# Lecture Outline 1
# Topics in Graph Algorithms (CSCI5320-19S)

CAI Leizhen
Department of Computer Science and Engineering
The Chinese University of Hong Kong
lcai@cse.cuhk.edu.hk

January 9, 2019

**Keywords:** TSP, Clique, Vertex Cover, and FPT algorithm.

Graphs are effective models for numerous applications, and algorithmic problems on graphs arise naturally from various applications. Three key words in establishing mathematical models: *simplification, abstraction, and approximation.*

1. **Problem 1: 3D Printing**

   **Task:** For a given layer, determine nozzle movement to minimize the traveling distance of the nozzle.

   **Special case:** The layer consists of isolated dots each of minimum feature size.

   **Graph model**: Construct a weighted complete graph $G$ whose vertices correspond to dots and the weight of an edge indicates the distance between two dots.

   Our problem is exactly the NP-complete metric Traveling Salesman Problem on $G$ if we want the nozzle to return to its start position after finishing the layer (otherwise the problem is to find a Hamiltonian path of minimum weight).

   **2-Approximation algorithm**

   *Find a MST $T$ in $G$, walk around the tree to get a closed walk $W$, and then convert $W$ into a Hamiltonian cycle $C$ by taking short cuts to next unvisited vertices.*

   Note that edge weights of metric TSP satisfy *triangle inequality*: for any three vertices $x, y$ and $z$, $w(xy) \le w(xz)+w(yz)$. It follows that $w(C) \le w(W) = 2w(T)$ as $w(W) = 2w(T)$. On the other hand, any Hamiltonian cycle $H$ of $G$ contains a spanning path $P$ which is a spanning tree. Therefore $w(H) > w(P) \ge w(T)$, and hence $\text{opt}(G) > w(T)$. It follows that $w(C)/\text{opt}(G) < 2$.

   **1.5-Approximation algorithm**

   *We use a different way to form a closed walk $W$: add edges to $T$ to form an Eulerian graph $G^*$, and use an Eulerian cycle in $G^*$ as $W$.*

   For this purpose, we construct a weighted complete graph $G_T$ from $T$ by taking odd vertices of $T$ as vertices, and setting the weight of edge $uv$ in $G_T$ to be their

distance. We find a minimum-weight perfect matching $M$ from $G_T$ and add it to $T$ to form $G^*$. Note that $G^*$ may contain parallel edges.

Since $w(M) \leq \text{opt}(G)/2$, we have $w(C) \leq w(W) = w(T) + w(M) < \frac{3}{2}\text{opt}(G)$.

2. **Problem 2: Making Necklace**

   **Origin**: To make a perfect pearl necklace, one should use pearls as identical (size, color, shape, and texture) as possible.

   **Task:** Select $k$ pearls from $n$ pearls to make a perfect necklace.

   **Graph model**: Construct a weighted complete graph $G = (V, E; w)$ whose vertices are pearls, and the weight $w(xy)$ of edge $xy$ is the *similarity score* between pearls $x$ and $y$.

   Put a threshold $\epsilon$ for similarity scores: we can use pearls $x$ and $y$ together in a necklace if $w(xy) \geq \epsilon$.

   **New task:** Find a $k$ clique in the graph obtained from $G$ by deleting all edges with weight less than $\epsilon$.

3. **Problem 3: Density Reduction**

   **Origin**: Trees in a forest are too dense, and we need to cut down some trees.

   **Problem formation**: Given a point set $S$ in the plane, remove $k$ points from $S$ to maximize the minimum pairwise distance of the remaining points.

   **Graph model**: Construct a weighted complete graph $G = (V, E; w)$ whose vertices are points of $S$, and the weight $w(uv)$ of edge $uv$ is the distance between $u$ and $v$ in the plane.

   **General version**: Minimum Edge Weight

   *Input*: An edge-weighted graph $G$, integers $k$ and $w$.

   *Question*: Are there at most $k$ vertices $V'$ in $G$ such that the minimum edge weight in $G - V'$ is at least $w$?

   Note: Density Reduction is a special case of Minimum Edge Weight on weighted complete graphs.

   **Connection with** Vertex Cover

   Construct a graph $G'$ from $G$ by deleting all edges of $G$ with weight at least $w$.

   Then the answer for $G$ is yes iff $G'$ contains a vertex cover of size at most $k$.

   However, Vertex Cover is NP-complete and this reduction does not help us to solve Minimum Edge Weight.

   **Question**: Can you prove that Minimum Edge Weight is NP-complete?

4. Vertex Cover **for fixed** $k$

   Easily solved in time $O(n^k m)$ by exhaustive search, where $m$ and $n$, respectively, are numbers of edges and vertices of $G$.

   However this type of polynomial-time algorithms is normally meaningless in theory and useless in practice.

5. **FPT algorithm**

   An FPT (fixed-parameter tractable) algorithm measures its runing time in terms of the input size and a *parameter*, which is usually a part of input, and runs in time $O(f(k)n^c)$ for some computable function $f(k)$, where $n$ denotes the input size and $c$ is a constant independent of $k$ and $n$.

   Parameter $k$ tries to capture an aspect of a problem that most interests you, and FPT algorithms attempt to solve intractable problems such as NP-hard problems effectively in practice when parameter $k$ is "small".

   **Question**: Can you give an alternative definition of FPT algorithms that captures the key ideas of FPT algorithms?

6. **FPT algorithm for** Vertex Cover [Mehlhorn (1984) and Fellows (1988)]

   The algorithm uses the *bounded search tree* method based on the simple fact that for every edge $uv$, any $k$-vertex cover must contain either $u$ or $v$.

   We construct a binary tree $B$ of height at most $k$ as follows. Label the root of $B$ by $(G, \emptyset)$. Choose an edge $uv$ in the current graph $G$, branch out and label the two children of the root by $(G-u, u)$ and $(G-v, v)$ respectively. For each labeled node of $B$, label its two children in a similar way.

   Note that $G$ has a $k$-vertex cover $V'$ iff $B$ has a leaf $l$ with label $(\emptyset, x)$ for some vertex $x$, and $V'$ can be obtained from the labels of the nodes on the path from $l$ to the root.

   The above algorithm solves $k$-Vertex Cover in $O(2^k(m+n))$ time, which is effective for small $k$, say $k \leq 15$. The current fastest FPT algorithm for $k$-Vertex Cover runs in $O(1.28^k + kn)$ time (Chen, Kanj and Xia 2005), which makes Vertex Cover quickly solvable in practice for $n \leq 10^{15}$ and $k \leq 150$. Actually, the algorithm can handle $k \leq 400$ for most practical problems. Note that the exhaustive search algorithm for the problem can hardly handle a graph with $n = 100$ and $k = 10$.

7. Using FPT algorithms for $k$-Vertex Cover, we can obtain FPT algorithms for Minimum Edge Weight, and hence for Density Reduction. In fact, Cai and Zimmermann (2010) have obtained an $O(1.28^k + m)$-time algorithm for the problem.

8. For $k$-Vertex Cover, Cai (2017) used *indirect certificating* technique to obtain the following extremely simple algorithm, which finds a $k$-vertex cover in a graph $G$ with probability at least $4^{-k}$, if $G$ has one:

   Randomly mark each vertex with probability 1/2 and output $N(M)$.

   Note that $N(M)$ denotes the open neighbourhood of marked vertices, i.e., unmarked vertices that are adjacent to marked vertices.