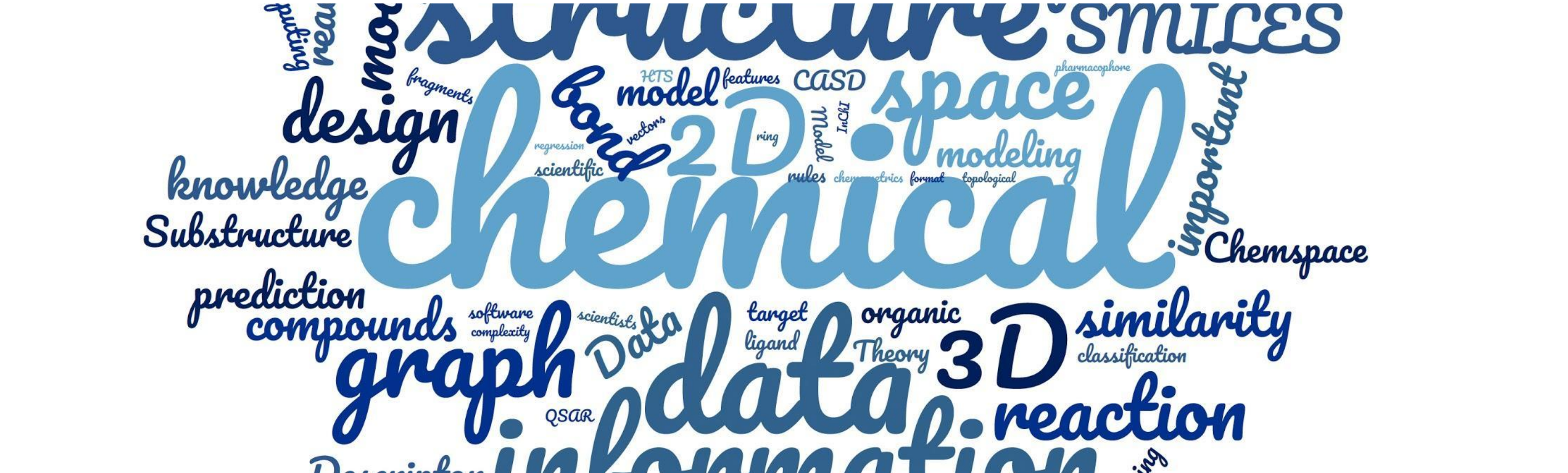
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Institute for Bioinformatics
and Medical Informatics



BIO-4372 **Cheminformatics**

L05 Topological Structure Comparison

Part II: Maximum Common Substructure

Winter Semester 2022-23

Philipp Thiel

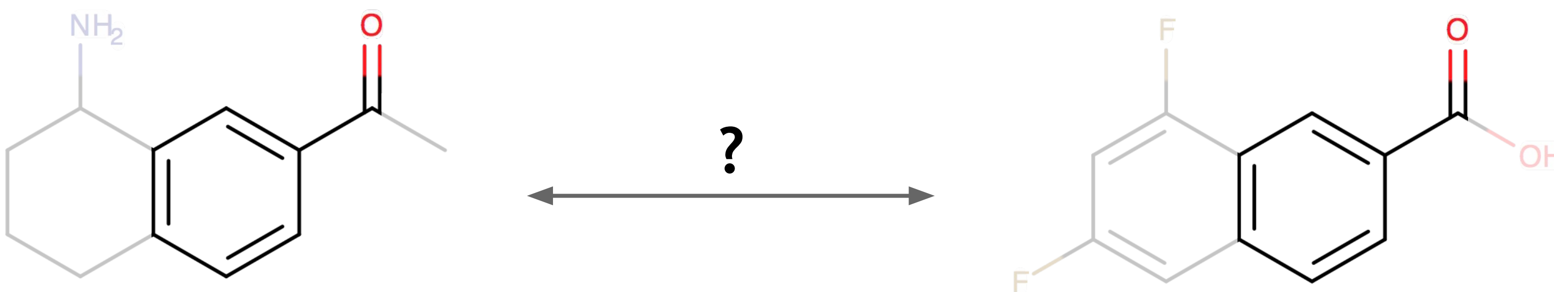
- Problem introduction

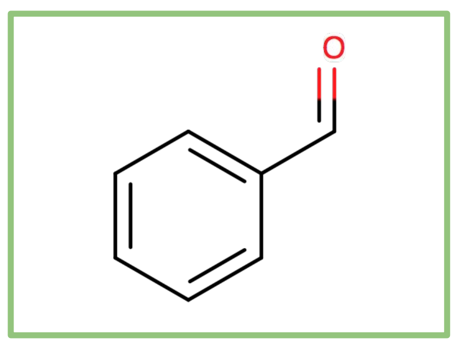
- Two molecule case
  - Example: atom mapping in chemical reactions
  - Reduction to a well-known graph-based problem
  - Maximum clique detection
  - Bron-Kerbosch algorithm

- Multiple molecule case
  - Example: Identification of active core structure
  - Extension non-trivial
  - Pairwise search for maximal common substructure

Motivation

- **Maximum Common Substructure: MCS**





Motivation

- **Maximum Common Substructure: MCS**
- The largest common substructure of two molecules
- Very important concept in cheminformatics
  - Also used in other molecular science areas
  - An overview can be found in Ehrlich and Rarey (2011) [1]

- Two problem variants have cheminformatic use cases:

1. **Two molecule case**
2. Multiple molecule case

1. Ehrlich H.C. and Rarey M. (2011) *WIREs Comput. Mol. Sci.*, 1, 68-79, 10.1002/wcms.5

Motivation

Two Molecule Case

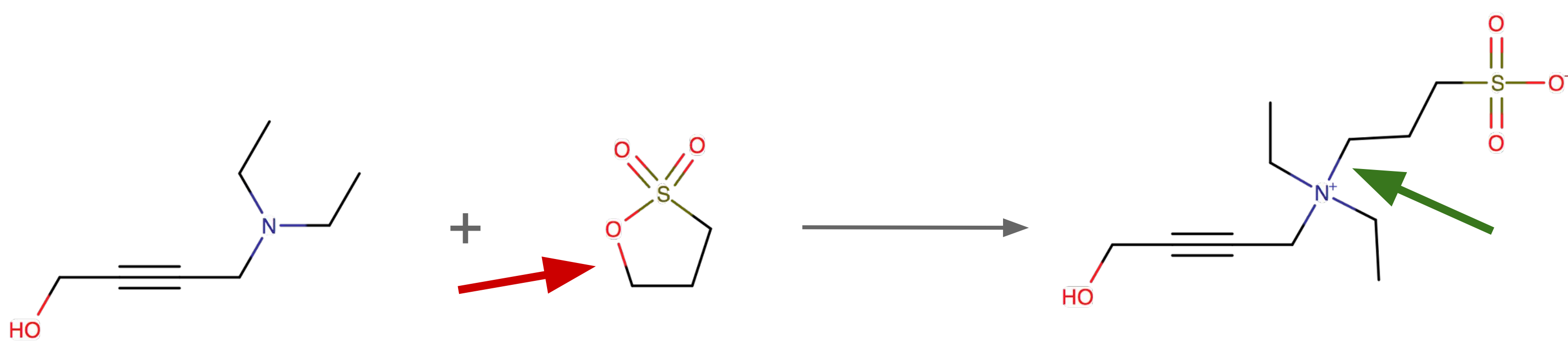
- **Atom mapping for chemical reactions**
- Chemical reactions transform educt(s) into product(s)
- Large databases of chemical reactions exist
- Learning from that information would be extremely useful
  - Prediction of chemical reactivity

- Required to achieve this goal(s):
  1. Reactions have to be balanced
  2. **Reactions have to be atom-mapped**

Motivation

Two Molecule Case

- **Atom mapping for chemical reactions**
- Experimental approach: isotope-labeling experiments and NMR
  - Expensive in time and money
- Computational strategies of utmost interest
  - Active research field
- **MCS is a key technique**

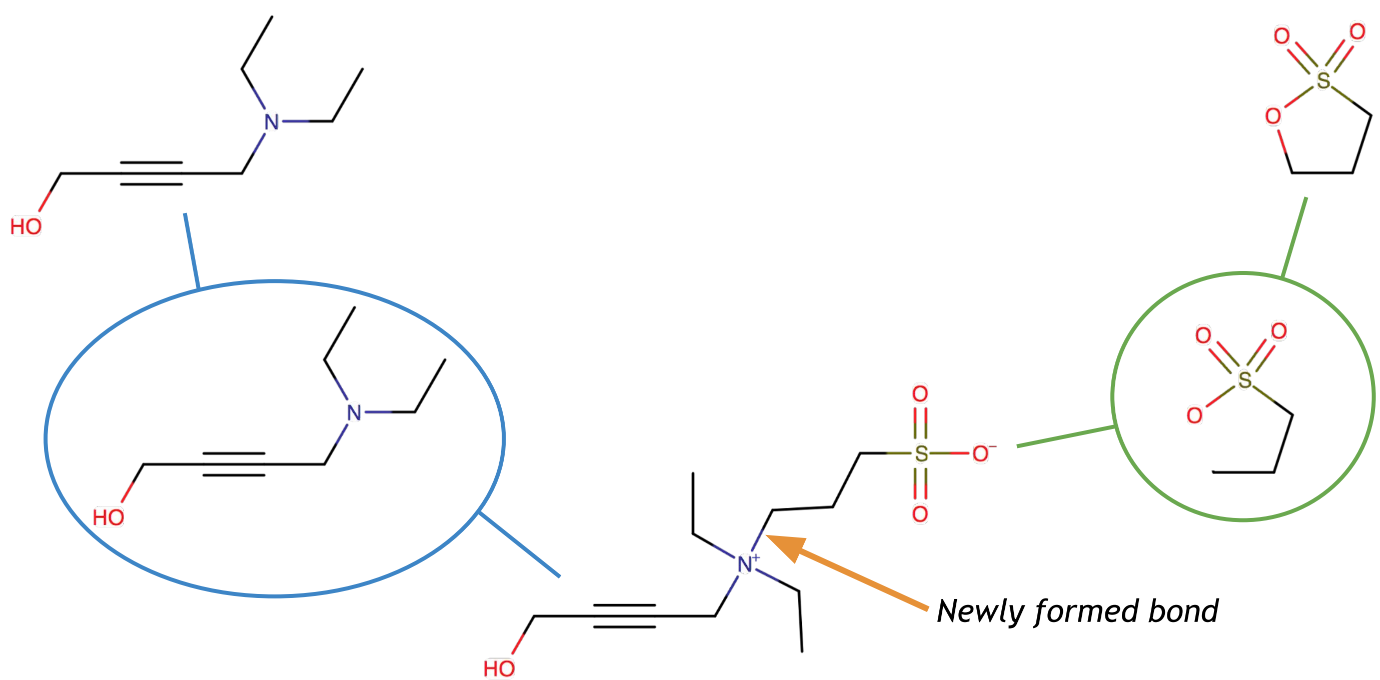


Modified after: Fooshee D. et al. (2013) *J. Chem. Inf. Model.*, 53, 2812-9

Motivation

Two Molecule Case

- **Atom mapping for chemical reactions**
- MCS mapping



Modified after: Fooshee D. et al. (2013) *J. Chem. Inf. Model.*, 53, 2812-9

Maximum Common Substructure
Essential Problem

- We first discuss the **two molecule case**
  - Multiple molecule case less explored

- Variant of **Maximum Common Subgraph Isomorphism**
  - **NP-complete** in the general case [1]

- **Maximum Common Substructure** for molecules
  - Labeled graph with bounded node degree
    - Cf. lecture *L03 Chemical Data Representation*
  - Solvable for most medium sized molecules in acceptable time
    - That is within seconds

1. [GJ]

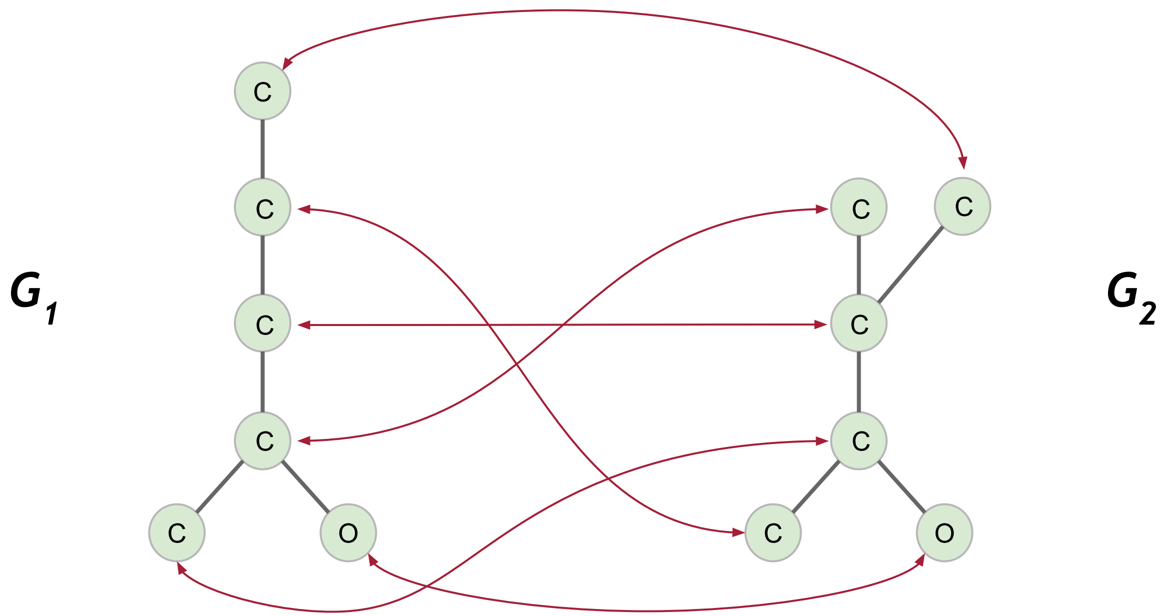
Maximum Common Substructure

Essential Problem

- Given: Molecular graphs $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ and a node labeling function μ: $V_1 \cup V_2 \rightarrow \Sigma$
- Problem: Find a bijection m: $V_1' \rightarrow V_2'$ mapping each node from $V_1' \subseteq V_1$ on a node from $V_2' \subseteq V_2$ such that $\mu(v) = \mu(m(v)) \;\; \forall \;\; v \in V_1'$
- As we search for a **maximum** common substructure the mapping m should be **maximal**. That is no other mapping exists that maps more than $|V_1'|$ nodes.

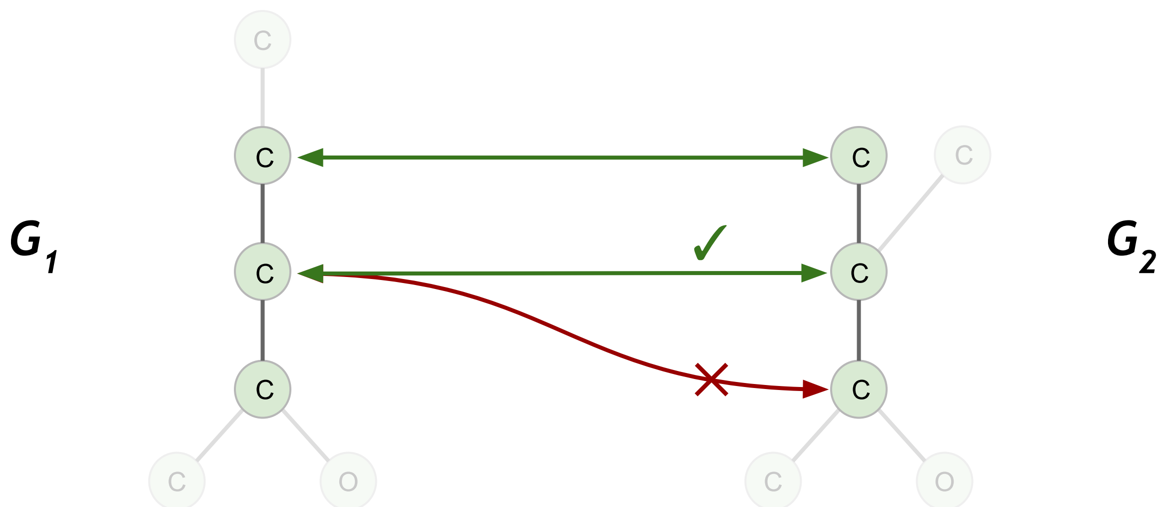
Maximum Common Substructure

Essential Problem: Topology



- Problem: mapping is **not topology preserving**
  ⇒ chemically not meaningful!

Maximum Common Substructure
Essential Problem: Topology

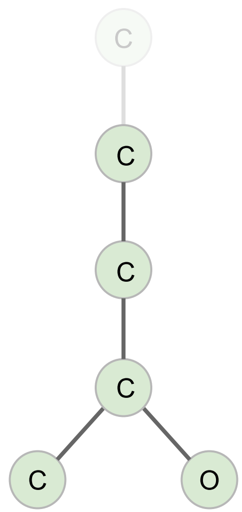


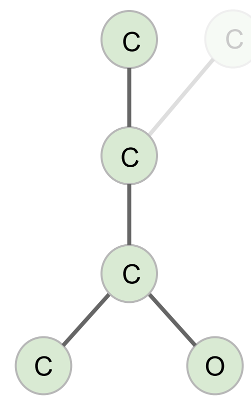
- We need to define **appropriate constraints**:
  To preserve topology we have to ensure that adjacent nodes in G_1 can only be mapped onto adjacent nodes in G_2

Maximum Common Substructure

Essential Problem: Topology

G_1





G_2

- **Topology constraints preserve chemistry**

Maximum Common Substructure

Essential Problem

- Formally, we have to add the following requirement:

  $(u, v) \in E_1 \Leftrightarrow (\mathrm{m}(u), \mathrm{m}(v)) \in E_2$

- Topology constraint significantly complicates problem

- Questions:

  1. How many such mappings exist?
  2. How to identify suitable mappings?

Maximum Common Substructure

Number of Possible Mappings

- Worst case:
  - In G_1 and G_2: **all labels are identical**
  - G_1 and G_2 are complete

- Consequence: all unique bijections are valid
- Assuming $|V_1| = |V_2| = n$ we have $n!$ possible mappings of G_1 onto G_2 and thus **exponentially many**!

- How can we identify such mappings efficiently?

Maximum Common Substructure
Problem Variants

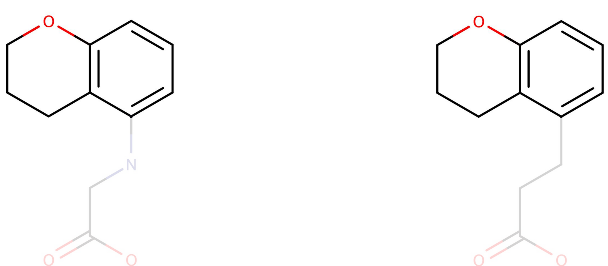
- MCS can refer to different problem variants
- Two groups of variants can be distinguished:
  1. Connected and Disconnected MCS
  2. MCIS and MCES

Maximum Common Substructure

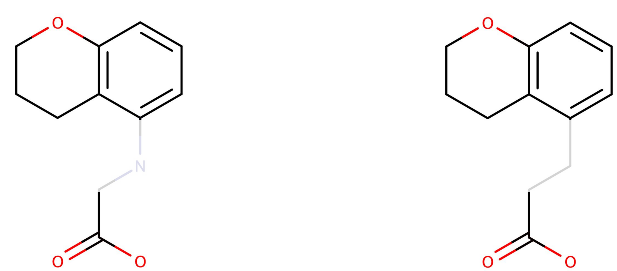
Problem Variants

- MCS can refer to different problem variants
- Two groups of variants can be distinguished:

1. **Connected and Disconnected MCS**
2. MCIS and MCES



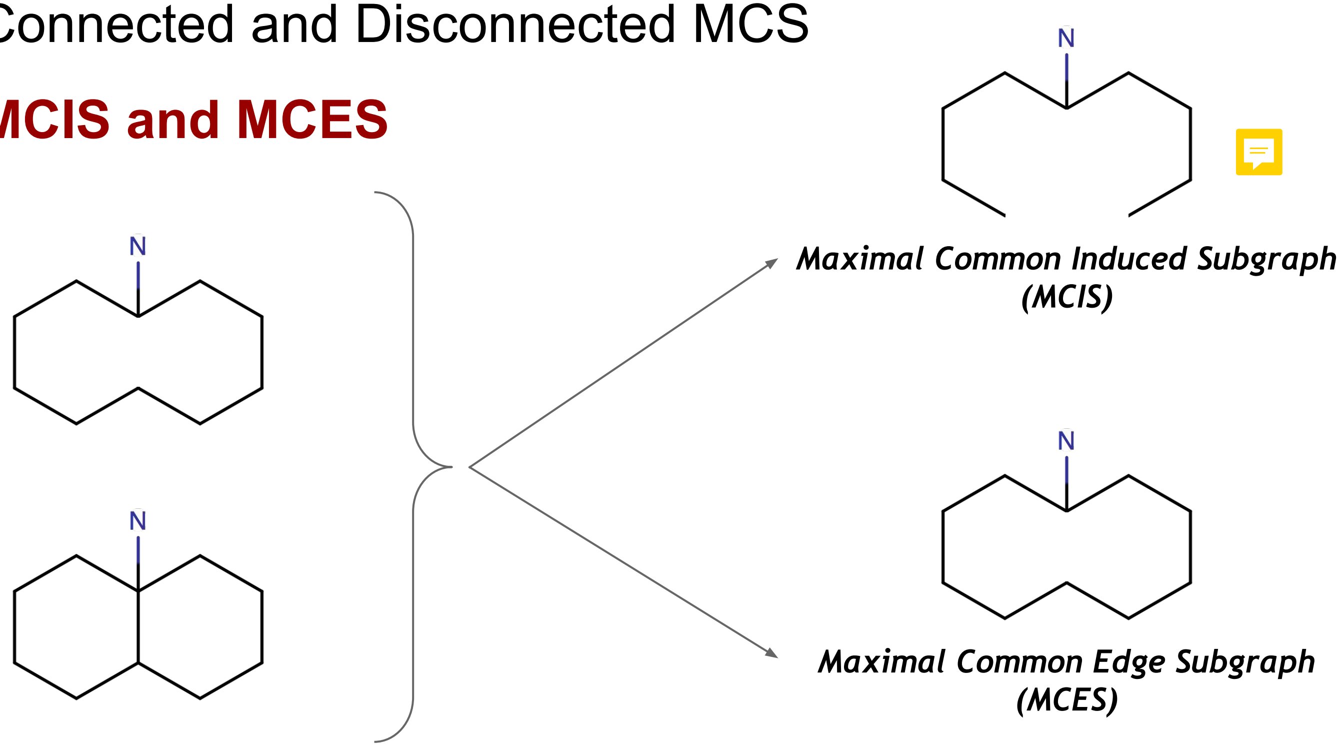
Connected MCS



Disconnected MCS

Maximum Common Substructure
Problem Variants

- MCS can refer to different problem variants
- Two groups of variants can be distinguished:
  1. Connected and Disconnected MCS
  2. **MCIS and MCES**



Maximum Common Substructure
Algorithmic Approaches

- A lot of algorithms have been proposed to solve MCS [1]
  - Exact algorithms
    - **Maximum Clique-based**
    - Backtracking
    - Dynamic programming [2]
  - Approximate algorithms
    - Genetic algorithms
    - Combinatorial optimization
    - Others

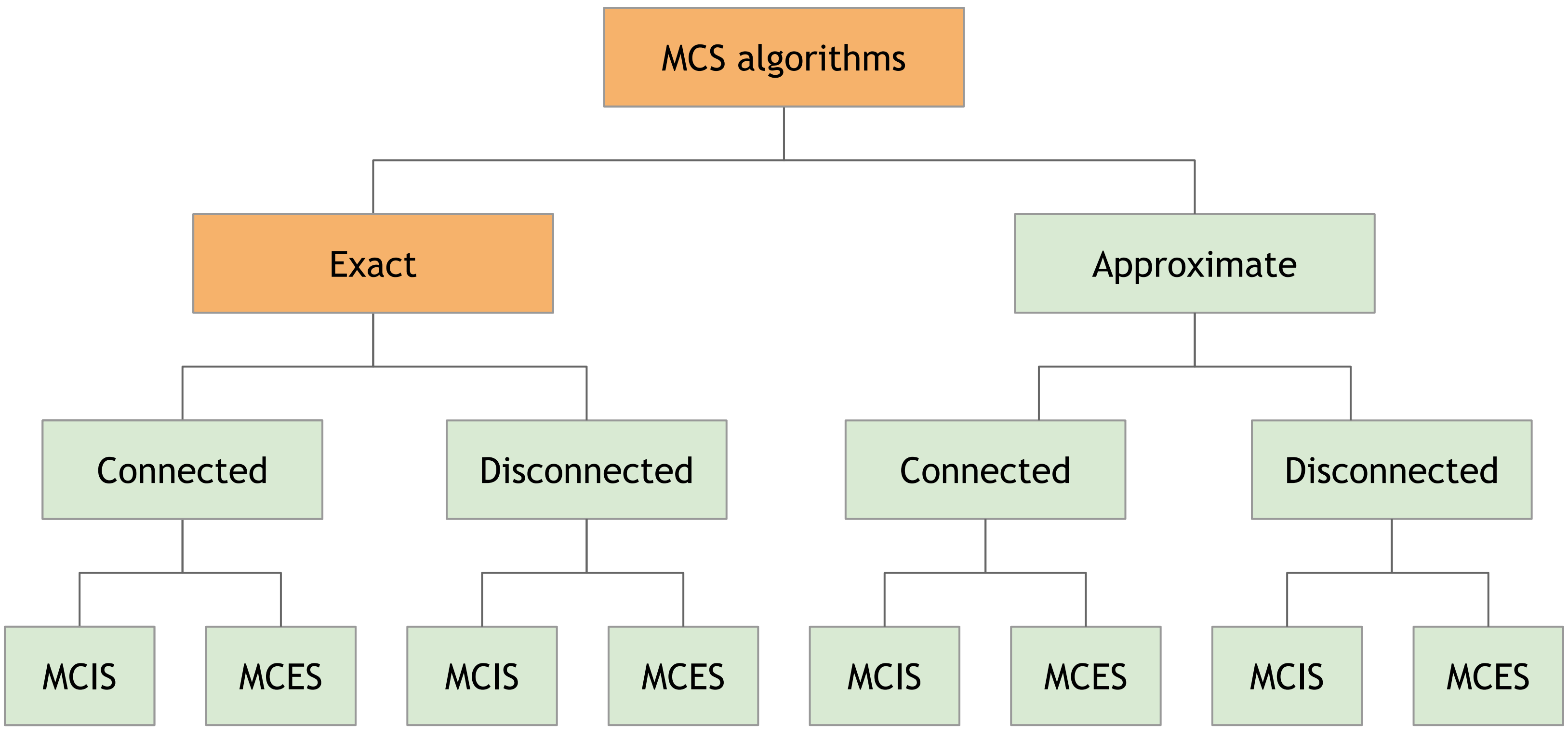
- Solution in **polynomial time** for tree-like graphs with **bounded node degree** [2]

1. Raymond J.W. and Willett P. (2002) *J. Comput. Aided Mol. Des.*, 16, 521-33
2. Akutsu T. (1993) *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E76-A, 1488

Maximum Common Substructure

Algorithmic Approaches

- Algorithms for MCS can thus also be classified [1,2]



1. Raymond J.W. and Willett P. (2002) *J. Comput. Aided Mol. Des.*, 16, 521-33
2. Ehrlich H.C. and Rarey M. (2011) *WIREs Comput. Mol. Sci.*, 1: 68-79, doi: 10.1002/wcms.5

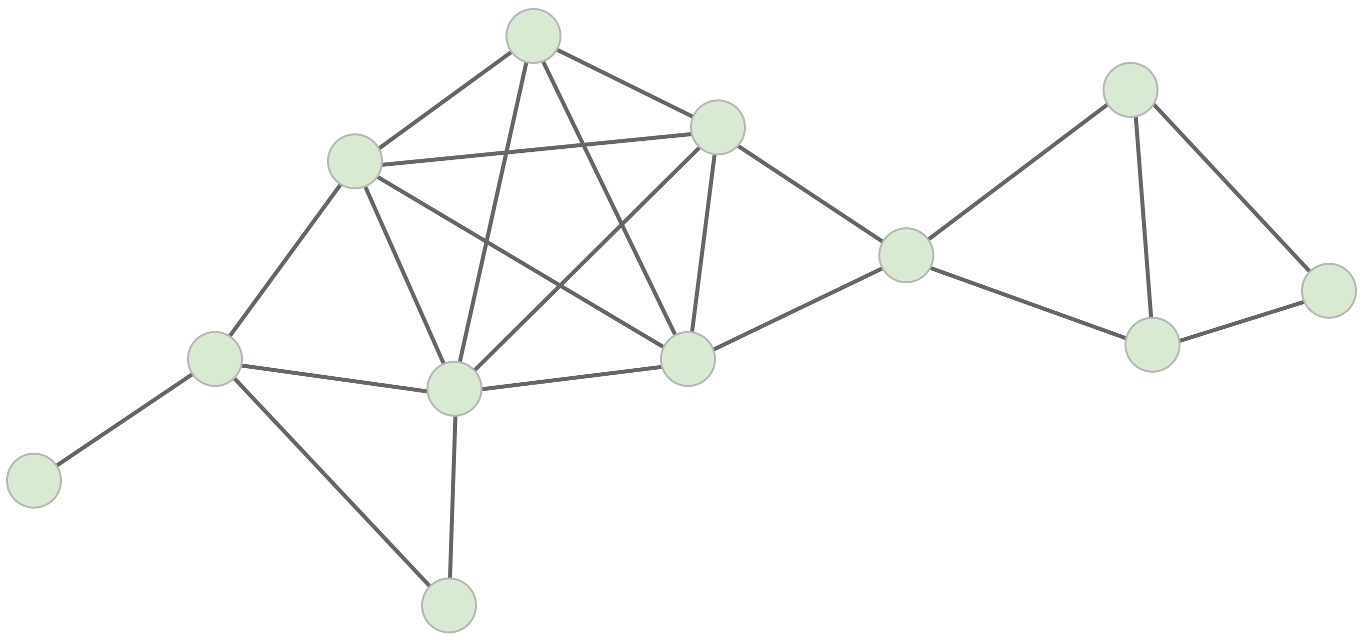
MCS: Maximum Clique Approach

Cliques

- MCS can be reduced to detection of a **maximum clique**

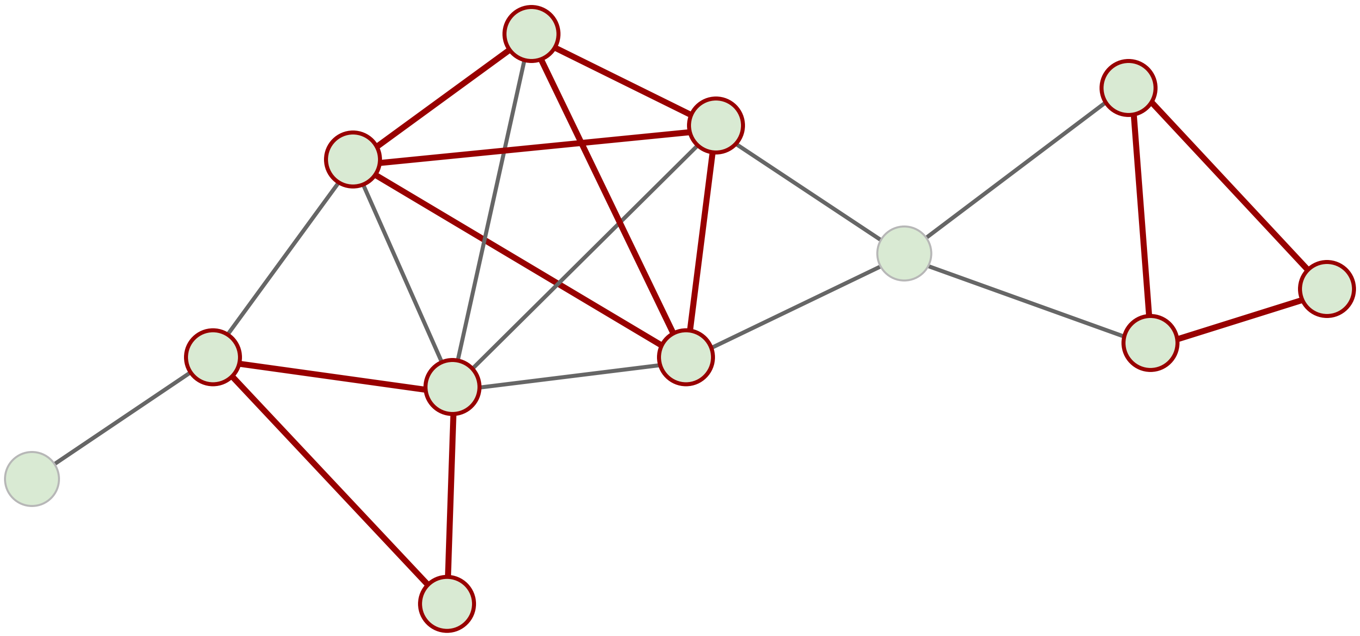
- Given a graph $G = (V, E)$
  - **Clique**: a complete subgraph of G.
  - **Maximal clique**: a clique where no further $v \in V$ can be added (including its induced edges) such that the resulting subgraph is again a clique.
  - **Maximum clique**: largest maximal clique(s) of G with respect to the number of nodes.

MCS: Maximum Clique Approach
Cliques



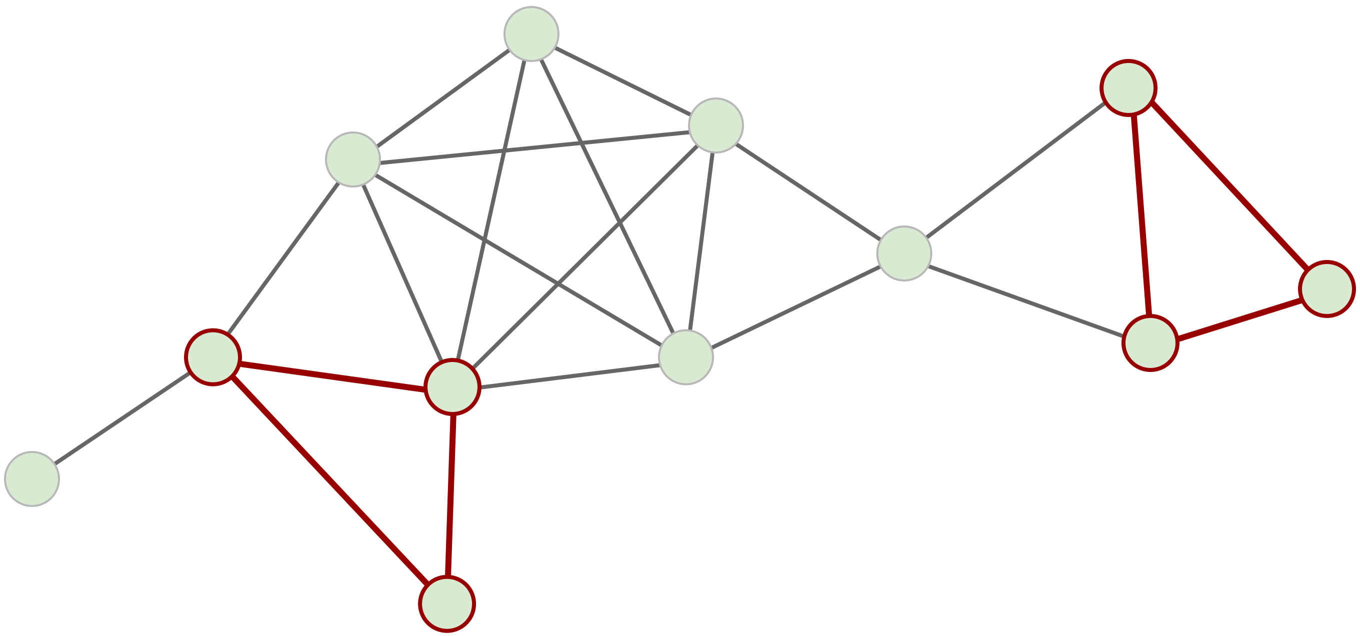
Graph *G*

MCS: Maximum Clique Approach
Cliques



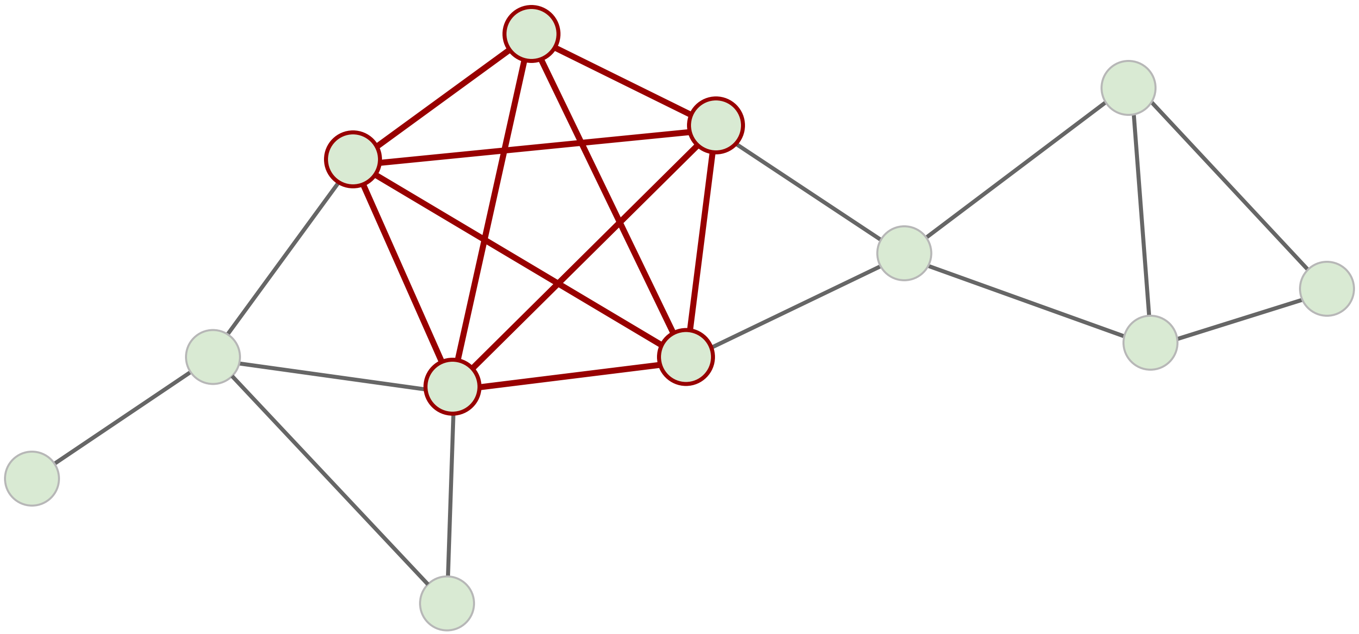
Graph *G*: some **cliques** ...

MCS: Maximum Clique Approach
Cliques



Graph *G*: two **maximal cliques** ...

MCS: Maximum Clique Approach
Cliques



Graph *G*: **maximum clique**

MCS: Maximum Clique Approach

Overview

- Clique detection works on a **single graph**

- Questions:
  1. What graph is that?
  2. How do we generate that graph from our molecular graphs?
  3. How can we calculate maximum cliques?

- We will discuss these steps in the following

MCS: Maximum Clique Approach
Compatibility Graph

- Target graph: **compatibility graph**
  - Association graph
  - Correspondence graph
  - Modular product graph

- We have to molecular graphs *A* and *B*
- We have one compatibility graph

  → Obviously the latter needs to be calculated from *A* and *B*

MCS: Maximum Clique Approach

Compatibility Graph

- Given two (molecular) graphs $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$, the compatibility graph G_C is defined as the vertex set $V_C \subseteq V_1 \times V_2$ where $\mu(v_{1i}) = \mu(v_{2j})$ for all $<v_{1i},v_{2j}> \in V_C$ and in which $<v_{1i},v_{2j}>$ and $<v_{1r},v_{2s}>$ are adjacent iff

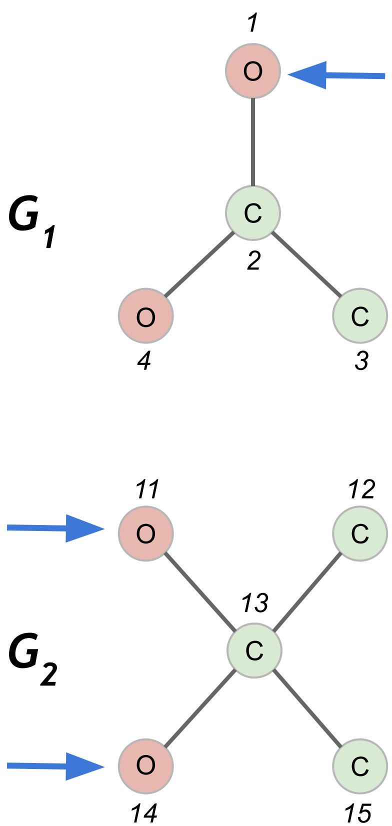
$$\left.\begin{array}{l} (v_{1i},v_{1r}) \in E_1 \;\text{ and }\; (v_{2j},v_{2s}) \in E_2 \\ \text{or} \\ (v_{1i},v_{1r}) \notin E_1 \;\text{ and }\; (v_{2j},v_{2s}) \notin E_2 \end{array}\right\} \text{Topology preservation!}$$

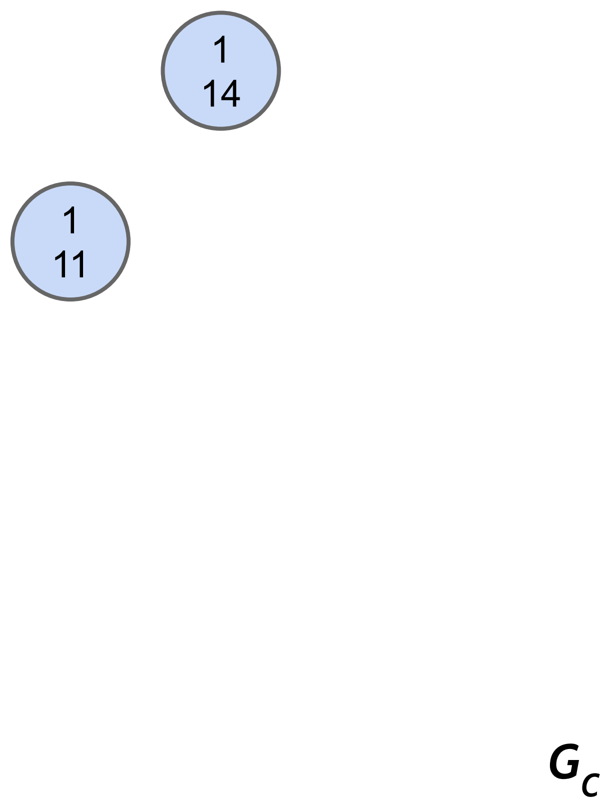
for $v_{1i} \neq v_{1r}$ and $v_{2j} \neq v_{2s}$

MCS: Maximum Clique Approach

Compatibility Graph

- Construction of G_C

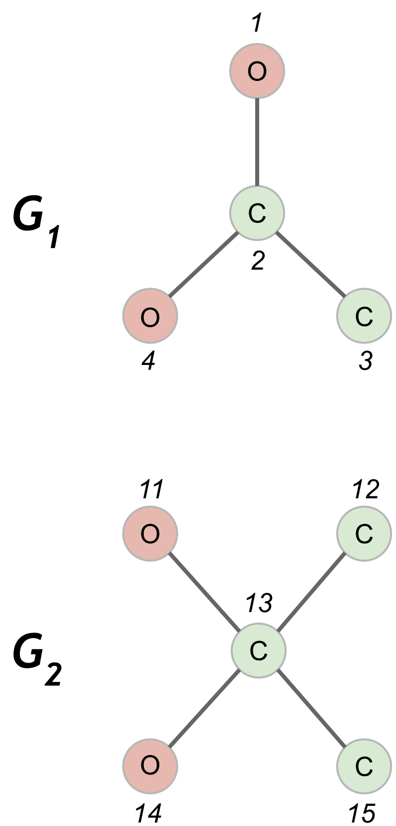


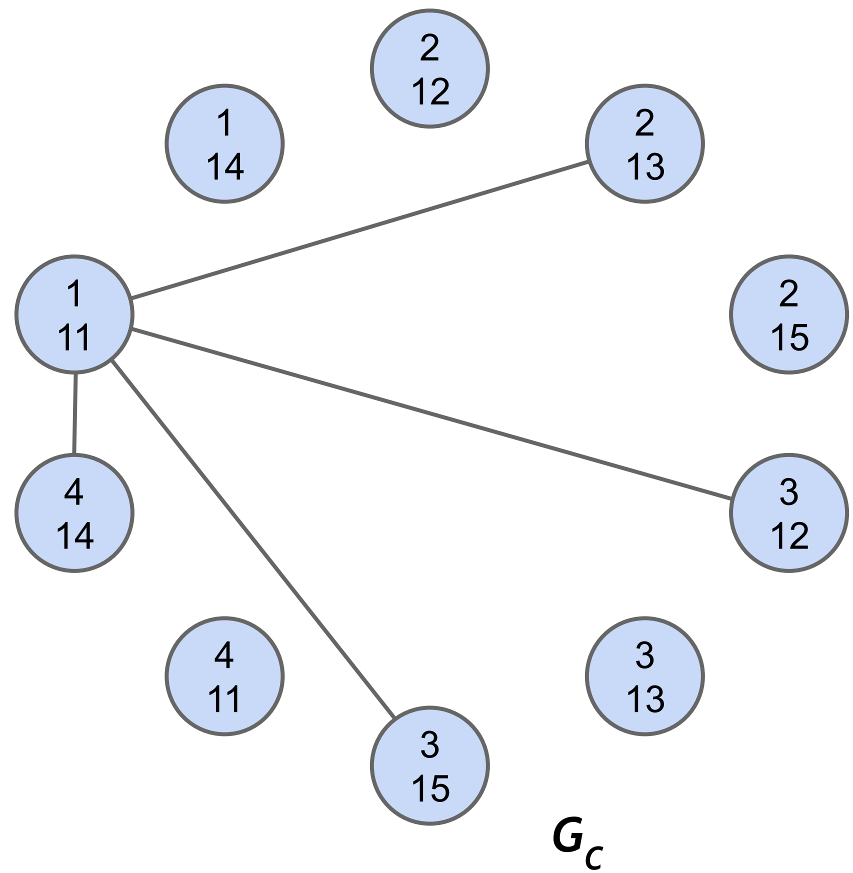


MCS: Maximum Clique Approach

Compatibility Graph

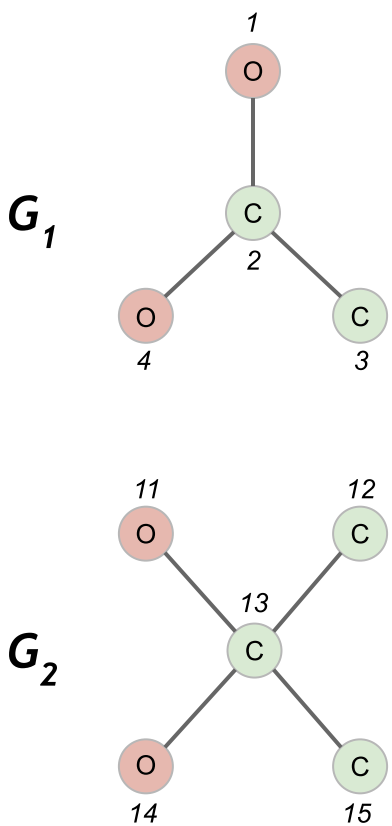
- Construction of G_C

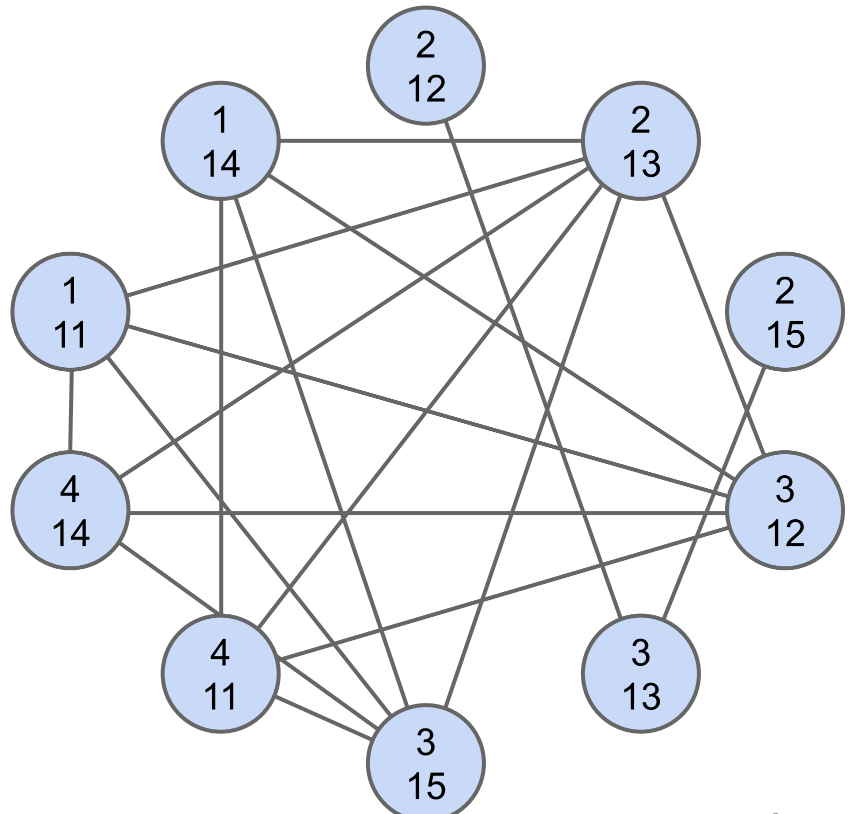




MCS: Maximum Clique Approach
Compatibility Graph

- Construction of G_C



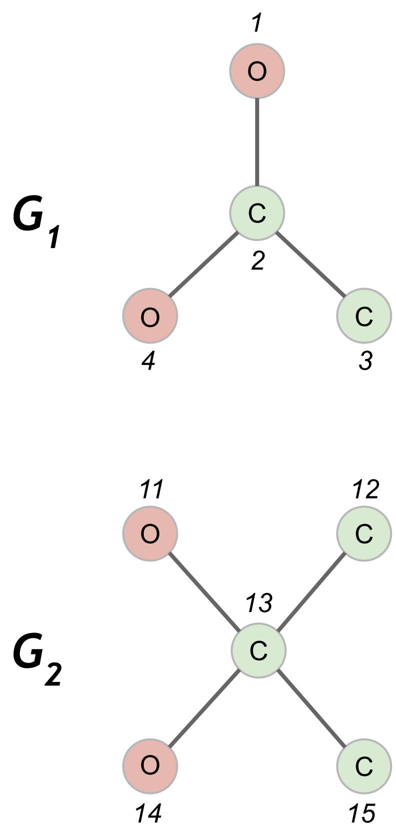


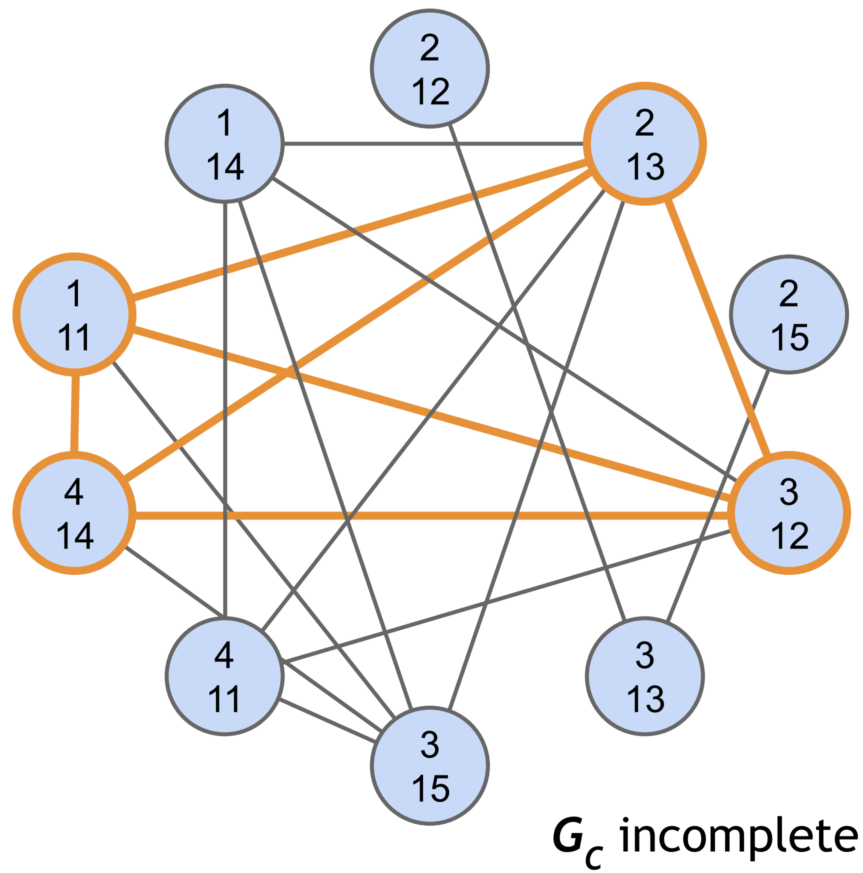
G_C incomplete!

MCS: Maximum Clique Approach

Compatibility Graph

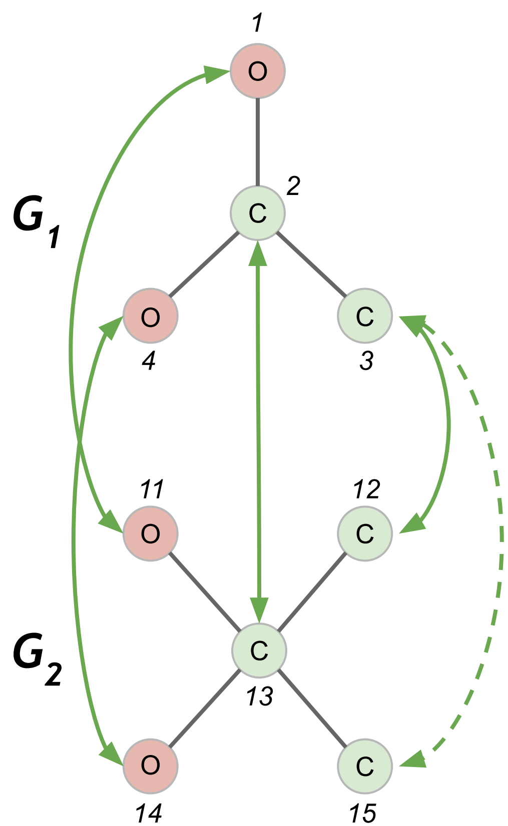
- Construction of G_C

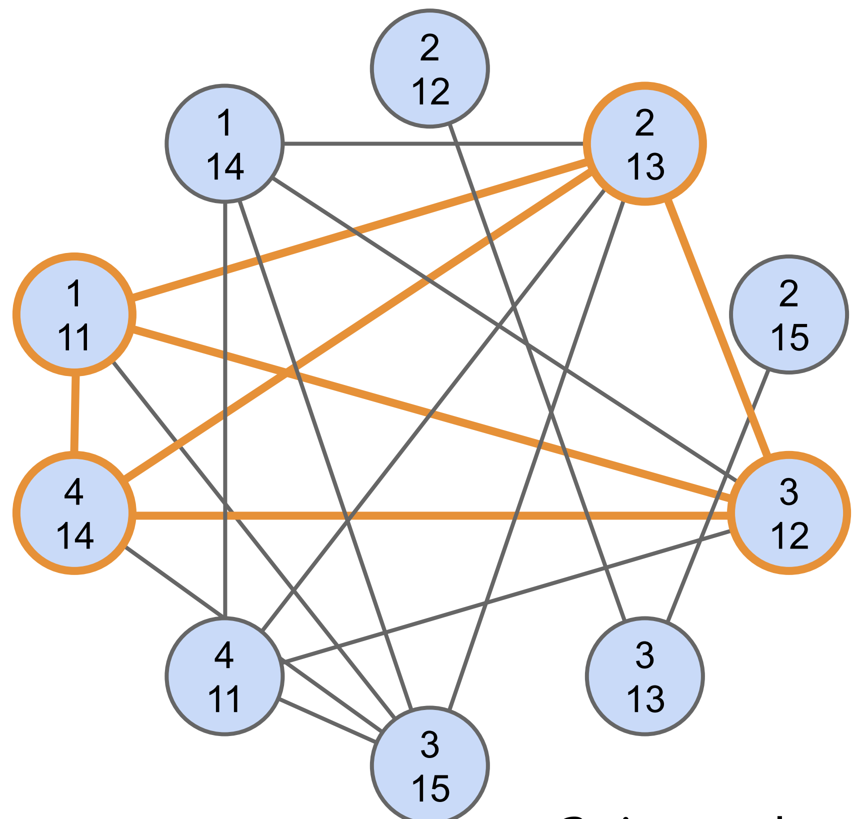




MCS: Maximum Clique Approach
Compatibility Graph

- **Maximum clique** in G_C corresponds to **MCS** between G_1 and G_2





G_C incomplete!

MCS: Maximum Clique Approach

Maximum Clique Problem

- Well known but also **NP-complete**
  - Reducible to 3-SAT
- A number of algorithms exist for solving this problem
- Example: **Bron-Kerbosch algorithm** [1]
  - Popular in cheminformatics
  - Easy to implement
- One key problem remains:

  MCS cannot be solved efficiently for large molecules and is still computationally expensive even for small to medium-sized molecules

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm

- **Given**: A graph $G = (V, E)$, e.g. a compatibility graph
- Bron and Kerbosch proposed a simple algorithm using **recursive tree-search with backtracking**
- It enumerates all maximal cliques
  - The maximum clique thus being part of it
- It uses three node lists:

  C: current clique candidate nodes

  M: nodes of next maximal clique

  N: tested nodes that are not part of the next maximum clique

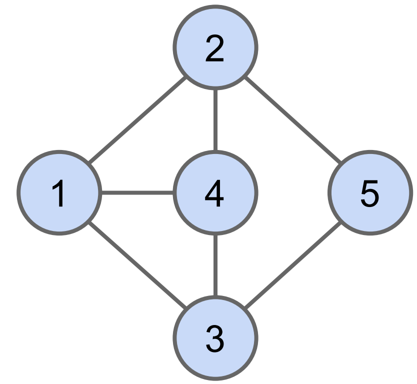
1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;

```
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



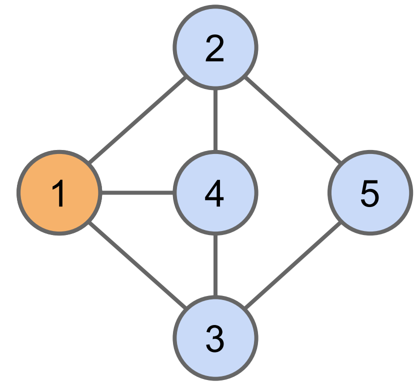
M	*C*	*N*
	1,2,3,4,5	

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



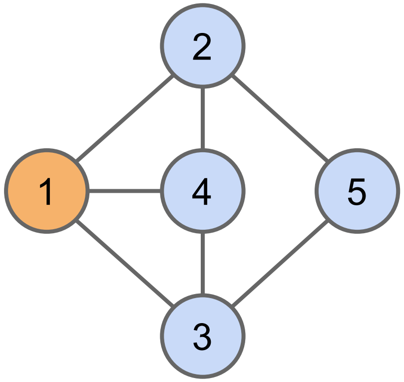
In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function `BronKerbosch`(M, C, N)
  if $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
    return;
  end
  foreach $u \in C$ **do**
    $M_{new} = M \cup \{u\}$;
    $C_{new} = \{v \in C \mid (u, v) \in E\}$;
    $N_{new} = \{v \in N \mid (u, v) \in E\}$;
    if $C_{new} == N_{new} == \emptyset$ **then**
      `printMaxClique`(M_{new});
    else
      `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
    end
    $N = N \cup \{u\}$;
    $C = C \setminus \{u\}$;
  end
end

d = 0	*M*	*C*	*N*
current		1,2,3,4,5	
new			

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u,v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u,v) ∈ E};
        N_new = {v ∈ N | (u,v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

d = 0	*M*	*C*	*N*
current		1,2,3,4,5	
new	1		

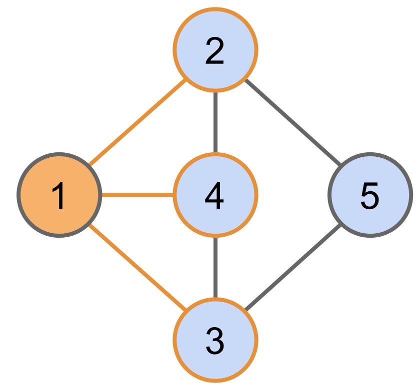
L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u,v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u,v) ∈ E};
        N_new = {v ∈ N | (u,v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 0	*M*	*C*	*N*
current		1,2,3,4,5	
new	1	2,3,4	

L

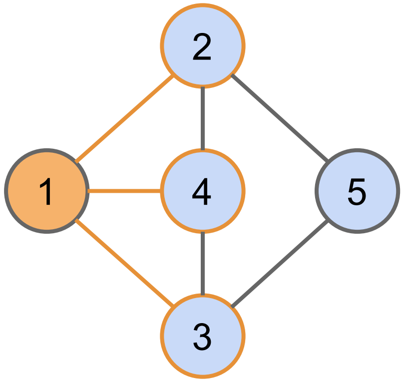
1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u,v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u,v) ∈ E};
        N_new = {v ∈ N | (u,v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 0	*M*	*C*	*N*
current		1,2,3,4,5	
new	1	2,3,4	

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

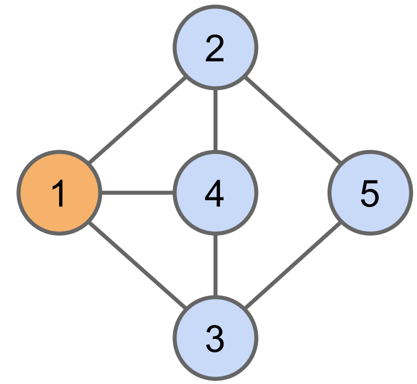
MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

$C = V$;
$M = N = \emptyset$;

Function BronKerbosch(M, C, N)
- **if** $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
  - **return**;
- **end**
- **foreach** $u \in C$ **do**
  - $M_{new} = M \cup \{u\}$;
  - $C_{new} = \{v \in C \mid (u, v) \in E\}$;
  - $N_{new} = \{v \in N \mid (u, v) \in E\}$;
  - **if** $C_{new} == N_{new} == \emptyset$ **then**
    - printMaxClique(M_{new});
  - **else**
    - BronKerbosch$(M_{new}, C_{new}, N_{new})$;
  - **end**
  - $N = N \cup \{u\}$;
  - $C = C \setminus \{u\}$;
- **end**

end

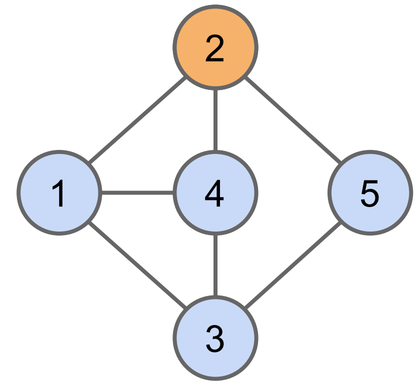


d = 0	*M*	*C*	*N*
current		1,2,3,4,5	
new	1	2,3,4	

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function `BronKerbosch`(M, C, N)
    if $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            `printMaxClique`(M_{new});
        else
            `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end

d = 1	*M*	*C*	*N*
current	1	2,3,4	
new			

L

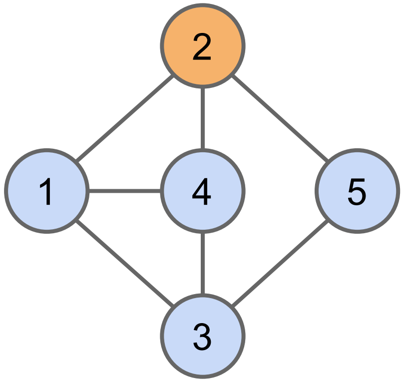
1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 1	*M*	*C*	*N*
current	1	2,3,4	
new	1,2		

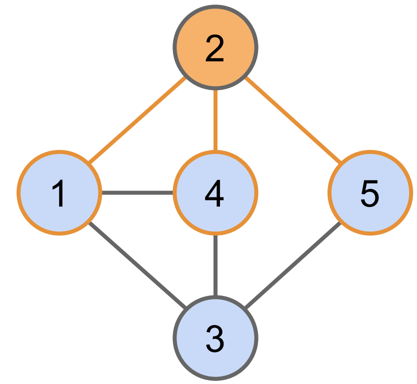
L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 1	*M*	*C*	*N*
current	1	2,3,4	
new	1,2	4	

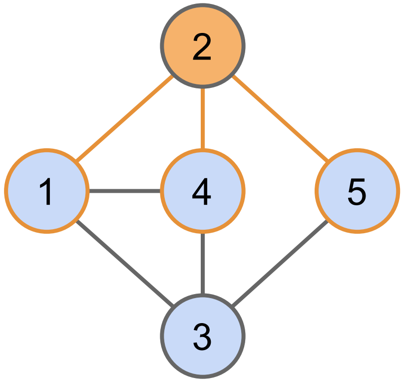
L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

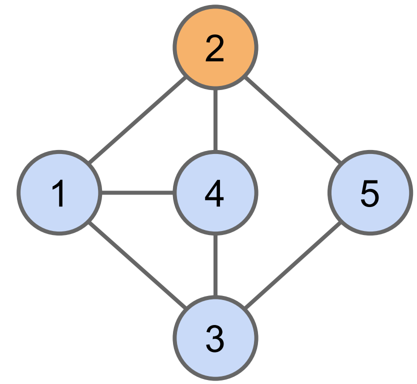


d = 1	*M*	*C*	*N*
current	1	2,3,4	
new	1,2	4	

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

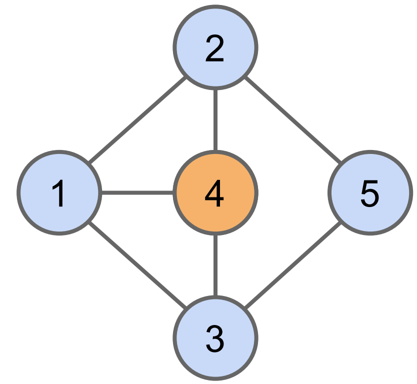
d = 1	*M*	*C*	*N*
current	1	2,3,4	
new	1,2	4	

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function `BronKerbosch`(M, C, N)
    if $\exists u \in N$ *with* $(u, v) \in E \; \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            `printMaxClique`(M_{new});
        else
            `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end



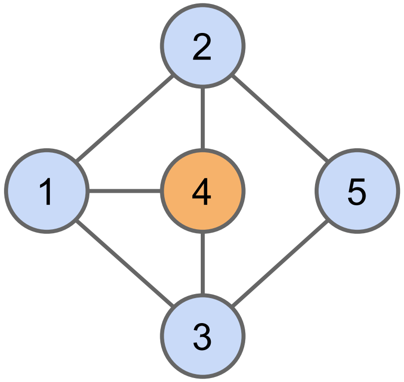
d = 2	*M*	*C*	*N*
current	1,2	4	
new			

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

$C = V$;
$M = N = \emptyset$;

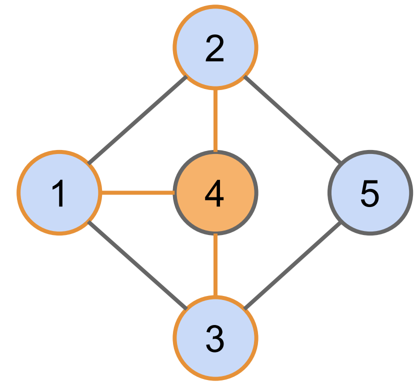
Function `BronKerbosch`(M, C, N)
    if $\exists u \in N$ *with* $(u, v) \in E \; \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            `printMaxClique`(M_{new});
        else
            `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end

***d* = 2**	*M*	*C*	*N*
current	1,2	4	
new	1,2,4		

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function $\texttt{BronKerbosch}(M, C, N)$
    if $\exists u \in N$ *with* $(u, v) \in E \; \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            $\texttt{printMaxClique}(M_{new})$;
        else
            $\texttt{BronKerbosch}(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end

***d* = 2**	***M***	***C***	***N***
current	1,2	4	
new	1,2,4		

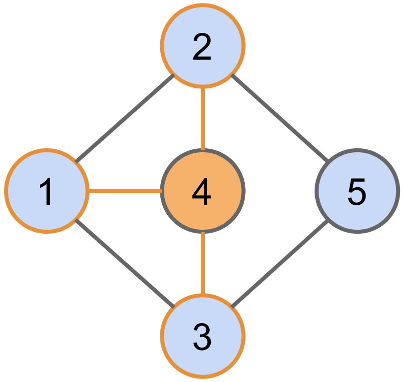
L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



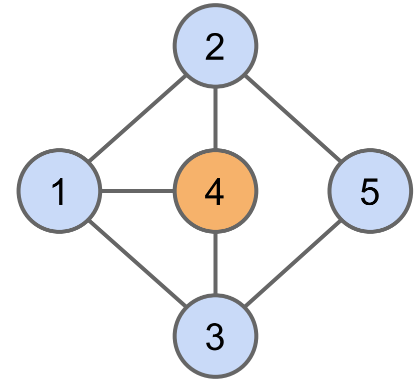
d = 2	*M*	*C*	*N*
current	1,2	4	
new	1,2,4		

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

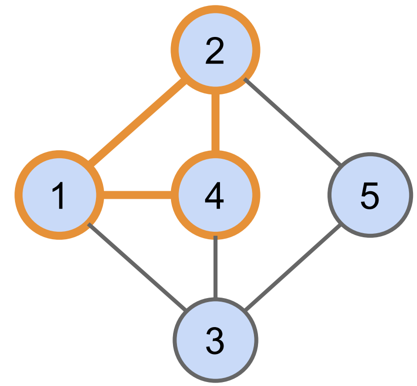
d = 2	*M*	*C*	*N*
current	1,2	4	
new	1,2,4		

L

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function `BronKerbosch`(M, C, N)
  if $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
    return;
  end
  foreach $u \in C$ **do**
    $M_{new} = M \cup \{u\}$;
    $C_{new} = \{v \in C \mid (u, v) \in E\}$;
    $N_{new} = \{v \in N \mid (u, v) \in E\}$;
    if $C_{new} == N_{new} == \emptyset$ **then**
      `printMaxClique`(M_{new});
    else
      `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
    end
    $N = N \cup \{u\}$;
    $C = C \setminus \{u\}$;
  end
end

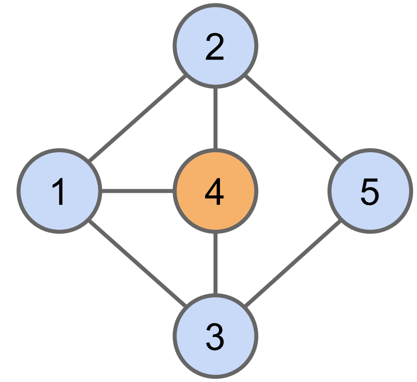


d = 2	*M*	*C*	*N*
current	1,2	4	
new	1,2,4		

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u,v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u,v) ∈ E};
        N_new = {v ∈ N | (u,v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

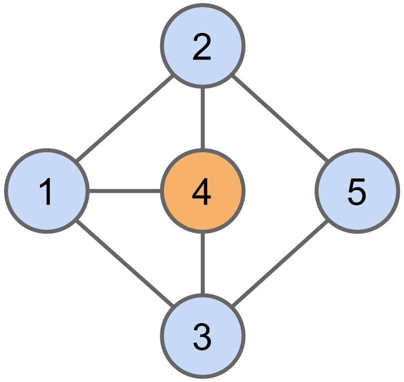
d = 2	*M*	*C*	*N*
current	1,2	4	
new	1,2,4		

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function $\texttt{BronKerbosch}(M, C, N)$
    if $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            $\texttt{printMaxClique}(M_{new})$;
        else
            $\texttt{BronKerbosch}(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end



d = 2	*M*	*C*	*N*
current	1,2		4
new	1,2,4		

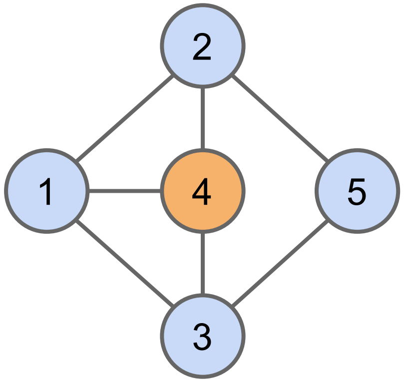
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

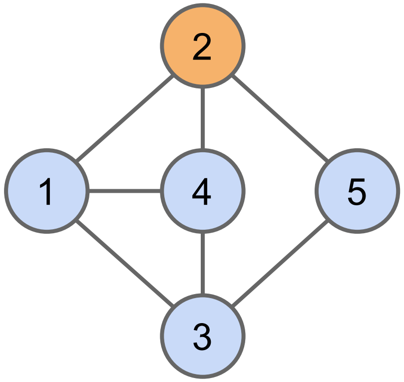


d = 2	*M*	*C*	*N*
current	1,2		4
new	1,2,4		

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

$C = V$;
$M = N = \emptyset$;

Function `BronKerbosch`(M, C, N)
  if $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
    return;
  end
  foreach $u \in C$ **do**
    $M_{new} = M \cup \{u\}$;
    $C_{new} = \{v \in C \mid (u, v) \in E\}$;
    $N_{new} = \{v \in N \mid (u, v) \in E\}$;
    if $C_{new} == N_{new} == \emptyset$ **then**
      `printMaxClique`(M_{new});
    else
      `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
    end
    $N = N \cup \{u\}$;
    $C = C \setminus \{u\}$;
  end
end

d = 1	*M*	*C*	*N*
current	1	2,3,4	
new	1,2	4	

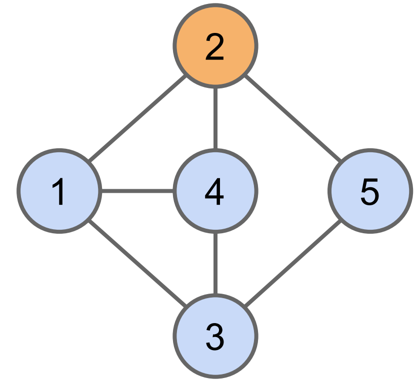
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 1	*M*	*C*	*N*
current	1	2,3,4	
new	1,2	4	

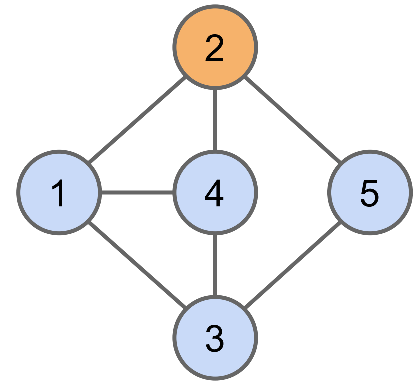
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 1	*M*	*C*	*N*
current	1	3,4	2
new	1,2	4	

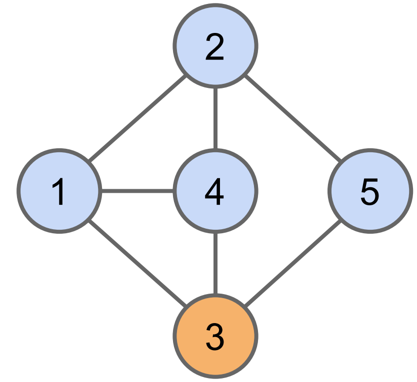
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



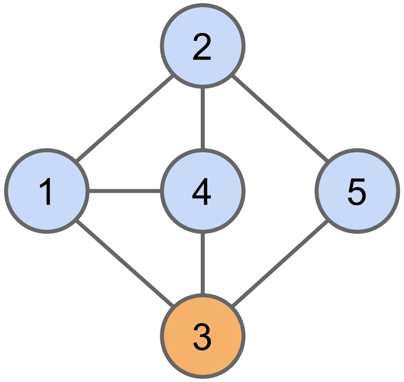
d = 1	***M***	***C***	***N***
current	1	3,4	2
new	1,2	4	

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



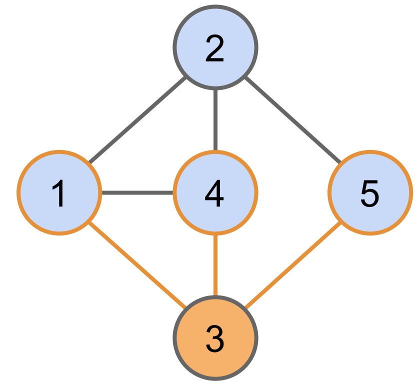
In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function `BronKerbosch`(M, C, N)
  if $\exists u \in N$ *with* $(u, v) \in E \; \forall v \in C$ **then**
    return;
  end
  foreach $u \in C$ **do**
    $\boxed{M_{new} = M \cup \{u\};}$
    $C_{new} = \{v \in C \mid (u, v) \in E\}$;
    $N_{new} = \{v \in N \mid (u, v) \in E\}$;
    if $C_{new} == N_{new} == \emptyset$ **then**
      `printMaxClique`(M_{new});
    else
      `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
    end
    $N = N \cup \{u\}$;
    $C = C \setminus \{u\}$;
  end
end

d = 1	*M*	*C*	*N*
current	1	3,4	2
new	1,3	4	

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```

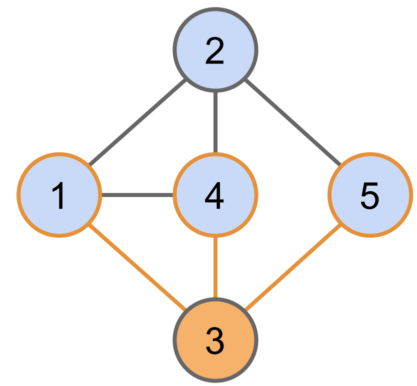
d = 1	*M*	*C*	*N*
current	1	3,4	2
new	1,3	4	

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function `BronKerbosch`(M, C, N)
    if $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            `printMaxClique`(M_{new});
        else
            `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end

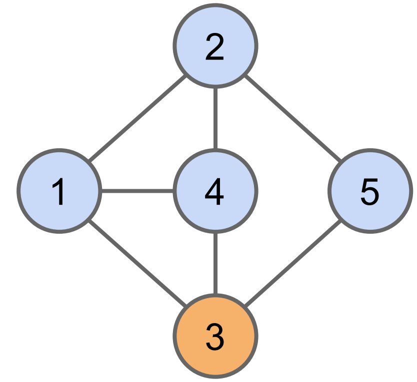
d = 1	*M*	*C*	*N*
current	1	3,4	2
new	1,3	4	

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function $\texttt{BronKerbosch}(M, C, N)$
    if $\exists u \in N$ *with* $(u, v) \in E \; \forall v \in C$ **then**
        return;
    end
    foreach $u \in C$ **do**
        $M_{new} = M \cup \{u\}$;
        $C_{new} = \{v \in C \mid (u, v) \in E\}$;
        $N_{new} = \{v \in N \mid (u, v) \in E\}$;
        if $C_{new} == N_{new} == \emptyset$ **then**
            $\texttt{printMaxClique}(M_{new})$;
        else
            $\texttt{BronKerbosch}(M_{new}, C_{new}, N_{new})$;
        end
        $N = N \cup \{u\}$;
        $C = C \setminus \{u\}$;
    end
end



d = 1	*M*	*C*	*N*
current	1	3,4	2
new	1,3	4	

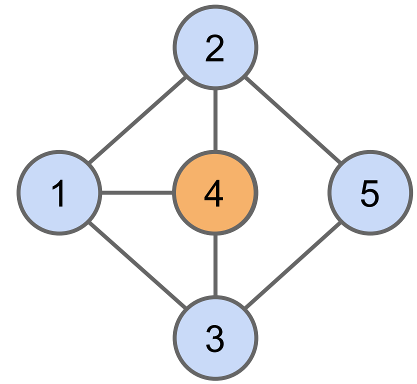
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



d = 2	*M*	*C*	*N*
current	1,3	4	
new			

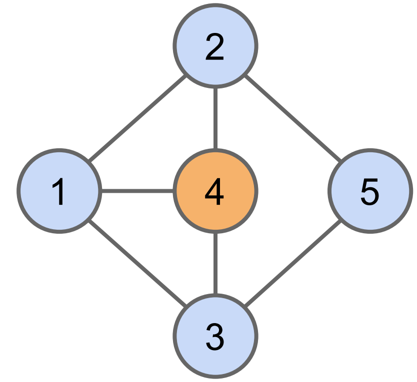
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



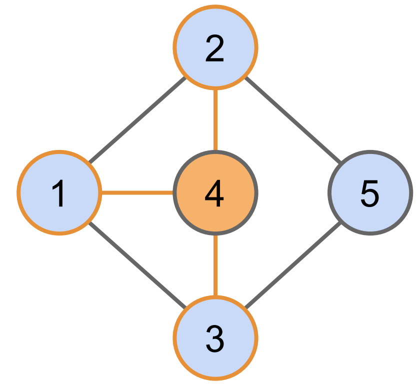
d = 2	*M*	*C*	*N*
current	1,3	4	
new	1,3,4		

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G
$C = V$;
$M = N = \emptyset$;
Function BronKerbosch(M, C, N)
  if $\exists u \in N$ *with* $(u, v) \in E \ \forall v \in C$ **then**
    return;
  end
  foreach $u \in C$ **do**
    $M_{new} = M \cup \{u\}$;
    $C_{new} = \{v \in C \mid (u, v) \in E\}$;
    $N_{new} = \{v \in N \mid (u, v) \in E\}$;
    if $C_{new} == N_{new} == \emptyset$ **then**
      printMaxClique(M_{new});
    else
      BronKerbosch$(M_{new}, C_{new}, N_{new})$;
    end
    $N = N \cup \{u\}$;
    $C = C \setminus \{u\}$;
  end
end

***d* = 2**	***M***	***C***	***N***
current	1,3	4	
new	1,3,4		

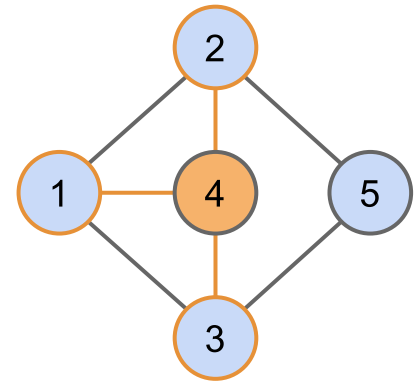
L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u, v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u, v) ∈ E};
        N_new = {v ∈ N | (u, v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



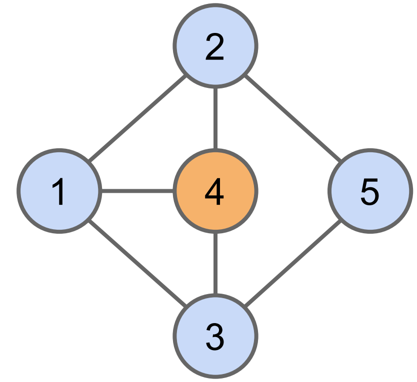
d = 2	*M*	*C*	*N*
current	1,3	4	
new	1,3,4		

L
1,2,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm



In : Graph $G = (V, E)$

Out: List L with all maximal cliques of G

$C = V$;

$M = N = \emptyset$;

Function `BronKerbosch`(M, C, N)
  if $\exists u \in N$ *with* $(u, v) \in E \ \forall v \in C$ **then**
    return;
  end
  foreach $u \in C$ **do**
    $M_{new} = M \cup \{u\}$;
    $C_{new} = \{v \in C \mid (u, v) \in E\}$;
    $N_{new} = \{v \in N \mid (u, v) \in E\}$;
    if $C_{new} == N_{new} == \emptyset$ **then**
      `printMaxClique`(M_{new});
    else
      `BronKerbosch`$(M_{new}, C_{new}, N_{new})$;
    end
    $N = N \cup \{u\}$;
    $C = C \setminus \{u\}$;
  end
end

d = 2	*M*	*C*	*N*
current	1,3	4	
new	1,3,4		

L
1,2,4

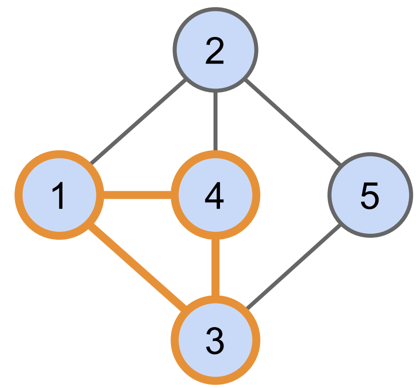
1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach

Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$
Out: List L with all maximal cliques of G

```
C = V;
M = N = ∅;
Function BronKerbosch(M, C, N)
    if ∃u ∈ N with (u,v) ∈ E ∀v ∈ C then
        return;
    end
    foreach u ∈ C do
        M_new = M ∪ {u};
        C_new = {v ∈ C | (u,v) ∈ E};
        N_new = {v ∈ N | (u,v) ∈ E};
        if C_new == N_new == ∅ then
            printMaxClique(M_new);
        else
            BronKerbosch(M_new, C_new, N_new);
        end
        N = N ∪ {u};
        C = C \ {u};
    end
end
```



***d* = 2**	***M***	***C***	***N***
current	1,3	4	
new	1,3,4		

L
1,2,4 1,3,4

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

In : Graph $G = (V, E)$

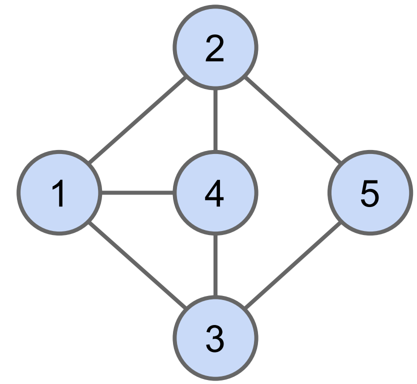
Out: List L with all maximal cliques of G

$C = V$;

$M = N = \emptyset$;

Function BronKerbosch(M, C, N)
- **if** $\exists u \in N$ *with* $(u, v) \in E\ \forall v \in C$ **then**
  - **return**;
- **end**
- **foreach** $u \in C$ **do**
  - $M_{new} = M \cup \{u\}$;
  - $C_{new} = \{v \in C \mid (u, v) \in E\}$;
  - $N_{new} = \{v \in N \mid (u, v) \in E\}$;
  - **if** $C_{new} == N_{new} == \emptyset$ **then**
    - printMaxClique(M_{new});
  - **else**
    - BronKerbosch$(M_{new}, C_{new}, N_{new})$;
  - **end**
  - $N = N \cup \{u\}$;
  - $C = C \setminus \{u\}$;
- **end**

end



d = ...	*M*	*C*	*N*
current			
new			

L
1,2,4 1,3,4 2,5 3,5

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7

MCS: Maximum Clique Approach
Bron-Kerbosch Algorithm

- Enumerates **all maximal cliques**
- Runtime exponential in the number of nodes
- Also used for other cheminformatics problems
  - Pharmacophore matching (discussed in a later lecture)
- Popularity of the algorithm is due to its trivial implementation
- Much more advanced algorithms exist
  - C.f. second DIMACS Challenge [2]
  - However, they are often very tricky to implement
- Efficient algorithms for approximate clique detection often yield very good results as well

1. Bron C. and Kerbosch J. (1973) *Commun. ACM*, 16, 575-7
2. *Johnson D.S. and Trick M.A. (1996) Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, Bellcore and the American Mathematical Society*

Motivation

- **Maximum Common Substructure: MCS**
- The largest common substructure of two molecules
- Very important concept in cheminformatics
  - Also used in other molecular science areas
  - An overview can be found in Ehrlich and Rarey (2011) [1]

- Two problem variants have cheminformatic use cases:
  1. Two molecule case
  2. **Multiple molecule case**

1. Ehrlich H.C. and Rarey M. (2011) *WIREs Comput. Mol. Sci.*, 1, 68-79, 10.1002/wcms.5

Motivation
Multiple Molecule Case

- We discussed substructure searching in detail
- Question:

  What are interesting substructures to search for?
- Possible answer:

  Molecules that are structurally related to known actives
- Pharmacologically active compounds

  Given a set of active molecules, e.g. identified by HTS,

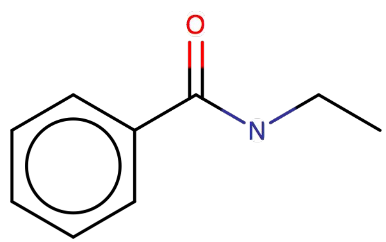
  identify largest common substructure and search for molecules

  that also contain it in order to be tested (SPP!).

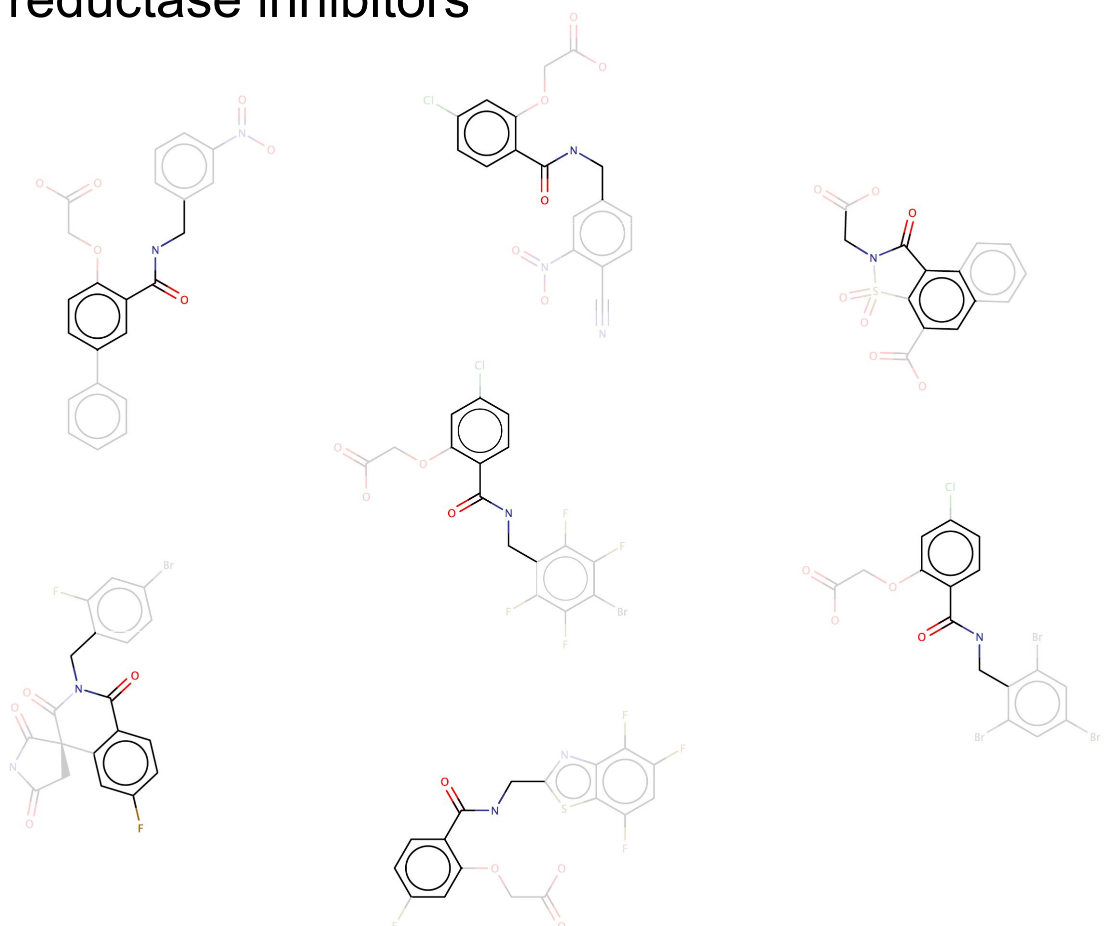
Motivation

Multiple Molecule Case

- Example: aldose reductase inhibitors



MCS



Motivation
Multiple Molecule Case

- **Given**: **set of compounds** with known property
  - Desired pharmacologic activity, identified e.g. by HTS
  - Same smell, desired material property, ...
- **Goal**: find new compounds possessing that property
- According to the **SPP** we should try to find structurally related compounds and test those

- **Approach**:
  Identify MCS of given compounds and use it as a query for a substructure search

Maximum Common Substructure

Multiple Molecule Case

- Maximum clique approach not easily extendible
- Compatibility graph size grows exponentially
- Assume *n* molecular graphs of size *m*

  ⇒ Worst case size of compatibility graph [1]:

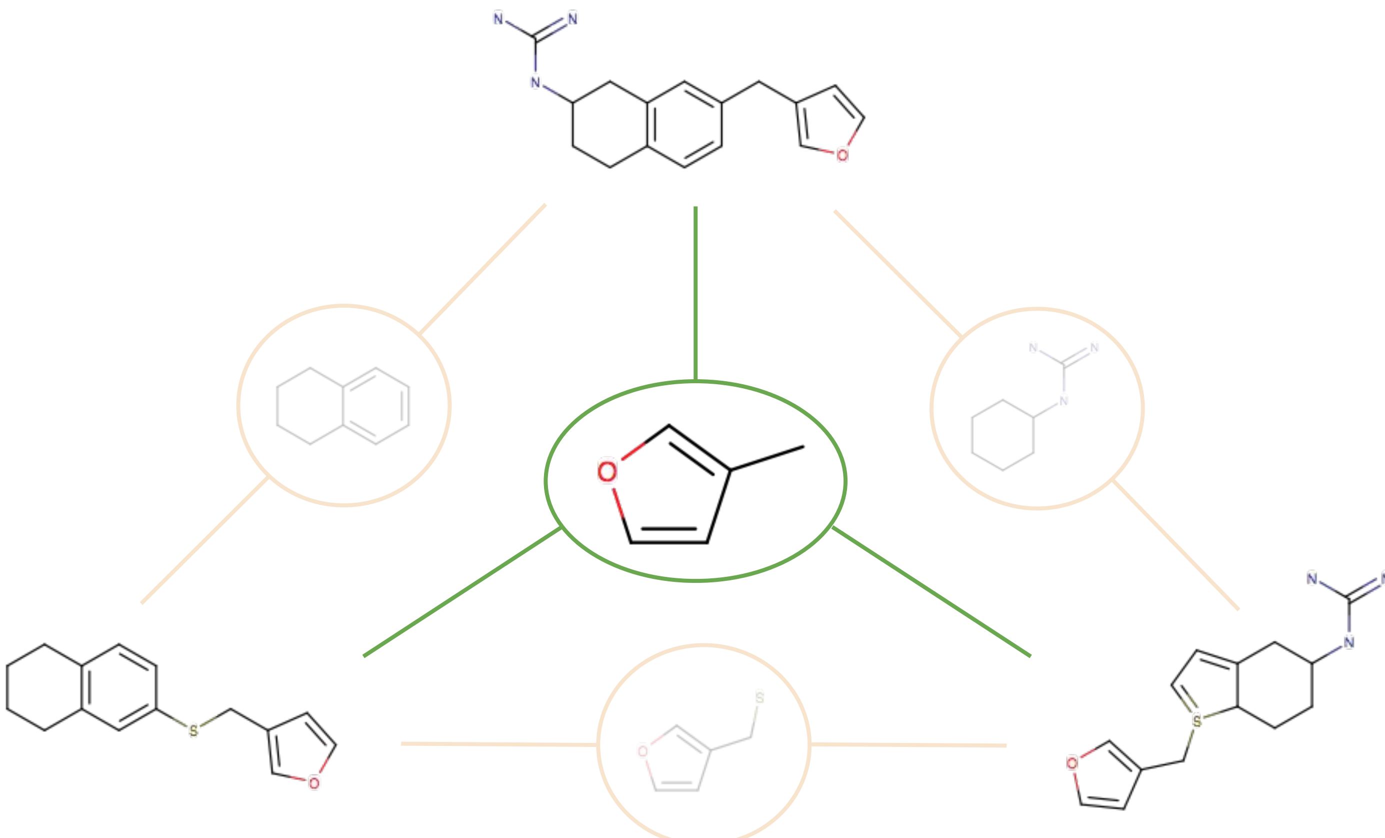
$$\prod_{i=1}^{n} m_i^2$$

- **Efficient clique detection is infeasible here**

1. Brint A.T. and Willett P. (1987) *J. Chem. Inf. Comput. Sci.*, 27, 152-8

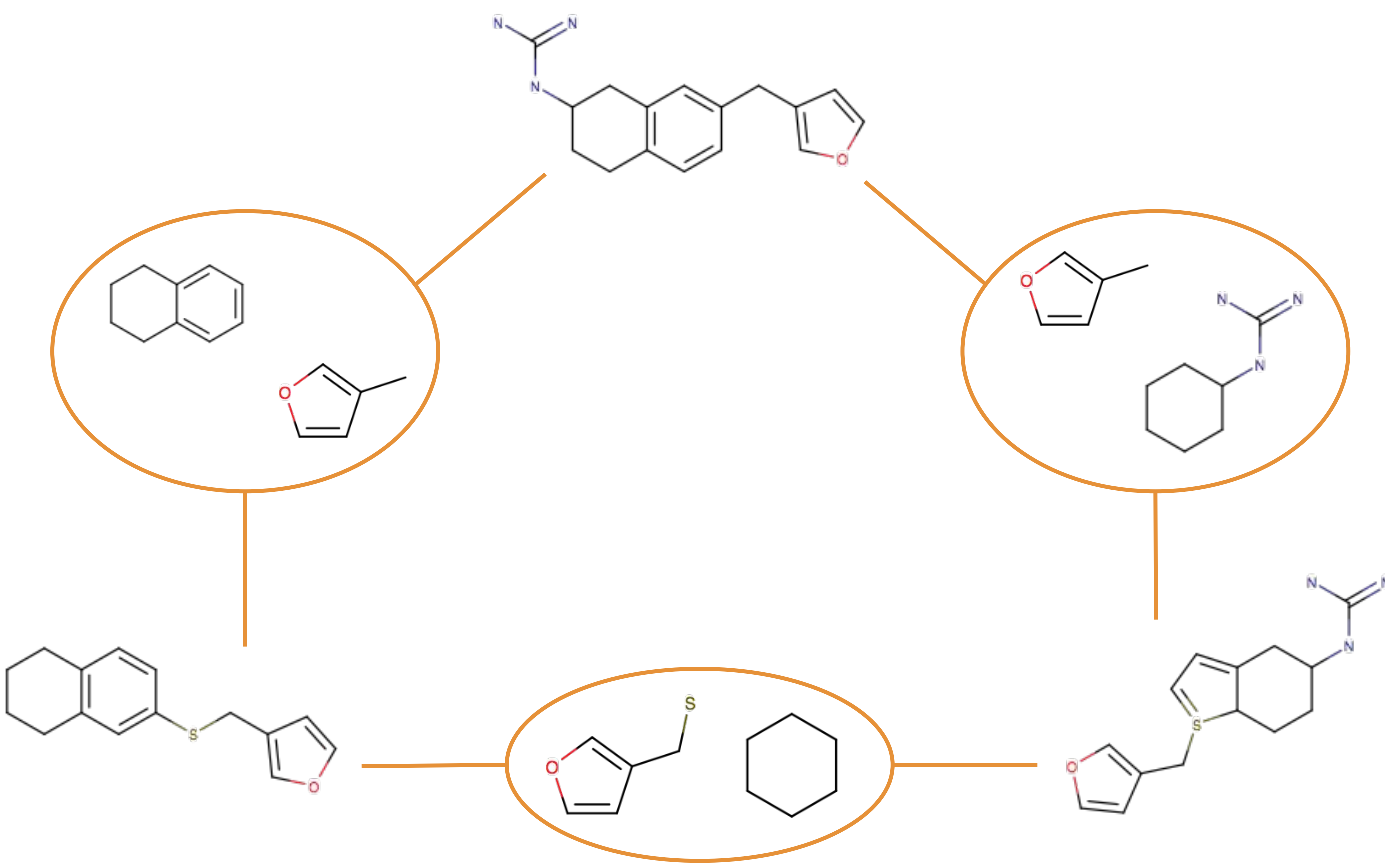
Maximum Common Substructure
Multiple Molecule Case

- Pairwise MCS detection is not sufficient



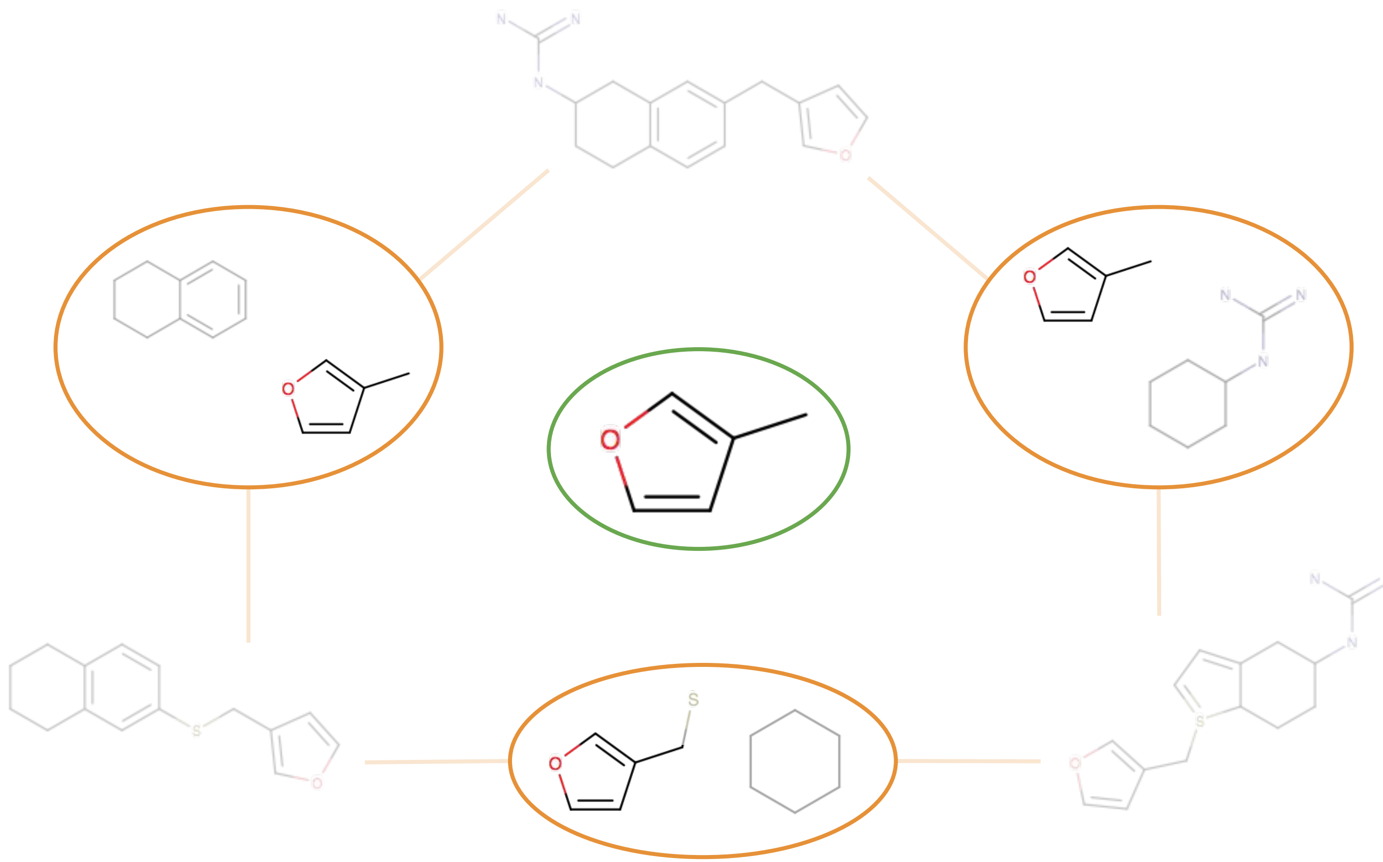
Maximum Common Substructure
Multiple Molecule Case

- Pairwise **Maximal Common Substructures (mCS)** detection



Maximum Common Substructure
Multiple Molecule Case

- **MCS** is contained in the set of all pairwise intersected **mCS**



Maximum Common Substructure
Multiple Molecule Case

- All **mCS** are enumerated by **Bron-Kerbosch**
- Idea:
  - Given a set of *n* molecules
  - Select a pivot molecule
  - Calculate mCS for pivot molecule and all other molecules
  - Iteratively intersect mCS sets
- Possible outcomes:
  - An empty set of intersections, thus no **MCS**
  - A **list of MCS candidates**
- Exemplary algorithm can look like the following

Maximum Common Substructure

Multiple Molecule Case: Ingredients

- `selectPivotMolecule(`M`)`:

  Select a pivot molecule from all molecules

- `getMaximalCS(`m_i`,`m_j`)`:

  Return all mCS for molecule pair m_i and m_j as substructures of m_j .

  Use for example the clique approach with Bron-Kerbosch

- `getLargestSubstructure(`S`)`:

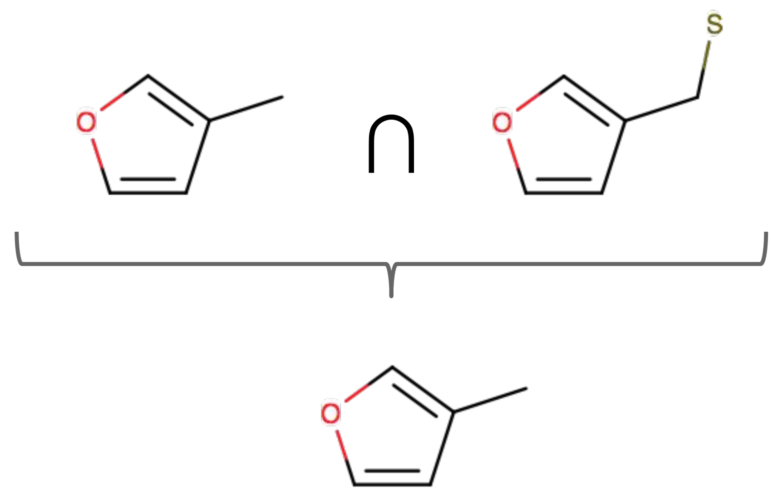
  Return largest substructure from S with respect to its number of atoms

Maximum Common Substructure
Multiple Molecule Case: Algorithm

In : Molecular Graphs $M = \{m_1, ..., m_n\}$

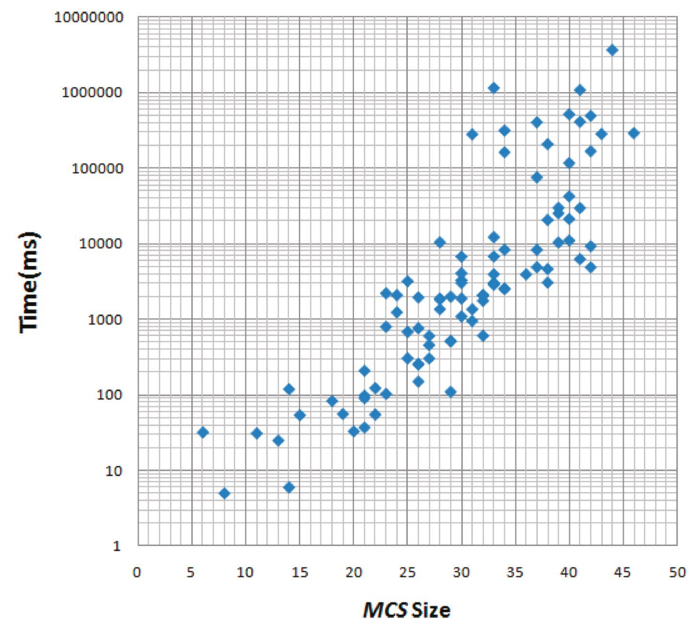
Out: MCS, the maximum common substructure of molecules in M

```
begin
    MCS = ∅;
    m_P = selectPivotMolecule(M);
    M = M \ {m_P};
    S = getMaximalCS(m_1, m_P);
    foreach m_i ∈ M with 2 ≤ i ≤ n − 1 do
        if S == ∅ then
            return;
        end
        S_new = getMaximalCS(m_i, m_P);
        S_tmp = ∅;
        foreach s ∈ S_new do
            foreach t ∈ S do
                S_tmp = S_tmp ∪ (s ∩ t);
            end
        end
        S = S_tmp
    end
    MCS = getLargestSubstructure(S);
end
```



Maximum Common Substructure
MultiMCS

- **How to select the pivot molecule?**
- Obvious choice: smallest molecule
  - Reduction of compatibility graph size
  - Speeding up mCS calculations
- This is still pretty time consuming
  - Figure shows benchmarks for 3-molecule instances



1. Hariharan R. et al. (2011) *J. Chem. Inf. Model.*, 51, 788-806

Maximum Common Substructure
MultiMCS

- Hariharan et al. presented an efficient approach [1]: **MultiMCS**
- Divide-and-conquer strategy

- **Key ideas**:
  - Split pivot molecule m_P into small fragments $\{m_{P1}, \ldots, m_{Pn}\}$
    - Splitting by removal of chain bonds
  - Solve mCS task for all fragments against all other molecules
  - Restore original mCS set for complete molecule pairs

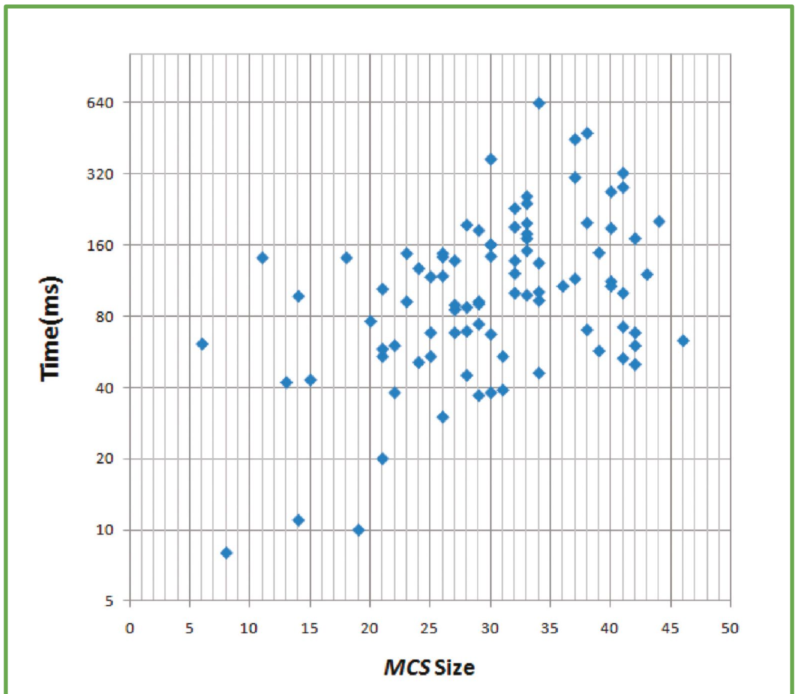
- Choice of pivot molecule:

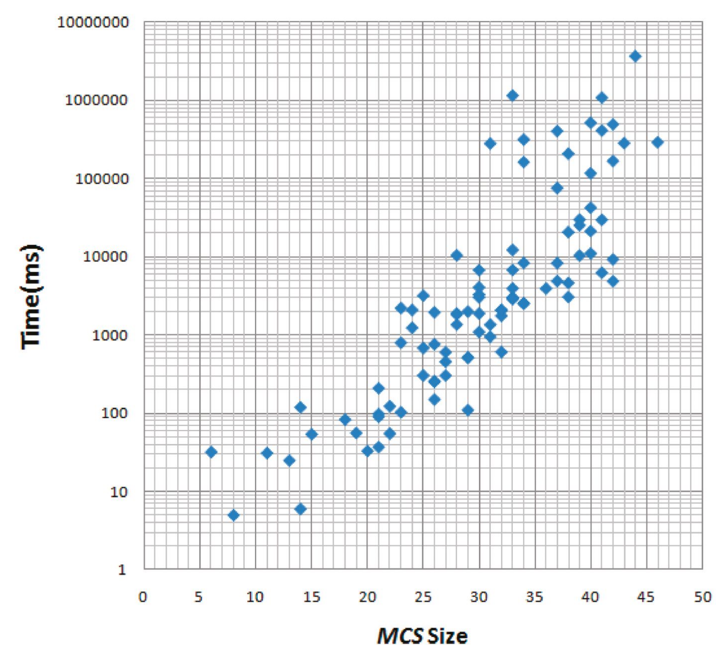
  Molecule that can best be decomposed into small fragments

1. Hariharan R. et al. (2011) *J. Chem. Inf. Model.*, 51, 788-806

Maximum Common Substructure
MultiMCS

- Hariharan et al. presented an efficient approach [1]: **MultiMCS**
- Significant speedup over naive approach





1. Hariharan R. et al. (2011) *J. Chem. Inf. Model.*, 51, 788-806

Summary

- Maximum Common Substructure (MCS)
- Variant of Maximum Common Subgraph Isomorphism
- Reaction mapping of educts and products: pairwise MCS
- MCS problem reduced to into search for maximum clique
- Bron-Kerbosch algorithm calculates all maximal cliques
- Common structural property of active compounds: multiple MCS
- MCS for multiple molecules not trivial
- Select pivot molecule and pairwise mCS problem
- Intersecting the mCS lists yields MCS
- MultiMCS employs a very efficient divide-and-conquer approach

Text Books:

- GJ — Garey M. and Johnson D.S., W. H. Freeman & Co., New York, 1979
  Computers and Intractability: A Guide to the Theory of NP-Completeness
- GE — Gasteiger J. and Engel T. (Eds.), 1st Ed., Wiley-VCH, 2003
  Chemoinformatics - A Textbook
- KA — Kerber A. et al.
  Mathematical Chemistry and Chemoinformatics, De Gruyter, 2014

Acknowledgments:

- 2D structure drawings were generated with ChemAxon **MarvinSketch**
  - `https://www.chemaxon.com/products/marvin/marvinsketch`
- 3D structures were generated with **BALLView**
  - `http://www.ball-project.org`
  - Hildebrandt A. et al. (2010) *BMC Bioinformatics*, 11, 531
  - Moll A. et al. (2006) *Bioinformatics*, 22, 365-6