Planning: Representation and Example

CPSC 322 – Planning 1

Textbook §11.1 - 11.2

Lecture Overview

Greedy Descent with Min-Conflict Heuristic

This is one of the best techniques for solving CSP problems:

- At random, select one of the variables v that participates in a violated constraint
- Set v to one of the values that minimizes the number of unsatisfied constraints

Simulated Annealing

- Pick a variable at random and a new value at random.
- If it is an improvement, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on a temperature parameter, T.
- Temperature reduces over time, according to an annealing schedule

Tabu lists

- SLS algorithms can get stuck in plateaus
- To prevent cycling we can maintain a tabu list of the k last nodes visited.
- Don't visit a node that is already on the tabu list.

Stochastic Beam Search

- Maintain k nodes at every step.
- When updating, probabilistically choose k new nodes from the neighbors of the original nodes.
- The probability that a neighbor is chosen is proportional to its scoring function value.

Genetic Algorithms

- Like stochastic beam search, but pairs of nodes are combined to create the offspring:
- For each generation:
  - Randomly choose pairs of nodes, with the best-scoring nodes being more likely to be chosen.
  - For each pair, perform a cross-over: form two offspring each taking different parts of their parents
  - Mutate some values
- Report best node found.

Lecture Overview

Planning

- With CSPs, we looked for solutions to essentially atemporal problems.
  - find a single variable assignment (state) that satisfies all of our constraints.

  Now consider a problem where we are given:
  - A description of an initial state
  - A description of the effects and preconditions of actions
  - A goal to achieve

  ...and want to find a sequence of actions that is possible and will result in a state satisfying the goal.
  - note: here we want not a single state that satisfies our constraints, but rather a sequence of states that gets us to a goal

State-Based Representation of a Planning Domain

We solved similar planning problems when we considered state-based representations. How was the problem represented?

- The domain is characterized by states, actions and goals
  - note: a given action may not be possible in all states
- Key issue: representing the way we transition from one state to another by taking actions
- A state-based representation has no structure
  - there's no sense in which we can say that states a and b are more similar than states a and z
- Thus, we can't do better than a tabular representation:

Starting state	Action	Resulting state
⋮	⋮	⋮

Problems with the Tabular Representation

Problems:

- Usually too many states for a tabular representation to be feasible
- Small changes to the model can mean big changes for the representation
  - e.g., if we added another feature, all the states would change
- There may be structure and regularity to the actions, and to the states themselves. If there is, there's no way to capture it with this representation.

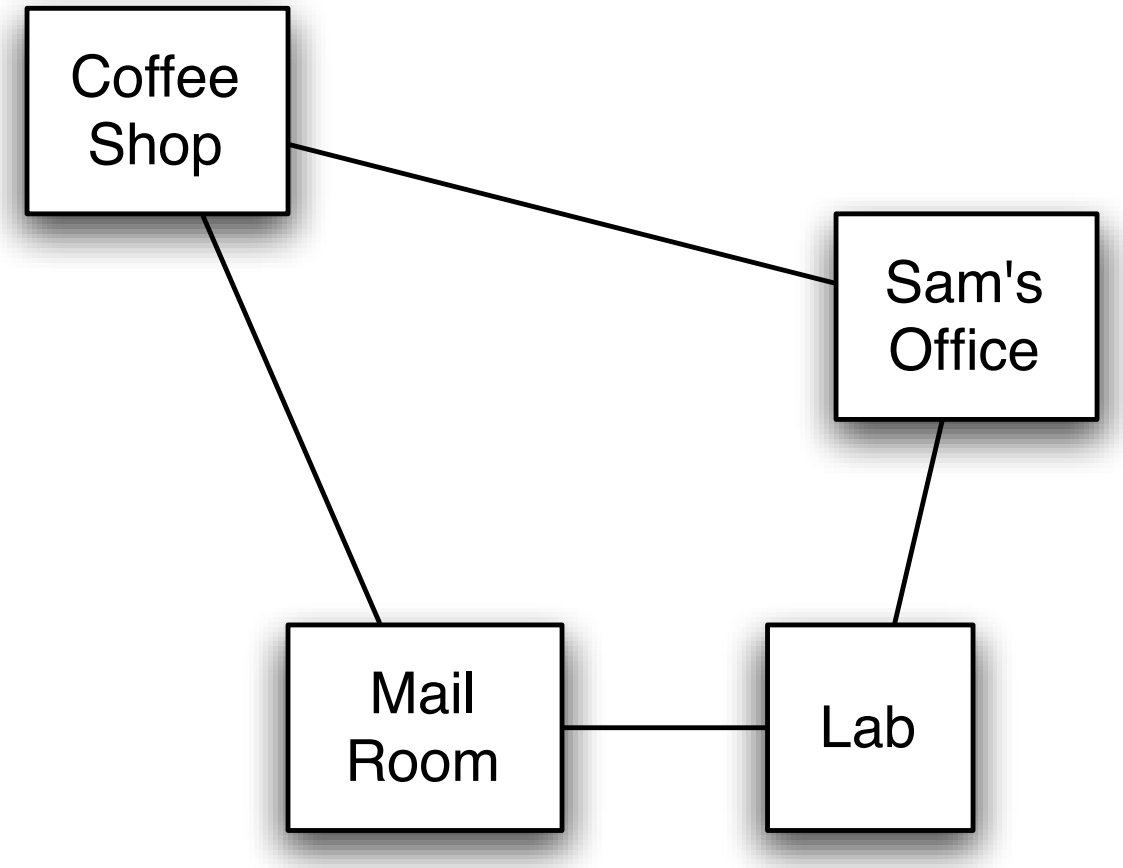
Feature-Based Representation

- Features helped us to represent CSPs more compactly than states could.
  - The main idea: factor states into joint variable assignments
  - This allowed us to represent constraints compactly by discussing only relevant variables rather than full states.
- Now what we want to find is a sequence of variable assignments that
  - begins at an initial state
  - proceeds from one state to another by taking valid actions
  - ends up at a goal
- Instead of having one variable for every feature, we must instead have one variable for every feature at each time step, indicating the value taken by that feature at that time step!
- Can this representation be more useful than the state-based one?

Lecture Overview

Delivery Robot Example

Consider a delivery robot named Rob, who must navigate the following environment:



Delivery Robot Example

The state is defined by the following features:

$RLoc$ — Rob's location

- domain: coffee shop (cs), Sam's office (off), mail room (mr), or laboratory (lab)

RHC — Rob has coffee

- domain: true/false. By rhc indicate that Rob has coffee and by $\overline{rhc}$ that Rob doesn't have coffee.

SWC — Sam wants coffee (T/F)

MW — Mail is waiting (T/F)

RHM — Rob has mail (T/F)

An example state is $\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$. How many states are there:

Delivery Robot Example

The state is defined by the following features:

$RLoc$ — Rob's location

- domain: coffee shop (cs), Sam's office (off), mail room (mr), or laboratory (lab)

RHC — Rob has coffee

- domain: true/false. By rhc indicate that Rob has coffee and by $\overline{rhc}$ that Rob doesn't have coffee.

SWC — Sam wants coffee (T/F)

MW — Mail is waiting (T/F)

RHM — Rob has mail (T/F)

An example state is $\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$. How many states are there: $4 \times 2 \times 2 \times 2 \times 2 = 64$.

Delivery Robot Example

The robot's actions are:

$Move$ — Rob's move action
- move clockwise (mc), move anti-clockwise (mac) not move (nm)

PUC — Rob picks up coffee
- must be at the coffee shop

$DelC$ — Rob delivers coffee
- must be at the office, and must have coffee

PUM — Rob picks up mail
- must be in the mail room, and mail must be waiting

$DelM$ — Rob delivers mail
- must be at the office and have mail

Assume that Rob can perform one action of each kind in a single step. Thus, an example action is $\langle mc, \overline{puc}, \overline{dc}, \overline{pum}, dm\rangle$; we can abbreviate it as $\langle mc, dm\rangle$.

How many actions are there:

Delivery Robot Example

The robot's actions are:

$Move$ — Rob's move action
- move clockwise (mc), move anti-clockwise (mac) not move (nm)

PUC — Rob picks up coffee
- must be at the coffee shop

$DelC$ — Rob delivers coffee
- must be at the office, and must have coffee

PUM — Rob picks up mail
- must be in the mail room, and mail must be waiting

$DelM$ — Rob delivers mail
- must be at the office and have mail

Assume that Rob can perform one action of each kind in a single step. Thus, an example action is $\langle mc, \overline{puc}, \overline{dc}, \overline{pum}, dm\rangle$; we can abbreviate it as $\langle mc, dm\rangle$.

How many actions are there: $3 \times 2 \times 2 \times 2 \times 2 = 48$.

Example State-Based Representation

State	Action	Resulting State
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mc \rangle$	$\langle mr, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mac \rangle$	$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle nm \rangle$	$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mac, dm \rangle$	$\langle cs, \overline{rhc}, \overline{swc}, \overline{mw}, \overline{rhm} \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mac, \overline{dm} \rangle$	$\langle cs, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mc, dm \rangle$	$\langle lab, \overline{rhc}, \overline{swc}, \overline{mw}, \overline{rhm} \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mc, \overline{dm} \rangle$	$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
...	...	...

Lecture Overview

Feature-Based Representation

Where we stand so far:

- the state-based representation is unworkable
- a feature-based representation might help.

How would a feature-based representation work?

- the state space is easy: joint assignment to variables
- initial state and goal state are also easy
- the key is modeling actions

Modeling Actions

To "model actions" in the feature-based representation, we need to solve two problems:

1. Model when the actions are possible, in terms of the values of the features of the current state
2. Model the state transitions in a "factored" way:

Why might this be more tractable/managable than the tabular representation?

Modeling Actions

To "model actions" in the feature-based representation, we need to solve two problems:

1. Model when the actions are possible, in terms of the values of the features of the current state
2. Model the state transitions in a "factored" way:

Why might this be more tractable/managable than the tabular representation?

- if some action doesn't depend on or modify some feature of the state, we can achieve some representational savings here
- The representation can be easier to modify/update.

The STRIPS Representation

- The book discusses a feature-centric representation
  - for every feature, where does its value come from?
- causal rule: ways a feature's value can be changed by taking an action.
- frame rule: requires that a feature's value is unchanged if no action changes it.

- STRIPS is an action-centric representation:
  - for every action, what does it do?
- This leaves us with no way to state frame rules.
- The STRIPS assumption:
  - all variables not explicitly changed by an action stay unchanged

STRIPS Actions

- In STRIPS, an action has two parts:
  1. Precondition: a logical test about the features that must be true in order for the action to be legal
  2. Effects: a set of assignments to variables that are caused by the action

- If the feature V has value v after the action a has been performed, what can we conclude about a and/or the state of the world?

STRIPS Actions

- In STRIPS, an action has two parts:
  1. Precondition: a logical test about the features that must be true in order for the action to be legal
  2. Effects: a set of assignments to variables that are caused by the action

- If the feature V has value v after the action a has been performed, what can we conclude about a and/or the state of the world?
  - either $V = v$ was true in the state of the world immediately preceding execution of action a, or a sets $V = v$, or both.

Example

STRIPS representation of the action pick up coffee, PUC:

- **preconditions** $Loc = cs$ and $RHC = \overline{rhc}$
- **effects** $RHC = rhc$

STRIPS representation of the action deliver coffee, $DelC$:

- **preconditions** $Loc = off$ and $RHC = rhc$
- **effects** $RHC = \overline{rhc}$ and $SWC = \overline{swc}$

Note that Sam doesn't have to want coffee for Rob to deliver it; one way or another, Sam doesn't want coffee after delivery.

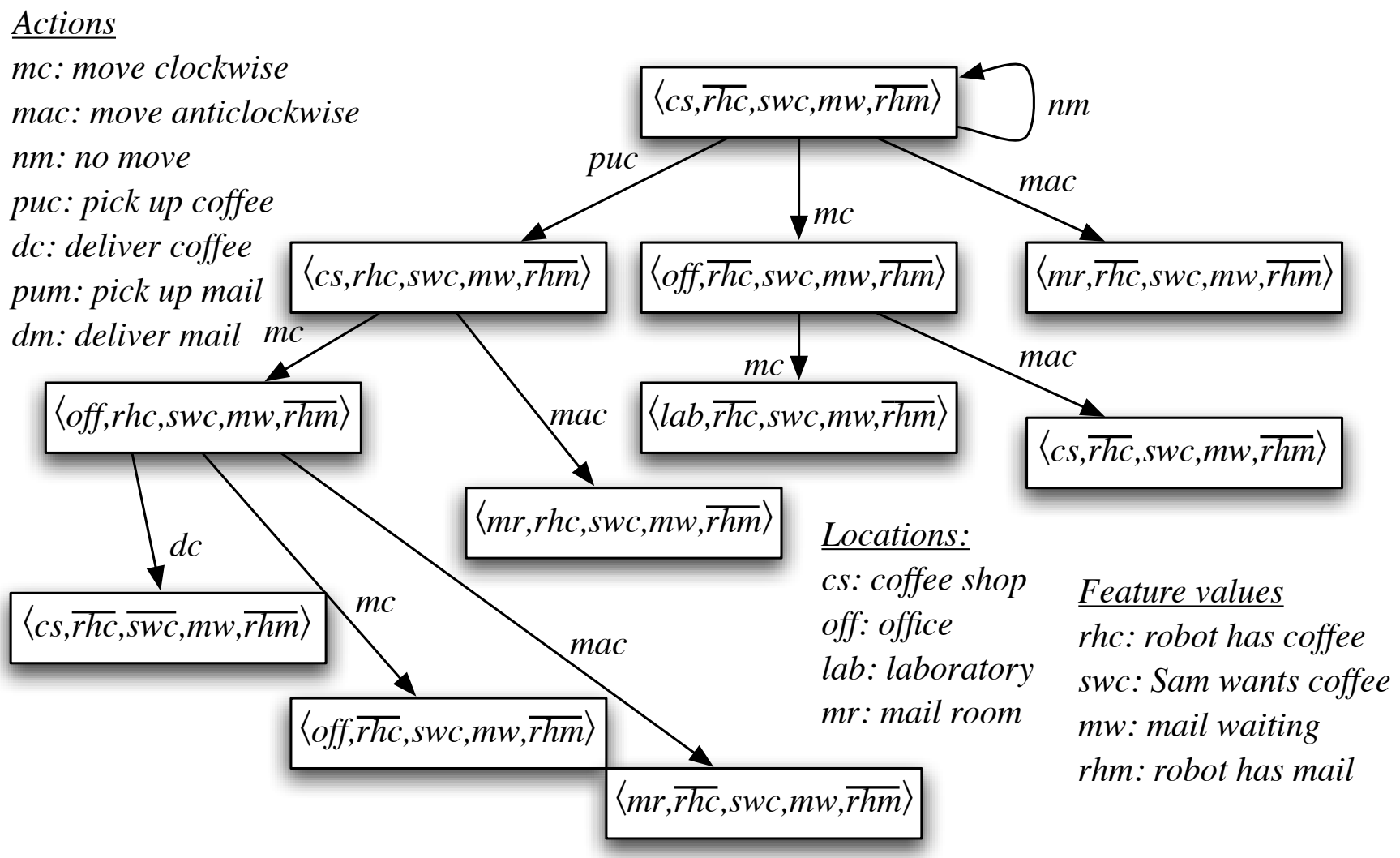
Lecture Overview

Forward Planning

Idea: search in the state-space graph.

- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state s represent all of the actions that are legal in state s.
- A plan is a path from the state representing the initial state to a state that satisfies the goal.

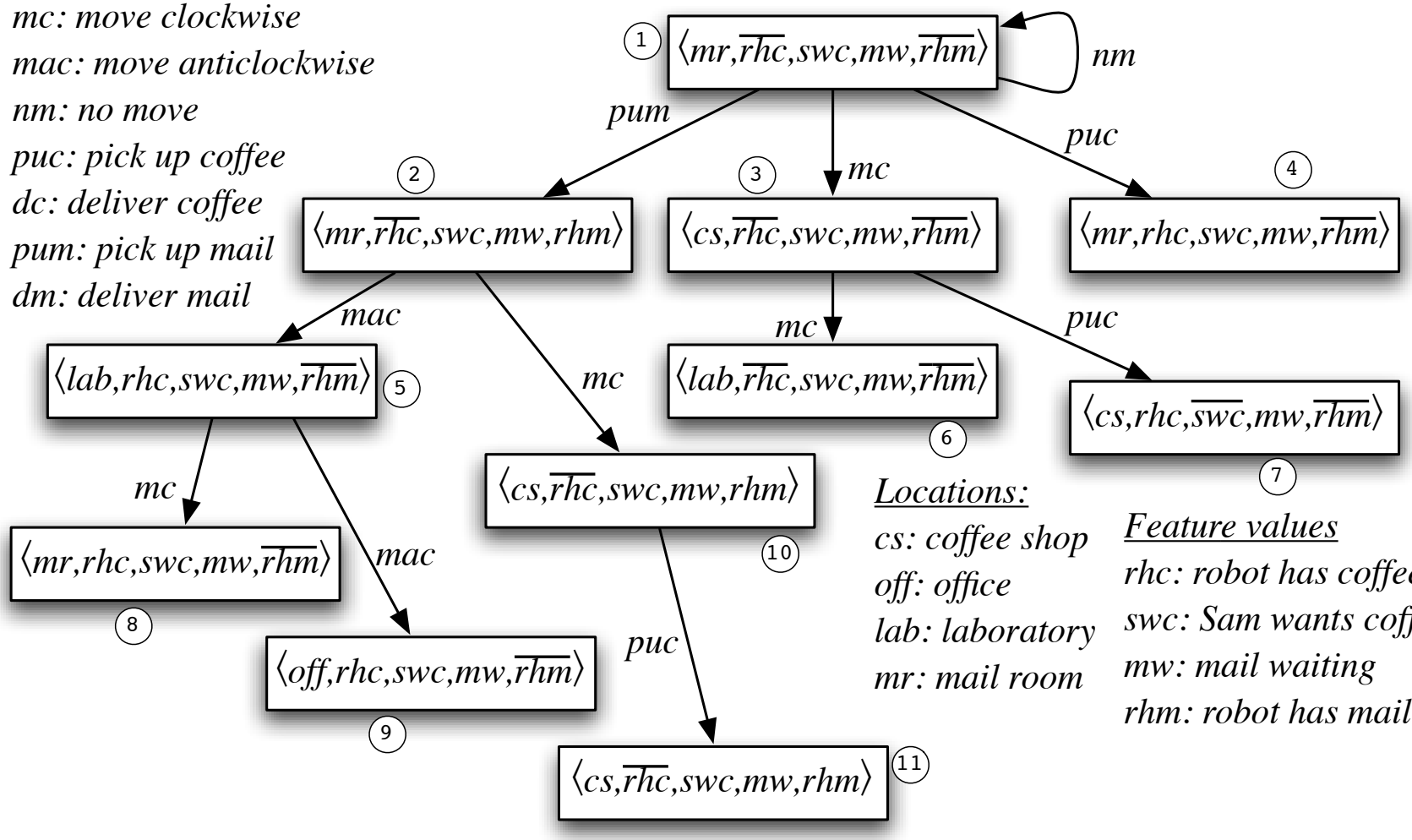
Example state-space graph



What are the errors (none involve room locations)?

Actions

mc: move clockwise
mac: move anticlockwise
nm: no move
puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail



Locations:

cs: coffee shop
off: office
lab: laboratory
mr: mail room

Feature values

rhc: robot has coffee
swc: Sam wants coffee
mw: mail waiting
rhm: robot has mail

Improving Search Efficiency

Forward search can use domain-specific knowledge specified as:

- a heuristic function that estimates the number of steps to the goal
- domain-specific pruning of neighbors:
  - don't go to the coffee shop unless "Sam wants coffee" is part of the goal and Rob doesn't have coffee
  - don't pick-up coffee unless Sam wants coffee
  - unless the goal involves time constraints, don't do the "no move" action.