An Empirical Case Study on Symmetry Handling in Cost-Optimal Planning as Heuristic Search

Silvan Sievers[1] Martin Wehrle[1]
Malte Helmert[1] Michael Katz[2]

[1]University of Basel
Basel, Switzerland

[2]IBM Research
Haifa, Israel

September 23, 2015

Motivation

- Successful usage of symmetries:
  - Planning: duplicate pruning in $A^\star$, improved merge-and-shrink heuristics
  - Heuristic search: symmetrical/dual lookups

Motivation

- Successful usage of symmetries:
  - Planning: duplicate pruning in $A^\star$, improved merge-and-shrink heuristics
  - Heuristic search: symmetrical/dual lookups
- Contribution of this work:
  - Quantitative analysis of symmetries in planning benchmarks
  - Empirical comparison of different symmetry-based techniques (adapted to planning)

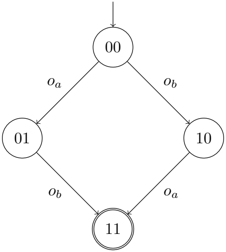
Outline

Classical Planning

- SAS^+ planning task Π:
  - Finite-domain state variables
  - Initial state: complete variable assignment
  - Goal description: partial variable assignment
  - Operators: preconditions, effects, cost

Classical Planning

- SAS^+ planning task Π:
  - Finite-domain state variables
  - Initial state: complete variable assignment
  - Goal description: partial variable assignment
  - Operators: preconditions, effects, cost
- State transition graph $\mathcal{T}_\Pi$:

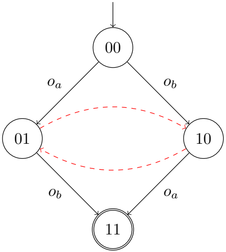


Structural Symmetries (Shleyfman et al. 2015)

- Structural symmetry of a planning task Π:
  - Maps facts (variable/value pairs) to facts and operators to operators
  - Induced symmetry σ on the state transition graph $\mathcal{T}_\Pi = (V, E)$ is a goal-stable automorphism:
    - $(s, o, s') \in E$ iff $(\sigma(s), \sigma(o), \sigma(s')) \in E$
    - s goal state iff $\sigma(s)$ goal state

Structural Symmetries (Shleyfman et al. 2015)

- Structural symmetry of a planning task Π:
  - Maps facts (variable/value pairs) to facts and operators to operators
  - Induced symmetry σ on the state transition graph $\mathcal{T}_\Pi = (V, E)$ is a goal-stable automorphism:
    - $(s, o, s') \in E$ iff $(\sigma(s), \sigma(o), \sigma(s')) \in E$
    - s goal state iff $\sigma(s)$ goal state
- Example symmetry:
  $\sigma(o_a) = o_b$
  $\sigma(o_b) = o_a$

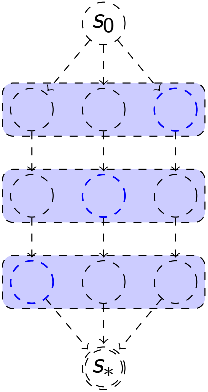


Orbit Space Search (Domshlak et al. 2015)

- Orbit: equivalence class of symmetrical states

Orbit Space Search (Domshlak et al. 2015)

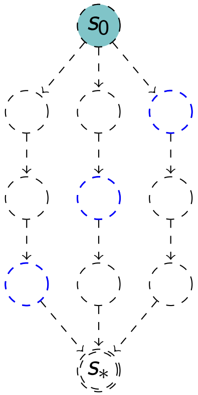
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

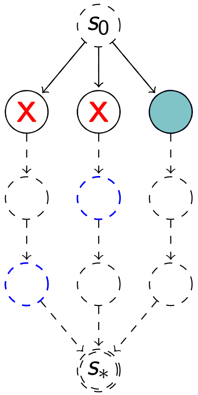
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run A* as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators
    $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

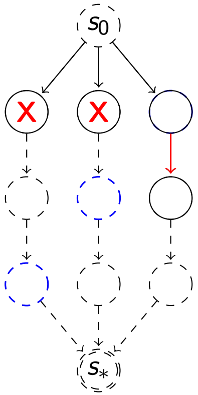
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
    - Run A* as usual
    - When expanding state s, replace successors by orbit representatives, but save regular operators
      $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

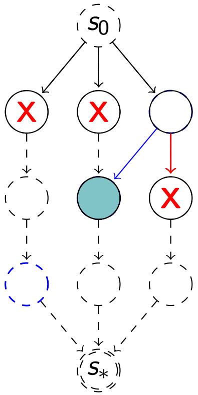
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run $A^\star$ as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators
    $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

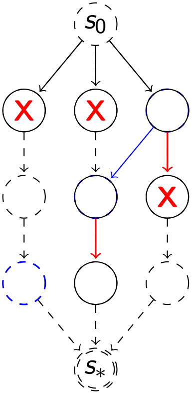
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run $A^\star$ as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators
    $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

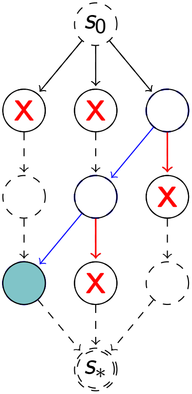
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
    - Run A* as usual
    - When expanding state s, replace successors by orbit representatives, but save regular operators
      $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

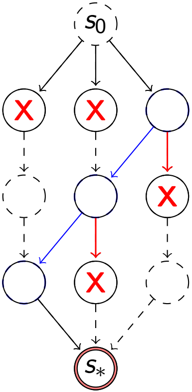
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run A* as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators
    $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

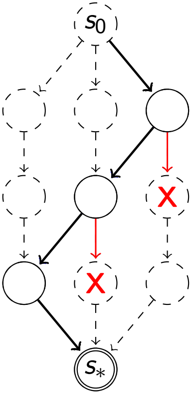
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run A* as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators
    $\rightarrow$ symmetrical duplicate pruning



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

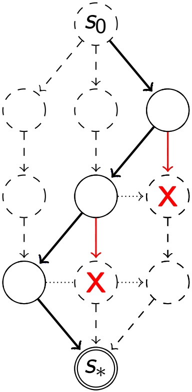
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run A* as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators
    $\rightarrow$ symmetrical duplicate pruning
- Non-standard plan extraction:
  - Compute the "real" state sequence
  - Find operators connecting the sequence



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

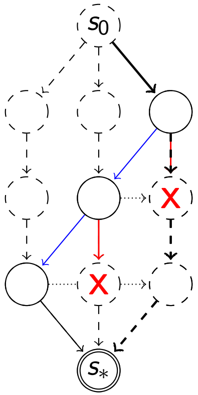
- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run A* as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators $\rightarrow$ symmetrical duplicate pruning
- Non-standard plan extraction:
  - Compute the "real" state sequence
  - Find operators connecting the sequence



Credits to A. Shleyfman

Orbit Space Search (Domshlak et al. 2015)

- Orbit: equivalence class of symmetrical states
- Before search: find (some) generators of the automorphism group
- During search:
  - Run $A^\star$ as usual
  - When expanding state s, replace successors by orbit representatives, but save regular operators $\rightarrow$ symmetrical duplicate pruning
- Non-standard plan extraction:
  - Compute the "real" state sequence
  - Find operators connecting the sequence



Credits to A. Shleyfman

Symmetrical Lookups for Planning

- (For heuristic search: Felner et al. 2005, Zahavi et al. 2008)
- Before search: find (some) generators of the automorphism group
- During search, for a given state s and heuristic h:
  - Compute (a subset of) the orbit containing s: $S := \{s, s^1, \ldots s^m\}$
  - Compute heuristic as $\bar{h}(s) := \max\{h(s') \mid s' \in S\}$
- Properties:
  - S can be chosen arbitrarily
  - $\bar{h}(s)$ is still admissible (if h is)

Bidirectional Pathmax for Planning

- (For heuristic search: Felner et al. 2011)
- Symmetrical lookups usually render heuristics inconsistent
- Consistency: $h(s) \leq cost(o) + h(s')$ for a transition from s to s' with operator o
- Bidirectional pathmax (BPMX) rule: $h(s') = \max(h(s'), h(s) - cost(o))$

Merge-and-Shrink Heuristic (Helmert et al. 2014)

- Represent state space as set $\mathcal{T}$ of small finite transition systems, with a shared label set L
- State space corresponds to product of transition systems
- Transform transition systems to obtain distance heuristic for state space

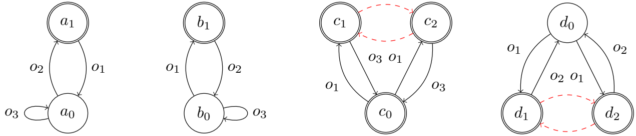
Factored Symmetries (Sievers et al. 2015)

- Work on a set $\mathcal{T}$ of transition systems as encountered during the merge-and-shrink computation
- Locally map abstract states to abstract states within elemets of $\mathcal{T}$ and globally map transition labels to transition labels in L
- Goal states must be preserved

Factored Symmetries (Sievers et al. 2015)

- Work on a set $\mathcal{T}$ of transition systems as encountered during the merge-and-shrink computation
- Locally map abstract states to abstract states within elemets of $\mathcal{T}$ and globally map transition labels to transition labels in L
- Goal states must be preserved
- Example:

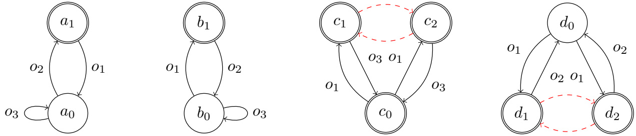
$$\sigma(o_1) = o_1$$
$$\sigma(o_2) = o_2$$
$$\sigma(o_3) = o_3$$



Factored Symmetries (Sievers et al. 2015)

- Work on a set $\mathcal{T}$ of transition systems as encountered during the merge-and-shrink computation
- Locally map abstract states to abstract states within elemets of $\mathcal{T}$ and globally map transition labels to transition labels in L
- Goal states must be preserved
- Example:

$$\sigma(o_1) = o_1$$
$$\sigma(o_2) = o_2$$
$$\sigma(o_3) = o_3$$



- Usage: improve merging strategies

Outline

Quantitative Analysis

- Benchmark set: 44 domains with 1396 tasks
- Amount of symmetries:
  - Only 3 domains with no symmetries
  - 1103 tasks contain symmetries
  - In 38 domains, more than 50% of tasks contain symmetries
  - In most of the 38 domains, almost all tasks contain symmetries
- Influence of the representation and the symmetry tool?

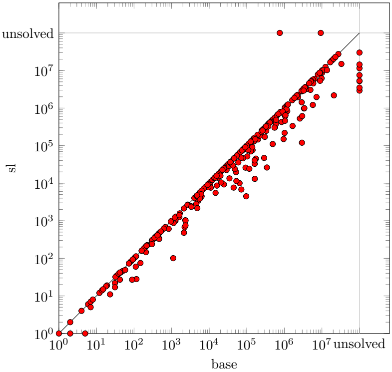
Symmetrical Lookups

Merge-and-Shrink	**base**	**1 state**	**5 states**	**10 states**	**orbit**
Coverage	652	656	**658**	**658**	**658**
Expansions sum	607602428	501671723	493848579	**471769190**	493848579

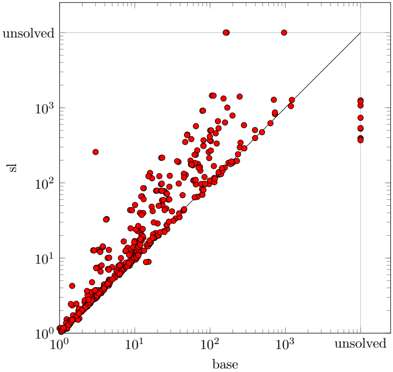
Symmetrical Lookups

Merge-and-Shrink	base	1 state	5 states	10 states	orbit
Coverage	652	656	**658**	**658**	**658**
Expansions sum	607602428	501671723	493848579	**471769190**	493848579

Expansions:



Runtime:



Bidirectional Pathmax

Merge-and-Shrink	base	sl	sl-bpmx
Coverage	652	**658**	**658**
Expansions sum	607602428	**471769190**	471769236

- Marginal reduction in expansions, no increase in coverage
- Explanation: pathmax corrections only in 2% of the tasks for which the merge-and-shrink heuristic was constructed

Combinations of Techniques

Merge-and-Shrink	base	oss	sl	fs
Coverage	652	696	658	654
Expansions sum	5.16e+8	2.68e+8	4.01e+8	3.65e+8

- All techniques improve performance

Combinations of Techniques

Merge-and-Shrink	base	oss	sl	fs
Coverage	652	696	658	654
Expansions sum	5.16e+8	2.68e+8	4.01e+8	3.65e+8

- All techniques improve performance

Merge-and-Shrink	oss-sl	oss-fs	sl-fs	all
Coverage	691	**698**	655	692
Expansions sum	2.54e+8	2.39e+8	3.44e+8	**2.32e+8**

- Including orbit space search always helpful
- Including symmetrical lookups not very helpful (for coverage)

More Results . . .

. . . on the poster!

Conclusions

- Planning benchmarks contain lots of symmetries
- Symmetry-based techniques improve state-of-the-art planning techniques
- Orbit space search achieves best performance
- BMPX does not help as much as in heuristic search problems