Symbolic Search and Abstraction Heuristics for Cost-Optimal Planning

Álvaro Torralba
Advisors: Daniel Borrajo and Carlos Linares López

Universidad Carlos III de Madrid – June 2, 2015

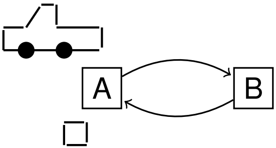
Outline

1. Introduction
   - Cost-Optimal Planning
2. Symbolic Search
   - (Background) Symbolic Search
   - Image Computation
   - State Invariants
3. Abstraction Heuristics
   - (Background) Abstractions
   - Merge-and-Shrink for Symbolic Search
   - Symbolic Perimeter Merge-and-Shrink
4. Symbolic Bidirectional Heuristic Search
5. Conclusions
   - Final Results: IPC14
   - Conclusions

Automated Planning

Given a planning task:

- A *logical description* of the initial situation, goal condition and a set of possible actions



$$\mathcal{V} = \{\text{at-T}= \{A, B\}, \text{at-P}= \{A, B, T\}\ \}$$
$$s_0 = \{\text{at-T } A, \text{at-P } A\}$$
$$s_\star = \{\text{at-P } B\}$$
$$\mathcal{O} = \{\text{move-T } (A, B), \text{move-T } (B, A), \text{load-P}(A), \dots\ \}$$

$\rightarrow$ Find a plan (sequence of actions)

$\rightarrow$ Cost-optimal: plan of minimum cost (prove it)

Empirical Evaluation Methods

SAS+ task $\longrightarrow$ $\boxed{\text{Planner}}$ $\longrightarrow$ Optimal plan

$\rightarrow$ Domain independent!! a planner can deal with any task

Empirical Evaluation Methods

SAS+ task → Planner → Optimal plan

→ Domain independent!! a planner can deal with any task

- Empirical evaluation methods:
  - International Planning Competition: 1998, 2000, 2002, 2004, 2006, 2008, 2011, 2014, . . .
  - Standard set of benchmark domains: 1998-2011
  - Time limit: 30 minutes
  - Memory limit: 4GB RAM
  - Coverage: number of problems solved
  - Time: solve problems faster

Motivation of this Thesis

- Improve state-of-the-art optimal planning

→ Efficiently solve optimal planning problems

- Techniques considered
    - Bidirectional search
    - Symbolic search
    - Abstraction heuristics

- Understand strengths/weaknesses
- Understand relation between techniques

Motivation of this Thesis

- Improve state-of-the-art optimal planning

$\rightarrow$ Efficiently solve optimal planning problems

- Techniques considered
  - Bidirectional search
  - Symbolic search $\Rightarrow$ GAMER: winner of IPC 2008
  - Abstraction heuristics

    $\Rightarrow$ Merge-and-shrink: runner-up and part of the winner of IPC 2011
- Understand strengths/weaknesses
- Understand relation between techniques

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A*
- Uniform-Cost

- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^{+}
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^{m}
- Flow
  - max
  - add
  - LP

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A*
- Uniform-Cost
- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^{+}
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^{m}
- Flow
  - max
  - add
  - LP

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A^*
- Uniform-Cost

- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^+
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^m
- Flow
  - max
  - add
  - LP

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A^*
- Uniform-Cost
- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^+
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^m
- Flow
- max | add | LP

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A^*
- Uniform-Cost
- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^{+}
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^{m}
- Flow
- max | add | LP

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A^*
- Uniform-Cost
- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^+
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^m
- Flow
  - max
  - add
  - LP

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning

State of the Art in Cost-Optimal Planning

Explicit search | Symbolic search

Algorithms

- A^*
- Uniform-Cost
- forward
- backward
- bidirectional

Heuristics

- Delete-relaxation: h^{max}, h^+
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S
- Critical paths: h^m
- Flow

max | add | LP

Pruning techniques

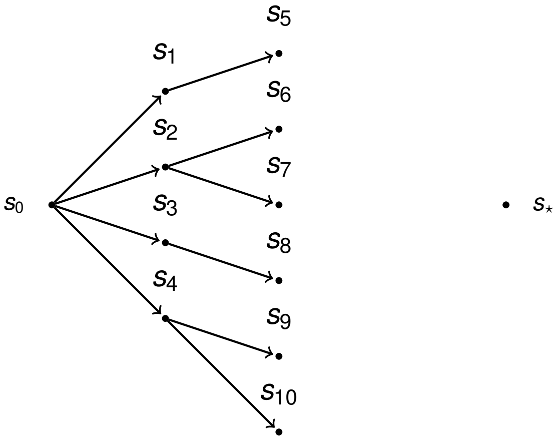
- State invariants
- Symmetries
- Partial-order pruning

Outline

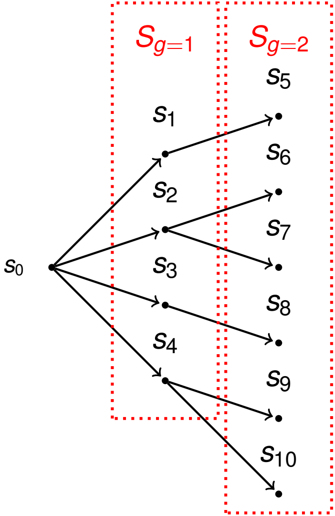
From Explicit to Symbolic Search

From Explicit to Symbolic Search

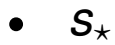
From Explicit to Symbolic Search



From Explicit to Symbolic Search



Reason with sets of states!



Binary Decision Diagrams (BDDs)

- Sets of states represented with Binary Decision Diagrams
  - Variable ordering
  - Reduction rules
- Possible exponential gain in memory/time
- Efficient operations (polynomial in BDD size)

1. (at Truck A) (at Package A)
2. (at Truck A) (in Package Truck)
3. (at Truck B) (at Package A)

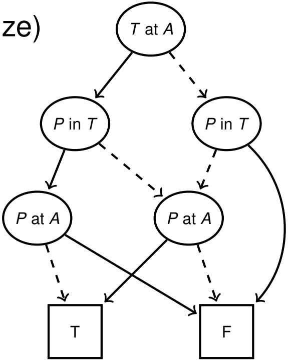
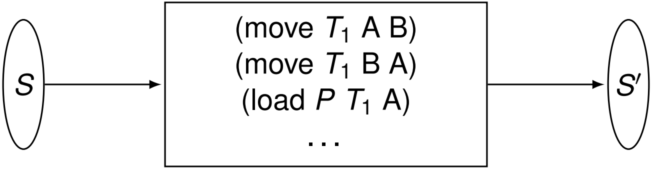


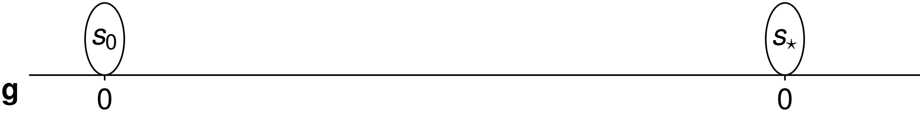
Image Computation

- Expand a set of states and generate the successor states
- Transition Relation: BDD that represents one or more planning actions with the same cost



$$S' \leftarrow image(S, T) = \exists x \,.\, S(x) \wedge T(x, x')[x' \leftrightarrow x]$$

Symbolic Bidirectional Breadth-First Search



Symbolic Bidirectional Breadth-First Search

Symbolic Bidirectional Breadth-First Search

- Decide forward or backward direction at each step

Symbolic Bidirectional Breadth-First Search

- Decide forward or backward direction at each step

Symbolic Bidirectional Breadth-First Search

- Decide forward or backward direction at each step

Outline

Optimizing Image Computation

- Image computation is the main bottleneck in symbolic search
- How to represent the Transition Relation?
  - Monolithic relation ⇒ may use exponential memory
  - Solution in GAMER ⇒ One TR for each action

move-T (A, B)
load-P (A)
move-T (B, A)
...

Optimizing Image Computation

- Image computation is the main bottleneck in symbolic search
- How to represent the Transition Relation?
  - Monolithic relation ⇒ may use exponential memory
  - Solution in GAMER ⇒ One TR for each action
- Idea 1: Separate preconditions and effects

  → avoid using auxiliary variables!

move-T (A, B)	pre: at-T A	eff: at-T B
load-P (A)	pre: at-T A, at-P A	eff: at-P T
move-T (B, A)	pre: at-T B	eff: at-T A

...

Optimizing Image Computation

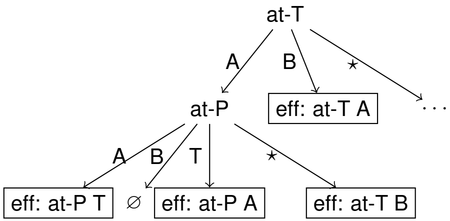
- Image computation is the main bottleneck in symbolic search
- How to represent the Transition Relation?
  - Monolithic relation ⇒ may use exponential memory
  - Solution in GAMER ⇒ One TR for each action
- Idea 1: Separate preconditions and effects
  → avoid using auxiliary variables!
- Idea 2: Conjunction Tree
  → check preconditions of all operators simultaneously

move-T (A, B)

load-P (A)

move-T (B, A)

...



Optimizing Image Computation

- Image computation is the main bottleneck in symbolic search
- How to represent the Transition Relation?
  - Monolithic relation $\Rightarrow$ may use exponential memory
  - Solution in GAMER $\Rightarrow$ One TR for each action
- Idea 1: Separate preconditions and effects
  $\rightarrow$ avoid using auxiliary variables!
- Idea 2: Conjunction Tree
  $\rightarrow$ check preconditions of all operators simultaneously
- Idea 3: Aggregate TRs
  $\rightarrow$ different strategies to group the actions

move-T (A, B)
load-P (A)
move-T (B, A)
...

move-T (A, B) load-P (A)
move-T (B, A)

Empirical Results

- Compare image computation methods:
  1. TR^1: baseline approach
  2. TR^{1+}: avoid using x' variables
  3. CT^{L}_{20}: conjunction tree
  4. T^{DT}_{100k}: aggregate TRs

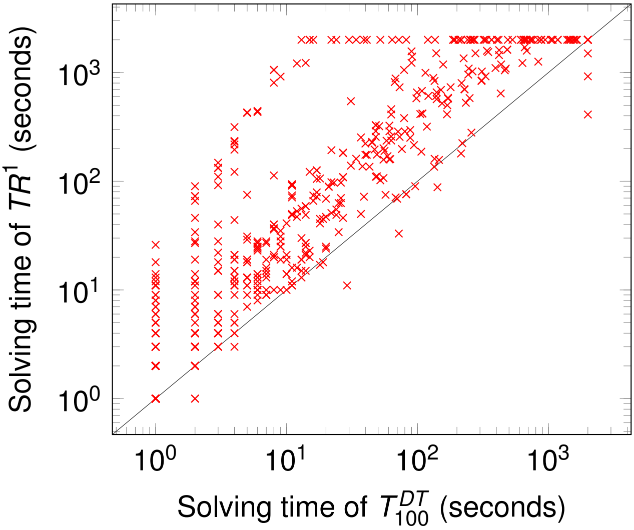
Total coverage of symbolic search algorithms over 1375 instances:

	TR^1	TR^{1+}	CT^{L}_{20}	T^{DT}_{100k}
Forward uniform-cost search	699	676	724	**742**
Backward uniform-cost search	444	525	529	**532**
Bidirectional uniform-cost search	729	763	769	**793**
BDDA* with SPDBs	705	717	724	**764**

$$TR^1 \leq TR^{1+} \leq CT^{L}_{20} \leq T^{DT}_{100k}$$

(across all domains)

Time of Bidirectional Search



Outline

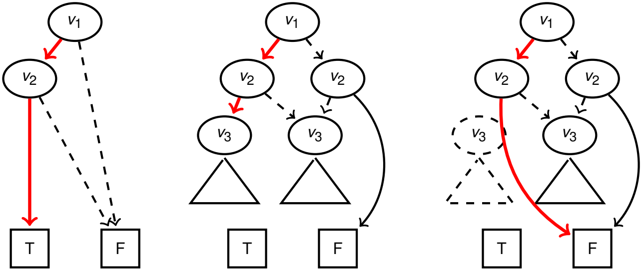
Motivation: State Invariants in Symbolic Search

- Invariant: holds in all states that may belong to a solution path
  1. Mutex: pair of facts that cannot be true in the same state
     $\rightarrow$ a truck cannot be simultaneously at two locations
  2. Invariant group: Set of facts such that exactly one is true
     $\rightarrow$ a truck must be somewhere

- Generated computing h^2 in both directions
- Useful for:
  1. Removing operators from the planning task
  2. Pruning invalid states during the search

Encoding Constraints with *cBDD*

- *cBDD*: BDD that describes invalid states
  1. Mutex: $f_i \wedge f_j$
  2. "At-least-1" invariant: $\neg(f_1 \vee f_2 \vee \ldots \vee f_n)$
- Remove invalid states from S_g: $S_g \setminus cBDD$



(c) *cBDD* (d) S_g (e) $S_g \setminus cBDD$

e-deletion: encode invariants in the TRs

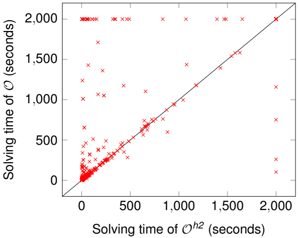
$\rightarrow$ no invalid states are generated

Experimental Results

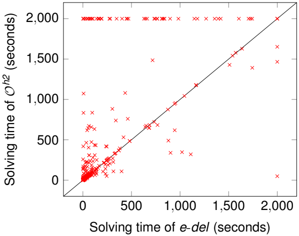
- Constraints found in 35 out of 43 domains
- 10%-74% invalid operators found in 17 out of 43 domains
- Mutex types:
  - Baseline (B)
  - Not pruning invalid states: $\mathcal{M}_\emptyset$
  - Pruning invalid states: *cBDD* or *e-deletion* (*e-del*)

		Remove invalid ops		
	B	$\mathcal{M}_\emptyset$	*cBDD*	*e-del*
Forward uniform-cost search	699	742	745	750
Backward uniform-cost search	509	532	677	696
Bidirectional uniform-cost search	765	793	836	**841**
BDDA* with SPDBs	736	764	777	781

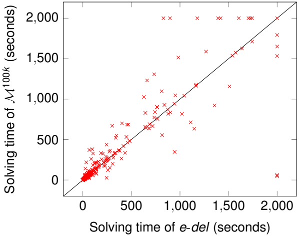
Time of Bidirectional Uniform-Cost Search



(f) remove operators

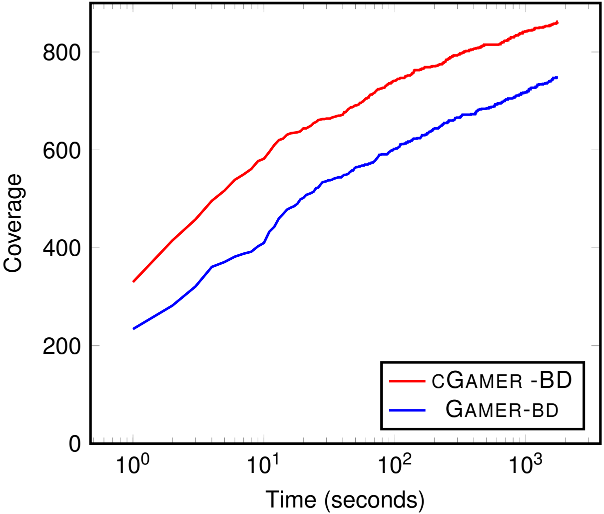


(g) prune invalid states

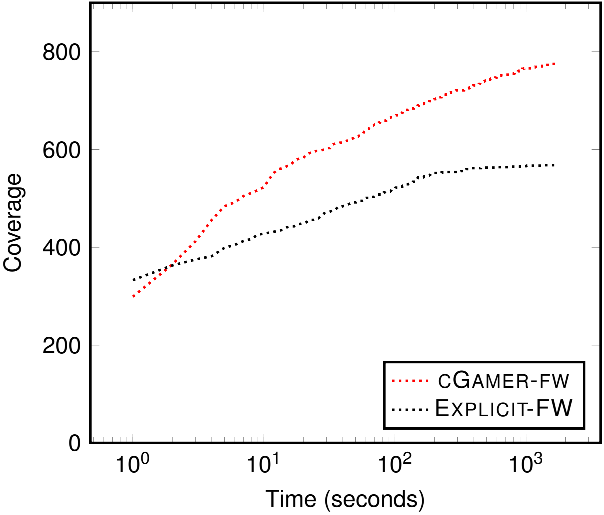


(h) e-deletion vs cBDD

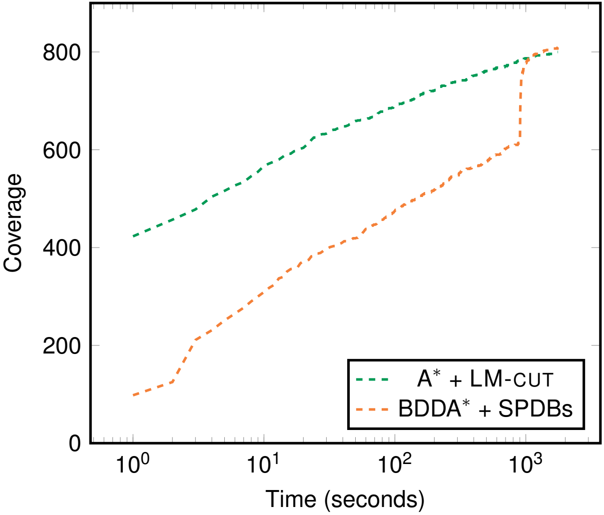
Comparison with State-of-the-Art Planners



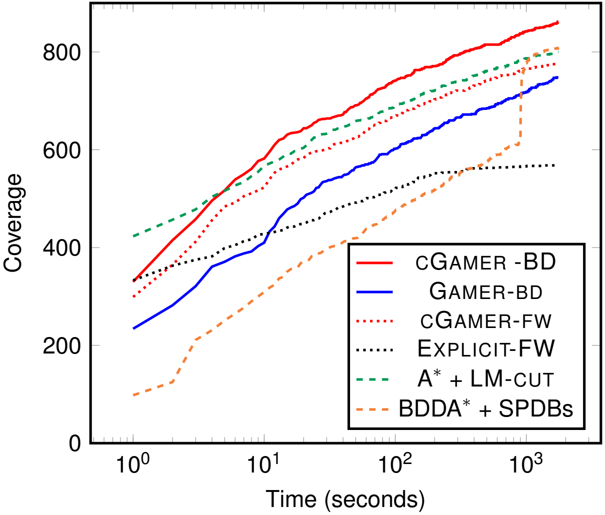
Comparison with State-of-the-Art Planners



Comparison with State-of-the-Art Planners



Comparison with State-of-the-Art Planners



Summary

1. Image computation
   - Analyzed different methods for image computation
   - Best method: aggregate TRs
2. State invariants
   - Pruning invalid states (specially useful in bw search)
   - Best encoding for symbolic search: e-edeletion

These significantly improved performance of symbolic planning

$\rightarrow$ Symbolic bidirectional blind search is the current state-of-the-art for cost-optimal planning

Outline

Motivation: Heuristics in Symbolic Search

Heuristics

- Delete-relaxation: h^{max}, h^{+}
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S, CEGAR, Fork
- Critical paths: h^{m}
- Flow

max | add | LP

Motivation: Heuristics in Symbolic Search

Heuristics
Delete-relaxation: h^{max}, h^{+}
Landmarks: h^{LA}, LM-cut
Abstractions: PDBs, M&S, CEGAR, Fork
Critical paths: h^{m}
Flow
max · add · LP

Challenge: How to evaluate $h(s)$ on a set of states?

Motivation: Heuristics in Symbolic Search

Heuristics

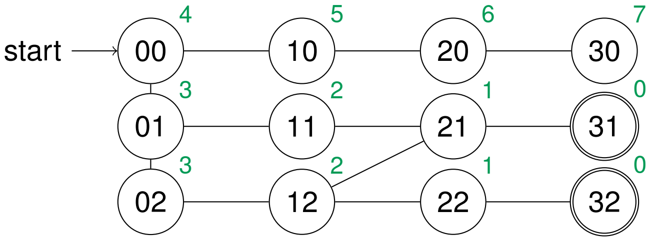
- Delete-relaxation: h^{max}, h^{+}
- Landmarks: h^{LA}, LM-cut
- Abstractions: PDBs, M&S, CEGAR, Fork
- Critical paths: h^{m}
- Flow

max | add | LP

Challenge: How to evaluate $h(s)$ on a set of states?

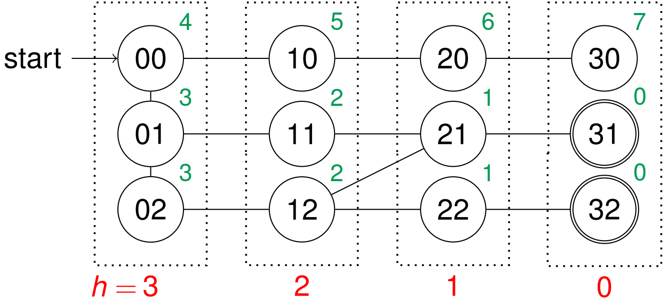
Abstraction Heuristics

- Abstraction: Mapping from states to abstract states
  - Smaller abstract state space → easier to search
  - Use optimal distances in abstract state space as heuristic
  - Preserve transitions → admissible estimation



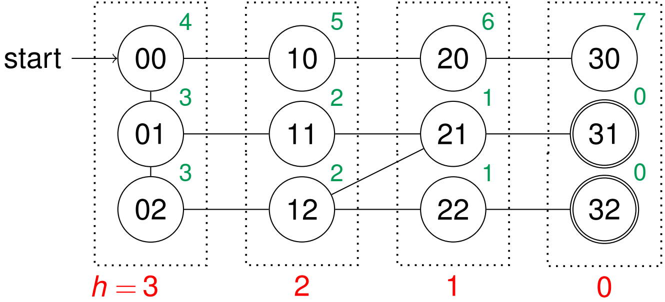
Abstraction Heuristics

- Abstraction: Mapping from states to abstract states
  - Smaller abstract state space → easier to search
  - Use optimal distances in abstract state space as heuristic
  - Preserve transitions → admissible estimation



Abstraction Heuristics

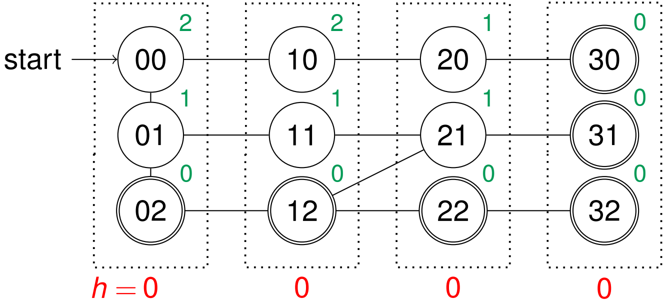
- Abstraction: Mapping from states to abstract states
  - Smaller abstract state space $\rightarrow$ easier to search
  - Use optimal distances in abstract state space as heuristic
  - Preserve transitions $\rightarrow$ admissible estimation



- Pattern Databases (PDBs)
  - Ignore some variables in the problem
  - Limitation: ignoring a single variable may relax too much

Abstraction Heuristics

- Abstraction: Mapping from states to abstract states
  - Smaller abstract state space $\rightarrow$ easier to search
  - Use optimal distances in abstract state space as heuristic
  - Preserve transitions $\rightarrow$ admissible estimation



- Pattern Databases (PDBs)
  - Ignore some variables in the problem
  - Limitation: ignoring a single variable may relax too much

Merge-and-Shrink Algorithm (M&S)

Algorithm 1: M&S

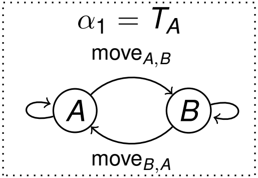
$\alpha_1 \leftarrow \Pi_{v_1}$
foreach $v \in \{v_2 \dots v_n\}$**:**
  if $|\alpha| > N$**:**
    $\texttt{shrink}(\alpha_{i-1}) \otimes \Pi_i$
  $\alpha_i \leftarrow \alpha_{i-1} \otimes \Pi_i$
return α

- Merge strategy: **Linear**
  $\rightarrow$ variable ordering
- Shrink strategy
  $\rightarrow$ reduce abstraction size

Merge-and-Shrink Algorithm (M&S)

Algorithm 1: M&S

$\alpha_1 \leftarrow \Pi_{v_1}$
foreach $v \in \{v_2 \dots v_n\}$**:**
  if $|\alpha| > N$**:**
    $\mathtt{shrink}(\alpha_{i-1}) \otimes \Pi_i$
  $\alpha_i \leftarrow \alpha_{i-1} \otimes \Pi_i$
return α



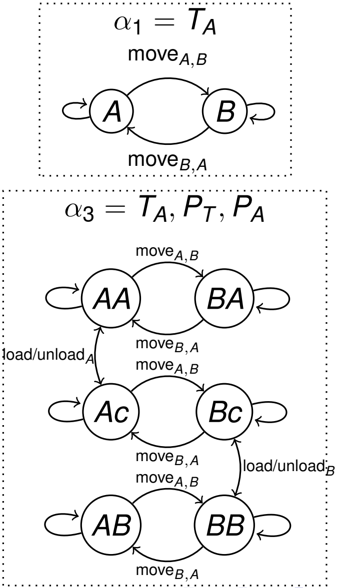
- Merge strategy: **Linear**
  $\rightarrow$ variable ordering
- Shrink strategy
  $\rightarrow$ reduce abstraction size

Merge-and-Shrink Algorithm (M&S)

Algorithm 1: M&S

$\alpha_1 \leftarrow \Pi_{v_1}$
foreach $v \in \{v_2 \ldots v_n\}$**:**
  if $|\alpha| > N$**:**
    $\texttt{shrink}(\alpha_{i-1}) \otimes \Pi_i$
  $\alpha_i \leftarrow \alpha_{i-1} \otimes \Pi_i$
return α

- Merge strategy: **Linear**
  - $\rightarrow$ variable ordering
- Shrink strategy
  - $\rightarrow$ reduce abstraction size

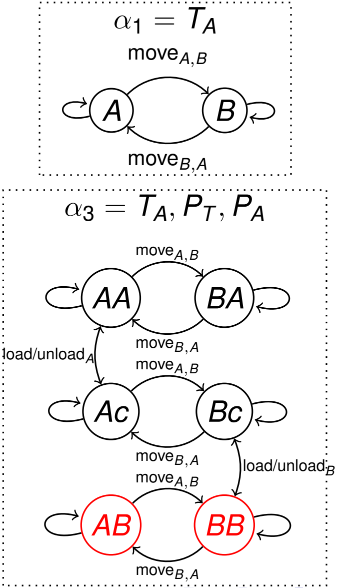


Merge-and-Shrink Algorithm (M&S)

Algorithm 1: M&S

$\alpha_1 \leftarrow \Pi_{v_1}$
foreach $v \in \{v_2 \dots v_n\}$**:**
  if $|\alpha| > N$**:**
    $\texttt{shrink}(\alpha_{i-1}) \otimes \Pi_i$
  $\alpha_i \leftarrow \alpha_{i-1} \otimes \Pi_i$
return α

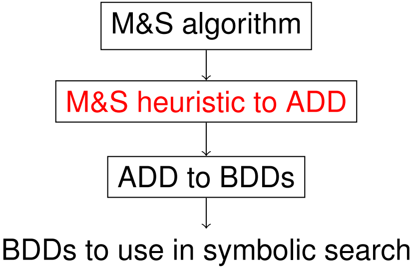
- Merge strategy: **Linear**
  - $\rightarrow$ variable ordering
- Shrink strategy
  - $\rightarrow$ reduce abstraction size



Outline

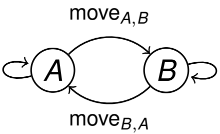
Merge-and-Shrink for Symbolic Search

- Hypothesis: BDDA* lacks good heuristics
  $\rightarrow$ BDDA* + M&S can improve results
- How to use M&S in symbolic search:

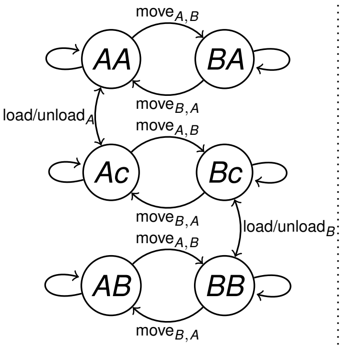


Merge-and-Shrink as ADDs

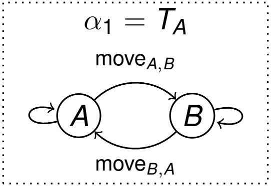
$\alpha_1 = T_A$

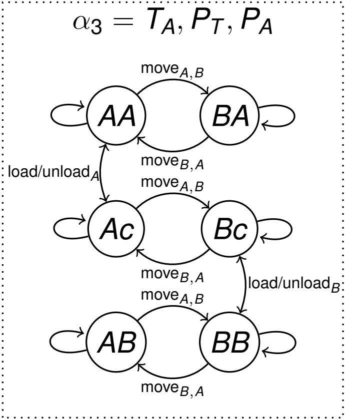


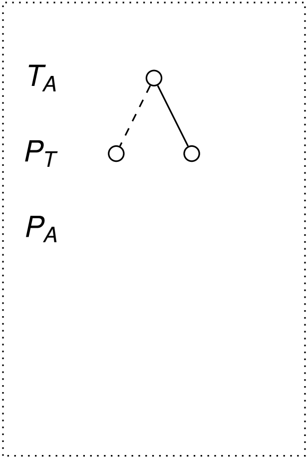
$\alpha_3 = T_A, P_T, P_A$



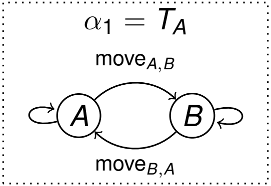
Merge-and-Shrink as ADDs

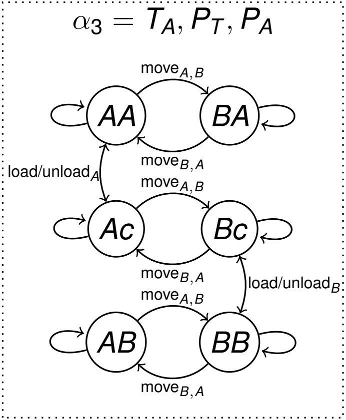


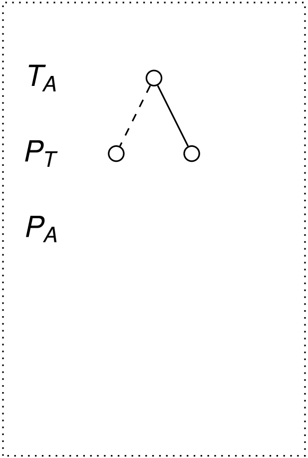




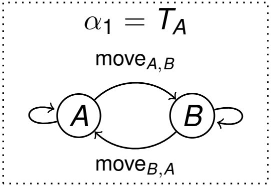
Merge-and-Shrink as ADDs

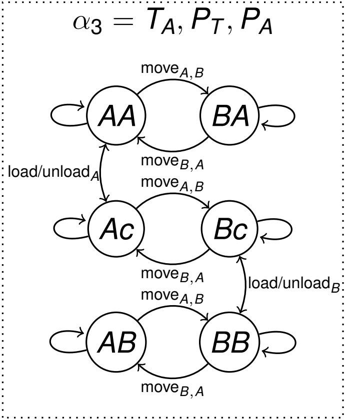


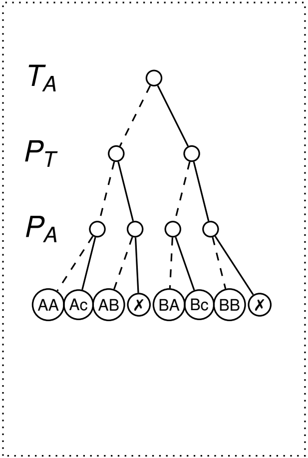




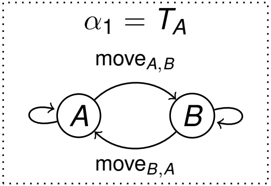
Merge-and-Shrink as ADDs

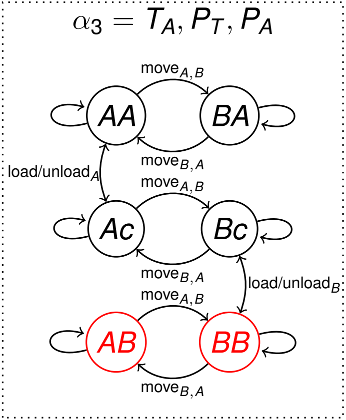


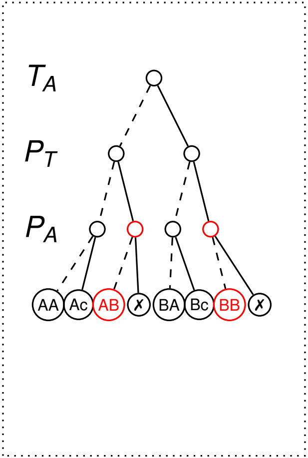




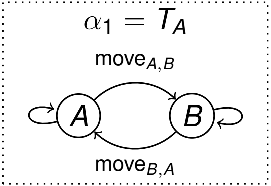
Merge-and-Shrink as ADDs

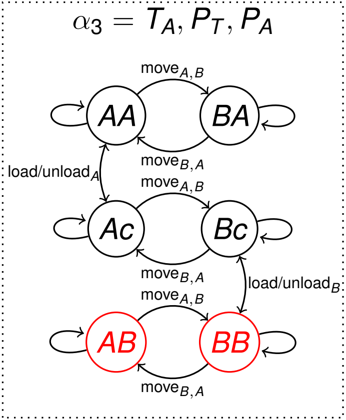


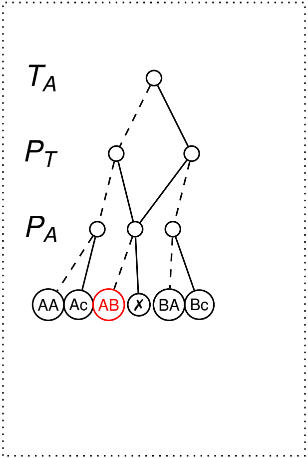




Merge-and-Shrink as ADDs







Theoretical Results

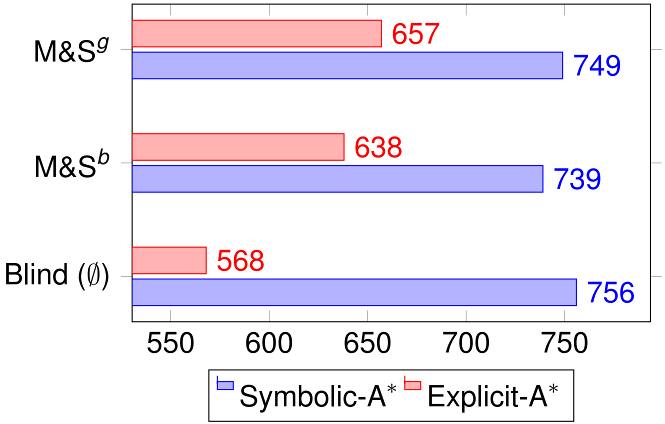
- M&S to ADDs/BDDs in polynomial time
- Related empirical results:
  - ADD representation of heuristics reduces memory
  - Variable ordering has a huge impact

- ADD/BDD reduction rules may achieve exponential gain in memory with respect to shrinking perfect strategies

  $\rightarrow$ shows potential of improvement for M&S strategies

Empirical Results

- Used M&S in symbolic search → Worse than symbolic PDBs



  - Contradicts our hypothesis

Outline

Motivation: Combine Symbolic Search and M&S

1. Symbolic PDBs: larger abstract state spaces
2. M&S: flexible abstractions

Can we get the best of both worlds?

Motivation: Combine Symbolic Search and M&S

1. Symbolic PDBs: larger abstract state spaces
2. M&S: flexible abstractions

Can we get the best of both worlds?

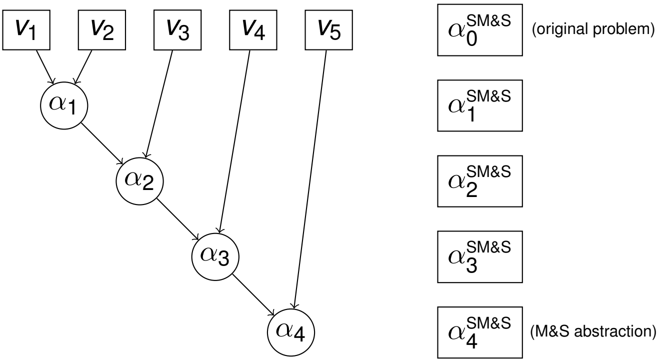
$\rightarrow$Use symbolic search to search M&S abstractions!

Symbolic Perimeter M&S:

1. Symbolic M&S abstractions: larger M&S abstract state spaces
2. Perimeter abstractions

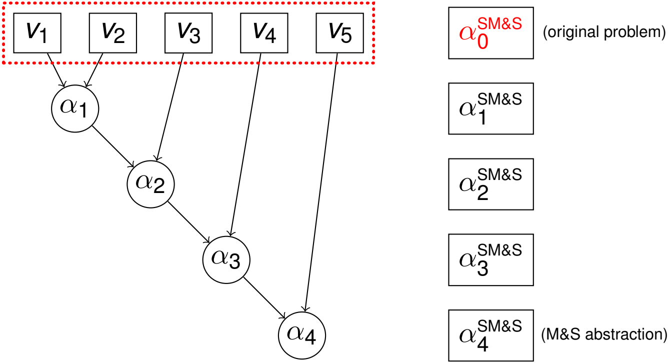
SM&S Hierarchy

Enlarged M&S abstractions: to perform symbolic search



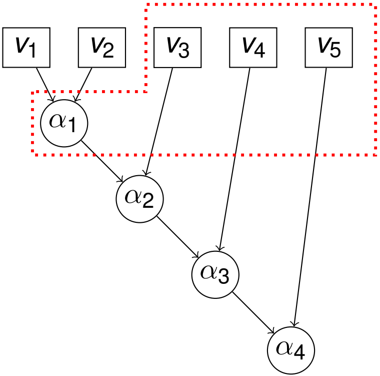
SM&S Hierarchy

Enlarged M&S abstractions: to perform symbolic search



SM&S Hierarchy

Enlarged M&S abstractions: to perform symbolic search



$\alpha_0^{\text{SM\&S}}$ (original problem)

$\alpha_1^{\text{SM\&S}}$

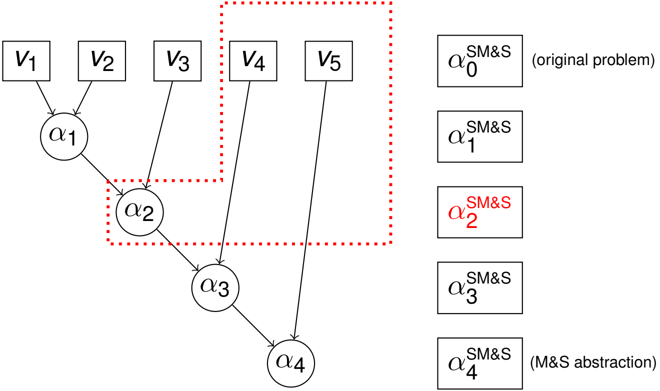
$\alpha_2^{\text{SM\&S}}$

$\alpha_3^{\text{SM\&S}}$

$\alpha_4^{\text{SM\&S}}$ (M&S abstraction)

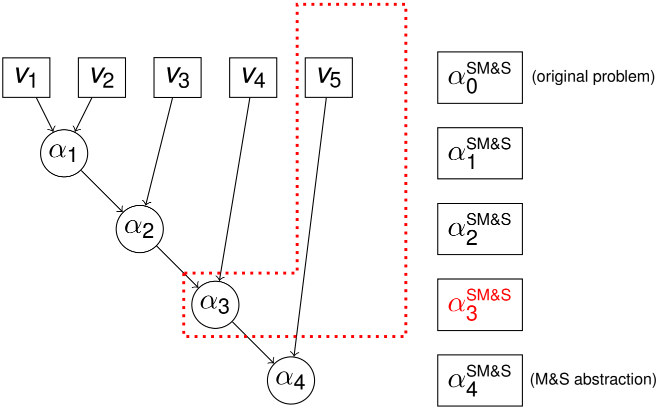
SM&S Hierarchy

Enlarged M&S abstractions: to perform symbolic search



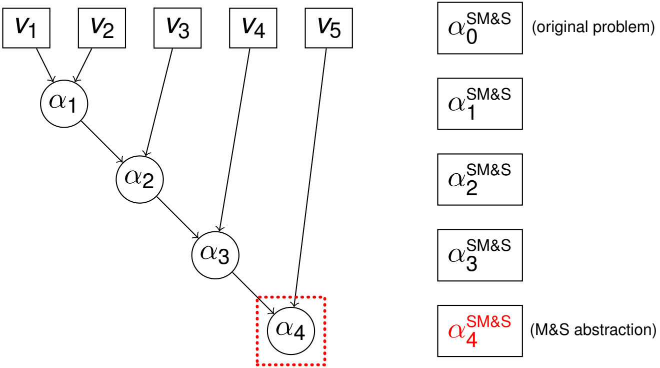
SM&S Hierarchy

Enlarged M&S abstractions: to perform symbolic search



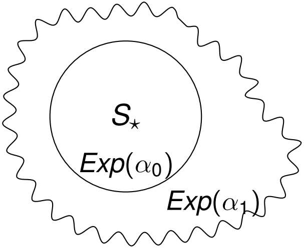
SM&S Hierarchy

Enlarged M&S abstractions: to perform symbolic search



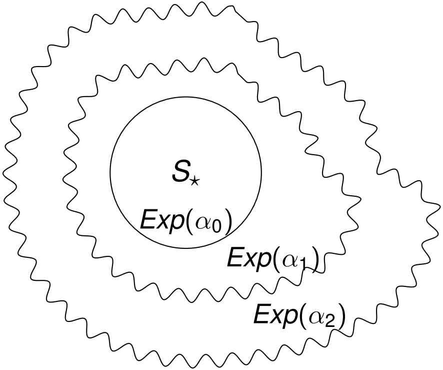
Perimeter Abstractions

- Challenges addressed with symbolic search
  1. Regression
  2. Expensive operations:
     - ★ membership in perimeter
     - ★ frontier mapping
  3. Set perimeter radius
- Contributions
  1. Multiple abstraction levels
  2. Improved initialization of abstract searches

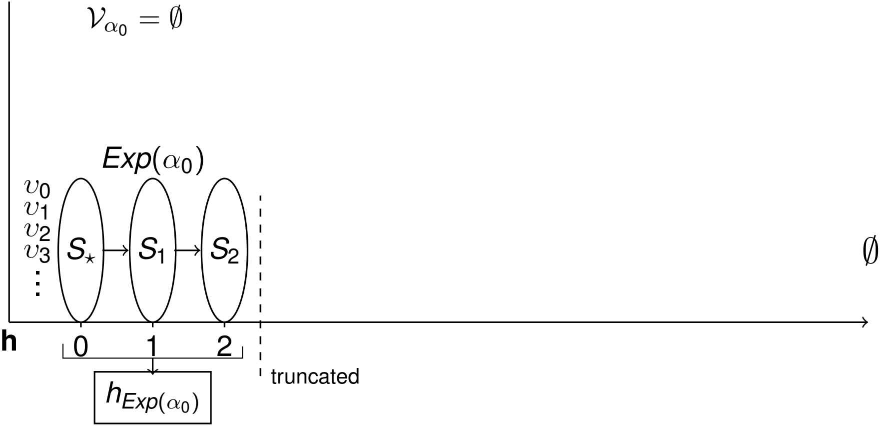


Perimeter Abstractions

- Challenges addressed with symbolic search
  1. Regression
  2. Expensive operations:
     - ★ membership in perimeter
     - ★ frontier mapping
  3. Set perimeter radius
- Contributions
  1. Multiple abstraction levels
  2. Improved initialization of abstract searches

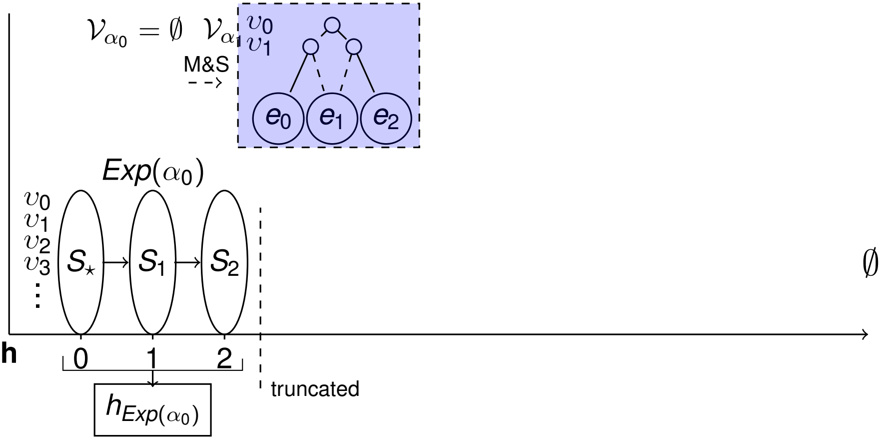


Symbolic Perimeter Merge-and-Shrink



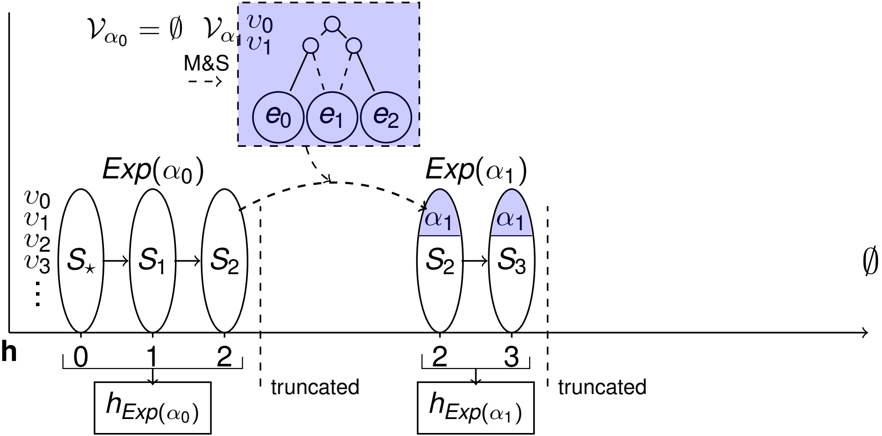
- $h_{\text{SPM\&S}}$ heuristic is admissible and consistent

Symbolic Perimeter Merge-and-Shrink



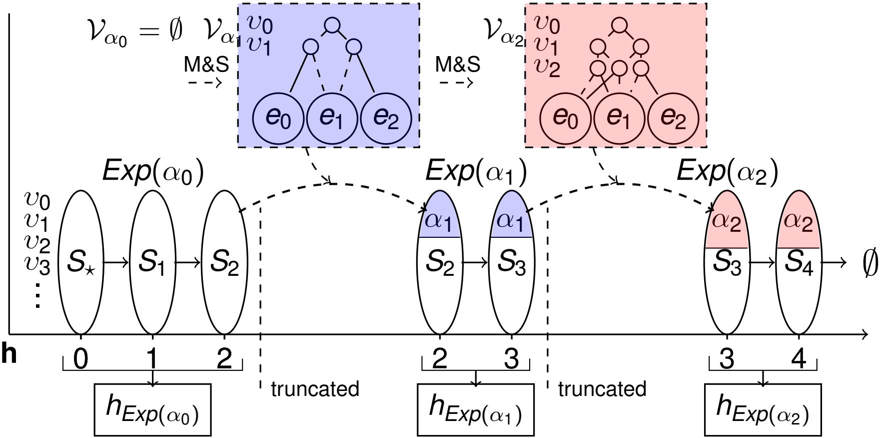
- $h_{\text{SPM\&S}}$ heuristic is admissible and consistent

Symbolic Perimeter Merge-and-Shrink



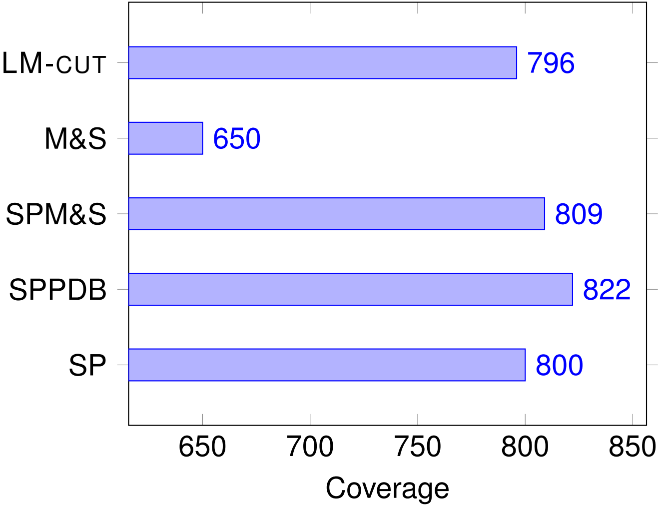
- $h_{\text{SPM\&S}}$ heuristic is admissible and consistent

Symbolic Perimeter Merge-and-Shrink

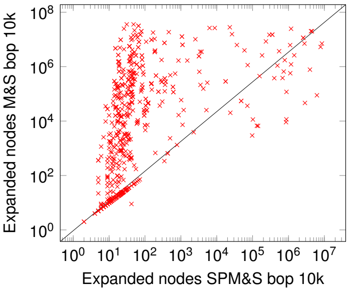


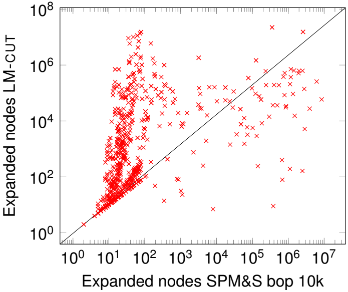
- $h_{\text{SPM\&S}}$ heuristic is admissible and consistent

Empirical Results

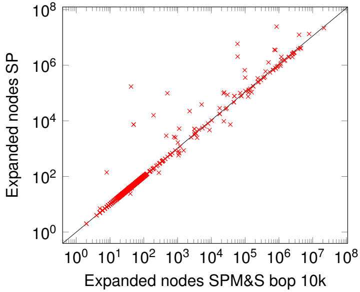


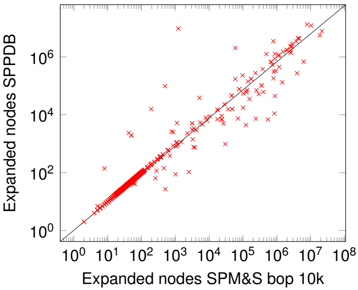
Empirical Results: Expanded Nodes





Empirical Results: Expanded Nodes





Summary

Symbolic Perimeter M&S

- Combines M&S, perimeter abstractions and symbolic search
- Improvements to perimeter abstractions
- Synergy between symbolic search and perimeter abstractions
- More accurate heuristic than both!

But...

Results still worse than symbolic bidirectional uniform-cost search

Outline

Motivation: Heuristics in Symbolic Bidirectional Search

- Observations
  1. Bidirectional brute-force search is a state-of-the-art technique
  2. Good symbolic abstraction heuristics

Motivation: Heuristics in Symbolic Bidirectional Search

- Observations
  1. Bidirectional brute-force search is a state-of-the-art technique
  2. Good symbolic abstraction heuristics

- Use abstraction heuristics in symbolic bidirectional search!

Motivation: Heuristics in Symbolic Bidirectional Search

- Observations
  1. Bidirectional brute-force search is a state-of-the-art technique
  2. Good symbolic abstraction heuristics

- Use abstraction heuristics in symbolic bidirectional search!

- However, bidirectional heuristic search is not so easy:
  - Very promising since years ago
  - Never really able to outperform A^* or bidirectional uniform-cost search

Algorithm

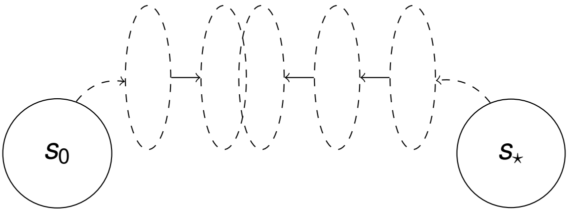
- Main idea:
  1. Start symbolic bidirectional uniform-cost search
     - ★ If it succeeds → done!
  2. Detect when it is going to fail and activate heuristics
- Abstraction heuristics: Bidirectional, Partial, Perimeter

Algorithm

- Main idea:
  1. Start symbolic bidirectional uniform-cost search
     - ⋆ If it succeeds → done!
  2. Detect when it is going to fail and activate heuristics
- Abstraction heuristics: Bidirectional, Partial, Perimeter
- Decide which search advance: **useful** and **feasible**

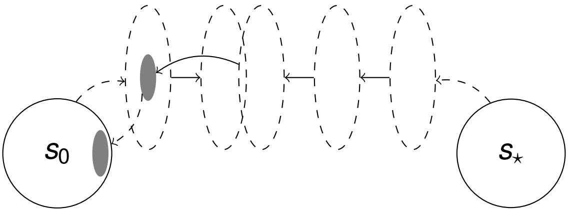
Algorithm

- Main idea:
  1. Start symbolic bidirectional uniform-cost search
     - ★ If it succeeds → done!
  2. Detect when it is going to fail and activate heuristics
- Abstraction heuristics: Bidirectional, Partial, Perimeter
- Decide which search advance: **useful** and **feasible**



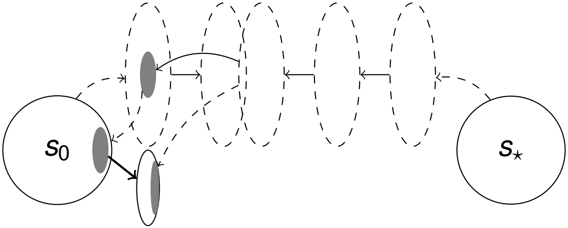
Algorithm

- Main idea:
  1. Start symbolic bidirectional uniform-cost search
     - ⋆ If it succeeds → done!
  2. Detect when it is going to fail and activate heuristics
- Abstraction heuristics: Bidirectional, Partial, Perimeter
- Decide which search advance: **useful** and **feasible**

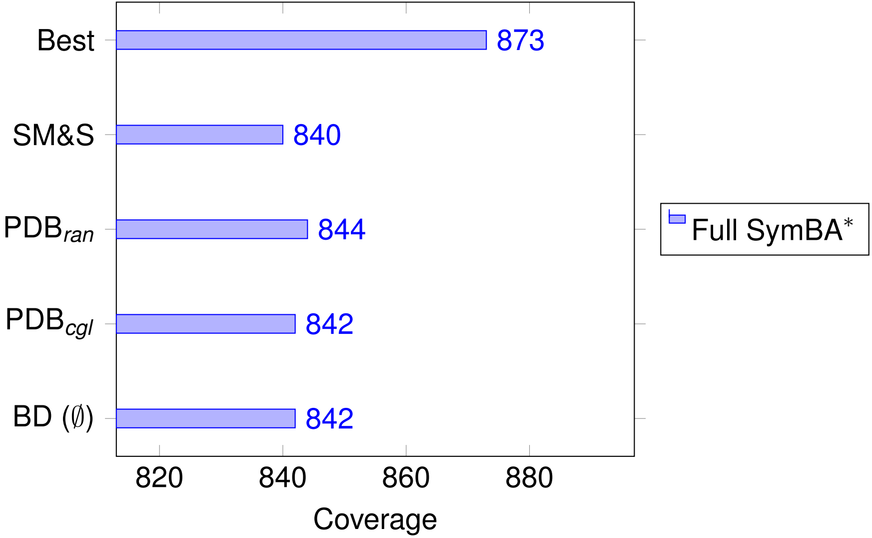


Algorithm

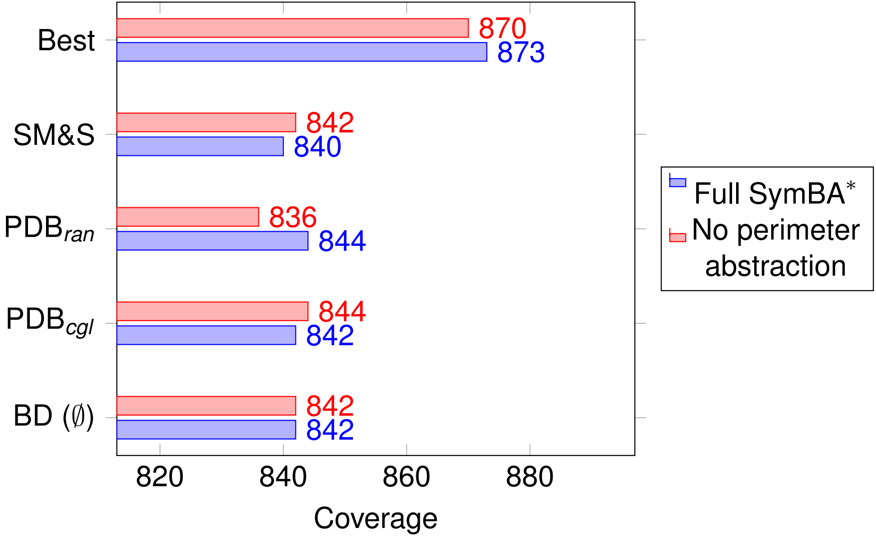
- Main idea:
  1. Start symbolic bidirectional uniform-cost search
     - ★ If it succeeds → done!
  2. Detect when it is going to fail and activate heuristics
- Abstraction heuristics: Bidirectional, Partial, Perimeter
- Decide which search advance: **useful** and **feasible**



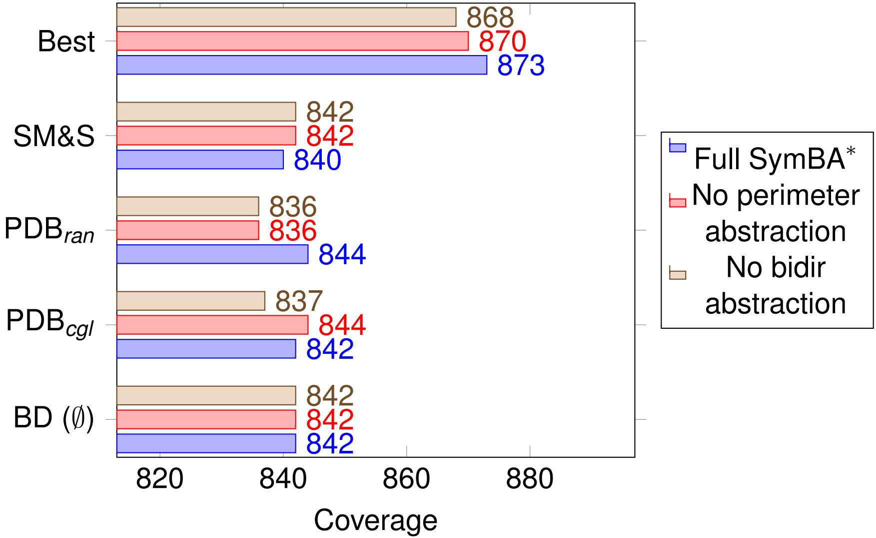
Empirical Results



Empirical Results



Empirical Results



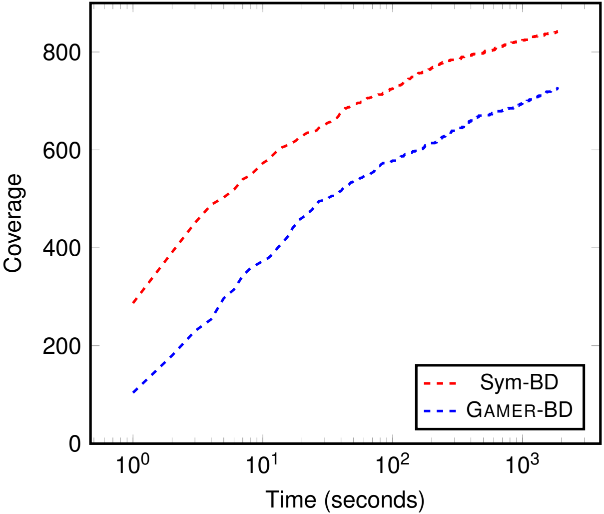
Summary

- Contributions:
  - SymBA*: a symbolic bidirectional heuristic search algorithm
  - Bidirectional search in abstract state spaces
  - Synergy: Symbolic search + Bidirectional search + Perimeter abstractions

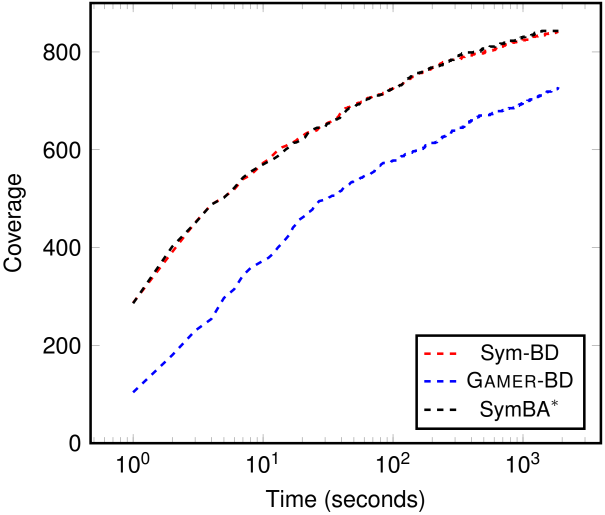
- Symbolic Bidirectional A* is possible
  - Future work: domain-independent abstraction strategies (better than a random selection)

Outline

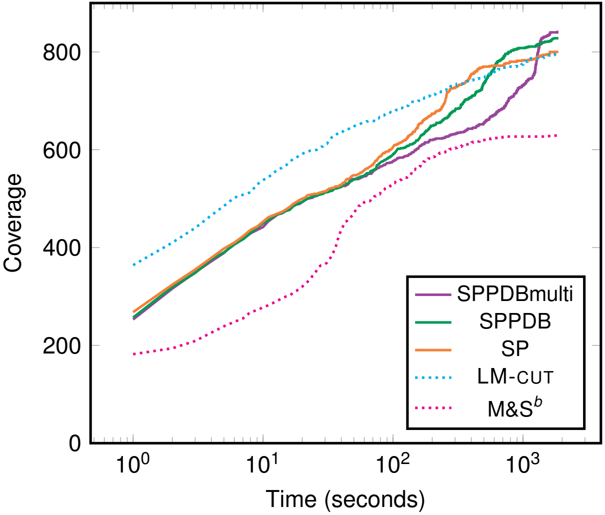
Final Results



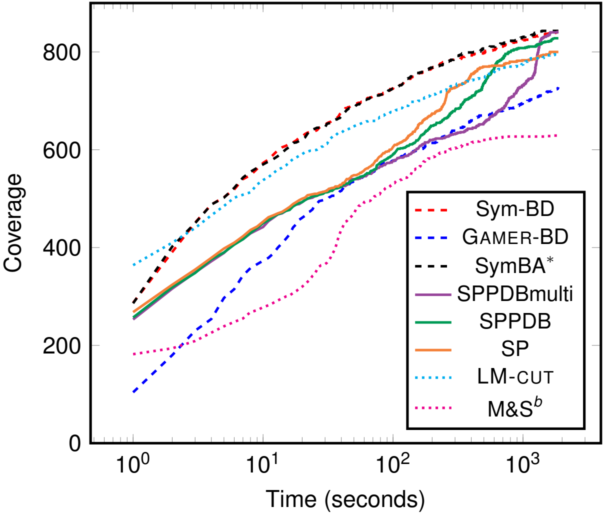
Final Results



Final Results



Final Results



2014 International Planning Competition

- Submitted our planners to the 2014-IPC
    1. cGAMER: Symbolic Bidirectional uniform-cost search with image computation and state-invariant constraints
    2. SPM&S: A^* with symbolic perimeter PDBs and M&S
    3. SymBA*: Symbolic Bidirectional A^* with SPM&S
- Competed against:
    - GAMER: baseline symbolic planner
    - Top explicit-state search planners and portfolios
- Disclaimer: IPC results are not everything
    - Domains/Instances selection, bugs, . . .

2014 International Planning Competition

	Barman	Cave	Childsnack	Citycar	Floortile	GED	Hiking	Maintenan	Openstack	Parking	Tetris	Tidybot	Transport	Visitall	Total
symba-2	**6**	3	4	**18**	**20**	**20**	**20**	4	**20**	0	10	10	**9**	7	**151**
symba-1	**6**	3	4	**18**	**20**	19	**20**	4	**20**	0	10	4	**9**	6	143
cgamer-bd	**6**	0	1	**18**	**20**	0	15	0	19	3	**11**	**13**	8	6	120
spmas	5	3	2	1	**20**	18	12	4	14	4	7	8	**9**	7	114
rida	0	3	0	16	5	19	17	**5**	3	**6**	8	8	8	**15**	113
dynamic-gamer	3	3	**10**	15	14	0	17	3	19	0	2	0	7	6	99
all-paca	0	**7**	0	17	6	15	13	**5**	8	**6**	3	1	5	12	98
cedalion	0	**7**	0	14	5	15	13	**5**	1	2	5	7	6	13	93
metis	3	**7**	6	0	8	15	13	**5**	3	4	8	7	6	6	91
nucelar	0	**7**	0	13	0	15	13	**5**	3	5	9	0	7	13	90
rlazya	0	**7**	0	17	5	15	9	**5**	2	4	6	7	6	5	88
gamer	3	3	2	**18**	13	0	14	0	16	0	3	0	6	5	83
hflow	0	3	0	0	3	7	4	**5**	1	0	10	0	5	**15**	53
miplan	0	**7**	0	11	0	0	10	**5**	0	1	0	0	0	13	47
dpmplan	0	**7**	0	8	0	0	0	**5**	0	5	0	0	6	12	43
hpp-ce	0	0	0	7	0	3	0	**5**	0	0	0	0	0	0	15
hpp	0	0	0	6	0	3	0	**5**	0	0	0	0	0	0	14

2014 International Planning Competition

	Barman	Cave	Childsnack	Citycar	Floortile	GED	Hiking	Maintenan	Openstack	Parking	Tetris	Tidybot	Transport	Visitall	Total
symba-2	**6**	3	4	**18**	**20**	**20**	**20**	4	**20**	0	10	10	**9**	7	**151**
symba-1	**6**	3	4	**18**	**20**	19	**20**	4	**20**	0	10	4	**9**	6	143
cgamer-bd	**6**	0	1	**18**	**20**	0	15	0	19	3	**11**	**13**	8	6	120
spmas	5	3	2	1	**20**	18	12	4	14	4	7	8	**9**	7	114
rida	0	3	0	16	5	19	17	**5**	3	**6**	8	8	8	**15**	113
dynamic-gamer	3	3	**10**	15	14	0	17	3	19	0	2	0	7	6	99
all-paca	0	**7**	0	17	6	15	13	**5**	8	**6**	3	1	5	12	98
cedalion	0	**7**	0	14	5	15	13	**5**	1	2	5	7	6	13	93
metis	3	**7**	6	0	8	15	13	**5**	3	4	8	7	6	6	91
nucelar	0	**7**	0	13	0	15	13	**5**	3	5	9	0	7	13	90
rlazya	0	**7**	0	17	5	15	9	**5**	2	4	6	7	6	5	88
gamer	3	3	2	**18**	13	0	14	0	16	0	3	0	6	5	83
hflow	0	3	0	0	3	7	4	**5**	1	0	10	0	5	**15**	53
miplan	0	**7**	0	11	0	0	10	**5**	0	1	0	0	0	13	47
dpmplan	0	**7**	0	8	0	0	0	**5**	0	5	0	0	6	12	43
hpp-ce	0	0	0	7	0	3	0	**5**	0	0	0	0	0	0	15
hpp	0	0	0	6	0	3	0	**5**	0	0	0	0	0	0	14

2014 International Planning Competition

	Barman	Cave	Childsnack	Citycar	Floortile	GED	Hiking	Maintenan	Openstack	Parking	Tetris	Tidybot	Transport	Visitall	Total
symba-2	**6**	3	4	**18**	**20**	**20**	**20**	4	**20**	0	10	10	**9**	7	**151**
symba-1	**6**	3	4	**18**	**20**	19	**20**	4	**20**	0	10	4	**9**	6	143
cgamer-bd	**6**	0	1	**18**	**20**	0	15	0	19	3	**11**	**13**	8	6	120
spmas	5	3	2	1	**20**	18	12	4	14	4	7	8	**9**	7	114
rida	0	3	0	16	5	19	17	**5**	3	**6**	8	8	8	**15**	113
dynamic-gamer	3	3	**10**	15	14	0	17	3	19	0	2	0	7	6	99
all-paca	0	**7**	0	17	6	15	13	**5**	8	**6**	3	1	5	12	98
cedalion	0	**7**	0	14	5	15	13	**5**	1	2	5	7	6	13	93
metis	3	**7**	6	0	8	15	13	**5**	3	4	8	7	6	6	91
nucelar	0	**7**	0	13	0	15	13	**5**	3	5	9	0	7	13	90
rlazya	0	**7**	0	17	5	15	9	**5**	2	4	6	7	6	5	88
gamer	3	3	2	**18**	13	0	14	0	16	0	3	0	6	5	83
hflow	0	3	0	0	3	7	4	**5**	1	0	10	0	5	**15**	53
miplan	0	**7**	0	11	0	0	10	**5**	0	1	0	0	0	13	47
dpmplan	0	**7**	0	8	0	0	0	**5**	0	5	0	0	6	12	43
hpp-ce	0	0	0	7	0	3	0	**5**	0	0	0	0	0	0	15
hpp	0	0	0	6	0	3	0	**5**	0	0	0	0	0	0	14

2014 International Planning Competition

	Barman	Cave	Childsnack	Citycar	Floortile	GED	Hiking	Maintenan	Openstack	Parking	Tetris	Tidybot	Transport	Visitall	Total
symba-2	**6**	3	4	**18**	**20**	**20**	**20**	4	**20**	0	10	10	**9**	7	**151**
symba-1	**6**	3	4	**18**	**20**	19	**20**	4	**20**	0	10	4	**9**	6	143
cgamer-bd	**6**	0	1	**18**	**20**	0	15	0	19	3	**11**	**13**	8	6	120
spmas	5	3	2	1	**20**	18	12	4	14	4	7	8	**9**	7	114
rida	0	3	0	16	5	19	17	**5**	3	**6**	8	8	8	**15**	113
dynamic-gamer	3	3	**10**	15	14	0	17	3	19	0	2	0	7	6	99
all-paca	0	**7**	0	17	6	15	13	**5**	8	**6**	3	1	5	12	98
cedalion	0	**7**	0	14	5	15	13	**5**	1	2	5	7	6	13	93
metis	3	**7**	6	0	8	15	13	**5**	3	4	8	7	6	6	91
nucelar	0	**7**	0	13	0	15	13	**5**	3	5	9	0	7	13	90
rlazya	0	**7**	0	17	5	15	9	**5**	2	4	6	7	6	5	88
gamer	3	3	2	**18**	13	0	14	0	16	0	3	0	6	5	83
hflow	0	3	0	0	3	7	4	**5**	1	0	10	0	5	**15**	53
miplan	0	**7**	0	11	0	0	10	**5**	0	1	0	0	0	13	47
dpmplan	0	**7**	0	8	0	0	0	**5**	0	5	0	0	6	12	43
hpp-ce	0	0	0	7	0	3	0	**5**	0	0	0	0	0	0	15
hpp	0	0	0	6	0	3	0	**5**	0	0	0	0	0	0	14

Outline

Conclusions

- Symbolic search for cost-optimal planning:
  - Analysis of image computation
  - State-invariant pruning
- M&S heuristics in symbolic search planning
- SPM&S: new perimeter abstraction heuristic based in symbolic search and M&S
- Big question: can we use heuristics in symbolic planning?
  1. Used M&S and SPM&S in BDDA*
  2. SymBA*: symbolic bidirectional search + perimeter abstractions

Conclusions

- Symbolic bidirectional blind search
  - $\rightarrow$ Currently, the best method for cost-optimal planning (only beaten by heuristics in domains where the heuristics are very precise).
- SPM&S: state-of-the-art heuristic
- Highlighted the relevance of symbolic search and regression
- Synergy of symbolic bidirectional search and perimeter abstractions

List of Publications

Álvaro Torralba, Stefan Edelkamp, and Peter Kissmann. Transition trees for cost-optimal symbolic planning.
In *ICAPS*, 2013

Álvaro Torralba and Vidal Alcázar. Constrained symbolic search: On mutexes, BDD minimization and more.
In *SoCS*, 2013

Stefan Edelkamp, Peter Kissmann, and Álvaro Torralba. Symbolic A* search with pattern databases and the merge-and-shrink abstraction.
In *ECAI*, 2012

Álvaro Torralba, Carlos Linares López, and Daniel Borrajo. Symbolic merge-and-shrink for cost-optimal planning.
In *IJCAI*, 2013

Thank you for your attention!

Questions?