Generalized Planning: Synthesizing Plans that Work for Multiple Environments

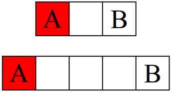
Toby Hu (University of Toronto, Canada)
Giuseppe De Giacomo (Sapienza Università di Roma, Italy)

Presentation at IJCAI'11, Barcelona, Spain
July 20, 2011

Classical Planning

Classical planning: Given a deterministic dynamic system with a single known initial state and a goal condition, find a plan that achieves the goal.

For example, starting from the leftmost cell A of a linear grid with left and right move actions, act to be in the rightmost cell B.

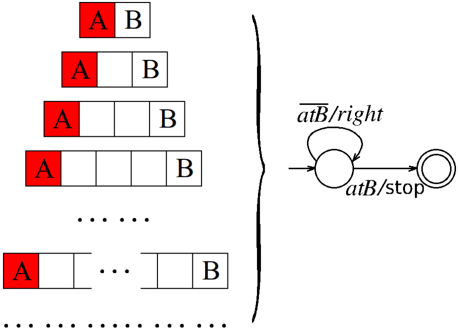


right, right

right, right, right, right

Generalized Planning

Generalized planning: Find a plan that achieves the goal for a number of "similar" planning problems.



Motivation

Existing work:

- [Srivastava, Immerman and Zilberstein 2008; 2011]
  Learning generalized plans using abstract counting.
- [Bonet, Palacios and Geffner 2009]
  Automatic derivation of memoryless policies and fnite-state controllers using classical planners.
- [Hu and Levesque 2010]
  A correctness result for reasoning about one-dimensional planning problems.
- ...

Question: Is there a general definition of generalized planning independent of any specific representation formalism?

Definitions

Two aspects need to be considered for planning:

1. The **agent** $A = \langle Acts, Obs \rangle$ that executes the plan;

   A plan is then a (prefix-closed) partial function $p : Obs^* \to Acts \cup \{\texttt{stop}\}$, often in a less general, finite-state form.

   Plans are independent of the environment.

Definitions

Two aspects need to be considered for planning:

1. The **agent** $A = \langle Acts, Obs \rangle$ that executes the plan;

   A plan is then a (prefix-closed) partial function $p : Obs^* \rightarrow Acts \cup \{\texttt{stop}\}$, often in a less general, finite-state form.

   Plans are independent of the environment.

2. The **environment** $E = \langle Events, S, s_0, \delta \rangle$ in which the plan is executed.

   A trace is a sequence $t = s_0 e_1 s_1 e_2 \cdots e_n s_n$, where
   - s_0 is the initial state, and
   - $s_i = \delta(s_{i-1}, e_i)$.

   A goal is a specification of desired traces in E. For now: $last(t) \in G \subseteq S$.

   Goals are independent of the agent.

Definitions

To relate agent $A = \langle Acts, Obs \rangle$ and environment $E = \langle Events, S, s_0, \delta \rangle$:

- an observation function $obs : S \to Obs$,
- an execution function $exec : Acts \to Events$.

Definitions

To relate agent $A = \langle Acts, Obs \rangle$ and environment $E = \langle Events, S, s_0, \delta \rangle$:

- an observation function $obs : S \to Obs$,
- an execution function $exec : Acts \to Events$.

A run r of plan p is the trace $trace(p) = s_0 e_1 s_1 e_2 \cdots e_n s_n$, such that

$$\begin{aligned} e_1 &= exec\big(p(obs(s_0))\big), \\ e_2 &= exec\big(p(obs(s_0), obs(s_1))\big), \\ &\cdots \quad \cdots \\ e_i &= exec\big(p(obs(s_0), \cdots, obs(s_{i-1}))\big). \end{aligned}$$

r is complete if $p(obs(s_0), \cdots, obs(s_n)) = \texttt{stop}$.

Definitions

A **basic planning problem** is a tuple

$$P = \langle \mathit{Acts}, \mathit{Obs}, \mathit{Events}, S, s_0, \delta, G, \mathit{obs}, \mathit{exec} \rangle.$$

A plan p is a solution to P iff $r = \mathit{trace}(p)$ is a complete run, and the final state $\mathit{last}(r) \in G$.

Definitions

A **basic planning problem** is a tuple

$$P = \langle Acts, Obs, Events, S, s_0, \delta, G, obs, exec \rangle.$$

A plan p is a solution to P iff $r = trace(p)$ is a complete run, and the final state $last(r) \in G$.

Definitions

A **basic planning problem** is a tuple

$$P = \langle Acts, Obs, Events, S, s_0, \delta, G, obs, exec\rangle.$$

A plan p is a solution to P iff $r = trace(p)$ is a complete run, and the final state $last(r) \in G$.

A **generalized planning problem** $\mathcal{P} = \{P_1, P_2, \cdots\}$ is a possibly infinite set of basic planning problems that share the same agent, *i.e.*,

$$P_i = \langle Acts, Obs, Events_i, S_i, s_{i0}, \delta_i, G_i, obs_i, exec_i\rangle.$$

A plan p is a solution to $\mathcal{P}$ iff p is a solution for all $P_i \in \mathcal{P}$.

The Linear Grid Example

Now we can capture the linear grid example as a generalized planning problem

$$\mathcal{P} = \{P_2, P_3, \cdots\},$$

where each $P_i = \langle Acts, Obs, Events_i, S_i, s_{i0}, \delta_i, G_i, obs_i, exec_i \rangle$ is a basic planning problem with i cells

- modeled completely independently
  (*e.g.*, using one proposition per celll or using a planning parameter),
- except that all P_i share the same *Acts* and *Obs*.

Standard Planning

Classical Planning

- Generalized planning with singleton set $\mathcal{P} = \{P\}$.
- The plan does not use the history of observations, except for its size: $p(o_1, \cdots, o_n) = p(n)$. (Sequential plan.)
- PSPACE-complete assuming a logarithmic encoding of the states (Bylander 1994).

Conditional Planning

- Generalized planning with finite set $\mathcal{P} = \{P_1, \cdots, P_N\}$, where $P_i = \langle Acts, Obs, Events, S, s_{i0}, \delta, G, obs, exec \rangle$ are identical except for s_{i0}.
- EXPSPACE-complete assuming a logarithmic encoding of the states (Haslum and Jonsson 1999; Rintanen 2004).

Conformant Planning

- Same as conditional, except $Obs = \{nil\}$ and $obs(s) = nil$ for all s.

Generalized Planning: Finite Case

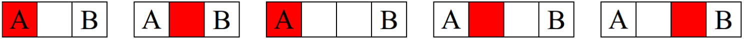
In general, for any finite set $\mathcal{P} = \{P_1, \cdots, P_N\}$, a plan can always be found if one exists.

How? Compile into a classical planning problem $P_{\mathcal{P}}$ with goal $G_{\mathcal{P}}$:
(similar in spirit to [Palacios and Geffner 2009] and [Albore *et al.* 2009])

- the actions are N-vectors of the action alphabet plus no-op;
- the state space is the cross-product of the N state spaces, augmented by
  - an equivalence relation
    (to remember for which problems identical actions must be prescribed),
  - an achievement set
    (to remember for which problems has the goal been achieved);
- the goal is to make every problem in the achievement set;
- no observation;
- the initial state, transition functions and other parts constructed accordingly.

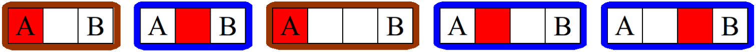
Generalized Planning: Finite Case

An Example



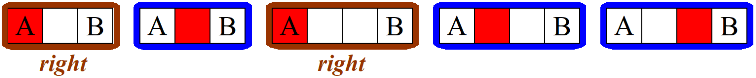
Generalized Planning: Finite Case

An Example



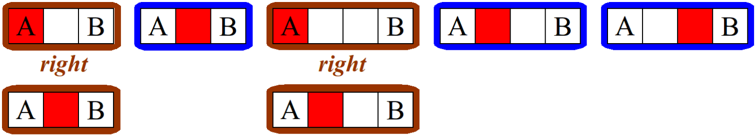
Generalized Planning: Finite Case

An Example



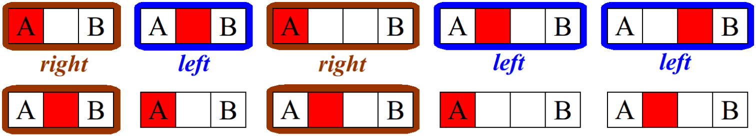
Generalized Planning: Finite Case

An Example



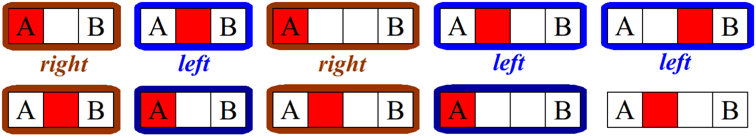
Generalized Planning: Finite Case

An Example



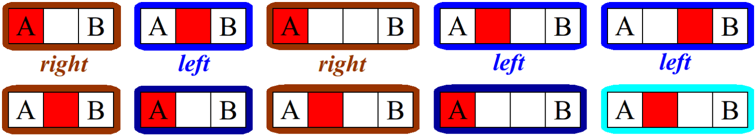
Generalized Planning: Finite Case

An Example



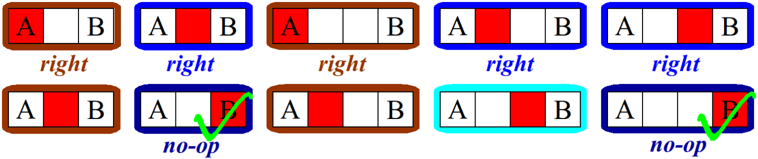
Generalized Planning: Finite Case

An Example



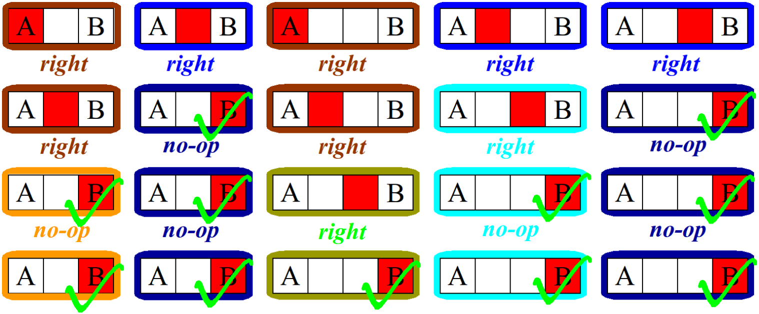
Generalized Planning: Finite Case

An Example

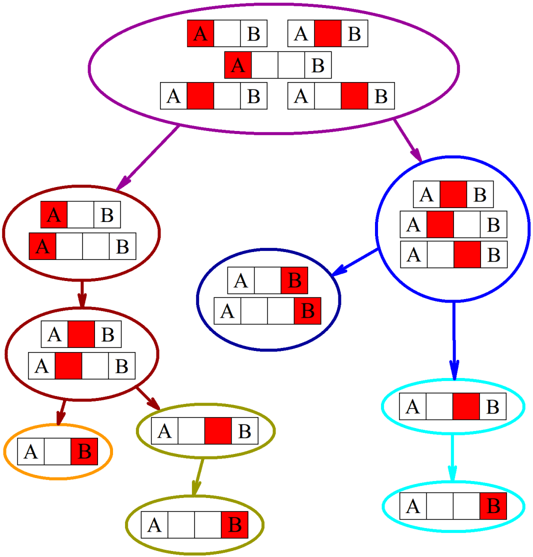


Generalized Planning: Finite Case

An Example



Generalized Planning: Finite Case



Generalized Planning: Finite Case

Theorem (Soundness and Completeness)

A generalized plan for G in the problem set $\mathcal{P}$ exists iff a classical plan exists for $G_{\mathcal{P}}$ in $P_{\mathcal{P}}$.

Theorem (Complexity)

Generalized planning for finite environments is PSPACE-complete (EXPSPACE-complete assuming a logarithmic encoding of the states).

Generalized Planning: Infinite Case

Existing approaches to generalized planning are essentially planning for infinite environment sets.

To model in our framework

- [Bonet *et al.* 2009]: Create a basic planning problem for each of their problem instances.
- [Hu and Levesque 2010]: Create a basic planning problem for each interpretation of their basic action theory.
- [Srivastava *et al.* 2008]: Create multiple basic planning problems for each of their problem instance, simulating all possible concretization of the abstract actions.

Note that all three approaches use a finite-state plan representation.

Finite Verifiability for One-Dimensional Problems

A reformulation (assuming no sequence functions) of the existing result in [Hu and Levesque 2010]:

A generalized planning problem $\mathcal{P} = \{P_0, P_1, P_2, \cdots\}$ is one-dimensional (simplified), or 1ds, if all $P_i = \langle Acts, Obs, Events, S_i, s_{i0}, \delta_i, G, obs_i, exec\rangle$ satisfy

- $S_i = \{\langle n, \vec{b}\rangle \mid n \in \{0, \cdots, i\}, \vec{b} \in \{\mathrm{true}, \mathrm{false}\}^m\}$;
- $s_{i0} = \langle i, \vec{b}_0\rangle$;
- if $\delta_i(\langle n, \vec{b}\rangle, e) = \langle n - d, \vec{b}'\rangle$ for $n > 0$, then $d \in \{0, 1\}$; furthermore, for all $n' > 0$, $\delta_i(\langle n', \vec{b}\rangle, e) = \langle n' - d, \vec{b}'\rangle$;
- $G \subseteq \{0\} \times \{\mathrm{true}, \mathrm{false}\}^m$;
- $obs_i(\langle n_1, \vec{b}\rangle) = obs_i(\langle n_2, \vec{b}\rangle)$ for all $n_1, n_2 > 0$.

Theorem

Given a 1ds problem $\mathcal{P} = \{P_0, P_1, \cdots\}$ that shares a final-state goal G, and a finite-state plan p with l states, if p is a plan for all of $\{P_0, P_1, \cdots, P_N\}$, where $N = l \cdot 2^m + 1$, then p is a plan for $\mathcal{P}$.

New Insights into One-Dimensional Problems

Is it possible that there exists a generalized plan but no finite-state plan?

Theorem

If a 1ds generalized planning problem has a generalized plan, then it has a finite-state plan.

This justifies the use of finite-state plans, and makes planning for 1ds problems at least recursively enumerable. Even better:

Theorem

A 1ds generalized planning problem $\mathcal{P} = \{P_0, P_1, \cdots\}$ *has a plan, if and only if the finite subset* $\mathcal{P}' = \{P_0, P_1, \cdots, P_N\}$ *has a plan where* $N = 2^m$.

So we can use, *e.g.*, compilation to classical planning, to solve 1ds problems.

Theorem

Generalized planning for 1ds problems is EXPSPACE.

Summary

Contributions

- Proposed a framework to represent and analyze generalized planning; (capturing existing formalisms of standard and generalized planning)
- Proved that generalized planning for finite sets is EXPSPACE- complete;
- Showed 1ds problem is EXPSPACE, and solvable by planning for finite sets.

Future Work

- Use the framework to identify more general problem classes with interesting correctness and computational properties (*e.g.*, full 1d and more);
- Consider more general goals (*e.g.*, temporally-extended and long-running goals);
- Consider unrelated concrete actions with shared abstract actions.