Induction of Node Label Controlled Graph Grammar Rules

Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.BE
Department of Computer Science, Katholieke Universiteit Leuven
& Leiden Institute of Advanced Computer Science, Leiden University

Siegfried Nijssen SIEGFRIED.NIJSSEN@CS.KULEUVEN.BE
Department of Computer Science, Katholieke Universiteit Leuven

Abstract

Algorithms for inducing graph grammars from sets of graphs have been proposed before. An important class of such algorithms are those based on the Subdue graph mining system. But the rules learned by Subdue and its derivatives do not fit easily in any of the well-studied graph grammars formalisms. In this paper, we discuss how Subdue-like algorithms could be made to work in the context of NLC grammars, an important class of node replacement graph grammars. More specifically, we show how, given a set of occurrences of a subgraph, an NLC grammar rule can be induced such that the given occurrences could have been generated by it.

1. Introduction

Grammar induction is a well-studied area in machine learning. In most cases, the grammars that are considered define languages over strings, but languages over trees or over graphs have also been considered. In this text, we focus specifically on graph grammars.

There is a large body of work on graph grammars, but much less on learning such grammars. Perhaps the most widely known work in this area is that by Jonyer et al. (2004), who developed an algorithm for graph grammar induction that builds on Cook and Holder's (1994) Subdue approach for learning from graphs. One could say that Jonyer et al.'s approach is a practical one, which has led to interesting results, but where the link with existing theory on graph grammars is not always clear. This paper presents a first step towards a similar algorithm that will induce *node label controlled graph grammars* (NLC grammars for short) (Engelfriet & Rozenberg, 1990), a well-studied subclass of graph grammars. Specifically, we adapt the main operator used by Subdue (replacing a frequently occurring subgraph by a non-terminal node) so that it can be used for learning correct NLC grammars. In this text we consecutively describe NLC grammars, discuss Subdue and its limitations, and present our own results.

Appearing in the 6^{th} *International Workshop on Mining and Learning with Graphs*, Helsinki, Finland, 2008.

2. NLC graph grammars

Graph grammars are often divided in two categories: node replacement grammars, where the grammar consists of rules that define how a single (non-terminal) node can be replaced by a subgraph; and hyperedge replacement grammars, with rules showing how a hyperedge can be replaced by a graph. Node replacement grammars come in a number of variants, among which NLC grammars are the simplest.

In the following we consider undirected, node-labeled, graphs. Abusing notation somewhat, we will write $x \in G$, $\{x, y\} \in G$, $S \subseteq G$ to denote $x \in Nodes(G)$, $\{x, y\} \in Edges(G)$, and $Nodes(S) \subseteq Nodes(G) \wedge Edges(S) \subseteq Edges(G)$. We define the neighborhood of S in G as $Nbh(S, G) = \{y | \{x, y\} \in G \wedge x \in S \wedge y \notin S\}$.

A rewrite rule in a node replacement grammar is of the form $N \rightarrow S/E$, where N is a node label, S is a graph, and E is an *embedding rule*. Applying the rule to a graph G consists of taking a node x with label N, removing all its incident edges, replacing the node with S, and reconnecting S to the nodes originally connected to N according to the rules specified in E. (The embedding rule E does not occur in string grammars; it is needed here because, contrary to strings, when a node is replaced by a graph, the graph could be reconnected to the rest of the graph in many ways.)

In the case of NLC grammars, E is a set of couples (a, b) where a and b are node labels. $(a, b) \in E$ indicates that each node of S with label a will be connected to each node with label b in the neighborhood of x.

3. Subdue

Starting from a graph G, Subdue repeatedly performs the following operation: find a large subgraph S that occurs often in G, and replace each occurrence[1] S_i of S by a single node x_i with a new label N, connecting all edges formerly incident to S_i to x_i, and remembering that any node labeled N actually stands for that subgraph (i.e., a rewrite rule $N \rightarrow S$ is stored). Let us denote the resulting graph with $G_{S/N}$. S is chosen such that it leads to maximal compression, i.e., $G_{S/N}$ is as small as possible. The same operator is next applied to $G_{S/N}$, and this is continued until no further compression is possible.

Subdue does not learn an embedding rule. As a consequence, the graph compression is lossy: after replacing each occurrence of S with a single node labeled N_S, the original graph cannot be recovered by applying the rewrite rule (the information about how S was connected to the rest of the graph is lost). *The question we address here is how to learn rules of the form $N \rightarrow S/E$, with E a valid embedding rule, thus achieving lossless compression.*

4. Learning NLC grammar rules

The compression achieved by the Subdue operator is roughly $|S| \cdot f(S)$ with $|S|$ the size of S and $f(S)$ its frequency (number of occurrences). Here, we want to find a rule of the form $N \rightarrow S/E$ such that $|S| \cdot f'(S)$ is maximal, with $f'(S)$ the number of occurrences where S is embedded in the graph consistently with E. We want to find E such that $f'(S)$ is maximal.

Clearly, $f'(S) \leq f(S)$. $f'(S) < f(S)$ may hold for three reasons. (1) An occurrence of a single subgraph S in G may be such that no E exists such that this occurrence of S in G could have been generated by a rule of the form $N \rightarrow S/E$. (2) If we have two occurrences S_1 and S_2, for each of which embedding rules E_1 and E_2 exist, there may not exist a single E such that both S_1 and S_2 can be embedded in the graph according to the single E. (3) When two occurrences *touch*,[2] replacing one occurrence with N affects the neighborhood of the other occurrence, possibly destroying its embeddability.

[1] Non-overlapping occurrences are assumed here.

[2] We say that two subgraphs touch if they overlap or one graph overlaps with the other's neighborhood.

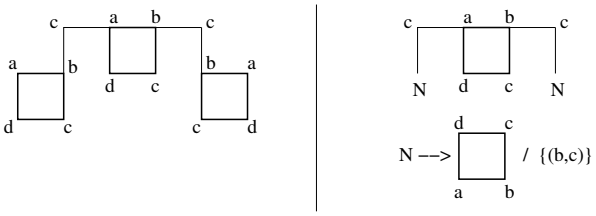

Figure 1. Left: an "a-b-c-d cycle" (indicated in bold) occurs three times in the graph, but only twice in such a way that it could have been generated from a non-terminal node following a particular embedding rule. Right: a compressed version of the graph together with the NLC graph grammar rule that generates the original graph from it.

We call a set of occurrences $\{S_1, \ldots, S_k\}$ in G of a graph S *compatible* if an embedding rule E exists such that repeatedly applying the rule $N \rightarrow S/E$ to $G_{S/N}$ will again yield G. In general, when the set of all occurrences of S in G is not compatible, $f'(S) < f(S)$.

For example: the middle structure in Figure 1 is such that no embedding rule exists such that S could be generated from a non-terminal node (the node with label a should be connected to all nodes with label c in the neighborhood of S). Both other occurrences, though, are compatible with each other.

4.1. Possible embedding rules for one subgraph

Problem: Given a subgraph S of a graph G, what are the possible embedding rules E such that G could have been obtained from some graph F by applying the NLC grammar rule $N \rightarrow S/E$?

Solution: E is a set of couples (a, b) with a and b labels. Recall that when $(a, b) \in E$, each node $x \in S$ with label a will be connected to each node $y \in Nbh(N)$ with label b, when applying $N \rightarrow S/E$. Let L denote the set of all possible node labels.

We will define an *inset* I, which is the set of all (a, b) couples that must be in E, and an *outset* O, which is the set of all (a, b) couples that cannot possibly be in E. I is then a lower approximation of E, and $L^2 - O$ an upper approximation. Any E such that $I \subseteq E \subseteq L^2 - O$ will be a possible embedding rule.

In order to define I and O, we first need a lower approximation of $Nbh(N)$. Define NI, the *node-inset*, as the set of nodes in $G - S$ that *must* have been part of $Nbh(N)$. Clearly, $NI \supseteq Nbh(S)$. In fact, it is possible to show (we omit the proof here) that $NI = Nbh(S)$.

Whenever a node $x \in S$ is connected to a node $y \in G-S$, we know that $(l(x), l(y)) \in E$, with $l(\cdot)$ denoting a node's label. Hence, for each (x, y) such that $x \in S$, $y \in NI$, and $(x, y) \in G$, we have: $(l(x), l(y)) \in I$.

$$I = \{(l(x), l(y)) | x \in S \wedge y \in NI \wedge (x, y) \in G\}.$$

Similarly, whenever we have an $x \in S$ and $y \in NI$ where $(x, y) \notin G$, we know that $(l(x), l(y)) \notin E$. Indeed, if x and y are not connected, although y was in $Nbh(N)$, then $(l(x), l(y))$ must not have been in E.

$$O = \{(l(x), l(y)) | x \in S \wedge y \in NI \wedge (x, y) \notin G\}.$$

If $I \cap O \neq \emptyset$, then no embedding rule E exists such that a rule $N \rightarrow S/E$ could have generated S.

4.2. Possible embedding rules for a set of occurrences of a subgraph

Let S_i, $1 \leq i \leq n$, be a set of isomorphic graphs, all of which are mutually non-touching subgraphs of a graph G. Let I_i and O_i be the inset and outset of S_i.

If there exists an embedding rule E compatible with S_i and with S_j, it must hold that $I_i \subseteq E \subseteq L^2 - O_i$ as well as $I_j \subseteq E \subseteq L^2 - O_j$, and hence

$$I_i \cup I_j \subseteq E \subseteq L^2 - (O_i \cup O_j).$$

Thus, extending the notion of inset and outset to sets of graphs (instead of single graphs), we can say that the inset (outset) of a set of subgraphs is the union of the insets (outsets) of its elements.

A set of non-touching occurrences is compatible if and only if its inset and outset do not overlap.

When two subgraphs touch, replacing one of them with a non-terminal node may destroy the other one or affect its neighborhood. Correctness of our algorithm is only guaranteed for non-touching subgraphs.

4.3. Maximal compatible subset

Problem: Given a set of isomorphic subgraphs, find a maximal compatible subset of non-touching subgraphs in it, i.e., a maximal subset for which a single embedding rule E exists. (Note: $f'(S)$ will be equal to the cardinality of this subset.)

Solution: First note that a set of non-touching subgraphs $\{S_1, \ldots, S_k\}$ is compatible if and only if all its elements are pairwise compatible, i.e., $\forall i, j : \{S_i, S_j\}$ is compatible. Indeed, $\{S_1, \ldots, S_k\}$ is compatible iff $(\bigcup_i I_i) \cap (\bigcup_j O_j) = \emptyset$. Because of distributivity, $(\bigcup_i I_i) \cap (\bigcup_j O_j) = \bigcup_{ij}(I_i \cap O_j)$. This equals $\emptyset$ if and only if $\forall i, j : I_i \cap O_j = \emptyset$.

For any set of subgraphs $\{S_1, \ldots, S_k\}$, we can construct a "compatibility graph" with the S_i as nodes, in which an edge $\{S_i, S_j\}$ indicates that S_i and S_j do not touch and are pairwise compatible. Our problem now reduces to finding a maximal clique in this graph. Kuramochi and Karypis (2004) use a similar approach to compute subgraph supports in a frequent subgraph miner.

While the maximal clique finding problem is NP-complete, Kuramochi and Karypis found this approach to be feasible in practice. Since, for a given graph mining problem, our compatibility graph is always a subgraph of Kuramochi and Karypis's graph (in their graph, two nodes are connected if they do not overlap, which is a strictly weaker condition than ours), we expect our approach to be at least as feasible. This expectation remains to be validated empirically.

5. Conclusions

We have studied the following problem: given a set of occurrences of a subgraph S in a graph G, find a maximal subset of non-touching occurrences that could have been generated by a single NLC grammar rule $N \rightarrow S/E$ (identifying E at the same time). We have proposed an algorithmic solution for this problem. Embedded in a Subdue-like system, this algorithm paves the way towards lossless graph compression using NLC grammar rules, and towards induction of NLC grammars.

Practical consequences of using this algorithm (such as its effect on efficiency) remain to be investigated. The question of how to find compatible touching subgraphs also remains open. Finally, we note that NLC grammars are the simplest of a series of graph grammars, of which the so-called edNCE grammars (which have more powerful embedding rules) are the most powerful. Those edNCE grammars still resemble NLC grammars quite well in structure, so it makes sense to study whether the proposed approach can be generalized towards learning edNCE grammars.

Acknowledgments

H.B. is a post-doctoral fellow of the Research Foundation - Flanders (FWO-Vlaanderen). S.N. was supported by the EU FET IST project "Inductive Querying", contract number FP6-516169. Work also supported by Vidi project "Annotated Graph Mining" of the Dutch NWO and Project G.0306.07 "Graph logic:

representation, inference and learning" funded by the Research Foundation - Flanders.

References

Cook, D. J., & Holder, L. B. (1994). Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, *1*, 231–255.

Engelfriet, J., & Rozenberg, G. (1990). Graph grammars based on node rewriting: An introduction to nlc graph grammars. *Graph-Grammars and Their Application to Computer Science* (pp. 12–23). Springer-Verlag.

Jonyer, I., Holder, L., & Cook, D. (2004). MDL-based context-free graph grammar induction and applications. *International Journal on Artificial Intelligence Tools*, *13*, 65–79.

Kuramochi, M., & Karypis, G. (2004). Finding frequent patterns in a large sparse graph. *Proceedings of the Fourth SIAM International Conference on Data Mining*.