Better than exponential time algorithms for NP-complete problems if some parameter is small.

Example

Vertex Cover: Given graph $G = (V, E)$ and $k \in \mathbb{N}$, is there a vertex cover of size $\leq k$? Set of vertices hitting all edges

What if we use $k$ itself as a parameter?

$k = 1, 2, 3$ Try all possible $k$-sets $\binom{n}{k}$ which is $O(n^k)$

Checking a subset takes $O(n \cdot k)$ so total time is $O(k \cdot n^{k+1})$.

This is poly time for constant $k$

Some idea works for clique or ind. set

but not for graph colouring - given a graph, is it 3-colourable?

$O(n^k)$ is still pretty bad.

We went $O(f(k) \cdot n^c)$ for some $f$, $f(k)$ ind. of $n$ and some constant $c$ ind. of $k$.

or even $O(f(k) + n^c)$

We can achieve this for Vertex Cover

(but no one knows how to do it for Ind. Set)

For each edge $(u,v)$ we need $u$ or $v$ in the Vertex Cover.

We can do exhaustive search, branching on this choice

At each tree node, we pick an uncovered edge.

$|c| \leq k \Rightarrow$ stop at depth $k$
Alg. VC2 $(G, k)$ - return True or False
- if $E = \emptyset$ return True
- if $k = 0$ return False
- pick $e = (u, v)$ $\in E$
- return VC2 $(G - u, k - 1)$ or VC $(G - v, k - 1)$
  remove $u$ and incident edges

Time $O(2^k \cdot n)$
# internal tree nodes
work for each node

EX: find the vertex cover too.

We can improve to $O(f(k) + m + n)$
technique called "kernelization".

Claim If $G$ has a vertex $v$ with $\deg(v) > k$ then $v$ must be in $C$.
(C vertex cover of size $k$)

Pf $\{ u \subseteq \frac{\deg(v)}{k} \}$ if $u \not\subseteq C$, we need all $> k$ neighbours.

Alg. VC3 $(G, k)$ - return True/False
- $C' \leftarrow$ all vertices of $\deg > k$
- $k' \leftarrow k - |C'|$
- $G' \leftarrow G - (C' \text{ and all incident edges})$
  and remove isolated vertices
  - if $G'$ has $k^2 + k$ vertices return False
  - else return VC2 $(G', k')$

Note:
$G'$ is not very big:
max degree in $G'$ is $\leq k$
if $G'$ has VC of size $\leq k$
then $G'$ has $\leq k^2 + k$
vertices
the actual vertex cover is $C^1 \cup$ vertex cover of $G^1, k$

idea due to Prof. J. Buss 1993

Analysis: call to VC takes $O\left(2^k \cdot \left(\ast \text{ vertices of } G^1\right)\right)$

finding $C^1$ takes $O(m+n)$

total run time $O\left(2^k \cdot k^2 + m + n\right)$

This 3rd alg. is even practical sometimes (much better than other two)

Comparison

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>n = 10^4</th>
<th>k = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>brute force</td>
<td>$k \cdot n^{k+1}$</td>
<td>$10^45$</td>
<td>$\times$</td>
</tr>
<tr>
<td>better</td>
<td>$2^k n$</td>
<td>$10^9$</td>
<td></td>
</tr>
<tr>
<td>kernelized</td>
<td>$2^k k^2 + n + m$</td>
<td>$2 \cdot 10^5 + m$</td>
<td>possible</td>
</tr>
</tbody>
</table>

Defn A problem is **fixed parameter tractable** in parameter $k$
if it has an alg. with run time $O(f(k) \cdot n^c)$

where $f(k)$ is a fn of $k$ (ind. of $n$)

and $c$ is a constant (ind. of $k$).

Two ideas so far
- branching search of limited depth
- kernelization (preprocessing step to get small instance of the problem)
Examples of parameters
- value of opt. e.g. Ind. set of size \( k \)
- max degree of graph
- dimension for geometric problems
- genus of graph genus 0 = planar
  \( 1 = \) on torus

Some examples of FPT algs:
- vertex cover of size \( k \)
- simple path of length \( k \)
- finding \( k \) disjoint triangles in graph
- drawing graph in plane with \( k \) edge crossings
- finding disjoint paths connecting \( k \) pairs of nodes

Hardness - some parameterized problems are not known to have FPT algorithms and there are classes of "equivalently hard problems" - but no result like FPT \( \forall X \implies P = NP \)

The kernelization method generalizes

**Thm.** If a problem with parameter \( k \) is FPT i.e. there's an alg. with run time \( O(f(k) \cdot n^c) \)
then there exists an alg. with run time \( O(f'(k) + n^c) \)

But proof is not constructive. (It's not hard either)
k-path: Given graph $G$, $k \in \mathbb{N}$, vertices $s,t$, find a simple $s$-$t$ path with exactly $k$ internal vertices.

A shortest $s$-$t$ path will be simple.

But if $k$ is larger, this problem is NP-hard — it generalizes Hamiltonian Path.

**Example**

A randomized FPT algorithm

Colour vertices $V \setminus \{s,t\}$ randomly with $k$ colours

Test for **colour-ful** $s$-$t$ path — each colour appears exactly once.

We'll see how to test later.

Error analysis:

no simple $s$-$t$ path $\Rightarrow$ no colour-ful $s$-$t$ path $\Rightarrow$ alg. returns NO (for any colouring)

3 simple $s$-$t$ path $P$

$$\Pr[P \text{ is colour-ful}] \geq \frac{k!}{k^k} \geq \frac{(\frac{k}{e})^k}{k^k} = \frac{1}{e^k}$$

So alg. outputs YES (i.e. is correct) with prob. $\geq \frac{1}{e^k}$

If prob. success $\geq \frac{1}{e}$ then

prob. failure after $\frac{1}{e}$ repetitions is

$$(1-p)^{\frac{1}{e}} < (e^p)^{\frac{1}{e}} = \frac{1}{e}$$

$1-p + \frac{p^2}{2} \ldots$ Taylor expansion.

So in our case, repeating $e^k$ times gives prob. error $\leq \frac{1}{e}$.
Finding a colourful s-t path.

The colours on such a path are a permutation of 1..k.

Try all $k!$ permutations.

The problem is easy for fixed permutation.

Run Time $O(k! \cdot m)$, $m=$ # edges

Fact: Can be improved to $O(2^k \cdot m)$.

So we have a FPT alg. with small prob. of error, $\leq \frac{1}{e}$

and with run time $O(e^k 2^k \cdot m)$.

Fact: Can be de-randomized (using $k$-perfect hash functions).