Coping with NP-Hardness

- Strategies to deal with hard problems
- Algorithms for Special Cases
- Fixed Parameter Tractability
- Intelligent Exhaustive Search
  - Backtracking
  - Branch and Bound
- Dynamic Programming based pseudo polynomial algorithm TSP

Imdad ullah Khan

Dynamic Programming: Review

- More general and powerful than divide and conquer
- Break up a problem into (in)(dependent) sub-problems
- Generally there is a sequence of problems
- Identify the optimal substructure: when optimal solution to a problem is made up of optimal solution to smaller subproblems
- Build up solution to larger and larger subproblems
- Identify redundancy and repetitions
- Use memoization or build up memo on the run

Dynamic Programming Formulation for TSP

Traveling Salesman Problem TSP(G) Given a complete graph G on n vertices with edge weights, find a minimum cost Hamiltonian cycle in G

Need ordering of subproblems ▷ Bellman (1962) and Held&Karp (1962)

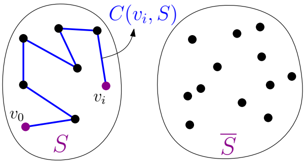
Let the vertex set of G be $V = \{v_0, v_1, \cdots, v_{n-1}\}$

WLOG assume the "start vertex" of the cycle is always v_0

Begin by constructing some sub path of a cycle starting form v_0

For $S \subset V$,

$C(v_i, S)$: the min cost path from $v_0 \in S$ to $v_i \in S$ that visits all and only vertices in S once

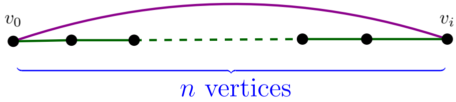


Dynamic Programming Formulation for TSP

For some $v_i \neq v_0$, $C(v_i, V)$ is a Hamiltonian path in G

▷ A lightest Hamiltonian path among those with v_0 as an endpoint

Adding the edge (v_i, v_0) to $C(v_i, V)$ gives a Hamiltonian cycle in G



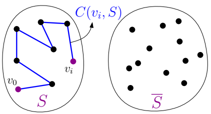
1. Initially, $S = \{v_0\}$ and $C(v_0, S)$ is the empty path with cost 0
2. Gradually increase S to get a Ham path in G

   ▷ Note: for $S = \{v_0\}$ and $i > 0$, $C(v_i, S)$ is not defined
3. Analyze the structure of the path $C(v_i, S)$ ▷ without knowing it

Dynamic Programming Formulation for TSP

For $S \subset V$,

$C(v_i, S)$: the min cost path from $v_0 \in S$ to $v_i \in S$ that visits all and only vertices in S once



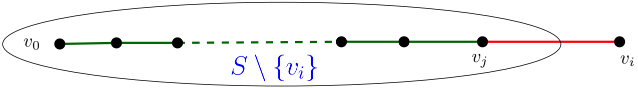
Analyze the structure of the path $C(v_i, S)$ ▷ without knowing it

Let $v_j \in S$ be the second to last vertex in $C(v_i, S)$

$$C(v_j, S \setminus \{v_i\}) \;=\; C(v_i, S) \setminus \{(v_j, v_i)\}$$

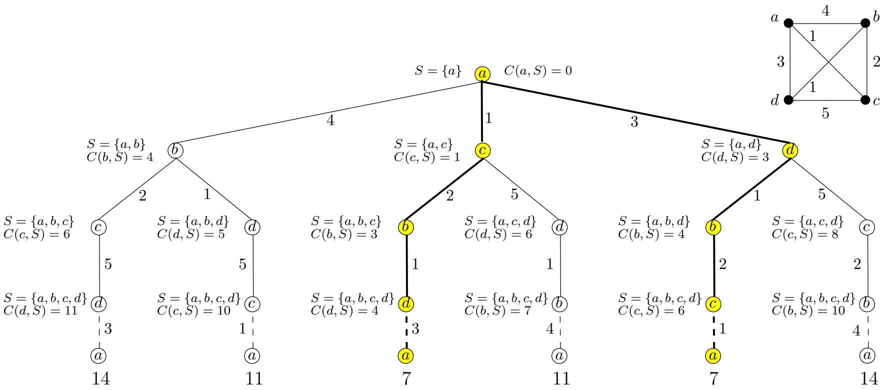
▷ min cost path from v_0 to v_j must be this subpath

because otherwise $C(v_i, S)$ would not be optimal



$C(v_j, S \setminus \{v_i\}) \cup \{(v_j, v_i)\}$ cannot be shorter than $C(v_i, S)$

Dynamic Programming Formulation for TSP: Example



Dynamic Programming Formulation for TSP

Let $c(v_i, S)$ be the weight of $C(v_i, S)$

Recurrence Relation for $c(v_i, S)$

$$c(v_i, S) = \begin{cases} 0 & \text{if } S = \{v_0\} \\ +\infty & \text{else if } v_i \notin S \vee i = 0 \\ \min\limits_{v_{j \neq i} \in S} \{c(v_j, S \setminus \{v_i\}) + w(v_j, v_i)\} & \text{else} \end{cases}$$

$$\text{VALUE-TSP}(G) \ = \ \min_{v_i \in V} \Big\{ c(v_i, V) + w(v_i, v_0) \Big\}$$

Dynamic Programming Formulation for TSP

$$\text{VALUE-TSP}(G) \;=\; \min_{v_i \in V} \Big\{ c(v_i, V) + w(v_i, v_0) \Big\}$$

- $2^n - 1$ possible $S \subset V$ and up to $n-1$ options for the end-vertex v_i
- Each of the $n \times 2^n$ sub-problems can be solved in $\mathcal{O}(n)$
- Runtime of DP solution $\mathcal{O}(n^2 2^n)$ i.e. $< \mathcal{O}(n!)$ of brute force solution
- What about space complexity? $\mathcal{O}(n2^n)$ i.e. $> \mathcal{O}(n^2)$ of brute force
- Actual Hamiltonian cycle can be found by backtracking in $\mathcal{O}(n^2)$
- If previous vertex in subpath (selected v_j with min cost) is stored for each step in DP, then backtracking can be done in $\mathcal{O}(n)$

Dynamic Programming Formulation for TSP

$$\text{VALUE-TSP}(G) \;=\; \min_{v_i \in V} \left\{ c(v_i, V) + w(v_i, v_0) \right\}$$

Runtime of DP solution $\mathcal{O}(n^2 2^n)$ i.e. $< \mathcal{O}(n!)$ of brute force solution