DM840 Algorithms in Cheminformatics:

Minimum (weight) Cycle Basis
—
Two Polynomial Time Algorithms

Daniel Merkle

Week 39/40, 2022

Horton's Algorithm

Horton, J. D. A polynomial-time algorithm to find a shortest cycle basis of a graph. *SIAM Journal of Computing* 16 (1987), 359–366.

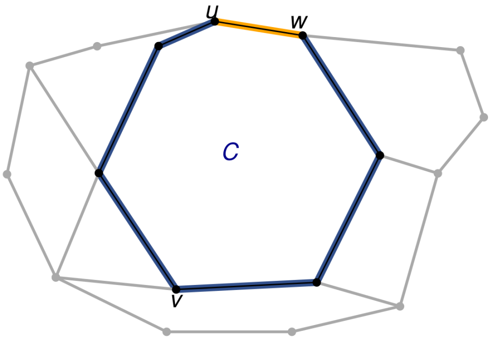
Horton's Algorithm

Theorem

For every cycle in G which is element of an MCB of G, there exists for every node $v \in G$ an edge $\{u, w\} \in G$, such that

$$C = SP(u, v) + SP(w, v) + \{u, w\}$$

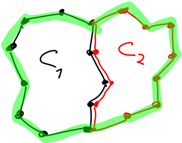
where $SP(x, y)$ denotes the shortest path from node x to node y.



Horton's Algorithm

Lemma 1

Let $\mathcal{B}$ be a cycle basis of G and $C \in \mathcal{B}$ with $C = C_1 \oplus C_2$. Then either $\mathcal{B} \setminus \{C\} \cup \{C_1\}$ or $\mathcal{B} \setminus \{C\} \cup \{C_2\}$ is a cycle basis.

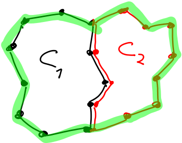


$$C = C_1 \oplus C_2$$

Horton's Algorithm

Lemma 1

Let $\mathcal{B}$ be a cycle basis of G and $C \in \mathcal{B}$ with $C = C_1 \oplus C_2$. Then either $\mathcal{B} \setminus \{C\} \cup \{C_1\}$ or $\mathcal{B} \setminus \{C\} \cup \{C_2\}$ is a cycle basis.



$C = C_1 \oplus C_2$

Proof.

Assume otherwise (none is a cycle basis[a]). Then there is a linear dependency in $\mathcal{B} \setminus \{C\} \cup \{C_1\}$ as well as in $\mathcal{B} \setminus \{C\} \cup \{C_2\}$. Therefore C_1 as well as C_2 can be expressed as a linear combination of $\mathcal{B} \setminus \{C\}$. But $C = C_1 \oplus C_2$, and thus C can also be expressed as a linear combination of $\mathcal{B} \setminus \{C\}$. This is a contradiction to the fact that $\mathcal{B}$ is a basis (details Horton 1987). □

[a] case of "both are a cycle basis" omitted

Horton's Algorithm

Lemma 2

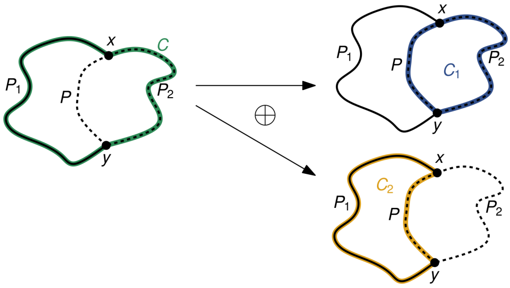
Let $\mathcal{B}$ by a cycle basis of G. For every pair of nodes $x, y \in V$ and a path $P \in G$ from x to y holds: Every cycle $C \in \mathcal{B}$ containing x and y can be replaced by a cycle C, that contains P.

Horton's Algorithm

Lemma 2

Let $\mathcal{B}$ by a cycle basis of G. For every pair of nodes $x, y \in V$ and a path $P \in G$ from x to y holds: Every cycle $C \in \mathcal{B}$ containing x and y can be replaced by a cycle C, that contains P.

Proof:

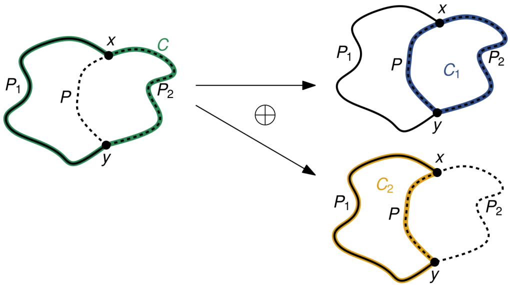


Horton's Algorithm

Lemma 2

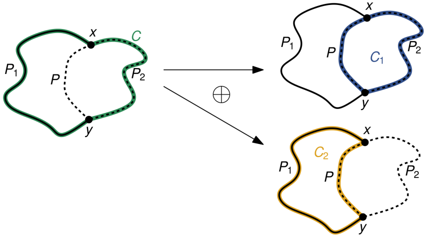
Let $\mathcal{B}$ by a cycle basis of G. For every pair of nodes $x, y \in V$ and a path $P \in G$ from x to y holds: Every cycle $C \in \mathcal{B}$ containing x and y can be replaced by a cycle C, that contains P.

Proof:



It holds $C = C_1 \oplus C_2$ and by Lemma 1 either $\mathcal{B} \setminus \{C\} \cup \{C_1\}$ or $\mathcal{B} \setminus \{C\} \cup \{C_2\}$ is a cycle basis.

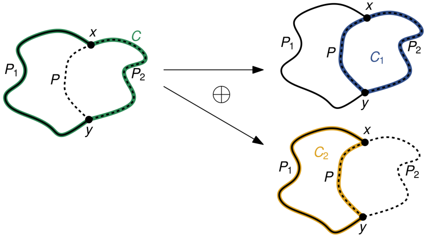
Horton's Algorithm



Implications:

Let neither P_1 nor P_2 be shortest paths between x and y and let P be a shortest path between x and y.

Horton's Algorithm



Implications:

Let neither P_1 nor P_2 be shortest paths between x and y and let P be a shortest path between x and y.

$\Longrightarrow$ $l(C_1) < l(C)$ and $l(C_2) < l(C)$

$\Longrightarrow$ every basis $\mathcal{B}$ containing C can be rewritten into a basis $\mathcal{B}'$, which contains either C_1 or C_2 instead of C

$\Longrightarrow$ $l(\mathcal{B}') < l(\mathcal{B})$

Therefore, if $\mathcal{B}$ is a MCB, then every cycle in $\mathcal{B}$ with nodes x and y also contains a shortest path from x to y.

Horton's Algorithm

Theorem

For every cycle in G which is element of an MCB of G, there exists for every node $v \in G$ an edge $\{u, w\} \in G$, such that

$$C = SP(u, v) + SP(w, v) + \{u, w\}$$

where $SP(x, y)$ denotes the shortest path from node x to node y.

Horton's Algorithm

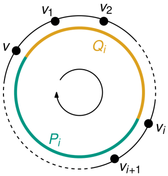
Theorem

For every cycle in G which is element of an MCB of G, there exists for every node $v \in G$ an edge $\{u, w\} \in G$, such that

$$C = SP(u, v) + SP(w, v) + \{u, w\}$$

where $SP(x, y)$ denotes the shortest path from node x to node y.

Proof:



Consider a cycle C and an arbitrary node v in C:

- wlog: Let $v = v_0, v_1, \ldots, v_{l-1}, v_l = v$ be the indices of the nodes.
- Let Q_k be a path from v to v_k in direction of the indexing.
- Let P_k be a path from v_k to v in direction of the indexing.

  $\Longrightarrow$ P_k or Q_k is a shortest path from v to v_k.
- Let i be the largest k such that Q_k is shortest path from v to v_k.

  $\Longrightarrow$ Q_i *and* P_{i+1} are a shortest path from v to v_i.

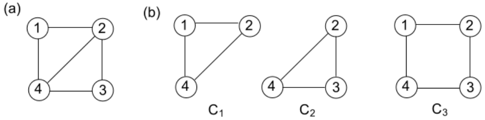
  $\Longrightarrow$ $C = Q_i \oplus \{v_i, v_{i+1}\} \oplus P_{i+1}$

Horton's Algorithm for an MCB

Input: Graph $G = (V, E)$
Output: A Minimum Cycle Basis
$\mathcal{H} \leftarrow \emptyset$
for $v \in V$ *and* $\{u, w\} \in E$ **do**
  $C_v^{uw} := SP(u, v) + \{u, w\} + SP(w, v)$
  if C_v^{uw} *is simple* **then**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{C_v^{uw}\}$
  end
end
Sort the cycles in $\mathcal{H}$ increasingly: $C_1, C_2, \ldots$
$\mathcal{B}^* \leftarrow \emptyset$; i:=1
while $(|\mathcal{B}^*| < |E| - |V| + c(G))$ **do**
  if $\mathcal{B}^* \cup \{C_i\}$ *is linear independent* **then**
    $\mathcal{B}^* \leftarrow \mathcal{B}^* \cup \{C_i\}$
  end
  i++
end

Horton's Algorithm

Runtime of Binary Gaussian elimination: $O(k \times |E|)$



(c)

	[1,2]	[1,4]	[2,3]	[2,4]	[3,4]
C_1	1	1	0	1	0
C_2	0	0	1	1	1
C_3	1	1	1	0	1

$\rightarrow$

	[1,2]	[1,4]	[2,3]	[2,4]	[3,4]
C_1	$\boxed{1}$	1	0	1	0
C_2	0	0	1	1	1
C_3	0	0	1	1	1

(d)

	[1,2]	[1,4]	[2,3]	[2,4]	[3,4]
C_1	1	1	0	1	0
C_2	0	0	1	1	1
C_3	0	0	1	1	1

$\rightarrow$

	[1,2]	[1,4]	[2,3]	[2,4]	[3,4]
C_1	$\boxed{1}$	1	0	1	0
C_2	0	0	$\boxed{1}$	1	1
C_3	0	0	0	0	0

(e)

	[1,2]	[1,4]	[2,3]	[2,4]	[3,4]
C_1	1	1	0	1	0
C_2	0	0	1	1	1
C_3	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~

$\rightarrow$

	[1,2]	[1,4]	[2,3]	[2,4]	[3,4]
C_1	1	1	0	1	0
C_2	0	0	1	1	1

Horton's Algorithm

Runtime:

- Pre-Computation:
  All-Pairs-Shortest Paths (e.g. Floyd-Warshall) $O(|V|^3)$
- Size of Horton set $\mathcal{H} \in O(|V| \times |E|)$
- Sorting the Horton set $O(|\mathcal{H}| \log |\mathcal{H}|)$
- Independence check of *one* cycle:
  One iteration of Gaussian elimination with $|\mathcal{B}^*| = k$ $O(|E| \times k)$
  As $k \leq |E| - |V| + |c(G)|$ $O(|E|^2)$
- Maximal number of iterations of the while-loop: $O(|V| \times |E|)$
- Overall runtime: $O(|V| \times |E|^3)$

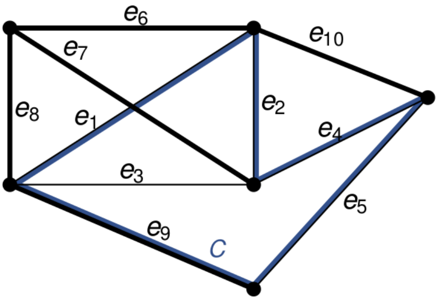
Can be brought down to $O(|V| \times |E|^\omega)$, where $O(n^\omega)$ is the runtime of matrix-matrix multiplication (known: $\omega \leq 2.373$).

de Pina's Algorithm

de Pina, J. *Applications of Shortest Path Methods*. PhD thesis, University of Amsterdam, Netherlands, 1995.

de Pina's Algorithm

Let E be the set of edges not included in an (arbitrarily) chosen spanning tree T.



$$C = \begin{array}{cc} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} & \begin{array}{c} e_1 \\ \\ \vdots \\ \\ e_5 \end{array} \end{array}$$

$$C = C_1 \oplus C_2 \oplus C_4 \oplus C_5$$

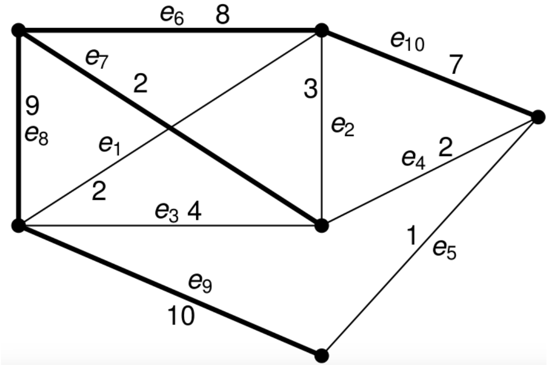
Let C_i be the cycle in T when edge e_i is used (here: $1 \leq i \leq 5$).

It holds (without proof): any cycle C in G can be written as a linear combination of cycles $\{C_1, C_2, \ldots\}$

de Pina's Algorithm

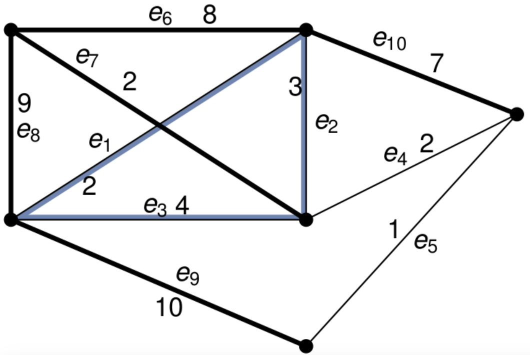
Input: Edges $e_1, \ldots, e_N$ not included in an arbitrary spanning tree of G
Output: A Minimum Cycle Basis of G
for $j = 1, \ldots, N$ **do**
  $S_j := \{e_j\}$
end
for $k = 1, \ldots, N$ **do**
  C_k := shortest cycle in G with $\langle C_k, S_k \rangle = 1$
  for $j = k + 1, \ldots, N$ **do**
    if $\langle C_k, S_j \rangle = 1$ **then**
      $S_j := S_j \oplus S_k$
    end
  end
end
$\{C_1, \ldots, C_N\}$ is a MCB

de Pina's Algorithm



An arbitrarily chosen spanning tree. The edges $\{e_1, \ldots, e_5\}$ are not in the spanning tree.
$S_i := e_i \quad (i = 1, \ldots, 5)$

de Pina's Algorithm ($k = 1$)

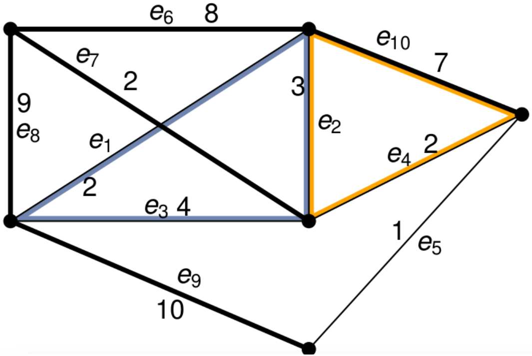


$S_1 = \{e_1\}$

$S_2 := S_2 \oplus S_1 = \{e_1, e_2\}$
$S_3 := S_3 \oplus S_1 = \{e_1, e_3\}$
$S_4 := S_4 = \{e_4\}$
$S_5 := S_5 = \{e_5\}$

The shortest cycle C_1 containing an odd number of edges from S_1 (equivalent: $\langle C_1, S_1 \rangle = 1$)

de Pina's Algorithm ($k = 2$)

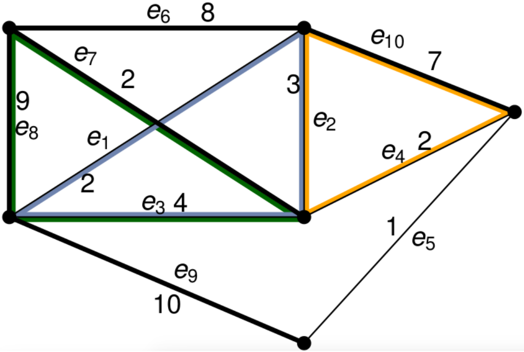


$S_2 = \{e_1, e_2\}$

$S_3 := S_3 = \{e_1, e_3\}$
$S_4 := S_4 \oplus S_2 = \{e_1, e_2, e_4\}$
$S_5 := S_5 = \{e_5\}$

The shortest cycle C_2 containing an odd number of edges from S_2 (equivalent: $\langle C_2, S_2 \rangle = 1$)

de Pina's Algorithm ($k = 3$)

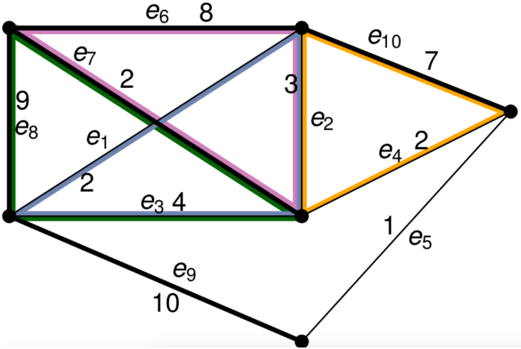


$S_3 = \{e_1, e_3\}$

$S_4 := S_4 = \{e_1, e_2, e_4\}$
$S_5 := S_5 = \{e_5\}$

The shortest cycle C_3 containing an odd number of edges from S_3 (equivalent: $\langle C_3, S_3 \rangle = 1$)

de Pina's Algorithm ($k = 4$)

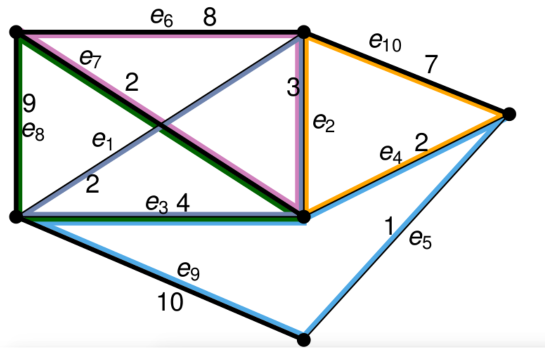


$S_4 = \{e_1, e_2, e_4\}$

$S_5 := S_5 = \{e_5\}$

The shortest cycle C_4 containing an odd number of edges from S_4 (equivalent: $\langle C_4, S_4 \rangle = 1$)

de Pina's Algorithm ($k = 5$)



$S_5 = \{e_5\}$

The shortest cycle C_5 containing an odd number of edges from S_5 (equivalent: $\langle C_5, S_5 \rangle = 1$)

de Pina's Algorithm

- Invariant of the second outer loop:

$$\forall 1 \leq i < j \leq N : \langle C_i, S_j \rangle = 0$$

- Or: the updating of the sets S_j with $j > i$ is nothing more than maintaining a basis $\{S_{i+1}, \ldots, S_{|N|}\}$ of the subspace orthogonal to $\{C_1, \ldots, C_i\}$.

- Can be used to show correctness.

de Pina's Algorithm

Remember Horton's overall runtime: $O(|V| \times |E|^3)$

Runtime (without details):

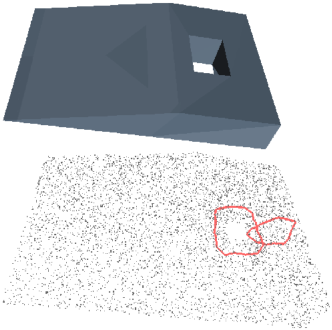
- Finding shortest cycle per phase $O(|V| \times (|E| + |V| log |V|))$
- Update sets per phase $O(|E|^2)$
- Number of phases $O(|E|)$
- Overall runtime $O(|E|^3 + |E|^2 \times |V| + |E| \times |V|^2 \log |V|)$
- Can be improved to $O(|E|^2 \times |V| + |E| \times |V|^2 \log |V|)$

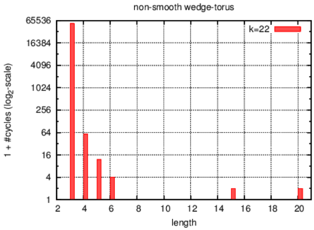
Currently the best known (de Pina/Horton hybrid)[1]: $O\left(\frac{|E|^2 \times |V|}{\log |V|}\right)$

[1] Amaldi, Edoardo; Iuliano, Claudio; Rizzi, Romeo (2010), "Efficient deterministic algorithms for finding a minimum cycle basis in undirected graphs", doi:10.1007/978-3-642-13036-6_30

MCB Applications

Besides characterization of molecules: *Genus Determination*[1]:





Used for surface reconstruction if the sample is "dense enough".

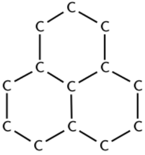
More applications: see e.g.

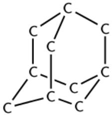
`http://en.wikipedia.org/wiki/Cycle_basis#Applications`

[1] Gotsman, Craig; Kaligosi, Kanela; Mehlhorn, Kurt; Michail, Dimitrios; Pyrga, Evangelia (2007), *Cycle bases of graphs and sampled manifolds*, Computer Aided Geometric Design **24** (8-9): 464480, doi:10.1016/j.cagd.2006.07.001

MCB Applications

Graph invariants in molecular graphs.





- Left: MCB has three hexagons.
  Right: Three of the four visible hexagons make up the forth, they are "interchangeable".
- Define "interchangability"-classes
- Left: has three $\sim_6$ equivalence classes of relative rank 1
  Right: has one $\sim_6$ equivalence classes of relative rank 3
- Use these equivalence classes to characterize the ring system of a molecule by a unique molecular descriptor
- MCB algorithms allow to do this in polynomial time

MCB Applications - Molecular Graph Invariant

Interchangeability classes $\sim_\kappa$:

Lemma

Let C, C' be two relevant cycles of weight κ. Then $C \sim_\kappa C'$ if and only if there is a representation $C = C' \oplus \bigoplus_{D \in \mathcal{I}} D$, where $\{C'\} \cup \mathcal{I}$ is a (linearly) independent subset of the relevant cycles of weight smaller than or equal to κ.

Berger, Franziska; Gritzmann, Peter; de Vries, Sven (2009), *Minimum cycle bases and their applications*, Algorithmics of Large and Complex Networks, Lecture Notes in Computer Science 5515, pp. 3449

MCB Applications - Molecular Graph Invariant

The *relative rank* $|\mathcal{B} \cap W^\kappa|$ of an equivalence class for $\sim_\kappa$:

Lemma

Let $\kappa > 0$ be the weight of some relevant cycle, let $\mathcal{B}, \mathcal{B}'$ be two different minimum cycle bases and let W^κ be an equivalence class for $\sim_\kappa$. Then $|\mathcal{B} \cap W^\kappa| = |\mathcal{B}' \cap W^\kappa|$.

This can be computed in polynomial time, namely $O(|E|^4 \times |V|)$.

MCB Applications - Molecular Graph Invariant

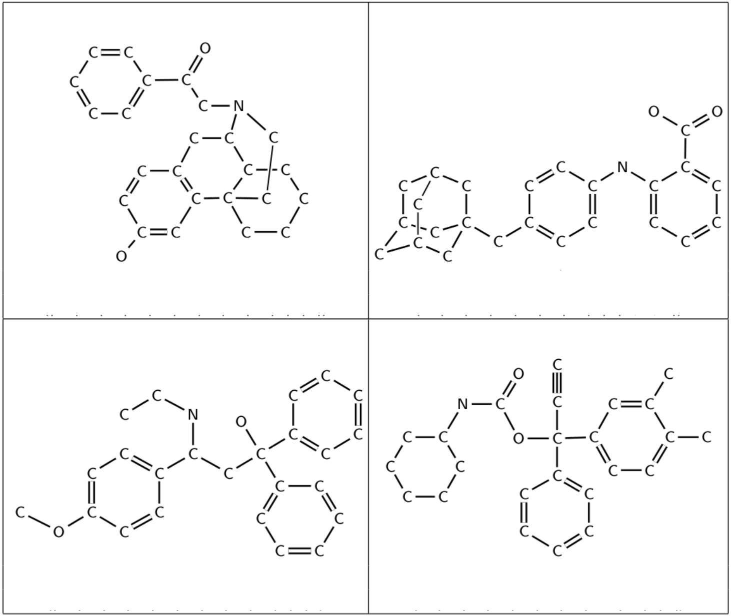
The ordered vector $\beta(G)$ containing the relative ranks of $\sim_\kappa$ equivalence classes is a graph invariant.

MCB Applications - Molecular Graph Invariant

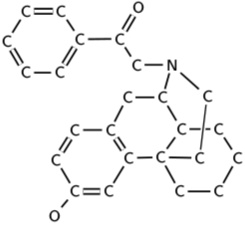
More precise graph invariant: Encode the information gained from a minimum cycle basis and the relative ranks of the interchangeability classes within one vector $w(G)$:

Vertical lines separate the entries in $w(G)$ according to the $\sim_\kappa$ equivalence classes. The relative rank of each class corresponds to the number of entries between two vertical lines, sorted by increasing rank. A subscript e means that the corresponding equivalence class has cardinality 1.

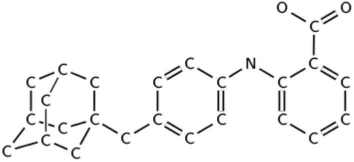
MCB Applications - Molecular Graph Invariant



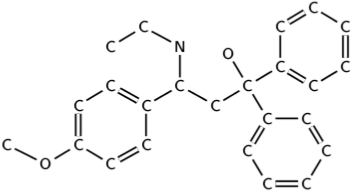
MCB Applications - Molecular Graph Invariant



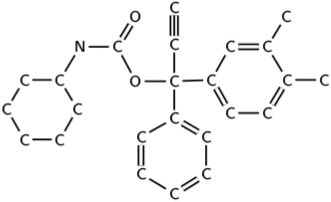
1. Levophenacylmorphan.
$(|2_e|2_e|2_e|2_e|2_e|2_e|2_e|6_e|6_e|6|6|6|)$



2.[1]
$(2_e|2_e|2_e|2_e|2_e|2_e|2_e|6|6|6,6,6|)$



3.[2]
$(|2_e|2_e|2_e|2_e|2_e|2_e|2_e|2_e|2_e|6|6|6)$



4. [3]
$(2_e|2_e|2_e|2_e|2_e|2_e|2_e|2,2|6_e|6|6|)$