State Projection via AI Planning

Shirin Sohrabi and **Anton V. Riabov** and **Octavian Udrea**
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA
{ssohrab, riabov, udrea}@us.ibm.com

Abstract

Imagining the future helps anticipate and prepare for what is coming. This has great importance to many, if not all, human endeavors. In this paper, we develop the Planning Projector system prototype, which applies plan-recognition-as-planning technique to both explain the observations derived from analyzing relevant news and social media, and project a range of possible future state trajectories for human review. Unlike the plan recognition problem, where a set of goals, and often a plan library must be given as part of the input, the Planning Projector system takes as input the domain knowledge, a sequence of observations derived from the news, a time horizon, and the number of trajectories to produce. It then computes the set of trajectories by applying a planner capable of finding a set of high-quality plans on a transformed planning problem. The Planning Projector prototype integrates several components including: (1) knowledge engineering: the process of encoding the domain knowledge from domain experts; (2) data transformation: the problem of analyzing and transforming the raw data into a sequence of observations; (3) trajectory computation: characterizing the future state projection problem and computing a set of trajectories; (4) user interface: clustering and visualizing the trajectories. We evaluate our approach qualitatively and conclude that the Planning Projector helps users understand future possibilities so that they can make more informed decisions.

Introduction

Being prepared for the future is of great importance to many, if not all, human endeavors. In this paper, we develop the Planning Projector system prototype that helps users (e.g., analysts) to not only explain the past, based on the observed events, but also project the future by providing a range of possible trajectories, where each trajectory consists of a future state, and an explanation (i.e., an action sequence) that led to that state. The Planning Projector applies plan-recognition-as-planning technique to explain the observations, derived from relevant news and social media posts, and projects alternative trajectories, in accordance with a planning domain definition that captures the expert knowledge. Hence, the Planning Projector infers (or recognizes) plans for the past and projects the future.

Consider the following energy domain example, where the objective is to project the price of oil and volume of oil produced 2 years into the future. Our objective is not to find a precise estimate of the price of oil, but rather to project the possible range of values and provide an explanation that led to those values. The Planning Projector relies on domain knowledge that can either be provided by the domain experts, or encoded by non-experts after reviewing various sources of available knowledge, such as research papers, textbooks or Wikipedia. The domain knowledge in this example describes possible actions affecting the oil price directly or indirectly, for example by affecting supply levels. For instance, the decision of the leaders of Organization of the Petroleum Exporting Countries (OPEC) to meet is an action that is likely to affect both the price and the supply of oil, depending on the outcome of such a meeting. The decision to limit production will decrease the supply and increase the price, or the decision to increase production can lead to lower prices. The observations associated with these actions, confirming or contradicting them, can be derived from news reports. Similarly, several other events or actions can be modeled: the discovery of a new oil field, drilling activity in known fields, hurricanes or other natural disasters affecting oil production, and changes in currency rates.

We present the Planning Projector system prototype that integrates multiple components in order to explain the observations and generate future trajectories. The components include: (1) knowledge engineering: the process to encode the domain knowledge in planning terms; (2) data transformation: the component required to transform the raw data into a sequence of observations; (3) trajectory computation: the component responsible to compute a set of trajectories given no planning goals as the input; (4) user interface: the component needed in order to present the space of possibilities in a compact and efficient way.

Figure 1 shows the system architecture of the Planning Projector. The system takes as input news articles and social media posts, analyses and transforms the raw data via the *Data Transformation* component and sends selected analytic results as a sequence of observations into the *Trajectory Computation* component, which produces a set of trajectories. The *Knowledge Engineering* component ensures that the required domain knowledge is captured and available to the *Trajectory Computation* component. The *User Interface*

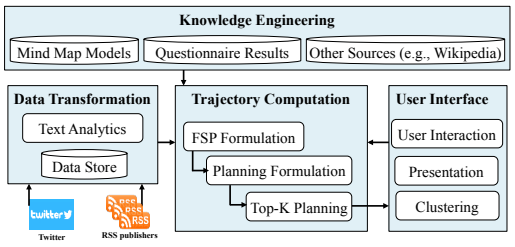

Figure 1: Planning Projector system architecture

component interacts with the user to obtain additional inputs such as the number of trajectories to compute and a time horizon and presents the results.

The *Knowledge Engineering* component addresses one of the main bottlenecks of applying AI planning technology, the often manual process of obtaining the domain knowledge from domain experts and encoding it in a planning language. While it is generally assumed that a planning domain is given as an input that may not be possible in general. That is because the domain expert often has no AI planning background and is not able to encode the knowledge in a planning language. Instead the domain expert might be able to capture the knowledge in a variety of tools available and known to them. For example, the domain expert may be able to easily encode the domain knowledge in a diagramming tool such as Mind Maps (Faste and Lin 2012). The system prototype we have developed is then able to automatically translate the available domain knowledge captured in Mind Maps into a planning language. Further, we have developed a questionnaire in order to obtain additional domain knowledge, such as the weights of the edges on the Mind Maps from the domain experts. It is also possible that domain knowledge is extracted from online sources such as articles, textbooks or Wikipedia or is learned (e.g., (Yang, Wu, and Jiang 2007; Zhuo, Nguyen, and Kambhampati 2013)).

The *Trajectory Computation* component is responsible for computing the trajectories given the domain knowledge and a sequence of observations, which can be unreliable (i.e., missing or unexplainable). To this end, it first formulates the Future State Projection (FSP) problem and then applies the plan-recognition-as-planning theory that addresses unreliable observations to transform the FSP problem into a planning problem (Sohrabi, Riabov, and Udrea 2016; Ramírez and Geffner 2010). To compute the trajectories, the system computes a set of top-k plans (Riabov, Sohrabi, and Udrea 2014) given the time horizon and the sequence of observations. The *User Interface* component post-processes the set of produced plans and presents the set of trajectories to the user. It also applies clustering and plan similarity metrics (Nguyen et al. 2012; Sohrabi et al. 2016) to better visualize similar plans and produce the summary reports.

Background: Plan Recognition as Planning

In this section, we briefly review the definition of AI planning problem and plan recognition problem. We borrow terminology from our recent work, (Sohrabi, Riabov, and Udrea 2016), which in turn builds on the work of Ramírez and Geffner 2010/2009 to address observations over fluents (i.e., observations over properties of a state), infer both goals as well as plans, and explicitly address unreliable observations in the theory and transformation to planning.

Definition 1 *A planning problem is a tuple* $P = (F, A, I, G)$*, where* F *is a finite set of fluent symbols,* A *is a set of actions with preconditions,* $\text{PRE}(a)$*, add effects,* $\text{ADD}(a)$*, delete effects,* $\text{DEL}(a)$*, and action costs,* $\text{COST}(a)$*,* $I \subseteq F$ *defines the initial state, and* $G \subseteq F$ *defines the goal state.*

A state, s, is a set of fluents that are true. An action a is *executable* in a state s if $\text{PRE}(a) \subseteq s$. The successor state is defined as $\delta(a, s) = ((s \backslash \text{DEL}(a)) \cup \text{ADD}(a))$ for the executable actions. The sequence of actions $\pi = [a_0, ..., a_n]$ is executable in s if state $s' = \delta(a_n, \delta(a_{n-1}, \ldots, \delta(a_0, s)))$ is defined. Moreover, π is the solution to P if it is executable from the initial state and $G \subseteq \delta(a_n, \delta(a_{n-1}, \ldots, \delta(a_0, I)))$.

Definition 2 *A Plan Recognition problem is a tuple* $PR = (F, A, I, O, \mathcal{G}, \text{PROB})$*, where* (F, A, I) *is the planning domain as defined above,* $O = [o_1, ..., o_m]$*, where* $o_i \in F$*,* $i \in [1, m]$ *is the sequence of observations,* $\mathcal{G}$ *is the set of possible goals* G*,* $G \subseteq F$*, and* PROB *is the goal priors,* $P(G)$*.*

Unexplainable or noisy observations are defined as those that have not been added to the state as a result of an effect of any of the actions in a plan for a particular goal, while missing observations are those that are added to the state but are not observed (i.e., are not part of the observation sequence). To address the noisy observations, the definition of satisfaction of an observation sequence by an action sequence is modified to allow for observations to be left unexplained. Given an execution trace and an action sequence, an observation sequence is said to be satisfied by an action sequence and its execution trace if there is a non-decreasing function that maps the observation indices into the state indices as either explained or discarded.

The solution to the PR problem is the posterior probabilities, the probability of a plan given the observations, $P(\pi|O)$, and the probability of a goal given the observations, $P(G|O)$. $P(\pi|O)$ can be approximated by normalizing three objectives over a set of sample plans: (1) cost of the original actions, (2) number of missing observations, and (3) number of noisy observations. $P(G|O)$ can be computed by a summation over $P(\pi|O)$ for all plans that achieve G and satisfy O. The sampled set of plans can be computed using either diverse planning or top-k planning on the transformed planning problem (Sohrabi, Riabov, and Udrea 2016).

Future State Projection Problem

In this section, we discuss how we extend the plan-recognition-as-planning theory to define the projection problem and how to modify the transformation to planning to allow for explanation of past observations as well as projection of the future. We will also discuss how to compute the trajectories using an AI planner.

Definition 3 *A Future State Projection problem is defined as a tuple $FSP = (F, A, I, O, T, K)$, where (F, A, I) is the planning domain as defined above, $O = [o_1, ..., o_m]$, where $o_i \in F$, $i \in [1, m]$ is the sequence of observations, T is the number of time steps into the future, K is the number of trajectories to produce.*

Similar to the plan recognition problem, each observation is over a fluent rather than an action, as it is often the case that the actions are not directly observable, but their effects through the change in the state of the world are observable. Note that this problem definition does not include a set of possible goals as the input, instead, T, a number of time steps into the future, is given. In other words, we define T as the number of actions that must be executed after the last observation is explained or discarded. Hence, a trajectory that considers T steps into the future, considers T many future actions. However, it is possible that each action can be associated with a duration, and that T can measure the real time. In that case, a temporal planner can be used to compute the trajectories (Benton, Coles, and Coles 2012).

Definition 4 *Given a FSP problem (F, A, I, O, T, K) as defined above, a trajectory is a tuple (s, π), where (1) $\pi = [a_0, ..., a_n, a_{n+1}, ..., a_{n+T}]$ is an action sequence that is executable from the initial state I and results in state $s = \delta(a_{n+T}, \ldots, \delta(a_0, I))$, and (2) the observation sequence O is satisfied by the action sequence $[a_0, ..., a_n]$. A solution to the FSP problem is a collection of K trajectories.*

The trajectory includes the final state s together with its "explanation", π. Each action sequence π, comprises of actions that explain or discard the observations (i.e., $[a_0, ..., a_n]$) and T many future actions reachable in T steps after the last observation, o_m, is either explained or discarded, according to the domain description. While there are many trajectories for a given FSP problem, a trajectory with an action sequence that has the lowest cost, lowest number of missing, and lowest number of noisy observations is more probable. Note that this is the same objective function defined for the PR problem that is used to estimate the posterior probabilities. This objective function also maps to the cost of the plan in the transformed planning problem.

Transformation to Planning

Next, we discuss how to transform the FSP problem into a planning problem with action costs. We build on our recent work but modify it in order to be able to address the different inputs to the problem (Sohrabi, Riabov, and Udrea 2016). The time horizon together with the observations now comprise the goal for the planning problem rather than the set of given goals $\mathcal{G}$ in prior work.

To address generation of future T many actions, we add a special observable fluent φ to the set of fluents, F, and also to the add effect of all the original actions (i.e., for all actions $a \in A$, $\varphi \in \text{ADD}(a)$). We also explicitly modify the sequence of observations O to add T many observations of type φ. This simple modification to O will generate T many future actions based on the planning domain.

We also ensure the order of observations is preserved, that is, first the past observations are explained or discarded, and then T many future observations are explained. To do so, we use a set of special fluents, one for each observation that is set to true if that observation is either explained or discarded by the extra actions, "explain" or "discard", which we add for each observation to the set of actions, A. Note, we disallow discarding the future observation. Each of the "explain" or "discard" actions can only be executed if the previous observation was either explained or discarded. Furthermore, to explain the past observation, the fluent associated with that observation must be true in the state, and to explain the future observation, the special fluent φ must be true in the state, so it must have been added by an action. We also add to the goal state, the special fluent associated with the final observation. This ensures that all of the observations are either explained or discarded. Note, we set the cost of the discard action higher than the explain action to encourage explaining as many observations as possible.

Computing Trajectories via Top-k planning

There are several methods that can be used to compute a set of trajectories; however, we propose the use of a top-k planner that is capable of finding a set of plans with high quality. Top-k planning (Riabov, Sohrabi, and Udrea 2014; Sohrabi et al. 2016) is the problem of finding k plans that have the highest quality. The set of top-k plans can be a set of optimal plans, or a mix of optimal plans and suboptimal plans depending on k. There are several techniques that can be used to compute the top-k plans. In this paper, we use the top-k planning planner, TK*, that is based on the use of a k shortest paths algorithm called K^* (Aljazzar and Leue 2011) because it has been shown that TK* outperforms other techniques for top-k planning. Each plan generated by the top-k planner, π' is post-processed into a trajectory, (s, π) as follows: s is the final state reached by execution of actions in π', and π is constructed by removing the extra actions (i.e., discard, explain) from π'. Hence, the set of K trajectories is computed by calling the top-k planner on the transformed FSP problem and by post-processing the produced K plans.

Knowledge Engineering

While most AI planning systems assume that the domain knowledge is expressed in Planning Domain Description Language (PDDL) (McDermott 1998) or similar, that is often not the case in practice. Several knowledge engineering tools have been developed (e.g., (Simpson, Kitchin, and McCluskey 2007; Vaquero et al. 2007)) with the objective of helping domain experts to write the domain knowledge. While some tools accept as input a language other than PDDL (Vaquero et al. 2007; Riabov et al. 2015), it is often the case that the existing tools assume that the domain expert has some AI planning background and are there to provide an additional support in writing the PDDL language; for example, by providing an editor that has syntax highlighting, error checking, and solver plug-ins (Muise 2016).

The approach we take in this paper is driven by two main requirements: (1) the domain expert may not have any AI

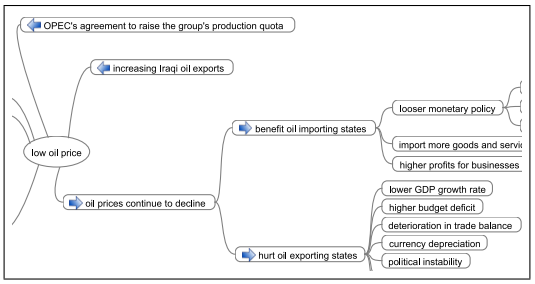

Figure 2: A sample Mind Map

Figure 3: A sample question

planning background and is not willing to write the domain knowledge in PDDL, for example, (2) the domain expert may be comfortable writing their knowledge in a lightweight, already existing graphically tool such as the Mind Maps (Faste and Lin 2012). Mind Maps represent the concepts and their relationship to other concepts in a simple form. A sample Mind Map for the energy domain is shown in Figure 2. In this Mind Map, the concepts are linked by edges, most of which have directions to indicate the cascading effects or the order of which they happen. For example, if OPEC agrees to increase the oil production, then the price of oil may fall. This subsequently may result in looser monetary policy for the oil importing countries, while, it may cause currency depreciation for the oil exporting countries.

The domain knowledge can be encoded by one or more Mind Maps connected by the same concept used in multiple Mind Maps. The Mind Maps can be created in a tool such as FreeMind that produces an XML representation of the Mind Maps, which can serve as an input to our system. The system then translates the Mind Maps into an AI planning problem automatically. This translation is novel and an integral part of our system. The two requirements we impose on the Mind Maps are that the Mind Maps have directions on the edges and that there is a cascading of events in the Mind Maps to enable generation of rich plans.

To generate the planning model, we develop one PDDL domain file with many possible groundings of the actions based on the given Mind Maps. The domain file includes actions that represent the change in the transitions between two concepts. For example, the domain file includes an action "transition ?from ?to", with a precondition "(next ?from ?to)". The problem file then provides the grounding of the actions, as indicated by the edges between two concepts. Following Figure 2, the problem file provides a grounding of the transition action, by the predicate "(next OPECs-agreement-to-raise-the-groups-production-quota low-oil-price)". Special considerations is given to the actions at the root of the Mind Maps using special predicates in order to avoid repeatedly executing the same action multiple times. Note that the size of the Mind Map leads to a larger problem file, as the domain file is fixed.

In addition, we also automatically develop a questionnaire based on the given Mind Maps. A sample question is shown in Figure 3. The answers to the questionnaire provide additional information on the weights of the edges between the two concepts in a Mind Map. This is categorized into three levels, low, medium, and high. The likelihood and impact levels are then encoded as a cost of the transition action in the planning domain, assigning a higher cost/penalty for the "low" option, a medium cost for the "medium" option, and a lower cost for the "high" option. The produced trajectories differ based on the answers from the questionnaire.

Data Transformation

As shown in Figure 1, the input to the system is the raw social media posts from Twitter for example, and/or news articles with RSS feed. The *Data Transformation* component must analyze the large amount of raw data available and select with the help of an analyst a sequence of observations to be fed to the next component in order to compute the trajectories. Observations are the required input in many research areas including but not limited to the model-based diagnosis (Bauer et al. 2011; Cordier and Thiébaux 1994; McIlraith 1994), plan recognition (Ramírez and Geffner 2009; Zhuo, Yang, and Kambhampati 2012), and explanation generation (Sohrabi, Baier, and McIlraith 2011; Sohrabi, Udrea, and Riabov 2013). However, the related work often does not discuss how the observations are obtained from the raw data.

Our implemented system continuously monitors multiple real-world sources (e.g., news channels, social media posts) to identify the sequence of observations. To this end, several text analytics are implemented in order to find the information that is relevant for a particular domain in the vast amount of information available to crawl. We define a set of predefined relevant news sources (e.g., CNN), topics (e.g., oil), and organizations (e.g., OPEC) in order to refine and filter the information. Users/analysts can also add a set of keywords that is important for a particular domain. Analysts then review the generated results and select the observations that are the most relevant and important for them. Note that we do not assume that the observations are perfect since they are coming from various different possibly contradicting sources. It is also possible that a particular important observation was not found and is missing. However, our system can deal with unreliable observations (i.e., noisy and missing observation) and still generate a set of plans.

User Interface

Figure 4 shows the main input screen of the Planning Projector. At the top, the observation sequence is entered. The observations can be entered one by one by selecting from

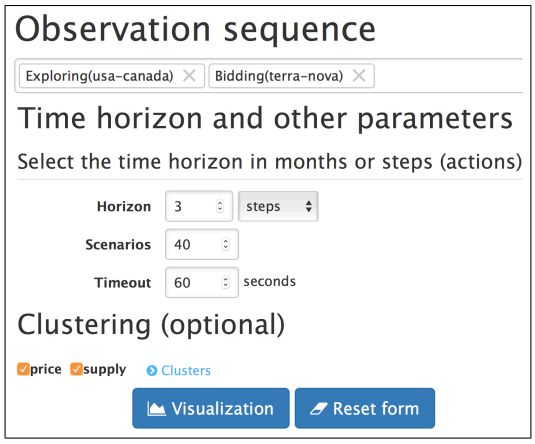

Figure 4: Planning Projector input screen

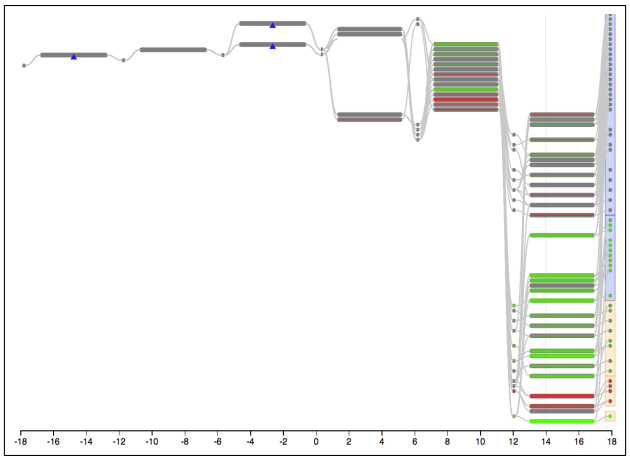

Figure 5: Planning Projector results page

a drop down menu containing known fluents. Note, the analysts may interact with the system through the command line to feed the system observations in real-time and view and manipulate the generated plans, which can be exported in JSON object format, in addition to being shown visually.

The selected observations are of ongoing exploration field in North America, and of bidding on an oil field called Terra Nova from our energy domain. The time horizon, number of trajectories to compute, and the timeout can be specified in the input screen. Optionally the user can express interest in grouping the results in clusters around their values of interest. These numbers can be entered by clicking on "Clusters" at the bottom of the page. The system then post-processes the results and shows clusters of trajectories, by picking the entered values as the representative of the clusters. Furthermore, the system generates additional clusters using a predefined threshold. The users will benefit from seeing clusters of trajectories around other not user-predicted values.

Figure 5 shows the results page produced by our system, which is generated by formulating a FSP problem, translating it to a planning problem, and running a planner to compute a set of plans, as described in the previous sections. The x-axis shows the time horizon, where zero indicates the current time, the positive numbers indicate the future, and the negative numbers indicate the past. Each action in this domain is associated with a duration of six months; hence, the 3 steps into the future, maps to 18 months. Note, our mapping between time steps to months is fixed and for the purpose of visualization only. That is while actions have duration, we do not use a temporal planner and the actions are non-concurrent. Each circle corresponds to a state, and each rectangle represents an action. Hence, the set of trajectories are shown by their state-action sequences. The small blue triangle indicates the action that explained an observation in the past. The user can move the mouse over the states and actions in order to get a more detailed description, including all fluents that are true in the state, action name, duration, and how the action influences the price and supply of oil. The actions are color-coded for a quick visual analysis, where red indicates a decrease in the price of oil and green indicates an increase. The intensity of the colors indicates the degree of the change in price. The color coding can be set in a configuration file and can be modified based on values of interest of a particular domain.

The last states, represented by circles at time point 18, are of a particular interest to the user. These states can be clustered optionally around given values of price or volume of oil. Clusters are computed by considering similarities of plans as defined in previous work that address finding diverse plans (e.g., (Coman and Muñoz-Avila 2011; Nguyen et al. 2012)) combined with known clustering technique (e.g., (Xu and Wunsch 2005; Sohrabi et al. 2016)). There are two kinds of clusters, as shown in dark blue and in light orange. The dark blue clusters are those that are centered on the given values provided by the user, while the light orange clusters are discovered by the system without user guidance. The states in the blue clusters and the state sequences leading to them confirm the prior information that the user may have, while the values in the orange clusters are unexpected. The orange clusters provide valuable information to the users of the system.

Experimental Evaluation

This section is divided into two parts. First, we provide details associated with a use case we have worked on including numbers associated with the various aspect of the domain knowledge, and the raw data. For this use case we had access to analysts who provided us with the Mind Maps and were able to provide keywords to filter the relevant sources in order to obtain the observations. For confidentiality reasons, we cannot elaborate on the use case, but the energy example we discussed throughout the paper is similar in nature to this use case. In the second part, we provide a qualitative evaluation on the computation of plans using different recent plan-recognition-as-planning approaches (Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016) on the energy domain. All our experiments were run on a 2.5 GHz Intel Core i7 processor with 16 GB RAM.

We ran the Planning Projector on a specific use case (de-

tails of the use case omitted). The end user (i.e., a number analysts) provided us with a list of possible keywords, such as organizations of interest, key people, key topics, and were able to pick the relevant sequence of observations when we presented them with the results of the Data Transformation component. For RSS publications, around 3,000 news abstracts from 64 publishers, and for Twitter, around 73,000 tweets from around 32,000 users matched our keyword search criteria. The domain experts, different from the analysts, also provided us with many Mind Maps that have over 300 concepts in total, each focused on cascading of events for a particular main concept such as the oil price as shown in Figure 2. We also automatically generated a set of questions to ask both the domain experts as well as the analysts in order to understand the weights of the edges. This resulted in around 100 questions. The resulting planning problem that aggregates the knowledge of all Mind Maps (i.e., the grounding of the actions based on the edges on the Mind Maps) has around 350 predicates. The planner is able to find 100 plans with a time horizon of 5, within a minute, which meets our objective of this use case.

We evaluate our approach to two recent plan-recognition-as-planning approaches: (1) Ramírez and Geffner's 2010 approach configured to use the LM-Cut planner (Pommerening and Helmert 2012) and (2) Sohrabi, Riabov, and Udrea's 2016 approach configured to use the TK* planner (Riabov, Sohrabi, and Udrea 2014) as well as LPG-d (Nguyen et al. 2012). The plan-recognition-as planning approaches require as input a set of goals. We argue that in many cases, the set of goals is not known a priori, even if known, it may be difficult to enumerate them. Furthermore, providing the set of goals ahead of time may impose a restriction on the search space. Instead, our approach requires a time horizon and a number of trajectories to compute as its input. The analysts can optionally provide a set of predefined known values and the system will cluster the trajectories around them. The system will also present clusters around other non-given values as shown in light orange at the bottom of Figure 5.

Table 1 shows a summary of our comparison on the energy domain. The objective of this experiment is to evaluate the performance of the proposed approach and also to see if the analysts can benefit by using the proposed approach by seeing that new goals are discoverable. Time is measured in seconds. In the proposed approach, the time measures both the planner performance as well as the clustering performance. The column "# of Goals" is the provided number of goals as the input. This is the same as the number of cluster representatives the analysts selected for the proposed approach. The column "# of clusters" is the total number of clusters presented to the user. For the previous approaches, this number is not defined. The column T is the time horizon while "# of Plans" is the total number of plans produced by each approach. Note, we defined the cluster representatives to best resembles the given set of goals using the predicates in the domain. The results show that as the number of goals increases, the performance of Ramírez and Geffner's 2010 approach slows down as it requires to call the planner twice per each goal. The performance of Sohrabi, Riabov, and Udrea's 2016 approach does not increase; however, this approach requires the provided goals as the input and does not project into the future. In our proposed approach, new clusters/goals that the analysts were not aware are created. The new clusters provide useful, valuable information to the analysts. Note, in the case of 1000 plans, the time to compute the clusters and visualization can add to the overall time in the proposed approach. The ideal case is to call the Planning Projector with a small set of goals (i.e., provided known cluster representatives), and let the projector create additional clusters based on the result.

Approach	Time (Sec)	# of Goals	# of Clusters	T	# of Plans
Ramírez	53.52	5	NA	NA	10
& Geffner,	92.66	10	NA	NA	20
LM-Cut	145.68	15	NA	NA	30
Sohrabi	0.24	5	NA	NA	1000
et al.,	0.26	10	NA	NA	1000
TK*	0.25	15	NA	NA	1000
Sohrabi	2.70	5	NA	NA	50
et al.,	1.80	10	NA	NA	50
LPG-d	2.06	15	NA	NA	50
	0.16	5	27	4	100
	0.35	5	36	6	100
	0.15	10	23	4	100
Proposed	0.37	10	24	6	100
Approach	0.15	15	28	4	100
	0.35	15	29	6	100
	6.31	10	43	4	1000
	7.31	10	61	4	1000

Table 1: Comparison of two plan-recognition-as-planning approaches with our proposed approach: Ramírez and Geffner's approach configured to use the LM-Cut planner, Sohrabi, Riabov, and Udrea's approach configured to use the TK* planner as well as LPG-d. Time is measured in seconds. "# of Goals" is the given number of goals, "# of clusters" is the total number of clusters produced, T is the time horizon, and "# of Plans" is the total number of plans produced by each approach. NA indicates not applicable.

Discussion and Summary

We present the Planning Projector system prototype that is able to infer the past observations and project the future states by computing a set of trajectories using AI planning. The system is comprised of four components: knowledge engineering, data transformation, trajectory computation, and user interface. The main contributions of this paper are: (1) characterization of the FSP problem where instead of a set of given goals, a future time horizon and number of trajectories is given; (2) computation of the trajectories via planning; (3) translation of the domain knowledge expressed in a tool, such as Mind Maps into the planning language; (4) transformation of the raw data into observations; (5) presentation of the set of plans in a compact and efficient way. While we are unable to discuss the use case, the energy domain closely resembles this use case. Furthermore, our approach is not domain specific, and can be applied in a variety of different use cases. In future, we plan to extend the formalism to better capture concurrent actions as well as the interactions between multiple agents acting within one domain.

There exist a body of work on the plan recognition prob-

lem with several different approaches (e.g., (Zhuo, Yang, and Kambhampati 2012)). However, most approaches assume that the observations are perfect, mainly because they do not take as input the raw data and that they do not have to analyze and transform the raw data into observations (Sukthankar et al. 2014). Also, most approaches assume plan libraries are given as the input, whereas, we use AI planning (e.g., (Goldman, Geib, and Miller 1999)). Furthermore, there is a body of work on learning the domain knowledge (e.g., (Yang, Wu, and Jiang 2007; Zhuo, Nguyen, and Kambhampati 2013)). Learning can be beneficial in domains in which plan traces are available.

Acknowledgements

We thank Fang Yuan and Finn McCoole at IBM for providing the domain expertise. We thank Nagui Halim for his guidance and support. We also thank our LAS collaborators. This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

References

Aljazzar, H., and Leue, S. 2011. K*: A heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence* 175(18):2129–2154.

Bauer, A.; Botea, A.; Grastien, A.; Haslum, P.; and Rintanen, J. 2011. Alarm processing with model-based diagnosis of discrete event systems. In *Proceedings of the 22nd International Workshop on Principles of Diagnosis (DX)*, 52–59.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 2–10.

Coman, A., and Muñoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)*, 946–951.

Cordier, M.-O., and Thiébaux, S. 1994. Event-based diagnosis of evolutive systems. In *Proceedings of the 5th International Workshop on Principles of Diagnosis (DX)*, 64–69.

Faste, H., and Lin, H. 2012. The untapped promise of digital mind maps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1017–1026.

Goldman, R. P.; Geib, C. W.; and Miller, C. A. 1999. A new model of plan recognition. In *Proceedings of the 15th Conference in Uncertainty in Artificial Intelligence (UAI)*, 245–254.

McDermott, D. V. 1998. PDDL — The Planning Domain Definition Language. Technical Report TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

McIlraith, S. 1994. Towards a theory of diagnosis, testing and repair. In *Proceedings of the 5th International Workshop on Principles of Diagnosis (DX)*, 185–192.

Muise, C. 2016. Planning.Domains. In *International Conference on Automated Planning and Scheduling – Demonstrations*.

Nguyen, T. A.; Do, M. B.; Gerevini, A.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence* 190:1–31.

Pommerening, F., and Helmert, M. 2012. Optimal planning for delete-free tasks with incremental LM-Cut. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1778–1783.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*.

Riabov, A. V.; Sohrabi, S.; Sow, D. M.; Turaga, D. S.; Udrea, O.; and Vu, L. H. 2015. Planning-based reasoning for automated large-scale data analysis. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*, 282–290.

Riabov, A. V.; Sohrabi, S.; and Udrea, O. 2014. New algorithms for the top-k planning problem. In *Proceedings of the Scheduling and Planning Applications woRKshop (SPARK) at the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 10–16.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning domain definition using GIPO. *Knowledge Engineering Review* 22(2):117–134.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)*, 261–267.

Sohrabi, S.; Riabov, A. V.; Udrea, O.; and Hassanzadeh, O. 2016. Finding diverse high-quality plans for hypothesis generation. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*. Also appears in the Proceedings of the 8th International Workshop on Modeling and Reasoning in Context (MRC).

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*.

Sohrabi, S.; Udrea, O.; and Riabov, A. V. 2013. Hypothesis exploration for malware detection using planning. In *Proceedings of the 27th National Conference on Artificial Intelligence (AAAI)*, 883–889.

Sukthankar, G.; Geib, C.; Bui, H. H.; Pynadath, D. V.; and Goldman, R. P. 2014. *Plan, Activity, and Intent Recognition*. Boston: Morgan Kaufmann.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE 2.0: An integrated tool for designing planning domains. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*.

Xu, R., and Wunsch, I. 2005. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks* 16(3):645–678.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence* 171(2-3):107–143.

Zhuo, H. H.; Nguyen, T.; and Kambhampati, S. 2013. Refining incomplete planning domain models through plan traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2451–2457.

Zhuo, H. H.; Yang, Q.; and Kambhampati, S. 2012. Action-model based multi-agent plan recognition. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 377–385.