Investigation Planning in Data Analysis

Ilbin Lee[1], **Shirin Sohrabi**[2], **Anton Riabov**[3], **Octavian Udrea**[2],
[1] University of Alberta School of Business
[2] IBM T.J. Watson Research Center
[3] Logitech International S.A.
ilbin@ualberta.ca, ssohrabi@us.ibm.com, ariabov@logitech.com, udrea@us.ibm.com

Abstract

In many applications of artificial intelligence, such as intensive care delivery or malware detection, the state of an entity of interest is only indirectly assessed via observations, which can indicate multiple states. An observation is typically obtained through analysis of data; however, it is usually impossible to obtain all observations due to limited resources to analyze the data and its fast accumulation. An agent's goal is to direct the entity of interest to a desired state with high probability with an uncertain starting point and imperfect observational capabilities. Thus, the agent's plan must optimally balance further analysis of data to reveal observations (reduce uncertainty) and actions that change the entity's state. In this paper we examine this problem, which we call the *investigation planning* in data analysis. We propose a novel hypothesis-based formulation of the problem. To this end, we use an AI planner that computes highly likely hypotheses from which we form a belief-state that estimates the current state and then choose a set of "promising" analyses. Due to the large space of belief-states, we apply a limited look-ahead online approach. We experimentally evaluate our approach over a large benchmark problem set.

1 Introduction

In many applications of artificial intelligence, the state of an entity or a system of interest is not directly assessed. Instead, we have available some indicators of one or more states associated with the entity – we call these indicators observations. In practice, an observation is obtained through data analysis of data, e.g., comparing medical record of a patient to a large database. It is usually impossible to analyze all available data because of limited resources and time, and fast accumulation of data. Hence, we must deal with partial observability of possibly unreliable sequence of observations and may need to revisit past data with more in-depth analyses to obtain additional observations. Observations that are already obtained by analyzing data are called *revealed* and the ones that could be obtained from the data but are yet to be revealed are called *hidden*. In this scenario, the agent's goal is to move the entity of interest to a desired state. To achieve this goal, the agent's plan must balance actions that reduce uncertainty with those that directly affect the state of the entity of interest.

Let us consider the following example for intensive care delivery. Suppose that we are given a simplified model of a patient consisting of states, transitions between states, and many-to-many correspondence between states and observables, shown in Fig. 1(a). This model can be specified by a domain expert (i.e., doctors) or could be learned given enough data (e.g., by process mining). The rounded rectangles are states, the callouts are observations associated with these states, and the arrows are possible transitions. For example, from *Highrisk* state, the patient may get to the *Lowrisk*, *Infection*, *Infarction*, or the *DCI* (*Delayed Cerebral Ischemia*) state. The observations are obtained based on the raw data captured by sensors as well as other measurements and computations provided by doctors and nurses. For example, given the patient's heart rate, blood pressure, and temperature, a SIRS score is computed, producing an integer from 0 to 4. Similarly, a result of CT Scan, or a lab test will indicate other possible observations about the patient.

For the model of Fig. 1(a), consider the following sequence of revealed observations: *HH2, HRVL*. This indicates the patient has a Hunt and Hess score of 2 (a score classifying the severity of subarachnoid hemorrhage), followed by low heart rate variability. We call a sequence of revealed observations a *trace*. Based on the trace, the agent has to re-construct the current state of the patient and determine whether to analyze past data to reveal more observations. For example, the agent may check if *HH3* is hidden between *HH2* and *HRVL*, which may indicate the patient went through the *Highrisk* state. Once the agent has a good idea about the state of the patient, it may then decide how to treat the patient – with the goal of bringing the patient to a *Discharged* state. That is, it must decide whether to intervene and change the current state of the entity, and how. There is a natural trade-off between analyzing more data to reveal more observations (called a *test action*) in order to better understand the entity's current state and taking actions to change the entity's state to a desired one (called a *preventive action*). We call this problem *investigation planning*.

While the example in Fig. 1(a) is simplified, most real-world problems face the issue of deciding on a course of action with imperfect information. Finite resources typically

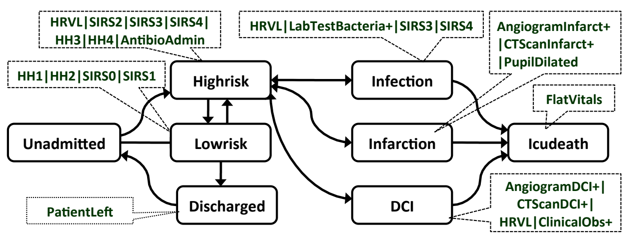


Figure 1 (a): Intensive care model

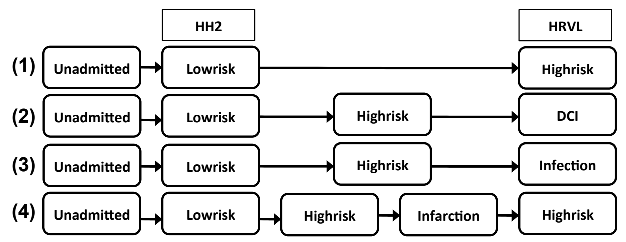


Figure 1 (b): Hypotheses for trace *HH2, HRVL*

prevent us from executing all possible analyses on all data in real time. For example, in intensive care delivery, we may execute simple electrocardiogram (ECG) analyses such as computing the heart rate in real time, but we typically run complex analyses such as computing the heart rate variability only during the course of an investigation into a specific suspected condition. Observations coming from non-computational sources have similar characteristics – for example, blood tests provide observations that are generally useful, but specialized (retrospective) tests may be required to look for markers of a suspected condition. The requirement for revealing missed observations retrospectively before deciding on a course of action is even more evident in domains such as malware detection, where simple analyses such as malware signature detection can be performed in real time for all network traffic, but complex analyses such as deep content inspection are performed retrospectively only on a small subset of traffic. However, existing approaches to partial observability cannot handle this requirement. In particular, a newly obtained observation in Partially Observable Markov Decision Processes (POMDPs) always indicates the current state, but in our problem, it may indicate a past state and this makes it impossible to update the belief-state as POMDPs do (for other reasons, see the Related Work). As a further source of complexity in certain domains such as cybersecurity, we are often faced with adversarial behavior that adapts to our real-time monitoring, in which case many possible analyses are relegated to post-incident forensic investigations.

In this paper, we consider generating a number of sequences of states for a given trace, which we call *hypotheses*, in order to re-construct the current state and determine which observations to look for at which position. For the above example, some possible hypotheses are given in Fig. 1(b). For example, the hypothesis (4) suggests that *HH2* was observed at state *Lowrisk* and *HRVL* at *Highrisk* (in this case, for example, we say *HRVL* is *explained* by *Highrisk*). Given this hypothesis, for instance, the agent may consider checking if *PupilDilated* is hidden in data between *HH2* and *HRVL* to see whether the patient went through *Infarction*.

We use an AI planner to compute a finite number of hypotheses that "explain the trace well". Using these, we define *belief-state* which approximates the entity's current state. The goal is to reach a belief-state where we "believe" the entity is in a desired state, while minimizing the total cost of tests and preventive actions taken. Since tests have different possible outcomes, we formulate this problem as finding an optimal conditional plan that reaches a desired belief-state with the minimum cost. Note, the set of belief-states that can be reached from an initial belief-state by simulating the effects of test and preventive actions is exponential in the number of actions that can be executed. To overcome this problem, we propose a limited look-ahead online approach.

The main contributions of this paper are: (1) proposing a novel hypothesis-based formulation of the investigation planning problem; (2) proposing a new scheme to generate hypotheses and their probabilities; (3) implementing a limited look-ahead online approach; (4) evaluating the approach by empirically analyzing how different ways to generate hypotheses and algorithm parameters impact its performance. We evaluated the approach over a large benchmark problem set and our experiments show that the plans generated by our system lead to a *good* state in more than 90% of cases while minimizing cost, and plans for complex problems can be constructed in seconds.

2 Problem Formulation

Let S be the set of states of an entity of interest. We assume a classification of S into two sets, *good* and *bad* states denoted as S_g and S_b, respectively. State is only indirectly observed via observations. Let O be the set of all observables. Transition law $\tau : S \times S \rightarrow [0,1]$ gives the probability distribution that governs the state transition and observation law $\omega : S \times O \rightarrow [0,1]$ governs which observation is observed in a state. When the entity enters a state s, an observation o is generated (but may be hidden) with probability $\omega(s,o)$ and the entity makes a transition to a state t with probability $\tau(s,t)$.

We assume that the entity already made a number of transitions, and only some of the observations are revealed. We assume that the last observation in data is revealed and that the agent does not know how many observations are hidden. Let $\phi_0 = [o_1, ..., o_n]$, where $o_i \in O$, denote the sequence of initially revealed observations (also called initial trace).

For a trace ϕ, a test action is written as a pair (i,o) where i is a natural number less than the length of ϕ and $o \in O$. This means past data is analyzed to determine if the observation o is hidden between the ith and $(i+1)$st observations in ϕ. If o is hidden in the data, outcome of the test is *true* and o is added to ϕ between the two observations. Otherwise, the outcome is *false*, which results in no change in the trace but we know o is not hidden between the two observations. The set of all test actions is determined by the trace ϕ and O. A preventive action is a function from S to S, which means we assume that their effects are deterministic and known. We make this assumption to simplify the modeling aspects of the problem definition; since our algorithms already take into account non-deterministic effects of test actions, an extension to non-deterministic preventive actions is straightforward. In our example, there can be a preventive action prescribing an antibiotic, which moves a patient from *Infection* to *Highrisk*. However, preventive actions may have side effects as they are conditioned on the state of the entity. Let A^p denote the set of preventive actions.

Each action has a positive cost. In applications, a clinical action has an economic cost, and an analytical action can have a computational cost. Let c denote the cost function that maps an action to its cost. We assume the entity does not make a transition while the agent plans and takes actions, thus, further transitions are made only by the agent taking preventive actions. The objective of our problem is to hypothesize the state of the entity, refine the belief by performing test actions, and create a plan of recovery (preventive actions). Because the effect of a preventive action is conditioned on the current state of the entity, it is important for the agent to get a better understanding about the current state by using test actions to reveal observations in data. Since the agent wants to minimize the total cost of actions taken, there is a natural tradeoff between test and preventive actions.

Next, we formally define a few terms.

Definition 1 (Belief-state) *Given a set of states S, a belief-state is defined as a probability distribution over S.*

We compute belief-states by generating *hypotheses* similar to those in Fig. 1(b). Note that in Fig. 1(b), the state sequences are aligned with the observations, representing which observation is generated from which state. For a given trace, we call a sequence of states along with a correspondence between states in the sequence and observations in the trace, a *hypothesis*. For the purpose of this paper, we assume the existence of a hypothesis generator.

Definition 2 (Hypothesis Generator) *Given S, O, τ, and ω, hypothesis generator $\mathcal{H}$ receives a trace ϕ, and a number K as an input and outputs at most K highly likely hypotheses along with their probabilities, where S is the set of states, O is the set of observables, and τ and ω are the transition and observation laws.*

Going back to our example, consider the following probabilities for the four hypotheses: 0.4, 0.3, 0.2, 0.1. Then, the belief-state has probabilities 0.5, 0.3, 0.2 on *Highrisk*, *DCI*, *Infection*, respectively. Now, if a test action is taken and its outcome is true, then the trace is updated to insert the new observation. Hypotheses and their probabilities are then recomputed for the new trace. In the case of false outcome, the belief-state remains unchanged. If a preventive action is taken, the belief-state is updated depending on its effect. For example, consider a preventive action that prescribes an antibiotic which moves a patient from *Infection* to *Highrisk* but causes no change at other states. If the action is taken at the belief-state {*Highrisk*:0.5, *DCI*:0.3, *Infection*:0.2}, then the updated belief-state is {*Highrisk*:0.7, *DCI*:0.3}.

Definition 3 (Goal Belief-State) *A goal belief-state is defined as a belief-state whose probability of being in bad states is less than a pre-specified threshold δ.*

Definition 4 (Investigation Planning) *An investigation planning problem is defined as $P = (S_g, S_b, O, \tau, \omega, A^p, c, \phi_0, \mathcal{H})$, where S_g and S_b are the set of good and bad states, O is the set of observables, τ and ω are the transition and observation laws, A^p is the set of preventive actions, c is the cost function, ϕ_0 is the initial trace, and $\mathcal{H}$ is the hypothesis generator. The solution to P is a conditional plan that reaches a goal belief-state with minimum cost.*

Note, the set of test actions is not included in the above definition as it is constructed from O and the trace, and is not known a priori, a major reason for why defining a classical planning problem is difficult for this problem.

3 Investigation Planning

Our approach to investigation planning for data analysis is modeled as a three component system (Fig. 2(a)) that includes: (1) Data analysis applications running on middleware platforms transform input raw data into observation traces. For instance, to obtain a *HRVL* (low heart rate variability) observation, a signal processing application has to analyze the ECG signal from the patient, eliminate abnormal beats, and determine the distance between peaks. (2) The hypothesis generator produces a set of at most K highly likely hypotheses and their probabilities. (3) The investigation planner produces test and preventive actions that change (submit, cancel or modify) the set of data analysis applications. In this section, we describe the hypothesis generator and our online algorithm for investigation planning.

3.1 Hypothesis Generation

We generate hypotheses by extending the work in [Sohrabi *et al.*, 2013]. Their approach generates a number of "high quality" hypotheses given a similar state transition and observation model (without probabilities) by formulating the hypothesis generation problem as a planning problem. To evaluate quality of a hypothesis, their approach assigns a heuristically determined cost (not to be confused with the cost of actions, in this subsection cost is only to evaluate hypotheses) to each transition and each observation explained by a state. On the other hand, in order to produce K highly likely hypotheses and their probabilities, we assign costs based on the transition and observation laws τ and ω. For a hypothesis h for a trace ϕ, our costs are defined as follows: $-log(\tau(s, s'))$ for each state transition $s \rightarrow s'$ in h; $-log(\omega(s, o))$ for each observation o in ϕ explained by a state s in h; a fixed cost $-log(P_{noisy})$ for each unexplained observation o in ϕ, where P_{noisy} is the estimated probability of having a noisy observation (in practice, observations resulting from analyses may be unreliable due to missing or noisy data). With this cost assignment, the sum of all costs for state transitions in the hypothesis h is $-log(P(h))$, and the sum of all costs for explained and unexplained observations in the trace is $-log(P(\phi|h))$. Therefore, the total cost of the hypothesis is $c(h) = -log(P(h)) - log(P(\phi|h)) = -log(P(h) \times P(\phi|h))$. Therefore, according to Bayes' formula, $P(h|\phi) \propto e^{-c(h)}$. Since we limit the set of hypotheses to K highly likely hypotheses, we re-normalize to obtain $\forall i \in [1, K], P(h_i|\phi) \approx e^{-c(h_i)} / (\sum_{i=1}^{K} e^{-c(h_i)})$.

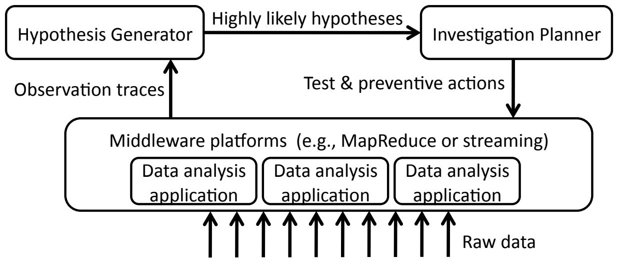

Hypothesis Generator
Highly likely hypotheses
Investigation Planner
Observation traces
Test & preventive actions
Middleware platforms (e.g., MapReduce or streaming)
Data analysis application
Data analysis application
Data analysis application
Raw data

Figure 2 (a): System architecture

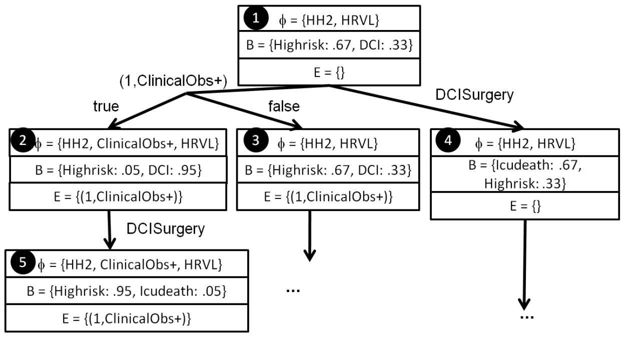

1 ϕ = {HH2, HRVL}
B = {Highrisk: .67, DCI: .33}
E = {}
(1,ClinicalObs+)
true
false
DCISurgery
2 ϕ = {HH2, ClinicalObs+, HRVL}
B = {Highrisk: .05, DCI: .95}
E = {(1,ClinicalObs+)}
3 ϕ = {HH2, HRVL}
B = {Highrisk: .67, DCI: .33}
E = {(1,ClinicalObs+)}
4 ϕ = {HH2, HRVL}
B = {Icudeath: .67, Highrisk: .33}
E = {}
DCISurgery
5 ϕ = {HH2, ClinicalObs+, HRVL}
B = {Highrisk: .95, Icudeath: .05}
E = {(1,ClinicalObs+)}
...
...

Figure 2 (b): Example scenario tree

3.2 Online Planning Algorithm

Our algorithm constructs a *scenario tree* starting from the belief-state computed over the initial trace ϕ_0. Each node in the scenario tree contains three elements: (i) the trace ϕ in that node; (ii) the belief-state $\mathcal{B}$; and (iii) the list E of test actions that have been previously considered on the path to the node. Essentially, the scenario tree simulates the possible effects of applying test and preventive actions, and the information is used to determine the best action in the root node.

Algorithm 1 Construction of the scenario tree

```
Input: Root node $R = (\phi_0, \mathcal{B}_0, \{\})$, depth limit $D$, threshold $\delta$,
    hypothesis count $K$
Output: Scenario tree with the given root
 1: add all leaf nodes in tree rooted at $R$ to queue $\mathcal{F}$
 2: while $\mathcal{F} \neq \emptyset$ do
 3:     remove first node $n = (\phi, \mathcal{B}, E)$ from $\mathcal{F}$
 4:     let $\mathcal{A}_n$ be the set of applicable actions in $n$ (possibly heuris-
        tically determined)
 5:     for all $a \in \mathcal{A}_n$ do
 6:         if $a$ is preventive action then
 7:             $\mathcal{B}_1 \leftarrow$ apply $a$ to belief state $\mathcal{B}$
 8:             create node $(\phi, \mathcal{B}_1, E)$ as a child of $n$ and add to $\mathcal{F}$
 9:         else
10:             $a = (i, o)$ {is a testing action}
11:             $\phi' \leftarrow$ insert $o$ at position $i$ in $\phi$
12:             $\mathcal{B}_2 \leftarrow$ apply hypothesis generator $\mathcal{H}$ to $\phi'$ for $K$ hy-
                potheses
13:             create node $(\phi', \mathcal{B}_2, E \cup \{a\})$ as child of $n$ and add to
                $\mathcal{F}$ {true outcome node}
14:             create node $(\phi, \mathcal{B}, E \cup \{a\})$ as child of $n$ and add to $\mathcal{F}$
                {false outcome node}
15:         end if
16:     end for
17:     remove from $\mathcal{F}$ all nodes of depth $D$
18:     remove from $\mathcal{F}$ all nodes with probability of being in a bad
        state less than $\delta$
19: end while
20: return scenario tree rooted at $R$
```

The scenario tree is constructed as follows (outlined in Algorithm 1). We start by constructing the root $(\phi_0, \mathcal{B}_0, \{\})$, where $\mathcal{B}_0$ is the belief-state obtained by applying the hypothesis generator $\mathcal{H}$ to ϕ_0, and add it to the frontier F. We continue by iteratively extracting nodes from the frontier F whose belief-state is **not** a goal belief-state and creating its child nodes. Given a frontier node $n = (\phi, \mathcal{B}, E)$, we first select a set of applicable actions. In general, this set is comprised of *all* preventive and test actions not been previously executed. Practically, since this is the branching factor of the tree, it is desirable to heuristically restrict this set to actions that are highly likely to lead to a goal belief-state.

We continue by creating child nodes of n for each action. For each preventive action a_p, we create a child node $n' = (\phi, \mathcal{B}_1, E)$, where the belief-state $\mathcal{B}_1$ is obtained by applying a_p to the belief-state $\mathcal{B}$. For each test action $a_t = (i, o)$, we create two child nodes for its true and false outcomes, respectively: the node $n^{a_t}_{true} = (\phi', \mathcal{B}_2, E \cup \{a_t\})$ where ϕ' is the trace ϕ with the observation o inserted between positions i and $i + 1$ and $\mathcal{B}_2$ is obtained by applying $\mathcal{H}$ to ϕ'; the node $n^{a_t}_{false} = (\phi, \mathcal{B}, E \cup \{a_t\})$ where the trace and belief-state are the same as the parent n, but we record that a_t has been tried to avoid running it again.

Fig. 2(b) illustrates the trade-off between test and preventive actions. Assume that the preventive action *DCISurgery* transitions the patient from *DCI* to *Highrisk*, from *Highrisk* to *Icudeath*, and keeps every other state the same; also assume that *Highrisk* is a good state and every other state in the example is bad. In addition, assume that the observation *ClinicalObs+* can be obtained by a retrospective analysis of the data from patient monitors for a given time interval. The root (node (1)) contains the initial trace {*HH2, HRVL*} and an arbitrary belief-state is given. The test action *(1, ClinicalObs+)* may have two outcomes (nodes (2) and (3)). The *true* outcome inserts *ClinicalObs+* in the trace and correspondingly makes *DCI* the most likely state of the patient. From this state, applying *DCISurgery* leads us to a goal belief-state (node (5)) if we assume a threshold $\delta = 0.1$. On the other hand, applying *DCISurgery* without disambiguating the state of the patient shows a 0.67 probability of ending up in *Icudeath* (node (4)).

To obtain the best action at the root, we start by computing a *cost* for each leaf node that estimates how far we are from a goal belief-state. A simple way is to assign a value proportional to the total probability of being in a bad state according to the belief-state of that node. Then we perform backward induction, assuming for test actions that *true* and *false* outcomes are equally likely. In the example in Fig. 2(b), let us assume that the cost of the *(1, ClinicalObs+)* is 0.1, the cost of *DCISurgery* is 1. We can compute the cost of node (5) as 0.05 – the probability of being in a bad state. The running cost of the *DCISurgery* action in node (2) is $1 + 0.05 = 1.05$.

The cost of node (3) is 0.33 and that of node (4) is 0.67. As a result, the running cost of *DCISurgery* at the root is 1.67 and the running cost of *(1, ClinicalObs+)* at the root is $0.1 + 0.5 \times 0.33 + 0.5 \times 1.05 = 0.7675$. As a result, the best action at the root is the test action. Note that the execution of *DCISurgery* as the second action is conditioned on the test action *(1, ClinicalObs+)* having a *true* outcome. Also, note that the subtree of the scenario tree that selects only the *best* action in every node form a conditional investigation plan. In the example, the conditional plan is the subtree formed of nodes (1), (2), (3), (5).

Since we execute the hypothesis generator $\mathcal{H}$ in almost all nodes (except *false* test action outcome nodes), in practice we must limit the tree size and we do this by limiting both the depth and the branching factor of the tree. To limit the branching factor, we heuristically select a subset of actions likely to lead to a goal belief-state in each node. Given a hypothesis h, an example heuristic selects test actions associated with any state that does not explain any observation in ϕ. For example, in hypothesis (2) in Fig. 1(b), it selects test actions for every observation associated with *Highrisk* between positions 1 and 2 in the trace (e.g., *(1, SIRS2)*). If any such test is successful, the probability of hypothesis (2) will increase. We compared this heuristic with others and experimentally determined that combining this heuristic with limiting the number of testing and preventive actions considered in each node yields the best trade-offs between performance criteria discussed in the next section.

Algorithm 2 Online investigation planning algorithm

```
Input: Initial trace $\phi_0$, depth limit $D$, threshold $\delta$, hypothesis count $K$
 1: $\mathcal{B}_0 \leftarrow$ apply hypothesis generator $\mathcal{H}$ to $\phi_0$ for $K$ hypotheses
 2: $R \leftarrow (\phi_0, \mathcal{B}_0, \{\})$
 3: construct and solve scenario tree rooted at $R$ with depth $D$, threshold $\delta$, $K$ hypotheses
 4: execute the best action at $R$
 5: loop
 6:     wait until trace $\phi_0$ is updated to new trace $\phi$
 7:     if there exists child $n$ of $R$ with trace $\phi$ then
 8:         $R \leftarrow n$
 9:         if probability of bad state in $R$ less than $\delta$ then
10:             exit {completed}
11:         end if
12:         construct and solve scenario tree rooted at $R$ with $D$, $\delta$, $K$
13:         execute the best action at $R$
14:     else
15:         restart online planning algorithm with $\phi$, $D$, $\delta$, $K$
16:     end if
17: end loop
```

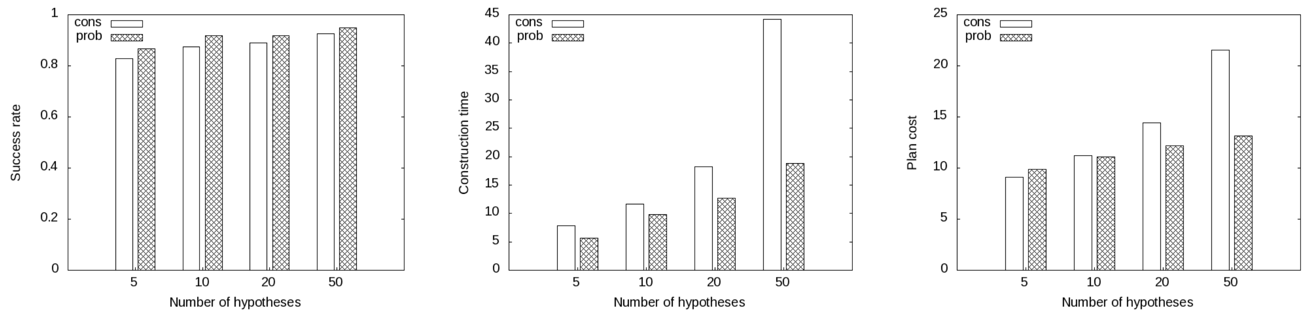
Limiting the depth of the tree is essentially a limited look-ahead approach, since not all leaf nodes at a fixed depth will be a goal belief-state. Thus, our algorithm is an online planning algorithm (Algorithm 2) that repeats these steps until reaching a goal belief-state: (1) construct the scenario tree to a fixed depth (2) execute the best action in the root node (3) based on the outcome determine the new root (should be a child of the current root) (4) expand the new tree an additional level. Note that depending on how many ambiguous (associated with multiple states) observations there are, it may not be possible to construct a complete scenario tree since we do not know how many observations are hidden.

4 Experimental Evaluation

We had three main objectives: (i) show that our algorithm takes the entity to a good state with high probability and has good performance, (ii) examine how different ways to compute belief-state affect the performance, and (iii) evaluate the impact of parameters.

We generated a large benchmark set of problems with varying complexity and size, using a similar technique as described in [Sohrabi *et al.*, 2013]. We started with two domain models provided by domain experts for intensive care delivery and malware detection. The intensive care delivery model was highly connected and with a lot of ambiguous observations. In the malware detection model we could observe distinct infection and exploitation stages, each consisting of multiple states with distinct observations. We then generated a number of finite state machines (FSMs) with similar connectivity characteristics (as well as low connectivity) of varied sizes. For each FSM, we generated a set of observables and many-to-many correspondence between observables and states like the example in the Introduction; only 40% of the states are chosen to be *good* and the rest are *bad*. This percentage is based on the real-world models, for which domain experts model the bad states in detail, and a few good states that either lead into the bad states or exhibit similar sets of observables with bad states and thus have to be distinguished from the bad states (e.g., in malware detection a machine crawling the Web has similar observables to an infected machine). We assigned uniform probabilities to possible state transitions. For each state, we distributed probability 0.975 uniformly to the associated observables and the remaining probability 0.025 was uniformly distributed to the other observables, in order to model noisy observations in applications. We generated one preventive action for each bad state and randomly added side effects. We also randomly chose costs of actions and cost of a test action only depends on the observation it tests. For each FSM, we generated ground truths by "walking" through the state transition model according to the transition law τ. For each state, we generated observations according to the observation law ω. For each FSM, we created 10 traces of length 5, 10, and 20, and in each trace we randomly revealed 60% of the observations on start, a percentage based on the number of observations typically missed in our real-world scenarios. In the end, we had more than a thousand distinct problems.

Since there is no existing algorithm applicable to the investigation planning and in order to evaluate our method to compute a belief-state based on probabilities of hypotheses, we compared our approach to a version that does not use probabilities. Specifically, this version, which we call the *conservative* approach, uses the hypothesis generator described in [Sohrabi *et al.*, 2013]; the belief-state is defined as a subset of S, consisting of the last states of generated hypotheses. We compare the *conservative* approach to our *probabilistic* approach with the threshold $\delta = 0.1$. The first evaluation



(a) Success rate (b) Construction time (c) Plan cost

Figure 3: Evaluation of the two approaches with varying number of hypotheses K

criterion is the success rate, i.e., the percentage of cases in which the entity reached a good state through the investigation plan. We also measured the construction time of the scenario tree and the total cost of all test and preventive actions executed (the plan cost). We used a depth limit of 3 for both approaches (Fig. 3). All experiments discussed were performed on a set of identical 16-core Intel(R) Xeon(R) E7-8837 CPUs 2.67GHz with 512GB RAM.

Fig. 3 shows the three criteria for the two approaches with varying number of hypotheses K, because we found that among the parameters K has the greatest influence. In the plots, *cons* and *prob* stand for the conservative and the probabilistic approach, respectively. In Fig. 3(a), the success rate of the two approaches are around 90% for all Ks considered, indicating that when a goal belief-state is reached, the entity reaches a good state with high probability. Also, the better success rates of the *prob* approach imply that the suggested method to compute belief-states better estimates the state of the entity. In Fig. 3(b), the average construction time of the *prob* approach is approximately 10 seconds for $K = 10$, implying our algorithm finds a plan to get the entity to a good state in a short time. Fig. 3 also shows that all criteria increase with K for both approaches, because larger values of K mean more possibilities in the belief-state, hence a better expected success rate, but also more actions to either narrow the possibilities down (by test actions) or treat them (by preventive actions). Notice that construction time and plan cost of the *cons* approach increase much faster as K increases. The difference in construction time can be explained by the fact that in the probabilistic approach we can set a threshold δ, whereas in the conservative approach the belief-state does not have probabilities, therefore we have to avoid all bad states.

We also performed a set of experiments where we varied the depth limit of the scenario tree (the look-ahead) from 2 to 7. We found that the average success rate for $K = 10$ grows from 87% to 91% from depth 2 to depth 3, but only to 92% for depth 4. Construction time is on average 4, 10, and 45 seconds for depths 2, 3, and 4, respectively. We did not see a significant success rate improvement for depths larger than 4, although the construction time was greatly increased.

5 Related Work

Planning under partial observability is commonly addressed by probabilistic approaches through Hidden Markov Models (HMM) and Partially Observable Markov Decision Processes (POMDP) (e.g.,[Hauskrecht and Fraser, 2000]) or non-probabilistic approaches through contingent planning (e.g., [Maliah *et al.*, 2014; Muise *et al.*, 2014]). However, several properties of the investigation planning problem present challenges. Observations linked to past states can arrive late and out of order, and any number of state transitions or observations can be hidden between any two revealed observations. The belief state following an execution of a test action is not determined by computation applied to a previous belief-state, and instead requires a call to an external hypothesis generator. Test actions are sensing actions, but describing the set of test actions is challenging because it depends on the current trace and the set of observations. Under these considerations, applying existing methods becomes non-trivial.

From another perspective, the problem can be thought similar in spirit to the problem of active diagnosis (e.g., [Sampath *et al.*, 1998; Haar *et al.*, 2013]) as we need to decide how to refine the set of hypotheses by running tests. However, the work in active diagnosis is focused on the context of fault diagnosis and the problem of diagnosability. Moreover, preventive actions are not considered, and our definition of test actions is most closely related to *truth test* in [McIlraith, 1994] rather than test actions in dynamic observers [Cassez and Tripakis, 2008]. Active probing is also relevant [Rish *et al.*, 2005], but we use heuristics to restrict the set of tests rather than inferences in Baysian networks. The problem also has similarities to plan recognition, but there are important differences in definitions and formulations (our observations do not uniquely identify state, we do not require plan libraries, and the set of possible goals is not given as input).

Our hypotheses-based approach is similar to diagnosis (e.g., [Sampath *et al.*, 1995; Cordier and Thiébaux, 1994]) and plan recognition (e.g., [Sukthankar *et al.*, 2014]). Planning has been applied to these problems (e.g., [Ramírez and Geffner, 2009; Sohrabi *et al.*, 2010]). We assume hypotheses are generated following [Sohrabi *et al.*, 2013], but our framework allows alternatives that agree with our definitions. In particular, previous work on diagnosis has some similar features to the problem we discuss. [Sampath *et al.*, 1995] proposed a discrete-event system approach to the problem of failure diagnosis where the system's behavior was described as a finite state machine. [Cordier and Thiébaux, 1994] considered event-based diagnosis of dynamic systems with ambiguous observations.

In [Sohrabi *et al.*, 2010], it was suggested that planning could be used to deal with incomplete information and rich preferences, and also for efficient generation of diagnoses. [Sohrabi *et al.*, 2011] identified a correspondence between planning and generating preferred explanations for observed behavior with respect to a model of a dynamic system, and [Sohrabi *et al.*, 2013] used the correspondence to generate hypothesis for malware detection application. In this paper, we use the planning approach in [Sohrabi *et al.*, 2013] to generate diagnoses for a patient based on which we find optimal tests and preventive actions.

[McIlraith, 1994] introduced a situation calculus framework for a general diagnostic problem including testing and repair and [McIlraith and Scherl, 2000] also defined a situation calculus that includes detailed definitions for different kinds of testing actions. However, the applications motivating this work allow more specific definitions of testing and repair (which we call *prevention*) than the ones in the situation calculus. Those definitions help us formulate a planning problem that unifies diagnosis, testing, and prevention for the applications, to which we can apply existing planning methods and heuristics. [Schumann *et al.*, 2010] defined a notion of definitely discriminating test (DDT) and discussed how to find a DDT with the minimum cost using the techniques from knowledge compilation and satisfiability (SAT), but did not considered repairing.

6 Conclusion

In this paper we introduced the problem of investigation planning for data analysis inspired from real-world scenarios. We proposed a novel hypothesis-based formulation of this problem, proposed a new way to generate hypotheses and their probabilities, and provided an online limited look-ahead algorithm to cost-optimally achieve a desired state with high likelihood. Our approach addresses the problem of planning with hypotheses based on unreliable observations revealed by testing actions, not previously studied in related work. Our experiments over an extensive set of benchmark problems show that our system achieves goals with a success rate around 90% in seconds.

There are several directions to expand our approach to better support real-world scenarios. First, we would like to consider more expressive action definitions. For testing actions, duration is a very important practical consideration; often we do not have the luxury to execute testing actions for an extended period of time before we must decide on a preventive action. Second, we would like to complement our online algorithm with learning. In particular, in this paper we have considered true and false outcomes of a testing action equally likely; however, the true probability of a true or false outcome can be learned over time, which is particularly useful for actions that have low true outcome probability (e.g., medical tests for very rare conditions). Finally, we would like to expand the set of action types to include for instance actions that suggest changes to the model to better explain observed traces.

References

[Cassez and Tripakis, 2008] Franck Cassez and Stavros Tripakis. Fault diagnosis with static and dynamic observers. *Fundam. Inf.*, 88(4):497–540, 2008.

[Cordier and Thiébaux, 1994] Marie-Odile Cordier and Sylvie Thiébaux. Event-based diagnosis of evolutive systems. In *Proceedings of the 5th International Workshop on Principles of Diagnosis (DX)*, pages 64–69, 1994.

[Haar *et al.*, 2013] Stefan Haar, Serge Haddad, Tarek Melliti, and Stefan Schwoon. Optimal constructions for active diagnosis. In *Proceedings of The 33nd International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 527–539, 2013.

[Hauskrecht and Fraser, 2000] Milos Hauskrecht and Hamish S. F. Fraser. Planning treatment of ischemic heart disease with partially observable Markov decision processes. *Artificial Intelligence in Medicine*, 18(3):221–244, 2000.

[Maliah *et al.*, 2014] Shlomi Maliah, Ronen I. Brafman, Erez Karpas, and Guy Shani. Partially observable online contingent planning using landmark heuristics. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[McIlraith and Scherl, 2000] Sheila McIlraith and Richard B. Scherl. What sensing tells us: Towards a formal theory of testing for dynamical systems. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, pages 483–490, 2000.

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

[Muise *et al.*, 2014] Christian Muise, Vaishak Belle, and Sheila A. McIlraith. Computing contingent plans via fully observable non-deterministic planning. In *Proceedings of the 28th National Conference on Artificial Intelligence (AAAI)*, 2014.

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

[Rish *et al.*, 2005] I. Rish, M. Brodie, Sheng Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. *Trans. Neur. Netw.*, 16(5):1088–1109, 2005.

[Sampath *et al.*, 1995] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[Sampath *et al.*, 1998] M. Sampath, S. Lafortune, , and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43:908–929, 1998.

[Schumann *et al.*, 2010] Anika Schumann, Jinbo Huang, and Martin Sachenbacher. Computing cost-optimal definitely discriminating tests. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, pages 161–166, 2010.

[Sohrabi *et al.*, 2010] Shirin Sohrabi, Jorge Baier, and Sheila McIlraith. Diagnosis as planning revisited. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 26–36, 2010.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)*, pages 261–267, 2011.

[Sohrabi *et al.*, 2013] Shirin Sohrabi, Octavian Udrea, and Anton Riabov. Hypothesis exploration for malware detection using planning. In *Proceedings of the 27th National Conference on Artificial Intelligence (AAAI)*, pages 883–889, 2013.

[Sukthankar *et al.*, 2014] Gita Sukthankar, Robert P. Goldman, David V. Pynadath, and Hung Hai Bui. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann, 2014.