Studies on Graph Optimization and Network Indicators in Economic Structure Analysis

土中, 哲秀

https://doi.org/10.15017/1931691

出版情報：九州大学, 2017, 博士（経済学）, 課程博士
バージョン：
権利関係：

Studies on
Graph Optimization and Network Indicators in Economic Structure Analysis

Thesis by

Tesshu Hanaka

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Kyushu University

Graduate School of Economics, Hakozaki, Fukuoka, Japan

February, 2018

Preface

In the field of economic structure analysis, a nature of economy as a network has been focused on. A network is a discrete structure that consists of (weighted) vertices and (weighted) edges that connect vertices. In the context of economic analysis, vertices represent industries, sectors, companies or individuals, and edges represent transactions between them. In weighted cases, the weight of a vertex or an edge represents its magnitude. For example, if a vertex corresponds to an industry, its production volume is represented as the vertex weight, and if an edge corresponds to a transaction between two industries, the amount of money or materials transferred between them is represented as the edge weight.

In general, a graphical/network representation gives an intuitive observation about the local/global structure of connections. Thus a typical usage of economic network model is to identify a group of industries/transactions that play a key role in economy. This type of analyses are also useful to capture flow of certain things on a network. In fact, in the field of environmental economics, economic network analysis is used to discover industries and transactions with high environmental burden of pollutants caused by economic activities.

The network analysis methods are roughly categorized into two approaches. One is graph optimization approach, and the other is network indicator approach. The graph optimization approach works as follows: We first model the task of analysis as a graph optimization problem. We then apply an algorithm to solve the problem, and obtain a solution. The solution implies analytical results about the network to be considered.

The network indicators reflect the characteristics of the network. Typically, these indicators are defined for vertices and/or edges, which represent their importance. The network indicator approach quantitatively argues the features of the network via the indicators.

These approaches have been applied to economic network analysis and they are promising in the

field of economic structure analysis. In this thesis, we reorganize and develop the graph optimization approach and the network indicator approach from the viewpoint of economic structure analysis.

February, 2018

Tesshu HANAKA

Acknowledgments

First of all, I would like to express my sincere appreciation to my supervisor, Professor Hirotaka Ono of Nagoya University for his unconditional support from the beginning to the end. He kindly gave me his deep knowledge, insightful suggestions, and generous encouragement. He also taught me that research is very exciting and pleasant. I am deeply grateful to him for his polite and persistent guidance. A lot of valuable experience he granted to me will become an irreplaceable treasure in my life. Without his dedicated help, I would never complete this dissertation.

I wish to express my deepest appreciation to Professor Shigemi Kagawa of Kyushu University for being a great adviser to me. He gave me his deep insight and extensive knowledge. I am indebted to him for his constructive advice and warm encouragement.

I would also like to express my deep gratitude to my committee members, Professor Tetsuya Furukawa and Professor Toshio Ohnishi of Kyushu University for their generous support and helpful comments.

My heartfelt gratitude goes to Professor Hans L. Bodlaender of Utrecht University, Professor Naomi Nishimura of University of Waterloo, Professor Keiichiro Kanemoto of Shinshu University, and Mr. Tom van der Zanden of Utrecht University for their kind cooperation as coauthors. I have greatly benefited from their scientific expertise, and their generous help and advice advanced my research.

I would like to thank Professor Eiji Miyano of Kyushu Institute of Technology, Professor Yasushi Kondo of Waseda University, Professor Yota Otachi of Kumamoto University, Professor Toshiki Saitoh of Kyushu Institute of Technology, Professor Yen Kaow Ng of Universiti Tunku Abdul Rahman, Professor Michael Lampis of Université Paris-Dauphine, Professor Jesper Nederlof of Eindhoven University of Technology, Professor Johan M. M. van Rooij of Utrecht University, Professor Hiroshi Eto of Kyushu University, Mr. Ioannis Katsikarelis of Université Paris-Dauphine,

and Mr. Krishna Vaidyanathan of University of Waterloo for many enlightening discussions.

It is impossible to conclude my acknowledgment without all the members of Ono laboratory. Since I started my research life, Dr. Omar Rifki and Mr. Petar Tzenov have supported me as my colleagues and my friends. The remaining members of Ono laboratory have also given me fulfilling days.

I would also like to thank Professor Yasunori Hanamatsu, Mr. Shota Tokunaga, and the remaining members of Decision Science for a Sustainable Society of Kyushu University for their constructive help.

Finally, but not least, I would like to thank my family, my mother, my father and my siblings. Without their care, support and encouragement, this dissertation would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A *network* is a pattern of interconnections among a set of things [21]. It consists of some objects, called *vertices* or *nodes*, and some pairs of the objects, called *edges* or *links*. Networks can express any "relationships" or "connections". Typical examples are computer networks, which represent the connection between computers, the World Wide Web, which represent the connections between web pages, traffic networks, in which vertices represent intersections or stations and edges represent roads or rails, friendship networks, which consist of vertices of persons and edges between them, and trade networks in which vertices represent industries or companies and edges represent direct transactions [21,86,87]. Also bioinformatics uses protein-protein interaction networks, which represent the physical contacts between proteins in the cell [47, 99].

By expressing a set of relations as a network, we can intuitively understand structures of relations. For example, by using social networks, we can observe how complicated human relations are, who is a key person in the group, which relationship between persons is strong, and so on. We can also find which route is the shortest in a map by using transportation networks, and analyze transaction structures among industries by using trade networks. As above, observing networks gives us a lot of insight and /or deep understanding.

On the other hand, the amount of information in the world rapidly increases nowadays, and many networks become large and complicated. Consequently, it gets harder and harder to observe the structure and the property of a network. To solve this situation, it is required to observe the

network by extracting the key structure, or developing indicators that reflect the structures without seeing the whole network.

For this, network analysis is studied in many research fields for a long time. Network analysis is a research method that analyzes the structures and properties of a network by using mathematical tools such as graph theory, linear algebra, and so on. Among them, the *graph optimization approach* is a typical approach of network analysis. This approach uses graph optimization problems and algorithms for solving them. Another well-known approach is *network indicator approach*, which uses indicators that represent the characteristics of a network.

In this thesis, we develop analytical methods for economic networks based on the graph optimization approach and the network indicator approach. An economic network consists of vertices corresponding to industries, sectors or companies and edges corresponding to transactions. That is, an economic network represents the economic structure and analyzing the economic network leads to the grasp of economic structure. Consequently, economic network analysis is useful to carry out appropriate economic policies and managements.

1.1.1 Graph optimization approach

In mathematical literature, a network is often called a *graph*. The graph optimization approach is a typical network analysis approach based on graph optimization problems and algorithms. In the graph optimization approach, we first model the task to discover a critical structure as a graph optimization problem. We then solve the problem by using the algorithm and obtain the desired structure.

For example, given a map, a person would like to go to the destination as fast as possible, he or she needs to take the "best" route on the map. How can he/she find such a route? This can be done by using the graph optimization approach: First, we describe the map as a graph. Next, we model the task to find an optimal route in the map as a graph optimization problem. In an edge-weighted graph, a *shortest path* from s to t is defined as a path from s to t with minimum weight. The weight of an edge represents the distance between two points. The task to find an optimal route can be modeled as the graph optimization problem to find a shortest path. This problem is called the SHORTEST PATH problem [25]. Finally, by solving the SHORTEST PATH problem, we can obtain an optimal route in the map.

A shortest path is one of the important graph structures, and there are other structures in the graph as well. In a graph, a *separator* is a subset of vertices that divides the graph into two parts. Similarly, a *cut* is a subset of edges that divides the graph into two parts. The problems to find a minimum separator and a minimum cut are well-studied as MINIMUM SEPARATOR and MINIMUM CUT, respectively. Finding a minimum separator and a minimum cut correspond to finding a bottleneck in the network. Also, when a cut is small, the two divided sets are less related in the sense. By dividing a network with minimum cuts iteratively, we can detect the dense structures that implies strong relations between vertices, so-called communities in the network. Splitting a large network into subnetworks also makes it easier to observe the network structure. The division of the network is used in various aspects such as community analysis, image segmentation, and so on [36,101,102]. The generic name of that method is *graph clustering*. The task is grouping the vertices of the graph into subsets of them, called clusters, such that there exist few edges between clusters [101]. A lot of clustering methods have been proposed since the criteria of goodness of division depends on the type of networks. Some of them are based on the graph optimization approach, such as finding minimum cuts iteratively [17, 26, 46, 50, 73, 102, 104, 115].

In the graph optimization approach, we need to design the algorithm to solve the corresponding graph optimization problem if any good algorithm is not known. From the aspect of algorithm design, the running time is a critical factor to obtain a solution at high speed. If a problem has a polynomial-time algorithm, it can be solved in realistic time as one theoretical criteria. In fact, polynomial-time algorithms are useful in practice. For example, SHORTEST PATH, MINIMUM SEPARATOR, and MINIMUM CUT have polynomial-time algorithms [25, 35].

On the other hand, if there is no polynomial-time algorithm for a problem, it may not be solved in realistic time when the size of input is large. Actually, there exist a lot of problems that are anticipated to have no polynomial-time algorithm, which are known as NP-hard (or NP-complete) problems. For example, FEEDBACK ARC SET and NORMALIZED CUT that will appear in Section 1.2.1 are NP-hard [59, 102].

To cope with such NP-hard problems, algorithm design has been studied in the field of computer science. For an NP-hard problem, an *approximation algorithm* is an efficient algorithm that finds an approximate solution. When designing an approximation algorithm, it is important how close the solution that the algorithm outputs is to an optimal solution, in other words, the accuracy of the

solution.

A *fixed-parameter algorithm* is another efficient algorithm for an NP-hard problem. The key concept of fixed-parameter algorithm is to give up the universality. Its running time is exponential for the worst case, but when the input satisfies certain conditions, it outputs an optimal solution in a realistic running time. These "efficient" algorithms could be important tools in the graph optimization approach in the case when the employed problem is NP-hard.

1.1.2 Network indicator approach

The network indicator approach is the other major method of network analysis. As the name implies, this approach uses network indicators, which reflect the characteristics of vertices, edges, and network itself.

An example of indicators for the whole network is *density*, which represents how dense a graph is. If the density of a social network is high, it means that a human relation of the group is close and strong.

Among network indicators for vertices and edges, one of the most well-known concepts of network indicators is *centrality*. The centrality quantifies how important vertices (or edges) are in a networked system [86]. Many kinds of centralities are previously defined [11, 14, 37–39, 44, 85, 86, 100, 103, 106]. The most basic centrality among them is *degree centrality* for a vertex. It is defined as the degree of a vertex, that is, the number of neighbors of a vertex in the network. For example, a person with many friends has the high degree centrality in the social network.

The *vertex betweenness centrality* is also a well-studied centrality index proposed by Freeman [37–39]. The vertex betweenness centrality of vertex v is defined as the number of shortest paths between pairs of other vertices that run through v [37–39, 44]. Intuitively, a vertex with the high betweenness centrality represents an important point such as a hub. Thus, in an aviation network, vertices corresponding to hub airports such as Narita, Incheon and Frankfurt could have the high vertex betweenness centralities. In a communication network, a vertex with the high betweenness centrality is also an important point where the information concentrates.

Similarly, the betweenness centrality is defined for an edge. The *edge betweenness centrality* is defined as the number of shortest paths between pairs of vertices that run along it [44, 88]. For example, in a traffic network, an edge with the high edge betweenness centrality represents a

road on which many cars concentrate. The edge betweenness centrality is also used for the graph clustering [21, 44, 84, 88, 101].

1.2 Network analysis in Economics

In Economics, economic structure analysis by using the concept of network has been studied. An *economic network* consists of vertices represent industries, sectors, companies or individuals, and edges represent transactions between them. In weighted cases, the weight of a vertex or an edge represents its magnitude. For example, if a vertex corresponds to an industry, its production volume is represented as the vertex weight, and if an edge corresponds to a transaction between two industries, the amount of money or materials transferred between them is represented as the edge weight. Economic networks include international trade networks, supply-chain networks, and so on. By using economic networks, we can analyze the economic structure in terms of networks and the results are used to carry out appropriate economic policies and managements.

One of the most well-known research fields which study economic network analysis is *input-output analysis*. Input-output analysis is a quantitative economic analysis framework [70, 72, 77]. This framework is based on an input-output table, which is a matrix such that its rows and columns correspond to sectors and its element (i, j) represents the amount of money or materials in the transaction from sector i to sector j. In the sense, an input-output table is an interindustry transactions table [77]. Since an input-output table is a matrix, we can interpret it as a transaction network between sectors. In the field of input-output analysis, we deal with not only simple transaction networks, but also direct and indirect transaction networks by using the *Leontief inverse matrix*. It can be regarded as supply-chain networks between sectors associated with the final demands of specific sectors. In other words, input-output analysis considers the economic ripple effect. We call an economic network based on the input-output model, in particular, an *input-output network*.

1.2.1 Graph optimization approach for economic networks

Input-output analysis has long been related to the graph optimization approach. The *triangulation* is one of the traditional graph optimization methods in input-output analysis. The triangulation in input-output analysis reveals the characteristics of economic structures [63, 64, 71, 82]. This

method rearranges rows and columns of input-output tables so that the sum of the upper triangular elements is maximum. In other words, it finds such a linear ordering of rows and columns. In terms of the graph optimization approach, the triangulation is well-known as the (weighted) FEEDBACK ARC SET problem, which is a well-studied graph optimization problem [59, 63]. The triangulation, or FEEDBACK ARC SET extracts a main stream of the economic structure from the upstream sectors to the downstream sectors in the sense.

Input-output analysis framework can be also extended to environmental analysis by using environmentally extended input-output tables [55–57, 68, 69, 91, 114]. We call a network based on an environmentally extended input-output table an *environmental input-output network*. In an environmental input-output network, the weight of an edge represents the amount of embodied emissions such as CO_2 associated with a transaction. Kagawa et al. proposed a new graph optimization method for the environmental input-output network analysis by using graph clustering [55–57]. This method can identify industrial clusters with the high environmental emissions and is expected to lead to more efficient environmental policies that encourage the collaboration between industries in the cluster than those for a specific industry. In this method, they found industrial clusters by solving NORMALIZED CUT, which is a graph optimization problem [102].

1.2.2 Network indicator approach for economic networks

In the economic field, there are many traditional indicators, such as GDP, consumer price index, and unemployment rate, and so on. Some of them are developed a long time ago. Recently, the development of indicators based on economic networks is started. The merit of indicators based on economic networks is that we can observe the critical economic structures quantitatively. As the result, we can carry out efficient policies.

The economic network indicators are mainly studied on environmental economic analysis. Liang et al. proposed the vertex betweenness centrality for environmental input-output networks [75]. The vertex betweenness centrality in environmental input-output networks is defined as the sum of environmental emissions associated with the supply-chain paths passing through the specific sector. It can identify a critical transmission sector in the sense that many sectors supply their products to final consumers by passing through the specific sector, and consequently the transmission sector contributes to emitting a large amount of environmental emissions in the economy. The

point of betweenness centrality for an environmental input-output network is considering the economic ripple effect and the economic circulation. That is, this indicator is specialized in economic networks.

1.3 Motivation and contribution

As seen the previous sections, the graph optimization approach and the network indicator approach for economic network analysis are making achievements and promising. In the graph optimization approach, the triangulation method and the graph clustering approach for economic networks were introduced in Section 1.2.1. The triangulation method extracts a directed acyclic subgraph that represents a main stream of the economic structure and the graph clustering approach for an environmental input-output network finds industrial clusters with the high emissions. However, the tools of economic network analysis are still inadequate since there are many types of structures to analyze, for example, a vulnerable part in supply chains, transactions with high environmental impacts, and so on. Thus, we need to expand the methods by modeling the task to discover the other important structures in economic networks as a graph optimization problem and by designing its algorithm. The same is true for the network indicator approach. Although Liang et al. proposed a network indicator for industries in economic networks [75], there are a lot of objectives of network indicators (e.g., transactions and a whole network). On the basis of these principles, we expand the methods of economic network analysis in terms of the graph optimization and the network indicator approaches in this thesis.

When we analyze economic networks, there are important characteristics of them. Among them, we focus on two important characteristics, “direction” and “weight”. The direction of an edge is one of the essential elements in economic network analysis. When a policymaker considers an appropriate economic policy for some industries, he or she has to take the properties of industries into account since the upstream industry and the downstream industry in a transaction differ in the situation such as a business model, production system, service, marketing and so on. Thus, the edge direction in an economic network is clearly essential in order to make economic policies considering the characteristics of industries.

Having weights is another important feature of an economic network. In an economic network,

the vertex weight represents the sector's importance such as the production volume and the amount of emissions. Also the edge weight represents the amount of money, materials or emissions associated with the transaction. In the sense, the edge weight reflects how important and strong a relationship between industries is.

In the graph optimization approach, a lot of graph optimization problems and their algorithms are proposed, but many of them are for undirected and unweighted graphs. Since the direction and the weight are essential characteristics of economic networks described above, we need a graph optimization problem that takes them into account and its algorithm to find a good solution at high speed. Following these concepts, we re-organize and expand the graph optimization approach for economic networks. In detail, we propose two types of new graph optimization problems. One focuses on the direction, and the other focuses on the weight. For both of them, we present high-performance algorithms for them.

For the former, we define two graph optimization problems on directed graphs, called DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET. These problems are studied in Chapter 3. The motivation of these problems comes from the extraction of critical industries and transactions in a "directed" economic network. In theoretical computer science, these problems are categorized as variants of classical graph optimization problems on undirected graphs, which are known as VERTEX COVER and EDGE DOMINATING SET [41, 116].

For the latter, as a graph optimization problem focusing on the weight, we define the MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR problem in Chapter 4. The problem is to find a maximum weight minimal (s-t) separator, which is a variant of separators in graphs. The point is that each vertex in the graph has the weight. This problem arises in the context of supply-chain network analysis. When a vertex-weighted network represents a supply chain where a vertex represents an industry, s and t are virtual vertices representing source and sink respectively, and the weight of a vertex represents its financial importance, the maximum weight minimal s-t separator is interpreted as the most important set of industries that is influential or vulnerable in the supply-chain network.

Unfortunately, these problems are NP-hard as shown in Chapters 3 and 4. On the other hand, we design fixed-parameter algorithms for DIRECTED r-IN (OUT) VERTEX COVER, DIRECTED (p, q)-EDGE DOMINATING SET, and MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR. This implies that we can obtain the optimal solutions in polynomial time when the instances satisfy

certain conditions. We also give approximation algorithms for DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET in Chapter 3.

As for the network indicator approach, we propose a new network indicator for environmental input-output networks in this thesis. In a previous study, Liang et al. proposed the vertex betweenness centrality in an environmental input-output network [75]. However, the vertex betweenness centrality is not necessarily useful to observe the connections between sectors since it is a network indicator for a sector. To grasp the economic structure in detail, a network indicator specified for a transaction is preferable.

We newly develop such a network indicator for a transaction in Chapter 5. The indicator is named the *edge betweenness centrality* for an environmental input-output network. The edge betweenness centrality in an environmental input-output network indicates how much 'embodied' environmental emissions of products flow through the transaction and to what extent sectors are connecting through a specific edge (i.e., a transaction) in terms of supply-chain complexity. As with the vertex betweenness centrality, the edge betweenness centrality considers the economic ripple effect and the economic circulation. Thus, this indicator is also specialized in economic networks.

Moreover, we calculate the vertex and edge betweenness centralities for a real input-output table. We then identify critical sectors and transactions for environmental and economic policies to reduce the pollutions efficiently. We also visualize the CO_2 circulation networks in global supply chains based on the vertex and edge betweenness centralities. Visualizing networks is useful to understand the fundamental characteristics of them intuitively [47, 98, 99]. In economic network analysis, the visualization of economic networks also gives useful information and leads good implications [55–57, 69, 89, 96]. We reveal the environmental economic structures in global supply chains by the visualization of economic networks based on the vertex and edge betweenness centralities.

1.4 Thesis overview

This thesis is organized as follows. Chapter 2 is the preliminary part. We give common definitions and notations used in this thesis. We also introduce a very useful technique of algorithm design, which is called a *tree decomposition*.

In Chapter 3, we give two graph optimization problems on directed graphs, called DIRECTED

r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET. For these problems, we first study the computational complexity. We show that DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET are NP-complete on restricted graphs. Moreover, we prove that if r is larger than one, DIRECTED r-IN (OUT) VERTEX COVER is $W[2]$-hard and if p or q is larger than one, DIRECTED (p, q)-EDGE DOMINATING SET is $W[2]$-hard on directed acyclic graphs. For these problems, we also showed that there is no polynomial-time $c \ln k$-approximation algorithm for any constant $c < 1$ unless P=NP, where k is the size of an optimal solution, though they can be approximated within ratio $O(\log n)$ by a greedy algorithm. On the other hand, we give polynomial-time algorithms on trees. Moreover, we show that DIRECTED (p, q)-EDGE DOMINATING SET is fixed-parameter tractable with respect to treewidth when $(p, q) = (0, 1), (1, 0), (1, 1)$. Finally, we design a $2^{O(k)}n$-time algorithm for DIRECTED (p, q)-EDGE DOMINATING SET when $(p, q) = (0, 1), (1, 0), (1, 1)$ where k is the solution size.

In Chapter 4, we study the MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR problem. We first prove that MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR is NP-hard on bipartite graphs. On the other hand, we show that MAXIMUM WEIGHT MINIMAL SEPARATOR is fixed-parameter tractable with respect to the weight of the solution. We then propose two fixed-parameter algorithms for MAXIMUM WEIGHT (s-t) MINIMAL SEPARATOR with respect to treewidth. One is an $\omega^{O(\omega)}n^{O(1)}$-time deterministic algorithm and the other is a $9^{\omega}W^2n^{O(1)}$-time randomized algorithm, where ω is the width of a tree decomposition.

In Chapter 5, we propose a new analytical method based on economic network indicators to find environmentally critical transmission sectors, transactions and paths in global supply-chain networks. We first introduce the edge betweenness centrality for input-output analysis. Then we give the mathematical relationship between the edge betweenness centrality proposed in Chapter 5 and the vertex betweenness centrality proposed by Liang et al. [75]. As the empirical analysis, we use the world input-output database (WIOD) covering 35 industrial sectors and 41 countries and regions in 2008 [24, 109]. We compute the vertex and edge betweenness centralities for WIOD. Moreover, we visualize the CO_2 circulation networks in global supply chains based on the vertex and edge betweenness centralities. Finally, we discuss effective environmental policies from the results.

In Chapter 6, we summarize our study in this thesis and discuss our contribution to this field.

Chapter 2

Preliminaries

In this chapter, we give notations and definitions. A graph is an ordered pair $G = (V(G), E(G))$ consisting of the vertex set $V(G)$ and the edge set $E(G)$. We denote a vertex in $V(G)$ by $v \in V(G)$ and an edge of a pair of vertices (u, v) in $E(G)$ by $(u, v) \in E(G)$. We sometimes denote an edge by e, that is, $e = (u, v)$ for simplicity. We will generally use $n = |V(G)|$ and $m = |E(G)|$ as the number of vertices and edges of G, respectively. For simplicity, we sometimes denote $V(G)$ and $E(G)$ by V and E, respectively.

For two vertices u, v, if $(u, v) \in E(G)$, u and v are said to be *adjacent* in G. If an edge (u, v) is unordered, a graph is said to be *undirected*. On the other hand, if an edge (u, v) is ordered, a graph is said to be *directed* and edge (u, v) is oriented from u to v. For example, given a vertex set $V(G) = \{u, v, w\}$ and an edge set $E(G) = \{(u, v), (v, w)\}$, an undirected graph G is depicted in Figure 2.1, and a directed graph G is depicted in Figure 2.2.

A graph $H = (V(H), E(H))$ of G is a *subgraph* if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We

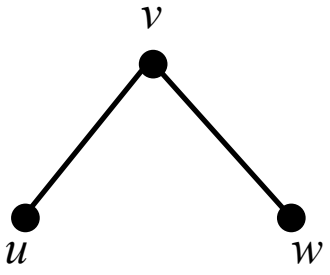

Figure 2.1. An undirected graph

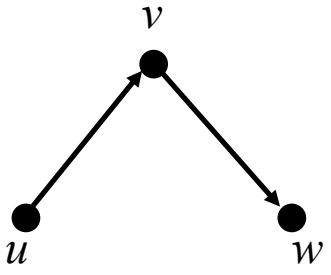

Figure 2.2. A directed graph

denote the subgraph of G that consists of $V' \subseteq V(G)$ by $G(V')$. A subgraph H of G is said to be *induced* by V' if H contains every edge in $E(G)$ whose endpoints are both in V'. For $V' \subseteq V(G)$, we denote by $G[V']$ the subgraph of G induced by V'.

A *path* from v_1 to v_k in a directed or undirected graph G is a vertex sequence $v_1, v_2, \ldots, v_k$ such that $v_i \neq v_j$ for $1 \leq i < j \leq k$ and $(v_i, v_{i+1}) \in E(G)$ for $1 \leq i \leq k-1$. The *length* of a path is defined as the number of edges, that is, $k-1$. Similarly, a *cycle* is a vertex sequence $v_1, v_2, \ldots, v_k$ such that $v_i \neq v_j$ for $1 \leq i < j \leq k$, $(v_i, v_{i+1}) \in E(G)$ for $1 \leq i \leq k-1$, and $(v_k, v_1) \in E(G)$. The length of a cycle is defined as the number of edges. In this case, it is k. For two vertices $u, v \in V$, if the length of a path from u to v is minimum, it is called *shortest*. The *distance* from u to v, denoted by $dist(u, v)$, is defined as the length of a shortest path from u to v. Note that if a graph is undirected, $dist(u, v) = dist(v, u)$.

2.1 Undirected graph

We say an undirected graph G is *connected* if there exists a path from u to v for any $u, v \in V(G)$. A *connected component* of G is an inclusion-wise maximal connected induced subgraph.

In an undirected graph, a vertex u is called a *neighbor* of v if there exists an edge (u, v). A vertex u such that $dist(u, v)$ is at most r is called an *r-neighbor* of v. We denote the set of neighbors of v by $N(v)$ and the set of r-neighbors of v by $N_r(v)$. Note that $N_1(v) = N(v)$. The number of neighbors of v is called the *degree* of v and it is denoted by $d(v) := |N(v)|$. We define $\max_{v \in V} d(v)$ as the maximum degree of undirected graph G.

A set of edges without common vertices is called a *matching*. Furthermore, a matching is *maximal* if no proper superset is a matching. An edge dominating set is an edge set X such that every edge in $E \setminus X$ is adjacent to at least one edge in X. Therefore, a maximal matching is an edge dominating set.

2.2 Directed graph

In a directed graph, a vertex u is called an *in-neighbor* of v if there exists an edge (u, v) and a vertex w is called an *out-neighbor* of v if there exists an edge (v, w). Moreover, the set of in (out)-neighbors of v is denoted by $N^{in}(v)$ (resp., $N^{out}(v)$). The number of in (out)-neighbors of v

is called the *in (out)-degree* of v and denoted by $d^{in}(v) := |N^{in}(v)|$ (resp., $d^{out}(v) := |N^{out}(v)|$). A vertex u such that $dist(u, v)$ is at most r is called an *r-in-neighbor* of v and a vertex w such that $dist(v, w)$ is at most r is called an *r-out-neighbor* of v. The set of r-in (out)-neighbors of v is denoted by $N_r^{in}(v)$ (resp., $N_r^{out}(v)$). Note that $N_1^{in}(v) = N^{in}(v)$ and $N_1^{out}(v) = N^{out}(v)$. We define $\Delta(G) = \max_{v \in V}(d^{in}(v) + d^{out}(v))$ as the maximum degree of directed graph G. For simplicity, we sometimes use Δ instead of $\Delta(G)$.

2.3 Graph classes

Undirected graph

An undirected graph is a *forest* if it has no cycle. In particular, a forest is called a *tree* if it is connected. A graph is *bipartite* if a graph does not contain any odd-length cycles. If a graph can be embedded in the plane without any edges crossing, it is called a *planar graph*. A graph is *r-regular* if $d(v) = r$ for any vertex v. In particular, a 3-regular graph is called a *cubic graph*. A graph is *chordal* if every cycle of length at least four has a *chord*, which is an edge that is not part of the cycle but connects two vertices in the cycle.

Directed graph

A directed graph G is a forest, a tree, and a planar graph if the underlying undirected graph of G is a forest, a tree, and a planar graph, respectively. A directed graph G is called a *directed acyclic graph* (*DAG*) if G has no directed cycle.

2.4 Algorithm design for NP-hard problems

In the graph optimization approach, we design an algorithm to solve a graph optimization problem. From the aspect of algorithm design, the running time of an algorithm is a critical factor. In fact, whether the algorithm can fast compute the solution depends on the running time. Generally, we evaluate the running time by using a function of the input size n. Also we often use the upper bound of the running time in which any instance of the problem can be solved by the algorithm, in other words, we evaluate the worst running time for any instance. We denote the running time of an algorithm by $O(f(n))$ by using big O notation, which is a kind of the Bachmann-Landau notations.

Although the definition of the running time of an algorithm that can be solved in "realistic time" is ambiguous, one criterion is whether the running time is polynomial in n or not. An algorithm is said to be a *polynomial-time algorithm* in n if its running time is $O(n^c)$ where c is some constant, for example, $O(n), O(n^2), n \log n, O(n^{100})$. If there exists a polynomial-time algorithm for a problem, it is considered to be tractable in the theoretical sense. Such problems are said to be in P.

On the other hand, if there is no polynomial-time algorithm for a problem, it may not be solved in realistic time when the input size is large. In fact, there exist a lot of problems that are anticipated to have no polynomial-time algorithms. Such problems are known to be NP-hard (or NP-complete if problems are decision problems). To prove that a problem is NP-hard, we use a *polynomial-time reduction* from the other NP-hard problem to the problem [65]. If there is a polynomial-time reduction, it implies that the problem is as hard as the other one or more.

An *approximation algorithm* is an efficient algorithm for an NP-hard problem that find the approximate solution. When designing an approximation algorithm, it is important how close the solution is to the optimal solution, in other words, the accuracy. For this, by giving a mathematical proof, we evaluate the accuracy of an approximation algorithm. Such an algorithm is described as an α-approximation algorithm and α is called an *approximation ratio* [111]. The approximation ratio of an algorithm represents the accuracy guarantee of the solution. The closer approximation ratio α is to 1, the better solution the algorithm outputs.

A *fixed-parameter algorithm* or an *FPT algorithm* is the other efficient algorithm for an NP-hard problem. Given a parameter k and an input size n, a fixed-parameter algorithm is defined as an algorithm that computes an optimal solution in time $f(k)n^c$ where c is a constant independent of both n and k, and f is a computational function. In a sense, a fixed-parameter algorithm is conditional because we can regard the algorithm as a polynomial-time algorithm if k is enough small. The parameter k depends on the input instance. When designing a fixed-parameter algorithm, the goal is to make both the $f(k)$ factor and the constant c in the bound on the running time as small as possible [19]. The problem that has a fixed-parameter algorithm in time $f(k)n^c$ is said to be *fixed-parameter tractable* with respect to k.

The *parameterized complexity* is computational complexity in terms of parameters and the *W*-

Hierarchy is a hierarchy of complexity classes as follows [33]:

$$\text{FPT} = W[0] \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[t] \subseteq \cdots \subseteq W[SAT] \subseteq \cdots \subseteq W[P].$$

As for fixed-parameter tractability, it is most important that FPT $= W[0]$ versus $W[t]$-hard where $t \geq 1$. Intuitively, this relation is similar to P versus NP. In other words, if the problem is $W[t]$-hard where $t \geq 1$, then it does not presumably have a fixed-parameter algorithm.

2.5 Tree decomposition

In this section, we introduce the definition of tree decomposition.

Definition 2.5.1. A *tree decomposition* of an undirected graph $G = (V, E)$ is defined as a pair $\langle \mathcal{X}, T \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N \subseteq V\}$, and T is a tree whose nodes are labeled by $I \in \{1, 2, \ldots, N\}$, such that:

1. $\bigcup_{i \in I} X_i = V$,

2. For all $\{u, v\} \in E$, there exists an X_i such that $\{u, v\} \subseteq X_i$,

3. For all $i, j, k \in I$, if j lies on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

In the following, we use the term "nodes" (not "vertices") for the elements of T to avoid confusion. Moreover, we call a subset of V corresponding to a node $i \in I$ a *bag* and denote it by X_i. The *width* of a tree decomposition $\langle \mathcal{X}, T \rangle$ is defined by $\max_{i \in I} |X_i| - 1$, and the *treewidth* of G, denoted by $\mathbf{tw}(G)$, is the minimum width over all tree decompositions of G. We sometimes denote $\mathbf{tw}(G)$ by $\mathbf{tw}$ for simplicity. Roughly speaking, the treewidth is a graph parameter that means how much a graph is similar to a tree. If the treewidth of a graph is one, it is a forest. Thus, the treewidth is small implies a graph is tree-like. Figure 2.3 shows a tree decomposition of a graph. Since a tree-like structure of a graph enables to solve the problem efficiently, a tree decomposition of the small width is desirable. However, in general, computing the treewidth of a given graph G is NP-hard [1].

On the other hand, it is known that computing treewidth is fixed-parameter tractable with respect to itself and there exists a linear time algorithm if treewidth is fixed [7]. Moreover, there are FPT approximation algorithms for computing treewidth [8, 19].

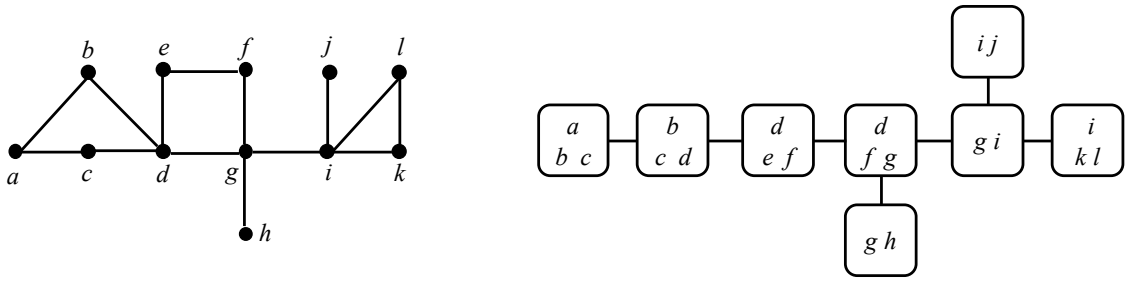

Figure 2.3. A graph (left) and its tree decomposition of width two (right)

We also use a very useful type of tree decomposition for use in algorithms, called a *nice tree decomposition* introduced by Kloks [61]. More precisely, it is a special binary tree decomposition which has four types of nodes, named *leaf*, *introduce vertex*, *forget* and *join*. A variant of the notion, using a new type of node named *introduce edge*, was introduced by Cygan et al. [20].

Definition 2.5.2. A tree decomposition $\langle \mathcal{X}, T \rangle$ is called a *nice tree decomposition* if it satisfies the following:

1. T is rooted at a designated node $r \in I$ satisfying $|X_r| = 0$, called the *root node*.

2. Each node of the tree T has at most two children.

3. Each node in T has one of the following five types:

- A *leaf* node i which has no children and its bag X_i satisfies $|X_i| = 0$,
- An *introduce vertex* node i has one child j with $X_i = X_j \cup \{v\}$ for a vertex $v \in V$,
- An *introduce edge* node i has one child j and labeled with an edge $(u, v) \in E$ where $u, v \in X_i = X_j$,
- A *forget* node i has one child j and satisfies $X_i = X_j \setminus \{v\}$ for a vertex $v \in V$,
- A *join* node i has two children nodes j_1, j_2 and satisfies $X_i = X_{j_1} = X_{j_2}$.

We can transform any tree decomposition to a nice tree decomposition with $O(n)$ nodes in linear time [19]. Without loss of generality, we can assume that the parent node of an introduce edge (u, v) node is an introduce edge (u, w), introduce edge (v, x), forget u, or forget v node for some w or x [19].

Chapter 3

On Directed Covering and Domination Problems

3.1 Introduction

Covering and domination problems are well-studied problems in theory and in applications of graph algorithms, for example, VERTEX COVER [41], DOMINATING SET [41] and EDGE DOMINATING SET [116]. However, almost all of these problems are studied on undirected graphs. In particular, VERTEX COVER and EDGE DOMINATING SET on directed graphs have not been studied although there are some results on directed DOMINATING SET [18, 29, 40, 66]. This seems surprising, but maybe one reason might be that it is difficult to expand the definition naturally to directed graphs due to the unclear relationship between "direction" and "domination".

In this chapter, we study directed versions of VERTEX COVER and EDGE DOMINATING SET. First, we give formal definitions of directed VERTEX COVER and directed EDGE DOMINATING SET. In the definitions, we consider several scenarios that reflect how the selected set influences edges via directed edges. It should be noted that the definition follows from r-DOMINATING SET [22, 30, 66]. These definitions are also motivated by economic network analysis. We mention applications of these problems in Section 3.1.2.

In a directed graph, vertex v is said to *in-cover* every incoming edge (u, v) and *out-cover* every outgoing edge (v, u) for some u. A vertex v is also said to *r-in-cover* all edges in the directed path to v of length at most r. Similarly, v is said to *r-out-cover* all edges in the directed path from v.

Here, for a path $v_1, v_2, \ldots, v_\ell$, the *length* of the path is defined as the number of edges, that is, $\ell - 1$. In particular, if $r = 0$, a vertex is not considered to cover any edge. Then DIRECTED r-IN (OUT) VERTEX COVER is the following problem.

Definition 3.1.1. DIRECTED r-IN (OUT) VERTEX COVER (r-IN (OUT) VC) is the problem that given a directed graph $G = (V, E)$ and two positive integers k and r, determines whether there exists a vertex subset $S \subseteq V$ of size at most k such that every edge in E is r-in (out)-covered by S. Such an S is called an *r-in (out)-vertex cover*.

Furthermore, we define DIRECTED (p, q)-EDGE DOMINATING SET. An edge $e = (u, v)$ is said to (p, q)-*dominate* itself and all edges that vertex u p-in-covers and vertex v q-out-covers. In particular, edge (u, v) is said to $(p, 0)$-*dominate* (resp., $(0, q)$-*dominate*) itself and all edges p-in-covered by u (resp., q-out-covered by v).

Then DIRECTED (p, q)-EDGE DOMINATING SET is defined as follows.

Definition 3.1.2. DIRECTED (p, q)-EDGE DOMINATING SET ((p, q)-EDS) is the problem that given a directed graph $G = (V, E)$, one positive integer k, and two non-negative integers p, q, determines whether there exists an edge subset $K \subseteq E$ of size at most k such that every edge is (p, q)-dominated by K. Such a K is called a (p, q)-*edge dominating set*.

We also define DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET as optimization problems, that is, the problems are to find a minimum r-in (out)-vertex cover and a minimum (p, q)-edge dominating set, respectively.

The undirected EDGE DOMINATING SET problem is DOMINATING SET on (undirected) line graphs. We can see the same relationship between DIRECTED $(0, 1)$-EDGE DOMINATING SET and DOMINATING SET on directed line graphs. For a directed graph, a *directed line graph* is defined as follows:

Definition 3.1.3 ([49]). A *directed line graph* of $G = (V, E)$ is $L(G) = (E, E_2)$ such that

$$E_2 = \{((x, y), (z, w)) \mid (x, y), (z, w) \in E \wedge y = z\}.$$

It is obvious that a directed $(0, 1)$-edge dominating set on a directed graph G corresponds to a (directed) dominating set on the line graph of G. Furthermore, DIRECTED $(1, 1)$-EDGE DOMI-

Graph class	Tree	Planar DAG of bounded Δ	DAG	General
1-IN (OUT) VC	-	-	-	$O(n)$
r-IN (OUT) VC ($r \geq 2$)	$O(r(r+\Delta)n)$	NP-c	$W[2]$-h	$W[2]$-h
$(0,1),(1,0)$-EDS	$O(n)$	NP-c	NP-c	$2^{O(k)}n$
$(1,1)$-EDS	$O(n)$	NP-c	NP-c	$2^{O(k)}n$
(p,q)-EDS (p or $q \geq 2$)	$O(\gamma^2\Delta^2 n)$	NP-c	$W[2]$-h	$W[2]$-h

Table 3.1. Our results for graph classes. NP-c and $W[2]$-h stand for NP-complete and $W[2]$-hard, respectively. Note that $\gamma = \max\{p,q\}$ and $\Delta = \max_{v \in V}(d^{in}(v) + d^{out}(v))$.

NATING SET corresponds to undirected DOMINATING SET on the underlying undirected graph of a directed line graph. These relations imply that our definition of DIRECTED (p,q)-EDGE DOMINATING SET is quite natural from the viewpoint of line graphs.

One interesting aspect of directed versions, but not undirected versions, is the asymmetry of the problem structures. For DIRECTED r-IN VERTEX COVER, a vertex in-covers only incoming edges when $r = 1$. Thus, all vertices whose in-degree is at least one must be included the solution to cover every edge due to the asymmetry of covering. Therefore, it is trivial that DIRECTED 1-IN (OUT) VERTEX COVER is solvable in linear time, while undirected VERTEX COVER is NP-complete. The problem DIRECTED $(1,1)$-EDGE DOMINATING SET, in a sense, corresponds to (undirected) EDGE DOMINATING SET. For the optimization version, EDGE DOMINATING SET is equivalent to MINIMUM MAXIMAL MATCHING [116]. However, DIRECTED $(1,1)$-EDGE DOMINATING SET does not necessarily correspond to matching on the undirected graphs underlying directed graphs due to the asymmetry of domination.

For DIRECTED (p,q)-EDGE DOMINATING SET, there exists another source of asymmetry. That is, we can consider the case in which p and q are different. In the case in which $(p,q) = (0,1)$, edge (u,v) dominates itself and edges out-covered by v. Although DIRECTED $(0,1)$-EDGE DOMINATING SET is similar to DIRECTED 1-OUT VERTEX COVER, surprisingly, it is NP-complete even on directed acyclic graphs.

3.1.1 Our contributions

Table 3.1 shows our results. In this chapter, we first give hardness results for DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p,q)-EDGE DOMINATING SET on restricted graphs,

even on directed acyclic planar graphs of bounded degree. The hardness on directed acyclic graphs implies that we cannot design parameterized algorithms with respect to *directed treewidth* [54] and *DAG-width* [4] unless P=NP. The fact that DIRECTED $(0, q)$-EDGE DOMINATING SET is NP-complete even if $q = 1$ implies that r-DOMINATING SET on directed line graphs is NP-complete even if $r = 1$. Moreover, we prove that DIRECTED r-IN (OUT) VERTEX COVER is $W[2]$-hard and $c \ln k$-inapproximable on directed acyclic graphs when $r \geq 2$, and DIRECTED (p, q)-EDGE DOMINATING SET is $W[2]$-hard and $c \ln k$-inapproximable on directed acyclic graphs when either p or q is greater than 1. These results hold even if there are no multiple edges or loops.

On the other hand, we obtain algorithms for certain cases, including algorithms for all problems when restricted to trees, for any values of p, q, and r. The interplay among distance, direction, and domination results in a complex dynamic programming solution for DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET, running in time $O(r(r + \Delta)n)$ and $O(\gamma^2\Delta^2 n)$, respectively. Note that $\gamma = \max\{p, q\}$ and $\Delta = \max_{v \in V}(d^{in}(v) + d^{out}(v))$. Because an edge can either dominate or be dominated by edges outside of a subtree depending on how it is directed, at each step of the algorithm we need to maintain extensive information not only about the subtree itself but also potential outside influence.

We also show that DIRECTED $(0, 1)$-EDGE ($(1, 0)$-EDGE, $(1, 1)$-EDGE) DOMINATING SET can be solved in linear time on graphs whose underlying undirected graphs have bounded treewidth. Note that given a directed graph G and its underlying undirected graph G^*, the directed treewidth of G is no greater than its DAG-width which, in turn, is no greater than the treewidth of G^* [4].

Finally, we show that DIRECTED $(0, 1)$-EDGE ($(1, 0)$-EDGE, $(1, 1)$-EDGE) DOMINATING SET is fixed-parameter tractable with respect to k. In particular, we give $2^{O(k)}n$-time algorithms. We emphasize that the running time of these algorithms is single exponential in k and linear in n.

3.1.2 Motivation and application

As practical motivation, a number of network models employ directed graphs. For example, directed graphs are used to represent economic networks in which vertices correspond to industries and edges correspond to transactions of money or materials between industries [48, 67].

Recently, economists have used graph algorithms to analyze these economic networks in terms of graph structures in order to find critical industries and transactions [56,96]. Based on the analyses,

economists discuss which kinds of economic policies should be adopted, and so on. However, there are some problems. Such analyses in economics are based on undirected graph algorithms instead of directed graph algorithms; the algorithms first transform directed graphs to undirected graphs, and then apply undirected graph algorithms to the graphs thus obtained. This is because there are many more results on graph optimization on undirected graphs than on directed graphs. Of course, such substitute algorithms might extract some information from the processed graph, but some important information is definitely lost. For example, when we would like to find a critical transaction in an economic network, the edge direction is clearly essential.

The theoretical motivation is a relationship between directed DOMINATING SET and DIRECTED (p, q)-EDGE DOMINATING SET. As we mentioned above, DIRECTED $(0, 1)$-EDGE DOMINATING SET is directed DOMINATING SET on directed line graphs and DIRECTED $(1, 1)$-EDGE DOMINATING SET is undirected DOMINATING SET on an underlying undirected graph of a directed line graph. Directed line graphs are used for DNA sequencing and have some useful properties and characterizations [5, 49]. As for combinatorial problems on graphs, (directed) HAMILTONIAN PATH on directed line graphs can be solved in time $O(n^2 + m^2)$ [6] while HAMILTONIAN PATH on undirected line graphs is NP-complete [3]. Therefore, some directed problems could be easier than the undirected versions on line graphs. Unfortunately, however, our results show that directed DOMINATING SET and the distance version, that is, directed r-DOMINATING SET, remain NP-complete even on directed line graphs.

3.1.3 Related problems

One of the most famous covering problems is VERTEX COVER. This is a classical NP-complete problem on undirected graphs, but known to be fixed-parameter tractable [15]. In terms of graph parameters, the size of the minimum vertex cover of G is called the *vertex cover number* of G. For any graph, it is easily seen that the vertex cover number is greater than or equal to the treewidth [32].

EDGE DOMINATING SET is the problem that given an undirected graph $G = (V, E)$ and an integer k, determines whether there exists a set of edges X of size at most k such that any edge in $E \setminus X$ has at least one incident edge in X. This problem is NP-complete even on bipartite, planar, and bounded degree graphs [116], but fixed-parameter tractable in general [31]. As we have seen, the EDGE DOMINATING SET problem is equivalent to DOMINATING SET on line graphs. More-

over, the (optimization) EDGE DOMINATING SET problem is equivalent to MINIMUM MAXIMAL MATCHING [116].

DOMINATING SET is a classical domination problem. This problem is known to be $\Omega(\log n)$-inapproximable, but $O(\log n)$-approximable by a simple greedy algorithm on general graphs [27]. With respect to parameterized complexity, DOMINATING SET is $W[2]$-complete, unlike VERTEX COVER and EDGE DOMINATING SET [28]. Therefore, this problem is well-studied on restricted graphs. Recently, Dawar et al. [22] and Drange et al. [30] considered fixed-parameter tractability and the existence of problem kernels for some sparse graph classes. Their results include the distance version, that is, r-DOMINATING SET. This approach was generalized to directed graphs because the directed DOMINATING SET problem is also $W[2]$-complete [66].

The remainder of this chapter is organized as follows. In Section 3.2, we prove hardness results. In Section 3.3, we give polynomial-time algorithms on trees and fixed-parameter algorithms on general graphs. Finally, we present conclusions and directions for further work in Section 3.4.

3.2 Hardness results

In this section, we discuss the hardness of DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET.

3.2.1 Directed $(0, 1)$-Edge ($(1, 0)$-Edge) Dominating Set

We first show that DIRECTED $(0, 1)$-EDGE ($(1, 0)$-EDGE) DOMINATING SET is NP-complete. Although DIRECTED $(0, 1)$-EDGE DOMINATING SET is very similar to 1-OUT VERTEX COVER, there is a large gap in terms of time complexity.

To show this, we introduce a variant of the SAT problem. Let $(X, \mathcal{C})$ be an instance I of SAT, where $X = \{x_1, x_2, \ldots, x_n\}$ is the set of variables and $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ is the set of clauses. We consider a bipartite graph $G_I = (X \cup \mathcal{C}, E)$, where $E = \{\{x, C\} \mid x \in X, C \in \mathcal{C}$ such that $x \in C$ or $\bar{x} \in C\}$. An instance I of SAT is called *planar* if G_I is planar. Much is known concerning the planar version of SAT, called PLANAR SAT. For example, 3SAT is known to be NP-complete even if the instance is restricted to being planar [76]. The restricted version of 3SAT is called PLANAR 3SAT.

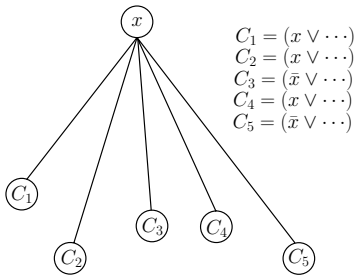

Figure 3.1. Original clauses where x's appear

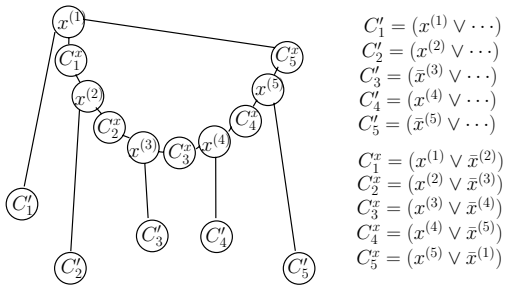

Figure 3.2. Replaced clauses

Here, we consider another restriction of PLANAR SAT. In the restricted instances, each literal appears at most twice, that is, for all $y \in X \cup \overline{X}$, the number of clauses containing y is at most two. Instead, the size of each clause is relaxed to be not exactly three but at most three. We call this version PLANAR AT-MOST3SAT(L2), where L2 means that each literal appears at most twice.

Lemma 3.2.1. PLANAR AT-MOST3SAT(L2) is NP-complete.

Proof: This problem clearly belongs to NP. We show the hardness. The hardness is shown by a reduction from PLANAR 3SAT, which is NP-complete [76]. Let $I = (X, \mathcal{C})$ be an instance of PLANAR 3SAT. If all the literals appear at most twice, we do not need to do anything. Otherwise, there is a literal that appears at least three times. We assume that the literal x is positive, without loss of generality. Let ℓ be the total number of appearances of x and $\bar{x}$.

Next, we create ℓ new variables $x^{(1)}, x^{(2)}, \ldots, x^{(\ell)}$, and new clauses $C^x_1 = \{x^{(1)}, \bar{x}^{(2)}\}$, $C^x_2 = \{x^{(2)}, \bar{x}^{(3)}\}$, ..., $C^x_{\ell-1} = \{x^{(\ell-1)}, \bar{x}^{(\ell)}\}$, $C^x_\ell = \{x^{(\ell)}, \bar{x}^{(1)}\}$. Later in the proof, we will define a mapping of the $x^{(i)}$'s to clauses in the original CNF formula that contains x. The order implicit in the mapping will be used to guarantee planarity of the new formula. These new clauses are introduced in order to guarantee that all $x^{(i)}$'s take the same value (1 or 0). Furthermore, for C the clause in which the i-th x or $\bar{x}$ appears, we define C' to be the same as C except that x or $\bar{x}$ is replaced with $x^{(i)}$ or $\bar{x}^{(i)}$. Note that $x^{(i)}$ (or $\bar{x}^{(i)}$) appears only in C' and C^x_i (resp., C^x_{i-1}), that is, at most twice. It is obvious that the original instance is satisfiable if and only if the new instance is, where x's and $\bar{x}$'s are replaced with $x^{(i)}$ or $\bar{x}^{(i)}$, $i = 1, 2, \ldots, \ell$. By doing this replacement for all the variables, we obtain a new equivalent SAT instance I', whose clauses contain at most three literals and literals appear at most twice.

We now map the $x^{(i)}$'s to clauses containing x and demonstrate that planarity is preserved. We construct a planar drawing of $G_{I'}$ from a planar drawing P of G_I. We focus on the part of the graph

consisting of x and the C's containing x in P, as shown in Figure 3.1. Suppose that x appears in C_{x1}, C_{x2}, ..., $C_{x\ell}$, where C's are drawn in this order in P. Our mapping assigns $x^{(i)}$ (or $\bar{x}^{(i)}$) to C_{xi}. Now we are ready to show that $G_{I'}$ has a planar drawing. We first draw vertices corresponding to C''s in the order in which the C's are drawn in P. We then draw $x^{(i)}$'s and $C_1^x, C_2^x, \ldots, C_\ell^x$ as the cycle $(x^{(1)}, C_1^x, x^{(2)}, \ldots, x^{(\ell)}, C_\ell^x, x^{(1)})$ taking the place of x in P, with scale reduced as needed for the cycle to fit in the allotted space. Since a cycle is planar, this can be done in a planar way. Finally, we draw edges between $x^{(i)}$'s and C_{xi}'s, which does not yield any crossing among vertices derived by variable x and clauses containing x due to the way we chose to map values of i to clauses (see Figure 3.2).The planarity of P guarantees that the whole $G_{I'}$ can be drawn in the plane. This completes the proof. ■

By using Lemma 3.2.1, we can obtain Theorem 3.2.2.

Theorem 3.2.2. DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE) DOMINATING SET is NP-complete on directed planar graphs such that $\Delta \leq 3$ holds.

Proof: We only consider DIRECTED $(0,1)$-EDGE DOMINATING SET as the other proof is similar. This problem is clearly in NP. Thus, we show the hardness. The reduction is from PLANAR AT-MOST3SAT(L2).

Let n be the number of variables, m be the number of clauses, and l be the number of literals in an input Φ for PLANAR AT-MOST3SAT(L2). Then, we construct a graph as in Figure 3.3. First, we create n cycles of length four corresponding to the variables in Φ and m paths of length five corresponding to the clauses in Φ. For a variable's gadget, if we include the two horizontal edges in the $(0,1)$-edge dominating set, it corresponds to setting the variable to true in Φ. Otherwise, we include the two vertical edges, which corresponds to setting the variable to false. Note that the size of a minimum $(0,1)$-edge dominating set for a cycle of length four is two. In Figure 3.3, thick lines represent that they are included in the solution (we use the same convention in the other figures).

We connect each clause gadget to the variable gadgets corresponding to the literals in the clause, as follows. For $v_1, v_2, \ldots, v_6$ the vertices in the clause gadget, each of v_1, v_3, and v_5 is connected by a path of length two, called a *linking path*, to one of the vertices in a variable gadget. We can observe that there are l linking paths in the constructed graph. For each variable, there are at most two occurrences of true literals and at most two of false literals. Because the variable gadget has

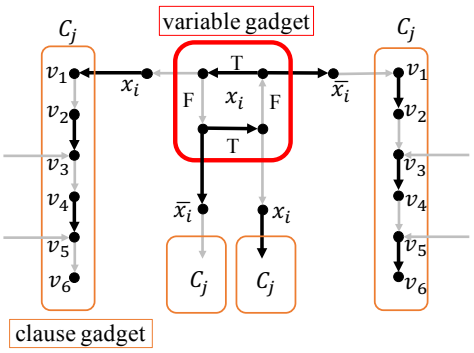

Figure 3.3. Constructed graph of the reduction from 3SAT to $(0, 1)$-EDS

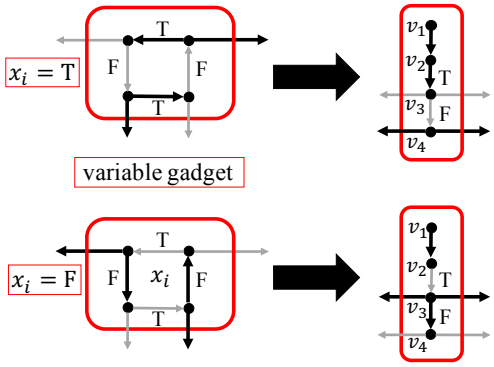

Figure 3.4. Replacing a cycle by a directed path for a variable's gadget

four vertices corresponding to literals, by connecting each vertex in the variable gadget to a clause gadget, for any vertex v in the constructed graph, $\Delta \leq 3$ holds.

Finally, we show that an input Φ for PLANAR AT-MOST3SAT(L2) has a satisfying truth assignment if and only if there exists a $(0, 1)$-edge dominating set of size $2n + l + 2m$ in the constructed graph. Given a truth assignment for Φ, we take the corresponding vertical or horizontal pair of edges in each variable gadget since we need exactly two edges in order to dominate the cycle. For each linking path, we add to the $(0, 1)$-edge dominating set either the edge incident on a variable gadget if that edge is not dominated, or otherwise the other edge on the linking path. For a clause gadget, if a literal in the clause is true, an edge adjacent to the vertex corresponding to the literal is dominated. Note that if at least one literal is true, the clause gadget can be dominated by only two edges; otherwise we need three edges to dominate it. Because every clause is true and there are n variable gadgets, l linking paths, and m clauses, we can dominate all edges by a set of $2n + l + 2m$ edges.

Conversely, suppose that we are given a $(0, 1)$-edge dominating set of size $2n + l + 2m$. We need at least two edges to dominate each variable gadget and at least one edge to dominate each linking path. Because there are n variable gadgets and l linking paths, all clause gadgets must be dominated by at most $2m$ edges. Note that we need at least two edges to dominate a clause gadget. Two edges will suffice only when the $(0, 1)$-edge dominating set contains at least one adjacent edge in a linking path. This means that each clause has at least one true literal. Moreover, we can observe that a $(0, 1)$-edge dominating set contains exactly two, one, and two edges in each variable gadget, linking path, and clause gadget, respectively. In a variable gadget, either the vertical or horizontal

pair of edges is in the $(0,1)$-edge dominating set. Thus, we give an assignment for each variable such that if the horizontal pair is included, we assign the variable to be true, and otherwise we assign it to be false. Because each clause has at least one true literal, this is a satisfying truth assignment. ■

By replacing each variable gadget by a path v_1, v_2, v_3, v_4 of length three, connecting vertex v_3 and true literals in a clause, and connecting vertex v_4 and false literals (see Figure 3.4), we can also show that DIRECTED $(0,1)$-EDGE DOMINATING SET is NP-complete on directed acyclic planar graphs of bounded degree. Note that edge (v_1, v_2) is contained in any $(0,1)$-edge dominating set. Moreover, including edge (v_2, v_3) in the $(0,1)$-edge dominating set corresponds to assigning true to the variable and including edge (v_3, v_4) corresponds to assigning false. Now, we only replace variable gadgets in Theorem 3.2.2, and the other parts in the constructed graph satisfy $\Delta \leq 3$. Since each literal appears at most twice in PLANAR AT-MOST3SAT(L2), v_3 and v_4 have at most two edges in linking paths. Thus, the degrees of v_3 and v_4 are at most four and three, respectively. Consequently, $\Delta \leq 4$ holds for any vertex in the constructed graph.

Corollary 3.2.3. DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE) DOMINATING SET is NP-complete on directed acyclic planar graphs such that $\Delta \leq 4$ holds.

3.2.2 Directed $(1,1)$-Edge Dominating Set

As for DIRECTED $(1,1)$-EDGE DOMINATING SET, we obtain a stronger result in terms of a degree constraint. To show this, we first introduce a variant of planar graphs. A graph is *planar almost cubic* if it is planar, there are exactly two vertices of degree two, and the degree of all other vertices is three. We show that VERTEX COVER remains NP-complete on planar almost cubic graphs.

Lemma 3.2.4. VERTEX COVER on planar almost cubic graphs is NP-complete.

Proof: We show a reduction from VERTEX COVER on planar cubic graphs, which is NP-complete [79]. Given a planar cubic graph $G = (V, E)$, choose an edge (u, v) and insert two vertices w and x into (u, v), that is, change (u, v) to a path consisting of u, w, x, v. Let $G' = (V', E')$ be the constructed graph where $V' = V \cup \{w, x\}$ and $E' = (E \setminus \{(u, v)\}) \cup \{(u, w), (w, x), (x, v)\}$. Note that the degree of w and x is two, and the degree of any other vertex is three. Since we only

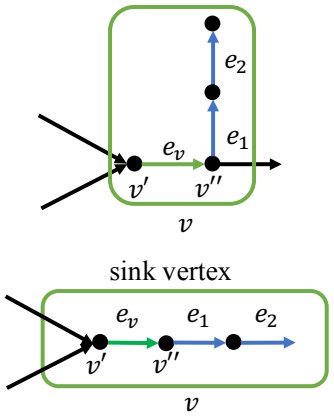

Figure 3.5. Vertex gadgets for v with $d^{in}(v) = 2$ in the reduction to $(1, 1)$-EDS

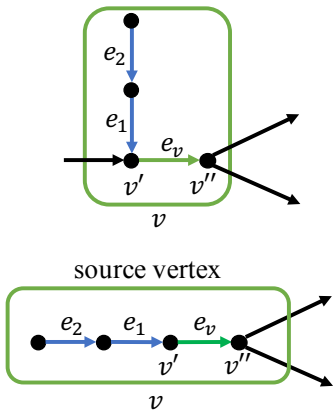

Figure 3.6. Vertex gadgets for v with $d^{out}(v) = 2$ in the reduction to $(1, 1)$-EDS

replace an edge by a path of length three in a planar graph, G' remains planar. Thus, G' is a planar almost cubic graph.

Then, we show that an instance (G, k) of VERTEX COVER on planar cubic graphs is a yes-instance if and only if an instance $(G', k + 1)$ of VERTEX COVER on planar almost cubic graphs is a yes-instance.

Let C be a vertex cover of size at most k in G. Then, C covers every edge in $E \setminus \{(u, v)\}$ and at least one of (u, w) and (x, v) in G' since at least one of u and v is in C. If both of (u, w) and (x, v) are covered by C, we add either w or x. Because (w, x) is the only edge not covered, we can obtain a vertex cover of size at most $k + 1$. Otherwise, without loss of generality, we suppose that (u, w) is not covered. Now, the only edges not covered by C are (u, w) and (w, x). Therefore, by setting $C' = C \cup \{w\}$, we obtain a vertex cover in G' of size at most $k + 1$.

Conversely, let C' be a vertex cover of size at most $k + 1$ in G'. To cover edge (w, x), C' contains at least one of w and x. First, we suppose C' contains exactly one of w and x. Without loss of generality, we suppose that $w \in C'$ and $x \notin C'$. Then we can observe that $v \in C'$ in order to cover (x, v). Thus $C' \setminus \{w\}$ is a vertex cover of size at most k in G because v covers (u, v) in G. If C' contains both x and w, we can replace x by v because (w, x) is covered by w. Then, $(C' \setminus \{w, x\}) \cup \{v\}$ is a vertex cover of size at most k in G because v covers (u, v) in G. ■

By using Lemma 3.2.4, we show the following theorem.

Theorem 3.2.5. DIRECTED $(1, 1)$-EDGE DOMINATING SET is NP-complete on directed acyclic

planar graphs such that $\Delta \leq 3$ holds.

Proof: Since DIRECTED $(1,1)$-EDGE DOMINATING SET is clearly in NP, we prove hardness. We show a reduction from VERTEX COVER on planar almost cubic graphs. Suppose that we are given an instance (G,k) of VERTEX COVER. For an undirected planar almost cubic graph G, we choose the two vertices with degree two in G as source and sink vertices. We then arrange each vertex in a horizontal line such that the two vertices of degree two become ends of the line and orient every edge from left to right. Note that there exist exactly one source vertex such that the in-degree is zero and out-degree is two and exactly one sink vertex such that the in-degree is two and out-degree is zero. For other vertices v, it holds that $d^{in}(v) = 1$ and $d^{out}(v) = 2$ or $d^{in}(v) = 2$ and $d^{out}(v) = 1$.

Next, we attach paths of length three to each vertex to form a vertex gadget. First, for $G = (V,E)$, let $G' = (V' \cup V'', E_o \cup E_v)$, where $V' = \{v' \mid v \in V\}$, $V'' = \{v'' \mid v \in V\}$, $E_o = \{(u'', v') \mid (u,v) \in E)$, and $E_v = \{(v', v'') \mid v \in V\}$. Graph G' is a graph formed from G such that v is split into two vertices v', v'' connected by the edge (v', v''). Then, we attach a path from v'' of length two to v'' for $v'' \in \{w'' \in V'' \mid d^{in}(w) = 2, w \in V\}$ (see Figure 3.5), and a path to v' of length two to v' for $v' \in \{w' \in V' \mid d^{out}(w) = 2, w \in V\}$ (see Figure 3.6). For each path, we denote the edge incident to v' or v'' in a path by e_1 and the other edge by e_2. An edge $e_v \in E_v$ in G' corresponds to vertex v in G and each oriented edge in E_o corresponding to an edge in G is called an *original edge*. We refer to a path consisting of e_1, e_2, and e_v as a *vertex gadget* of v. Let G'' be the graph constructed in this way. Since we only replace vertices in G by paths, G'' remains planar and acyclic and for any vertex v in G'', $\Delta \leq 3$ holds.

Finally, we show that an instance (G,k) of VERTEX COVER on planar almost cubic graphs is a yes-instance if and only if an instance $(G'', n+k)$ of DIRECTED $(1,1)$-EDGE DOMINATING SET is a yes-instance. Let $G = (V,E)$ and C be a vertex cover of size at most k in G. In G'', we add each e_1 to the solution set in order to dominate e_2 and e_v for each vertex gadget. We also add e_v's such that $v \in C$ in G. Because edge e_v corresponds to v in G, each edge e_v dominates all original edges. Therefore, such a solution set is a $(1,1)$-edge dominating set and its size is $n+k$.

Conversely, let K be a $(1,1)$-edge dominating set of size at most $n+k$ in G''. To dominate e_2, K contains either e_1 or e_2 for each vertex gadget. If K contains e_2 for some gadgets, e_2 can be replaced by e_1 while maintaining a $(1,1)$-edge dominating set. Thus, we can assume that K contains all e_1's and does not contain any e_2's. Note that every e_v is dominated by e_1. From this

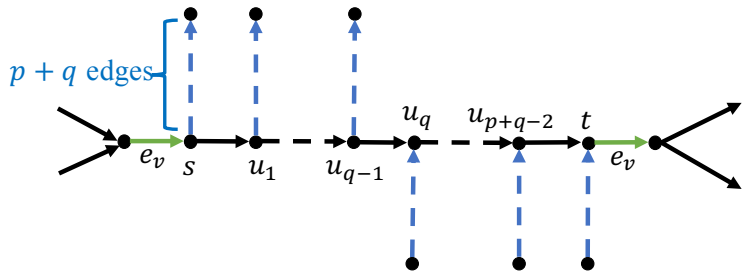

Figure 3.7. Replacing an edge and attaching path gadgets for the reduction to (p, q)-EDS

fact, if K includes some original edges, we can exclude them because they can only dominate e_v's. Therefore, we can suppose K only contains e_1's and e_v's. Because the set of e_v's in K dominates all original edges, the set of vertices in G corresponding to e_v's in K covers all edges in G. Thus, this is a vertex cover for G. Since the size of K is at most $n + k$ and the number of e_1's included in K is n, the number of e_v's in K is at most k. Therefore, the size of the vertex cover is at most k. ■

We also obtain the following result on the distance-generalized version.

Corollary 3.2.6. DIRECTED (p, q)-EDGE DOMINATING SET is NP-complete on directed acyclic planar graphs such that $\Delta \leq 3$ holds when $p, q \geq 1$.

Proof: For graph G'' in Theorem 3.2.5, let E_v, E_1 and E_2 be the sets of e_v's, e_1's, and e_2's, respectively. First, we remove every edge in $E_1 \cup E_2$ from G''. Then, we replace each original edge (s, t) by a path of length $p + q - 1$, consisting of $s, u_1, u_2, \ldots, u_{p+q-2}, t$. We consider the edge (u_{q-1}, u_q) to correspond to an edge in E.

Next, we attach a path of length $p + q$, called a *path gadget*, to each u_i as in Figure 3.7. As with DIRECTED $(1, 1)$-EDGE DOMINATING SET, for each edge $(w, x) \in E_v$ except for the source and the sink, if $d^{in}(w) = 1$, we attach a path gadget to w. Otherwise, that is, if $d^{out}(x) = 1$, we attach it to x. Finally, we attach a path gadget to each of the source and the sink.

Let G''' be the created graph. Because we only attach the paths to vertices satisfying $\Delta \leq 2$, it holds that $\Delta \leq 3$ for any v. Furthermore, G''' remains planar and acyclic.

Then, we show that an instance (G, k) of VERTEX COVER is a yes-instance if and only if an instance $(G''', (p + q - 1)n + k)$ of DIRECTED (p, q)-EDGE DOMINATING SET is a yes-instance. The rest of the argument is the same as that in the proof for DIRECTED $(1, 1)$-EDGE DOMINATING SET. ■

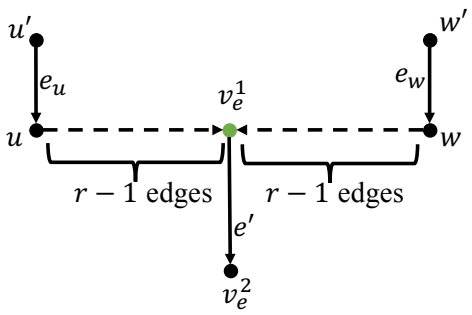

Figure 3.8. Replacing $e = (u, v)$ by a gadget for r-VC

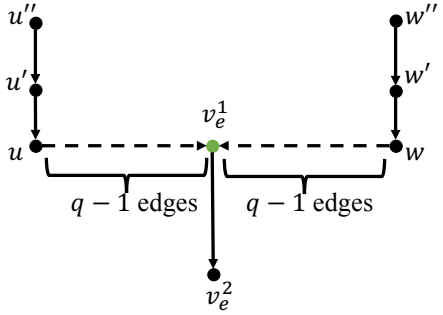

Figure 3.9. Replacing $e = (u, v)$ by a gadget for $(0, q)$-EDS

3.2.3 Distance generalization

In this subsection, we consider the distance-generalized versions as with Corollary 3.2.6. We first show that DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED $(0, q)$-EDGE ($(p, 0)$-EDGE) DOMINATING SET are NP-complete on directed acyclic planar graphs of bounded degree.

Theorem 3.2.7. When r, p and q are greater than 1, DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED $(0, q)$-EDGE ($(p, 0)$-EDGE) DOMINATING SET are NP-complete on directed acyclic planar graphs such that $\Delta \leq 4$.

Proof: We show a reduction from VERTEX COVER on planar cubic graphs, which is NP-complete [79], to DIRECTED r-OUT VERTEX COVER.

Given an instance $(G = (V, E), k)$ of VERTEX COVER, we create a graph G'. First, we replace each edge $e = (u, w)$ in E by an edge gadget as in Figure 3.8. We insert a center vertex v_e^1 in each edge e, that is, we replace one edge $e = (u, w)$ by two directed edges (u, v_e^1) and (w, v_e^1). We denote the set of v_e^1's by V_e^1. Moreover, for each e in E we add a vertex v_e^2 and an edge $e' = (v_e^1, v_e^2)$. Let V_e^2 be the set of v_e^2's and E' be the set of e''s.

Next, we replace (u, v_e^1) by a directed path from u to v_e^1 of length $r-1$ and (w, v_e^1) by a directed path from w to v_e^1 of length $r - 1$. Let V_P be the set of vertices in the directed paths except for u, v_e^1, and w and E_P be the set of edges in the directed paths.

As a vertex gadget, we attach an edge $e_v = (v', v)$ to every v. We denote the set of v''s by V' and the set of e_v's by E_v.

Let $G' = (V \cup V' \cup V_e^1 \cup V_e^2 \cup V_P, E' \cup E_v \cup E_P)$ be the created graph. Note that G' remains planar and acyclic. Moreover, for any vertex v in G', $\Delta \leq 4$ holds because we only replace each

edge by a path and attach edges.

Now, we show that an instance (G, k) of VERTEX COVER is a yes-instance if and only if an instance $(G', n + k)$ of DIRECTED r-OUT VERTEX COVER is a yes-instance. Suppose that we are given a vertex cover $C = \{v_1, \ldots, v_k\} \subseteq V$ of size k. Let $C' = V' \cup C$. Then the set V' covers all edges in $E_v \cup E_P$. Furthermore, since C is a vertex cover in G, it covers every edge in E' due to the construction. Thus, C' is an r-out-vertex cover of size at most $n + k$.

Conversely, suppose that we are given an r-out-vertex cover C' of size $n + k$. Note that C' always includes V' in order to cover each edge $e_v = (v', v)$.

Because V' covers every edge in $E_v \cup E_P$, and all edges covered by $V_e^1 \cup V_e^2 \cup V_P$ are covered by V, if C' includes a vertex in $V_e^1 \cup V_e^2 \cup V_P$, we can replace it by a vertex in V while maintaining the coverage of each edge by C'. Let $C = C' \setminus V' \subseteq V$. Then C is a vertex cover in G of size k because C covers all edges in E' corresponding to E and $|V'| = n$.

By attaching a new edge (v'', v') for each v' in the reduced graph G' and a corresponding additional vertex v'', and setting $r = q$ for DIRECTED $(0, q)$-EDGE DOMINATING SET, we obtain the reduced graph of DIRECTED $(0, q)$-EDGE DOMINATING SET (see Figure 3.9). Then, we can similarly prove DIRECTED $(0, q)$-EDGE DOMINATING SET is NP-complete on directed acyclic planar graphs such that $\Delta \leq 4$.

By reversing the orientation of every edge in the reduced graph of DIRECTED r-OUT VERTEX COVER and DIRECTED $(0, q)$-EDGE DOMINATING SET, we can also show that DIRECTED r-IN VERTEX COVER and DIRECTED $(p.0)$-EDGE DOMINATING SET are NP-complete on directed acyclic planar graphs such that $\Delta \leq 4$, respectively. ■

From Theorems 3.2.2 and 3.2.7, we can conclude directed r-DOMINATING SET on directed line graphs is NP-complete.

Corollary 3.2.8. The (directed) r-DOMINATING SET problem is NP-complete on directed line graphs even if $r = 1$.

Finally, we show in Theorems 3.2.10, 3.2.12 and Corollaries 3.2.11, 3.2.13 that DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET are $W[2]$-hard parameterized by the solution size k and $c \ln k$-inapproximable for any constant $c < 1$ on directed acyclic graphs unless P=NP. Here, we introduce the SET COVER problem, which is used for reductions to

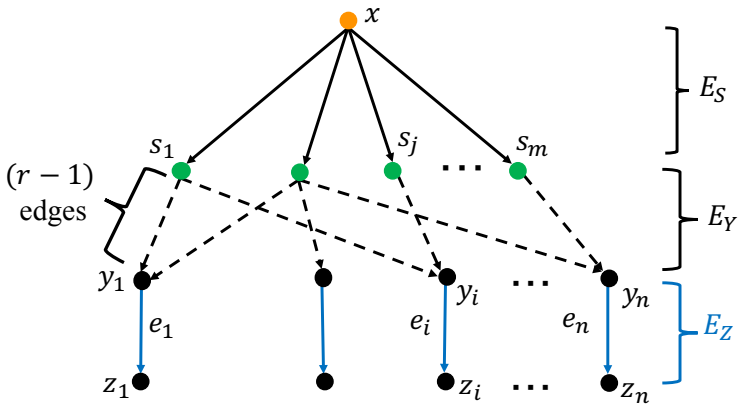

Figure 3.10. Constructed graph of the reduction from SET COVER to DIRECTED r-OUT VERTEX COVER

show the hardness results above.

Definition 3.2.9. SET COVER is the problem that given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ defined over a domain of elements $U = \{e_1, e_2, \ldots, e_n\}$ and an integer k, determines whether there exists a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ of size at most k such that $\bigcup_{S \in \mathcal{S}'} S = U$.

It should be noted that SET COVER is $W[2]$-complete and $\Omega(\log n)$-inapproximable [27, 28]. Since the setting of SET COVER is very general, many problems can be formulated as SET COVER. Also it should be noted that SET COVER can be approximated within ratio $O(\log n)$ by a simple greedy algorithm, which yields approximation algorithms with performance guarantee for such problems. In fact, our problems DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET can be approximated within $O(\log n)$ under the framework, where n is the number of vertices.

Theorem 3.2.10. DIRECTED r-IN (OUT) VERTEX COVER is $W[2]$-hard parameterized by the solution size k on directed acyclic graphs when $r \geq 2$.

Proof: We show that there is a parameterized reduction from SET COVER to DIRECTED r-IN (OUT) VERTEX COVER. In this proof, we focus on DIRECTED r-OUT VERTEX COVER since we can also prove the other case by a slight modification. If $k = m$, it is trivial. Hence, we suppose that $k < m$. Without loss of generality, suppose that every $S \in \mathcal{S}$ is not empty.

Given an instance of SET COVER, that is, $(\mathcal{S}, U, k)$, we create a graph $G = (V, E)$ where $V = \{x\} \cup V_S \cup V_Y \cup V_Z$ and $E = E_S \cup E_Y \cup E_Z$ by the following method (see Figure 3.10). Let

x be a *super vertex* and $V_S = \{s_1, s_2, \ldots, s_m\}$ be a vertex set corresponding to $\mathcal{S}$ in the instance of SET COVER. Then we connect x and each $s_j \in V_S$ by adding edges (x, s_j), where $1 \leq j \leq m$. We denote the set of (x, s_j)'s by E_S.

Let $V_Y = \{y_1, \ldots, y_n\}$ and $V_Z = \{z_1, \ldots, z_n\}$. Then, for each element e_i of SET COVER, we create the corresponding edge (y_i, z_i) for $1 \leq i \leq n$. Finally, we connect each s_j and y_i by a path of length $r-1$ from s_j to y_i if $S_j \in \mathcal{S}$ covers element $e_i \in U$. We denote the set of (y_i, z_i)'s by E_Z and the set of edges in the paths of length $r-1$ by E_Y. Obviously, G is directed acyclic due to the construction. Constructing the graph can be done in time polynomial in the size of the input to SET COVER.

Then we show that an instance of SET COVER $(\mathcal{S}, U, k)$ is a yes-instance if and only if the instance of DIRECTED r-OUT VERTEX COVER $(G, r, k+1)$ is a yes-instance, where G is the graph created by the above procedure.

If $(\mathcal{S}, U, k)$ is a yes-instance of SET COVER, let $\mathcal{S}^* = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$ be a solution where $\{j_1, j_2, \ldots, j_k\} \subseteq \{1, 2, \ldots, m\}$. In G, we select $C = \{s_{j_1}, s_{j_2}, \ldots, s_{j_k}, x\}$. Then we can immediately confirm that C covers every edge in E since x covers all edges in $E \setminus E_Z$.

Conversely, if $(G, r, k+1)$ is a yes-instance of DIRECTED r-OUT VERTEX COVER, let C be a solution of size $k+1$. Note that C contains x because every edge in E_S is covered only by vertex x. If C includes a vertex u_j on the path from s_j to z_i, then because every edge covered by u_j is covered by s_j, we can replace u_j by vertex s_j if $s_j \in V_S \setminus C$, and any vertex $s \in V_S \setminus C$ otherwise, keeping every edge covered by C.

Therefore, we assume that $C \setminus \{x\} \subseteq V_S$ and let $C = \{s_{j_1}, s_{j_2}, \ldots, s_{j_k}, x\}$. Then we choose $\mathcal{S}^* = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\} \subseteq \mathcal{S}$. Note that element e_i is covered by S_j if and only if edge $e_i = (y_i, z_i)$ is covered by s_j in G since vertex x does not cover any edge in U. From the construction of G, we conclude that $\bigcup_{S \in \mathcal{S}^*} S = U$, and hence $(\mathcal{S}, U, k)$ is a yes-instance. This implies that DIRECTED r-OUT VERTEX COVER is $W[2]$-hard parameterized by k. By reversing the orientation of every edge, we can also obtain a reduction from SET COVER to DIRECTED r-IN VERTEX COVER. ■

Corollary 3.2.11. When $r \geq 2$, for DIRECTED r-IN (OUT) VERTEX COVER, there is no polynomial-time $c \ln k$-approximation algorithm for any constant $c < 1$ on directed acyclic graphs unless P=NP, where k is the size of an optimal solution, though it can be approximated within ratio

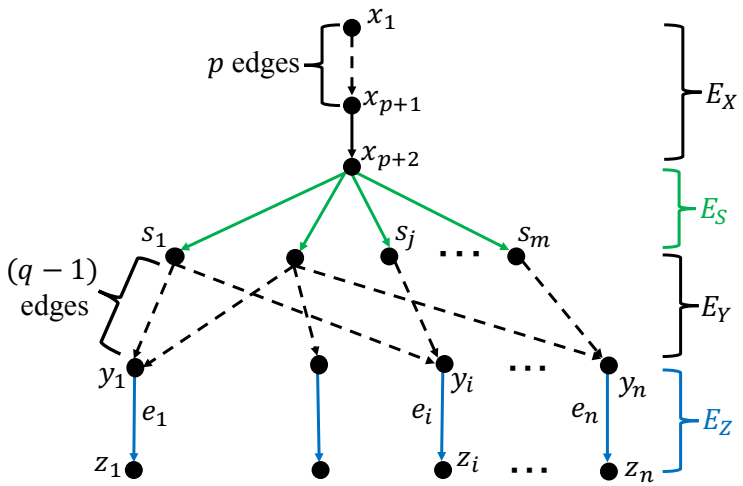

Figure 3.11. Constructed graph of the reduction from SET COVER to DIRECTED (p, q)-EDGE DOMINATING SET

$O(\log n)$ by a greedy algorithm.

Proof: We use the same reduction as in Theorem 3.2.10. For SET COVER, there is no polynomial-time algorithm whose approximation ratio is better than $c \ln |U|$ with any $c < 1$ unless P=NP [27]. Since the parameter k of SET COVER corresponds to $(k + 1)$ of an r-in (out) vertex cover, it is hard to find an r-in (out) vertex cover with size at most $(k + 1)c \ln |U|$. Here, $|U|$ is bounded below by k, and this implies that r-IN (OUT) VERTEX COVER is hard to approximate within $c \ln k$ for any $c < 1$.

On the other hand, since we can describe r-IN (OUT) VERTEX COVER as SET COVER, it can be approximated within ratio $O(\log n)$ by a greedy algorithm. ■

By almost the same method as for r-IN (OUT) VERTEX COVER, we obtain a proof as for DIRECTED (p, q)-EDGE DOMINATING SET. That is, we replace a super vertex by a path of length $p + 1$.

Theorem 3.2.12. DIRECTED (p, q)-EDGE DOMINATING SET is $W[2]$-hard parameterized by the solution size k on directed acyclic graphs when $p \geq 2$ or $q \geq 2$.

Proof: We show that there is a parameterized reduction from SET COVER to DIRECTED (p, q)-EDGE DOMINATING SET. In this proof, we show only the case in which $q \geq 2$ since we can also prove the other case by a slight modification.

Given an instance of SET COVER, that is, $(\mathcal{S}, U, k)$, we create graph G by the following method (see Figure 3.11). This graph is similar to a constructed graph for DIRECTED r-IN (OUT) VERTEX COVER. We replace r by $q-1$ and x by a path $x_1, x_2, \ldots, x_{p+2}$ of length $p+1$. Let E_X be an edge set of the path. Note that each edge (x_{p+2}, s_j) corresponds to $S_j \in \mathcal{S}$. The rest of the proof is almost the same as that for DIRECTED r-IN (OUT) VERTEX COVER.

We show that an instance of SET COVER $(\mathcal{S}, U, k)$ is a yes-instance if and only if the instance of DIRECTED (p,q)-EDGE DOMINATING SET $(G, k+1)$ is a yes-instance, where G is the graph created by the above procedure.

Given a yes-instance of SET COVER $(\mathcal{S}, U, k)$, let $\mathcal{S}^* = \{S_{j_1}, \ldots, S_{j_k}\}$ be a solution where $\{j_1, \ldots, j_\ell\} \subseteq \{1, \ldots, m\}$. In graph G, we select the edge set $K = \{(x_{p+2}, s_{j_1}), \ldots, (x_{p+2}, s_{j_k}), (x_{p+1}, x_{p+2})\}$. Then we can immediately confirm that edge (x_{p+1}, x_{p+2}) dominates every edge in $E \setminus E_Z$ and $K \setminus \{(x_{p+1}, x_{p+2})\}$ dominates every edge in E_Z.

Conversely, if $(G, k+1)$ is a yes-instance of DIRECTED (p,q)-EDGE DOMINATING SET, let K be a solution of size $k+1$. Note that K contains at least one edge in E_X and we can assume that K only contains one edge (x_{p+1}, x_{p+2}) because it dominates every edge dominated by $e \in E_X \setminus \{(x_{p+1}, x_{p+2})\}$.

If there is edge $e \in (E \setminus E_X \setminus E_S) \cap K$ on the path from s_j to z_i, we can replace e by $(x_{p+2}, s_j) \in E_S$ if it is not in K, and otherwise by any edge $e_s \in E_S$, keeping every edge of G dominated by K. Hence every edge in $E \setminus E_Z$ is already dominated by (x_{p+1}, x_{p+2}). Therefore, we assume that $K \subseteq E_S$.

Let $K = \{(x_{p+2}, s_{j_1}), (x_{p+2}, s_{j_2}), \ldots, (x_{p+2}, s_{j_k}), (x_{p+1}, x_{p+2})\}$ be a solution of size $k+1$. We then choose $\mathcal{S}^* = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\} \subseteq \mathcal{S}$. Note that element e_i is covered by S_j if and only if edge (y_i, z_i) is dominated by (x_{p+2}, s_j) in G. From the construction of G, we conclude that $\bigcup_{S \in \mathcal{S}^*} S = U$, and hence $(\mathcal{S}, U, k)$ is a yes-instance. This implies that DIRECTED (p,q)-EDGE DOMINATING SET is $W[2]$-hard parameterized by k when $q \geq 2$.

By reversing the orientation of every edge, we can obtain a reduction from SET COVER to DIRECTED (p,q)-EDGE DOMINATING SET when $p \geq 2$. ■

Corollary 3.2.13. When $p \geq 2$ or $q \geq 2$, for DIRECTED (p,q)-EDGE DOMINATING SET, there is no polynomial-time $c \ln k$-approximation algorithm for any constant $c < 1$ on directed acyclic graphs unless P=NP, where k is the size of an optimal solution, though it can be approximated

within ratio $O(\log n)$ by a greedy algorithm for any p and q.

Proof: We use the same reduction as in Theorem 3.2.12. For SET COVER, there is no polynomial-time algorithm whose approximation ratio is better than $c \ln |U|$ with any $c < 1$ unless P=NP [27]. Since the parameter k of SET COVER corresponds to $(k+1)$ of a (p,q)-edge dominating set, it is hard to find an (p,q)-edge dominating set with size at most $(k+1)c \ln |U|$. Here, $|U|$ is bounded below by k, and this implies that (p,q)-EDGE DOMINATING SET is hard to approximate within $c \ln k$ for any $c < 1$.

On the other hand, since we can describe DIRECTED (p,q)-EDGE DOMINATING SET as SET COVER, it can be approximated within ratio $O(\log n)$ by a greedy algorithm. ■

3.3 Algorithms

In this section, we give polynomial-time algorithms for DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p,q)-EDGE DOMINATING SET on trees. Moreover, we also present a fixed-parameter algorithm with respect to treewidth and the solution size respectively for DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE, $(1,1)$-EDGE) DOMINATING SET on general graphs.

3.3.1 Algorithms on trees

In this subsection, we first present a polynomial-time algorithm for DIRECTED (p,q)-EDGE DOMINATING SET of trees, which shows the following theorem.

Theorem 3.3.1. Let $\gamma = \max\{p,q\}$. There is an algorithm that solves DIRECTED (p,q)-EDGE DOMINATING SET on trees in time $O(\gamma^2 \Delta^2 n)$.

Proof: We show this theorem by presenting a concrete algorithm to solve DIRECTED (p,q)-EDGE DOMINATING SET on trees in time $O(\gamma^2 \Delta^2 n)$. Our algorithm is based on dynamic programming. Because the underlying undirected graph of an instance graph G is a tree, we can root it at an arbitrary vertex; henceforth we use $\hat{G}$ to denote such a rooted tree. When we use the terms *parent*, *child*, *ancestor*, and *descendant*, we are referring to the relationships between vertices in $\hat{G}$.

We first extend the definition of distance to specify distances between vertices and edges. For an edge $e = (u,v)$ and vertices w and x, we define $dist(w,e)$ to be $dist(w,u)$ and $dist(e,x)$ to be $dist(v,x)$. Moreover, for two edges $e = (u,v)$ and $f = (x,y)$, we define $dist(e,f)$ to be

$dist(v, x)$. An edge e *i-in-dominates* (or just *in-dominates*) all edges f such that $dist(f, e) \leq i - 1$ and an edge e *j-out-dominates* (or just *out-dominates*) all edges f such that $dist(e, f) \leq j - 1$. In a directed path containing edges e and f, the edges (not including e and f) traversed along the path are *between* e and f. If there are k edges between e and f, then e $(k + 1)$-out-dominates f and f $(k + 1)$-in-dominates e.

In $\hat{G}$, we use T_v to denote the subtree rooted at the vertex v, and $G[T_v]$ to denote the subgraph of (the directed graph) G induced on the vertices in T_v. We call $G[T_v]$ the *subtree of G rooted at v* and use $conn(v)$ to denote the edge connecting v to its parent, if it has one. We refer to a vertex v as an *out-vertex* if $conn(v)$ is directed from v to its parent and an *in-vertex* if $conn(v)$ is directed from v's parent to v. If v is the root of $\hat{G}$, it is neither an out-vertex nor an in-vertex. We use $same(v)$ and $diff(v)$ to denote the sets of children of v that are out-vertices and in-vertices, respectively, if v is an out-vertex and that are in-vertices and out-vertices, respectively, if v is an in-vertex. Furthermore, we use $ST(v)$ to denote the set of subtrees rooted at vertices in $same(v)$ and $DT(v)$ to denote the set of subtrees rooted at vertices in $diff(v)$; these are considered to be two different *types of subtrees*. In addition, we use C_s to denote the set of edges between v and vertices in $same(v)$, and C_d to denote the set of edges between v and vertices in $diff(v)$; just as there are two types of subtrees, we consider these sets to constitute two types of *connecting edges*.

Our dynamic-programming algorithm processes vertices in an order such that a vertex v is processed after all its descendants, where we use information about the subtrees rooted at the children of v to determine how to dominate edges in $G[T_v]$. We store not only the sizes of edge dominating sets, but also the sizes of edge dominating sets defined in terms of their *reach* and *deficit*, which are measures of the impact of edges inside a subtree in the domination of edges outside the subtree and the impact of edges outside a subtree in the domination of edges inside the subtree.

To see how edges in subtrees rooted at children of v can have an impact on each other, suppose v has two children w and x such that w is an out-vertex and x is an in-vertex. Furthermore, consider an edge e_w in $G[T_w]$ such that $dist(e_w, w) = i$ and an edge e_x in $G[T_x]$ such that $dist(x, e_x) = j$. We can form a directed path that starts at e_w and traverses the edges (w, v) and (v, x) to end at e_x. Since the number of edges between e_w and e_x is $i + j + 2$, this means that e_w $(i + j + 3)$-out-dominates e_x and that e_x $(i + j + 3)$-in-dominates e.

To determine the reach of a set of edges K in $G[T_v]$, we first determine the shortest distance

i from an edge in K to v, if v is an out-vertex, or the shortest distance i from v to an edge in K, if v is an in-vertex. When v is an endpoint of an edge in K (that is, $i = 0$), that edge will be able to q-out-dominate an edge outside of $G[T_v]$, if v is an out-vertex, or p-in-dominate an edge outside of $G[T_v]$, if v is an in-vertex. We thus define $maxreach(v) = q$ for each out-vertex v and $maxreach(v) = p$ for each in-vertex v. More generally, we define the *reach of K beyond $G[T_v]$* to be $maxreach(v) - i$.

To measure which edges depend on outside edges for domination, we define the *deficit of K within $G[T_v]$* to be maximum over $dist(e, v)$ (resp., $dist(v, e)$) over all edges e in $G[T_v]$ not (p, q)-dominated by any edge in K, for v an out-vertex (resp., in-vertex). Since the edge between v and its parent is the outside edge that can cover the largest deficit, we set $maxdeficit(v) = p$ for v an out-vertex and $maxdeficit(v) = q$ for v an in-vertex. We refer to all edges e with $dist(e, v) \leq d - 1$ (resp., $dist(v, e) \leq d - 1$) to be *edges of deficit of d* (≥ 1) in $G[T_v]$, for v an out-vertex (resp., an in-vertex). If $d = 0$, the deficit does not exist, that is, all edges in $G[T_v]$ are dominated by K. Should an edge outside a subtree have sufficient reach to dominate all edges of deficit of at most d, we will say that the edge *covers the deficit*.

Using these concepts, we say that a set of edges K is a *reach-r-deficit-d edge dominating set* for $G[T_v]$ if the reach of K beyond $G[T_v]$ is r, and K (p, q)-dominates $G[T_v \backslash J]$ where J is the set of edges of deficit at most d in $G[T_v]$. In our algorithm, we use $D[v, r, d]$ to store the minimum number of edges in a reach-r-deficit-d edge dominating set for $G[T_v]$.

When processing a vertex v, we determine $D[v, r, d]$ for values of r and d in the ranges $0 \leq r \leq maxreach(v)$ and $0 \leq d \leq maxdeficit(v)$. For the base cases, for each leaf v in $\hat{G}$, we set $D[v, r, d] = 0$ for all values of r and d.

The observations below result from the fact that the size of the minimum edge dominating set of a smaller subgraph is never bigger than the size of a minimum edge dominating set for a larger subgraph.

Observation 3.3.2. The following properties hold:

1) $D[u, r, d] \leq D[v, r, d]$ for u a descendant of v,

2) $D[v, r, d] \leq D[v, r', d]$ for $r \leq r'$, and

3) $D[v, r, d] \leq D[v, r, d']$ for $d \geq d'$.

Observation 3.3.3. For a given value of v, the minimum values will be $D[v, 0, maxdeficit(v)]$; the solution to DIRECTED (p, q)-EDGE DOMINATING SET will be $D[v, 0, 0]$, for v the root of $\hat{G}$.

Before detailing the calculation of $D[v, r, d]$, we first consider the roles of C_s, $ST(v)$, C_d, and $DT(v)$. For both types of subtrees and connecting edges, a single edge may cover the deficit for all the subtrees and connecting edges of the opposite type. However, the roles of the two types of subtrees and connecting edges are not symmetric. Specifically, since edges in C_d cannot form directed paths with $conn(v)$, only edges in C_s and $ST(v)$ will have an impact on the reach r and deficit d for v. In contrast, in the choice of edges for K in $G[T_v]$, all edges in C_d and all deficits in trees in $DT(v)$ must be covered.

We first observe in the following two lemmas that a single edge in C_s will suffice to ensure that $r = maxreach(v)$, and that if there is no edge in $K \cap C_s$, then it suffices for a single subtree in $ST(v)$ to have reach $r + 1$. In our calculations, this implies that the reach for every other tree in $ST(v)$ can be assumed to be 0 (or for all to have reach 0, if $K \cap C_s \neq \emptyset$).

Lemma 3.3.4. If $K \cap C_s = \emptyset$, the reach of K beyond $G[T_v]$ is one less than the maximum over all vertices $u \in same(v)$ of the reach of K restricted to $G[T_u]$.

Proof of Lemma 3.3.4: If $K \cap C_s = \emptyset$, then the reach of K beyond $G[T_v]$ is one less than $maxreach(v)$, since it will be one less than the maximum over all $u \in same(v)$ of the reach of K beyond $G[T_u]$, and since $u \in same(v)$, $(u, v) \notin K$, and $maxreach(u) = maxreach(v)$. ■

Lemma 3.3.5. The reach of K beyond $G[T_v]$ is $maxreach(v)$ if and only if $K \cap C_s \neq \emptyset$.

Proof of Lemma 3.3.5: Clearly, an edge in $K \cap C_s$ can $maxreach(v)$-out-dominate (resp., $maxreach(v)$-in-dominate) edges outside of $G[T_v]$ if v is an out-vertex (resp., in-vertex). The opposite direction is immediately shown by Lemma 3.3.4. ■

To determine deficit, we first make a few observations that apply to subtrees of both types. For any child u of v, if $conn(u)$ is in K, then we can assume that the deficit within $G[T_u]$ is $maxdeficit(u)$. Furthermore, if $conn(u)$ is not in K, then the maximum deficit possible within $G[T_v]$ is $maxdeficit(u) - 1$. This is summarized in the following two lemmas.

Lemma 3.3.6. For any child u of v, $conn(u)$ covers a deficit of $maxdeficit(u)$ in $G[T_u]$.

Proof of Lemma 3.3.6: If $conn(u) = (u, v)$ (resp., $conn(u) = (u, v)$), $conn(u)$ dominates all

edges e with $dist(e,u) \leq maxdeficit(u) - 1$ (resp., $dist(u,e) \leq maxdeficit(u) - 1$). Thus, $conn(u)$ covers a deficit of $maxdeficit(u)$ in $G[T_u]$. ■

Lemma 3.3.7. For any child u of v, if $conn(u)$ is not included in K, then the maximum possible deficit within $G[T_v]$ that can be covered by K is $maxdeficit(u) - 1$.

Proof of Lemma 3.3.7: If $conn(u) = (u,v)$ (resp., $conn(u) = (v,u)$) is not included in K, $dist(e,v) = dist(e,u) + 1$ (resp., $dist(v,e) = dist(u,e) + 1$) for edge e in the deficit of K within $G[T_u]$. Thus, the maximum possible deficit within $G[T_v]$ that can be covered by K is $maxdeficit(u) - 1$. ■

We make use of the following terminology to capture the idea of determining whether or not to include $conn(u)$ in K, where the deficit can be an arbitrary value j. We define $min(u,j) = \min\{1 + D[u,0,maxdeficit(u)],\ D[u,0,j]\}$; the first case represents choosing to include $conn(u)$ and the second case not to include $conn(u)$. If the latter is the smaller for all $u \in same(v)$ ($u \in diff(v)$, respectively), we may choose to add an arbitrary edge in C_s (C_d, respectively) to cover the deficit for subtrees of the opposite type. Accordingly, we define $\alpha_s(j) = 0$ if there exists $u \in same(v)$ such that $1 + D[u,0,maxdeficit(u)] \leq D[u,0,j]$, and 1 otherwise. We define $\alpha_d(j)$ similarly, for $u \in diff(v)$.

As a consequence of Lemma 3.3.7, we observe that the reach of a connecting edge of the opposite type is sufficient to cover any deficit.

Lemma 3.3.8. For any child u of v, if $conn(u)$ is not included in K, the deficit in $G[T_v]$ will be covered by any single connecting edge of the opposite type. Thus, if $K \cap C_d \neq \emptyset$, $d = 0$.

Proof of Lemma 3.3.8: If $conn(u)$ is not included in K, by Lemma 3.3.7, the maximum possible deficit within $G[T_v]$ that can be covered by K is $maxdeficit(u) - 1$. Any single connecting edge e of the opposite type of $conn(u)$ can $maxdeficit(u)$-in-dominate (resp., $maxdeficit(u)$-out-dominate) edges if u is an out-vertex (resp., in-vertex). Thus, the deficit in $G[T_v]$ will be covered by any single connecting edge of the opposite type.

Moreover, if there exists an edge in $K \cap C_d$, the edge dominates all edges in $ST(u)$ and C_s from the above discussion. Thus, if $K \cap C_d \neq \emptyset$, $d = 0$. ■

We consider using a subtree T_x in $DT(v)$ to cover the deficit in a subtree rooted at a vertex w in $same(v)$. In this case, the reach beyond $G[T_x]$ must be two greater than the deficit in $G[T_w]$, due

to the need for the path to pass through $conn(x)$ and $conn(w)$ en route to $G[T_x]$. This reach for a single subtree in $DT(v)$ will cover the deficit for all subtrees in $ST(v)$.

The analogous result holds for the reach of a single subtree in $ST(v)$ covering the deficit for all subtrees rooted at vertices in $DT(v)$. The key difference is that for the subtrees in $ST(v)$, there remains the option of not covering the deficit. No such option exists for subtrees in $DT(v)$, for which all deficits must be covered.

To determine the value of $D[v, r, d]$, we will consider all possible options for adding edges between v and its children to K, a reach-r-deficit-d edge dominating set for $G[T_v]$, as the choice of edges of K in the subtrees rooted at the children of v will be represented by already-computed table entries. We demonstrate how to compute $D[v, r, d]$ as four different cases, depending on the values of d and r.

Case 1: $r = maxreach(v)$ and $d > 0$

Since $r = maxreach(v)$, by Lemma 3.3.5 $K \cap C_s \neq \emptyset$, and since $d > 0$, by Lemma 3.3.8 $K \cap C_d = \emptyset$.

Since K contains at least one edge in C_s, by Lemma 3.3.8, all edges in C_d are covered, as well as deficits of $maxdeficit(u) - 1$ for each $u \in diff(v)$, which by Lemma 3.3.7 is the maximum possible.

Any $u \in same(v)$ for which $conn(u) \in K$ will contribute $D[u, 0, maxdeficit(u)]$, by Lemma 3.3.6.

If for $u \in same(v)$, $conn(u) \notin K$, then $G[T_u]$ can have a deficit of $d-1$, which in conjunction with $conn(u)$ will result in deficit d. Thus, for each $u \in same(v)$, the contribution is $min(u, d-1)$. We may need to add an arbitrary edge in C_s to ensure $r = maxreach(v)$, as indicated by $\alpha_s(d-1)$.

$$D[v, r, d] = \sum_{u \in diff(v)} D[u, 0, maxdeficit(u) - 1] + \sum_{u \in same(v)} min(u, d-1) + \alpha_s(d-1).$$

Case 2: $r < maxreach(v)$ and $d > 0$

Since $r < maxreach(v)$, by Lemma 3.3.5 $K \cap C_s = \emptyset$, and since $d > 0$, by Lemma 3.3.8 $K \cap C_d = \emptyset$.

To ensure that the edges in C_d are covered and that the deficit of all trees in $DT(v)$ are covered, we make use of an edge in the subtree in $ST(v)$ that gives rise to reach r. We consider all choices

of $w \in same(v)$, for a contribution of $D[w, r+1, d-1]$ for that choice, and $D[u, 0, d-1]$ for all $u \in same(v) \backslash \{w\}$. For each $u \in diff(v)$, the contribution will be $D[u, 0, r-1]$.

Thus,

$$D[v,r,d] = \min_{w \in same(v)} \{ D[w, r+1, d-1] + \sum_{u \in same(v) \backslash \{w\}} D[u, 0, d-1] \} + \sum_{u \in diff(v)} D[u, 0, r-1].$$

Case 3: $r = maxreach(v)$ and $d = 0$

Since $r = maxreach(v)$, by Lemma 3.3.5 $K \cap C_s \neq \emptyset$. Any $u \in same(v)$ for which $conn(u) \in K$ will contribute $1 + D[u, 0, maxdeficit(u)]$, by Lemma 3.3.6.

The edges in C_s cover all deficits of trees T_u in $DT(v)$ up to a deficit of $maxdeficit(u) - 1$; the only other option for a tree in $DT(v)$ is to include $conn(u)$ in K to cover a deficit of up to $maxdeficit(u)$.

To cover the deficits of trees in $ST(v)$, we consider the minimum over all choices of j in the range from 0 to $maxdeficit(v) - 1$ as the deficit for any $T_u \in ST(v)$ such that $conn(u) \notin K$. When $j = maxdeficit(v) - 1$, an edge in C_d must be in K. For all other values of j, one tree in $DT(v)$ must have reach $j + 2$, and all others will have reach 0.

We set $D[v, r, d] = \min_{j \in \{0, \ldots, maxdeficit(v)-1\}} Cost(j)$ for $Cost(j)$ as defined below.

$$Cost(maxdeficit(v) - 1) = \sum_{u \in same(v) \cup diff(v)} min(u, maxdeficit(u) - 1) + \alpha_s(maxdeficit(u) - 1) + \alpha_d(maxdeficit(u) - 1).$$

For any value of j in the range from 0 to $maxdeficit(v) - 2$,

$$Cost(j) = \min_{w \in diff(v)} \{ D[w, j+2, maxdeficit(w) - 1] + \sum_{u \in diff(v) \backslash \{w\}} D[u, 0, maxdeficit(u) - 1] + \sum_{u \in same(v)} min(u, j) + \alpha_s(j) \}.$$

Case 4: $r < maxreach(v)$ and $d = 0$

Since $r < maxreach(v)$, by Lemma 3.3.5 $K \cap C_s = \emptyset$.

If K does not contain any edge in C_d, we need to cover the deficits in trees in $ST(v)$ and $DT(v)$ as well as the edges in C_s and C_d, all without being able to select any of the connecting edges. Since an edge in a subtree in $DT(v)$ can cover the deficit for all trees in $ST(v)$ (as well as all edges in C_s), we consider all trees in $ST(v)$ to have the same deficit, j. We then require a single subtree in $DT(v)$ to have reach $j + 2$, with the rest having reach 0. We consider all possible values of j, $0 \leq j \leq maxdeficit(v) - 2$, and all choices of a subtree in $DT(v)$ to have sufficient reach.

Similarly, we need to ensure that the edges in C_d are covered and that the deficits of all trees in $DT(v)$ are covered. This will be accomplished by an edge in a subtree in $ST(v)$, which is also the one that results in reach r (the rest will have reach 0). Thus all subtrees in $DT(v)$ will have the same deficit, $r - 2$. This immediately implies that $r \geq 2$ in this case. We consider all possible choices of subtrees in $ST(v)$.

If instead K contains any edge in C_d, then by Lemma 3.3.8, K covers the deficits in all subtrees in $ST(v)$, as well as all edges in C_s. To cover the edges of C_d and the deficits in all trees in $DT(v)$, we consider $D[w, r + 1, maxdeficit(w) - 1]$, for all possibilities of $w \in same(v)$, and $D[u, 0, maxdeficit(u) - 1]$ for all $u \in same(v) \backslash \{w\}$ (only one tree needs to contribute to the reach of r for v). Since this reach will cover a deficit of $r - 1$ in any tree in $DT(v)$, the contribution for each $u \in diff(v)$ will be $min(u, r - 1)$, with the possible addition of an arbitrary edge in C_d (represented by $\alpha_d(r - 1)$.

We set $D[v, r, d] = \min_{j \in \{0,\ldots,maxdeficit(v)-1\}} Cost(j)$ for $Cost(j)$ as defined below; the value $j = maxdeficit(v) - 1$ handles the case in which K contains an edge in C_d. Thus,

$$\begin{aligned} Cost(maxdeficit(v) - 1) = \min_{w \in same(v)} \{ & D[w, r + 1, maxdeficit(w) - 1] \\ & + \sum_{u \in same(v) \backslash \{w\}} D[u, 0, maxdeficit(u) - 1] \\ & + \sum_{u \in diff(v)} min(u, r - 1) + \alpha_d(r - 1)\}. \end{aligned}$$

For any value of j in the range from 0 to $maxdeficit(v) - 2$,

$$Cost(j) = \min_{w \in same(v), x \in diff(v)} \{ \ D[w, r + 1, j] + \sum_{u \in same(v) \backslash \{w\}} D[u, 0, j]$$

$$+ D[x, j+2, r-2] + \sum_{u \in diff(v) \setminus \{x\}} D[u, 0, r-2]\}.$$

Let $\gamma = \max\{p, q\}$, and for vertex v let $d(v) = d^{in}(v) + d^{out}(v)$, which represents the degree of v in the underlying undirected graph of G; $d(v) \leq \Delta$ holds. For each v, r, d, the algorithm computes $D[v, r, d]$ in time $O(d(v))$ in Case 1, $O(d(v)^2)$ in Case 2, $O(\gamma d(v)^2)$ in Case 3, and $O(\gamma d(v)^3)$ in Case 4. Thus, for each v, $D[v, r, d]$ can be computed in time $O(\gamma d(v))$ for $r = maxreach(v)$ and all $d > 0$ in Case 1. Similarly, time $O(\gamma^2 d(v)^2)$ suffices for all $r < maxreach(v)$ and $d > 0$ in Case 2, $O(\gamma^2 d(v)^2)$ for $r = maxreach(v)$ and $d = 0$ in Case 3, and $O(\gamma^2 d(v)^3)$ for all $r < maxreach(v)$ and $d = 0$ in Case 4. Therefore, the total running time is in $\sum_{v \in V} O(\gamma^2 d(v)^3) = O(\gamma^2 \sum_{v \in V} d(v)^3) = O(\gamma^2 \Delta^2 \sum_{v \in V} d(v)) = O(\gamma^2 \Delta^2 n)$. Note that since the underlying undirected graph of G is a tree, $\sum_{v \in V} d(v) = 2(n-1)$ by the handshaking lemma. This completes the proof of Theorem 3.3.1. ■

Next, we present a polynomial-time algorithm for DIRECTED r-IN (OUT) VERTEX COVER on trees.

Theorem 3.3.9. There is an algorithm that solves DIRECTED r-IN (OUT) VERTEX COVER on trees in time $O(r(r + \Delta)n)$.

Proof: Since we will use r to denote reach, here we will consider the (renamed) DIRECTED q-OUT VERTEX COVER. The proof for DIRECTED q-IN VERTEX COVER, which is similar, has been omitted. We assume that $q \geq 2$ because the case in which $q = 1$ is trivial in general graphs.

As with DIRECTED (p, q)-EDGE DOMINATING SET, we use $D[v, r, d]$ to store the minimum number of vertices in a reach-r-deficit-d vertex cover for $G[T_v]$.

When processing a vertex v, we determine $D[v, r, d]$ for values of r and d in the ranges $0 \leq r \leq q$ and $0 \leq d \leq q$. For the base cases, for each leaf v and its parent u in $\hat{G}$, if there is an edge (v, u) from v to its parent u, we must include v in the solution in order to cover (v, u). Thus, we define $D[v, r, d]$ as follows:

$$D[v, r, d] = \begin{cases} 1 & (1 \leq r \leq q \wedge d = 0) \\ +\infty & \text{(otherwise).} \end{cases}$$

If there is an edge (u, v) from parent u for a leaf v, there is no reachable path from v to any

vertex. Moreover, v is not included in any minimum q-out vertex cover because v does not cover any edge. Thus, we define $D[v, r, d]$ as follows:

$$D[v, r, d] = \begin{cases} 0 & (r = 0 \wedge 0 \leq d \leq q - 1) \\ +\infty & \text{(otherwise).} \end{cases}$$

As with DIRECTED (p, q)-EDGE DOMINATING SET, the observations below result from the fact that the size of the minimum q-out vertex cover of a smaller subgraph is never bigger than the size of a minimum q-out vertex cover for a larger subgraph.

Observation 3.3.10. The following properties hold:

1) $D[u, r, d] \leq D[v, r, d]$ for u a descendant of v,

2) $D[v, r, d] \leq D[v, r', d]$ for $r \leq r'$, and

3) $D[v, r, d] \leq D[v, r, d']$ for $d \geq d'$.

Observation 3.3.11. For a given value of v, the minimum value will be $D[v, 0, r]$; the solution to DIRECTED q-OUT VERTEX COVER will be $D[v, 0, 0]$, for v the root of $\hat{G}$.

Thus, we compute $D[v, 0, 0]$ by using dynamic programming from the leaves. For each v, we define the recursive formulas. We consider two types of vertices.

Case 1: v such that there is an edge (v, u) to parent u.

In this case, if $r = 0$ or $d > 0$, we set $D[v, r, d] = +\infty$ because uncovered edges in $G[T_v]$ and (v, u) are never covered henceforth. Otherwise, we consider two cases:

(a) When $r = q$ and $d = 0$, v must be included in the solution. Therefore, we set

$$D[v, r, d] = \sum_{u \in diff(v)} D[u, 0, q - 1] + \sum_{w \in same(v)} D[w, 1, 0] + 1.$$

Because v and any vertex above v in a tree never cover any edge (w, v) for $w \in same(v)$, we need the reach of 1 in $D[w, 1, 0]$.

(b) When $1 \leq r \leq q-1$ and $d = 0$, v must not be included in the solution. Therefore, we set

$$D[v, r, d] = \min_{x \in same(v)} \{ D[x, r+1, 0] + \sum_{u \in diff(v)} D[u, 0, r-1] + \sum_{w \in same(v) \setminus \{x\}} D[w, 1, 0] \}.$$

In the recursive formula, every edge in $DT(v)$ is covered by x. As with Case 1(a), we need the reach of 1 in $D[w, 1, 0]$ since v and any vertex above v in a tree never cover any edge (w, v).

Case 2: v such that there is an edge (u, v) from parent u.
In this case, if $r > 0$, we set $D[v, r, d] = +\infty$ because v does not reach any vertex above itself in $G[T_v]$. Otherwise, we consider two cases:

(a) When $r = 0$ and $d > 0$, v must not be included in the solution. Therefore, we set

$$D[v, r, d] = \sum_{u \in diff(v)} D[u, 1, 0] + \sum_{w \in same(v)} D[w, 0, d-1].$$

Because v and any vertex above v in a tree never cover any edge (u, v) for $u \in diff(v)$, we need the reach of 1 in $D[u, 1, 0]$.

(b) When $r = 0$ and $d = 0$, we set

$$D[v, r, d] = \min \Bigg\{ \sum_{u \in diff(v)} D[u, 1, 0] + \sum_{w \in same(v)} D[w, 0, q-1] + 1, \\ \min_{0 \leq j \leq q-2} \Big\{ \min_{x \in diff(v)} \{ D[x, j+2, 0] + \sum_{u \in diff(v) \setminus \{x\}} D[u, 1, 0] + \sum_{w \in same(v)} D[w, 0, j] \} \Big\} \Bigg\}.$$

The first part is the case in which v is included in the solution. Since v q-covers edges in $ST(v)$, we sum up $D[w, 0, q]$ for $w \in same(v)$. The second part is the other case, that is, v is not included in the solution. In this case, we consider every possible combination of the reach of $DT(v)$ and the deficit of $ST(v)$ such that every edge in $G[T_v]$ is covered. Then we set a minimum possible solution in $G[T_v]$ as $D[v, r, d]$.

Let $d(v) = d^{in}(v) + d^{out}(v)$ for vertex v, which represents the degree of v in the underlying undirected graph of G; $d(v) \leq \Delta$ holds. For each v, the algorithm computes $D[v, r, d]$ in time $O(q^2) + O(d(v)) + O(qd(v)^2)$ in Case 1 and $O(q^2) + O(qd(v)) + O(qd(v)^2)$ in Case 2, respectively. Therefore, the total running time is in

$\sum_{v \in V} O(q^2 + qd(v)^2) = O(q^2 n + q\Delta \sum_{v \in V} d(v)) = O(q(q+\Delta)n)$ by the same argument of the proof of Theorem 3.3.1. ■

3.3.2 Algorithms on graphs of bounded treewidth

In this subsection, we present a $25^{\omega}\omega^{O(1)}n$-time algorithm on a tree decomposition of width at most ω for DIRECTED $(1,0)$-EDGE ($(0,1)$-EDGE, $(1,1)$-EDGE) DOMINATING SET. We first show an algorithm for DIRECTED $(1,1)$-EDGE DOMINATING SET, which shows the following theorem.

Theorem 3.3.12. Given a directed graph G, let G^* be the underlying undirected graph of G. Then given a tree decomposition of G^* of width at most ω, there exists an algorithm that solves DIRECTED $(1,1)$-EDGE DOMINATING SET in time $25^{\omega}\omega^{O(1)}n$.

Proof: We use dynamic programming based on a tree decomposition of width ω. For vertex v, (u,v) is called an *incoming edge* of v and (v,u) is called an *outgoing edge* of v. Let D be the solution set. We fill table entries for each node. For a representation of the state of v in a bag, we define the *coloring* function:

$$c : V \to \{\mathbf{0}, \mathbf{0}_{in}, \mathbf{0}_{out}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}.$$

Each element of $\{\mathbf{0}, \mathbf{0}_{in}, \mathbf{0}_{out}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$ is called a *state* and has the following meanings:

- $\mathbf{0}$: v is an endpoint of an edge not included in D.
- $\mathbf{0}_{in}$: v is an endpoint of an edge not included in D, but will become an endpoint of an incoming edge of v included in D.
- $\mathbf{0}_{out}$: v is an endpoint of an edge not included in D, but will become an endpoint of an outgoing edge of v included in D.
- $\mathbf{0}_{inout}$: v is an endpoint of an edge not included in D, but will become an endpoint of both an incoming edge and an outgoing edge of v included in D.
- $\mathbf{0}^{in}_{inout}$: v is an endpoint of an incoming edge of v included in D and will also become an endpoint of an outgoing edge of v included in D.

- $\mathbf{0}_{inout}^{out}$: v is an endpoint of an outgoing edge of v included in D and will also become an endpoint of an incoming edge of v included in D.
- $\mathbf{1}_{in}$: v is an endpoint of an incoming edge of v included in D.
- $\mathbf{1}_{out}$: v is an endpoint of an outgoing edge of v included in D.
- $\mathbf{1}_{inout}$: v is an endpoint of both an incoming edge and an outgoing edge of v included in D.

Let $\Sigma = \{\mathbf{0}, \mathbf{0}_{in}, \mathbf{0}_{out}, \mathbf{0}_{inout}, \mathbf{0}_{inout}^{in}, \mathbf{0}_{inout}^{out}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$. Given two vertex sets V and W, we denote their colorings by $c_V \in \Sigma^{|V|}$ and $c_W \in \Sigma^{|W|}$, respectively. Suppose that $c_V = (c(v_1), \ldots, c(v_{|V|}))$ and $c_W = (c(w_1), \ldots, c(w_{|W|}))$, where $v_i \in V$ and $w_i \in W$. Then we define the *concatenation* $c_V \times c_W \in \Sigma^{|V|+|W|}$ of c_V and c_W as the coloring $(c(v_1), \ldots, c(v_{|V|}), c(w_1), \ldots, c(w_{|W|}))$.

Suppose that $W \subseteq V$. Then we define the *separation* $c_V \setminus c_W \in \Sigma^{|V|-|W|}$ as the coloring $(c(v_{i_1}), \ldots, c(v_{i_{|V|-|W|}}))$, where $v_{i_1}, \ldots, v_{i_{|V|-|W|}} \in V \setminus W$.

Given a tree decomposition $\langle \mathcal{X}, T \rangle$, we define a subgraph $G_i = (V_i, E_i)$ for each node i where V_i is the union of all bags X_j with $j = i$ or j a descendant of i in T, and $E_i \subseteq E$ is the set of edges introduced in the subtree rooted at node i. Then we define a *partial solution* in node i as a subset of E_i that possibly dominates every edge in an induced subgraph of V_i.

For each node i, we define the function:

$$f_i : \Sigma^{|X_i|} \to \mathbb{N} \cup \{+\infty\}.$$

This function's value represents the minimum size of a possible partial solution in node i. Let $c_i \in \Sigma^{|X_i|}$ be a coloring of node i. If $f_i(c_i) = +\infty$, it means that the coloring c_i is invalid. We obtain the minimum size of the solution by computing $f_i(c_i)$ by dynamic programming on a tree decomposition. For simplicity, we sometimes denote c_i by c.

Leaf node: In leaf nodes, we define $f_i(\emptyset) := 0$.

Introduce vertex v node: If $c(v) \in \{\mathbf{0}_{inout}^{in}, \mathbf{0}_{inout}^{out}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$, we set $f_i(c) := +\infty$. That is because there is no edge incident to v in node i while each state in

(u,v)	$\mathbf{0}$	$\mathbf{0}_{in}$	$\mathbf{0}_{out}$	$\mathbf{0}_{inout}$	$\mathbf{0}^{in}_{inout}$	$\mathbf{0}^{out}_{inout}$	$\mathbf{1}_{in}$	$\mathbf{1}_{out}$	$\mathbf{1}_{inout}$
$\mathbf{0}$	-	-	(1)	(1)	(1)	(1)	-	(1)	(1)
$\mathbf{0}_{in}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{0}_{out}$	-	-	(1)	(1)	(1)	(1)	-	(1)	(1)
$\mathbf{0}_{inout}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{0}^{in}_{inout}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{0}^{out}_{inout}$	(1)	(1)	(1)	(1)	(2)	(1)	(3)	(1)	(4)
$\mathbf{1}_{in}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{1}_{out}$	-	-	(1)	(1)	(5)	(1)	(6)	(1)	(7)
$\mathbf{1}_{inout}$	(1)	(1)	(1)	(1)	(8)	(1)	(9)	(1)	(10)

Table 3.2. Entries indicate case numbers that apply for the possible states of u and v for the introduce node for the edge (u,v). The labels for the rows correspond to $c(u)$ and the labels for the columns correspond to $c(v)$.

$\{\mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$ represents that v is incident on at least one edge. Note that an introduce vertex node only adds v, and does not add any edge. Thus, the fact that vertex v has no edge is inconsistent with $c(v) \in \{\mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$. Otherwise, we set $f_i(c) := f_j(c \setminus \{c(v)\})$, that is, we set $f_i(c)$ as the value of f_j for the same coloring as c in its child j.

Introduce edge (u,v) node: Table 3.2 shows all cases of the possible states of u and v for the introduce node for the edge (u,v). The point of the recursive formulas is to check whether the states of the endpoints of (u,v) in node j are consistent with the ones in node i. We set the recursive formula taking the minimum value of a valid coloring in node j.

Case 1. In case 1 (see Table 3.2), we cannot add edge (u,v) to D, since adding (u,v) to D is inconsistent with the states of u and v in introduce node i for the edge (u,v). For example, if $c(u) = \mathbf{0}$ and $c(v) = \mathbf{0}_{out}$ in node i, the states of u and v imply that u is an endpoint of edge in D and v is an endpoint of an edge not included in D. However, if (u,v) is added to D in node i, u becomes an endpoint of an edge in D and v becomes an endpoint of an edge not included in D. This is inconsistent with the states of u and v. Thus, (u,v) must not be included in D. The other cases are similar. In these cases, we set $f_i(c) := f_j(c)$ since we do not add edge (u,v) to D and consequently the size of a partial solution does not change.

Case 2. If $c(u) = \mathbf{0}^{out}_{inout}$ and $c(v) = \mathbf{0}^{in}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\})\}. \end{aligned}$$

In a child node j, if $c_j(u), c_j(v) = \mathbf{0}_{inout}$, u and v do not have any edge in the solution yet. If we add (u, v) to the solution in the introduce edge (u, v) node, the states of u and v are changed to $\mathbf{0}^{out}_{inout}$ and $\mathbf{0}^{in}_{inout}$ since u becomes incident to an outgoing edge in the solution and v becomes incident to an incoming edge. The second and third equations are similar cases in which the states of u and v are changed to $\mathbf{0}^{out}_{inout}$ and $\mathbf{0}^{in}_{inout}$. For the last equation, u and v already have an outgoing and incoming edge in the solution, respectively. Moreover, edge (u, v) is dominated even if it is not included in the solution. Thus, we set $f_i(c \times \{c(u)\} \times \{c(v)\})$ as described above. Similar arguments apply to the remaining cases below.

Case 3. If $c(u) = \mathbf{0}^{out}_{inout}$ and $c(v) = \mathbf{1}_{in}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}_{in}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{1}_{in}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}_{in}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{1}_{in}\})\}. \end{aligned}$$

Case 4. If $c(u) = \mathbf{0}^{out}_{inout}$ and $c(v) = \mathbf{1}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{1}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{1}_{inout}\})\}. \end{aligned}$$

Case 5. If $c(u) = \mathbf{1}_{out}$ and $c(v) = \mathbf{0}^{in}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}^{in}_{inout}\})\}. \end{aligned}$$

Case 6. In the case in which $c(u) = \mathbf{1}_{out}$ and $c(v) = \mathbf{1}_{in}$, if we do not add (u, v) to the solution, it is never dominated. Thus, edge (u, v) must be added to the solution in any case. Thus we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}_{in}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{1}_{in}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}_{in}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{1}_{in}\}) + 1\}. \end{aligned}$$

Case 7. If $c(u) = \mathbf{1}_{out}$ and $c(v) = \mathbf{1}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{1}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{1}_{inout}\})\}. \end{aligned}$$

Case 8. If $c(u) = \mathbf{1}_{inout}$ and $c(v) = \mathbf{0}^{in}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\})\}. \end{aligned}$$

Case 9. If $c(u) = \mathbf{1}_{inout}$ and $c(v) = \mathbf{1}_{in}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}_{out}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{1}_{out}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}_{out}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{1}_{out}\})\}. \end{aligned}$$

Case 10. If $c(u) = \mathbf{1}_{inout}$ and $c(v) = \mathbf{1}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{1}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{1}_{inout}\})\}. \end{aligned}$$

For each entry denoted by "-" in Table 3.2, we set $f_i(c) := +\infty$ since edge (u, v) is never dominated.

Forget v node: Suppose that $X_i = X_j \setminus \{v\}$ for a forget node i and its child j. Let $c_j = c \times \{c_j(v)\}$ and D be a set of c_j's such that one of these sets of conditions holds:

1) $c_j(v) = \mathbf{0}$ and $\forall u \in N^{in}(v) \cap X_j, c(u) \in \{\mathbf{0}_{in}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{inout}\}$ and $\forall w \in N^{out}(v) \cap X_j, c(w) \in \{\mathbf{0}_{out}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$,

2) $c_j(v) = \mathbf{1}_{in}$ and $\forall u \in N^{in}(v) \cap X_j, c(u) \in \{\mathbf{0}_{in}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$,

3) $c_j(v) = \mathbf{1}_{out}$ and $\forall w \in N^{out}(v) \cap X_j, c(w) \in \{\mathbf{0}_{out}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$,

4) $c_j(v) = \mathbf{1}_{inout}$.

The conditions of set D mean that every edge incident to v is dominated. In condition 2, note that any $(u, v) \in E$ such that $c_j(u) = \mathbf{1}_{out}$ is included in K in the introduce edge (u, v) node. Similarly, in condition 3, any $(u, v) \in E$ such that $c_j(u) = \mathbf{1}_{in}$ is included in K. Then, we set $f_i(c) := \min_{c_j \in D} f_j(c_j)$.

	$\mathbf{0}$	$\mathbf{0}_{in}$	$\mathbf{0}_{out}$	$\mathbf{0}_{inout}$	$\mathbf{0}^{in}_{inout}$	$\mathbf{0}^{out}_{inout}$	$\mathbf{1}_{in}$	$\mathbf{1}_{out}$	$\mathbf{1}_{inout}$
$\mathbf{0}$	$\mathbf{0}$								
$\mathbf{0}_{in}$		$\mathbf{0}_{in}$					$\mathbf{1}_{in}$		
$\mathbf{0}_{out}$			$\mathbf{0}_{out}$					$\mathbf{1}_{out}$	
$\mathbf{0}_{inout}$				$\mathbf{0}_{inout}$	$\mathbf{0}^{in}_{inout}$	$\mathbf{0}^{out}_{inout}$			$\mathbf{1}_{inout}$
$\mathbf{0}^{in}_{inout}$				$\mathbf{0}^{in}_{inout}$	$\mathbf{0}^{in}_{inout}$	$\mathbf{1}_{inout}$			$\mathbf{1}_{inout}$
$\mathbf{0}^{out}_{inout}$				$\mathbf{0}^{out}_{inout}$	$\mathbf{1}_{inout}$	$\mathbf{0}^{out}_{inout}$			$\mathbf{1}_{inout}$
$\mathbf{1}_{in}$		$\mathbf{1}_{in}$					$\mathbf{1}_{in}$		
$\mathbf{1}_{out}$			$\mathbf{1}_{out}$					$\mathbf{1}_{out}$	
$\mathbf{1}_{inout}$				$\mathbf{1}_{inout}$	$\mathbf{1}_{inout}$	$\mathbf{1}_{inout}$			$\mathbf{1}_{inout}$

Table 3.3. Entries indicate $c_i(v)$ for possible combinations of $c_{j_1}(v)$ and $c_{j_2}(v)$ for v in a join node i which has two children node j_1 and j_2.

Join node: We assume that a join node i has two children nodes j_1, j_2. According to Table 3.3, we have 25 combinations of states for each v in a join node. Let D' be a tuple of two colorings c_{j_1}, c_{j_2} such that for each vertex v, $c_{j_1}(v)$ and $c_{j_2}(v)$ satisfy the condition of Table 3.3. Then we set the recursive formula as follows:

$$f_i(c) := \min_{c_{j_1}, c_{j_2} \in D'} f_{j_1}(c_{j_1}) + f_{j_2}(c_{j_2}).$$

Note that there is no edge (u, v) for $u, v \in X_i$ in G_i because we assume that the parent node of an introduce edge (u, v) node is an introduce edge (u, w), introduce edge (v, x), forget u, or forget v node for some w or x.

Finally, we bound the running time. In each introduce vertex, introduce edge, and forget node, we can compute every recursive formula in time $9^\omega \omega^{O(1)}$. For each join node, we have 25 combinations of states for each v, and hence it takes $25^\omega \omega^{O(1)}$-time to compute every recursive formula. Therefore, the total running time is $25^\omega \omega^{O(1)} n$. ■

By a slight modification, we also obtain a $25^\omega \omega^{O(1)} n$-time algorithm for DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE) DOMINATING SET.

Theorem 3.3.13. Given a directed graph G, let G^* be an undirected graph underlying in G. Then given a tree decomposition of G^* of width at most ω, there exists an algorithm that solves DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE) DOMINATING SET in time $25^\omega \omega^{O(1)} n$.

(u,v)	$\mathbf{0}$	$\mathbf{0}_{in}$	$\mathbf{0}_{out}$	$\mathbf{0}_{inout}$	$\mathbf{0}^{in}_{inout}$	$\mathbf{0}^{out}_{inout}$	$\mathbf{1}_{in}$	$\mathbf{1}_{out}$	$\mathbf{1}_{inout}$
$\mathbf{0}$	-	-	-	-	-	-	-	-	-
$\mathbf{0}_{in}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{0}_{out}$	-	-	-	-	-	-	-	-	-
$\mathbf{0}_{inout}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{0}^{in}_{inout}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{0}^{out}_{inout}$	(1)	(1)	(1)	(1)	(2)	(1)	(3)	(1)	(4)
$\mathbf{1}_{in}$	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
$\mathbf{1}_{out}$	-	-	-	-	(5)	-	(6)	-	(7)
$\mathbf{1}_{inout}$	(1)	(1)	(1)	(1)	(8)	(1)	(9)	(1)	(10)

Table 3.4. Entries indicate case numbers that apply for possible states of u and v for the introduce node for the edge (u,v). The labels for the rows correspond to $c(u)$ and the labels for the columns correspond to $c(v)$.

Proof: We only change the recursive formulas of introduce edge and forget nodes in the algorithm for DIRECTED $(1,1)$-EDGE DOMINATING SET.

Introduce edge (u,v) node: Table 3.4 shows all cases of state of u and v in an introduce edge (u,v) node.

Case 1. In case 1 (see Table 3.4), we cannot add edge (u,v) to D, since adding (u,v) to D is inconsistent with the states of u and v as with DIRECTED $(1,1)$-EDGE DOMINATING SET. Therefore, we set $f_i(c) := f_j(c)$ since we do not add edge (u,v) to D and consequently the size of a partial solution does not change.

Case 2. If $c(u) = \mathbf{0}^{out}_{inout}$ and $c(v) = \mathbf{0}^{in}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\})\}. \end{aligned}$$

Case 3. If $c(u) = \mathbf{0}^{out}_{inout}$ and $c(v) = \mathbf{1}_{in}$, we define:

$$f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}_{in}\}) + 1,$$

$$f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{1}_{in}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}_{in}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{1}_{in}\})\}.$$

Case 4. If $c(u) = \mathbf{0}^{out}_{inout}$ and $c(v) = \mathbf{1}_{inout}$, we define:

$$f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}_{inout}\} \times \{\mathbf{1}_{inout}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{1}_{inout}\})\}.$$

Case 5. In the case in which $c(u) = \mathbf{1}_{out}$ and $c(v) = \mathbf{0}^{in}_{inout}$, if we do not add (u, v) to the solution, it is never dominated. Thus, we must add edge (u, v) to the solution in any case. Thus we define:

$$f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}_{inout}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1,$$
$$f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}_{inout}\}) + 1,$$
$$f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1\}.$$

Case 6. In the case in which $c(u) = \mathbf{1}_{out}$ and $c(v) = \mathbf{1}_{in}$, if we do not add (u, v) to the solution, it is never dominated. Thus, we must add edge (u, v) to the solution in any case. Thus we define:

$$f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}_{in}\}) + 1,$$
$$f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{1}_{in}\}) + 1,$$
$$f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}_{in}\}) + 1,$$
$$f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{1}_{in}\}) + 1\}.$$

Case 7. In the case in which $c(u) = \mathbf{1}_{out}$ and $c(v) = \mathbf{1}_{inout}$, if we do not add (u, v) to the solution, it is never dominated. Thus, we must add edge (u, v) to the solution in any case. Thus we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}_{out}\} \times \{\mathbf{1}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{out}\} \times \{\mathbf{1}_{inout}\}) + 1\}. \end{aligned}$$

Case 8. If $c(u) = \mathbf{1}_{inout}$ and $c(v) = \mathbf{0}^{in}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}^{in}_{inout}\})\}. \end{aligned}$$

Case 9. If $c(u) = \mathbf{1}_{inout}$ and $c(v) = \mathbf{1}_{in}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{0}_{out}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{out}_{inout}\} \times \{\mathbf{1}_{out}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}_{out}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{1}_{out}\})\}. \end{aligned}$$

Case 10. If $c(u) = \mathbf{1}_{inout}$ and $c(v) = \mathbf{1}_{inout}$, we define:

$$\begin{aligned} f_i(c \times \{c(u)\} \times \{c(v)\}) := \min\{\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{0}^{in}_{inout}\} \times \{\mathbf{1}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{0}^{out}_{inout}\}) + 1, \\ & f_j(c \times \{\mathbf{1}_{inout}\} \times \{\mathbf{1}_{inout}\})\}. \end{aligned}$$

For the case we denote by "-" in Table 3.2, we set $f_i(c) := +\infty$ since edge (u, v) is never dominated.

Forget v node: Suppose that $X_i = X_j \setminus \{v\}$ for a forget node i and its child j. Let $c_j = c \times \{c_j(v)\}$ and D be a set of c_j's such that one of these sets of conditions holds:

1) $c_j(v) \in \{\mathbf{1}_{in}, \mathbf{1}_{inout}\}$ and $\forall u \in N^{in}(v) \cap X_i, c(u) \in \{\mathbf{0}_{in}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$,

2) $c_j(v) = \mathbf{1}_{out}$ and $\forall u \in N^{in}(v) \cap X_i, c(u) \in \{\mathbf{0}_{in}, \mathbf{0}_{inout}, \mathbf{0}^{in}_{inout}, \mathbf{0}^{out}_{inout}, \mathbf{1}_{out}, \mathbf{1}_{inout}\}$ and $\forall w \in N^{out}(v) \cap X_i, c(w) \in \{\mathbf{0}^{in}_{inout}, \mathbf{1}_{in}, \mathbf{1}_{inout}\}$.

The conditions of set D mean that every edge incident to v is dominated. Then, we set $f_i(c) := \min_{c_j \in D} f_j(c_j)$.

The running time of this algorithm is dominated by join nodes. Therefore, this algorithm runs in time $25^{\omega}\omega^{O(1)}n$. ■

3.3.3 Fixed-parameter algorithms with respect to k

In this subsection, we give fixed-parameter algorithms in time $2^{O(k)}n$ for DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE, $(1,1)$-EDGE) DOMINATING SET. The procedures of our algorithms are as follows. We first obtain a tree decomposition of small width, and then we run $25^{\omega}\omega^{O(1)}n$-time algorithms in Theorems 3.3.12 and 3.3.13, respectively.

To obtain a tree decomposition of small width, we use the following theorem proved by Bodlaender et al. [8].

Theorem 3.3.14 ([8]). There exists an algorithm that, given an n-vertex graph G and an integer ω, in time $2^{O(\omega)}n$ either outputs that the treewidth of G is larger than ω, or constructs a tree decomposition of G of width at most $5\omega + 4$.

Then, we show the treewidth of the underlying undirected graph of G is bounded by twice the minimum size of a directed $(1,1)$-edge dominating set.

Lemma 3.3.15. Given a directed graph G, let G^* be the underlying undirected graph of G and s be the minimum size of a directed $(1,1)$-edge dominating set on G. Then the following inequality holds: $\mathbf{tw}(G^*) \leq 2s$.

Proof: Let G^* be an undirected graph, $\mathbf{tw}(G^*)$ be the treewidth of G^*, and $\mathbf{vc}(G^*)$ be the size of minimum vertex cover. Then we have $\mathbf{tw}(G^*) \leq \mathbf{vc}(G^*)$ [32]. Let M^* be a minimum maximal matching in G^*. A minimum $(1,1)$-edge dominating set in G is a (not necessarily minimum) edge dominating set in G^*. Otherwise, there is an edge not dominated by the $(1,1)$-edge dominating set in G. Moreover, for any edge dominating set D in undirected graphs, $|D| \geq |M^*|$ holds because a minimum maximal matching is a minimum edge dominating set [116]. Therefore, $s \geq |M^*|$ holds. On the other hand, we have a well-known result that for any maximal matching M, $\mathbf{vc}(G^*) \leq 2|M|$ [41]. Moreover, we already know that $\mathbf{tw}(G^*) \leq \mathbf{vc}(G^*)$ holds. Finally, we can obtain $\mathbf{tw}(G^*) \leq 2s$. ■

Finally, DIRECTED $(1,1)$-EDGE DOMINATING SET can be solved in the following time.

Theorem 3.3.16. An instance (G,k) of DIRECTED $(1,1)$-EDGE DOMINATING SET can be solved in time $2^{O(k)}n$.

Proof: Given an instance (G,k), we first determine whether the treewidth of G^* is at most $2k$ in time $2^{O(k)}n$ by using Theorem 3.3.14. If $\mathbf{tw}(G^*) > 2k$, we conclude that it is a no-instance by Lemma 3.3.15. Otherwise, we use the $25^{\omega}\omega^{O(1)}n$-time algorithm based on a tree decomposition of width at most $10k+4$ obtained by Theorem 3.3.14. Therefore, the total running time is $2^{O(k)}n + 25^{10k+4}(10k+4)^{O(1)}n = 2^{O(k)}n$. ■

Thus, the DIRECTED $(1,1)$-EDGE DOMINATING SET problem is fixed-parameter tractable with respect to k. We emphasize that the running time of this algorithm is single exponential in k and linear in n.

In the same way, we can prove DIRECTED $(0,1)$-EDGE ($(1,0)$-EDGE) DOMINATING SET is fixed-parameter tractable with respect to k. First, we obtain the following lemma corresponding to Lemma 3.3.15.

Lemma 3.3.17. Given a directed graph G, let G^* be the underlying undirected graph of G and s' be the minimum size of a directed $(0,1)$-edge ($(1,0)$-edge) dominating set on G. Then the following inequality holds: $\mathbf{tw}(G^*) \leq 2s'$.

Proof: Let s be the minimum size of a directed $(1,1)$-edge dominating set on G. From the definition of directed $(0,1)$-edge ($(1,0)$-edge) dominating set and directed $(1,1)$-edge dominating set, $s' \geq s$. Thus, we have $\mathbf{tw}(G^*) \leq 2s'$ from Lemma 3.3.15. ■

Using an argument similar to that used in the proof of Theorem 3.3.16, we obtain the following theorem.

Theorem 3.3.18. An instance (G, k) of DIRECTED $(0, 1)$-EDGE, $((1, 0)$-EDGE) DOMINATING SET can be solved in time $2^{O(k)}n$.

3.4 Conclusions and future directions

In this chapter, we study covering and domination problems on directed graphs, that is, DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET. We first showed the NP-hardness of these problems even on some restricted graphs. We also proved that DIRECTED r-IN (OUT) VERTEX COVER is $W[2]$-hard when $r \geq 2$ and DIRECTED (p, q)-EDGE DOMINATING SET is $W[2]$-hard when $p \geq 2$ or $q \geq 2$ on directed acyclic graphs. Moreover, for these problems, we showed that there is no polynomial-time $c \ln k$-approximation algorithm for any constant $c < 1$ unless P=NP, where k is the size of an optimal solution, though they can be approximated within ratio $O(\log n)$ by a greedy algorithm.

On the other hand, we designed polynomial-time algorithms for these problems on trees. We finally showed DIRECTED $(0, 1)$-EDGE ($(1, 0)$-EDGE, $(1, 1)$-EDGE) DOMINATING SET are fixed-parameter tractable with respect to treewidth and the solution size, respectively. In particular, the running time of these algorithms are linear in n, and single exponential in treewidth and the solution size, respectively.

Directions for future work include improving the running time and the computational complexity on other restricted graphs.

Chapter 4

On the Maximum Weight Minimal Separator

4.1 Introduction

Given a connected graph $G = (V, E)$ and two vertices $s, t \in V$, a set $S \subseteq V$ of vertices is called an s-t *separator* if s and t belong to different connected components in $G \setminus S$, where $G \setminus S = (V \setminus S, E)$. If a set S is an s-t separator for some s and t, it is simply called a *separator*. If an s-t separator S is minimal in terms of set inclusion, that is, no proper subset of S also separates s and t, it is called a *minimal s-t separator*. Similarly, if a separator is minimal in terms of set inclusion, it is called a *minimal separator*.

Separators and minimal separators are important in several contexts and have indeed been studied intensively. For example, they are related to the connectivity of graphs, which is an important notion in many practical applications, such as network design, supply chain analysis and so on. From a theoretical point of view, minimal separators are related to treewidth or potential maximal cliques, which play key roles in designing fast algorithms [10, 13].

Assume that G does not have the edge (s, t), that is, $(s, t) \notin E$. In this chapter, we consider the problem of finding a maximum weight minimal s-t separator of a given vertex-weighted graph. More precisely, the problem is defined as follows: Given a connected graph $G = (V, E)$, vertices $s, t \in V$ and a weight function $w : V \to \mathbb{N}^+$, find a minimal s-t separator whose weight $\sum_{v \in S} w(v)$ is maximum. The decision version of the problem is to decide the existence of minimal s-t separator

with weight W. We call this problem MAXIMUM WEIGHT MINIMAL s-t SEPARATOR. Similarly, MAXIMUM WEIGHT MINIMAL SEPARATOR is the following problem: Given a connected graph $G = (V, E)$ and a weight function $w : V \to \mathbb{N}^+$, find a minimal separator whose weight $\sum_{v \in S} w(v)$ is maximum. The decision version of the problem is to decide the existence of minimal separator with weight W.

This problem arises in the context of supply chain network analysis. When a weighted network represents a supply chain where a vertex represents an industry, s and t are virtual vertices representing source and sink respectively, and the weight of a vertex represents its financial importance, the maximum weight minimal s-t separator is interpreted as the most important set of industries that is influential or vulnerable in the supply chain network.

On the negative side, we show that these problems are NP-hard on bipartite graphs, even if all the vertex weights are identical, that is, the problem is to find the maximum size of minimal (s-t) separator. On the other hand, we show that MAXIMUM WEIGHT MINIMAL SEPARATOR is fixed-parameter tractable with respect to the solution size and weight. We then design FPT algorithms with respect to treewidth. It should be noted that since the condition of s-t connectivity can be written in Monadic Second Order Logic, given a tree decomposition of width at most ω, it can be solved in $f(\omega)n$ time by Courcelle's meta-theorem, where f is a computable function. However, the function f forms a tower of exponentials; the existence of an FPT algorithm with better running time is not obvious.

In this chapter, we propose two parameterized algorithms for MAXIMUM WEIGHT s-t MINIMAL SEPARATOR with respect to treewidth. One is an $\omega^{O(\omega)}n$-time deterministic algorithm and the other is an $O^*(c^\omega \cdot W^2)$-time randomized algorithm for the decision version, where c is a constant, ω is the width of a tree decomposition, and O^* is the order notation omitting the polynomial factor. The former algorithm is based on a standard dynamic programming approach, whereas the latter utilizes two techniques recently developed. The first technique is called *Cut & Count*, and by using this, the running time is bounded by a single exponential of treewidth. Furthermore, by applying the second technique called *fast convolution*, we improve the running time by reducing the base of the exponent from $c = 21$ to $c = 9$; the total running time of the resulting algorithm is $O^*(9^\omega \cdot W^2)$, which can be further improved when the graph is unweighted. It is noted that MAXIMUM WEIGHT MINIMAL SEPARATOR can be solved by applying the above algorithms for

all possible combinations of s and t, i.e., at most n^2 times.

4.2 Related work

4.2.1 The number of minimal separators

Minimal separators have been investigated for a long time in many aspects. As mentioned above, they are related to treewidth or potential maximal cliques, for example [10, 13]. In general, a graph may have exponentially many minimal separators, and in fact there exists a graph with $\Omega(3^{n/3})$ minimal separators [34]. Recently, this bound was improved to $\Omega(1.4521^n)$ [43]. On the other hand, some graph classes have only polynomially (even linearly) many minimal separators. For example, Bouchitté showed that weakly triangulated (weakly chordal) graphs have a polynomial number of separators [12]. As examples of other graph classes with polynomially many minimal separators, there are circular-arc graphs [62] and polygon-circle graphs, which are a superclass of circle graphs [107].

On the other hand, Berry et al. presented an $O(n^3 R_{sep})$-time algorithm that enumerates all the minimal separators where R_{sep} is the number of these [2]. By combining these results, we know that MAXIMUM WEIGHT MINIMAL s-t SEPARATOR can be solved in polynomial time for the graph classes mentioned above. That is, we just enumerate all the minimal separators and evaluate the weights of these for such graphs.

Proposition 4.2.1. MAXIMUM WEIGHT MINIMAL s-t SEPARATOR and MAXIMUM WEIGHT MINIMAL SEPARATOR can be solved in polynomial time for a graph classes that have a polynomial number of minimal separators.

4.2.2 Relationship between minimal separators and treewidth

Minimal separators and treewidth are strongly related. As for the number of minimal separators, if a graph has a polynomially many minimal separators, we can compute its treewidth in polynomial time [12, 13]. Such graph classes include (amongst others) circular-arc graphs ($O(n^2)$ [62]), polygon-circle graphs ($O(n^2)$ [107]), weakly triangulated graphs ($O(n^2)$ [12]). Furthermore, computing treewidth is fixed-parameter tractable with respect to the maximum size of a minimal separator [105]. This parameter corresponds to the solution size of MAXIMUM WEIGHT MINIMAL

SEPARATOR on unweighted graphs. In this sense, this thesis focuses on the converse relation of these two parameters: maximum size of a minimal separator and treewidth. That is, for treewidth as the parameter, we consider the fixed parameter tractability of MAXIMUM WEIGHT MINIMAL SEPARATOR.

4.2.3 Comparison with Max Cut

The MAX CUT problem is a classical graph problem where, given an undirected and edge-weighted graph $G = (V, E)$, we have to find a set $S \subseteq V$ that maximizes $\sum_{u \in S, v \in V \setminus S} w_{uv}$, where w_{uv} is the weight of edge (u, v). Both MAX CUT and MAXIMUM WEIGHT MINIMAL SEPARATOR are in some sense connectivity problems. However, in the former problem the value of a solution is based on edge weights, whereas in the latter problem it is given by vertex weights. As such, the problems can behave quite differently: It is known that MAX CUT is NP-hard on chordal graphs [9], but on bipartite graphs, it is trivial. In contrast, MAXIMUM WEIGHT MINIMAL SEPARATOR is NP-hard on bipartite graphs but can be solved in polynomial time on chordal graphs.

4.3 Basic results

In this section, we give two basic results for MAXIMUM WEIGHT MINIMAL SEPARATOR. On the negative side, we show that MAXIMUM WEIGHT MINIMAL SEPARATOR and MAXIMUM WEIGHT MINIMAL s-t SEPARATOR are NP-hard. On the other hand, we show that it is fixed-parameter tractable with respect to W.

4.3.1 NP-hardness

Theorem 4.3.1. MAXIMUM WEIGHT MINIMAL s-t SEPARATOR is NP-hard on bipartite graphs even if all the vertex weights are identical.

Proof: We give a reduction from a well-known NP-hard problem, MAX CUT ([42]), which, given an unweighted graph $G = (V, E)$, asks whether there exists a cut $(C, V \setminus C)$ whose value $(|\{(u, v) \in E \mid u \in C, v \in V \setminus C\}|)$ is at least k.

We construct an instance (G', p) of MAXIMUM WEIGHT MINIMAL s-t SEPARATOR from $G = (V, E)$ and positive integer k. Although the instance we construct is weighted, the proof

can easily be modified to the unweighted case.

We build the vertex set of G' by taking the union of 3 copies (V, V', V'') of the vertex set V of G, where we write v' (resp., v'') to denote the vertex corresponding to the copy of $v \in V$ in V' (resp., V''). Furthermore, for each edge e in the edge set E of G, we create a corresponding vertex e in G'. Finally, we also create two vertices s and t.

We build the edge set of G' by making every vertex of V' adjacent to s, and every vertex of V'' adjacent to t. We make every vertex $v \in V$ adjacent to its corresponding copies v' (in V') and v'' (in V''). Finally, for each edge vertex e (corresponding to some edge $e = (u, v)$ in G), we take the edges (e, u) and (e, v).

Formally, let $G' = (V \cup E \cup V' \cup V'' \cup \{s, t\}, E_1 \cup E_2)$, where $V' = \{v' \mid v \in V\}$, $V'' = \{v'' \mid v \in V\}$, $E_1 = \bigcup_{e=(u,v)\in E}\{(u, e), (v, e)\}$ and $E_2 = \bigcup_{u\in V}\{(s, u'), (u', u)\} \cup \bigcup_{u\in V}\{(t, u''), (u'', u)\}$.

The vertex weights of G' are defined to be $w_v = 3n + 1$ if $v \in E$ and 1 otherwise (where $n = |V|$). G' can be easily seen to be bipartite, by partitioning the vertices into sets $\{s, t\} \cup V$ and $V' \cup V'' \cup E$.

We now show that if G has a cut C of weight at least k, then G' has a minimal s-t separator S whose weight is at least $p = (3n + 1)k$.

Given C, we construct S by taking the union of the vertices in V'' that correspond to some vertex in C, the vertices in V' that correspond to some vertex not in C, and the vertices corresponding to edges bisected by C. Formally, let $S = \{u'' \in V'' \mid u \in V \cap C\} \cup \{v' \in V' \mid v \in V \setminus C\} \cup \{e = (u, v) \in E \mid u \in C, v \in V \setminus C\}$.

S is a s-t separator: it is easy to see that the vertices reachable from s (after removing S) are precisely the vertices in V and V' that correspond to vertices in C, together with vertices corresponding to edges between vertices in C. On the other hand, the vertices reachable from t are precisely the vertices in V and V'' that correspond to vertices *not* in C, together with vertices corresponding to edges between vertices *not* in C.

Furthermore, S is minimal, since removing from S any vertex e corresponding to an edge $e = (u, v)$ gives rise to an s-t path s, u', u, e, v, v'', t (if $u \in C, v \notin C$) or s, v', v, e, u, u'', t (if $u \notin C, v \in C$). Removing from S any vertex v' or v'' gives rise to an s-t path s, v', v, v'', t.

The weight of S is at least $(3n + 1)k$ since $|\{e = (u, v) \in E \mid u \in C, v \in V \setminus C\}| \geq k$ (i.e.,

since C is a cut that bisects at least k edges, our separator S includes at least k edge vertices).

We next show that if G' has a minimal s-t separator S whose weight is at least $p = (3n+1)k$, then G has a cut C of weight at least k. By the weighting, S contains at least k vertices in E. Note that for any $v \in V(G)$, at least one of v, v', v'' is included in S, otherwise S does not separate s and t. If S does not contain any $v \in V$, let C be vertices in V that are reachable from s after removing S; C is actually a cut, and its weight is k. Otherwise, S contains a vertex $v \in V$. In this case, S does not contain any e forming $e = (v, x)$ because otherwise it contradicts the minimality. Then, we construct $S' := S \setminus \{v\} \cup \{v'\} \cup \{e = (v, x) \in E\}$ - obtaining a minimal separator of greater weight. By repeating this procedure, we obtain a minimal separator S of weight at least $(3n+1)k$ that does not contain any $v \in V$. This completes the correctness of the reduction.

As mentioned above, this reduction can be modified to the unweighted case. To this end, we create $3n+1$ identical copies of each edge vertex e (and of its incident edges). In the new reduction, to block the path between u and v, we need to remove $3n+1$ copies of e $(= (u, v))$, which plays the same role as the original vertex having weight $3n+1$. ■

Next, we show how to adapt this proof to the case of MAXIMUM WEIGHT MINIMAL SEPARATOR, which does not require the separator to separate s and t (but rather only requires that the separator separates *some* pair of vertices).

Corollary 4.3.2. MAXIMUM WEIGHT MINIMAL SEPARATOR is NP-hard on bipartite graphs, even if all the vertex weights are identical.

Proof: We give a reduction from MAXIMUM WEIGHT MINIMAL s-t SEPARATOR. Given an instance $(G = (V, E), p, s, t, w)$ of MAXIMUM WEIGHT MINIMAL s-t SEPARATOR, we add an additional vertex x which we make adjacent to both s and t and we give x weight $w(x) = \Sigma_{v \in V} w(v) + 1$. We ask whether there exists a minimal separator of weight at least $w(x) + p$. If S is a separator of weight at least $w(x) + p$, it must necessarily include x, and thus, be an s-t separator. It then follows that $S \setminus \{x\}$ must be an s-t separator of weight at least p in G. The converse easily follows: if S is an s-t separator in G of weight at least p, then $S \cup \{x\}$ is a separator of weight at least $p + w(x)$ in the modified graph.

The proof for the unweighted case follows similarly to above, by creating $w(x)$ identical copies of x (and noting that our hardness proof for MAXIMUM WEIGHT MINIMAL s-t SEPARATOR uses

polynomial weights). ■

4.3.2 Fixed-parameter tractability

As a positive result, we show that MAXIMUM WEIGHT MINIMAL SEPARATOR is fixed-parameter tractable with respect to the solution size k.

Theorem 4.3.3. For unweighted graphs G, MAXIMUM WEIGHT MINIMAL SEPARATOR is fixed-parameter tractable with respect to the solution size k.

Proof: We first determine whether G contains $k \times k$ grid minor or not in $f(k)n^2$-time [60]. If G has an $k \times k$ grid minor, then G also has a minimal separator of size at least k. Otherwise, the treewidth of G is at most $g(k)$ by the excluded grid theorem [16,97]. Therefore, we can use dynamic programming on a tree decomposition of width bounded by a function of k. As the MAXIMUM WEIGHT MINIMAL SEPARATOR problem can be formulated in Monadic Second Order Logic, we can solve the problem in linear time for fixed k. In this chapter, we give more efficient dynamic programming algorithms. ■

For vertex-weighted graphs, MAXIMUM WEIGHT MINIMAL SEPARATOR is also fixed-parameter tractable with respect to W.

Corollary 4.3.4. For vertex-weighted graphs G, MAXIMUM WEIGHT MINIMAL SEPARATOR is fixed-parameter tractable with respect to W.

4.4 Dynamic programming on tree decompositions

In this section, we give an FPT algorithm with respect to treewidth. It is a standard dynamic programming algorithm based on tree decompositions, and given a tree decomposition of width at most ω, the running time is $\omega^{O(\omega)}n$.

To be able to use "standard" algorithmic techniques, we reformulate the MAXIMUM WEIGHT MINIMAL s-t SEPARATOR as a connectivity problem. In doing so, we can use the Cut & Count technique [19, 20]. We start with defining the notion of connected partition:

Definition 4.4.1. A *connected partition* of weight W is a partition (S, A, B, Q) of V such that: (1) $s \in A, t \in B$, (2) $G[A]$ is connected, (3) $G[B]$ is connected, (4) $\sum_{v \in S} w(v) = W$, (5) for $\forall v \in S$,

there exist vertices $a \in A, b \in B$ such that $(a, v) \in E, (v, b) \in E$ and (6) for sets A, B, Q, there does not exist an edge (u, v) such that u and v are in different sets.

The key to the design of our algorithms is the following lemma, which states that connected partitions correspond to minimal separators.

Theorem 4.4.2. There exists a minimal s-t separator of weight W if and only if there exists a connected partition (S, A, B, Q) of weight W.

To prove Theorem 4.4.2, we use the following lemma. This lemma appears in many papers and books, for example, as an exercise in [45].

Lemma 4.4.3 ([45])**.** Let S be a minimal s-t separator and A, B be the connected components of $G[V \setminus S]$ containing s and t, respectively. Then every vertex of S has a neighbor in A and a neighbor in B.

Using this lemma, we can prove Theorem 4.4.2.

Proof: [Theorem 4.4.2] ($\Rightarrow$) Let S be a minimal s-t separator of weight W. Let A be the subset of vertices of $V \setminus S$ such that $G[A]$ is the connected component containing s and B be the subset of vertices such that $G[B]$ is the connected component containing t. Moreover, let $Q = V \setminus A \setminus B \setminus S$. Note that $S \cap A \cap B \cap Q = \emptyset$ and that there is no edge between A, B and Q. By Lemma 4.4.3, all vertices in S have neighbors in both A and B. Therefore, (S, A, B, Q) is a connected partition of weight W.

($\Leftarrow$) Suppose that (S, A, B, Q) is a connected partition of weight W. We claim that S is a minimal s-t separator. To show this by contradiction, suppose that there exists a vertex $v \in S$ such that $S \setminus \{v\}$ separates s and t. Then there exist vertices $a \in A, b \in B$ such that $(a, v) \in E, (v, b) \in E$. Since $G[A]$ is connected, there exists a path (in $G[A]$) from s to a. Similarly, there exists a path (in $G[B]$) from b to t. By joining these paths with the edges (a, v) and (v, b), we obtain a path from s to t, which contradicts that $S \setminus \{v\}$ is a separator. Hence S is a minimal s-t separator, and by definition it is of weight W. ■

Using connected partitions, we design an $\omega^{O(\omega)}n$-time algorithm for MAXIMUM WEIGHT MINIMAL s-t SEPARATOR. First, we partition S into $S_\emptyset, S_A, S_B$ and S_{AB} (see Figure 4.1). Set $S_\emptyset$ consists of the vertices in S that have no neighbor in A and B, but may have neighbors in S_A, S_B, S_{AB}

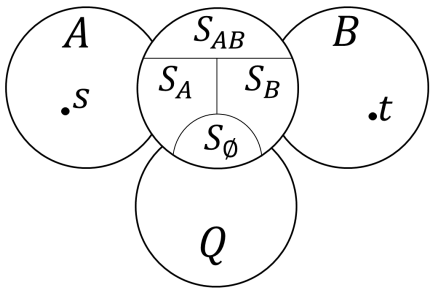

Figure 4.1. Connection between vertex sets

or Q. Set S_A (resp., S_B) consists of the vertices in S that have at least one neighbor in A (resp., B), but no neighbor in B (resp., A). They may have neighbors in S_A, S_B, S_{AB}, Q. Set S_{AB} consists of the vertices in S that have neighbors in A and in B and may have neighbors in S_A, S_B, S_{AB}, Q.

Since we eventually want the sets A and B to become connected, we need to track their connectivity. We define two partitions $\mathcal{P}^A = \{P_1^A, P_2^A, \ldots, P_\alpha^A\}$ of $X_i \cap A$ and $\mathcal{P}^B = \{P_1^B, P_2^B, \ldots, P_\beta^B\}$ of $X_i \cap B$. We call each element of a partition P_ℓ^A (resp., P_ℓ^B) a *block*. They correspond to the intersection of X_i and the vertex sets of connected components of $G_i^A = (A \cap V_i, E_i)$ (resp., $G_i^B = (B \cap V_i, E_i)$). Note that there are at most $|X_i|^{O(|X_i|)}$ partitions for each node X_i. α and β are the number of connected components in $G_i^A = (A \cap V_i, E_i)$ and $G_i^B = (B \cap V_i, E_i)$, respectively. Note that $\alpha \leq |X_i \cap A|$ and $\beta \leq |X_i \cap B|$. Intuitively, one block $\{\{v\}\}$ is added to $\mathcal{P}^A$ in each introduce vertex v node; then blocks may be merged in introduce edge nodes and join nodes. In a forget node, note that a vertex may not be the last of its block, else, the solution is invalid and must be discarded since its component of A is not connected.

With these sets, we define a *partial solution* as follows.

Definition 4.4.4. Given a node i of the tree decomposition of G, a *partial solution* for node i is a partition $(S_\emptyset, S_A, S_B, S_{AB}, A, B, Q)$ of V_i with $\mathcal{P}^A$ and $\mathcal{P}^B$, such that:

- $S_\emptyset \cup S_A \cup S_A \cup S_{AB} \cup A \cup B \cup Q = V_i$,
- $\forall v \in S_\emptyset$, $N(v) \cap (A \cup B) = \emptyset$,
- $\forall v \in S_A$, $N(v) \cap B = \emptyset$ and $N(v) \cap A \neq \emptyset$,
- $\forall v \in S_B$, $N(v) \cap B \neq \emptyset$ and $N(v) \cap A = \emptyset$,
- $\forall v \in S_{AB}$, $N(v) \cap B \neq \emptyset$ and $N(v) \cap A \neq \emptyset$,

- $\forall v_1, v_2 \in A$, v_1, v_2 are in the same block in $\mathcal{P}^A \leftrightarrow v_1, v_2$ are connected in G_i^A,
- $\forall v_1, v_2 \in B$, v_1, v_2 are in the same block in $\mathcal{P}^B \leftrightarrow v_1, v_2$ are connected in G_i^B,
- $s \in V_i \Rightarrow s \in A$, and
- $t \in V_i \Rightarrow t \in B$.

Let $\Sigma = \{s_\emptyset, s_A, s_B, s_{AB}, a, b, q\}$. Each element of Σ is called *state* of vertices (e.g., we say that vertex $v \in S_\emptyset$ has state $s_\emptyset$). Then, we define the *coloring* function $c : V \to \Sigma$. The coloring function represents which set of the partition a vertex is in, for example, if v is in $S_\emptyset$ then $c(v) = s_\emptyset$.

Given two sets V and W, we denote their colorings by $c_V \in \Sigma^{|V|}$ and $c_W \in \Sigma^{|W|}$, respectively. Suppose that V and W are disjoint, $c_V = (c(v_1), \ldots, c(v_{|V|}))$, and $c_W = (c(w_1), \ldots, c(w_{|W|}))$, where $v_i \in V$ and $w_i \in W$. We then define the *concatenation* $c_V \times c_W \in \Sigma^{|V|+|W|}$ of c_V and c_W as the coloring $(c(v_1), \ldots, c(v_{|V|}), c(w_1), \ldots, c(w_{|W|}))$. Moreover, if $W \subseteq V$. then we define the *separation* $c_V \setminus c_W \in \Sigma^{|V|-|W|}$ as the coloring $(c(v_{i_1}), \ldots, c(v_{i_{|V|-|W|}}))$, where $v_{i_1}, \ldots, v_{i_{|V|-|W|}} \in V \setminus W$.

In our algorithm, we assume we are given a nice tree decomposition of width at most ω. We transform this tree decomposition by adding $\{s, t\}$ to all bags; thus we can suppose that the root bag X_r contains exactly two vertices s, t. The width of this tree decomposition is at most $\omega + 2$.

We define the function $f_i(c, \mathcal{P}^A, \mathcal{P}^B)$ to be the possible maximum weight of vertices in $S \cap V_i$ of a partial solution $(S_\emptyset, S_A, S_B, S_{AB}, A, B, Q)$ of V_i with $\mathcal{P}_A$ and $\mathcal{P}_B$. If $c, \mathcal{P}^A, \mathcal{P}^B$ deviate from the definition of a partial solution, then let $f_i(c, \mathcal{P}^A, \mathcal{P}^B) = -\infty$.

We now give recursive formulas for computing f_i in each node i. In the root node, $f_r(\{c(s)\} \times \{c(t)\}, \{\{s\}\}, \{\{t\}\}) = f_r(\{a\} \times \{b\}, \{\{s\}\}, \{\{t\}\})$ is the maximum weight of minimal s-t separators because $X_r = \{s, t\}$.

In the following, we let i denote a parent node, and let j denote its corresponding child node. For a join node, we write j_1 and j_2 to denote its two children. To emphasize that we are dealing with two different colorings, we denote parent node colorings by c_i and child node colorings by c_j.

Leaf node: In leaf nodes, if $c(s) = a$ and $c(t) = b$, we define $f_i(\{c(s)\} \times \{c(t)\}, \{\{s\}\}, \{\{t\}\})$ $:= 0$. Otherwise, $f_i(\{c(s)\} \times \{c(t)\}, \mathcal{P}^A, \mathcal{P}^B) := -\infty$ since there are only two vertices s, t in X_i.

Introduce vertex v node: In introduce vertex nodes, we consider three cases for colorings. If $c(v) = s_\emptyset$, we add $w(v)$ to $f_j(c, \mathcal{P}^A, \mathcal{P}^B)$ because v is added in S. If $c(v) \in \{a, b, q\}$, the value of f_i does not change since $v \notin S$. Moreover, we add a block $\{\{v\}\}$ to $\mathcal{P}^A$ or $\mathcal{P}^B$ depending on whether $c(v) = a$ or $c(v) = b$, respectively. Finally, if $c(v) \in \{s_A, s_B, s_{AB}\}$, a partial solution is invalid by the definition because v has no incident edge and hence no neighbor in A or B. Therefore, we define f_i as follows:

$$f_i(c, \mathcal{P}^A, \mathcal{P}^B) := \begin{cases} f_j(c \setminus \{c(v)\}, \mathcal{P}^A, \mathcal{P}^B) + w(v) & \text{if } c(v) = s_\emptyset \\ f_j(c \setminus \{c(v)\}, \mathcal{P}^A \setminus \{\{v\}\}, \mathcal{P}^B) & \text{if } c(v) = a \\ f_j(c \setminus \{c(v)\}, \mathcal{P}^A, \mathcal{P}^B \setminus \{\{v\}\}) & \text{if } c(v) = b \\ f_j(c \setminus \{c(v)\}, \mathcal{P}^A, \mathcal{P}^B) & \text{if } c(v) = q \\ -\infty & \text{otherwise.} \end{cases}$$

Introduce edge (u, v) node: In introduce edge nodes, we define f_i for the following cases of $c(u)$, $c(v)$.

- If $c(u) = a$ and $c(v) = a$, the vertices u, v are in A. If u and v are in the different blocks, we set $f_i(c, \mathcal{P}^A, \mathcal{P}^B) := -\infty$ because u and v are in the same block of partition $\mathcal{P}^A$ in node i due to edge (u, v). Then, there are two cases: the partitions in $A \cap X_i$ (parent) and $A \cap X_j$ (child) are same or not. In the former case, u and v are in the same block in the partition of $A \cap X_j$, and we then set $f_i(c, \mathcal{P}^A, \mathcal{P}^B) := f_j(c, \mathcal{P}^A, \mathcal{P}^B)$. In the latter case, let $\mathcal{P}'^A$ be a partition of $A \cap X_j$ such that $\mathcal{P}^A \neq \mathcal{P}'^A$ but $\mathcal{P}'^A$ changes to $\mathcal{P}^A$ by merging two blocks of $\mathcal{P}'^A$ including u and v respectively with edge (u, v). Therefore, we take a $\mathcal{P}'^A$ that maximizes $f_i(c, \mathcal{P}'^A, \mathcal{P}^B)$. Then, we set f_i as follows:

$$f_i(c, \mathcal{P}^A, \mathcal{P}^B) := \max\{f_j(c, \mathcal{P}^A, \mathcal{P}^B), \max_{\mathcal{P}'^A} f_j(c, \mathcal{P}'^A, \mathcal{P}^B)\}.$$

- The case that $c(u) = b$ and $c(v) = b$ is almost the same as the case that $c(u) = a$ and $c(v) = a$. If u and v are not in the same block of partition $\mathcal{P}^B$, we then set $f_i(c, \mathcal{P}^A, \mathcal{P}^B) := -\infty$. Let $\mathcal{P}'^B$ be a partition of $B \cap X_j$ such that $\mathcal{P}^B \neq \mathcal{P}'^B$ but $\mathcal{P}'^B$ changes to $\mathcal{P}^B$ by merging two blocks of $\mathcal{P}'^B$ including u and v respectively with edge (u, v).

Then, we define f_i as follows:

$$f_i(c, \mathcal{P}^A, \mathcal{P}^B) := \max\{f_j(c, \mathcal{P}^A, \mathcal{P}^B), \max_{\mathcal{P}'^B} f_j(c, \mathcal{P}^A, \mathcal{P}'^B)\}.$$

- If $c(u), c(v) \in \{s_\emptyset, s_A, s_B, s_{AB}, q\}$, we define f_i as follows:

$$f_i(c, \mathcal{P}^A, \mathcal{P}^B) = f_j(c, \mathcal{P}^A, \mathcal{P}^B).$$

  In this case, (u, v) is irrelevant to the partitions and the value is not changed because only one edge (u, v) is added.

- If $(c(u), c(v)) = (s_A, a), (a, s_A)$, we consider two cases. One case is that $u \in S_A$ and $v \in A$ in the child node and the other case is that $u \in S_\emptyset$ and $v \in A$ in the child node. In the other case, u is moved from $S_\emptyset$ into S_A by adding (u, v), because u has a neighbor v in A. Thus, we define f_i as follows:

$$f_i(c \times \{s_A\} \times \{a\}, \mathcal{P}^A, \mathcal{P}^B) := \max\{\ f_j(c \times \{s_A\} \times \{a\}, \mathcal{P}^A, \mathcal{P}^B), \\ f_j(c \times \{s_\emptyset\} \times \{a\}, \mathcal{P}^A, \mathcal{P}^B)\}.$$

- If $(c(u), c(v)) = (s_B, b), (b, s_B)$, we consider almost the same cases as above; that is, $u \in S_B$, $v \in B$ and $u \in S_\emptyset$ and $v \in B$ in the child node.

$$f_i(c \times \{s_B\} \times \{b\}, \mathcal{P}^A, \mathcal{P}^B) := \max\{\ f_j(c \times \{s_B\} \times \{b\}, \mathcal{P}^A, \mathcal{P}^B), \\ f_j(c \times \{s_\emptyset\} \times \{b\}, \mathcal{P}^A, \mathcal{P}^B)\}.$$

- If $(c(u), c(v)) = (s_{AB}, a), (a, s_{AB})$, there are two cases: (1) $u \in S_{AB}$ and $v \in A$ in the child node and (2) $u \in S_B$ and $v \in A$ in the child node. In the latter case, u is moved from S_B to S_{AB} by adding (u, v), because u has a neighbor v in B. Therefore, we define f_i as follows:

$$f_i(c \times \{s_{AB}\} \times \{a\}, \mathcal{P}^A, \mathcal{P}^B) := \max\{\ f_j(c \times \{s_{AB}\} \times \{a\}, \mathcal{P}^A, \mathcal{P}^B), \\ f_j(c \times \{s_B\} \times \{a\}, \mathcal{P}^A, \mathcal{P}^B)\}.$$

- If $(c(u), c(v)) = (s_{AB}, b), (b, s_{AB})$, we consider almost the same cases as above; that is, $u \in S_{AB}$, $v \in B$ and $u \in S_A$ and $v \in B$ in the child node.

$$f_i(c \times \{s_{AB}\} \times \{b\}, \mathcal{P}^A, \mathcal{P}^B) := \max \{ f_j(c \times \{s_{AB}\} \times \{b\}, \mathcal{P}^A, \mathcal{P}^B), \\ f_j(c \times \{s_A\} \times \{b\}, \mathcal{P}^A, \mathcal{P}^B)\}.$$

- Otherwise, we set $f_i(c, \mathcal{P}^A, \mathcal{P}^B) := -\infty$ because the rest of the cases are invalid by the definition of states and partial solution. There must not exist edge (u, v) for such cases.

Forget v node: In a forget v node, if $c_j(v) \in \{s_\emptyset, s_A, s_B\}$, vertex v will never have neighbors both in A and in B and hence this case is invalid because of the definition of the connected partition, which requires that each vertex in S has neighbors in both A and B. If $c_j(v) \in \{s_{AB}, q\}$, we need not consider the connectivity of v. In the case that $c_j(v) = a$, we only consider partitions such that there exists at least one vertex u in A included in the same block as v. If there is no such vertex, the block including v is never merged. Consequently, $G[A]$ would not be connected in the root node. The case that $c_j(v) = b$ is almost the same. Let $\mathcal{P}'^A, \mathcal{P}'^B$ be a partition satisfying such conditions, then we define f_i as follows:

$$f_i(c, \mathcal{P}^A, \mathcal{P}^B) := \max \{ f_j(c \times \{s_{AB}\}, \mathcal{P}^A, \mathcal{P}^B), f_j(c \times \{q\}, \mathcal{P}^A, \mathcal{P}^B), \\ \max_{\mathcal{P}'^A} f_j(c \times \{a\}, \mathcal{P}'^A, \mathcal{P}^B), \max_{\mathcal{P}'^B} f_j(c \times \{b\}, \mathcal{P}^A, \mathcal{P}'^B)\}.$$

Join node: For a parent node i and two child nodes j_1, j_2, we denote the corresponding colorings by c_i, c_{j_1}, c_{j_2} and the corresponding partitions by $\mathcal{P}^A_{j_1}, \mathcal{P}^B_{j_1}, \mathcal{P}^A_{j_2}, \mathcal{P}^B_{j_2}$. We then define a subset D of tuples of $((c_{j_1}, \mathcal{P}^A_{j_1}, \mathcal{P}^B_{j_1}), (c_{j_2}, \mathcal{P}^A_{j_2}, \mathcal{P}^B_{j_2}))$ such that the combinations of colorings for c_{j_1}, c_{j_2} satisfy the following conditions (see Table 4.1):

- $\forall v \in c_i^{-1}(\{s_\emptyset, a, b, q\}), (c_{j_1}(v), c_{j_2}(v)) = (c_i(v), c_i(v))$,
- $\forall v \in c_i^{-1}(\{s_A\}), (c_{j_1}(v), c_{j_2}(v)) = (s_A, s_\emptyset), (s_\emptyset, s_A), (s_A, s_A)$,
- $\forall v \in c_i^{-1}(\{s_B\}), (c_{j_1}(v), c_{j_2}(v)) = (s_B, s_\emptyset), (s_\emptyset, s_B), (s_B, s_B)$, and
- $\forall v \in c_i^{-1}(\{s_{AB}\}), (c_{j_1}(v), c_{j_2}(v)) = (s_{AB}, s_\emptyset), (s_{AB}, s_A), (s_{AB}, s_B), (s_{AB}, s_{AB})$,

	$s_\emptyset$	s_A	s_B	s_{AB}	a	b	q
$s_\emptyset$	$s_\emptyset$	s_A	s_B	s_{AB}			
s_A	s_A	s_A	s_{AB}	s_{AB}			
s_B	s_B	s_{AB}	s_B	s_{AB}			
s_{AB}	s_{AB}	s_{AB}	s_{AB}	s_{AB}			
a					a		
b						b	
q							q

Table 4.1. This table represents combinations of states of two child nodes j_1, j_2 for each vertex in $X_i = X_{j_1} = X_{j_2}$. The rows and columns correspond to states of j_1, j_2 respectively and inner elements correspond to states of x. For example, if $c_i(v) = s_A$, there are three combinations such that $(c_{j_1}(v), c_{j_2}(v)) = (s_A, s_\emptyset)$, $(c_{j_1}(v), c_{j_2}(v)) = (s_\emptyset, s_A)$ and $(c_{j_1}(v), c_{j_2}(v)) = (s_A, s_A)$.

$$(s_\emptyset, s_{AB}), (s_A, s_{AB}), (s_B, s_{AB}), (s_A, s_B), (s_B, s_A),$$

and the partition obtained by merging $\mathcal{P}^A_{j_1}$ and $\mathcal{P}^A_{j_2}$ equals $\mathcal{P}^A$ and the partition obtained by merging $\mathcal{P}^B_{j_1}, \mathcal{P}^B_{j_2}$ equals $\mathcal{P}^B$. If $D = \emptyset$ for $c_i, \mathcal{P}^A, \mathcal{P}^B$, we set $f_i(c_i, \mathcal{P}^A, \mathcal{P}^B) := -\infty$. Otherwise, we set $S^* := c_i^{-1}(\{s_\emptyset, s_A, s_B, s_{AB}\})$. Then we define f_i as follows:

$$f_i(c_i, \mathcal{P}^A, \mathcal{P}^B) := \max_{((c_{j_1}, \mathcal{P}^A_{j_1}, \mathcal{P}^B_{j_1}), (c_{j_2}, \mathcal{P}^A_{j_2}, \mathcal{P}^B_{j_2})) \in D} \{ f_{j_1}(c_{j_1}, \mathcal{P}^A_{j_1}, \mathcal{P}^B_{j_1}) + f_{j_2}(c_{j_2}, \mathcal{P}^A_{j_2}, \mathcal{P}^B_{j_2}) - w(S^*) \}.$$

The subtraction in the right hand side of the equation above is because the weight $w(S^*)$ is counted twice; once in each child node.

We recursively calculate f_i on the tree decomposition. Note that all bags have $|X_i|$ vertices and the number of combinations of colorings and partitions $(c, \mathcal{P}^A, \mathcal{P}^B)$ in each node is $|X_i|^{O(|X_i|)} = \omega^{O(\omega)}$. The running time to compute all f_i's in X_i is dominated by join nodes and it is roughly $(\omega^{O(\omega)})^3 = \omega^{O(\omega)}$ since we scan every coloring and partition in two children nodes X_{j_1} and X_{j_2} for each coloring c_i and each partition $\mathcal{P}^A, \mathcal{P}^B$ and then check all combinations. Therefore, the total running time is $\omega^{O(\omega)}n$ and we conclude with the following theorem.

Theorem 4.4.5. Given a tree decomposition of width at most ω, there exists an algorithm that solves MAXIMUM WEIGHT MINIMAL s-t SEPARATOR in time $\omega^{O(\omega)}n$.

MAXIMUM WEIGHT MINIMAL SEPARATOR can be solved by applying the above $\omega^{O(\omega)}n$-

time algorithm for all possible combinations of s and t, i.e., at most n^2 times. We thus obtain the following result:

Theorem 4.4.6. Given a tree decomposition of width at most ω, there exists an algorithm that solves MAXIMUM WEIGHT MINIMAL s-t SEPARATOR in time $\omega^{O(\omega)}n^3$.

4.5 Algorithm using Cut & Count

In this section, we give a Monte-Carlo algorithm that solves the decision version of MAXIMUM WEIGHT MINIMAL s-t SEPARATOR, that is, to decide whether there exists a minimal s-t separator with weight W in time $O^*(9^\omega \cdot W^2)$, where ω is the width of a tree decomposition. This algorithm is based on the Cut & Count technique.

4.5.1 Isolation lemma

In this subsection, we explain the isolation lemma introduced by Mulmuley et al. [80]. The main idea of the Cut & Count technique is to obtain a single solution with high probability; we count modulo 2, and the isolation lemma guarantees the existence of such a single solution.

Definition 4.5.1 ([80]). A function $w' : U \to \mathbb{Z}$ *isolates* a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $w'(S') = \min_{S \in \mathcal{F}} w'(S)$ where $w'(X) = \sum_{u \in X} w'(u)$.

Lemma 4.5.2 (Isolation lemma [80]). Let $F \subseteq 2^U$ be a set family over a universe U with $|F| > 0$. For each $u \in U$, choose a weight $w'(u) \in \{1, 2, \ldots N\}$ uniformly and independently at random. Then $Pr[w' \text{ isolates } \mathcal{F}] \geq 1 - |U|/N$.

4.5.2 Cut & Count

The Cut & Count technique was introduced by Cygan et al. for solving connectivity problems [20]. The concept of Cut & Count is counting the number of relaxed solutions, that is, we do not consider whether they are connected or disconnected. Then we compute the number of relaxed solutions modulo 2 and we determine whether there exists a connected solution by cancellation tricks. Now, we define a *consistent cut* to explain the details of Cut & Count.

Definition 4.5.3 ([20]). A cut (V_1, V_2) of $V' \subseteq V$ such that $V_1 \cup V_2 = V'$ and $V_1 \cap V_2 = \emptyset$ is

consistent if $v_1 \in V_1$ and $v_2 \in V_2$ implies $(v_1, v_2) \notin E$.

This means that a cut (V_1, V_2) of V' is consistent if there are no edges between V_1 and V_2. We fix an arbitrary vertex v in V_1. Then, if $G[V]$ is connected, then there only exists one consistent cut, namely $(V_1, V_2) = (V, \emptyset)$. Therefore, the number of consistent cuts is odd. Otherwise, if V does not induce a connected subgraph, the number of consistent cuts is a multiple of two. Therefore, we just need to compute the number of consistent cuts modulo 2 and return yes if the number of consistent cuts is odd, and return no otherwise. the isolation lemma allows us to guarantee (with high probability) that there exists a unique solution (and thus, that we indeed end up with an odd number of consistent cuts).

Let $\mathcal{S} \subseteq 2^U$ be a set of solutions. According to [19, 20], the Cut & Count technique is divided into two parts as follows.

- **The Cut part** : Relax the connectivity requirement by considering the set $\mathcal{R} \supseteq \mathcal{S}$ of possibly connected or disconnected candidate solutions. Moreover, consider the set $\mathcal{C}$ of pairs $(X; C)$ where $X \in \mathcal{R}$ and C is a consistent cut of X.

- **The Count part** : Isolate a single solution by sampling weights of all elements in U by the isolation lemma. Then, compute $|\mathcal{C}|$ modulo 2 using a sub-procedure. Disconnected candidate solutions $X \in \mathcal{R} \setminus S$ cancel since they are consistent with an even number of cuts. If the only connected candidate $x \in \mathcal{S}$ exists, we obtain the odd number of cuts.

Given a set U and a tree decomposition $\langle \mathcal{X}, T \rangle$, the general scheme of Cut & Count is as follows:

Step 1. Set the integer weight for every vertex uniformly and independently at random by $w' : U \to \{1, \ldots, 2|U|\}$.

Step 2. For each integer weight $0 \leq W' \leq 2|U|^2$, compute the number of relaxed solutions of weight W' with consistent cuts modulo 2 on a tree decomposition. Then return yes if it is odd, otherwise no in the root node.

We use the Cut & Count technique to determine whether there exists a connected partition (S, A, B, Q) of weight W so that A and B are connected. To apply the above scheme, we newly give the following definition of a *partial solution*. Note that we have to consider two consistent cuts of A and B.

Definition 4.5.4. Given a node i of the tree decomposition of G, a *partial solution* for that node is a tuple $(S_\emptyset, S_A, S_B, S_{AB}, A_l, A_r, B_l, B_r, Q, w)$, such that:

- $V_i = S_\emptyset \cup S_A \cup S_A \cup S_{AB} \cup A_l \cup A_r \cup B_l \cup B_r \cup Q$,
- (A_l, A_r) is a consistent cut: there exists no edge $(u, v) \in E$ such that $u \in A_l$ and $v \in A_r$,
- (B_l, B_r) is a consistent cut: there exists no edge $(u, v) \in E$ such that $u \in B_l$ and $v \in B_r$,
- $w = \Sigma_{v \in S} w(v)$,
- $\forall v \in S_\emptyset$, $N(v) \cap (A_l \cup A_r \cup B_l \cup B_r) = \emptyset$,
- $\forall v \in S_A$, $N(v) \cap (B_l \cup B_r) = \emptyset$ and $N(v) \cap (A_l \cup A_r) \neq \emptyset$,
- $\forall v \in S_B$, $N(v) \cap (B_l \cup B_r) \neq \emptyset$ and $N(v) \cap (A_l \cup A_r) = \emptyset$ and
- $\forall v \in S_{AB}$, $N(v) \cap (B_l \cup B_r) \neq \emptyset$ and $N(v) \cap (A_l \cup A_r) \neq \emptyset$.
- $s \in V_i \Rightarrow s \in A_l$
- $t \in V_i \Rightarrow t \in B_l$

For each vertex v, we set another weight $w'(v)$ by choosing from $\{1, \ldots, 2|V|\}$ independently at random. We also define the coloring function $c : V \to \{s_\emptyset, s_A, s_B, s_{AB}, a_l, a_r, b_l, b_r, q\}$. Now, we give a dynamic programming algorithm that computes the number of relaxed solutions with consistent cuts modulo 2. To compute that, for each c, w and w', we define the *counting* function $h_i : \{s_\emptyset, s_A, s_B, s_{AB}, a_l, a_r, b_l, b_r, q\}^{|X_i|} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ in each node i on a nice tree decomposition. Here, we define the function $[p]$ as follows: if p is true, then $[p] = 1$, otherwise $[p] = 0$.

Leaf node: In a leaf node, we define $h_i(\emptyset, 0, 0) = 1$, if $S_\emptyset = S_A = S_B = S_{AB} = A_l = A_r = B_l = B_r = \emptyset$ and $w, w' = 0$. Otherwise, $h_i(c, w, w') = 0$.

Introduce vertex v node: The function h_i has five cases in introduce vertex nodes. Note that we only add one vertex v without edges. Thus, if $c(v) \in \{s_A, s_B, s_{AB}\}$, the partial solution is invalid by definition because v has no neighbor. If $c(v) = s_\emptyset$, vertex v is chosen as a vertex of S, and we hence add each weight $w(v)$, $w'(v)$ to w, w', respectively. Moreover, v must not be s, t because s

(resp., t) should be in A_l (resp., B_l). If not, it is not a connected partition. Similarly, if $c(v) = a_l$ (resp., b_l), we check whether v is not t (resp., s). As for $c(v) \in \{a_r, b_r, q\}$, we also check whether v is neither s nor t. Therefore, we define h_i in introduce vertex nodes as follows:

$$h_i(c \times \{c(v)\}, w, w') := \begin{cases} [v \neq s, t]h_j(c, w - w(v), w' - w'(v)) & \text{if } c(v) = s_\emptyset \\ [v \neq t]h_j(c, w, w') & \text{if } c(v) = a_l \\ [v \neq s]h_j(c, w, w') & \text{if } c(v) = b_l \\ [v \neq s, t]h_j(c, w, w') & \text{if } c(v) \in \{a_r, b_r, q\} \\ 0 & \text{otherwise.} \end{cases}$$

Introduce edge (u, v) node: In introduce edge nodes, we check each state of the endpoints of the edge (u, v) and define f_i for each case.

- If $c(u) = s_\emptyset$, vertex u has no neighbor in A and B. Hence, we define the function h_i in this case as follows:

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) \notin \{a_l, a_r, b_l, b_r\}] \cdot h_j(c \times \{s_\emptyset\} \times \{c(v)\}, w, w').$$

- If $c(u) = s_A$, vertex u has neighbors in A but no neighbor in B. In this case, we have two cases. The first case is that $u \in S_\emptyset$ and $v \in A$ in the child node, because by adding edge (u, v) in the introduce edge (u, v) node, vertex u is moved from $S_\emptyset$ to S_A. The other case is that $u \in S_A$ and $v \notin B$ in the child node. If $v \in B$, vertex u is in S_{AB} in the parent node. We define h_i as follows. Note that only if $c(v) \in \{a_l, a_r\}$, we sum up two cases. If $c(v) \in \{b_l, b_r\}$, $h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := 0$, otherwise $h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := h_j(c \times \{s_A\} \times \{c(v)\}, w, w')$.

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) \in \{a_l, a_r\}]h_j(c \times \{s_\emptyset\} \times \{c(v)\}, w, w') + [c(v) \notin \{b_l, b_r\}]h_j(c \times \{s_A\} \times \{c(v)\}, w, w').$$

- The case that $c(u) = s_B$ is almost the same as in the above case, however, we swap the roles of A and B.

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) \in \{b_l, b_r\}] h_j(c \times \{s_\emptyset\} \times \{c(v)\}, w, w') \\ + [c(v) \notin \{a_l, a_r\}] h_j(c \times \{s_B\} \times \{c(v)\}, w, w').$$

- If $c(u) = s_{AB}$, we consider three cases: $u \in S_A$ and $v \in B$, $u \in S_B$ and $v \in A$, and $u \in S_{AB}$ and v is in an arbitrary set in the child node. For first and second case, vertex u is moved from S_A (resp., S_B) into S_{AB} by adding edge (u, v). If $u \in S_{AB}$, v is allowed to be in any set because a vertex in S_{AB} could connect to all sets. Therefore, we define f_i as follows:

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) \in \{b_l, b_r\}] h_j(c \times \{s_A\} \times \{c(v)\}, w, w') \\ + [c(v) \in \{a_l, a_r\}] h_j(c \times \{s_B\} \times \{c(v)\}, w, w') \\ + h_j(c \times \{s_{AB}\} \times \{c(v)\}, w, w').$$

- If $c(u) \in \{a_l, a_r\}$, then $c(v) \notin \{b_l, b_r, q\}$ since there is no edge between A, B and Q by the definition of connected partition. There is also no edge between A_l and A_r because (A_l, A_r) is a consistent cut. Therefore, if u is in A_l or A_r, then v is in the same set as u or is in one of S_A and S_{AB}. Note that because u is in A, v is not in $S_\emptyset$, S_B.

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) = c(u)] h_j(c \times \{c(u)\} \times \{c(v)\}, w, w') \\ + [c(v) \in \{s_A, s_{AB}\}] h_j(c \times \{c(u)\} \times \{c(v)\}, w, w').$$

- The case that $c(u) \in \{b_l, b_r\}$ is almost the same as in the above case, however, we replace A

by B.

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) = c(u)]h_j(c \times \{c(u)\} \times \{c(v)\}, w, w') + [c(v) \in \{s_B, s_{AB}\}]h_j(c \times \{c(u)\} \times \{c(v)\}, w, w').$$

- If $c(u) = q$, vertex u is in Q. Hence, v must be in $S_\emptyset, S_A, S_B, S_{AB}$, or Q because a vertex in Q has no neighbor in A and B by the definition of connected partition.

$$h_i(c \times \{c(u)\} \times \{c(v)\}, w, w') := [c(v) \in \{s_\emptyset, s_A, s_B, s_{AB}, q\}] \cdot h_j(c \times \{c(u)\} \times \{c(v)\}, w, w').$$

Forget v node: For forget nodes, if $c_j(v) \in \{s_\emptyset, s_A, s_B\}$, a partial solution does not satisfy the condition of connected partitions because any $v \in S$ must have neighbors of both A and B. For this reason, we only sum up for each state $c_j(v) \in \{s_{AB}, a_l, a_r, b_l, b_r, q\}$. The function h_i in forget nodes is defined as follows:

$$h_i(c, w, w') := \sum_{c_j(v) \in \{s_{AB}, a_l, a_r, b_l, b_r, q\}} h_j(c \times \{c_j(v)\}, w, w').$$

Join node: We denote the coloring and weight for each partial solution in i, j_1, j_2 by c_i, c_{j_1}, c_{j_2} and w_i, w_{j_1}, w_{j_2}, w'_i, w'_{j_1}, w'_{j_2}, respectively. Moreover, for a state subset $L \subseteq \{s_\emptyset, s_A, s_B, s_{AB}, a_l, a_r, b_l, b_r, q\}$, we define $c^{-1}(L)$ as the vertex set such that all vertices satisfy $c(v) \in L$. For a coloring c_i, we also define the subset D of tuples of (c_{j_1}, c_{j_2}) as the combinations of colorings of c_{j_1}, c_{j_2} like Section 3 such that:

- $\forall v \in c_i^{-1}(\{s_\emptyset, a_l, a_r, b_l, b_r, q\})$, $(c_{j_1}(v), c_{j_2}(v)) = (c_i(v), c_i(v))$,
- $\forall v \in c_i^{-1}(\{s_A\})$, $(c_{j_1}(v), c_{j_2}(v)) = (s_A, s_\emptyset), (s_\emptyset, s_A), (s_A, s_A)$,
- $\forall v \in c_i^{-1}(\{s_B\})$, $(c_{j_1}(v), c_{j_2}(v)) = (s_B, s_\emptyset), (s_\emptyset, s_B), (s_B, s_B)$, and
- $\forall v \in c_i^{-1}(\{s_{AB}\})$, $(c_{j_1}(v), c_{j_2}(v)) = (s_{AB}, s_\emptyset), (s_{AB}, s_A), (s_{AB}, s_B)$, $(s_{AB}, s_{AB}), (s_\emptyset, s_{AB}), (s_A, s_{AB}), (s_B, s_{AB}), (s_A, s_B), (s_B, s_A)$.

Let S^* be the vertex subset $c_i^{-1}(\{s_\emptyset, s_A, s_B, s_{AB}\})$. To sum up all combinations of vertex states and weights for counting, we define the function h_i. If $D = \emptyset$, we define $h_i(c_i, w_i, w_i') := 0$. Otherwise,

$$h_i(c_i, w_i, w_i') := \sum_{\substack{w_{j_1}+w_{j_2} \\ =w_i+w(S^*)}} \sum_{\substack{w'_{j_1}+w'_{j_2} \\ =w'_i+w'(S^*)}} \sum_{(c^*_{j_1}, c^*_{j_2})\in D} h_{j_1}(c^*_{j_1}, w_{j_1}, w'_{j_1}) h_{j_2}(c^*_{j_2}, w_{j_2}, w'_{j_2}).$$

From now, we analyze the running time of this algorithm. In leaf, introduce vertex, introduce edge, and forget nodes, we can compute f_i for each coloring c and weight w, w' in $O(1)$-time because we only use $O(1)$-operations. Therefore, the total running time for them is $O^*(9^\omega \cdot W \cdot W')$. However, in join nodes, we sum up all weight combinations and coloring combinations satisfying some conditions. There are 21 coloring combinations for each vertex and $W \cdot W'$ weight combinations. Therefore, we compute all f_i's in a join node in time $O^*(21^\omega \cdot W^2)$. Note that by definition, $O(W'^2)$ is a polynomial factor.

Theorem 4.5.5. Given a tree decomposition of width at most ω, there exists a Monte-Carlo algorithm that solves the decision version of MAXIMUM WEIGHT MINIMAL s-t SEPARATOR in time $O^*(21^\omega \cdot W^2)$. It cannot give false positives and may give false negatives with probability at most $1/2$.

Using the convolution technique [110], we can obtain a faster Monte-Carlo algorithm. The technique helps to speed up the computation for join nodes. First, we set the new coloring $\hat{c} : V \to \{s_{\bar{A}\bar{B}}, s_{\bar{A}}, s_{\bar{B}}, s_{all}, a_l, a_r, b_l, b_r, q\}$. The state $s_{\bar{A}\bar{B}}$ represents that a vertex v is in S and has no neighbor of A and B. The state $s_{\bar{A}}$ (resp., $s_{\bar{B}}$) represents a vertex v is in S and has no neighbor of A (resp., B), respectively. Finally, the state s_{all} represents a vertex v is in S without constraints.

Then, we show the following lemma to transform between c and $\hat{c}$.

Lemma 4.5.6. Let i be a node of a tree decomposition and $h_i(c, w, w')$ be a counting function to count the number of partial solutions of MAXIMUM WEIGHT MINIMAL s-t SEPARATOR of each weight w, w', corresponding to each coloring c, $\hat{c}$ of a node i. Then we can transform from one coloring to another coloring for each function without loss of information. Moreover, it is computed in $O(W \cdot W' \cdot 9^\omega \cdot |X_i|)$.

Proof: This proof scheme follows [110]. We consider the immediate ℓ-th step in the transformation.

For $h_i(c \times \{c(v)\} \times \hat{c}, w, w')$, v is a vertex which turns into the state of another coloring in the ℓ-th step, and c is the partial coloring of size $\ell - 1$ and $\hat{c}$ is also the partial coloring of size $|X_i| - \ell$. Here, for simplicity, we denote $h_i(c \times \{c(v)\} \times \hat{c}, w, w')$ and $h_i(c \times \{\hat{c}(v)\} \times \hat{c}, w, w')$ by $h_i(c(v))$ and $h_i(\hat{c}(v))$.

Since h_i is the number of partial solutions with a consistent cut, the transformation from c to $\hat{c}$ of h_i in ℓ-th step is as follows:

- $h_i(s_{\bar{A}\bar{B}}) = h_i(s_\emptyset)$
- $h_i(s_{\bar{A}}) = h_i(s_\emptyset) + h_i(s_B)$
- $h_i(s_{\bar{B}}) = h_i(s_\emptyset) + h_i(s_A)$
- $h_i(s_{all}) = h_i(s_\emptyset) + h_i(s_A) + h_i(s_B) + h_i(s_{AB})$.

Conversely, we can transform from $\hat{c}$ to c as follows:

- $h_i(s_\emptyset) = h_i(s_{\bar{A}\bar{B}})$
- $h_i(s_A) = h_i(s_{\bar{B}}) - h_i(s_{\bar{A}\bar{B}})$
- $h_i(s_B) = h_i(s_{\bar{A}}) - h_i(s_{\bar{A}\bar{B}})$
- $h_i(s_{AB}) = h_i(s_{all}) - h_i(s_{\bar{A}}) - h_i(s_{\bar{B}}) + h_i(s_{\bar{A}\bar{B}})$.

These equations follow from equations of the transformation from c to $\hat{c}$.

For each of the colorings $c, \hat{c}$, each transformation can be performed in $O(|X_i|)$-steps. Thus, the total running time of each direction is $O(W \cdot W' \cdot 9^\omega \cdot |X_i|)$. ■

Therefore, we first transform the original coloring c to the new coloring $\hat{c}$ in $O(W \cdot W' \cdot 9^\omega \cdot |X_i|)$-time. Then we compute the following function h_i for the new coloring $\hat{c}$ in $O(9^\omega \cdot W^2)$-time:

$$h_i(\hat{c}, w, w_i') := \sum_{w_{j_1} + w_{j_2} = w_i + w(\hat{S}^*)} \sum_{w'_{j_1} + w'_{j_2} = w_i' + w'(\hat{S}^*)} h_{j_1}(\hat{c}, w_{j_1}, w'_{j_1}) h_{j_2}(\hat{c}, w_{j_2}, w'_{j_2}),$$

where $\hat{S}^* = \hat{c}^{-1}(\{s_\emptyset, s_A, s_B, s_{AB}\}) \subseteq V$. Note that $\hat{c}_i, \hat{c}_{j_1}, \hat{c}_{j_2}$ are the same coloring. Finally, we transform $\hat{c}$ to c. Thus, total running time of this algorithm is $O^*(9^\omega \cdot W^2)$.

	$s_\emptyset$	s_A	s_B	s_{AB}	a_l	a_r	b_l	b_r	q
$s_\emptyset$	$s_\emptyset$	s_A	s_B	s_{AB}					
s_A	s_A	s_A	s_{AB}	s_{AB}					
s_B	s_B	s_{AB}	s_B	s_{AB}					
s_{AB}	s_{AB}	s_{AB}	s_{AB}	s_{AB}					
a_r					a_r				
a_l						a_l			
b_r							b_r		
b_l								b_l	
q									q

Table 4.2. Combinations of the original coloring in join node

	s_{all}	$s_{\bar{A}}$	$s_{\bar{B}}$	$s_{\bar{A}\bar{B}}$	a_l	a_r	b_l	b_r	q
s_{all}	s_{all}								
$s_{\bar{A}}$		$s_{\bar{A}}$							
$s_{\bar{B}}$			$s_{\bar{B}}$						
$s_{\bar{A}\bar{B}}$				$s_{\bar{A}\bar{B}}$					
a_r					a_r				
a_l						a_l			
b_r							b_r		
b_l								b_l	
q									q

Table 4.3. Combinations of the new coloring in join node

Theorem 4.5.7. Given a tree decomposition of width at most ω, there exists a Monte-Carlo algorithm that solves the decision version of MAXIMUM WEIGHT MINIMAL SEPARATOR and MAXIMUM WEIGHT MINIMAL s-t SEPARATOR in time $O^*(9^\omega \cdot W^2)$. It cannot give false positives and may give false negatives with probability at most $1/2$. If the input graph is unweighted, the running time is $9^\omega n^{O(1)}$.

As usual for this type of algorithm, the probability of a false negative can be made arbitrarily small by repeating the algorithm.

4.6 Conclusion

In this chapter, we studied MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR. We first showed MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR is NP-hard even on unweighted bipartite graphs. On the other hand, we designed a deterministic $\omega^{O(\omega)}n^{O(1)}$-algorithm. However, this algorithm is not a single exponential algorithm. We then designed an $O^*(9^\omega \cdot W^2)$-randomized algorithm based on the Cut & Count.

Chapter 5

Finding Environmentally Critical Transmission Sectors, Transactions, and Paths in Global Supply Chain Networks

5.1 Introduction

Environmentally extended input-output analysis has been widely used to estimate production- and consumption-based emissions in many countries [52, 90, 93], and in Japan a national emission inventory has been provided to the public every five years [83]. A major advantage of using environmental energy input-output analysis is that product supply-chain networks can be easily modeled and environmentally and ecologically important sectors and paths can be identified from the network [68, 69, 91, 114]. Clustering analysis has also been applied to environmental energy input-output analysis in order to find environmentally important industry groups (i.e., industry clusters) that induce higher CO_2 emissions along their supply chains [55–57].

It is important to note that the environmentally critical sectors, transactions, paths, and clusters identified by input-output analysis play important roles in reducing consumption-based emissions through the entire economy. A supply-chain path is composed of transactions between two sectors.

An environmentally critical sector as identified by key sector analysis [51,53,68,95] is considered to be a sector that contributes to emitting larger environmental emissions (e.g., CO_2 emissions) in the economy through not only purchasing emission- and energy-intensive products from other upstream sectors but producing emission- and energy-intensive products in its own sector. Therefore, an effective emission reduction policy (i.e., technology policy) is to improve the production technology of the critical sector toward a cleaner one that has less energy consumption and environmental emissions along the product supply chain.

Information on environmentally critical paths [68] and clusters [55–57, 96] can support a technology policy in the sense that policymakers can find high priority supply-chain paths (i.e., upstream products) and clusters (i.e., upstream product systems). However, it is not an easy task to find high-priority paths and clusters from the supply-chain complexity due to the problem of computation (e.g., [56]).

Liang et al. (2016) proposed a useful indicator, vertex betweenness centrality, of a specific sector by combining input-output analysis with social network analysis [75]. More specifically, Liang et al. (2016) formulated the vertex betweenness centrality index by applying the notion of network centrality [37–39] to structural path analysis [23, 68]. Liang et al. (2016) defined a specific sector with higher vertex betweenness centrality in a supply-chain network as a critical transmission sector in the sense that many sectors supply their products to final consumers by passing through the specific sector, and consequently the transmission sector contributes to emitting a large amount of environmental emissions in the economy. Although key sector analysis [51, 53, 68, 95] implicitly considers the transmission power of a specific sector, previous analyses failed to define this as the vertex betweenness centrality consistent with network theory [37–39]. A technology policy for reducing environmental emissions should target high-priority sectors with higher vertex betweenness centrality.

In this study, we develop another centrality index, edge betweenness centrality, consistent with environmental energy input-output analysis following Liang et al. (2016) and prove a mathematical relationship between the "edge" betweenness centrality and the "vertex" betweenness centrality. The edge betweenness centrality indicates how much 'embodied' environmental emissions of products flow through the transaction and to what extent sectors are connecting through a specific edge (i.e., a transaction) in terms of supply-chain complexity. It is important to note that the embodied

environmental emissions (e.g., embodied CO_2 emissions) in supply-chain networks are caused by embodied energy consumption in the entire economy [52, 94].

The edge betweenness centrality for a particular transaction developed in this study can be regarded as the sum of environmental emissions associated with 'infinite' supply-chain paths that include the specific transaction identified using structural path analysis (SPA) (e.g., [68, 81, 91, 92]). However, a technical problem of SPA is that it is impossible to identify 'infinite' paths that include the particular transaction; therefore, the edge betweenness centrality developed in this study can be a useful indicator to express the importance (or criticality) of a particular transaction in the entire supply-chain network.

Liang et al. (2015) also proposed strongest path betweenness [74]. This index represents the importance of a sector in the supply-chain network as a center transmitting or facilitating the creation of environmental impacts. Roughly speaking, it is defined as the sum of the strengths of all strongest paths in the supply-chain network passing through a sector. The strongest path in [74] is defined as the environmentally important path that causes the largest CO_2 emissions in sector i owing to one unit of value added in sector j. It is important to note that the strongest path developed in [74] is defined for a path, whereas the edge betweenness centrality developed in this study is defined for a transaction. A strongest path from sector i to sector j represents the most inefficient path among all possible paths from i to j. The point of difference from the vertex betweenness centrality and the edge betweenness centrality is that the strongest path betweenness does not consider 'infinite' supply-chain paths.

In this study, we compute the edge and vertex betweenness centralities using the environmentally extended multi-regional input-output table covering 35 industrial sectors and 41 countries and regions in 2008 [24, 109] and identify high-priority sectors with higher vertex betweenness centrality and high-priority transactions with higher edge betweenness centrality in global supply-chain networks. Finally, we argue how those high-priority sectors and transactions can contribute to reducing CO_2 emissions as climate mitigation.

It should be noted that although we mainly focus on CO_2 emissions as a case study, the method developed in this chapter can be applied to energy consumption and other environmental pollutants such as NO_x and SO_x. In particular, although energy consumption and CO_2 emissions are both typical analysis subjects for these methods, our reason for mainly focusing on CO_2 emissions is

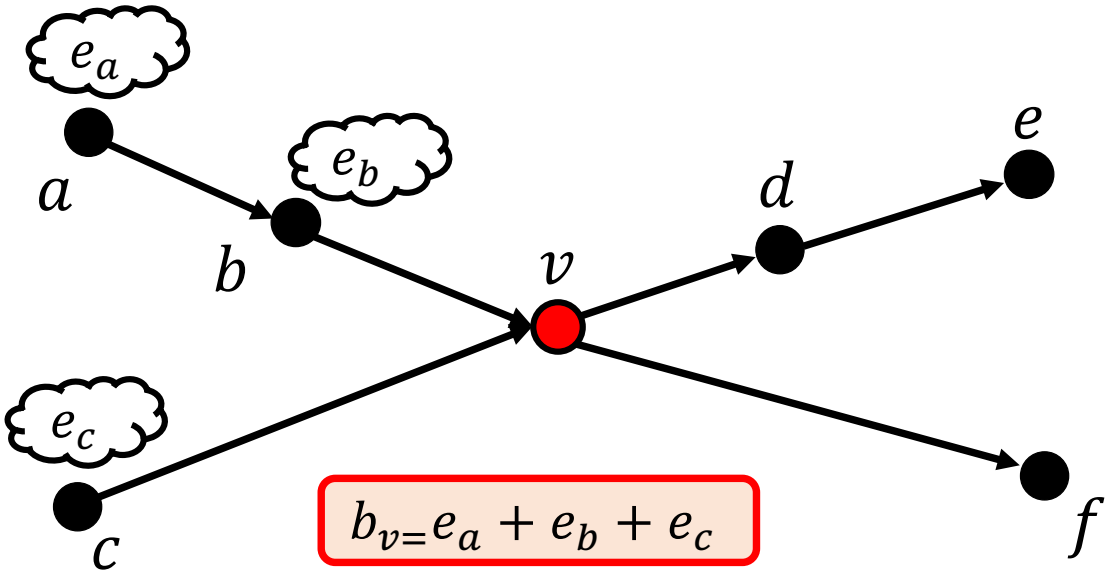

Figure 5.1. An example of the vertex betweenness centrality for sector v

interest in the effects on recent global warming. For this purpose, we conducted similar betweenness centrality analyses focused on energy consumption and computed rank correlation coefficients for both vertex and edge betweenness centralities for CO_2 emissions and energy consumption. Finally, we discuss a more direct relationship between CO_2 emissions and energy consumption with a focus on the vertex and edge betweenness centralities in Section 5.5.

The remainder of this chapter is organized as follows: Section 5.2 revisits the vertex betweenness centrality as proposed in [75]; Section 5.3 develops the edge betweenness centrality consistent with input output analysis; Section 5.4 presents the mathematical relationship between the vertex betweenness centrality and the edge betweenness centrality; Section 5.5 presents and discusses the results; and finally Section 5.6 concludes this chapter.

5.2 Vertex betweenness centrality proposed in Liang et al. (2016)

The vertex betweenness centrality of a specific sector proposed in Liang et al. (2016) was defined as the sum of environmental emissions associated with the supply-chain paths passing through the specific sector [75]. Figure 5.1 illustrates the vertex betweenness centrality of sector v in a supply-chain network with seven vertices and six edges. In the figure, e_a, e_b, and e_c are respectively the environmental emissions in upstream sectors a, b, and c triggered by the transactions among

downstream sectors, d, e, and f. In this case, the vertex betweenness centrality of sector v can be calculated as $b_v = e_a + e_b + e_c$.

Following Liang et al. (2016), the vertex betweenness centrality b_v of a specific sector v can be formulated as:

$$b_v = \sum_{s=1}^{n} \sum_{t=1}^{n} \sum_{r=1}^{\infty} q_r \times w(s, k_1, k_2, \ldots, k_r, t), \tag{5.1}$$

where $w(s, k_1, k_2, \ldots, k_r, t)$ is the environmental emissions associated with all supply-chain paths from sector s to sector t passing through r sectors, sector $s \rightarrow$ sector $k_1 \rightarrow$ sector $k_2 \rightarrow \cdots \rightarrow$ sector $k_r \rightarrow$ sector t via sector v, and q_r is the number of times that sector v appears in the supply-chain paths. Following the idea of structural path analysis (e.g., [23, 68]), $w(s, k_1, k_2, \ldots, k_r, t)$ can be formulated as $f_s a_{sk_1} a_{k_1 k_2} \cdots a_{k_r t} y_t$, where y_t is the final demand of sector t, $a_{k_1 k_2}$ is the intermediate input from sector k_1 directly required for producing one unit of output in sector k_2, and f_s is the direct environmental emissions per unit of output of sector s. Accordingly, $a_{sk_1} a_{k_1 k_2} \cdots a_{k_r t}$ represents the intermediate input from sector s indirectly required for producing one unit of output in sector t. It should be noted that if $r = 1$ and $k_1 = v$, we have $b_v = \sum_{s=1}^{n} \sum_{t=1}^{n} f_s a_{sv} a_{vt} y_t$ from Eq. (5.1).

We further define the following supply-chain path:

$$b_v(\ell_1, \ell_2) = \sum_{1 \le k_1, \ldots, k_{\ell_1} \le n} \; \sum_{1 \le j_1, \ldots, j_{\ell_2} \le n} f_{k_1} a_{k_1 k_2} \cdots a_{k_{\ell_1} v} a_{v j_1} \cdots a_{j_{\ell_2 - 1} j_{\ell_2}} y_{j_{\ell_2}}, \tag{5.2}$$

where $b_v(\ell_1, \ell_2)$ represents the environmental emissions associated with supply-chain paths that pass through sector v that has upstream industrial supply chains with path lengths of ℓ_1 and downstream industrial supply chains with path lengths of ℓ_2. Then, we can rewrite $b_v(\ell_1, \ell_2)$ as the following matrix multiplication:

$$\begin{aligned} b_v(\ell_1, \ell_2) &= \sum_{1 \le k_1, \ldots, k_{\ell_1} \le n} \; \sum_{1 \le j_1, \ldots, j_{\ell_2} \le n} f_{k_1} a_{k_1 k_2} \cdots a_{k_{\ell_1} v} a_{v j_1} \cdots a_{j_{\ell_2 - 1} j_{\ell_2}} y_{j_{\ell_2}} \\ &= \sum_{1 \le k_1, \ldots, k_{\ell_1} \le n} f_{k_1} a_{k_1 k_2} \cdots a_{k_{\ell_1} v} \sum_{1 \le j_1, \ldots, j_{\ell_2} \le n} a_{v j_1} \cdots a_{j_{\ell_2 - 1} j_{\ell_2}} y_{j_{\ell_2}} \\ &= (\mathbf{f}' \mathbf{A}^{\ell_1})_v (\mathbf{A}^{\ell_2} \mathbf{y})_v \end{aligned} \tag{5.3}$$

$$= \mathbf{f}'\mathbf{A}^{\ell_1}\mathbf{J}_v\mathbf{A}^{\ell_2}\mathbf{y},$$

where $\mathbf{y} = (y_j)$ represents an $(n \times 1)$ final demand vector expressing the final demand for sector j, $\mathbf{A} = (a_{ij})$ $(n \times n)$ is an input coefficient matrix expressing the intermediate input from sector i per unit of output of sector j, $\mathbf{J}_v$ is an $(n \times n)$ matrix whose (v, v) element is 1 and other matrix elements are 0, and $\mathbf{f} = (f_i)$ represents an $(n \times 1)$ emission coefficient vector expressing direct environmental emissions per unit of output of sector i. The prime denotes the matrix transpose. $(\mathbf{f}'\mathbf{A}^{\ell_1})_v$ and $(\mathbf{A}^{\ell_2}\mathbf{y})_v$ represent the v-th elements of the respective computed vectors.

It is well known that the direct and indirect requirement matrix (i.e., Leontief inverse matrix) can be obtained as the following power series expansion [77, 112]:

$$\mathbf{L} = (\mathbf{I} - \mathbf{A})^{-1} = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots . \tag{5.4}$$

If we define the indirect requirement matrix as $\mathbf{T} = \mathbf{LA} = \mathbf{AL} = \mathbf{L} - \mathbf{I} = \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 \cdots$ and from Eqs. 5.1 and 5.3 we notice that $b_v = \sum_{\ell_1=1}^{\infty} \sum_{\ell_2=1}^{\infty} b_v(\ell_1, \ell_2)$, then the vertex betweenness centrality of sector v can be reformulated as follows:

$$\begin{aligned} b_v &= \sum_{\ell_1=1}^{\infty} \sum_{\ell_2=1}^{\infty} b_v(\ell_1, \ell_2) \\ &= \sum_{\ell_1=1}^{\infty} \sum_{\ell_2=1}^{\infty} \mathbf{f}'\mathbf{A}^{\ell_1}\mathbf{J}_v\mathbf{A}^{\ell_2}\mathbf{y} \\ &= (\sum_{\ell_1=1}^{\infty} \mathbf{f}'\mathbf{A}^{\ell_1})\mathbf{J}_v(\sum_{\ell_2=1}^{\infty} \mathbf{A}^{\ell_2}\mathbf{y}) \\ &= \mathbf{f}'(\sum_{\ell_1=1}^{\infty} \mathbf{A}^{\ell_1})\mathbf{J}_v(\sum_{\ell_2=1}^{\infty} \mathbf{A}^{\ell_2})\mathbf{y} \\ &= \mathbf{f}'\mathbf{T}\mathbf{J}_v\mathbf{T}\mathbf{y}. \end{aligned} \tag{5.5}$$

The matrix computation in Eq. (5.5) yields the vertex betweenness centrality of sector v. Hence we can compute the vertex betweenness centrality for each node (i.e., each sector) in time $O(n^\omega)$, where ω denotes the matrix multiplication exponent. It is well known that $\omega < 2.373$ [113]. Since

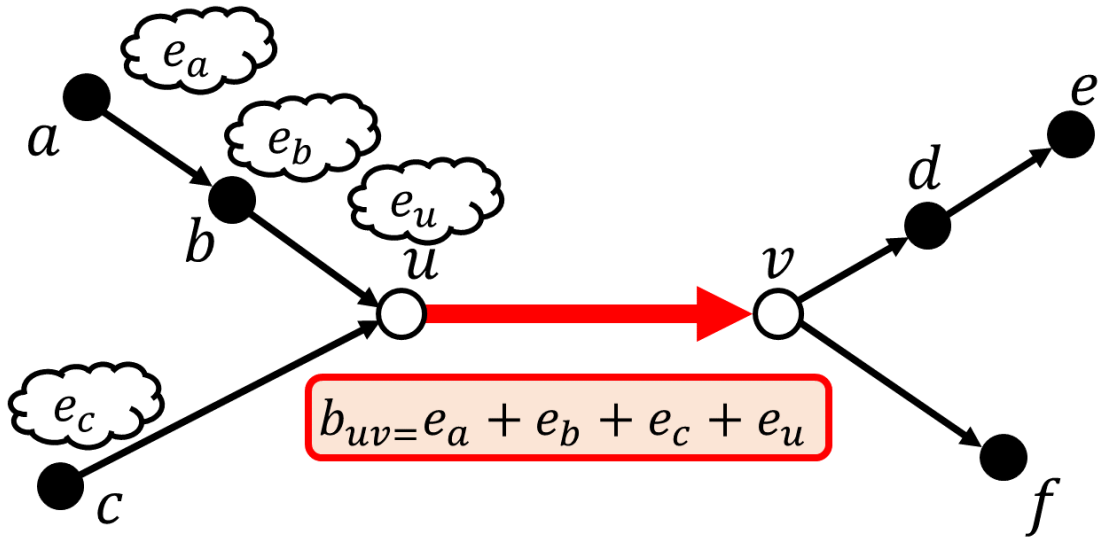

Figure 5.2. An example of the edge betweenness centrality for transaction (u, v)

the number of sectors is n, the total running time is $O(n^{\omega+1})$ and therefore very low.

5.3 Edge betweenness centrality in input-output networks

In this study, we develop another betweenness centrality, edge betweenness centrality, for supply-chain networks. Let (u, v) be a directed edge (i.e., a transaction from sector u to sector v in a supply-chain network). We then define the edge betweenness centrality for a specific transaction (u, v) as the amount of environmental emissions associated with all supply-chain paths passing through the transaction (u, v). As in Eq. (5.1), we can formulate the edge betweenness centrality for the transaction (u, v) as follows:

$$b_{uv} = \sum_{s=1}^{n} \sum_{t=1}^{n} \sum_{r=1}^{\infty} p_r \times w(s, k_1, k_2, \ldots, k_r, t), \tag{5.6}$$

where p_r represents the number of times that transaction (u, v) appears in all supply-chain paths from sector s to sector t passing through r sectors. As with the vertex betweenness centrality, the edge betweenness centrality is explained using a simple example: Figure 5.2 illustrates the betweenness centrality of transaction (u, v) in a supply-chain network with eight vertices and seven edges. In Figure 5.2, e_a, e_b, e_c, and e_u are respectively the environmental emissions in upstream sectors a,

b, c, and u triggered by the transactions among downstream sectors, d, e, f, and v. In this case, the edge betweenness centrality of transaction (u, v) can be calculated as $b_{uv} = e_a + e_b + e_c + e_u$.

As in Eq. (5.2), we denote by $b_{uv}(\ell_1, \ell_2)$ the environmental emissions associated with supply-chain paths that pass through a transaction from sector u to sector v that has upstream industrial supply chains with path lengths of ℓ_1 and downstream industrial supply chains with path lengths of ℓ_2 as follows:

$$b_{uv}(\ell_1, \ell_2) = \sum_{1 \le k_1, \ldots, k_{\ell_1} \le n} \sum_{1 \le j_1, \ldots, j_{\ell_2} \le n} f_{k_1} a_{k_1 k_2} \cdots a_{k_{\ell_1} u} a_{uv} a_{v j_1} \cdots a_{j_{\ell_2 - 1} j_{\ell_2}} y_{j_{\ell_2}}. \tag{5.7}$$

Subsequently, we reformulate $b_{uv}(\ell_1, \ell_2)$ as the following matrix multiplication:

$$\begin{aligned} b_{uv}(\ell_1, \ell_2) &= \sum_{1 \le k_1, \ldots, k_{\ell_1} \le n} \sum_{1 \le j_1, \ldots, j_{\ell_2} \le n} f_{k_1} a_{k_1 k_2} \cdots a_{k_{\ell_1} u} a_{uv} a_{v j_1} \cdots a_{j_{\ell_2 - 1} j_{\ell_2}} y_{j_{\ell_2}} \\ &= a_{uv} \sum_{1 \le k_1, \ldots, k_{\ell_1} \le n} f_{k_1} a_{k_1 k_2} \cdots a_{k_{\ell_1} u} \sum_{1 \le j_1, \ldots, j_{\ell_2} \le n} a_{v j_1} \cdots a_{j_{\ell_2 - 1} j_{\ell_2}} y_{j_{\ell_2}} \\ &= a_{uv} (\mathbf{f}' \mathbf{A}^{\ell_1})_u (\mathbf{A}^{\ell_2} \mathbf{y})_v \\ &= a_{uv} \mathbf{f}' \mathbf{A}^{\ell_1} \mathbf{J}_{uv} \mathbf{A}^{\ell_2} \mathbf{y}. \end{aligned} \tag{5.8}$$

Here $\mathbf{J}_{uv}$ represents an $(n \times n)$ matrix whose (u, v) element is 1 and other elements are 0. It should be noted that when $\ell_1 = \ell_2 = 0$, $b_{uv}(0, 0) = f_u a_{uv} y_v$. Considering all the supply-chain paths passing through transaction (u, v), its edge betweenness centrality b_{uv} in Eq. (5.6) can be computed based on the following matrix computation:

$$\begin{aligned} b_{uv} &= \sum_{\ell_1=0}^{\infty} \sum_{\ell_2=0}^{\infty} b_{uv}(\ell_1, \ell_2) \\ &= \sum_{\ell_1=0}^{\infty} \sum_{\ell_2=0}^{\infty} a_{uv} \mathbf{f}' \mathbf{A}^{\ell_1} \mathbf{J}_{uv} \mathbf{A}^{\ell_2} \mathbf{y} \\ &= a_{uv} (\sum_{\ell_1=0}^{\infty} \mathbf{f}' \mathbf{A}^{\ell_1}) \mathbf{J}_{uv} (\sum_{\ell_2=0}^{\infty} \mathbf{A}^{\ell_2} \mathbf{y}) \\ &= a_{uv} \mathbf{f}' (\sum_{\ell_1=1}^{\infty} \mathbf{A}^{\ell_0}) \mathbf{J}_{uv} (\sum_{\ell_2=0}^{\infty} \mathbf{A}^{\ell_2}) \mathbf{y} \\ &= a_{uv} \mathbf{f}' \mathbf{L} \mathbf{J}_v \mathbf{L} \mathbf{y}. \end{aligned} \tag{5.9}$$

If a specific transaction from u to v does not exist in the supply-chain network, its edge betweenness centrality is zero because we have $a_{uv} = 0$. As in the vertex betweenness centrality, each edge betweenness centrality is computed in $O(n^{\omega})$. The total running time is $O(n^{\omega+2})$ because there may exist $O(n^2)$ transactions. Moreover, let m be the number of transactions in the input-output table. If an input-output matrix is sparse, we can compute all edge betweenness centralities in time $O(mn^{\omega})$. Therefore, the computation cost is also very low.

5.4 Relationship between vertex betweenness centrality and edge betweenness centrality

We describe the relationship between the vertex betweenness centrality and the edge betweenness centrality from the point of view of graph theory. First, we define the edge betweenness centrality matrix.

Definition 5.4.1. The matrix $\mathbf{B} = (b_{uv})$ is called an *edge betweenness centrality matrix*.

Given a directed and weighed graph $G = (V, E)$, we denote the weight for edge (u, v) by w_{uv}. We further define $d_w^{in}(v) = \sum_{u \in N^{in}(v)} w_{uv}$ as the weighted in-degree of vertex v and $d_w^{out}(v) = \sum_{u \in N^{out}(v)} w_{vu}$ as the weighted out-degree of vertex v.

If we regard the edge betweenness centrality matrix $\mathbf{B} = (b_{uv})$ as the above adjacency matrix $\mathbf{W} = (w_{uv})$, then the weighted in-degree of sector v, $d_w^{in}(v)$, expressing the environmental emissions associated with supply-chain paths that pass through the transactions from other sectors to sector v and the weighted out-degree of sector v, $d_w^{out}(v)$, expressing the environmental emissions associated with supply-chain paths that pass through the transactions from sector v to other sectors can be formulated as follows.

Lemma 5.4.2. The following two equations for respectively the weighted in-degree of sector v and the weighted out-degree of sector v hold:

$$d_w^{in}(v) = \mathbf{f}'\mathbf{T}\mathbf{J}_v\mathbf{L}\mathbf{y}, \tag{5.10}$$

$$d_w^{out}(v) = \mathbf{f}'\mathbf{L}\mathbf{J}_v\mathbf{T}\mathbf{y}. \tag{5.11}$$

Proof: We only prove Eq. (5.10) because the proof of Eq. (5.11) is almost the same.

$$\begin{aligned} d_w^{in}(v) &= \sum_{u \in N^{in}(v)} w_{uv} = \sum_{u=1}^{n} b_{uv} \\ &= \sum_{u=1}^{n} a_{uv} \mathbf{f}' \mathbf{L} \mathbf{J}_{uv} \mathbf{L} \mathbf{y} \\ &= \mathbf{f}' \mathbf{L} \sum_{u=1}^{n} a_{uv} \mathbf{J}_{uv} \mathbf{L} \mathbf{y} \\ &= \mathbf{f}' \mathbf{L} \mathbf{A} \mathbf{J}_v \mathbf{L} \mathbf{y} \\ &= \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{L} \mathbf{y}. \end{aligned}$$

■

We finally obtain the following theorem using Lemma 5.4.2.

Theorem 5.4.3.

$$b_v = d_w^{in}(v) - \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{y} = d_w^{out}(v) - \mathbf{f}' \mathbf{J}_v \mathbf{T} \mathbf{y}. \tag{5.12}$$

Proof: According to Lemma 5.4.2, $d_w^{in}(v) = \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{L} \mathbf{y}$ (Eq. (5.10)) holds and $\mathbf{T} = \mathbf{L} - \mathbf{I}$ also holds. Therefore, from Eq. (5.5) we have:

$$\begin{aligned} b_v &= \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{T} \mathbf{y} \\ &= \mathbf{f}' \mathbf{T} \mathbf{J}_v (\mathbf{L} - \mathbf{I}) \mathbf{y} \\ &= \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{L} \mathbf{y} - \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{y} \\ &= d_w^{in}(v) - \mathbf{f}' \mathbf{T} \mathbf{J}_v \mathbf{y}. \end{aligned}$$

We can prove similarly that $b_v = d_w^{out}(v) - \mathbf{f}' \mathbf{J}_v \mathbf{T} \mathbf{y}$ also holds. ■

Theorem 5.4.3 shows that the vertex betweenness centrality of sector v is equal to the weighted in-degree of sector v minus the amount of all sectors' embodied emissions triggered by the final demand of sector v, and it also shows that the weighted out-degree of sector v minus the amount of v's embodied emissions triggered by the final demands of sectors coincides with the vertex betweenness centrality of sector v.

5.5 Data and results

5.5.1 Consumption-based emissions of nations

Using the inventory database of energy consumption and CO_2 emissions from fossil fuels combustion of 35 industrial sectors of 41 countries and regions provided by the 2008 world input-output database (see Tables A.1 and A.2 in appendices) [24, 109], we first calculated the consumption-based emissions of nations estimated by the conventional carbon footprint method [52, 58, 93, 108]. The total number of sectors in this study is $n = 35 \times 41 = 1435$, and thus the network analysis formulated in the previous section can be easily extended to a multi-regional framework.

The result based on the world input-output database shows that the total of all consumption-based CO_2 emissions in 2008 was 25,598 Mt-CO_2, of which the U.S.A. and China accounted for 21% and 18%, respectively (Figure 5.3). Thus, it is well known that the U.S.A. and China have contributed a large amount of CO_2 emissions through consumption of goods and services [52]. An important point is that the U.S.A. transferred a large portion of its consumption-based CO_2 emissions to developing countries such as China and India, whereas China and India transferred relative small portions of their CO_2 emissions to other countries (Figure 5.4) (e.g., [56, 94]).

As shown in Figure 5.4, CO_2 emission transfers are greatly influenced by the structure of global supply-chain networks (e.g., [56]). Accordingly, if we adopt emission reduction measures focused on the sectors and transactions of highest priority within supply-chain networks, we can expect to efficiently reduce emissions associated with entire supply chains. Up to now, however, no one has proposed a centrality index that reflects the complex structure of global supply-chain networks associated with the final demand for particular products or analyzed such structures.

5.5.2 Vertex betweenness centrality in global supply-chain networks

As explained in Section 5.2, Liang et al. (2016) developed a new network index called vertex betweenness centrality. This index indicates the intermediate nodes of huge supply-chain paths, or in other words, the "mediating function" of sectors. For the induced CO_2 emissions in the global supply chains considered in this study, sectors having the high vertex betweenness centrality represent environmentally important sectors along a supply chain.

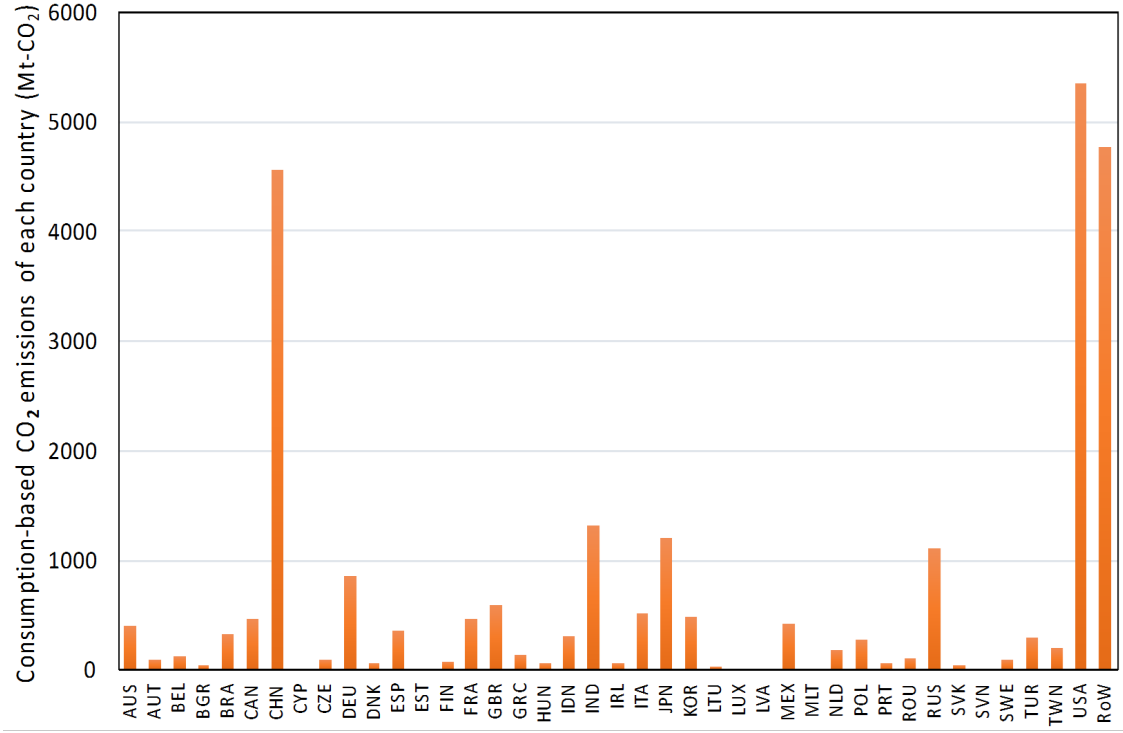

Figure 5.3. Consumption-based CO_2 emissions of each country

Table 5.1 summarizes the results for the top 30 sectors in terms of the vertex betweenness centrality in global supply-chain networks associated with the final demand in each demand country. It is clear from the table that most of the CO_2 emissions indirectly generated via the sectors of the high vertex betweenness centrality are induced by the final demand in the same country (Table 5.1). More specifically, we see that China's Basic Metals and Fabricated Metal sector — the sector having the highest vertex betweenness centrality (1253 Mt-CO_2) — is an important sector in global supply chains formed by the final demand in China (Table 5.1). Since approximately 90% of the 4.6 Gt of CO_2 emissions induced by the final demand in China (i.e., China's consumption-based emissions) are emitted in China (Figure 5.3), it follows that to effectively reduce CO_2 emissions associated with global supply-chain networks within China, the Basic Metals and Fabricated Metal sector — the sector having the highest vertex betweenness centrality — requires more environmentally friendly supply-chain management of raw materials inside China (Table 5.1).

The Electrical and Optical Equipment sector of China, ranked at both No. 5 and No. 20 in Table 5.1, is clearly an important sector in the global supply chains formed by the final demands

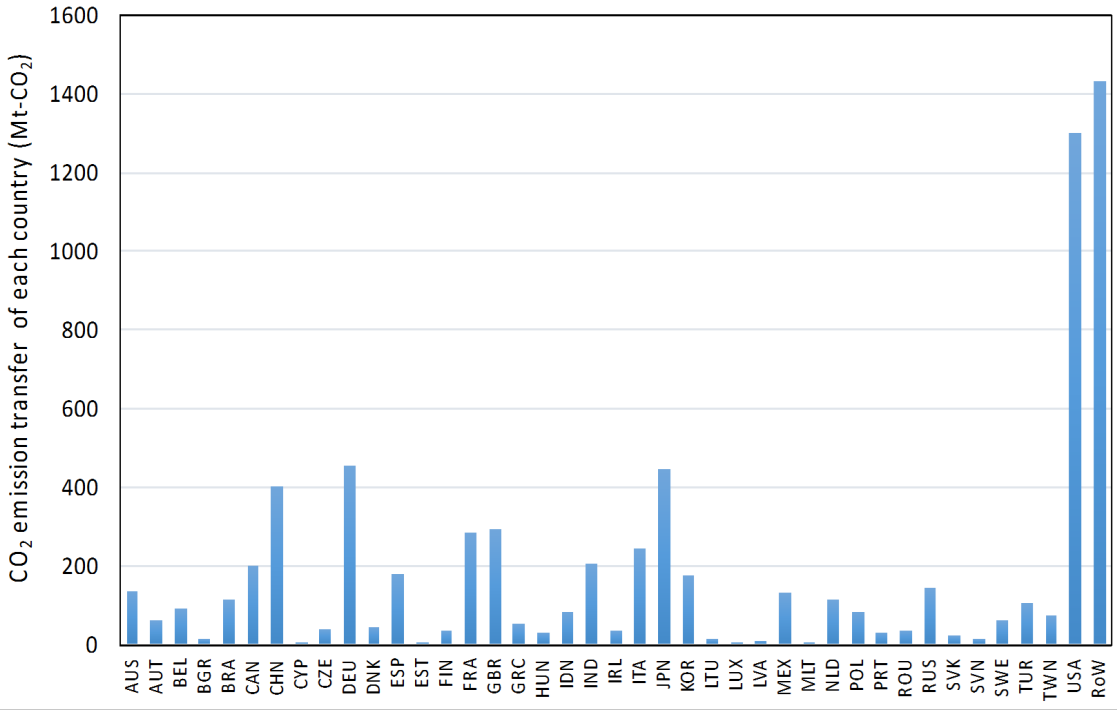

Figure 5.4. CO_2 emission transfer of each country

in both China and the U.S.A. Based on the betweenness centrality, 414 Mt of the CO_2 emissions induced by the final demand in China is indirectly generated via China's Electrical and Optical Equipment sector, but in addition 164 Mt of the CO_2 emissions induced by the final demand in theU.S.A. are also emitted via this sector (Table 5.1). This result demonstrates that substantial amounts of CO_2 emissions are induced by the final demands in other countries.

To obtain the vertex betweenness centrality for a network created by superimposing the global supply-chain networks formed by the final demand in each country, we substitute $\mathbf{y}^g = \sum_{r=1}^{41} \mathbf{y}^r$, the global final demand obtained by summing the final demand $\mathbf{y}^r$ in each country, into the right-hand side of Eq. (5.5) (Table 5.2). From Table 5.2, we see that of the CO_2 emissions associated with the final demands in all countries (41 countries and regions), emissions from China's Electrical and Optical Equipment sector amounted to 1037 Mt (Table 5.2), which are similar with Japan's total CO_2 emissions in 2008 (1282 Mt) [78]. If we compare the value of vertex betweenness centrality of China's Electrical and Optical Equipment sector associated with the final demand in China (No. 5 in Table 5.1) with the value of vertex betweenness centrality of the same sector relating to world final

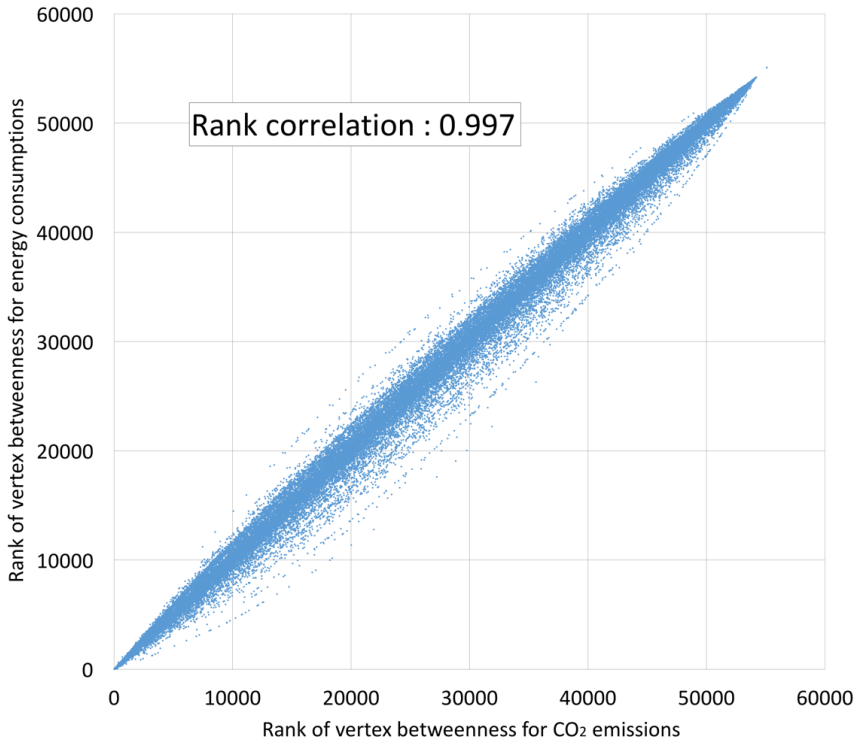

Figure 5.5. Rank correlation of vertex betweenness centrality indices between CO_2 emissions and energy consumptions associated with the final demand in each demand country

demand (No. 4 in Table 5.2), we find that approximately 60% of CO_2 emissions induced by the final demands outside China are generated in this specific sector of the Chinese economy. Remarkably, the final demand in the U.S.A. accounts for approximately 16% of the CO_2 emissions, while the final demands in Japan and Germany account for approximately 4% each. Undoubtedly therefore, this is a key sector for the CO_2 emission transfer to China via supply chains (Tables 5.1, 5.2).

From the above observations, we can conclude that environmentally friendly production management and raw material procurement in China's Electrical and Optical Equipment sector, which has the high vertex betweenness centrality in global supply-chain networks with respect to the total final demand worldwide, has the potential for effectively reducing CO_2 emissions over entire global supply-chain networks.

We also found that the Spearman's rank correlation of vertex betweenness indices of CO_2 emissions and energy consumption associated with the final demand by each demand country is very high, 0.997 (Figure 5.5), which means that the above-mentioned CO_2 mitigation policy (i.e., environmentally friendly production management and raw material procurement) focused on high-

ranked sectors listed in Table 5.1 should be straightforwardly consistent with an energy reduction policy in those sectors. The detailed results for the vertex betweenness centrality of energy consumption and a comparison between CO_2 emissions and energy consumption are shown in Tables B.1, B.2, B.5, and B.6.

Table 5.1. Top 30 sectors for vertex betweenness centrality (VB) for CO_2 emissions in the global supply-chain networks associated with the final demand in each demand country

Rank	Demand	Country	Sector	VB (Mt-CO_2)
1	CHN	CHN	Basic Metals and Fabricated Metal	1253
2	CHN	CHN	Electricity, Gas and Water Supply	1089
3	CHN	CHN	Chemicals and Chemical Products	701
4	CHN	CHN	Other Non-Metallic Mineral	605
5	CHN	CHN	Electrical and Optical Equipment	414
6	CHN	CHN	Mining and Quarrying	402
7	CHN	CHN	Machinery, Nec	330
8	CHN	CHN	Rubber and Plastics	234
9	USA	USA	Basic Metals and Fabricated Metal	224
10	CHN	CHN	Transport Equipment	224
11	USA	USA	Renting of M&Eq and Other Business Activities	212
12	IND	IND	Basic Metals and Fabricated Metal	202
13	CHN	CHN	Food, Beverages and Tobacco	202
14	USA	CHN	Basic Metals and Fabricated Metal	194
15	USA	USA	Chemicals and Chemical Products	186
16	CHN	CHN	Agriculture, Hunting, Forestry and Fishing	184
17	CHN	CHN	Pulp, Paper, Paper, Printing and Publishing	180
18	CHN	CHN	Coke, Refined Petroleum and Nuclear Fuel	179
19	CHN	CHN	Textiles and Textile Products	170
20	USA	CHN	Electrical and Optical Equipment	164
21	JPN	JPN	Basic Metals and Fabricated Metal	162
22	USA	USA	Coke, Refined Petroleum and Nuclear Fuel	152
23	CHN	CHN	Renting of M&Eq and Other Business Activities	147
24	IND	IND	Electricity, Gas and Water Supply	141
25	USA	USA	Food, Beverages and Tobacco	135
26	USA	CHN	Electricity, Gas and Water Supply	118
27	USA	USA	Financial Intermediation	117
28	CHN	CHN	Wood and Products of Wood and Cork	113
29	USA	CHN	Chemicals and Chemical Products	110
30	CHN	CHN	Inland Transport	107

Table 5.2. Top 30 sectors of vertex betweenness centrality (VB) for CO_2 emissions in the global supply-chain network associated with the global final demand

Rank	Country	Sector	VB (Mt-CO_2)
1	CHN	Basic Metals and Fabricated Metal	2084
2	CHN	Electricity, Gas and Water Supply	1591
3	CHN	Chemicals and Chemical Products	1171
4	CHN	Electrical and Optical Equipment	1037
5	CHN	Other Non-Metallic Mineral	715
6	CHN	Mining and Quarrying	608
7	CHN	Machinery, Nec	544
8	CHN	Rubber and Plastics	439
9	CHN	Textiles and Textile Products	400
10	CHN	Transport Equipment	356
11	USA	Basic Metals and Fabricated Metal	326
12	JPN	Basic Metals and Fabricated Metal	298
13	IND	Basic Metals and Fabricated Metal	290
14	USA	Chemicals and Chemical Products	285
15	CHN	Coke, Refined Petroleum and Nuclear Fuel	275
16	CHN	Food, Beverages and Tobacco	260
17	CHN	Pulp, Paper, Paper, Printing and Publishing	259
18	USA	Renting of M & Eq and Other Business Activities	248
19	CHN	Agriculture, Hunting, Forestry and Fishing	246
20	CHN	Renting of M & Eq and Other Business Activities	241
21	KOR	Basic Metals and Fabricated Metal	239
22	RUS	Basic Metals and Fabricated Metal	195
23	USA	Coke, Refined Petroleum and Nuclear Fuel	192
24	IND	Electricity, Gas and Water Supply	177
25	RUS	Coke, Refined Petroleum and Nuclear Fuel	166
26	CHN	Wood and Products of Wood and Cork	161
27	CHN	Wholesale Trade and Commission Trade	158
28	USA	Food, Beverages and Tobacco	154
29	CHN	Inland Transport	152
30	RUS	Mining and Quarrying	140

Table 5.3. Top 30 transactions for edge betweenness centrality (EB) for CO_2 emissions in the global supply-chain networks associated with the final demand in each demand country

Rank	Demand	Country	Upstream sector		Country	Downstream sector	EB (Mt-CO_2)
1	CHN	CHN	Other Non-Metallic Mineral (4)	→	CHN	Construction (49)	845
2	CHN	CHN	Basic Metals and Fabricated Metal (1)	→	CHN	Construction (49)	422
3	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Basic Metals and Fabricated Metal (1)	369
4	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Chemicals and Chemical Products (3)	251
5	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Mining and Quarrying (6)	228
6	CHN	CHN	Basic Metals and Fabricated Metal (1)	→	CHN	Machinery, Nec (7)	200
7	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Other Non-Metallic Mineral (4)	197
8	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Construction (49)	164
9	CHN	CHN	Basic Metals and Fabricated Metal (1)	→	CHN	Electrical and Optical Equipment (5)	152
10	USA	USA	Electricity, Gas and Water Supply (71)	→	USA	Public Admin and Defence(121)	138
11	CHN	CHN	Agriculture, Hunting, Forestry and Fishing (16)	→	CHN	Food, Beverages and Tobacco (13)	128
12	IND	IND	Other Non-Metallic Mineral (43)	→	IND	Construction (50)	125
13	CHN	CHN	Mining and Quarrying (6)	→	CHN	Basic Metals and Fabricated Metal (1)	122
14	CHN	CHN	Chemicals and Chemical Products (3)	→	CHN	Health and Social Work (125)	111
15	IND	IND	Electricity, Gas and Water Supply (24)	→	IND	Basic Metals and Fabricated Metal (12)	99
16	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Machinery, Nec (7)	98
17	CHN	CHN	Chemicals and Chemical Products (3)	→	CHN	Rubber and Plastics (8)	98
18	IND	IND	Basic Metals and Fabricated Metal (12)	→	IND	Construction (50)	97
19	CHN	CHN	Mining and Quarrying (6)	→	CHN	Coke, Refined Petroleum and Nuclear Fuel (18)	95
20	CHN	CHN	Basic Metals and Fabricated Metal (1)	→	CHN	Transport Equipment (10)	94
21	USA	USA	Other Non-Metallic Mineral (47)	→	USA	Construction (52)	90
22	USA	USA	Mining and Quarrying (33)	→	USA	Coke, Refined Petroleum and Nuclear Fuel (22)	88
23	RUS	RUS	Other Non-Metallic Mineral (61)	→	RUS	Construction (216)	82
24	USA	USA	Agriculture, Hunting, Forestry and Fishing (37)	→	USA	Food, Beverages and Tobacco (25)	81
25	USA	USA	Electricity, Gas and Water Supply (71)	→	USA	Food, Beverages and Tobacco (25)	79
26	USA	USA	Electricity, Gas and Water Supply (71)	→	USA	Chemicals and Chemical Products (15)	77
27	USA	USA	Electricity, Gas and Water Supply (71)	→	USA	Hotels and Restaurants (74)	76
28	CHN	CHN	Mining and Quarrying (6)	→	CHN	Electricity, Gas and Water Supply (2)	75
29	CHN	CHN	Mining and Quarrying (6)	→	CHN	Other Non-Metallic Mineral (4)	74
30	CHN	CHN	Electricity, Gas and Water Supply (2)	→	CHN	Education (130)	70

Table 5.4. Top 30 transactions of edge betweenness centrality (EB) for CO_2 emissions in the global supply-chain network associated with the global final demand

Rank	Country	Upstream sector		Country	Downstream sector	EB (Mt-CO_2)
1	CHN	Other Non-Metallic Mineral	→	CHN	Construction	853
2	CHN	Electricity, Gas and Water Supply	→	CHN	Basic Metals and Fabricated Metal	610
3	CHN	Basic Metals and Fabricated Metal	→	CHN	Construction	425
4	CHN	Electricity, Gas and Water Supply	→	CHN	Chemicals and Chemical Products	419
5	CHN	Basic Metals and Fabricated Metal	→	CHN	Electrical and Optical Equipment	396
6	CHN	Electricity, Gas and Water Supply	→	CHN	Mining and Quarrying	343
7	CHN	Basic Metals and Fabricated Metal	→	CHN	Machinery, Nec	312
8	CHN	Electricity, Gas and Water Supply	→	CHN	Other Non-Metallic Mineral	235
9	CHN	Mining and Quarrying	→	CHN	Basic Metals and Fabricated Metal	202
10	CHN	Chemicals and Chemical Products	→	CHN	Rubber and Plastics	187
11	CHN	Electricity, Gas and Water Supply	→	CHN	Construction	166
12	CHN	Electricity, Gas and Water Supply	→	CHN	Machinery, Nec	153
13	CHN	Electricity, Gas and Water Supply	→	CHN	Electrical and Optical Equipment	153
14	CHN	Agriculture, Hunting, Forestry and Fishing	→	CHN	Food, Beverages and Tobacco	152
15	CHN	Mining and Quarrying	→	CHN	Coke, Refined Petroleum and Nuclear Fuel	146
16	USA	Electricity, Gas and Water Supply	→	USA	Public Admin and Defence; Compulsory Social Security	139
17	IND	Electricity, Gas and Water Supply	→	IND	Basic Metals and Fabricated Metal	136
18	CHN	Basic Metals and Fabricated Metal	→	CHN	Transport Equipment	129
19	IND	Other Non-Metallic Mineral	→	IND	Construction	129
20	CHN	Chemicals and Chemical Products	→	CHN	Health and Social Work	116
21	USA	Electricity, Gas and Water Supply	→	USA	Chemicals and Chemical Products	111
22	CHN	Mining and Quarrying	→	CHN	Electricity, Gas and Water Supply	108
23	USA	Mining and Quarrying	→	USA	Coke, Refined Petroleum and Nuclear Fuel	107
24	IND	Basic Metals and Fabricated Metal	→	IND	Construction	100
25	CHN	Electricity, Gas and Water Supply	→	CHN	Textiles and Textile Products	99
26	CHN	Mining and Quarrying	→	CHN	Chemicals and Chemical Products	92
27	USA	Other Non-Metallic Mineral	→	USA	Construction	91
28	USA	Agriculture, Hunting, Forestry and Fishing	→	USA	Food, Beverages and Tobacco	90
29	RUS	Electricity, Gas and Water Supply	→	RUS	Mining and Quarrying	89
30	CHN	Mining and Quarrying	→	CHN	Other Non-Metallic Mineral	89

5.5.3 Edge betweenness centrality in global supply-chain networks

The vertex betweenness centrality as discussed above is an indicator that focuses on the centrality of sectors, but it does not take into consideration the role played by transactions between two different sectors in global supply-chain networks. For this reason, for segments of high vertex betweenness centrality, implementing supply-chain management practices that focus on the environmental burden associated with all the raw materials from the upstream sectors needed for manufacturing products is very difficult. Since the edge betweenness centrality formulated as Eq. (5.9) in Section 5.3 makes it possible to identify transactions between sectors through which large amounts of CO_2 emissions flow, promoting technological and economic cooperation between sectors of high edge betweenness centrality has the potential to effectively reduce emissions for entire supply chains.

Table 5.3 summarizes the results for the top 30 transactions in terms of the edge betweenness centrality in global supply-chain networks associated with the final demand in each demand country. The numbers in brackets in Table 5.3 are the ranks in terms of the vertex betweenness centrality shown in Table 5.1. We also show the top 30 transactions in terms of the edge betweenness centrality in global supply-chain networks associated with the global final demand in Table 5.4.

In Table 5.3, it is shown that the edge betweenness centrality of domestic transactions associated with the final demand in the same country is high, in particular, the edge betweenness centrality of transactions within China associated with the final demand in China is high. This demonstrates that the majority of CO_2 arising from the final demand in the same country is not emitted abroad in global supply chains via sectors that have a high value of edge betweenness centrality.

The comparison with the vertex betweenness centrality shows that the vertex betweenness centrality of at least one of the endpoints of a transaction with the high edge betweenness centrality also tends to be high (Table 5.3). If the vertex betweenness centrality of both the upstream sector u and the downstream sector v in a particular transaction (u, v) is high, a large amount of CO_2 is emitted from the sectors upstream of upstream sector u associated with the final demands of sectors downstream of downstream sector v.

For a transaction from the U.S.A.'s Electricity, Gas and Water Supply sector to its Public Admin and Defence; Compulsory Social Security sector, the edge betweenness centrality is high, despite the fact that the vertex betweenness centrality at both endpoints is low (Table 5.3). The reason is

that a high proportion of the CO_2 emissions associated with these transactions are higher primary ripple effect ($f_u a_{uv} y_v$) CO_2 emissions arising from upstream sectors (e.g., U.S.A.'s Electricity, Gas and Water Supply) associated with the final demands of downstream sectors (e.g., U.S.A.'s Public Admin and Defence; Compulsory Social Security). For these kinds of transactions, characterized by sectors of low vertex betweenness centrality and high edge betweenness centrality, emission reduction measures focused on the particular transaction do not contribute much to limiting emissions over the entire global supply chain. Therefore, emission reduction efforts need to be focused on the relevant sectors themselves in these cases.

In order to view the relationship between CO_2 emissions and energy consumption in terms of the edge betweenness centrality, we computed the Spearman's rank correlation of edge betweenness indices of CO_2 emissions and energy consumption associated with the final demand in each demand country and found it to be 0.998. This implies that a very high rank correlation of CO_2 emissions and energy consumption also supports emission reduction efforts being made by effective energy reduction measures in the relevant high-priority sectors indicated in Table 5.1 and the relevant high-priority transactions in Table 5.3. The detailed results for the edge betweenness centrality of energy consumption and a comparison with CO_2 emissions are shown in Tables B.3, B.4, B.7, and B.8.

Next, we examine the edge betweenness centrality matrix $\mathbf{B} = (b_{uv})$, composed of the edge betweenness centrality values b_{uv} of each transaction (u, v). Figures 5.6 and 5.7 illustrate the networks consisting of the top 100 transactions in terms of the edge betweenness centrality for CO_2 emissions in global supply chains associated with the final demands in the U.S.A. and Germany, both of which have high emissions transfers, as shown in Figure 5.4. In the global supply-chain networks shown in these two figures, a higher vertex betweenness centrality is indicated by a larger circle, while a higher edge betweenness centrality is indicated by a thicker edge. Both in the case of the U.S.A. and Germany, the edge betweenness centrality of transactions that have Electricity, Gas and Water Supply as an upstream sector is high, which means that this sector tends to spread CO_2 emissions to domestic sectors (Figures 5.6 and 5.7). In addition, since the vertex betweenness centrality of the Electricity, Gas and Water Supply sector is low, the quantity of CO_2 emissions indirectly generated via this sector is low; it generates CO_2 itself.

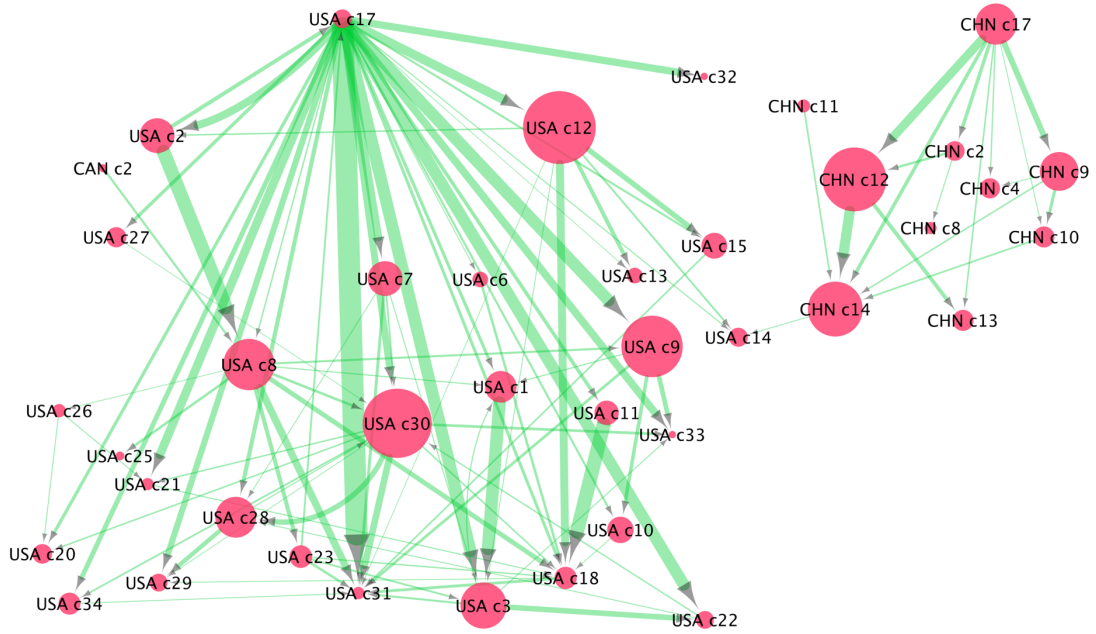

Figure 5.6. High-priority supply-chain network with higher edge betweenness centrality for CO_2 emissions associated with the final demand in the U.S.A. (circle size indicates amount of vertex betweenness centrality and edge width indicates amount of edge betweenness centrality)

If we look at the important supply-chain network graphs for global warming alleviation, that is, having the strong edge betweenness centrality as shown in Figures 5.6 and 5.7, we find that although the final demands in the U.S.A. and Germany induce a huge amount of CO_2 emissions in China, they hardly induce cross-border emissions outside of China. However, we can confirm that CO_2 generated within China by the final demands in the U.S.A. and Germany flows outside the country through global supply chains. If we look at China's Electrical and Optical Equipment sector, which has the high vertex betweenness centrality, we note that the large amount of CO_2 emitted by China's Electricity, Gas and Water Supply sector associated with the final demand in the U.S.A. flows into and concentrates in this Electrical and Optical Equipment sector via China's Basic Metals and Fabricated Metal sector (Figure 5.6). Furthermore, a large amount of CO_2 flows into the Electrical and Optical Equipment sector outside of China via China's Electrical and Optical Equipment sector (Figure 5.6). A similar observation can be made in the case of CO_2 associated with the final demand in Germany (Figure 5.7). Accordingly, China's Electrical and Optical Equipment sector, which this study's centrality analysis has identified as having the high vertex betweenness centrality, is a decisively important sector for reducing CO_2 emissions across entire global supply chains. In addition, we can see that formulating measures to promote cooperation across a specific

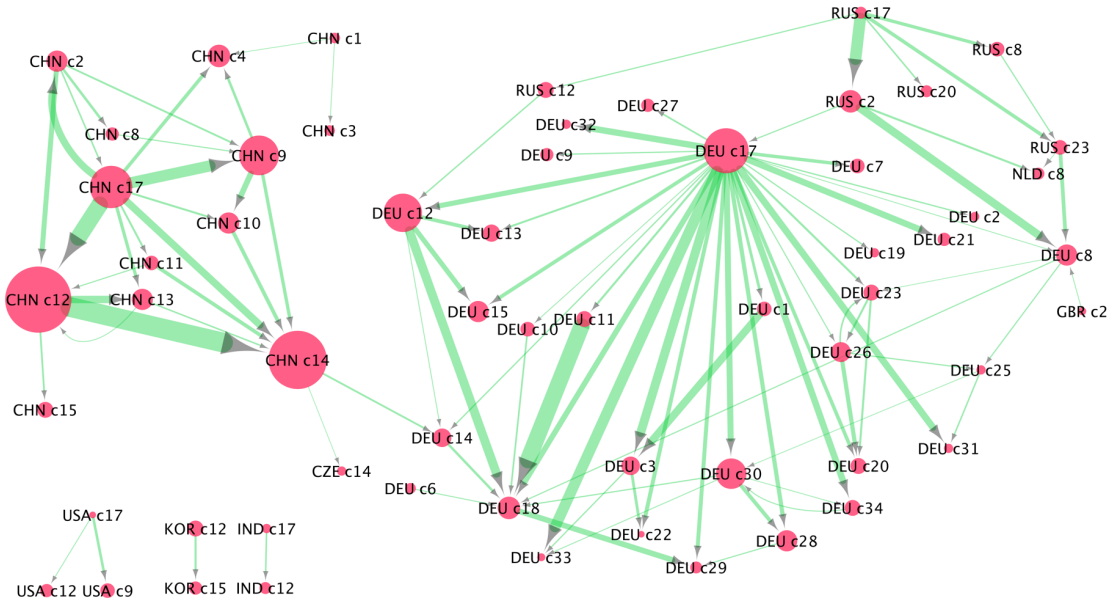

Figure 5.7. High-priority supply-chain network with higher edge betweenness centrality for CO_2 emissions associated with the final demand in Germany (circle size indicates amount of vertex betweenness centrality and edge width indicates amount of edge betweenness centrality)

group of transactions having the high edge betweenness centrality — China's Electricity, Gas and Water Supply → China's Basic Metals and Fabricated Metal sector → China's Electrical and Optical Equipment sector → U.S.A.'s Electrical and Optical Equipment sector → U.S.A. final demand — has the potential for effectively reducing CO_2 emissions across global supply chains.

5.6 Discussion and conclusions

In this study, we identified key sectors and key transactions for effectively reducing CO_2 emissions associated with global supply-chain networks by applying the concept of vertex betweenness centrality proposed by Liang et al. [75] together with the concept of edge betweenness centrality proposed in this study, using the environmentally extended multi-regional input-output table covering 35 industrial sectors and 41 countries and regions in 2008 [24, 109].

Notably, the results of this study demonstrate that emission transfers to China associated with the final demands in Europe and North America are high, and that if these regions cooperate with China on environmentally friendly supply chain management, substantial emission reductions can be effectively achieved. More specifically, China's Electrical and Optical Equipment sector, which

has the high vertex betweenness centrality, is the most instrumental sector in global supply-chain networks in terms of the reduction of CO_2 emissions. Due to its central role in global supply-chain networks associated with the final demand of countries except for China, it is a key sector for global supply-chain management to focus on. Therefore, we could effectively reduce the total CO_2 emissions of global supply-chain networks by achieving procurement of low-carbon raw materials that focuses on China's Electrical and Optical Equipment sector and other high-priority sectors identified in Table 5.1.

For one particular supply-chain path that we identified consisting of transactions having the high edge betweenness centrality — China's Electricity, Gas and Water Supply $\rightarrow$ China's Basic Metals and Fabricated Metal sector $\rightarrow$ China's Electrical and Optical Equipment sector $\rightarrow$ U.S.A.'s Electrical and Optical Equipment sector $\rightarrow$ U.S.A. final demand — we propose to adopt a style in which high-priority product supply chains are emphasized in reporting life-cycle CO_2 emissions of companies. When raw materials are procured so that all involved companies adopt more environmentally friendly practices for raw material procurement and product design, very substantial progress can be made toward global warming mitigation.

Chapter 6

Conclusion

In this thesis, we studied the graph optimization approach and the network indicator approach in terms of economic structure analysis.

As for the graph optimization approach, we proposed new graph optimization problems considering important characteristics of economic networks and designed their high-performance algorithms. In Chapter 3, we studied covering and domination problems on directed graphs, called DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET. We first showed that DIRECTED r-IN (OUT) VERTEX COVER and DIRECTED (p, q)-EDGE DOMINATING SET are NP-complete on restricted graphs. We then proved that if r is larger than one, DIRECTED r-IN (OUT) VERTEX COVER is $W[2]$-hard and if p or q is larger than one, DIRECTED (p, q)-EDGE DOMINATING SET is $W[2]$-hard on directed acyclic graphs. For these problems, we also showed that there is no polynomial-time $c \ln k$-approximation algorithm for any constant $c < 1$ unless P=NP, where k is the size of an optimal solution, though they can be approximated within ratio $O(\log n)$ by a greedy algorithm. On the other hand, we designed polynomial time algorithms on trees. Moreover, we showed that DIRECTED (p, q)-EDGE DOMINATING SET is fixed-parameter tractable with respect to treewidth when $(p, q) = (0, 1), (1, 0), (1, 1)$. We finally gave a $2^{O(k)}n$-time algorithm for DIRECTED (p, q)-EDGE DOMINATING SET when $(p, q) = (0, 1), (1, 0), (1, 1)$ where k is the solution size. This implies that DIRECTED (p, q)-EDGE DOMINATING SET is fixed parameter tractable with respect to k when $(p, q) = (0, 1), (1, 0), (1, 1)$.

In Chapter 4, we defined MAXIMUM WEIGHT MINIMAL $(s$-$t)$ SEPARATOR. We first showed MAXIMUM WEIGHT MINIMAL $(s$-$t)$ SEPARATOR is NP-hard even on unweighted bipartite graphs.

On the other hand, we showed that MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR is fixed-parameter tractable with respect to the weight of the solution. We then designed an $\omega^{O(\omega)}n^{O(1)}$-time deterministic algorithm, where ω is the width of a tree decomposition. This implies MAXIMUM WEIGHT MINIMAL (s-t) SEPARATOR is fixed-parameter tractable with respect to treewidth. However, this algorithm is not a single exponential algorithm. To improve the running time, we designed a $9^{\omega}W^2n^{O(1)}$-time randomized algorithm based on the Cut & Count.

In Chapter 5, we studied the network indicator approach. We first proposed the edge betweenness centrality in input-output analysis. The edge betweenness centrality is a network indicator for a transaction whereas the vertex betweenness centrality proposed in [75] is a network indicator for a sector. These indicators consider the economic ripple effect and reflect the structures of supply chains. We also showed that the relationship between the edge betweenness centrality and the vertex betweenness centrality. The edge betweenness centrality also enabled us to observe the connections between sectors by visualizing the network based on it.

By applying them to the environmentally extended multi-regional input-output table covering 35 industrial sectors and 41 countries and regions in 2008 [24, 109], we identified key sectors and key transactions for effectively reducing CO_2 emissions associated with global supply-chain networks. Moreover, we visualized the CO_2 circulation networks in global supply chains based on the vertex and edge betweenness centralities. From the results, we claimed the total CO_2 emissions of global supply-chain networks could be effectively reduced by achieving procurement of low-carbon raw materials that focuses on China's Electrical and Optical Equipment sector and the other high-priority sectors. We also proposed to adopt a style in which high-priority product supply chains are emphasized in reporting life-cycle CO_2 emissions of companies for one particular supply-chain path identified by the edge betweenness centrality.

Through this thesis, we studied new analytical methods for economic structure analysis via economic networks. With the globalization of economic activity, the importance of these types of analytical methods definitely gets larger and larger.

In the graph optimization approach, we modeled the tasks to discover the important structures in economic networks as graph optimization problems and designed algorithms for them; our work gives a mathematical interpretation of an analytical method for economic network, which is a contribution in the field of economic structure analysis. Furthermore, our graph modeling demonstrates

that graph optimization with direction and/or weight are well motivated; it might open up new research topics in the field of computer science. In fact, graph optimization problems that we investigated in this thesis are one with direction and one with weight; one with both direction and weight is left as important future work. Of course, graph optimization for some other topological structure that reflects important economic relations should be studied also.

As for the network indicator approach, we proposed a new indicator that considers the economic ripple effect. However, there are still few indicators specialized on economic networks. Therefore, the further development of economic network indicators is desired.

Finally, combining the graph optimization approach and the network indicator approach would further expand the field of economic network analysis, for example, applying graph algorithms to the economic network based on the edge betweenness centrality.

Appendix A

Information of World Input-Output Database

Table A.1. Industry classification of the World Input-Output Table (WIOT)

Industry code	Industry name
c1	Agriculture, Hunting, Forestry and Fishing
c2	Mining and Quarrying
c3	Food, Beverages and Tobacco
c4	Textiles and Textile Products
c5	Leather, Leather and Footwear
c6	Wood and Products of Wood and Cork
c7	Pulp, Paper, Paper , Printing and Publishing
c8	Coke, Refined Petroleum and Nuclear Fuel
c9	Chemicals and Chemical Products
c10	Rubber and Plastics
c11	Other Non-Metallic Mineral
c12	Basic Metals and Fabricated Metal
c13	Machinery, Nec
c14	Electrical and Optical Equipment
c15	Transport Equipment
c16	Manufacturing, Nec; Recycling
c17	Electricity, Gas and Water Supply
c18	Construction
c19	Sale, Maintenance and Repair of Motor Vehicles and Motorcycles; Retail Sale of Fuel
c20	Wholesale Trade and Commission Trade, Except of Motor Vehicles and Motorcycles
c21	Retail Trade, Except of Motor Vehicles and Motorcycles; Repair of Household Goods
c22	Hotels and Restaurants
c23	Inland Transport
c24	Water Transport
c25	Air Transport
c26	Other Supporting and Auxiliary Transport Activities; Activities of Travel Agencies
c27	Post and Telecommunications
c28	Financial Intermediation
c29	Real Estate Activities
c30	Renting of M&Eq and Other Business Activities
c31	Public Admin and Defence; Compulsory Social Security
c32	Education
c33	Health and Social Work
c34	Other Community, Social and Personal Services
c35	Private Households with Employed Persons

Table A.2. Country classification of the World Input-Output Table (WIOT)

Country code	Country name
AUS	Australia
AUT	Austria
BEL	Belgium
BGR	Bulgaria
BRA	Brazil
CAN	Canada
CHN	China
CYP	Cyprus
CZE	Czech Republic
DEU	Germany
DNK	Denmark
ESP	Spain
EST	Estonia
FIN	Finland
FRA	France
GBR	United Kingdom
GRC	Greece
HUN	Hungary
IDN	Indonesia
IND	India
IRL	Ireland
ITA	Italy
JPN	Japan
KOR	Korea
LTU	Lithuania
LUX	Luxembourg
LVA	Latvia
MEX	Mexico
MLT	Malta
NLD	Netherlands
POL	Poland
PRT	Portugal
ROM	Romania
RUS	Russia
SVK	Slovak Republic
SVN	Slovenia
SWE	Sweden
TUR	Turkey
TWN	Taiwan
USA	United States
RoW	Rest of the world

Appendix B

Comparison Between CO_2 Emission and Energy Consumption Based on the Betweenness Centrality

Table B.1. Top 30 sectors of vertex betweenness centrality (VB) for energy consumptions in the global supply-chain network associated with the global final demand

Rank	Country	Sector	VB (PJ)
1	CHN	Basic Metals and Fabricated Metal	27794
2	CHN	Electricity, Gas and Water Supply	20188
3	CHN	Chemicals and Chemical Products	15872
4	CHN	Electrical and Optical Equipment	14332
5	CHN	Other Non-Metallic Mineral	8931
6	CHN	Mining and Quarrying	7939
7	CHN	Machinery, Nec	7356
8	CHN	Rubber and Plastics	6110
9	CHN	Textiles and Textile Products	5752
10	USA	Basic Metals and Fabricated Metal	5444
11	USA	Chemicals and Chemical Products	5194
12	JPN	Basic Metals and Fabricated Metal	4992
13	CHN	Transport Equipment	4891
14	USA	Renting of M&Eq and Other Business Activities	4526
15	CHN	Coke, Refined Petroleum and Nuclear Fuel	3915
16	KOR	Basic Metals and Fabricated Metal	3787
17	IND	Basic Metals and Fabricated Metal	3753
18	CHN	Pulp, Paper, Paper , Printing and Publishing	3655
19	CHN	Food, Beverages and Tobacco	3638
20	CHN	Agriculture, Hunting, Forestry and Fishing	3425
21	USA	Coke, Refined Petroleum and Nuclear Fuel	3386
22	CHN	Renting of M&Eq and Other Business Activities	3370
23	RUS	Basic Metals and Fabricated Metal	3345
24	RUS	Coke, Refined Petroleum and Nuclear Fuel	3279
25	USA	Food, Beverages and Tobacco	2853
26	RUS	Mining and Quarrying	2769
27	USA	Financial Intermediation	2606
28	USA	Pulp, Paper, Paper , Printing and Publishing	2388
29	IND	Electricity, Gas and Water Supply	2315
30	CHN	Wood and Products of Wood and Cork	2242

Table B.2. Top 30 sectors of vertex betweenness centrality (VB) for energy consumptions in the global supply-chain networks associated with the final demand in each demand country

Rank	Demand	Country	Sector	VB (PJ)
1	CHN	CHN	Basic Metals and Fabricated Metal	16713
2	CHN	CHN	Electricity, Gas and Water Supply	13817
3	CHN	CHN	Chemicals and Chemical Products	9505
4	CHN	CHN	Other Non-Metallic Mineral	7561
5	CHN	CHN	Electrical and Optical Equipment	5724
6	CHN	CHN	Mining and Quarrying	5244
7	CHN	CHN	Machinery, Nec	4471
8	USA	USA	Renting of M&Eq and Other Business Activities	3879
9	USA	USA	Basic Metals and Fabricated Metal	3750
10	USA	USA	Chemicals and Chemical Products	3386
11	CHN	CHN	Rubber and Plastics	3257
12	CHN	CHN	Transport Equipment	3084
13	CHN	CHN	Food, Beverages and Tobacco	2819
14	JPN	JPN	Basic Metals and Fabricated Metal	2707
15	USA	USA	Coke, Refined Petroleum and Nuclear Fuel	2678
16	IND	IND	Basic Metals and Fabricated Metal	2615
17	USA	CHN	Basic Metals and Fabricated Metal	2594
18	CHN	CHN	Agriculture, Hunting, Forestry and Fishing	2560
19	CHN	CHN	Coke, Refined Petroleum and Nuclear Fuel	2540
20	CHN	CHN	Pulp, Paper, Paper , Printing and Publishing	2538
21	USA	USA	Food, Beverages and Tobacco	2505
22	CHN	CHN	Textiles and Textile Products	2442
23	USA	CHN	Electrical and Optical Equipment	2271
24	USA	USA	Financial Intermediation	2206
25	CHN	CHN	Renting of M&Eq and Other Business Activities	2059
26	USA	USA	Pulp, Paper, Paper , Printing and Publishing	1921
27	RUS	RUS	Coke, Refined Petroleum and Nuclear Fuel	1920
28	IND	IND	Electricity, Gas and Water Supply	1848
29	RUS	RUS	Basic Metals and Fabricated Metal	1753
30	USA	USA	Mining and Quarrying	1698

Table B.3. Top 30 transactions of edge betweenness centrality (EB) for energy consumptions in the global supply-chain network associated with the global final demand

Rank	Country	Upstream sector		Country	Downstream sector	EB (PJ)
1	CHN	Other Non-Metallic Mineral	→	CHN	Construction	9043
2	CHN	Electricity, Gas and Water Supply	→	CHN	Basic Metals and Fabricated Metal	7605
3	CHN	Basic Metals and Fabricated Metal	→	CHN	Construction	5756
4	CHN	Basic Metals and Fabricated Metal	→	CHN	Electrical and Optical Equipment	5356
5	CHN	Electricity, Gas and Water Supply	→	CHN	Chemicals and Chemical Products	5216
6	CHN	Electricity, Gas and Water Supply	→	CHN	Mining and Quarrying	4278
7	CHN	Basic Metals and Fabricated Metal	→	CHN	Machinery, Nec	4219
8	CHN	Electricity, Gas and Water Supply	→	CHN	Other Non-Metallic Mineral	2929
9	CHN	Mining and Quarrying	→	CHN	Basic Metals and Fabricated Metal	2787
10	CHN	Chemicals and Chemical Products	→	CHN	Rubber and Plastics	2602
11	USA	Electricity, Gas and Water Supply	→	USA	Public Admin and Defence; Compulsory Social Security	2540
12	CHN	Agriculture, Hunting, Forestry and Fishing	→	CHN	Food, Beverages and Tobacco	2181
13	CHN	Electricity, Gas and Water Supply	→	CHN	Construction	2065
14	USA	Electricity, Gas and Water Supply	→	USA	Chemicals and Chemical Products	2020
15	CHN	Mining and Quarrying	→	CHN	Coke, Refined Petroleum and Nuclear Fuel	2014
16	CHN	Electricity, Gas and Water Supply	→	CHN	Machinery, Nec	1912
17	CHN	Electricity, Gas and Water Supply	→	CHN	Electrical and Optical Equipment	1908
18	USA	Mining and Quarrying	→	USA	Coke, Refined Petroleum and Nuclear Fuel	1834
19	RUS	Electricity, Gas and Water Supply	→	RUS	Mining and Quarrying	1801
20	IND	Electricity, Gas and Water Supply	→	IND	Basic Metals and Fabricated Metal	1779
21	CHN	Basic Metals and Fabricated Metal	→	CHN	Transport Equipment	1744
22	RUS	Electricity, Gas and Water Supply	→	RUS	Coke, Refined Petroleum and Nuclear Fuel	1720
23	USA	Agriculture, Hunting, Forestry and Fishing	→	USA	Food, Beverages and Tobacco	1702
24	CHN	Chemicals and Chemical Products	→	CHN	Health and Social Work	1607
25	USA	Electricity, Gas and Water Supply	→	USA	Food, Beverages and Tobacco	1594
26	USA	Electricity, Gas and Water Supply	→	USA	Basic Metals and Fabricated Metal	1584
27	CHN	Mining and Quarrying	→	CHN	Electricity, Gas and Water Supply	1482
28	USA	Electricity, Gas and Water Supply	→	USA	Hotels and Restaurants	1415
29	CHN	Mining and Quarrying	→	CHN	Chemicals and Chemical Products	1270
30	IND	Basic Metals and Fabricated Metal	→	IND	Construction	1257

Table B.4. Top 30 transactions of edge betweenness centrality (EB) for energy consumptions in the global supply-chain networks associated with the final demand in each demand country

Rank	Demand	Country	Upstream sector		Country	Downstream sector	EB (PJ)
1	CHN	CHN	Other Non-Metallic Mineral	→	CHN	Construction	8962
2	CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Construction	5704
3	CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Basic Metals and Fabricated Metal	4594
4	CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Chemicals and Chemical Products	3124
5	CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Mining and Quarrying	2844
6	CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Machinery, Nec	2700
7	USA	USA	Electricity, Gas and Water Supply	→	USA	Public Admin and Defence; Compulsory Social Security	2514
8	CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Other Non-Metallic Mineral	2457
9	CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Electrical and Optical Equipment	2060
10	CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Construction	2046
11	CHN	CHN	Agriculture, Hunting, Forestry and Fishing	→	CHN	Food, Beverages and Tobacco	1827
12	CHN	CHN	Mining and Quarrying	→	CHN	Basic Metals and Fabricated Metal	1683
13	CHN	CHN	Chemicals and Chemical Products	→	CHN	Health and Social Work	1541
14	USA	USA	Agriculture, Hunting, Forestry and Fishing	→	USA	Food, Beverages and Tobacco	1528
15	USA	USA	Mining and Quarrying	→	USA	Coke, Refined Petroleum and Nuclear Fuel	1513
16	USA	USA	Electricity, Gas and Water Supply	→	USA	Food, Beverages and Tobacco	1431
17	USA	USA	Electricity, Gas and Water Supply	→	USA	Chemicals and Chemical Products	1411
18	USA	USA	Electricity, Gas and Water Supply	→	USA	Hotels and Restaurants	1385
19	CHN	CHN	Chemicals and Chemical Products	→	CHN	Rubber and Plastics	1356
20	CHN	CHN	Mining and Quarrying	→	CHN	Coke, Refined Petroleum and Nuclear Fuel	1304
21	IND	IND	Electricity, Gas and Water Supply	→	IND	Basic Metals and Fabricated Metal	1292
22	CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Transport Equipment	1265
23	IND	IND	Basic Metals and Fabricated Metal	→	IND	Construction	1226
24	CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Machinery, Nec	1223
25	IND	IND	Other Non-Metallic Mineral	→	IND	Construction	1139
26	USA	USA	Electricity, Gas and Water Supply	→	USA	Basic Metals and Fabricated Metal	1089
27	USA	USA	Electricity, Gas and Water Supply	→	USA	Retail Trade, Except of Motor Vehicles and Motorcycles	1040
28	CHN	CHN	Mining and Quarrying	→	CHN	Electricity, Gas and Water Supply	1036
29	USA	USA	Other Non-Metallic Mineral	→	USA	Construction	1031
30	CHN	CHN	Mining and Quarrying	→	CHN	Other Non-Metallic Mineral	1025

Table B.5. The comparison of Top 30 sectors of vertex betweenness centrality between CO_2 emissions and energy consumptions in the global supply-chain network associated with the global final demand

Country	Sector	CO_2	Energy
CHN	Basic Metals and Fabricated Metal	1	1
CHN	Electricity, Gas and Water Supply	2	2
CHN	Chemicals and Chemical Products	3	3
CHN	Electrical and Optical Equipment	4	4
CHN	Other Non-Metallic Mineral	5	5
CHN	Mining and Quarrying	6	6
CHN	Machinery, Nec	7	7
CHN	Rubber and Plastics	8	8
CHN	Textiles and Textile Products	9	9
CHN	Transport Equipment	10	13
USA	Basic Metals and Fabricated Metal	11	10
JPN	Basic Metals and Fabricated Metal	12	12
IND	Basic Metals and Fabricated Metal	13	17
USA	Chemicals and Chemical Products	14	11
CHN	Coke, Refined Petroleum and Nuclear Fuel	15	15
CHN	Food, Beverages and Tobacco	16	19
CHN	Pulp, Paper, Paper , Printing and Publishing	17	18
USA	Renting of M&Eq and Other Business Activities	18	14
CHN	Agriculture, Hunting, Forestry and Fishing	19	20
CHN	Renting of M&Eq and Other Business Activities	20	22
KOR	Basic Metals and Fabricated Metal	21	16
RUS	Basic Metals and Fabricated Metal	22	23
USA	Coke, Refined Petroleum and Nuclear Fuel	23	21
IND	Electricity, Gas and Water Supply	24	29
RUS	Coke, Refined Petroleum and Nuclear Fuel	25	24
CHN	Wood and Products of Wood and Cork	26	30
CHN	Wholesale Trade and Commission Trade	27	33
USA	Food, Beverages and Tobacco	28	25
CHN	Inland Transport	29	35
RUS	Mining and Quarrying	30	26

Table B.6. The comparison of Top 30 sectors of vertex betweenness centrality between CO_2 emissions and energy consumptions in the global supply-chain network associated with the final demand in each demand country

Demand	Country	Sector	CO_2	Energy
CHN	CHN	Basic Metals and Fabricated Metal	1	1
CHN	CHN	Electricity, Gas and Water Supply	2	2
CHN	CHN	Chemicals and Chemical Products	3	3
CHN	CHN	Other Non-Metallic Mineral	4	4
CHN	CHN	Electrical and Optical Equipment	5	5
CHN	CHN	Mining and Quarrying	6	6
CHN	CHN	Machinery, Nec	7	7
CHN	CHN	Rubber and Plastics	8	11
USA	USA	Basic Metals and Fabricated Metal	9	9
CHN	CHN	Transport Equipment	10	12
USA	USA	Renting of M&Eq and Other Business Activities	11	8
IND	IND	Basic Metals and Fabricated Metal	12	16
CHN	CHN	Food, Beverages and Tobacco	13	13
USA	CHN	Basic Metals and Fabricated Metal	14	17
USA	USA	Chemicals and Chemical Products	15	10
CHN	CHN	Agriculture, Hunting, Forestry and Fishing	16	18
CHN	CHN	Pulp, Paper, Paper , Printing and Publishing	17	20
CHN	CHN	Coke, Refined Petroleum and Nuclear Fuel	18	19
CHN	CHN	Textiles and Textile Products	19	22
USA	CHN	Electrical and Optical Equipment	20	23
JPN	JPN	Basic Metals and Fabricated Metal	21	14
USA	USA	Coke, Refined Petroleum and Nuclear Fuel	22	15
CHN	CHN	Renting of M&Eq and Other Business Activities	23	25
IND	IND	Electricity, Gas and Water Supply	24	28
USA	USA	Food, Beverages and Tobacco	25	21
USA	CHN	Electricity, Gas and Water Supply	26	33
USA	USA	Financial Intermediation	27	24
CHN	CHN	Wood and Products of Wood and Cork	28	31
USA	CHN	Chemicals and Chemical Products	29	34
CHN	CHN	Inland Transport	30	36

Table B.7. The comparison of Top 50 transactions of edge betweenness centrality between CO_2 emissions and energy consumptions in the global supply-chain network associated with the global final demand

Country	Upstream sector		Country	Downstream sector	CO_2	Energy
CHN	Other Non-Metallic Mineral	→	CHN	Construction	1	1
CHN	Electricity, Gas and Water Supply	→	CHN	Basic Metals and Fabricated Metal	2	2
CHN	Basic Metals and Fabricated Metal	→	CHN	Construction	3	3
CHN	Electricity, Gas and Water Supply	→	CHN	Chemicals and Chemical Products	4	5
CHN	Basic Metals and Fabricated Metal	→	CHN	Electrical and Optical Equipment	5	4
CHN	Electricity, Gas and Water Supply	→	CHN	Mining and Quarrying	6	6
CHN	Basic Metals and Fabricated Metal	→	CHN	Machinery, Nec	7	7
CHN	Electricity, Gas and Water Supply	→	CHN	Other Non-Metallic Mineral	8	8
CHN	Mining and Quarrying	→	CHN	Basic Metals and Fabricated Metal	9	9
CHN	Chemicals and Chemical Products	→	CHN	Rubber and Plastics	10	10
CHN	Electricity, Gas and Water Supply	→	CHN	Construction	11	13
CHN	Electricity, Gas and Water Supply	→	CHN	Machinery, Nec	12	16
CHN	Electricity, Gas and Water Supply	→	CHN	Electrical and Optical Equipment	13	17
CHN	Agriculture, Hunting, Forestry and Fishing	→	CHN	Food, Beverages and Tobacco	14	12
CHN	Mining and Quarrying	→	CHN	Coke, Refined Petroleum and Nuclear Fuel	15	15
USA	Electricity, Gas and Water Supply	→	USA	Public Admin and Defence; Compulsory Social Security	16	11
IND	Electricity, Gas and Water Supply	→	IND	Basic Metals and Fabricated Metal	17	20
CHN	Basic Metals and Fabricated Metal	→	CHN	Transport Equipment	18	21
IND	Other Non-Metallic Mineral	→	IND	Construction	19	34
CHN	Chemicals and Chemical Products	→	CHN	Health and Social Work	20	24
USA	Electricity, Gas and Water Supply	→	USA	Chemicals and Chemical Products	21	14
CHN	Mining and Quarrying	→	CHN	Electricity, Gas and Water Supply	22	27
USA	Mining and Quarrying	→	USA	Coke, Refined Petroleum and Nuclear Fuel	23	18
IND	Basic Metals and Fabricated Metal	→	IND	Construction	24	30
CHN	Electricity, Gas and Water Supply	→	CHN	Textiles and Textile Products	25	32
CHN	Mining and Quarrying	→	CHN	Chemicals and Chemical Products	26	29
USA	Other Non-Metallic Mineral	→	USA	Construction	27	41
USA	Agriculture, Hunting, Forestry and Fishing	→	USA	Food, Beverages and Tobacco	28	23
RUS	Electricity, Gas and Water Supply	→	RUS	Mining and Quarrying	29	19

Table B.8. The comparison of Top 30 transactions of edge betweenness centrality between CO_2 emissions and energy consumptions in the global supply-chain network associated with the final demand in each demand country

Demand	Country	Upstream sector		Country	Downstream sector	CO_2	Energy
CHN	CHN	Other Non-Metallic Mineral	→	CHN	Construction	1	1
CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Construction	2	2
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Basic Metals and Fabricated Metal	3	3
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Chemicals and Chemical Products	4	4
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Mining and Quarrying	5	5
CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Machinery, Nec	6	6
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Other Non-Metallic Mineral	7	8
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Construction	8	10
CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Electrical and Optical Equipment	9	9
USA	USA	Electricity, Gas and Water Supply	→	USA	Public Admin and Defence; Compulsory Social Security	10	7
CHN	CHN	Agriculture, Hunting, Forestry and Fishing	→	CHN	Food, Beverages and Tobacco	11	11
IND	IND	Other Non-Metallic Mineral	→	IND	Construction	12	25
CHN	CHN	Mining and Quarrying	→	CHN	Basic Metals and Fabricated Metal	13	12
CHN	CHN	Chemicals and Chemical Products	→	CHN	Health and Social Work	14	13
IND	IND	Electricity, Gas and Water Supply	→	IND	Basic Metals and Fabricated Metal	15	21
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Machinery, Nec	16	24
CHN	CHN	Chemicals and Chemical Products	→	CHN	Rubber and Plastics	17	19
IND	IND	Basic Metals and Fabricated Metal	→	IND	Construction	18	23
CHN	CHN	Mining and Quarrying	→	CHN	Coke, Refined Petroleum and Nuclear Fuel	19	20
CHN	CHN	Basic Metals and Fabricated Metal	→	CHN	Transport Equipment	20	22
USA	USA	Other Non-Metallic Mineral	→	USA	Construction	21	29
USA	USA	Mining and Quarrying	→	USA	Coke, Refined Petroleum and Nuclear Fuel	22	15
RUS	RUS	Other Non-Metallic Mineral	→	RUS	Construction	23	32
USA	USA	Agriculture, Hunting, Forestry and Fishing	→	USA	Food, Beverages and Tobacco	24	14
USA	USA	Electricity, Gas and Water Supply	→	USA	Food, Beverages and Tobacco	25	16
USA	USA	Electricity, Gas and Water Supply	→	USA	Chemicals and Chemical Products	26	17
USA	USA	Electricity, Gas and Water Supply	→	USA	Hotels and Restaurants	27	18
CHN	CHN	Mining and Quarrying	→	CHN	Electricity, Gas and Water Supply	28	28
CHN	CHN	Mining and Quarrying	→	CHN	Other Non-Metallic Mineral	29	30
CHN	CHN	Electricity, Gas and Water Supply	→	CHN	Education	30	41

Bibliography

[1] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. doi:10.1137/0608024.

[2] Anne Berry, Jean Paul Bordat, and Olivier Cogis. Generating all the minimal separators of a graph. *International Journal of Foundations of Computer Science*, 11(03):397–403, 2000. doi:10.1142/S0129054100000211.

[3] Alan A. Bertossi. The edge hamiltonian path problem is NP-complete. *Information Processing Letters*, 13(4):157–159, 1981. doi:10.1016/0020-0190(81)90048-X.

[4] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdržálek. The dag-width of directed graphs. *Journal of Combinatorial Theory, Series B*, 102(4):900–923, 2012. doi:10.1016/j.jctb.2012.04.004.

[5] Jacek Blazewicz, Alain Hertz, Daniel Kobler, and Dominique de Werra. On some properties of DNA graphs. *Discrete Applied Mathematics*, 98(1):1–19, 1999. doi:10.1016/S0166-218X(99)00109-2.

[6] Jacek Blazewicz, Marta Kasprzak, Benjamin Leroy-Beaulieu, and Dominique de Werra. Finding hamiltonian circuits in quasi-adjoint graphs. *Discrete Applied Mathematics*, 156(13):2573–2580, 2008. doi:10.1016/j.dam.2008.03.014.

[7] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

[8] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. doi:10.1137/130947374.

[9] Hans L. Bodlaender and Klaus Jansen. On the complexity of the maximum cut problem. *Nordic Journal of Computing*, 7(1):14–31, 2000.

[10] Hans L. Bodlaender, Ton Kloks, and Dieter Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM Journal on Discrete Mathematics*, 8(4):606–616, 1995. doi:10.1137/S089548019223992X.

[11] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[12] Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM Journal on Computing*, 31(1):212–232, 2001. doi:10.1137/S0097539799359683.

[13] Vincent Bouchitté and Ioan Todinca. Listing all potential maximal cliques of a graph. *Theoretical Computer Science*, 276(1):17–32, 2002. doi:10.1016/S0304-3975(01)00007-X.

[14] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001. doi:10.1080/0022250X.2001.9990249.

[15] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P^*. *SIAM Journal on Computing*, 22(3):560–572, 1993. doi:10.1137/0222038.

[16] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *Journal of the ACM*, 63(5):40:1–40:65, 2016. doi:10.1145/2820609.

[17] C. K. Cheng and T. C. Hu. Maximum concurrent flows and minimum cuts. *Algorithmica*, 8(1):233, 1992. doi:10.1007/BF01758845.

[18] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008. doi:10.1016/j.ic.2008.07.003.

[19] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015. doi:10.1007/978-3-319-21275-3.

[20] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011)*, pages 150–159, 2011. doi:10.1109/FOCS.2011.23.

[21] Easley David and Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[22] Anuj Dawar and Stephan Kreutzer. Domination problems in nowhere-dense classes. In *Proceedings of the 29th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009)*, pages 157–168, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2315.

[23] Jacques Defourny and Erik Thorbecke. Structural path analysis and multiplier decomposition within a social accounting matrix framework. *The Economic Journal*, 94(373):111–136, 1984.

[24] Erik Dietzenbacher, Bart Los, Robert Stehrer, Marcel Timmer, and Gaaitzen de Vries. The construction of world input-output tables in the WIOD project. *Economic Systems Research*, 25(1):71–98, 2013. doi:10.1080/09535314.2012.761180.

[25] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.

[26] Chris Ding, Xiaofeng He, and Horst D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 606–610, 2005. doi:10.1137/1.9781611972757.70.

[27] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 624–633, 2014. doi:10.1145/2591796.2591884.

[28] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995. doi:10.1137/S0097539792228228.

[29] Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. In *Feasible Mathematics II*, pages 219–244, 1995. doi:10.1007/978-1-4612-2566-9_7.

[30] Pål Grønås Drange, Markus Dregi, Fedor V. Fomin, Stephan Kreutzer, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, Felix Reidl, Fernando Sánchez Villaamil, Saket Saurabh, Sebastian Siebertz, and Somnath Sikdar. Kernelization and sparseness: the case of dominating set. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 31:1–31:14, 2016. doi:10.4230/LIPIcs.STACS.2016.31.

[31] Henning Fernau. EDGE DOMINATING SET: Efficient enumeration-based exact algorithms. In *Proceedings of the Parameterized and Exact Computation: Second International Workshop (IWPEC 2006)*, pages 142–153, 2006. doi:10.1007/11847250_13.

[32] Jiří Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412(23):2513–2523, 2011. doi:10.1016/j.tcs.2010.10.043.

[33] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag New York, Inc., 2006. doi:10.1007/3-540-29953-X.

[34] Fedor V. Fomin, Dieter Kratsch, Ioan Todinca, and Yngve Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM Journal on Computing*, 38(3):1058–1079, 2008. doi:10.1137/050643350.

[35] Lester R. Ford Jr. and Delbert Ray Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

[36] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. doi:https://doi.org/10.1016/j.physrep.2009.11.002.

[37] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. doi:10.2307/3033543.

[38] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978. doi:10.1016/0378-8733(78)90021-7.

[39] Linton C. Freeman, Douglas Roeder, and Robert R. Mulholland. Centrality in social networks: ii. experimental results. *Social Networks*, 2(2):119–141, 1979. doi:10.1016/0378-8733(79)90002-9.

[40] Robert Ganian, Petr Hliněný, Joachim Kneis, Alexander Langer, Jan Obdržálek, and Peter Rossmanith. Digraph width measures in parameterized algorithmics. *Discrete Applied Mathematics*, 168:88–107, 2014. doi:10.1016/j.dam.2013.10.038.

[41] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[42] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

[43] Serge Gaspers and Simon Mackenzie. On the number of minimal separators in graphs. *CoRR*, abs/1503.01203v2, 2015.

[44] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. doi:10.1073/pnas.122653799.

[45] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[46] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992. doi:10.1109/43.159993.

[47] Jing-Dong J. Han, Nicolas Bertin, Tong Hao, Debra S. Goldberg, Gabriel F. Berriz, Lan V. Zhang, Denis Dupuy, Albertha J. M. Walhout, Michael E. Cusick, Frederick P. Roth, and Marc Vidal. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430:88–93, 2004. doi:10.1038/nature02555.

[48] Tesshu Hanaka, Shigemi Kagawa, Hirotaka Ono, and Keiichiro Kanemoto. Finding environmentally critical transmission sectors, transactions, and paths in global supply chain networks. *Energy Economics*, 68:44–52, 2017. doi:10.1016/j.eneco.2017.09.012.

[49] Frank Harary and Robert Z. Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, 1960. doi:10.1007/BF02854581.

[50] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4):175–181, 2000. doi:10.1016/S0020-0190(00)00142-3.

[51] Bharat R. Hazari. Empirical identification of key sectors in the indian economy. *The Review of Economics and Statistics*, 52(3):301–305, 1970.

[52] Edgar G. Hertwich and Glen P. Peters. Carbon footprint of nations: A global, trade-linked analysis. *Environmental Science & Technology*, 43(16):6414–6420, 2009. doi:10.1021/es803496a.

[53] Albert O. Hirschman. *The Strategy of Economic Development*. Yale University Press, 1958.

[54] Thor Johnson, Neil Robertson, P.D. Seymour, and Robin Thomas. Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82(1):138–154, 2001. doi:10.1006/jctb.2000.2031.

[55] Shigemi Kagawa, Shunsuke Okamoto, Sangwon Suh, Yasushi Kondo, and Keisuke Nansai. Finding environmentally important industry clusters: Multiway cut approach using nonnegative matrix factorization. *Social Networks*, 35(3):423–438, 2013. doi:10.1016/j.socnet.2013.04.009.

[56] Shigemi Kagawa, Sangwon Suh, Klaus Hubacek, Thomas Wiedmann, Keisuke Nansai, and Jan Minx. CO_2 emission clusters within global supply chain networks: Implications for climate change mitigation. *Global Environmental Change*, 35:486–496, 2015. doi:10.1016/j.gloenvcha.2015.04.003.

[57] Shigemi Kagawa, Sangwon Suh, Yasushi Kondo, and Keisuke Nansai. Identifying environmentally important supply chain clusters in the automobile industry. *Economic Systems Research*, 25(3):265–286, 2013. doi:10.1080/09535314.2012.730992.

[58] Keiichiro Kanemoto, Manfred Lenzen, Glen P. Peters, Daniel D. Moran, and Arne Geschke. Frameworks for comparing emissions associated with production, consumption, and international trade. *Environmental Science & Technology*, 46(1):172–179, 2012. doi:10.1021/es202239t.

[59] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

[60] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

[61] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 1994. doi:10.1007/BFb0045375.

[62] Ton Kloks, Dieter Kratsch, and C.K. Wong. Minimum fill-in on circle and circular-arc graphs. *Journal of Algorithms*, 28(2):272–289, 1998. doi:10.1006/jagm.1998.0936.

[63] Yasushi Kondo. Triangulation of input–output tables based on mixed integer programs for inter-temporal and inter-regional comparison of production structures. *Journal of Economic Structures*, 3(1):2, 2014. doi:10.1186/2193-2409-3-2.

[64] Bernhard Korte and Walter Oberhofer. Triangularizing input-output matrices and the structure of production. *European Economic Review*, 2(4):493–522, 1971. doi:10.1016/0014-2921(71)90024-9.

[65] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag Berlin Heidelberg, 5th edition, 2012.

[66] Stephan Kreutzer and Siamak Tazari. Directed nowhere dense classes of graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1552–1562, 2012. doi:10.1137/1.9781611973099.123.

[67] Michael Lahr and Erik Dietzenbacher. *Input-Output Analysis: Frontiers and Extensions*. Palgrave Macmillan UK, 2001.

[68] Manfred Lenzen. Environmentally important paths, linkages and key sectors in the australian economy. *Structural Change and Economic Dynamics*, 14(1):1–34, 2003. doi:10.1016/S0954-349X(02)00025-5.

[69] Manfred Lenzen, Daniel D. Moran, Keiichiro Kanemoto, Barney Foran, Leonarda Lobefaro, and Arne Geschke. International trade drives biodiversity threats in developing nations. *Nature*, 486:109–112, 2012. doi:10.1038/nature11145.

[70] Wassily W. Leontief. Quantitative input and output relations in the economic systems of the united states. *The Review of Economics and Statistics*, 18(3):105–125, 1936.

[71] Wassily W. Leontief. The structure of development. *Scientific American*, 209(3):148–167, 1963. doi:10.1038/scientificamerican0963-148.

[72] Wassily W. Leontief. *Input-Output Economics*. Oxford University Press, 2nd edition, 1986.

[73] Xiaobin Li and Zheng Tian. Optimum cut-based clustering. *Signal Processing*, 87(11):2491–2502, 2007. doi:10.1016/j.sigpro.2007.03.017.

[74] Sai Liang, Yu Feng, and Ming Xu. Structure of the global virtual carbon network: Revealing important sectors and communities for emission reduction. *Journal of Industrial Ecology*, 19(2):307–320, 2015. doi:10.1111/jiec.12242.

[75] Sai Liang, Shen Qu, and Ming Xu. Betweenness-based method to identify critical transmission sectors for supply chain environmental pressure mitigation. *Environmental Science & Technology*, 50(3):1330–1337, 2016. doi:10.1021/acs.est.5b04855.

[76] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

[77] Ronald E. Miller and Peter D. Blair. *Input-Output Analysis: Foundations and Extensions*. Cambridge University Press, 2nd edition, 2009.

[78] Ministry of the Environment Government of Japan. URL: http://www.env.go.jp/earth/ondanka/ghg/2008ghg.pdf.

[79] Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001.

[80] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

[81] Fumiya Nagashima, Shigemi Kagawa, Sangwon Suh, Keisuke Nansai, and Daniel Moran. Identifying critical supply chain paths and key sectors for mitigating primary carbonaceous pm2.5 mortality in asia. *Economic Systems Research*, 29(1):105–123, 2017. doi:10.1080/09535314.2016.1266992.

[82] Shinichiro Nakamura, Yasushi Kondo, Kazuyo Matsubae, Kenichi Nakajima, and Tetsuya Nagasaka. Upiom: A new tool of mfa and its application to the flow of iron and steel associated with car production. *Environmental Science & Technology*, 45(3):1114–1120, 2011. doi:10.1021/es1024299.

[83] National Institute for Environmental Studies (NIES), Japan. 3EID: embodied energy and emission intensity data for japan, 2016. URL: http://www.cger.nies.go.jp/publications/report/d031/eng/index_e.htm.

[84] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004. doi:10.1103/PhysRevE.69.066133.

[85] Mark E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, 2005. doi:10.1016/j.socnet.2004.11.009.

[86] Mark E. J. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[87] Mark E. J. Newman, Albert-Laszlo Barabasi, and Duncan J. Watts. *The Structure and Dynamics of Networks: (Princeton Studies in Complexity)*. Princeton University Press, Princeton, NJ, USA, 2006.

[88] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004. doi:10.1103/PhysRevE.69.026113.

[89] Hajime Ohno, Philip Nuss, Wei-Qiang Chen, and Thomas E. Graedel. Deriving the metal and alloy networks of modern technology. *Environmental Science & Technology*, 50(7):4082–4090, 2016. doi:10.1021/acs.est.5b05093.

[90] Intergovernmental Panel on Climate Change. *Climate Change 2014: Mitigation of Climate Change: Working Group III Contribution to the IPCC Fifth Assessment Report*. Cambridge University Press, 2015. doi:10.1017/CBO9781107415416.

[91] Yuko Oshita. Identifying critical supply chain paths that drive changes in co2 emissions. *Energy Economics*, 34(4):1041–1050, 2012. doi:10.1016/j.eneco.2011.08.013.

[92] Anne Owen, Richard Wood, John Barrett, and Andrew Evans. Explaining value chain differences in mrio databases through structural path decomposition. *Economic Systems Research*, 28(2):243–272, 2016. doi:10.1080/09535314.2015.1135309.

[93] Glen P. Peters. From production-based to consumption-based national emission inventories. *Ecological Economics*, 65(1):13–23, 2008. doi:10.1016/j.ecolecon.2007.10.014.

[94] Glen P. Peters, Jan C. Minx, Christopher L. Weber, and Ottmar Edenhofer. Growth in emission transfers via international trade from 1990 to 2008. *Proceedings of the National Academy of Sciences*, 108(21):8903–8908, 2011. doi:10.1073/pnas.1006388108.

[95] Poul Nørregaard Rasmussen. *Studies in inter-sectoral relations*. PhD thesis, Københavns universitet, 1956.

[96] Omar Rifki, Hirotaka Ono, and Shigemi Kagawa. The robustest clusters in the input–output networks: global CO_2 emission clusters. *Journal of Economic Structures*, 6(1):3, 2017. doi:10.1186/s40008-017-0062-2.

[97] Neil Robertson and P.D Seymour. Graph minors. V. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

[98] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. doi:10.1073/pnas.0706851105.

[99] Jean-François Rual, Kavitha Venkatesan, Tong Hao, Tomoko Hirozane-Kishikawa, Amélie Dricot, Ning Li, Gabriel F. Berriz, Francis D. Gibbons, Matija Dreze, Nono Ayivi-Guedehoussou, Niels Klitgord, Christophe Simon, Mike Boxem, Stuart Milstein, Jennifer Rosenberg, Debra S. Goldberg, Lan V. Zhang, Sharyl L. Wong, Giovanni Franklin, Siming Li, Joanna S. Albala, Janghoo Lim, Carlene Fraughton, Estelle Llamosas, Sebiha Cevik, Camille Bex, Philippe Lamesch, Robert S. Sikorski, Jean Vandenhaute, Huda Y. Zoghbi, Alex Smolyar, Stephanie Bosak, Reynaldo Sequerra, Lynn Doucette-Stamm, Michael E. Cusick, David E. Hill, Frederick P. Roth, and Marc Vidal. Towards a proteome-scale map of the human protein-protein interaction network. *Nature*, 437:1173–1178, 2005. doi:10.1038/nature04209.

[100] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966. doi:10.1007/BF02289527.

[101] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. doi:10.1016/j.cosrev.2007.05.001.

[102] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi:10.1109/34.868688.

[103] Alfonso Shimbel. Structural parameters of communication networks. *The bulletin of mathematical biophysics*, 15(4):501–507, 1953. doi:10.1007/BF02476438.

[104] Jiří Šíma and Satu Elisa Schaeffer. On the np-completeness of some graph cluster measures. In *Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2006)*, pages 530–537, 2006. doi:10.1007/11611257_51.

[105] Konstantin Skodinis. Efficient analysis of graphs with small minimal separators. In *Proceedings of the Graph-Theoretic Concepts in Computer Science, 25th International Workshop (WG 1999)*, pages 155–166, 1999. doi:10.1007/3-540-46784-X_16.

[106] Karen Stephenson and Marvin Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, 1989. doi:10.1016/0378-8733(89)90016-6.

[107] Karol Suchan. Minimal separators in intersection graphs. Master's thesis, Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, 2003.

[108] Jan Minx Thomas Wiedmann. A definition of 'carbon footprint'. In Carolyn C. Pertsova, editor, *Ecological Economics Research Trends*, volume 2, chapter 1, pages 1–11. Nova Science Publishers, Inc., 2008.

[109] Marcel P. Timmer, Erik Dietzenbacher, Bart Los, Robert Stehrer, and Gaaitzen J. de Vries. An illustrated user guide to the world input-output database: the case of global automotive production. *Review of International Economics*, 23(3):575–605, 2015. doi:10.1111/roie.12178.

[110] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Proceedings of the17th Annual European Symposium on Algorithms (ESA 2009)*, pages 566–577, 2009. doi:10.1007/978-3-642-04128-0_51.

[111] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag Berlin Heidelberg, 2003. doi:10.1007/978-3-662-04565-7.

[112] Frederick V. Waugh. Inversion of the leontief matrix by power series. *Econometrica*, 18(2):142–154, 1950.

[113] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

[114] Richard Wood and Manfred Lenzen. Structural path decomposition. *Energy Economics*, 31(3):335–341, 2009. doi:10.1016/j.eneco.2008.11.003.

[115] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993. doi:10.1109/34.244673.

[116] Mihalis Yannakakis and Fanica Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980. doi:10.1137/0138030.