Planning and Optimization

G3. Symbolic Search: Uniform-cost and A* search

Malte Helmert and Gabriele Röger

Universität Basel

December 18, 2017

Planning and Optimization
December 18, 2017 — G3. Symbolic Search: Uniform-cost and A* search

G3.1 Introduction

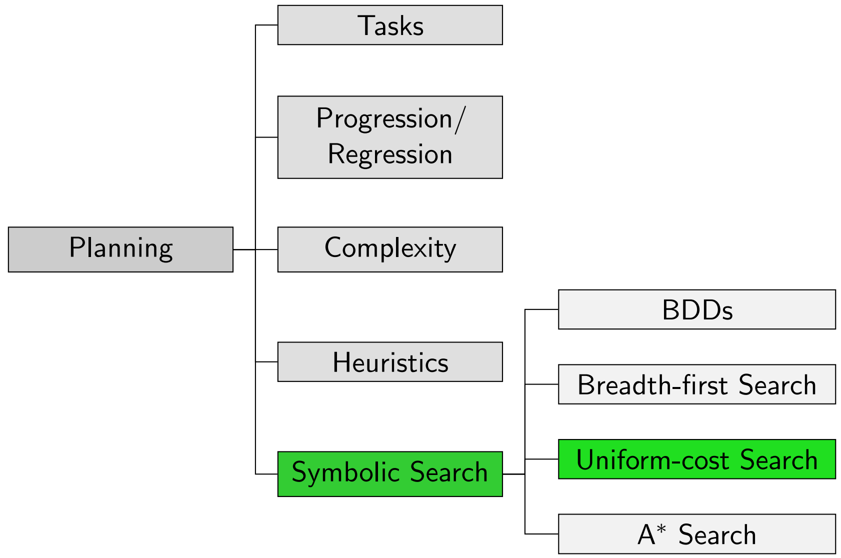
Introduction

- Previous chapter: Symbolic breadth-first search
- Optimal plans only guaranteed for unit-cost tasks (= all operators same cost)
- Optimal planning in explicit-state forward search:
  - (uninformed) uniform-cost search
  - (informed) A* search
  - ...

Analogous algorithms for symbolic (BDD-based) search?

G3.2 Symbolic Uniform-Cost Search

Content of this Course



Cost-separated Transition Relations

- Previously: one transition relation $T_V(O)$ for all operators
- Now: several transition relations for operators of same cost
- Set $\mathcal{T}$ of pairs (T, c), where T is a transition relation for one/some/all operators of cost c
  - All operators must be covered (and nothing else):
    $\bigcup_{(T,c)\in\mathcal{T}} r(T) = r(T_V(O))$
  - The cost must be correct:
    For $(T, c) \in \mathcal{T}$: if $t \in r(T)$ then $t \models \bigvee_{o\in O:cost(o)=c} \tau_V(o)$

Many possibilities to split up $T_V(O)$ (discussed later)

Image Computation

- The apply function (previous chapter) computes the set of states S' that can be reached from a set of states S by applying one operator.
- This is called the image of S wrt. transition relation $T_V(O)$.
- Now: image computation for arbitrary transition relations.

```
def image(B, T):
    B := bdd-intersection(B, T)
    for each v ∈ V:
        B := bdd-forget(B, v)
    for each v ∈ V:
        B := bdd-rename(B, v', v)
    return B
```

Exactly like apply but gets transition relation as argument.

Symbolic Uniform-Cost Search (Positive Operator Costs)

```
def symbolic-uniform-cost(V, I, O, γ):
    goal := build-BDD(γ)
    𝒯 := make-transition-relations(V, O)
    open_0 := bdd-state(I)
    while ∃g : open_g ≠ 0:
        g := min{g | open_g ≠ 0}
        closed_g := open_g
        if bdd-intersection(open_g, goal) ≠ 0:
            return construct-plan(I, O, goal, closed_*, g)
        for all (T, c) ∈ 𝒯:
            open_{g+c} := bdd-union(open_{g+c},
                                    image(open_g, T))
        open_g := 0
    return unsolvable
```


Pre-image Computation

- The image of S wrt. transition relation T computes the set of states that can be reached from S by applying a transition represented by T.
- The pre-image of S wrt. T is the set of states from which we can reach S by applying a transition represented by T.

```
def pre-image(B, T):
    for each v ∈ V:
        B := bdd-rename(B, v, v')
    B := bdd-intersection(B, T)
    for each v ∈ V:
        B := bdd-forget(B, v')
    return B
```


Plan Extraction (Positive Operator Costs)

```
def construct-plan(I, O, goal, closed_*, g):
    cut := bdd-intersection(goal, closed_g)
    init := bdd-state(I)
    π := ⟨⟩
    while bdd-intersection(cut, init) = 0:
        for o ∈ O:
            pre := pre-image(cut, τ_V(o))
            if c := bdd-intersection(pre, closed_{g−cost(o)}) ≠ 0:
                cut := c
                g := g − cost(o)
                π := ⟨o⟩π
                break
    return π
```


Zero-cost Operators

What is the problem with zero-cost operators?

- Search: could re-open $open_g$ after it was moved to $closed_g$, possibly running into an infinite loop
  → Apply all zero-cost operators before closing
- Plan extraction: could loop in zero-cost cycles
  → special treatment

Breadth-first Exploration with Zero-cost Operators

def bfs-zero(B, g, $\mathcal{T}$, *goal*):
    $i := 0$
    $closed_{g,i} := B$
    while $B \neq \mathbf{0}$ and $bdd\text{-}intersection(B, goal) = \mathbf{0}$:
        $B' := \mathbf{0}$
        for $\langle T, c\rangle \in \mathcal{T}, c = 0$:
            $B' := bdd\text{-}union(B', image(B, T))$
        $B := bdd\text{-}intersection(B', bdd\text{-}complement(closed_{g,i}))$
        $i := i + 1$
        $closed_{g,i} := bdd\text{-}union(B, closed_{g,i-1})$
    return $closed_{g,i}$

Symbolic Uniform-Cost Search

def symbolic-uniform-cost(V, I, O, γ):
    $goal := build\text{-}BDD(\gamma)$
    $\mathcal{T} := make\text{-}transition\text{-}relations(V, O)$
    $open_0 := bdd\text{-}state(I)$
    while $\exists g : open_g \neq \mathbf{0}$:
        $g := \min\{g \mid open_g \neq \mathbf{0}\}$
        $open_g := bfs\text{-}zero(open_g, g, \mathcal{T}, goal)$
        $closed_g := open_g$
        if $bdd\text{-}intersection(open_g, goal) \neq \mathbf{0}$:
            return $construct\text{-}plan(I, O, goal, closed_*, g)$
        for all $\langle T, c\rangle \in \mathcal{T}$ with $c > 0$:
            $open_{g+c} := bdd\text{-}union(open_{g+c},$
                                    $image(open_g, T))$
        $open_g := \mathbf{0}$
    return unsolvable

Plan Extraction with Zero-cost Operators

Needs all closed sets form bfs-zero and symbolic-uniform-cost.

def construct-plan(I, O, *goal*, $closed_{*,*}$, g):
    $cut := bdd\text{-}intersection(goal, closed_g)$
    $init := bdd\text{-}state(I)$; $\pi := \langle\rangle$
    while $bdd\text{-}intersection(cut, init) = \mathbf{0}$:
        $cut, \pi := get\text{-}to\text{-}bfs\text{-}level\text{-}0(cut, g, closed_{g,*}, \pi, O)$
        if $g = 0$:
            return π
        for $o \in O$ with $cost(o) > 0$:
            $pre := pre\text{-}image(cut, \tau_V(o))$
            if $c := bdd\text{-}intersection(pre, closed_{g-cost(o)}) \neq \mathbf{0}$:
                $cut := c$; $\pi := \langle o\rangle\pi$
                $g := g - cost(o)$
                break
    return π

Plan Extraction: Zero-Cost Plan Fragment

def $get\text{-}to\text{-}bfs\text{-}level\text{-}0(cut, g, closed_{g,*}, \pi, O)$:
    $level := 0$
    while $bdd\text{-}intersection(cut, closed_{g,level}) = \mathbf{0}$:
        $level := level + 1$
    while $level \neq 0$:
        for $o \in O$ with $cost(o) = 0$:
            $pre := pre\text{-}image(cut, \tau_V(o))$
            if $c := bdd\text{-}intersection(pre, closed_{g,level-1}) \neq \mathbf{0}$:
                $cut := c$
                $level := level - 1$
                $\pi := \langle o\rangle\pi$
                break
    return cut, π

Pruning of Closed States

- In explicit-state uniform-cost search, we never re-expand closed states.
- We can easily introduce such pruning in symbolic uniform-cost search.

Uniform-Cost Search with Pruning of Closed States

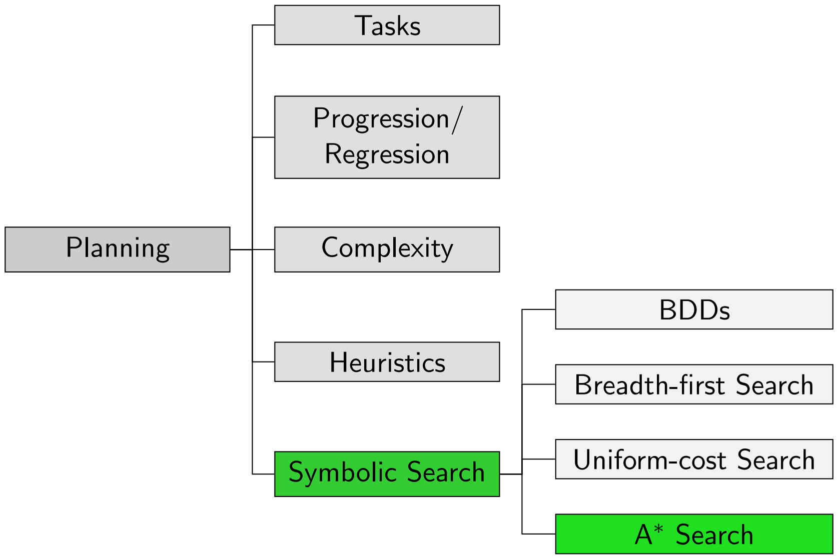
```
def symbolic-uniform-cost(V, I, O, γ):
    goal := build-BDD(γ)
    T := make-transition-relations(V, O)
    open_0 := bdd-state(I)
    while ∃g : open_g ≠ 0:
        g := min{g | open_g ≠ 0}
        open_g := bfs-zero(open_g, g, T, goal, closed_*)
        closed_g := open_g
        if bdd-intersection(open_g, goal) ≠ 0:
            return construct-plan(I, O, goal, closed_*, g)
        for all (T, c) ∈ T with c > 0:
            open_{g+c} := bdd-union(open_{g+c},
                                    image(open_g, T))
        open_g := 0
    return unsolvable
```


bfs-zero with Pruning of Closed States

```
def bfs-zero(B, g, T, goal, prune):
    for P ∈ prune:
        B := bdd-intersection(B, bdd-complement(P))
    i := 0
    closed_{g,i} := B
    while B ≠ 0 and bdd-intersection(B, goal) = 0:
        B' := 0
        for (T, c) ∈ T, c = 0:
            B' := bdd-union(B', image(B, T))
        B := bdd-intersection(B', bdd-complement(closed_{g,i}))
        for P ∈ prune:
            B := bdd-intersection(B, bdd-complement(P))
        i := i + 1
        closed_{g,i} := bdd-union(B, closed_{g,i-1})
    return closed_{g,i}
```


G3.3 Symbolic A*

Content of this Course



Symbolic A*

- Difference between explicit-state uniform-cost search and A*: heuristic to guide search
- $f = g + h$
- Analogously in symbolic search
- Heuristic given as set *heur* of BDDs $heur_h$ for each heuristic estimate h

Symbolic A* (with Consistent Heuristic)

def symbolic-AStar(V, I, O, γ, $heur$):
    $goal := \textit{build-BDD}(\gamma)$
    $\mathcal{T} := \textit{make-transition-relations}(V, O)$
    $open_{0,h(I)} := \textit{bdd-state}(I)$
    while $\exists g, h : open_{g,h} \neq \mathbf{0}$:
        $f := \min\{f \mid \exists g, h : open_{g,h} \neq \mathbf{0}, f = g + h\}$
        $g := \min\{g \mid \exists h : open_{g,h} \neq \mathbf{0}, f = g + h\}$
        $open_{g,*} := \textit{expand}_0(open_{*,*}, g, h, \mathcal{T}, goal, heur, closed_*)$
        $closed_g := \textit{bdd-union}(closed_g, open_{g,h})$
        if $\textit{bdd-intersection}(open_{g,h}, goal) \neq \mathbf{0}$:
            return $\textit{construct-plan}(I, O, goal, closed_*, g)$
        $open_{*,*} := \textit{expand}_{>0}(open_{*,*}, g, h, \mathcal{T}, heur)$
        $open_{g,h} := \mathbf{0}$
    return unsolvable

For performance it is important to expand the minimum g value.

Expand States and Update Open Lists

def expand$_0$($open_{*,*}$, g, h, $\mathcal{T}$, $goal$, $heur$, $prune$):
    $\textit{open-zero} := \textit{bfs-zero}(open_{g,h}, (g, h), \mathcal{T}, goal, prune)$
    for $heur_{h'} \in heur, h \leq h' < \infty$:
        $B' := \textit{bdd-intersection}(heur_{h'}, \textit{open-zero})$
        $open_{g,h'} := \textit{bdd-union}(open_{g,h'}, B')$
    return $open_{g,*}$

def expand$_{>0}$($open_{*,*}$, g, h, $\mathcal{T}$, $heur$):
    for all $(T, c) \in \mathcal{T}, c > 0$:
        $successors := image(open_{g,h}, T)$
        for $heur_{h'} \in heur, h - c \leq h' < \infty$:
            $B' := \textit{bdd-intersection}(heur_{h'}, successors)$
            $open_{g+c,h'} := \textit{bdd-union}(open_{g+c,h'}, B')$
    return $open_{*,*}$

Heuristics

How can we generate symbolic heuristics?

- Symbolic Pattern Databases
  - Uniform-cost search can easily be adapted to regression search.
  - Can search backwards in abstract transition systems
  - BDD for closed states with (backwards-) g-value i is heuristic BDD for $h = i$.
- Merge-and-Shrink Abstractions
  - Algebraic Decision Diagrams are like BDDs but sink nodes are labeled with arbitrary numbers.
  - Can map states to numbers.
  - Cascading tables of merge-and-shrink heuristics with linear merge strategy can efficiently be transformed into an ADD.
  - Result can be used in symbolic search instead of BDD set.

G3.4 Discussion

Importance of Variable Ordering

- For good performance, we need a good variable ordering.
  - Variables that refer to the same state variable before and after operator application (v and v') should be neighbors in the transition relation BDD.
- This is important for the performance of *BDD-rename* in the *image* and *pre-image* computation.

Transition Relations in $\mathcal{T}$

- We only required that all operators are represented by some $(T, c) \in \mathcal{T}$ and that the costs are correct.
- Extreme cases:
  - One element $(\tau_V(o), cost(o))$ for each operator o
  - Only one element for each operator cost, covering all operators of that cost.
- Trade-off:
  - Large number of entries leads to large number of image computations.
  - Size of T can grow exponentially with number of covered operators.
- There exist different aggregation strategies.

Performance

- In symbolic planning, blind search is often better than informed search.
- Practical implementations also perform regression or bidirectional search.
- This is only a minor modification of uniform-cost search.

G3.5 Summary

Summary

- Symbolic search operates on sets of states instead of individual states as in explicit-state search.
- State sets and transition relations can be represented as BDDs.
- A good variable ordering and an efficient image computation are crucial for performance.

Literature I

Randal E. Bryant.
Graph-Based Algorithms for Boolean Function Manipulation.
IEEE Transactions on Computers 35.8, pp. 677–691, 1986.
Reduced ordered BDDs.

Kenneth L. McMillan.
Symbolic Model Checking.
PhD Thesis, 1993.
Symbolic search with BDDs.

Literature II

Stefan Edelkamp and Frank Reffel.
OBDDs in Heuristic Search.
Proc. KI 1998, pp. 81–92, 1998.
Symbolic A*.

Stefan Edelkamp.
Symbolic Pattern Databases in Heuristic Search Planning.
Proc. AIPS 2002, pp. 274–283, 2002.
Symbolic PDB heuristics.

Literature III

Álvaro Torralba, Carlos Linares López, and Daniel Borrajo.
Symbolic Merge-and-Shrink for Cost-Optimal Planning.
Proc. IJCAI 2013, pp. 2394–2400, 2013.
Symbolic merge-and-shrink heuristics.

Álvaro Torralba.
Symbolic Search and Abstraction Heuristics for Cost-Optimal Planning.
PhD Thesis, 2015.
Aggregation strategies for transition relations and good overview of state of the art.