Alternatives to Explicit State Space Search
Symbolic Search

Álvaro Torralba & Daniel Gnad

SAARLAND UNIVERSITY
SAARBRÜCKEN GRADUATE SCHOOL OF COMPUTER SCIENCE

June 19, 2017

About us

Dr. Álvaro Torralba

Daniel Gnad

Saarland University, Saarbrücken, Germany

About you

Target audience:

Ideally, you are..

- .. familiar with Classical Planning Formalisms (FDR/SAS$^+$).
- .. familiar with Planning as Heuristic Search.
- .. aware of an important issue in Explicit State Space Search
  $\rightarrow$ State Space Explosion

About you

Target audience:

Ideally, you are..

- .. familiar with Classical Planning Formalisms (FDR/SAS$^+$).
- .. familiar with Planning as Heuristic Search.
- .. aware of an important issue in Explicit State Space Search
  $\rightarrow$ State Space Explosion

Don't hesitate to ask questions if something is unclear!

About the tutorial

Symbolic Search:

There have been many tutorials on the usefulness of Decision Diagrams:
$\rightarrow$Here: focus on symbolic search algorithms

Agenda

Agenda

Classical Planning

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

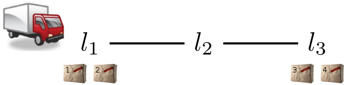
- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v*.*
- A *is a set of actions; each* $a \in A$ *is a triple* (pre_a, eff_a, c_a)*, of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}^{0+}$*.*
- *Initial state* I *(complete assignment), 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 $D_{p_i} = \{t, l_1, l_2, l_3\}$.
- $A = \{load(p_i, x), unload(p_i, x), drive(x, x')\}$

Semantics – The State Space of a Planning Task

Definition. *Let $\Pi = (V, A, I, G)$ be an FDR planning task. The state space of Π is the labeled transition system $\Theta_\Pi = (S, L, c, T, I, S^G)$ where:*

- *The states S are the complete variable assignments.*
- *The labels $L = A$ are Π's actions; the cost function c is that of Π.*
- *The transitions are $T = \{s \xrightarrow{a} s' \mid pre_a \subseteq s, s' = s[\![a]\!]\}$.*

  If $pre_a \subseteq s$, then a is applicable in s and, for all $v \in V$, $s[\![a]\!][v] := eff_a[v]$ if $eff_a[v]$ is defined and $s[\![a]\!][v] := s[v]$ otherwise.

  If $pre_a \not\subseteq s$, then $s[\![a]\!]$ is undefined.
- *The initial state I is identical to that of Π.*
- *The goal states $S^G = \{s \in S \mid G \subseteq s\}$ are those that satisfy Π's goal.*

Semantics – The State Space of a Planning Task

Definition. *Let $\Pi = (V, A, I, G)$ be an FDR planning task. The state space of Π is the labeled transition system $\Theta_\Pi = (S, L, c, T, I, S^G)$ where:*

- *The states S are the complete variable assignments.*
- *The labels $L = A$ are Π's actions; the cost function c is that of Π.*
- *The transitions are $T = \{s \xrightarrow{a} s' \mid pre_a \subseteq s, s' = s[\![a]\!]\}$.*

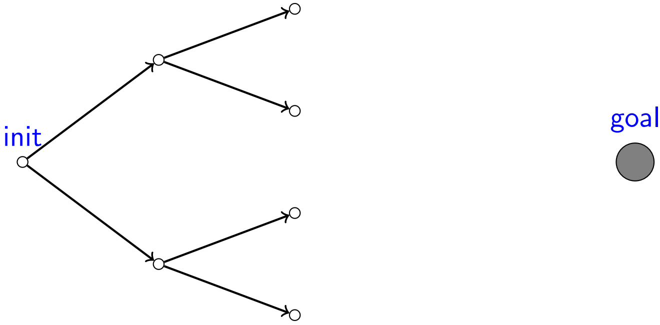
  If $pre_a \subseteq s$, then a is applicable in s and, for all $v \in V$, $s[\![a]\!][v] := \mathit{eff}_a[v]$ if $\mathit{eff}_a[v]$ is defined and $s[\![a]\!][v] := s[v]$ otherwise.

  If $pre_a \not\subseteq s$, then $s[\![a]\!]$ is undefined.
- *The initial state I is identical to that of Π.*
- *The goal states $S^G = \{s \in S \mid G \subseteq s\}$ are those that satisfy Π's goal.*

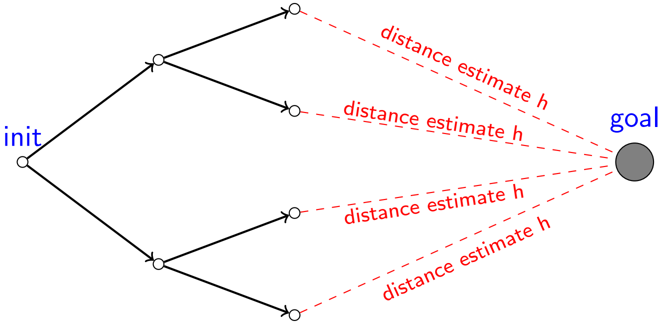
→ Solution ("Plan"): Action sequence mapping I into $s \in S^G$.
Optimal plan: Minimum summed-up cost.

A successful approach: Heuristic Search

A successful approach: Heuristic Search



A successful approach: Heuristic Search

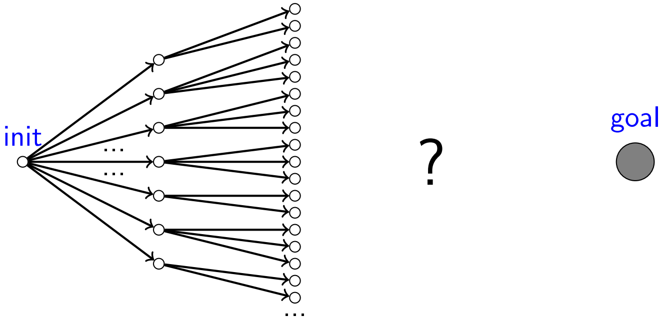


→ Forward state space search. Heuristic function h maps states s to an estimate $h(s)$ of goal distance.

Alternatives to State Space Search (not covered here)

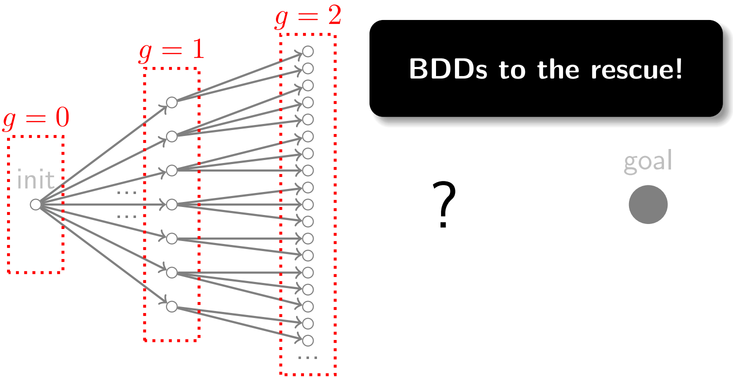
- **Planning as SAT**: Extensions use, e. g., heuristics, symmetry breaking. [KS92, KS96, EMW97, Rin98, Rin03, Rin12]
- **Property Directed Reachability** [Bra11, EMB11, Sud14]
- Planning via **Petri Net Unfolding** [GW91, McM92, ERV02, ELL04, HRTW07, BHHT08, BHK$^+$14]
- **Partial-order Planning** [Sac75, KKY95, YS03, BGB13]
- **Factored Planning** [Kno94, AE03, BD06, KBHT07, BD08, BD13, FJHT10]
- ...

State Space Explosion



Huge branching factor → state space *explosion*

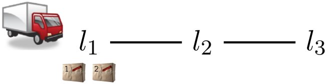
State Space Explosion



Huge branching factor $\rightarrow$ state space *explosion*

Agenda

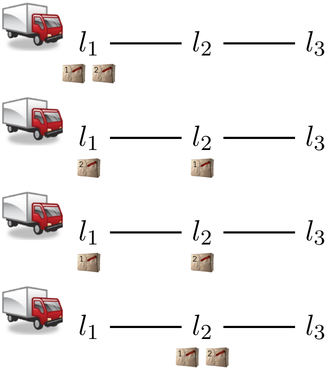
Sets of States as Logical Formulas



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

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

Sets of States as Logical Formulas



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

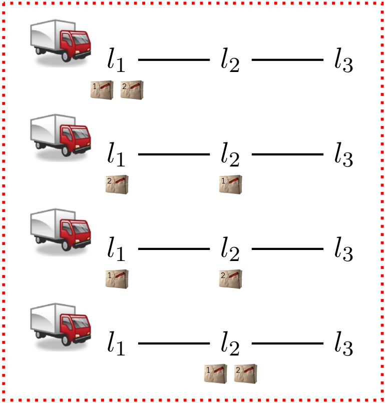
$$\langle t, l_1 \rangle \wedge \langle p_1, l_2 \rangle \wedge \langle p_2, l_1 \rangle$$

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

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

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

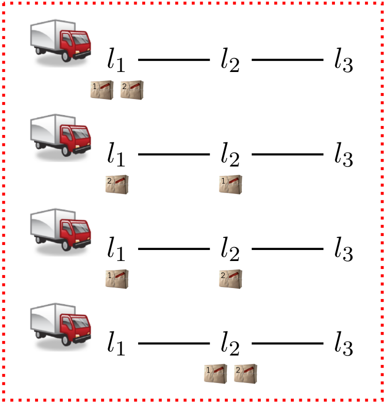
Sets of States as Logical Formulas



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

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: $\langle t, l_1\rangle \leftrightarrow (\neg\langle t, l_2\rangle \wedge \neg\langle t, l_3\rangle), \ldots$

Operating with Sets of States as Logical Formulas

Sets	Logic
Empty set	$\bot$
All states	$\top$
All states in which the truck is at l_1	$\langle t, l_1 \rangle$

Operating with Sets of States as Logical Formulas

Sets	Logic
Empty set	$\bot$
All states	$\top$
All states in which the truck is at l_1	$\langle t, l_1 \rangle$
Union ($\cup$)	Disjunction ($\vee$)
Inersection ($\cap$)	Conjunction ($\wedge$)
Complement	Negation ($\neg$)

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.

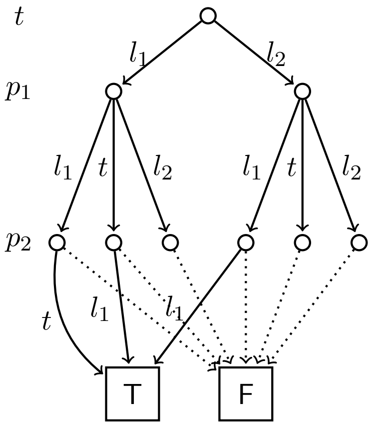
P: polynomial in the size of the representation

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)

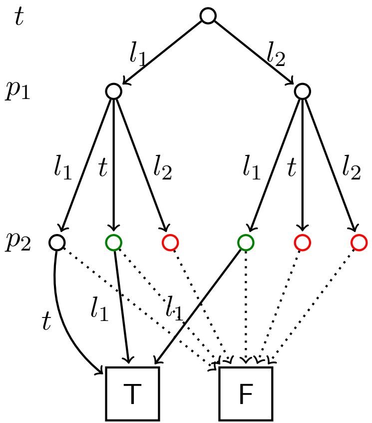
Multi-valued Decision Diagram (MDD)

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)

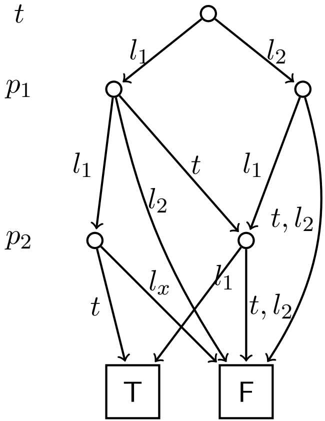
Multi-valued Decision Diagram (MDD)

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)

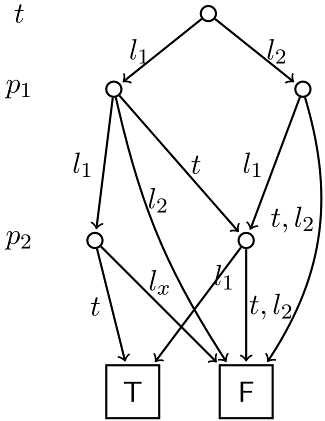
Multi-valued Decision Diagram (MDD)

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
$\rightarrow$Compilation that uses $log_2|D_v|$ binary variables per FDR variable v

Binary Decision Diagrams (BDDs)

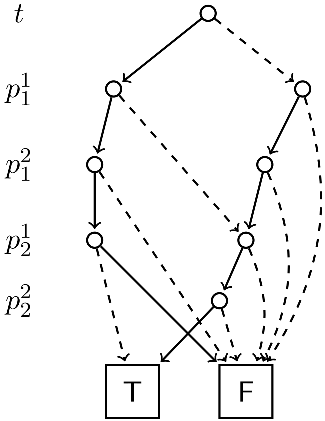
Multi-valued Decision Diagram (MDD)

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

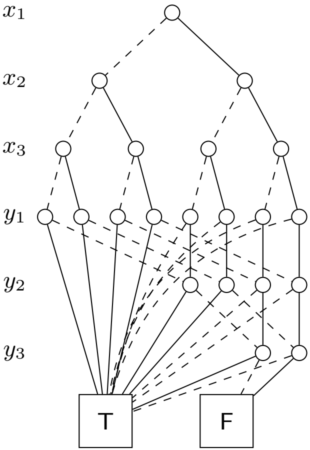


Binary Decision Diagrams: MDDs where variables are all binary
$\rightarrow$Compilation that uses $log_2|D_v|$ binary variables per FDR variable v

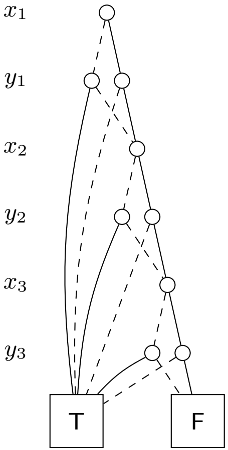
BDD Variable Ordering

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

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



Polynomial ($3n + 2$)



Practical Strategies for a Good Variable Ordering

Static Variable Ordering: Put causally-related variables close [KE11]

$$\text{Choose ordering } o \text{ that minimizes } \sum_{v_i, v_j \in CG} d_o(v_i, v_j)^2$$

$\rightarrow$No strong theoretical guarantees [KH13] but compares well against other alternatives [BRKM91, CHP93, Mai09, MWBSV88, MIY90]

Practical Strategies for a Good Variable Ordering

Static Variable Ordering: Put causally-related variables close [KE11]

$$\text{Choose ordering } o \text{ that minimizes } \sum_{v_i, v_j \in CG} d_o(v_i, v_j)^2$$

→No strong theoretical guarantees [KH13] but compares well against other alternatives [BRKM91, CHP93, Mai09, MWBSV88, MIY90]

Dynamic Variable Ordering: variable re-ordering to minimize the size of the BDDs generated so far

- Finding the optimal BDD ordering is NP-hard [Bry86]
- But practical approximations (based on local-search) exist [Rud93].
- Applied in planning with good results by dynamic-Gamer [KH14]

Complexity Results

BDD Representation of Interesting Sets of States [EK11]

Goal States/Reachable states	Polynomial	Exponential
Polynomial	Gripper	Blocksworld N-puzzle
Exponential	Connect-4 Tic-tac-toe Gomoku	

Complexity Results

BDD Representation of Interesting Sets of States [EK11]

Goal States/Reachable states	Polynomial	Exponential
Polynomial	Gripper	Blocksworld N-puzzle
Exponential	Connect-4 Tic-tac-toe Gomoku	

Can variable orderings schemas based on the causal graph give us theoretical guarantees for the size of BDDs in the search? [KH14]
$\rightarrow$Mostly not.

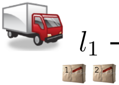
Planning Actions as Logical Formulas

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'.

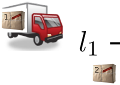
Planning Actions as Logical Formulas

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 $e\!f\!f : \{\langle p_1, t\rangle\}$ (prevail: $\{\langle t, l_1\rangle\}$)



$l_1 - l_2 - l_3$



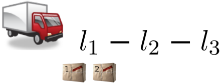
$l_1 - l_2 - l_3$

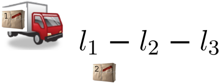
$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_1\rangle \wedge$
$\langle t, l_1\rangle' \wedge \langle p_1, t\rangle' \wedge \langle p_2, l_1\rangle'$

Planning Actions as Logical Formulas

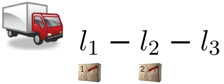
Transition Relation: represents an action a as the relation (set of pairs of states) containing (s, s') where a is applicable in s resulting in s'.

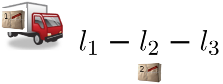
$load(p_1, l_1)$: $pre : \{\langle t, l_1\rangle, \langle p_1, l_1\rangle\}$ and $eff : \{\langle p_1, t\rangle\}$ (prevail: $\{\langle t, l_1\rangle\}$)



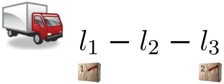


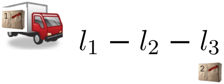
$\langle t, l_1\rangle \land \langle p_1, l_1\rangle \land \langle p_2, l_1\rangle \land$
$\langle t, l_1\rangle' \land \langle p_1, t\rangle' \land \langle p_2, l_1\rangle'$





$\langle t, l_1\rangle \land \langle p_1, l_1\rangle \land \langle p_2, l_2\rangle \land$
$\langle t, l_1\rangle' \land \langle p_1, t\rangle' \land \langle p_2, l_2\rangle'$



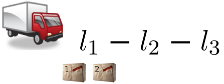


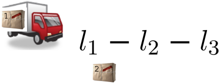
$\langle t, l_1\rangle \land \langle p_1, l_1\rangle \land \langle p_2, l_3\rangle \land$
$\langle t, l_1\rangle' \land \langle p_1, t\rangle' \land \langle p_2, l_3\rangle'$

Planning Actions as Logical Formulas

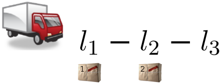
Transition Relation: represents an action a as the relation (set of pairs of states) containing (s, s') where a is applicable in s resulting in s'.

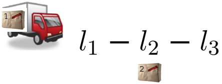
$load(p_1, l_1)$: $pre : \{\langle t, l_1\rangle, \langle p_1, l_1\rangle\}$ and $eff : \{\langle p_1, t\rangle\}$ (prevail: $\{\langle t, l_1\rangle\}$)



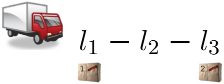


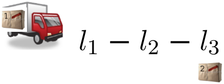
$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_1\rangle \wedge$
$\langle t, l_1\rangle' \wedge \langle p_1, t\rangle' \wedge \langle p_2, l_1\rangle'$





$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_2\rangle \wedge$
$\langle t, l_1\rangle' \wedge \langle p_1, t\rangle' \wedge \langle p_2, l_2\rangle'$





$\langle t, l_1\rangle \wedge \langle p_1, l_1\rangle \wedge \langle p_2, l_3\rangle \wedge$
$\langle t, l_1\rangle' \wedge \langle p_1, t\rangle' \wedge \langle p_2, l_3\rangle'$

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

Computing the Successors (Image Computation)

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

$$\text{image}(S(x), T(x, x')) = \exists x \, . \, S(x) \land T(x, x')[x' \leftrightarrow x]$$

Computing the Successors (Image Computation)

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

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

$S(x)$:

t	p_1	p_2
l_1	l_1	l_1
l_1	l_1	l_2
l_2	l_1	l_3
l_1	l_3	l_1

$T(x, x')$:
$(load(p_1, l_1))$

t	p_1	p_2	t'	p'_1	p'_2
l_1	l_1	l_1	l_1	t	l_1
l_1	l_1	l_2	l_1	t	l_2
l_1	l_1	l_3	l_1	t	l_3

Computing the Successors (Image Computation)

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

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

$S(x)$:

t	p_1	p_2
l_1	l_1	l_1
l_1	l_1	l_2
l_2	l_1	l_3
l_1	l_3	l_1

$T(x, x')$: $(load(p_1, l_1))$

t	p_1	p_2	t'	p'_1	p'_2
l_1	l_1	l_1	l_1	t	l_1
l_1	l_1	l_2	l_1	t	l_2
l_1	l_1	l_3	l_1	t	l_3

Result:

t	p_1	p_2	t'	p'_1	p'_2
l_1	l_1	l_1	l_1	t	l_1
l_1	l_1	l_2	l_1	t	l_2

Computing the Successors (Image Computation)

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

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

$S(x)$:

t	p_1	p_2
l_1	l_1	l_1
l_1	l_1	l_2
l_2	l_1	l_3
l_1	l_3	l_1

$T(x, x')$: $(load(p_1, l_1))$

t	p_1	p_2	t'	p'_1	p'_2
l_1	l_1	l_1	l_1	t	l_1
l_1	l_1	l_2	l_1	t	l_2
l_1	l_1	l_3	l_1	t	l_3

Result:

t	p_1	p_2	t'	p'_1	p'_2
			l_1	t	l_1
			l_1	t	l_2

Computing the Successors (Image Computation)

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

$$\text{image}(S(x), T(x, x')) = \exists x \, . \, S(x) \land T(x, x')[x' \leftrightarrow x]$$

$S(x)$:

t	p_1	p_2
l_1	l_1	l_1
l_1	l_1	l_2
l_2	l_1	l_3
l_1	l_3	l_1

$T(x, x')$: $(load(p_1, l_1))$

t	p_1	p_2	t'	p'_1	p'_2
l_1	l_1	l_1	l_1	t	l_1
l_1	l_1	l_2	l_1	t	l_2
l_1	l_1	l_3	l_1	t	l_3

Result:

t	p_1	p_2	t'	p'_1	p'_2
l_1	t	l_1			
l_1	t	l_2			

Computing the Successors (Image Computation)

Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

$S(x)$:

t	p_1	p_2
l_1	l_1	l_1
l_1	l_1	l_2
l_2	l_1	l_3
l_1	l_3	l_1

$T(x, x')$: $(load(p_1, l_1))$

t	p_1	p_2	t'	p'_1	p'_2
l_1	l_1	l_1	l_1	t	l_1
l_1	l_1	l_2	l_1	t	l_2
l_1	l_1	l_3	l_1	t	l_3

Result:

t	p_1	p_2	t'	p'_1	p'_2
l_1	t	l_1			
l_1	t	l_2			

$\exists$-quantification: exponential in the number of variables

Computing the Predecessors (Pre-Image Computation)

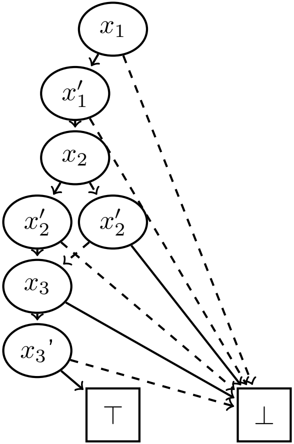
Image: Given a set of states $S(x)$ and a TR $T(x,x')$ generate the predecessor states

$$\text{pre-image}(S(x), T(x,x')) = \exists x' \, . \, S(x)[x' \leftrightarrow x] \wedge T(x,x')$$

- Corresponds to regression [Rin08]

Efficient Image Computation: Variable Ordering

Variable Ordering: Interleave variables x and x'



$\rightarrow$The TR of an action has linear size on the number of variables

Efficient Image Computation

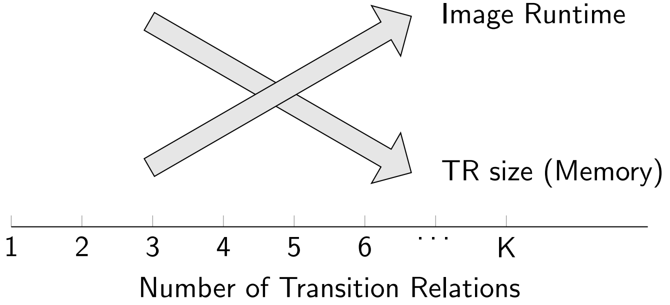
Transition Relation Partitioning [BCL91, JVB08]

Given a set of K actions with the same cost, replace $T_i(x, x')$ and $T_j(x, x')$ by $T_i(x, x') \vee T_j(x, x')$

Efficient Image Computation

Transition Relation Partitioning [BCL91, JVB08]

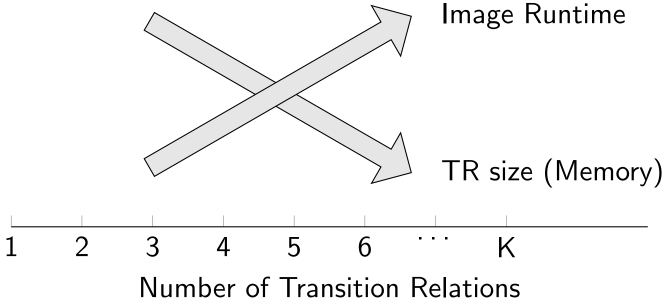
Given a set of K actions with the same cost, replace $T_i(x, x')$ and $T_j(x, x')$ by $T_i(x, x') \vee T_j(x, x')$



Efficient Image Computation

Transition Relation Partitioning [BCL91, JVB08]

Given a set of K actions with the same cost, replace $T_i(x, x')$ and $T_j(x, x')$ by $T_i(x, x') \vee T_j(x, x')$



→Merge as many TRs as possible given the available memory [TEK13, TAKE17]

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

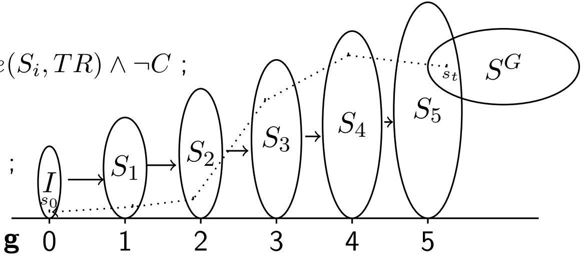
- Representation of state-dependent action costs [GKM15]
- Subsumption of partial states [AFB14]
- Dominance pruning [TH15]

→Here: symbolic search

Agenda

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

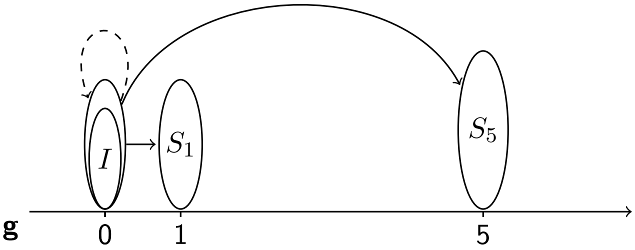
Expand set of states S_i with minimum g-value i

- 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

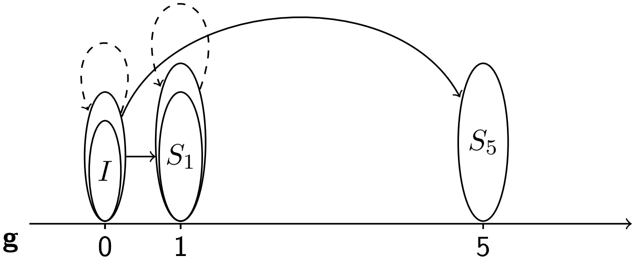
- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)



Symbolic Uniform-Cost Search

Expand set of states S_i with minimum g-value i

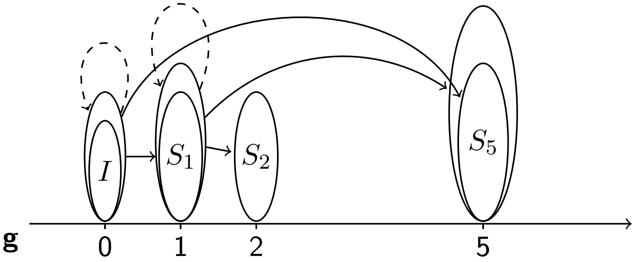
- Zero-cost breadth-first search to obtain all states reachable with $g = i$
- For each TR with action cost c:
  - Use image to compute states reachable with $i + c$
  - Insert the result in the corresponding bucket (disjunction)



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

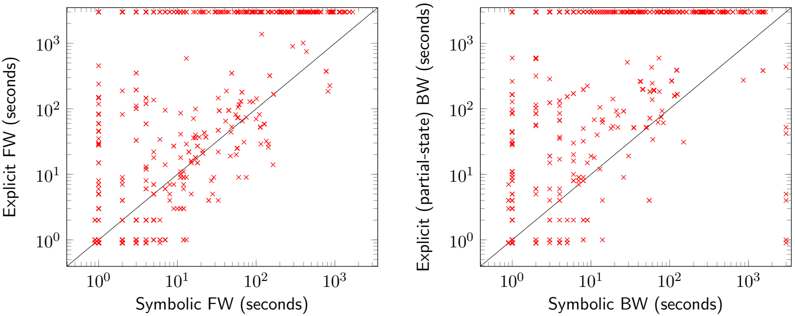
We can perform the search in backward direction:

- 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

Symbolic Uniform-Cost Search: Results



Forward Backward

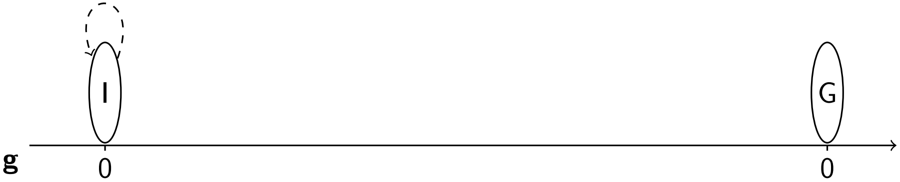
Symbolic Bidirectional Uniform-Cost Search

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

Symbolic Bidirectional Uniform-Cost Search

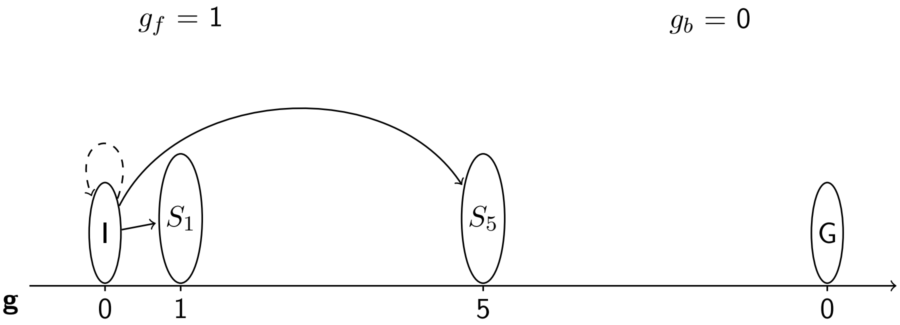
- Do forward and backward search at the same time
- Decide forward or backward direction at each step

$g_f = 0$ $g_b = 0$



Symbolic Bidirectional Uniform-Cost Search

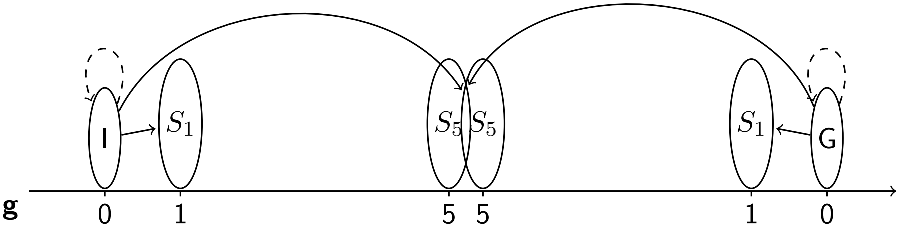
- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Stop when $g_f + g_b + min_{a \in A} c(a) \geq Sol$



Symbolic Bidirectional Uniform-Cost Search

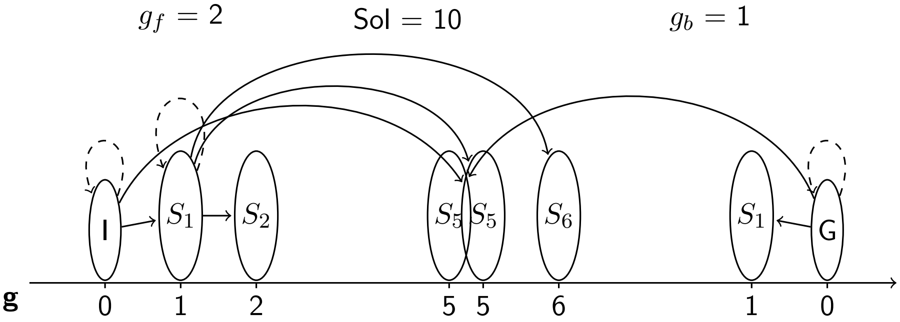
- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Stop when $g_f + g_b + min_{a \in A} c(a) \geq Sol$

$g_f = 1$ Sol $= 10$ $g_b = 1$

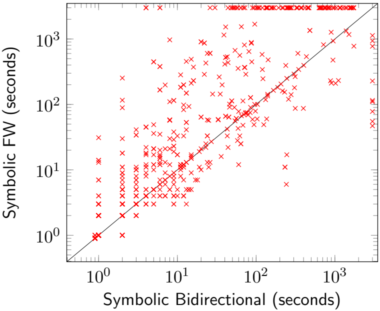


Symbolic Bidirectional Uniform-Cost Search

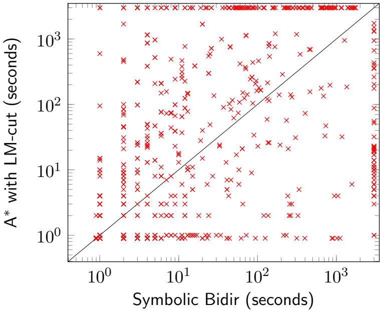
- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Stop when $g_f + g_b + min_{a \in A} c(a) \geq Sol$



Symbolic Bidirectional Uniform-Cost Search: Results



vs Symbolic Forward



vs A* with LM-cut

Agenda

Planning Heuristics

Definition *A* *heuristic* h *is a function* $h : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$*. Its value* $h(s)$ *for a state* s *is referred to as the state's* *heuristic value**, or* h *value.*

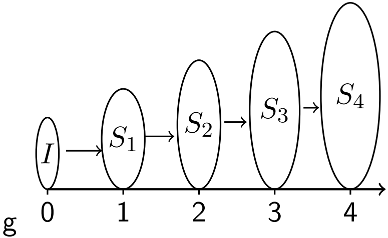
Planning Heuristics

Definition *A heuristic* h *is a function* $h : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$*. Its value* $h(s)$ *for a state* s *is referred to as the state's heuristic value, or* h *value.*

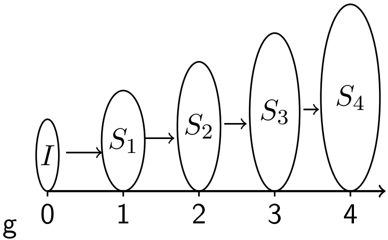
Definition *For a state* $s \in S$*, the perfect heuristic value* h^* *of* s *is the cost of an optimal plan for* s*, or* ∞ *if there exists no plan for* s*.*

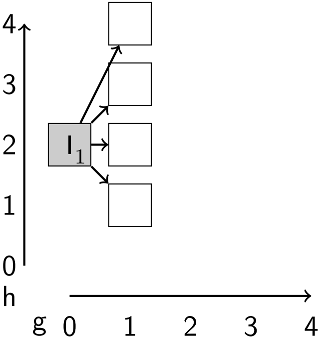
$\rightarrow$ Heuristic functions h estimate the remaining cost h^*.

How to Exploit Heuristics in Symbolic Search?



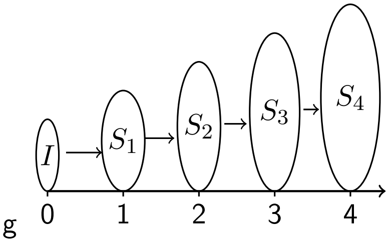
How to Exploit Heuristics in Symbolic Search?

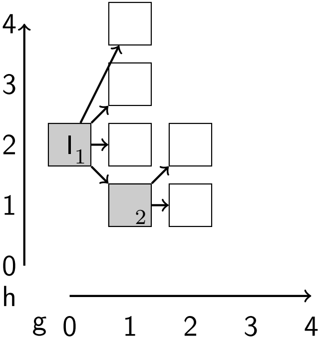




Split a BDD into subsets of states according to their h-value!

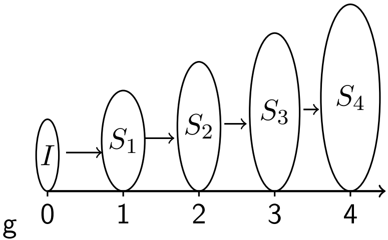
How to Exploit Heuristics in Symbolic Search?

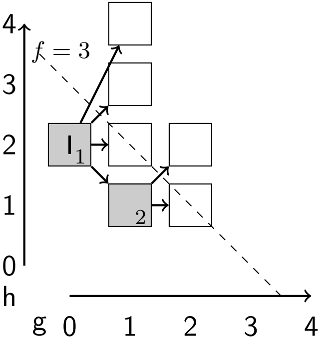




Split a BDD into subsets of states according to their h-value!

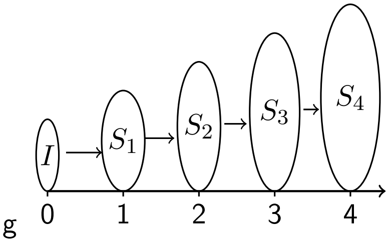
How to Exploit Heuristics in Symbolic Search?

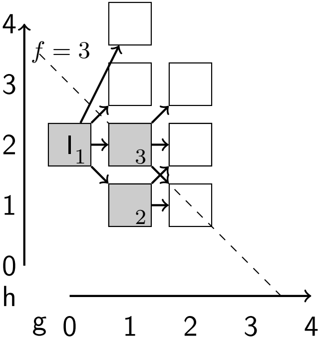




Split a BDD into subsets of states according to their h-value!

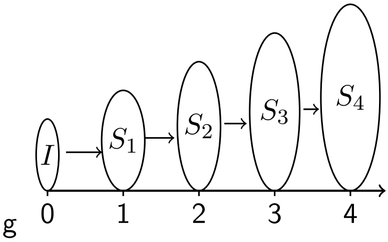
How to Exploit Heuristics in Symbolic Search?

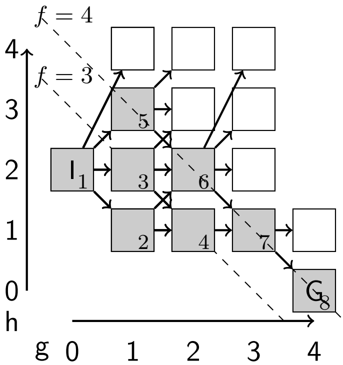




Split a BDD into subsets of states according to their h-value!

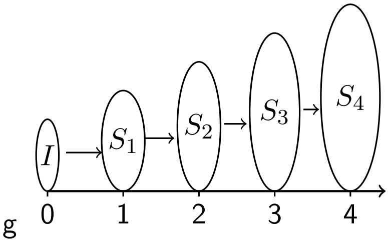
How to Exploit Heuristics in Symbolic Search?

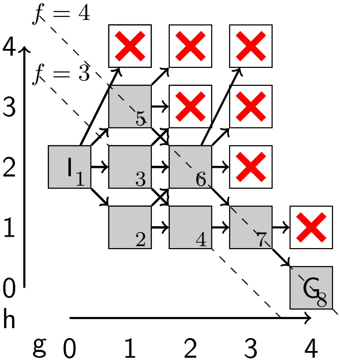




Split a BDD into subsets of states according to their h-value!

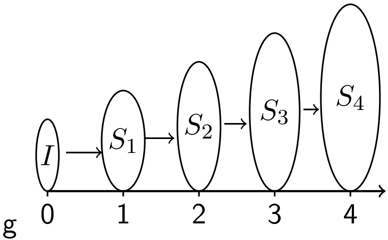
How to Exploit Heuristics in Symbolic Search?

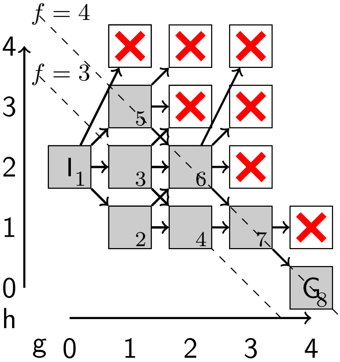




Split a BDD into subsets of states according to their h-value!

How to Exploit Heuristics in Symbolic Search?





Split a BDD into subsets of states according to their h-value!

- Heuristic computation: how to evaluate a set of states?
- Does the heuristic improve the search performance?

Heuristic Computation: How to Evaluate a Set of States?

1. Iterate over all states in the BDD, computing $h(s)$

Heuristic Computation: How to Evaluate a Set of States?

1. ~~Iterate over all states in the BDD, computing $h(s)$~~
2. Precompute the heuristic in form of BDDs: A BDD H_i for each possible h-value representing the set of states with $h(s) = i$

$$\boxed{H_0} \quad \boxed{H_1} \quad \boxed{H_2} \quad \boxed{H_3} \quad \boxed{H_4}$$

Heuristic Computation: How to Evaluate a Set of States?

1. ~~Iterate over all states in the BDD, computing $h(s)$~~
2. Precompute the heuristic in form of BDDs: A BDD H_i for each possible h-value representing the set of states with $h(s) = i$

Given a set of states S, split it according to their h-value

$$\boxed{H_0} \quad \boxed{H_1} \quad \boxed{H_2} \quad \boxed{H_3} \quad \boxed{H_4}$$

$$\boxed{S}$$

$$\boxed{S_0} \quad \boxed{S_1} \quad \boxed{S_2} \quad \boxed{S_3} \quad \boxed{S_4}$$

Heuristic Computation: How to Evaluate a Set of States?

1. ~~Iterate over all states in the BDD, computing $h(s)$~~
2. Precompute the heuristic in form of BDDs: A BDD H_i for each possible h-value representing the set of states with $h(s) = i$

Given a set of states S, split it according to their h-value: $S_i = S \wedge B_i$

$$\begin{array}{cccccc} & \boxed{H_0} & \boxed{H_1} & \boxed{H_2} & \boxed{H_3} & \boxed{H_4} \\ \boxed{S} & \wedge & \wedge & \wedge & \wedge & \wedge \\ & \boxed{S_0} & \boxed{S_1} & \boxed{S_2} & \boxed{S_3} & \boxed{S_4} \end{array}$$

Heuristic Computation: How to Evaluate a Set of States?

1. ~~Iterate over all states in the BDD, computing $h(s)$~~
2. Precompute the heuristic in form of BDDs: A BDD H_i for each possible h-value representing the set of states with $h(s) = i$

Given a set of states S, split it according to their h-value: $S_i = S \wedge B_i$

$$\begin{array}{cccccc} & \boxed{H_0} & \boxed{H_1} & \boxed{H_2} & \boxed{H_3} & \boxed{H_4} \\ \boxed{S} & \wedge & \wedge & \wedge & \wedge & \wedge \\ & \boxed{S_0} & \boxed{S_1} & \boxed{S_2} & \boxed{S_3} & \boxed{S_4} \end{array}$$

$\rightarrow$Can we efficiently precompute a heuristic into BDDs?

- Yes, for some types of abstraction heuristics
- Not in the general case. Finding tractable cases is an open research question!

Agenda

Abstractions

Abstraction: *function* $\alpha : S \mapsto S^\alpha$*.* Induces an abstract state space s.t.:

(i) $I^\alpha = \alpha(I)$.

(ii) $S^{\alpha G} = \{\alpha(s) \mid s \in S^G\}$. /* preserve goal states */

(iii) $T^\alpha = \{(\alpha(s), l, \alpha(t)) \mid (s, l, t) \in T\}$./* preserve transitions */

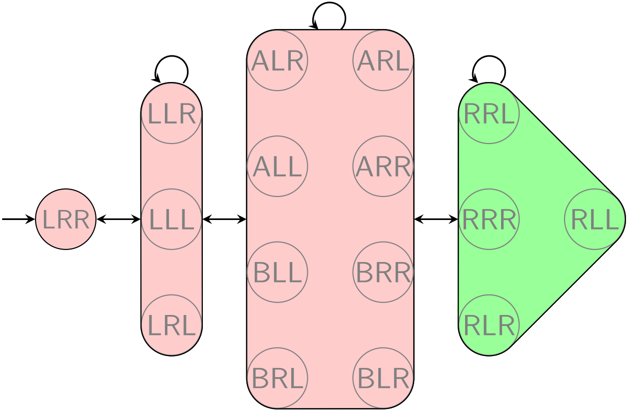
Abstractions

Abstraction: *function* $\alpha : S \mapsto S^\alpha$. Induces an abstract state space s.t.:

(i) $I^\alpha = \alpha(I)$.

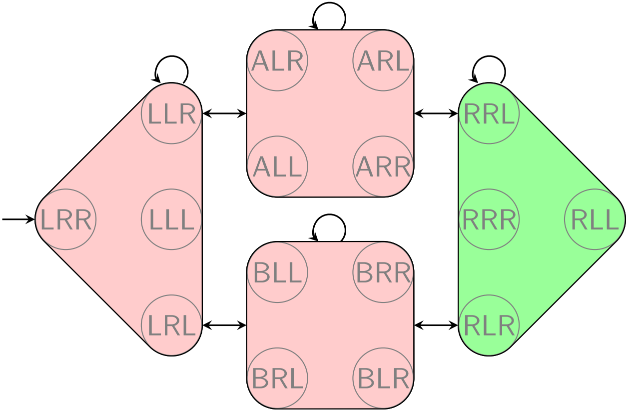
(ii) $S^{\alpha G} = \{\alpha(s) \mid s \in S^G\}$. /* preserve goal states */

(iii) $T^\alpha = \{(\alpha(s), l, \alpha(t)) \mid (s, l, t) \in T\}$./* preserve transitions */



Pattern Databases [CS98, Ede01, HBH+07]

Pattern Databases: Select a subset of variables $V^\alpha \subseteq V$ (pattern). The mapping α is defined as the projection onto V^α.



Abstraction Heuristics

Use the optimal goal-distance in the abstract state space as an (admissible) estimate for the distance in the concrete state space:

$$h(s) = h^*(\alpha(s))$$

Abstraction Heuristics

Use the optimal goal-distance in the abstract state space as an (admissible) estimate for the distance in the concrete state space:

$$h(s) = h^*(\alpha(s))$$

1. Precompute h^* for all $\alpha(s) \in S^\alpha$ by performing a backward uniform-cost search in the abstract state space
2. Store them in a look-up table

Abstraction Heuristics

Use the optimal goal-distance in the abstract state space as an (admissible) estimate for the distance in the concrete state space:

$$h(s) = h^*(\alpha(s))$$

1. Precompute h^* for all $\alpha(s) \in S^\alpha$ by performing a backward uniform-cost search in the abstract state space
   - $\rightarrow$ Searching the entire abstract state space? That's what symbolic search is good for!
2. Store them in a look-up table

Abstraction Heuristics

Use the optimal goal-distance in the abstract state space as an (admissible) estimate for the distance in the concrete state space:

$$h(s) = h^*(\alpha(s))$$

1. Precompute h^* for all $\alpha(s) \in S^\alpha$ by performing a backward uniform-cost search in the abstract state space
   - $\rightarrow$ Searching the entire abstract state space? That's what symbolic search is good for!
2. Store them in a look-up table
   - $\rightarrow$ In the form of BDDs, so we can use them in BDDA*

Symbolic Pattern Databases

Do symbolic backward uniform-cost search with only a subset of variables

Symbolic Pattern Databases

Do symbolic backward uniform-cost search with only a subset of variables

- Do not have a limit on the number of variables to consider
- Truncate the search if it takes too much time or memory [AHS07]
- Using all variables: Symbolic Perimeter

Symbolic Pattern Databases

Do symbolic backward uniform-cost search with only a subset of variables

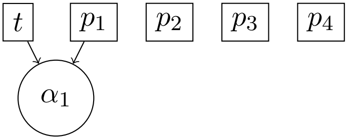
- Do not have a limit on the number of variables to consider
- Truncate the search if it takes too much time or memory [AHS07]
- Using all variables: Symbolic Perimeter

→Really strong for heuristic search planners too [FTLB17]!

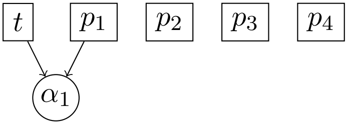
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

$\boxed{t}$ $\boxed{p_1}$ $\boxed{p_2}$ $\boxed{p_3}$ $\boxed{p_4}$

Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

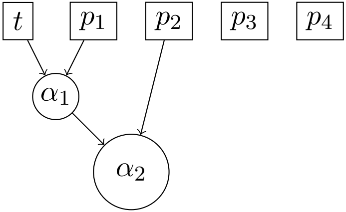


Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]



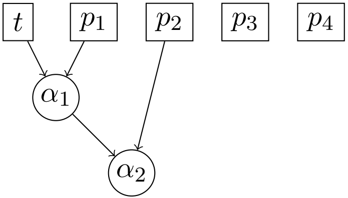
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



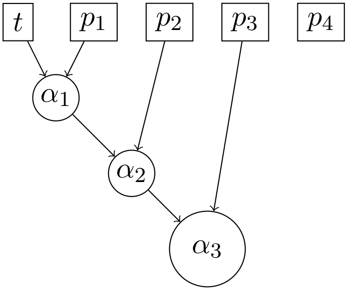
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



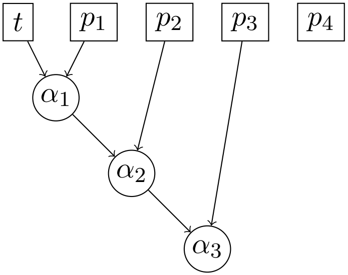
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



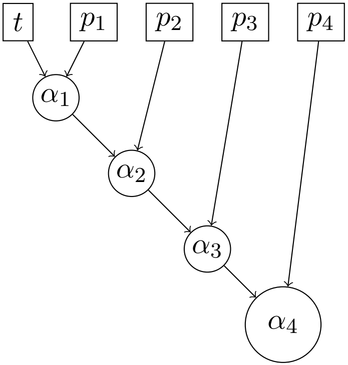
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



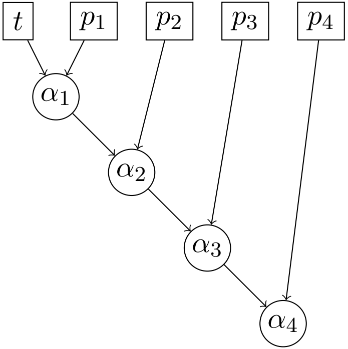
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



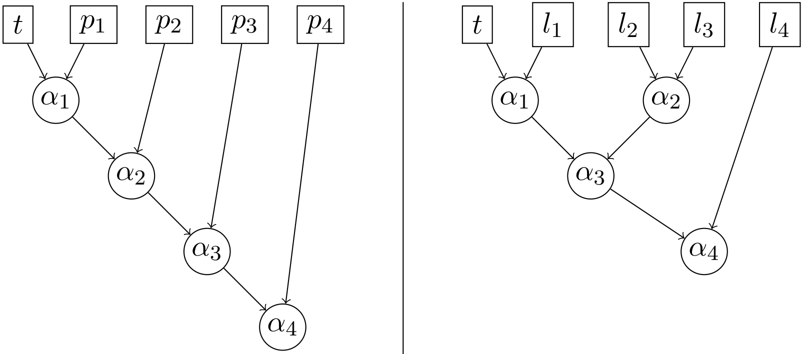
Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



Abstraction Heuristics: Merge-and-Shrink [HHHN14, SWH14]

- Shrink strategy: What abstraction to apply to reduce the abstract state space size?
- Merge strategy: What two abstractions to merge next?



Linear Merge

Non-linear merge

Symbolic Merge-and-Shrink

Are $M\&S$ heuristics efficiently representable by BDDs?

Symbolic Merge-and-Shrink

Are $M\&S$ heuristics efficiently representable by BDDs?

- Yes, if a linear merge strategy is used following the BDD variable ordering [EKT12, Tor15]

Symbolic Merge-and-Shrink

Are $M\&S$ heuristics efficiently representable by BDDs?

- Yes, if a linear merge strategy is used following the BDD variable ordering [EKT12, Tor15]
- No, if a non-linear merge strategy is used (though the exponential overhead can be bounded on the non-linearity of the merge tree) [HRS15]

Symbolic Merge-and-Shrink

Are $M\&S$ heuristics efficiently representable by BDDs?

- Yes, if a linear merge strategy is used following the BDD variable ordering [EKT12, Tor15]
- No, if a non-linear merge strategy is used (though the exponential overhead can be bounded on the non-linearity of the merge tree) [HRS15]

$\rightarrow$We can use M&S heuristics in symbolic search under some restrictions

SymBA*

- We have good symbolic abstraction heuristics
- But, bidirectional blind search is often the best symbolic search algorihm

Can we use heuristics in symbolic bidirectional search?

SymBA*

- We have good symbolic abstraction heuristics
- But, bidirectional blind search is often the best symbolic search algorihm

Can we use heuristics in symbolic bidirectional search? Yes! [TGDH16]

1. Start symbolic bidirectional uniform-cost search
   - If it succeeds → done!
2. Detect when it is going to fail and activate heuristics
   - Perimeters abstractions take advantage of the search already performed in the concrete state space [TLB13]

Agenda

1. About this Tutorial
2. Classical Planning: Models, Approaches
3. Symbolic Representation of Planning Tasks
4. Symbolic Blind Search
5. Heuristic Search
6. Symbolic Abstraction Heuristics
7. Implementation
8. Conclusions and Open Challenges

Planners using Symbolic Search

- **MIPS**: Stefan Edelkamp and Malte Helmert
  `http://www.tzi.de/~edelkamp/mips/mips-bdd.html`
- **MIPS-XXL**: Stefan Edelkamp, Shahid Jabbar, and Mohammed Nazih. Extension to net-benefit with external planning
  `http://sjabbar.com/mips-xxl-planner`
- **BDDPlan**: Hans-Peter Strr `http://www.stoerr.net/bddplan.html`
- **Gamer** (IPC08-IPC11): Peter Kissmann and Stefan Edelkamp
  `https://fai.cs.uni-saarland.de/kissmann/planning/downloads/`
  Extensions (IPC14):
  1. cGamer: improved image computation and state invariant pruning
  2. dynamic Gamer: dynamic variable reordering
- **SymBA*** (IPC14): Based on Fast Downward
  `http://fai.cs.uni-saarland.de/torralba/software.html`

BDD Packages

Library	Language	Reference
CUDD	C/C++	[Som]
CacBDD	C++	[LSX13]
BuDDy	C	[CWWG]
CAL	C	
Sylvan	C	[vDvdP15]
JDD	Java	[Vah]
BeeDeeDee	Java	[LMS14]

→Not clear best performer. CUDD, BuDDy, CacBDD have good results in symbolic model-checking [vDHJ+15].
→There are interfaces that adapt these libraries for other languages like Java, Python, Haskell, . . .

Managing BDDs in CUDD in C++

- Supports:
    - BDDs
    - ZDDs
    - ADDs
- Really easy to perform operations:
    - Disjunction (A + B)
    - Conjunction (A * B).
- Possible to set time and memory limits for BDD operations
    $\rightarrow$ Critical to avoid failure due to an exponential-time BDD operation
- Integrated many other potentially useful algorithms
    - BDD minimization
    - Variable re-ordering
    - ...

Symbolic Search in Fast Downward

- SymVariables: BDD representation of FD variables
  - Obtain the BDD that represents a (partial-)state
  - Check if a BDD contains a given state
- SymStateSpaceManager: Search-related structures to a state space
  - Retrieve initial state/goal states
  - Transition relation and image computation
  - State invariants
- Search algorithms: breadth-first, uniform-cost, A^*, bidirectional, ...

`http://fai.cs.uni-saarland.de/torralba/software.html`

Agenda

Conclusions

- Symbolic search is useful for classical planning

→ Takes advantage of the structure of the planning task implicitly

Conclusions

- Symbolic search is useful for classical planning

→ Takes advantage of the structure of the planning task implicitly

- Advantages:
  - Compact representation of sets of states
  - Time/memory efficient exploration of state spaces

Conclusions

- Symbolic search is useful for classical planning

→ Takes advantage of the structure of the planning task implicitly

- Advantages:
  - Compact representation of sets of states
  - Time/memory efficient exploration of state spaces
- Disadvantages:
  - Heuristics/pruning methods must be adapted to leverage the symbolic representation

Explicit vs. Symbolic Search in Cost-Optimal Planning

Algorithms

- A^*
- Uniform-Cost
- Forward
- Bidirectional

Heuristics

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

max | Disjoint sum

Cost-partitioning

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning
- Dominance pruning

Explicit vs. Symbolic Search in Cost-Optimal Planning

Algorithms

A*	Forward
Uniform-Cost	Bidirectional

Heuristics

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

max | Disjoint sum

Cost-partitioning

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning
- Dominance pruning

Explicit vs. Symbolic Search in Cost-Optimal Planning

Algorithms

- A^*
- Uniform-Cost
- Forward
- Bidirectional

Heuristics

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

max | Disjoint sum | Cost-partitioning

Pruning techniques

- State invariants
- Symmetries
- Partial-order pruning
- Dominance pruning

And What can YOU do?

Still lots of open questions:

- Are there any alternatives to BDDs?
- Can BDDs be split for a more efficient exploration of the state space?
- Keep optimizing operations
  - How to do efficient image operations [TEK13, TAKE17]?
  - How to choose good BDD variable orderings [KE11, KH13]?

BDDs can be useful for you to represent and operate with sets of states even if you are not planning to use symbolic search!

- Partial-state regression search: Keep set of all states expanded so far [AFB14]
- Dominance Pruning: Keep set of all states dominated by any expanded state [TH15]

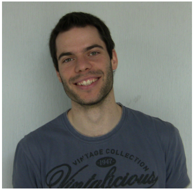
Acknowledgements

Stefan Edelkamp

Peter Kissmann

Jussi Rintanen

Vidal Alcázar



Carlos Linares López

Daniel Borrajo

References I

[AE03] Eyal Amir and Barbara Engelhardt, *Factored planning*, Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03) (Acapulco, Mexico) (G. Gottlob, ed.), Morgan Kaufmann, August 2003, pp. 929–935.

[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.

[AHS07] Kenneth Anderson, Robert Holte, and Jonathan Schaeffer, *Partial pattern databases*, Proceedings of the 7th International Symposium on Abstraction, Reformulation, and Approximation (SARA-07) (Whistler, Canada) (Ian Miguel and Wheeler Ruml, eds.), Lecture Notes in Computer Science, vol. 4612, Springer-Verlag, 2007, pp. 20–34.

References II

[BCL91] Jerry R. Burch, Edmund M. Clarke, and David E. Long, *Symbolic model checking with partitioned transition relations*, Proceedings of the International Conference on Very Large Scale Integration (VLSI-91) (Edinburgh, Scotland) (Arne Halaas and Peter B. Denyer, eds.), IFIP Transactions, vol. A-1, North-Holland, 1991, pp. 49–58.

[BD06] Ronen Brafman and Carmel Domshlak, *Factored planning: How, when, and when not*, Proceedings of the 21st National Conference of the American Association for Artificial Intelligence (AAAI'06) (Boston, Massachusetts, USA) (Yolanda Gil and Raymond J. Mooney, eds.), AAAI Press, July 2006, pp. 809–814.

[BD08] Ronen I. Brafman and Carmel Domshlak, *From one to many: Planning for loosely coupled multi-agent systems*, Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08) (Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, eds.), AAAI Press, 2008, pp. 28–35.

[BD13] Ronen Brafman and Carmel Domshlak, *On the complexity of planning for agent teams and its implications for single agent planning*, Artificial Intelligence **198** (2013), 52–71.

References III

[BGB13] Pascal Bercher, Thomas Geier, and Susanne Biundo, *Using state-based planning heuristics for partial-order causal-link planning*, Proceedings of the 36th Annual German Conference on Artificial Intelligence (KI'11) (Ingo J. Timm and Matthias Thimm, eds.), Lecture Notes in Computer Science, vol. 8077, Springer, 2013, pp. 1–12.

[BHHT08] Blai Bonet, Patrik Haslum, Sarah L. Hickmott, and Sylvie Thiébaux, *Directed unfolding of petri nets*, Transactions on Petri Nets and Other Models of Concurrency **1** (2008), 172–198.

[BHK+14] Blai Bonet, Patrik Haslum, Victor Khomenko, Sylvie Thiébaux, and Walter Vogler, *Recent advances in unfolding technique*, Theoretical Computer Science **551** (2014), 84–101.

[Bra11] Aaron R. Bradley, *Sat-based model checking without unrolling*, Proceedings of the 12th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'11), 2011, pp. 70–87.

References IV

[BRKM91] Kenneth M. Butler, Don E. Ross, Rohit Kapur, and M. Ray Mercer, *Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams*, Proceedings of the 28th Conference on Design Automation (DAC-91) (San Francisco, CA, USA), ACM, 1991, pp. 417–420.

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

[CHP93] Pi-Yu Chung, Ibrahim N. Hajj, and Janak H. Patel, *Efficient variable ordering heuristics for shared ROBDD*, Proceedings of the 1993 IEEE International Symposium on Circuits and Systems (ISCAS-93) (Chicago, IL, USA), IEEE, 1993, pp. 1690–1693.

[CS98] Joseph C. Culberson and Jonathan Schaeffer, *Pattern databases*, Computational Intelligence **14** (1998), no. 3, 318–334.

[CWWG] H. Cohen, J. Whaley, J. Wildt, and N. Gorogiannis, *BuDDy*, At `http://sourceforge.net/p/buddy/`.

References V

[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.

[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.

[Ede01] Stefan Edelkamp, *Planning with pattern databases*, Proceedings of the 6th European Conference on Planning (ECP'01) (A. Cesta and D. Borrajo, eds.), Springer-Verlag, 2001, pp. 13–24.

References VI

[EK11] Stefan Edelkamp and Peter Kissmann, *On the complexity of BDDs for state space search: A case study in connect four*, 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.

[EKT12] Stefan Edelkamp, Peter Kissmann, and Álvaro Torralba, *Symbolic A* search with pattern databases and the merge-and-shrink abstraction*, Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12) (Montpellier, France) (Luc De Raedt, ed.), IOS Press, August 2012, pp. 306–311.

[ELL04] Stefan Edelkamp, Stefan Leue, and Alberto Lluch-Lafuente, *Partial-order reduction and trail improvement in directed model checking*, International Journal on Software Tools for Technology Transfer **6** (2004), no. 4, 277–301.

[EMB11] Niklas Eén, Alan Mishchenko, and Robert K. Brayton, *Efficient implementation of property directed reachability*, International Conference on Formal Methods in Computer-Aided Design, FMCAD '11, Austin, TX, USA, 2011, pp. 125–134.

References VII

[EMW97] Michael D. Ernst, Todd D. Millstein, and Daniel S. Weld, *Automatic SAT-compilation of planning problems*, Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97) (Nagoya, Japan) (M. Pollack, ed.), Morgan Kaufmann, August 1997, pp. 1169–1177.

[ERV02] Javier Esparza, Stefan Römer, and Walter Vogler, *An improvement of mcmillan's unfolding algorithm*, Formal Methods in System Design **20** (2002), no. 3, 285–310.

[FJHT10] Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux, *Cost-optimal factored planning: Promises and pitfalls*, Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10) (Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, eds.), AAAI Press, 2010, pp. 65–72.

[FTLB17] Santiago Franco, Álvaro Torralba, Levi H.S. Lelis, and Mike Barley, *On creating complementary pattern databases*, Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17) (Carles Sierra, ed.), AAAI Press/IJCAI, 2017.

References VIII

[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.

[GW91] Patrice Godefroid and Pierre Wolper, *Using partial orders for the efficient verification of deadlock freedom and safety properties*, Proceedings of the 3rd International Workshop on Computer Aided Verification (CAV'91), 1991, pp. 332–342.

[HBH+07] Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig, *Domain-independent construction of pattern database heuristics for cost-optimal planning*, Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07) (Vancouver, BC, Canada) (Adele Howe and Robert C. Holte, eds.), AAAI Press, July 2007, pp. 1007–1012.

[HHHN14] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim, *Merge & shrink abstraction: A method for generating lower bounds in factored state spaces*, Journal of the Association for Computing Machinery **61** (2014), no. 3.

References IX

[HRS15] Malte Helmert, Gabriele Röger, and Silvan Sievers, *On the expressive power of non-linear merge-and-shrink representations*, Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15) (Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, eds.), AAAI Press, 2015, pp. 106–114.

[HRTW07] Sarah L. Hickmott, Jussi Rintanen, Sylvie Thiébaux, and Langford B. White, *Planning via petri net unfolding*, Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07) (Hyderabad, India) (Manuela Veloso, ed.), Morgan Kaufmann, January 2007, pp. 1904–1911.

[JVB08] Rune M. Jensen, Manuela M. Veloso, and Randal E. Bryant, *State-set branching: Leveraging BDDs for heuristic search*, Artificial Intelligence **172** (2008), no. 2-3, 103–139.

[KBHT07] Elena Kelareva, Olivier Buffet, Jinbo Huang, and Sylvie Thiébaux, *Factored planning using decomposition trees*, Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07) (Hyderabad, India) (Manuela Veloso, ed.), Morgan Kaufmann, January 2007, pp. 1942–1947.

References X

[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.

[KH13] Peter Kissmann and Jörg Hoffmann, *What's in it for my BDD? On causal graphs and variable orders in planning*, Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13) (Rome, Italy) (Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, eds.), AAAI Press, 2013, pp. 327–331.

[KH14] ———, *BDD ordering heuristics for classical planning*, Journal of Artificial Intelligence Research **51** (2014), 779–804.

[KKY95] Subbarao Kambhampati, Craig A. Knoblock, and Qiang Yang, *Planning as refinement search: a unified framework for evaluating design tradeoffs in partial-order planning*, Artificial Intelligence **76** (1995), no. 1–2, 167–238.

[Kno94] Craig Knoblock, *Automatically generating abstractions for planning*, Artificial Intelligence **68** (1994), no. 2, 243–302.

References XI

[KS92] Henry A. Kautz and Bart Selman, *Planning as satisfiability*, Proceedings of the 10th European Conference on Artificial Intelligence (ECAI'92) (Vienna, Austria) (B. Neumann, ed.), Wiley, August 1992, pp. 359–363.

[KS96] ______, *Pushing the envelope: Planning, propositional logic, and stochastic search*, Proceedings of the 13th National Conference of the American Association for Artificial Intelligence (AAAI'96) (Portland, OR) (William J. Clancey and Daniel Weld, eds.), MIT Press, July 1996, pp. 1194–1201.

[LMS14] Alberto Lovato, Damiano Macedonio, and Fausto Spoto, *A thread-safe library for binary decision diagrams*, Software Engineering and Formal Methods - 12th International Conference, SEFM 2014, Grenoble, France, September 1-5, 2014. Proceedings (Dimitra Giannakopoulou and Gwen Salaün, eds.), Lecture Notes in Computer Science, vol. 8702, Springer, 2014, pp. 35–49.

[LSX13] Guanfeng Lv, Kaile Su, and Yanyan Xu, *Cacbdd: A BDD package with dynamic cache management*, Lecture Notes in Computer Science, vol. 8044, Springer, 2013, At `http://www.kailesu.net/CacBDD/`, pp. 229–234.

References XII

[Mai09] Vivien Maisonneuve, *Automatic heuristic-based generation of MTBDD variable orderings for PRISM models*, Internship report, Oxford University Computing Laboratory, 2009.

[McM92] Kenneth L. McMillan, *Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits*, Proceedings of the 4th International Workshop on Computer Aided Verification (CAV'92) (Gregor von Bochmann and David K. Probst, eds.), Lecture Notes in Computer Science, Springer, 1992, pp. 164–177.

[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.

[MIY90] Shin-ichi Minato, Nagisa Ishiura, and Shuzo Yajima, *Shared binary decision diagram with attributed edges for efficient boolean function manipulation*, Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC-90) (Orlando, FL, USA), IEEE Computer Society Press, 1990, pp. 52–57.

References XIII

[MWBSV88] S. Malik, A.R. Wang, R.K. Brayton, and A. Sangiovanni-Vincentelli, *Logic verification using binary decision diagrams in a logic synthesis environment*, Proceedings of the 1988 International Conference on Computer-Aided Design (ICCAD-98), IEEE Computer Society Press, 1988, pp. 6–9.

[Rin98] Jussi T. Rintanen, *A planning algorithm not based on directional search*, Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-98) (Trento, Italy) (A. Cohn, L. Schubert, and S. Shapiro, eds.), Morgan Kaufmann, May 1998, pp. 953–960.

[Rin03] Jussi Rintanen, *Symmetry reduction for SAT representations of transition systems*, Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03) (Trento, Italy) (Enrico Giunchiglia, Nicola Muscettola, and Dana Nau, eds.), Morgan Kaufmann, 2003, pp. 32–41.

References XIV

[Rin08] ______, *Regression for classical and nondeterministic planning*, Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08) (Patras, Greece) (Malik Ghallab, ed.), Wiley, July 2008, pp. 568–572.

[Rin12] ______, *Planning as satisfiability: Heuristics*, Artificial Intelligence **193** (2012), 45–86.

[Rud93] Richard Rudell, *Dynamic variable ordering for ordered binary decision diagrams*, Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design (ICCAD-93) (Santa Clara, CA, USA) (Michael R. Lightner and Jochen A. G. Jess, eds.), IEEE Computer Society, 1993, pp. 42–47.

[Sac75] Earl D. Sacerdoti, *The nonlinear nature of plans*, Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI'75) (Tiblisi, USSR), William Kaufmann, September 1975, pp. 206–214.

[Som] Fabio Somenzi, *CUDD: CU decision diagram package release 3.0.0.*, At `http://vlsi.colorado.edu/~fabio/`.

References XV

[Sud14] Martin Suda, *Property directed reachability for automated planning*, Journal of Artificial Intelligence Research **50** (2014), 265–319.

[SWH14] Silvan Sievers, Martin Wehrle, and Malte Helmert, *Generalized label reduction for merge-and-shrink heuristics*, Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14) (Austin, Texas, USA) (Carla E. Brodley and Peter Stone, eds.), AAAI Press, January 2014, pp. 2358–2366.

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

[TEK13] Álvaro Torralba, Stefan Edelkamp, and Peter Kissmann, *Transition trees for cost-optimal symbolic planning*, Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13) (Rome, Italy) (Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, eds.), AAAI Press, 2013, pp. 206–214.

References XVI

[TGDH16] Álvaro Torralba, Daniel Gnad, Patrick Dubbert, and Jörg Hoffmann, *On state-dominance criteria in fork-decoupled search*, Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16) (Subbarao Kambhampati, ed.), AAAI Press/IJCAI, 2016.

[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.

[TLB13] Álvaro Torralba, Carlos Linares López, and Daniel Borrajo, *Symbolic merge-and-shrink for cost-optimal planning*, Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13) (Francesca Rossi, ed.), AAAI Press/IJCAI, 2013, pp. 2394–2400.

[Tor15] Álvaro Torralba, *Symbolic search and abstraction heuristics for cost-optimal planning*, Ph.D. thesis, Universidad Carlos III de Madrid, 2015.

[Vah] A. Vahidi, *JDD, a pure java BDD and z-BDD library*.

References XVII

[vDHJ+15] Tom van Dijk, Ernst Moritz Hahn, David N. Jansen, Yong Li, Thomas Neele, Mariëlle Stoelinga, Andrea Turrini, and Lijun Zhang, *A comparative study of BDD packages for probabilistic symbolic model checking*, Dependable Software Engineering: Theories, Tools, and Applications - First International Symposium, SETTA 2015, Nanjing, China, November 4-6, 2015, Proceedings (Xuandong Li, Zhiming Liu, and Wang Yi, eds.), Lecture Notes in Computer Science, vol. 9409, Springer, 2015, pp. 35–51.

[vDvdP15] Tom van Dijk and Jaco van de Pol, *Sylvan: Multi-core decision diagrams*, Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings (Christel Baier and Cesare Tinelli, eds.), Lecture Notes in Computer Science, vol. 9035, Springer, 2015, pp. 677–691.

[YS03] Håkan L. S. Younes and Reid G. Simmons, *VHPOP: versatile heuristic partial order planner*, Journal of Artificial Intelligence Research **20** (2003), 405–430.