Algorithms and Satisfiability

11. Planning as SAT and Symbolic Search

Solving Planning with Satisfiability and BDDs

Álvaro Torralba

AALBORG UNIVERSITET

Spring 2023

Agenda

The Question for this Lecture

The Satisfiability part of this course so far: Algorithms to solve hard problems that cannot be solved in polynomial time (unless **P**=**NP**)

- Satisfiability (NP-complete)
  →DPLL + Clause Learning
- Binary Decision Diagrams
- Planning (PSPACE-complete)
  →Heuristic Search + Delete-relaxation

Question?

If these are general problem solving techniques, can't we just use SAT and/or BDDs to solve planning problems?

- Planning as SAT
- Symbolic Search

States as Logical Formulas

Example: "Logistics"

l_1 — l_2 — l_3

- Facts P: $\{at(x,l) \mid x \in \{t,p_1,p_2\}, l \in \{l_1,l_2,l_3,t\}\}$
- Initial state I: $\{at(t,l_1), at(p_1,l_1), at(p_2,l_1)\}$
- Goal G: $\{at(t,l_1), at(p_1,l_3), at(p_2,l_3)\}$
- Actions A: (Notated as "precondition $\Rightarrow$ adds, $\neg$ deletes")
  - $drive(x,y)$, where x,y have a road: "$at(t,x) \Rightarrow at(t,y), \neg at(t,x)$".
  - $load(p,x)$: "$at(t,x), at(p,x) \Rightarrow at(p,t), \neg at(p,x)$".
  - $unload(p,x)$: "$at(t,x), at(p,t) \Rightarrow at(p,x), \neg at(p,t)$".

We can represent states as conjunction of literals:

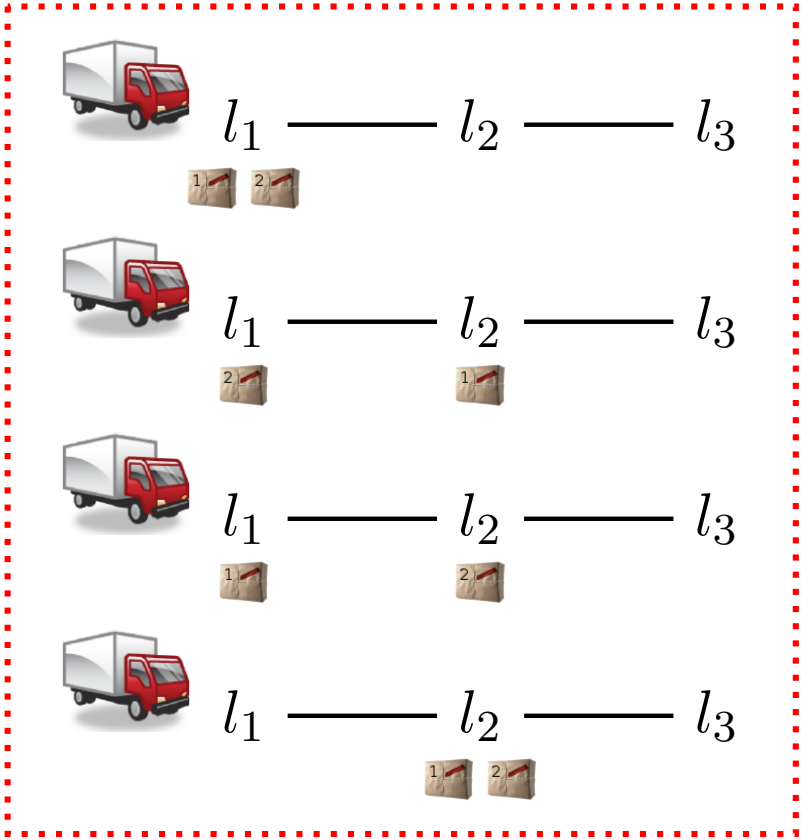
$$\phi_I = at(t,l_1) \wedge at(p_1,l_1) \wedge at(p_2,l_1) \wedge \neg at(t,l_2) \wedge \neg at(p_1,l_2) \wedge \ldots$$

In propositional logic there is no closed-world assumption: we also need to explicitly represent the negated literals!

Notation: $\langle t,l_1 \rangle = at(t,l_1) \wedge \neg at(t,l_2) \wedge \neg at(t,l_3)$

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

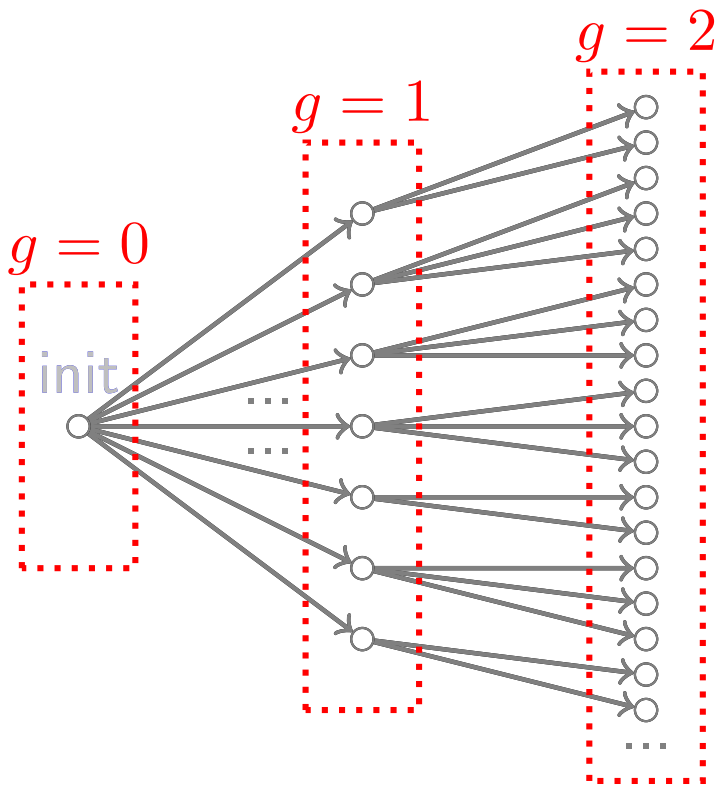
Sets of States as Logical Formulas



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

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

State Space Explosion



BDDs to the rescue!

?

goal

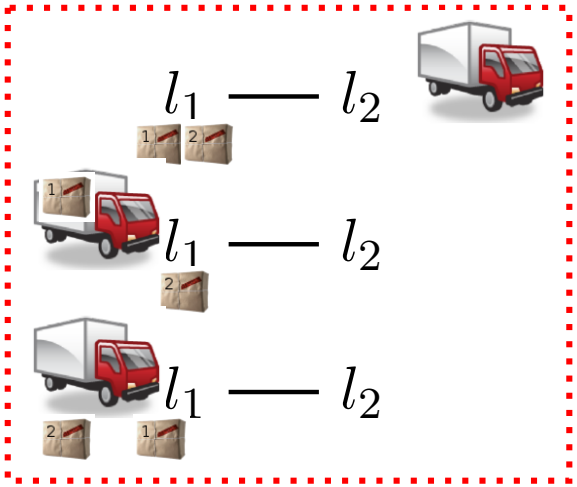
Huge branching factor $\rightarrow$ state space *explosion*

Sets of States as BDDs

In heuristic search, we process one state at a time

Idea: Can we perform search where we process multiple states at once?

Sets of States $\leftrightarrow$ Logical Formulas $\leftrightarrow$ BDDs



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

$\rightarrow$The straightfoward encoding would have 11 layers, this is a simplification

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 ($\neg$)

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'.

$load(p_1, l_1)$:$pre : \{at(t, l_1), at(p_1, l_1)\}$,$add : \{in(p_1, t)\}$,$del : \{at(p_1, l_1)\}$

$l_1 - l_2 - l_3$

$l_1 - l_2 - l_3$

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

$l_1 - l_2 - l_3$

$l_1 - l_2 - l_3$

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

$l_1 - l_2 - l_3$

$l_1 - l_2 - l_3$

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

$$at(p_1 l_1) \wedge at(t, l_1) \wedge in(p_1, t)' \wedge (at(t, l_1) \leftrightarrow at(t, l_1)') \wedge$$
$$(at(p_2, l_1) \leftrightarrow at(p_2, l_1)') \wedge \ldots$$

Transition Relation

For each action a:

$$TR_a = \bigwedge_{p \in pre(a)} p \wedge \bigwedge_{p \in add(a)} p' \wedge \bigwedge_{p \in del(a)} \neg p' \wedge \bigwedge_{p \in P \setminus (add(a) \cup del(a))} (p \leftrightarrow p')$$

- Preconditions must hold on the predecessor state
- Effects (adds and deletes) must hold on the successor state
- All other facts remain unmodified

$\rightarrow$Corresponds to all pairs (s, s') such that $s \xrightarrow{a} s'$

Full transition relation:

$$TR = \bigvee_{a \in A} TR_a$$

$\rightarrow$Corresponds to all pairs (s, s') such that $s \xrightarrow{a} s'$ for some a

Representing States and Actions as BDDs

- BDDs have $2|P|$ variables: x and x'
  - BDDs representing states use only standard variables x
  - BDDs representing actions use both

Variable Ordering: Interleave variables x and x'

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

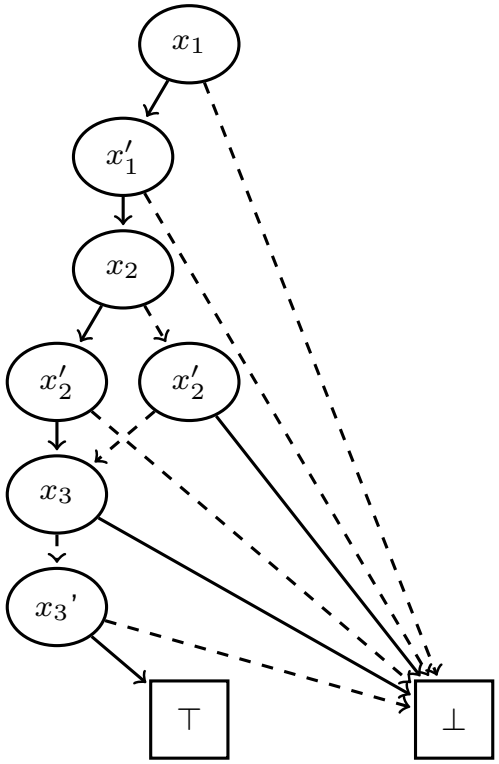


Image Operation

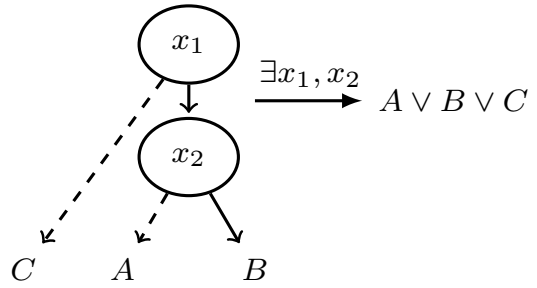
Image: Given a set of states $S(x)$ and a TR $T(x, x')$ generate the successor states

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

Uses two new BDD operations:

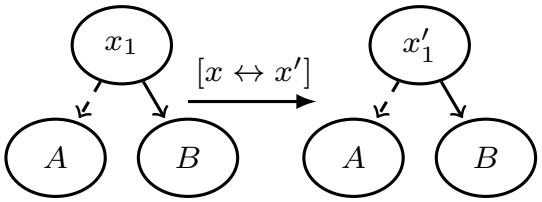
Existential quantification $\exists x$:

- Removes some variables x from the BDD, computing the disjunction of their children
- Worst-case exponential as it may compute the disjunction of many sub-BDDs



Variable Replacement $[x' \leftrightarrow x]$:

- Changes the BDD variables
- Worst-case exponential (changes variable ordering)



Computing the Successors (Image Computation) Example

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			

$\exists$-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

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

Questionnaire

Question!

Let C be the set of closed states (that have already been expanded), and let S be the newly generated set of states. How do we compute the subset of states in S that have not been expanded yet?

(A): $C \wedge S$ (B): $\neg C \wedge S$

(C): $C \vee S$ (D): $\neg C \vee S$

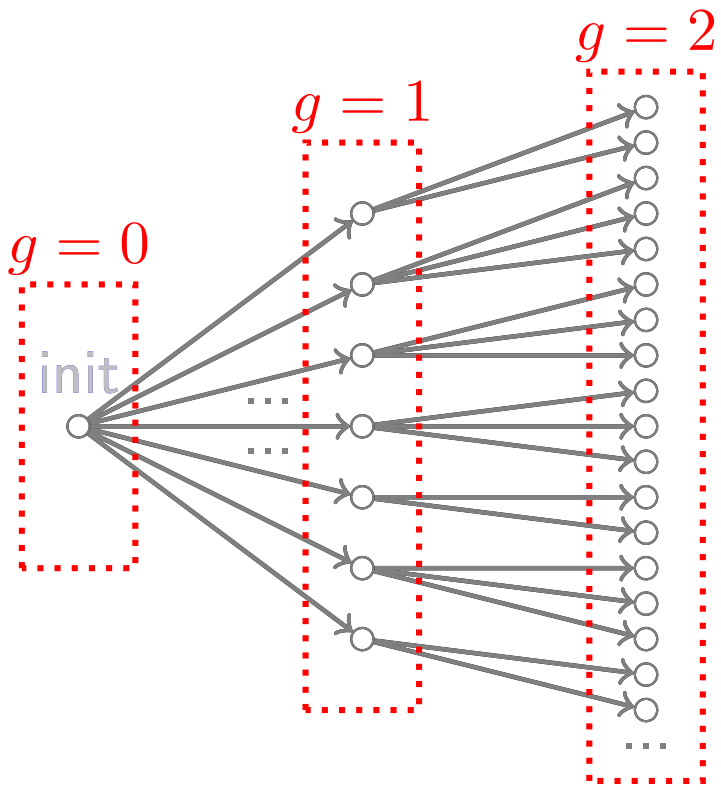
Bonus: What is the time complexity of this operation?

(B) The intersection between the states that are in S and the ones that have not been explored yet.

This has complexity $\mathcal{O}(|S| \cdot |C|)$, as negation takes constant time, and conjunction is quadratic in the number of nodes of the input BDDs.

$\rightarrow$This is only the worst-case, often the result will be smaller than S!

State Space Explosion



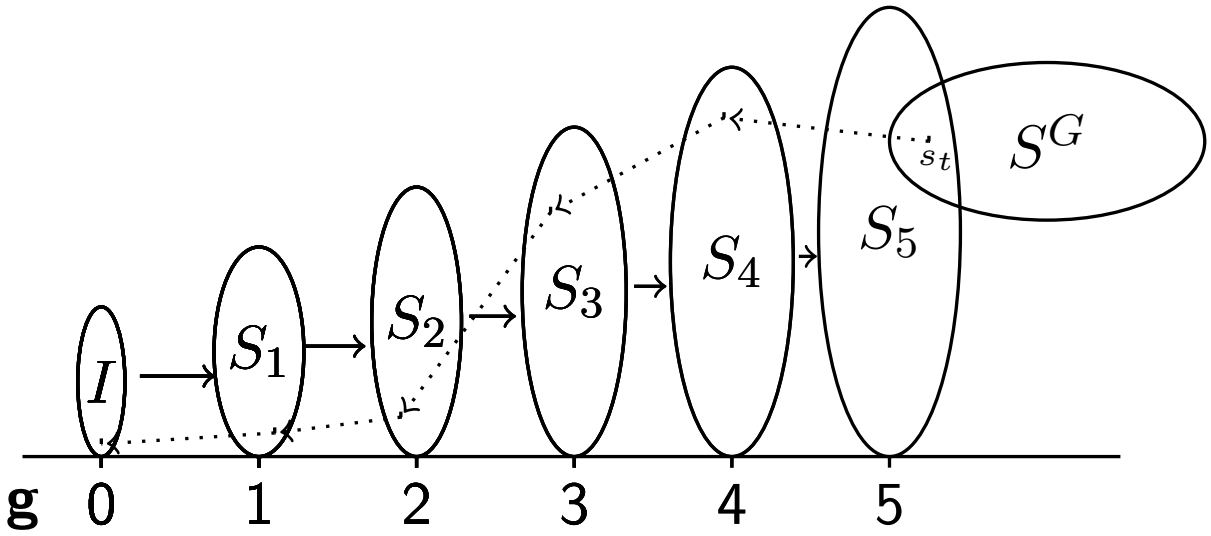
BDDs to the rescue!

?

goal

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

Symbolic Breadth-First Search



At every step:

- Image to generate states reachable with one action
- Insert new states into closed; Remove closed from states
- Check intersection with goal

Symbolic Breadth-First Search: Pseudocode

- C: Closed list (set of states explored so far)
- S_i: Set of states seen after i steps

Input: Planning Task $\Pi = (V, A, I, G)$

1 $S_0 \leftarrow I$;
2 $C \leftarrow \emptyset$;
3 $i \leftarrow 0$;
4 **while** $S_i \neq \emptyset$ **do**
5 **if** $S_i \wedge G$ **then**
6 **return** Plan ;
7 $C \leftarrow C \vee S_i$;
8 $S_{i+1} \leftarrow image(S_i, TR) \wedge \neg C$;
9 $i \leftarrow i + 1$;
10 **return** Unsolvable ;

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

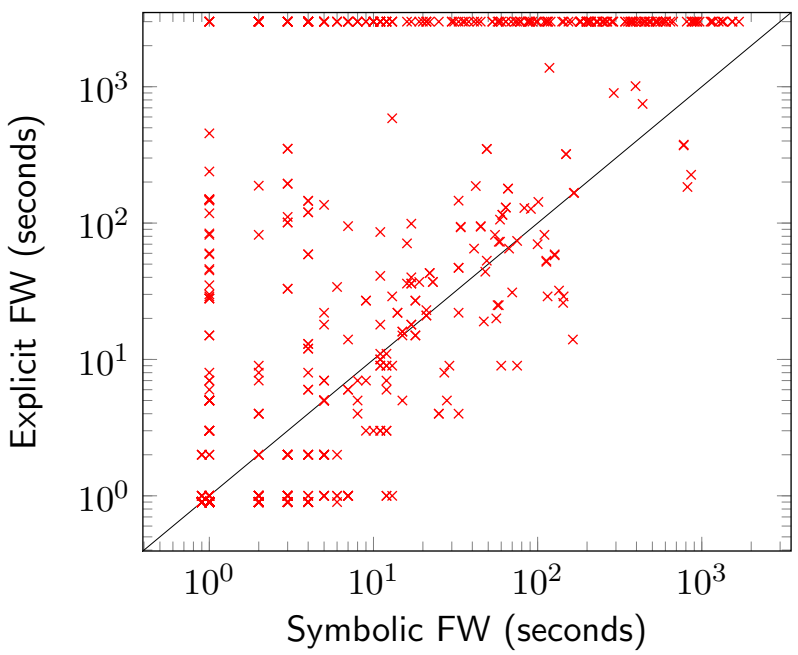
Challenge:

- Multiple goal states $\rightarrow$Not a problem in symbolic search!

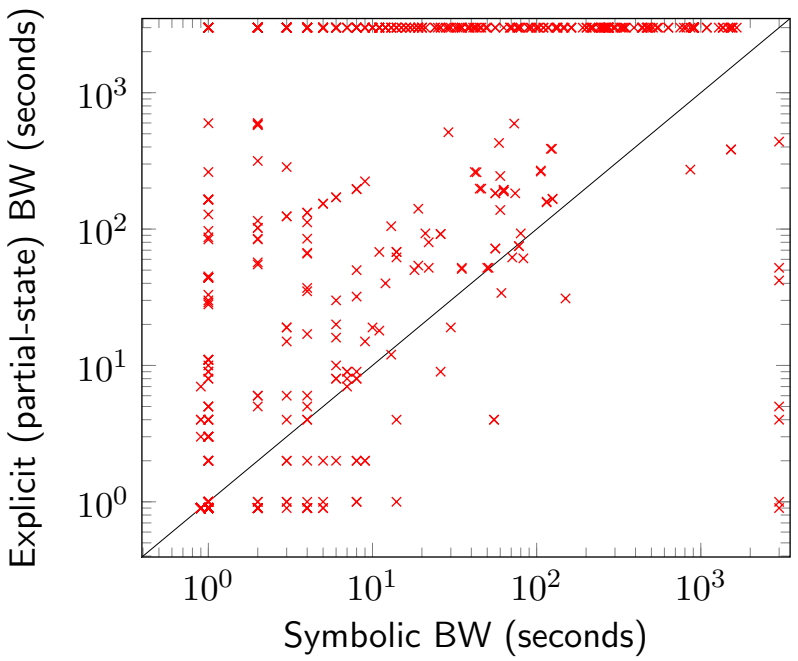
$\rightarrow$Remark: Performance can benefit a lot from encoding state-invatiants in the BDDs and/or TR (e.g. A truck cannot be in two locations at the same time: $\neg(at(t, l_1) \wedge at(t, l_2)))$

Symbolic Uniform-Cost Search: Results

- Performance of symbolic search depends on how compact is the BDD representation of the sets of states with g=1, g=2, etc.
- Still is often much better than enumerating all states one by one!



Forward

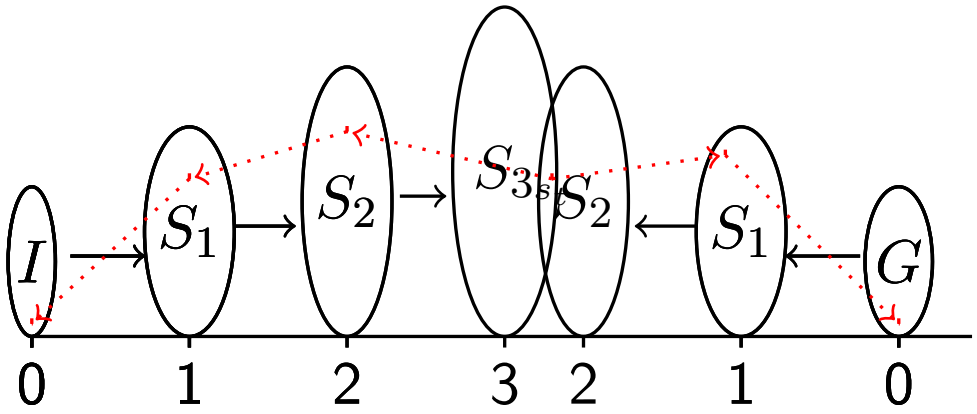


Backward

Symbolic Bidirectional Search

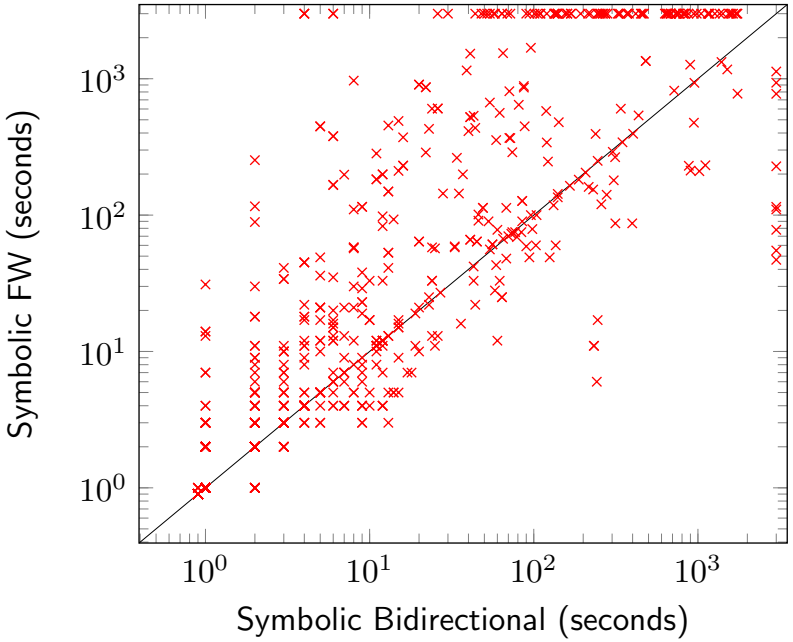
- Do forward and backward search at the same time
- Decide forward or backward direction at each step
- Check intersection against opposite frontier

$\rightarrow$Solution is still guaranteed to be optimal!

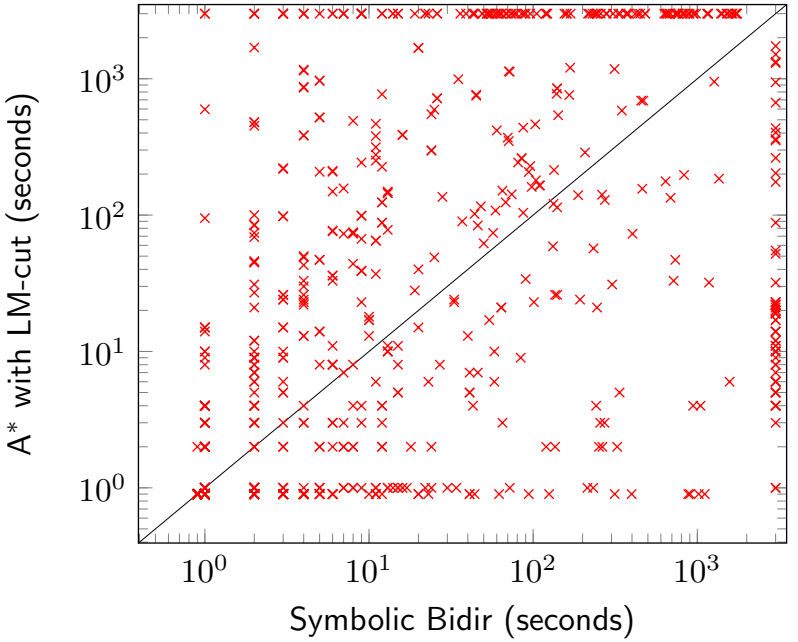


Symbolic Bidirectional Uniform-Cost Search: Results

- Much better than unidirectional search!
- Complementary with heuristic search approaches (c.f. Chapters 10, 11)
- Good in optimal planning



vs Symbolic Forward



vs A* with LM-cut

?

Idea

Given a planning task Π, construct a formula ϕ_Π such that ϕ_Π is satisfiable if and only if Π is solvable

$\rightarrow$A plan for Π can be recovered from the satisfying assignment for ϕ_Π

Reminder: PlanEx/PlanLen are PSPACE-complete; SAT is NP-complete.

Question!

Can we reduce Planning to SAT?

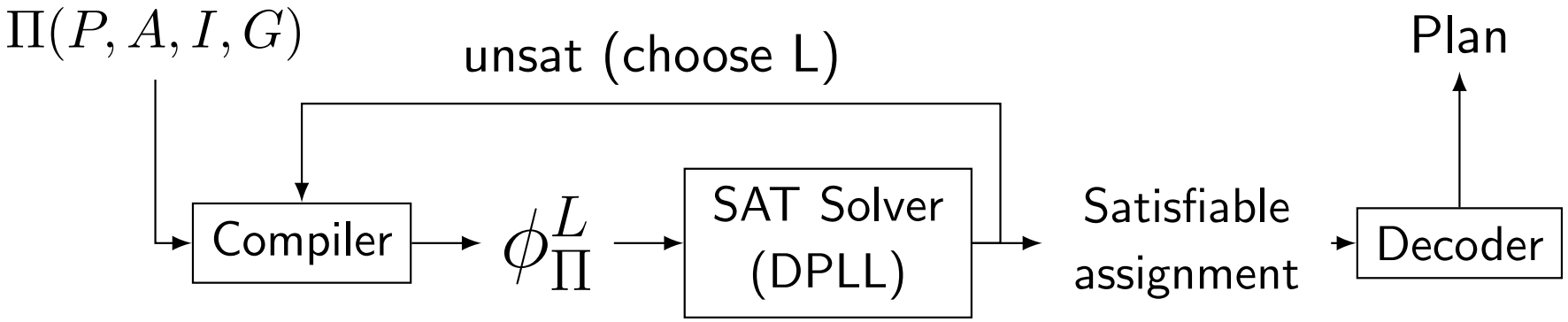
(A): Yes (B): No

No, unless we prove that PSPACE = NP

However, PolyPlanLen is NP-complete:
$\rightarrow$We can reduce PolyPlanLen to SAT!

Planning as SAT

$\phi_{\Pi,L}$: Given a planning task $\Pi = (P, A, I, G)$, does there exist a plan of "length" L or less?



Two separate questions:

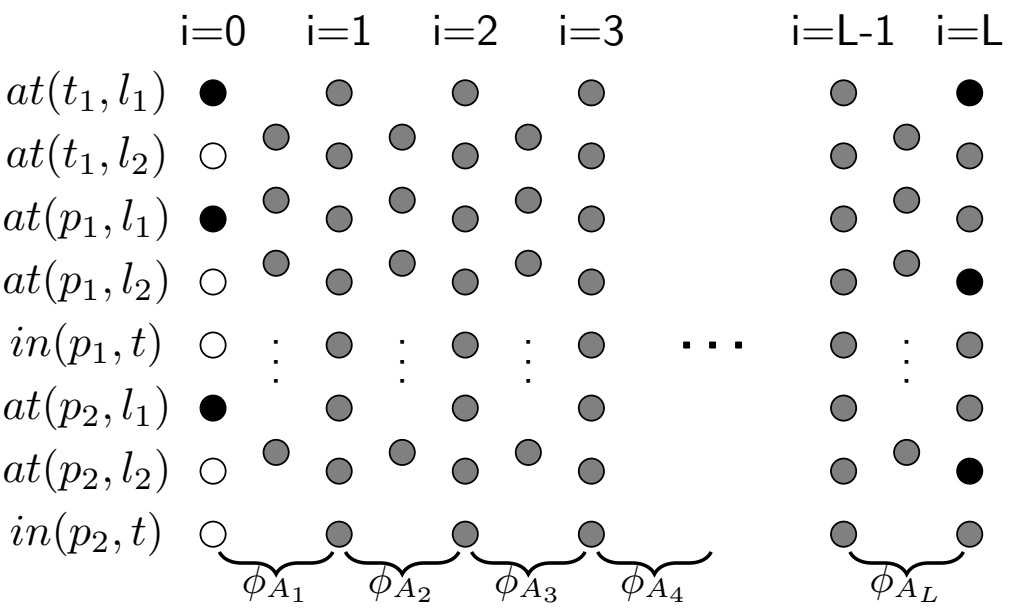
1. How to construct $\phi_{\Pi,L}$?
2. How to choose L?

How to construct $\phi_{\Pi,L}$?

Propositions:

- p^i for $p \in P$ and $i \in [0, \ldots, L]$ →Is p true after i steps?
- a^i for $a \in A$ and $i \in [1, \ldots, L]$ →Is a applied at step i?

$$\phi_{\Pi,L} = \phi_I \wedge \phi_G \wedge (\bigwedge_{k=1}^{L} \phi_{A_k})$$



- Layer 0 is the initial state:

$$\phi_I = \bigwedge_{p \in I} p^0 \wedge \bigwedge_{p \in P \setminus I} \neg p^0$$

- The goal is true in layer L:

$$\phi_G = \bigwedge_{p \in G} p^L$$

- Layer k can be obtained from layer $k-1$ by applying some actions:

ϕ_{A_k} (multiple encodings)

Defining ϕ_{A_k}: Encoding Actions

$$\phi_{A_k} = \bigwedge_{a \in A} \left(\boldsymbol{a^k} \implies \bigwedge_{p \in pre(a)} \boldsymbol{p^{k-1}} \wedge \bigwedge_{p \in add(a)} \boldsymbol{p^k} \bigwedge_{p \in del(a)} \neg \boldsymbol{p^k} \right)$$

$$\wedge \bigwedge_{p \in P} \left((\boldsymbol{p^{k-1}} \wedge \neg \boldsymbol{p^k}) \implies (\bigvee_{a \in A, p \in del(a)} \boldsymbol{a^k}) \right)$$

$$\wedge \bigwedge_{p \in P} \left((\neg \boldsymbol{p^{k-1}} \wedge \boldsymbol{p^k}) \implies (\bigvee_{a \in A, p \in add(a)} \boldsymbol{a^k}) \right) \wedge \ldots \text{(next slides)}$$

Intuitive meaning:

- If we apply action a at step k, their preconditions hold at step $k-1$, and their effects hold at step k
- If p changes from false to true, we have applied some action that adds p
- If p changes from true to false, we have applied some action that deletes p

Questionnaire

Question!

According to the encoding in the previous slide, we can choose to make true more than an action in the same step. For which of the following pairs of actions, will this be problematic when reconstructing the plan?

(A): drive(t, l_1, l_2),drive(t, l_1, l_3) (B): drive(t, l_1, l_2),drive(t, l_3, l_2)

(C): load(p_1, l_1),load(p_2, l_1) (D): load(t, p_1, l_1),drive(t, l_1, l_2),

(A) This is a problem! After applying one of the actions, we cannot apply the other. Within the assignment at step k, we will have the truck in two locations at the same time and there is no corresponding plan that can achieve this.

(B) Not a problem (though the preconditions of both actions will never be true at step $k-1$)

(C) Not a problem at all! We can load the packages on the truck in any order! ($\forall$-step)

(D) Not a problem, but it will make plan reconstruction more difficult as we need to make sure that we load the package before driving! ($\exists$-step)

Defining ϕ_{A_k}: Step Semantics

How do we make sure that no conflicting actions are applied at the same step?

Sequential: At most one action per step

→Simple, but it requires many more steps (larger formulas and harder to solve)

Parallel: Multiple actions per step (multiple semantics). A set of actions A_k can be simultaneously applied in s_k to reach s_{k+1} if:

- $\forall$-step: all possible orderings are executable on s_k and result in s_{k+1} [Kautz and Selman (1996)]
- $\exists$-step: there exists an ordering that is executable on s_k and results in s_{k+1} [Rintanen *et al.* (2006)]
- $R\exists$-step/$R^2\exists$-step: relaxed variants that do not require the precondition of all actions in A_k to be executable in s_k or their effects to hold in s_{k+1} if a valid sequence exists [Wehrle and Rintanen (2007); Balyo (2013)].

→For each of these, multiple encodings are possible (e.g. linear or quadratic)

Defining ϕ_{A_k}: Sequential Semantics

Sequential Semantics: At most one action per step

$$\phi_{A_k} = \text{(see slide 30)} \cdots \wedge \bigwedge_{a_1,a_2 \in A, a_1 \neq a_2} \neg(a_1^k \wedge a_2^k)$$

Notation: We will use $\phi^{seq}_{\Pi,L}$ to refer to formulas that use sequential semantics

$\rightarrow$**Remark:** With this encoding ϕ_{A_k} has quadratic size in the number of actions. A better (linear) encoding exists [Rintanen *et al.* (2006)].

Defining ϕ_{A_k}: Parallel ($\forall$-step) Semantics

$\forall$-step Parallel Semantics ($\phi^{\forall\text{-step}}_{\Pi,L}$): (*) all possible orderings are executable on s_k and result in s_{k+1}

$$\phi_{A_k} = \text{(see slide 30)} \cdots \wedge \bigwedge_{a_1,a_2 \in A, \mathit{interfere}(a_1,a_2)} \neg(a_1^k \wedge a_2^k)$$

Definition (Interference): Actions a_1 and a_2 interfere if $(pre(a_1) \cup add(a_1)) \cap del(a_2) \neq \emptyset$, or $(pre(a_2) \cup add(a_2)) \cap del(a_i) \neq \emptyset$.

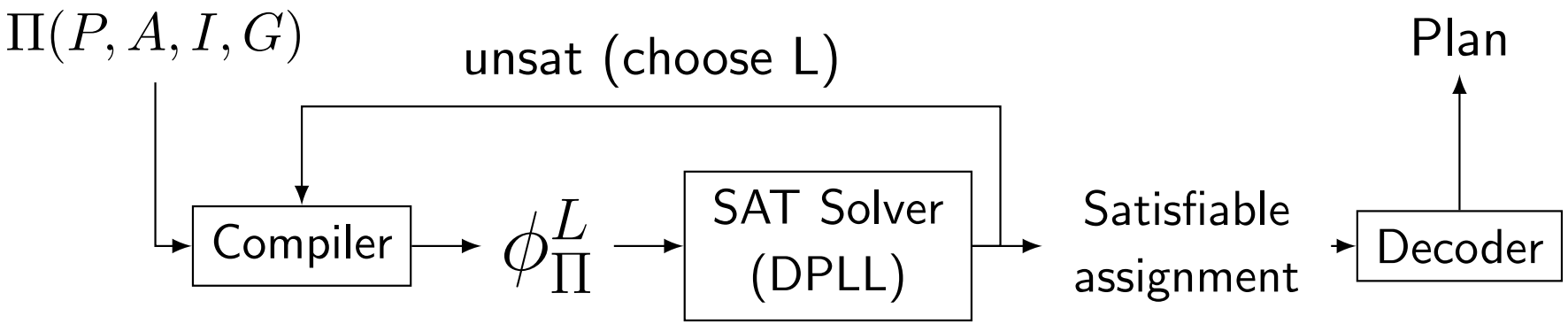
Theorem: Let s_k be a state, and A_k a set of actions that do not interfere and are applicable on s_k. Then, any ordering of A_k is executable in s_k and results in a unique state s_{k+1}.

Proof sketch: No precondition is deleted, and no fact is added and deleted by different actions in A_k. □

$\rightarrow$**Remark:** With this encoding ϕ_{A_k} has quadratic size in the number of actions. A better (linear) encoding exists [Rintanen *et al.* (2006)].

Planning as SAT

ϕ_{Π}^{L}: Given a planning task $\Pi = (P, A, I, G)$, does there exist a plan of "length" L or less?



Two separate questions:

1. How to construct $\phi_{\Pi,L}$?
2. How to choose L?

How to Choose L?

Given a planning task $\Pi = (P, A, I, G)$, does there exist a plan of length L or less?

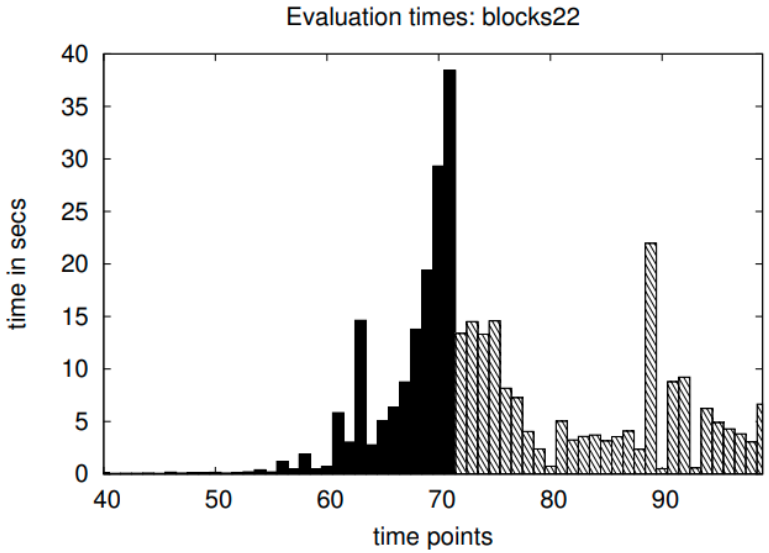
What value should we choose for L?

- If L is too small ($L < h^*$ for the sequential encoding), the formula will be unsatisfiable
  $\rightarrow$We do not gain any information
- Size of formula is linear on L, but SAT is NP-hard $\rightarrow$Time of solving the formula may be much larger if we choose the wrong L!

How to Choose L?

Multiple Strategies:

- Sequentially: 1 2 3 4 5 6 ...
  - Guaranteed to find the solution with minimum number of steps
  - Issue: sometimes the last unsatisfiable step is much harder!



Rintanen *et al.* **?** (2006)

- Parallel: Start n SAT solvers simultaneously 1, 2, 3, 4, 5, 6, ...
- Parallel Exponential: Solve horizon lengths $1, 2, 4, 8, 16, \ldots$ in parallel, up to some finite length (some hundreds or thousands), with the same amount of CPU assigned to each horizon length

Some Remarks

1. Theory $\rightarrow$ Practice:
   - $\rightarrow$ Cook-Levin Theorem (c.f. Computability and Complexity course) does not seem "practical" but similar ideas can be used to solve practical problems using SAT!
2. SAT solvers can be useful to solve problems beyond NP!
3. Performance can benefit a lot from encoding state-invatiants in the BDDs and/or TR (e.g. A truck cannot be in two locations at the same time: $\neg(at(t, l_1) \wedge at(t, l_2))$)

Summary

Can't we just use SAT and/or BDDs to solve planning problems? $\rightarrow$ Yes, we can!

- Symbolic Search
  - Search by exploring sets of states (instead of individual states)
  - Use BDDs to efficiently represent and manipulate sets of states
  - Image computation: Operation for expanding a set of states
  - Very useful for exhaustive exploration (optimal planning)

- Planning as SAT
  - We can construct a formula $phi_{\Pi,L}$ that is satisfiable if and only if Π has a solution on L steps.
  - Sequential and parallel encodings
  - Very useful for satisficing planning

Alternatives to Heuristic Search (not all covered here)

- **Planning as SAT**: Extensions use, e.g., heuristics, symmetry breaking.
  Kautz and Selman (1992, 1996); Ernst *et al.* (1997); Rintanen (1998, 2003, 2012)
- **Symbolic Search**: (originated in Model Checking McMillan (1993)) Edelkamp and Helmert (2001); Kissmann and Edelkamp (2011); Torralba *et al.* (2017)
- **Property Directed Reachability**
  Bradley (2011); Eén *et al.* (2011); Suda (2014)
- Planning via **Petri Net Unfolding**
  Godefroid and Wolper (1991); McMillan (1992); Esparza *et al.* (2002); Edelkamp *et al.* (2004); Hickmott *et al.* (2007); Bonet *et al.* (2008, 2014)
- **Partial-order Planning**
  Sacerdoti (1975); Kambhampati *et al.* (1995); Younes and Simmons (2003); Bercher *et al.* (2013)
- **Factored Planning**
  Knoblock (1994); Amir and Engelhardt (2003); Brafman and Domshlak (2006); Kelareva *et al.* (2007); Brafman and Domshlak (2008, 2013); Fabre *et al.* (2010)
- **Decoupled Search**: Gnad and Hoffmann (2018)
- ...

References I

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

Tomás Balyo. Relaxing the relaxed exist-step parallel planning semantics. pages 865–871. IEEE Computer Society, 2013.

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

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

Blai Bonet, Patrik Haslum, Victor Khomenko, Sylvie Thiébaux, and Walter Vogler. Recent advances in unfolding technique. *Theoretical Computer Science*, 551:84–101, 2014.

References II

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

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

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

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

Stefan Edelkamp and Malte Helmert. MIPS: The model checking integrated planning system. *AI Magazine*, 22(3):67–71, 2001.

References III

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(4):277–301, 2004.

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

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

Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.

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

References IV

Daniel Gnad and Jörg Hoffmann. Star-topology decoupled state space search. *Artificial Intelligence*, 257:24 – 60, 2018.

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

Sarah L. Hickmott, Jussi Rintanen, Sylvie Thiébaux, and Langford B. White. Planning via petri net unfolding. In Veloso **?**, pages 1904–1911.

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(1–2):167–238, July 1995.

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

Henry A. Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In William J. Clancey and Daniel Weld, editors, *Proceedings of the 13th National Conference of the American Association for Artificial Intelligence (AAAI'96)*, pages 1194–1201, Portland, OR, July 1996. MIT Press.

References V

Elena Kelareva, Olivier Buffet, Jinbo Huang, and Sylvie Thiébaux. Factored planning using decomposition trees. In Veloso **?**, pages 1942–1947.

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

Craig Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302, 1994.

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

Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12-13):1031–1080, 2006.

References VI

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

Jussi Rintanen. Symmetry reduction for SAT representations of transition systems. In Enrico Giunchiglia, Nicola Muscettola, and Dana Nau, editors, *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, pages 32–41, Trento, Italy, 2003. AAAI Press.

Jussi Rintanen. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193:45–86, 2012.

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

Martin Suda. Property directed reachability for automated planning. *Journal of Artificial Intelligence Research*, 50:265–319, 2014.

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

References VII

Manuela Veloso, editor. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, Hyderabad, India, January 2007. Morgan Kaufmann.

Martin Wehrle and Jussi Rintanen. Planning as satisfiability with relaxed $-step plans. volume 4830 of *Lecture Notes in Computer Science*, pages 244–253. Springer, 2007.

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