Case-Based Plan Recognition Under Imperfect Observability

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

[1]NRC Postdoctoral Fellow; Naval Research Laboratory (Code 5514); Washington, DC; USA
[2]Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5514); Washington, DC; USA
{swaroop.vattam.ctr.in, david.aha}@nrl.navy.mil

Abstract. SET-PR is a novel case-based recognizer that is robust to three kinds of input errors arising from imperfect observability, namely missing, mislabeled and extraneous actions. We extend our previous work on SET-PR by empirically studying its efficacy on three plan recognition datasets. We found that in the presence of higher input error rates, SET-PR significantly outperforms alternative approaches, which perform similarly to or outperform SET-PR in the presence of no input errors.

Keywords: Case-based reasoning, plan recognition, imperfect observability, graph representation, plan matching

1 Introduction

A plan recognizer observes the actions executed by an actor and attempts to infer the actor's plan. A plan recognizer typically receives its input observations (of actions) from a lower-level action recognition system that can be noisy. A sophisticated plan recognizer therefore needs to relax the assumption of perfect observability and expect at least three kinds of input errors: a *mislabeled* action occurs in an input when an actor's true action is recognized as some other action; a *missing* action occurs when a true action is unrecognized (i.e., classified as a non-action); and an *extraneous* action occurs when a non-action is classified as some valid action.

Single-Agent Error-Tolerant Plan Recognizer (SET-PR) is a novel case-based plan recognizer that has shown promise in tolerating these kinds of input errors. We previously introduced SET-PR and highlighted its representation and reasoning techniques (Vattam, Aha, & Floyd 2014). This paper extends our preliminary empirical study of SET-PR, which was limited to just one dataset (the Blocks World domain) (Vattam, Aha, & Floyd 2015). Here we conduct a more comprehensive empirical investigation of SET-PR by (1) expanding the scope of the investigation to three datasets (Blocks World, Linux, and Monroe), (2) adopting a wider range of plan-recognition performance metrics, and (3) comparing the performance of SET-PR to baseline algorithms.

This paper is organized as follows. Section 2 describes related work on plan recognition. Section 3 gives an overview of SET-PR including its novel plan representation and retrieval mechanism. Section 4 presents our more comprehensive study of SET-PR, including the hypotheses we address, data used, evaluation method, empirical results, and their analysis. In this investigation, we found that SET-PR significantly outperformed the baseline algorithms in the presence of higher levels of input error, although the baselines performed similar to or outperformed SET-PR in the presence of no input errors. We conclude and discuss future research plans in Section 5.

2 Related Research

Early work on plan recognition (e.g., Kautz & Allen, 1986) assumed that the observed actor's actions follow a hierarchical plan structure, requiring the plan recognizer to infer plans and sub-plans at multiple abstraction levels. However, it assumed perfect observability, which is unrealistic. Since then, a number of important probabilistic (e.g., Charniak & Goldman, 1993; Bauer, 1994; van Beek, 1996) and statistical parsing approaches (e.g., Pynadath & Wellman, 1995; Geib & Goldman, 2009) have been proposed that address issues of uncertainty. They frame plan recognition as a problem of probabilistic inference in a stochastic process that models the actor's action execution. While this offers a general and coherent framework for modeling different sources of uncertainty, they have not focused on problems due to imperfect observability. In contrast, activity recognition (Duong et al., 2005) algorithms, which apply signal processing techniques to discretize sensor information into coherent actions, have addressed imperfect observability issues. Bridging the gap between low-level, often noisy activity models and higher-level plans remains a research challenge.

Recently Ramirez and Geffner (2010) proposed a novel approach to plan recognition by formulating it in terms of plan synthesis and solving it using off-the-shelf planners. They extended their approach to perform plan recognition in POMDP settings (Ramirez & Geffner 2011), which they claim can tolertae different kinds of input errors. They demonstrated that it tolerates one kind of input error, namely missing actions (i.e., incomplete observations). However, like most plan recognition approaches theirs is "model-heavy"; they require accurate models of an actor's possible actions and how those actions interact to accomplish different goals. Engineering these models is difficult and time consuming. Furthermore, these plan recognizers perform poorly when confronted with novel situations and are brittle when the operating conditions deviate from model parameters.

SET-PR exemplifies case-based plan recognition (CBPR), a model-lite, lesser studied approach to plan recognition. Existing CBPR approaches (e.g., Cox & Kerkez, 2006; Tecuci &Porter, 2009) eschew generalized models and instead use plan libraries that contain plan instances that can be gathered from experience. CBPR algorithms can respond to novel inputs outside the scope of their plan library using plan adaptation techniques. However, to our knowledge they have not been designed for imperfect observability, which is the unique focus of SET-PR.

Cox and Kerkez (2006) proposed a novel representation for storing and organizing plans in a plan library, modeled as action-state pairs and abstract states, which counts the number of instances of each type of generalized state predicate. SET-PR uses a similar representation, but stores and processes plans in an action-sequence graph. Our encoding was inspired by planning encoding graphs (Serina, 2010). These are syntactically similar to our graphs but encode a planning problem while ours instead encode a solution (i.e., a grounded plan).

Plan retrieval is an important step in CBPR algorithms and presents an efficiency bottleneck. Our previous contribution presented an algorithm for speeding plan retrieval in SET-PR that uses plan projection and clustering (Maynord, Vattam, & Aha 2015). Sánchez-Ruiz and Ontañón (2014) use Least Common Subsumer Trees for the same purpose, but they are not applicable to our representation.

3 SET-PR

3.1 Representation

SET-PR learns to recognize plans from a given plan library C (i.e., a set of cases). Each case is a tuple $c = (\pi, g)$, where π is a known plan (the problem part), and g is its corresponding goal (the solution part).

3.1.1 Action state sequences. Each case's plan $c.\pi$ is modeled as an *action state sequence* $\mathbb{s} = \langle(\boldsymbol{a_0}, \boldsymbol{s_0}), \ldots, (\boldsymbol{a_n}, \boldsymbol{s_n})\rangle$, where each action $\boldsymbol{a_i}$ is a ground operator in the planning domain, and $\boldsymbol{s_i}$ is a ground state obtained by executing $\boldsymbol{a_i}$ in $\boldsymbol{s_{i-1}}$, with the caveat that $\boldsymbol{s_0}$ is an initial state, $\boldsymbol{a_0}$ is null, and $\boldsymbol{s_n}$ is a goal state. An action $\boldsymbol{a}$ in $(\boldsymbol{a}, \boldsymbol{s}) \in \mathbb{s}$ is a ground literal $\boldsymbol{p} = p(o_1\colon t_1, \ldots, o_n\colon t_n)$, where $p \in \boldsymbol{P}$ (a finite set of predicate symbols), $o_i \in \boldsymbol{O}$ (a finite set of objects), and t_i is an instance of o_i (e.g., stack(block:A, block:B)). A state $\boldsymbol{s}$ in $(\boldsymbol{a}, \boldsymbol{s}) \in \mathbb{s}$ is a set of ground literals (e.g., {on(block:A,block:B), on(block:B,substrate:TABLE)}).

Inputs to SET-PR consist of sequences (observed parts of a plan). An input to SET-PR $\mathbb{s}^{query}$ is also modeled as an action state sequence. However, unlike a plan, $\boldsymbol{s_0}$ and $\boldsymbol{s_n}$ in $\mathbb{s}^{query}$ need not be initial and goal states, and $\boldsymbol{a_0}$ need not be null.

Each case's goal $c.g$ is modeled as a task to be achieved (using the HTN vocabulary) or as a state to be achieved depending on the domain. This reduces a goal to an instance of a task ($c.g$ is an $\boldsymbol{a}$) or a state ($c.g$ is a $\boldsymbol{s}$) respectively. The representation of a goal can be flexible because it is the solution part of a case and does not participate in matching during retrieval.

3.1.2 Action Sequence Graphs. An action sequence graph is a graphical representation of an action state sequence, which is propositional. This graph preserves the topology of the sequence it encodes (including the order of the propositions and their arguments). We mentioned that plans are modeled as action state sequences. SET-PR does not store the propositional representation of an action state sequence $\mathbb{s}$. Instead, $\mathbb{s}$ is encoded as an action sequence graph $\mathcal{E}^{\mathbb{s}}$ and stored in $c.\pi$. Similarly an input sequence $\mathbb{s}^{query}$ is also encoded as an action sequence graph $\mathcal{E}^{\mathbb{s}^{query}}$ and used in retrieval.

A labeled directed graph G is a 3-tuple $G = (V, E, \lambda)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $\lambda\colon V \cup E \to 2^L$ assigns labels to vertices and edges. Here, an edge $e = [v, u] \in E$ is directed from v to u, where v is the edge's source node and u is the target node. Also, L is a finite set of symbolic labels and 2^L is a set of all the multisets on L; this permits multiple non-unique labels for a node or edge.

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

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

Definition: Given ground atom $\boldsymbol{p}$ representing an action $\boldsymbol{a}$ or a 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 $\mathcal{E}^p(\boldsymbol{p}) = \left(V_p, E_p, \lambda_p\right)$ where:

$$V_p = \begin{cases} \left\{A_{k_p}, o_1, \ldots, o_n\right\}, \text{if } \boldsymbol{p} \text{ is an action} \\ \left\{S_{k_p}, o_1, \ldots, o_n\right\}, \text{if } \boldsymbol{p} \text{ is a state fact} \end{cases}$$

$$E_p = \begin{cases} \left[A_{k_p}, o_1\right] \bigcup \bigcup_{i=1,n-1; j=i+1,n} \left[o_i, o_j\right] \text{ if } \boldsymbol{p} \text{ is an action} \\ \left[S_{k_p}, o_1\right] \bigcup \bigcup_{i=1,n-1; j=i+1,n} \left[o_i, o_j\right] \text{ if } \boldsymbol{p} \text{ is a state fact} \end{cases}$$

$$\lambda_p\left(A_{k_p}\right) = \left\{A_{k_p}\right\};\ \lambda_p\left(S_{k_p}\right) = \left\{S_{k_p}\right\};\ \lambda_p(o_i) = \{t_i\} \text{ for } i = 1, \ldots, n$$

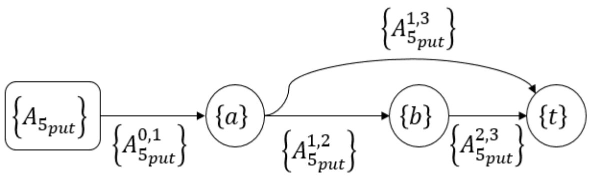
$$\lambda_p\left(\left[A_{k_p}, o_1\right]\right) = \left\{A_{k_p}^{0,1}\right\};\ \lambda_p\left(\left[S_{k_p}, o_1\right]\right) = \left\{S_{k_p}^{0,1}\right\};$$

$$\forall\left[o_i, o_j\right] \in E_p, \lambda_p\left(\left[o_i, o_j\right]\right) = \begin{cases} \left\{A_{k_p}^{i,j}\right\}, \text{if } \boldsymbol{p} \text{ is an action} \\ \left\{S_{k_p}^{i,j}\right\}, \text{if } \boldsymbol{p} \text{ is a state fact} \end{cases}$$

Interpretation: Suppose we have a ground literal $\boldsymbol{p} = p(o_1{:}\,t_1, \ldots, o_n{:}\,t_n)$. 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 $\left\{A_{k_p}\right\}$ or $\left\{S_{k_p}\right\}$). Suppose it is an action predicate. A_{k_p} is then connected to the second node of this graph, the object node o_1 (labeled $\{t_1\}$), through the edge $\left[A_{k_p}, o_1\right]$ (labeled $\left\{A_{k_p}^{0,1}\right\}$). Next, o_1 is connected to the third node o_2 (labeled $\{t_2\}$) through the edge $[o_1, o_2]$ (labeled $\left\{A_{k_p}^{1,2}\right\}$), then to the fourth node o_3 (labeled $\{t_3\}$) through the edge $[o_1, o_3]$ (labeled $\left\{A_{k_p}^{1,3}\right\}$), and so on. Suppose also the third node o_2 is connected to o_3 through $A_{k_p}^{2,3}$, to o_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 $\left\{A_{5_{put}}\right\}$, $\{a\}$, $\{b\}$, and $\{t\}$. The edges are $\left[A_{5_{put}}, a\right]$, $[a, b]$, $[a, t]$, and

$[b, t]$, with respective labels $\left\{A^{0,1}_{5_{put}}\right\}$, $\left\{A^{1,2}_{5_{put}}\right\}$, $\left\{A^{1,3}_{5_{put}}\right\}$, and $\left\{A^{2,3}_{5_{put}}\right\}$. The predicate encoding graph for $\boldsymbol{p}$ is shown in Figure 1.



$\left\{A^{1,3}_{5_{put}}\right\}$
$\left\{A_{5_{put}}\right\}$
$\{a\}$
$\{b\}$
$\{t\}$
$\left\{A^{0,1}_{5_{put}}\right\}$
$\left\{A^{1,2}_{5_{put}}\right\}$
$\left\{A^{2,3}_{5_{put}}\right\}$

Figure 1: A predicate encoding graph corresponding to $\boldsymbol{p} = put(block\colon a, block\colon b, table\colon t)$

Definition: An action sequence graph of an action state sequence $\mathbb{s}$ is a labeled directed graph $\mathcal{E}^{\mathbb{s}} = \bigcup_{(\boldsymbol{a},\boldsymbol{s})\in\mathbb{s}}\left(\mathcal{E}(\boldsymbol{a}) \cup_{\boldsymbol{p}\in\boldsymbol{s}} \mathcal{E}(\boldsymbol{p})\right)$, a union of the predicate encoding graphs of the action and state facts in $\mathbb{s}$.

Figure 2 shows an example of a sample action state sequence and its corresponding action sequence graph.

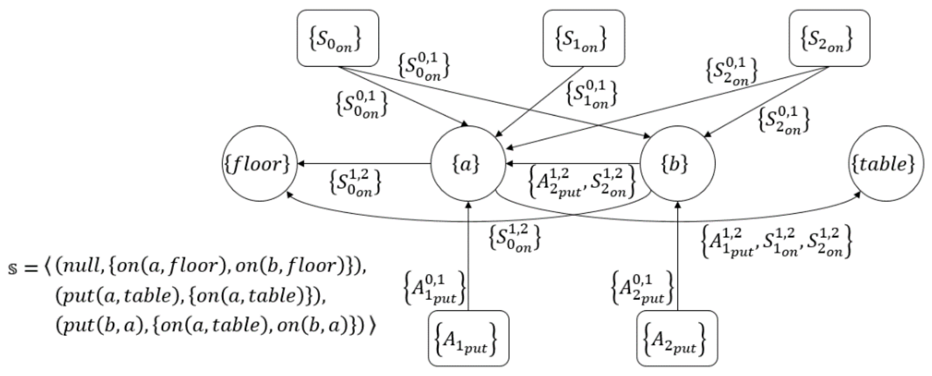


Figure 2: An example action-state sequence $\mathbb{s}$ and corresponding action sequence graph $\mathcal{E}^{\mathbb{s}}$

3.2 Retrieval

Case retrieval requires a similarity metric. Because we represent the input and the stored plans as graphs, our metric uses graph matching; this can be formulated as the task of computing the maximum common subgraph (MCS) of two graphs. Computing the MCS between two or more graphs is an NP-Complete problem, restricting applicability to only small plan recognition problems. Alternatively, a plethora of approximate graph similarity measures exist. For example, similarity metrics that compute graph degree sequences have been used successfully to match chemical structures (Raymond & Willett, 2002).

In SET-PR, we use one of the degree sequence similarity metrics called Johnson's similarity metric (Johnson, 1985). This metric, denoted as sim_{str}, computes the similarity between plans based on the approximate structural similarity of their graph representations. Previously, we tried alternative degree sequence metrics and found that Johnson's metric performed the best (Vattam, Aha & Maynord, 2015).

Let G_1 and G_2 be the two action-sequence graphs. To compute their similarity, we first separate the set of vertices in each graph into l partitions by label type, and then sort them in a non-increasing total order by degree (of a vertex v is the number of edges that touch v). 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 can then be computed as:

$\left|\text{mcs}(G_{1,}G_2)\right| = V(G_1, G_2) + E(G_1, G_2)$, where

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

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

where $v_1^{i,j}$ denotes the jth vertex of the L_1^i sorted degree sequence, and $E\left(v_1^{i,j}\right)$ denotes the set of edges connected to $v_1^{i,j}$. Johnson's similarity metric is given by:

$$sim_{str}(G_1, G_2) = \frac{\left(\left|\text{mcs}(G_{1,}G_2)\right|\right)^2}{|G_1| \cdot |G_2|}$$

Two plans that are similar in structure can differ drastically in semantics. For instance, a plan to travel to a grocery store to buy milk might coincidentally be structurally similar to a plan to travel to the airport to receive a visitor. To mitigate this issue, we use a weighted combination of structural and semantic similarity, denoted as sim_{obj}, as our final similarity metric:

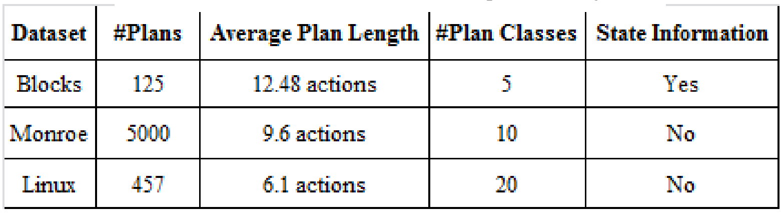
$$\text{sim}(G_1, G_2) = \alpha\, \text{sim}_{str}(G_1, G_2) + (1 - \alpha)\text{sim}_{obj}(G_1, G_2),$$

where $\text{sim}_{obj}(G_1, G_2) = \frac{O_{\mathbb{s}} \cap O_{\pi_i}}{O_{\mathbb{s}} \cup O_{\pi_i}}$ is the Jaccard coefficient of the set of (grounded) objects in G_1 and G_2, and α $(0 \leq \alpha \leq 1)$ governs the weights for sim_{str} and sim_{obj}.

SET-PR matches an input action-sequence graph $\mathbb{s}^{\boldsymbol{query}}$ with each case $c = (\pi, g)$ in C using $\text{sim}(\mathbb{s}^{\boldsymbol{query}}, c.\pi)$, and retrieves the top-ranked matching case (k=1 in k-NN). This case's plan is output as the recognized plan and its goal is output as the recognized goal.

SET-PR keeps track of its most recent previous prediction and uses it to resolve ambiguity if multiple cases are retrieved with nearly similar scores. In other words, selection preference favors a case that maintains continuity in plan prediction. If none of the cases in that set match the previous prediction, then one of them is selected randomly.

Table 1. Datasets used in this empirical study



Dataset	#Plans	Average Plan Length	#Plan Classes	State Information
Blocks	125	12.48 actions	5	Yes
Monroe	5000	9.6 actions	10	No
Linux	457	6.1 actions	20	No

3.3 Error Tolerance

The ability of SET-PR to tolerate input errors is a direct benefit of its representation and retrieval mechanism. By adding state information to plan representation, SET-PR reduces the overreliance on action information (which causes poor performance when they are error-prone) and increases the total amount of information that is used for recognition. SET-PR's graph representation of plans permits inexact matching, trading off higher recall for lower precision. We claim that this tradeoff allows SET-PR to generalize better in the presence of input errors compared to other approaches that favor propositional representations and symbol matching, and test this in Section 4.

4 Empirical Evaluation

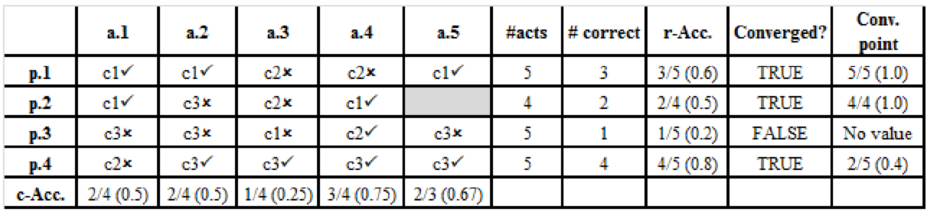
We empirically test the following claim: for the task of plan recognition, SET-PR's approach, which employs a graph-based representation and similarity metric, offers more robustness to input errors compared to alternative CBPR approaches that use propositional representation and symbol matching. We test this claim by subjecting the approaches to increasing levels of input errors. At each level, we measure and compare their plan recognition performance. We perform this experiment across three different plan recognition datasets and note if similar performance trends emerge.

In the following sections we describe the approaches tested, performance metrics, datasets used, methodology, the results and their analysis.

4.1 Compared CBPR Approaches

1. **SET-PR**: This approach uses action sequence graph representation and Johnson's similarity metric for performing plan recogntion as described above.
2. **EDIT**: This approach uses propositional representation and an ordered symbolic similarity metric for performing plan recognition. It treats inputs and plans as symbol sequences and computes their Edit distance (Levenshtein 1966).
3. **JACC**: This approach uses propositional representation and an unordered symbolic similarity metric for performing plan recognition. It treats inputs and plans as a set of action propositions (A_1 and A_2) and computes their Jaccard distance $(1 - (A_1 \cap A_2 / A_1 \cup A_2))$.
4. **RAND**: This is the baseline condition. It performs plan recognition by randomly selecting a plan from the plan library in response to its inputs.

Table 2. A sample convergence matrix



	a.1	a.2	a.3	a.4	a.5	#acts	# correct	r-Acc.	Converged?	Conv. point
p.1	c1✓	c1✓	c2✗	c2✗	c1✓	5	3	3/5 (0.6)	TRUE	5/5 (1.0)
p.2	c1✓	c3✗	c2✗	c1✓		4	2	2/4 (0.5)	TRUE	4/4 (1.0)
p.3	c3✗	c3✗	c1✗	c2✓	c3✗	5	1	1/5 (0.2)	FALSE	No value
p.4	c2✗	c3✓	c3✓	c3✓	c3✓	5	4	4/5 (0.8)	TRUE	2/5 (0.4)
c-Acc.	2/4 (0.5)	2/4 (0.5)	1/4 (0.25)	3/4 (0.75)	2/3 (0.67)					

4.2 Datasets Used

In this study we used three datasets (Table 1). We repeated our evaluation method described below in all three datasets. Blocks World is a synthetic dataset that we generated using the HTN planner SHOP2 (Nau et al., 2003), which we modified to capture state information in the generated plans. Monroe (Blaylock & Allen, 2005) and Linux (Blaylock & Allen, 2004) are two datasets that are commonly used to assess plan recognizers. Monroe is also a synthetic dataset generated using SHOP2, while Linux is a corpus of plans collected from human users performing assigned tasks. Because the plans in these latter two datasets contain no state information, SET-PR's plans also contain only action information. This reduces the size of the encoding of the plans in these two datasets.

4.3 Evaluation Method

For each dataset, we developed an error simulator that takes as input a plan (π), an error-type (t), and an error-percentage (p). It outputs π^{err}, which contains p% errors of type t. The values for t include mislabeled (MLAB), missing (MSNG), extraneous (EXTR), and mixed (MXD). For MLAB, a specified percentage of actions was randomly chosen, and each was replaced with another action randomly chosen from the domain. For MSNG, a percentage of actions was randomly chosen, and each was replaced with an unidentified marker '*'. For EXTR, a percentage of randomly chosen actions from the domain were introduced at random locations in the plan. For MXD, a uniform distribution of all three types of errors was introduced.

For each dataset D, we obtained a set of datasets $D_{t,p}$ that combine t = {MLAB, MSNG, EXTR, MXD} and p = {0, 0.15, 0.3, 0.45, 0.6}. For example, $Monroe_{MXD,0.6}$ is a Monroe version containing plans with 60% mixed error.

For each $D_{t,p}$, we tested our compared conditions (SET-PR, EDIT, JACC, and RAND) using five-fold cross-validation (with shuffle). That is, for each plan in the test set, we *incrementally queried* the training set to predict a plan. For example, if a test plan had four actions $\{a1, a2, a3, a4\}$, the evaluator performed 4 queries $\{a1\}$, $\{a1, a2\}$, $\{a1, a2, a3\}$, and $\{a1, a2, a3, a4\}$ to obtain a predicted plan after observing each action in succession. For a prediction to be correct, the plan class of the predicted plan must match the plan class of the test plan.

Table 3. Confusion matrix

Actl. \ Pred.	c1	c2	c3	Recall
c1	5	3	1	5/9
c2	1	1	3	1/5
c3	0	1	4	4/5
Precision	5/6	1/5	4/8	

For each compared condition for each $D_{t,p}$, the results of the cross validation was tabulated in a *convergence matrix* (example in Table 2). The rows in matrix are plan indices and columns are action indices. After observing the j^{th} action of the i^{th} plan in the test set, *cell*$_{ij}$ registers (1) the predicted value (the goal class), and (2) a Boolean value indicating a correct or incorrect prediction. We maintain two additional columns, total number of actions (#acts) per row and correct predictions (#correct) per row.

From the convergence matrix, we derive a *confusion matrix* (Table 3) by counting the instances where the predicted plan class agrees or disagrees with the actual class.

4.4 Performance Metrics

We defined the following four plan recognition performance metrics from the convergence matrix depicted in Table 2:

Percent convergence: Convergence indicates whether the final prediction in each row was correct. For each condition, the percentage of True values is computed.

Convergence point: If a prediction converged, the convergence point (CP) is the point in the input that the recognizer starts to output only the correct prediction. A smaller value for this metric indicates a better performance. For each condition, we also compute the average convergence point.

r-Accuracy: We can compute row-wise prediction accuracy as the ratio of the total number of correct predictions in a row versus the total number of actions observed in that row (#correct/#acts). We compute r-Accuracy as the average of this value for each test plan. It has often been referred to as "precision" (e.g., Blaylock & Allen 2006) in plan recognition literature, which differs from the traditional meaning of precision in the general classification literature. Table 3 displays the traditionally-defined precision and recall values from the confusion matrix.

c-Accuracy: We calculate column-wise prediction accuracy as the ratio of the total number of or correct predictions in a column versus its total number of plans (#correct in col/#plans in col). c-Accuracy is the average of these values.

From the confusion matrix depicted in Table 3, we define a final performance metric, ***F1-Score***, which is the harmonic mean of average precision and recall.

4.5 Results

Figure 3 shows the plots for percent convergence and convergence point of SET-PR, EDIT, JACC, and RAND at varying levels of input errors of type MXD (mixed error) for the three datasets. These are mean values averaged over five-fold cross validation. Similarly, Figure 4 shows the plots for mean *r-Accuracy* and *c-Accuracy*, and Figure 5 shows plots for the mean *F1-Score*. Due to space restrictions, we do not show the plots and other significance test results for other error types. However, we note that the trends observed in MXD hold for other error conditions as well. We highlight MXD because it contains a uniform distribution of the three kinds of errors (we chose uniform distribution because we currently lack domain-specific error models).

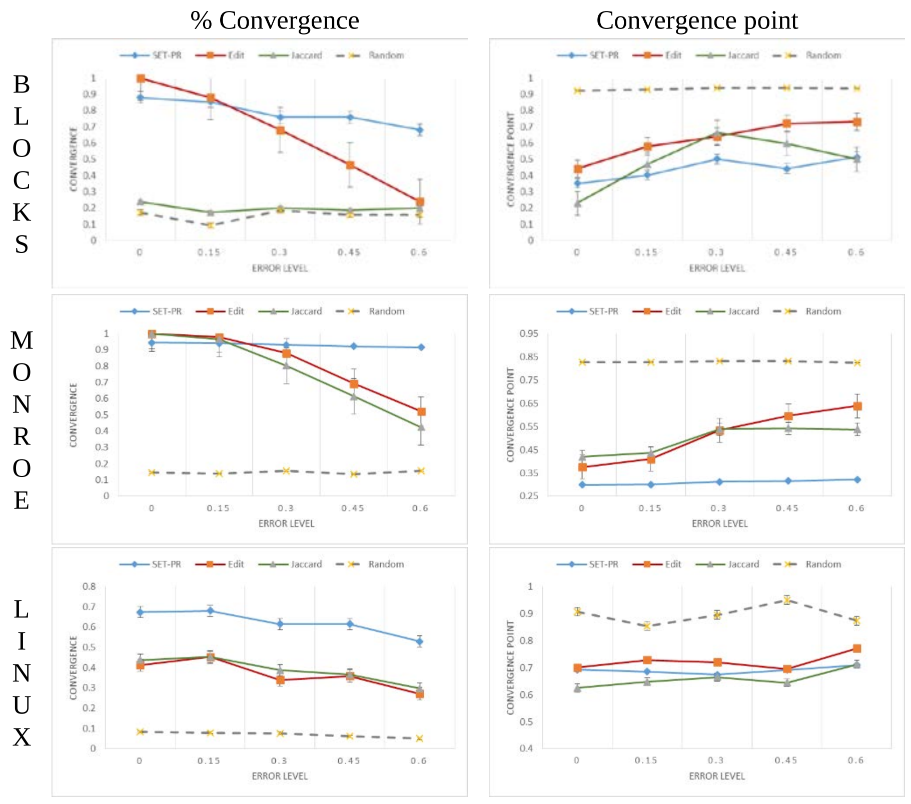


Figure 3. Percent Convergence and Convergence-point vs. Error level for the 3 datasets

4.6 Analysis

At 0% error level, the plots in Figures 3, 4 and 5 indicate the following. (1) For Blocks: with the exception of one metric, EDIT and SET-PR perform comparably, but JACC performs poorly, closer to RAND. (2) For Monroe: with the exception of one metric, all three perform comparably. (3) For Linux: SET-PR shows performance advantage in 3 out of 5 metrics, while EDIT and JACC perform comparably with each other. Overall, for 0% error, in majority of the experiments, SET-PR’s performance is comparable to EDIT, JACC or both.

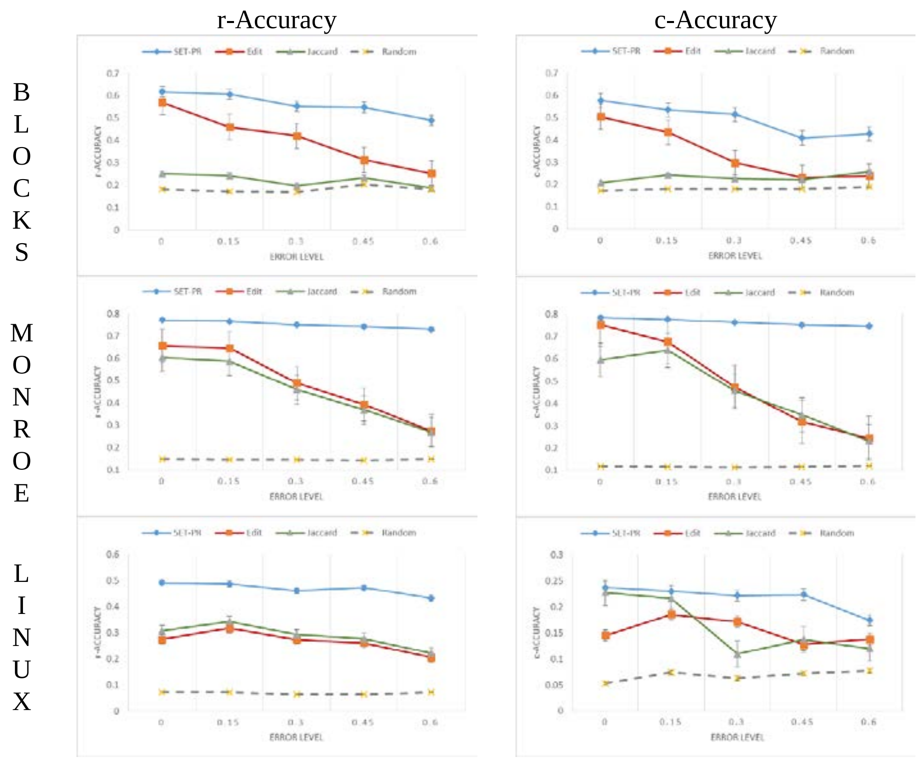


Figure 4. Mean r-Accuracy and c-Accuracy vs. Error level for the 3 datasets

At 15% error level, we see small to negligible performance declines for SET-PR, but more declines for EDIT and JACC. Finally, at higher levels of error, we see moderate declines in performance for SET-PR, but steep declines for EDIT and JACC. This trend can be observed across datasets and across different error types in a majority of experiments.

To assess the impact of the two independent factors (CBPR approach and error level) on the value of a performance metric, we compared the means of the performance metric values across these two factors. For each dataset and for each error type, we subjected this two factor data to a two-way ANOVA test to measure the statistical significance of the outcomes of the comparison, amounting to a total of 60 tests. In all 60 tests, there was a statistically significant effect observed for both factors as well as for their interaction (p < 0.05 for all tests; for *error level*, F(4,59) ranged between 650 and 2229; for *CBPR approach*, F(3,59) ranged between 1042 and 17654; and their *interaction* factor, F(12,59) ranged between 100 and 409).

From these results we can conclude that SET-PR has a *significantly* higher tolerance for the three kinds of input errors compared to EDIT and JACC although the latter two can perform similarly to or outperform SET-PR in the 0% error condition.

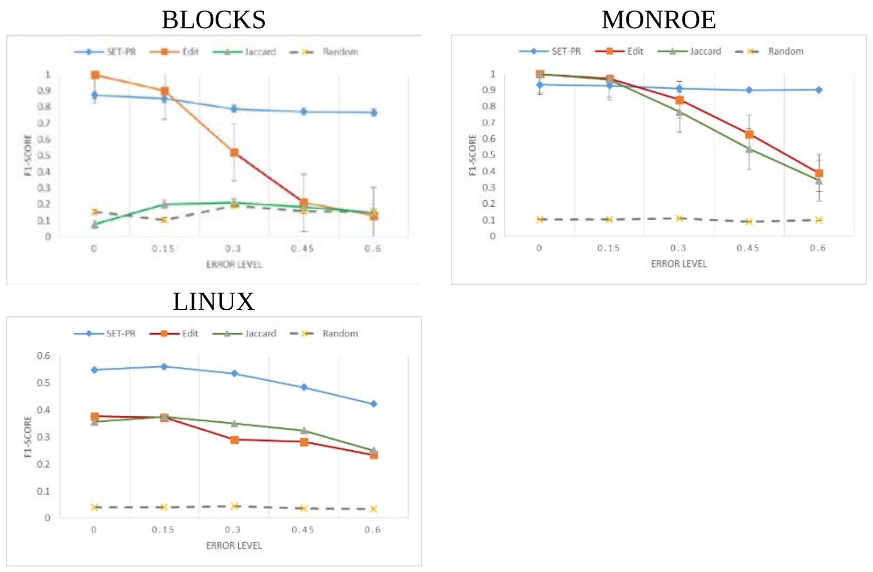


Figure 5: Mean F1-Score vs. Error level for the 3 datasets

5 Discussion

In Section 3.3 we argued that the superior performance of SET-PR under imperfect observability can be attributed to two factors: (1) the content of the plans, which includes action and state information, and (2) the graph representation of the plans, which permits inexact matching. Our evaluation lends support to (2) because only SET-PR uses graph representations. Regarding (1), in our earlier pilot studies with Blocks world (Vattam, Aha, & Floyd 2015), we compared SET-PR with and without state information, keeping all else constant. There, we found preliminary evidence to support (1), but our current investigation does not focus on (1) because no state information is included in SET-PR for the Monroe and Linux plan libraries.

One of the limitations of our study is that we do not compare SET-PR with other state-of-the-art plan recognizers. In the future, we plan to obtain and run these experiments with other well-known plan recognizers.

Given that graph matching is generally considered a hard problem, what can we say about the computational efficiency of SET-PR's matching process? Using its degree sequence metric, others showed that the similarity between two graphs can be computed in $O(n \cdot log\, n)$ time, where $n = max_i\left(|L_1^i|, |L_2^i|\right)$ (Raymond & Willett, 2002).

Without efficient indexing techniques, plan retrieval time scales linearly with the size of SET-PR's library C. This can be prohibitively expensive for online plan recognition. Thus, we use Plan Projection Clustering (PPC), a method to increase the plan retrieval speed of SET-PR (Maynord, Vattam, & Aha 2015). PPC is a domain-general approach for organizing SET-PR's plans in a hierarchy. It employs a metric d (e.g., Johnson's similarity metric) that measures distances among plans. PPC computes $d(p_1, p_2)$ for each pair of plans $p_1, p_2 \in C$ to produce a distance matrix M. PPC then projects M into N-dimensional Euclidean space by applying multi-dimensional scaling (Kruskal, 1964). All cases are placed into a single group constituting the top level of a hierarchy. We then recursively apply a clustering algorithm m to these cases until the desired depth of the hierarchy, k, is reached. Hyper-parameters d, N, m, and k can be tuned for optimal performance.

PPC processes a query q by recursively matching it down the hierarchy. At each level, d is used to determine the distance between q and the case c closest to each candidate cluster center. At each step, q is matched to a cluster for which this distance is smallest. Once a leaf is reached, q's nearest neighbor is retrieved.

Our pilot study (Maynord, Vattam, & Aha 2015) indicated that PPC can reduce retrieval time by up to 72% while sacrificing only a small amount in retrieval accuracy (approximately 4%), because queries are partial (rather than complete) plans.

6 Conclusions and Future Work

We described SET-PR, a case-based plan recognition algorithm that represents plans as action sequence graphs. Unlike most prior algorithms, we designed SET-PR to be tolerant of input errors in the observed actions (i.e., missing, mislabeled, or extra action labels). We use Johnson's (1985) similarity metric for plan retrieval in SET-PR because it is an approximation of the maximal common subgraph function for matching graphs. In our empirical studies on plan recognition tasks involving three data sets, which we modified by adding input errors, we compared the performance of SET-PR with alternative approaches that use propositional representation and similarity functions for plan retrieval. We found that SET-PR's use of a graph representation for plans contributed to its superior performance when error rates are high. This complements earlier work (Vattam, Aha, & Floyd 2015) that showed the incorporation of state information in its plan representation is another contributing factor.

In our future work, we will compare the performance of SET-PR versus other state-of-the-art plan recognition algorithms. We also plan to investigate more sophisticated graph similarity functions (e.g., graph kernels) and compare them versus SET-PR's current similarity function. Current plan recognizers, including SET-PR, assume that the observed actor's plans remains static during plan recognition. We will relax this assumption and extend SET-PR to tolerate dynamics changes to an actor's plans. Finally, we plan to integrate SET-PR with sensory perceptual systems on simulated and real robotic platforms so that we can study its performance on the ground in real time.

Acknowledgements

Thanks to OSD ASD (R&E) for sponsoring this research. Swaroop Vattam performed this work while an NRC post-doctoral research associate at NRL. Thanks also to the reviewers for their comments. 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 of NRL or OSD.

References

Bauer, M. (1994). Integrating probabilistic reasoning into plan recognition. In *Proceedings of the Eleventh European Conference on Artificial Intelligence* (pp. 620-624). Amsterdam, The Netherlands: Wiley & Sons.

Blaylock, N. & Allen, J. (2004). Statistical goal parameter recognition. *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling* (pp 297-304). Whistler, BC, Canada.

Blaylock, N., & Allen, J. (2005). Generating artificial corpora for plan recognition. *User Modeling* (pp. 179-188). Berlin, Germany: Springer.

Blaylock, N., & Allen, J. (2006). Hierarchical instantiated goal recognition. In G. Kaminka, D. Pynadath, & C. Geib (Eds.) *Modeling Others from Observations: Papers from the AAAI Workshop* (Technical Report WS-06-13). Boston, MA: AAAI Press.

Charniak, E., & Goldman, R.P. (1993). A Bayesian model of plan recognition. *Artificial Intelligence*, *64*(1), 53-79.

Cox, M.T., & Kerkez, B. (2006). Case-based plan recognition with novel input. *Control and intelligent systems*, *34*(2), 96-104.

Duong, T. V., Bui, H.H., Phung, D.Q. & Venkatesh, S. (2005). Activity recognition and abnormality detection with the switching hidden semi-Markov model. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 838–845). San Diego, CA: IEEE Press.

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

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

Kautz, H., & Allen, J. F. (1986). Generalized plan recognition. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 32–37). Philadelphia, PA: AAAI Press.

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*(8), 707-710.

Maynord, M., Vattam, S., & Aha, D.W. (2015). Increasing the runtime speed of case-based plan recognition. To appear in *Proceedings of the Twenty-Eighth Florida Artificial Intelligence Research Society Conference*. Hollywood, FL: AAAI Press.

Nau, D. S., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379-404.

Pynadath, D. V. & Wellman, M. P. (1995). Accounting for context in plan recognition with application to traffic monitoring. *Proceedings of Uncertainty in Artificial Intelligence* (pp. 472–481). Montreal, Quebec: Morgan Kaufmann.

Ramirez, M., & Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence*. Atlanta, GA: AAAI Press.

Ramirez, M., & Geffner, H. (2011). Goal recognition over POMDPs: Inferring the intention of a POMDP agent. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (pp. 2009-2014). Barcelona, Spain: AAAI Press.

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

Sánchez-Ruiz, A. A., & Ontañón, S. (2014). Least common subsumer trees for plan retrieval. *Proceedings of the Twenty-Second International Conference on Case-Based Reasoning* (pp. 405-419). Cork, Ireland: Springer.

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

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

van Beek, P. (1996). An investigation of probabilistic interpretations of heuristics in plan recognition. *Proceedings of the Fifth International Conference on User Modeling* (pp. 113-120).

Vattam, S., Aha, D.W., & Floyd, M. (2015). Error tolerant plan recognition: An empirical investigation. To appear in *Proceedings of the Twenty-Eighth Florida Artificial Intelligence Research Society Conference*. Hollywood, FL: AAAI Press.

Vattam, S.S., Aha, D.W., & Floyd, M. (2014). Case-based plan recognition using action sequence graphs. *Proceedings of the Twenty-Second International Conference on Case-Based Reasoning* (pp. 495-510). Cork, Ireland: Springer.