Iterative improvement algorithms

Prof. Tuomas Sandholm
Carnegie Mellon University
Computer Science Department
Iterative improvement algorithms
  = iterative refinement = local search

Usable when the solution are states, not paths.

Start with a complete configuration and make modifications to improve its quality.
Iterative improvement algorithms try to find peaks on a surface of states where height is defined by the evaluation function.
Hill-climbing Search…

function HILL-CLIMBING(problem) returns a solution state

inputs: problem, a problem
static: current, a node
    next, a node

current \leftarrow \text{MAKE-NODE}(\text{INITIAL-STATE}[problem])
loop do
    next \leftarrow \text{a highest-valued successor of current}
    if VALUE[next] < VALUE[current] then return current
    current \leftarrow next
end

Best-swap vs. first-swap
Hill-climbing Search…

Problems:
1. Local maxima
   • No progress
2. Plateaux (essentially flat evaluation fn)
   • Random walk
3. Ridges
   • Search may oscillate from side to side, making little progress

Potential solutions: random restarts
• Eventually finds the optimal solution
• On NP-complete problems it is likely that there are exponentially many local optima

Usually good solutions can be found quickly.
Performance depends on the “state-space surface”.

How to find feasible neighbors?
E.g. 2-swap in TSP

3-swaps...
Simulated Annealing
Simulated Annealing…

function SIMULATED-ANNEALING(problem, schedule) returns a solution state
inputs: problem, a problem
    schedule, a mapping from time to “temperature”
static: current, a node
    next, a node
    T, a “temperature” controlling the probability of downward steps

current ← MAKE-NODE(INITIAL-STATE[problem])
for i ← 1 to ∞ do
    T ← schedule[t]
    if T=0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] – VALUE[current]
    if ΔE > 0 then current ← next
    else current ← next only with probability e^{ΔE/T}

Does not always find an optimal solution, and does not know whether it has found an optimal solution.
[Theoretical results show that if T is lowered slowly enough (extremely slowly), the optimum will be found]
Heuristic Repair

Iterative improvement in CSPs called *heuristic repair*.

**Min-conflicts heuristic**: choose a value that results in a minimum number of conflicts with other variables.
Heuristic Repair (cont.)

A two-step solution for an 8-queen problem using min-conflicts. At each stage, a queen is chosen for reassignment in its column. The number of conflicts (in this case, the number of attacking queens) is shown in each square. The algorithm moves the queen to the min-conflict square, breaking ties randomly.

Surprisingly effective
- $10^6$ queens in 50 steps on average
- Hubble space telescope scheduling
  (3 weeks $\rightarrow$ 10 minutes for scheduling a week of observations)