

# Classical Planning Algorithms

## 3. Planning Algorithms

Malte Helmert

ICAPS 2018 Summer School

June 21, 2018

# The Big Three

# Classical Planning Algorithms

## In This Part

very high-level overview of classical planning algorithms

- **bird's eye view**: no details, just some very brief ideas
- some in-depth coverage in Gabi's presentation tomorrow
- basic literature pointers provided, but please get in touch during or after the summer school if you want more!

# The Big Three

Of the many planning approaches, three techniques stand out:

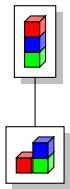
- ① heuristic state-space search
- ② SAT planning
- ③ symbolic search

# Heuristic State-Space Search

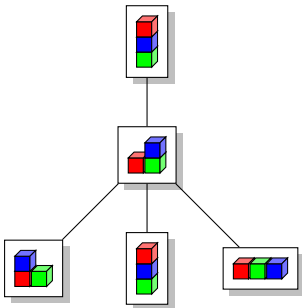
# State-Space Search in a Nutshell



# State-Space Search in a Nutshell

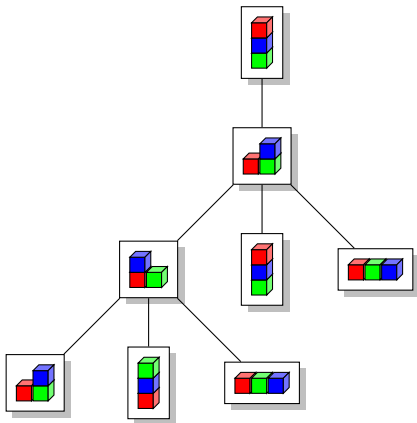


# State-Space Search in a Nutshell

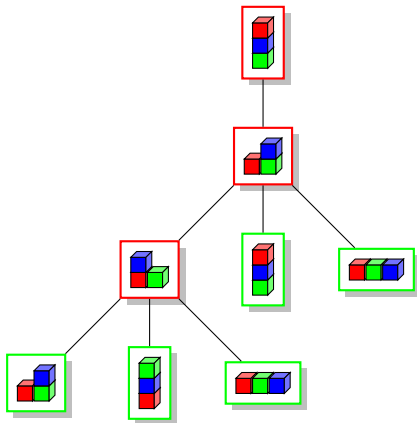




# State-Space Search in a Nutshell



# State-Space Search in a Nutshell



Closed node

Open node

Which open node  
should we select  
for expansion?

# Heuristic Search

- prioritize open nodes with **heuristic**
- **heuristic**
  - **estimates cost** of path from state **to closest goal state**
  - $h : S \rightarrow \mathbb{R}_0^+$
- search algorithms differ in how they exploit the heuristic:
  - $h$ : heuristic estimate of state
  - $g$ : cost of path from initial state to open node
  - **greedy best-first search**: expand node with minimum  $h$
  - $A^*$ : expand node with minimum  $g + h$

# Planning Heuristics

The central question for heuristic search approaches to classical planning:

How do we find good heuristics in a domain-independent way?

↪ Gabi's presentation tomorrow

# SAT Planning

# SAT Planning: Basic Ideas

- formalize problem of finding plan **with a given horizon** (length bound) as a **propositional satisfiability problem** and feed it to a generic SAT solver
- to obtain a (semi-) complete algorithm, try with increasing horizons until a plan is found (= the formula is satisfiable)
- **important optimization**: allow applying several non-conflicting actions “at the same time” so that a shorter horizon suffices

# SAT Formulas: Variables

- assume STRIPS encoding  $\langle V, I, G, A \rangle$
- given **horizon**  $T \in \mathbb{N}_0$

## Variables of SAT Encoding

- propositional variables  $v^i$  for all  $v \in V$ ,  $0 \leq i \leq T$   
encode **state after  $i$  steps** of the plan
- propositional variables  $a^i$  for all  $a \in A$ ,  $1 \leq i \leq T$   
encode **actions applied in  $i$ -th step** of the plan

# SAT Formulas: Clauses

- assume STRIPS encoding  $\langle V, I, G, A \rangle$
- given **horizon**  $T \in \mathbb{N}_0$

## Clauses of SAT Encoding

- unit clauses encoding **initial state**:  
 $v^0$  for all  $v \in I$  and  $\neg v_0$  for all  $v \notin I$
- unit clauses encoding **goal**:  
 $v^T$  for all  $v \in G$
- ...



# SAT Formulas: Clauses

- assume STRIPS encoding  $\langle V, I, G, A \rangle$
- given **horizon**  $T \in \mathbb{N}_0$

## Clauses of SAT Encoding

for all  $1 \leq i \leq T$ :

- subformulas encoding **action preconditions**:

$$a^i \rightarrow \bigwedge_{v \in \text{pre}(a)} v^{i-1}$$

- subformulas encoding **action conflicts**:

$$\neg(a^i \wedge b^i) \text{ for all } a, b \in A \text{ with } a \neq b, \\ (\text{pre}(a) \cup \text{add}(a)) \cap \text{del}(b) \neq \emptyset$$

- subformulas encoding **successor states**:

$$v^i \leftrightarrow \left( \bigvee_{a \in A: v \in \text{add}(a)} a^i \vee (v^{i-1} \wedge \neg \bigvee_{a \in A: v \in \text{del}(a)} a^i) \right)$$

# Advanced SAT Planning

Much fancier SAT encodings and SAT planning techniques exist.

- e.g., **Madagascar planner** (Rintanen, IPC 2011 & 2014)
- many papers by Jussi Rintanen (e.g., Rintanen; AIJ 2012)
- related: **property-directed reachability** (Suda, JAIR 2014)

# Symbolic Search

# Symbolic Search Planning: Basic Ideas

- search processes **sets of states** at a time
- operators, goal states, state sets reachable with a given cost etc. represented by **binary decision diagrams (BDDs)** (or related data structures)
- **hope**: exponentially large state sets can be represented as polynomially sized BDDs, which can be efficiently processed
- perform **symbolic Dijkstra search** on these set representations

# Symbolic Breadth-First Progression Search

simple symbolic search for unit-cost problems:

## Progression Breadth-first Search

```
def bfs-progression( $V, I, O, G$ ):  
     $goal\_states := conjunction(G)$   
     $reached_0 := \{I\}$   
     $i := 0$   
    loop:  
        if  $reached_i \cap goal\_states \neq \emptyset$ :  
            return solution found  
         $reached_{i+1} := reached_i \cup apply(reached_i, O)$   
        if  $reached_{i+1} = reached_i$ :  
            return no solution exists  
         $i := i + 1$ 
```

# Symbolic Breadth-First Progression Search

simple symbolic search for unit-cost problems:

## Progression Breadth-first Search

```
def bfs-progression( $V, I, O, G$ ):  
     $goal\_states := conjunction(G)$   
     $reached_0 := \{I\}$   
     $i := 0$   
    loop:  
        if  $reached_i \cap goal\_states \neq \emptyset$ :  
            return solution found  
         $reached_{i+1} := reached_i \cup apply(reached_i, O)$   
        if  $reached_{i+1} = reached_i$ :  
            return no solution exists  
         $i := i + 1$ 
```

↪ If we can implement operations *conjunction*,  $\{I\}$ ,  $\cap$ ,  $\neq \emptyset$ ,  $\cup$ , *apply* and  $=$  efficiently, this is a reasonable algorithm.

# Symbolic Search Planning: Some Pointers

- symbolic planning systems:
  - **Gamer** (Edelkamp & Kissmann, IPC 2008)
  - **SymBA\*** (Torralba et al., IPC 2014)
- comprehensive treatment:  
Álvaro Torralba's PhD thesis (Torralba, 2015)
- red-hot paper using **EVMDDs** for symbolic search at ICAPS next week (Speck et al., 2018)

# Summary



# Summary

three major algorithmic approaches to classical planning:

- heuristic state-space search
- SAT planning
- symbolic search