

Planning

Jussi Rintanen

NICTA, Canberra

August 13, 2008

Introduction

Definition

Heuristics

Conclusion

What is planning?

- **Domain-independent** language for **describing** search problems.
- **Domain-independent** algorithms for **solving** search problems.

Introduction

Definition

Heuristics

Conclusion

What is planning?

Definition:

- Choose **actions** and their **ordering**, to achieve a pre-defined goal.

Application areas:

- high-level planning for intelligent robots
- autonomous systems: NASA Deep Space One, ...
- production planning
- problem-solving (games like Rubik's cube)

Introduction

Definition

Heuristics

Conclusion

Why is planning difficult?

- Solutions to simplest planning problems are **paths from an initial state to a goal state** in the transition graph. Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
- Q: Why don't we solve all planning problems this way?
- A: State spaces are often huge: $10^9, 10^{12}, 10^{15}, \dots$ states. Constructing the transition graph explicitly is not feasible!!
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

Introduction

Definition

Heuristics

Conclusion

Why is planning difficult?

- Solutions to simplest planning problems are **paths from an initial state to a goal state** in the transition graph. Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
- Q: Why don't we solve all planning problems this way?
- A: State spaces are often huge: $10^9, 10^{12}, 10^{15}, \dots$ states. Constructing the transition graph explicitly is not feasible!!
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

Introduction

Definition

Heuristics

Conclusion

Why is planning difficult?

- Solutions to simplest planning problems are **paths from an initial state to a goal state** in the transition graph. Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
- Q: Why don't we solve all planning problems this way?
- A: State spaces are often huge: $10^9, 10^{12}, 10^{15}, \dots$ states. Constructing the transition graph explicitly is not feasible!!
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

- 1 Generic search algorithms (A^* , ...) and **automatically derived** heuristics.
- 2 **Constraint-based** methods (planning as CSP, SAT)
- 3 **Symbolic** methods based on **Binary Decision Diagrams** and similar data structures (symbolic breadth-first search, symbolic heuristic search)

Representation of transition systems

- state = valuation of a **finite set** of state variables

Example

HOUR : $\{0, \dots, 23\} = 13$

MINUTE : $\{0, \dots, 59\} = 55$

LOCATION : $\{51, 52, 82, 101, 102\} = 101$

WEATHER : $\{\text{sunny, cloudy, rainy}\} = \text{cloudy}$

HOLIDAY : $\{T, F\} = F$

- Any n -valued state variable can be represented by $\lceil \log_2 n \rceil$ Boolean (2-valued) state variables.
- Actions change the values of the state variables.

Introduction

Definition

State variables

Actions

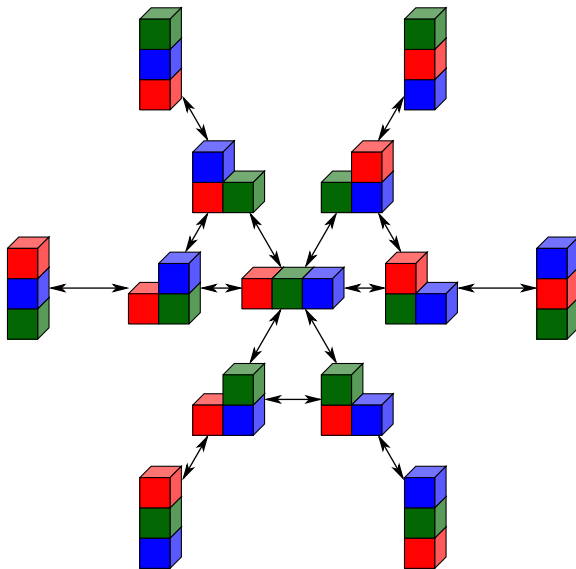
Plans

Heuristics

Conclusion

Blocks world

The transition graph for three blocks



Introduction

Definition

State variables

Actions

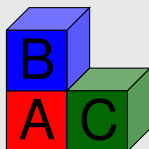
Plans

Heuristics

Conclusion

Blocks world with Boolean state variables

Example



$s(\text{clearA}) = 0$	$s(\text{clearB}) = 1$	$s(\text{clearC}) = 1$
$s(\text{AonB}) = 0$	$s(\text{AonC}) = 0$	$s(\text{AonTABLE}) = 1$
$s(\text{BonA}) = 1$	$s(\text{BonC}) = 0$	$s(\text{BonTABLE}) = 0$
$s(\text{ConA}) = 0$	$s(\text{ConB}) = 0$	$s(\text{ConTABLE}) = 1$

Not all valuations correspond to an intended state, e.g. if $s(\text{AonB}) = 1$ and $s(\text{BonA}) = 1$.

Introduction

Definition

State variables

Actions

Plans

Heuristics

Conclusion

Actions

Actions

An action $\langle p, e \rangle$ consists of

- 1 a **precondition** $v_1 = b_1, \dots, v_n = b_n$ where v_i are state variables and b_i are 0 or 1,
- 2 an **effect** $v_1 := b_1, \dots, v_m := b_m$ where v_i are state variables and b_i are 0 or 1.

Short-hands for conditions:

a for $a = 1$

$\neg a$ for $a = 0$

Short-hands for assignments:

a for $a := 1$

$\neg a$ for $a := 0$

Introduction

Definition

State variables

Actions

Plans

Heuristics

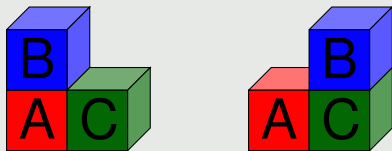
Conclusion

Actions

Example

Action that moves B from A onto C :

$\langle \{BonA, clearB, clearC\}, \{BonC, clearA, \neg BonA, \neg clearC\} \rangle$.



Introduction

Definition

State variables

Actions

Plans

Heuristics

Conclusion

Actions

The successor state of a state

Executability

An action $\langle p, e \rangle$ is **executable in a state** s iff the precondition p is true in s .

Successor states

The **successor state** $exec_o(s)$ of s with respect to $o = \langle p, e \rangle$ is obtained from s by making the assignments in e .

Example

$\langle \{a\}, \{\neg a, b\} \rangle$ is executable in state s that satisfies $a = 1, b = 1, c = 1$ because $s(a) = 1$.

Hence $a = 0, b = 1, c = 1$ hold in $exec_{\langle \{a\}, \{\neg a, b\} \rangle}(s)$.

Introduction

Definition

State variables

Actions

Plans

Heuristics

Conclusion

Planning problem

Transition system $\langle V, I, A, G \rangle$

- V is a finite set of **state variables**.
- I is an **initial state** (a valuation of V).
- A is a set of **actions** over V .
- G is a set of conditions over V , the **goals**.

Introduction

Definition

State variables

Actions

Plans

Heuristics

Conclusion

Plans

A **plan** for $\langle V, I, A, G \rangle$ is a sequence $\pi = o_1, \dots, o_n$ of actions such that $o_1, \dots, o_n \in A$ and there is a sequence of states s_0, \dots, s_n (the **execution** of π) so that

- 1 $s_0 = I$,
- 2 $s_i = \text{exec}_{o_i}(s_{i-1})$ for every $i \in \{1, \dots, n\}$, and
- 3 s_n satisfies G .

This can be equivalently expressed as

$$\text{exec}_{o_n}(\text{exec}_{o_{n-1}}(\dots \text{exec}_{o_1}(I) \dots)) \text{ satisfies } G.$$

Search algorithms: A*

Example



Introduction

Definition

State variables

Actions

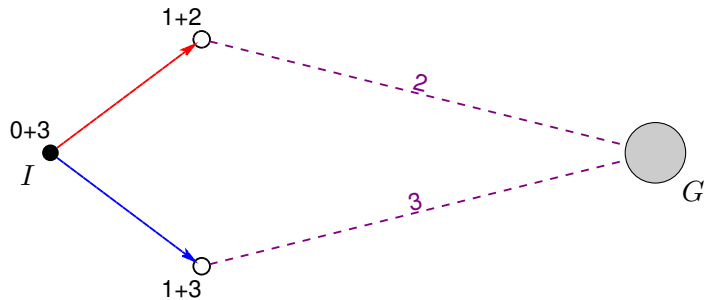
Plans

Heuristics

Conclusion

Search algorithms: A*

Example



Introduction

Definition

State variables

Actions

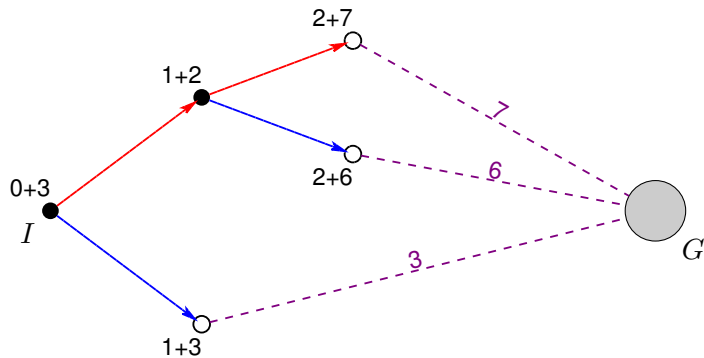
Plans

Heuristics

Conclusion

Search algorithms: A*

Example



Introduction

Definition

State variables

Actions

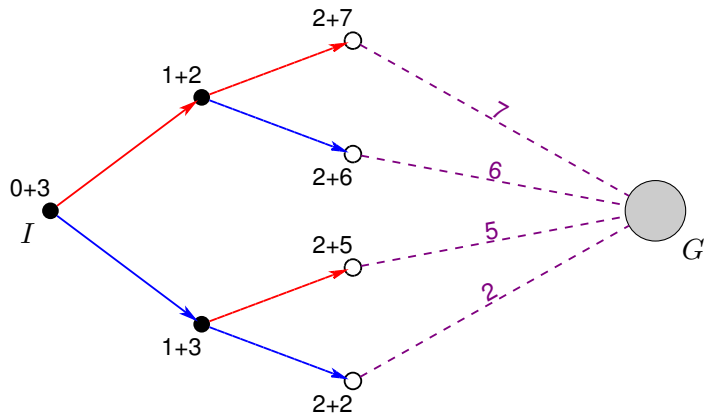
Plans

Heuristics

Conclusion

Search algorithms: A*

Example



Introduction

Definition

State variables

Actions

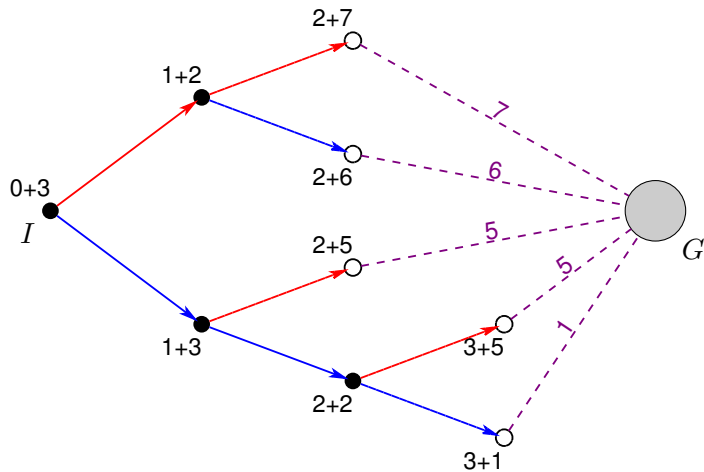
Plans

Heuristics

Conclusion

Search algorithms: A^*

Example



Introduction

Definition

State variables

Actions

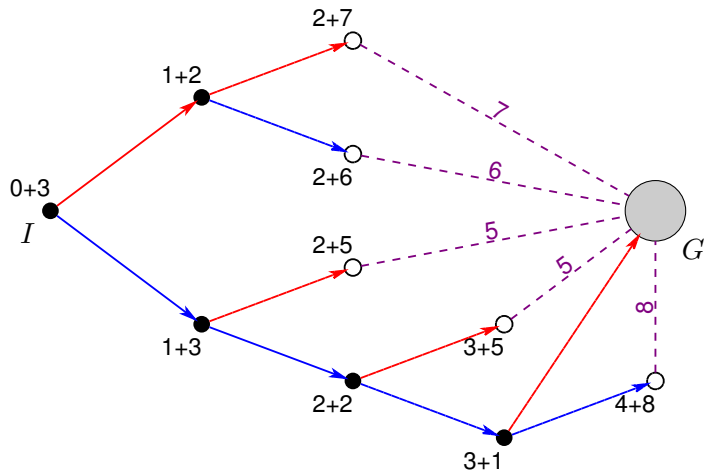
Plans

Heuristics

Conclusion

Search algorithms: A^*

Example



Introduction

Definition

State variables

Actions

Plans

Heuristics

Conclusion

Methods for constructing heuristics

Different forms of relaxation:

- 1 ignore **dependencies between variables**
- 2 project to a **subset of variables** and solve (optimally)

Introduction

Definition

Heuristics

Relaxations

Abstraction

Conclusion

Ignoring Dependencies between Variables

- Basic insight: compute a “distance” for each state variable, not individual states.
- This can be done in low-polynomial time in the size of the problem instance.

Introduction

Definition

Heuristics

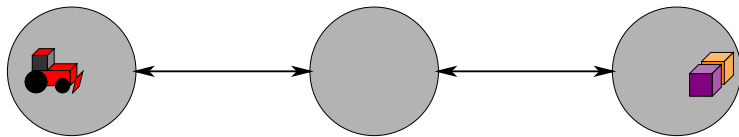
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



1 Tractor moves:

- from 1 to 2: $T_{12} = \langle \{T1\}, \{T2, \neg T1\} \rangle$
- from 2 to 1: $T_{21} = \langle \{T2\}, \{T1, \neg T2\} \rangle$
- from 2 to 3: $T_{23} = \langle \{T2\}, \{T3, \neg T2\} \rangle$
- from 3 to 2: $T_{32} = \langle \{T3\}, \{T2, \neg T3\} \rangle$

2 Tractor pushes A:

- from 2 to 1: $A_{21} = \langle \{T2, A2\}, \{T1, A1, \neg T2, \neg A2\} \rangle$
- from 3 to 2: $A_{32} = \langle \{T3, A3\}, \{T2, A2, \neg T3, \neg A3\} \rangle$

3 Tractor pushes B:

- from 2 to 1: $B_{21} = \langle \{T2, B2\}, \{T1, B1, \neg T2, \neg B2\} \rangle$
- from 3 to 2: $B_{32} = \langle \{T3, B3\}, \{T2, B2, \neg T3, \neg B3\} \rangle$

Introduction

Definition

Heuristics

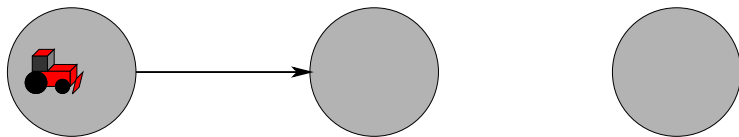
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Execute $T12 = \langle \{T1\}, \{T2, -T1\} \rangle$

Introduction

Definition

Heuristics

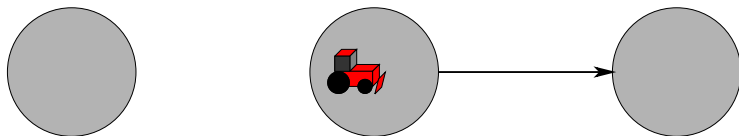
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Execute $T23 = \langle \{T2\}, \{T3, -T2\} \rangle$

Introduction

Definition

Heuristics

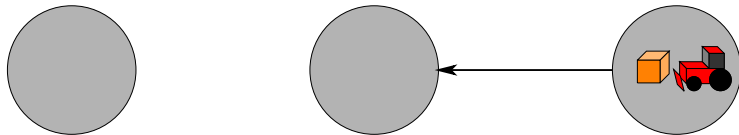
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Execute $A32 = \langle \{T3, A3\}, \{T2, A2, \neg T3, \neg A3\} \rangle$

Introduction

Definition

Heuristics

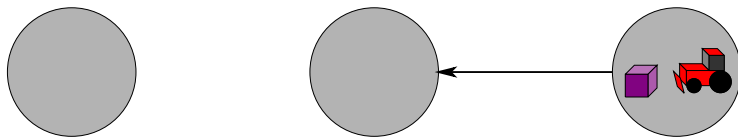
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Execute $B32 = \langle \{T3, B3\}, \{T2, B2, \neg T3, \neg B3\} \rangle$

Introduction

Definition

Heuristics

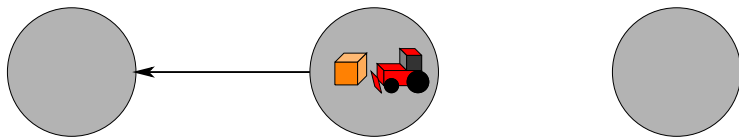
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Execute $A21 = \langle \{T2, A2\}, \{T1, A1, \neg T2, \neg A2\} \rangle$

Introduction

Definition

Heuristics

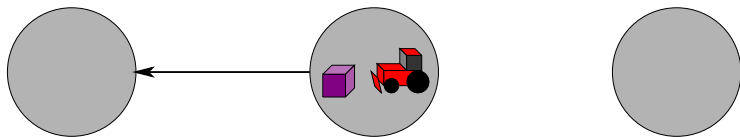
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Execute $B21 = \langle \{T2, B2\}, \{T1, B1, \neg T2, \neg B2\} \rangle$

Introduction

Definition

Heuristics

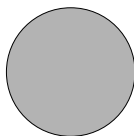
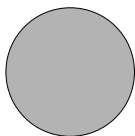
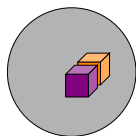
Relaxations

Abstraction

Conclusion

Distance estimation

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	1	0	0	0	0	1	0	0	1
1	10	10	0	0	0	1	0	0	1
2	10	10	10	0	0	1	0	0	1
3	10	10	10	0	10	10	0	10	10
4	10	10	10	10	10	10	10	10	10

Heuristic estimate of distance of $A1, B1$ is 4.

Actual distance of $A1, B1$ is 8.

Introduction

Definition

Heuristics

Relaxations

Abstraction

Conclusion

Abstraction Heuristics

Key observation

Eliminating any state variable can only reduce the length of the shortest plan.

- Any abstraction, with some variables eliminated, yields a smaller state space.
- Distances in the **abstract state space** are **lower bounds** for the distances in the state space itself.

Introduction

Definition

Heuristics

Relaxations

Abstraction

Conclusion

Abstraction Heuristics

The tractor example, abstracted to $\{A1, A2, A3, B1, B2, B3\}$ (eliminating the tractor) yields actions

1 Tractor moves:

- from 1 to 2: $T12 = \langle \{\}, \{\} \rangle$
- from 2 to 1: $T21 = \langle \{\}, \{\} \rangle$
- from 2 to 3: $T23 = \langle \{\}, \{\} \rangle$
- from 3 to 2: $T32 = \langle \{\}, \{\} \rangle$

2 Tractor pushes A:

- from 2 to 1: $A21 = \langle \{A2\}, \{A1, \neg A2\} \rangle$
- from 3 to 2: $A32 = \langle \{A3\}, \{A2, \neg A3\} \rangle$

3 Tractor pushes B:

- from 2 to 1: $B21 = \langle \{B2\}, \{B1, \neg B2\} \rangle$
- from 3 to 2: $B32 = \langle \{B3\}, \{B2, \neg B3\} \rangle$

The abstract state space has 9 states (as opposed to 27).
Reaching $A1, B1$ from the abstract initial state $A3, B3$ takes 4 abstract actions.

Introduction

Definition

Heuristics

Relaxations

Abstraction

Conclusion

Abstraction Heuristics

Aggregation of several abstractions

In practice it is only possible to use abstractions that retain only **very few state variables**. These typically yield **very weak lower bounds**.

Useful strategy: **aggregate** several abstractions.

- 1 **Maximum** of lower bounds from different abstractions
- 2 **Sum** of lower bounds from different abstractions, **provided that** no action **gets counted twice**.
- 3 More sophisticated aggregation methods exist.

Central problem: Which abstractions to aggregate?

Introduction

Definition

Heuristics

Relaxations

Abstraction

Conclusion

Conclusions

- Planning: problem-independent solution methods for search problems
- We considered **one solution method**: state-space search, with problem-independent methods for deriving admissible heuristics.
- Other solution methods exist (CSP, SAT, ...)
- Planning is more general than state-space search:
 - uncertainty (nondeterminism),
 - different types of objectives (goals vs. rewards),
 - rational, real timeleading to infinite state spaces, infinite executions, ...

Introduction

Definition

Heuristics

Conclusion