IJCAI workshop on Generalization in Planning (GenPlan 22)

Learning Generalized Policy Automata for Relational Stochastic Shortest Path Problems

Rushang Karia, Rashmeet Nayyar, Siddharth Srivastava

AAIR
Autonomous Agents and Intelligent Robots

Arizona State University

Motivation

- A planetary rover needs to transport rocks back to the base for analysis.
  - The rover has a slippery gripper that can hold a single rock.
  - It is possible to reach any location from any given location.
- Actions have certain predictable, but stochastic elements.
  - Given the nature of the mission, good models of the effects of actions are available.

Image credit: https://mars.nasa.gov/msl/mission/overview/

How do we represent states and actions?

Image-based

- Some aspects of the state are not visible *eg. power, coeff. friction*
- Many problems cannot be naturally described using images.

Logic-based

$$s = \{\texttt{rock}(r_1), \texttt{loc}(l_1), \texttt{rock-at}(r_1, l_1)\}$$

- Many problems well-suited to such representations.
- Good heuristics available.

Image credit: https://mars.nasa.gov/msl/mission/overview/

How do we represent states and actions?

Image-based

- Some aspects of the state are not visible *eg. power, coeff. friction*
- Many problems cannot be naturally described using images.

Logic-based

$$s = \{\texttt{rock}(r_1), \texttt{loc}(l_1), \texttt{rock-at}(r_1, l_1)\}$$

- Many problems well-suited to such representations.
- Good heuristics available.

How do we efficiently solve such problems?

Image credit: https://mars.nasa.gov/msl/mission/overview/

Generalized Relational Stochastic Shortest Path Problems (GSSPs)

- A domain D consisting of predicates **P** and actions **A**.

- A GSSP problem for a domain D is a tuple <O, S, A, s_0, G, T, C>
    - **O** is a set of objects.
    - **S** and **A** are sets of states and actions instantiating using **O** and **D**..
    - $\mathbf{s_0}$ is the initial state and **G** is a set of goal states.
    - **T** is the transition function and **C** is the cost function.

- We chose GSSPs over SSPs since they allow deadends to exist.

Solution to GSSPs

- Solutions are expressed as policies π: S →A

- GSSP solvers compute policies by iteratively solving Bellman equations over states reachable from the initial state.

$$v^i(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^{i-1}(s')]$$

- Many solvers typically initialize values with heuristic estimates; $v^0(s) = h(s)$

The Need for Generalization

- State spaces grow exponentially as state variables increase.

- Most of the existing solvers are “stateless”.

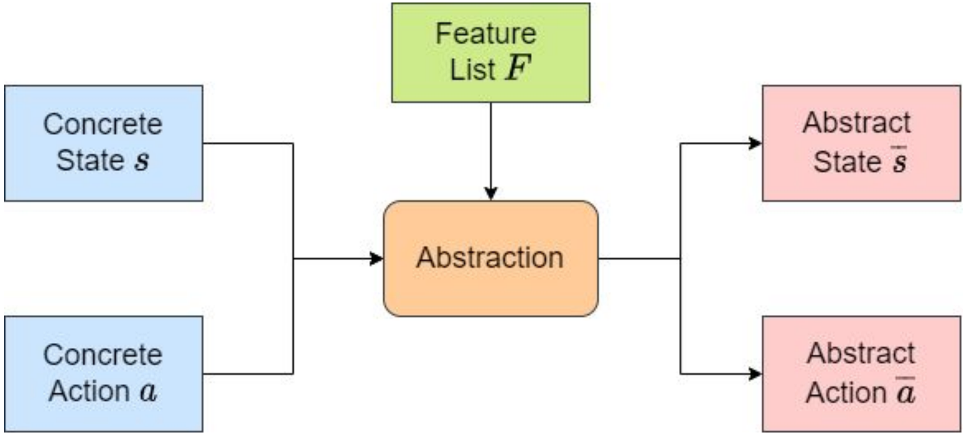
Example

1. Planetary rover problem with 1 rock across 1 location.
2. Planetary rover problem with 10 rocks across 10 locations.

Our Approach: GPA-accelerated SSP Solvers

- We use abstraction to create Generalized Policy Automata (GPA): abstract hypergraphs that encode partial policies.
- GPAs can be used to prune away large parts of the search space.
- SSP solvers operate over this reduced search space to quickly find solutions.
- Theoretical guarantees of hierarchical optimality and completeness.

Abstraction



- Given a list of features, we usefeature kernels that convert concrete states and actions into abstract states and actions.
- We use Canonical Abstraction [Sagiv et al.; 2002] to automatically generate features and feature kernels in a domain-independent fashion.

Learning GPAs

Given a set of solution policies $\pi_1, \ldots, \pi_n$, we learn GPAs by using abstraction to iteratively merge GPAs.

Learning GPAs

1. We represent the solution policy as a directed hypergraph.
2. We convert each directed hypergraph to a GPA using abstraction.
3. We iteratively merge each GPA one by one.

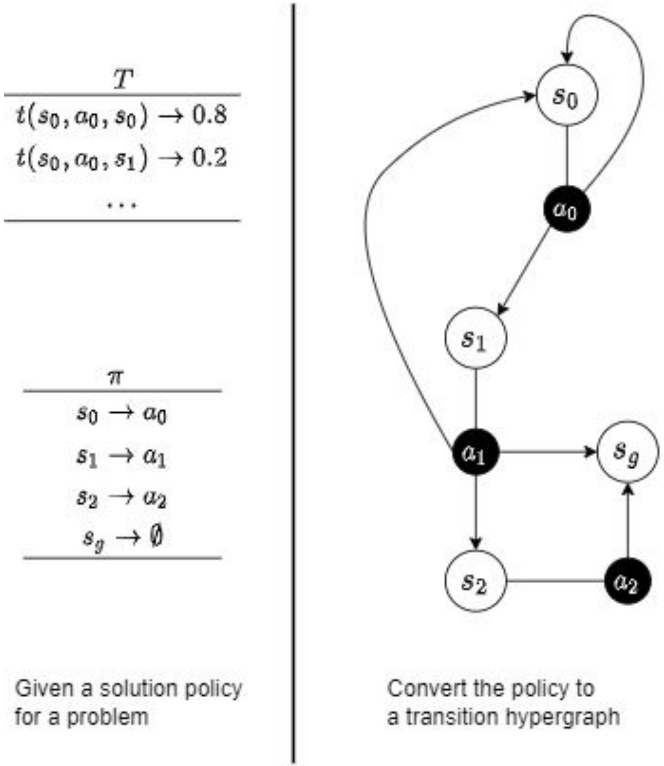
T
$t(s_0, a_0, s_0) \rightarrow 0.8$
$t(s_0, a_0, s_1) \rightarrow 0.2$
$\ldots$

π
$s_0 \rightarrow a_0$
$s_1 \rightarrow a_1$
$s_2 \rightarrow a_2$
$s_g \rightarrow \emptyset$

Given a solution policy
for a problem

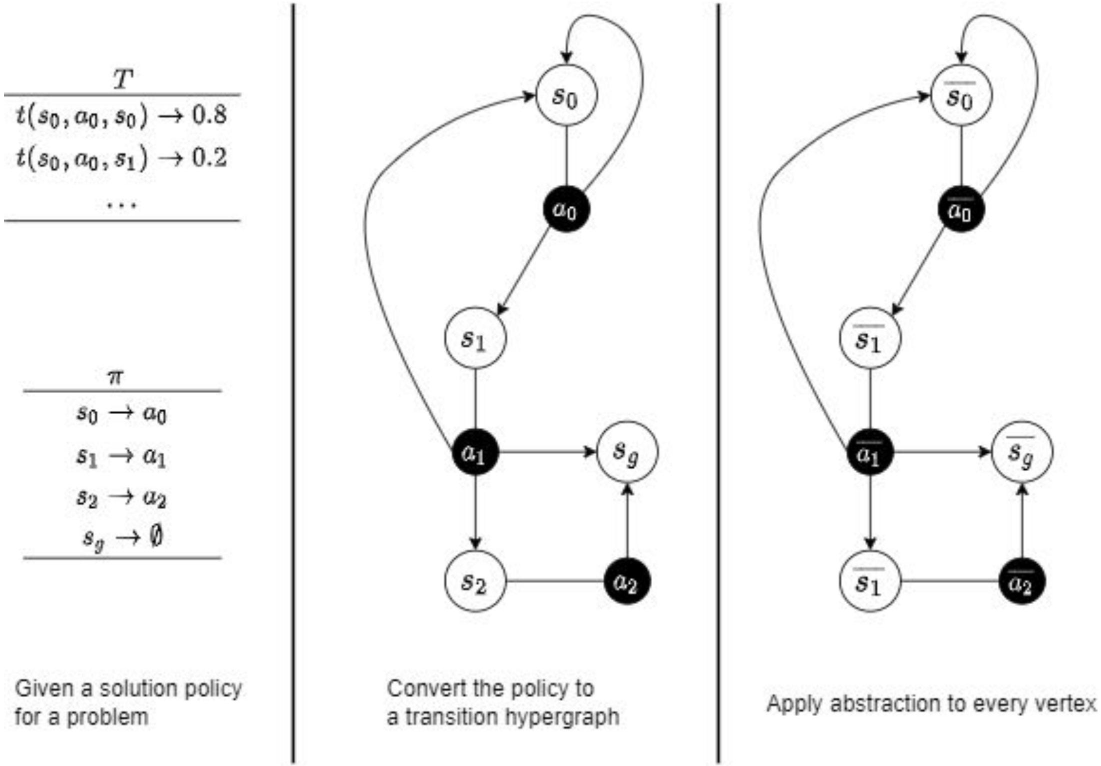
Learning GPAs

1. **We represent the solution policy as a directed hypergraph.**
2. We convert each directed hypergraph to a GPA using abstraction.
3. We iteratively merge each GPA one by one.



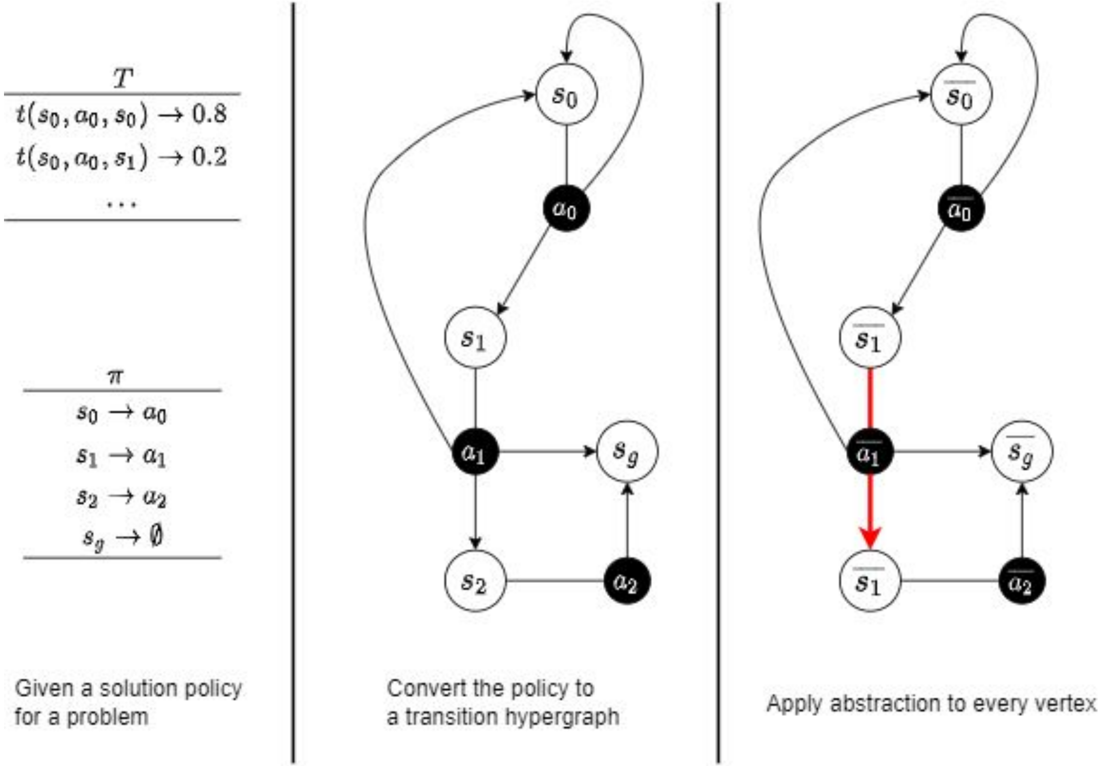
Learning GPAs

1. We represent the solution policy as a directed hypergraph.
2. **We convert each directed hypergraph to a GPA using abstraction.**
3. We iteratively merge each GPA one by one.



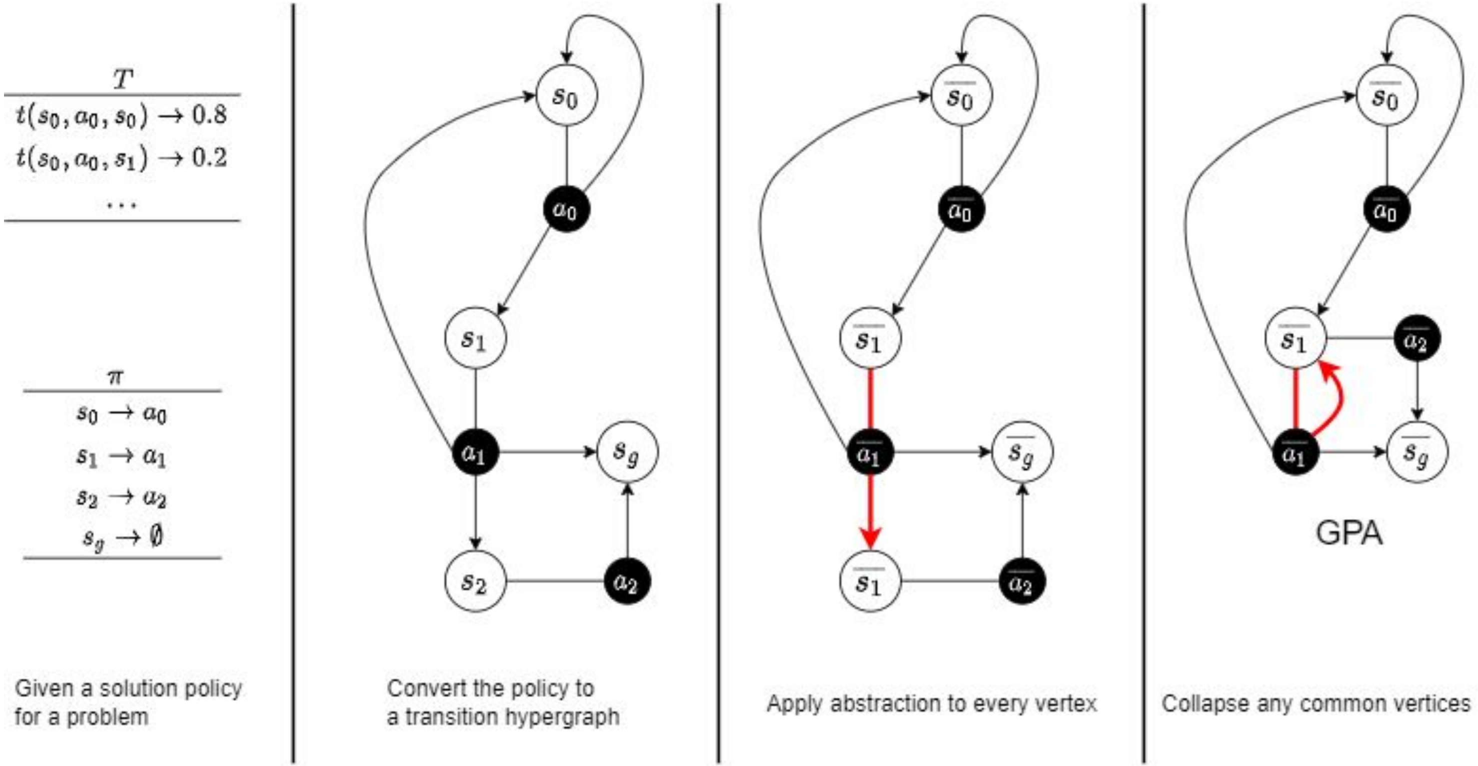
Learning GPAs

1. We represent the solution policy as a directed hypergraph.
2. **We convert each directed hypergraph to a GPA using abstraction.**
3. We iteratively merge each GPA one by one.



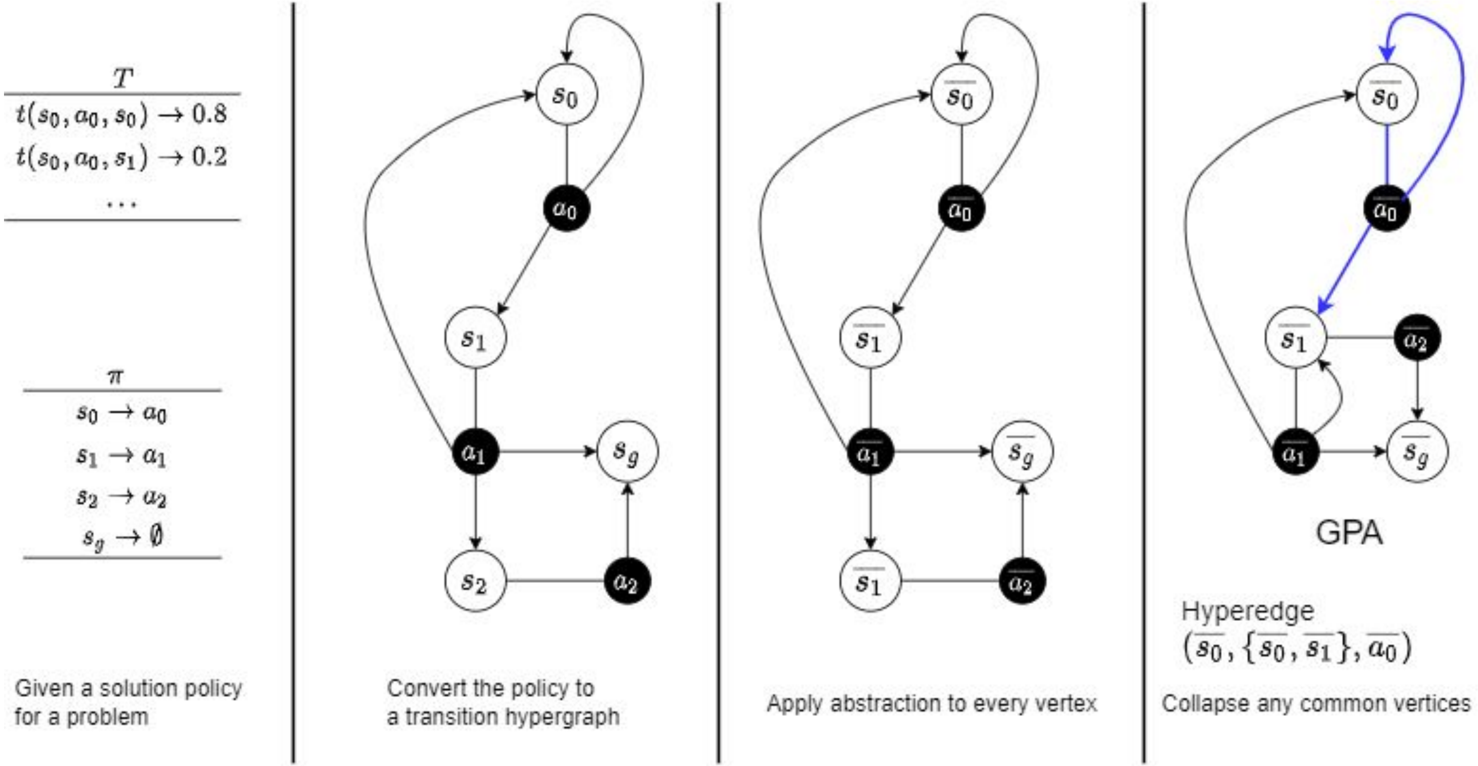
Learning GPAs

1. We represent the solution policy as a directed hypergraph.
2. **We convert each directed hypergraph to a GPA using abstraction.**
3. We iteratively merge each GPA one by one.



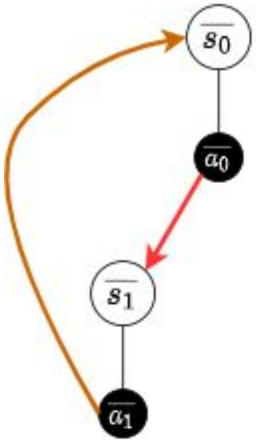
Learning GPAs

1. We represent the solution policy as a directed hypergraph.
2. **We convert each directed hypergraph to a GPA using abstraction.**
3. We iteratively merge each GPA one by one.

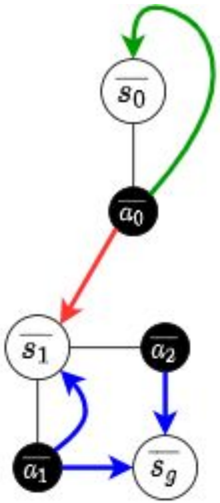


Merging GPAs

1. We represent the solution policy as a directed hypergraph.
2. We convert each directed hypergraph to a GPA using abstraction.
3. **We iteratively merge each GPA one by one.**



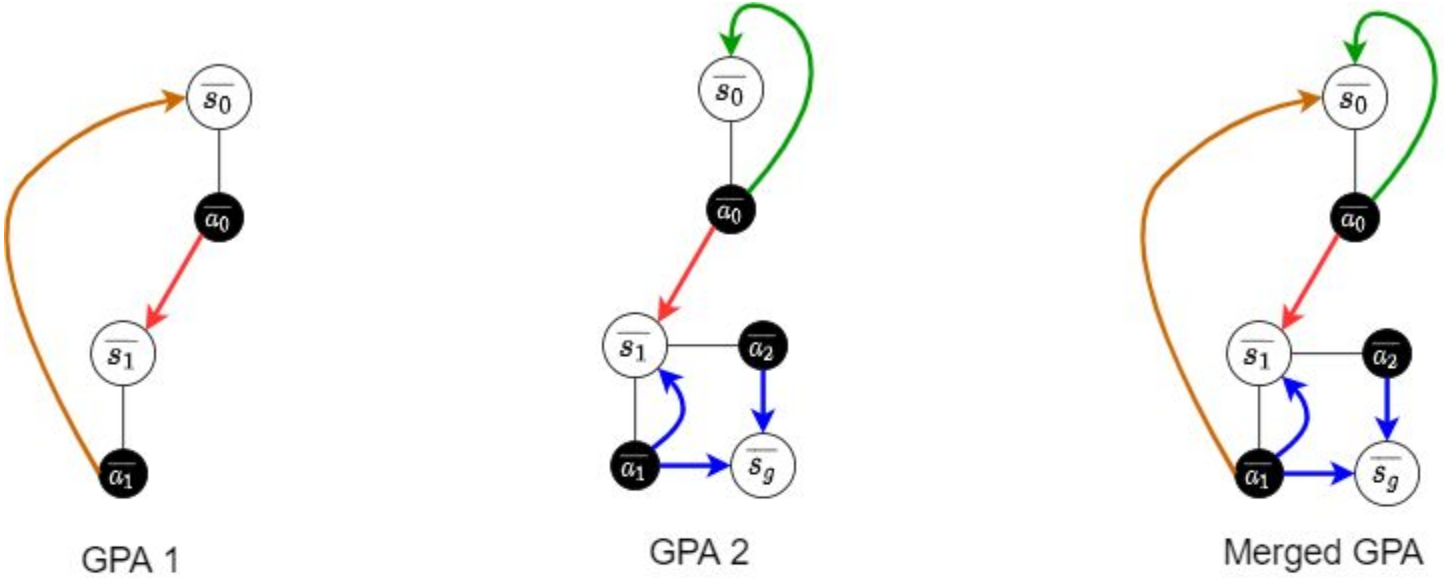
GPA 1



GPA 2

Merging GPAs

1. We represent the solution policy as a directed hypergraph.
2. We convert each directed hypergraph to a GPA using abstraction.
3. **We iteratively merge each GPA one by one.**



Using GPAs for solving SSPs

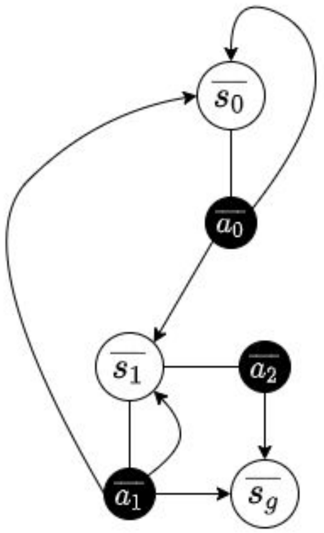
Intuitively, we modify the cost function for transitions that do not appear in the GPA to prune the search space.

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s, a, s')[c(s, a, s') + v^0(s')]$$

Consider the transition

$$t(s_0, a_0, s_2) = 1$$



GPA

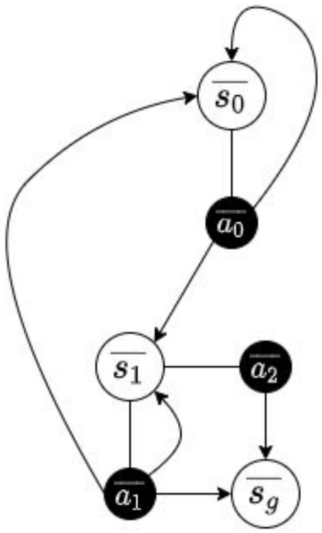
Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition

$$t(s_0, a_0, s_2) = 1$$

$$v^1(s_0) = \quad c(s_0, a_0, s_2) + v^0(s_2)$$

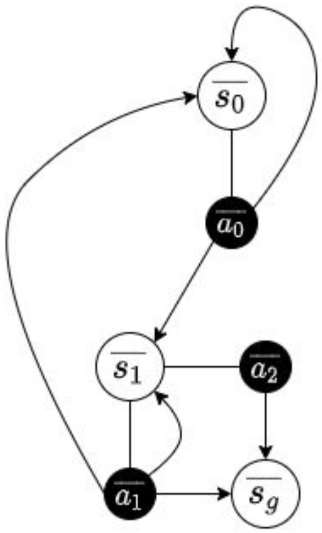


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$



GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition

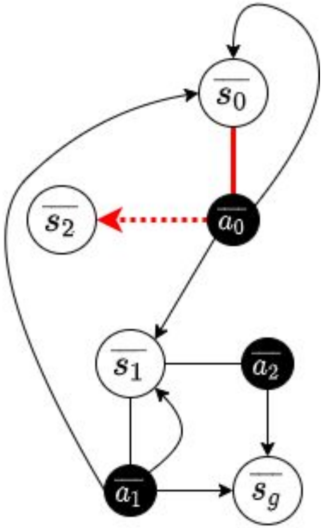
$$t(s_0, a_0, s_2) = 1$$

$v^1(s_0) =$

$$c(s_0, a_0, s_2) + v^0(s_2)$$

Apply Abstraction

$$(\overline{s_0}, \overline{a_0}, \overline{s_2})$$

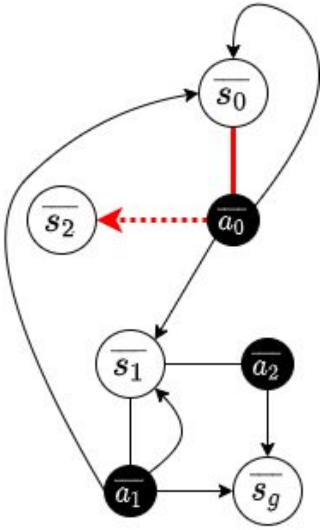


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$
Modify the cost function to ∞ for transitions $t \notin$ GPA	$\infty + v^0(s_2)$

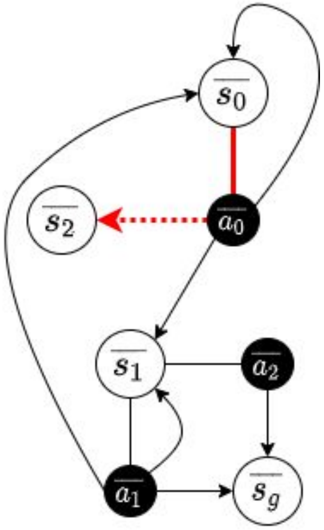


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$
Modify the cost function to ∞ for transitions $t \notin$ GPA	$\infty + v^0(s_2)$
	Pruned

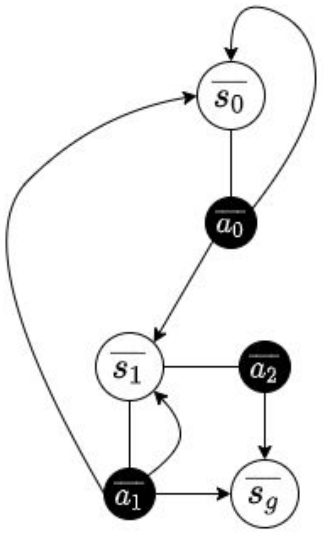


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$	$t(s_0, a_1, s_3) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$	$c(s_0, a_1, s_3) + v^0(s_3)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$	$(\overline{s_0}, \overline{a_0}, \overline{s_1})$
Modify the cost function to ∞ for transitions $t \notin$ GPA	$\infty + v^0(s_2)$	
	Pruned	

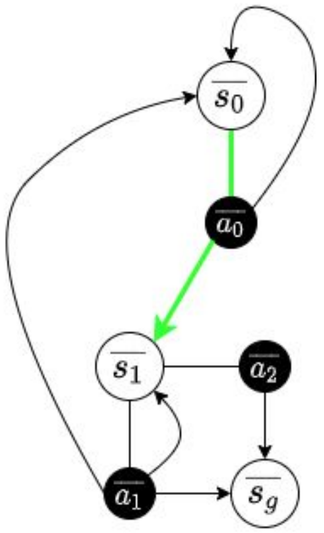


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$	$t(s_0, a_1, s_3) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$	$c(s_0, a_1, s_3) + v^0(s_3)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$	$(\overline{s_0}, \overline{a_0}, \overline{s_1})$
Modify the cost function to ∞ for transitions $t \notin$ GPA	$\infty + v^0(s_2)$	
	Pruned	

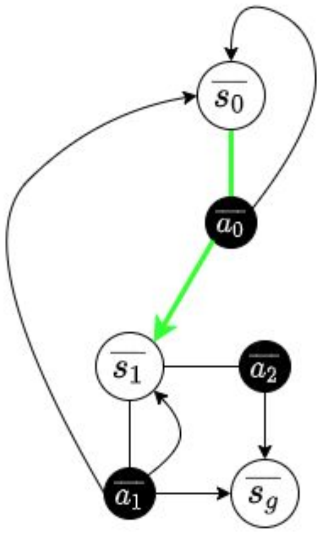


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$	$t(s_0, a_1, s_3) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$	$c(s_0, a_1, s_3) + v^0(s_3)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$	$(\overline{s_0}, \overline{a_0}, \overline{s_1})$
Modify the cost function to ∞ for transitions $t \notin$ GPA	$\infty + v^0(s_2)$	$c(s_0, a_1, s_3) + v^0(s_3)$
	Pruned	

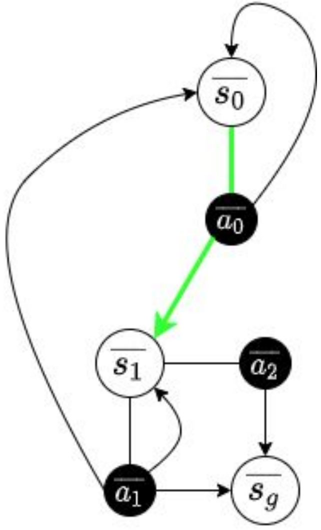


GPA

Example

$$v^1(s) = \min_a \sum_{s' \in S} t(s,a,s')[c(s,a,s') + v^0(s')]$$

Consider the transition	$t(s_0, a_0, s_2) = 1$	$t(s_0, a_1, s_3) = 1$
$v^1(s_0) =$	$c(s_0, a_0, s_2) + v^0(s_2)$	$c(s_0, a_1, s_3) + v^0(s_3)$
Apply Abstraction	$(\overline{s_0}, \overline{a_0}, \overline{s_2})$	$(\overline{s_0}, \overline{a_0}, \overline{s_1})$
Modify the cost function to ∞ for transitions $t \notin$ GPA	$\infty + v^0(s_2)$	$c(s_0, a_1, s_3) + v^0(s_3)$
	Pruned	**Allowed**



GPA

Theoretical Analysis

- **Thm 3.1:** Our approach is complete.

- **Thm 3.2:** Our approach is guaranteed to find hierarchically optimal policies.
    - Hierarchically optimal **given** the training data.
    - Policy improvement is guaranteed when falling back to the original cost function.

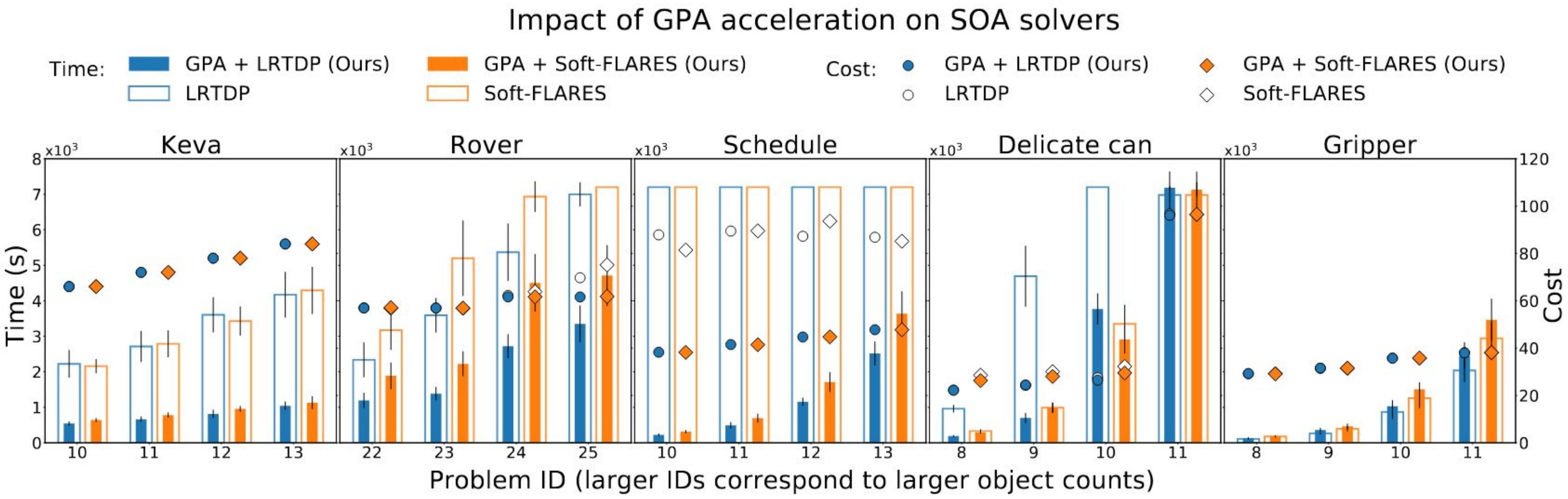
Empirical Evaluation

- 4 benchmark domains.
  - 1 from the International Probabilistic Planning Competition (IPPC)
  - 1 from robotics
  - 2 stochastic versions of International Planning Competition (IPC) domains.
- 2 baseline solvers.
  - LRTDP (Bonet and Geffner; 2003)
  - Soft-FLARES (Pineda et al.; 2019)
- Few-shot training.
  - Solution policies from less than 20 problems.
  - Training time was less than 10 seconds.
- Large test problems.
  - Test set consisted of problems that had more than 2x the object counts of the training set.

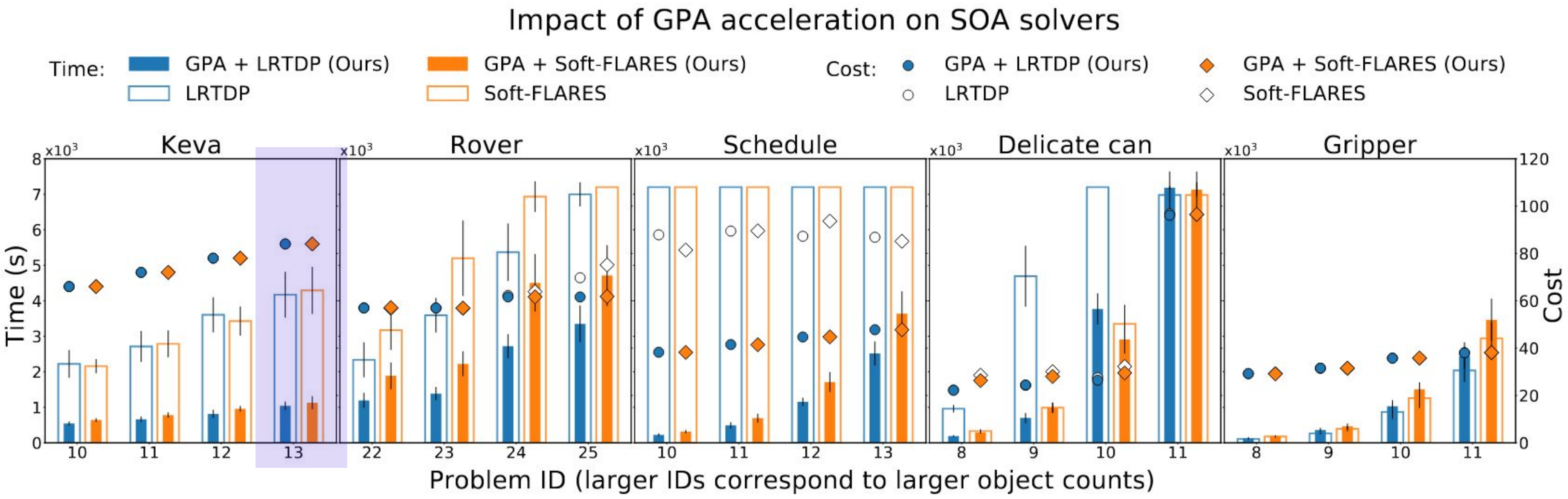
Evaluation Methodology

- Run a solver until convergence or 7200 seconds.
- Simulate policy computed 100 times with a horizon limit of 100.
- Report avg. cost incurred.
- 10 total runs.

Empirical Evaluation

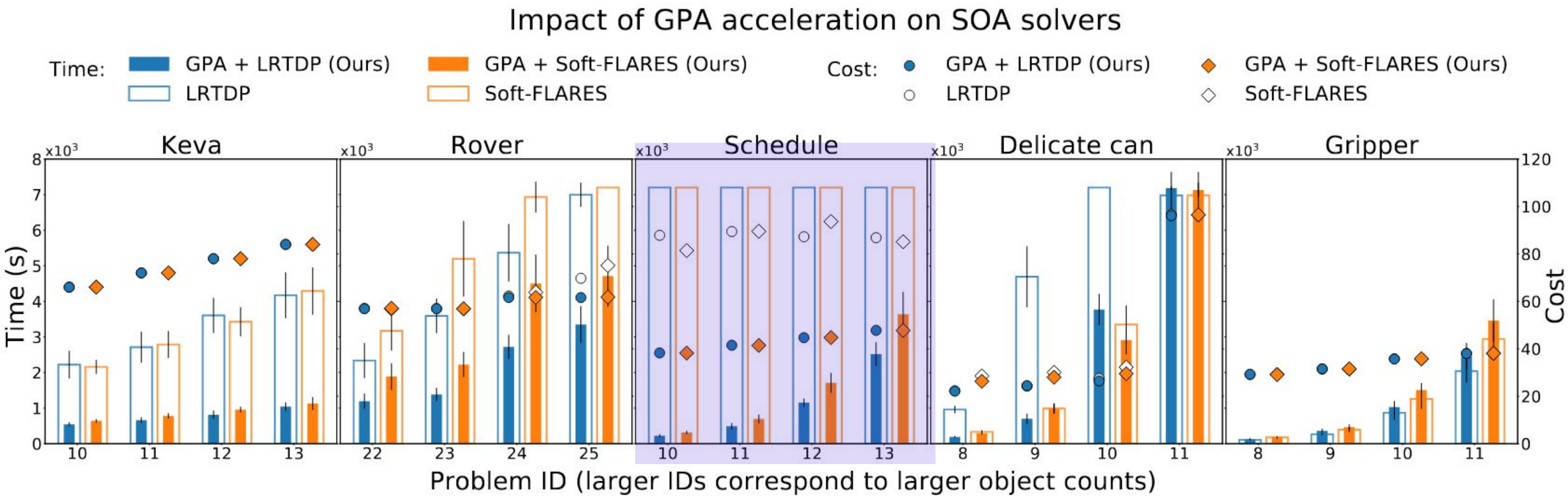


Empirical Evaluation



GPA accelerated solvers can solve problems much faster than baselines.

Empirical Evaluation



Sometimes baselines do find a policy that can yield equivalent cost, but they provide no convergence guarantees whereas our approach hierarchically converges in a fraction of the time.

Conclusions

- We introduced an approach that uses abstraction to synthesize GPAs.
- GPAs can be used to prune large parts of the search space.
- Our results show that embedding GPAs results in significant time savings.

Future Work

- Utilize description logic based abstractions.
- Add memory to the GPAs.

Conclusions

- We introduced an approach that uses abstraction to synthesize GPAs.
- GPAs can be used to prune large parts of the search space.
- Our results show that embedding GPAs results in significant time savings.

Future Work

- Utilize description logic based abstractions.

Thank you!