Probabilistic Plan Recognition for Hostile Agents

Christopher W. Geib and Robert P. Goldman

May 21st, 2001

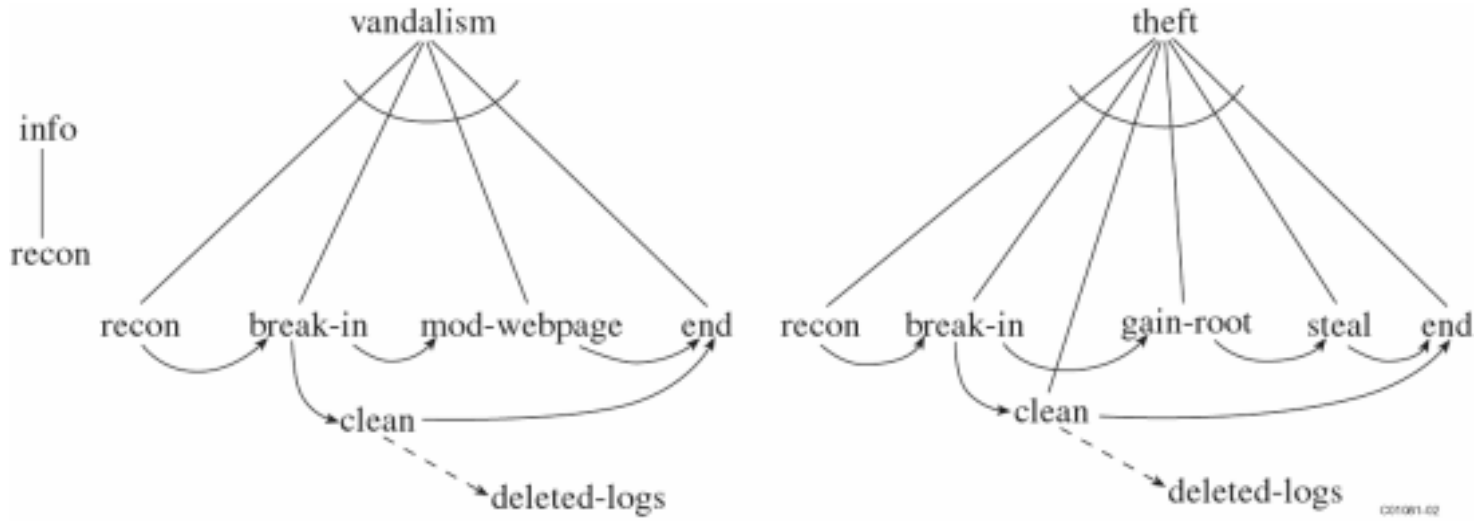
Background: definitions

- What is plan recognition: The process of recognizing the intentions or goals of an agent on the basis of observations of their actions.
  - AKA intent recognition, task trackingÖ

- Substantive work has been done in the area making different assumptions about the agentís attitude:
  - Agnostic: The agent doesnít care or is completely unaware of the fact that you are observing their actions.(monitoring systems)
  - Cooperative: The agent is aware and wants you to observe and understand the actions they are performing (collaborative or advisory systems)
  - Hostile: The agent is hostile to the inference of their intent.
    - (not studied in the AI literature)

Background: plan library

- And/Or tree representation of the set of possible plans that could be used by the agent to be watched.
- Distinguished nodes of the trees (most often the roots) represent the agentís possible goals,
- Partial ordering constraints
- Effects of select actions



Background: previous approaches

- Set covering (Kautz and Allen ë86)
- Probabilistic abduction (Charniak and Goldman ë93)
- Grammar formalisms (Sidner ë85, Vilain ë90, Pynadath and Wellman ë00)
- Reactive systems (Huber ë94, Rao ë94)
- Game theory, Mini-max search (Carmel and Markovitch ë96)

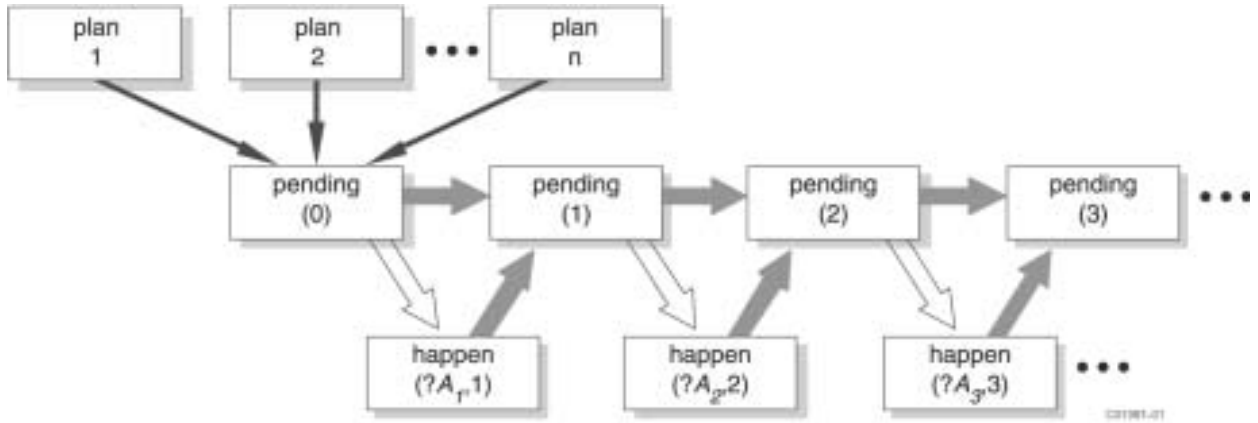
Problems

- Previous work suffers from a number of limitations that make them problematic for the domains we want to look at:
  - partial order plans (peanut butter or jelly first?)
  - effect of world state on goals (plant monitoring Vs. shutting down)
  - overloading of actions (you only need to scan the network once)
  - multiple goal Vs. single goal explanations
  - cumulative effect of not seeing something (The dog didnít bark)

- Hostile agents are generally not covered since most work assumes agnostic or cooperative agents.
  - Hidden actions/missing observations
  - Observation of state changes rather than actions

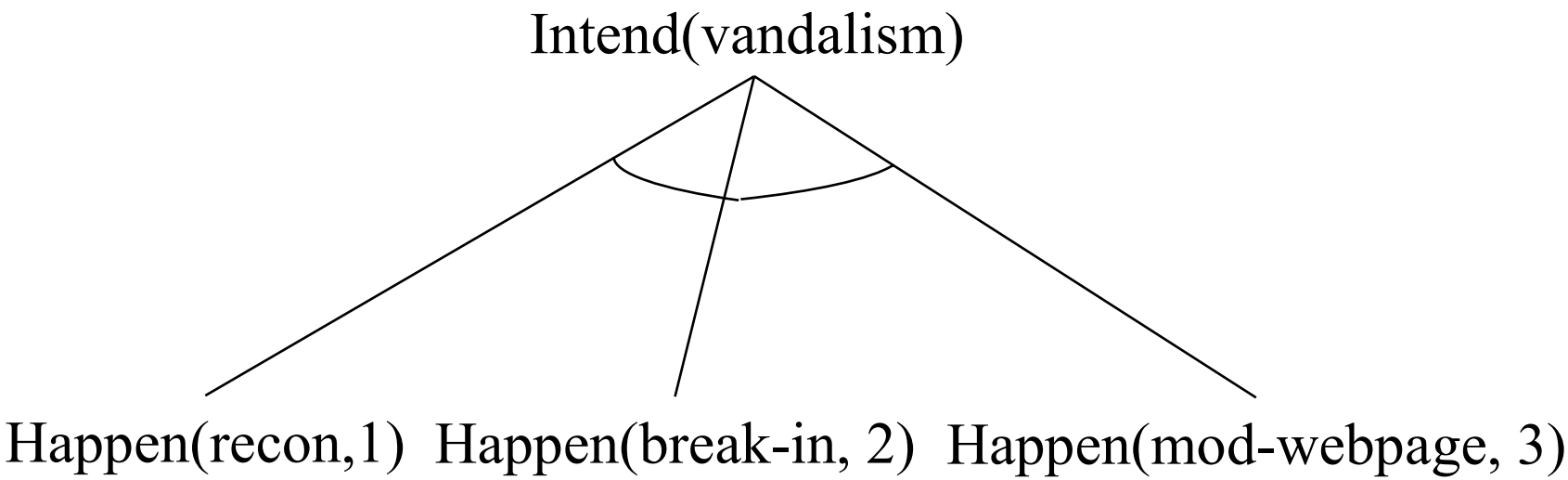
Solution: Pending sets

- Central Insight: the agent will only execute actions that are consistent with their goals and are enabled by the previous actions they have performed.
- We define a ***pending set*** as the set of actions that are currently enabled by the agentís hypothesized goals and the actions the agent has taken.
- We use the sequence of observed actions to generate the pending set and as a result the hypothesized goals of the agent.



Solution: explanations

- Define: an ***Explanation*** for a set of observations as the substructure of the plan library with pending sets and hypothesized goals sufficient to explain the current set of observed actions.
- Example: observed **recon, break-in, mod-webpage**



Solution: algorithm intuition

- Build an exclusive and exhaustive set of the explanations for the observed actions.
- Establish the probability of each of the explanations (P(Exp).)
- The conditional probability of the goal given the observations is just the sum of those explanations that have the goal divided by the probability of the observations. (P(Goal | Observations))

Solution: what probabilities do we need?

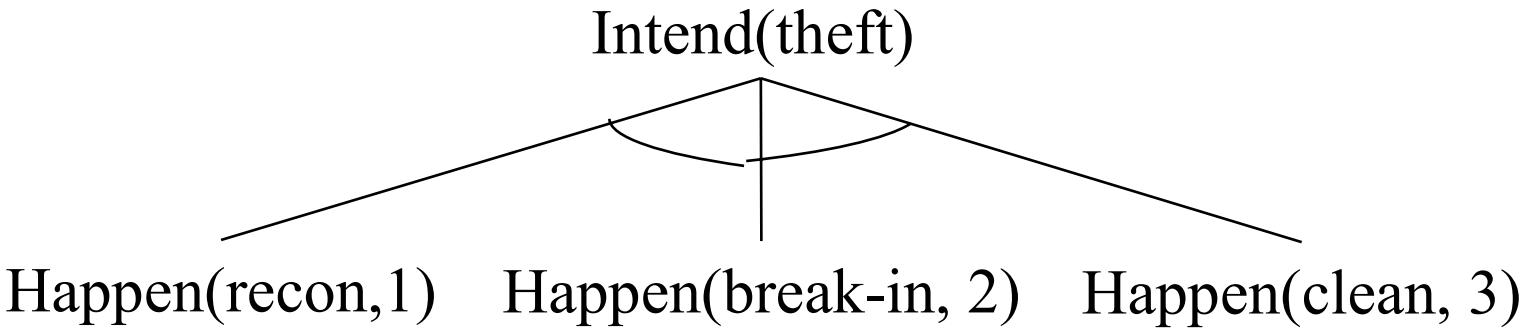
- Not many that you have to know and they are easy to get
  - Prior probability of a given intention or
  - The conditional probability of the intention given a state of the world

- The other probabilities that we need we can compute:
  - The probability of choosing a given action from the pending set.
  - The probability of choosing a specific method
    (when you have a choice of plans.)

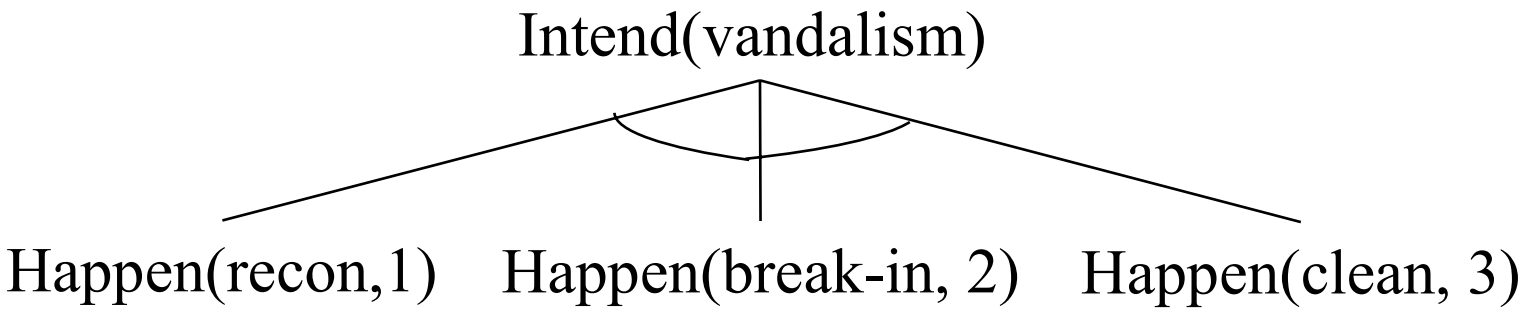
  - Our implementation assumes a uniform distribution for these
    (but this is not required)

Solution: example

- Observations: **recon, break-in, clean**
- Explanation1: cp(theft | recon, break-in, clean) = 0.5



- Explanation2: cp(vandalism | recon, break-in, clean) = 0.5



Solution: midpoint summary

- This approach will handle:
  - partial order plans
  - effect of world state on goals
  - overloading of actions
  - multiple goal Vs. single goal explanations
  - cumulative effect of not seeing something

- But we can also extend it to handle hostile agents

Solution: hostile agent problems

- This algorithm wonít work for hostile agents.
- Hostile agentís like to hide their actions so that you donít know what they are doing. (Think of you dog or cat or small child (or even big child))
- The given algorithm relies on the fact that we have a complete record of the actions performed by the agent to compute the pending sets.
- This assumption doesnít hold anymore.
- In the parlance of Markov Decision Processes (MDP) we have moved from a fully observable case to the partially observable case (POMDP).

Solution: general approach

- If we just had a complete set of observations then we wouldnít have a problem.
- Intuition:
  - Infer the unobserved actions from other information.
  - Use the set of inferred unobserved actions to insert actions into the observations sequence to produce the complete set.
  - Use the algorithm as before.

- How can we infer the execution of unobserved actions?
  - Observations of unenabled actions
  - Observations of unexplained state changes

Solution: intuition unenabled actions

- Definition: an ***unenabled action*** is an executed action that is not enabled by the agents previous actions. The observation of an unenabled action implies the execution of enabling actions that were performed unobserved.

- Example: all you observe is a **clean** action you can infer the execution of a **recon** and a **break-in**.

- The intuitive implementation of this would be to reason backwards once an unenabled action is found. This is NOT what we do. Instead we work forward inserting possible unobserved actions. We will return to discuss this later.

Solution: intuition state change

- Definition: an ***unexplained state change*** is an observed state change that does not have a causal explanation within the observed actions.

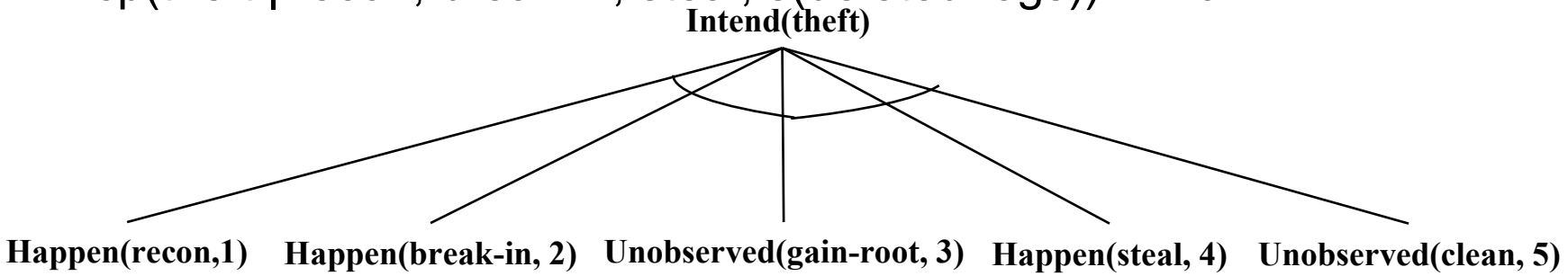
- Example: All you observe is **deleted-logs** you can infer the execution of a **clean** action.

- The addition of reports of state change to intent recognition is not common in the AI literature. This kind of information isnít important when you have a complete set of observations.
- Interestingly reports of state changes are often the most common in real domains. Hiding the effects of actions can be a great deal more difficult than hiding the actions. (think of nuclear weapons testing)

Solution: algorithm intuition

- Insert unobserved actions that are consistent with the
  - unobserved actions
  - unexplained state changes
  - number of unobserved actions is less than a chosen threshold
- Produces a complete execution trace
- User the algorithm as before to compute the pending sets and explanations on the basis of the new ìcompleteî set of observations.
- Example: observations: **recon, break-in, steal, s(deleted-logs)**

  cp(theft | recon, break-in, steal, s(deleted-logs)) = 1.0

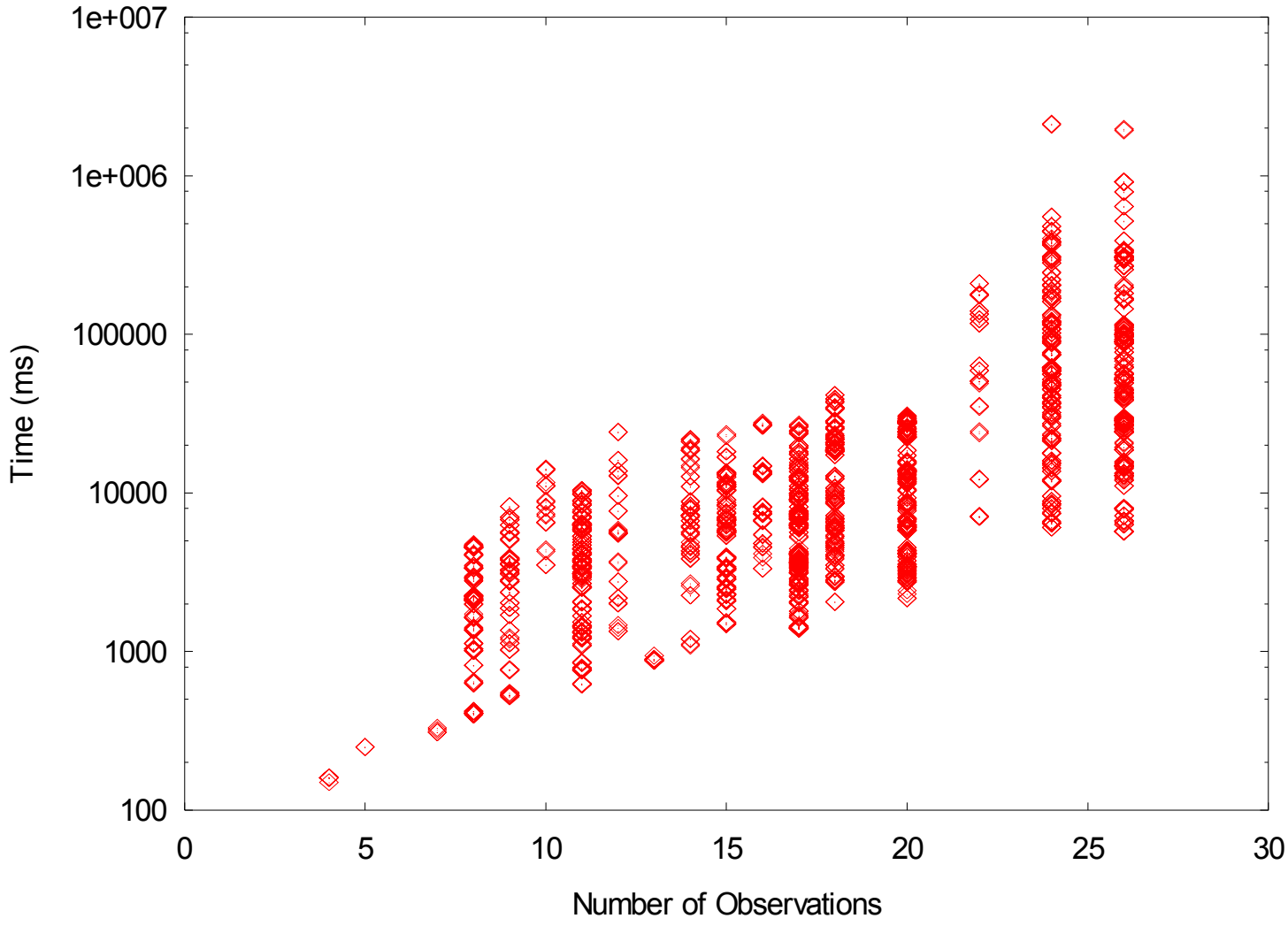


Implementation

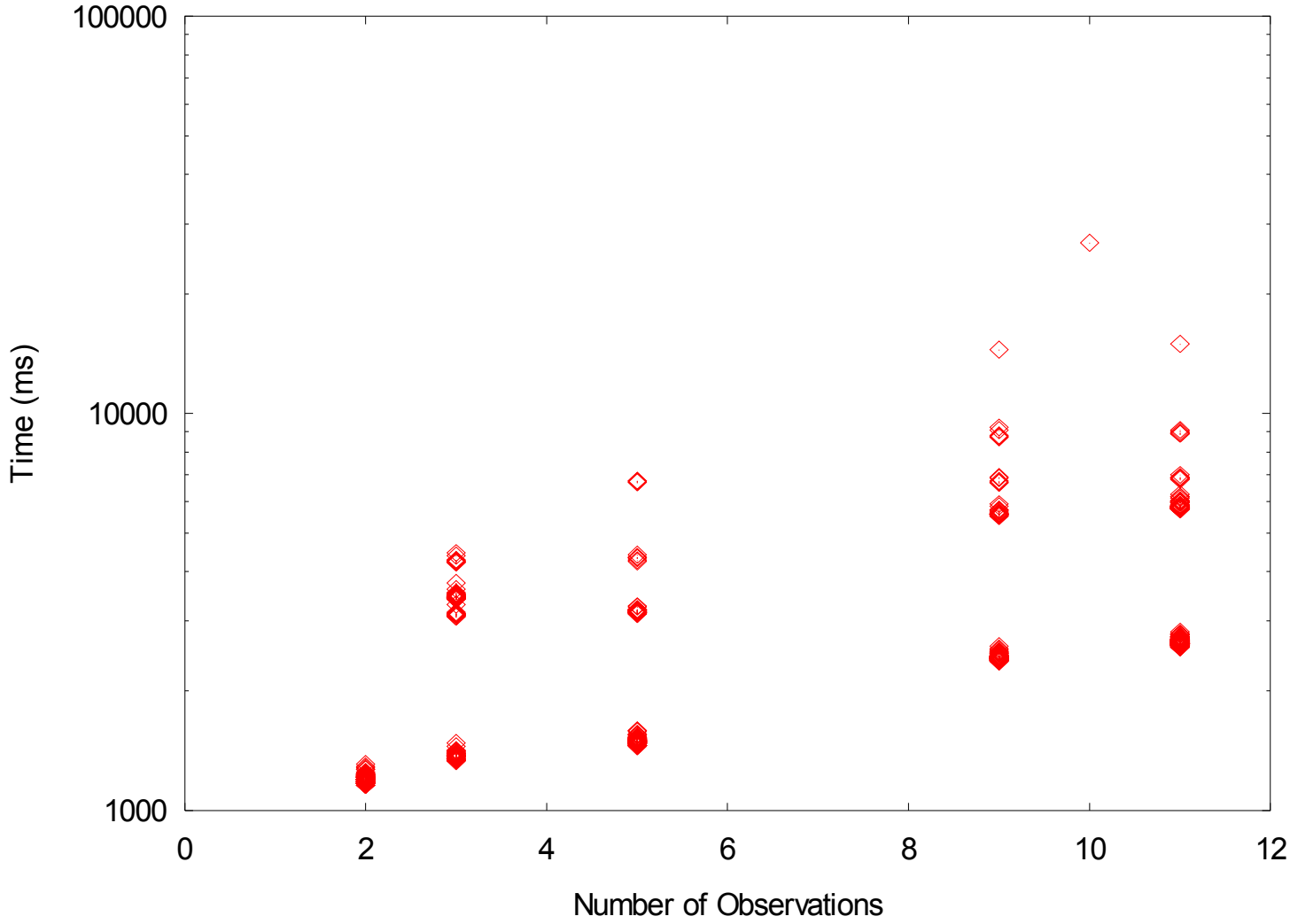
- Based on Pooleís Probabilistic Horn Clause Abduction
  - Push all probability out to axioms of a proof
  - Use first order logic to do your reasoning
  - Requires an exclusive and exhaustive set of explanations
  - Compute the probability of a single explanation by considering the ìaxiomsî of the explanation and their likelihood.

- Our current implementation
  - in prolog
  - not as efficient as it could be.

Implementation with no unobserveds



Implementation with unobserved actions



Implementation: analysis

- HEY WAIT A MINUTE! That looks exponential!

- Preliminary analysis: the good news -
  - The length of the observations is NOT the problem

- Preliminary analysis: the bad news -
  - The number of possible non-differentiated plans IS the problem

- Factors related to the number of non-differentiated plans
  - Unobserved action threshold (too low and too high both bad)
  - Branching factor of the plan library
  - Longest non differentiating prefix

Conclusions

- A new, abductive, probabilistic, theory of plan recognition that is based on taking the sequential nature of observations of actions and state changes seriously. It is able to handle a wide variety of phenomena that other work finds problematic and we have applied it to the problem of inferring the intent of hostile agents.

- Implementation that is ìefficient.î

- Preliminary analysis of the algorithm

End

Background: applications

- NIST Abnormal Situation Management ê99: study project for assistant systems for oil refineries. Watch plan operatorís actions and infer their goals. When possible suggest actions that might assist them in their task.
- DARPA CyberPanel project ê00-í01: Use reports from existing computer network intrusion detection systems as observations of

- Associate systems
- Educational or mentoring systems
- Intelligence analysis applications (business or governmental)

Solution: algorithm

- Inputs: a sequence of action observations, and a plan library
- Output: the conditional probability for each root intention
- Code:

```
For each observation
    Progress pending set
        . remove executed action
        . add newly enabled actions
    Add to possible hypotheses set any new explanations indicated by the actions
    Remove from possible hypothesis set any explanations inconsistent with the executed action.
    Loop
For each explanation compute the explanations probability
For each root compute the conditional probability
```

Solution: more probabilities needed.

- To do this we will need more probabilities.
- Again its just priors of the kind you would expect to get from simple observations of the functioning system.
- Prior probability of the action being performed unobserved.
- Threshold of how many unobserved actions you are willing to accept.

Future work

- More complete analysis of the features affecting the runtime
- Industrial grade implementation.
  - More efficient.
  - Incremental algorithm
  - precompilation of sequences and use of string matching algorithms.
  - application of Tree-Adjoining Grammars and O(n^6) parsing methods.
- Misdirection
  - Execution of actions simply to mislead the observer.
- Multiple agents
- Observations of failed actions

Solution: bounding the explanations

- Problem: You canít allow the insertion of unobserved actions to go on forever. You would generate ever more unlikely explanations for the same set of observations.

- Realization: Some actions are more likely to be done unobserved than others. (a stealthy port scan Vs. a denial of service)

- Solution:
  - Represent how likely it is that a given action is performed and not observed.
  - For each explanation compute the probability that the unobserved actions could have been performed unobserved.
  - Bound how unlikely an explanation we are willing to accept.
  - Reject all explanations that are below the threshold.

Talk Outline

- Background
- Problem
- Solution
  - General approach
  - Extension to hostile agents
- Implementation
  - Implementation details
  - Happy graphs
  - Analysis
- Conclusions

- Joint work with Robert Goldman