5CCS2FC2: Foundations of Computing II

Approximation Algorithms & the Travelling Salesman Problem

Week 7

Dr Christopher Hampson

Department of Informatics

King's College London

Optimisation Problems and Approximation Algorithms

Decision vs Optimisation Problems

Decision Problems

Decide whether a solution *exists* or not.

VS

Optimisation Problems

Identify the *best* / *optimal* solution.

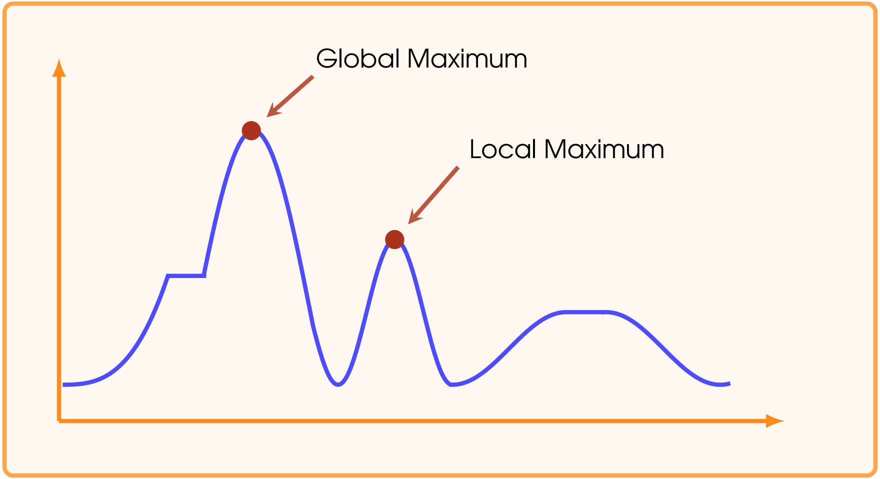
- **Examples**
  - The Boolean Satisfiability problem
  - The Halting Problem
  - The Hamiltonian Cycle Problem
  - The Graph Isomorphism Problem, *etc.*

- **Examples**
  - The Clique Problem
  - The Vertex Cover problem
  - The Travelling Salesman Problem, *etc.*

Optimisation Problems

- **Optimisation Problems**
  - **Solution Space** $S = \{ \text{ set of all possible solutions } \}$
  - **Cost function** $h(\text{solution}) \;=\; \text{value of solution}$
    - **Global Maximum** $h(s_{\text{global}}) \;\geq h(s) \text{ for all } s \in S$
    - **Local Maximum** $h(s_{\text{global}}) \;\geq h(s) \text{ for all } \textit{'neighbouring'}\ s \in S$

Optimisation Problems



The Travelling Salesman Problem

The Travelling Salesman Problem (TSP)

The Travelling Salesman (Decision) Problem TSP$_{dec}$

Input) — A *complete* weighted graph $M = (V, d)$,

where $d : V \times V \to \mathbb{Q}$ *is a distance function*

— A value $D \in \mathbb{Q}$

Output) **True** if and only if G contains a *Hamiltonian Cycle* H such that length_of $(H) \leq D$

The Travelling Salesman Problem (TSP)

The Travelling Salesman Problem TSP

Input) A *complete* weighted graph $G = (V, d)$,

where $d : V \times V \to \mathbb{Q}$ *is a distance function*

Output) The **shortest** Hamiltonian cycle, visiting all vertices.

$$S \;=\; \{\text{ all possible routes } H \;\}$$

$$h(H) \;=\; \text{length\_of}\,(H) \;=\; \sum_{(x,y)\in H} d(x,y)$$

The Travelling Salesman Problem (TSP)

Theorem The Travelling Salesman Probelm $\mathbf{TSP}_{dec}$ is NP-complete.

The Travelling Salesman Problem (TSP)

Theorem The optimisation version **TSP** can be solved in polynomial time if and only if the decision version **TSP**$_{dec}$ can be solved in polynomial time.

Proof:

TSP-DEC (G, D) :

1. Use algorithm to find optimal route H
2. **If** length_of $(H) \leq D$ **Return** Route H
3. **Else Return False**

The Travelling Salesman Problem (TSP)

TSP (G) :

1. $D^- \leftarrow 0$
2. $D^+ \leftarrow$ total weight of all edges of G
3. **While** $(D^+ - D^-) <$ shortest edge of G
    4. **mid** $\leftarrow (D^+ + D^-)/2$
    5. **If** TSP-DEC$(G, m) = H$ **Then** $D^+ \leftarrow$ **mid**
    6. **Else** $D^- \leftarrow$ **mid**
7. **Return** Route H of length **mid**

Q.E.D.

Approximation Ratio

Approximation Ratio

- **The Approximation Ratio**
  - The **approximation ratio** for a given approximation algorithm M on input w is the real value $R_M(w) \geq 1$ such that

$$\boxed{R_M(w) \;=\; \max\left(\frac{C_{\textbf{approx}}}{C_{\textbf{global}}}, \frac{C_{\textbf{global}}}{C_{\textbf{approx}}}\right)}$$

    - $C_{\textbf{approx}}$ is the cost of the ***approximate* solution**
    - $C_{\textbf{global}}$ is the cost of the ***optimal* solution**

Approximation Ratio

- **The Approximation Ratio**
  - The **approximation ratio** for a given approximation algorithm M on input w is the real value $R_M(w) \geq 1$ such that

$$\boxed{R_M(w) \;=\; \begin{cases} C_{\text{approx}}/C_{\text{global}} & \text{if } C_{\text{global}} \leq C_{\text{approx}} \\ C_{\text{global}}/C_{\text{approx}} & \text{if } C_{\text{global}} > C_{\text{approx}} \end{cases}}$$

    - C_{approx} is the cost of the ***approximate* solution**
    - C_{global} is the cost of the ***optimal* solution**

Approximation Ratio

- R-approximable problems
  - A language or problem L is said to be R-approximable if there is some polynomial-time approximation algorithm M such that

$$\boxed{R_M(w) \;\leq\; R \qquad \text{for all } w \in \Sigma^*}$$

$$\boxed{C_{\textsf{approx}} \;\leq\; R \times C_{\textsf{global}}}$$

Approximation Ratio

- **Unapproximable problems**
  - A language or problem L is said to be **unapproximable** if there L is not R-approximable for *any* fixed value $R > 0$

> Any polynomial-time approximation algorithm M must give **arbitrarily bad** solutions for some inputs.

End of Slides!