Mining frequent 'easy' graphs in incremental polynomial time

Jan Ramon

K.U.Leuven

DBDBD, October 2006
Introduction

Easy graph classes

Related work and optimisations

Conclusions
Pattern mining

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

Output:
- List all patterns $p \in \mathcal{L}$ for which $\text{interesting}(p)$, 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)
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

Introduction

Easy graph classes

Related work and optimisations

Conclusions
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.
Introduction

Easy graph classes

Related work and optimisations

Conclusions

Outerplanar

Outerplanar

Not outerplanar

Not outerplanar

Outerplanar

J. Ramon (K.U.Leuven)
Outerplanar molecules

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

<table>
<thead>
<tr>
<th></th>
<th>NCI database</th>
<th>Outerplanar</th>
<th>Non-outerplanar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250251</td>
<td>236180</td>
<td>14071</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>94.3%</td>
<td>5.7%</td>
</tr>
</tbody>
</table>
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

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

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

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

Introduction

Easy graph classes

Related work and optimisations

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

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

e(a,d(b,c))
f(a,d(b,c))
g(a,d(b,c))
Experiments

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