

Automatic Planning

Jörg Hoffmann

University of Innsbruck
DERI

Masterstudium Informatik Vertiefungsmodul (WM 9)

Automatic Planning

1. Introduction

- ▶ **What is AI “Planning”?**
- ▶ VL Goals and Prerequisites
- ▶ Transition Systems and the Reachability Problem
- ▶ Some Examples
- ▶ Search Methods
- ▶ VL Outline

What is AI “Planning”?

Some misunderstandings:

- ▶ Planning a lecture
- ▶ Planning to go out
- ▶ “Urban Planning” (= Verkehrsplanung und so)
 - Encyclopedia Britannica, entry 1
- ▶ “Economic Planning” (= Planwirtschaft)
 - Encyclopedia Britannica, entry 2
- ▶ “Planned Parenthood” (= Familienplanung . . .)
 - Encyclopedia Britannica, entry 3

“AI/Automatic Planning”? No hits. (Not in the first 20, at least.)

What is AI “Planning”?

So: what *is* AI Planning?

Suffers from similar ambiguities as the question of what
“Artificial Intelligence” is ...

- ▶ Historical view ...
- ▶ Idealistic AI view ...
- ▶ Pragmatic view ...

Common: a planning system is a *general*, not specific, problem solver. “Problems” are to find *solution paths in transition systems*.

What is AI “Planning”?

Historical view:

Planning is the process that takes decisions about action in a robot.

- ▶ Robots are supposed to react to a dynamic environment – particularly, *new* situations
- ▶ You don't want a human with a remote control, there's gotta be some “intelligent” reasoning about what's going on
- ▶ Long-term (=SciFi) application: Mars Rover
- ▶ Best application known to me so far: intelligent vacuum cleaners

What is AI “Planning”?

Idealistic AI view:

One of the distinguishing qualities of humans is their pronounced ability to anticipate the effects of their actions, and form plans to achieve their goals.

- ▶ Put simpler: confronted with some new situation (a new job, a new card game), we can all figure out an “OK” coarse of action pretty quickly
- ▶ How do we do that?

What is AI “Planning”?

Pragmatic view:

Planning is a flexible approach for taking complex decisions.

- ▶ Decide about the schedule of a production line
- ▶ Decide about the movements of an elevator
- ▶ Decide about the flow of paper through a (big) copy machine
- ▶ Decide about robot actions

“Flexible”: you just **describe the problem to the planning system, in a generic language. A (good) solution is found fully automatically.**

What is AI “Planning”?

Our view here: pragmatic.

- ▶ We look at generic “problem description languages”
- ▶ We look at methods to solve **any problem that can be described in the language**
- ▶ If the problem changes, all you need to change is its description
- ▶ Of course, flexibility comes with a loss of efficiency wrpto. specific solvers . . .
- ▶ . . . we don't expect to *beat* specialized solvers!

What is AI “Planning”?

The “easy” goal: **a method that efficiently solves (all) everyday-life problems.**

- ▶ Going shopping, vacuum-cleaning, . . .
- ▶ Already achieved, to a certain extent
- ▶ More formally: all problems in **P**

The “hard” goal: **a method that “competitively” solves (all) critical decision problems.**

- ▶ Production schedule, elevator moves, . . .
- ▶ Far from being achieved

Automatic Planning

1. Introduction

- ▶ What is AI “Planning”?
- ▶ **VL Goals and Prerequisites**
- ▶ Transition Systems and the Reachability Problem
- ▶ Some Examples
- ▶ Search Methods
- ▶ VL Outline

- ▶ Understand the principles behind the different solution (search) methods
- ▶ Get an impression of the state of the art
- ▶ Be able to understand research literature in AI Planning
- ▶ Be able to complete a project in this area, and in the related area of Web Service Composition

- ▶ Basic knowledge in CS
- ▶ Basic knowledge of complexity theory (**P**, **NP**, **PSPACE**)
- ▶ Basic understanding of AI, Search, and Logics:
recommended
- ▶ The lecture is relatively self-contained in these respects

Automatic Planning

1. Introduction

- ▶ What is AI “Planning”?
- ▶ VL Goals and Prerequisites
- ▶ **Transition Systems and the Reachability Problem**
- ▶ Some Examples
- ▶ Search Methods
- ▶ VL Outline

Transition Systems

A transition system is given by:

- ▶ A finite set V of (state) **variables** v (Booleans, finite-domain, natural numbers, reals, ...)
- ▶ A finite set A of **transition rules** a (=actions) of the form
$$condition(a) \longrightarrow effect(a)$$
- ▶ $condition(a)$: a formula specifying applicability
- ▶ $effect(a)$: an instruction how the variable values change

System **states**: value assignments.

Language: which kinds of state variables, which kinds of expressions in $condition(a)$ and $effect(a)$?

The Reachability Problem

- ▶ Assume we have a transition system, an **initial state** I , and a **goal condition** G
- ▶ The 3 together form a problem **instance**, or **task**

*Is there a **solution** (**plan**), i.e. a sequence of transitions that maps I into a state that satisfies G ?*

*If yes, what is a (**optimal**, e.g. shortest) solution?*

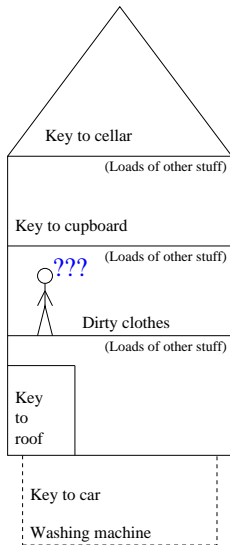
- ▶ The worst-case nr. of reachable states is $\prod_{v \in V} |domain(v)|$
- ▶ Turing machines can naturally be modeled as transition systems
- ▶ Even the simplest language known is **PSPACE**-complete

Automatic Planning

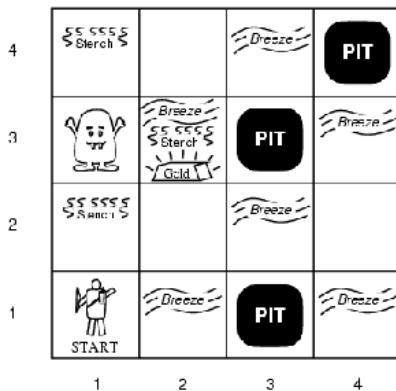
1. Introduction

- ▶ What is AI “Planning”?
- ▶ VL Goals and Prerequisites
- ▶ Transition Systems and the Reachability Problem
- ▶ **Some Examples**
- ▶ Search Methods
- ▶ VL Outline

“Easy” Example 1



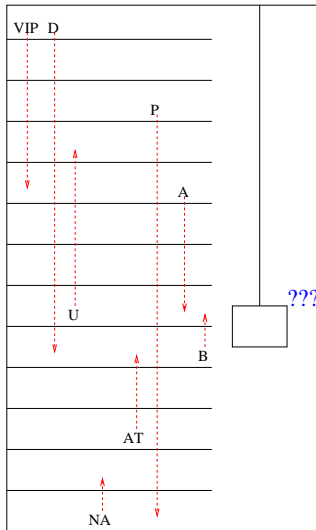
“Easy” Example 2



“Hard” Example 1



“Hard” Example 2



Automatic Planning

1. Introduction

- ▶ What is AI “Planning”?
- ▶ VL Goals and Prerequisites
- ▶ Transition Systems and the Reachability Problem
- ▶ Some Examples
- ▶ **Search Methods**
- ▶ VL Outline

Search Methods, *for* what?

There are several things to try to do:

1. Find out if there is, or if there isn't, a solution
2. Find an optimal solution
3. Find some solution

*These sound related, but make **a lot** of difference! The techniques best suited for each one are almost disjoint!*

- ▶ In AI Planning, one usually assumes that there is a solution
- ▶ We concentrate on 2. and 3.

Search Methods, *what?*

There are essentially three known search methods:

- ▶ **State space search** – also called “directed search”
- ▶ **Constraint-based search** – also called “undirected search”
- ▶ **Symbolic search**

What this is all about:

1. How do we make these methods efficient?
2. Which one is best suited for which examples?

State Space Search

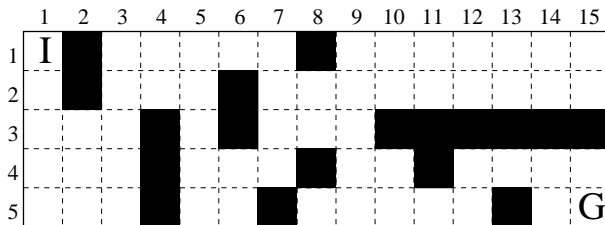
- ▶ Search **forward** from the initial state ...
- ▶ ... until a goal state is reached
- ▶ A **heuristic function** estimates goal distance

- ▶ Search **backward** from the goal condition ...
- ▶ ... until /the initial state is reached
- ▶ A **heuristic function** estimates initial state distance

- ▶ Prefer states with lower heuristic values

- ▶ “Directed” search: the search goes either forward or backward

State Space Search, Example



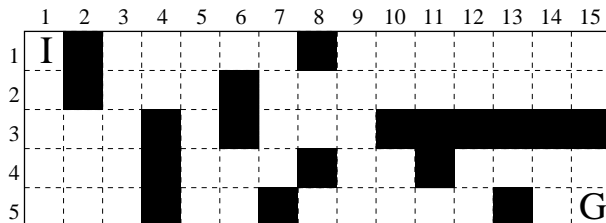
- ▶ Forward: $\{(1, 1)\}, (1, 2)\}, (1, 3)\}, (2, 3), (1, 4)\}, \dots$
- ▶ Backward: $\{(15, 5)\}, (14, 5), (15, 4)\}, (14, 4)\}, \dots$
- ▶ Possible heuristic function: $dx + dy$ (“Manhattan distance”)
- ▶ We will see a (general) heuristic that is *exact* here

Constraint-Based Search

- ▶ (Additionally) constrain the task by a **time horizon b**
- ▶ Formulate this as a **constraint satisfaction problem**
- ▶ Use **backtracking methods** to solve it: “do x at time t yes/no”, “do y at time t' yes/no”, ...
- ▶ If there is no solution, increment b
- ▶ Inside backtracking: **constraint propagation**, conflict analysis, ...

- ▶ “Undirected” search: there is no fixed time-order to the decisions done by backtracking

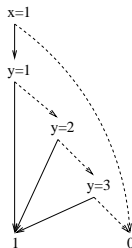
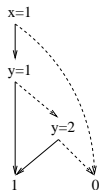
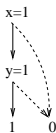
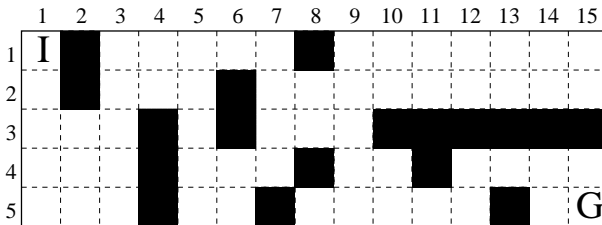
Constraint-Based Search, Example



- ▶ Variables: $x(t), y(t)$ for $0 \leq t \leq b$;
 $a(t) \in \{UP, DOWN, LEFT, RIGHT\}$ at $1 \leq t \leq b$
- ▶ $x(0) = 1, y(0) = 1; x(b) = 15, y(b) = 5$
- ▶ For all $1 \leq t \leq b$: $a(t) = DOWN \implies x(t) = x(t-1), y(t) = y(t-1) + 1$
- ▶ ...

- ▶ Forward or backward breadth-first search
- ▶ Represent set of reached states (by time t) through its characteristic function
- ▶ Most wide-spread: **BDDs** (Binary Decision Diagrams)
- ▶ Rooted DAG, nodes either:
 - ▶ 0 sons and labeled 0/1, or
 - ▶ 2 sons – 0 and 1 – and labeled with a variable
- ▶ Members of the set: those states that reach a label 1
- ▶ Main difficulty: find a good **variable order**

Symbolic Search, Example



- ▶ State space: state of the art for satisficing Planning (find some plan);
I implemented 8+ planners of this kind

Main topic

- ▶ Constraint-based: state of the art for optimal Planning (find an optimal plan);
I implemented 3+ planners of this kind

Will do as much as we got time left

- ▶ Symbolic: currently, not competitive in Planning;
I implemented 0 planners of this kind

Not covered by this VL

Automatic Planning

1. Introduction

- ▶ What is AI “Planning”?
- ▶ VL Goals and Prerequisites
- ▶ Transition Systems and the Reachability Problem
- ▶ Some Examples
- ▶ Search Methods
- ▶ **VL Outline**

1. Introduction
2. Planning Formalisms
- Part I. State Space Search**
3. Search Algorithms
4. The “Pattern Database” Relaxation
5. The “ k -Subsets” Relaxation
6. The “Monotonicity” Relaxation
7. Planning Systems, and the IPC
8. Local Search Topology

Part II. Constraint-Based Search

9. SAT and DPLL
10. Planning as Satisfiability
11. Clause Learning