Structure Searching

Problem: Identify a particular molecule and associated information in a chemical database.

The solution boils down to solve the graph isomorphism problem.

Two graphs are isomorphic if there exists a one-to-one mapping between the atoms, which preserves the connectivity of the graphs.

If the two graphs are the same the one-to-one mapping is called automorphism.

Automorphic atoms are equivalent with respect to the constitutional symmetry.

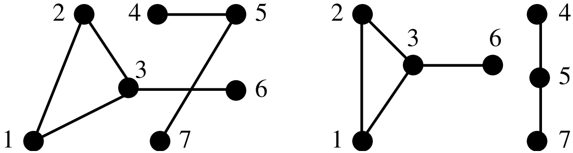
Graph theoretical algorithms for (sub)graph isomorphism are well established but **usually too slow** for typical chemical databases.

A filtering step rapidly eliminates molecules that can not match.

Primer: Graphs

$$G = (V, E) \quad \text{where} \quad e_i = (v_i, v_j)$$

- A graph is a tupel of two sets, the vertex set and the edge set.
- The edge set members are tupels of vertex set members.
- Graphs preserve neighborhood relations.



$$\begin{aligned}
\text{vertex set } V \quad &= \quad \{1, 2, 3, 4, 5, 6, 7\} \\
\text{edge set } E \quad &= \quad \{(1, 2), (2, 3), (1, 3), (3, 6), (4, 5), (5, 7) \\
&\qquad (2, 1), (3, 2), (3, 1), (6, 3), (5, 4), (7, 5)\}
\end{aligned}$$

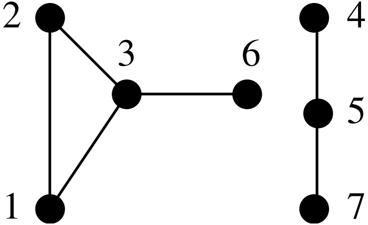
Chemical graphs are **undirected simple graphs**, which are loop-free and do not contain multiple edges.

Representation of Graphs

The adjacency matrix $\mathbf{A} = \mathbf{A}(G)$ of graph G with N vertices is the square $N \times N$ symmatric matrix whose elements $[\mathbf{A}]_{ij}$ are defined as

$$[\mathbf{A}]_{ij} = \begin{cases} 1 \text{ if } i \neq j \text{ and } e_{ij} \in E(G) \\ 0 \text{ if } i = j \text{ or } e_{ij} \notin E(G) \end{cases}$$

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



Algorithms for Chemical Graphs

The most common tasks in cheminformatics involve only three classes of algorithms which operate on chemical graphs:

1. **Canonical coding problem** i.e. the generation of a unique representation of a chemical compound.
2. **Automorphism partitioning problem** (= constitutional symmetry problem) i.e. the detection of equivalent atoms and bonds in a chemical compound.
3. **Graph isomorphism problem** i.e. the determination if two connection tables represent the same chemical compound.

Ullmann algorithm

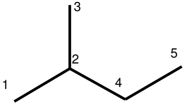
- One of the most efficient subgraph isomorphism methods.
- Backtracking algorithm with relaxation.
- Operates on the adjacency matrix.
- Produces all matching matrices (= subgraph isomorphisms).

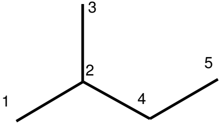
$$M \cdot (M \cdot H)^T = Q$$

A matching matrix M fulfills the following conditions:

1. Each row contains just one element equal to “1”.
2. Each column contains no more than one element equal to “1”.

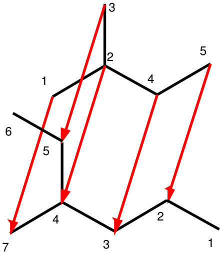
Ullmann algorithm





$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

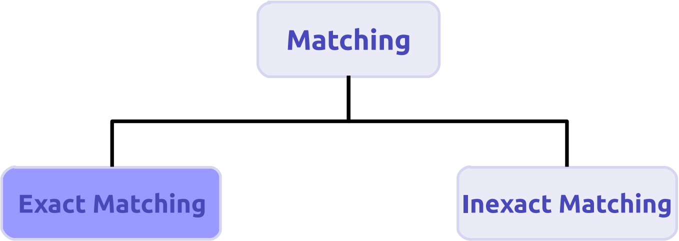
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ullmann JR (1976), **An Algorithm for Subgraph Isomorphism.** *J Assoc Comput Mach* **23**:31-42 | DOI:10.1145/321921.321925.

Exact Matching



Exact Matching Complexity

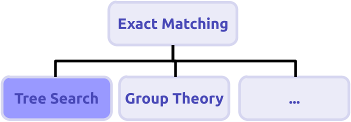
Graph Isomorphism : In NP, but unknown if P or NP complete.

Subgraph Isomorphism, Monomorphism, MCS... : NP complete.

There exists algorithms for special graphs with polynomial runtime.

Exact Matching

Tree-Search approach



Basic Idea

- Iteratively expand partial match by adding new pairs of matched nodes.
- The pair is chosen using some necessary conditions.
- Prune unfruitful search paths.
- If no further vertex pairs may be added due to constraint, undo last additions (backtracking)
- Algorithm stops if match has been found or all matchings that satisfy the constraints has been tried.

Exact Matching

Ullmann's Algorithm [J.R. Ullmann 1976]

- Tree-Search algorithm (Depth-Search-First)
- Uses adjacency matrices and additional constraints for matching and pruning.
- Application for graph isomorphism, subgraph isomorphism and monomorphism, also for MCS problem

Exact Matching

Ullmann's Algorithm

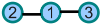
- Given: Two graphs $G_A(V_A, E_A)$ and $G_B(V_B, E_B)$ and their adjacency matrices: A and B
- Idea: $n = |V_a|$, $m = |V_b|$, $n \times m$ permutation matrix M with following form:
  - M contains only '0' and '1'
  - Exact one '1' in each row
  - Not more than one '1' in each column
- Permutate adjacency matrix B by multiplying it with M, and compare adjacency.

Exact Matching

Ullmann's Algorithm

- $M \times B$: Move row j to row i $\forall M_{ij} = 1$

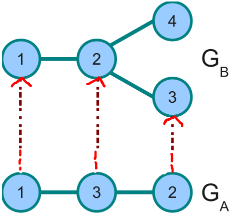
$$\underset{M = M^T}{\begin{bmatrix} 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \text{ x } \underset{B = B^T}{\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$



- $(MB)^T$: Move column j to column i
- $M(MB)^T$: Move column j to column i and row j to row i

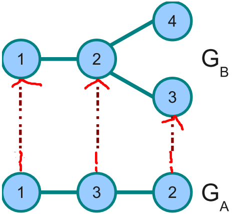
Exact Matching

Ullmann's Algorithm



Exact Matching

Ullmann's Algorithm



0	1	0	0
1	0	1	1
0	1	0	0
0	1	0	0

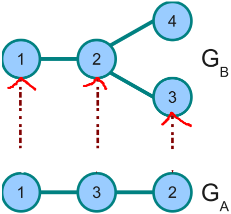
$B=B^T$

1	0	0	0
0	0	**1**	0
0	**1**	0	0

M

Exact Matching

Ullmann's Algorithm



$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$B=B^T$

$$\begin{bmatrix} \textcircled{1} & 0 & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 \\ 0 & \textcircled{1} & 0 & 0 \end{bmatrix}$$

M

$$M(MB)^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \right)^T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} = C$$

Exact Matching

Ullmann's Algorithm

Creating pairs of nodes by exchanging rows and columns (renaming).

Adjacency condition

Let $C = M(MB)^T$,

monomorphism

A is a ~~(subgraph-) isomorphism~~ iff

$$A_{ij} = 1 \Rightarrow C_{ij} = 1 \forall i, j$$

How do we get M?

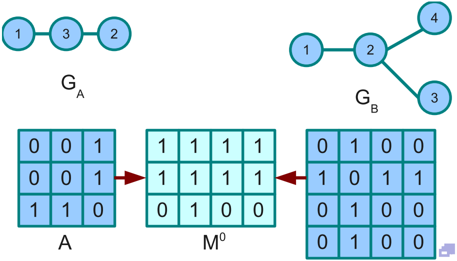
Exact Matching

Ullmann's Algorithm

- Build Startmatrix M^0 by setting all values to 1 (allow all permutations)
- Set values to 0 for all M^0_{ij} where $deg(B_j) < deg(A_i)$ (remove impossible permutations)

$$M^0_{ij} = \begin{cases} 1 & if \quad deg(B_j) \geq deg(A_i) \\ 0 & \text{otherwise} \end{cases} \quad , \forall i, j$$

- Generate systematically permutation matrices M^d.



0	0	1
0	0	1
1	1	0

A

1	1	1	1
1	1	1	1
0	1	0	0

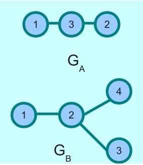
M^0

0	1	0	0
1	0	1	1
0	1	0	0
0	1	0	0

Exact Matching

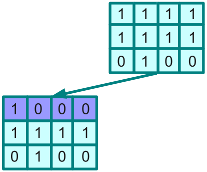
Ullmann's Algorithm

1	1	1	1
1	1	1	1
0	1	0	0



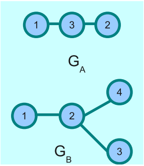
Exact Matching

Ullmann's Algorithm



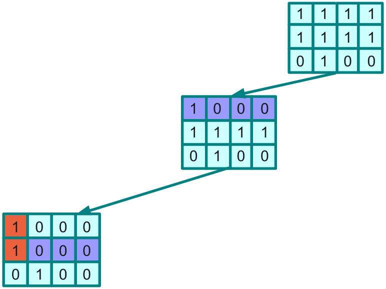
$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



Exact Matching

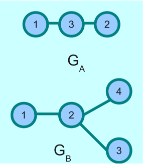
Ullmann's Algorithm



1	1	1	1
1	1	1	1
0	1	0	0

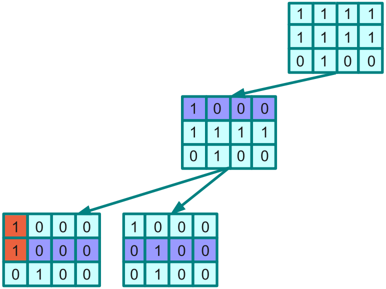
1	0	0	0
1	1	1	1
0	1	0	0

1	0	0	0
1	0	0	0
0	1	0	0



Exact Matching

Ullmann's Algorithm

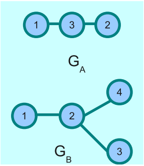


1	1	1	1
1	1	1	1
0	1	0	0

1	0	0	0
1	1	1	1
0	1	0	0

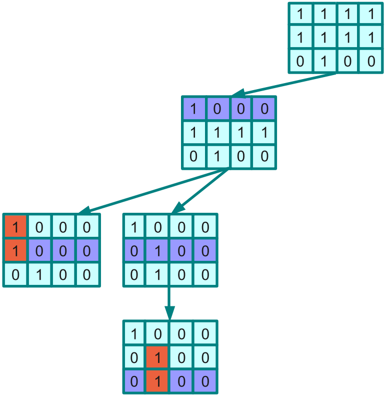
1	0	0	0
1	0	0	0
0	1	0	0

1	0	0	0
0	1	0	0
0	1	0	0



Exact Matching

Ullmann's Algorithm



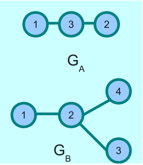
1	1	1	1
1	1	1	1
0	1	0	0

1	0	0	0
1	1	1	1
0	1	0	0

1	0	0	0
1	0	0	0
0	1	0	0

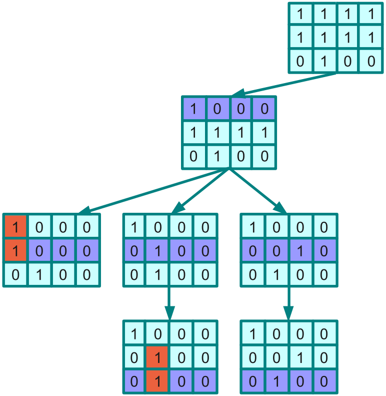
1	0	0	0
0	1	0	0
0	1	0	0

1	0	0	0
0	1	0	0
0	1	0	0



Exact Matching

Ullmann's Algorithm

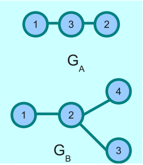


$$\begin{bmatrix}1&1&1&1\\1&1&1&1\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\1&1&1&1\\0&1&0&0\end{bmatrix}$$

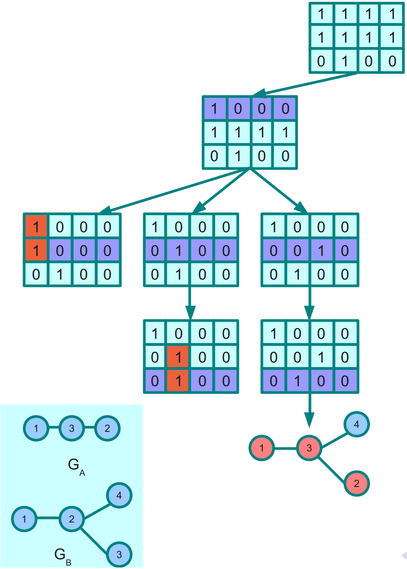
$$\begin{bmatrix}1&0&0&0\\1&0&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix}$$



Exact Matching

Ullmann's Algorithm



$$\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{matrix}$$

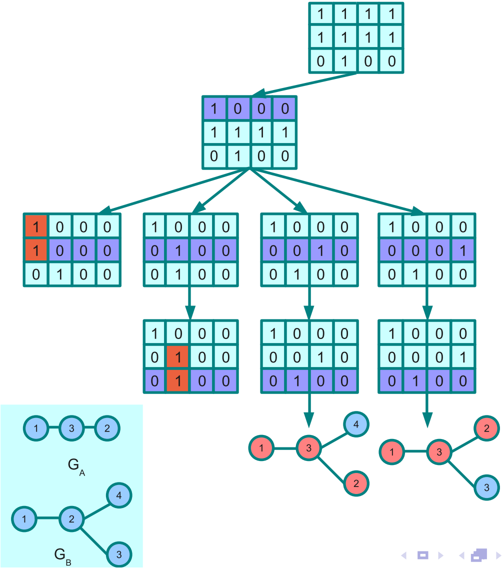
$$\begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Exact Matching

Ullmann's Algorithm



$$\begin{bmatrix}1&1&1&1\\1&1&1&1\\0&1&0&0\end{bmatrix}$$

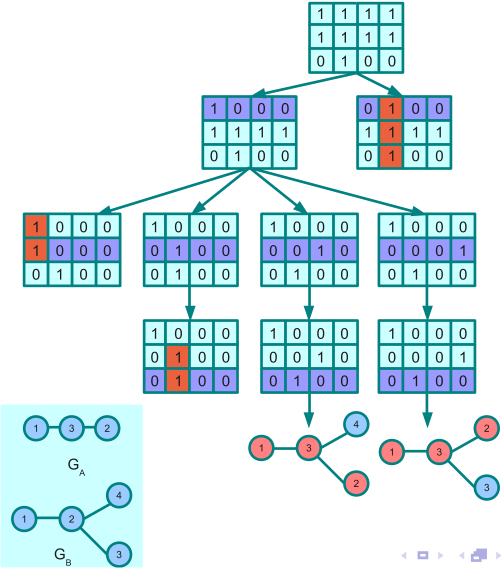
$$\begin{bmatrix}1&0&0&0\\1&1&1&1\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\1&0&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\end{bmatrix}$$

Exact Matching

Ullmann's Algorithm



$$\begin{bmatrix}1&1&1&1\\1&1&1&1\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\1&1&1&1\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}0&1&0&0\\1&1&1&1\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\1&0&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\end{bmatrix}$$

Exact Matching

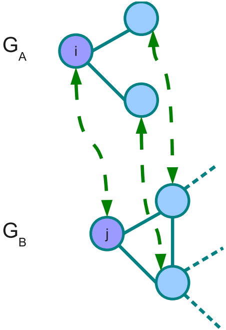
Ullmann's Algorithm V2

Refinement Procedure:

- For all neighbours in A there must be proper neighbours in B.
- Formally:

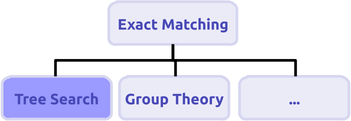
$$\forall k(A_{ik} = 1 \Rightarrow \exists p(M_{kp}B_{pj} = 1))$$

- Set $M^d_{ij} = 0$ where conditions are not complied.



Exact Matching

Tree-Search approaches

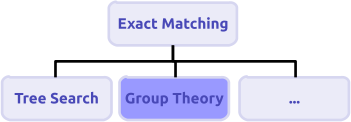


VF and VF2 algorithm

- Application for isomorphism and subgraph isomorphism
- VF algorithm defines a heuristic based on the analysis of the sets of nodes adjacent to the ones already considered in the partial mapping.

Exact Matching

Group theory approach



McKay's Nauty

- Nauty - No automorphisms, yes?
- Application for isomorphism only
- It uses the property that the canonical labeling for isomorph graphs is identical.
- It constructs the automorphism group of each of the input graphs and derives a canonical labeling

Structure Searching

Problem: Identify a particular molecule and associated information in a chemical database.

The solution boils down to solve the graph isomorphism problem.

Two graphs are isomorphic if there exists a one-to-one mapping between the atoms, which preserves the connectivity of the graphs.

If the two graphs are the same the one-to-one mapping is called automorphism.

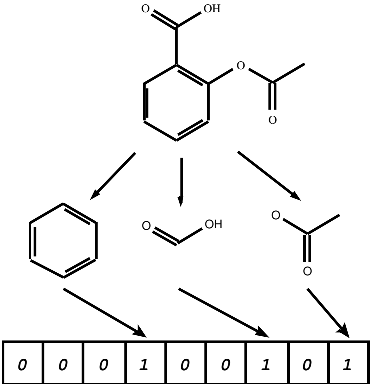
Automorphic atoms are equivalent with respect to the constitutional symmetry.

Graph theoretical algorithms for (sub)graph isomorphism are well established but **usually too slow** for typical chemical databases.

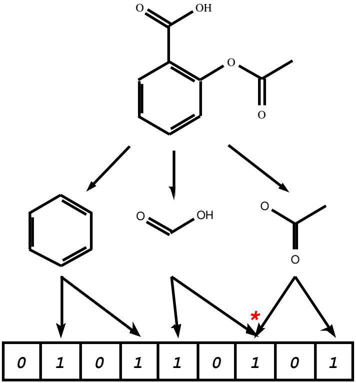
A filtering step rapidly eliminates molecules that can not match.

Fragment Information

Fragment Code



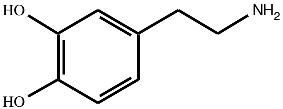
Hash Fingerprints

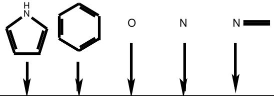


- Bitstrings depend on the choice of the fragments.
- Bitstrings are ambiguous.
- Bitstrings can be compared and manipulated very rapidly.

Pre-screening example

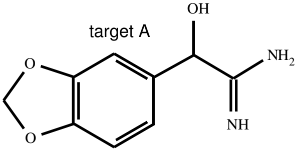
query





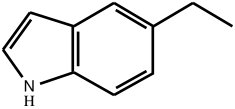
0	1	1	1	0

target A



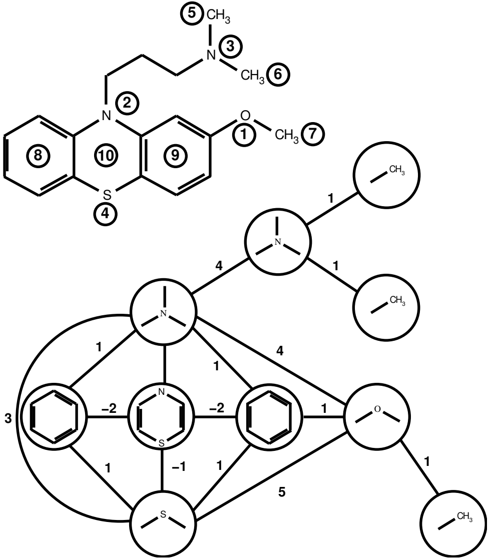
0	1	1	1	1

target B



1	1	0	1	0

Descriptor Center Connection Graphs (DCCG)



- Reduced graph representation (10 vertices 16 edges).
- Preserves topological information.
- Biology-oriented fragment description.
- Level of generalization can be controlled.
- Good for pharmacophore based virtual screening.

Avidon VV et al (1982), **Structure-activity relationship oriented languages for chemical structure representation**, *J Chem Inf Comput Sci*, **22**:207-214 | DOI:10.1021/ci00036a006

Algorithms for Chemical Graphs

The most common tasks in cheminformatics involve only three classes of algorithms which operate on chemical graphs:

1. **Canonical coding problem** i.e. the generation of a unique representation of a chemical compound.
2. **Automorphism partitioning problem** (= constitutional symmetry problem) i.e. the detection of equivalent atoms and bonds in a chemical compound.
3. **Graph isomorphism problem** i.e. the determination if two connection tables represent the same chemical compound.

Canonical Numbering

A molecular graph

$$G = G(V, E)$$

consists of a non-empty set V of vertices representing atoms and a set E of edges representing chemical bonds.

A labeling Lb of a graph G composed of N vertices consist of a one-to-one mapping

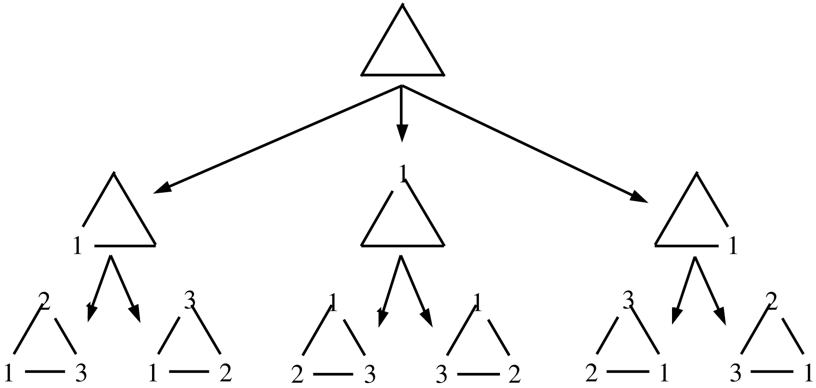
$$Lb : V(G) \to \{1, 2, \ldots, N\}$$

The integer $Lb(v) \in \{1, 2, \ldots, N\}$ assigned to a vertex $v \in V(G)$ is called a label of the vertex v.

For a graph G with N vertices there exist $N!$ permutation labelings.

Graph Labeling

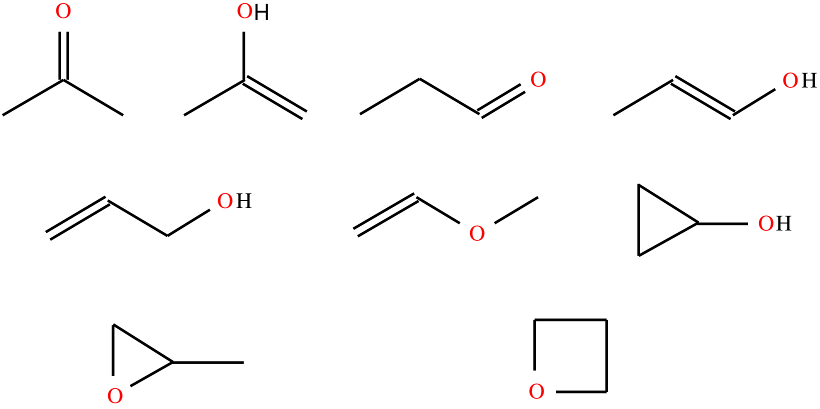
The tree representation of the $3! = 6$ permutation labelings of cyclopropane.



The tree of permutation labelings can be explored by breadth-first or depth-first order.

Constitutional Isomeres and Isomorphism

The empiric formula C_3H_6O can be expressed by the following nine structural diagrams:



Generation of all structural isomeres from an empiric formula is an important task in automatic structure elucidation.

Constitutional Symmetry of Graphs

Consider two graphs

$$G = (V, E) \text{ and } G' = (V', E') \quad \text{with } |V| = |V'|$$

and a mapping $m : V \rightarrow V'$ which assigns each vertex $v \in V$ a vertex $v' \in V'$ in such a way that if $v_i \neq v_j$ then $m(v_i) \neq m(v_j)$.

The two graphs G and G' are isomorphic if there exists a mapping $m : V \rightarrow V'$ which **preserves the adjacency** of vertices.

$$e_{ij} \in E \text{ with } v_k = m(v_i) \text{ and } v_l = m(v_j) \implies e_{kl} \in E'$$

An isomorphism of a graph with itself is called an automorphism and can be represented by a permutation matrix $\mathbf{P}$

$$\mathbf{P} = \begin{pmatrix} 1 & 2 & 3 & \dots & i & \dots & N \\ p_1 & p_2 & p_3 & \dots & p_i & \dots & p_N \end{pmatrix}$$

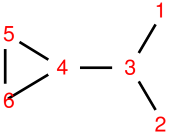
Automorphism group $Aut(G)$

The automorphism group $Aut(G)$ describes **all** symmetry properties of a graph and satisfies the folowing conditions:

1. For any two permutations $\mathbf{A}, \mathbf{B} \in Aut(G)$ there exists a unique element $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ with $\mathbf{C} \in Aut(G)$.
2. The operations respect the associative law: $\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} \forall \mathbf{A}, \mathbf{B}, \mathbf{C} \in Aut(G)$.
3. For every permutation $\mathbf{A} \in Aut(G)$ there exists an inverse permutation $\mathcal{A}^{-1} \in Aut(G)$ such that $\mathbf{A} \otimes \mathbf{A}^{-1} = \mathbf{A}^{-1} \otimes \mathbf{A} = \mathbf{E}$.
4. The set $Aut(G)$ contains a unique permutation $\mathbf{E}$ such that $\mathbf{A} \otimes \mathbf{E} = \mathbf{E} \otimes \mathbf{A} = \mathbf{A} \forall \mathbf{A} \in Aut(G)$. $\mathbf{E}$ is the **identity** permutation.

An orbit is the set of all atoms that are transformed from one into another by the action of all automorphisms from $Aut(G)$.

The automorphism group of Isopropylcycloporan



$$\mathbf{E} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 4 & 5 & 6 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 6 & 5 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 4 & 6 & 5 \end{pmatrix}$$

P	**E**	**A**	**B**	**C**
$\mathbf{P}^{-1}$	**E**	**A**	**B**	**C**

	E	**A**	**B**	**C**
E	**E**	**A**	**B**	**C**
A	**A**	**E**	**C**	**B**
B	**B**	**C**	**E**	**A**
C	**C**	**B**	**A**	**E**

Orbits: $X_1 = \{1, 2\}, X_2 = \{3\}, X_3 = \{4\}, X_4 = \{5, 6\}$.

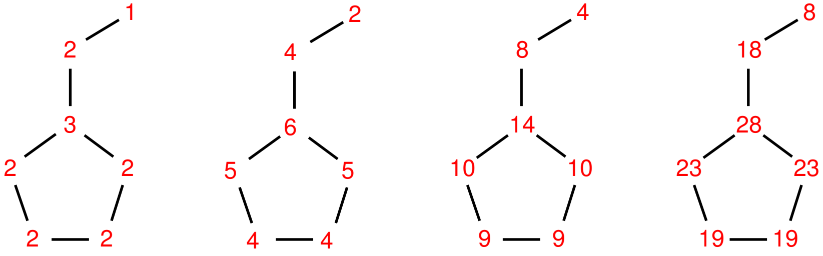
The Morgan Algorithm

The extended connectivity (EC) algorithm efficiently partitions atoms into equivalence classes.

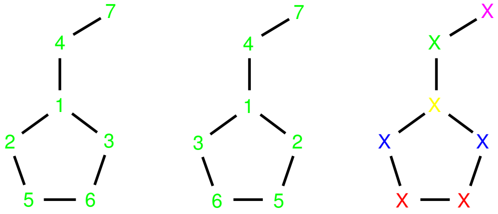
1. Set the EC^1 value of each atom to the value of its degree.
2. Determine the number of different EC^1 v njalues, NECV^1.
3. Set the EC^{n+1} value of each atom to the sum of the EC^n values of the adjacent atoms.
4. Determine NVEC^{n+1}.
5. If $\mathrm{NVEC}^{n+1} > \mathrm{NVEC}^n$ goto step (3).
6. The EC^n values are the final ones.

Morgan HL, J Chem Doc **5**:107-113, (1965)

Canonical labeling of Ethylcyclopentane



The Morgan algorithm reduces the search for a canonical labeling of ethylcyclopentane from $7! = 5040$ to only two labelings.



Canonical Coding of Graphs

A code $Cd(G, Lb)$ of a labeled graph $G(Lb(v))$ is a string obtained from G by a set of rules.

A code is a complete representation of $G(Lb(v))$ because the labeled graph can be reconstructed from the code $Cd(G, Lb)$.

The code is not a structural invariant, because different labelings of G usually give different codes.

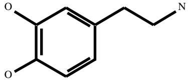
The lexicographical/numerical relations between two strings induce an ordering of the codes. (A minimal/maximal code exists).

For a given molecular graph the canonical code is unique. This property of codes is used in graph isomorphism testing.

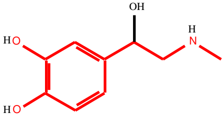
Heuristic approaches are used to reduce the number of permutation labelings, that need to be searched to detect the canonical one.

Substructure Searching

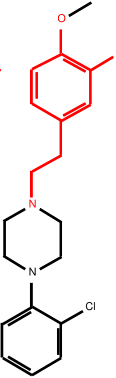
Problem: Identify all molecules containing a specific substructure.



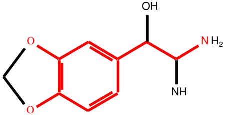
query



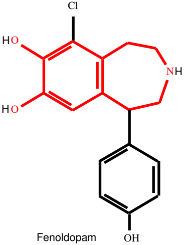
Adrenaline



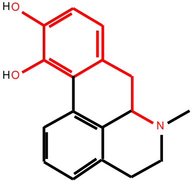
Mefeclorazine



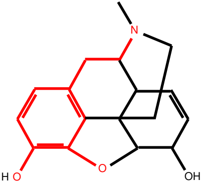
Olmidine



Fenoldopam



Apomorphine



Morphine

Graph theoretical methods for *subgraph isomorphism* can be used.

- These methods are usually too slow for large databases.
- Hence pre-screen database for possibly matching candidates.
- Ideally discard more than 99% of the database.

(Sub)structure Search

Search strategy:

1. Eliminate molecules using the bitstring search.
2. Perform subgraph isomorphism search on remaining molecules.

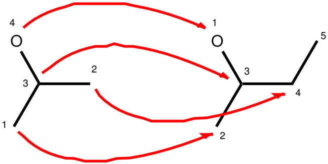
The subgraph isomorphism search belongs to the class of *NP-complete* problems i.e. the runtime of the algorithms scales in the worst case exponentially with the number of nodes in the graphs.

"Brute-force" approach:

1. Generate all possible ways to map the atoms from the query molecule Q onto the host molecules H (from the database).
2. Foreach mapping check if all atom and bond types match.

$$\text{Number of Mappings} = \frac{N_H!}{(N_H - N_Q)!}$$

Possibilities for Performance Improvements

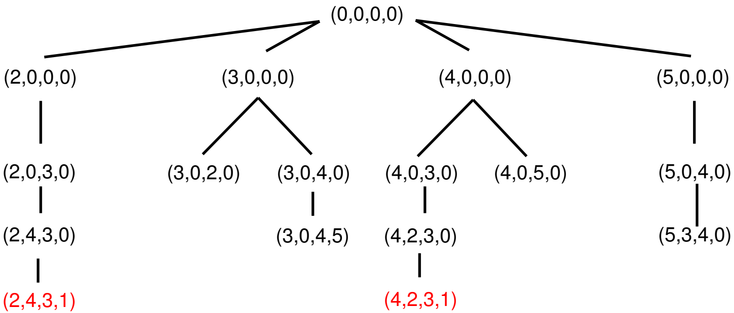


$$N_{\text{maps}} = \frac{5!}{(5-4)!} = 120$$

only two mappings are isomorphisms $(2, 4, 3, 1)$ and $(4, 2, 3, 1)$.

1. Optimize hard and software technologies (e.g. faster computers or parallel architecture).
2. Change heuristic such that partial matchings can be recognized/rejected early during search.
3. Preprocess time consuming operations, which are independent of query structure and store them with the target structure.

Backtracking Algorithms



1. All mappings can be organized hierarchically.
2. Use neighbors of already mapped nodes to extend the partial mappings.

Note: Unfruitful partial mappings are recognized relatively early.

Ray LC, Kirsch RA (1957), Finding Chemical Records by Digital Computers. Science 126:521-533.
Xu J (1996), GMA: A Generic Match Algorithm. J Chem Inf Comput Sci 36:25-34.

Optimization of Backtracking Algorithm

Runtime complexity of the backtracking algorithm is

$$\mathcal{O}(m_H \cdot b^{n_Q})$$

1. Reduce mean value of branching factor
   - Put more information into the node labels (e.g. neighborhood information).
   - Order in which alternatives are examined (e.g. pick unusual heteroatoms with high degree first).
2. Node partitioning (similar to topological symmetry perception)

Class description	Atoms from G_Q	Atoms from G_H
C with 1 single bond	1,2	2, 5
C with 2 single bonds	–	4
C with 3 single bonds	3	3
O with 1 single bond	4	1

Failure of isomorphism search is guaranteed:

- An atom from G_Q does not have a candidate in G_H.
- # of atoms in class i from G_Q is larger than # of candidates in the same class from G_H.