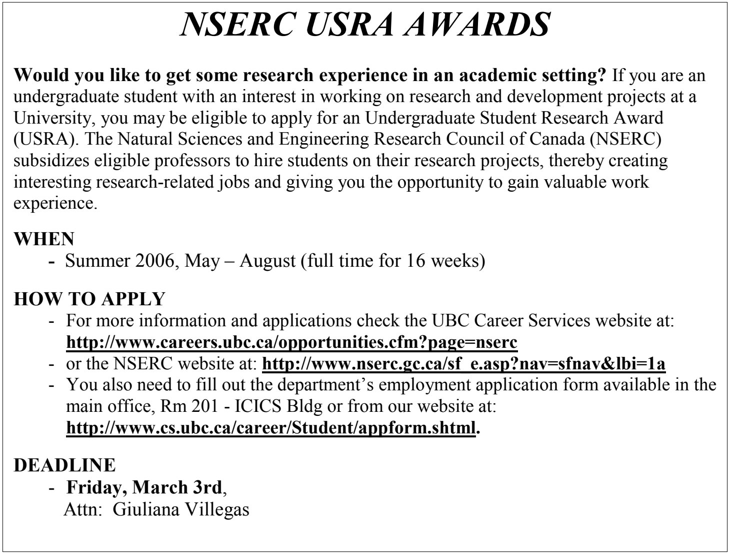
Planning: Representation

CPSC 322 Lecture 14

February 3, 2006
Textbook §11.1

Lecture Overview



NSERC USRA AWARDS

Would you like to get some research experience in an academic setting? If you are an undergraduate student with an interest in working on research and development projects at a University, you may be eligible to apply for an Undergraduate Student Research Award (USRA). The Natural Sciences and Engineering Research Council of Canada (NSERC) subsidizes eligible professors to hire students on their research projects, thereby creating interesting research-related jobs and giving you the opportunity to gain valuable work experience.

WHEN

- Summer 2006, May – August (full time for 16 weeks)

HOW TO APPLY

- For more information and applications check the UBC Career Services website at: **http://www.careers.ubc.ca/opportunities.cfm?page=nserc**
- or the NSERC website at: **http://www.nserc.gc.ca/sf_e.asp?nav=sfnav&lbi=1a**
- You also need to fill out the department's employment application form available in the main office, Rm 201 - ICICS Bldg or from our website at: **http://www.cs.ubc.ca/career/Student/appform.shtml.**

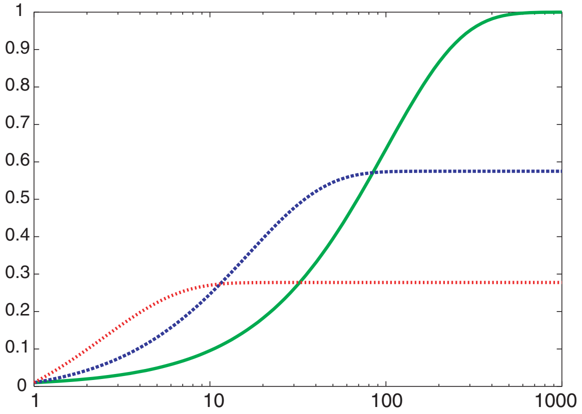
DEADLINE

- **Friday, March 3rd**,
  Attn: Giuliana Villegas

Lecture Overview

Runtime Distribution

- To compare SLS algorithms, use runtime distributions



SLS Variants

- Greedy Descent with Min-Conflict Heuristic
  - randomly select a variable that participates in a violated constraint; set it to the value that minimizes num violated constraints
- Simulated Annealing
  - randomly choose a variable/value combination
  - always keep it if it's an improvement; if it's not an improvement, keep it anyway with a probability that depends on the "temperature"
  - gradually "cool down"
- Tabu lists
  - keep a list of the last k nodes visited
  - don't revisit nodes on this list

More SLS Variants

- Parallel search
  - Maintain k nodes instead of one
  - Run an independent search with each node
- Beam search
  - Maintain k nodes
  - Update: the k best nodes out of all the original nodes' neighbours
- Stochastic Beam Search
  - Same as the above, but instead of updating with the k best nodes, choose k nodes at random, but in proportion to their scores

Genetic Algorithms

- Like stochastic beam search, but pairs are 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

- To a large extent, CSPs and search have been focused on atemporal problems.
  - Before, when solving a CSP, we wanted to find a single state of the world that satisfied all of our constraints
- How should an agent reason and act when it will take a sequence of actions to achieve a goal?
  - Now we want to find a *sequence* of states of the world that get us from our initial state to the goal state

Planning: Assumptions

- We make the following assumptions about the world:
  - the world is deterministic
  - the agent knows what state it is in and understands the consequences of its actions
  - the agent is the only thing that changes the state of the world
  - time passes in discrete steps
  - goals are predicates of states ("goal tests")

Goals

- Achievement goals: conditions on the final state we want to achieve
- Maintenance goals: conditions that we want to have remain true throughout the plan
- Transient goals: conditions that we want to achieve before the final achievement goal, but which do not need to remain true
- Resource goals: functions we want to minimize in the pursuit of our achievement goals (e.g., energy expended; distance traveled.)

Lecture Overview

State-Based Representation of a Planning Domain

- 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.

Lecture Overview

Feature-Based Representation

- Features helped us to represent CSPs more compactly than states could. Can they help us here?
- Note that our problem is trickier than the original CSP problem:
    - we used to be looking for a single variable assignment, out of the space of all variable assignments, that satisfies our constraints
- Now we are looking for a sequence of variable assignments that
    - begins at the initial state
    - proceeds from one state to another by taking valid actions
    - ends up at a goal
- This means that 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!

Feature-Based Representation

- Model when the actions are possible, in terms of the values of the features of the current state
- Model the state transitions in a “factored” way:
    - describe how each feature in the next state is affected by the features of the current state and the action
    - if some action doesn’t depend on or modify some feature of the state, we can achieve some representational savings here

Feature-Based Representation

We need two things to replace the tabular representation:

1. Modeling when actions are possible:
   - Provide a function that indicates when an action can be executed
   - Precondition of an action: a function (proposition) of the state variables that is true when the action can be carried out

Feature-Based Representation

We need two things to replace the tabular representation:

1. Modeling when actions are possible
2. Modeling state transitions in a “factored” way:
   - causal rules: explain how the value of a variable describing a feature at time step t depends on the action taken at time $t-1$
     - things that are changed in the world
     - example: *act* causes $V_1 = v_1$

Feature-Based Representation

We need two things to replace the tabular representation:

1. Modeling when actions are possible
2. Modeling state transitions in a “factored” way:
    - causal rules
    - frame rules: explain how the value of a variable describing a feature at time step t depends on the value of the variable that describes the same feature at time step $t - 1$
        - things that are not changed in the world
        - example: $V_4 = v_7$ is maintained when condition c holds
        - the need for frame rules is counter-intuitive: but remember, a computer doesn’t know that the variables describing the same feature at different time steps have anything to do with each other, unless they are related to each other using appropriate constraints.

Lecture Overview

The STRIPS Representation

- The previous representation was feature-centric:
  - for every feature, where does its value come from?
- 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.
- Historically, STRIPS divided the effects list into an add list and delete list
  - this makes sense when all variables are Boolean