Induction of Node Label Controlled Graph Grammar Rules

Hendrik Blockeel \(^1,2\)
Siegfried Nijssen \(^1\)

\(^1\) Katholieke Universiteit Leuven
\(^2\) Leiden Institute of Advanced Computer Science
Overview

- Motivation
  - Limitations of Subdue-like grammar induction

- Introduction to node label controlled graph grammars (NLC-GGs)

- An algorithm for learning NLC-GG rewrite rules from graphs

- Conclusions & future work
Motivation

- Grammar induction is a popular approach to learning from strings, and a well-studied problem.
- Induction of graph grammars might be an interesting approach to learning from graphs.
- While graph grammars are well studied (a lot of literature exists on them), there seems to be very little work on learning such grammars.
- Yet, learning such grammars might be useful:
  - Understanding common structure of graphs.
  - Active learning: generate new graphs.
  - Studying dynamic behavior of networks.
  - ...
Existing work on learning graph grammars

- Perhaps best known in the learning/mining community: *Subdue* family of algorithms (Holder, Cook, et al., 1994-)
  - Finds frequently occurring subgraph G
  - Compresses graphs by replacing G with a node N and adding *rewrite rule* N -> G
  - Set of rewrite rules can be seen as a graph grammar
  - Heuristic for finding good grammars: maximal compression of graphs
Disadvantage of Subdue

- Disadvantage 1: *compression is lossy*
  - From the point of view of minimal description length (MDL), this is not very nice

- Disadvantage 2: not well in line with existing, well-studied, graph grammars

- Goal of this work is to remove these disadvantages
Theory on graph grammars

- How to define a “graph grammar”?
- Many different methods have been proposed
- Often, on a high level, two kinds of graph grammars are distinguished:
  - **Hyperedge replacement** grammars
    - Rewrite rule replaces (hyper)edge by new graph
  - **Node replacement** grammars
    - Rewrite rule replaces node by new graph
- Here we will consider *node replacement grammars*
NLC graph grammars

- **Node Label Controlled** graph grammars (see, e.g., Engelfriet & Rozenberg, 1991)
- = node replacement grammars with rules of the form:

\[ N \rightarrow G / E \]

Replace any node with label N by G, connecting G to N’s neighborhood according to the embedding rules listed in E. *Embedding rules are based on node labels.*
Example NLC-GG rule

\[ N \rightarrow b_{\{a,b\}, \{b,c\}} \]
Example NLC-GG rule

\[ N \rightarrow b \quad / \quad \{(a,b), (b,c)\} \]
Another example

\[ N \rightarrow \begin{array}{c}
\text{N} \\
\text{N} \\
\end{array} / \{(a,b), (b,a)\} \]

\[ N \rightarrow \varepsilon / \{\} \]

\[ N \rightarrow \begin{array}{c}
a \\
b \\
\end{array} \rightarrow \begin{array}{c}
a \\
b \\
a \\
b \\
\end{array} \rightarrow \ldots \rightarrow \begin{array}{c}
a \\
a \\
b \\
b \\
\end{array} \rightarrow \begin{array}{c}
b \\
a \\
b \\
b \\
b \\
a \\
b \\
\end{array} \]
Research Question

- Question: can we adapt the Subdue operator so that it learns rules of the form $N \rightarrow G / E$ (instead of $N \rightarrow G$) ?
  
  - This would be a first step towards learning “real” graph grammars (i.e., better in line with existing graph grammar theory).
Task: learn rewrite rule

- Subdue learns a rule $N \rightarrow G$ that leads to maximal compression

- Our goal: Learn a rule $N \rightarrow G / E$ that leads to maximal compression
  - Find a large $G$ that occurs frequently in the graph, and a set $E$ that is compatible with how all these occurrences are embedded in the surrounding graph
Substitutability

Observation 1: given a single occurrence of some subgraph $G$, there may not exist a set of embedding rules $E$ such that $G$ could be generated and embedded by a rule $N \rightarrow G / E$

We say that a subgraph $G$ is substitutable if such an $E$ does exist

– In that case, we can substitute some node $N$ for $G$, and add the rule $N \rightarrow G / E$
No ruleset $E$ exists such that the encircled graph could have been generated from a node $N$ through $N \rightarrow G / E$:

1) 3 nodes (a,a,d) must have been in the environment of $N$
2) Since we have an edge (b,a), (b,a) must have been in $E$
3) But then, b should have been connected also to the other a node
Compatibility

- Observation 2: for 2 substitutable occurrences of the same subgraph G, there may or may not exist a single rule $N \rightarrow G / E$ that could have generated both of them.

- We say that the occurrences are compatible if such a rule does exist.
Compatibility: example

\[ E \supseteq \{(a,a), (b,a), (c,d)\} \]

\[ E \supseteq \{(b,b), (c,d)\} \]

\[ E \supseteq \{(a,a), (b,a), (b,b), (c,d)\} \]
Compatibility: example

E \supseteq \{(a,a), (b,a), (c,d)\}
E \not\supset \{(c,a), (a,d), (b,d)\}
E \supseteq \{(a,a), (b,a), (b,b), (c,d)\}; E \not\supset \{(a,b), (a,d), (b,d), (c,a), (c,b)\}

Rule-Inset (must be in E)

Rule-outset (must not be in E)
Determining E

Auxiliary concepts:
- Given $G \subseteq G'$, and assuming $G$ was generated by some rule $N \rightarrow G / E$:
  - The *Node-InSet* of $G$, $NIS(G)$, contains all nodes in $G'$ – $G$ that must have been in the neighborhood of $N$
  - The *Rule-InSet* $RIS(G)$, also denoted $I$, contains all couples $(l_1, l_2)$ that must have been in $E$
  - The *Rule-OutSet* $ROS(G)$, also denoted $O$, contains all couples $(l_1, l_2)$ that cannot have been in $E$
- We have $I \subseteq E \subseteq L^2-O$ (with $L$ set of all labels)
1: Determining NIS

- The NIS of a graph G equals the set of all nodes outside G connected to it
  - Each node connected to G must have been in the environment of N (otherwise G couldn’t have been connected to it)
  - For each node not connected to G, either:
    - 1) We know it was not in N’s environment
    - Or 2) we don’t know whether it was or wasn’t
    - (Proof: if node x is not connected to G, any E that yields this embedding from N connected to x would yield the same embedding from N not connected to x)
2: Determining I

- I is the set of couples \((a,b)\) such that \(E\) must contain \((a,b)\)

- **I contains \((a,b)\) if and only if a node with label \(a\) in \(G\) is connected to a node with label \(b\) outside \(G\)**
  
  - **If:** if edge \((a,b)\) exists, \((a,b)\) must have been in \(E\), otherwise this edge couldn’t have been generated
  
  - **Only if:** if no edge \((a,b)\) exists, then for any \(E\), \(E - \{(a,b)\}\) would have given the same embedding; hence, \((a,b)\) not in \(I\)
3: Determining $O$

- $O$ is the set of couples $(a,b)$ that cannot possibly be in $E$.

- $O$ contains $(a,b)$ if and only if there is an $a$-node in $G$ and a $b$-node in $\text{NIS}(G)$ that are not connected.
  - If: if $(a,b)$ were in $E$, then the $a$-node and the $b$-node would have been connected, since the $b$-node is in $\text{NIS}(G)$. Since they are not connected, $(a,b)$ must not be in $E$.
  - Only if: $O$ contains $(a,b)$ implies that $E$ cannot contain $(a,b)$, i.e., there is a contradiction if $(a,b)$ is in $E$. Such a contradiction only occurs if there is an $a$-node in $G$ and a $b$-node in $\text{NIS}(G)$ such that $a$ and $b$ are not connected.
Thus, given G (subgraph of G’):  
- Can determine NIS(G) (= nbh(G))
- Can determine I (= \{(l(x), l(y)) | x \in G \land y \in nbh(G) \land (x, y) \in G' \})
- From NIS(G), can determine O (= \{(l(x), l(y)) | x \in G \land y \in nbh(G) \land (x, y) \notin G' \})
- E is a possible embedding rule that might have generated this graph from a graph containing N, using the rule N \rightarrow G / E, if and only if I \subseteq E \subseteq L^2-O

If I and O overlap, there are no E’s fulfilling the above condition, hence G is not substitutable
Sets of occurrences

- Take a set of subgraphs $G_i$ (or “occurrences $G_i$ of some subgraph $G$”), with corresponding $I_i$ and $O_i$
- $E$ is a possible embedding for all $G_i$ if and only if
  - for all $i$: $I_i \subseteq E$; in other words, $\bigcup_i I_i \subseteq E$
  - for all $i$: $E \subseteq L^2 - O_i$; that is, $E \subseteq L^2 - \bigcup_i O_i$
- $\Rightarrow$ can define the RIS and ROS of a set of subgraphs (or occurrences of a single subgraph) as follows:
  - $\text{RIS}(S) = \bigcup_{G \in S} \text{RIS}(G)$
  - $\text{ROS}(S) = \bigcup_{G \in S} \text{ROS}(G)$
- If $\text{RIS}(S) \cap \text{ROS}(S) \neq \emptyset$, there are incompatible graphs in $S$
Maximal compatible subset

- Given a set of occurrences $S = \{G_1, \ldots, G_n\}$, find a maximal subset $S'$ such that $S'$ is compatible.

Solution:
- Call two occurrences $G_i$ and $G_j$ substitution-compatible iff they do not overlap nor touch, and are compatible.
- Construct graph with the $G_i$ as nodes and an edge $(G_i, G_j)$ iff $G_i$ and $G_j$ are substitution-compatible.
- Maximal compatible subset = maximal clique in this graph.
  - Indeed, a set of $n$ occurrences is compatible iff all these occurrences are pairwise compatible.
- Can use existing algorithms for maximal clique finding.
Example
Subdue operator successfully upgraded to learning NLC grammar rules

Computations seem feasible in practice

- Computational bottleneck is maximum clique problem, which frequent graph miners already handle with reasonable efficiency
Future work

- Learn recursive rules
  - Currently only non-recursive rules are handled
  - To learn recursive rules, should drop “do not touch” criterion in substitution-compatibility
    - Can it always be dropped safely?
- Extend to ed-NCE grammars
  - Like NLC grammars, but: directed edges, edge labels, E contains (x,a) where x is node in G and a is label in neighborhood
  - Shown to be a very powerful (expressive) class of grammars
- Find interesting applications