

Lecture 3

Search and Constraints

Wednesday 27 August 2003

William H. Hsu

Department of Computing and Information Sciences, KSU

<http://www.kddresearch.org>

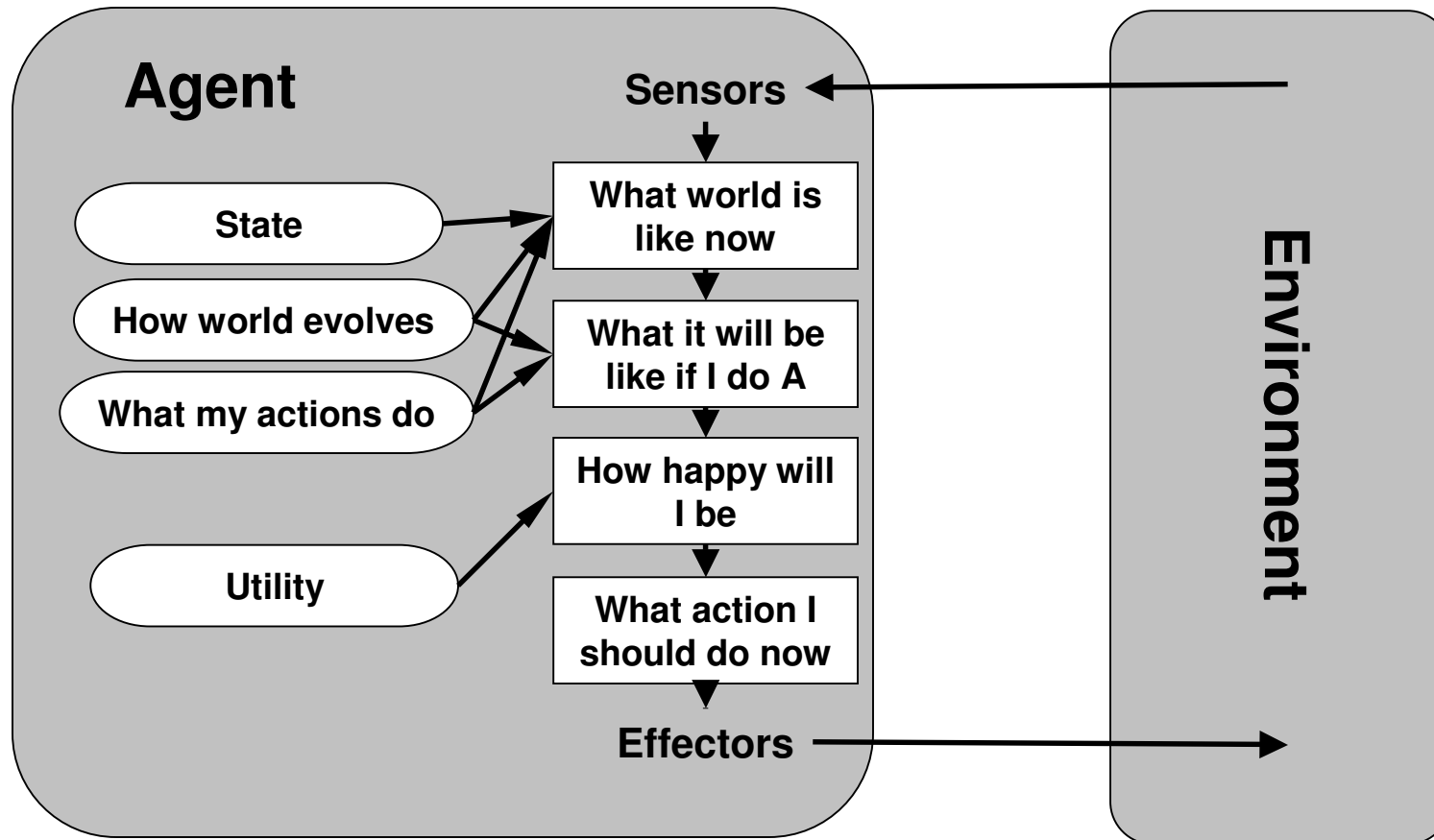
<http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:
Sections 4.1-4.2, Russell and Norvig

Lecture Outline

- **Today's Reading: Sections 3.5-3.8, Russell and Norvig**
- **Thinking Exercises (Discussion): 3.3 (a, b, e), 3.9**
- **Solving Problems by Searching**
 - Problem solving agents: design, specification, implementation
 - Specification components
 - Problems – formulating *well-defined* ones
 - Solutions – requirements, constraints
 - Measuring performance
- **Formulating Problems as (State Space) Search with Backtracking**
- **Example Search Problems**
 - Toy problems: 8-puzzle, 8-queens, cryptarithmic, toy robot worlds, constraints
 - Real-world problems: layout, scheduling
- **Data Structures Used in Search**
- **Uninformed Search: Depth-First, Breadth-First, Branch-and-Bound**
- **Next Tuesday: Informed Search Strategies (see handouts)**

Agent Frameworks: Utility-Based Agents



Review: Problem-Solving Agents

- **function *Simple-Problem-Solving-Agent* (*p*: percept) returns *a*: action**
 - **inputs**: *p*, percept
 - **static**:
 - s*, action sequence (initially empty)
 - state*, description of current world state
 - g*, goal (initially null)
 - problem*, problem formulation
 - *state* ← *Update-State* (*state*, *p*)
 - if *s.Is-Empty*() then
 - *g* ← *Formulate-Goal* (*state*) // focus of today's class
 - *problem* ← *Formulate-Problem* (*state*, *g*) // focus of today's class
 - *s* ← *Search* (*problem*) // next 3 classes
 - *action* ← *Recommendation* (*s*, *state*)
 - *s* ← *Remainder* (*s*, *state*) // discussion: meaning?
 - return (*action*)
- **Chapters 3-4: Implementation of *Simple-Problem-Solving-Agent***

Formulating Problems [1]: Single versus Multi-State

- **Single-State Problems**
 - Goal state is reachable in one action (one move)
 - World is fully accessible
 - Example: vacuum world (Figure 3.2, R&N) – simple robot world
 - Significance
 - Initial step analysis
 - “Base case” for problem solving by regression (General Problem Solver)
- **Multi-State Problems**
 - Goal state may not be reachable in one action
 - Assume limited access: *effects of actions known* (may or may not have sensors)
 - Significance
 - Need to reason over states that agent can get to
 - *May be able to guarantee reachability of goal state anyway*
- **Determining A State Space Formulation**
 - State space – single-state problem
 - State set space – multi-state problems

Formulating Problems [2]: Issues

- **Contingency**
 - Scenario: requirement for sensing *during* execution phase
 - e.g., “turn on vacuum *only if* sensors show dirt present”
 - Basis for advanced planning (Chapter 13 R&N) – e.g., conditional
- **Interleaving**
 - Scenario: agent can act *before* it has found complete plan
 - Basis for
 - Concurrent search and execution (Chapters 5, 13 R&N)
 - Anytime algorithms: online, incremental
- **Exploration Problems**
 - Scenario: agent does *not* know effects of actions (“navigating without map”)
 - Experimentation in environment required (Chapter 20, R&N – CIS830)
- **Other**
 - Uncertainty in problem solving (Chapters 14-17 R&N)
 - Learning to solve problems (Chapters 18-21 R&N)

Defining Problems

- **Definition**
 - *Collection of information used by agent to decide on actions*
 - **First specification: single-state problem**
- **State Space: Definitions**
 - **State space**: set of states reachable from initial state by any action sequence
 - **Path**: sequence of actions leading from one state to another
- **Given**
 - **Initial state**: agent's knowledge of current location, situation of world
 - **Operator set**: agent's knowledge of possible action
 - **Operator**: *description* of action in terms of **state transition** mapping
 - **Successor function**: alternative formulation – reachable states in one action
 - **Goal test**: boolean test for termination (e.g., explicit set of “accepting” states)
 - **Path cost function**: sum, g , of individual costs over sequence of actions
- **datatype *Problem* of (Initial-State, Operators, Goal-Test, Cost-Function)**

Defining Solutions

- **What Is A Solution?**
 - Based on previous *problem definition*
 - Requirements
 - Satisfies goal test
 - Consists of sequence of legal actions
 - Possible constraints (criteria)
 - Plausibility: adaptation of “legal” to uncertain domains
 - Optimality: path cost minimization (*online*)
 - Efficiency: search (*offline*)
- **Towards Finding Solutions**
 - State space search
 - Process: systematic exploration of representation of state space
 - One implementation: graph search
 - Subject to objectives: requirements, possible constraints

Measuring Problem-Solving Performance

- **Search Cost**
 - Measures cost of applying actions
 - Some typical units: time, computer memory (primary / secondary)
 - Incurred during interaction with environment
 - Called offline cost in theoretical computer science
 - Incurred during interaction with environment
 - Formal analytical indicator of search cost: asymptotic complexity
- **Path Cost**
 - Measures cost of applying actions
 - Some typical units: distance, energy, resources, risk (e.g., micromorts)
 - Often attributed (as satellite data) to edges of state space graph
 - Called online cost in theoretical computer science
 - Incurred during interaction with environment
 - Discussion: is path cost always incurred “later”?
- **Total Cost = Search Cost + Path Cost**

Choosing States and Actions

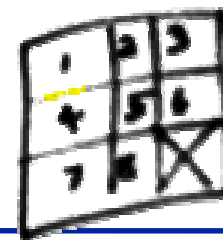
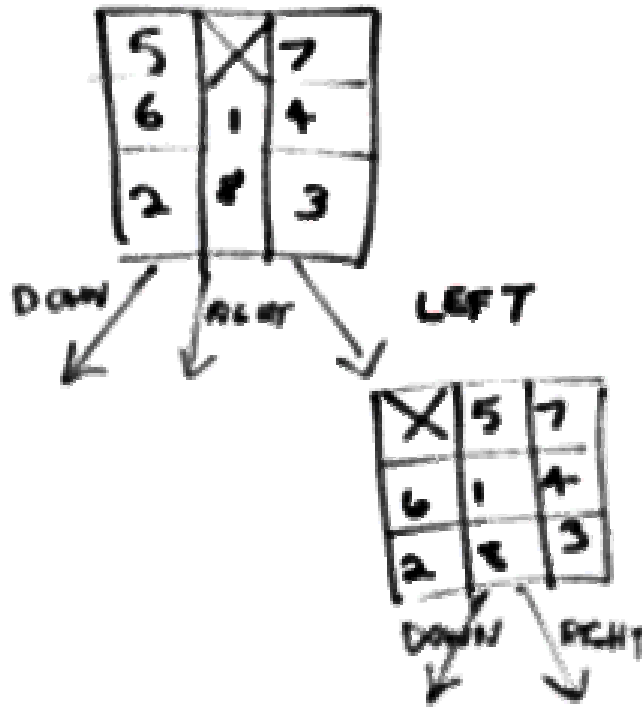
- **Intuitive Ideas**
 - Now have: specification of problem, solution
 - Example: “Drive from A to B using the roads in the map in Figure 3.3 R&N”
 - *How to determine path cost function?*
 - Depends on goals
 - Example 1: total mileage
 - Example 2: expected travel time
 - Examples 3a, 3b: cities visited (*positive or negative?!*)
 - May *itself* be problem to be optimized (by *search!*)
 - *What aspects of world state should be represented?*
 - Again, depends: on details of operators, states needed to *make decisions*
 - Example: traveling companions, radio broadcast, resources (food / fuel)
- **Example: Navigation (Simplified Single-Pairs Shortest Path)**
 - Suppose path cost is number of cities visited (to be minimized)
 - What assumptions are made? (hint: what does agent know?)
 - Is regression (abstraction in problem formulation) needed in “real life”?

Abstraction in AI

- **Why Not Exhaustively Represent World?**
 - Too much detail – intractable:
 - Representation
 - Problem solving (e.g., search and decision problems)
 - Not feasible to implement perception of state of world
 - Sampling (sensor bandwidth)
 - Updating (memory bandwidth)
- **Eliminating Irrelevant Detail**
 - Eliminate granularity (e.g., frequency of measurement, *aka* resolution)
 - Spatial (location, distance)
 - Temporal (time, ordering of events)
 - What to reduce
 - Precision of measurements
 - Exactness or crispness of qualitative and quantitative assertions
 - Some times need to do this in vague domains anyway (what is “vague”?)
- **Discussion: How Can Abstraction Be Generalized to Other Problems?**

Toy Problem Example [1]: 8-Puzzle

- **Objectives (Informal)**
 - Given: permutation of 8 squares plus “blank”, allowable moves (of blank)
 - Achieve: specified ordering (1, 2, 3, 4, 5, 6, 7, 8,_)
- **States**
 - (x, n) denoting that square n is at x
 - Could also use Cartesian coordinates – ramifications?
 - Initial state: “scrambled” but a reachable permutation
- **Operators**
 - Move blank
 - Precondition: $(x, n), (x', _)$
 - Assert: $(x', n), (x, _)$ – here’s where representation helps...
 - Delete: (x, n)
- **Goal Test: Specified Ordering Achieved?**
 - How to represent test?
 - Efficiency issues?
- **Path Cost: Number of Moves**



Toy Problem Example [5]: Simple Constraint Problems

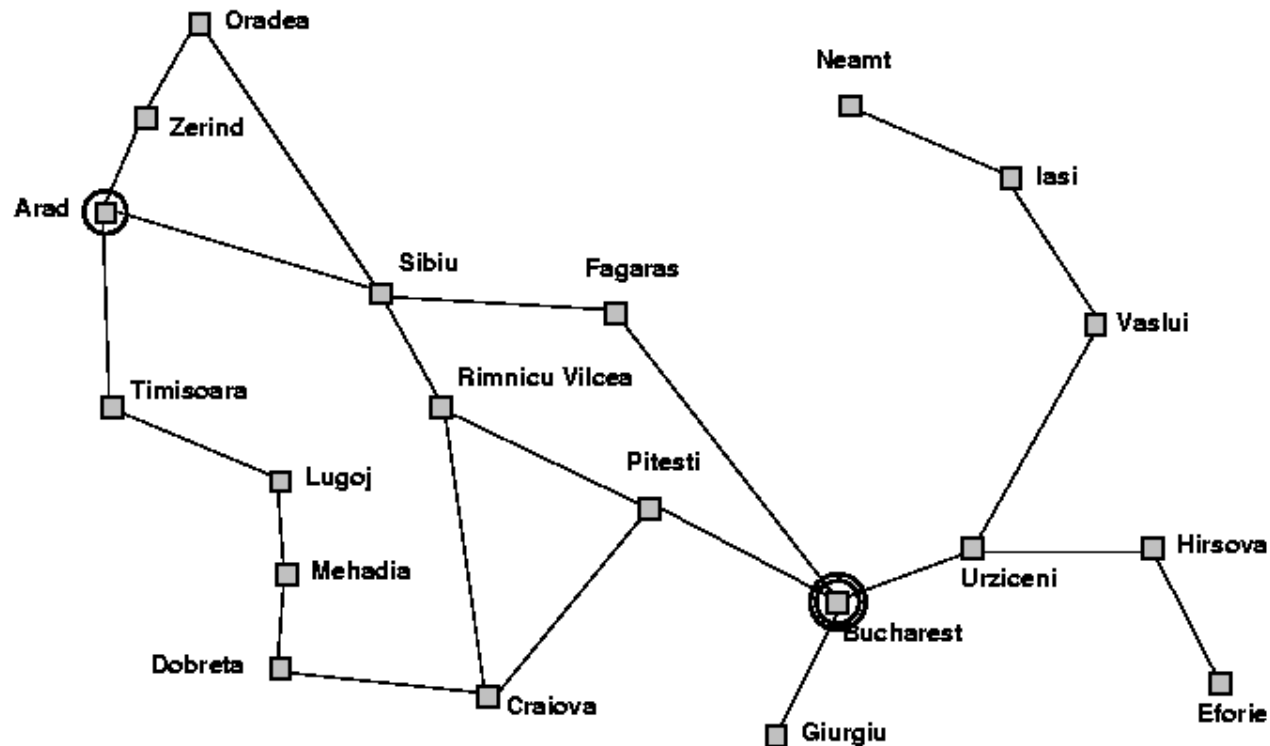
- **Missionaries and Cannibals (Microserfs and Open Source Advocates)**
 - Objectives (informal)
 - Given: M1, M2, M3, C1, C2, C3, 2-person canoe (holds 1-2 people)
 - Achieve: all people on opposite bank without violating constraint
 - States: people on each bank (exercise: better rep?)
 - Operators: ferry (*Passenger-1*, *Passenger-2*)
 - Parity can be implicit or not
 - Constraint on postcondition: cannibals can't outnumber missionaries on bank
 - Goal test: trivial
 - Discussion: <http://www-formal.stanford.edu/jmc/elaboration/node2.html>
- **Farmer, Fox, Goose, Grain**
 - Objectives (informal): (F,X,G,R | empty) → (empty | F,X,G,R)
 - States: entities on each side
 - Operators: ferry (*Entity-1*, *Entity-2*)
 - Goal test: unique final state (equality)
- **Other Constraint Problems: Cryparithmetic, Monkeys and Bananas**

Real-World Problems

- **Route Finding**
 - Objectives (informal): finding shortest path for situated agent – exploration cost
 - States: graph representation (see Machine Problem 1); implicit representations
 - Operators: move from location to location; other degrees of freedom (navigation)
 - Goal test: “are we there yet?”; “did we get there in time?”; “found target?”
- **Travelling Salesperson Problems (TSP) and Other Touring Problems**
 - aka Hamiltonian tour
 - Objectives (informal): finding shortest cost tour that visits all $v \in V$ *exactly once*
 - States: current location in V of agent
 - Operators: visit a neighbor (constraint: previously unvisited)
 - Goal test:
- **Other**
 - Very Large-Scale Integrated (VLSI) circuit layout
 - Robot navigation
 - Assembly sequencing (possible real-time scheduling application)

Blind Search Example (Russell and Norvig)

Example: Romania



Terminology

- **State Space Search**
 - Initial state / conditions, goal test, operator set, path costs
 - Graph formulation
 - Definitions: vertex (node) set V , edge (link, arc) set $E \subseteq V \times V$
 - Unbounded graphs: infinite V , E sets
- **Constraint Satisfaction Problems**
- **Uninformed (Blind) Search Algorithms**
 - Properties of *algorithms*: completeness, optimality, optimal efficiency
 - Depth-first search (DFS)
 - “British Museum” search
 - “Breadth-first search (BFS)
 - Branch-and-bound search – from operations research (OR)
- **Problems**
 - Dealing with path costs
 - Heuristics (next)

Summary Points

- **Today's Reading: Sections 3.5-3.8, Russell and Norvig**
- **Solving Problems by Searching**
 - Problem solving agents: design, specification, implementation
 - Specification components
 - Problems – formulating *well-defined* ones
 - Solutions – requirements, constraints
 - Measuring performance
- **Formulating Problems as (State Space) Search**
- **Example Search Problems**
 - Toy problems: 8-puzzle, 8-queens, cryptarithmic, toy robot worlds, constraints
 - Real-world problems: layout, scheduling
- **Data Structures Used in Search**
- **Uninformed Search Algorithms: BFS, DFS, Branch-and-Bound**
- **Next Tuesday: Informed Search Strategies**
 - State space search handout (Winston)
 - Search handouts (Ginsberg, Rich and Knight)