Grammar-based Compression for Directed and Undirected Generalized Series-parallel Graphs using Integer Linear Programming

Morihiro Hayashida[1], Hitoshi Koyano[2] and Tatsuya Akutsu[3]
[1]*National Institute of Technology, Matsue College, 14-4, Nishiikumacho, Matsue, Shimane 690–8518, Japan*
[2]*Quantitative Biology Center, Riken, 2-2-3, Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo 650–0047, Japan*
[3]*Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611–0011, Japan*

Keywords: Generalized Series-parallel Graph, Grammar-based Compression, Integer Linear Programming.

Abstract: We address a problem of finding generation rules from biological data, especially, represented as directed and undirected generalized series-parallel graphs (GSPGs), which include trees, outerplanar graphs, and series-parallel graphs. In the previous study, grammars for edge-labeled rooted ordered and unordered trees, called SEOTG and SEUTG, respectively, were defined, and it was examined to extract generation rules from glycans and RNAs that can be represented by rooted tree structures, where integer linear programming-based methods for finding the minimum SEOTG and SEUTG that produce only given trees were developed. In nature and organisms, however, there are various kinds of structures such as gene regulatory networks, metabolic pathways, and chemical structures that cannot be represented as rooted trees. In this study, we relax the limitation of structures to be compressed, and propose grammars representing edge-labeled directed and undirected GSPGs based on context-free grammars by extending SEOTG and SEUTG. In addition, we propose an integer linear programming-based method for finding the minimum GSPG grammar in order to analyze more complicated biological networks and structures.

1 INTRODUCTION

Data compression for a structure is related with the amount of information that it contains. The amount of information would be large if the size of compressed data is still large. Otherwise, the data include redundant data, and the amount of information is small. Our purpose is to extract useful information and knowledge from data through compression. In particular, we focus on biological structured data constructed in nature. Such structures could be often explained by several simple generation rules.

In previous studies, biological data represented by rooted trees such as glycans and RNAs were compressed and analyzed (Zhao et al., 2010; Zhao et al., 2015). It is known that glycans are composed of multiple monosaccharides bound by glycosidic bonds, take various structures in accordance with biosynthetic reactions, and the function of a glycan depends on its structure. Hence, it is important to analyze the glycan structures, and to extract rules of the biosyntheses. They developed integer linear programming-based methods, called the minimum SEOTG and SEUTG, for finding the minimum grammar that produces only given single ordered and unordered rooted trees, and applied them to biological data such as glycans with up to 36 nodes and 5 distinct labels, where these methods are based on a kind of tree grammar, the simple elementary ordered (unordered) tree grammar (SEO(U)TG) (Akutsu, 2010). Furthermore, they extended the methods to multiple rooted trees because generation rules are commonly utilised among these multiple trees. It, however, is considered that structures generated in nature cannot be always represented by rooted trees. In this paper, we extend their grammar to directed and undirected generalized series-parallel graphs (GSPGs), which include trees and outerplanar graphs. In addition, we propose an integer linear programming-based method for finding the minimum GSPG grammar that produces only a given generalized series-parallel graph.

A series-parallel graph is defined by two procedures, called series and parallel compositions, and two special nodes in the graph are labeled with source and sink as terminal nodes (Eppstein, 1992; Eikel et al., 2015). A generalized series-parallel graph is defined by the addition of another series-type composition, called generalized series composition, where

Hayashida, M., Koyano, H. and Akutsu, T.
Grammar-based Compression for Directed and Undirected Generalized Series-parallel Graphs using Integer Linear Programming.
DOI: 10.5220/0006583001050111
In *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018) - Volume 3: BIOINFORMATICS*, pages 105-111
ISBN: 978-989-758-280-6

the shared node between two composed graphs is labeled with a terminal node (Korneyenko, 1994). Ho et al. proposed a decomposition method for GSPGs using many processors in parallel (Ho et al., 1999). However, it is not guaranteed that their method always finds the minimum decomposition tree. It has been proved that the problem of finding the minimum SEO(U)TG for a given rooted tree is NP-hard (Akutsu, 2010). Hence, the problem of finding the minimum grammar for a given GSPG is also NP-hard, and it means that there does not exist any polynomial time algorithm for finding the minimum GSPG grammar.

Since production rules of a SEO(U)TG can be regarded as two types of series compositions in GSPGs, we define a grammar by adding a production rule corresponding to the parallel composition to their grammar, and develop an integer linear programming-based method for finding the minimum GSPG grammar.

2 METHOD

We briefly review the simple elementary ordered (unordered) tree grammar (SEO(U)TG) and the integer linear programming-based methods for finding the minimum SEOTG and SEUTG, and propose grammars for edge-labeled directed and undirected generalized series-parallel graphs (GSPGs) and an integer linear programming-based method for finding the minimum GSPG grammar.

2.1 SEOTG and SEUTG

SEOTG and SEUTG are context-free grammar (CFG)-like grammars for edge-labeled ordered and unordered rooted trees, respectively. In CFG, a nonterminal symbol is replaced with several (non)terminal symbols (Hopcroft et al., 2001). In SEO(U)TG, an edge having a nonterminal symbol is replaced with one or two edges having (non)terminal symbols. SEOTG and SEUTG are defined as follows.

Definition 1 (Simple Elementary Ordered Tree Grammar (SEOTG))**.** *A SEOTG is defined as 4-tuple* $(\Sigma, \Gamma, S, \Delta)$*, where* Σ *is a set of terminal symbols,* Γ *is a set of nonterminal symbols, S is a start nonterminal symbol in* Γ*, and* Δ *is a set of production rules (R1), (R1t), (R2), (R2t), (R3), (R3r), (R3l) as shown in Fig. 1.*

Definition 2 (Simple Elementary Unordered Tree Grammar (SEUTG))**.** *A SEUTG is defined as 4-tuple* $(\Sigma, \Gamma, S, \Delta)$ *where* Σ *is a set of terminal symbols,* Γ *is a set of nonterminal symbols, S is a start nonterminal symbol in* Γ*, and* Δ *is a set of production rules (R1), (R1t), (R2), (R2t), (R3), (R3r).*

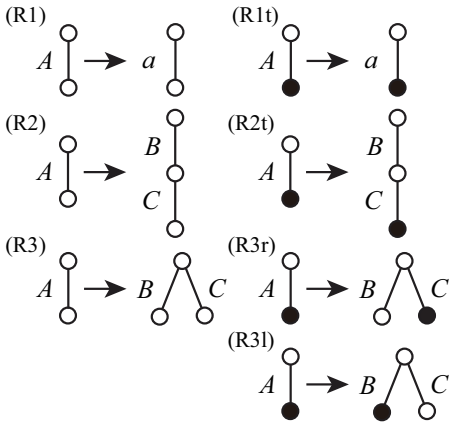

Figure 1: Three main types (R1), (R2), (R3) of production rules of SEOTG for rooted ordered trees. A black circle denotes a tag.

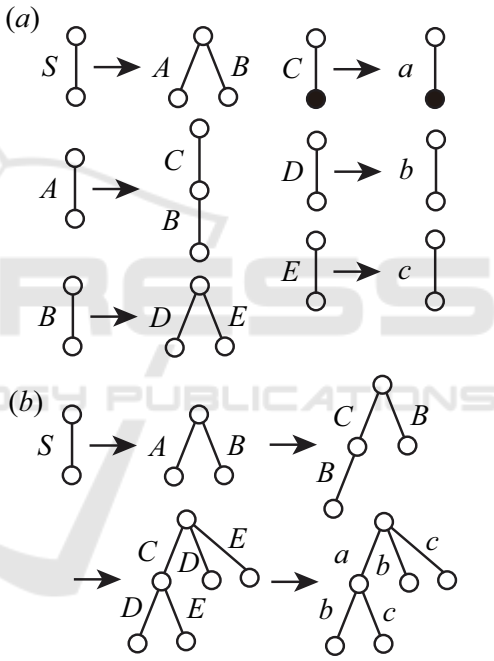

Figure 2: Example of a SEOTG with $(\{a,b,c\},\{S,A,B,C,D,E\},S,\Delta)$ and the tree generated by this grammar. (a) Production rules of Δ. (b) The derivation of the generated tree.

It is noted that (R3r) becomes equivalent to (R3l) because the edge order is ignored.

These production rules do not construct any cycle but trees. A tree generated from a nonterminal symbol by SEOTG and SEUTG has at most two special nodes, its root and a tag node, where a tag means a terminal node to which another tree structure can be attached.

Fig. 2 shows an example of a SEOTG with $(\{a,b,c\},\{S,A,B,C,D,E\},S,\Delta)$, and the tree generated by the grammar, where Δ is shown in Fig. 2(a). The generation starts from S, production rules are applied to edges with nonterminal symbols, and the tree with only terminal symbols is generated (see Fig. 2(b)).

2.2 The Minimum SEOTG and SEUTG

We can obtain a clue of generation mechanisms of biological structures by finding the minimum grammar. For a rooted ordered tree T, the following integer linear programming problem was formulated for finding the minimum SEOTG that produces only the tree T.

$$\text{Minimize} \sum_{u\in U} p_u$$

Subject to

$$x_{i,\varepsilon,j,j} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|ch(j)| = 0),$$
$$x_{i,j,j,j} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|ch(j)| > 0),$$
$$x_{1,\varepsilon,lch(1),rch(1)} = 1,$$
$$x_{i,\varepsilon,h,k} \le \sum_{l=h}^{k-1} y_{i,\varepsilon,h,l,k} + \sum_{t\in I(T_{i,\varepsilon,h,k})} z_{i,\varepsilon,h,k,t} \quad \text{for all } i,\ h \le k \in ch(i),$$
$$y_{i,\varepsilon,h,l,k} \le \frac{1}{2}(x_{i,\varepsilon,h,l} + x_{i,\varepsilon,l+1,k}) \quad \text{for all } i,\ h \le l < k \in ch(i),$$
$$z_{i,\varepsilon,h,k,t} \le \frac{1}{2}(x_{i,t,h,k} + x_{t,\varepsilon,lch(t),rch(t)}) \quad \text{for all } i,\ h \le k \in ch(i), t \in I(T_{i,\varepsilon,h,k}),$$
$$x_{i,j,h,k} \le \sum_{l=h}^{k-1} y_{i,j,h,l,k} + \sum_{t\in anc(j)} z_{i,j,h,k,t} \quad \text{for all } i,\ h \le k \in ch(i),\ j \in I(T_{i,\varepsilon,h,k}),$$
$$y_{i,j,h,l,k} \le \frac{1}{2}(x_{i,\varepsilon,h,l} + x_{i,j,l+1,k}) \quad \text{for all } i,\ h \le l < k \in ch(i),\ j \in I(T_{i,\varepsilon,l+1,k}),$$
$$y_{i,j,h,l,k} \le \frac{1}{2}(x_{i,j,h,l} + x_{i,\varepsilon,l+1,k}) \quad \text{for all } i,\ h \le l < k \in ch(i),\ j \in I(T_{i,\varepsilon,h,l}),$$
$$z_{i,j,h,k,t} \le \frac{1}{2}(x_{i,t,h,k} + x_{t,j,lch(t),rch(t)}) \quad \text{for all } i,\ h \le k \in ch(i),\ j \in I(T_{i,\varepsilon,h,k}),\ t \in anc(j),$$
$$p_u \ge \frac{1}{|S(u)|} \sum_{T_{i,j,h,k}\in S(u)} x_{i,j,h,k} \quad \text{for all } u \in U,$$
$$p_u < 1 + \frac{1}{|S(u)|} \sum_{T_{i,j,h,k}\in S(u)} x_{i,j,h,k} \quad \text{for all } u \in U,$$
$$x_{i,j,h,k}, y_{i,j,h,l,k}, z_{i,j,h,k,t}, p_u \in \{0,1\},$$

where $lch(i)$, $rch(i)$, and $ch(i)$ denote the leftmost child of the node v_i in T, the rightmost child of v_i, and the set of child nodes of v_i, respectively. $T_{i,t,h,k}$ denotes the subtree rooted at v_i, with the child nodes v_j $(h \le j \le k)$ and v_t labeled with a tag in T, which does not have a tag when $t = \varepsilon$. $I(T)$ denotes the set of internal nodes, except for the root and leaves of tree T. $anc(j)$ denotes the set of ancestor nodes of v_j, where $j \notin anc(j)$ and $anc(\varepsilon) = \emptyset$.

Each variable of $x_{i,j,h,k}, y_{i,j,h,l,k}, z_{i,j,h,k,t}$ takes either 0 or 1. $x_{i,j,h,k} = 1$ if $T_{i,j,h,k}$ is generated by the grammar, $x_{i,j,h,k} = 0$ otherwise. $y_{i,j,h,l,k} = 1$ if both of $T_{i,j,h,l}$ and $T_{i,j,l+1,k}$ are generated, $y_{i,j,h,l,k} = 0$ otherwise. $z_{i,j,h,k,t} = 1$ if both of $T_{i,t,h,k}$ and $T_{i,j,lch(t),rch(t)}$ are generated, $z_{i,j,h,k,t} = 0$ otherwise.

In this formulation, the Euler string $es(T)$ is used to determine if two edge-labeled rooted trees T_1 and T_2 are isomorphic to each other, where $es(T)$ for a tree T is defined by the sequence of edge labels l and its opposite $\bar{l}$, along the depth-first search traversal of T (Akutsu, 2010). It is noted that for two edge-labeled rooted trees T_1 and T_2, T_1 is isomorphic to T_2 if (and only if) $es(T_1) = es(T_2)$. U denotes the set of all Euler strings for all connected subtrees of T. $S(u)$ denotes the set of all subtrees $T_{i,j,h,k}$ of T such that $es(T_{i,j,h,k})$ is equivalent to u. Then, $p_u = 1$ means that the minimum grammar generates the subtree corresponding to u, and $\sum_{u\in U} p_u$ represents the number of nonterminal symbols.

Similarly to the minimum SEOTG, the minimum SEUTG was formulated.

2.3 Directed and Undirected Generalized Series-parallel Graph Grammars (GSPGGs)

Let $G(V,E)$ be an undirected GSPG with a set V of nodes and a set E of edges labeled with $l(e)$ for $e \in E$. For example, Fig. 4(a) shows the benzene ring, which is regarded as an undirected GSPG with six nodes and six edges, and is constructed by several series compositions after one parallel composition.

We define an undirected generalized series-parallel graph grammar (GSPGG) as follows.

Definition 3 (Undirected generalized series-parallel graph grammar)**.** *An undirected GSPGG is defined as 4-tuple* $(\Sigma, \Gamma, S, \Delta)$*, where* Σ *and* Γ *are sets of nonterminal and terminal symbols, every terminal symbol is an undirected labeled edge,* S *is a start nonterminal symbol, and* Δ *is a set of production rules as shown in Fig. 3.*

In Fig. 3, a head and a tail of each arrow denote two terminal nodes of its edge. If the graph with only terminal symbols generated from a nonterminal symbol is symmetric, then the source and sink nodes can be changed to each other. White and black squares mean that in a production rule, the node with a white (black) square in the left-hand side corresponds to the node with a white (black) square in the right-hand

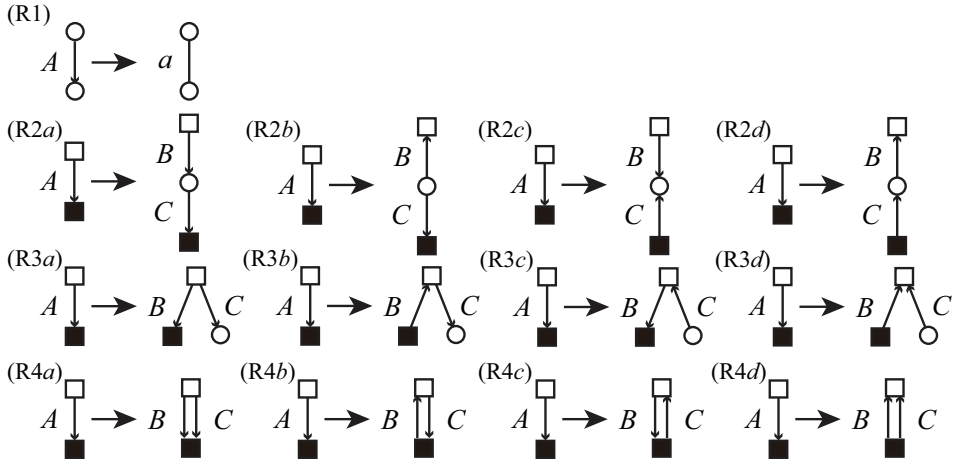

Figure 3: Four main types of production rules of undirected GSPGGs. A head and a tail of each arrow denote two terminal nodes of its edge. White and black squares mean that the node with a white (black) square in the left-hand side corresponds to the node with a white (black) square in the right-hand side.

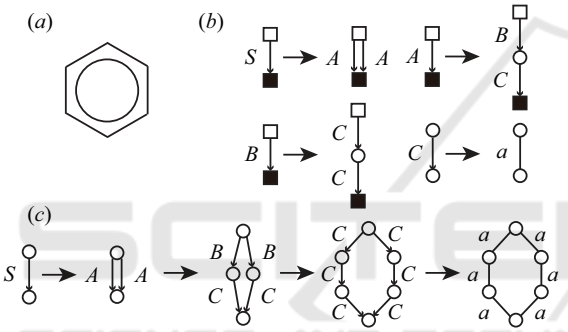

Figure 4: Example of an undirected generalized series-parallel graph and its grammar. (a) The benzene ring. (b) An undirected GSPGG of the benzene ring. (c) The derivation of the benzene ring using the grammar, where terminal symbol '*a*' denotes a bond with order 1.5 of the benzene ring.

side. In the production rule (R4a-d), an edge between the source and sink nodes is replaced with two edges, and a cycle is generated.

Fig. 4(b) shows an example of an undirected GSPGG that produces the benzene ring (Fig. 4(a)), where each bond in the benzene ring is represented as an edge with label '*a*' because six bonds are equivalent to each other. Fig. 4(c) shows the derivation of the benzene ring using the undirected GSPGG. The start symbol '*S*' is replaced with two nonterminal symbols '*A*' making a cycle. '*A*' is replaced with '*B*' and '*C*'. '*B*' is replaced with two '*C*'s. '*C*' is replaced with '*a*'. Then, the number of production rules is equal to the number of nonterminal symbols, $|\Sigma| = 4$ ($\Sigma = \{S, A, B, C\}$). For finding the minimum GSPGG, it is enough to find GSPGGs with the minimum number of nonterminal symbols.

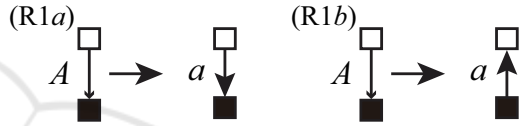

Figure 5: Production rules of replacement of a nonterminal symbol with a terminal symbol in directed GSPGGs. In each production rule, the arrow in the right-hand side denotes a directed edge.

Similarly to the definition of undirected GSPGGs, we define a directed GSPGG as follows.

Definition 4 (Directed generalized series-parallel graph grammar)**.** *A directed GSPGG is defined as 4-tuple (Σ, Γ, S, Δ), where Σ and Γ are sets of nonterminal and terminal symbols, every terminal symbol is a directed labeled edge, S is a start nonterminal symbol, Δ is a set of the same types of production rules of undirected GSPGGs except (R1), and (R1) is replaced with (R1a-b) as shown in Fig. 5.*

Fig. 6 shows an example of a directed GSPG and its grammar that produces only the graph, where the chemical structure of the purine (Fig. 6(a)) is transformed to a directed graph as shown in Fig. 6(b). If it is transformed to an undirected graph, two endpoints of a terminal symbol cannot be distinguished, and atom types are not determined in the graph produced by an undirected GSPGG.

2.4 The Minimum Directed and Undirected GSPGGs

Let $G(V,E)$ be a directed (undirected) GSPG with a set V of nodes and a set E of labeled edges. To consider all combinations of compositions of subgraphs

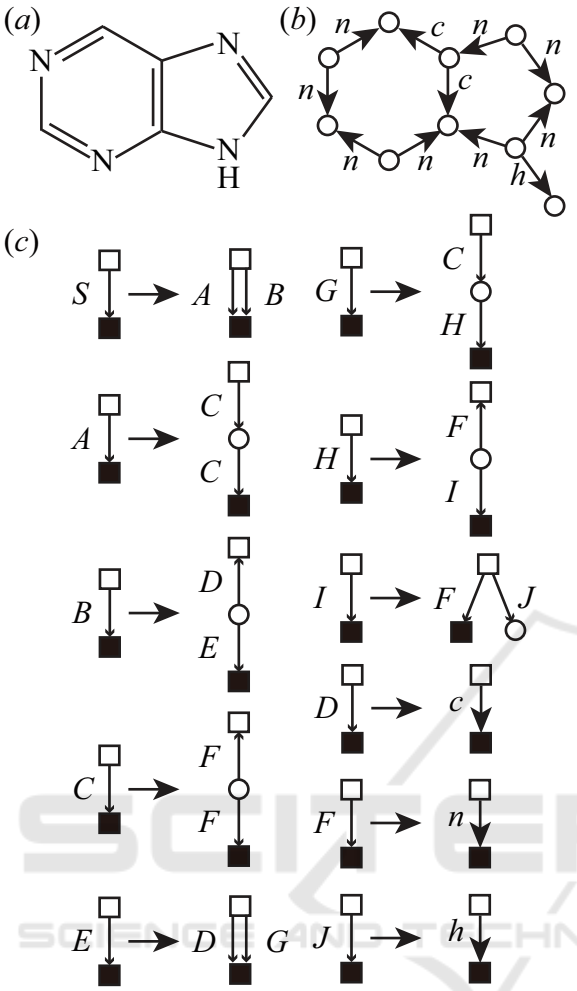

Figure 6: Example of a directed generalized series-parallel graph and its directed GSPGG. (a) The purine. (b) A transformed directed graph. (c) A directed GSPGG that generates the graph (b).

of G, we repeatedly partition subgraphs into two connected components until only edges remain. A GSPG has two terminal nodes, whereas G does not have any terminal node. Hence, we require that a partitioned subgraph has at most two terminal nodes. Suppose that $G_{i,S,j,T}$ represents a connected subgraph with terminal nodes i and j, where S and T are subsets of adjacent nodes of i and j, respectively. If a partitioned subgraph has one terminal node, we represent the subgraph as $G_{i,S}$, $G_{i,S,\varepsilon,\emptyset}$, or $G_{\varepsilon,\emptyset,i,S}$. G is also represented as $G_{\varepsilon,\emptyset,\varepsilon,\emptyset}$. If $G_{i,S,j,T}$ is isomorphic to $G_{i',S'}$ except node j, and is generated by a production rule, then $G_{i',S'}$ is also generated by the same production rule.

A subgraph with at most two terminal nodes can be partitioned into two subgraphs with one or two terminal nodes.

Fig. 7 shows an example of an undirected GSPG

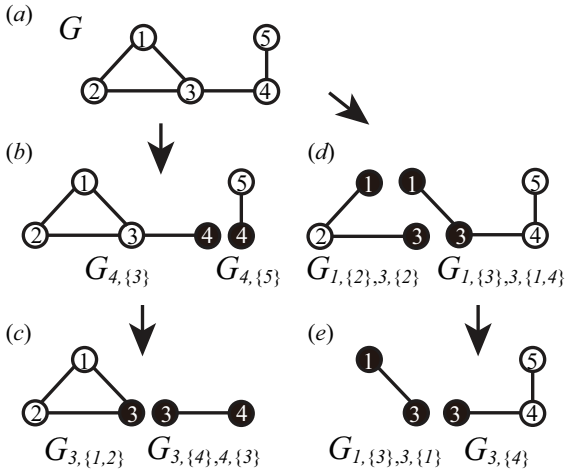

Figure 7: Example of an undirected GSPG and its partitioning. (a) An example graph G with five nodes. (b) The partitioned graphs $G_{4,\{3\}}$ and $G_{4,\{5\}}$ at node 4 of G. (c) The partitioned graphs $G_{3,\{1,2\}}$ and $G_{3,\{4\},4,\{3\}}$ at node 3 of $G_{4,\{3\}}$. (d) The partitioned graphs $G_{1,\{2\},3,\{2\}}$ and $G_{1,\{3\},3,\{1,4\}}$ at nodes 1 and 3 of G. (e) The partitioned graphs $G_{1,\{3\},3,\{1\}}$ and $G_{3,\{4\}}$ at node 3 of $G_{1,\{3\},3,\{1,4\}}$. A black circle denotes a terminal node.

G with five nodes and its partitioning. G is partitioned into two subgraphs $G_{4,\{3\}}$ and $G_{4,\{5\}}$ at node 4 that does not belong to any cycle as shown in Fig. 7(b). If a node to be partitioned does not belong to any cycle, only the node can be a new terminal node. Then, production rules of (R2) and (R3) can be constructed, and G is generated from $G_{4,\{3\}}$ and $G_{4,\{5\}}$ by series compositions. In Fig. 7(c), we cannot partition $G_{4,\{3\}}$ at node 1 or 2 because two connected components are not generated. $G_{4,\{3\}}$ is partitioned into $G_{3,\{1,2\}}$ and $G_{3,\{4\},4,\{3\}}$ at node 3. Then, production rules of (R2) and (R3) can be constructed, and $G_{4,\{3\}}$ is generated from $G_{3,\{1,2\}}$ and $G_{3,\{4\},4,\{3\}}$ by series compositions. On the other hand, if a node to be partitioned belongs to only a cycle, another node belonging to the cycle is needed. In Fig. 7(d), G is partitioned into $G_{1,\{2\},3,\{2\}}$ and $G_{1,\{3\},3,\{1,4\}}$ at nodes 1 and 3. Then, a production rule of (R4) can be constructed, and G is generated from $G_{1,\{2\},3,\{2\}}$ and $G_{1,\{3\},3,\{1,4\}}$ by parallel composition. In Fig. 7(e), $G_{1,\{3\},3,\{1,4\}}$ cannot be partitioned at node 4 because only subgraphs with at most two terminal nodes are allowed. Hence, $G_{1,\{3\},3,\{1,4\}}$ is partitioned into $G_{1,\{3\},3,\{1\}}$ and $G_{3,\{4\}}$ at node 3. Then, a production rule of (R3) can be constructed, and $G_{1,\{3\},3,\{1,4\}}$ is generated from $G_{1,\{3\},3,\{1\}}$ and $G_{3,\{4\}}$ by generalized series composition.

Suppose that $I(G)$ is a set of indices (i,S,j,T) of all subgraphs $G_{i,S,j,T}$ of G obtained by repeatedly partitioning, $\mathcal{S}(G)$ is a set of all distinct subgraphs $G_{i,S,j,T}$, and $\mathcal{E}(u)$ is a set of all sub-

graphs $G_{i,S,j,T}$ that are isomorphic to u. Consider the case that $G_{i,S,j,T}(V_{i,S,j,T}, E_{i,S,j,T})$ is correctly partitioned into $G_{i',S',j',T'}(V_{i',S',j',T'}, E_{i',S',j',T'})$ and $G_{i'',S'',j'',T''}(V_{i'',S'',j'',T''}, E_{i'',S'',j'',T''})$. Let $\mathcal{C}(G_{i,S,j,T})$ be a set of all index combinations $(i', S', j', T', i'', S'', j'', T'')$ that $V_{i',S',j',T'} \cup V_{i'',S'',j'',T''} = V_{i,S,j,T}$, $V_{i',S',j',T'} \cap V_{i'',S'',j'',T''} = \{i, j\}$, $E_{i',S',j',T'} \cup E_{i'',S'',j'',T''} = E_{i,S,j,T}$, $E_{i',S',j',T'} \cap E_{i'',S'',j'',T''} = \emptyset$, $E_{i',S',j',T'} \neq \emptyset$, and $E_{i'',S'',j'',T''} \neq \emptyset$ in such cases. Then, we propose the following integer linear programming formulation for finding the minimum directed and undirected GSPGGs that produce only a given generalized series-parallel graph G.

$$\text{Minimize} \sum_{u \in \mathcal{S}(G)} p_u$$

Subject to

$$x_{\varepsilon,\emptyset,\varepsilon,\emptyset} = 1, \quad (1)$$

$$x_{i,S,j,T} = 1 \quad \text{for all } (i,S,j,T) \in I(G) \text{ s.t. } |E_{i,S,j,T}| = 1, \quad (2)$$

$$x_{i,S,j,T} \leq \sum_{(i',S',j',T',i'',S'',j'',T'') \in \mathcal{C}(G_{i,S,j,T})} y_{i',S',j',T',i'',S'',j'',T''} \quad \text{for all } (i,S,j,T) \in I(G) \text{ s.t. } |E_{i,S,j,T}| \geq 2, \quad (3)$$

$$y_{i',S',j',T',i'',S'',j'',T''} \leq \frac{1}{2}(x_{i',S',j',T'} + x_{i'',S'',j'',T''}) \quad \text{for all } (i',S',j',T',i'',S'',j'',T'') \in \mathcal{C}(G_{i,S,j,T}), \quad (4)$$

$$p_u \geq \frac{1}{|E|} \sum_{G_{i,S,j,T} \in \mathcal{E}(u)} x_{i,S,j,T} \quad \text{for all } u \in \mathcal{S}(G), \quad (5)$$

$$p_u < 1 + \frac{1}{|E|} \sum_{G_{i,S,j,T} \in \mathcal{E}(u)} x_{i,S,j,T} \quad \text{for all } u \in \mathcal{S}(G), \quad (6)$$

$$x_{i,S,j,T}, y_{i',S',j',T',i'',S'',j'',T''}, p_u \in \{0,1\}.$$

In this formulation, $x_{i,S,j,T} = 1$ if $G_{i,S,j,T}$ is generated by the minimum GSPGG, otherwise 0. In the minimum SEO(U)TG, the Euler string is used to determine whether or not partitioned subtrees are isomorphic. However, it cannot be used for GSPGs, and we must investigate whether or not $G_{i,S,j,T}$ is isomorphic to $G_{i',S',j',T'}$. Eqs. (5) and (6) represent that $p_u = 1$ for $u \in \mathcal{S}(G)$ if and only if a subgraph $G_{i,S,j,T}$ isomorphic to u is generated by the minimum GSPGG, otherwise 0. The objective function indicates the number of nonterminal symbols in the grammar, and the integer linear programming problem finds the minimum GSPGG. Eq. (1) represents that G is constructed by the grammar. Eq. (2) represents that each edge in G is constructed by the grammar. Eq. (3) represents that $G_{i,S,j,T}$ is constructed by some production rule. Eq. (4) represents that a production rule can be candidate in the grammar if both of $G_{i',S',j',T'}$ and $G_{i'',S'',j'',T''}$ are constructed by the grammar. Since the problem of finding the minimum directed and undirected GSPGGs that produce only a given GSPG is NP-hard, it is reasonable to solve it by utilising integer linear programs.

3 CONCLUSION

In this paper, we proposed the definition of directed and undirected generalized series-parallel graph (GSPG) grammars, and an integer linear programming-based method for finding the minimum GSPG grammar that produces only a given GSPG. It has been proved that any outerplanar graph is a GSPG. We can find the minimum grammar for trees, outerplanar graphs, and GSPGs. As future work, we would like to apply our method to biological structured data, and extract production rules to construct the structure. Our integer linear programming formulation can take exponential time of the size of a GSPG. Hence, we would like to analyze the time complexity for the case that the degree of every node is less than a constant value. Furthermore, we would like to uncover what kind of graphs other than trees and outerplanar graphs can be handled by directed and undirected GSPGGs.

ACKNOWLEDGEMENTS

This work was partially supported by Grants-in-Aid #16K00392, #16KT0020, and #26240034 from JSPS, Japan.

REFERENCES

Akutsu, T. (2010). A bisection algorithm for grammar-based compression of ordered trees. *Information Processing Letters*, 110:815–820.

Eikel, M., Scheideler, C., and Setzer, A. (2015). Minimum linear arrangement of series-parallel graphs. *Lecture Notes in Computer Science*, 8952:168–180.

Eppstein, D. (1992). Parallel recognition of series-parallel graphs. *Information and Computation*, 98:41–55.

Ho, C., Hsieh, S., and Chen, G. (1999). Parallel decomposition of generalized series-parallel graphs. *Journal of Information Science and Engineering*, 15:407–417.

Hopcroft, J., Motwani, R., and Ullman, J. (2001). *Introduction to Automata Theory, languages, and Computation*, chapter Chapter 5: Context-Free Grammars and Languages, pages 169–218. Addison-Wesley, Boston, 2 edition.

Korneyenko, N. (1994). Combinatorial algorithms on a class of graphs. *Discrete Applied Mathematics*, 54:215–217.

Zhao, Y., Hayashida, M., and Akutsu, T. (2010). Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures. *BMC Bioinformatics*, 11(Suppl 11):S4.

Zhao, Y., Hayashida, M., Cao, Y., Hwang, J., and Akutsu, T. (2015). Grammar-based compression approach to extraction of common rules among multiple trees of glycans and RNAs. *BMC Bioinformatics*, 16:128.