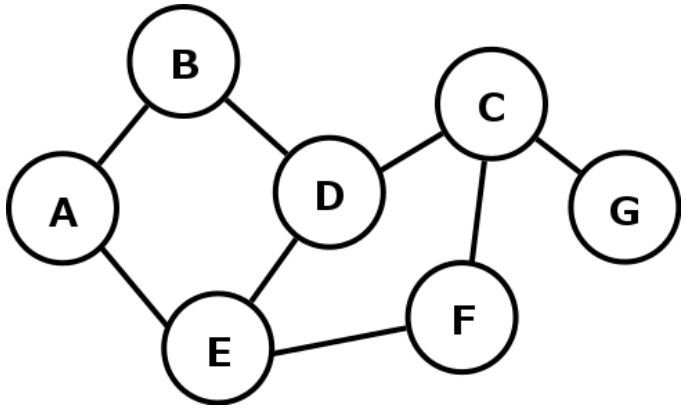
Nauty – McKay 1981

- Assign a **unique (canonical) label** to graphs such that two isomorphic graphs have the same label
- Label = concatenated rows of the adjacency matrix (i.e. integer)

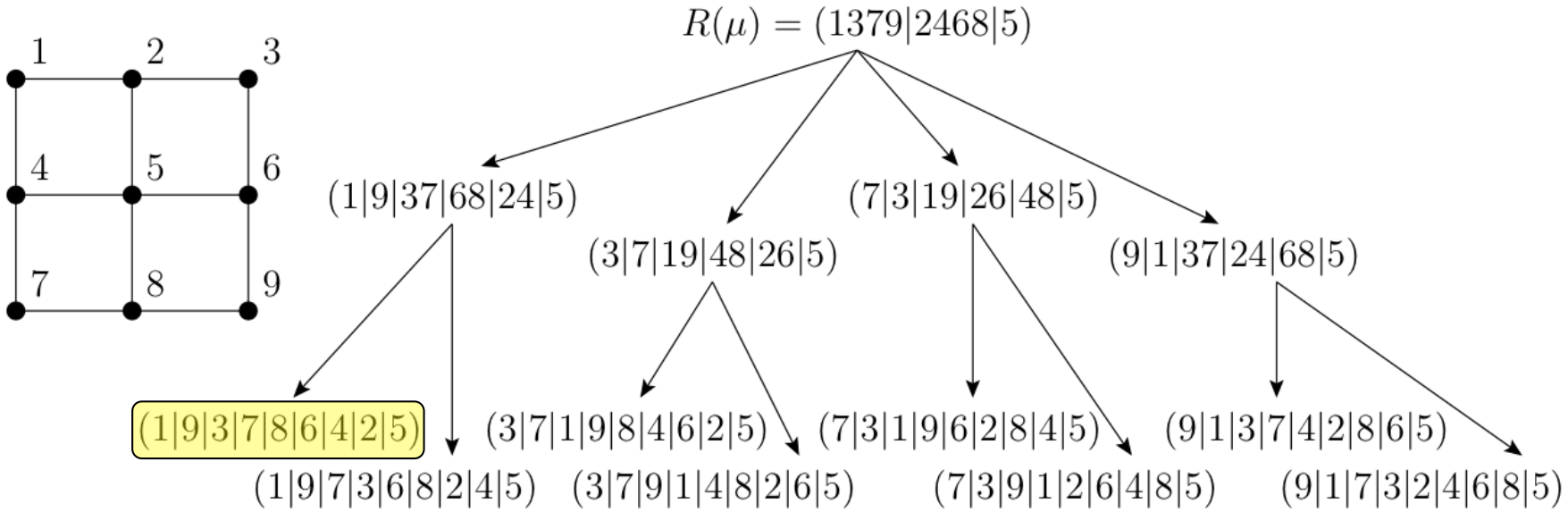



```
A 1 0 0 1 0 0
1 B 0 1 0 0 0
0 0 C 1 0 1 1
0 1 1 D 1 0 0
1 0 0 1 E 1 0
0 0 1 0 1 F 0
0 0 1 0 0 0 G
```

- Canonical label = smallest label of all the automorphisms of G

Nauty – Automorphism enumeration

- Idea: group vertices that are undistinguishable, ie. that have the same degree (and whose neighbors have the same degrees, etc.)
- Recursively split the list of vertices into **equitably ordered partitions** V1, V2, …, Vn where for every i, j, d(v, Vj) = d(w, Vj) for every v, w in Vi

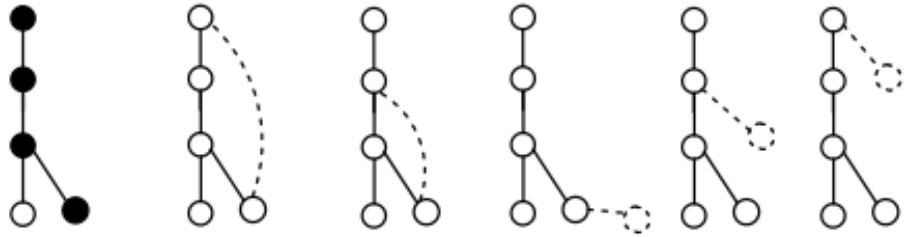


- Smallest automorphism maps 1 to 1, 2 to 9, 3 to 3, …, 9 to 5

Figures from S. G. Hartke and A. J. Radcliffe, *McKay's Canonical Graph Labeling Algorithm*, Contemporary Mathematics 479, 2009

gSpan – Yan & Han, ICMD 2002

- Assign a unique (canonical) label to graphs such that two isomorphic graphs have the same label
- **Minimum DFS encoding**:
  - Create DFS tree
  - Order edges in traversal order DFS encoding
  - Use the one with minimal lexicographic order
- Patterns can only be grown from the right-most path



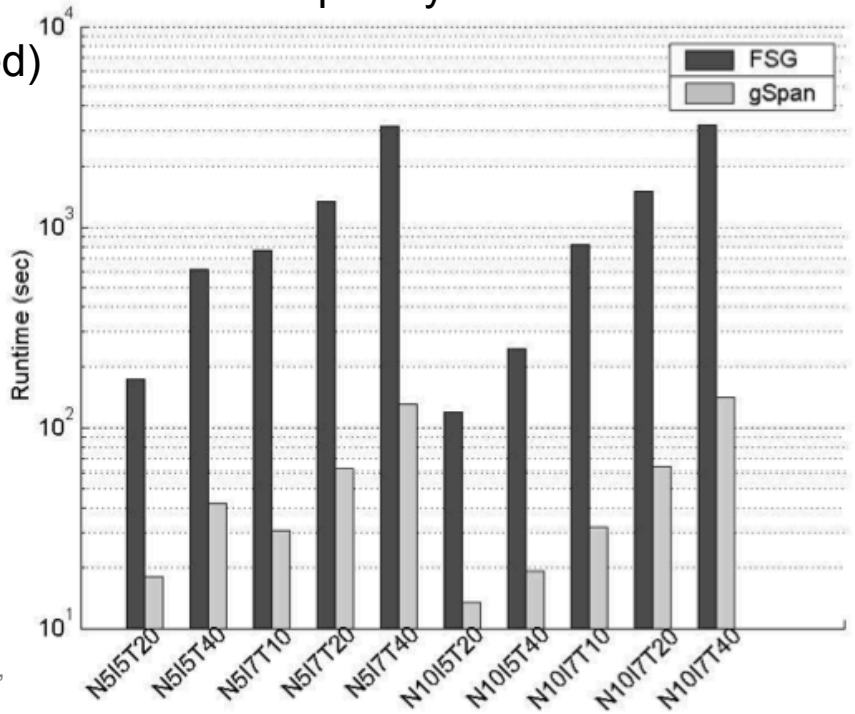
- α = (a1, a2, …, an) can only be extended to β = (a1, a2, …, an, b)
- α is the **parent** of β; β is a **child** of α

Figures from X. Yan and J. Han, *gSpan: graph-based substructure pattern mining*, ICDM 2002

gSpan Algorithm (sketch)

- Sort all vertices and edge labels appearing in the data set according to frequency. Keep only the frequent ones. Re-label them by frequency order (e.g A-a-B is the most frequent edge)
- For all frequent edges **e**:
  - Add **e** and all its frequent children to the set of frequent patterns:
    - Check **e** is not the duplicate of an already explored pattern
    - Recursively grow **e** to compute its children and their frequency

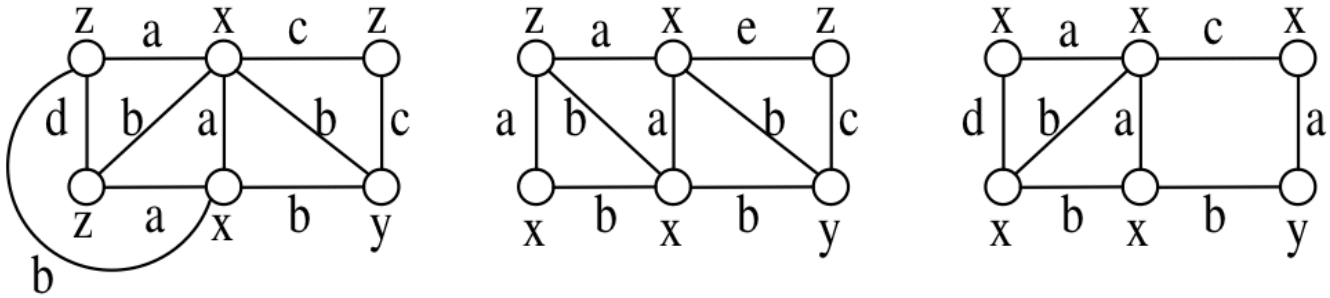
    (non-frequent children will not be extended)
  - Remove e from all graphs in the data set



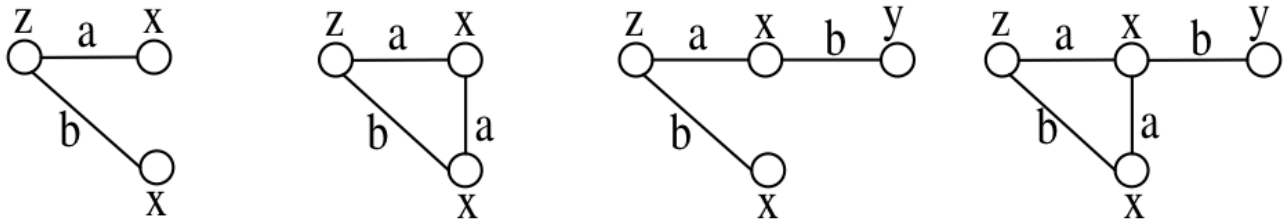
Figures from X. Yan and J. Han, *gSpan: graph-based substructure pattern mining*, ICDM 2002

CloseGraph – Yan & Han, KDD 2003

- Extension of gSpan so as to avoid growing patterns guaranteed to only have non-closed descendants.



- If wherever pattern H1 occurs in the data, pattern H2 = (H1 + e) occurs as well, then for any pattern H*, if H1 is a subgraph of H* and H2 is not, then H* is not closed.

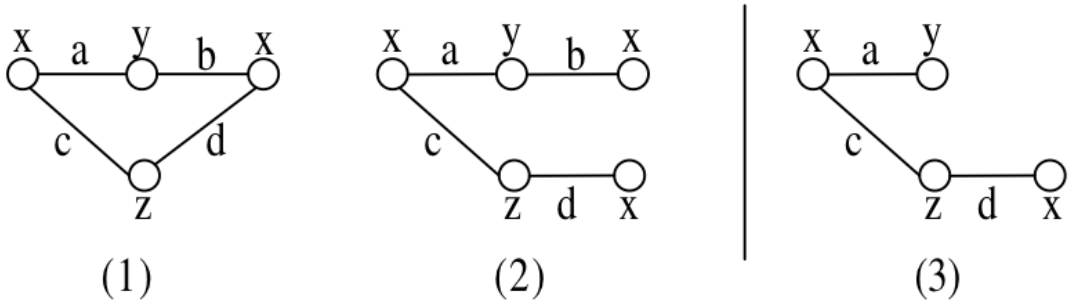


(1) (2) (3) (4)

- (1) and (2) systematically co-occur in D. Therefore (3) cannot be closed – indeed (4) is a supergraph of (3) with identical support. **We need to grow from (2) and not from (1).**

Figures from X. Yan and J. Han, *CloseGraph: mining closed frequent graph patterns*, KDD 2003

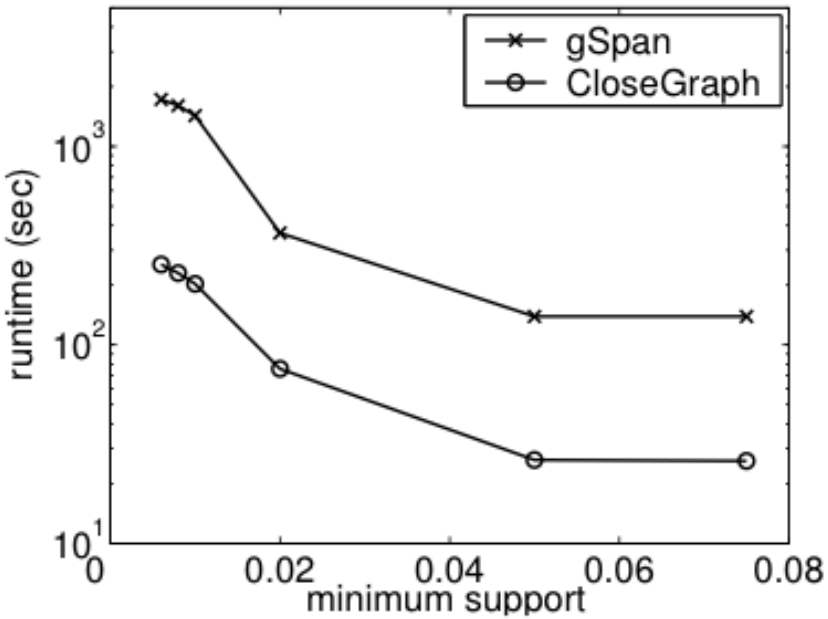
Failure of Early Termination



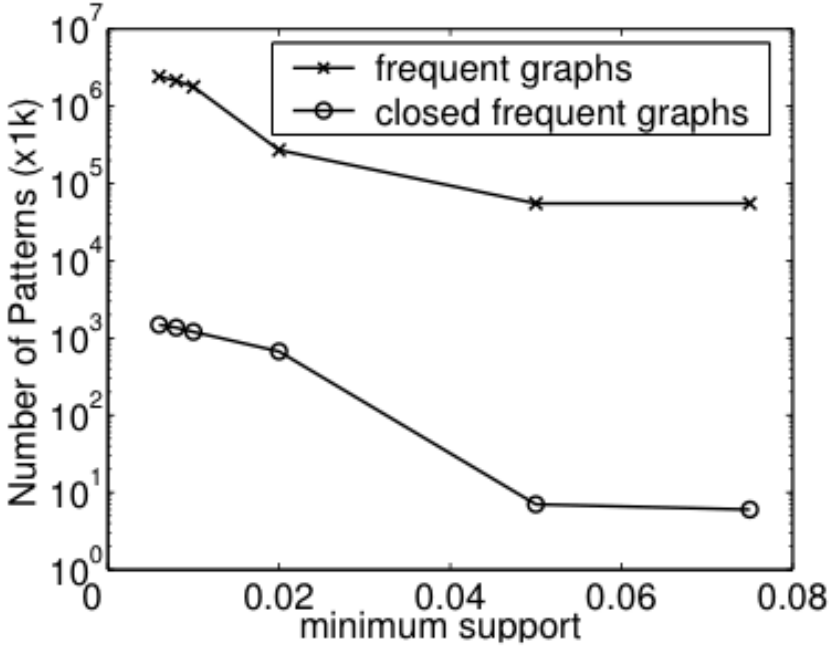
- (x, a, y) and (y, b, x) co-occur in (1) and (2)
- if we only extend from (x, a, y, b, x) then we miss pattern (3), which also co-occurs in (1) and (2)
- Early Termination only works if for every pattern H* such that H1 is a subgraph of H* and (H1 + e) is not, then either H* and (H* + e) where e creates a new vertex or H* and (H* + e) where e does not create a new vertex systematically co-occur in the data

Figures from X. Yan and J. Han, *CloseGraph: mining closed frequent graph patterns*, KDD 2003

gSpan vs. CloseGraph



(a) performance



(b) number of patterns

Figures from X. Yan and J. Han, *CloseGraph: mining closed frequent graph patterns*, KDD 2003