Local Search

Computer Science cpsc322, Lecture 14

(Textbook Chpt 4.8)

February, 4, 2009

Lecture Overview

- Recap CSP (solving CSPs systematically)
- Local search
- Greedy Descent / Hill Climbing: Problems

Systematically solving CSPs: Summary

- Build Constraint Network
- Apply Arc Consistency
  - One domain is empty → no solution
  - Each domain has a single value → unique solution
  - Some domains have more than one value →
    may or may not be a solution
- Apply Depth-First Search with Pruning
- Split the problem in a number of disjoint cases
  - Apply Arc Consistency to each case

Lecture Overview

- Recap
- Local search
- Greedy Descent / Hill Climbing: Problems

Local Search motivation: Scale

- Many CSPs (scheduling, DNA computing, more later) are simply too big for systematic approaches
- If you have 10^5 vars with dom(var_i) = 10^4

- Systematic Search

  $bf = 10^4$

  $d = 10^5$

  $\boxed{(10^4)^{10^5}}$

  branching factor

  depth

  → # path (worst case)

  → # possible worlds

- Constraint Network

  $\sim 10^{10}$ max # of nodes

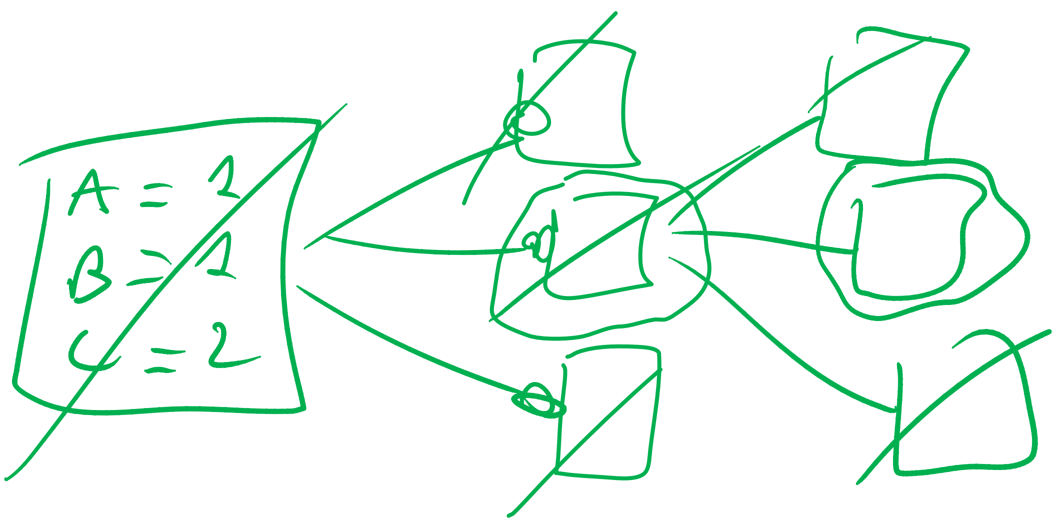
  $\sim (\underbrace{10^5}_{\text{var nodes}} + \underbrace{10^5 * 10^5}_{\text{constraints nodes}})$

- but if solutions are densely distributed…….

Local Search: General Method

- Start from a possible world
- Generate some neighbors (“similar” possible worlds)
- Move from the current node to a neighbor, selected according to a particular strategy

  - Example: A,B,C same domain {1,2,3}



Local Search: Selecting Neighbors

How do we determine the neighbors?

- Usually this is simple: some small incremental change to the variable assignment
  - a) assignments that differ in one variable's value, by a value difference of (+1 for instance)
  - b) assignments that differ in one variable's value
  - c) assignments that differ in two variables' values, etc.

- Example: A,B,C same domain {1,2,3}

a)

A = 1, B = 1, C = 2

A = 2, B = 1, C = 2

A = 1, B = 2, C = 2

A = 1, B = 1, C = 3

3 neighbors

b)

A = 1, B = 1, C = 2

A = 2, B = 1, C = 2

A = 3, B = 1, C = 2

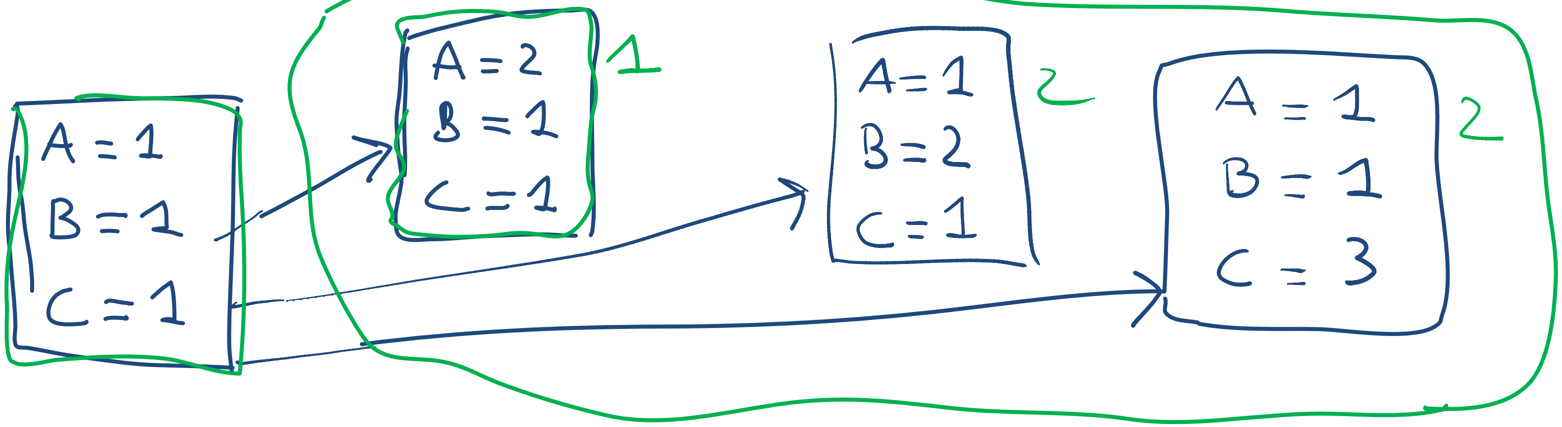
A = 1, B = 2, C = 2

⋮

6 neighbors

Selecting the best neighbor

- Example: A,B,C same domain {1,2,3} , (A=B, A>1, C≠3)



A common component of the scoring function (heuristic) =>
select the neighbor that results in the ……

least # of unsatisfied cons.

- the min conflicts heuristics

Example: *n*-queens

Put *n* queens on an *n* *n* board with no two queens on the same row, column, or diagonal

Example: 4-Queens

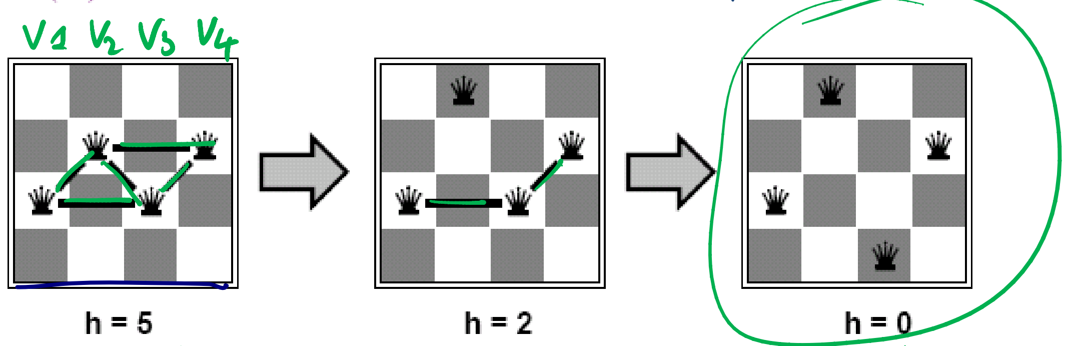
~~States~~: 4 queens in 4 columns ($4^4 = 256$ ~~states~~) possible worlds

Operators: move queen in column (to generate neighbors)

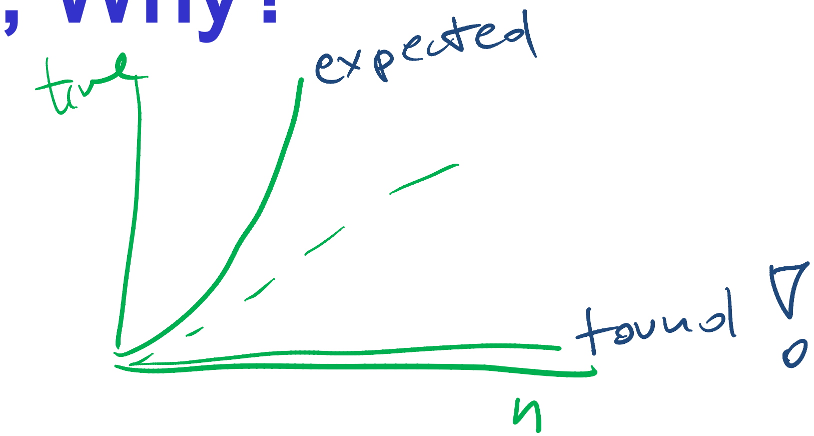
Goal test: no attacks

Evaluation: $h(n)$ = number of attacks ← min conflict scoring function

4 vars one for each column each var can have 4 values {0 1 2 3} ← position of the queen in that column



n-queens, Why?



Why this problem?

Lots of research in the 90’ on local search for CSP was generated by the observation that the run-time of local search on n-queens problems is **independent of problem size!**

Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)

Solutions might have different values

- Example: A,B,C same domain {1,2,3} , (A=B, A>1, C≠3)
- Value = (C+A) and we want a solution that maximize that

A = 1
B = 1
C = 1

n_1: A = 2, B = 1, C = 1 — 1, 3

n_2: A = 1, B = 2, C = 1 — 2, 2

n_3: A = 1, B = 1, C = 3 — 3, 4

we should select n_1

The scoring function we'd like to maximize might be:

f = (C + A) – #-of-conflicts

In our example $f(n_1) = 2$ $f(n_3) = 1$
$f(n_2) = 0$

Greedy Descent means selecting the neighbor which minimizes a scoring function. (min-conflict + cost)
involving the cost of the poss. world

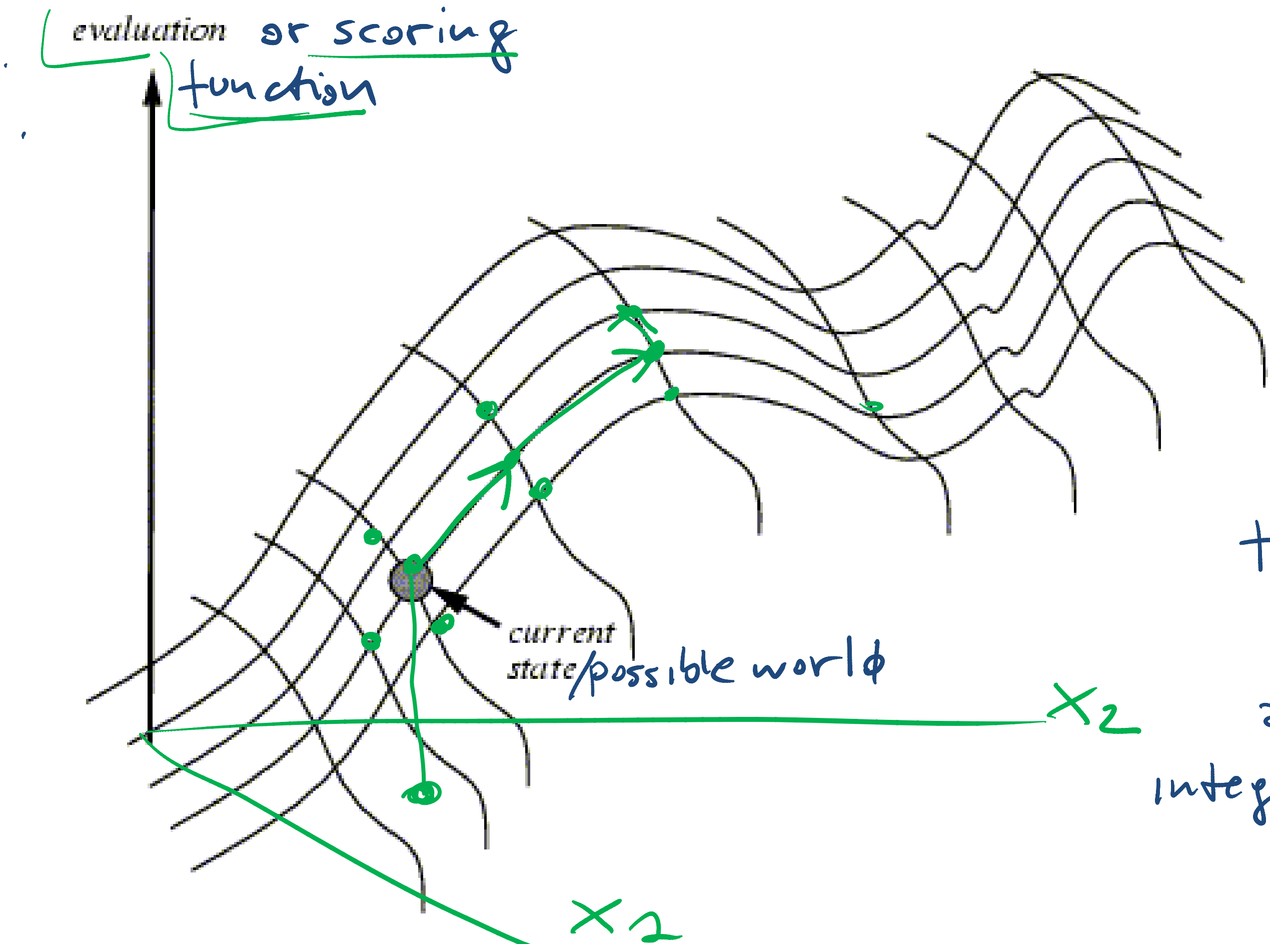
Hill Climbing means selecting the neighbor which best improves a scoring function. involving the value/quality of the poss. world

Lecture Overview

- Recap
- Local search
- **Greedy Descent / Hill Climbing: Problems**

Hill Climbing

NOTE: Everything that will be said for Hill Climbing is also true for Greedy Descent



two vars

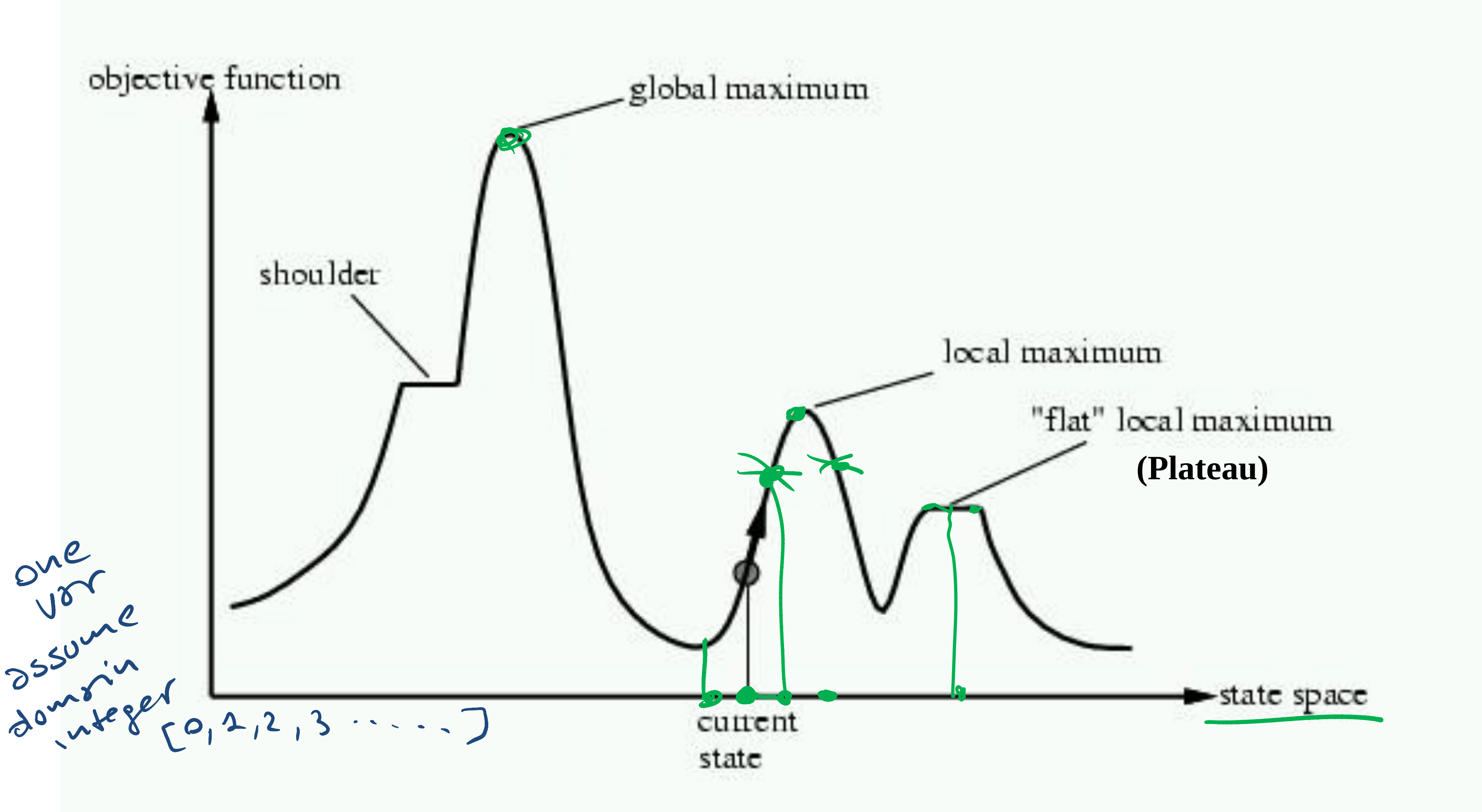
X_1 X_2

assume domain integer [0,1,2....]

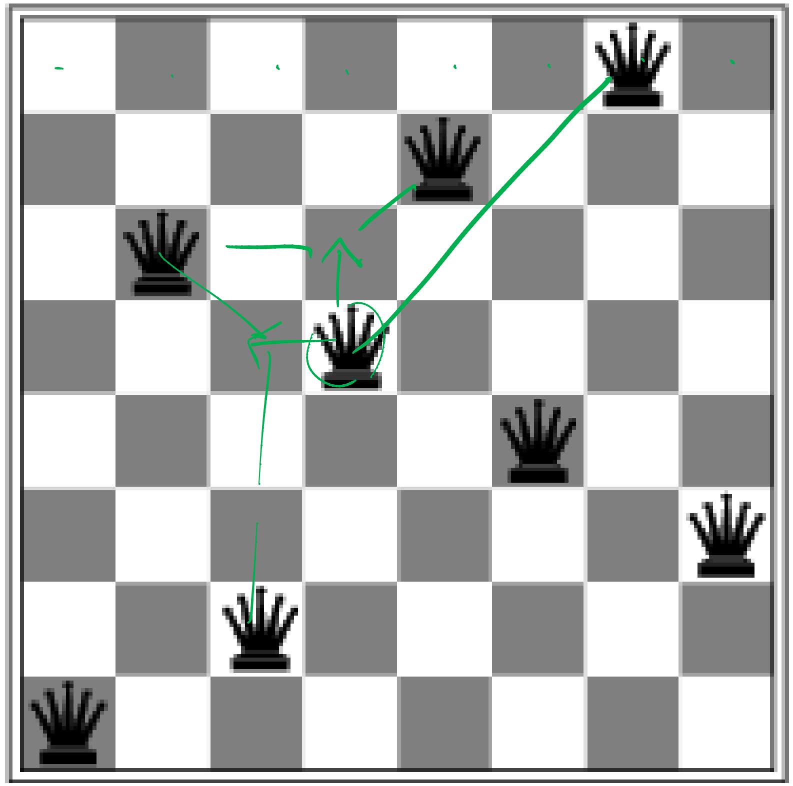
Problems with Hill Climbing

Local Maxima.

Plateau - Shoulders



Corresponding problem for GreedyDescent
Local minimum example: 8-queens problem



for all the moves $h > 1$

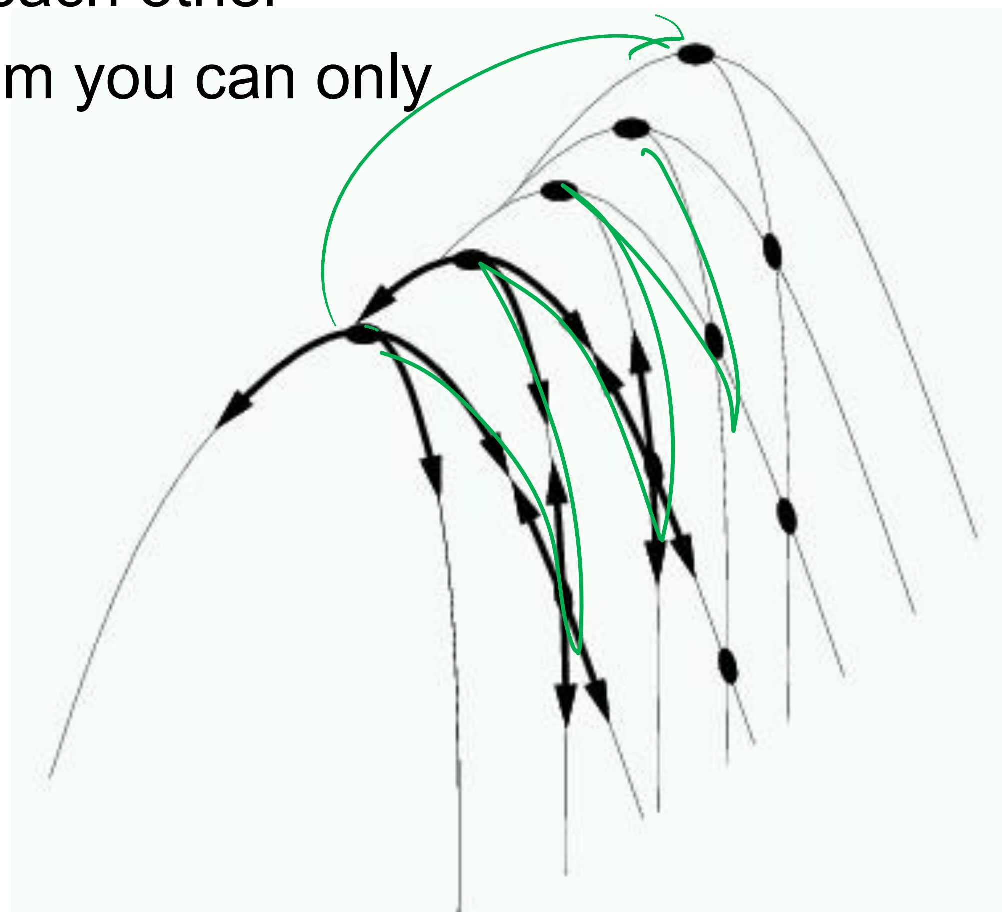
A local minimum with *h = 1*

$\boxed{h = 0}$ for solution

Even more Problems in higher dimensions

E.g., Ridges – sequence of local maxima not directly connected to each other

From each local maximum you can only go downhill



Learning Goals for today's class

You can:

- Implement local search for a CSP.
  - Implement different ways to generate neighbors
  - Implement scoring functions to solve a CSP by local search through either greedy descent or hill-climbing.

Next Class

- How to address problems with Greedy Descent / Hill Climbing?

Stochastic Local Search (SLS)

322 Feedback ☺ or ☹

- Lectures
- Slides
- Practice Exercises
- Assignments
- AIspace
- ……
- Textbook
- Course Topics / Objectives
- TAs
- Learning Goals
- ……