Leveraging and Learning Propositional Function for Large State Spaces

D Ellis Hershkowitz, David Abel, Gabriel Barth-Maron, Stephen Brawner, Kevin O'Farrell, James MacGlashan, Stefanie Tellex
Humans to Robots Laboratory
Department of Computer Science, Brown University

Leveraging

Goal: Enable autonomous agents to learn how to plan efficiently in massive stochastic state spaces using PFs

Want to Infer:

$$\Pr(a_i \in \mathcal{A}^*_{s,G} \mid s, G)$$

(Probability that action i is optimal for state s given goal G — i.e. a Bernoulli)

Estimating Action Optimality:

Tabularly solve a policy for easy worlds in domain, estimate action optimality

Use a set of binary features, (ϕs), to featurize states where p is a propositional function and g is a goal

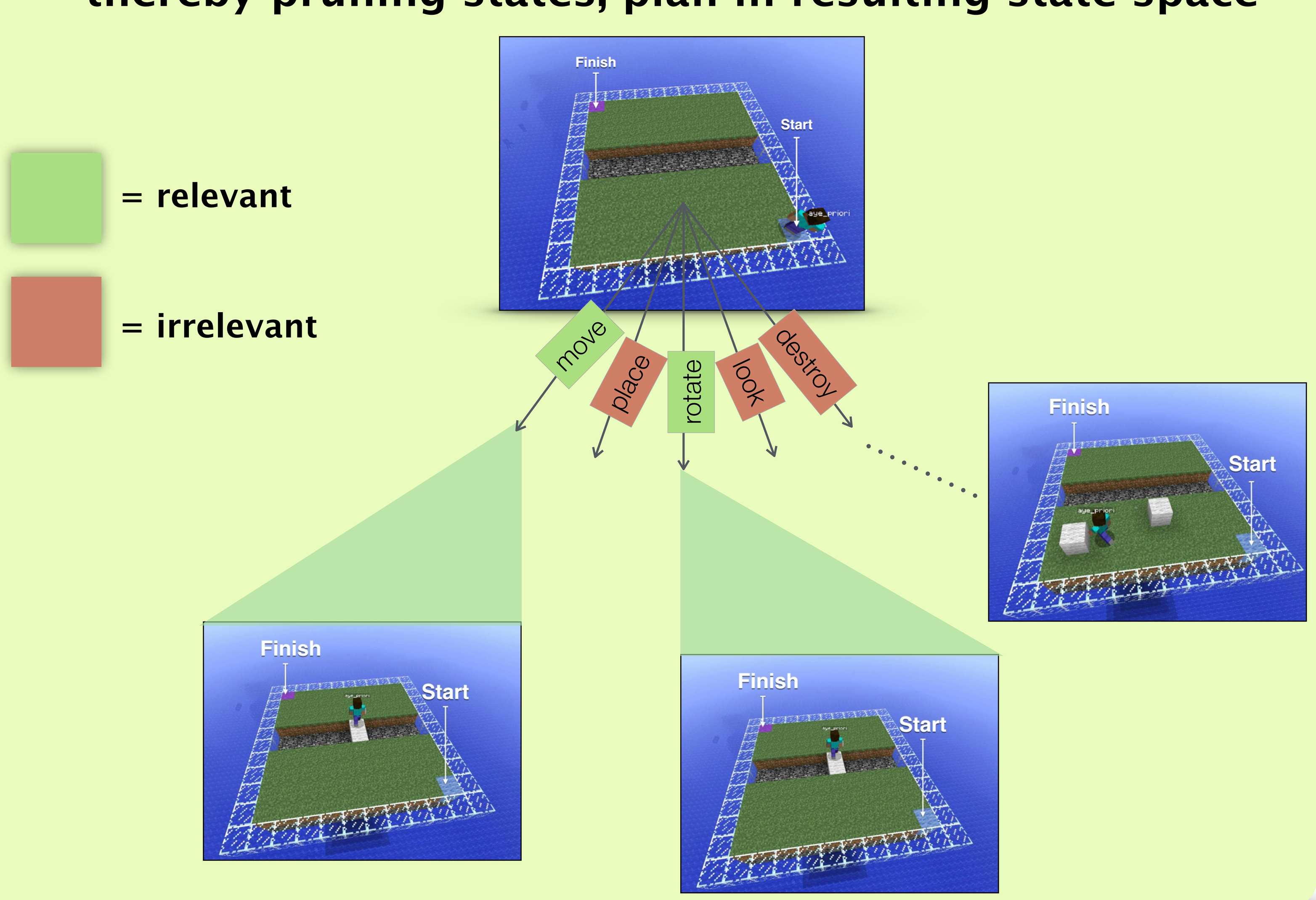
$$\phi_i = \begin{cases} 1 & p_i(s) \wedge g_i(G) \\ 0 & \text{otherwise.} \end{cases}$$

Train Naive Bayes classifiers on easy world policies

$$\Pr(a_i \in \mathcal{A}^* \mid s, G) \approx \alpha C(\phi_j, a_i) \frac{C(a_i)}{C(a_i) + C(\bar{a}_i)}$$

Action Pruning:

Prune actions unlikely to be involved in optimal plan, thereby pruning states, plan in resulting state space



Learning

Goal: Produce useful propositional functions given only an OO-MDP problem representation.

Relationally Featurize States:

First gather RL agent observations in domain

Featurize observed states based on the relative values of object attributes.

$$v(s) = \Big\Vert_{\forall (oClass_1, oClass_2, l) \in (\mathcal{O}^2 \times \mathcal{L})} \Big\Vert_{\forall (att_1, att_2,) \in (\mathrm{Att}(oClass_1) \times (oClass_2))} l_{att_1, att_2, oClass_1, oClass_2}(s)$$

Create Data Sets for Each DOORMAX Prediction:

Algorithm 7 DOORMAXLearn Modified to Generate D_v

```
INPUT: a set of observations O, a state featurizer v, a k in the DOORMAX sense
OUTPUT: a set of labeled datasets D_v

D_v ← ∅
ω ← ∅
//Loop over observations
for all (s, a, s') ∈ O do
  for all (oClass, att) ∈ (O × Att(oClass)) do
    updatedDataSets← ∅
    //Loop over hypothesized effects
    for all hypEffect ∈ eff_{oClass,att}(s, s') do
      //Check for dataset with prediction for this effect
      if ∃ d ∈ D_v s.t. d.pred.effect = hypEffect then
        d.add((v(s), true))
      else
        if (hypEffect.type, oClass, att, a) ∉ ω then
          newPrediction ← (a, null, hypEffect)
          newDataSet ← {(v(s), true)}
          newDataSet.prediction = newPrediction
          D_v.add(newPrediction)
          //Rule out effect type if more than K predictions
          relatedDataSets ← ∪d ∈ D_v s.t. d.prediction is related to newPrediction
          if |relatedDataSets|> k then
            ω.add((pred.effect, oClass, att, a))
            D_v.removeAll(relatedDataSets)
          end if
        end if
      end if
    end for
    //Update all datasets that did not receive a true with a false
    for all d ∈ D_v and ∉ updatedDataSets do
      d.add((v(s), false))
    end for
  end for
end for
return D_v
```

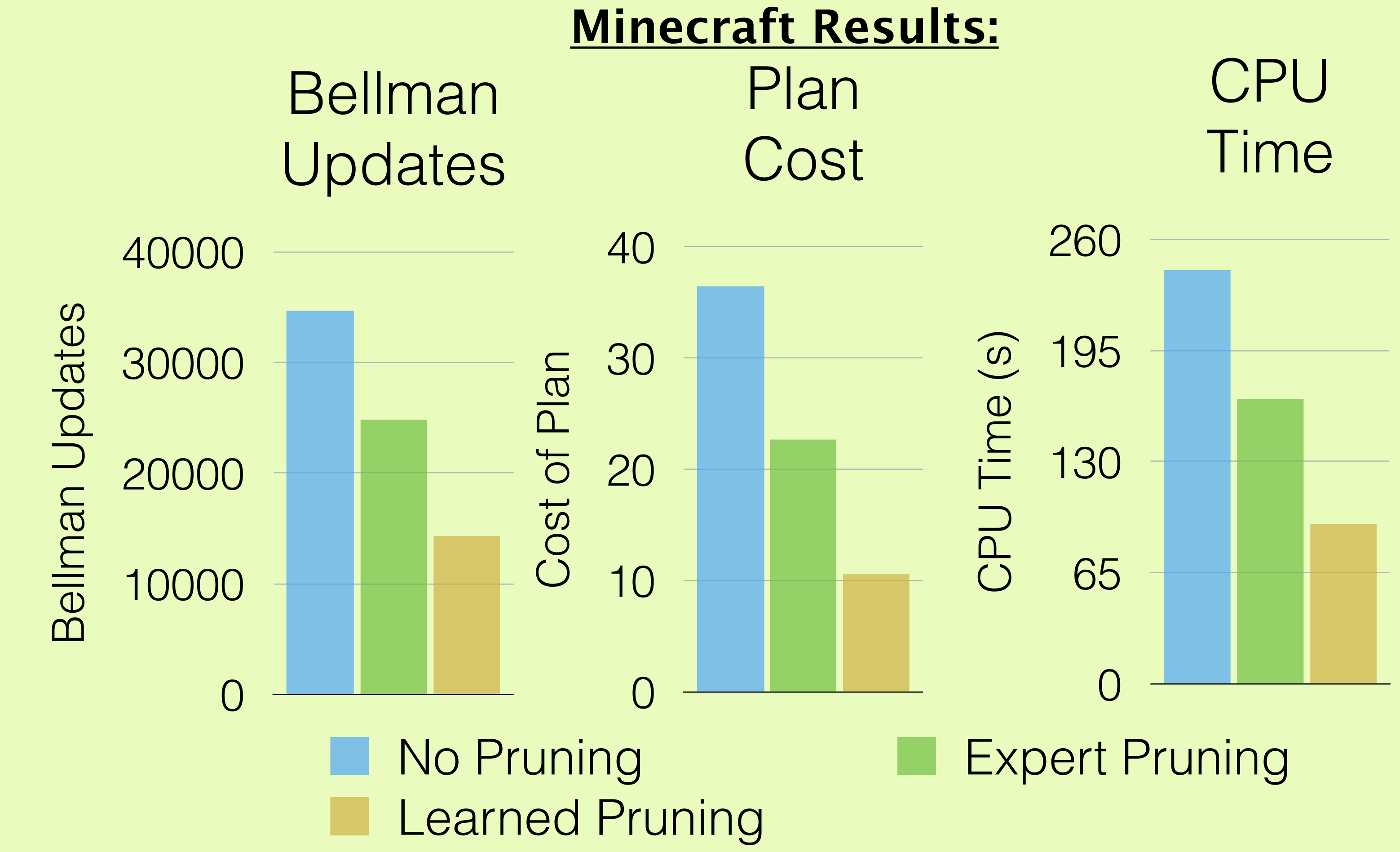
Turn Each Data Set into a PF:

Each dataset becomes a propositional function that results from the classify() routine of a J48 decision tree.

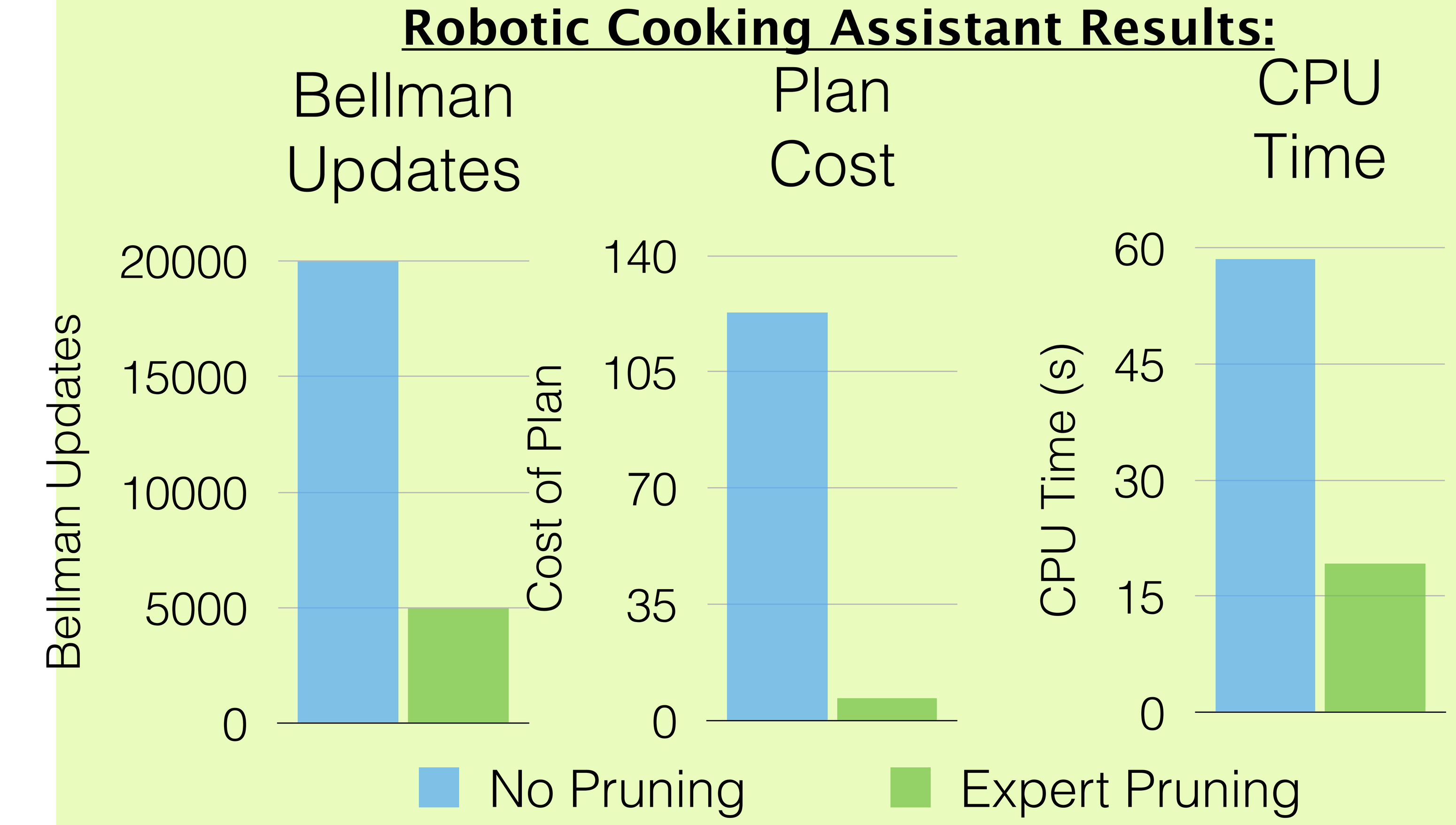
Experimental Results

Leveraging Results:

Minecraft Results:

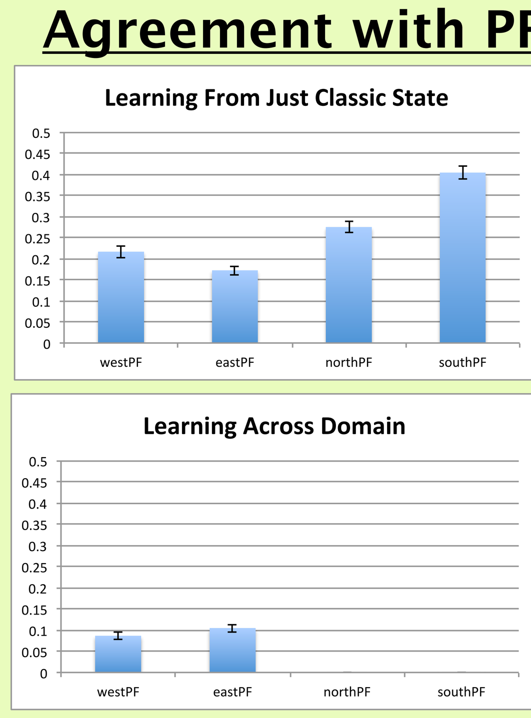


Robotic Cooking Assistant Results:



Learning Results:

Agreement with PFs



DOORMAX Without PFs