Heuristic Search for Planning

Sheila McIlraith

University of Toronto

Acknowledgements

Many of the slides used in today's lecture are modifications of slides developed by Malte Helmert, Bernhard Nebel, and Jussi Rintanen.

Some material comes from papers by Daniel Bryce and Rao Kambhampati.

I would like to gratefully acknowledge the contributions of these researchers, a nd thank them for generously permitting me to use aspects of their presentation material.

Outline

A simple heuristic for deterministic planning

STRIPS (Fikes & Nilsson, 1971) used the number of state variables that differ in current state s and a STRIPS goal $l_1 \wedge \cdots \wedge l_n$:

$$h(s) := |\{i \in \{1, \ldots, n\} \mid s(a) \not\models l_i\}|.$$

Intuition: more true goal literals ⇝ closer to the goal

⇝ STRIPS heuristic (properties?)

Note: From now on, for convenience we usually write heuristics as functions of states (as above), not nodes.
Node heuristic h' is defined from state heuristic h as $h'(\sigma) := h(\mathit{state}(\sigma))$.

Criticism of the STRIPS heuristic

What is wrong with the STRIPS heuristic?

- quite uninformative:
  the range of heuristic values in a given task is small;
  typically, most successors have the same estimate
- very sensitive to reformulation:
  can easily transform any planning task into an equivalent one where $h(s) = 1$ for all non-goal states
- ignores almost all problem structure:
  heuristic value does not depend on the set of operators!

⇝ need a better, principled way of coming up with heuristics

Outline

Coming up with heuristics in a principled way

General procedure for obtaining a heuristic

Solve an easier version of the problem.

Two common methods:

- relaxation: consider less constrained version of the problem
- abstraction: consider smaller version of real problem

Both have been very successfully applied in planning.
We consider both in this course, beginning with relaxation.

Relaxing a problem

How do we relax a problem?

Example (Route planning for a road network)

The road network is formalized as a weighted graph over points in the Euclidean plane. The weight of an edge is the road distance between two locations.

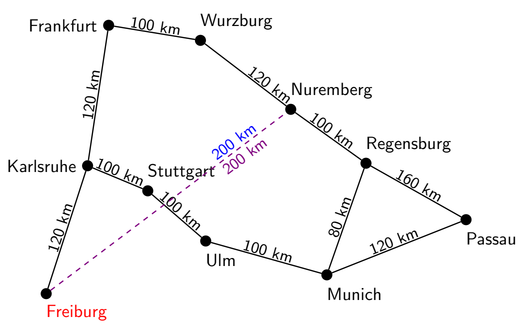
A relaxation drops constraints of the original problem.

Example (Relaxation for route planning)

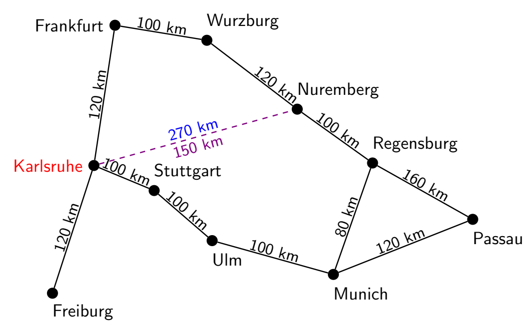
Use the Euclidean distance $\sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2}$ as a heuristic for the road distance between (x_1, x_2) and (y_1, y_2)
This is a lower bound on the road distance ($\rightsquigarrow$ admissible).

$\rightsquigarrow$ We drop the constraint of having to travel on roads.

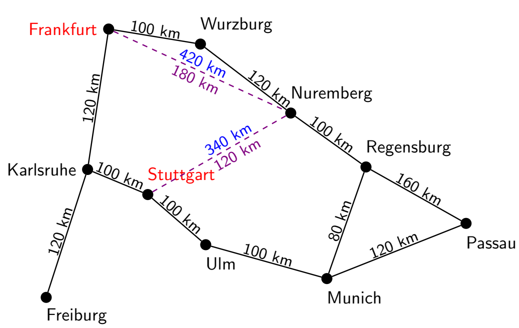
A* using the Euclidean distance heuristic



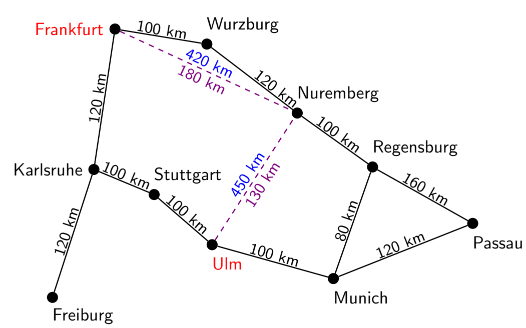
A* using the Euclidean distance heuristic



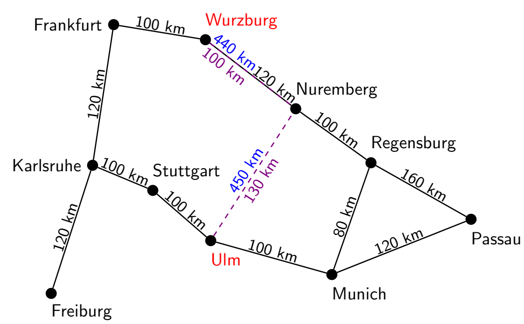
A* using the Euclidean distance heuristic



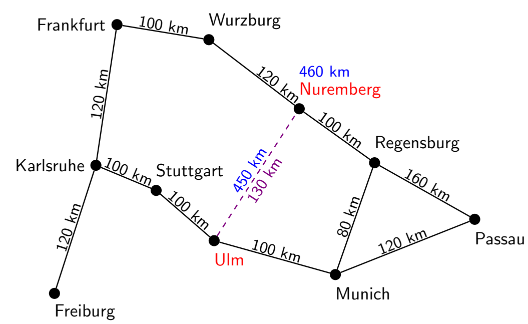
A* using the Euclidean distance heuristic



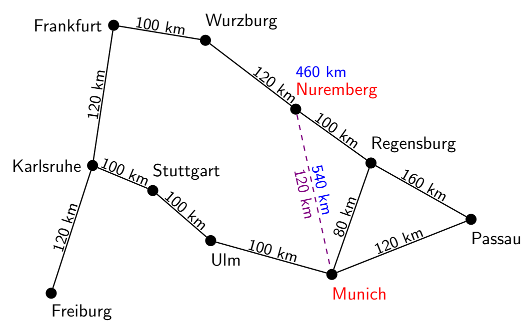
A* using the Euclidean distance heuristic



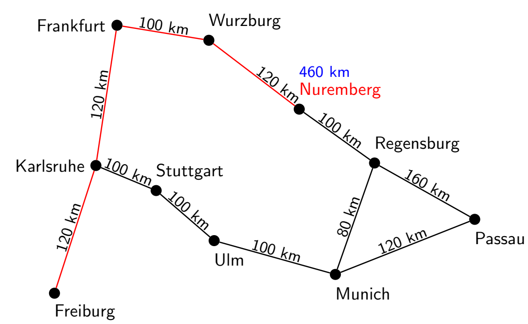
A* using the Euclidean distance heuristic



A* using the Euclidean distance heuristic



A* using the Euclidean distance heuristic



Outline

Relaxations for planning

- Relaxation is a general technique for heuristic design:
  - Straight-line heuristic (route planning): Ignore the fact that one must stay on roads.
  - Manhattan heuristic (15-puzzle): Ignore the fact that one cannot move through occupied tiles.
- We want to apply the idea of relaxations to planning.
- Informally, we want to ignore bad side effects of applying operators.

What is a good or bad effect?

Question: Which operator effects are good, and which are bad?

Difficult to answer in general, because it depends on context:

- Locking the entrance door is good if we want to keep burglars out.
- Locking the entrance door is bad if we want to enter.

We will now consider a reformulation of planning tasks that makes the distinction between good and bad effects obvious.

Outline

Notation Review

The notation we use here is a generalization of the notation used in previous introductory lectures, which was based on the GNT textbook. Recall:

Definition

An operator $\langle c, e\rangle$ is a STRIPS operator if

1. precondition c is a conjunction* of literals, and
2. effect e is a conjunction of atomic effects.

**We previously used "set" rather than "conjunction".*

Notation Review (cont.)

Here we extend the expressiveness of our operator definition as follows:

- precondition c is an arbitrary propositional formula.
- (Deterministic) effect e is defined recursively as follows:
  1. If $a \in A$ is a state variable, then a and $\neg a$ are effects (atomic effects).
  2. If $e_1, \dots, e_n$ are effects, then $e_1 \wedge \dots \wedge e_n$ is an effect (conjunctive effects). The special case with $n = 0$ is the empty conjunction $\top$.
  3. If c is a propositional formula and e is an effect, then $c \rhd e$ is an effect (conditional effects).

  Atomic effects a and $\neg a$ are best understood as assignments $a := 1$ and $a := 0$, respectively.

Positive normal form

Definition (operators in positive normal form)

An operator $o = \langle c, e\rangle$ is in positive normal form if it is in normal form, no negation symbols appear in c, and no negation symbols appear in any effect condition in e.

Definition (planning tasks in positive normal form)

A planning task $\langle A, I, O, G\rangle$ is in positive normal form if all operators in O are in positive normal form and no negation symbols occur in the goal G.

Positive normal form: existence

Theorem (positive normal form)

Every planning task Π has an equivalent planning task Π' in positive normal form.
Moreover, Π' can be computed from Π in polynomial time.

Note: Equivalence here means that the represented transition systems of Π and Π', limited to the states that can be reached from the initial state, are isomorphic.

We prove the theorem by describing a suitable algorithm. (However, we do not prove its correctness or complexity.)

Positive normal form: algorithm

Transformation of $\langle A, I, O, G\rangle$ to positive normal form

Convert all operators $o \in O$ to normal form.
Convert all conditions* to negation normal form (NNF).
while any condition contains a negative literal $\neg a$:
  Let a be a variable which occurs negatively in a condition.
  $A := A \cup \{\hat{a}\}$ for some new state variable $\hat{a}$
  $I(\hat{a}) := 1 - I(a)$
  Replace the effect a by $(a \wedge \neg\hat{a})$ in all operators $o \in O$.
  Replace the effect $\neg a$ by $(\neg a \wedge \hat{a})$ in all operators $o \in O$.
  Replace $\neg a$ by $\hat{a}$ in all conditions.
Convert all operators $o \in O$ to normal form (again).

* Here, *all conditions* refers to all operator preconditions, operator effect conditions and the goal.

Outline

Positive normal form: example

Example (transformation to positive normal form)

$$\begin{aligned}A &= \{\textit{home}, \textit{uni}, \textit{lecture}, \textit{bike}, \textit{bike-locked}\}\\ I &= \{\textit{home} \mapsto 1, \textit{bike} \mapsto 1, \textit{bike-locked} \mapsto 1,\\ &\quad \textit{uni} \mapsto 0, \textit{lecture} \mapsto 0\}\\ O &= \{\langle \textit{home} \wedge \textit{bike} \wedge \neg\textit{bike-locked}, \neg\textit{home} \wedge \textit{uni}\rangle,\\ &\quad \langle \textit{bike} \wedge \textit{bike-locked}, \neg\textit{bike-locked}\rangle,\\ &\quad \langle \textit{bike} \wedge \neg\textit{bike-locked}, \textit{bike-locked}\rangle,\\ &\quad \langle \textit{uni}, \textit{lecture} \wedge ((\textit{bike} \wedge \neg\textit{bike-locked}) \rhd \neg\textit{bike})\rangle\}\\ G &= \textit{lecture} \wedge \textit{bike}\end{aligned}$$

Positive normal form: example

Example (transformation to positive normal form)

$$\begin{aligned}A &= \{\textit{home}, \textit{uni}, \textit{lecture}, \textit{bike}, \textit{bike-locked}\}\\ I &= \{\textit{home} \mapsto 1, \textit{bike} \mapsto 1, \textit{bike-locked} \mapsto 1,\\ &\quad \textit{uni} \mapsto 0, \textit{lecture} \mapsto 0\}\\ O &= \{\langle \textit{home} \wedge \textit{bike} \wedge \neg\textit{bike-locked}, \neg\textit{home} \wedge \textit{uni}\rangle,\\ &\quad \langle \textit{bike} \wedge \textit{bike-locked}, \neg\textit{bike-locked}\rangle,\\ &\quad \langle \textit{bike} \wedge \neg\textit{bike-locked}, \textit{bike-locked}\rangle,\\ &\quad \langle \textit{uni}, \textit{lecture} \wedge ((\textit{bike} \wedge \neg\textit{bike-locked}) \rhd \neg\textit{bike})\rangle\}\\ G &= \textit{lecture} \wedge \textit{bike}\end{aligned}$$

Identify state variable a occurring negatively in conditions.

Positive normal form: example

Example (transformation to positive normal form)

$$\begin{aligned}A &= \{\textit{home}, \textit{uni}, \textit{lecture}, \textit{bike}, \textit{bike-locked}, \textit{bike-unlocked}\}\\ I &= \{\textit{home} \mapsto 1, \textit{bike} \mapsto 1, \textit{bike-locked} \mapsto 1,\\ &\quad \textit{uni} \mapsto 0, \textit{lecture} \mapsto 0, \textit{bike-unlocked} \mapsto 0\}\\ O &= \{\langle \textit{home} \wedge \textit{bike} \wedge \neg\textit{bike-locked}, \neg\textit{home} \wedge \textit{uni}\rangle,\\ &\quad \langle \textit{bike} \wedge \textit{bike-locked}, \neg\textit{bike-locked}\rangle,\\ &\quad \langle \textit{bike} \wedge \neg\textit{bike-locked}, \textit{bike-locked}\rangle,\\ &\quad \langle \textit{uni}, \textit{lecture} \wedge ((\textit{bike} \wedge \neg\textit{bike-locked}) \rhd \neg\textit{bike})\rangle\}\\ G &= \textit{lecture} \wedge \textit{bike}\end{aligned}$$

Introduce new variable $\hat{a}$ with complementary initial value.

Positive normal form: example

Example (transformation to positive normal form)

$$\begin{aligned}A &= \{\textit{home}, \textit{uni}, \textit{lecture}, \textit{bike}, \textit{bike-locked}, \textit{bike-unlocked}\}\\ I &= \{\textit{home} \mapsto 1, \textit{bike} \mapsto 1, \textit{bike-locked} \mapsto 1,\\ &\quad \textit{uni} \mapsto 0, \textit{lecture} \mapsto 0, \textit{bike-unlocked} \mapsto 0\}\\ O &= \{\langle \textit{home} \wedge \textit{bike} \wedge \neg\textit{bike-locked}, \neg\textit{home} \wedge \textit{uni}\rangle,\\ &\quad \langle \textit{bike} \wedge \textit{bike-locked}, \neg\textit{bike-locked}\rangle,\\ &\quad \langle \textit{bike} \wedge \neg\textit{bike-locked}, \textit{bike-locked}\rangle,\\ &\quad \langle \textit{uni}, \textit{lecture} \wedge ((\textit{bike} \wedge \neg\textit{bike-locked}) \rhd \neg\textit{bike})\rangle\}\\ G &= \textit{lecture} \wedge \textit{bike}\end{aligned}$$

Identify effects on variable a.

Positive normal form: example

Example (transformation to positive normal form)

$$A = \{home, uni, lecture, bike, bike\text{-}locked, bike\text{-}unlocked\}$$
$$I = \{home \mapsto 1, bike \mapsto 1, bike\text{-}locked \mapsto 1, uni \mapsto 0, lecture \mapsto 0, bike\text{-}unlocked \mapsto 0\}$$
$$O = \{\langle home \land bike \land \neg bike\text{-}locked, \neg home \land uni\rangle, \langle bike \land bike\text{-}locked, \neg bike\text{-}locked \land bike\text{-}unlocked\rangle, \langle bike \land \neg bike\text{-}locked, bike\text{-}locked \land \neg bike\text{-}unlocked\rangle, \langle uni, lecture \land ((bike \land \neg bike\text{-}locked) \rhd \neg bike)\rangle\}$$
$$G = lecture \land bike$$

Introduce complementary effects for $\hat{a}$.

Positive normal form: example

Example (transformation to positive normal form)

$$A = \{home, uni, lecture, bike, bike\text{-}locked, bike\text{-}unlocked\}$$
$$I = \{home \mapsto 1, bike \mapsto 1, bike\text{-}locked \mapsto 1, uni \mapsto 0, lecture \mapsto 0, bike\text{-}unlocked \mapsto 0\}$$
$$O = \{\langle home \land bike \land \neg bike\text{-}locked, \neg home \land uni\rangle, \langle bike \land bike\text{-}locked, \neg bike\text{-}locked \land bike\text{-}unlocked\rangle, \langle bike \land \neg bike\text{-}locked, bike\text{-}locked \land \neg bike\text{-}unlocked\rangle, \langle uni, lecture \land ((bike \land \neg bike\text{-}locked) \rhd \neg bike)\rangle\}$$
$$G = lecture \land bike$$

Identify negative conditions for a.

Positive normal form: example

Example (transformation to positive normal form)

$$A = \{home, uni, lecture, bike, bike\text{-}locked, bike\text{-}unlocked\}$$
$$I = \{home \mapsto 1, bike \mapsto 1, bike\text{-}locked \mapsto 1, uni \mapsto 0, lecture \mapsto 0, bike\text{-}unlocked \mapsto 0\}$$
$$O = \{\langle home \land bike \land bike\text{-}unlocked, \neg home \land uni\rangle, \langle bike \land bike\text{-}locked, \neg bike\text{-}locked \land bike\text{-}unlocked\rangle, \langle bike \land bike\text{-}unlocked, bike\text{-}locked \land \neg bike\text{-}unlocked\rangle, \langle uni, lecture \land ((bike \land bike\text{-}unlocked) \rhd \neg bike)\rangle\}$$
$$G = lecture \land bike$$

Replace by positive condition $\hat{a}$.

Positive normal form: example

Example (transformation to positive normal form)

$$A = \{home, uni, lecture, bike, bike\text{-}locked, bike\text{-}unlocked\}$$
$$I = \{home \mapsto 1, bike \mapsto 1, bike\text{-}locked \mapsto 1, uni \mapsto 0, lecture \mapsto 0, bike\text{-}unlocked \mapsto 0\}$$
$$O = \{\langle home \land bike \land bike\text{-}unlocked, \neg home \land uni\rangle, \langle bike \land bike\text{-}locked, \neg bike\text{-}locked \land bike\text{-}unlocked\rangle, \langle bike \land bike\text{-}unlocked, bike\text{-}locked \land \neg bike\text{-}unlocked\rangle, \langle uni, lecture \land ((bike \land bike\text{-}unlocked) \rhd \neg bike)\rangle\}$$
$$G = lecture \land bike$$

What does this transformation achieve?

We have expanded the size of our domain by introducing new propositions to ensure that all the conditions that affect planning:

- preconditions
- conditions of conditional effects
- goals

are expressed in terms of positive literals, and we've adjusted the effects of operators to ensure that they are consistent with the introduction of these new propositions.

Outline

Relaxed planning tasks: idea

In positive normal form, good and bad effects are easy to distinguish:

- Effects that make state variables true are good (add effects).
- Effects that make state variables false are bad (delete effects).

***** Idea for the heuristic: Ignore all delete effects. ****

Relaxed planning tasks

Definition (relaxation of operators)

The relaxation o^+ of an operator $o = \langle c, e\rangle$ in positive normal form is the operator which is obtained by replacing all negative effects $\neg a$ within e by the do-nothing effect $\top$.

Definition (relaxation of planning tasks)

The relaxation Π^+ of a planning task $\Pi = \langle A, I, O, G\rangle$ in positive normal form is the planning task $\Pi^+ := \langle A, I, \{o^+ \mid o \in O\}, G\rangle$.

Definition (relaxation of operator sequences)

The relaxation of an operator sequence $\pi = o_1 \ldots o_n$ is the operator sequence $\pi^+ := {o_1}^+ \ldots {o_n}^+$.

Relaxed planning tasks: terminology

- Planning tasks in positive normal form without delete effects are called relaxed planning tasks.
- Plans for relaxed planning tasks are called relaxed plans.
- If Π is a planning task in positive normal form and π^+ is a plan for Π^+, then π^+ is called a relaxed plan for Π.

Outline

Greedy algorithm for relaxed planning tasks

The relaxed planning task can be solved in polynomial time using a simple greedy algorithm:

Greedy planning algorithm for $\langle A, I, O^+, G\rangle$

$s := I$
$\pi^+ := \epsilon$
forever:
  if $s \models G$:
    return π^+
  else if there is an operator $o^+ \in O^+$ applicable in s
      with $app_{o^+}(s) \neq s$:
    Append such an operator o^+ to π^+.
    $s := app_{o^+}(s)$
  else:
    return unsolvable

Correctness of the greedy algorithm

The algorithm is sound:

- If it returns a plan, this is indeed a correct solution.
- If it returns "unsolvable", the task is indeed unsolvable

What about completeness (termination) and runtime?

- Each iteration of the loop adds at least one atom to the set of true state variables in s.
- This guarantees termination after at most $|A|$ iterations.
- Thus, the algorithm can clearly be implemented to run in polynomial time.

Outline

Using the greedy algorithm as a heuristic

We can apply the greedy algorithm within heuristic search:

- In a search node σ, solve the relaxation of the planning task with *state*(σ) as the initial state.
- Set $h(\sigma)$ to the length of the generated relaxed plan.

Is this an admissible heuristic?

- Yes if the relaxed plans are optimal (due to the plan preservation corollary).
- However, usually they are not, because our greedy planning algorithm is very poor.

Generating an admissible heuristic is NP-hard

- To obtain an *admissible* heuristic, we need to generate an *optimal* relaxed plan.
- The problem of deciding whether a given relaxed planning task has a length at most K is NP-complete (through a reduction of part of the problem to the set cover problem).
- Thus, generating an optimal relaxed plan for the purposes of generating a heuristic (not even solving the problem!) is not a good strategy.

Outline

Using relaxations in practice

How can we use relaxations for heuristic planning in practice?

Different possibilities:

- Implement an optimal planner for relaxed planning tasks and use its solution lengths as an estimate, even though it is NP-hard.
  $\rightsquigarrow$ h^+ heuristic
- Do not actually solve the relaxed planning task, but compute an estimate of its difficulty in a different way.
  $\rightsquigarrow$ h_{max} heuristic, h_{add} heuristic
- Compute a solution for relaxed planning tasks which is not necessarily optimal, but "reasonable".
  $\rightsquigarrow$ h_{FF} heuristic

Outline

Towards better relaxed plans

Why does the greedy algorithm compute low-quality plans?

- It may apply many operators which are not goal-directed.

How can this problem be fixed?

- Reaching the goal of a relaxed planning task is most easily achieved with forward search.
- Analyzing relevance of an operator for achieving a goal (or subgoal) is most easily achieved with backward search.

Idea: Use a forward-backward algorithm that first finds a path to the goal greedily, then prunes it to a relevant subplan. *Does this sound similar to an algorithm we've seen before?*

The Relaxed Plan Graph Heuristic and FF

In the tutorial today you will learn about the Relaxed Plan Graph (RPG) heuristic and how it is used in one particular planner, Fast-Forward (FF) (Hoffmannn & Nebel, JAIR-01).

- **Heuristic:** Solve the relaxed planning problem using a planning graph approach.
- **Search:** Hill-climbing extended by breadth-first search on plateaus and with pruning
- **Pruning:** Only those successors are considered that are part of a relaxed solution – i.e., the result of so-called *helpful actions*
- **Fall-back strategy:** Complete best-first search