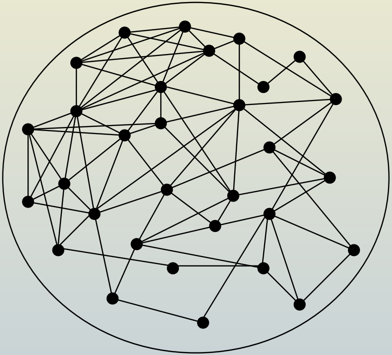
University of Glasgow

When can an efficient decision algorithm be used to find and count witnesses?

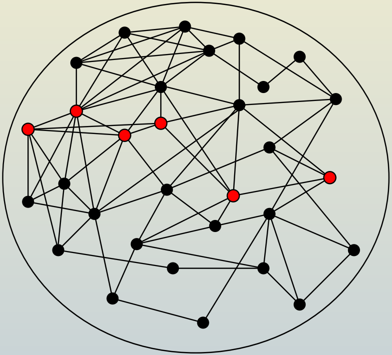
1st March 2016

Kitty Meeks

Given a graph on n vertices, we are interested in subgraphs with k vertices that have particular properties.



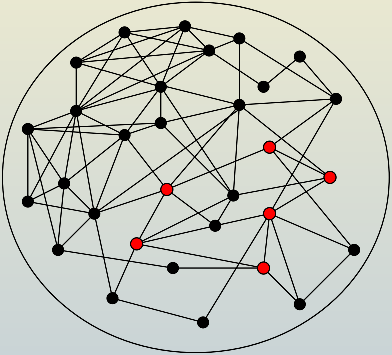
Given a graph on n vertices, we are interested in subgraphs with k vertices that have particular properties.



For example:

- Paths on k vertices

Given a graph on n vertices, we are interested in subgraphs with k vertices that have particular properties.

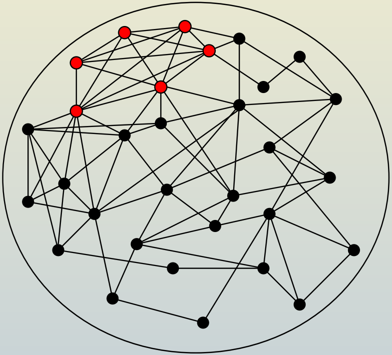


For example:

- Paths on k vertices
- Cycles on k vertices

The problem

Given a graph on n vertices, we are interested in subgraphs with k vertices that have particular properties.

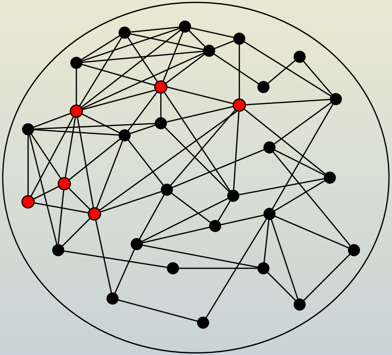


For example:

- Paths on k vertices
- Cycles on k vertices
- Cliques on k vertices

The problem

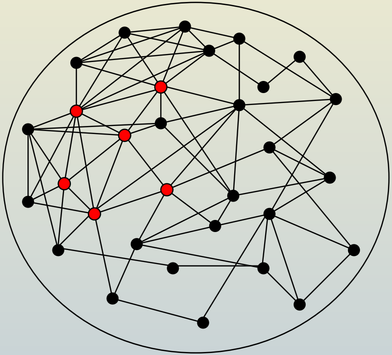
Given a graph on n vertices, we are interested in subgraphs with k vertices that have particular properties.



For example:

- Paths on k vertices
- Cycles on k vertices
- Cliques on k vertices
- Connected k-vertex induced subgraphs

Given a graph on n vertices, we are interested in subgraphs with k vertices that have particular properties.



For example:

- Paths on k vertices
- Cycles on k vertices
- Cliques on k vertices
- Connected k-vertex induced subgraphs
- k-vertex induced subgraphs with an even number of edges

DECISION
Is there a witness?

DECISION
Is there a witness?

APPROX COUNTING
Approximately how many witnesses?

Deciding, counting and enumerating

DECISION
Is there a witness?

APPROX COUNTING
Approximately how many witnesses?

EXACT COUNTING
Exactly how many witnesses?

Deciding, counting and enumerating

DECISION
Is there a witness?

EXTRACTION
Identify a single witness

APPROX COUNTING
Approximately how many witnesses?

EXACT COUNTING
Exactly how many witnesses?

Deciding, counting and enumerating

DECISION
Is there a witness?

EXTRACTION
Identify a single witness

APPROX COUNTING
Approximately how many witnesses?

UNIFORM SAMPLING
Pick a single witness uniformly at random

EXACT COUNTING
Exactly how many witnesses?

Deciding, counting and enumerating

DECISION Is there a witness?	**EXTRACTION** Identify a single witness
APPROX COUNTING Approximately how many witnesses?	**UNIFORM SAMPLING** Pick a single witness uniformly at random
EXACT COUNTING Exactly how many witnesses?	**ENUMERATION** List all witnesses

The values of n and k

Consider the k-cycle problem:

- if $k = 3$ then we are interested in triangles
- if $k = n$ then we are interested in Hamilton Cycles

The values of n and k

Consider the k-cycle problem:

- if $k = 3$ then we are interested in triangles
- if $k = n$ then we are interested in Hamilton Cycles

We are interested in what happens as n and k both tend to infinity, independently, with $k << n$.

The values of n and k

Consider the k-cycle problem:

- if $k = 3$ then we are interested in triangles
- if $k = n$ then we are interested in Hamilton Cycles

We are interested in what happens as n and k both tend to infinity, independently, with $k << n$.

- We can consider all possible k-vertex subgraphs in time $O(n^k)$.

The values of n and k

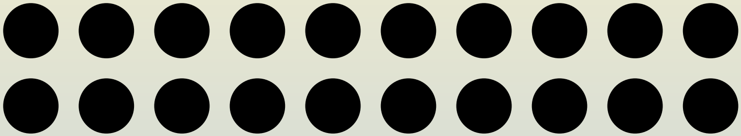
Consider the k-cycle problem:

- if $k = 3$ then we are interested in triangles
- if $k = n$ then we are interested in Hamilton Cycles

We are interested in what happens as n and k both tend to infinity, independently, with $k << n$.

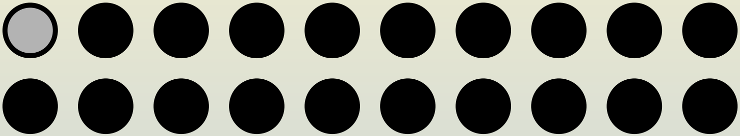
- We can consider all possible k-vertex subgraphs in time $O(n^k)$.
- We would like to be able to answer questions about k-vertex subgraphs in time $f(k) \cdot n^{O(1)}$.

If we can decide, we can find a witness



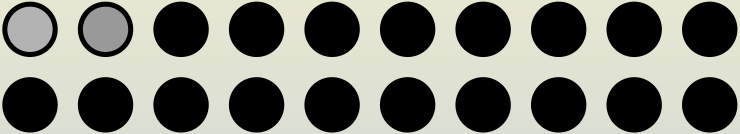
University of Glasgow

If we can decide, we can find a witness

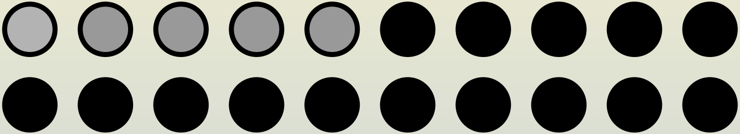


University of Glasgow

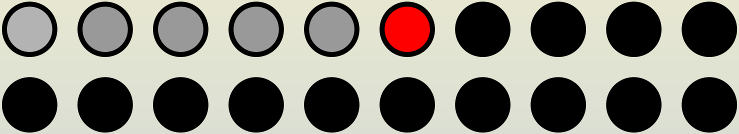
If we can decide, we can find a witness



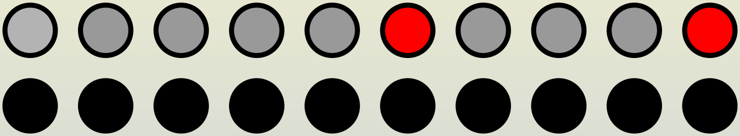
If we can decide, we can find a witness



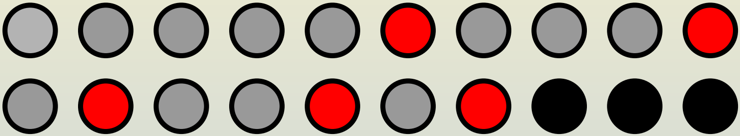
If we can decide, we can find a witness



If we can decide, we can find a witness



If we can decide, we can find a witness



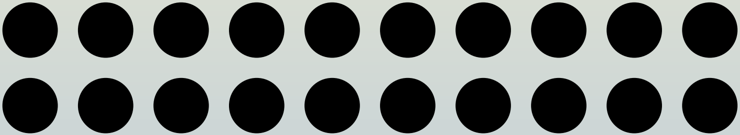
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



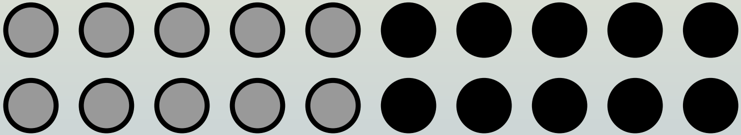
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



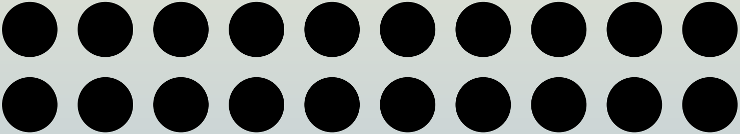
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.

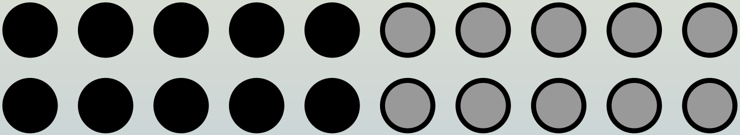


Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



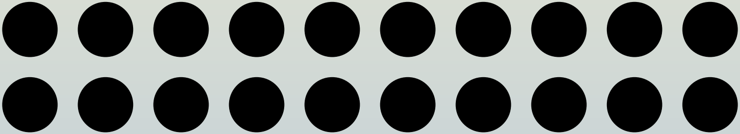
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



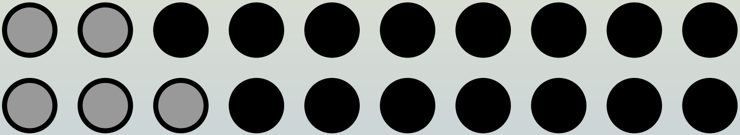
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



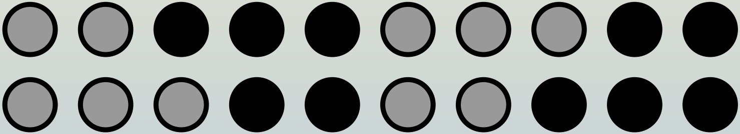
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



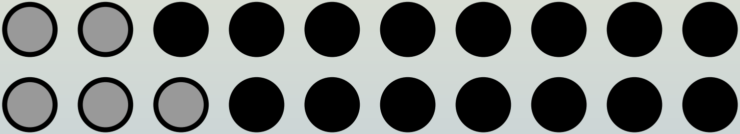
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



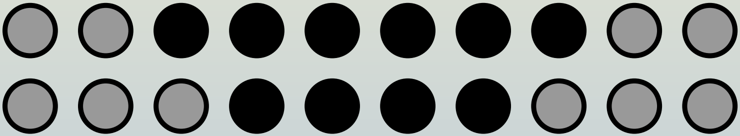
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



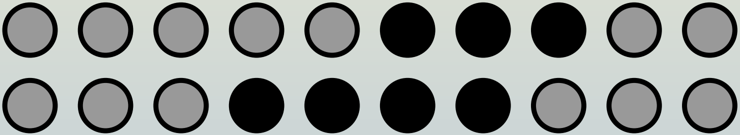
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



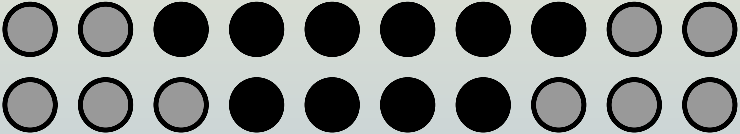
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



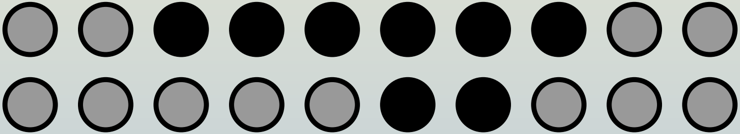
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



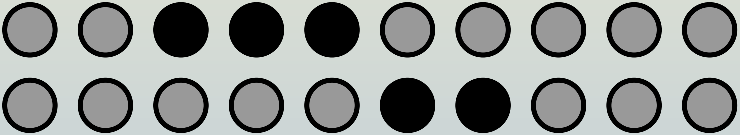
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



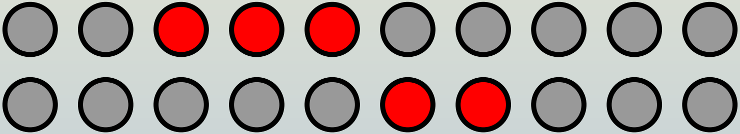
If we can decide, we can find a witness

Theorem (Björklund, Kaski and Kowalik, 2014)

There exists an algorithm that extracts a witness using at most

$$2k\left(\log_2 \frac{n}{k} + 2\right)$$

queries to a deterministic decision algorithm.



Deciding, counting and enumerating

DECISION Is there a witness?	**EXTRACTION** Identify a single witness
APPROX COUNTING Approximately how many witnesses?	**UNIFORM SAMPLING** Pick a single witness uniformly at random
EXACT COUNTING Exactly how many witnesses?	**ENUMERATION** List all witnesses

If we can count approximately, we can decide

If we can count approximately, we can decide

... at least with high probability.

An FPRAS for a counting problem Π is a randomised approximation scheme that takes an instance I of Π (with $|I| = n$), and numbers $\epsilon > 0$ and $0 < \delta < 1$, and in time $poly(n, 1/\epsilon, \log(1/\delta))$ outputs a rational number z such that

$$\mathbb{P}[(1-\epsilon)\Pi(I) \leq z \leq (1+\epsilon)\Pi(I)] \geq 1 - \delta.$$

If we can count approximately, we can decide

... at least with high probability.

An FPRAS for a counting problem Π is a randomised approximation scheme that takes an instance I of Π (with $|I| = n$), and numbers $\epsilon > 0$ and $0 < \delta < 1$, and in time $poly(n, 1/\epsilon, \log(1/\delta))$ outputs a rational number z such that

$$\mathbb{P}[(1-\epsilon)\Pi(I) \leq z \leq (1+\epsilon)\Pi(I)] \geq 1-\delta.$$

Set $\epsilon < \frac{1}{2}$, and we will distinguish between 0 and at least 1 with probability at least $1-\delta$.

Deciding, counting and enumerating

DECISION Is there a witness?	**EXTRACTION** Identify a single witness
APPROX COUNTING Approximately how many witnesses?	**UNIFORM SAMPLING** Pick a single witness uniformly at random
EXACT COUNTING Exactly how many witnesses?	**ENUMERATION** List all witnesses

Uniform sampling is harder than finding a witness

GENCYCLE
Input: A directed graph G.
Output: A cycle selected uniformly, at random, from the set of all directed cycles of G.

Theorem (Jerrum, Valiant, Vazirani, 1986)

Suppose there exists a polynomial time bounded Probabilistic Turing Machine which solves the problem GENCYCLE. *Then NP = RP.*

Deciding, counting and enumerating

DECISION
Is there a witness?

EXTRACTION
Identify a single witness

APPROX COUNTING
Approximately how many witnesses?

UNIFORM SAMPLING
Pick a single witness uniformly at random

EXACT COUNTING
Exactly how many witnesses?

ENUMERATION
List all witnesses

Self-reducibility

A relation $R \subseteq \Sigma^* \times \Sigma^*$ is *self-reducible* if and only if:

- there exists a polynomial time computable function $g \in \Sigma^* \to \mathbb{N}$ such that $xRy \implies |y| = g(x)$;
- there exist polynomial time computable functions $\psi \in \Sigma^* \times \Sigma^* \to \Sigma^*$ and $\sigma \in \Sigma^* \to \mathbb{N}$ satisfying:
  - $\sigma(x) = O(\log |x|)$
  - $g(x) > 0 \implies \sigma(x) > 0 \quad \forall x \in \Sigma^*$
  - $|\psi(x, w)| \leq |x| \quad \forall x, w \in \Sigma^*$,

  and such that, for all $x \in \Sigma^*$, $y = y_1 \ldots y_n \in \Sigma^*$,

$$\langle x, y_1 \ldots y_n \rangle \in R \iff \langle \psi(x, y_1 \ldots y_{\sigma(x)}), y_{\sigma(x)+1} \ldots y_n \rangle \in R.$$

Self-reducibility

A relation $R \subseteq \Sigma^* \times \Sigma^*$ is *self-reducible* if and only if:

- there exists a polynomial time computable function $g \in \Sigma^* \to \mathbb{N}$ such that $xRy \implies |y| = g(x)$;
- there exist polynomial time computable functions $\psi \in \Sigma^* \times \Sigma^* \to \Sigma^*$ and $\sigma \in \Sigma^* \to \mathbb{N}$ satisfying:
  - $\sigma(x) = O(\log |x|)$
  - $g(x) > 0 \implies \sigma(x) > 0 \quad \forall x \in \Sigma^*$
  - $|\psi(x, w)| \leq |x| \quad \forall x, w \in \Sigma^*$,

  and such that, for all $x \in \Sigma^*$, $y = y_1 \dots y_n \in \Sigma^*$,

$$\langle x, y_1 \dots y_n \rangle \in R \iff \langle \psi(x, y_1 \dots y_{\sigma(x)}), y_{\sigma(x)+1} \dots y_n \rangle \in R.$$

Theorem (Jerrum, Valiant, Vazirani, 1986)

For self-reducible problems, approximate counting and almost-uniform sampling are polynomial-time inter-reducible.

Self-reducibility and equivalent notions

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$. Then the following statements are equivalent.

Self-reducibility and equivalent notions

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$. Then the following statements are equivalent.

1. The problem is self-reducible.

Self-reducibility and equivalent notions

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$. Then the following statements are equivalent.

1. The problem is self-reducible.
2. We can decide whether there is at least one **multicoloured** witness in time $f(k) \cdot n^{O(1)}$.

Self-reducibility and equivalent notions

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$. Then the following statements are equivalent.

1. The problem is self-reducible.
2. We can decide whether there is at least one **multicoloured** witness in time $f(k) \cdot n^{O(1)}$.
3. We can decide whether there is at least one witness that is an **extension** of a given partial solution in time $f(k) \cdot n^{O(1)}$.

Deciding, counting and enumerating

DECISION Is there a witness?	**EXTRACTION** Identify a single witness
APPROX COUNTING Approximately how many witnesses?	**UNIFORM SAMPLING** Pick a single witness uniformly at random
EXACT COUNTING Exactly how many witnesses?	**ENUMERATION** List all witnesses

A special case

Theorem (Arvind and Raman (2002); Jerrum and M. (2015); M. (2016))

If there is an efficient ($f(k) \cdot n^{O(1)}$) algorithm for the decision version of a self-reducible subgraph problem, and adding edges cannot decrease the number of witnesses, then we can count witnesses approximately.

Many witnesses

Proposition

Suppose that, for each k and any graph G on n vertices, the number of k-vertex witnesses is either

1. *zero, or*
2. *at least*

$$\frac{1}{g(k)p(n)}\binom{n}{k}.$$

Then there is an FPTRAS to count witnesses.

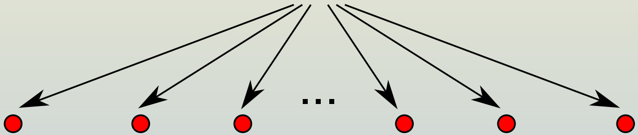
Theorem

Suppose that we have an $f(k) \cdot n^{O(1)}$ decision algorithm for the multicolour version of the problem. Then we can enumerate (and hence count) all witnesses in time $g(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses.

Few witnesses

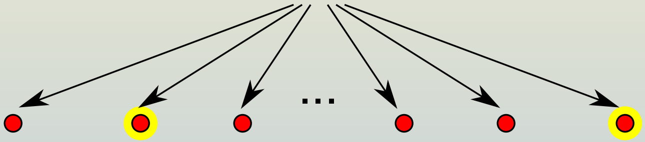
Theorem

Suppose that we have an $f(k) \cdot n^{O(1)}$ decision algorithm for the multicolour version of the problem. Then we can enumerate (and hence count) all witnesses in time $g(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses.



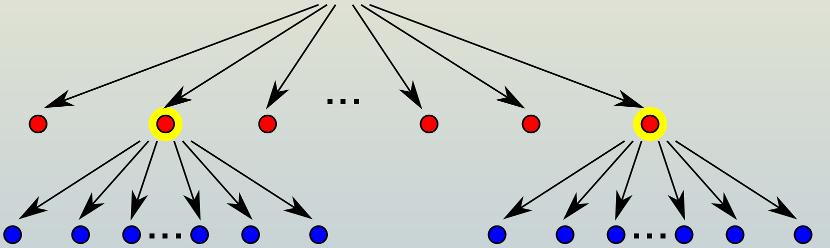
Theorem

Suppose that we have an $f(k) \cdot n^{O(1)}$ decision algorithm for the multicolour version of the problem. Then we can enumerate (and hence count) all witnesses in time $g(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses.



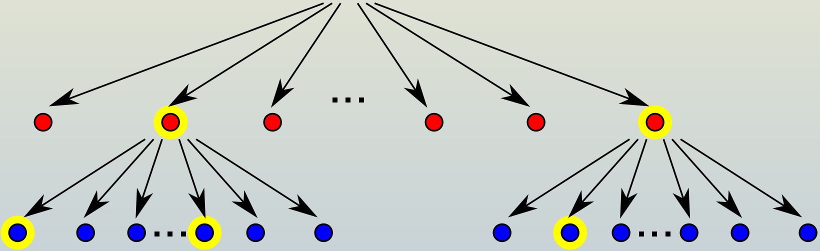
Theorem

Suppose that we have an $f(k) \cdot n^{O(1)}$ decision algorithm for the multicolour version of the problem. Then we can enumerate (and hence count) all witnesses in time $g(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses.



Theorem

Suppose that we have an $f(k) \cdot n^{O(1)}$ decision algorithm for the multicolour version of the problem. Then we can enumerate (and hence count) all witnesses in time $g(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses.



Not all subgraph problems are self-reducible

Suppose that a k-vertex subset is a witness if it either induces a clique or an independent set.

Not all subgraph problems are self-reducible

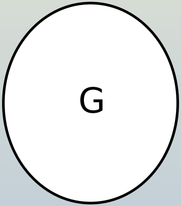
Suppose that a k-vertex subset is a witness if it either induces a clique or an independent set.

- The decision problem can be solved in time $f(k)$:
  - By Ramsey, for sufficiently large graphs the answer is always "yes".

Not all subgraph problems are self-reducible

Suppose that a k-vertex subset is a witness if it either induces a clique or an independent set.

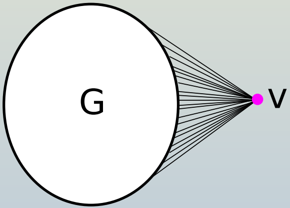
- The decision problem can be solved in time $f(k)$:
  - By Ramsey, for sufficiently large graphs the answer is always "yes".
- Unless the exponential time hypothesis fails, there is no $f(k) \cdot n^{o(1)}$ algorithm for the extension version:
  - Reduction from **p-**CLIQUE.



Not all subgraph problems are self-reducible

Suppose that a k-vertex subset is a witness if it either induces a clique or an independent set.

- The decision problem can be solved in time $f(k)$:
    - By Ramsey, for sufficiently large graphs the answer is always "yes".
- Unless the exponential time hypothesis fails, there is no $f(k) \cdot n^{o(1)}$ algorithm for the extension version:
    - Reduction from **p-**CLIQUE.



Theorem (Alon, Yuster, Zwick, 1995)

For all $n, k \in \mathbb{N}$ there is a k-perfect family $\mathcal{F}_{n,k}$ of hash functions from $[n]$ to $[k]$ of cardinality $2^{O(k)} \cdot \log n$. Furthermore, given n and k, a representation of the family $\mathcal{F}_{n,k}$ can be computed in time $2^{O(k)} \cdot n \log n$.

Theorem (Alon, Yuster, Zwick, 1995)

For all $n, k \in \mathbb{N}$ there is a k-perfect family $\mathcal{F}_{n,k}$ of hash functions from $[n]$ to $[k]$ of cardinality $2^{O(k)} \cdot \log n$. Furthermore, given n and k, a representation of the family $\mathcal{F}_{n,k}$ can be computed in time $2^{O(k)} \cdot n \log n$.

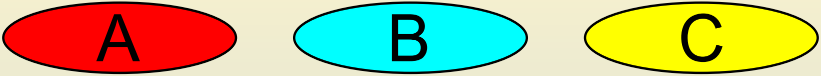
- IDEA: create many coloured instances, and enumerate the colourful copies in each (omitting duplicates)

Theorem (Alon, Yuster, Zwick, 1995)

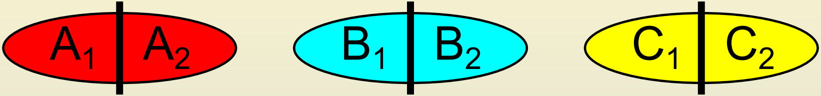
For all $n, k \in \mathbb{N}$ there is a k-perfect family $\mathcal{F}_{n,k}$ of hash functions from $[n]$ to $[k]$ of cardinality $2^{O(k)} \cdot \log n$. Furthermore, given n and k, a representation of the family $\mathcal{F}_{n,k}$ can be computed in time $2^{O(k)} \cdot n \log n$.

- IDEA: create many coloured instances, and enumerate the colourful copies in each (omitting duplicates)
- PROBLEM: although we're now looking for colourful witnesses, we still only have a decision algorithm for the uncoloured version...

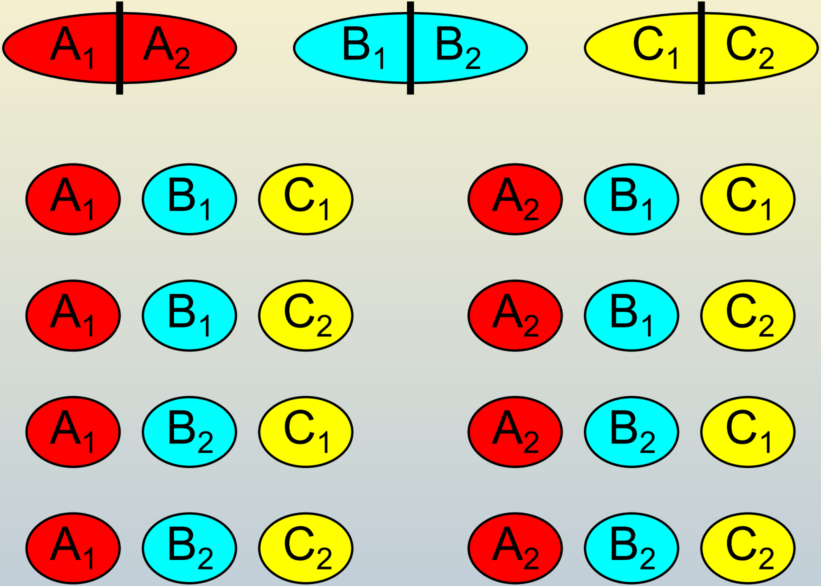
A randomised approach

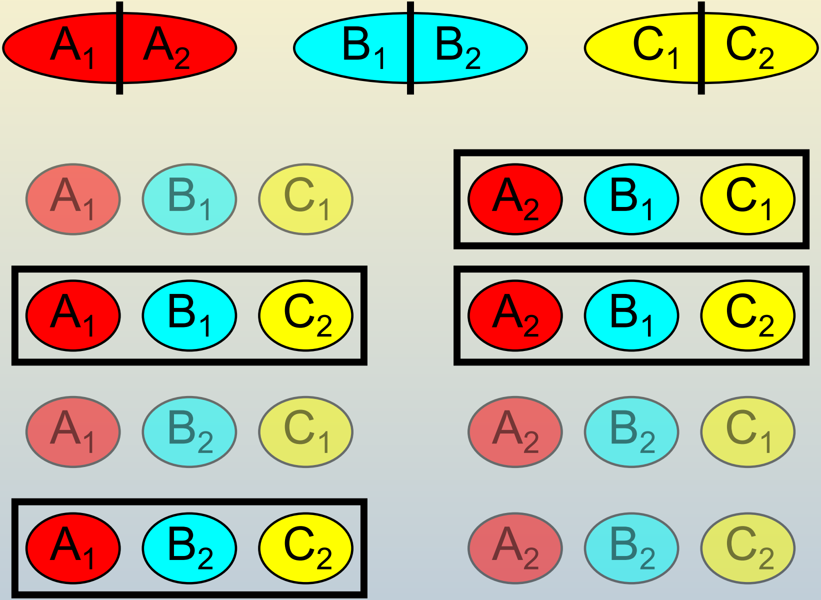


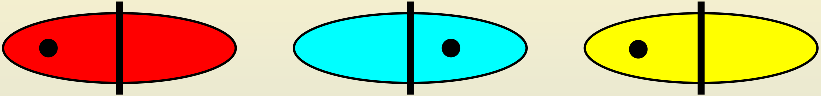
A randomised approach



A randomised approach



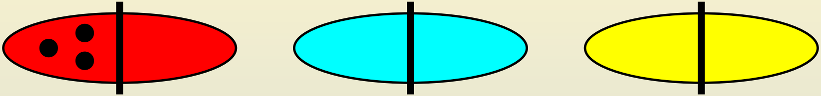

A1 A2
B1 B2
C1 C2
A1 B1 C1
A2 B1 C1
A1 B1 C2
A2 B1 C2
A1 B2 C1
A2 B2 C1
A1 B2 C2
A2 B2 C2



If a witness is colourful:

- It will always survive in exactly one combination

A randomised approach



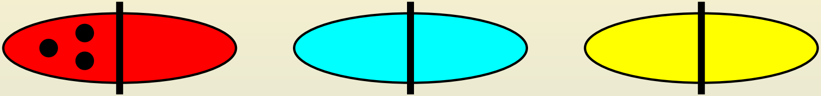
If a witness is colourful:

- It will always survive in exactly one combination

If a witness contains vertices of only $\ell < k$ colours:

- the probability it survives in at least one combination is at most $2^{-(k-\ell)}$
- if it survives in any combination, it will survive in exactly $2^{k-\ell}$ combinations

A randomised approach



If a witness is colourful:

- It will always survive in exactly one combination

If a witness contains vertices of only $\ell < k$ colours:

- the probability it survives in at least one combination is at most $2^{-(k-\ell)}$
- if it survives in any combination, it will survive in exactly $2^{k-\ell}$ combinations

It can then be shown that, for **any** witness, the **expected** number of combinations in which it survives at each level is at most one.

Few witnesses, revisited

Theorem

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$. Then there is a randomised algorithm which enumerates all witnesses in expected time $f(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses in the instance.

Few witnesses, revisited

Theorem

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$. Then there is a randomised algorithm which enumerates all witnesses in expected time $f(k) \cdot n^{O(1)} \cdot N$, where N is the total number of witnesses in the instance.

Corollary

Suppose that we can decide if there is at least one witness in time $f(k) \cdot n^{O(1)}$ and that, for each k and any graph G on n vertices, the total number of witnesses is at most $f(k)n^{O(1)}$. Then there exists an FPTRAS to count witnesses.

Open problems

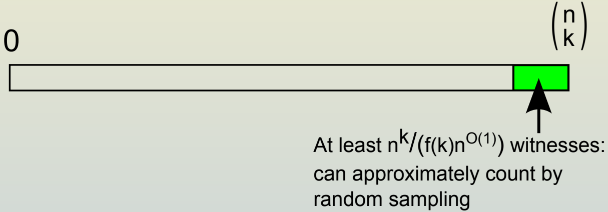
- Can the randomised enumeration process be derandomised?

Open problems

- Can the randomised enumeration process be derandomised?
- How common are non-self-reducible subgraph problems?

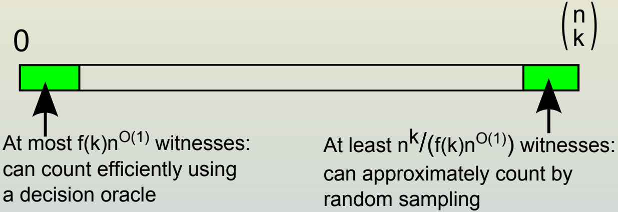
Open problems

- Can the randomised enumeration process be derandomised?
- How common are non-self-reducible subgraph problems?
- Can we close the gap?



Open problems

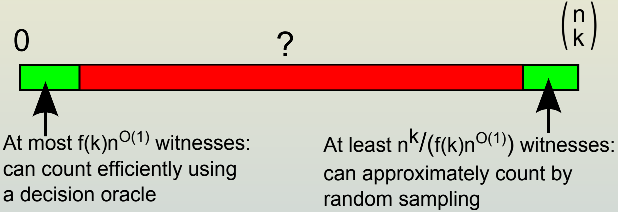
- Can the randomised enumeration process be derandomised?
- How common are non-self-reducible subgraph problems?
- Can we close the gap?



University of Glasgow

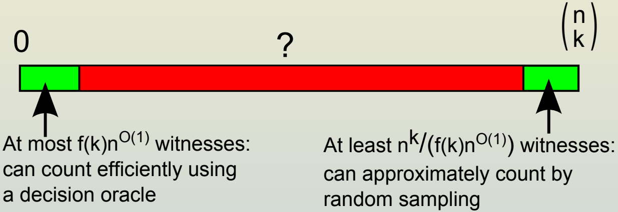
Open problems

- Can the randomised enumeration process be derandomised?
- How common are non-self-reducible subgraph problems?
- Can we close the gap?



Open problems

- Can the randomised enumeration process be derandomised?
- How common are non-self-reducible subgraph problems?
- Can we close the gap?



Thank you