## Kinds of Algorithms

<table>
<thead>
<tr>
<th>Speed</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>Short and sweet</td>
</tr>
<tr>
<td>slow</td>
<td>Slowly but surely</td>
</tr>
<tr>
<td></td>
<td>approximate</td>
</tr>
<tr>
<td></td>
<td>Quick and dirty</td>
</tr>
<tr>
<td></td>
<td>Too little, too late</td>
</tr>
</tbody>
</table>
The Minimum Spanning Tree Problem

Given: graph of “cities” and “distances”

Problem: Make all cities reachable from each other with minimum road construction.
Algorithms

Solution

<table>
<thead>
<tr>
<th>Speed</th>
<th>exact</th>
<th>approximate</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>Short and sweet</td>
<td>Quick and dirty</td>
</tr>
<tr>
<td>slow</td>
<td>Slowly but surely</td>
<td>Too little, too late</td>
</tr>
</tbody>
</table>
The Non-Attacking Queens Problem

- Put N queens on an N x N chessboard such that no queen attacks another queen

A queen in chess can attack any square on the same row, column, or diagonal. Given an N x N chessboard, we want to place N queens onto squares of the chessboard, such that no queen attacks another queen.

This example: placement (red squares) of N = 4 mutually non-attacking queens.

How many placements are possible?

How can we go through them systematically?
Backtrack in DFS of Decision Tree
Algorithms

Solution

<table>
<thead>
<tr>
<th>Speed</th>
<th>fastest</th>
<th>exact</th>
<th>approximate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Short and sweet</td>
<td></td>
<td>Quick and dirty</td>
</tr>
<tr>
<td>slow</td>
<td>Slowly but surely</td>
<td></td>
<td>Too little, too late</td>
</tr>
</tbody>
</table>
From **Spanning Trees** to **Steiner Trees**

- The **red dot** is called a **Steiner point** (= an “intermediate junction”)
- The maximum cost savings from adding Steiner points
  = \( \min \text{ ratio of } (\min \text{ Steiner tree cost}) / (\min \text{ spanning tree cost}) \)
  depends on the **metric**, or **distance function**
  - Cf. “Manhattan”, “Euclidean”, “Chebyshev”, …

Adapted from Prof. G. Robins, UVA
Minimum Steiner Tree is a **Hard** Problem

*(Minimum Spanning Tree was Easy)*

• How would you (greedily, iteratively) approach finding a low-cost Steiner tree?

Adapted from Prof. G. Robins, UVA
Iterated 1-Steiner Algorithm (1990)

Given a pointset $S$, what point $p$ minimizes $\text{MST}(S \cup \{p\})$

Algorithmic idea: Iterate! (greedily)

Theorem: within $\frac{4}{3}$ of OPT for “difficult” pointsets

In practice: solution cost is within $0.5\%$ of OPT on average


Adapted from Prof. G. Robins, UVA
“Light” AND “Shallow” Trees (1991)

What is the Minimum Spanning Tree?

Adapted from Prof. G. Robins, UVA
“Light” AND “Shallow” Trees (1991)

What is the Shortest Path Tree (with center point as source)?

Adapted from Prof. G. Robins, UVA
“Light” AND “Shallow” Trees (1991)

What is the Minimum Spanning Tree?
What is the Shortest Path Tree (with center point as source)?

What tree is both “light” and “shallow”?

1991 Algorithm:
Input:
• points, or graph
• $\varepsilon > 0$
Output: tree with
• radius $\leq (1 + \varepsilon) \cdot \text{OPT}$
• cost $\leq (1 + 2/\varepsilon) \cdot \text{OPT}$

See: http://vlsicad.ucsd.edu/Publications/Journals/j3.pdf
Adapted from Prof. G. Robins, UVA
“Light” AND “Shallow” Trees (1991)

Adapted from Prof. G. Robins, UVA
K-Center Problem

Given a planar pointset S, find a set of k “centers” C, such that the maximum distance of any point of S from its nearest center in C is minimized.

(We devised a greedy algorithm for this at the start of Lecture 8 on Greed. Your HW asks you to prove an approximation ratio for this greedy algorithm.)
List of Provably Good Approximations

- 1-Steiner Tree \((4/3)\)
- Bounded-Radius-Bounded-Cost Tree
  \(1 + \varepsilon \) radius, \(1 + 2/\varepsilon\) cost
- Engineer’s Method for Number Partitioning \((7/6)\)
- Next-Fit Bin Packing \((2)\) (Monday discussion)
- Christofides’ Euclidean TSP \((3/2)\)
- Greedy “farthest-first” k-center \((2)\)
“Metaheuristic”
Metaheuristic

- Method for dealing with large, practical, intractable optimization problems in the real world
  - Planning and scheduling
  - Routing and flows
  - Assignment and placement
  - Clustering and classification

- Intractable problems
  - = NP-hard
  - = instance complexity (“n”) too large to handle with known efficient algorithms
Outline

• Iterative Global Optimization
• Simulated Annealing
• Tabu Search
• Genetic Algorithms
Iterative Global Optimization

• **S = universe of solutions**  // aka “solution space”
  – n! TSP tours over n cities
  – $C(n, n/2)$ graph bisections
  – $k^n$ assignments of k colors to graph vertices

• **cost or objective function**
  – Cost of TSP tour
  – Cutsizes (# edges cut) in graph bisection
  – #colors needed for valid graph coloring

• **N(s) = neighborhood of a given solution s ∈ S**
  – Interchange positions of two cities in tour $\rightarrow C(n,2)$ neighbors
  – Swap two vertices between partitions $\rightarrow n^2/4$ neighbors
  – Change the color of a vertex $\rightarrow (k - 1) \cdot n$ neighbors
Iterative Global Optimization

- $S = \text{universe of solutions}$  // aka “solution space”
- cost or objective function
- $N(s) = \text{neighborhood of a given solution } s \in S$

- **Iterative Global Optimization**

start with an initial solution $s_0$

for $i = 1$ to $M$  // $M = \text{time limit, stop criterion, etc.}$

generate candidate solution $s \in N(s_{i-1})$

decide between $s_i = s_{i-1}$ or $s_i = s$

return $s_M$  // “where you are” == $s_M$

// “best so far” == best over $s_0$, ..., $s_M$
Simulated Annealing (SA)

- **Kirkpatrick, Gelatt, Vecchi, Science (1983):** *One of the 10 most cited scientific papers ever*

- SA is one of many “metaheuristics” that are used to deal with instances of intractable (NP-hard) combinatorial problems
  - Genetic algorithms (Holland, U. Michigan)
  - Tabu search (Glover, U. Colorado)
  - Etc.

- Combinatorial optimization has a physical analogy to the annealing (slow cooling) of metals to produce a perfectly-ordered, minimum-energy state: a “state” is a “solution”, “energy” is “cost”, etc.
Simulated Annealing Basic Idea

- **Initialize** – Start with a random initial solution. Initialize high “temperature”.
- **Step 2: “Move”** – Perturb current solution to obtain a ‘neighbor’ solution
- **Step 3: Calculate cost change** – calculate the change in solution cost due to the move (minimization: negative change is better, positive change is worse)
- **Step 4: Accept/Reject** – Depending on the cost change, accept or reject the move. Probability of acceptance depends on current “temperature”.
- **Step 5: Update** – Update temperature, current solution. Go to Step 2.
- Continue until termination condition (‘freezing’ or ‘quenching’) is satisfied
Algorithm SIMULATED-ANNEALING

Begin

\textit{temp} = INIT-TEMP;
\textit{currentSol} = INIT-SOLUTION;

\textbf{for} i = 1 to M

\textit{candidateSol} = NEIGHBOR\textsf{(currentSol)};
\Delta C = \text{COST}\textsf{(candidateSol)} - \text{COST}\textsf{(currentSol)};
\textbf{if} (\Delta C < 0) \textbf{then}

\textit{currentSol} = \textit{candidateSol};

\textbf{else with Pr = } e^{-\left(\Delta C/\text{temp}\right)}

\textit{currentSol} = \textit{candidateSol};

\textit{temp} = \text{SCHEDULE\textsf{(temp)};}

End

What happens when \textit{temp} = +\infty ?
What happens when \textit{temp} = 0 ?
Simulated Annealing Facts

- **Fact 1.** NEIGHBOR(solution) defines a topology over all solutions in the solution space.
- **Fact 2.** At a fixed value of \( temp \), SA behavior corresponds to a *homogeneous Markov chain*:
  - Fixed \( temp \) \( \rightarrow \) fixed matrix of transition probabilities between states.

Greed gets stuck here, in a local optimum.

SA converges to global opt solution with \( Pr = 1 \)
(in limit of infinite time, infinitely slow cooling)
Fact 3. The steady-state (= equilibrium) probability of the Markov chain being in state A is proportional to $e^{(-\text{cost}(A)/\text{temp})}$

- When temp $\to 0$, exponentially more likely to be in the global optimum state
- “SA is optimal” (in the limit of ‘infinite time’)
- Of course, we spend only a finite amount of time (#moves) at any temperature value
- Is cooling the best strategy with finite time? See Boese/Kahng, 1993
Optimal SA Temperature Schedules

- 6-city Traveling Salesman instance
- $M = 160$ steps
- “Where-You-Are” (top)
- “Best-So-Far” (bottom)
Optimal SA Temperature Schedules

- 8-vertex Graph Bisection instance
- $M = 160$ steps
- “Where-You-Are” (top)
- “Best-So-Far” (bottom)
Useful Idea: “Large-Step Markov Chain”

• (1) Run with Temp = 0 to find a local minimum
• (2) Run (briefly) with Temp = \( \infty \) as a “kick move” to escape the local minimum
• Alternate (1) and (2) until move budget is expended

• Cf. “optimal” (WYA) schedule on previous slide
• http://vlsicad.ucsd.edu/Publications/Journals/j29.pdf
Outline

• Iterative Global Optimization
• Simulated Annealing
• Tabu Search
• Genetic Algorithms
Tabu Search (Glover, 1986)

• **What**
  – Neighborhood search + memory
  • Neighborhood search
  • **Memory**
    – Record the search history – the “tabu list”
    – Forbid cycling search

[Source: Gang Quan, Van Laarhoven, Aarts, http://web.cecs.pdx.edu/~mperkows/CLASS_574/574-fall-08/0001.SimulatedAnnealingAndTabuSearch.ppt]
Tabu Search Algorithm

• (1) Choose initial solution $s_0$
• (2) Find best $s' \in N(s_i)$ that is not on tabu list
• (3) If $F(s') > F(s_i)$ \hspace{1cm} // $x'$ is better than $x$
  – $s_{i+1} = s'$
  – update tabu list
• (4) Go to step (2)

Note: This is Iterative Global Optimization!

[Source: Gang Quan, Van Laarhoven, Aarts, http://web.cecs.pdx.edu/~mperkows/CLASS_574/574-fall-08/0001_SimulatedAnnealingAndTabuSearch.ppt]
Tabu Search Tuning Parameters

• Local search procedure
• Neighborhood structure
• Criteria for tabu moves
• Update (and maximum size) of tabu list
  – Smaller list: “intensification” of search
  – Larger list: “diversification” of search
• Aspiration conditions
  – “aspiration” allows a tabu move to be used if it is sufficiently helpful
• Stopping condition/rule

Lots of parameters to tune!

[Source: Lecture slides from Lei Li, HongRui Liu, Roberto Lu, http://www.cs.ucla.edu/~rosen/161/TabuSearch.ppt]
Useful Idea: “Go With The Winners”

• (1) Run iterative global optimization on K processors
  == population of K “walkers”
• (2) Every so often, identify the K’ << K best solutions
  == “winners”
• (3) Replicate the K’ solutions onto the K processors
  == “go with the winners”
• (4) Return to (1)

Useful with parallel/distributed compute resources
Outline

• Iterative Global Optimization
• Simulated Annealing
• Tabu Search
• Genetic Algorithms
Genetic Algorithms

• A genetic algorithm (or GA) uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination)

• (1) Given a population at Generation i of solution representations (encodings)
• (2) Evaluate individuals’ fitness according to solution attributes
• (3) Propagate individuals’ attributes to next Generation i + 1 via selection (according to fitness), mutation, crossover, inversion, etc. operators

[Source: Muhannad Harrim, Western Michigan University]  
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt
Evolutionary Biology Metaphor

• **Individual** - Any possible solution
• **Population** - Group of all *individuals*
• **Chromosome** - Blueprint for an *individual*
• **Genome** - Collection of all *chromosomes* for an *individual*

• **Trait** - Possible aspect (*features*) of an *individual*
• **Allele** - Possible settings of trait (blue, brown, etc.)
• **Locus** - The position of a *gene* on the *chromosome*

[Source: Muhannad Harrim, Western Michigan University]
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt
Chromosome, Genes and Genomes

[Source: Muhannad Harrim, Western Michigan University]
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt
GA Pseudocode

Choose initial population

Evaluate the fitness of each individual in the population

Repeat

   Select best-ranking individuals to reproduce

   Breed new generation through crossover and mutation (genetic operations) and give birth to offspring

   Evaluate the individual fitnesses of the offspring

   Replace worst-ranked part of population with offspring

Until <terminating condition>

[Source: Muhannad Harrim, Western Michigan University]
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt
Representation

Chromosomes could be:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

[Source: Muhannad Harrim, Western Michigan University]
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt
A Fitness Function

[Source: Muhannad Harrim, Western Michigan University]
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt
Crossover and Mutation Operators

[Source: Muhannad Harrim, Western Michigan University]
https://www.cs.wmich.edu/elise/courses/cs6800/Genetic-Algorithms.ppt