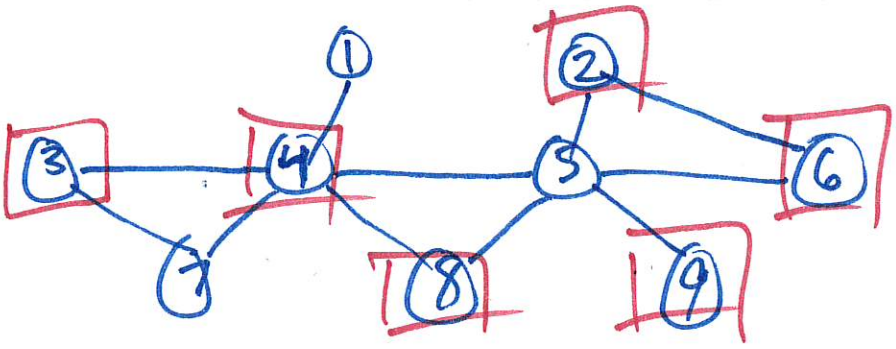
Introduction to Fixed-Parameter Tractability

A Motivating Application

A set of scientific experiments was conducted, resulting in $n = 100,000$ different sample data points. For whatever reason (e.g., measurement error or improper testing) the resulting data points are unreliable and contradict each other. For this reason, some of the results should be thrown out before conducting further analysis. Which results should be tossed? How should we find them?

To answer this, suppose that the contradictions are pairwise in nature and stored as a graph. That is, if data point i contradicts data point j, then we represent this contradiction by an edge $\{i, j\}$. Our task is to remove some of the data points so that the remaining data points are consistent with each other (i.e., do not contradict). Perhaps the best approach is to remove a *minimum* number of data points to achieve consistency. This follows Occam's razor and gives the most parsimonious explanation for the data inconsistencies.



possible to toss $k=6$ data points

We suppose that relatively few of the data points are bad, say $k \ll n$. (Otherwise, the entire data set is unreliable, and we should conduct the experiments all over again.) We would like a reasonably fast algorithm to solve instances like this.

If $k = 10$, how long would the brute-force algorithm take?

$$\binom{n}{k} m = O(n^k m)$$

$$k = 10 \Rightarrow O(n^{10} m)$$

$$\binom{100{,}000}{10} = 2.8 \times 10^{43}$$

What would be nice is a fixed-parameter tractable algorithm.

Definition 1. *An algorithm is fixed-parameter tractable (fpt) if its runtime is bounded by $f(k)n^{O(1)}$, where n is the input size, k is the parameter of choice, and f is a function that depends only on k.*

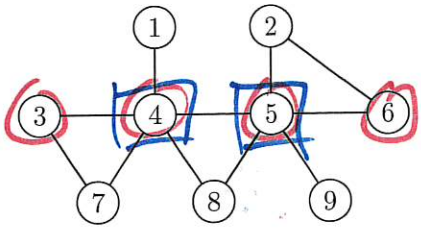
e.g., n^2, kn^2, $2^k n^3$, $(k)^{k!} n^{100}$

Class XP $\rightarrow n^k$ ← NOT

The (Parameterized) Vertex Cover Problem

Definition 2. *A subset $S \subseteq V$ of vertices is called a vertex cover for graph $G = (V, E)$ if, for every edge $\{u, v\} \in E$, either $u \in S$ or $v \in S$ or both.*

Problem: VERTEX COVER.
Input: A simple graph $G = (V, E)$ and a positive integer k.
Parameter: A positive integer k.
Question: Does G have a vertex cover of size k?

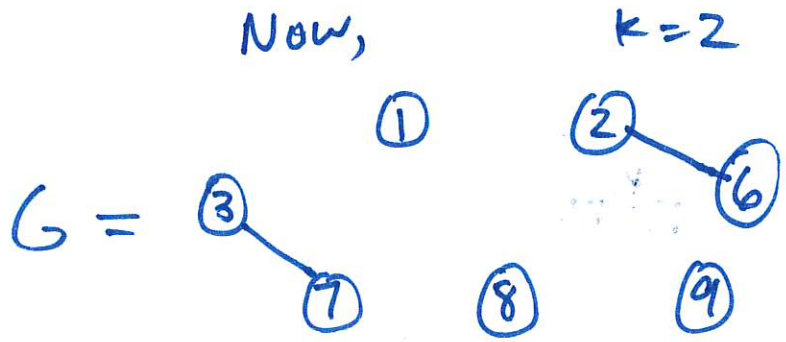


Does this graph G have a vertex cover of size $k = 4$, i.e., is (G, k) a "yes" instance?

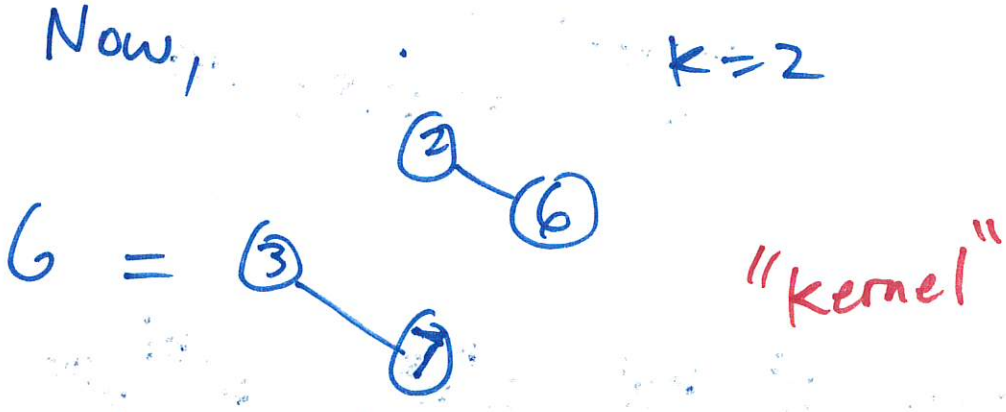
yes

Preprocessing Tricks

Lemma 1. *If G has a vertex v with $\deg(v) > k$, then replace (G, k) with $(G - v, k - 1)$.*



Lemma 2. *If G has an isolated vertex v, then replace (G, k) with $(G - v, k)$.*



The Buss Kernel

Repeatedly applying the preprocessing tricks leads to the Buss *kernelization* procedure:

kn — 1. let $S := \{v \in V(G) \mid \deg_G(v) > k\}$; — $S = \{4, 5\}$

n — 2. if $|S| > k$, return "no"; otherwise let $k' := k - |S|$. — $k' = k - |S| = 4 - 2 = 2$

O(kn) — 3. let $G' := G - S \cup I$, where I is the set of isolated vertices in $G - S$;

4. if $|E(G')| > kk'$, return "no";

k^2 — 5. else return the *kernel* (G', k').

Proposition 1. *The kernel (G', k') can be found in time $O(kn)$ and has at most k^2 edges and at most $2k^2$ vertices.*

$$|E(G')| \leq k(k') \leq k(k)$$

$$|V(G')| \leq 2|E(G')| \leq 2k^2$$

Proposition 2. *The k-vertex cover problem can be solved in time $O(kn + 2^k k^{2k+2})$.*

Alg: 1. find Buss kernel

2. Solve kernel w/ Brute Force

$$\binom{n'}{k'} m' = O((n')^{k'} m')$$

$$= O((2k^2)^k (k^2))$$

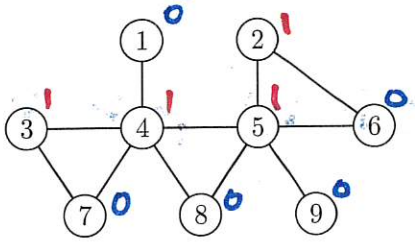
$$= O(2^k k^{2k+2})$$

The Nemhauser-Trotter Kernel

Here is an LP relaxation for the vertex cover problem.

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ & x_i + x_j \geq 1 \qquad \{i,j\} \in E \\ & 0 \leq x_v \leq 1 \qquad v \in V. \end{aligned}$$

Solve LP to get



Theorem 1 (Nemhauser and Trotter, 1975). *The LP above admits an optimal solution $x^* \in \mathbb{R}^n$ such that $x^* \in \{0, \frac{1}{2}, 1\}^n$. Moreover, there is a minimum vertex cover S of G such that*

$$V_1 \subseteq S \subseteq V_1 \cup V_{\frac{1}{2}}.$$

where $V_i = \{v \in V \mid x_v^* = i\}$ for $i \in \{0, \frac{1}{2}, 1\}$.

Here, $V_0 = \{1, 6, 7, 8, 9\}$

$V_{1/2} = \emptyset$

$V_1 = \{2, 3, 4, 5\}$

$V_1 \subseteq S \subseteq V_1 \cup \emptyset$

so $S = V_1$

Proposition 3. *An optimal solution $x^* \in \{0, \frac{1}{2}, 1\}^n$ to the LP can be found in time $O(m\sqrt{n})$.*

Follows by transformation to bipartite matching & Hopcroft-Karp.

Theorem 1 leads to the Nemhauser-Trotter kernel:

1. compute an optimal solution $x^* \in \{0, \frac{1}{2}, 1\}^n$ to the vertex cover LP;
2. if the LP objective is greater than k, return "no";
3. let $G' := G - V_0 - V_1$ and $k' := k - |V_1|$;
4. return (G', k').

$m\sqrt{n}$ — (or kn?) —

LP obj $= \frac{1}{2}|V_{1/2}| + |V_1|$

Proposition 4. *The kernel (G', k') can be found in time $O(m\sqrt{n})$ and has at most $2k$ vertices.*

$n' = |V_{1/2}| \leq |V_{1/2}| + 2|V_1| = 2(LP) \leq 2k.$

Remark 1 (Chen et al., 2001). *The kernel (G', k') can be found in time $O(kn + k^3)$.*

$O(kn)$ 1. Apply Buss

2. Get N.T. kernel of Buss kernel

$O(m'\sqrt{n'}) = O(k^2\sqrt{2k^2}) = O(k^3)$

Proposition 5. *The k-vertex cover problem can be solved in time $O(kn + 2^k k^{k+2})$.*

$kn + k^3$ —

1. do remark 1
2. Brute force

$O((n')^{k'} m') = O((2k)^k k^2) = O(2^k k^{k+2})$

Kernelization, Generally

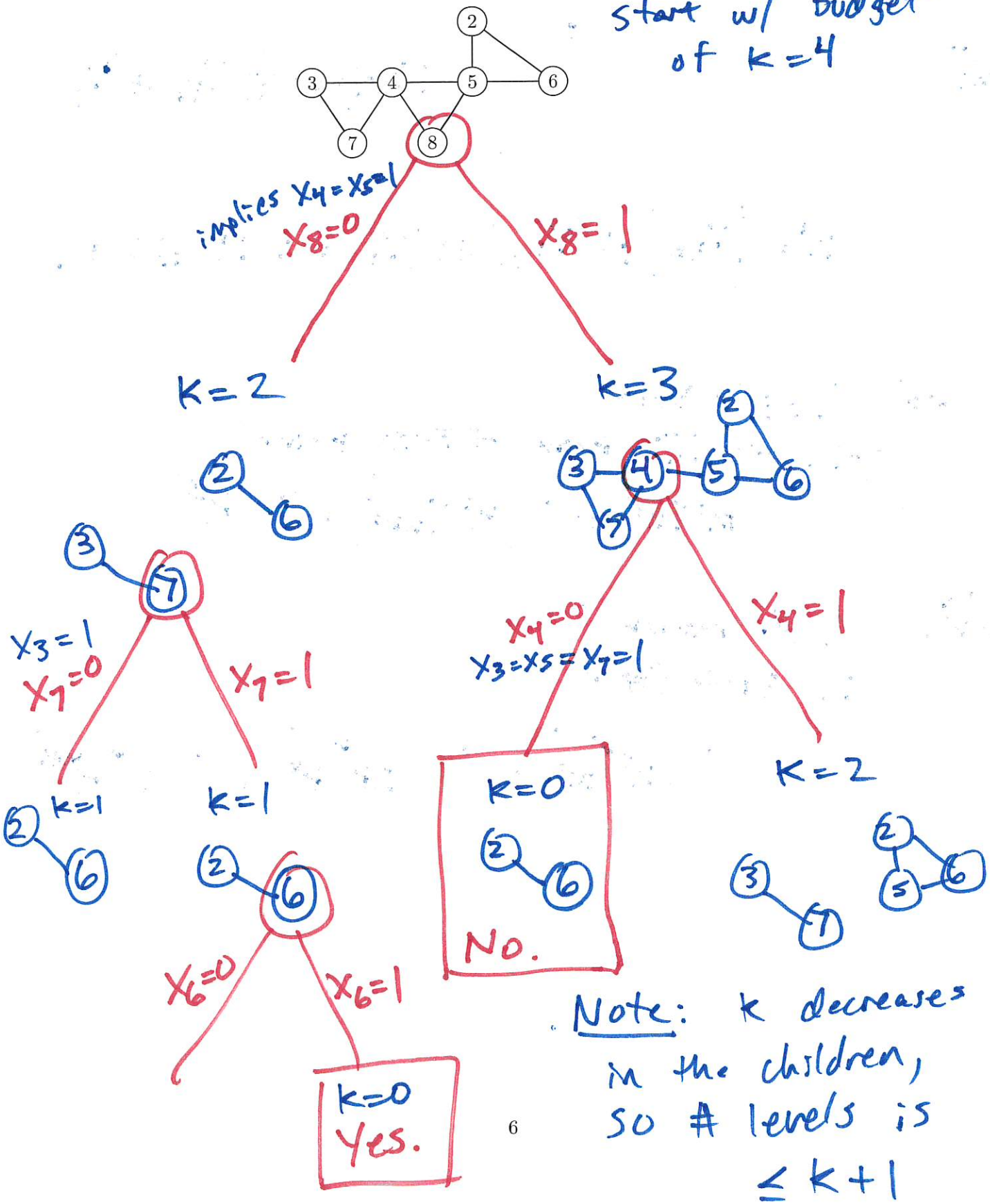
Definition 3 (Kernelization). *A kernelization is an algorithm that, when given an instance (G, k) of a parameterized decision problem, runs in polynomial time and returns another instance (G', k') such that:*

1. *(G, k) is a "yes" instance if and only if (G', k') is a "yes" instance;*
2. *the size of the instance (G', k') is bounded by some computable function of k.*

Notice that the second condition ensures that the kernel's size is bounded by a function that depends only on k; it does not depend on the input size G. Typically, $k' \leq k$, but not necessarily!

How to Avoid Brute Force: Branch!

For simplicity, suppose that vertices 1 and 9 did not exist.



Bounded Search Tree Algorithm

Here is recursive pseudocode for the bounded search tree (BST) algorithm.

vc(G, k) ← returns yes/no

1. if $k < 0$, return "no";
2. if $|E(G)| = 0$, return "yes";
3. pick a vertex $v \in V(G)$ with $|N_G(v)| \geq 1$;
4. return **vc**$(G - v, k - 1) \vee$ **vc**$(G - N[v], k - |N_G(v)|)$;

m+n →

$x_v = 1$ or $x_v = 0$

Proposition 6. *The BST algorithm solves the k-vertex cover problem in time* $O(2^k(n+m))$.

- at most $k+1$ levels in search tree
- so # nodes explored is at most:

$$2^0 + 2^1 + 2^2 + \cdots + 2^k = 2^{k+1} - 1$$

↑ root ↑ level k

- in each node perform $O(m+n)$ work.

Proposition 7. *The k-vertex cover problem can be solved in time* $O(2^k k^2 + kn)$.

1. Get N.T. kernel in time $O(kn + k^3)$
2. Solve kernel w/ B.S.T. in time:

$$O\left(2^{k'}(n'+m')\right) = O\left(2^k(2k + 2k^2)\right)$$
$$= O\left(2^k k^2\right)$$

Note:

$2^{10}(10)^2 + 10(100{,}000) = 1{,}102{,}400$

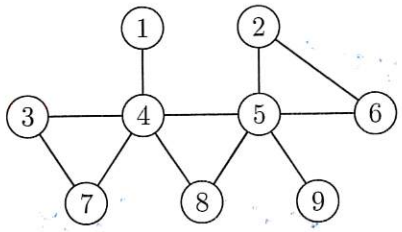
Theorem 2 (Chen et al., 2010). *The k-vertex cover problem can be solved in time* $O(1.2738^k + kn)$.

$1.2738^{10} \approx 12$

The (Parameterized) Dominating Set Problem

Definition 4. *A subset $S \subseteq V$ of vertices is called a dominating set for graph $G = (V, E)$ if, for every vertex $v \in V$, v or (at least) one of its neighbors belongs to S.*

Problem: DOMINATING SET.
Input: A simple graph $G = (V, E)$ and a positive integer k.
Parameter: A positive integer k.
Question: Does G have a dominating set of size k?



How long does brute force take for k-dominating set?

$\binom{n}{k} m$

The Strong Exponential Time Hypothesis (SETH) is an unproven complexity assumption.

Definition 5. *SETH is the assumption that for every $\varepsilon > 0$ there exists a k such that k-CNF-SAT cannot be solved in time $O((2 - \varepsilon)^n)$.*

Theorem 3 (Pătraşcu and Williams, 2010). *If SETH is true, then, for every $k \geq 3$ and every $\varepsilon > 0$, there is no algorithm that solves k-dominating set in time $O(n^{k-\varepsilon})$.*

Definition 6. *A parameterized problem belongs to class XP if instances with input size n and parameter k can be solved in time $n^{f(k)}$ for some computable function f.*

Theorem 4. *Determining whether a graph is 3-colorable is NP-hard.*

if $P \neq NP$ then k-coloring is not FPT and not in XP.