Case-Based Plan Recognition Using Action Sequence Graphs

Swaroop S. Vattam[1], David W. Aha[2], and Michael Floyd[3]

[1] NRC Postdoctoral Fellow, Naval Research Laboratory (Code 5514), Washington, DC
[2] Navy Center for Applied Research in Artificial Intelligence,
Naval Research Laboratory (Code 5514), Washington, DC, USA
[3] Knexus Research Corporation, Springfield, VA, USA
{swaroop.vattam.ctr.in,david.aha}@nrl.navy.mil,
michael.floyd@knexusresearch.com

Abstract. We present SET-PR, a novel case-based plan recognition algorithm that is tolerant to missing and misclassified actions in its input action sequences. SET-PR uses a novel representation called action sequence graphs to represent stored plans in its plan library and a similarity metric that uses a combination of graph degree sequences and object similarity to retrieve relevant plans from its library. We evaluated SET-PR by measuring plan recognition convergence and precision with increasing levels of missing and misclassified actions in its input. In our experiments, SET-PR tolerated 20%-30% of input errors without compromising plan recognition performance.

Keywords: Case-based reasoning, plan recognition, error tolerance, graph representation of plans, approximate graph matching.

1 Introduction

Plan recognition is considered the inverse problem of plan synthesis. It involves an observed and an observing agent. Given an input sequence of actions executed by the observed agent, the observing agent attempts to map this observed sequence to a plan such that the observed sequence is a subsequence of actions in the recognized plan.

One of the fundamental assumptions of classical plan recognition is that observed actions are reliable. This assumption is unrealistic for agents acting in the real world who may frequently fail to notice the actions of others (because they have to attend to several actors and events in their environment) or misclassify the observed actions (due to uncertainty in the real world and incomplete or inaccurate agent models). We relax this assumption, and present a single-agent keyhole plan recognition algorithm that is tolerant to two kinds of input errors: missing and misclassified actions.

Our plan recognition algorithm, called *Single-agent Error-Tolerant Plan Recognizer* (SET-PR), assumes the existence of a plan library consisting of a set of cases. Each case includes a specification of a planning problem and a fully-grounded plan that is a solution to this problem. Inputs to SET-PR are subsequences of plans, which are matched to plans in the plan library to retrieve candidate plans. Currently,

L. Lamontagne and E. Plaza (Eds.): ICCBR 2014, LNCS 8765, pp. 495–510, 2014.

the top-ranked plan is selected as the solution (the recognized plan). For online or dynamic plan recognition, SET-PR is executed each time a new set of observations arrive, obtaining an any-time hypothesized plan in each observation cycle.

Although case-based plan recognition (CBPR) is not novel, we explore a new representation for stored plans that impacts the similarity function we propose for case retrieval. More specifically, we use *action sequence graph*s to represent plans in a plan library (case base); they encode a detailed topology of a plan trace (a sequence of action-state pairs). Our similarity function uses a combination of graph degree sequences and object similarity for computing the similarity between input action sequences and stored plans.

Our paper is organized as follows. Section 2 describes related work on plan recognition. Section 3 then introduces essential notation that formalizes the CBPR problem. Section 4 introduces action sequence graphs and their use in SET-PR. Section 5 details our similarity function. Finally, Section 6 presents an initial empirical study that evaluates the robustness of SET-PR's plan recognition algorithm in the presence of input errors. We found that SET-PR is highly tolerant to increasing levels of input error until a yield point is reached, after which its performance degrades sharply.

2 Related Work

The ability to recognize the plans and goals of other agents is a fundamental aspect of intelligence that allows one to reason about what other agents are doing, why they are doing it, and what they will do next. Many AI researchers have focused on plan recognition approaches which can be broadly classified into *keyhole* or *intended* plan recognition. In keyhole recognition, the observing agent monitors the actions of an ambivalent observed agent. In contrast, in intended recognition, the observed and observing agents cooperate to convey the intentions of the observed agent. Another dimension of classification relates to the presence of single or multiple observed agents. We restrict ourselves to the single-agent keyhole plan recognition problem.

Several approaches has been proposed to address the problem of plan recognition (Sukthankar et al., 2014), including *consistency-based* approaches (e.g., Hong, 2001; Kautz & Allen, 1986; Lesh & Etzioni, 1996; Lau et al., 2004; Kumaran, 2007), and *probabilistic* approaches (e.g., Bui, 2003; Charniak & Goldman, 1991, 1993; Geib & Goldman, 2009; Goldman, Geib & Miller, 1999; Pynadath & Wellman, 2000). The former include hypothesize and revise algorithms, version space techniques, and other closed-world reasoning algorithms, while probabilistic algorithms include those that use stochastic grammars and probabilistic relational models. Both these approaches are sensitive to (1) an incomplete plan library and (2) missing or misclassified actions in the input observations. There have been few attempts to address this issues within these frameworks (e.g., using background goals (Lesh, 1996) or focus stacks (Rich et al., 2001)), but current solutions are usually problem-specific and lack generalizable qualities.

A lesser known approach to the single-agent keyhole plan recognition problem is the case-based approach as exemplified by Cox & Kerkez (2006) and Tecuci & Porter (2009). These investigations focus on the issues of incomplete plan library, incrementally learning the plan library, and responding to novel inputs. But they do not address the issue of error-prone input action sequences. Cox & Kerkez (2006) proposed a novel representation for storing and organizing plans in the plan library based on action-state pairs and abstract states, which counts the number of instances of each type of generalized state predicate. In our work, we start with a similar action-state representation, but process it using a graph representation and store our cases as graphs. As a result, our similarity metrics also operate on graphs.

Most other research on single-agent keyhole CBPR has an application focus. For instance Fagan and Cunningham (2003) acquire cases (state-action sequences) to predict a human's next action while playing SPACE INVADERS. Cheng and Thawonmas (2004) propose a CBPR system for assisting players with low-level management tasks in WARGUS. Lee et al. (2008) integrate Kerkez and Cox's technique with a reinforcement learner to predict opponent actions on a simplified WARGUS task. Similarly, Molineaux et al. (2009) integrate a plan recognition system with a case-based reinforcement learner for an adversarial action selection task involving an American football simulator. In contrast to an application thrust, the focus of our work is to produce a more general single-agent keyhole CBPR approach that is tolerant to uncertainty in observed actions.

In other related work, user traces have been used in a variety of case-based reasoning systems. In CBR systems that learn by observation, user traces are used to automatically extract knowledge and cases so that the system can learn the expert's behavior. These cases typically store state-action pairs (Rubin & Watson, 2010) or state-plan pairs (Ontañón et al., 2007), and retrieval is based on a single state rather than, like our approach, an entire trace. Temporal Backtracking (Floyd & Esfandiari, 2011) has been used in learning by observation to include additional states and actions from a trace during retrieval such that traces are dynamically resized as necessary. Similarly, trace-based reasoning (Zarka et al., 2013) and episode-based reasoning (Sánchez-Marré, 2005) store fixed-length traces in cases and compare the entire trace during retrieval. The primary difference between these approaches and our own is that they represent traces as linear sequences whereas we represent them as graphs.

3 Case-Based Plan Recognition

We now introduce notation that formalizes the general problem of CBPR. A *planning problem* is a 3-tuple $\Pi = (O, s_0, g)$, where O is a set of planning operators, s_0 is the initial state, and g is the goal specification (Ghallab, Nau & Traverso, 2004).

An *action-state sequence* is a sequence $\mathbb{s} = \langle (a_0, s_0), (a_1, s_1), \ldots, (a_n, s_n) \rangle$ where an action $a_i \in A = \{\text{all ground instances of operators in } O\}$, and s_i is the state resulting from executing action a_i in state s_{i-1}. A *plan* is a special action-state sequence $\pi = \langle (null, s_0), (a_1, s_1), \ldots, (a_g, s_g) \rangle$ where s_0 is the initial state of Π and $s_g \in \{s | s \text{ satisfies } g\}$ is a goal state of Π, where *satisfies* has the same semantics as

originally laid out in Ghallab, Nau & Traverso (2004). Action a_0 is $null$ because the action preceding the initial state does not need to be specified. While most plan recognition systems represent plans as a sequence of actions, this kind of representation (augmenting action information with the following state information) was proposed originally by Cox and Kerkez (2006). Our rationale for choosing this representation is to offset the inaccuracies of missing or misclassified actions by including information about states.

A *case* is a tuple $c = (\Pi_0, \pi_0)$, where Π_0 is a planning problem and π_0 is a corresponding plan for solving it. A *plan library* (or *case base*) is a set of cases $C = \{(\Pi_i, \pi_i) | 1 \leq i \leq m\}$.

Definition: A *CBPR problem* is represented by a tuple $(C, \mathbb{s}^{target})$, where C is a plan library and $\mathbb{s}^{target} = \langle (null, s_0), \ldots, (a_k, s_k) | k \geq 1 \rangle$ is a target action-state sequence, a subsequence of a plan, that has to be recognized. A solution to this problem is a plan π^{sol} that is predicted using the following CBR process:

- *Plan retrieval*: Retrieve cases $\{(\Pi_i, \pi_i)\}$ from the plan library such that π_i is similar to $\mathbb{s}^{target}$ according to some similarity metric
- *Plan evaluation*: Evaluate the retrieved cases according to some evaluation criteria to select a source case $(\Pi^{source}, \pi^{source})$
- *Plan adaptation*: Adapt π^{source} to increase its similarity with $\mathbb{s}^{target}$ by resolving the differences between the two, resulting in π^{sol} (and corresponding Π^{sol})
- *Plan repair*: Test and revise π^{sol} to remove inconsistencies

For the sake of simplicity, we assume that the steps of plan adaptation and repair are not required, i.e., $(\Pi^{source}, \pi^{source})$ is equal to (Π^{sol}, π^{sol}). Table 1 captures the differences between the case-based plan synthesis and recognition problems expressed in this notation.

Table 1. Contrasting the general case-based plan synthesis and case-based plan recognition problems

	Case-Based Plan Synthesis	**Case-Based Plan Recognition**
Case Base	$C = \{(\Pi_i, \pi_i)\}$	$C = \{(\Pi_i, \pi_i)\}$
Input	$\langle C, \Pi^{target} \rangle$	$\langle C, \mathbb{s}^{target} \rangle$
Output	π^{sol}	π^{sol}
Retrieval	Match Π^{target} with Π_is from cases in C to get Π^{source} (and its π^{source})	Match $\mathbb{s}^{target}$ with π_is from cases in C to get π^{source}
Adaptation	Adapt π^{source} to resolve differences between Π^{target} and Π^{source} to get π^{sol}	Adapt π^{source} to resolve differences between $\mathbb{s}^{target}$ and π^{source} to get π^{sol}

In online or dynamic CBPR, the above CBR process is employed in recognizing an ongoing plan, before it has ended, by executing the process each time a new set of

observations arrive, obtaining an any-time hypothesized plan in each observation cycle.

4 Case Representation

We define a graph data structure called *action sequence graphs* to represent action-state sequences in our cases. Graphs provide a rich means for modelling structured objects and they are widely used in real-life applications to model many objects and processes, from molecules and images to buildings and entire ecosystems. Graphs have also been widely used in planning to represent task networks, goal networks, and plan graphs. The *planning encoding graph* (Serina, 2010) representation, used to encode planning problems, has been particularly influential in the design of our representation. Although there are syntactic similarities between planning encoding graphs and action sequence graphs, important semantic differences exist because the former encodes a problem while the latter encodes a solution (plan).

There are at least two advantages of using a graph representation for cases. First, graphs are well understood. We can rely on a solid theoretical foundation of graph theory to analyze and manipulate our representation. We can also leverage a huge body of work to identify reliable metrics and efficient algorithms. Second, our representation is domain-general; it can be applied to plan recognition problems for a wide range of domains.

4.1 Preliminaries

A *labeled directed graph* G is a 3-tuple $G = (V, E, \lambda)$ where V is the set of vertices, $E \subseteq V \times V$ is the set of directed edges or arcs, and $\lambda: V \cup E \rightarrow 2^L$ assigns labels to vertices and edges. Here, L is a finite set of symbolic labels and 2^L is a set of all the *multisets* on L; this labeling scheme permits multiple non-unique labels for a node or an arc.

An arc $e = [v, u] \in E$ is considered to be directed from v to u, where v is called the *source* node and u is called the *target* node of the arc. Also, u is a *direct successor* of v, v is a *direct predecessor* of u, and v is *adjacent* to vertex u and vice versa.

The *union* of two graphs $G_1 = (V_1, E_1, \lambda_1)$ and $G_2 = (V_2, E_2, \lambda_2)$, denoted by $G_1 \cup G_2$, is the graph $G = (V, E, \lambda)$ where $V = V_1 \cup V_2$, $E = E_1 \cup E_2$, and

$$\lambda(x) = \begin{cases} \lambda_1(x), if\ x \in (V_1 \setminus V_2) \vee x \in (E_1 \setminus E_2) \\ \lambda_2(x), if\ x \in (V_2 \setminus V_1) \vee x \in (E_2 \setminus E_1) \\ \lambda_1(x) \cup \lambda_2(x), otherwise \end{cases}$$

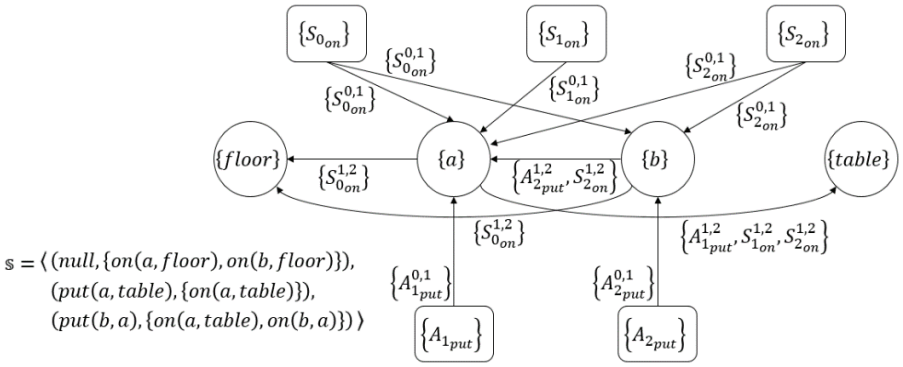


Fig. 1. An example action-state sequence $\mathbb{s}$ and its corresponding action sequence graph $\mathcal{E}^{\mathbb{s}}$

4.2 Action Sequence Graphs

An *action sequence graph* is an order-preserving encoding of an action-state sequence $\mathbb{s} = \langle (null, s_0), (a_1, s_1), \dots \rangle$. Two action-state sequences can be matched by matching their corresponding action sequence graphs. Figure 1 shows an example of $\mathbb{s}$ and its corresponding action sequence graph.

An action-state sequence $\mathbb{s}$ is defined over a planning language $\mathcal{L}$ consisting of $\boldsymbol{P}$, a set of finitely many *predicate* symbols, and $\boldsymbol{O}$, a finite set of typed constants representing distinct *objects* in the planning domain. An action $\boldsymbol{a}$ in $(\boldsymbol{a}, \boldsymbol{s}) \in \mathbb{s}$ is represented as a ground atom $p(c_1{:}\, t_1, \dots, c_n{:}\, t_n)$, where $p \in \boldsymbol{P}$ and represents an action, $c_i \in \boldsymbol{O}$, and t_i is an instance of c_i. Similarly a state $\boldsymbol{s}$ in $(\boldsymbol{a}, \boldsymbol{s}) \in \mathbb{s}$ is represented as a set of ground atoms $\{p(c_1{:}\, t_1, \dots, c_n{:}\, t_n) | p \in \boldsymbol{P}, c_i \in \boldsymbol{O}\}$.

Definition: Given a ground atom $\boldsymbol{p} = p(c_1{:}\, t_1, \dots, c_n{:}\, t_n)$ representing either an action $\boldsymbol{a}$ or a single fact of state $\boldsymbol{s}$ in the k^{th} action-state pair $(\boldsymbol{a}, \boldsymbol{s})_k \in \mathbb{s}$, a ***predicate encoding graph*** is a labeled directed graph $\mathcal{E}^p(\boldsymbol{p}) = (V_p, E_p, \lambda_p)$ such that:

- $V_p = \begin{cases} \{A_{k_p}, c_1, \dots, c_n\}, \text{if } \boldsymbol{p} \text{ is an action} \\ \{S_{k_p}, c_1, \dots, c_n\}, \text{if } \boldsymbol{p} \text{ is a state fact} \end{cases}$
- $E_p = \begin{cases} [A_{k_p}, c_1] \cup \bigcup_{i=1,n-1;j=i+1,n} [c_i, c_j], \text{if } \boldsymbol{p} \text{ is an action} \\ [S_{k_p}, c_1] \cup \bigcup_{i=1,n-1;j=i+1,n} [c_i, c_j], \text{if } \boldsymbol{p} \text{ is a state fact} \end{cases}$
- $\lambda_p(A_{k_p}) = \{A_{k_p}\};\ \lambda_p(S_{k_p}) = \{S_{k_p}\};\ \lambda_p(c_i) = \{t_i\}$ for $i = 1, \dots, n$
- $\lambda_p([A_{k_p}, c_1]) = \{A_{k_p}^{0,1}\};\ \lambda_p([S_{k_p}, c_1]) = \{S_{k_p}^{0,1}\};$
  $\forall [c_i, c_j] \in E_p, \lambda_p([c_i, c_j]) = \begin{cases} \{A_{k_p}^{i,j}\}, \text{if } \boldsymbol{p} \text{ is an action} \\ \{S_{k_p}^{i,j}\}, \text{if } \boldsymbol{p} \text{ is a state fact} \end{cases}$

Here is an interpretation of this definition. Suppose we have a predicate $\boldsymbol{p} = p(c_1{:}\, t_1, \ldots, c_n{:}\, t_n)$. Depending on whether $\boldsymbol{p}$ represents an action or a state fact, the first node of the predicate encoding graph $\mathcal{E}^p(\boldsymbol{p})$ is either A_{k_p}or S_{k_p} (labeled $\{A_{k_p}\}$ or $\{S_{k_p}\}$). Let us assume that it is an action predicate. A_{k_p} is then connected to the second node of this graph, the object node c_1 (labeled $\{t_1\}$), through the edge $[A_{k_p}, c_1]$ (labeled $\{A_{k_p}^{0,1}\}$). Next, c_1 is connected to the third node c_2 (labeled $\{t_2\}$) through the edge $[c_1, c_2]$ (labeled $\{A_{k_p}^{1,2}\}$), then to the fourth node c_3 (labeled $\{t_3\}$) through the edge $[c_1, c_3]$ (labeled $\{A_{k_p}^{1,3}\}$), and so on. Next, the third node c_2 is connected to c_3 through $A_{k_p}^{2,3}$, to c_4 through $A_{k_p}^{2,4}$, with appropriate labels, and so on.

Example: Suppose predicate $\boldsymbol{p} = put(block{:}\, a, block{:}\, b, table{:}\, t)$ appears in the fifth ($k = 5$) action-state pair of an observed sequence of actions. The nodes of this predicate are $\{A_{5_{put}}\}$, $\{a\}$, $\{b\}$, and $\{t\}$. The edges are $[A_{5_{put}}, a]$, $[a, b]$, $[a, t]$, and $[b, t]$, with respective labels $\{A_{5_{put}}^{0,1}\}$, $\{A_{5_{put}}^{1,2}\}$, $\{A_{5_{put}}^{1,3}\}$, and $\{A_{5_{put}}^{2,3}\}$. The predicate encoding graph for $\boldsymbol{p}$ is shown in Figure 2.

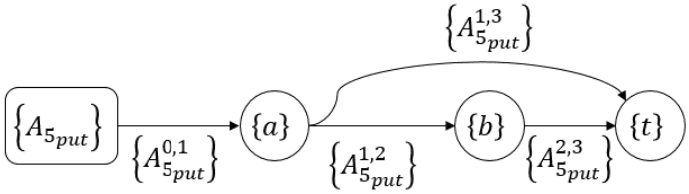


Fig. 2. An example predicate encoding graph $\mathcal{E}^p(\boldsymbol{p})$ corresponding to $\boldsymbol{p} = put(block{:}\, (a, block{:}\, b, table{:}\, t)$

Definition: An ***action sequence graph*** of an action-state sequence $\mathbb{s}$ is a labeled directed graph $\mathcal{E}^{\mathbb{s}} = \bigcup_{(\boldsymbol{a},\boldsymbol{s})\in\mathbb{s}}\big(\mathcal{E}(\boldsymbol{a}) \cup_{\boldsymbol{p}\in\boldsymbol{s}} \mathcal{E}(\boldsymbol{p})\big)$, a union of the predicate encoding graphs of all the action and state predicates in $\mathbb{s}$. (See Figure 1 for an example of a complete action sequence graph.)

Recall that a case is a tuple $c = (\Pi_0, \pi_0)$, where π_0 is an instance of the action-state sequence where s_0 is the initial state and s_n is the goal state of Π_0. Therefore, π_0 can be represented by an action sequence graph $\mathcal{E}^{\pi_0}$, Π_0 is implicit in π_0 (i.e., π_0 already contains the initial and goal states of Π_0), and it need not be captured explicitly in c. Given this, $\mathcal{E}^{\pi_0}$ can serve as a representation of case c.

5 Similarity and Retrieval

The first step in the CBR process is case retrieval. Given $\langle C, \mathbb{s}^{target}\rangle$, where C is a plan library and $\mathbb{s}^{target}$ is a target action-state sequence (also a subsequence of a plan), retrieval involves identifying those cases from the plan library that are similar to $\mathbb{s}^{target}$. This requires a similarity metric.

Cases in the plan library are represented as action sequence graphs, as is $\mathbb{s}^{target}$. Thus, we compute their *structural* similarity using graph matching. One reliable way to match two graphs is to find their *maximum common subgraph* (MCS) (Raymond & Willett, 2002), where similarity is an increasing function of the size of the MCS between G_1 and G_2. Given an action sequence graph $\mathcal{E}^{\mathbb{s}}$ for $\mathbb{s}^{target}$, the retrieval step would then compute the MCS between $\mathcal{E}^{\mathbb{s}}$ and each $\mathcal{E}^{\pi}$ in the case base, selecting the top k $\mathcal{E}^{\pi}$s based on the size of their MCS.

Unfortunately, computing the MCS between two or more graphs is an NP-Complete problem (Raymond & Willett, 2002). The worst-case time requirement of this retrieval step increases exponentially with the size of $\mathcal{E}^{\mathbb{s}}$, restricting its applicability to small plan recognition problems only. Thus, we seek an efficient approximation of the MCS similarity metric.

5.1 Degree Sequences Similarity Metric

We hypothesize that Johnson's (1985) similarity coefficient, based on graph *degree sequences*, can be used to derive an efficient approximation of the MCS similarity metric. The degree sequence of a graph is the non-increasing sequence of its vertex degrees. The degree sequence is a graph invariant, so isomorphic graphs have the same degree sequences. However, the degree sequence does not uniquely identify a graph. This similarity coefficient has been used for rapid graph matching in several applications, including in case-based planning with planning encoding graphs as a quick filter to *reject* unpromising cases during retrieval (Serina, 2010). In our approach, we will also use this similarity coefficient, but for the opposite purpose - we propose to use it to *select* promising cases.

Johnson's similarity coefficient for two graphs is calculated as follows. First, the set of vertices in each graph is divided into l partitions by label type, and then sorted in a non-increasing total order by degree[1]. Let L_1^i and L_2^i denote the sorted degree sequences of a partition i in the action sequence graphs G_1 and G_2, respectively. An upper bound on the number of vertices $V(G_1, G_2)$ and edges $E(G_1, G_2)$ of the MCS of these two graphs, $G_{1,2}$, can then be computed as:

$$V(G_1, G_2) = \sum_{i=1}^{l} min\left(\left|L_1^i\right|, \left|L_2^i\right|\right)$$

$$E(G_1, G_2) = \left\lfloor \sum_{i=1}^{l} \sum_{j=1}^{min\left(\left|L_1^i\right|, \left|L_2^i\right|\right)} \frac{min\left(\left|E\left(v_1^{i,j}\right)\right|, \left|E\left(v_2^{i,j}\right)\right|\right)}{2} \right\rfloor$$

where $v_1^{i,j}$ denotes the j^{th} vertex of the L_1^i sorted degree sequence, and $E\left(v_1^{i,j}\right)$ denotes the set of edges connected to vertex $v_1^{i,j}$. Then the structural similarity coefficient can be computed as follows:

$$sim_{str}(G_1, G_2) = \frac{\left(V(G_1, G_2) + E(G_1, G_2)\right)^2}{(|V(G_1)| + |E(G_1)|) \cdot (|V(G_2)| + |E(G_2)|)}$$

[1] The degree (or *valence*) of a vertex v of a graph G is the number of edges that touch v.

This computation can be performed in $O(n \cdot log\ n)$ time, where $n = max_i(|L_1^i|, |L_2^i|)$ (Raymond & Willett, 2002).

5.2 Example

Consider the two graphs G_1 and G_2 in Figure 3. The degree sequences of the partitions of G_1 and G_2 (partitioned by label type) are also tabulated in Figure 3. We can compute their (graph) similarity as follows:

$$V(G_1, G_2) = 3 + 1 + 1 + 1 + 1 + 1 = 8$$
$$E(G_1, G_2) = \left\lfloor \frac{(7 + 7 + 6)}{2} + \frac{2}{2} + \frac{1}{2} + \frac{2}{2} + \frac{2}{2} + \frac{1}{2} \right\rfloor = 14$$
$$sim_{str}(G_1, G_2) = \frac{(8 + 14)^2}{(8 + 12) \cdot (9 + 19)} = 0.8643$$

5.3 Combined Similarity Metric

We use a combination of structural and object similarity to compute the overall similarity of an input subsequence to a plan in the plan library:

$$sim(\mathbb{s}^{target}, \pi_i) = sim_{str}(G_1, G_2) + sim_{obj}(\mathbb{s}^{target}, \pi_i)/2,$$

where G_1 is the action sequence graph of $\mathbb{s}^{target}$, G_2 is the action sequence graph of π_i, $sim_{str}(G_1, G_2)$ is the degree sequences similarity function described above, and

$$sim_{obj}(\mathbb{s}^{target}, \pi_i) = \frac{O_{\mathbb{s}} \cap O_{\pi_i}}{O_{\mathbb{s}} \cup O_{\pi_i}}$$

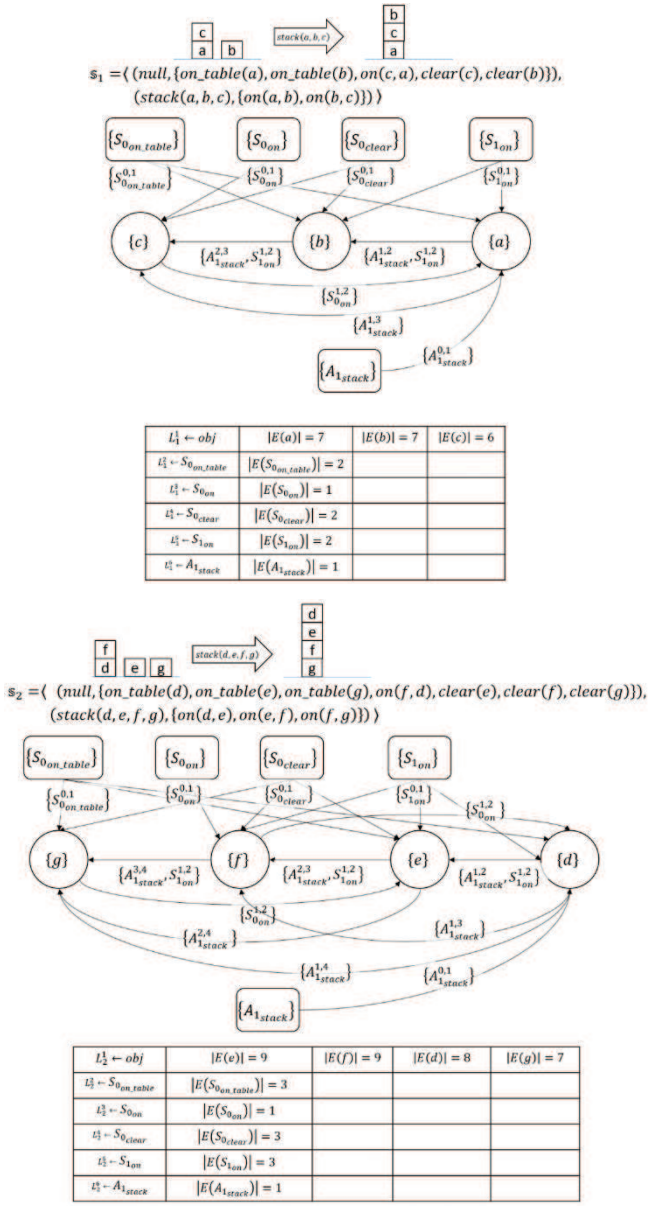
where $O_{\mathbb{s}}$ is the set of (grounded) objects in $\mathbb{s}^{target}$ and $O_{\pi_i}$are objects in π_i.

6 Evaluation

In this section we evaluate our claim that plan recognition in SET-PR is robust to two kinds of input errors: misclassified actions and missing actions.

6.1 Empirical Method

We conducted our experiments in the blocks world domain, which we chose because of its simplicity and affordance to quick automatic generation of the plan library with desired characteristics. We used a hierarchal task network (HTN) planner to generate plans for our plan library. Planning problems were created by randomly selecting an initial state and goal state (under the constraint that it is possible to reach the goal state from the initial state), and given as input to the HTN planner. The generated plan (actions and corresponding states) was converted into an action sequence graph and stored, along with the planning problem, as a case. In total, we used this method to generate 100 (error-free) cases for our plan library. We created 4 separate plan libraries that vary in their average plan length (8.94, 12.48, 16.36, and 19.46 actions respectively).



$L^1_1 \leftarrow obj$	$\lvert E(a)\rvert = 7$	$\lvert E(b)\rvert = 7$	$\lvert E(c)\rvert = 6$
$L^2_1 \leftarrow S_{0_{on\_table}}$	$\lvert E(S_{0_{on\_table}})\rvert = 2$		
$L^3_1 \leftarrow S_{0_{on}}$	$\lvert E(S_{0_{on}})\rvert = 1$		
$L^4_1 \leftarrow S_{0_{clear}}$	$\lvert E(S_{0_{clear}})\rvert = 2$		
$L^5_1 \leftarrow S_{1_{on}}$	$\lvert E(S_{1_{on}})\rvert = 2$		
$L^6_1 \leftarrow A_{1_{stack}}$	$\lvert E(A_{1_{stack}})\rvert = 1$		

$L^1_2 \leftarrow obj$	$\lvert E(e)\rvert = 9$	$\lvert E(f)\rvert = 9$	$\lvert E(d)\rvert = 8$	$\lvert E(g)\rvert = 7$
$L^2_2 \leftarrow S_{0_{on\_table}}$	$\lvert E(S_{0_{on\_table}})\rvert = 3$			
$L^3_2 \leftarrow S_{0_{on}}$	$\lvert E(S_{0_{on}})\rvert = 1$			
$L^4_2 \leftarrow S_{0_{clear}}$	$\lvert E(S_{0_{clear}})\rvert = 3$			
$L^5_2 \leftarrow S_{1_{on}}$	$\lvert E(S_{1_{on}})\rvert = 3$			
$L^6_2 \leftarrow A_{1_{stack}}$	$\lvert E(A_{1_{stack}})\rvert = 1$			

Fig. 3. Two graphs G_1 and G_2 and the degree sequences of their partitions

We varied the following variables in our evaluation of SET-PR: (1) length of the plans in the plan library; (2) percentage of absent ⟨action, state⟩ pairs; and (3) percentage of misclassified ⟨action, state⟩ pairs. Our metrics were convergence, convergence point, and precision. *Convergence* is defined as a boolean value that, for

a given query q with n ⟨action, state⟩ pairs, is true if the plan retrieved on the last ⟨action, state⟩ pair matches q. *Convergence point* is defined only for those queries that converge; if they converge after action k, it is defined as k/n. *Precision* is the total number of plans associated with the query's ⟨action, state⟩ pairs that are correct, divided by n.

We evaluated SET-PR's performance for each plan library L using the following method. For each randomly selected case $c = (\Pi, \pi)$ in L, we copied plan π, randomly distorted its action sequence $\langle (null, s_0), (a_1, s_1), \ldots, (a_g, s_g) \rangle$ (to introduce a fixed amount of error), and used it as an incremental query to SET-PR (i.e., initially with only its first ⟨action, state⟩ pair, and then repeatedly adding the next such pair in its sequence). The error levels tested, separately for both error types (missing and misclassified actions), were {10%, 20%, 30%, 40%, 50%}. For each case c, we also recorded the convergence, convergence point, and precision. Finally, we averaged these metrics across L.

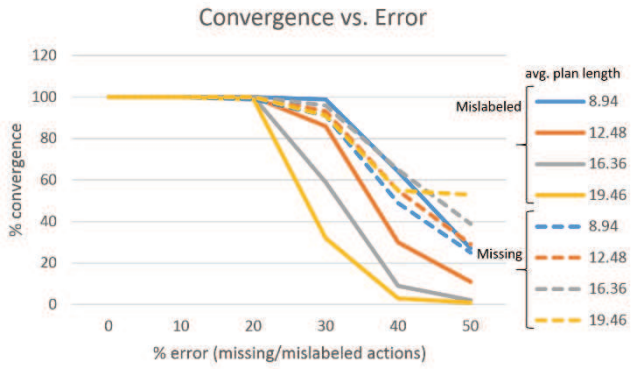


Fig. 4. Convergence rate vs. % error results

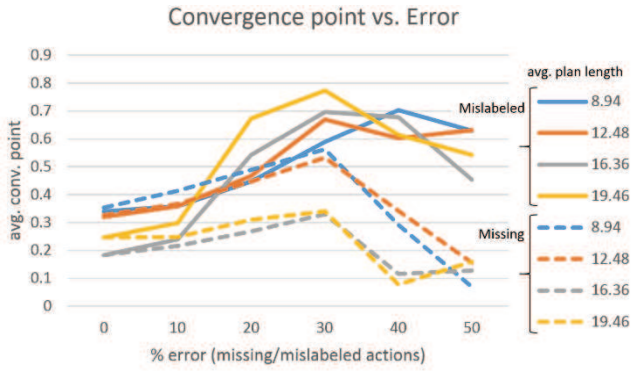


Fig. 5. Average convergence point vs. % error results

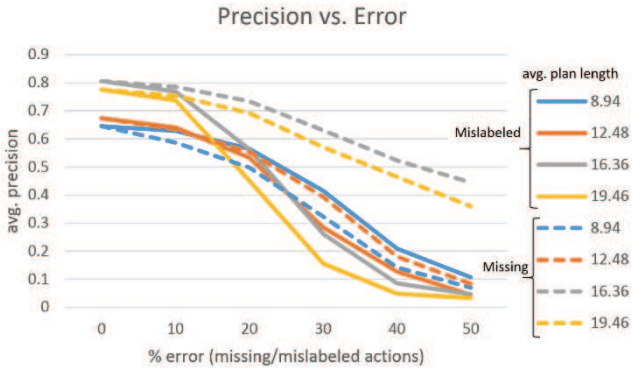


Fig. 6. Average precision vs. % error results

6.2 Trends and Discussion

Figures 4-6 summarize our results. The primary trends, for a given plan length, are as follows. As % error increases (for both error types):

- Convergence rate remains high and near constant until some (yield) point is reached and then sharply decreases
- Average convergence point increases until the yield point is reached, beyond which its value becomes unpredictable
- Average precision decreases steadily and approaches zero

The existence of yield point was surprising; we expected a gradual decrease in convergence rate with a gradual increase in % error. For the blocks world domain, the yield point was around 20% for misclassified and 30% for missing actions. The convergence rate was more than 90% until this yield point.

No clear trends emerged that distinguished the results by average plan length (across the four libraries), although for longer plan lengths, misclassified actions degraded performance of the algorithm more severely than did missing actions. More fine grained analysis is required to explain this.

After the yield point is reached, average convergence point is unreliable; the number of data points in the converged set were too few to indicate any clear trend.

Based on this analysis we conclude that SET-PR is robust to the presence input error until the yield point is reached, which in our experiments was 20%-30% of the input error (depending on the error type). To what extent the yield point is domain dependent and how we can push the yield point further to the right remain open research questions.

We used a variant of SET-PR for our baseline comparison. SET-PR$_{baseline}$ differs from SET-PR only with respect to the representation of action sequences in plan library and queries. While SET-PR uses ⟨action, state⟩ sequences, SET-PR$_{baseline}$ uses ⟨action⟩ sequences. SET-PR$_{baseline}$ better models the representations of classical plan recognition techniques which do not use explicit information about states to infer plans. Recall that the rationale for choosing to include state information in SET-PR

was to offset the inaccuracies introduced by missing or misclassified actions. Therefore, we expected the performance of SET-PR$_{baseline}$ to deteriorate more than SET-PR in the presence of input errors.

Table 2. Sample results for SET-PR$_{baseline}$ for one plan library for one type of error (misclassified)

	0%	10%	20%	30%	40%	50%
Conv. Rate (%)	76	14	9	7	1	1
Avg. conv. point	0.21	0.24	0.23	0.43	0.37	0.16
Avg. precision	0.72	0.13	0.11	0.08	0.04	0.02

Table 2 captures the convergence rate, average convergence point, and average precision for different % error levels for one of the plan libraries (average plan length = 12.4) and one type of error (misclassified actions). The results for other plan libraries were similar and we see the following trends in the case of SET-PR$_{baseline}$.

First, when the input error was 0%, we noted that the rate of convergence, average convergence point, and average precision were comparable to SET-PR (rate of convergence and average precision were better in SET-PR, but the convergence point was better in SET-PR$_{baseline}$).

Second, we noted the existence of yield point in SET-PR$_{baseline}$ as well, but it was reached even before the 10% input error level. This suggests that error increments smaller than 10% are required to find out exactly how much error SET-PR$_{baseline}$ can tolerate.

Third, after the yield point is reached, the rate of convergence and average precision in SET-PR$_{baseline}$ dropped sharply to levels seen for 50% error in SET-PR. This suggests that SET-PR$_{baseline}$ is extremely sensitive to input errors.

Fourth, the average convergence point in SET-PR$_{baseline}$ was higher compared to SET-PR, but this is not a meaningful statistic because the data points in the converged set were too few beyond the 0% error level.

Finally, we conclude that SET-PR compares favorably to SET-PR$_{baseline}$ for error tolerance.

7 Summary

We developed a case-based approach to the problem of plan recognition that can tolerate missing and misclassified actions in the input action sequences. We introduced a novel graph representation, called *action sequence graphs*, to represent plans in the plan library. We described a similarity function that uses a combination of graph degree sequences and object similarity for matching action sequence graphs, and used it to retrieve relevant stored plans from the plan library. We also presented an initial empirical investigation of our approach using the blocks world domain. We measured the extent to which our approach is tolerant to missing and misclassified actions, and found that its performance was robust to the presence of up to 20%-30% of error in the input actions (depending on error type). We also found that, in the

presence of input errors, our approach compared favorably to the baseline which used more traditional plan recognition representations.

Our study had several limitations. For example, we used a simple paradigmatic domain, the plan library was relatively small, errors were introduced in a systematic manner rather than reflecting a naturally occurring distribution, and while we introduced errors in the query we did not also introduce them in the plan library. As part of our future work, we will address these issues and test SET-PR in different domains, some of which will be real world application domains. One such domain we plan to target is human-robot teams. This will require extending SET-PR to work in domains involving multiple agents and multiple types of plan recognition tasks.

We also plan to compare and test SET-PR's performance using different degree sequence similarity metrics previously reported in literature (e.g., Wallis et al., 2001; Bunke and Shearer, 1998).

Acknowledgements. Thanks to OSD ASD (R&E) for sponsoring this research. Swaroop Vattam performed this work while an NRC post-doctoral research associate located at the Naval Research Laboratory. The views and opinions contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of NRL or OSD.

References

Bui, H.: A general model for online probabilistic plan recognition. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1309–1315. Morgan Kaufmann, Acapulco (2003)

Bunke, H., Shearer, K.: A graph distance metric based on the maximum common subgraph. Pattern Recognition 19(3), 255–259 (1998)

Charniak, E., Goldman, R.: A probabilistic model of plan recognition. In: Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 160–165. AAAI Press, Anaheim (1991)

Charniak, E., Goldman, R.: A Bayesian model of plan recognition. Artificial Intelligence 64, 53–79 (1993)

Cheng, D.C., Thawonmas, R.: Case-based plan recognition for real-time strategy games. In: Proceedings of the Fifth Game-On International Conference, pp. 36–40. University of Wolverhampton Press, Reading (2004)

Cox, M.T., Kerkez, B.: Case-based plan recognition with novel input. Control and Intelligent Systems 34(2), 96–104 (2006)

Fagan, M., Cunningham, P.: Case-based plan recognition in computer games. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 161–170. Springer, Heidelberg (2003)

Floyd, M.W., Esfandiari, B.: Learning state-based behaviour using temporally related cases. In: Petridis, M. (ed.) Proceedings of the Sixteenth UK Workshop on Case-Based Reasoning. Springer, Cambridge (2011)

Geib, C.W., Goldman, R.P.: A probabilistic plan recognition algorithm based on plan tree grammars. Artificial Intelligence 173(11), 1101–1132 (2009)

Ghallab, M., Nau, D., Traverso, P.: Automated planning: Theory and practice. Morgan Kaufmann, San Mateo (2004)

Goldman, R.P., Geib, C.W., Miller, C.A.: A new model of plan recognition. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 245–254. Morgan Kaufmann, Bled (1999)

Hong, J.: Goal recognition through goal graph analysis. Journal of Artificial Intelligence Research 15, 1–30 (2001)

Johnson, M.: Relating metrics, lines and variables defined on graphs to problems in medicinal chemistry. John Wiley & Sons, New York (1985)

Kautz, H., Allen, J.: Generalized plan recognition. In: Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 32–37. Morgan Kaufmann, Philadelphia (1986)

Kumaran, V.: Plan recognition as candidate space search (Master's Thesis). North Carolina State University, Department of Computer Science, Raleigh, NC (2007)

Lau, T., Wolfman, S.A., Domingos, P., Weld, D.S.: Programming by demonstration using version space algebra. Machine Learning 53(1-2), 111–156 (2003)

Lee, J., Koo, B., Oh, K.: State space optimization using plan recognition and reinforcement learning on RTS game. In: Proceedings of the International Conference on Artificial Intelligence, Knowledge Engineering, and Data Bases. WSEAS Press, Cambridge (2008)

Lesh, N.: Fast, adaptive, and empirically-tested goal recognition. In: Proceedings of the Fifth International Conference on User Modeling, pp. 231–233 (1996)

Lesh, N., Etzioni, O.: Scaling up goal recognition. In: Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning, pp. 178–189 (1996)

Molineaux, M., Aha, D.W., Sukthankar, G.: Beating the defense: Using plan recognition to inform learning agents. In: Proceedings of the Twenty-Second International FLAIRS Conference, pp. 337–343. AAAI Press, Sanibel Island (2009)

Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 164–178. Springer, Heidelberg (2007)

Pynadath, D.V., Wellman, M.P.: Probabilistic state-dependent grammars for plan recognition. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 507–514. Morgan Kaufmann, San Francisco (2000)

Raymond, J.W., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. Journal of Computer-Aided Molecular Design 16, 521–533 (2002)

Rich, C., Sidner, C.L., Lesh, N.: Collagen: Applying collaborative discourse theory to human-computer interaction. AI Magazine 22(4), 15–26 (2001)

Rubin, J., Watson, I.: Similarity-based retrieval and solution re-use policies in the game of Texas Hold'em. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 465–479. Springer, Heidelberg (2010)

Sánchez-Marré, M., Cortés, U., Martínez, M., Comas, J., Rodríguez-Roda, I.: An approach for temporal case-based reasoning: Episode-based reasoning. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 465–476. Springer, Heidelberg (2005)

Serina, I.: Kernel functions for case-based planning. Artificial Intelligence 174(16), 1369–1406 (2010)

Sukthankar, G., Goldman, R., Geib, C., Pynadath, D., Bui, H.: An introduction to plan, activity, and intent recognition. In: Sukthankar, G., Goldman, R., Geib, C., Pynadath, D., Bui, H. (eds.) Plan, Activity, and Intent Recognition. Elsevier, Philadelphia (2014)

Tecuci, D., Porter, B.W.: Memory based goal schema recognition. In: Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society Conference. AAAI Press, Sanibel Island (2009)

Wallis, W.D., Shoubridge, P., Kraetz, M., Ray, D.: Graph distances using graph union. Pattern Recognition Letters 22, 701–704 (2001)

Zarka, R., Cordier, A., Egyed-Zsigmond, E., Lamontagne, L., Mille, A.: Similarity measures to compare episodes in modeled traces. In: Delany, S.J., Ontañón, S. (eds.) ICCBR 2013. LNCS, vol. 7969, pp. 358–372. Springer, Heidelberg (2013)