Problem Solving by Search 3

Uwe Egly

Vienna University of Technology
Institute of Information Systems
Knowledge-Based Systems Group

Outline

Introduction

Formulation of the Problem

Basic Idea Behind Local Search

Hill Climbing

Simulated Annealing

Genetic Algorithms

Overview

Search: very important technique in CS and AI

Different kinds of search:

- Deterministic search
  - Uninformed (“blind”) search strategies ✔
  - Informed or heuristic search strategies: ✔
    use information about problem structure
- Local search
- Search in game trees (not covered in this course)

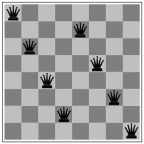
In this lecture: Local search

What is the Problem?

- Search methods mentioned so far are often too expensive
- They systematically explore the state space
  (state space = set of "complete" configurations)
- Solution was a path from the start to a goal
- From now on:
  - path to goal is irrelevant (goal state itself is the solution)
  - Each state n has a score (or objective function) $f(n)$
  - Goal: Find a goal state with best (or reasonable) score
- The focus is therefore on optimization problems

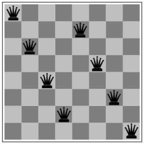
Examples

- N-queens problem:
  - Put N queens on the board without conflicts
  - Let $f(n)$ be the number of conflicting queens in state n
  - Search s with lowest score $f(s)$

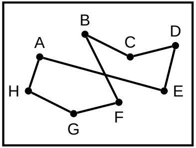


Examples

- N-queens problem:
  - Put N queens on the board without conflicts
  - Let $f(n)$ be the number of conflicting queens in state n
  - Search s with lowest score $f(s)$



- Traveling Salesperson problem (TSP)
  - Start at, e.g., A and make a tour
  - Visit each city ones and return to A
  - State: order of cities, $f(s)$: total kms
  - Problem is NP-hard

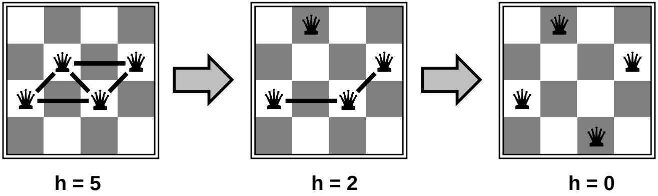


Basic Idea Behind Local Search

- Basic Idea: Perform an iterative improvement
  - Keep a single "current" state (rather than multiple paths)
  - Try to improve it
- Move iteratively to neighbors of the current state
- Do not retain search path
- Constant space, often rather fast, but incomplete
- What is a neighbor?
  - Neighborhood has to be defined application-dependent

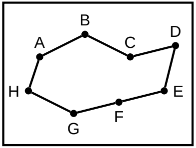
4-queens

- Take leftmost state s with $h(s) = 5$ and try to improve
- Generate neighbor by moving a queen in the column
- For the state t in the middle, $h(t) < h(s)$
- Again, generate neighbor by moving a queen in the column
- Resulting state has score 0 and is a solution



TSP

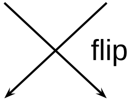
- State s: A-B-C-D-E-F-G-H-A
- $f(s)$: length of the tour
- One possibility: 2-change



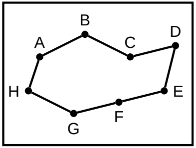
TSP

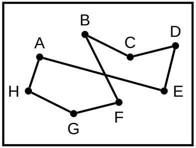
- State s: A-B-C-D-E-F-G-H-A
- $f(s)$: length of the tour
- One possibility: 2-change

A-B-C-D-E-F-G-H-A



A-E-D-C-B-F-G-H-A





Hill Climbing (or Gradient Ascent/Descent)

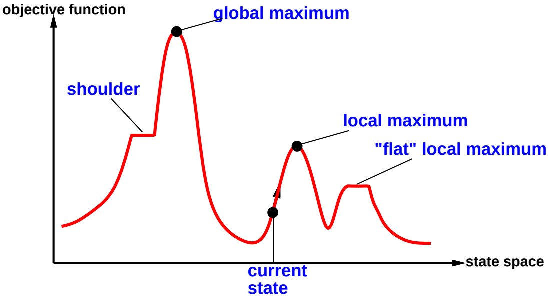
1. Choose an initial state n
2. Compute all neighbors m of n with largest $f(m)$
3. If $f(m) \leq f(n)$ then return n and stop
4. Otherwise, let $n = m$ and continoue with 2

Problems

- Very simple algorithm, get easily stuck in local optima
- How to get the neighbors?
  - By small changes of the state
  - Must be easy to compute
- Choise how to generate the neighbors crucial

Hill Climbing cont'd

How does *f* (the objective function) evolve for varying states?



- Random-restart hill climbing overcomes local maxima
- Random sideways moves
  - Escape from shoulders
  - Loop on flat maxima

Simulated Annealing

Basic Idea

- Escape local optima by allowing some “bad” moves
- But gradually increase their size and frequency

1. Choose an initial state n
2. Randomly pick m from the neighbors of n
3. If $f(m)$ is better than $f(n)$ then set $n = m$
4. Otherwise /* if m is worse than n */
   With a small probability p, set $n = m$
5. Continoue with 2. until time limit is reached

- p decreases over time and when $|f(n) - f(m)|$ increases

Simulated Annealing Cont'd

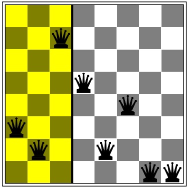
- If $f(m)$ better than $f(n)$, then always continoue with m
- Otherwise, take m with probablilty

$$\exp\left(-\frac{|f(n)-f(m)|}{T}\right)$$

- Probability decreases exponentially with the badness $|f(n)-f(m)|$
- The temperature parameter T is decreased over time ("cooling" or "annealing")
- If badness is large, the probability is small

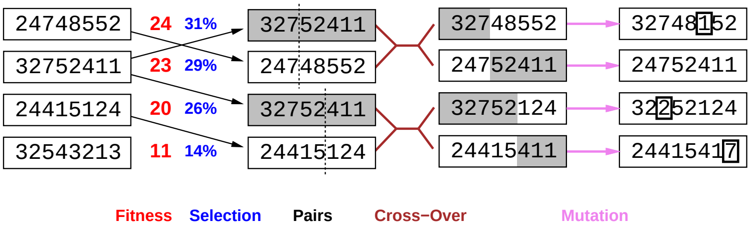
Genetic Algorithms (GAs)

- GAs: heuristic stochastic search algos
- GAs require states encoded as strings
- State (right) encodes as (3 2 7 5 2 4 1 1)
  (places of the queens from bottom)
- $f(n)$ is called the fitness of n
  Goal: Find fittest n, i.e., find global optimum
- Keep a fixed number of states
  (They are called the population)



Genetic Algorithms Cont'd

- GAs generate neighbors by crossover, mutation and natural selection



Genetic Algorithms Cont'd

- GAs generate neighbors by crossover, mutation and natural selection
- Crossover helps iff substrings are meaningful components
- GAs $\neq$ evolution: e.g., real genes encode replication machinery