AI Planning

20. Symmetry Reduction

How to Not Try "the Same Thing" Over and Over Again

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Agenda

Motivation

→ **Optimal Planning**: Admissible heuristics estimate distance to goal $h(s)$ and avoid the expansion of nodes whose $g(s) + h(s) > f^*(I)$

→ They are a very effective technique but not the only one!

→ Sometimes, even almost perfect heuristics are not good enough![Helmert and Röger (2008)]

Definition (Almost Perfect Heuristic). A heuristic is almost perfect if it differs from the perfect heuristic h^* only by an additive constant: $(h^* - c)(s) = max(h^*(s) - c, 0)$.

Definition (Search effort $N^c(\Pi)$). Let Π be a planning task. We denote $N^c(\Pi)$ to the number of states s with $g(s) + (h^* - c)(s) < h^*(\Pi)$.

→A*with $h^* - c$ will expand at least $N^c(\Pi)$ nodes.

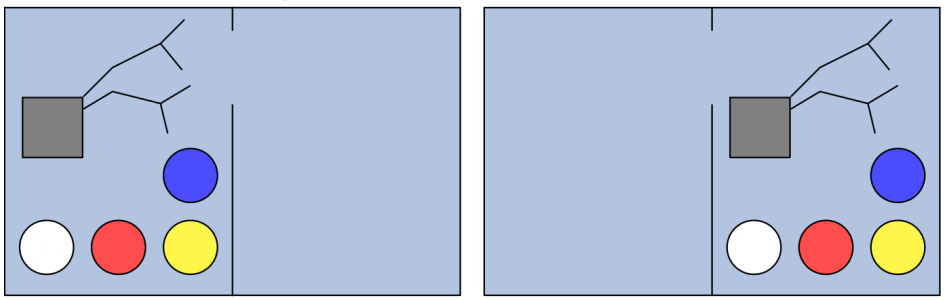
Theorem. There exist families of planning tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ where $\mathcal{T}_i$ is a planning task of size i and $N^c(\mathcal{T}_i)$ grows exponentially in i even for small c.

Proof. Gripper (see next slide), Miconic, Blocksworld, ...

Motivation: Example Gripper

Gripper:

$\mathcal{T}_n$ carry n balls from L to R.



initial state goal state

Reachable states: $S_n =$

Theorem. *Let $n \in \mathbb{N}_0$ with $n \geq 3$. If n is even, then $N^1(T_n) =$. If n is odd then $N^1(T_n) =$.*

Proof sketch.

Pruning Methods

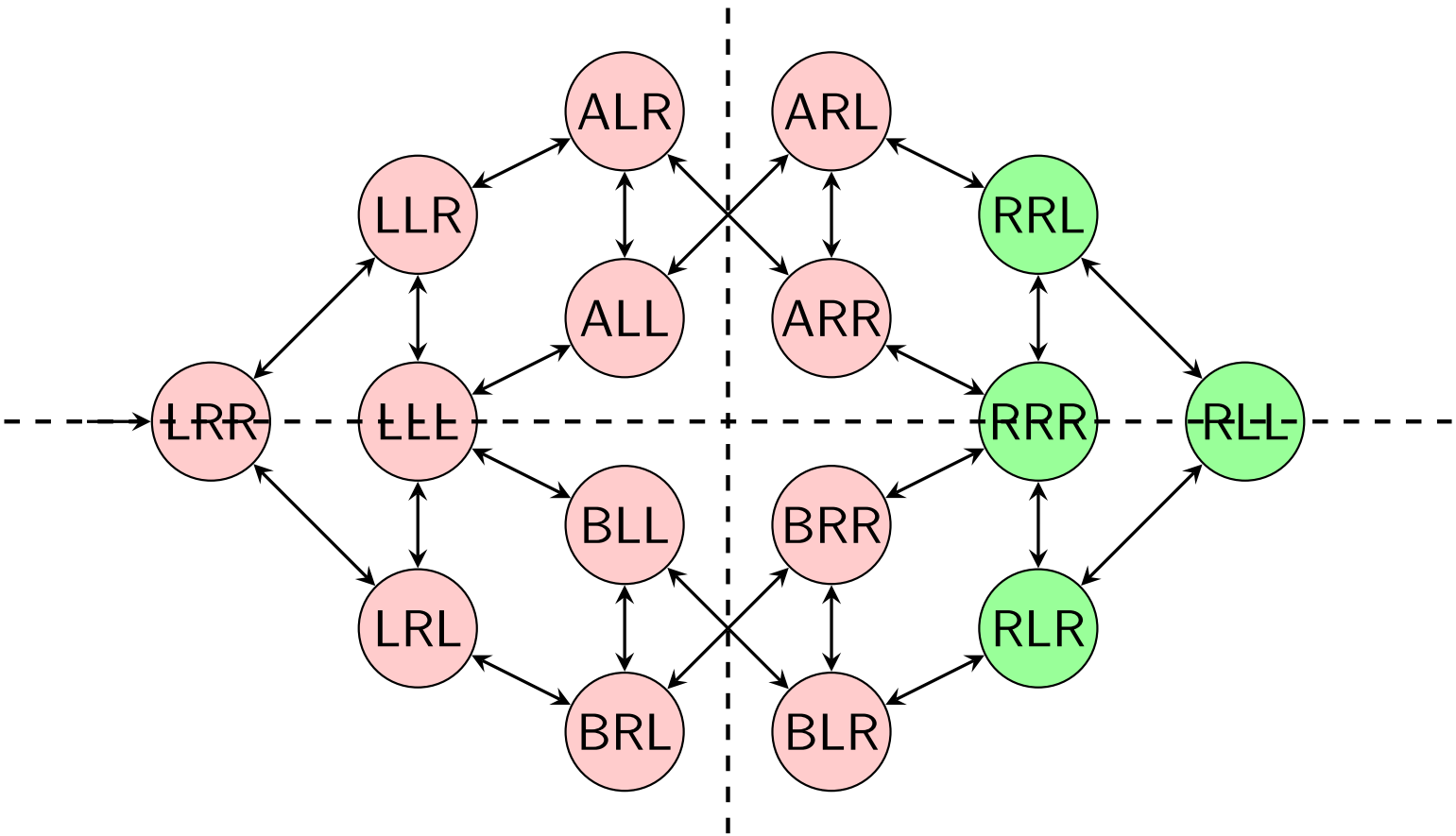
→ To the rescue: pruning methods

1. State Pruning: Reduces the search effort by not checking parts of the search space. Particular nodes are pruned.
2. Action Pruning: Reduces the search effort by considering only some of the applicable actions. Particular edges are pruned.

We cover 3 different methods for pruning:

- (State) Symmetry reduction: → **This Chapter**
- (State) Dominance pruning: → **Chapter 19**
- (Action) Partial-order reduction.→ **Chapter 18**

Symmetries in a Nutshell: Example



Symmetries in a Nutshell: Wrap-Up

Basic Algorithm

1. *Pre-search:* Find Symmetries
   - Find (some) generators of the automorphism group that fixes goal
   - Plug them into an *effective* symmetry detection black box
2. *Search:* Use Symmetries
   - Run A^*. When a successor node s' is generated, check if a symmetrical node s was already found. If *yes*, then
     (a) Update $g(s) := \min\{g(s), g(s')\}$
     (b) Prune s'
3. *Post-search:*
   - Extract plan going backwards from the goal using symmetries

Our Agenda for This Chapter

1. **Symmetry Basics:** Formal definition of symmetries and their associated structures; proving their basic properties.
2. **Finding Symmetries:** We take a look at algorithms that detect symmetric states from the definition of the planning task.
3. **Exploiting Symmetries:** We discuss how symmetries can be used during the search and how the solution plan can be extracted afterwards.

What Is A Symmetry?

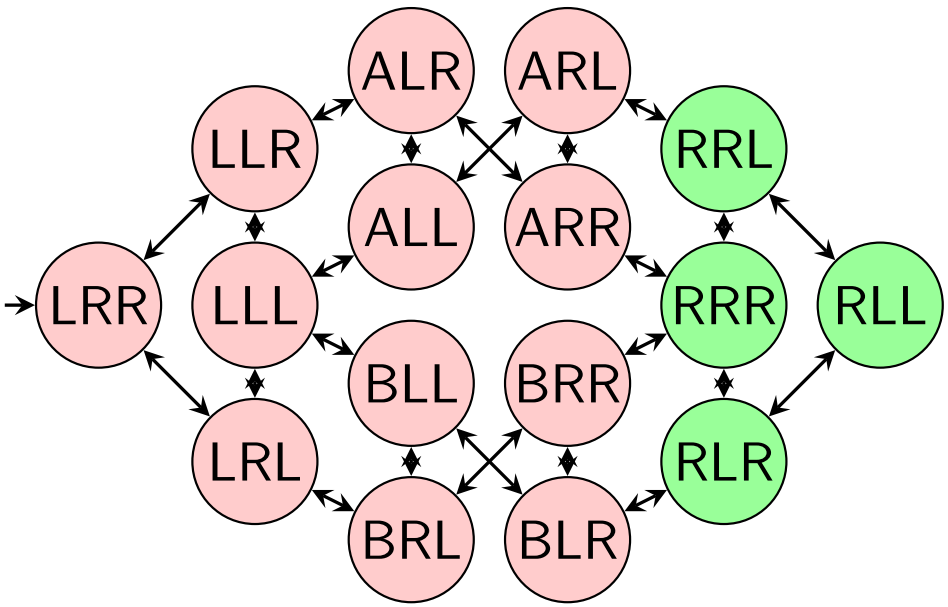
What is a symmetry? In abstract terms, an object is symmetric with respect to an operation if the operation preserves some property of the object. Here, we will consider automorphisms of the state space.

Definition (Automorphism). *An automorphism of a graph (S, E) is a permutation $\sigma : S \to S$ of the vertices of the graph that preserves the structure of the graph, i.e., for every two vertices $s_1, s_2 \in S$, we have that $(s_1, s_2) \in E$ iff $(\sigma(s_1), \sigma(s_2)) \in E$.*

$\to$ Since transitions in our state space have a cost, we require the action costs to be the same.

Definition (Automorphism in Θ). *An automorphism of a state space Θ is a permutation $\sigma : S \to S$ of the states that preserves the structure of Θ, i.e., for every two states $s_1, s_2 \in S$, we have that $s_1 \xrightarrow{a} s_2 \in T$ iff $\sigma(s_1) \xrightarrow{a'} \sigma(s_2) \in T$ with c(a) = c(a').*

Example of Automorphisms



Question!

Which of the following permutations are automorphisms?

(A): ARR-BRR (B): A-B

(C): XYZ-XZY (D): L-R

Definitions: Symmetry Group and Orbit

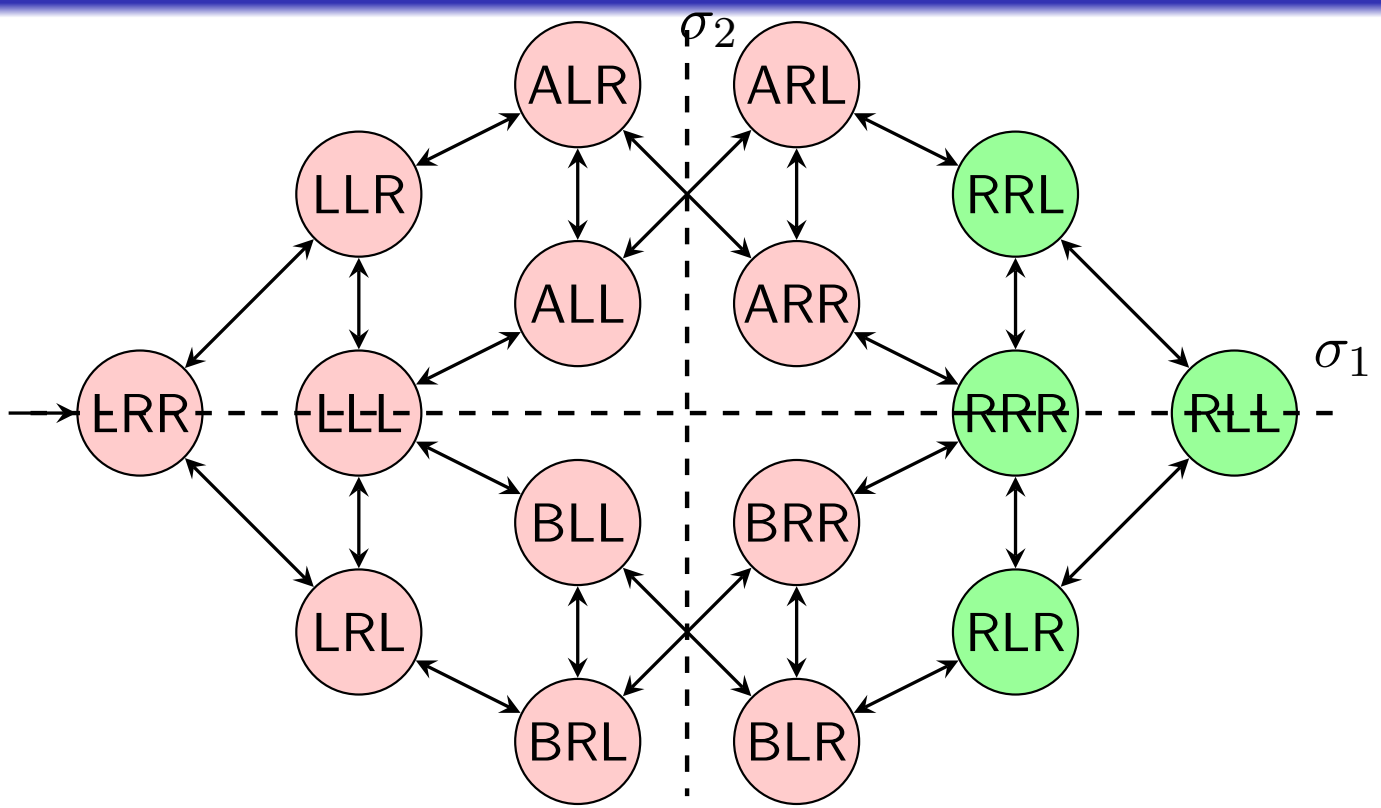
Definition (Symmetry Group). *A set of automorphisms $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ generates a group G, where G is the set of all permutations that can be obtained by composing elements of Σ.*

Definition (Orbit). *The orbit of a vertex s with respect to a group G is denoted by $G(s)$ and is simply the set of vertices to which elements in G map s.*

$$G(s) = \{\sigma(s) \mid \sigma \in G\}$$

Definition (Symmetric states).
Let G be a symmetry group of the state space of a planning task. Then, we say that s, t are symmetric, $s \sim_G t$ iff they belong to the same orbit $G(s) = G(t)$.

Example: Orbits



Question!

How many orbits are in the group generated by $\{\sigma_1, \sigma_2\}$?

(A): 4 (B): 5

(C): 8 (D): 10

What do we want to preserve?

Automorphisms characterize the transitions of our state space. However, they do not take into account the initial state and goals of the problem.

What we want is to consider symmetric states equivalent so that if we have considered a state in the search, we can skip all its symmetric states. In forward search we need to preserve goal distance: [1]

$$s \sim_G t \implies h^*(s) = h^*(t)$$

$\rightarrow$ Symmetry methods guarantee that for every plan valid for s, π, there exist an equivalent plan for t, $\sigma(\pi)$ and vice versa.

[1] In domains with 0-cost actions we also need to take into account the optimal number of 0-cost actions.

Stabilizing the symmetries

Definition (Stabilizer).
Let G be a symmetry group. The point-stabilizer of a vertex s with respect to G, denoted G_s is a subgroup of G that contains all the permutations that fix s. $G_s = \{\sigma \mid \sigma \in G, \sigma(s) = s\}$.

The set-stabilizer of a set of vertices S with respect to G, denoted G_S is a subgroup of G that contains all the permutations that fix S.
$G_S = \{\sigma \mid \sigma \in G$ *s.t.* $\sigma(s) = t$ *for all* $s, t \in S\}$.

$\rightarrow$We are interested in stabilizing the goal! We call G_G a goal-stabilizer group if is the subgroup that stabilizes the set of goal states.

$\rightarrow$ Goal-stabilizer groups preserve h^*.

Goal Stabilizer preserves h^*

Proposition (Goal Stabilizer preserves h^*). *Let Π be an FDR planning task and G a goal-stabilizer group of Θ_Π. Then, $s \sim_G t \implies h^*(s) = h^*(t)$.*

Proof. Let s, t be symmetric states, i.e., $s \sim_G t$. We show that for every plan for s (t), π, exists a symmetric plan for t (s) of the same cost.

By induction on the length of the plan, $|\pi|$. Base case, $|\pi| = 0$. Then s is a goal state and since G is goal-stabilizer, t is a goal state.

Inductive case. $|\pi| = n > 0$. Let a be the first action in π. Then, $s' = a(s)$ is the next state in the plan for s. Since s and t are symmetric, there exists $\sigma(a)$ such that $t \xrightarrow{\sigma(a)} t'$ where $t' \sim s'$. By induction, $h^*(s') = h^*(t')$. Since a and $\sigma(a)$ are symmetric, $c(a) = c(\sigma(a))$.

How to Find Symmetries?

Symmetry groups are characterized as automorphisms (i.e., permutations of states in our state space). However:

1. How to find a set of automorphisms?
   $\rightarrow$ look for structural symmetries in the problem description.
2. How to succinctly represent them?
   $\rightarrow$ permutations of facts (like in the example of slide 12).

Procedure:

1. Define Problem Description Graph, a graph that represents the problem.
2. Use state-of-the-art algorithms to find automorphisms in that graph.

Structural Symmetry

Definition (Structural Symmetry). *Let $\Pi = (V, A, I, G)$ be an FDR planning task. A permutation σ on $V \cup A \cup (\cup_{v \in V} D_v)$ is a structural symmetry if*

- $\sigma(V) = V$
- $\sigma(A) = A$, and for all $a \in A$:
  - $pre(\sigma(a)) = \sigma(pre(a))$
  - $\mathit{eff}(\sigma(a)) = \sigma(\mathit{eff}(a))$
  - $c(\sigma(a)) = c(a)$
- $\sigma(G) = G$

Theorem (Structural symmetries are goal-stabilizer). *Let Π be an FDR planning task and $\sigma_1, \ldots, \sigma_k$ structural symmetries of Π. Then $\{\sigma_1, \ldots, \sigma_k\}$ is a goal-stabilizer group of Θ_Π.*

Proof Intuition Since $\sigma(V) = V$; σ is a permutation ($\sigma(s)$ is a state).
Since $\sigma(A) = A$ (and pre, eff, c); σ is an automorphism of Π
Since $\sigma(G) = G$; σ is a goal-stabilizer subgroup.

Problem Description Graph

Definition (Problem Description Graph). *Let $\Pi = (V, A, I, G)$ be an FDR planning task. The problem description graph of Π is the undirected colored graph $PDG(\Pi)$ with four types of vertices colored of different colors:*

1. *one for each variable $N_v, v \in V$,*
2. *one for each value of a variable $N_d, d \in D_v$, and*
3. *two for each action corresponding to its preconditions and effects: N_{pre_a} and $N_{eff_a}, a \in A$.*

There is an arc between:

1. *N_v and N_d iff $d \in D_v$,*
2. *N_{pre_a} and N_{eff_a} for every $a \in A$, and*
3. *d and pre_a or eff_a iff $d \in pre_a$ or $d \in eff_a$.*

Problem Description Graph: Example

Using the PDG

Definition (State-permutation induced by PDG-permutation). *Let $\Pi = (V, A, I, G)$ be an FDR planning task. Let π be a permutation of $PDG(\Pi)$. The induced permutation on the state space of the planning task, $\pi' : S \to S$ is defined as $\pi'(s) = \{\pi(v), \pi(d) \mid \langle v, d \rangle \in s\}$.*

Theorem (PDG automorphisms to state space automorphisms). *Let Π be an FDR planning task. Let σ be an automorphism of $PDG(\Pi)$ and let σ' be the state-permutation induced by σ. Then σ' is a goal-stabilizer automorphism of the state space of Θ_Π.*

Proof Intuition *The PDG-permutation renames operators, variables and values in a way that preserves the semantics.*

Finding automorphisms in the PDG

- On the bad side – "Is the graph automorphism problem in P, or NP, or neither?" Is an open problem in CS. It is harder than graph isomorphism which is also not known to be in P.
- On the good side We compute the automorphisms only once before starting the search – We already in **PSPACE**, so how bad can it be?
- Open-source software tools that are available for this task:
  1. SAUCY – H. Katebi, K. A. Sakallah & I. L. Markov (2012)
  2. BLISS[2] – T. Junttila & P. Kaski (2011)
  3. NAUTY[2] – B. D. McKay & A. Piperno (2013)

[2] Also produce Canonical Labeling

Exploiting Symmetries: Basic Idea

Basic Idea

1. *Pre-search:* Find Symmetries
   Find (some) generators of the automorphism group that fixes goal
2. *Search:* Use Symmetries
   Run heuristic search. When a successor node s' is generated, check if a symmetrical node s was already found. If *yes*, then
   (a) If $g(s) > g(s')$, then update g, parent, and achieving action of s to those of s' and reopen (s)
   (b) Prune s'
3. *Post-search:*
   Non-standard plan extraction

Finding Symmetric States

Given state s, is s symmetric to another previously known state?

Problem 1. Finding symmetric states is intractable. Given a pair of states s, t and a symmetry group $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ decide whether $G(s) = G(t)$ is NP-hard.

Problem 2. Comparing s against all previously known states may be very expensive.

$\rightarrow$ Each orbit is implicitly represented by one of its states, called the canonical state, e.g., the lexicographically smallest state in the orbit. Whenever we generate s, replace it for the representative of its orbit. Duplicate elimination takes care of the rest.

$\rightarrow$ Given s, finding the canonical state of $G(s)$ is still NP-hard (see Problem 1)!

$\rightarrow$ We approximate! Use a greedy algorithm to find a lexicographically smaller state, but not necessarily the smallest.

Representative states

function FindRepresentativeGreedy ($G = \sigma_1, \ldots, \sigma_k$, *state* s)
  while $\exists \sigma \in G$, s.t., $\sigma(s) < s$ **do**
    $s \leftarrow \sigma(s)$
  return s

FindRepresentativeGreedy induces G_C, a new subgroup of G.

The orbit $G_C(s)$ is formed by all states s' s.t. *FindRepresentativeGreedy*$(G, s') = s$.

The equivalence relation is defined as $s \sim_{G_C} t$ iff $G_C(s) = G_C(t)$.

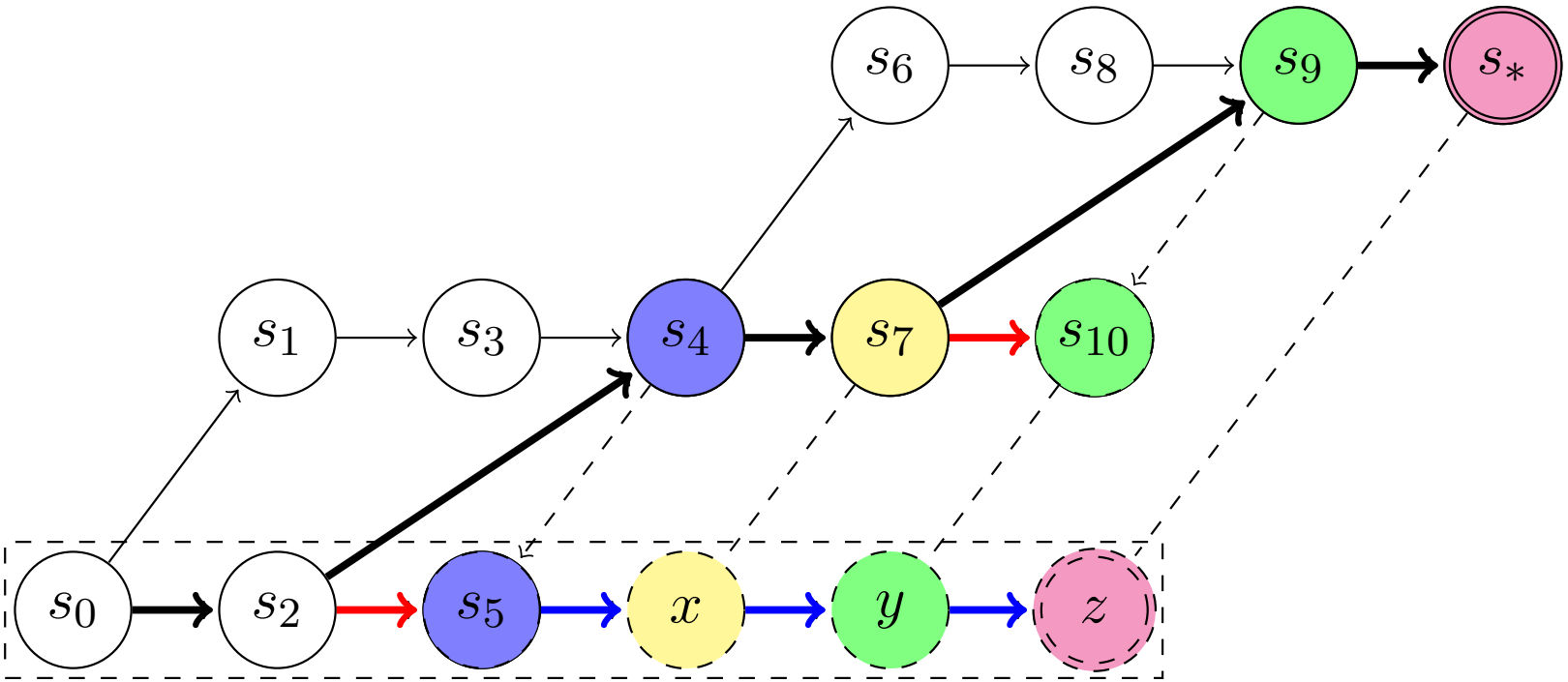
- $s \sim_{G_C} t \implies s \sim_G t$ so the approximation is still safe.
- $s \sim_{G_C} t \not\Longrightarrow s \sim_C t$ so the approximation is less powerful.

A* with Symmetries

1. *Search:* Run A*. When a successor node s' is generated, check if a symmetrical node s was already found (by checking if FindRepresentativeGreedy(s') is a duplicate). If *yes*, then
   (a) If $g(s) > g(s')$, then update g, parent, and achieving action of s to those of s' and reopen (s)
   (b) Prune s'
2. *Post-search:* Non-standard plan extraction

- s and s' are symmetric. Should we prune s or s'?
  $\rightarrow$ We prune s' because s might have already been expanded so we cannot easily prune it anymore.
- Is it possible that $g(s) > g(s')$? $\rightarrow$ Yes, $h^*(s) = h^*(s')$ but $g^*(s)$ and $g^*(s')$ may differ as well as $h(s)$ and $h(s')$.
- The new values of g, parent and achieving action of s are not "true", but we know that they are for some symmetric state s' and that suffices for plan extraction (see next slides)

A* with Symmetries: Example



Solution Reconstruction with Symmetries

The plan returned by $\mathrm{A}^*$with symmetries is not valid, but a valid plan can be reconstructed by "undoing" the symmetries.

function Trace-forward $(\pi = \langle (\epsilon, s_0), (a_1, s_1), \ldots, (a_m, a_m) \rangle)$
  let $\sigma_i \in \Gamma_{S_\star}$ be such that $\sigma_i(s_i) = s_{i-1}[a_i]$ for $0 < i \leq m$
  $\sigma := \sigma_{id}$
  $\rho := \langle \epsilon \rangle$
  for $i := 1$ **to** m **do**:
    $s := \sigma(s_{i-1})$, $\sigma := \sigma \circ \sigma_i$, $s' := \sigma(s_i)$
    append to ρ a cheapest action a such that $s[a] = s'$
  endfor
  return ρ

Solution Reconstruction with Symmetries: Example

$\rightarrow$ The figure shows a search tree after the goal has been found. All nodes that were pruned due to symmetries are omitted. Dashed edges represent the transitions that were substituted because a better path was found to a symmetric state. For example, $HHH00 \xrightarrow{t_{2,1}} HTH10$ is substituted by $HHH00 \xrightarrow{t_{1,1}} THH10$ because $HTH10$ and $THH10$ are detected as symmetric and the second has lower cost[3].

[3] In the example, $g(HTH10) = g(THH10)$ but we assume that the substitution has occurred anyway in order to show how the solution reconstruction works.

Summary

- A symmetry σ is an automorphism, i.e., a permutation of the states that preserves the structure (goal-distance) of the state space.
- A set of permutations $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ defines a group G, as is the set of all permutations that can be obtained by composing elements in Σ. The orbit of an state s, $G(s)$ is the set of states s' such that exists $\sigma \in G, \sigma(s) = s'$.
- A group G defines an equivalence class. s and t are symmetric according to G, $s \sim_G t$, iff $G(s) = G(t)$.
- In forward search we are interested in groups that stabilize the goal, i.e., goal states are only symmetric to other goal states.
- Symmetries can be obtained by computing the automorphisms of the Problem Description Graph (PDG). Those are structural symmetries of the domain and can be succinctly represented.

Remarks

Symmetries are also useful for satisficing planning.[Domshlak *et al.* (2013)]

Relation between symmetries and heuristics.[Shleyfman *et al.* (2015)]

Definition (Invariant Under Structural Symmetries). *A heuristic h is invariant under structural symmetries if $h(s) = h(t)$ for every $s \sim t$.*

- If a heuristic is invariant under symmetries we do not need to evaluate more than one state in the same orbit.
- If a heuristic is not invariant under symmetries, it is admissible to evaluate several states from the same orbit and take the maximum.

The coarsest bisimulation captures all symmetries.[Sievers *et al.* (2015)] Therefore, we can use bisimulation in M&S to capture all local symmetries to the variables that have been already merged.

Heuristics Invariant Under Structural Symmetries

Classification of heuristics ($\mathbb{E}$ stands for "with a tie-breaking invariant under symmetries"):

Non-symmetric

h^{FF} Hoffmann & Nebel

$h^{\text{FF}}/h^{\text{add}}$, $h^{\text{FF}}/h^{\text{max}}$ Keyder & Geffner

h^{PDB} Culberson & Schaeffer, Edelkamp

$h^{M\&S}$ Helmert, Haslum & Hoffmann

$h^{\text{LM-cut}}$ Helmert & Domshlak

Symmetric

h^{+} Hoffmann & Nebel

h^{max} Bonet & Geffner

h^{add} Bonet & Geffner

$\mathbb{E}h^{\text{FF}}$ Hoffmann & Nebel

h^{m} Haslum & Geffner

$\mathbb{E}h^{\text{LM-cut}}$ Helmert & Domshlak

Remarks: Orbit Search

In practice, one can just substitute each state by its canonical representative. Then, the pruning is just performed by standard duplicate elimination. We call this orbit search because each search node corresponds to a symmetry orbit.

Orbit Search Algorithm

1. *Pre-search:* Find Symmetries

   Find (some) generators of the automorphism group that fixes goal

2. *Search:* Use Symmetries

   Run heuristic search. When a successor node s' is generated replace s' by *FindRepresentativeGreedy*(G, s'). If a duplicate its found update g, parent, and achieving action as usual.

3. *Post-search:*

   Non-standard plan extraction

Reading

- *Exploiting Problem Symmetries in State-Based Planners* [Pochter *et al.* (2011)].

  Available at:

  `http://www.cs.hujcdi.ac.il/~avivz/pubs/12/PlanningSymmetries_AAAI11.pdf`

  Content: Introduces of symmetries as automorphisms of the state space in planning and the method based on the problem description graph.

- *Enhanced Symmetry Breaking in Cost-Optimal Planning as Forward Search* [Domshlak *et al.* (2012)].

  Available at:

  `http://iew3.technion.ac.il/~dcarmel/Papers/Sources/icaps12b.pdf`

  Content: Improvement over the work by Pochter *et al.*. Prove that it is not necessary to stabilize the symmetries with respect to the initial state if the Trace-Forward algorithm is used for solution reconstruction.

References I

Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Enhanced symmetry breaking in cost-optimal planning as forward search. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press, 2012.

Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Symmetry breaking: Satisficing planning and landmark heuristics. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy, 2013. AAAI Press.

Malte Helmert and Gabriele Röger. How good is almost perfect? In Dieter Fox and Carla Gomes, editors, *Proceedings of the 23rd National Conference of the American Association for Artificial Intelligence (AAAI'08)*, pages 944–949, Chicago, Illinois, USA, July 2008. AAAI Press.

References II

Nir Pochter, Aviv Zohar, and Jeffrey S. Rosenschein. Exploiting problem symmetries in state-based planners. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11)*, San Francisco, CA, USA, July 2011. AAAI Press.

Alexander Shleyfman, Michael Katz, Malte Helmert, Silvan Sievers, and Martin Wehrle. Heuristics and symmetries in classical planning. In Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 3371–3377. AAAI Press, January 2015.

Silvan Sievers, Martin Wehrle, Malte Helmert, Alexander Shleyfman, and Michael Katz. Factored symmetries for merge-and-shrink abstractions. In Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 3378–3385. AAAI Press, January 2015.