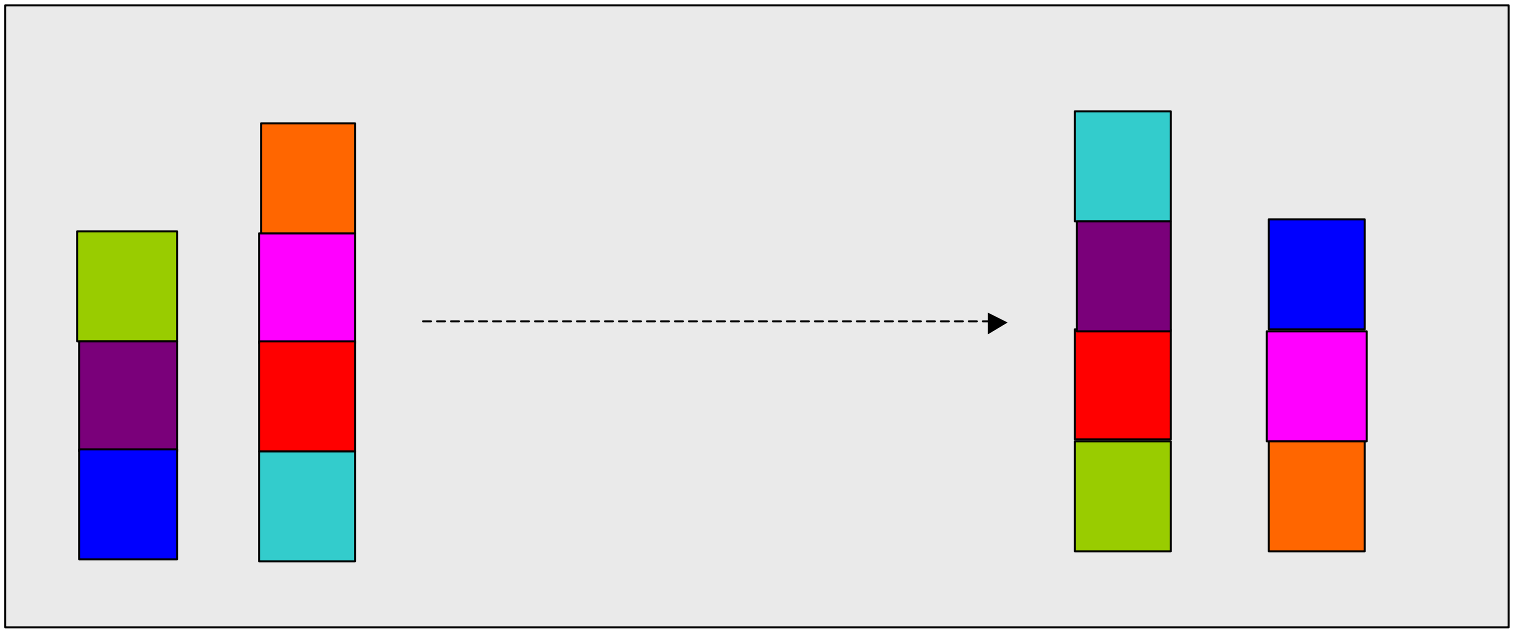
Reducing Planning to Classification

Alan Fern

Joint work w/ SungWook Yoon and Bob Givan

Electrical and Computer Engineering

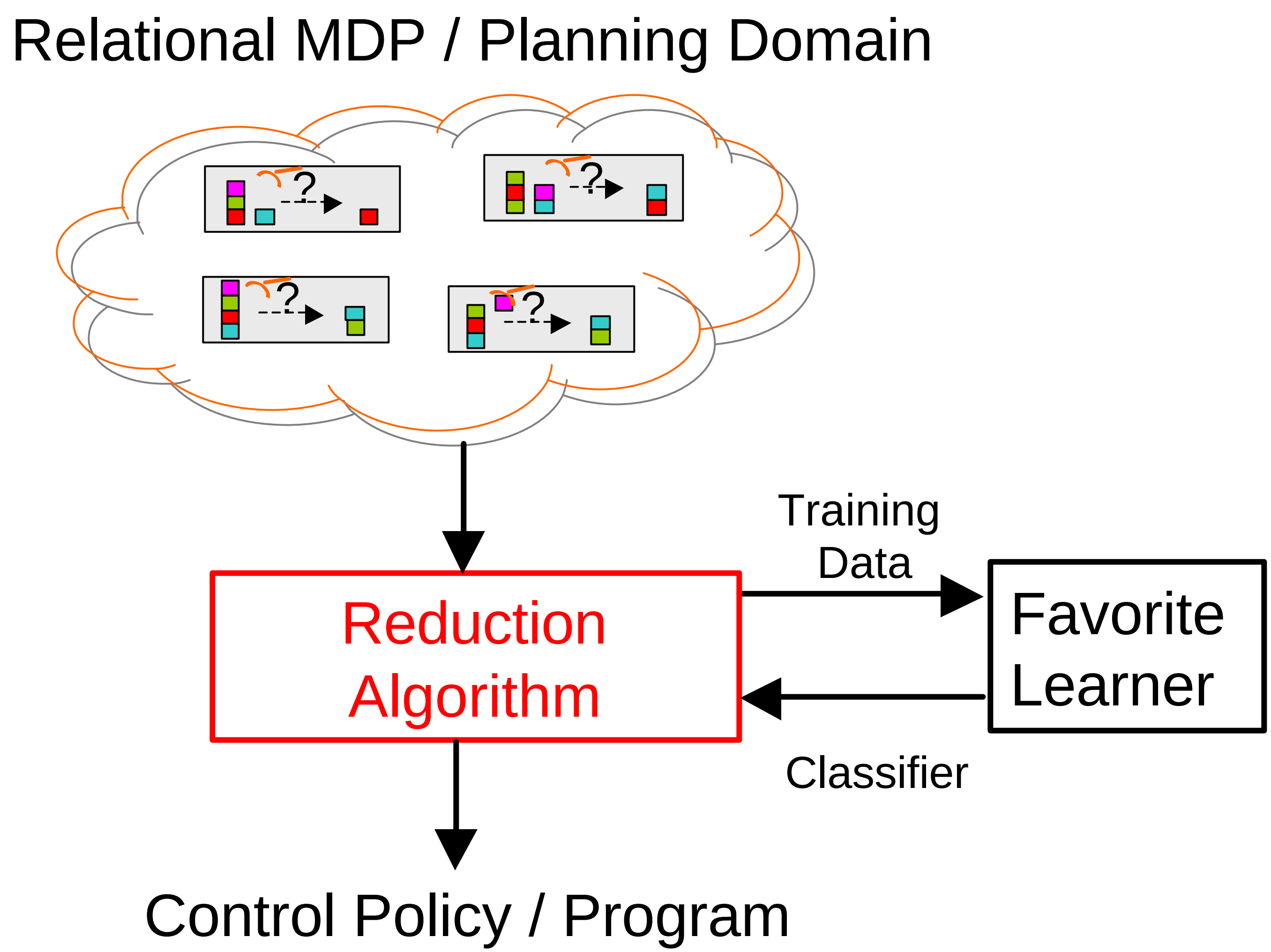
Purdue University



Available actions (possibly stochastic):

Pickup(x)

PutDown(x,y)



State-of-the-art on AI planning benchmarks.

Learning to Act

[Khardon, MLJ'99] gives PAC semantics linking classification and planning performance.

> Consider class of policies *C.*
> Observe $O(\log |C|)$ **trajectories** of target policy in *C*.
>
> If policy π in *C* is consistent with trajectories then quality of π is "probably close" to quality of target.

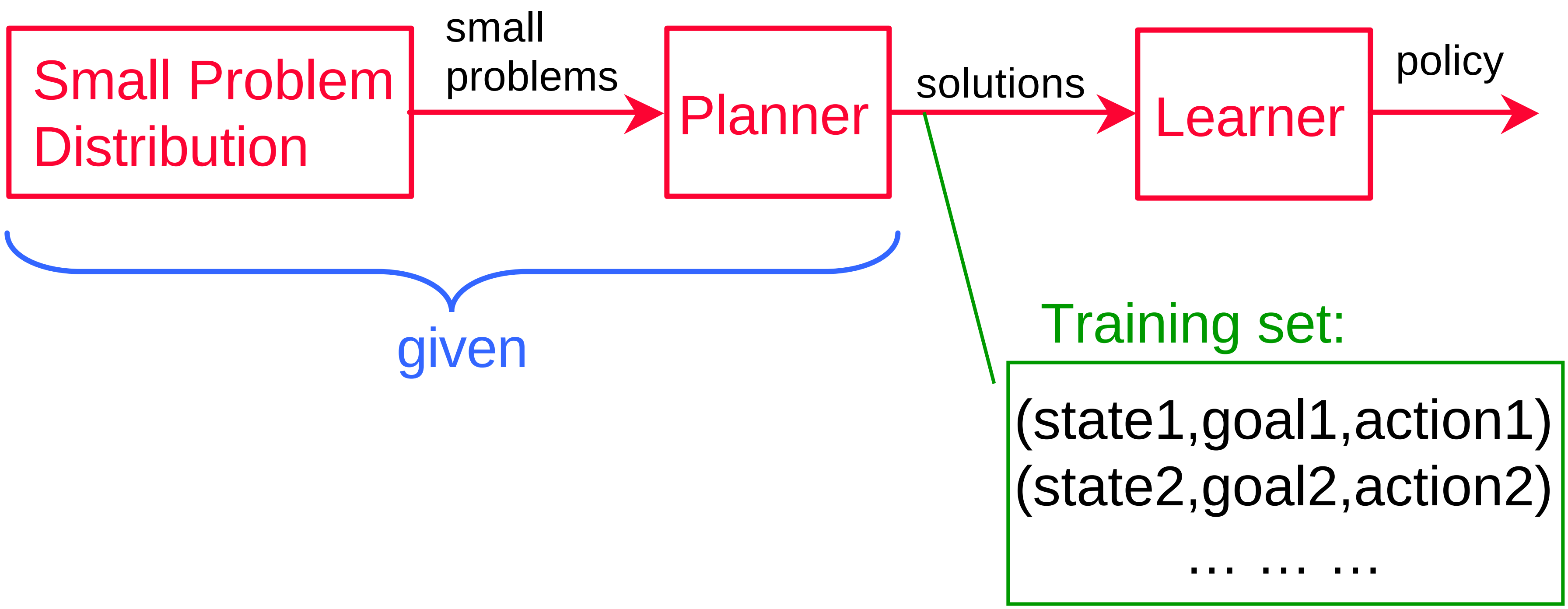
Suggests a type of reduction:

1) Somehow observe trajectories of a good policy.
2) Learn a classifier to (approximately) imitate the policy.

How can we observe a good policy?

Reduction 1: Learning to Solve Small Problems

[Khardon, AIJ 1999], [Martin&Geffner, KR 2000], [Yoon,Fern & Givan, 2002]



Generalizing to Large Problems

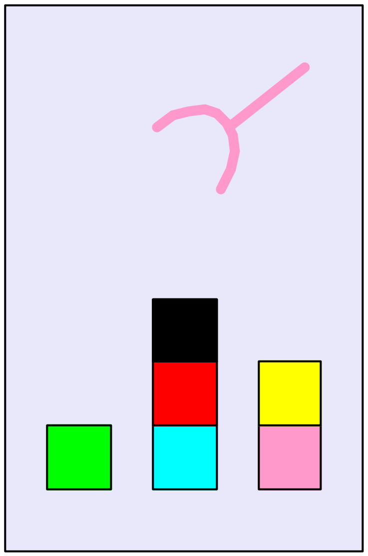
[Khardon, AIJ 1999], [Martin&Geffner, KR 2000], [Yoon,Fern & Givan, 2002]

Why expect policies to generalize to large problems?

- Select good policy language bias.
  - **Restrict expressiveness** to avoid overfitting.
  - But **expressive enough** to represent good policies.

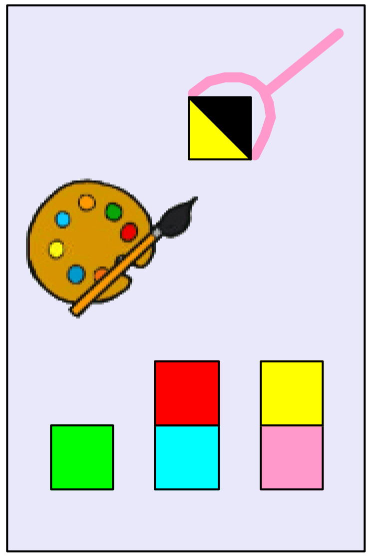
Experimental Domains

SBW(n)



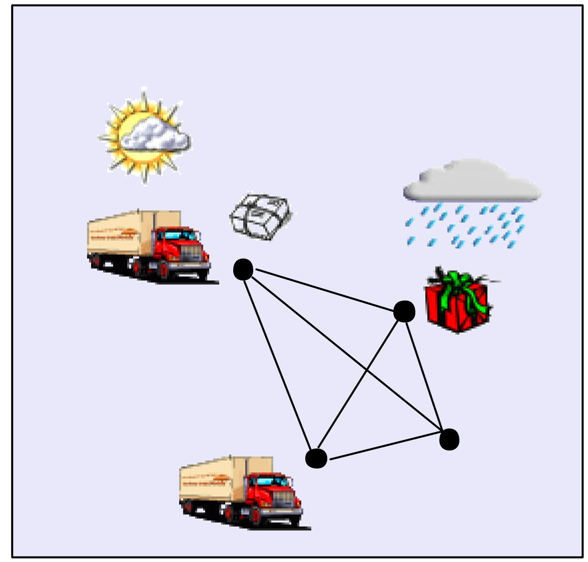
(Stochastic) Blocks World

SPW(n)

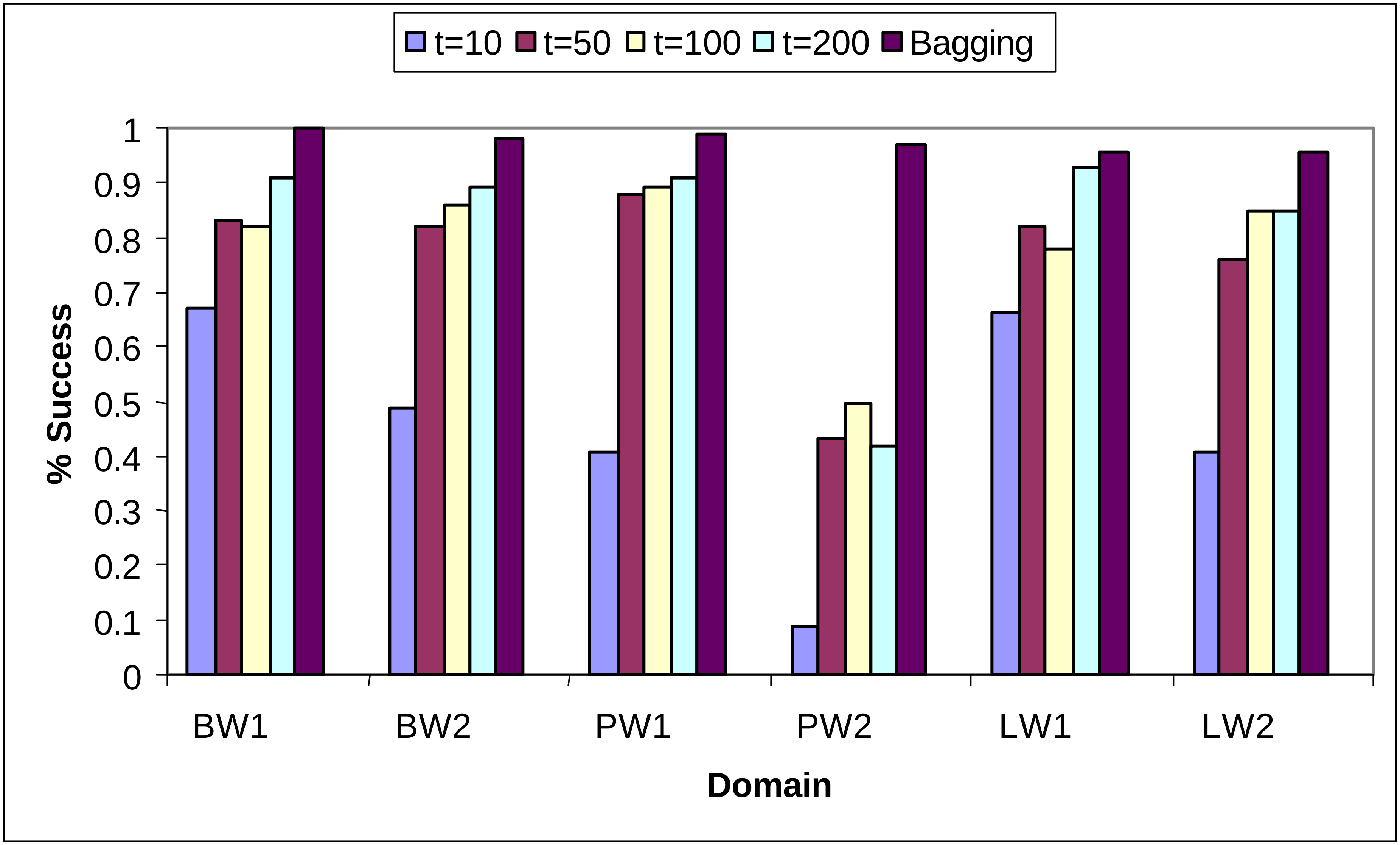


(Stochastic) Painted Blocks World

SLW(t,p,c)



(Stochastic) Logistics World

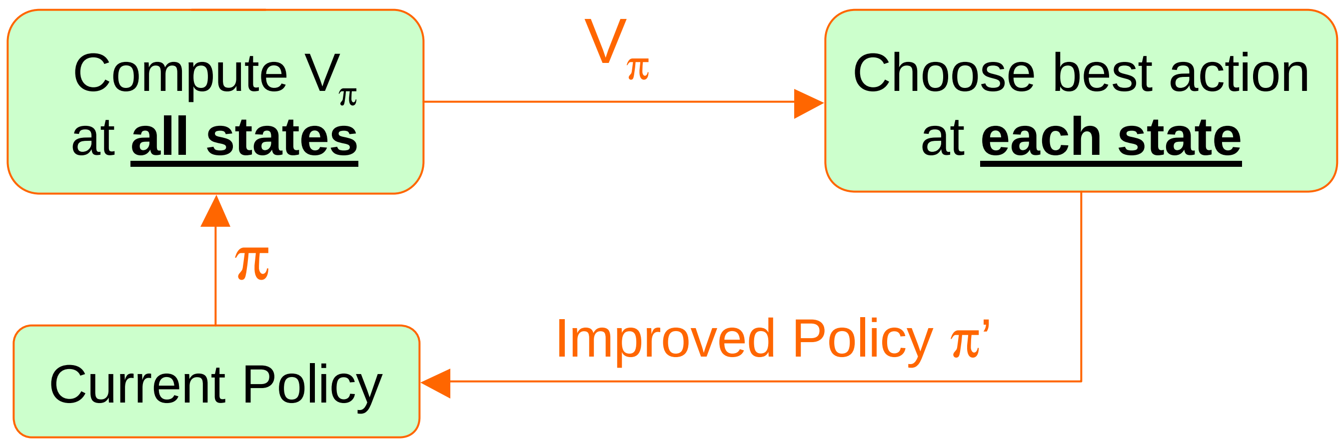

t=10
t=50
t=100
t=200
Bagging
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
% Success
BW1
BW2
PW1
PW2
LW1
LW2
Domain

Unsolved Problems

- Select policies without immediate access to small problems
  - Can we learn directly in a large domain?
- Improving buggy policies
  - All previous techniques produce policies with occasional fatal flaws.
- Our approach: use standard MDP technique of (approximate) policy iteration

Flowchart View of Policy Iteration

$V_\pi(s)$ = "value" of following π starting at s

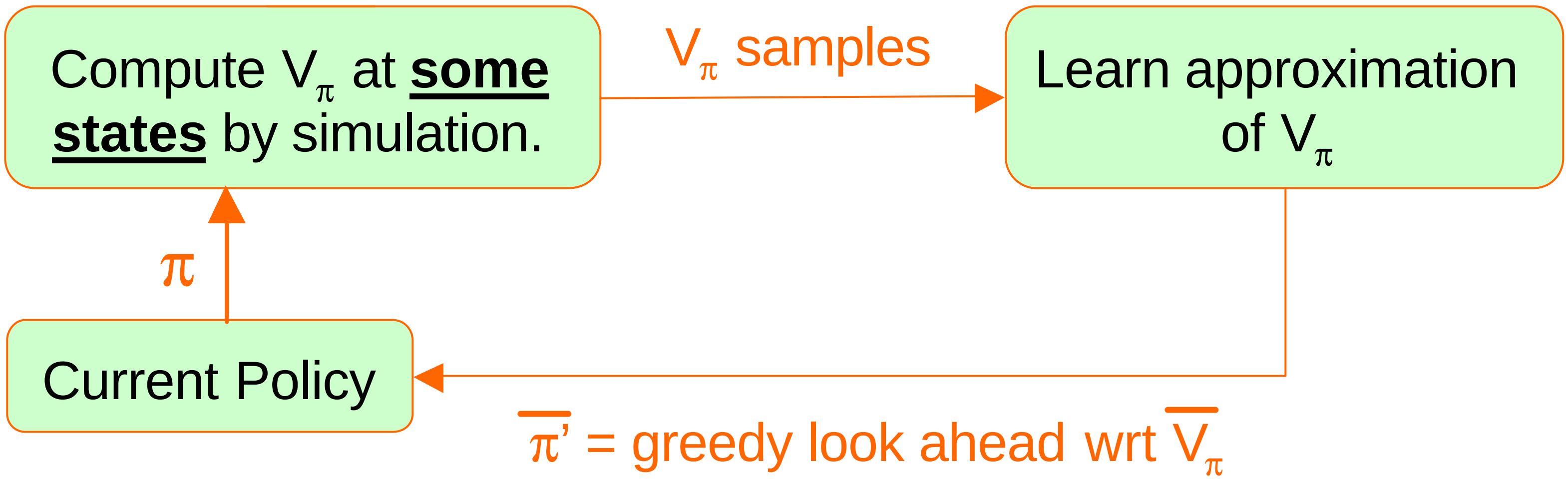


Guaranteed finite convergence to optimal policy.

Problem:
too many states

Approximate Policy Iteration

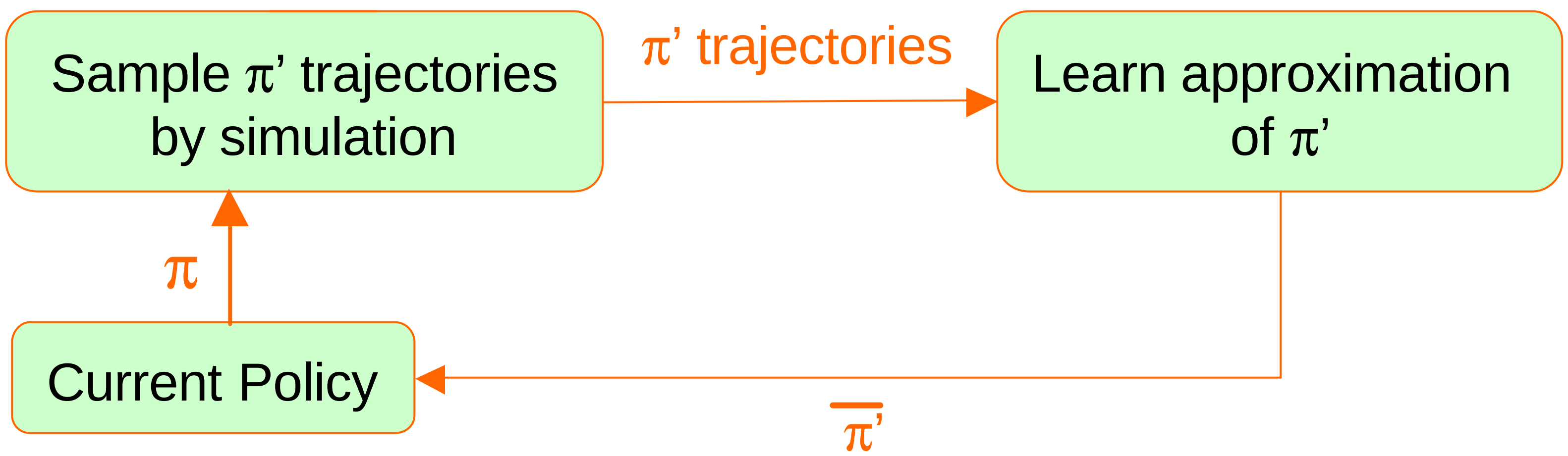
Usual Approach: reduce to value function approximation



- Value functions can be harder to represent than policies.
- Learning a policy directly may be more effective.

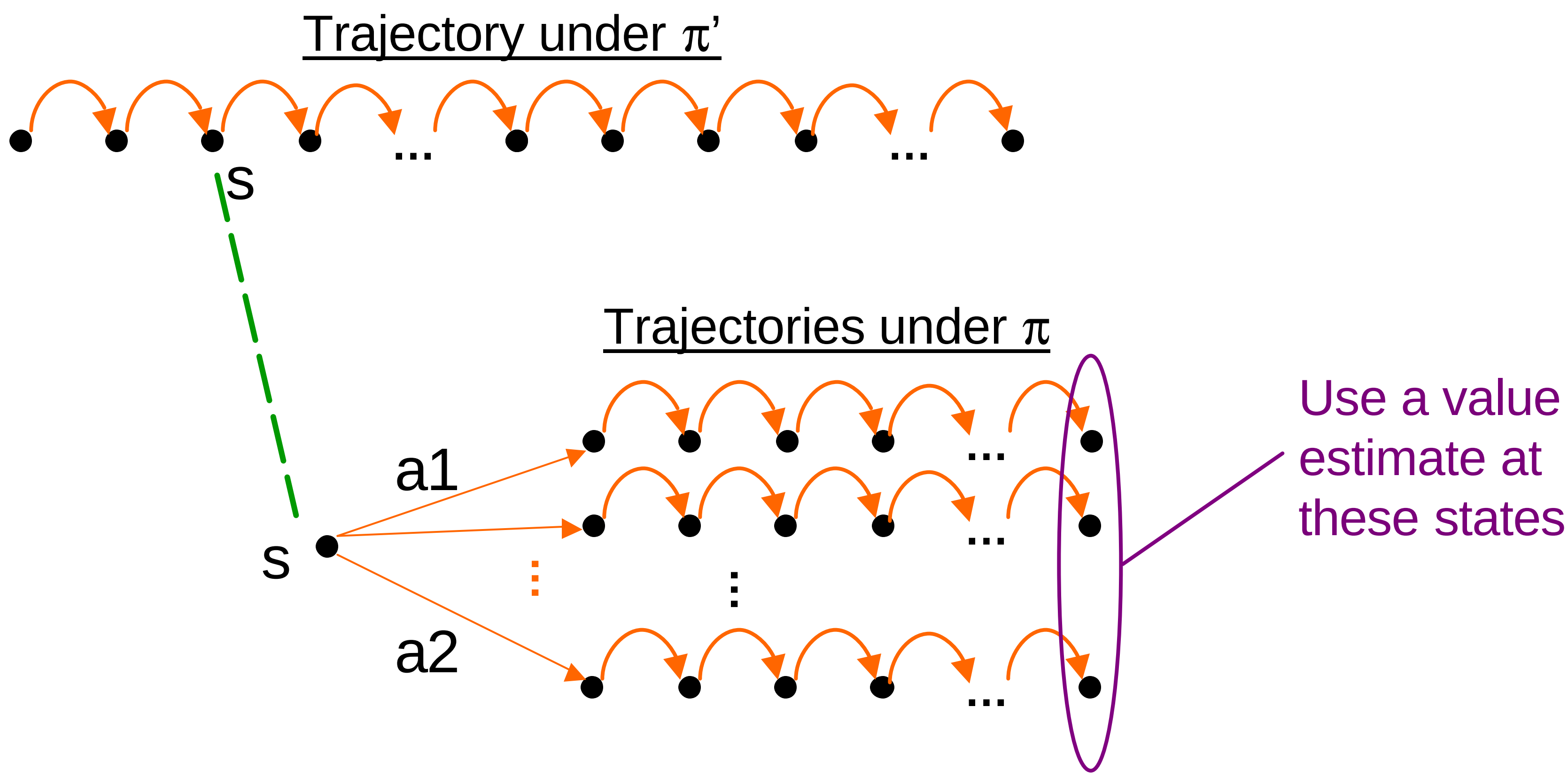
Approximate Policy Iteration

<u>Our Approach</u>: reduce to classifier learning



<u>Refinement</u>: Reduce to cost-sensitive classification. Costs based on Q-values.

Rollout: Computing π' Trajectories



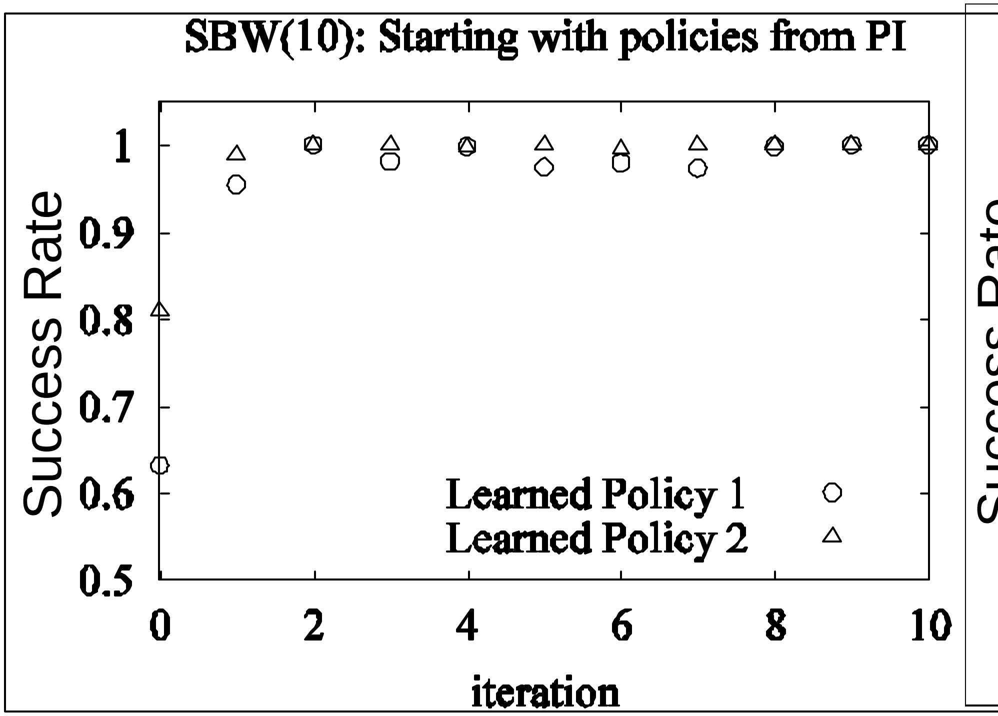
- For our relational planning domains we use the FF-plan plangraph heuristic

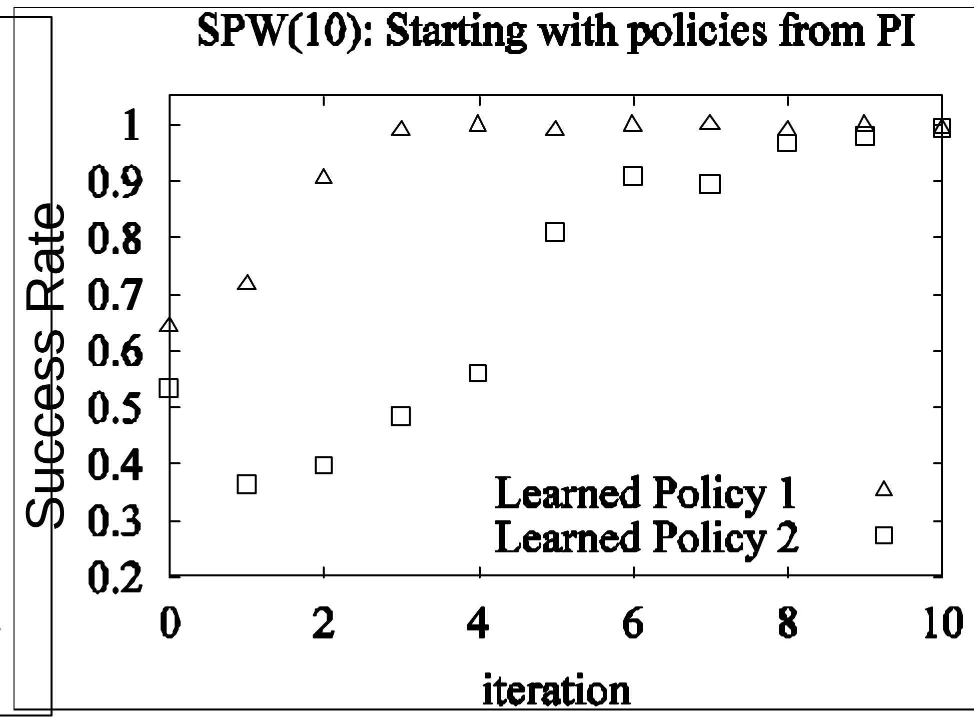
Initial Policy Choice

- Policy iteration requires an initial *base policy*
- Options include:
  - random policy
  - greedy policy with respect to a planning heuristic
  - policy learned from small problems

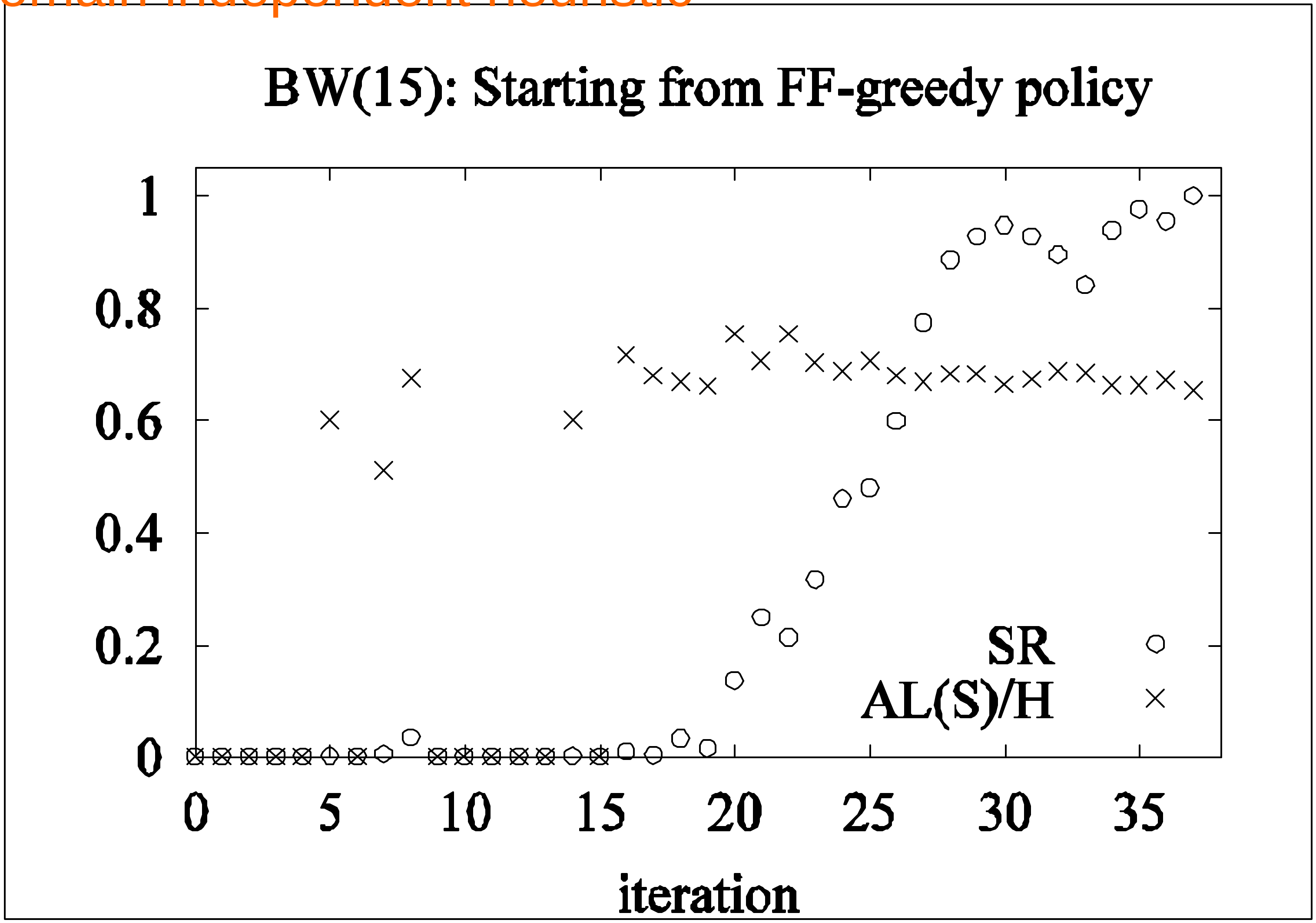
API Results

Starting with flawed policies learned from small problems





Starting with a policy greedy with respect to a domain independent heuristic



Ongoing and Future Work

- Explore new policy languages
  - E.g. relative value functions [Dietterich & Wang, NIPS'02]
- Approximation guarantees.
- Generalize to domains that "require search".
- Incorporating deductive reasoning.
- Generalize to games and partial observability.
  - E.g. the game of Hearts.

Questions?