Large-Scale Symbolic Search

Álvaro Torralba

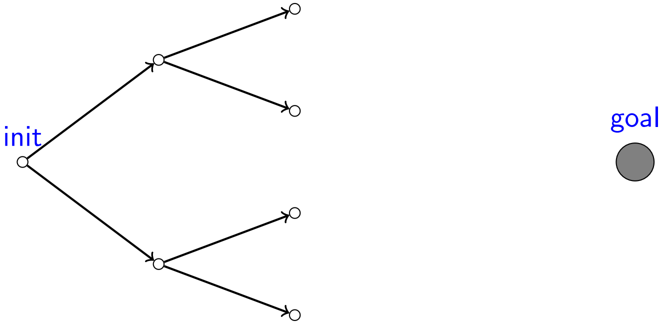
SAARLAND UNIVERSITY

SAARBRÜCKEN GRADUATE SCHOOL OF COMPUTER SCIENCE

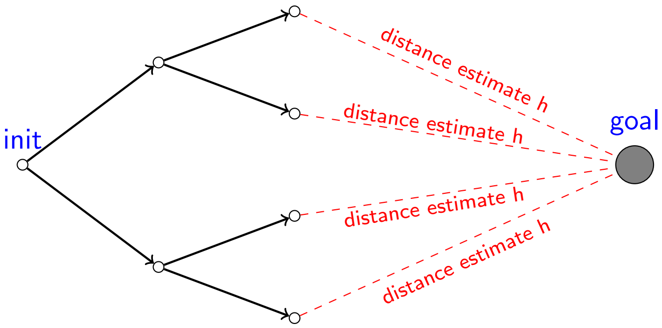
October 4, 2018

A successful approach: Heuristic Search

A successful approach: Heuristic Search

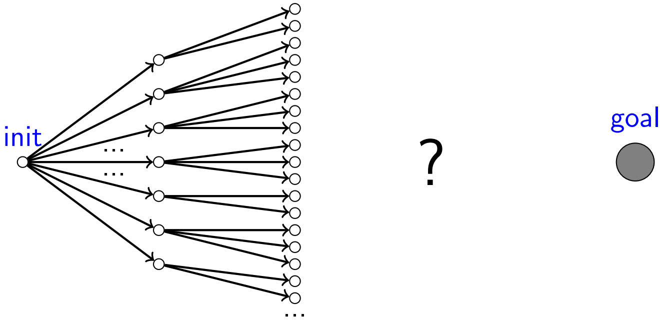


A successful approach: Heuristic Search



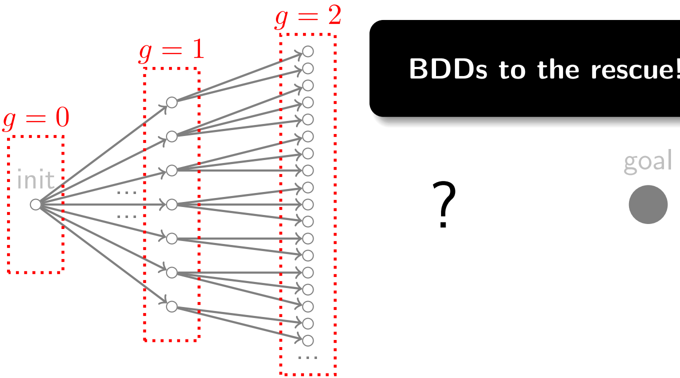
→ Forward state space search. Heuristic function h maps states s to an estimate $h(s)$ of goal distance.

State Space Explosion



Huge branching factor → state space *explosion*

State Space Explosion



BDDs to the rescue!

Huge branching factor → state space *explosion*

Classical Planning

Definition. *A planning task is a 4-tuple* $\Pi = (V, A, I, G)$ *where:*

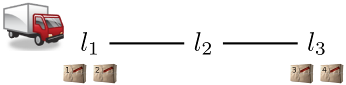
- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v.
- A *is a set of actions; each* $a \in A$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}^{0+}$.
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

Classical Planning

Definition. *A planning task is a 4-tuple* $\Pi = (V, A, I, G)$ *where:*

- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v.
- A *is a set of actions; each* $a \in A$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}^{0+}$.
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

Running Example:

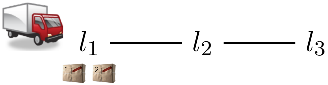


- $V = \{t, p_1, p_2, p_3, p_4\}$
  with $D_t = \{l_1, l_2, l_3\}$ and $D_{p_i} = \{t, l_1, l_2, l_3\}$.
- $A = \{load(p_i, x), unload(p_i, x), drive(x, x')\}$

Agenda

1. Symbolic Representation of Planning Tasks
2. Symbolic Search
3. Advantages And Limitations of Symbolic Search
4. Conclusions

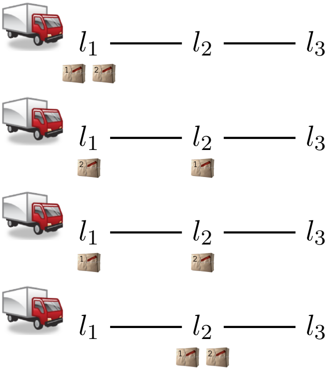
Sets of States as Logical Formulas



$$\langle t, l_1 \rangle \wedge \langle p_1, l_1 \rangle \wedge \langle p_2, l_1 \rangle$$

Disclaimer: In propositional logic there is no closed-world assumption. In our examples, we ignore state invariants: $\langle t, l_1 \rangle \leftrightarrow (\neg \langle t, l_2 \rangle \wedge \neg \langle t, l_3 \rangle), \ldots$

Sets of States as Logical Formulas



$\langle t, l_1 \rangle \wedge \langle p_1, l_1 \rangle \wedge \langle p_2, l_1 \rangle$

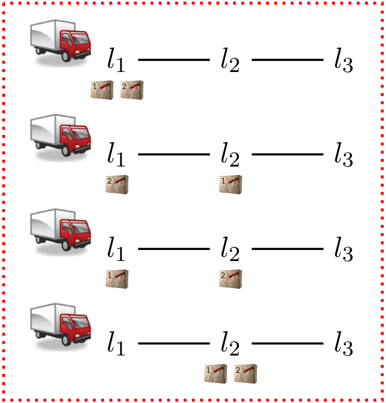
$\langle t, l_1 \rangle \wedge \langle p_1, l_2 \rangle \wedge \langle p_2, l_1 \rangle$

$\langle t, l_1 \rangle \wedge \langle p_1, l_1 \rangle \wedge \langle p_2, l_2 \rangle$

$\langle t, l_1 \rangle \wedge \langle p_1, l_2 \rangle \wedge \langle p_2, l_2 \rangle$

Disclaimer: In propositional logic there is no closed-world assumption. In our examples, we ignore state invariants: $\langle t, l_1 \rangle \leftrightarrow (\neg \langle t, l_2 \rangle \wedge \neg \langle t, l_3 \rangle), \ldots$

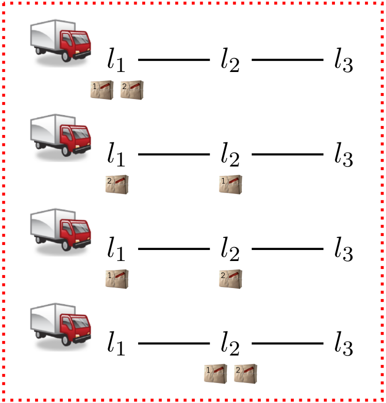
Sets of States as Logical Formulas



$$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_1\rangle$$
$$\vee$$
$$\langle t, l_1\rangle \wedge \langle p_1, l_2\rangle \wedge \langle p_2, l_1\rangle$$
$$\vee$$
$$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_2\rangle$$
$$\vee$$
$$\langle t, l_1\rangle \wedge \langle p_1, l_2\rangle \wedge \langle p_2, l_2\rangle$$

Disclaimer: In propositional logic there is no closed-world assumption. In our examples, we ignore state invariants: $\langle t, l_1\rangle \leftrightarrow (\neg\langle t, l_2\rangle \wedge \neg\langle t, l_3\rangle), \ldots$

Sets of States as Logical Formulas



$$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_1\rangle$$
$$\vee$$
$$\langle t, l_1\rangle \wedge \langle p_1, l_2\rangle \wedge \langle p_2, l_1\rangle$$
$$\vee$$
$$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_2\rangle$$
$$\vee$$
$$\langle t, l_1\rangle \wedge \langle p_1, l_2\rangle \wedge \langle p_2, l_2\rangle$$

$$\langle \mathbf{t}, \mathbf{l_1}\rangle \wedge (\langle \mathbf{p_1}, \mathbf{l_1}\rangle \vee \langle \mathbf{p_1}, \mathbf{l_2}\rangle) \wedge (\langle \mathbf{p_2}, \mathbf{l_1}\rangle \vee \langle \mathbf{p_2}, \mathbf{l_2}\rangle)$$

Disclaimer: In propositional logic there is no closed-world assumption. In our examples, we ignore state invariants: $\langle t, l_1\rangle \leftrightarrow (\neg\langle t, l_2\rangle \wedge \neg\langle t, l_3\rangle), \ldots$

How to Represent Logical Formulas in Practice?

Normal Form/Decision Diagram			
Negation NF (NNF)			
Disjunctive NF (DNF)			
Conjunctive NF (CNF)			
Binary DD (BDD)	[Bry86]		
Zero-sup DD (ZDD)	[Min93]		
Sentential DD (SDD)	[Dar11]		
Determ. DNNF (d-DNNF)	[Dar02]		
Decomp. NNF (DNNF)	[Dar01]		

How to Represent Logical Formulas in Practice?

Normal Form/Decision Diagram		$\vee$	$\wedge$	$\neg$	$\sigma \equiv \top$	$\sigma \equiv \bot$	$\sigma \equiv \sigma'$
Negation NF (NNF)		P	P	P	co-NP	NP	co-NP
Disjunctive NF (DNF)		P	E	E	co-NP	P	co-NP
Conjunctive NF (CNF)		E	P	E	P	NP	co-NP
Binary DD (BDD)	[Bry86]	E/P	E/P	P	P	P	P
Zero-sup DD (ZDD)	[Min93]	E/P	E/P	P	P	P	P
Sentential DD (SDD)	[Dar11]	E/P*	E/P*	P	P	P	P
Determ. DNNF (d-DNNF)	[Dar02]	P	E	E	co-NP	P	co-NP
Decomp. NNF (DNNF)	[Dar01]	P	E	E	co-NP	P	co-NP

$*$: In SDDs, $\vee$ and $\wedge$ with compression is not polynomial.

P: polynomial in the size of the representation

How to Represent Logical Formulas in Practice?

Normal Form/Decision Diagram		$\vee$	$\wedge$	$\neg$	$\sigma \equiv \top$	$\sigma \equiv \bot$	$\sigma \equiv \sigma'$
Negation NF (NNF)		P	P	P	co-NP	NP	co-NP
Disjunctive NF (DNF)		P	E	E	co-NP	P	co-NP
Conjunctive NF (CNF)		E	P	E	P	NP	co-NP
Binary DD (BDD)	[Bry86]	E/P	E/P	P	P	P	P
Zero-sup DD (ZDD)	[Min93]	E/P	E/P	P	P	P	P
Sentential DD (SDD)	[Dar11]	E/P*	E/P*	P	P	P	P
Determ. DNNF (d-DNNF)	[Dar02]	P	E	E	co-NP	P	co-NP
Decomp. NNF (DNNF)	[Dar01]	P	E	E	co-NP	P	co-NP

$*$: In SDDs, $\vee$ and $\wedge$ with compression is not polynomial.

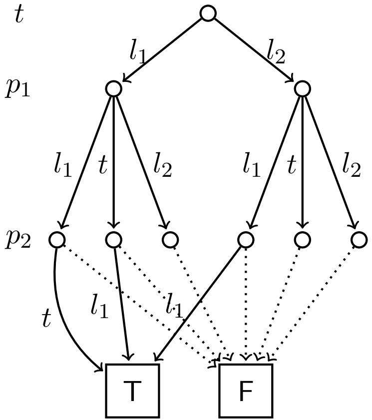
P: polynomial in the size of the representation

Binary Decision Diagrams (BDDs)

Multi-valued Decision Diagram (MDD)

DAG with a fixed variable ordering.

t	p_1	p_2
l_1	l_1	t
l_2	l_1	l_1
l_1	t	l_1



Binary Decision Diagrams (BDDs)

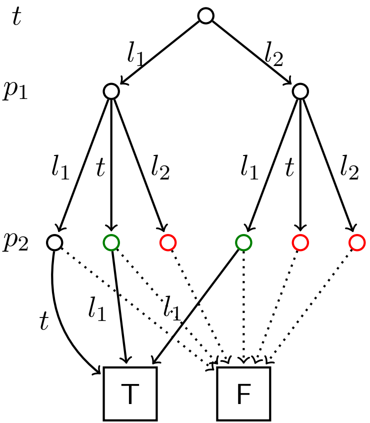
Multi-valued Decision Diagram (MDD)

DAG with a fixed variable ordering.

Reduction rules:

1. Each node represented only once
2. Nodes whose children are all the same are ommited

t	p_1	p_2
l_1	l_1	t
l_2	l_1	l_1
l_1	t	l_1



Binary Decision Diagrams (BDDs)

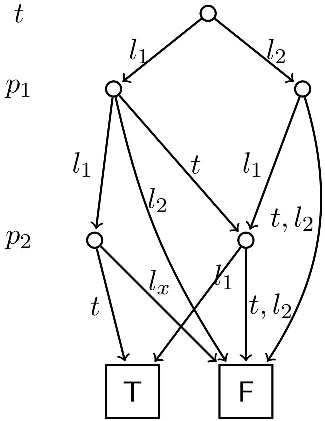
Multi-valued Decision Diagram (MDD)

DAG with a fixed variable ordering.

Reduction rules:

1. Each node represented only once
2. Nodes whose children are all the same are ommited

t	p_1	p_2
l_1	l_1	t
l_2	l_1	l_1
l_1	t	l_1



Binary Decision Diagrams (BDDs)

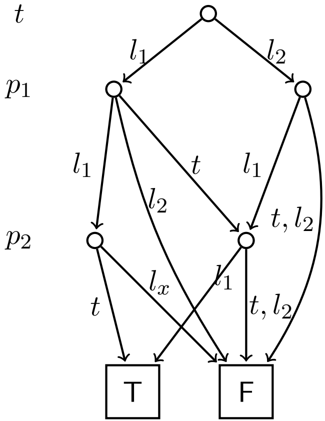
Multi-valued Decision Diagram (MDD)

DAG with a fixed variable ordering.

Reduction rules:

1. Each node represented only once
2. Nodes whose children are all the same are ommited

t	p_1	p_2
l_1	l_1	t
l_2	l_1	l_1
l_1	t	l_1



Binary Decision Diagrams: MDDs where variables are all binary
→Compilation that uses $log_2|D_v|$ binary variables per FDR variable v

Binary Decision Diagrams (BDDs)

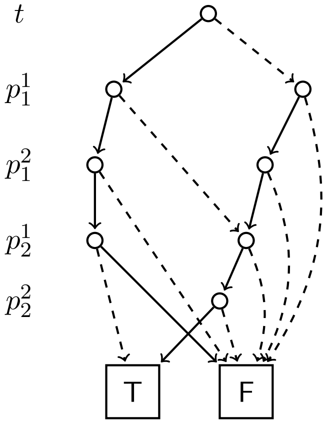
Multi-valued Decision Diagram (MDD)

DAG with a fixed variable ordering.

Reduction rules:

1. Each node represented only once
2. Nodes whose children are all the same are ommited

t	p_1	p_2
l_1	l_1	t
l_2	l_1	l_1
l_1	t	l_1

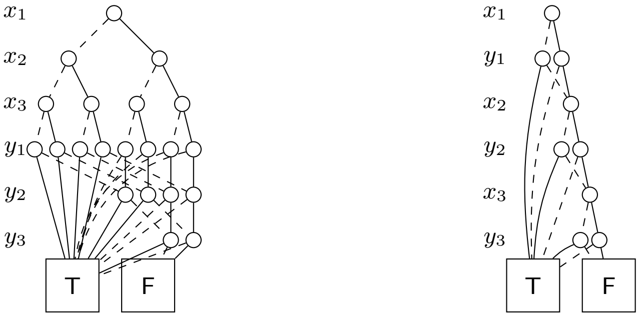


Binary Decision Diagrams: MDDs where variables are all binary
→Compilation that uses $log_2|D_v|$ binary variables per FDR variable v

BDD Variable Ordering

$$(x_1 \neq y_1) \vee (x_2 \neq y_2) \vee (x_3 \neq y_3)$$

Exponential ($> 2^{n+1}$) | Polynomial ($3n + 2$)



- Static Variable Ordering: Put causally-related variables close [KE11]
- Dynamic Variable Ordering: variable re-ordering to minimize the size of the BDDs generated so far

Uses of Decision Diagrams in Classical Planning

- Representation of state-dependent action costs [GKM15]
- Subsumption of partial states [AFB14]
- Dominance pruning [TH15]

Uses of Decision Diagrams in Classical Planning

- Representation of state-dependent action costs [GKM15]
- Subsumption of partial states [AFB14]
- Dominance pruning [TH15]

→Here: symbolic search

Agenda

Planning Actions as Logical Formulas

Transition Relation: represents an action a as the relation (set of pairs of states) containing (s, s') where a is applicable in s resulting in s'.

Planning Actions as Logical Formulas

Transition Relation: represents an action a as the relation (set of pairs of states) containing (s, s') where a is applicable in s resulting in s'.

$load(p_1, l_1)$: $pre : \{\langle t, l_1\rangle, \langle p_1, l_1\rangle\}$ and $eff : \{\langle p_1, t\rangle\}$ (prevail: $\{\langle t, l_1\rangle\}$)

$$\langle p_1, l_1\rangle \wedge \langle t, l_1\rangle \wedge \langle p_1, t\rangle' \wedge \langle t, l_1\rangle' \wedge (\langle p_2, l_1\rangle \leftrightarrow \langle p_2, l_1\rangle') \wedge (\langle p_2, l_2\rangle \leftrightarrow \langle p_2, l_2\rangle') \ldots$$

Planning Actions as Logical Formulas

Transition Relation: represents an action a as the relation (set of pairs of states) containing (s, s') where a is applicable in s resulting in s'.

$load(p_1, l_1)$: $pre : \{\langle t, l_1\rangle, \langle p_1, l_1\rangle\}$ and $eff : \{\langle p_1, t\rangle\}$ (prevail: $\{\langle t, l_1\rangle\}$)

$$\langle p_1, l_1\rangle \wedge \langle t, l_1\rangle \wedge \langle p_1, t\rangle' \wedge \langle t, l_1\rangle' \wedge (\langle p_2, l_1\rangle \leftrightarrow \langle p_2, l_1\rangle') \wedge (\langle p_2, l_2\rangle \leftrightarrow \langle p_2, l_2\rangle') \ldots$$

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states $\exists x \ . \ S(x) \wedge T(x, x')[x' \leftrightarrow x]$

Planning Actions as Logical Formulas

Transition Relation: represents an action a as the relation (set of pairs of states) containing (s, s') where a is applicable in s resulting in s'.

$load(p_1, l_1)$: $pre : \{\langle t, l_1\rangle, \langle p_1, l_1\rangle\}$ and $eff : \{\langle p_1, t\rangle\}$ (prevail: $\{\langle t, l_1\rangle\}$)

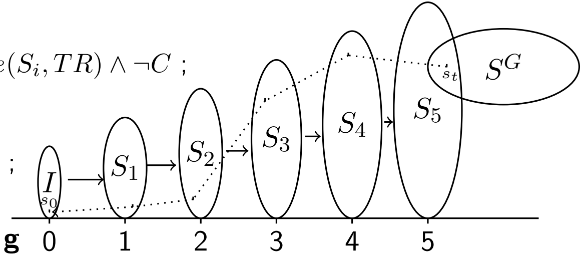
$$\langle p_1, l_1\rangle \wedge \langle t, l_1\rangle \wedge \langle p_1, t\rangle' \wedge \langle t, l_1\rangle' \wedge (\langle p_2, l_1\rangle \leftrightarrow \langle p_2, l_1\rangle') \wedge (\langle p_2, l_2\rangle \leftrightarrow \langle p_2, l_2\rangle') \ldots$$

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states $\exists x \, . \, S(x) \wedge T(x, x')[x' \leftrightarrow x]$

Pre-image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the predecessor states $\exists x' \, . \, S(x)[x' \leftrightarrow x] \wedge T(x, x')$

Symbolic Breadth-First Search

Input: Planning Task $\Pi = (V, A, I, G)$
$S_0 \leftarrow I$;
$C \leftarrow \emptyset$;
$i \leftarrow 0$;
while $S_i \neq \emptyset$ **do**
  if $S_i \wedge G$ **then**
    return Plan ;
  end
  $C \leftarrow C \vee S_i$;
  $S_{i+1} \leftarrow image(S_i, TR) \wedge \neg C$;
  $i \leftarrow i + 1$;
end
return Unsolvable ;



Symbolic Uniform-Cost Search

Expand set of states S_i with minimum g-value i

- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)

Symbolic Uniform-Cost Search

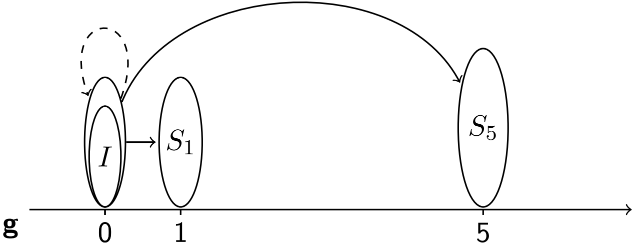
Expand set of states S_i with minimum g-value i

- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)

Symbolic Uniform-Cost Search

Expand set of states S_i with minimum g-value i

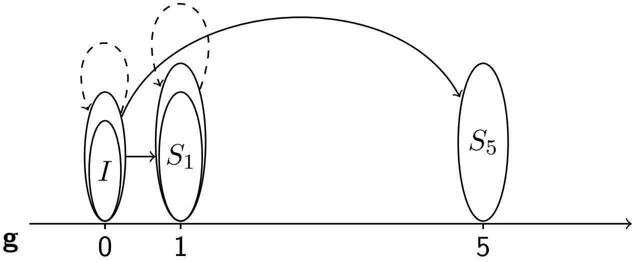
- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)



Symbolic Uniform-Cost Search

Expand set of states S_i with minimum g-value i

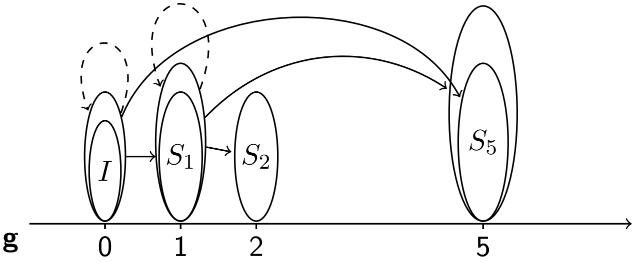
- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)



Symbolic Uniform-Cost Search

Expand set of states S_i with minimum g-value i

- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)



Symbolic Backward Uniform-Cost Search

We can perform the search in backward direction:

- Start with the set of goal states
- Use pre-image instead of image operation

Challenges:

1. Multiple goal states
2. Subsumption of partial states
3. Spurious states

Symbolic Backward Uniform-Cost Search

We can perform the search in backward direction:

- Start with the set of goal states
- Use pre-image instead of image operation

Challenges:

1. Multiple goal states $\rightarrow$Not a problem in symbolic search!
2. Subsumption of partial states $\rightarrow$Not a problem in symbolic search!
3. Spurious states

Symbolic Backward Uniform-Cost Search

We can perform the search in backward direction:

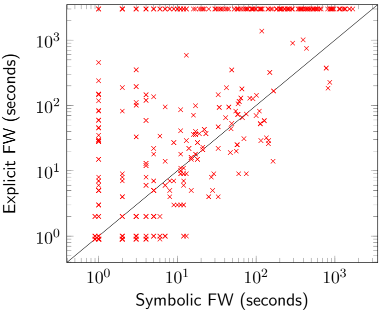
- Start with the set of goal states
- Use pre-image instead of image operation

Challenges:

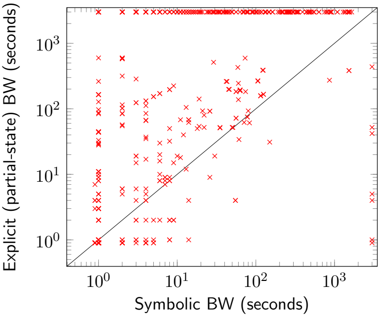
1. Multiple goal states →Not a problem in symbolic search!
2. Subsumption of partial states →Not a problem in symbolic search!
3. Spurious states →Solution: state-invariant pruning [TAKE17]
   - Compute state invariants, e.g., h^2 mutexes
   - Encode the set of spurious states as a BDD
   - Remove spurious states from the goal and the TRs

This is a Symbolic (Partial) PDB

Symbolic Uniform-Cost Search: Results



Forward



Backward

Symbolic Bidirectional Uniform-Cost Search

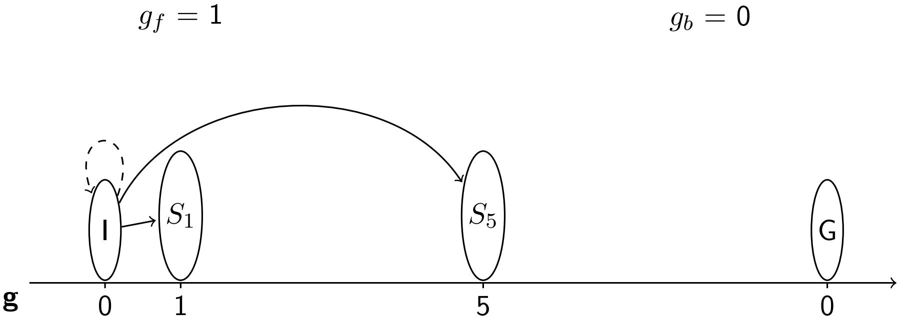
- Do forward and backward search at the same time
- Decide forward or backward direction at each step

Symbolic Bidirectional Uniform-Cost Search

- Do forward and backward search at the same time
- Decide forward or backward direction at each step

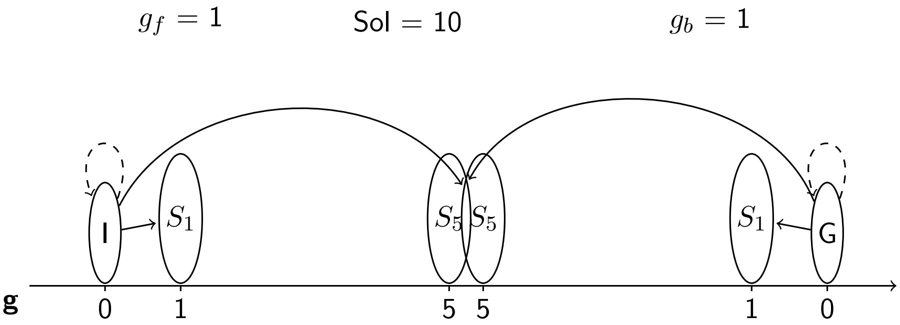
Symbolic Bidirectional Uniform-Cost Search

- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Stop when $g_f + g_b + min_{a \in A} c(a) \geq Sol$



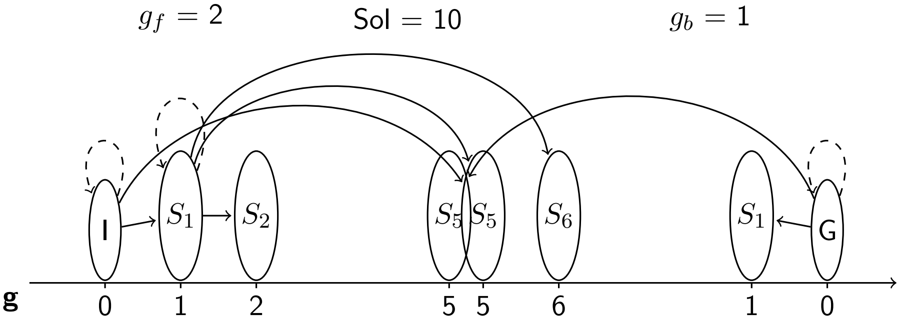
Symbolic Bidirectional Uniform-Cost Search

- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Stop when $g_f + g_b + min_{a \in A} c(a) \geq Sol$

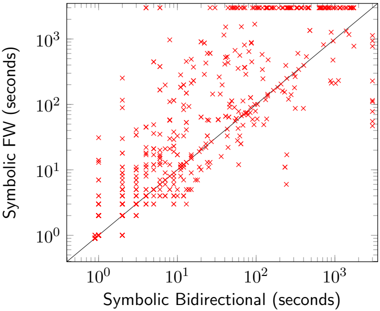


Symbolic Bidirectional Uniform-Cost Search

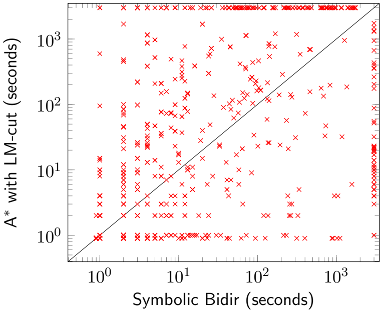
- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Stop when $g_f + g_b + min_{a \in A} c(a) \geq Sol$



Symbolic Bidirectional Uniform-Cost Search: Results



vs Symbolic Forward



vs A* with LM-cut

Agenda

1. Symbolic Representation of Planning Tasks
2. Symbolic Search
3. Advantages And Limitations of Symbolic Search
4. Conclusions

Advantages

- Optimal Planning: Finding shortest paths in a graph
  $\rightarrow$ Takes advantage of the structure of the planning task implicitly

Advantages

- Optimal Planning: Finding shortest paths in a graph

  $\rightarrow$ Takes advantage of the structure of the planning task implicitly

- Exhaustive (large-scale) search: computes shortest distance *from any given set of states* to *all states*. Beyond optimal planning:
  - Prove unsolvability
  - Generate symbolic PDBs
  - Generate all reachable states (e.g. for sampling)

Advantages

- Optimal Planning: Finding shortest paths in a graph
  - $\rightarrow$ Takes advantage of the structure of the planning task implicitly
- Exhaustive (large-scale) search: computes shortest distance *from any given set of states* to *all states*. Beyond optimal planning:
  - Prove unsolvability
  - Generate symbolic PDBs
  - Generate all reachable states (e.g. for sampling)
- Very flexible problem specification:
  - Conditional effects
  - Disjunctive preconditions

Limitations

- BDD representation: difficult to predict when the BDD representation will be compact. Typically not good in
  - permutation puzzles (N-puzzle, Spider?)
  - representing adjacent cells in grids (Connect-4, Snake)

Limitations

- BDD representation: difficult to predict when the BDD representation will be compact. Typically not good in
  - permutation puzzles (N-puzzle, Spider?)
  - representing adjacent cells in grids (Connect-4, Snake)
- Heuristics/pruning methods must be adapted to leverage the symbolic representation
  - → Heuristics split sets of states according to their heuristic value, which may be detrimental
  - → It is not easy to complement the perimeter

Open Challenges

- Integrate heuristics:
  - → Develop abstraction methods specifically designed for symbolic bidirectional search
  - → Can additive PDBs be used for symbolic search?

Open Challenges

- Integrate heuristics:
  - → Develop abstraction methods specifically designed for symbolic bidirectional search
  - → Can additive PDBs be used for symbolic search?
- BDD Partitioning: split a BDD in parts
  - → For distributed computing, external search, etc.

Open Challenges

- Integrate heuristics:
  - → Develop abstraction methods specifically designed for symbolic bidirectional search
  - → Can additive PDBs be used for symbolic search?
- BDD Partitioning: split a BDD in parts
  - → For distributed computing, external search, etc.
- Parallel computation:
  - → BDD packages that run in multiple cores/GPUs are being developed

Agenda

1. Symbolic Representation of Planning Tasks
2. Symbolic Search
3. Advantages And Limitations of Symbolic Search
4. Conclusions

Conclusions

- Symbolic search is useful for exhaustive exploration of classical planning task

Conclusions

- Symbolic search is useful for exhaustive exploration of classical planning task
- Advantages:
  - Compact representation of sets of states
  - Time/memory efficient exploration of state spaces

  → Takes advantage of the structure of the planning task implicitly

Conclusions

- Symbolic search is useful for exhaustive exploration of classical planning task
- Advantages:
  - Compact representation of sets of states
  - Time/memory efficient exploration of state spaces

  $\rightarrow$ Takes advantage of the structure of the planning task implicitly
- Limitations:
  - Heuristics/pruning methods must be adapted to leverage the symbolic representation

References I

[AFB14] Vidal Alcázar, Susana Fernández, and Daniel Borrajo, *Analyzing the impact of partial states on duplicate detection and collision of frontiers*, Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14) (Steve Chien, Minh Do, Alan Fern, and Wheeler Ruml, eds.), AAAI Press, 2014.

[Bry86] Randal E. Bryant, *Graph-based algorithms for boolean function manipulation*, IEEE Transactions on Computers **35** (1986), no. 8, 677–691.

[Dar01] Adnan Darwiche, *Decomposable negation normal form*, Journal of the Association for Computing Machinery **48** (2001), no. 4, 608–647.

[Dar02] ______, *A compiler for deterministic decomposable negation normal form*, Proceedings of the 18th National Conference of the American Association for Artificial Intelligence (AAAI'02) (Edmonton, AL, Canada) (Rina Dechter, Michael Kearns, and Richard S. Sutton, eds.), AAAI Press, July 2002, pp. 627–634.

References II

[Dar11] ______, *SDD: A new canonical representation of propositional knowledge bases*, Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11) (Toby Walsh, ed.), AAAI Press/IJCAI, 2011, pp. 819–826.

[GKM15] Florian Geißer, Thomas Keller, and Robert Mattmüller, *Delete relaxations for planning with state-dependent action costs*, Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15) (Qiang Yang, ed.), AAAI Press/IJCAI, 2015.

[KE11] Peter Kissmann and Stefan Edelkamp, *Improving cost-optimal domain-independent symbolic planning*, Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11) (San Francisco, CA, USA) (Wolfram Burgard and Dan Roth, eds.), AAAI Press, July 2011, pp. 992–997.

[Min93] Shin-ichi Minato, *Zero-suppressed bdds for set manipulation in combinatorial problems*, Proceedings of the 30th Design Automation Conference. Dallas, Texas, USA, June 14-18, 1993 (Alfred E. Dunlop, ed.), ACM Press, 1993, pp. 272–277.

References III

[TAKE17] Álvaro Torralba, Vidal Alcázar, Peter Kissmann, and Stefan Edelkamp, *Efficient symbolic search for cost-optimal planning*, Artificial Intelligence **242** (2017), 52–79.

[TH15] Álvaro Torralba and Jörg Hoffmann, *Simulation-based admissible dominance pruning*, Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15) (Qiang Yang, ed.), AAAI Press/IJCAI, 2015, pp. 1689–1695.