15.093: Optimization Methods

Lecture 15: Heuristic Methods

1 Outline

SLIDE 1

- Approximation algorithms
- Local search methods
- Simulated annealing

2 Approximation algorithms

SLIDE 2

- Algorithm H is an ϵ-**approximation** algorithm for a minimization problem with optimal cost Z^*, if H runs in polynomial time, and returns a feasible solution with cost Z_{H}:

$$Z_{\mathrm{H}} \leq (1+\epsilon)Z^*$$

- For a maximization problem

$$Z_{\mathrm{H}} \geq (1-\epsilon)Z^*$$

2.1 TSP

2.1.1 MST-heuristic

SLIDE 3

- Triangle inequality

$$c_{ij} \leq c_{ik} + c_{kj}, \qquad \forall\ i, k, j$$

- Find a minimum spanning tree with cost Z_T
- Construct a closed walk that starts at some node, visits all nodes, returns to the original node, and never uses an arc outside the minimal spanning tree
- Each arc of the spanning tree is used exactly twice

SLIDE 4

- Total cost of this walk is $2Z_T$
- Because of triangle inequality $Z_{\mathrm{H}} \leq 2Z_T$
- But $Z_T \leq Z^*$, hence

$$Z_{\mathrm{H}} \leq 2Z_T \leq 2Z^*$$

1-approximation algorithm

2.1.2 Matching heuristic

SLIDE 5

- Find a minimum spanning tree. Let Z_T be its cost
- Find the set of odd degree nodes. There is an even number of them. Why?
- Find the minimum matching among those nodes with cost Z_M
- Adding spanning tree and minimum matching creates a Eulerian graph, i.e., each node has even degree. Construct a closed walk
- Performance

$$Z_{\mathrm{H}} \leq Z_T + Z_M \leq Z^* + 1/2Z^* = 3/2Z^*$$

SLIDE 6

3 Local search methods

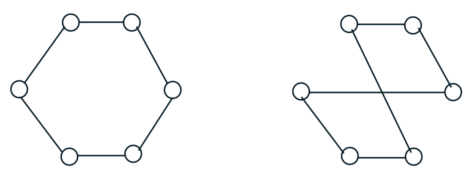
SLIDE 7

- Local Search: replaces current solution with a better solution by slight modification (searching in some neighbourhood) until a local optimal solution is obtained
- Recall the Simplex method

3.1 TSP-2OPT

SLIDE 8

- Two tours are neighbours if one can be obtained from the other by removing two edges and introducing two new edges



- Each tour has $O(n^2)$ neighbours. Search for better solution among its neighbourhood.

SLIDE 9

- Performance of 2-OPT on random Euclidean instances

-

Size N	100	1000	10000	100000	1000000
Matching	9.5	9.7	9.9	9.9	-
2OPT	4.5	4.9	5	4.9	4.9

3.2 Extensions

4 Extensions

Slide 10

- Iterated Local Search
- Large neighbourhoods (example 3-OPT)
- Simulated Annealing
- Tabu Search
- Genetic Algorithms

4.1 Large Neighbourhoods

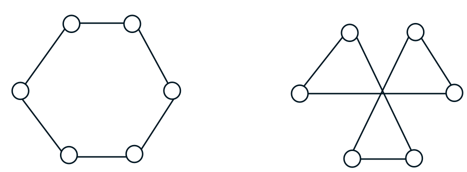
Slide 11

- Within a small neighbourhood, the solution may be locally optimal. Maybe by looking at a **bigger** neighbourhood, we can find a better solution.
- Increase in computational complexity

4.1.1 TSP Again

Slide 12

3-OPT: Two tours are neighbour if one can be obtained from the other by removing three edges and introducing three new edges



3-OPT improves on 2-OPT performance, with corresponding increase in execution time. Improvement from 4-OPT turns out to be not that substantial compared to 3-OPT.

5 Simulated Annealing

Slide 13

- Allow the possibility of moving to an inferior solution, to avoid being trapped at local optimum
- Idea: Use of randomization

5.1 Algorithm

SLIDE 14

- Starting at x, select a random neighbour y in the neighbourhood structure with probability q_{xy}

$$q_{xy} \geq 0, \quad \sum_{y \in \mathcal{N}(x)} q_{xy} = 1$$

- Move to y if $c(y) \leq c(x)$.
- If $c(y) > c(x)$, move to y with probability

$$e^{-(c(y)-c(x))/T},$$

  stay in x otherwise
- T is a positive constant, called temperature

5.2 Convergence

SLIDE 15

- We define a Markov chain.
- Under natural conditions, the long run probability of finding the chain at state x is given by

$$\frac{e^{-c(x)/T}}{A}$$

  with $A = \sum_z e^{-c(z)/T}$
- If $T \to 0$, then almost all of the steady state probability is concentrated on states at which $c(x)$ is minimum
- But if T is too small, it takes longer to escape from local optimal (accept an inferior move with probability $e^{-(c(y)-c(x))/T}$). Hence it takes much longer for the markov chain to converge to the steady state distribution

5.3 Cooling schedules

SLIDE 16

- $T(t) = R/\log(t)$. Convergence guaranteed, but known to be slow empirically.
- Exponential Schedule: $T(t) = T(0)a^n$ with $a < 1$ and very close to 1 (a=0.95 or 0.99) commonly used.

5.4 Knapsack Problem

SLIDE 17

$$\max \sum_{i=1}^{n} c_i x_i : \sum_{i=1}^{n} a_i x_i \leq b, \quad x_i \in \{0,1\}$$

Let $X = (x_1, ..., x_n) \in \{0,1\}^n$

- Neighbourhood Structure: $\mathcal{N}(X) = \{Y \in \{0,1\}^n : d(X,Y) = 1\}$. Exactly one entry has been changed

SLIDE 18

Generate random $Y = (y_1, ..., y_n)$:

- Choose j uniformly from $1, 2, ..., n$.
- $y_i = x_i$ if $i \neq j$. $y_j = 1 - x_j$.
- Accept if $\sum_i a_i y_i \leq b$.

5.4.1 Example

SLIDE 19

- c=(135, 139, 149, 150, 156, 163, 173, 184, 192, 201, 210, 214, 221, 229, 240)
- a=(70, 73, 77, 80,82, 87, 90,94, 98, 106, 110, 113, 115, 118, 120)
- $b = 750$
- $X^* = (1,0,1,0,1,0,1,1,1,0,0,0,0,1,1)$, with value 1458

SLIDE 20

Cooling Schedule:

- $T_0 = 1000$
- probability of accepting a downward move is between 0.787 ($c_i = 240$) and 0.874 ($c_i = 135$).
- Cooling Schedule: $T(t) = \alpha T(t-1)$, $\alpha = 0.999$
- Number of iterations: 1000, 5000

SLIDE 21

Performance:

- 1000 iterations: best solutions obtained in 10 runs vary from 1441 to 1454
- 5000 iterations: best solutions obtained in 10 runs vary from 1448 to 1456.

15.093J / 6.255J Optimization Methods
Fall 2009