



Lecture 6 of 42

Informed Search: A/A* Properties, Hill-Climbing, Beam Search

Wednesday, 06 September 2006

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course page: <http://snipurl.com/v9v3>

Course web site: <http://www.kddresearch.org/Courses/Fall-2006/CIS730>

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

Reading for Next Class:

Sections 5.1 – 5.3, p. 137 – 151, Russell & Norvig 2nd edition

Instructions for writing project plans, submitting homework



Lecture Outline

- **Reading for Next Class: Sections 5.1 – 5.3, R&N 2^e**
- **This Week: Chapter 4 concluded; Chapter 5**
 - * Properties of search algorithms, heuristics
 - * Local search (hill-climbing, Beam) vs. nonlocal search
 - * Constraint Satisfaction Problems (CSP)
 - * State space search: graph vs. constraint representations
 - * Search and games (start of Chapter 6)
- **Today: Sections 4.2 – 4.3**
 - * Properties of heuristics: consistency, admissibility, monotonicity
 - * Impact on A/A*
 - * Problems in heuristic search: plateaux, “foothills”, ridges
 - * Escaping from local optima
 - * Wide world of global optimization: genetic algorithms, simulated annealing
- **Friday, next Monday: Chapter 5 on CSP**





Search-Based Problem Solving: Quick Review

- **function** *General-Search* (*problem*, *strategy*) **returns a solution or failure**
 - * Queue: represents search **frontier** (see: Nilsson – OPEN / CLOSED lists)
 - * Variants: based on “add resulting nodes to search tree”
- **Previous Topics**
 - * Formulating *problem*
 - * Uninformed search
 - ⇒ No heuristics: only $g(n)$, if any cost function used
 - ⇒ Variants: BFS (uniform-cost, bidirectional), DFS (depth-limited, ID-DFS)
 - * Heuristic search
 - ⇒ Based on h – (heuristic) function, returns estimate of min cost to goal
 - ⇒ h only: **greedy (aka myopic)** informed search
 - ⇒ A/A*: $f(n) = g(n) + h(n)$ – **frontier** based on *estimated + accumulated cost*
- **Today: More Heuristic Search Algorithms**
 - * A* extensions: iterative deepening (IDA*), **simplified memory-bounded (SMA*)**
 - * Iterative improvement: hill-climbing, MCMC (simulated annealing)
 - * Problems and solutions (macros and global optimization)



Properties of Algorithm A/A*: Review

- **Admissibility: Requirement for A* Search to Find Min-Cost Solution**
- **Related Property: Monotone Restriction on Heuristics**
 - * For all nodes m, n such that m is a descendant of n : $h(m) \geq h(n) - c(n, m)$
 - * Discussion questions
 - ⇒ **Admissibility** ⇒ **monotonicity**? **Monotonicity** ⇒ **admissibility**?
 - ⇒ What happens if monotone restriction is violated? (Can we fix it?)
- **Optimality Proof for Admissible Heuristics**
 - * **Theorem**: If $\forall n . h(n) \leq h^*(n)$, A* will never return a suboptimal goal node.
 - * **Proof**
 - ⇒ Suppose A* returns x such that $\exists s . g(s) < g(x)$
 - ⇒ Let path from root to s be $\langle n_0, n_1, \dots, n_k \rangle$ where $n_k \equiv s$
 - ⇒ Suppose A* expands a subpath $\langle n_0, n_1, \dots, n_j \rangle$ of this path
 - ⇒ **Lemma**: by induction on i , $s = n_k$ is expanded as well
 - Base case: n_0 (root) always expanded
 - Induction step: $h(n_{j+1}) \leq h^*(n_{j+1})$, so $f(n_{j+1}) \leq f(x)$, Q.E.D.
 - ⇒ **Contradiction**: if s were expanded, A* would have selected s , not x





A/A*: Extensions (IDA*, RBFS, SMA*)

- **Memory-Bounded Search** (p. 101 – 104, R&N 2e)
 - * **Rationale**
 - ⇒ Some problems intrinsically difficult (intractable, exponentially complex)
 - ⇒ “Something’s got to give” – size, time or memory? (“Usually memory”)
- **Recursive Best-First Search** (p. 101 – 102 R&N 2e)
- **Iterative Deepening A*** – Pearl, Korf (p. 101, R&N 2e)
 - * **Idea:** use iterative deepening DFS with sort on f – expands node *iff* A* does
 - * **Limit on expansion:** f -cost
 - * **Space complexity:** linear in depth of goal node
 - * **Caveat:** could take $O(n^2)$ time – e.g., TSP ($n = 10^6$ could still be a problem)
 - * **Possible fix**
 - ⇒ Increase f cost limit by ϵ on each iteration
 - ⇒ **Approximation error bound:** no worse than ϵ -bad (ϵ -admissible)
- **Simplified Memory-Bounded A*** – Chakrabarti, Russell (p. 102-104)
 - * **Idea:** make space on queue as needed (compare: virtual memory)
 - * **Selective forgetting:** drop nodes (select victims) with highest f



Best-First Search Problems [1]: Global vs. Local Search

- **Optimization-Based Problem Solving as Function Maximization**
 - * **Visualize function space**
 - ⇒ **Criterion** (z axis)
 - ⇒ **Solutions** (x - y plane)
 - * **Objective:** *maximize criterion* subject to
 - ⇒ **Solution spec**
 - ⇒ **Degrees of freedom**
- **Foothills aka Local Optima**
 - * **aka relative minima (of error), relative maxima (of criterion)**
 - * **Qualitative description**
 - ⇒ All applicable operators produce suboptimal results (i.e., neighbors)
 - ⇒ *However, solution is not optimal!*
 - * **Discussion:** *Why does this happen in optimization?*





Best-First Search Problems [2]

- **Lack of Gradient aka Plateaux**
 - * Qualitative description
 - ⇒ All neighbors indistinguishable
 - ⇒ According to evaluation function f
 - * Related problem: jump discontinuities in function space
 - * Discussion: *When does this happen in heuristic problem solving?*
- **Single-Step Traps aka Ridges**
 - * Qualitative description: unable to move along steepest gradient
 - * Discussion: *How might this problem be overcome?*



Hill-Climbing aka Gradient Descent

- **function Hill-Climbing (problem) returns solution state**
 - * inputs: *problem*: specification of problem (structure or class)
 - * static: *current, next*: search nodes
 - * *current* ← *Make-Node (problem.Initial-State)*
 - * loop do
 - ⇒ *next* ← a highest-valued successor of *current*
 - ⇒ if *next.value()* < *current.value()* then return current
 - ⇒ *current* ← *next* // make transition
 - * end
- **Steepest Ascent Hill-Climbing**
 - * aka gradient ascent (descent)
 - * Analogy: finding “tangent plane to objective surface”
 - * Implementations
 - ⇒ Finding derivative of (differentiable) f with respect to parameters
 - ⇒ Example: error backpropagation in artificial neural networks (later)
- **Discussion: Difference Between Hill-Climbing, Best-First?**





Iterative Improvement: Framework

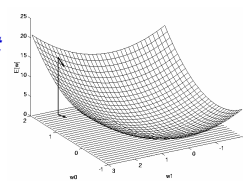
- **Intuitive Idea**
 - * “Single-point search frontier”
 - ⇒ Expand one node at a time
 - ⇒ Place children at head of queue
 - ⇒ Sort *only this sublist*, by f
 - * **Result** – direct convergence in direction of steepest:
 - ⇒ Ascent (in criterion)
 - ⇒ Descent (in error)
 - * Common property: proceed toward goal *from search locus (or loci)*
- **Variations**
 - * **Local** (steepest ascent hill-climbing) versus **global** (simulated annealing or SA)
 - * **Deterministic** versus **Monte-Carlo**
 - * **Single-point** versus **multi-point**
 - ⇒ Maintain frontier
 - ⇒ Systematic search (cf. OPEN / CLOSED lists): parallel SA
 - ⇒ Search with recombination: genetic algorithm



Hill-Climbing [1]: An Iterative Improvement Algorithm

- **function** *Hill-Climbing* (*problem*) **returns** solution state
 - * **inputs:** *problem*: specification of problem (structure or class)
 - * **static:** *current*, *next*: search nodes
 - * *current* ← *Make-Node* (*problem.Initial-State*)
 - * **loop do**
 - ⇒ *next* ← a highest-valued successor of *current*
 - ⇒ **if** *next.value()* < *current.value()* **then return** *current*
 - ⇒ *current* ← *next* // make transition
 - * **end**
- **Steepest Ascent Hill-Climbing**
 - * **aka gradient ascent** (descent)
 - * **Analogy:** finding “tangent plane to objective surface”
 - * **Implementations**
 - ⇒ Finding derivative of (differentiable) f with respect to parameters
 - ⇒ Example: error backpropagation in artificial neural networks (later)
- **Discussion:** Difference Between Hill-Climbing, Best-First?

$$\nabla E[\mathbf{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$





Hill-Climbing [2]: A Restriction of Best-First Search

- **Discussion:** How is Hill-Climbing a Restriction of Best-First?
- **Answer: Dropped Condition**
 - * **Best first:** sort by h or f over *current frontier*
 - ⇒ Compare: insert each element of expanded node into queue, in order
 - ⇒ Result: greedy search (h) or A/A* (f)
 - * **Hill climbing:** sort by h or f *within child list of current node*
 - ⇒ Compare: local bucket sort
 - ⇒ **Discussion (important):** *Does it matter whether we include g ?*
- **Impact of Modification on Algorithm**
 - * Search time complexity decreases
 - * Comparison with A/A* (Best-First using f)
 - ⇒ *Still optimal?* No
 - ⇒ *Still complete?* Yes
 - * Variations on hill-climbing (later): momentum, random restarts



Hill-Climbing [3]: Local Optima (Foothill Problem)

- **Local Optima aka Local Trap States**
- **Problem Definition**
 - * Point reached by hill-climbing may be maximal but not maximum
 - * **Maximal**
 - ⇒ **Definition:** *not dominated by any neighboring point (with respect to criterion measure)*
 - ⇒ In this partial ordering, maxima are incomparable
 - * **Maximum**
 - ⇒ **Definition:** *dominates all neighboring points (wrt criterion measure)*
 - ⇒ Different partial ordering imposed: "z value"
- **Ramifications**
 - * Steepest ascent hill-climbing will become trapped (*why?*)
 - * Need some way to break out of trap state
 - ⇒ Accept transition (i.e., search move) to dominated neighbor
 - ⇒ Start over: random restarts





Hill-Climbing [4]: Lack of Gradient (Plateau Problem)

- Zero Gradient Neighborhoods aka Plateaux
- **Problem Definition**
 - * Function space may contain points whose neighbors are *indistinguishable* (wrt criterion measure)
 - * Effect: “flat” search landscape
 - * Discussion
 - ⇒ *When does this happen in practice?*
 - ⇒ *Specifically, for what kind of heuristics might this happen?*
- **Ramifications**
 - * Steepest ascent hill-climbing will become trapped (*why?*)
 - * Need some way to break out of zero gradient
 - ⇒ Accept transition (i.e., search move) to random neighbor
 - ⇒ Random restarts
 - ⇒ *Take bigger steps* (later, in planning)



Hill-Climbing [5]: Single-Step Traps (Ridge Problem)

- Single-Step Traps aka Ridges
- **Problem Definition**
 - * Function space may contain points such that single move in any “direction” leads to suboptimal neighbor
 - * Effect
 - ⇒ There exists steepest gradient to goal
 - ⇒ None of allowed steps moves along that gradient
 - ⇒ Thin “knife edge” in search landscape, hard to navigate
 - ⇒ Discussion (important): *When does this occur in practice?*
 - * **NB:** ridges can lead to local optima, too
- **Ramifications**
 - * Steepest ascent hill-climbing will become trapped (*why?*)
 - * Need some way to break out of ridge-walking
 - ⇒ Formulate composite transition (multi-dimension step) – *how?*
 - ⇒ Accept multi-step transition (at least one to worse state) – *how?*
 - ⇒ Random restarts





Ridge Problem Solution: Multi-Step Trajectories (Macros)

- **Intuitive Idea: Take More than One Step in Moving along Ridge**
- **Analogy: Tacking in Sailing**
 - * Need to move against wind direction
 - * Have to compose move from multiple small steps
 - ⇒ *Combined move*: in (or more toward) direction of steepest gradient
 - ⇒ Another view: *decompose* problem into self-contained *subproblems*
- **Multi-Step Trajectories: Macro Operators**
 - * **Macros**: (inductively) generalize from 2 to > 2 steps
 - * **Example**: Rubik's Cube
 - ⇒ Can solve 3 x 3 x 3 cube by solving, interchanging 2 x 2 x 2 **subcubes**
 - ⇒ *Knowledge* used to formulate **subcube** (cubie) as macro operator
 - * *Treat operator as single step* (multiple **primitive** steps)
- **Discussion: Issues**
 - * *How can we be sure macro is **atomic**? What are pre-, postconditions?*
 - * *What is good **granularity** (size of basic step) for macro in our problem?*



Plateau, Local Optimum, Ridge Solution: Global Optimization

- **Intuitive Idea**
 - * Let search algorithm take some "bad" steps to escape from trap states
 - * *Decrease* probability of such steps *gradually* to *prevent return to traps*
- **Analogy: Marble(s) on Rubber Sheet**
 - * **Goal**: move marble(s) into global minimum from any starting position
 - * **Shake system**: hard at first, gradually decreasing vibration
 - * Tend to break out of local minima but have less chance of re-entering
- **Analogy: Annealing**
 - * Ideas from metallurgy, statistical thermodynamics
 - * **Cooling** molten substance: slow as opposed to rapid (quenching)
 - * **Goal**: maximize material strength of substance (e.g., metal or glass)
- **Multi-Step Trajectories in Global Optimization: Super-Transitions**
- **Discussion: Issues**
 - * What does **convergence** mean?
 - * What **annealing schedule** guarantees convergence?





Beam Search: “Parallel” Hill-Climbing

- **Idea**
 - * Teams of climbers
 - ⇒ Communicating by radio
 - ⇒ Frontier is only w teams wide ($w \equiv$ beam width)
 - ⇒ Expand cf. best-first but take best w only per layer
 - * Synchronous search: push frontier out to uniform depth from start node
- **Algorithm Details**
 - * How do we order OPEN (priority queue) by h ?
 - * How do we maintain CLOSED?
- **Question**
 - * What behavior does beam search with $w = 1$ exhibit?
 - * Hint: only one “team”, can’t split up!
 - * Answer: equivalent to hill-climbing
- **Other Properties, Design Issues**
 - * Another analogy: flashlight *beam* with adjustable radius (hence name)
 - * What should w be? How will this affect solution quality?



Iterative Improvement Global Optimization (GO) Algorithms

- **Idea: Apply Global Optimization with Iterative Improvement**
 - * Iterative improvement: local transition (primitive step)
 - * Global optimization algorithm
 - ⇒ “Schedules” exploration of landscape
 - ⇒ Selects next state to visit
 - ⇒ Guides search by specifying probability distribution over local transitions
- **Brief History of Markov Chain Monte Carlo (MCMC) Family**
 - * MCMC algorithms first developed in 1940s (Metropolis)
 - * First implemented in 1980s
 - ⇒ “Optimization by simulated annealing” (Kirkpatrick *et al.*, 1983)
 - ⇒ Boltzmann machines (Ackley, Hinton, Sejnowski, 1985)
 - * Tremendous amount of research and application since
 - ⇒ Neural, genetic, Bayesian computation
 - ⇒ See: CIS730 Class Resources page





Plan Interviews: Next Week

- **10-15 Minute Meeting**
- **Discussion Topics**
 - * Background resources
 - * Revisions needed to project plan
 - * Literature review: bibliographic sources
 - * Source code provided for project
 - * Evaluation techniques
 - * Interim goals
 - * Your timeline
- **Dates and Venue**
 - * Week of Mon 11 Sep 2006
 - * Sign up for times by e-mailing CIS730TA-L@listserv.ksu.edu
- **Come Prepared**
 - * Hard copy of plan draft
 - * Have demo running
 - ⇒ Installed on notebook if you have one
 - ⇒ Remote desktop, VNC, or SSH otherwise



Plan Selections

- **Game-Playing Expert System**
 - * Channell, Lamar (distance)
 - * Davis, Eric (distance)
 - * Evans, Ryan
 - * Hart, Jack
 - * Linda, Ondrej
- **Trading Agent Competition (TAC) – Supply Chain Management**
 - * Kugler, Tom
 - * Jordan, Kevin (distance)
 - * Wilsey, Nick
- **Evidence Ontology**
 - * Jantz, Karen (auditing / CIS 499)
 - * Schoenhofer, Aaron
 - * Xia, Jing
- **TBD: Bhatia, Erande, Forster (distance), Lupo, Hercula, Panday, Stampbach (send e-mail to CIS730TA-L today!)**





Instructions for Project Plans

- **Note: Project Plans Are Not Proposals!**
 - * Subject to (one) revision
 - * Choose one topic among three
- **Plan Outline: 1-2 Pages**
 - * **1. Problem Statement**
 - ⇒ Objectives
 - ⇒ Scope
 - * **2. Background**
 - ⇒ Related work
 - ⇒ Brief survey of existing agents and approaches
 - * **3. Methodology**
 - ⇒ Data resources
 - ⇒ Tentative list of algorithms to be implemented or adapted
 - * **4. Evaluation Methods**
 - * **5. Milestones**
 - * **6. References**



Project Calendar for CIS 490 and CIS 730

- **Plan Drafts – send by Fri 08 Sep 2006 (soft deadline, but by Monday)**
- **Plan Interviews – Mon 11 Sep 2006 – Wed 13 Sep 2006**
- **Revised Plans – submit by Fri 15 Sep 2006 (hard deadline)**
- **Interim Reports – submit by 11 Oct 2006 (hard deadline)**
- **Interim Interviews – around 18 Oct 2006**
- **Final Reports – 29 Nov 2006 (hard deadline)**
- **Final Interviews – around 04 Dec 2006**





Terminology

- **Heuristic Search Algorithms**
 - * **Properties of heuristics:** monotonicity, admissibility, completeness
 - * **Properties of algorithms:** (soundness), completeness, optimality, optimal efficiency
 - * Iterative improvement
 - ⇒ Hill-climbing
 - ⇒ Beam search
 - ⇒ Simulated annealing (SA)
 - * Function maximization formulation of search
 - * **Problems**
 - ⇒ Ridge
 - ⇒ Foothill aka local (relative) optimum aka local minimum (of error)
 - ⇒ Plateau, jump discontinuity
 - * **Solutions**
 - ⇒ Macro operators
 - ⇒ Global optimization (genetic algorithms / SA)
- **Constraint Satisfaction Search**



Summary Points

- **Properties of Search**
 - * **Properties of heuristics:** consistency, admissibility, monotonicity
 - * **Properties of search algorithms**
 - ⇒ Soundness
 - ⇒ Completeness
 - ⇒ Optimality
 - ⇒ Optimal efficiency
 - * How to prove properties of search algorithms
- **Algorithm A (Arbitrary Heuristic) vs. A* (Admissible Heuristic)**
- **Local Search**
 - * **Beam search**
 - * **Hill-climbing (beam width $w = 1$)**
- **Problems in Heuristic Search**
 - * **Plateaux, "foothills", ridges**
 - * **Combatting problems: global and local approaches**