Mining frequent 'easy' graphs in incremental polynomial time

Jan Ramon

K.U.Leuven

DBDBD, October 2006

Outline

Pattern mining

Given:

- A partially ordered language $\mathcal{L}, \preceq$ of patterns.
- An interestingness criterion $\mathit{interesting} : \mathcal{L} \rightarrow \{\mathit{true}, \mathit{false}\}$ (often frequency)

Output:

- List all patterns $p \in \mathcal{L}$ for which $\mathit{interesting}(p)$, preferentially in an order consistent with the order on $\mathcal{L}$.

Examples:

- Itemset mining (limited expressivity but easy)
- Graph mining (rich patterns, often computationally hard)

Complexity of mining

- Even for itemset mining, the output size is exponential, so no miner can be polynomial time in worst case.
- Note that in practice, the space of frequent patterns is sparse (that's why frequent pattern mining is interesting).
- An algorithm A runs in incremental polynomial time iff the time needed to compute the next bit of the output is bounded by a polynomial in the input and the number of bits already outputted.

Expressivity vs. efficiency

- Itemset mining (INC-P)
- Sequence mining (INC-P)
- Tree mining (INC-P)
- ????
- Graph mining (NP-hard)
- FOL mining (NP-hard)

Graph mining is hard

- Database with two examples:
  1. a cycle of length n
  2. an arbitrary graph with n vertices
- Let minimal frequency = 2.
- Possibly frequent patterns = n paths (length $0..n-1$) and 1 cycle.
- Finding hamiltonian cycle is hard.
- Conclusion: not in incremental polynomial time.

Applications

- The class of molecules is a subclass of the class of graphs
  - But usually molecular graphs are not very complex
- Maps are graphs
  - But they are (nearly) planar
- Can we exploit this knowledge?

Efficient graph classes

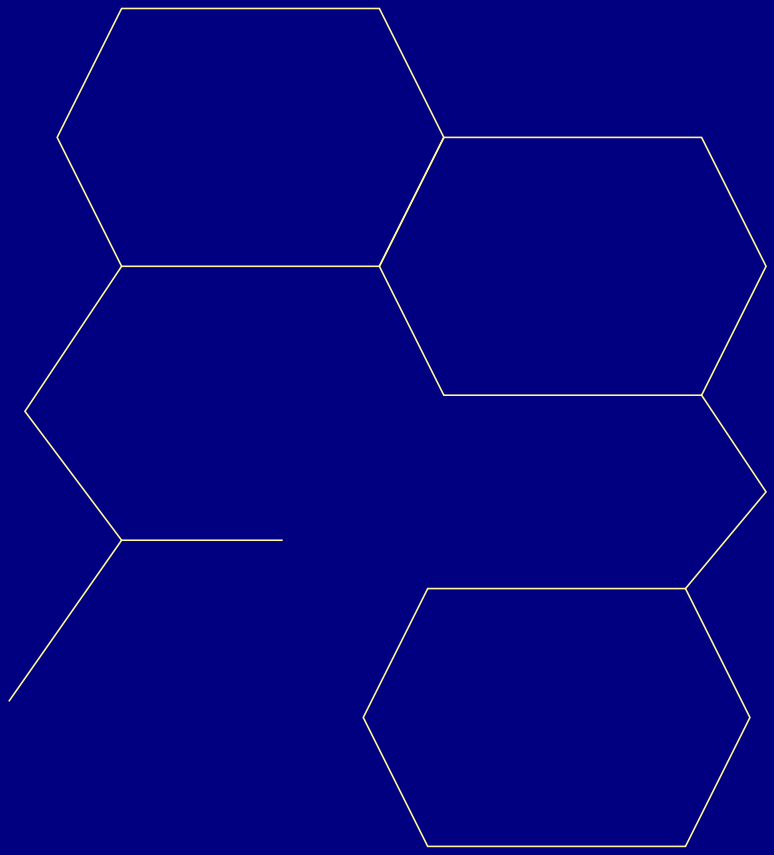
- Graphs become 'easier' if one of the following is small:
  - Pathwidth
  - *Treewidth*
  - Branchwidth
  - *Outerplanarity*
  - Size
  - ...
- Measures are often related, and usually measure how well the graph resembles an 'easy' structure (such as a tree).

Outline

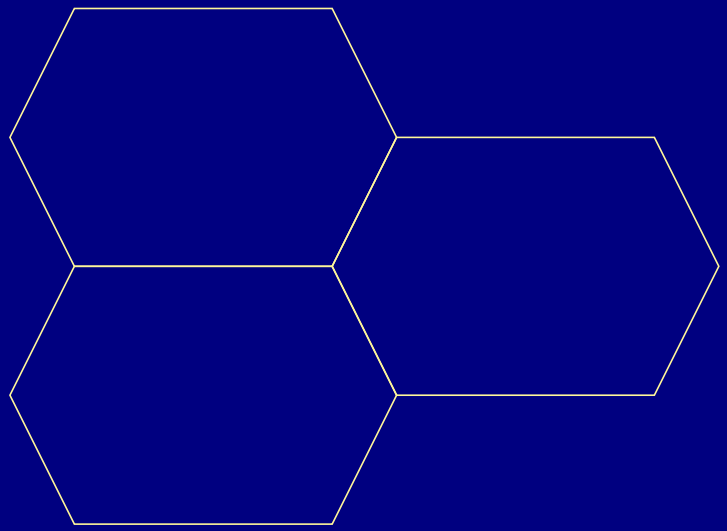
Outerplanar molecules

- Graph is (one-)outerplanar iff it can be embedded in the plane such that all its points are adjacent to the outer face.
- Graph is k outerplanar ($k \geq 2$) iff it can be embedded in the plane such that after removing all its points on the outer face, a $k-1$ graph remains.

Outerplanar



Outerplanar



Not outerplanar

Outerplanar molecules

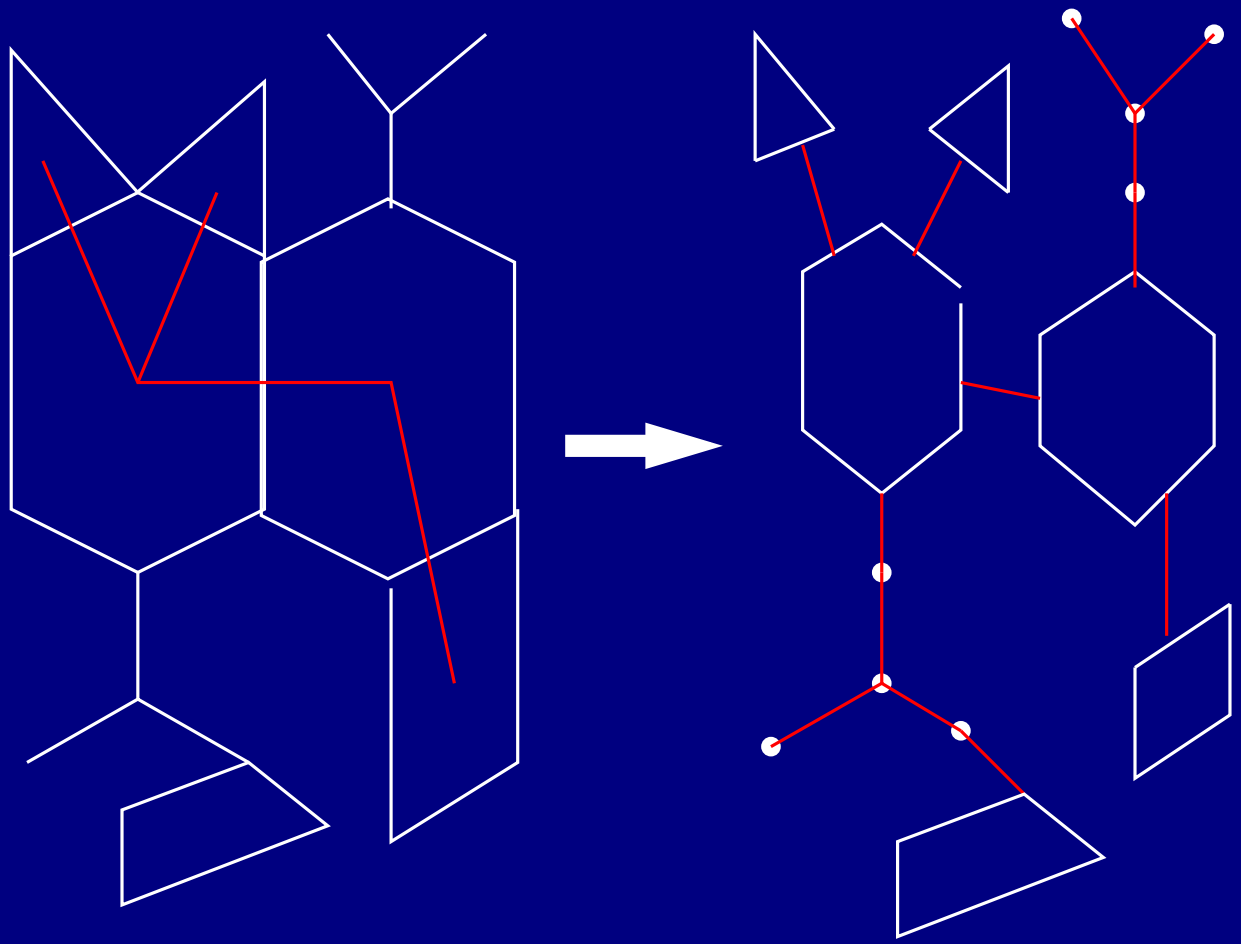
- NCI dataset : large database of chemical compounds, from several domains, e.g. includes a subdatabase with $\pm$40000 HIV-(non)active compounds.

NCI database	250251	100%
Outerplanar	236180	94.3%
Non-outerplanar	14071	5.7%

Mining of outerplanar graphs

- T. Horvath, J. Ramon and S.Wrobel, KDD'2006:
- Define special matching operator (BBP-subgraph isomorphism)
- One can enumerate all frequent outerplanar subgraphs in incremental polynomial time.
- This graph class is relevant for chemistry (NCI-dataset)

Representing outerplanar as tree



Experiments

size (k)	#C	#FP	10% TC	TE
1	84	7	0.00	46.55
2	67	14	0.01	227.58
3	96	33	0.03	373.52
4	109	73	0.03	788.92
5	206	125	0.10	1942.12
6	179	138	0.19	1943.38
7	100	70	0.26	988.89
8	55	42	0.11	524.76
9	21	14	0.06	184.6
10	8	5	0.01	108.58
11	0	0	0	0

Experiments

size	2%			1%		
(k)	#C	#FP	TE	#C	#FP	TE
1	98	23	51	124	40	66
2	739	74	718	3526	145	3332
3	1623	143	1237	5061	252	3566
..	...	...	...	...	...	...
13	1253	1250	1471	2935	2908	1687
14	2153	2151	2117	4210	4176	2365
15	3431	3422	2958	6182	6155	3567
16	5105	5098	4043	9692	9679	5459
17	6820	6789	5168	15999	15988	8770
18	8228	8210	6161	27697	27692	14369
..	...	...	...	...	...	...

Current & future work

- Other graph classes that
  - can be efficiently mined?
  - using practical algorithms ?
  - are relevant in practice ?
- E.g.
  - k-outerplanar
  - Limited treewidth
  - ...

Association rules

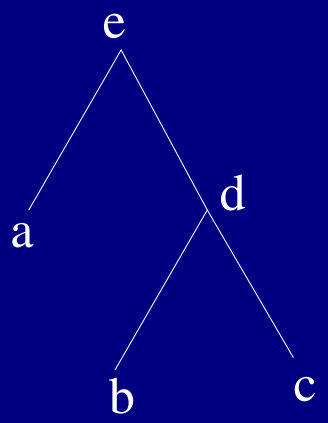
- Different types of association rules
  - $H_1 \preceq G \Rightarrow H_2 \preceq G$ (with $H_1 \prec H_2$) can be obtained easily from the set of frequent patterns (as in the propositional case).
  - $H_1 \preceq_\varphi G \Rightarrow \exists \varphi' \geq \varphi : H_2 \preceq_{\varphi'} G$ (i.e. all embeddings can be extended) is less trivial, but also possible in incremental polynomial time.

Outline

Bottom-up approach

- Most theoretical arguments w.r.t. the complexity of subgraph isomorphism algorithms are based on
  - the representation of the graphs as trees.
  - a bottom-up computation: first compute certain properties (e.g. partial embeddings) for the leaves, then for internal nodes, and so up to the root.

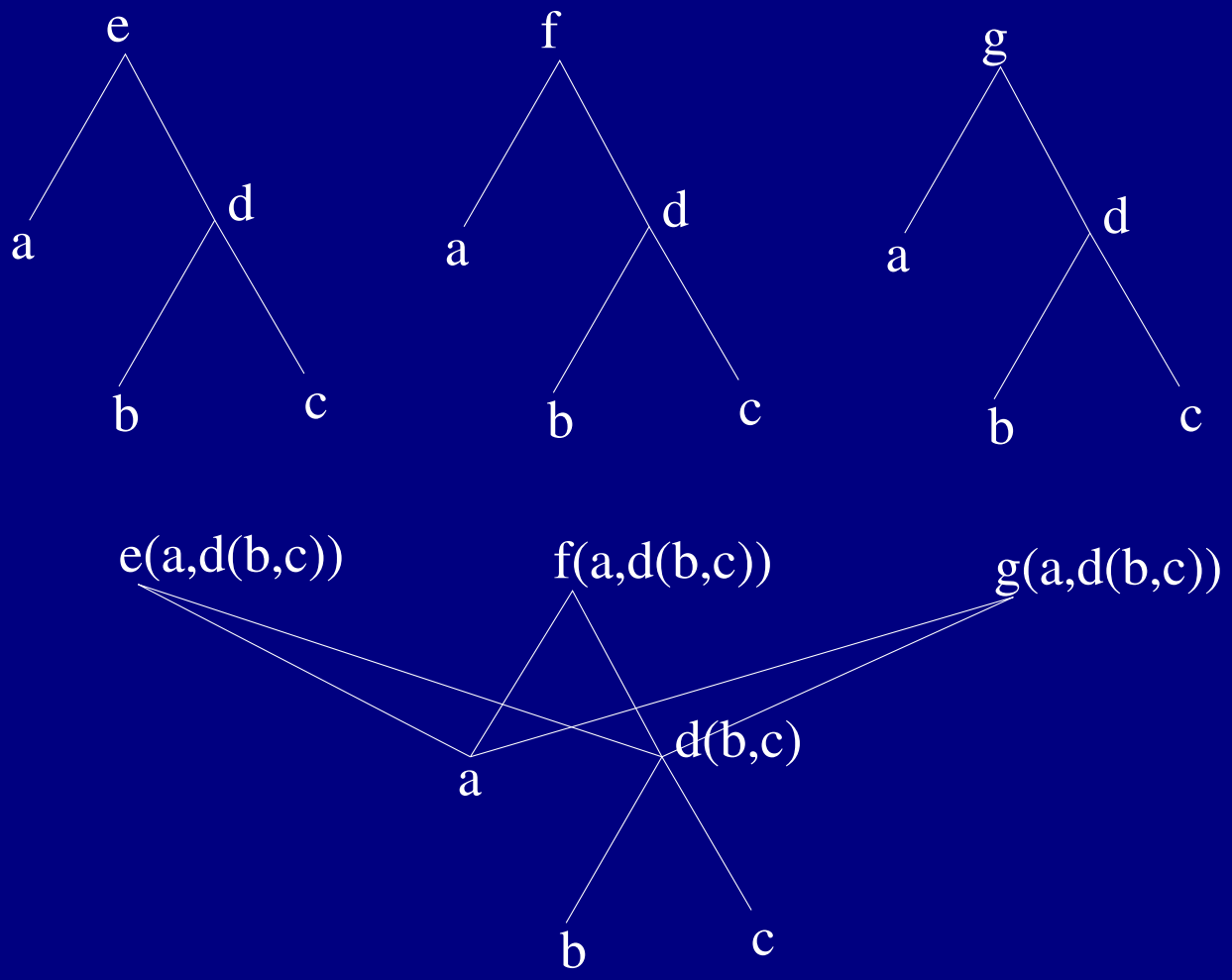
Databases



bottom-up approach as a sequence of joins and projections:

- Let $G(V, E)$ be a tree
- $r_a = \{(v, e) \in V \times E \mid \lambda(v) = a \wedge v \in e\}$
- $r_b = \{(v, e) \in V \times E \mid \lambda(v) = b \wedge v \in e\}$
- $r_c = \{(v, e) \in V \times E \mid \lambda(v) = c \wedge v \in e\}$
- $r_d = \{(v, e) \in V \times E \mid \exists (v_b, e_b) \in r_b, \exists (v_c, e_c) \in r_c : \lambda(v) = c \wedge v \in e \wedge \neq (e, e_b, e_c) \wedge e_b = (v_b, v) \wedge e_c = (v_c, v)\}$
- $r_e = \{v \in V \mid \exists (v_d, e_d) \in r_d, \exists (v_c, e_c) \in r_c : \lambda(v) = c \wedge e_d \neq e_c \wedge e_d = (v_d, v) \wedge e_c = (v_c, v)\}$

Query packs



Experiments

- NCI Database (250251) - Sharing common parts of queries: factor 4
- NCI Database (10000) - Remembering embeddings: factor 1.8

Outline

Conclusions

- We studied outerplanar graphs: they can be mined efficiently.
- Association rules, both existentially and universally quantified can be mined efficiently.
- We studied further optimisations.

Questions

Questions or comments?