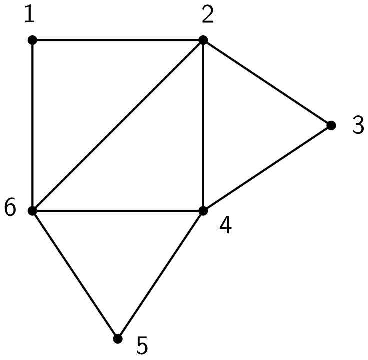
Enumeration algorithms in graphs

Aurélie Lagoutte
G-SCOP, Grenoble INP / Université Grenoble Alpes

G@g project – January 2026

Enumeration : a typical example

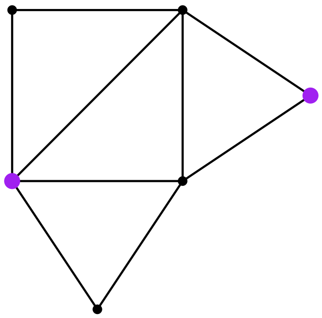
Input: Graph G
Output : The list of all **inclusion-wise maximal** stable sets of G



$\{1,3,5\},\{1,4\},\{2,5\},\{3,6\}$

Focus on easy problems

Input: Graph G
Output : one **inclusion-wise maximal** stable set. $\in$ P (greedy)



Not to be confused with :
Input: Graph G
Output : a stable set of maximum size

NP-complete

Enumerating in graphs : useful cases

- Graph databases : answer to a query
- Graph model is not exact : some solutions are *best* based on qualitative criteria, we have to examine them one by one
- Identify all problematic (or interesting !) patterns in a network

Application fields: bioinformatics (phylogenetic trees), chemistry (molecule structure), complex system modeling, databases...

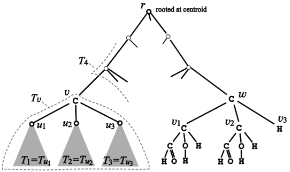


Diagram of a stereoisomer[1]

[1] Comparison and Enumeration of Chemical Graphs, T. Akutsu, H. Nagamochi, *Comp. and Struct. Biotechnology Journal*

Complexity for enumeration problems

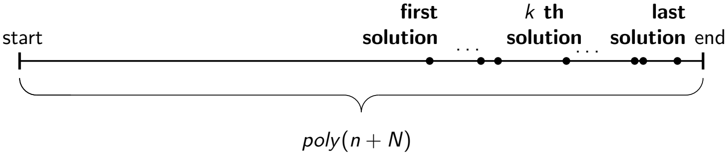
In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
⇒ *Good complexity measure?*

1. Output-polynomial

Input of size n, N solutions to output.

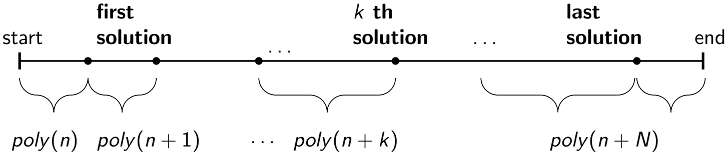


Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

1. Output-polynomial
2. Incremental polynomial

Input of size n, N solutions to output.

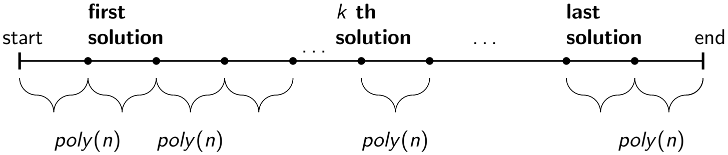


Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

1. Output-polynomial
2. Incremental polynomial
3. Polynomial delay

Input of size n, N solutions to output.



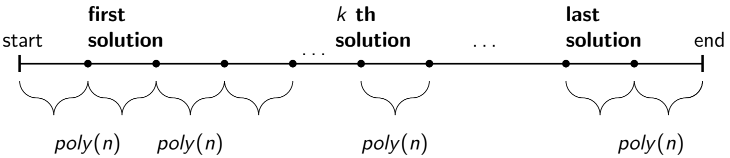
Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
⇒ *Good complexity measure?*

1. Output-polynomial
2. Incremental polynomial
3. Polynomial delay

poly space vs. exponential space

Input of size n, N solutions to output.



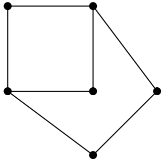
Interesting objects to enumerate

1. Inclusion-wise minimal transversal of a hypergraph
2. Inclusion-wise minimal dominating sets
3. Spanning trees
4. "Structured patterns" : inclusion-wise max. stable sets or cliques ...
5. **Inclusion-wise minimal "Π-fixings" of a graph**
   $\rightarrow$ Completions, deletion, induced subgraphs of a graph ...
   ... satisfying a given property Π

Minimal fixings

3 variants

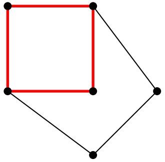
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



Minimal fixings

3 variants

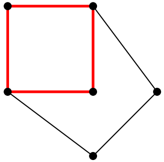
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



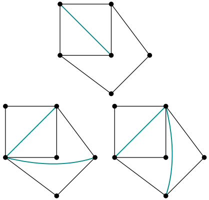
Minimal fixings

3 variants

We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



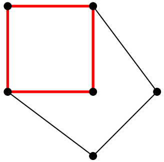
Fixing by adding edges
Min. Π-completion



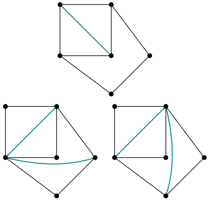
Minimal fixings

3 variants

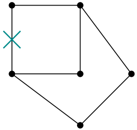
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



Fixing by
adding edges
Min. Π-completion



Fixing by
deleting edges
Min. Π-deletion

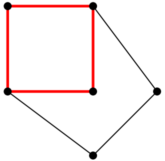


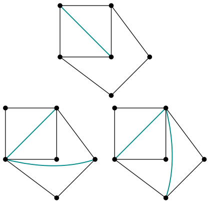
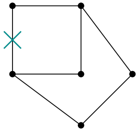
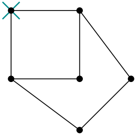
+ 3 others

Minimal fixings

3 variants

We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)

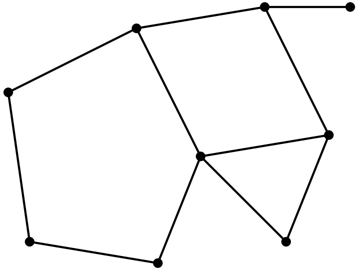


Fixing by adding edges	Fixing by deleting edges	Fixing by deleting vertices
Min. Π-completion	**Min. Π-deletion**	**Max. Π-induced subgraph**
	 + 3 others	 + 3 others

Chordal completion

Chordal completion of a graph G : a completion of G that is chordal (no chordless cycle of length ≥ 4).

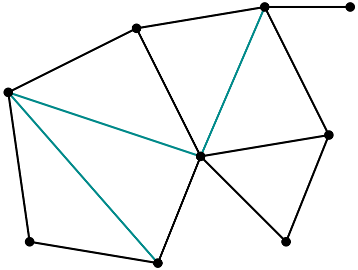
A chordal completion of G is also called a *triangulation* of G or sometimes a *fill-in*.



Chordal completion

Chordal completion of a graph G : a completion of G that is chordal (no chordless cycle of length ≥ 4).

A chordal completion of G is also called a *triangulation* of G or sometimes a *fill-in*.

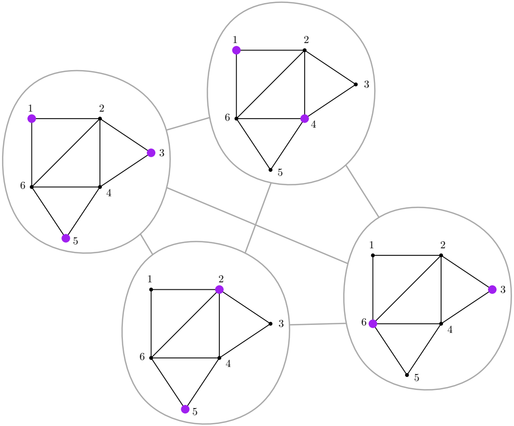


Algorithmic methods to enumerate

General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph

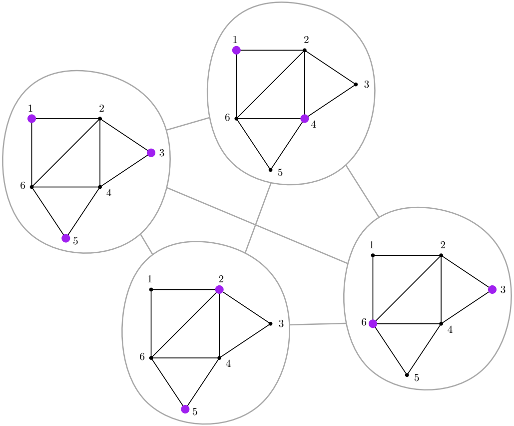
Example with
maximal stable sets
Solution metagraph



General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph
- outputting each solution once

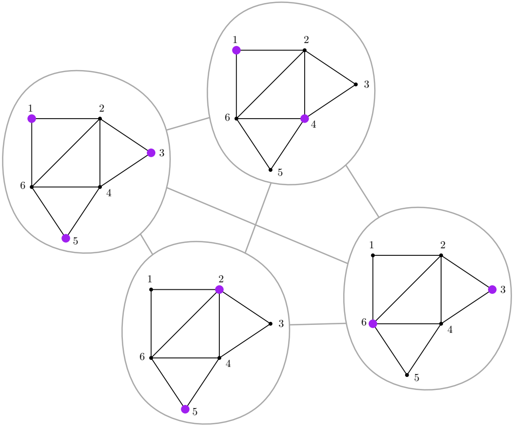
Example with
maximal stable sets
Solution metagraph



General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph
- outputting each solution once
- and **only** once

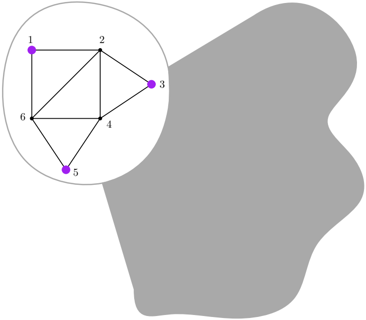
Example with
maximal stable sets
Solution metagraph



General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph
- outputting each solution once
- and **only** once

Example with
maximal stable sets
Solution metagraph



Three classical methods :

Goal : get polynomial delay
+ poly space for 1 et 2 (and sometimes 3)

1. *Flashlight search* or *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

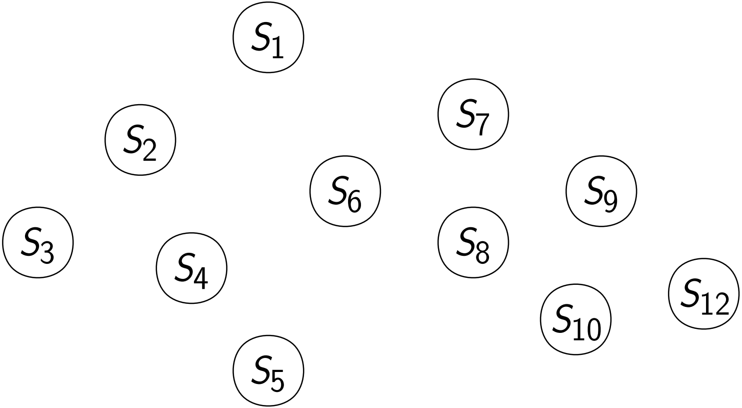
Three classical methods :

Goal : get polynomial delay
+ poly space for 1 et 2 (and sometimes 3)

1. *Flashlight search* or *Binary partition*
   [Read, Tarjan '75]
2. ***Reverse search***
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

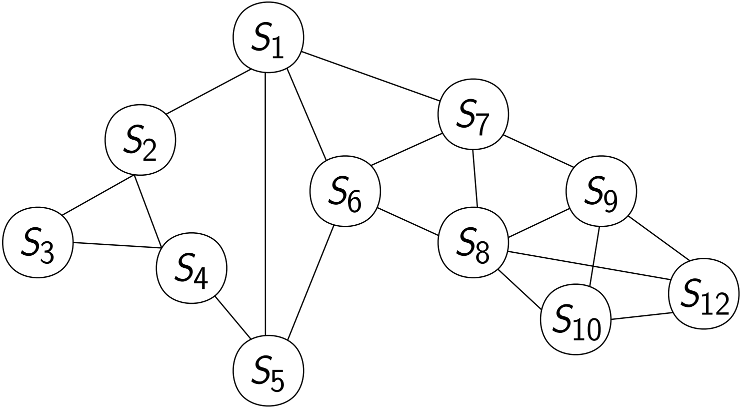
Reverse search

Solution space



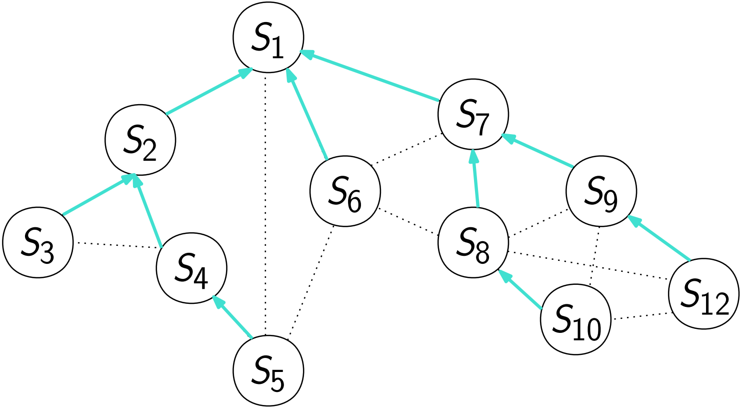
Reverse search

Solution metagraph



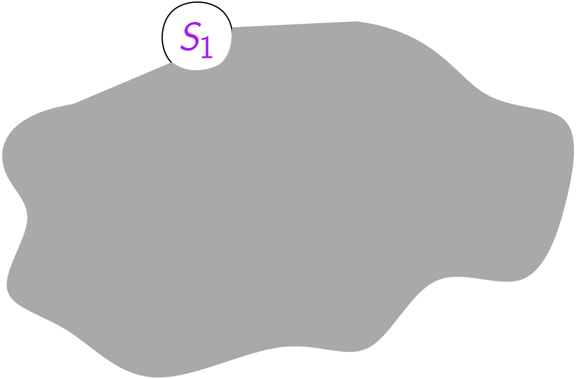
Reverse search

Solution tree

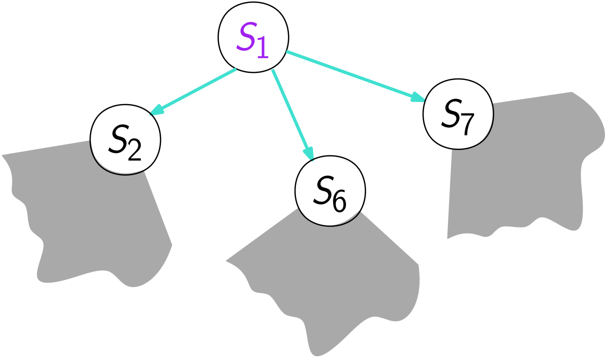


Reverse search

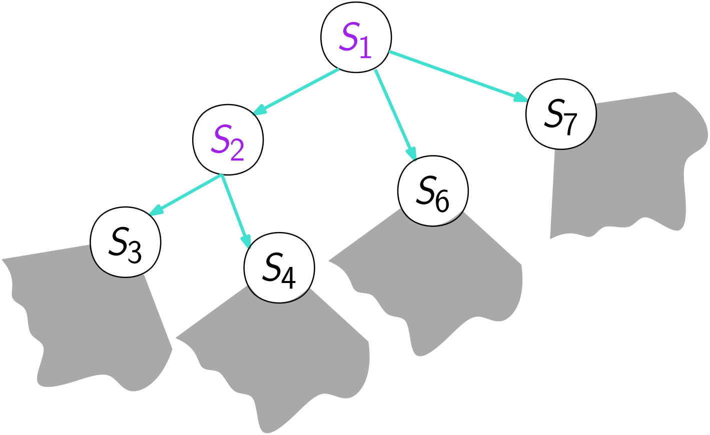
Solution tree



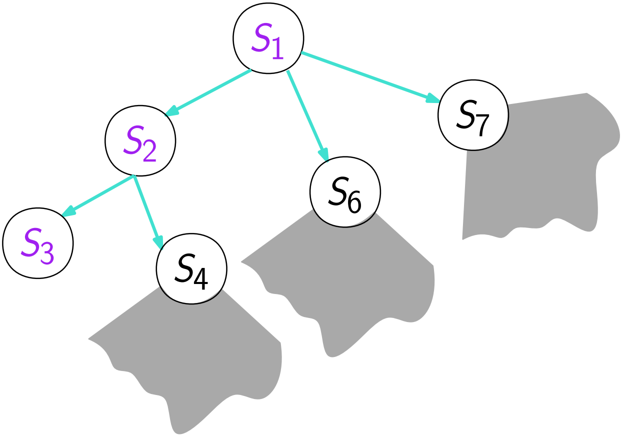
Reverse search



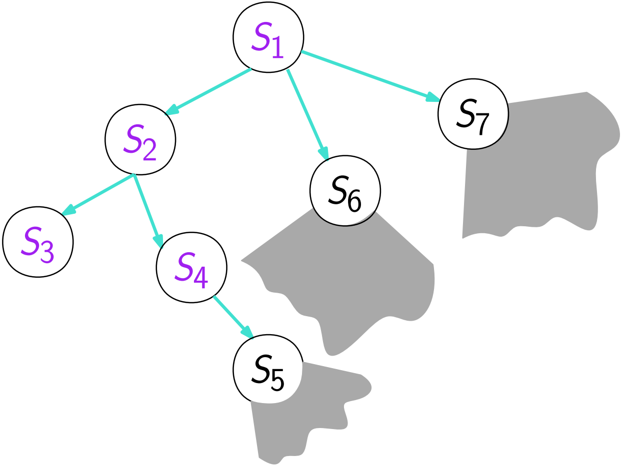
Reverse search



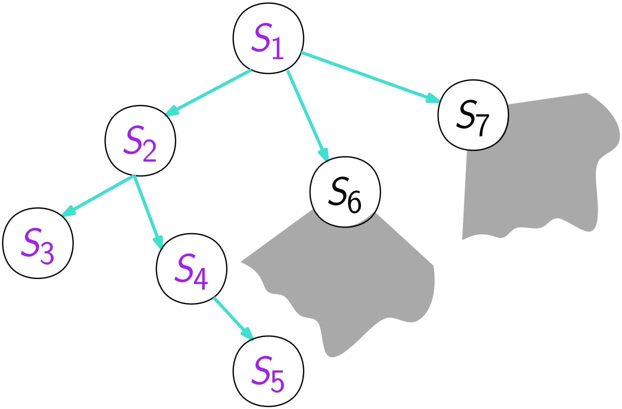
Reverse search



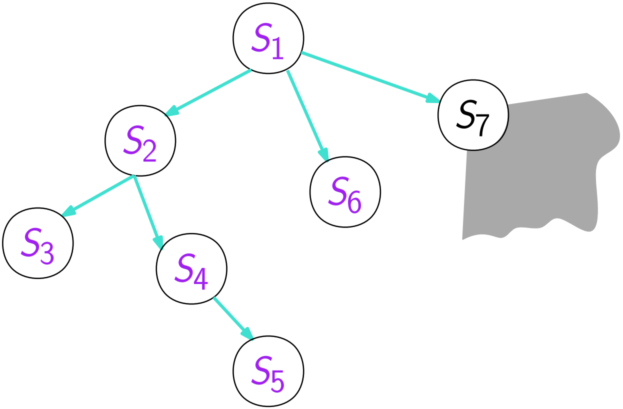
Reverse search



Reverse search



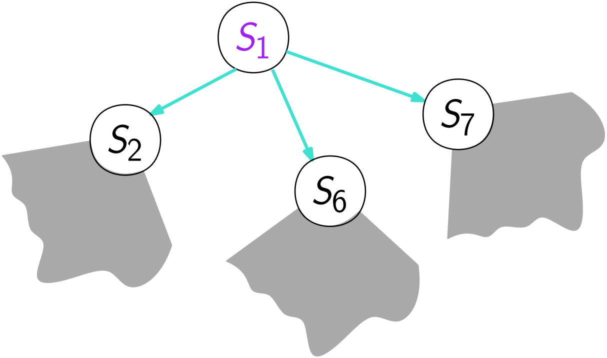
Reverse search



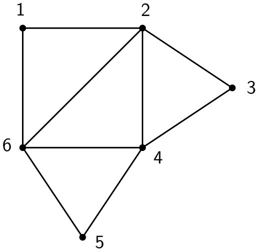
Reverse search

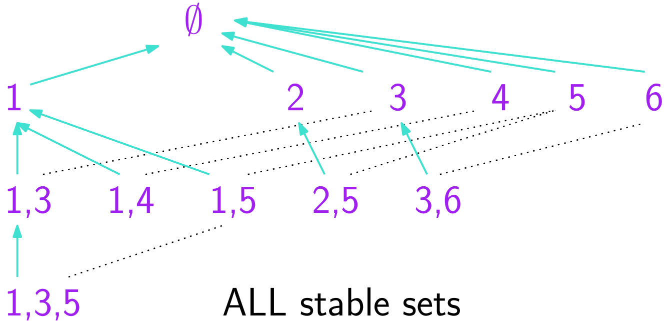
To have *Reverse search* run in poly delay and space :

Generate in poly time and space the *children* of a solution (each solution must have ***a single*** father)



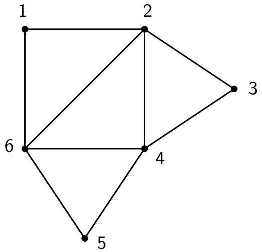
Reverse Search

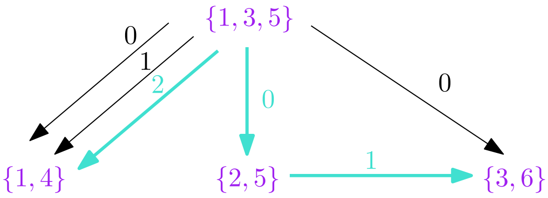




ALL stable sets

Reverse Search for Maximal Stable Sets





$S \xrightarrow{i} S'$ if $S \cap \{v_1, \ldots, v_i\} = S' \cap \{v_1, \ldots, v_i\}$ and S is the lexicographically smallest among all solutions containing $S' \cap \{v_1, \ldots, v_i\}$

P is the father of S if $P \xrightarrow{i} S$ and there is no arc to S indexed $> i$

Three classical methods :

Goal : get polynomial delay
+ poly space for 1 et 2 (and sometimes 3)

1. *Flashlight search* or *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. ***Proximity Search***
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

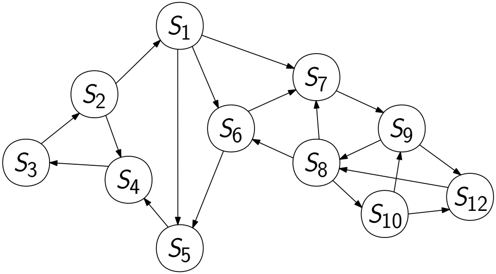
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  → *poly degree*

Solution metagraph



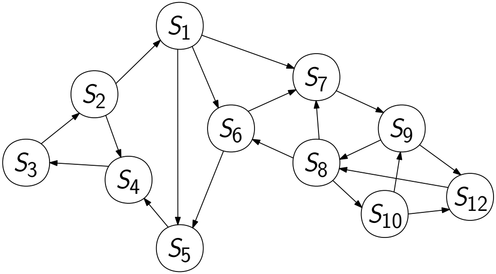
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  → *poly degree*
- check if a vertex has already been visited
  → *exponential space :-(*

Solution metagraph



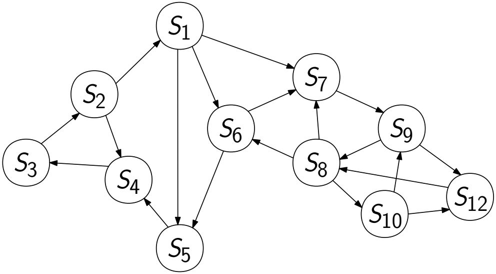
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  $\rightarrow$ *poly degree*
- check if a vertex has already been visited
  $\rightarrow$ *exponential space :-(*
- strong connectedness of the graph to be proven thanks to a notion of **proximity**

Solution metagraph



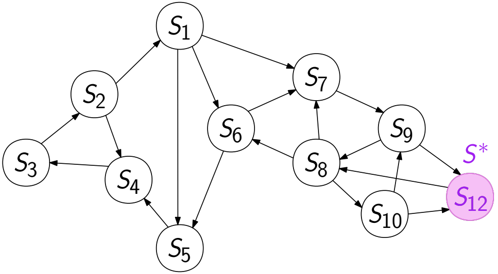
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  → *poly degree*
- check if a vertex has already been visited
  → *exponential space :-(*
- strong connectedness of the graph to be proven thanks to a notion of **proximity**

Solution metagraph



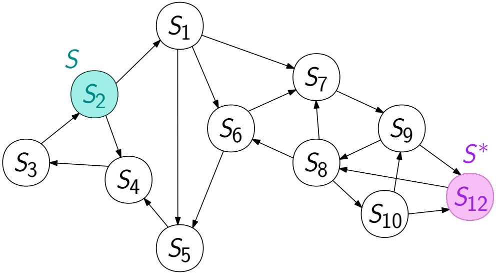
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  $\rightarrow$ *poly degree*
- check if a vertex has already been visited
  $\rightarrow$ *exponential space :-(*
- strong connectedness of the graph to be proven thanks to a notion of **proximity**

Solution metagraph



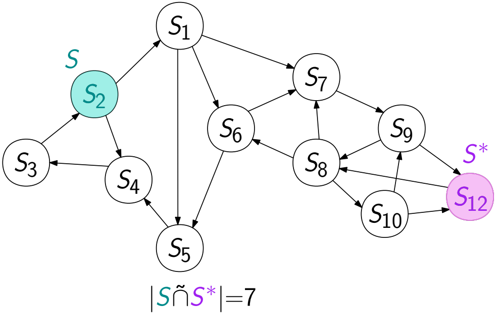
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  → *poly degree*
- check if a vertex has already been visited
  → *exponential space :-(*
- strong connectedness of the graph to be proven thanks to a notion of **proximity**

Solution metagraph



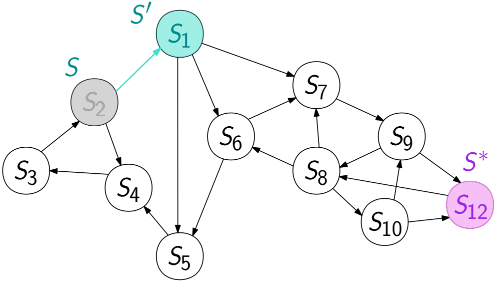
Proximity Search

Designed for enumerating *inclusion-wise maximal* (or min.) solutions to a problem.

Recursive depth-first search (rather classical) with conditions :

- generate **neighbors** of a vertex in poly time
  → *poly degree*
- check if a vertex has already been visited
  → *exponential space :-(*
- strong connectedness of the graph to be proven thanks to a notion of **proximity**

Solution metagraph



$|S' \tilde{\cap} S^*| = 10$

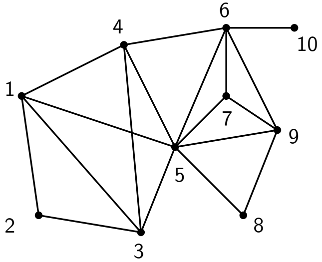
Enumerate chordal fixings

Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.

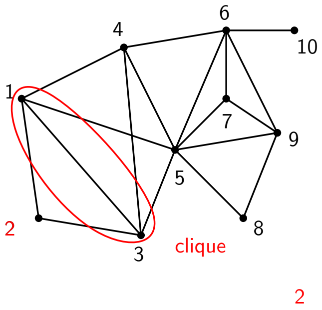


Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.

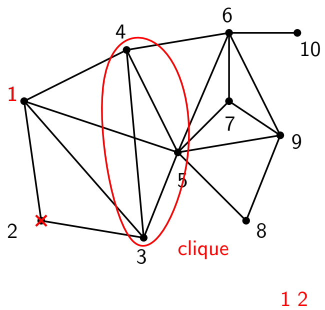


Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.

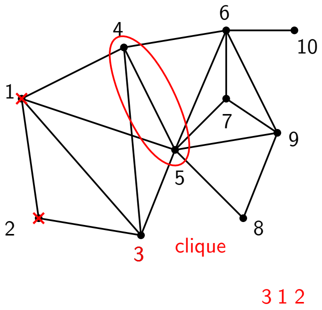


Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.

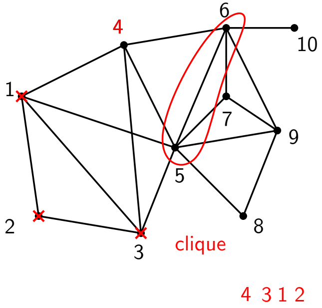


Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.

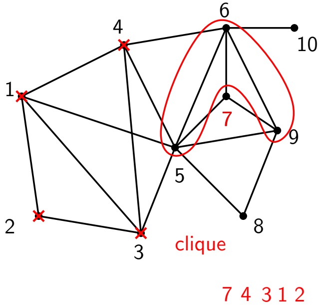


Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.

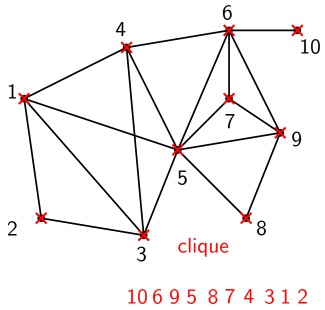


Chordal graph : no chordless cycle of length ≥ 4.

Simplicial vertex

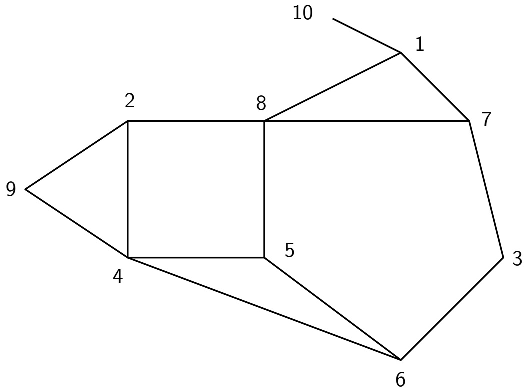
Any chordal graph has a **simplicial vertex** : its neighborhood is a clique.

$\Rightarrow$ **Perfect elimination ordering** : remove simplicial vertices one by one.



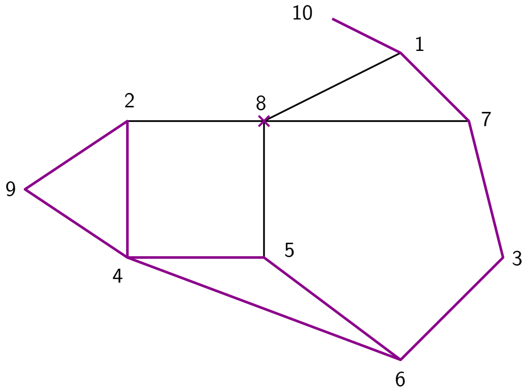
Proximity between solutions

Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)



Proximity between solutions

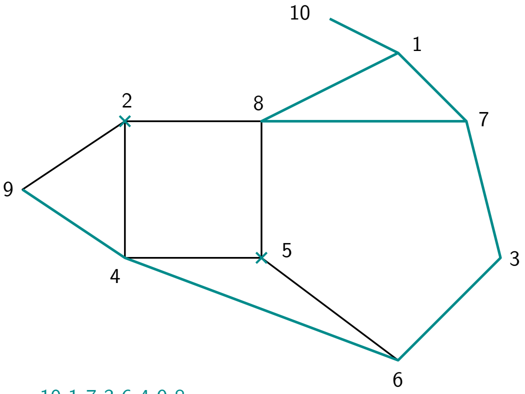
Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)



10 1 7 3 6 4 9 5 2

Proximity between solutions

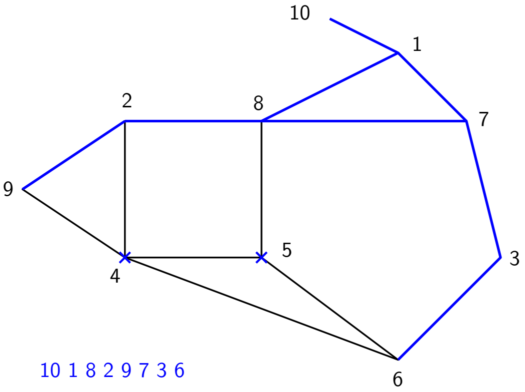
Goal : enumerate all **induced subgraphs** of *G* that are **chordal** (fixing by deleting vertices)



10 1 7 3 6 4 9 8

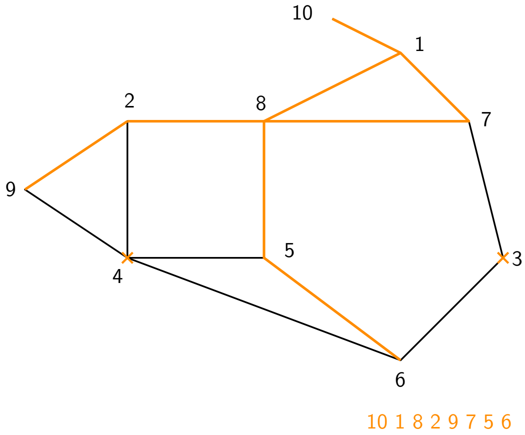
Proximity between solutions

Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)



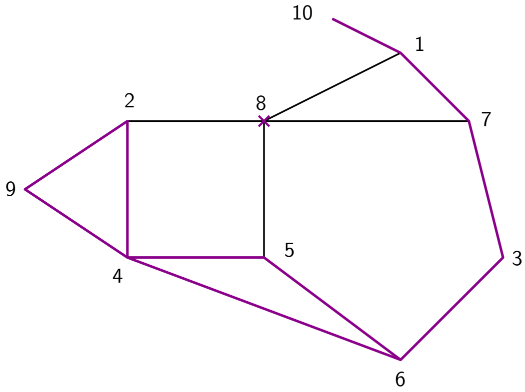
Proximity between solutions

Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)



Proximity between solutions

Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)

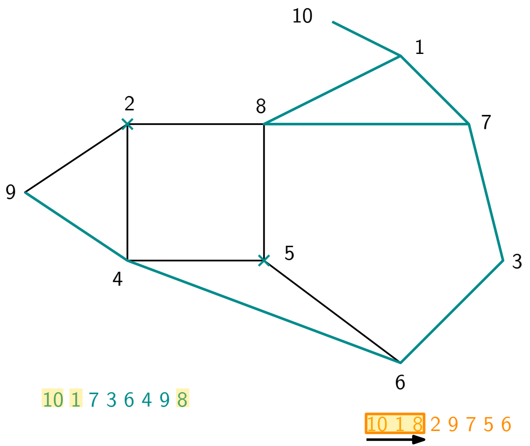


10 1 7 3 6 4 9 5 2

10 1 8 2 9 7 5 6

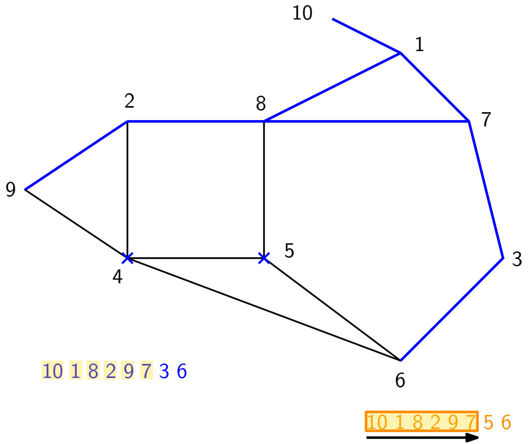
Proximity between solutions

Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)

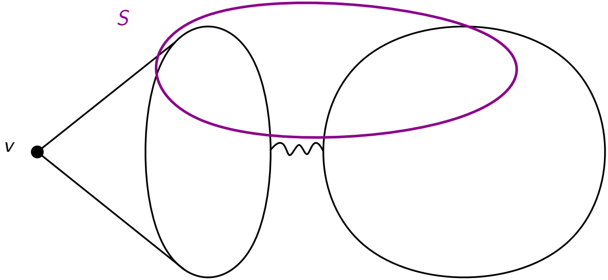


Proximity between solutions

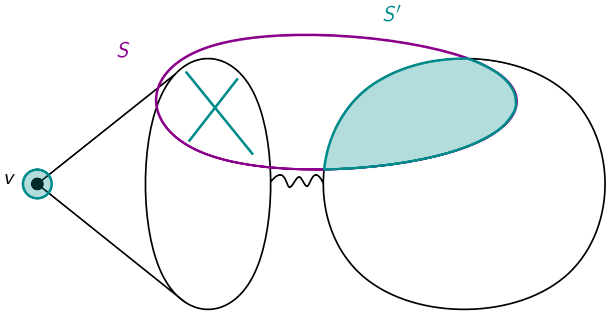
Goal : enumerate all **induced subgraphs** of G that are **chordal** (fixing by deleting vertices)



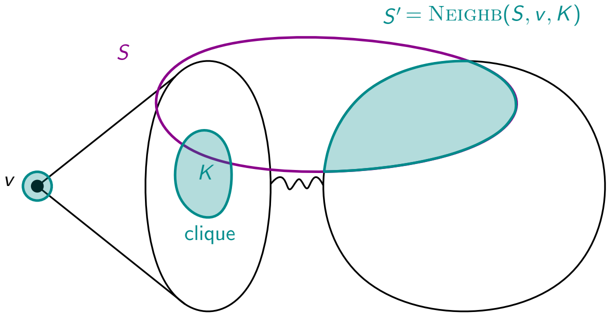
Neighbors in the solution metagraph



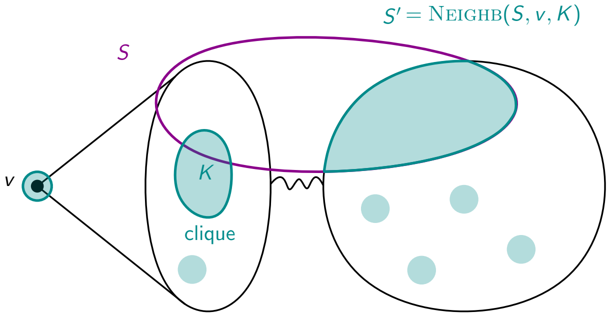
Neighbors in the solution metagraph



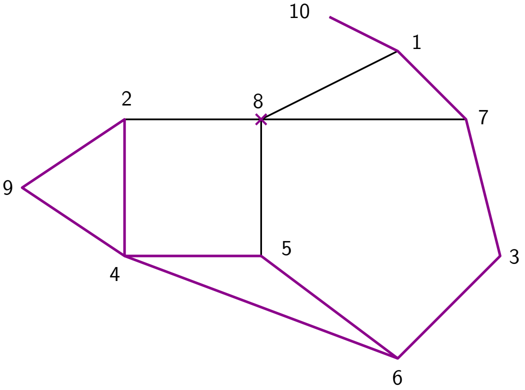
Neighbors in the solution metagraph



Neighbors in the solution metagraph



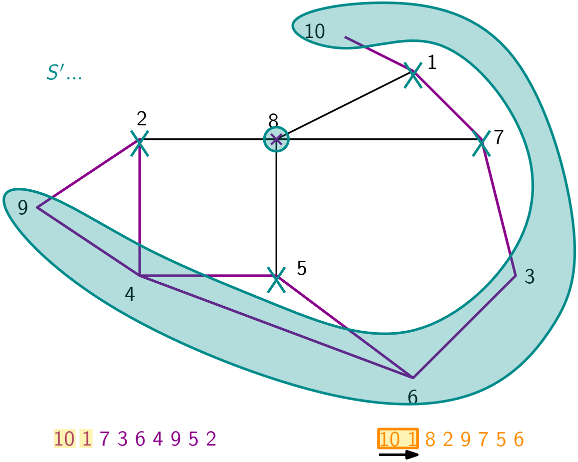
Neighbors in the solution metagraph : example



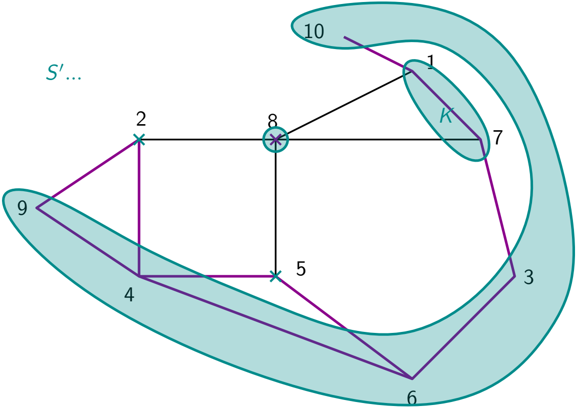
10 1 7 3 6 4 9 5 2

10 1 8 2 9 7 5 6

Neighbors in the solution metagraph : example



Neighbors in the solution metagraph : example

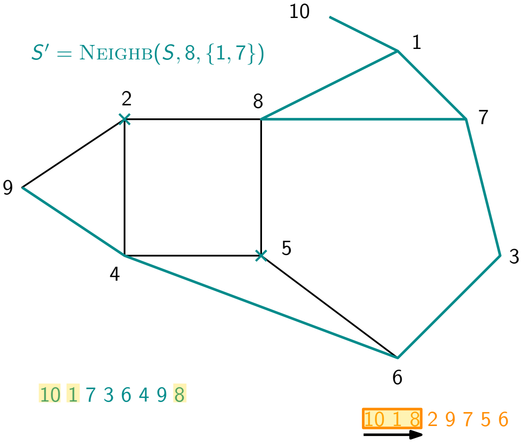


10 1 7 3 6 4 9 5 2

10 1 8 2 9 7 5 6

Neighbors in the solution metagraph : example

$S' = \textsc{Neighb}(S, 8, \{1, 7\})$



Chordal fixings : recap

We have all the ingredients for Proximity Search to work :

1. Proximity between two solutions S and S^* is defined
2. $\mathit{Neighbors}(S)$ is computable in polytime
3. Lemma proving that we can always find a neighbor with higher proximity with a target solution, thanks to the perfect elimination ordering (not shown here)

$\rightarrow$ Enumeration of chordal fixings by inclusion-wise min. deletion of vertices in polynomial delay

Other type of fixings : Chordal completions

$\rightarrow$ Enumerating minimal triangulations of a graph ?

Other type of fixings : Chordal completions

→ Enumerating minimal triangulations of a graph ?

Theorem [Brosse, Limouzy, Mary, 2021^{+}]

There exists a polynomial delay polynomial space algorithm to enumerate all inclusion-wise minimal chordal completion of a graph G given in input.

→ *Proximity Search* with careful arguments (to get polynomial space in particular)

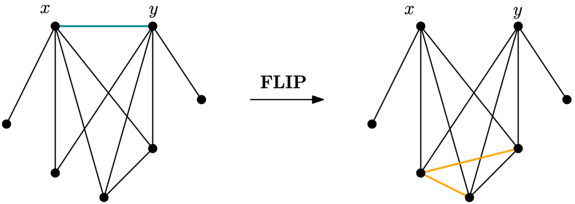
Other type of fixings : Chordal completions

→ Enumerating minimal triangulations of a graph ?

Theorem [Brosse, Limouzy, Mary, 2021^{+}]

There exists a polynomial delay polynomial space algorithm to enumerate all inclusion-wise minimal chordal completion of a graph G given in input.

→ *Proximity Search* with careful arguments (to get polynomial space in particular)



Removing edge xy → Common neighb. turned into clique

Thank you