# INAUGURAL-DISSERTATION

ZUR ERLANGUNG DER DOKTORWÜRDE DER
NATURWISSENSCHAFTLICH-MATHEMATISCHEN GESAMTFAKULTÄT DER
RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

VORGELEGT VON
Diplom-Mathematiker Stefan Wiesberg
aus Heilbronn am Neckar

TAG DER MÜNDLICHEN PRÜFUNG

_____________________

# LINK PATTERNS
# IN COMPLEX NETWORKS

BETREUER:

Prof. Dr. Gerhard Reinelt

Prof. Dr. Dres. h. c. Hans Georg Bock

## ZUSAMMENFASSUNG

Um komplexe Netzwerke dem menschlichen Betrachter zugänglich zu machen, definieren Netzwerktheoretiker den Begriff des Musters auf vielfältige Weise. Gebräuchliche Definitionen beruhen auf der Aufteilung der Netzwerkknoten in Gruppen. Diese Aufteilung geschieht in einer Weise, die Muster in den Verbindungen innerhalb und zwischen den Gruppen erkennen lässt. Bekannte Beispiele hierfür sind Cluster- und Blockmodell-Ansätze.
In dieser Arbeit betrachten wie die Mustersuche als diskretes mathematisches Optimierungsproblem. Von diesem Standpunkt aus entwickeln wir eine neue mathematische Klassifizierung für Cluster- und Blockmodell-Verfahren, welche beide Gebiete vereinigt und bisher unabhängig erbrachte Komplexitätsbeweise zusammenführt. Die Klassifizierung wird außerdem dazu verwendet, mathematische Programme für die Mustersuche zu entwickeln und neue Linearisierungsmethoden für die enthaltenen Polynomfunktionen zu diskutieren.
Wir wenden unsere Erkenntnisse auf ein neu formuliertes Mustersuch-Problem an. Trotz seiner einfachen kombinatorischen Struktur können wir bereits seine NP-Schwere nachweisen. Tatsächlich erweist es sich als Verallgemeinerung verschiedener bekannter Probleme, wie dem Traveling-Salesman- und dem Quadratic-Assignment-Problem. Unser abgeleitetes exaktes Mustersuch-Verfahren ist bis zu 10.000-mal schneller als vergleichbare Verfahren aus der Literatur. Zur Demonstration seiner Praktikabilität wenden wir das Verfahren schließlich auf das Welthandelsnetzwerk der Vereinten Nationen an und zeigen, dass das Netzwerk um weniger als 0,14% von den von uns entdeckten Mustern abweicht.

## ABSTRACT

Network theorists define patterns in complex networks in various ways to make them accessible to human beholders. Prominent definitions are thereby based on the partition of the network's nodes into groups such that underlying patterns in the link structure become apparent. Clustering and blockmodeling are two well-known approaches of this kind.
In this thesis, we treat pattern search problems as discrete mathematical optimization problems. From this viewpoint, we develop a new mathematical classification of clustering and blockmodeling approaches, which unifies these two fields and replaces several NP-hardness proofs by a single one. We furthermore use this classification to develop integer mathematical programming formulations for pattern search problems and discuss new linearization techniques for polynomial functions therein. We apply these results to a model for a new pattern search problem. Even though it is the most basic problem in combinatorial terms, we can prove its NP-hardness. In fact, we show that it is a generalization of well-known problems including the Traveling Salesman and the Quadratic Assignment Problem. Our derived exact pattern search procedure is up to $10,000$ times faster than comparable methods from the literature. To demonstrate its practicability, we finally apply the procedure to the world trade network from the United Nations' data base and show that the network deviates by less than $0.14\%$ from the patterns we found.

## ACKNOWLEDGEMENTS / DANKSAGUNG

Diese Arbeit konnte aufgrund der Hilfe zahlreicher Personen verfasst werden, denen ich im Einzelnen danken möchte. Zuerst bedanke ich mich bei meinen Betreuern Professor Gerhard Reinelt und Professor Hans Georg Bock für die Möglichkeit, eine Dissertation an der Fakultät für Mathematik und Informatik der Universität Heidelberg verfassen zu können, sowie für die regelmäßigen Beratungen. Meinen Kollegen Thorsten Bonato, Nam Ngyuễn und Katarina Gavrić verdanke ich sowohl eine angenehme Arbeitsatmosphäre als auch viele interessante wissenschaftliche Diskussionen während der seltenen Pausen. Catherine Proux-Wieland danke ich für ihre hervorragende Unterstützung bei allen organisatorischen und bürokratischen Aufgaben, die mir einige zusätzliche Zeit für die Forschung gewährte. Mein besonderer Dank gilt auch Dr. Marcus Oswald und Dr. Hanna Seitz, die mich sowohl direkt als auch durch ihre vorbildliche Art dazu motiviert haben, mich tiefergehend mit Kombinatorischer Optimierung zu beschäftigen und ein Promotionsstudium zu beginnen. Darüberhinaus danke ich Professorin Katharina Anna Zweig und Professor Johannes Glückler für spannende Projekte und unterstützende Diskussionen zum Thema Netzwerkanalyse. Professorin Elena Fernández, Professor Gábor Galambos und Dr. Miklós Krész bin ich für den Austausch während mehrwöchiger wissenschaftlicher Aufenthalte in Barcelona und Szegedin dankbar. Von Herzen aber danke ich Angela Jäschke und meiner Familie, die zu allen Zeiten den bestmöglichen Ausgleich zur Arbeit zu schaffen wussten.

# Link Patterns in Complex Networks

# Chapter 1

# PATTERNS OF LINK QUANTITY AND LINK EXISTENCE

This chapter gives a short survey on clustering and blockmodeling approaches. In contrast to surveys in literature, the network analytical approaches are classified by their underlying optimization problems, instead of their field of application, methodological affiliation, or historical order of invention. The survey hence aims to reflect the different mathematical ways to model the problem of finding link patterns in a given network. Especially, clustering approaches are introduced as special cases of blockmodeling approaches, even though the first are treated separately in existing surveys for historical reasons. We present a mathematical notation and framework to express all existing approaches as mathematical optimization problems.

## 1.1 PATTERNS OF LINK QUANTITY

### 1.1.1 Motivation

The structure of large networks is usually not comprehensible to the human beholder, especially if the network has not been designed by a human architect, but rather evolved over time in a complex (natural) process. Examples for such networks are social, chemical, economic trading, psychological, anthropological, or political networks. Nevertheless, researchers in the above mentioned fields use the network to gain insight into the underlying processes. A first step is often the reduction of the network's complexity with the help of algorithms.

A common approach is to reduce the high number of nodes in the network. This can be achieved by a grouping of nodes. The goal of the grouping which is discussed in this chapter is the following one. Group the nodes in a way such that for each pair of groups, there are either very *many* or very *few* links between the groups. In other words, search for a *pattern of link density* in the network.

Once such a pattern has been found, the network's complexity has been reduced in the following way. One can now shrink the groups to single nodes and connect two such nodes by an edge if the corresponding groups were densely connected prior to the shrinking. Figure 1.1 (left, center) gives an example of such a partition . A graph $G = (V,E)$ is randomly drawn on the left side. In the center, a partition of $V$ into 4 groups $V_A$, $V_B$, $V_C$, $V_D$, indicated by 4 different colors can be seen, such that a density pattern becomes appearent. Densely connected are the group pairs

$AB, BD, DD, CD, CA$, sparsely connected are $AA, BB, CC, AD, BC$. Note that we use a merely intuitive definition of density here for motivational reasons; strict mathematical definitions will be introduced shortly. The shrunken graph, called *image graph*, is depicted on the right-hand side.
In this section, we assume that our network is given as an undirected graph $G = (V, E)$ and that the groups are non-overlapping. More general cases, in which there are weights (on the arcs or on group pairs), multiple arc types, or overlapping groups are treated in Section 1.3.
We now formalize the notion of *density pattern*. Furthermore, we introduce functions which measure the quality of groupings with respect to patterns. We then formalize the optimizational goal to find the best possible density pattern for a given network.

**Density Pattern.** For a given network, one is hence interested in a partition $P$ of the nodes together with a density pattern. Given $P$, the pattern specifies for each pair of groups whether they are interpreted to be densly or sparsly connected. A pattern is usually notated as a binary square matrix $I$, where the dimension is the number of node groups. An entry $I_{AB}$ is 1 if groups $V_A$ and $V_B$ are interpreted to be densly connected, and 0 if they are interpreted to be sparsely connected. The matrix $I$ is usually called *image matrix*. It is symmetrical as the network graph is undirected. The graph whose adjacency matrix is the image matrix is called *image graph*. Figure 1.1 (right) shows the image graph to the density pattern described on the previous page. Note that the image graph can be seen as a simplification of the network structure: There is an edge in the image graph whereever there are many edges in the original network, and no edge whereever there are only few edges.

**Blockmodels.** The pair $(P, I)$ of a partition $P$ and its interpretation, an image matrix $I$ of appropriate dimension, is called a *blockmodel*. There are several ways to quantify the quality of a blockmodel for a given network. We will introduce them in Chapter 2. The underlying idea is usually that the quality is high if the dense (resp. sparse) parts – according to the image matrix – are actually very dense (resp. sparse), with respect to some density measure for the graph.

**Optimization Problem.** The specification of a quality function for blockmodels directly leads to a combinatorial optimization problem: Given a network and a quality function for blockmodels, find a feasible blockmodel with the highest quality.

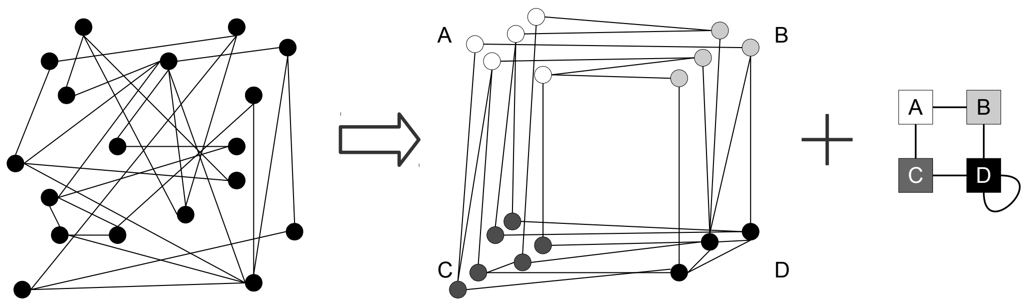


Figure 1.1: Example of a density pattern.

This problem can be formalized in the following way. We are given a graph $G = (V, E)$ with vertex set $V$ and edge set $E$, which represents the network to be searched for patterns. We denote by

- $\mathbf{G}$ the set of all undirected graphs,
- $\mathbf{P}_c(G)$ the set of all partitions of $V$ with $c$ groups,
- $\mathbf{I}_c = \{0,1\}^{c\times c} \cap \mathrm{Sym}_c$ the set of all $c \times c$ binary symmetric matrices,
- $\mathbf{B}_c(G) = \mathbf{P}_c(G) \times \mathbf{I}_c$ the set of all blockmodels with $c$ groups,
- $\mathbf{B}(G) = \cup_{c=1}^{|V|} \mathbf{B}_c(G)$ the set of all blockmodels on $G$,
- $\mathbf{B} = \cup_{G\in\mathbf{G}} \mathbf{B}(G)$ the set of all blockmodels on all graphs.

The following class of problems is parameterized by a set $\mathbf{F} \subseteq \mathbf{B}$ of blockmodels, called the set of *feasible* blockmodels, and a penalty function $p : \mathbf{B} \to \mathbb{R}$ assigning a penalty value to each blockmodel.

BLOCK($\mathbf{F}, p$).
Instance: A graph $G = (V, E)$.
Task: Minimize $p$ over $\mathbf{F} \cap \mathbf{B}(G)$.

Instead of minimizing a penalty function $p$, one could equivalently speak of maximizing a quality function $q$, say $q := -p$. Throughout this thesis, we will use the notion of a penalty function, for reasons that will become evident in Chapter 2. Besides this *deterministic* problem formulation, there are also so-called *stochastic* approaches. The latter include models on how the network was generated. See Nunkesser and Sawitzki [NS05] for an overview.
The definition of the Blockmodeling Problem depends on two parameters. First, the set of all possible blockmodels is restricted to a set $\mathbf{F}$ of *feasible* ones. Second, a penalty function $p$ needs to be specified. In the following two sections, we survey the most frequent choices for $\mathbf{F}$ and $p$ in research practice.

### 1.1.2 Feasibility of Blockmodels

The definition of the Blockmodeling problem restricts the set of all possible blockmodels to a set $\mathbf{F}$ of *feasible* ones. Even though $\mathbf{F}$ can be any subset of blockmodels theoretically, only a few special cases are frequently used in practical research. We discuss them in this section.

**Size and Number of Groups.** Network analysts sometimes bound the size of groups a priori. The *size* of a group is the number of vertices it contains. That is, a blockmodel for a graph is only in $\mathbf{F}$ if the groups in its partition are within given size limits. The reason is to exclude trivial solutions with very many small groups, as they do not increase the comprehensibility of the network. The same problem occurs for partitions with one very large group. In any case, the trivial solution in which every node forms a group on its own is often excluded from $\mathbf{F}$. Otherwise, this partition together with the image matrix $I = Adj(G)$ is the optimum blockmodel to many of the penalty functions $p$ being surveyed below. The condition that all groups should have approximately the same size can be modeled by the specification of an appropriate lower bound on the group sizes as well. Besides the *size*, the *number* of groups is often restricted for similar reasons. However, it is rarely bounded, but mostly fixed to a specific value $c \ll |V|$.

**Fixed Image Matrix.** For some applications, it is useful to test whether a given fixed pattern describes the network well. This pattern can either be a hypothesis formulated by a network analyst or be obtained from a fast heuristic algorithm (see Brusco and Steinley [BS09]). The solution of the Blockmodeling Problem serves the verification or falsification of the hypothetical pattern. The fixed pattern is notated by an image matrix $I'$. The set $\mathbf{F}$ of feasible blockmodels contains only blockmodels $(P,I)$ with image matrix $I = I'$. There are two hypotheses which are commonly tested: The core-periphery and the identity image matrix. The core-periphery matrix is $2 \times 2$ with $I_{11} = 1$ and $I_{22} = 0$. The other two matrix entries are not important and do not count; see Chapter 4.1 for weighted image matrices. One hence searches for a partition into two groups only: The core group and the periphery group. The first one is required to be dense, the latter one sparse (see Borgatti and Everett [BE00]). An example is depicted in Figure 1.2a. The most important image matrix is however the identity matrix. If it is fixed, say with size $c \times c$, one searches for a partition such that the groups themselves are dense, but their interconnections are sparse. This is a well-known *clustering problem* with $c$ clusters. An example is depicted in Figure 1.2b. Note that the image matrix can only be fixed if its dimension and hence the number of groups is also fixed. Otherwise, it is however possible to fix a family $\{I_k\}_{k=1}^{|V|}$ of image matrices; one image matrix for each dimension. For example, the family of identity matrices can be used. That is, $I_k$ is the $k \times k$ identity matrix. The Blockmodeling Problem is then a *clustering problem* with arbitrary many clusters.

**Rounded Image Matrix.** In some approaches, the penalty function $p$ depends solely on the partition $P$, but not on the image matrix $I$. For example, when $p$ measures the deviation of the inter-group densities from an average density; the larger the deviation, the lower the penalty. The edge density between two groups $A$ and $B$ is the number of edges between the groups divided by the maximum possible number of edges. The optimum image matrix can then be created a posteriori to the solution of the Blockmodeling Problem. This is usually done by rounding the edge densities. If the edge density between $V_A$ and $V_B$ is lower than a threshold $\alpha$, then $I_{AB}$ is set to 0, otherwise to 1 ($\alpha$ *density criterion*). The special cases $\alpha = 0$ and $\alpha = 1$ are called the *zeroblock* and the *oneblock criterion*. In this case, the set $\mathbf{F}$ has the following special form: If $(P,I_1) \in \mathbf{F} \cap \mathbf{B}(G)$ and $(P,I_2) \in \mathbf{F} \cap \mathbf{B}(G)$, then $I_1 = I_2$ follows.

**Objective Constraints.** Instead of minimizing a penalty function $p$, some researchers consider the pattern search as a pure feasibility problem. That is, $p$ is implicitly assumed to be constant, say $p \equiv 0$. In return, the density and sparsity requirements for a good pattern are formulated as *constraints*. The Blockmodeling Problem is then to find *any* blockmodel which is in $\mathbf{F} \cap \mathbf{B}(G)$ and thus satisfies all constraints. The constraints could for example require perfect density (all possible edges exist) for $I_{AB} = 1$ and perfect sparsity (not a single edge exists) for $I_{AB} = 0$. This strict requirement is however often softened in practice. Alba presents an alternative for the special case of *clustering* [Alb73]. Instead of allowing only perfect cliques within the groups (clusters), also $n$-cliques are considered feasible. An $n$-clique is a sub graph with the property that the shortest paths between all vertex pairs in the sub graph is at most $n$. A 1-clique is hence a clique. A 2-clique is depicted in Figure 1.2c. For a small fixed number $c$ of groups, $\mathbf{F} \cap \mathbf{B}(G)$ is less likely to be empty for $n > 1$. Furthermore, Alba states that $n$-cliques are also more suitable to describe friendship cliques (*sociometric cliques*) in social networks.

### 1.1.3 Penalties on Blockmodels

One parameter of the Blockmodeling Problem is a function $p : \mathbf{B} \to \mathbb{R}$. It assigns each blockmodel $(P,I)$ a penalty value $p(P,I)$. Chapter 2 gives a survey and classification of all penalty functions used in pattern search. For this reason, we only give three basic examples in this section, all of which are specific for patterns of link *quantity*: Given a partition $P$ and an image matrix $I$, they judge how well the demands of $I$ are met by $P$. That is, whether there are in fact many edges between groups $V_A$ and $V_B$ with $I_{AB} = 1$, and only few edges between $V_A$ and $V_B$ with $I_{AB} = 0$. The three functions could be paraphrased with “the more edges the better”, “the denser the better”, and “the denser than random the better”.
The following notation will be used. Let $[c] := \{1,\dots,c\}$ denote the set of groups of $P$. Let $|V_A|$ for $A \in [c]$ denote the number of vertices in group $V_A$ of $P$. Let *One* denote the set of group pairs which are supposed to be densely connected according to the image matrix. The complement set of pairs is called *Zero*. Furthermore, denote by $m(A,B)$ the number of actual edges between the groups $V_A$ and $V_B$ in $P$. Let $M(A,B)$ denote the maximum possible number of edges between $V_A$ and $V_B$ in $P$. That is, $One := \{A,B \in [c] \mid A \leq B, I_{AB} = 1\}$, $Zero := \{A,B \in [c] \mid A \leq B, I_{AB} = 0\}$, $m(A,B) := \sum_{u\in A,v\in B} Adj(G)_{u,v}$, and

$$M(A,B) := \begin{cases} |A| \cdot |B| & \text{if } A \neq B, \\ |A| \cdot (|A|-1)/2 & \text{otherwise.} \end{cases}$$

The simplest definition of $p$ counts the number of edges in the sparse parts, and the number of non-edges in the dense parts. The lower this sum, the higher is the quality of the blockmodel. The penalty $p$ can be expressed as in Formula 1.0a. The second function computes the densities within the blocks and compares them to the perfect densities 0 and 1. The results are again summed

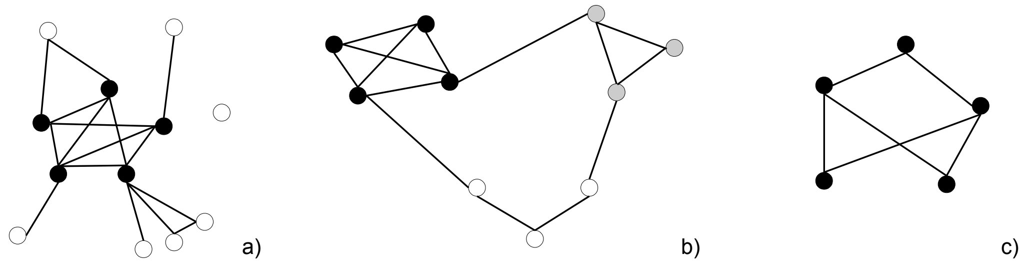


Figure 1.2: a) A core-periphery pattern. b) A clustering (identity pattern). c) A 2-clique.

up. This function is stated in Formula 1.0b. Girvan and Newman propose that the penalty for a blockmodel should not only depend on the absolute densities between the groups. Instead, a partition should be of low penalty only if it is denser than the same partition in a random graph. To this end, it is necessary that the random graph has the same vertex set as $G$. Otherwise, the groups could not be transferred. Let $exp(A)$ denote the number of expected edges within group $V_A$ in a random graph. Their formula 1.0c is restricted to the case where the image matrix is fixed to the identity matrix and only the diagonal entries count (see Section 4.1 on weighted image matrices). The number $exp(A)$ of expected edges in $V_A$ depends on the used random graph model. Girvan and Newman propose to consider graphs which have the same vertex degree distribution as $G$. From this model they deduce the formula $exp(A) = \sum_{u,v \in A, u \neq v} deg(u)deg(v)/2|E|$. They call the value $-p(P,I)/2|E| \in [-1,1]$ the *modularity* of the clustering. Note that in a complete graph $G$, any partition $P$ would yield a low penalty according to the first two functions $p_a$ and $p_b$. The modularity, however, would only yield the average value 0. This is because the same densities are achieved if $G$ is replaced by a random graph with the same vertex distribution, as there is only one such graph.

$$p_a(P,I) = \sum_{(A,B)\in Zero} m(A,B) + \sum_{(A,B)\in One} (M(A,B) - m(A,B)) \tag{1.0a}$$

$$p_b(P,I) = \sum_{(A,B)\in Zero} \frac{m(A,B)}{M(A,B)} + \sum_{(A,B)\in One} \left(1 - \frac{m(A,B)}{M(A,B)}\right) \tag{1.0b}$$

$$p_c(P,I) = \sum_{A\in[c]} \left(exp(A) - \frac{m(A,A)}{M(A,A)}\right) \tag{1.0c}$$

## 1.2 PATTERNS OF LINK EXISTENCE

This section treats the concept of patterns of link *existence*. Instead of demanding many links between two vertex groups $V_A$ and $V_B$, one demands that many vertices in $V_A$ have at least one link to $V_B$ and many vertices in $V_B$ have at least one link to $V_A$; the actual number of links is not taken into account. Analogously, the demand for *few* links from $V_A$ to $V_B$ is replaced by the demand for many vertices *without any* links from $V_A$ to $V_B$.

As in the previous section, a pattern can be notated as a binary matrix, the image matrix $I$. The image graph is again the graph whose adjacency matrix is $I$. Its interpretation differs however: Wherever there is a link in the image graph between two groups, they are interpreted to be connected (in the above sense). Otherwise, they are interpreted to be disconnected. The major combinatorial optimization problem is again the Blockmodeling Problem: Given a network graph, find a feasible blockmodel which minimizes a penalty function $p$.

**Application Example.** As an example, consider a digraph $D$ representing a trade network. The vertices represent companies, the arcs indicate flows of goods in a given period of time. In the case that the image graph of the optimal blockmodel is a path graph, we can conclude that the trading market has the structure of a production chain (see Figure 1.3). Group $V_A$ may in this case consist of the companies producing raw materials, group $V_B$ consists of resellers, group $V_C$ produces intermediate products out of the raw materials, and so on. Note that in this scenario, it is not necessary that every raw material producer sells to *every* reselling company. In fact, there might be only very few links between $V_A$ and $V_B$. Important for the partition is only the fact that every raw material producer sells to *at least one* reselling company.

**Feasibility of Blockmodels.** All of the feasibility constraints discussed in Section 1.1.2 can be applied to patterns of link existence as well. However, literature exploits only a few of these possibilities so far. In common practice, the rounding of the image matrix is rarely used. Image matrix fixation is used [BS09], but there is no distinguished interest in the identity or the core-periphery matrix. Concerning objective constraints, only the extreme case, but no relaxations are used: If $I_{AB} = 1$, then all vertices in $V_A$ must have at least one link to $V_B$ and vice versa. If $I_{AB} = 0$, no vertex in $V_A$ can have a link to $V_B$ and vice versa.

**Penalties on Blockmodels.** Penalty functions for general blockmodels are surveyed in Chapter 2. The most prominent example for patterns of link existence is introduced by Batagelj et al. [BDF92]. Their function $p$ is equivalent to counting the number of *incorrect* vertices for each group pair $(V_A, V_B)$ with $A \leq B$: A vertex $a \in V_A$ is correct if it has a link to $V_B$ in case that $I_{AB} = 1$ or it has no link to $B$ in case that $I_{AB} = 0$.

## 1.3 GENERALIZED GRAPH TYPES

We introduced the notions of patterns of link density and existence for undirected, simple graphs only. In this section, we show how the notions are generalized for other graph types. To our knowledge, there are four different directions of generalization:

1. Directed graphs.
2. Several types of edges in one network, which might also influence each other (*multi-relation models*).
3. Several types of nodes in one network, where the arcs are restricted to connect only certain combinations of node types (*multi-way models*).

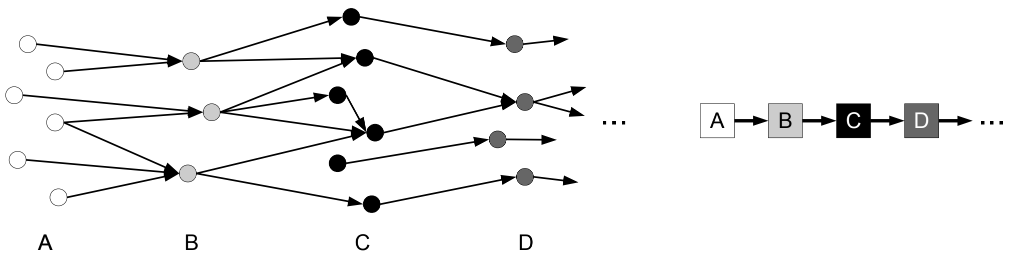


Figure 1.3: A good blockmodel $(P, I)$ for a trading network. Left: $P$. Right: Image graph to $I$.

4. $G$ is a hypergraph, i. e., an edge does not connect 2, but $n$ vertices to each other (*multi-mode models*).

Weighted networks, in which some links are stronger than others, sometimes occur in applications (see Section 1.4), but are usually reduced to the unweighted case: Either the weights are ignored [LBJE03] or all values below a threshold a set to zero; the remaining ones to one [Nor07].

**Directed Graphs.** Directed graphs represent networks with a directed link relation, such as "sends messages to" or "is chemically transformed into". A generalization of the pattern notions for undirected graphs is straightforward. Consider two vertex groups $V_A$ and $V_B$. In a pattern of link density, either many or few arcs exist *from* $V_A$ *to* $V_B$. In a pattern of link existence, many vertices in $V_A$ have an arc *to* $V_B$. Note that the image matrix is not necessarily symmetric. That is, there can be many arcs from $V_A$ to $V_B$, but only few arcs from $V_B$ to $V_A$.

**Multi-Relation Models.** In a multi-relation model graph $G = (V, E_1, \dots, E_k)$, there are several independent edge sets $E_i$. Each set represents a different kind of relation such as "is a friend of", "is a colleague of", "gives orders to", "is an enemy of", etc.
There are two ways of transferring the link pattern notions from the single-relation model. The first one is obvious: A partition of the vertices $V$ is considered to form a pattern if it forms a pattern for every single relation $E_1, \dots, E_k$. The second way is the *multiplex equivalence* approach: For each vertex pair $(u, v)$, all edges from $u$ to $v$ are merged to a single edge. This edge gets a label uniquely representing the combination of the edge types that have been merged. The resulting graph is called the *multiplex graph* $MPX(G)$ of $G$. Now the first way is applicable, if the different labels are interpreted as different kinds of edges $F_1, \dots, F_h$.
For an algebraic study on concatenations of relations, such as "is an enemy of a friend", see Lerner [Ler05] or Chapter 11 in Wasserman and Faust [WF94].

**Multi-Mode and Multi-Way Models.** The concepts of multi-way and multi-mode models are strongly related to each other and quite common in the analysis of data bases. In an $m$-mode model, $G = (V, E)$ is a hypergraph, which means that the adjacency matrix $Adj(G)$ of $G$ is not a mapping from $V \times V$ to $\{0, 1\}$, but from $V^m$ to $\{0, 1\}$. That is, every arc connects $m$ vertices at the same time; the matrix $Adj(G)$ is $m$-dimensional.
In practice, most multi-mode are also multi-way models. That is, $E$ is a subset of $V_1 \times \cdots \times V_m$, where for each two sets $V_i$ and $V_j$ holds that either $V_i = V_j$ or $V_i \cap V_j = \{\}$. For example, $V_1$ could be a set of crime types, $V_2$ a set of cities, $V_3$ a set of years, and an arc $(v_1, v_2, v_3)$ exists if a crime of type $v_1$ happened in city $v_2$ in year $v_3$. If $V_1, \dots, V_m$ consists of at most $w$ disjoint sets, we say that the model is an *m-mode w-way model*. The crime example is a 3-mode 3-way model. A model that considers if person $v_1$ met person $v_2$ on day $v_3$ is a 3-mode 2-way model. Clearly, the definition of $w$ is dependent of the given adjacency matrix, as otherwise $V_i$ and $V_j$ could always both be replaced by $V_i \cup V_j$, decreasing $w$ by 1.
Borgatti and Everett [BE92] extend the pattern notions to multi-mode and multi-way models, which we express here in a more elementary way and only for unweighted adjacency matrices. Instead of densly connected group pairs, we need to speak of densly connected tuples of groups.

In a pattern of link density, for each tuple $T := (V_1', \ldots, V_m')$ of groups, many edges $(v_1, \ldots, v_m)$ are demanded to exist in case that the group connection is dense. In the example, crimes in the same group mostly happen in the same cities in the same years. In a pattern of link existence, many nodes $v \in V_i'$ are part of an edge $(v_1, \ldots, v_{i-1}, v, v_{i+1}, \ldots, v_m)$, for all $i = 1, \ldots, m$. In the example, many crimes in one group happen in cities in the same group and in years in the same group.

## 1.4 SCIENTIFIC APPLICATIONS

Link patterns are used mostly to simplify the structure of complex networks. As we have seen, the image graph can be seen as a simplification of the network graph. This simplification loses only a small amount of information about the link structure in the following sense: The interpretation of an edge $AB$ in the image graph as "All possible edges between $V_A$ and $V_B$ exist." is almost true, since most of the edges actually exist. A second application of link patterns is the classification of networks by their image graphs. In Chapter 5, we will classify trading networks by searching for a small number of image graphs, such that all network graphs correspond to one of these image graphs.

Besides the link pattern, the groups themselves can reveal information about the structure of the network. This is, in fact, the main purpose of clustering, where the image matrix is fixed to the identity matrix. For example, a clustering of the friends of one person leads to groups that can often be labeled with "colleagues", "relatives", "friends from school", and so on. The members of one group mostly know each other, whereas only few of them know members of other groups. We give an overview on applications where the link pattern, not the groups themselves, reveals information about the network. We start with three examples for patterns of link *density*.

**Example (World Trade).** In an international trade network, vertices represent states. Two vertices $u$ and $v$ are connected by an arc $uv$ if trade goods are delivered from state $u$ to $v$ within a given period of time. Weights on the arcs represent the amount of trading goods. The search for patterns of link density groups states with almost identical trade partners and makes the network thus more comprehensible. Reichardt and White [RW07] examine the *United Nations Commodity Trade network* for the year 2000, containing trading goods of 55 different kinds. They partition the vertices into 2 to 9 groups, respectively. They conclude that the resulting groups allow for a real-world interpretation, as they can be easily labeled with terms such as "Central Europe", "Middle and South America", and "Eastern Europe and Northern Africa". The resulting density pattern for 5 groups is depicted in Figure 1.4 (bold arrows for strong trade, dashed arrows for limited trade).

**Example (International Conflicts).** In an international conflict network, the vertices represent again states. The edges are weighted, where the weights represent the amount of affinity between the two states. Obviously, there is no unique definition of *affinity*. Practically, it is computed from several viewpoints and sources, such as whether the states are allies, to which extent they trade with each other, and if are members in joint intergovernmental organizations. The search for patterns of link density can verify or falsify common hypotheses about the formation of conflicts between states. Maoz et al. [MKTT06] examine several such hypotheses, such as “The more similiar the affinities of two states $u$ and $v$ to other states are, the lower is the conflict potential between $u$ and $v$.” To test this hypothesis, they combine 5 data sources. All conflicts between politically relevant states in the years 1816 to 2001 are considered. Through an indirect search for link density patterns, they find that the hypothesis above is consistently confirmed by their data.

**Example (Technology Patents).** In a patent network, the vertices represent products. There is an arc $(u,v)$ if the production of product $v$ requires the use of a technology that has been patented for product $u$. The network can thus be interpreted to represent the flow of technological ideas. Weng et al. [WCHC10] examine a network of patents on methods in the insurance business. To create this network, they use a database of the *United States Patent and Trademark Office*. By searching for patterns of link density, they find the 4-group pattern depicted in Figure 1.4. They find the real-world interpretation that the patent transfer market has a so-called *core-periphery* structure. The technologies located in the *core group* $V_A$ have a better opportunity to become a dominant design. In contrast, the technologies in the *periphery groups* $V_B, V_C$ and $V_D$ are dependent on the technological progress in group $V_A$.

The following three application examples consider patterns of link *existence*.

**Example (Food Chain).** In a food web, the vertices represent organisms such as animals or plants. An arc $(u,v)$ indicates that organism $u$ consumes organism $v$. The computation of patterns of link existence can group the organisms into so-called trophic roles, e.g. the notions of “occupying a niche” or “being a top-predator”. Luczkovic et al. [LBJE03] compute patterns on a graph representing the *Florida Seagrass Ecosystem*. They find 10 groups of animals which allow a real-world interpretation: The “benthic producers”, the “planktonic producers”, the “planktonic consumers”, and so on. The pattern explains the structure of the food web, leading to insights such as “All fish and invertebrates eat at least one benthic and at least one planktonic producer”, and many more.

**Example (World Trade).** In a world trade network graph, the vertices represent countries. An arc $(u,v)$ indicates a flow of commodities from country $u$ to $v$; its weight gives the value of

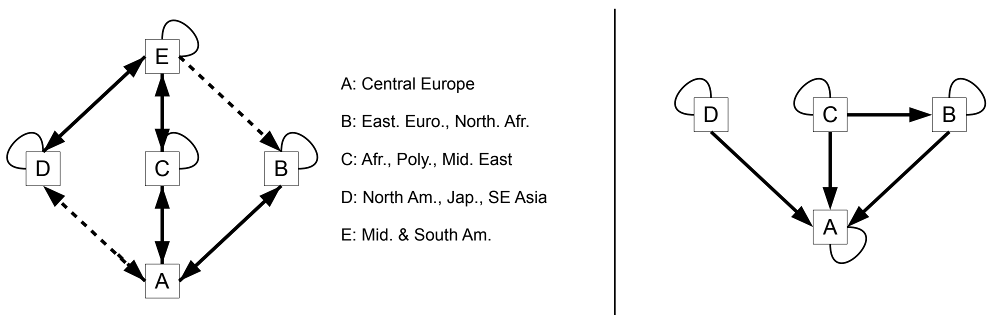

E
D
C
B
A
A: Central Europe
B: East. Euro., North. Afr.
C: Afr., Poly., Mid. East
D: North Am., Jap., SE Asia
E: Mid. & South Am.
D
C
B
A

Figure 1.4: Left: Pattern of link density for world trade. Right: Pattern of link density of technology patents in insurance business.

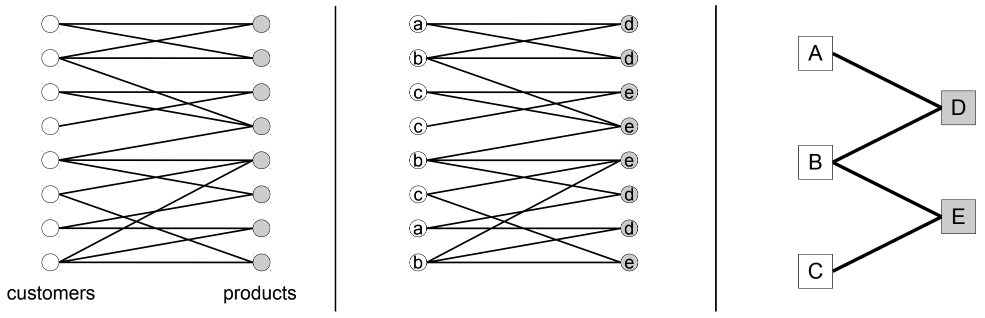


Figure 1.5: Left: A customer-product graph. Center: A partition of the vertices. Right: The pattern of the partition in the center.

the commodities within a given year. Patterns of link existence reveal the strategic positions of countries. It has been argued by Wallerstein that the world trade system does not have a core-periphery, but a core-semiperiphery-periphery structure. Smith and White [SW92] confirm this theory by a computation of link existence patterns. Using data on the world commodity trade flows from the years 1965–1980, semiperipheries are detected. The vertex groups have real-world interpretations. For example, they find that a new group is formed in the year 1980, which can be labeled “Fourth World group of very poor African countries”.

**Example (Customers and Products).** A customer-product network graph consists of two kinds of vertices: The customers and the products. An edge $cp$ connects a customer $c$ and a product $p$ if the former bought the latter within a given period of time. The graph is hence bipartite. According to Borgatti and Everett [BE92], the market analysts are interested in two questions: Which products compete with each other and which types of customers there are. The search for patterns of link existence can contribute to an answer. At the same time, groups of customers and groups of products are computed. Customers in the same group buy products from the same groups of products. Vice versa, products in the same group are the ones being bought by the customers of the same groups. Without using a priori information about the customers (“Young men with high income”, “elder workers not interested in sports”), the pattern search allows for an a posteriori interpretation, which is hence not restricted by uncertain pre-assumptions. Figure 1.5 (left) shows an example network. The center shows a partition of the customer vertices in groups $a,b,c$ as well as a partition of the product vertices in groups $d$ and $e$. If the vertices are partitioned in this way, a pattern of link existence is created. It is visualized on the right side. It shows that customers in $V_A$ buy only products in $V_D$, customers in $V_C$ buy only products in $V_E$, whereas customers in $V_B$ buy both kinds of products.

# Chapter 2

# A CLASSIFICATION OF QUALITY FUNCTIONS

The Blockmodeling Problem (Page 3) is a family of problems parameterized by a set $\mathbf{F}$ of feasible blockmodels and a penalty function $p : \mathbf{B} \to \mathbb{R}$ to measure the quality of blockmodels; the lower the value, the higher the quality.

BLOCK($\mathbf{F}, p$).
Instance: A graph $G = (V, E)$.
Task: Minimize $p$ over $\mathbf{F} \cap \mathbf{B}(G)$.

Recall that a blockmodel consists of a partition $P$ of the vertex set and its interpretation: a pattern given as a binary matrix $I$. Literature contains a large variety of penalty functions, which are often not explicitly stated as such.

In this chapter, we present a new classification for penalty functions. It shows that the design of $p$ and $\mathbf{F}$ go hand in hand in the following process: At the beginning, the set $\mathbf{F}$ contains only so-called *ideal* blockmodels; a notion to be defined in Section 2.1. The set of *all* ideal blockmodels is thereby restricted by the techniques explained in the first chapter (excluding trivial blockmodels, fixing image matrices, etc.). In the next step, this set is widened to contain also non-ideal blockmodels. The penalty function $p$ is then designed to measure the deviation of each blockmodel in the enlarged set to the ideal ones. This process is explained in detail in Section 2.2.

The new classification has three benefits. First, it is a common theory for the search for clusters, link density, and link existence patterns, which are usually treated separately. Second, the NP-hardness of the different realizations of the Blockmodeling Problem is a direct consequence of the classification. Several separate NP-hardness proofs in literature can now be replaced by a single one (Section 2.4). Third, it serves as a basis for a methodological discussion of which penalty function is most suitable under which circumstances.

## 2.1 IDEAL BLOCKMODELS

In this section, we define *ideal* blockmodels of link density and existence. In an *ideal* blockmodel $(P, I)$ for link density, *all* links exist in the dense parts and *no* links exist in the sparse ones. In an ideal blockmodel for link existence, either *all* or *no* vertices in $V_A$ have a neighbor in $V_B$ and vice versa, for all group pairs $(V_A, V_B)$.
We now formalize the notion of ideal partitions. There are three equivalent graph theoretical definitions of ideal partitions: The subgraph, the node pair, and the single node definition. They will be presented in the next three subsections.

### 2.1.1 The Subgraph Definition

In an ideal blockmodel for density patterns, certain subgraphs are required to be either complete (all edges exist) or empty (no edge exists). These subgraphs can be formally defined as follows. Given a partition $P$, there is one such subgraph $G_{P,A,B}$ for every pair $(V_A, V_B)$ of groups. It is obtained from $G$ by deleting all vertices but the ones in $V_A$ or $V_B$ and deleting all edges but those connecting an vertex in $V_A$ to a vertex in $V_B$. $G_{P,A,B}$ is hence bipartite for $A \neq B$. A similar observation can be made for ideal link existence blockmodels: That all vertices in $V_A$ have at least one neighbor in $V_B$, and vice versa, is equivalent to the statement that $G_{P,A,B}$ contains no isolated vertices.

**Definition 1.** Given a graph $G$, a partition $P$ of its vertex set $V$ into $c$ groups is an

(i) *ideal structural c-partition*, if for all pairs $A, B \in [c]$, the graph $G_{P,A,B}$ is either empty or a complete (complete bipartite for $A \neq B$) graph.

(ii) *ideal regular c-partition*, if for all pairs $A, B \in [c]$, the graph $G_{P,A,B}$ is either empty or contains no isolated vertices.

From this definition, it is obvious that Property (i) implies (ii). That is, every ideal structural $c$-partition is an ideal regular $c$-partition. Structural and regular partitions are extensively studied by Lorrain and White [LW71] in the form of equivalence relations (two vertices in the same group

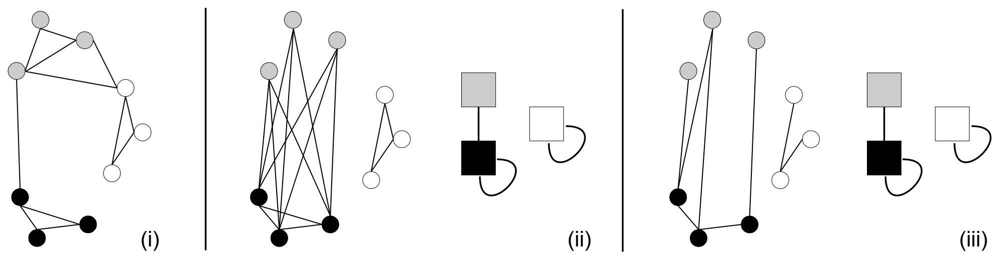


Figure 2.1: Ideal clique (i), structural (ii), and regular (iii) 3-partitions. In (ii) and (iii), the corresponding image graph is depicted.

are considered to be equivalent to each other), who motivate them by their sociological meaning in social networks.

The ideality definition can be easily transferred from partitions to *blockmodels*. An ideal blockmodel $(P,I)$ consists of an ideal partition $P$ and an image matrix $I$ which reflects its obvious density or connectivity pattern:

**Definition 2.** A blockmodel $(P,I)$ consisting of a $c$-partition $P$ and a $c \times c$ binary matrix $I$ is called *ideal structural* (resp. *ideal regular*) blockmodel if $P$ is an ideal structural (resp. ideal regular) $c$-partition and $I_{AB} = 0$ if and only if $G_{P,A,B}$ is empty. We denote by $\mathbf{B}^S_{\text{ideal}}(c,G)$ (resp. $\mathbf{B}^R_{\text{ideal}}(c,G)$) the set of all ideal structural (resp. regular) blockmodels with $c$ groups on $G$. We write $\mathbf{B}^X_{\text{ideal}}(c,G)$ in statements which hold for both structural and regular blockmodels. With $\mathbf{B}^X_{\text{ideal}}(G) = \cup_{c=1}^{|V|}\mathbf{B}^X_{\text{ideal}}(c,G)$ we denote the set of all ideal structural (resp. regular) blockmodels.

### 2.1.2 The Node Pair Definition

We have seen that ideal partitions can be defined by subgraph characterizations. Equivalently, they can be defined by properties of same-grouped vertices. In a structural $c$-partition, two vertices in the same group have exactly the same neighboring vertices in $G$. In a regular $c$-partition, two vertices in the same group have exactly the same groups in their neighborhoods. Formally, let $N(u)$ denote the set of vertices that are adjacent to vertex $u$. Recall that $P(u)$ denotes the group of vertex $u$ in the partition $P$. The following definition is equivalent to Definition 1 above.

**Definition 3.** Given a graph $G$, a partition $P$ of its vertex set $V$ into $c$ groups is an

(i) *ideal structural c-partition*, if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$ for all $u,v \in V$ with $P(u) = P(v)$.

(ii) *ideal regular c-partition*, if $\{P(w) \mid w \in V, uw \in E\} = \{P(w) \mid w \in V, vw \in E\}$ for all $u,v \in V$ with $P(u) = P(v)$.

**Proposition 1.** Definition 1 and Definition 3 are equivalent.

*Proof.* (i) Consider a partition $P$ which satisfies Definition 1(i). Consider two distinct vertices $u,v \in V$ in the same group $V_A$ with $A := P(u)$. Consider $B \in [c]$ such that $G_{P,A,B}$ is empty. Then both $u$ and $v$ do not have any neighbor in $V_B$. Now consider $B \in [c]$ such that $G_{P,A,B}$ is complete. Then $u$ and $v$ are both neighbors to all vertices in $V_B$. All in all, $u$ and $v$ have the same neighbors: $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. The partition $P$ hence satisfies Definition 3(i). Now consider a partition $P$ satisfying Definition 3(i). Consider $A,B \in [c]$ with vertices $a_1,\dots,a_{|A|}$ and $b_1,\dots,b_{|B|}$ respectively. In case that $a_1$ does not have a link to $V_B$, no other vertex in $V_A$ has a link to $V_B$, as the $a_i$ have the same neighbors pairwisely. In this case, $G_{P,A,B}$ is empty. In case that $a_1$ has a link to $V_B$, say to $b_1$, all $a_i$ have a link to $b_1$. As $b_1$ is hence linked to all $a_i$ and all $b_i$ have the same neighbors, all $b_i$ are linked to all $a_i$. The graph $G_{P,A,B}$ is hence complete. All in all, $P$ satisfies Definition 1(i).

(ii) Consider a partition $P$ which satisfies Definition 1(ii). Consider two distinct vertices $u,v \in V$ in the same group $V_A$ with $A := P(u)$. Consider $B \in [c]$ such that $G_{P,A,B}$ is empty. Then both $u$ and $v$ do not have any neighbor in $V_B$. Now consider $B \in [c]$ such that $G_{P,A,B}$ contains no isolated vertices. Then $u$ and $v$ have both links to $V_B$, as they would be isolated otherwise. All in all, $u$ and $v$ have the same neighbor groups. The partition $P$ hence satisfies Definition 3(ii). Now consider a partition $P$ satisfying Definition 3(ii). Consider $A,B \in [c]$ with vertices $a_1,\dots,a_{|A|}$ and $b_1,\dots,b_{|B|}$ respectively. In case that $a_1$ does not have a link to $V_B$, no other vertex in $V_A$ has a link to $V_B$, as the $a_i$ have the same neighbor groups pairwisely. In this case, $G_{P,A,B}$ is empty. In case that $a_1$ has a link to $V_B$, say to $b_1$, all $a_i$ have a link to $V_B$. Hence, no vertex in $V_A$ is isolated. Similarily, no vertex in $V_B$ is isolated. The graph $G_{P,A,B}$ is hence without isolated vertices. All in all, $P$ satisfies Definition 1(ii). □

### 2.1.3 The Single Node Definition

A definition from the perspective of *single* vertex is only possible with respect to a fixed image matrix $I$. In this case, the following single node definition is compatible with the two definitions above:

**Definition 4.** Given a graph $G$ and a $c \times c$ image matrix $I$, a partition $P$ of its vertex set $V$ into $c$ groups is an

(i) an *ideal structural c-partition w. r. t. I*, if for all $u \in V$ and all $C \in [c]$: $u$ is adjacent to all $v \in V$ with $P(v) = C$ if $I_{P(u)C} = 1$, and to no $v \in V$ with $P(v) = C$ if $I_{P(u)C} = 0$.

(ii) an *ideal regular c-partition w. r. t. I*, if for all $u \in V$ and all $C \in [c]$: $u$ is adjacent to at least one $v \in V$ with $P(v) = C$ if $I_{P(u)C} = 1$, and to no $v \in V$ with $P(v) = C$ if $I_{P(u)C} = 0$.

It is obvious that this definition is compatible to the subgraph definition. We thus omit the proof of the following proposition.

**Proposition 2.** *$P$ is an ideal structural (resp. regular) $c$-partition w.r.t. $I$ according to Definition 4 if and only if it is an ideal structural (resp. regular) $c$-partition with image matrix $I$ according to Definition 1.* □

### 2.1.4 Variants of Ideality

Patterns of link existence demand that many vertices in group $V_A$ have links to group $V_B$ in the case that the connection between $A$ and $B$ is “dense”. We have formalized this notion by the concept of ideal regular partitions. There, every vertex in $V_A$ is demanded to have *at least one* link to $V_B$. Several alternatives have been proposed in literature. We mention them briefly, as they are rarely used in network analysis practice.

**Exact Regular Partitions.** Ideal *exact regular partitions* demand that all vertices in $V_A$ have exactly the same number of neighbors in $V_B$. This number may however differ for each group pair $(V_A, V_B)$. Partitions of this kind are also called *equitable partitions*. Under the latter name, they have been well investigated in graph theory [Ler05]. Ideal regular partitions are hence a generalization of equitable partitions.

**Perfect Regular Partitions.** An ideal regular partition is called *perfect* if all vertex pairs that satisfy the regularity conditions are indeed in the same group. Perfect equivalence is mainly of theoretical interest, as there seems to be no practical reason to restrict regular equivalence this way [Ler05].

**Relative Regular Partitions.** Given a partition $P_0$ on $V$, a refinement $P$ of $P_0$ is called *relative regular to* $P_0$ if $\{P_0(w) \mid w \in V, uw \in E\} = \{P_0(w) \mid w \in V, vw \in E\}$ for all $u, v \in V$ with $P(u) = P(v)$. That is, members of the same group in $P$ have the same neighboring groups in $P_0$. An application is given in the study of interaction between friendship cliques [BE99].

## 2.2 DEVIATIONS FROM IDEALITY

Given a graph $G$ and one of the two ideal partition types above, there are efficient algorithms to compute any ideal partition (and thus ideal blockmodel) of this type. However, this is usually not done in practice. In Section 2.2.1, we list some common reasons for this decision. In Section 2.2.2, we show that the approaches used in practice can be interpreted as the solution of an optimization problem on a relaxed ideality definition.

### 2.2.1 Reasons for Relaxations

There are several reasons why ideal blockmodels are often not directly searched for in practice. We list four of them.

1. *Non-existence of solutions.* An ideal partition might only exist for a large number of groups.

2. *Real-world modeling reasons.* The ideal definition might be too restrictive for the application at hand. For example, the graph theoretical definition of *clique* might be too strict to describe friendship cliques in social networks, where some edges can be missing.

3. *Involvement of statistics.* The ideal definition does not allow the definition of statistically profound criteria for the quality of partitions. In fact, the criteria are purely graph theoretical.

4. *Robustness against measuring errors.* The extraction of graphs from complex networks can be erroneous, especially with biological or chemical networks. However, a regular partition on a graph can turn non-regular by the deletion or addition of one single edge. The ideal definitions are hence not useful to limit the influence of these errors on the partitions.

### 2.2.2 General Relaxation

In this section, we show how the penalty function $p$ and the set $\mathbf{F}$ of feasible blockmodels are constructed hand in hand. This procedure holds for all (non-stochastic) clustering and blockmodeling approaches which (directly or indirectly) quantify the quality of blockmodels and are reported in the following survey books: *Social Network Analysis* by Wasserman and Faust [WF94], *Network Analysis* by Brandes and Erlebach [BL10] (except *conductance*), and *Community Detection in Graphs* by Fortunato [For09].

1. A set $F \subseteq \mathbf{B}$ of feasible blockmodels is chosen as explained in Section 1.1.2. For example by excluding trivial solutions, demanding minimum group sizes, fixing the number of groups, or fixing an image matrix.

2. One of the three equivalent definitions of ideal blockmodels (single node, node pair, and subgraph definition) is chosen.

3. In the chosen definition of ideal blockmodels, some of the named requirements for the blockmodels in $B^X_{\text{ideal}} \cap F$ are dropped (*relaxation step*). Let $F' \subseteq \mathbf{B}$ denote the set of all blockmodels which satisfy the remaining requirements.

4. $\mathbf{F}$ is set to $F \cap F'$.

5. The penalty function $p$ is not arbitrary, but measures for each blockmodel $B \in \mathbf{B}$ its deviation from the members of $\mathbf{B}^X_{\text{ideal}} \cap F$.

We can now classify the penalty functions by the type of definition chosen in Step 2. Section 2.2.4 treats the use of the single node and the node pair definitions, Section 2.2.5 of the subgraph definition. First, however, we treat a case in which the choice of the definition is arbitrary:

### 2.2.3 **Partition Number Relaxations**

For many applications, a good choice for the number of groups is not a priori known and hence not fixed to a certain value $c$. As a low number of groups is usually more suitable for interpretation, one could penalize larger numbers of groups. We formulate it in terms of the 5-step procedure above. In Step 1, the set $F$ is chosen to contain only blockmodels with one group only. These are the "perfect" blockmodels with the lowest possible number of groups. In Step 2, the choice of the definition is not relevant, as we want to relax the definition of "partition" itself, which appears in all three definitions. In Step 3, in the requirement that the partition is a $c$-partition with $c = 1$, the requirement "$c = 1$" is dropped. In Step 5, the penalty function $p$ measures the violation against this requirement. For example, it can assign each partition $P$ the number $c$ of groups used by $P$. The lower the number of groups, the less the amount of penalty.

**Example (CATREGE).** The algorithm CATREGE [BE93] solves the problem BLOCK for such a $p$ and the case of *regular* partitions. I.e., given a graph, it finds an ideal regular $c$-partition with the smallest possible $c$.

### 2.2.4 **Single Node and Node Pair Relaxations**

In single node relaxations, the properties named in the single node definition (Definition 4) are relaxed. Single node definitions are only possible if the image matrix $I$ is fixed.

**Example (Nodal Degree Relaxations).** An example are the nodal degree relaxations for clusterings, i.e., for structural partitions with the image matrix fixed to the identity matrix. The set $F$ is hence the set of all blockmodels where the image matrix is the identity matrix. Seidman and Foster [SF78] relax the requirement that every vertex must be adjacent to all other vertices in the same group by the requirement that every vertex can be non-adjacent to at most $k$ other vertices in the same group. In a blockmodel in $F'$, the subgraphs induced by the groups are hence not cliques, but so-called *k-plexes*. The relaxation is not penalized. That is, $p$ is constant, say $p \equiv 0$.

Analogously, $k$-cores can be used instead of $k$-plexes.

We now turn to the more common node *pair* relaxations. Here, the properties for vertex pairs in Definition 3 are relaxed. Two forms of $p$ are most commonly used, which will be explained by the following two examples: $p$ is either constant or decomposable over the set of all vertex pairs.

**Example (Sociometric Cliques).** Alba [Alb73] finds the graph theoretical definition of *clique* to be not perfectly appropriate to describe friendship (or sociometric) cliques in social networks. He thus relaxes its definition to so-called $n$-cliques. Here, two vertices in the same group do not need to be connected by an edge. They need to be connected by a path of length at most $n$.
The set $F$ is again the set of all blockmodels where the image matrix is the identity matrix. The edge connection requirement is relaxed. Hence, the set $F'$ contains all blockmodels where the groups induce $n$-cliques. The penalty function $p$ is chosen to be constant. The problem thus merely consists in the search for *any* partition into $n$-cliques. Similar to the $n$-clique are the $n$-clan and $n$-club relaxations [Mok79].

We now treat a second common type of node pair relaxation: The *vertex similarity approaches*. The set $F$ consists of all blockmodels with rounded image matrix (the kind of rounding needs to be specified by the network analyst). The node pair definition says that two vertices are in the same group if they have exactly the same neighbors. This requirement is completely dropped. The set $F'$ is hence the set $\mathbf{B}$ of all blockmodels. The penalty function $p$ is constructed as follows. First, a dissimilarity value $p_{uv}$ is introduced for each pair $u, v$ of vertices. It measures how dissimilar the neighbors of vertex $u$ and $v$ in the same group are. Second, $p$ is defined as the sum over all dissimilarities:

$$p(P) = \sum_{u,v \in V} p_{uv} \delta(P(u), P(v)).$$

Here, $p_{uv} \geq 0$ are real numbers and $\delta$ denotes the Kronecker function. It is 1 if $P(u) = P(v)$ and 0 otherwise. The relaxation technique of using such a decomposable function $p$ is called *indirect blockmodeling approach* by Doreian et al. [DBF05].

**Example (Structural Partitions).** For structural partitions, several functions $p$ of the above form have been proposed. These propositions were made indirectly by a specification of the values $p_{uv}$. They quantify how much a partition violates this dropped requirement, that is, to quantify how dissimilar two vertices are with respect to common neighbors. Most directly, one can simply count the number of distinct neighbors of $u$ and $v$:

$$p_{uv} = \frac{|(N(u) \setminus v) \Delta (N(v) \setminus u)|}{k} \tag{2.1}$$

where $\Delta$ denotes the symmetric difference of the two sets and $k$ denotes a scalar, which could for example be constant ($k = 1$) or the total number of neighbors ($k = |(N(u) \cup N(v)) \setminus \{u, v\}|$). In the latter case, one has to set $p_{uv} = 0$ in case that both $u$ and $v$ have no neighbors at all, in order to sidestep a division by zero. Alternatively, Burt [Bur76] proposes to use the Euclidian distance between the image matrix entries of $u$ and $v$, that is,

$$p_{uv} = \sqrt{\sum_{w \in V, w \neq u,v} (Adj(G)_{uw} - Adj(G)_{vw})^2}. \tag{2.2}$$

### 2.2.5 Subgraph Relaxations

In subgraph relaxations, the set $F$ does not necessarily have to be restricted from $\mathbf{B}$. The requirements of Definition 1 for ideal partitions are relaxed. That is, the requirements "$G_{P,A,B}$ is a complete graph", "$G_{P,A,B}$ is an empty graph", or "$G_{P,A,B}$ contains no isolates" are dropped. The set $F'$ is the set $\mathbf{B}$ of all blockmodels. The penalty function $p$ then measures the difference between each sub graph $G_{P,A,B}$ and the ideal *complete, empty*, or *isolate free* subgraphs. It hence penalizes $G_{P,A,B}$'s deviations from ideality.
There are several ways to measure the distance $d$ between the subgraph $G_{P,A,B}$ and its ideal counterpart subgraph $H_{P,A,B}$ on the same vertex set $V$. Such distance measures on graph pairs are called *edit distances*. A simple but common exemplary form of an edit distance is given by

$$d(G_{P,A,B}, H_{P,A,B}) = \sum_{u,v \in V, u \neq v} |\mathrm{Adj}(G_{P,A,B})_{u,v} - \mathrm{Adj}(H_{P,A,B})_{u,v}|, \tag{2.3}$$

where Adj denotes the adjacency matrix of the graph. The function counts the number of different entries in the adjacency matrices of $G_{P,A,B}$ and $H_{P,A,B}$. More complex distance functions are discussed below. The function $d$ measures the distance of $G_{P,A,B}$ to a single ideal subgraph $H_{P,A,B}$. We can also measure its distance to a set $\mathbf{H}_{P,A,B}$ of ideal subgraphs, by defining it as the distance of $G_{P,A,B}$ to its closest element in $\mathbf{H}_{P,A,B}$. That is,

$$d(G_{P,A,B}, \mathbf{H}_{P,A,B}) := \min_{H \in \mathbf{H}_{P,A,B}} d(G_{P,A,B}, H).$$

In the remainder of this section, we will see how ideal graphs $\mathbf{H}_{P,A,B}$ are defined. Then, we give an overview on the distance functions which are used in practice. Subsequently, we show how the distance value for each group pair can be combined to the total penalty value. We close by some examples on how subgraph relaxation is used in literature.

**Ideal and Average Graphs.** The sets of ideal subgraphs $\mathbf{H}_{P,A,B}$ consist of subgraphs which are either complete, empty, or without isolated vertices, as demanded by Definition 1. Precisely, the set can be defined as follows for the two different pattern types:

(i) structural $c$-partition: For $A = B$, the empty graph and the complete graph on $V_A$. For $A \neq B$, the empty graph on $V_A \cup V_B$ and all complete bipartite graphs on $V_A \cup V_B$.

(ii) regular $c$-partition: For $A = B$, the empty graph on $V_A$ and all graphs on $V_A$ which do not contain isolated vertices. For $A \neq B$, the empty graph on $V_A \cup V_B$ and all bipartite graphs with shores $V_A$ and $V_B$ which do not contain isolated vertices.

Note that for every partition $P$, the set $\mathbf{H}_{P,A,B}$ is non-empty. Given a distance function $d$, the penalty value $p_{AB}$ for the group pair $(V_A, V_B)$ is set to $p_{AB} = d(G_{P,A,B}, \mathbf{H}_{P,A,B})$. An alternative to the comparison to ideal graphs is the comparison to *average* ones. It has been used for structural partitions. The average subgraphs are hence neither empty nor complete, but have an average density. The distance of $G_{P,A,B}$ to the average graphs $\mathbf{H}_{P,A,B}$ can then be positive or negative, depending on whether $G_{P,A,B}$ is denser or sparser than average. The exact definition of $\mathbf{H}_{P,A,B}$ depends on the way "average" is defined; see the example paragraph below.

**Overview on Distance Functions.** With Function (2.3), we already stated the simplest distance function to measure the distance between two graphs on the same vertex set. It counts the number of edges which are different in $G_{P,A,B}$ and the ideal subgraph $H_{P,A,B}$. However, if $G_{P,A,B}$ is compared to *average* graphs, the absolute value function is a problem. Here, we want to distinguish whether $G_{P,A,B}$ is worse or better than average. Hence, Function (2.4) below is more suitable in this case. The adjacency matrix of $H_{P,A,B}$ is possibly weighted, as average graphs usually do not have binary edge weights. There is a third function for the case that the edit distance $d$ does not count edge but *vertex* differences. Note that there can be no missing vertices in $G_{P,A,B}$, as adding a vertex cannot contribute to the transformation of $G_{P,A,B}$ into an ideal subgraph. For this reason, ideal subgraphs for $G_{P,A,B}$ are defined on vertex sets $V_{A'} \cup V_{B'}$ with $V_{A'} \subseteq V_A, V_{B'} \subseteq V_B$. They are again either complete, empty, or isolate-free graphs. We do not give a precise definition as vertex countings are not common in literature. The simple distance function (2.5) counts the number of vertices that need to be deleted from $G_{P,A,B}$ in order to obtain an ideal subgraph $H_{P,A,B}$.

$$d(G_{P,A,B}, H_{P,A,B}) = \sum_{u,v \in V, u \neq v} (\mathrm{Adj}(G_{P,A,B})_{u,v} - \mathrm{Adj}(H_{P,A,B})_{u,v}). \tag{2.4}$$

$$d(G_{P,A,B}, H_{P,A,B}) = |V(G_{P,A,B})| - |V(H_{P,A,B})|. \tag{2.5}$$

Beside these linear functions, several non-polynomial functions have been proposed. Being derived from general statistical matrix correlation measures, they can be used to compare the adjacency matrices of $G$ and $H$. See Wasserman and Faust [WF94] or Arabie et al. [ABL78] for an overview.

**Combining Subgraph Penalties.** In Definition 1, the ideal partition conditions are formulated as requirements for the subgraphs $G_{P,A,B}$ of $G$. There is hence a single penalty value $p_{AB}(P) = d(G_{P,A,B}, \mathbf{H}_{P,A,B})$ for each group pair $(A,B)$. The distance function $d$ is thereby always the same for all group pairs. There are several ways to combine the values $p_{AB}(P)$ to a total penalty $p(P)$ for the partition $P$. In most cases, the $p_{AB}$ values are simply summed up as in Formula (2.6). See Figure 2.2 for an example for structural 3-partitions: The distance $d(G_{P,A,B}, \mathbf{H}_{P,A,B})$ of the depicted partition $P$ of the drawn graph $G$ is 3. The reason is that 3 edge changes are at least necessary to obtain a structural 3-partition: Between groups Gray and Black, add two edges ($p_{GB} = 2$). Between groups White and White, delete one edge ($p_{WW} = 1$). This way, $P$ is an ideal 3-partition for the graph $H^*$ in the figure. Its image matrix $I^*$ is visualized as an image graph on the right side. Hence, the penalty value for this partition is $p(P) = 2 + 1 = 3$.
Instead of summing up, one can use scaling to give the penalty value $p_{AB}$ a higher weight if the groups $A$ and $B$ are large. The factor is usually $1/m_{AB}$, where $m_{AB}$ is the number of possible edges in the subgraph $G_{P,A,B}$. More precisely, $m_{AB} = |V_A| \cdot |V_B|$ if $A \neq B$, $m_{AA} = |V_A| \cdot (|V_A| - 1)$, and $p$ is

defined as in Formula 2.7. In some approaches, the squares of the penalties are summed up instead. This mostly occurs in so-called $\chi^2$ *approaches*, see Formula (2.8). As an alternative scaling factor, the distance of $G_{P,A,B}$ to $\mathbf{H}_{P,A,B}$ can be seen in relation to the maximum distance $d^{\max}_{P,A,B}$ of any graph, on the same vertex set, to $\mathbf{H}_{P,A,B}$. This notion is expressed by Formula (2.9).

$$p(P) = \sum_{A,B\in[c],A\le B} p_{AB}(P) \tag{2.6}$$

$$p(P) = \sum_{A,B\in[c],A\le B} \frac{1}{m_{AB}} \cdot p_{AB}(P) \tag{2.7}$$

$$p(P) = \sum_{A,B\in[c],A\le B} (p_{AB}(P))^2 \tag{2.8}$$

$$p(P) = \sum_{A,B\in[c],A\le B} m_{AB} \cdot \left(\frac{p_{AB}(P)}{d^{\max}_{P,A,B}}\right)^2 \tag{2.9}$$

**Examples.** We now give some examples on the use of subgraph relaxations in literature. For each example, we need to specify the following modeling choices:

- Whether ideal or average graphs are used (and how average is defined).
- Which edit distance is used.
- How $p(P)$ is combined from the $p_{AB}(P)$.

**Example (Cluster Performance).** The *performance* of a clustering counts the number of missing edges within the clusters and adds the number of existing edges between the clusters. It is hence a measure for structural partitions with image matrix fixed to the identity matrix (but can easily be generalized to the non-fixed case). According to our classification, ideal graphs are used, the edit distance measures edge differences, and $p(P)$ is simply the sum of the $p_{AB}(P)$.

**Example (Maximal Cluster Density.)** A basic measure for the quality of a clique partition on a graph $G = (V,E)$ is the sum over all *intra-cluster densities* $\delta_{int}(V_i)$. They give the proportion of actual edges to theoretically possible edges within the $i$-th cluster:

$$\delta_{int}(V_i) = \frac{\#\text{ internal edges of } V_i}{|V_i|(|V_i|-1)/2}.$$

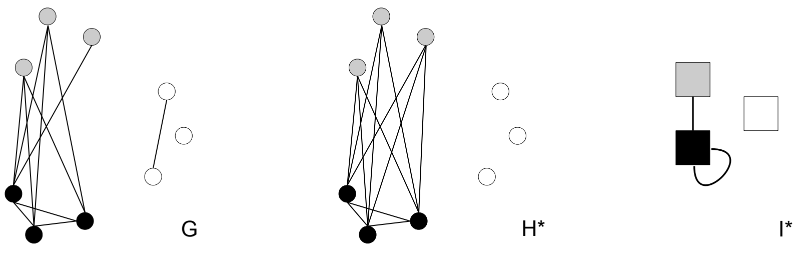


Figure 2.2: Example for distance function (2.3) when applied to a structural 3-partition problem.

The search for a partition $P^*$ with maximum total intra-cluster density has the form of the Blockmodeling Problem for structural partitions with image matrix fixed to the identity matrix. Ideal graphs are used, the edit distance measures edge differences, and the penalty values $p_{AB}(P)$ are linearly combined by Formula (2.7).

**Example (Maximal Structural Density.)** Wasserman and Faust explain a simple measure for structural partitions in their survey [WF94]. It is a generalization of the preceding example from clique to structural partitions. For each group pair $(V_A, V_B)$, they sum up the values $|I_{AB} - \Delta_{AB}|$. Here, $I$ denotes the image matrix and $\Delta_{AB}$ denotes the density. The density is defined as the number of edges from vertices in $V_A$ vertices in $V_B$, divided by the maximum possible number $m_{AB}$ of such edges. Hence, ideal graphs are used, the edit distance measures edge differences, and the penalties $p_{AB}(P)$ are linearly combined by Formula (2.7).

**Example (Newman-Girvan-Modularity.)** Newman and Girvan [NG04] present a relaxation for clusterings (structural partitions with image matrix fixed to the sparse identity matrix). That is, the groups are required to be dense, whereas the inter-group connections are not taken into account. $\mathbf{H}_{P,A,A}$ contains *average* graphs. More precisely, $\mathbf{H}_{P,A,A}$ consists of exactly one graph $H = (A, F)$. It is a complete graph and the edge weight of $uv \in F$ is $deg_G(u)deg_G(v)/2|E|$. This is precisely the expected number of edges between $u$ and $v$ in a random graph with the same degree distribution as $G$. For this reason, $H$ can be interpreted as the average graph w.r.t. the degree distribution of $G$. Hence, average graphs are used, edges are relaxed, and the penalties $p_{AA}(P)$ are simply summed up as in Formula (2.6). The value $p(P)/2|E|$ is called the *modularity* of $P$. The factor $1/2|E|$ is however constant and can thus be ignored in the solution of the Blockmodeling Problem. Its purpose is to scale the result to a value between $-1$ and $+1$. Other so-called *Newman-like modularities* can be modeled analogously.

**Example (Berkowitz-Carrington-Heil Index.)** The index [CHB80] is designed for structural partitions. The set $\mathbf{H}_{P,A,B}$ contains a single average graph $H$. The user is asked to specify an average density $\alpha$ from the interval between 0 and 1. The graph $H$ is then the complete (bipartite) graph with edge weights all $\alpha$, letting its density equal $\alpha$. The distance function $d$ is simply (2.3). Since the index is a $\chi^2$ approach, the function $p(P)$ is composed as in (2.9).

**Example (Vertex Relaxation.)** Batagelj et. al. [BDF92] relax vertices for regular partitions. They use ideal graphs, relax vertices, and simply sum up the penalties $p_{AB}(P)$. However, they restrict the set $\mathbf{H}_{P,A,B}$ of ideal graphs by allowing only those $H \in \mathbf{H}_{P,A,B}$ for which it holds that whenever vertex $u$ has been deleted from $G_{P,A,B}$ and there is an edge $uv \in E$, then $v$ must have been deleted as well. An optimization heuristic for this function is implemented in UCINET [BEF02]. Brusco and Steinley [BS09] present an exact optimization algorithm based on an integer programming model.

## 2.3 THE SPACE OF IDEAL BLOCKMODELS

On a given graph $G$, there can be several ideal structural (resp. regular) blockmodels. In this section, we analyze the space of all ideal blockmodels on a fixed graph $G$. The structure of this space will contribute to the computational complexity proofs in the following section.

To start with, observe that an ideal structural partition stays ideal if any of its groups is split into two groups. The same holds for ideal *regular* blockmodels $(P,I)$ in case that $I$ has a zero diagonal. Before we prove this observation, let us formalize the split procedure. It turns a blockmodel $B$ into a blockmodel $B'$ with one group more, by splitting one of the groups in $B$ into two groups. To this end, a subset $C$ of the vertex set of one group $V_i$ in $B$ needs to be specified. The following algorithm then splits this group into the two subgroups $C$ and $V_i \setminus C$. The image matrix $I$ is updated by adding a new row and a new column.

```
SPLIT$(B,i,C)$
Input: A blockmodel $B = (P,I)$, an index $i \in \{1,\dots,|P|\}$, and a subset $C \subset V_i$, $\emptyset \neq C \neq V_i$.
Output: A blockmodel $B' = (P',I')$.

    1. Set $P' := (V_1,\dots,V_{i-1},V_i \setminus C, V_{i+1},\dots,V_{|P|},C)$.

    2. Set $I'$ to a $|P'| \times |P'|$ binary matrix with
       $I'_{cd} := \begin{cases} I_{cd} & \text{if } c,d \in \{1,\dots,|P|\}, \\ I_{id} & \text{if } c = |P+1|, d \in \{1,\dots,|P|\}, \\ I_{di} & \text{if } d = |P+1|, c \in \{1,\dots,|P|\}, \\ I_{ii} & \text{if } c = d = |P|+1. \end{cases}$

    3. Return $(P',I')$.
```

In Step 1, the partition $P$ is updated. The old group $V_i$ is replaced by its subgroup $V_i \setminus C$. Furthermore, its complement group $C$ is added to the end of $P$. In Step 2, the image matrix remains unchanged for all partition pairs that do not involve $V_i$. Otherwise, the two new groups inherit the image matrix entries from their parent group $V_i$. That is, for every group $V_j$ with $j \neq i$, the image matrix entry for the pairs $(C,V_j)$ and $(V_i \setminus C, V_j)$ equals the entry for $(V_i,V_j)$.

According to the following statement, the split transformation preserves ideality for structural blockmodels. This observation can be found in several textbooks including Scott [Sco02].

**Proposition 3.** Let $B = (P,I)$ denote an ideal structural blockmodel. The blockmodel SPLIT$(B,i,C)$ is ideal for all feasible choices of $i$ and $C$. □

We make similar observations for regular blockmodels.

**Proposition 4.** Let $B = (P,I)$ denote a blockmodel and $i$ an index such that $I_{ii} = 0$. For all feasible choices of $C$, if the blockmodel SPLIT$(B,i,C)$ is regularly ideal then $B$ is regularly ideal. □

### 2.3.1 **Order-Theoretic Lattices**

For every graph $G$, the ideal blockmodel which requires the lowest number of groups is in fact uniquely determined. Moreover, all other ideal blockmodels can be obtained from this blockmodel by a sequence of calls of the SPLIT algorithm. More precisely, the space of all ideal blockmodels forms a *lattice*.

**Definition 5.** (Join, Meet) Given a partially ordered set $(X, \leq)$, the *meet* $x_1 \wedge x_2$ of the elements $x_1, x_2 \in X$ is the greatest $x_3 \in X$ with $x_3 \leq x_1, x_3 \leq x_2$. The *join* $x_1 \vee x_2$ is the least $x_3 \in X$ with $x_1 \leq x_3, x_2 \leq x_3$.

**Definition 6.** (Bounded Lattice) A partially ordered set $(X, \leq)$ is called a *lattice* if for every two elements $x_1, x_2 \in X$ both $x_1 \wedge x_2$ and $x_1 \vee x_2$ are in $X$. A lattice is *bounded* if there is a greatest element $x_G$ and a least element $x_L$, that is, $x_L \leq x \leq x_G$ for all $x \in X$.

To show that the set $(\mathbf{B}^X_{\text{Ideal}}(G), \leq)$ of all ideal blockmodels is a bounded lattice, we define a partial ordering $\leq$. For two blockmodels $B, B' \in \mathbf{B}^X_{\text{Ideal}}(G)$, we write $B = (P, I) \leq B' = (P', I')$ if for every group $V_i \in P$, there is a group $V_j \in P'$ with $V_i \subseteq V_j$. In other words, $B$ is lower than $B'$ if $B$ can be obtained from $B'$ by a sequence of split operations.

**Theorem 1.** [BE89, Sco02] The partially ordered sets $(\mathbf{B}^S_{\text{Ideal}}(G), \leq)$ and $(\mathbf{B}^R_{\text{Ideal}}(G), \leq)$ are bounded lattices.

The proof for regular blockmodels can be found in Borgatti and Everett [BE89]. Even though it is given for directed connected graphs without isolates, it can be easily generalized to our case. Scott [Sco02] states that the structural blockmodels form a sublattice of the regular lattice.

As $(\mathbf{B}^X_{\text{Ideal}}(G), \leq)$ is a bounded lattice, there is a greatest and a least element. The least one is the blockmodel in which every vertex defines its own group and the image matrix is the adjacency matrix of $G$. This holds for both structural and regular ideal blockmodels and for both directed and undirected graphs and also for (di)graphs with isolated vertices. The greatest element depends however on the case under consideration.

**Proposition 5.** The greatest element $B = (P, I)$ of the bounded lattice $(\mathbf{B}^X_{\text{Ideal}}(G), \leq)$ is defined in the following way:

a) structural ($X = S$): $B$ is the ideal blockmodel with $P(u) = P(v)$ if and only if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$ for all $u, v \in V$.

b) regular ($X = S$): $B$ is the ideal blockmodel in which $P$ consists of two groups: The isolated and the non-isolated vertices in $G$. $\square$

The proof follows from the observations that the respective blockmodel is in the lattice (it is ideal) and that ideality is lost by the merging of any groups. Note that the greatest element for regular blockmodels is non-trivial if directed graphs are considered. It can however still be computed in polynomial time by the algorithm CATREGE [BE93].

Since splitting preserves ideality, we can use the SPLIT procedure to decide whether an ideal $c$-partition exists on a given graph $G$ and for a given natural number $c$.

```
STRUCTURAL-C-PARTITION($G, c$)
Input: A graph $G \in \mathbf{G}$, a natural number $1 \leq c \leq |V|$.
Output: An ideal structural blockmodel on $G$ with $c$ groups (if one exists).
    1. Compute the greatest element $B = (P, I)$ of the bounded lattice $(\mathbf{B}^S_{\text{Ideal}}(G), \leq)$.
    2. While $|P| < c$:
        2a. Choose $i$ with $|V_i| \geq 2$ and $C \subset V_i$ with $\emptyset \neq C \neq V_i$.
        2b. Set $B =$ SPLIT($B, i, C$).
    3. If $|P| = c$: Return $B$. Else: "There is no such blockmodel."
```

## 2.4 COMPUTATIONAL COMPLEXITY OF QUALITY OPTIMIZATION

In this section, we discuss the computational hardness of minimizing the penalty functions introduced in this chapter. More precisely, we examine the hardness of the following family of optimization problems and their decision variants. Recall that the family is parameterized by a set $\mathbf{F} \subseteq \mathbf{B}$ of feasible blockmodels and a blockmodel penalty function $p : \mathbf{B} \to \mathbb{R}$.

BLOCK $(\mathbf{F}, p)$.
Instance: A graph $G = (V, E)$.
Task: Minimize $p$ over $\mathbf{F} \cap \mathbf{B}(G)$.

BLOCK-DECISION $(\mathbf{F}, p)$.
Instance: A graph $G = (V, E)$, a number $k \in \mathbb{Z}$.
Question: Is there a $B \in \mathbf{F} \cap \mathbf{B}(G)$ with $p(B) \leq k$?

We will see that the hardness proofs for problem BLOCK$(\mathbf{F}, p)$ mainly depend on the feasible set $\mathbf{F}$. They do however not depend on the precise form of function $p$. They use only the fact that $p$ has the property that *ideal* blockmodels are the optimal ones. We will state common NP-hardness proofs for all functions with this property. Since we have shown that the functions $p$ in literature are designed to measure deviations from ideality, this property is in fact very common. Our proofs can thus be applied simultaneously to most of the penalty functions discussed in this chapter.
In Section 2.4.1, we formalize the common property of the penalty functions and show that almost all presented functions share it. In Section 2.4.2, we give the NP-hardness proofs for those functions.

### 2.4.1 Sensitive Objective Functions

The following definition of *sensitive* functions is introduced by Doreian et al. [DBF05] in the more specific context of direct blockmodeling approaches.

**Definition 7.** A penalty function $p : \mathbf{B} \to \mathbb{R}_0^+$ is called *structurally* (resp. *regularly*) *sensitive* if for every blockmodel $B \in \mathbf{B}$ holds that $p(B) = 0$ if and only if $B \in \mathbf{B}^S_{\text{Ideal}}$ (resp. $B \in \mathbf{B}^R_{\text{Ideal}}$).

As we have shown that the blockmodel quality functions $q$ in literature can all be seen as penalty functions, which penalize deviations from ideal blockmodels, it is suggesting that these functions are sensitive. We will now show formally that this indeed holds for most of them.

**Node Pair Relaxations.** We show that the dissimilarity functions introduced in Section 2.2.4 are sensitive. It is easy to see that a function of the form $p(P) = \sum_{u,v \in V} p_{uv} \delta(P(u), P(v))$ is structurally sensitive if

1. $p_{uv} = 0$ if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$,

2. $p_{uv} > 0$ otherwise.

Consider an ideal structural blockmodel. Then, $N(u) \setminus \{v\} = N(v) \setminus \{u\}$ holds for all vertex pairs $u, v$ in the same group. Hence, $p(P) = \sum_{u,v \in V, P(u)=P(v)} p_{uv} = 0$. If the blockmodel is however not ideal structural, there is at least one vertex pair $u, v$ in the same group with $N(u) \setminus \{v\} \neq N(v) \setminus \{u\}$. Hence, $p(P) > 0$. A penalty function $p$ satisfying Conditions 1 and 2 is hence sensitive. These conditions apply to both presented vertex similarity approaches. In Formula (2.1), that is,

$$p_{uv} = \frac{|(N(u) \setminus v) \Delta (N(v) \setminus u)|}{k},$$

the symmetric difference is empty if and only if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. In case that $k = 0$, which means that both $u$ and $v$ do not have any neighbors at all, we defined $p_{uv} = 0$. Thus, the conditions apply. It is easy to see that also the Euclidian distance given by Formula (2.2) satisfies the conditions.

**Subgraph Relaxations.** We have seen in Section 2.2.5 that all subgraph relaxation approaches can be characterized by three modeling choices: Whether ideal or average graphs are used, which edit distance $d$ is used, and how the total penalty $p$ is combined from the group pair penalties $p_{AB}$. Such a total penalty function is sensitive if the following conditions hold:

1. ideal graphs are used,

2. $d$ and $p$ are non-negative: $d(G, H) \geq 0$, $p(P) \geq 0$,

3. $d$ is coincident: $d(G, H) = 0$ if and only if $G = H$,

4. $p(P) = 0$ if and only if $p_{AC}(P) = 0$ for all $V_A, V_C \in P$.

To see this, let $B = (P, I)$ denote an ideal structural or regular blockmodel. The distance value $d(G_{P,A,B}, \mathbf{H}_{P,A,B})$ equals 0 for every group pair $V_A, V_C \in P$. The reason is that $\mathbf{H}_{P,A,B}$ contains all ideal subgraphs (Definition see Page 20) and Condition 3 applies. Because of Condition 4, the total penalty $p(P)$ equals 0 in this case. Otherwise, if $B$ is not ideal, the total penalty is greater than 0 for a analogous argument. It is obvious that $d$ is non-negative and coincident for all presented edit distances (2.3), (2.4), and (2.5). Furthermore, the non-negativity and Condition 4 apply to all presented functions for $p$, that is, for Function (2.6) to (2.9). In case that ideal graphs are used, we conclude that all presented approaches have sensitive penalty functions.

### 2.4.2 NP-Hardness Proofs

Having shown that most of the presented penalty functions have the property of being *sensitive*, we can now state complexity results for the optimization of sensitive functions. The hardness of Problem BLOCK($\mathbf{F}, p$) (with a sensitive penalty function $p$) thereby depends on the choice of the set $\mathbf{F}$ of feasible blockmodels. We treat the most common definitions of $\mathbf{F}$: Only blockmodels with a fixed number of groups, only blockmodels with a fixed image-matrix, only blockmodels with non-empty classes, and only blockmodels without loops in the image graph. We formalize these sets with the following notation:

- $\mathbf{B}_c$: All $(P,I) \in \mathbf{B}$ where $P$ is a $c$-partition.
- $\mathbf{B}_I$: All $(P,I') \in \mathbf{B}$ where $I' = I$.
- $\mathbf{B}_{\text{non-empty}}$: All $(P,I) \in \mathbf{B}$ with $P = (V_1, \dots, V_k)$, where $|V_i| \geq 1$ for $i = 1, \dots, k$.
- $\mathbf{B}_{\text{no-loop}}$: All $(P,I) \in \mathbf{B}$, where $\text{diag}(I) = 0$.

If all blockmodels are chosen to be feasible ($\mathbf{F} = \mathbf{B}$), the Blockmodeling Problem is trivial, as the solution in which every vertex form its own group is already optimal:

**Proposition 6.** Let $p : \mathbf{B} \to \mathbb{R}$ be sensitive for $\mathbf{B}$. The blockmodel $(P,I) \in \mathbf{B}(G)$ with $P = V(G)$ and $I = Adj(G)$ is then an optimum solution the instance $G$ to BLOCK($\mathbf{B}, p$).

*Proof.* The proof follows from the fact that the stated blockmodel $(P,I)$ is both a structural and a regular partition. □

The Blockmodeling Problem turns however non-trivial if the number of groups is fixed to a natural number $c \geq 2$. In this case, the set $\mathbf{F}$ contains only those blockmodels $(P,I)$ where $P$ is a $c$-partition and $I$ a $c \times c$ matrix.

**Proposition 7.** Let $p : \mathbf{B} \to \mathbb{R}$ be regularly sensitive and $\mathbf{F} = \mathbf{B}_c$ with $c \geq 2$. The problem BLOCK($\mathbf{F}, p$) is NP-hard.

*Proof.* Fiala and Paulusma [FP03] prove that the following decision problem is NP-complete for all $c \geq 2$: Given a graph $G = (V,E)$, is there an ideal regular $c$-partition on $G$? This problem can be transformed to the instance $(G', k)$ of BLOCK DECISION($\mathbf{F}, p$) with sensitive $p$ in the following way. Set $G' = G$ and $k = 0$. If and only if the answer is "yes", there is a blockmodel $B^*$ with $p(B^*) = 0$ and $B^*$ is hence an ideal regular $c$-partition. The answer to Fiala and Paulusma's problem is hence "yes" if and only if the answer to BLOCK DECISION($\mathbf{F}, p$) is "yes". □

The problem stays hard if not only the number of groups $c$, but the whole image matrix $I$ is fixed. In this case, the hardness depends on the structure of the fixed image matrix:

**Proposition 8.** Let $p : \mathbf{B} \to \mathbb{R}$ be regularly sensitive and $\mathbf{F} = \mathbf{B}_I$. The problem BLOCK($\mathbf{F}, p$) is NP-hard in both of the following cases:

a) the image graph of $I$ is connected and has at least 3 vertices,

b) the image graph of $I$ has 2 vertices $u$ and $v$. The edges $uv$ and $vv$ exist ($uu$ may exist or not).

*Proof.* a) Fiala and Paulusma [FP03] show that the following decision problem is NP-complete: Given a graph $G$ and an image matrix $I$ as in a), is there an ideal regular blockmodel with image matrix $I$? It can be transformed to BLOCK DECISION$(\mathbf{F}, p)$ as in the proof of Proposition 7. b) Roberts and Sheng [RS01] show that the following decision problem is NP-complete: Given a graph $G$ and one of the two image matrices $I$ as in b), is there an ideal regular blockmodel with image matrix $I$? The transformation is the same as in Proposition 7. □

The following two propositions can be proven analogously. That is, there are polynomial transformations from problems which have been proven to be NP-complete by Fiala and Paulusma [FP03].

**Proposition 9.** Let $p : \mathbf{B} \to \mathbb{R}$ be regularly sensitive and $\mathbf{F} = \mathbf{B}_I \cap \mathbf{B}_{\text{non-empty}}$. BLOCK$(\mathbf{F}, p)$ is NP-hard if the image graph of $I$ is not connected, but has a connected component with at least 3 vertices. □

**Proposition 10.** Let $p : \mathbf{B} \to \mathbb{R}$ be regularly sensitive and $\mathbf{F} = \mathbf{B}_c \cap \mathbf{B}_{\text{no-loop}}$. BLOCK$(\mathbf{F}, p)$ is NP-hard. □

The preceeding hardness results treated the case of *regular* blockmodels. We will now see that there are also structural problems which are NP-hard:

**Proposition 11.** Let $p : \mathbf{B} \to \mathbb{R}$ be structurally sensitive and $\mathbf{F} = \mathbf{B}_I \cap \mathbf{B}_{\text{non-empty}}$. BLOCK$(\mathbf{F}, p)$ is NP-hard if $I$ is the sparse identity matrix in $\{-, 1\}^{c \times c}$.

*Proof.* If $I$ is the sparse $c \times c$ identity matrix, then the elements $(P, I)$ of $\mathbf{B}_{\text{Ideal}}$ are exactly those blockmodels where $P$ is a *c-clique cover*. The decision problem whether $G$ has a $c$-clique cover is NP-complete, see Karp [Kar72]. Due to a transformation analogous to the one in the proof of Proposition 7, the result follows. □

We conclude the section with the observation that the problem is polynomially solvable in two cases where only ideal blockmodels are considered feasible:

**Proposition 12.** Let $p : \mathbf{B} \to \mathbb{R}$ be the group counting function $p(P, I) = |P|$. BLOCK$(\mathbf{B}^X_{\text{Ideal}}, p)$ is polynomially solvable.

*Proof.* In this case, the optimum solutions are the greatest elements of the lattice of all ideal blockmodels (see Section 2.3). It follows from Proposition 5 that they can be computed in polynomial time. □

**Proposition 13.** Let $p : \mathbf{B} \to \mathbb{R}^+_0$ be structurally sensitive and $\mathbf{F} = \mathbf{B}^S_{\text{Ideal}} \cap \mathbf{B}_c$. The problem BLOCK$(\mathbf{F}, p)$ is polynomially solvable.

*Proof.* The algorithm STRUCTURAL-C-PARTITION$(G, c)$ on Page 26 computes an ideal structural blockmodel with $c$ groups in polynomial time. This blockmodel is the optimum solution to BLOCK$(\mathbf{F}, p)$. □

# Chapter 3

# VARIABLES IN INTEGER PROGRAMS FOR PATTERN SEARCH PROBLEMS

We have seen in Chapter 2 that pattern search is in practice performed in the following way: A penalty function $p$ is defined, which is minimized over a set of feasible blockmodels. A block-model $(P,I)$ consists of a partition $P$ of the graph's vertices and its pattern interpretation, an image matrix $I$.

We are interested in algorithms that either solve these problems to optimality or give a quality guarantee on the best solution found. To this end, we consider integer programming formulations. Such formulations need basic variables to model the partition $P$ of the vertices into groups. There are hence only a few possibilities to define these basic variables, even though the problems can largely differ in their objective functions or constraints.

In this chapter, we will treat four different kinds of basic variables: Vertex assignment variables $x$, newly introduced edge assignment variables $y$, equivalence relation variables $s$, and vertex subset variables $w$. They are taken from a survey including [AK08, BS09, CDW08, FMdS$^+$96, JMN93, KA91, MT98, XTP07]. We discuss each variable type separately: Its variable defining constraints, the ways to model the set of feasible blockmodels, and derive strong formulations from polyhedral studies. In Section 3.2.3, we present new methods to efficiently combine variable types.

## 3.1 VERTEX ASSIGNMENT VARIABLES $x$

In integer programs for clustering or blockmodeling problems, vertex assignment variables $x$ are most commonly used. For example, they are used by [FMdS$^+$96] and [JMN93] for clustering and by [BS09] for structural and regular equivalence. These variables indicate whether vertex $u$ is assigned to group $V_A$:

$$x_{uA} = \begin{cases} 1 & \text{if } P(u) = A, \\ 0 & \text{otherwise,} \end{cases}$$

for all $u \in V, A \in [c]$. The corresponding assignment constraints are the following ones, which simply state that every vertex must be assigned to a group.

$$\sum_{c \in C} x_{uA} = 1 \text{ for all } u \in V, \tag{3.1}$$

$$x_{uA} \in \{0,1\} \text{ for all } u \in V, A \in [c], \tag{3.2}$$

Usually, further constraints are added to restrict the number or sizes of groups.

### 3.1.1 Constraints on Group Sizes and Numbers of Groups

A common constraint on the set **F** of feasible blockmodels requires the number of groups to be fixed to an a priorily given number $c$. This restriction can be naturally modeled with $x$-variables. One needs to add the constraint

$$\sum_{u \in V} x_{uA} \geq 1 \text{ for all } A \in [c], \tag{3.3}$$

stating that every group in $P$ needs to contain at least one vertex. An alternative set of constraints is presented by Vinod [Vin69] and applied by Berry et al. [BHL$^+$07]. It is inspired by warehouse location models. Vertices are not assigned to groups, but to group leader vertices. Every group consists of its group leader and possibly some follower vertices. The basic variables $x'_{uv}$ hence take the value 1 if and only if vertex $u$ follows vertex $v$. We set the variable $L_v$ to 1 if vertex $v$ is a group leader, and 0 otherwise. The constraint set is then the following one: Constraint (3.5) states that there are exactly $k$ leaders. That is, the solution will have exactly $k$ groups. Constraint (3.6) states: If vertex $u$ is a follower of $v$, then $v$ must be a group leader. See Figure 3.1 for an example.

$$\sum_{v \in V} x'_{uv} = 1 \text{ for all } u \in V, \tag{3.4}$$

$$\sum_{v \in V} L_v = k, \tag{3.5}$$

$$\sum_{u \in V} x'_{uv} \leq |V| \cdot L_v \text{ for all } v \in V, \tag{3.6}$$

$$x'_{uv} \in \{0,1\} \text{ for all } u, v \in V, \tag{3.7}$$

$$L_v \in \{0,1\} \text{ for all } v \in V. \tag{3.8}$$

Another common constraint on **F** models the restriction of group sizes. Groups are required to have a minimum size $k$. Frequent special cases are the demand for non-empty groups ($k = 1$) and for groups of almost equal size ($k = \lfloor |V|/c \rfloor$). Ji [Ji04] examines such integer programming models for complete graphs $G$. For $x$-variables, the following constraint states that the number of nodes in group $V_A$ is at least $k$:

$$\sum_{u \in V} x_{uA} \geq k \text{ for all } A \in [c]. \tag{3.9}$$

### 3.1.2 Symmetry Breaking Constraints

Every partition $P = (V_1, \ldots, V_c)$ is represented by $c!$ distinct feasible $x$-vectors due to permutations of the group identification numbers $1, \ldots, c$. For every partition $P$, let $X_P$ denote the set of all

$x$-vectors representing $P$. A constraint set is called symmetry breaking if for every partition $P$, the constraint set is satisfied by exactly one vector in $X_P$. Such a symmetry breaking constraint set is proposed by Klein and Aronson [KA91] and later used by Xu et al. [XTP07]. Their constraint set models two statements:

1. Vertex $v$ can only be in a group $V_k$ with $k \in P_v := \{1, 2, \dots, \min\{v, c\}\}$. E. g., vertex 1 must be in group 1, vertex 2 must be either in group 1 or 2, and so on.

2. If a vertex $v \geq 3$ is in $V_k$, then there must be a vertex $u < v$ that is in group $V_{k-1}$.

$P_v$ denotes the set of all groups that vertex $v$ can be assigned to according to Statement 1. Let conversely denote $P_k^{-1}$ the set of all vertices that can be assigned to group $V_k$. The two statements above can then be modeled by

$$\sum_{A \in P_v} x_{vA} = 1 \text{ for all } v \in V, \tag{3.10}$$

$$\sum_{\substack{u \in P^{-1}(k-1),\\ u<v}} x_{u(k-1)} \geq x_{vk} \text{ for all } v \in V, v \geq 3, V_k \in P_v. \tag{3.11}$$

The above constraints have been applied to clustering problems only. They can also be applied to models for a problem BLOCK $(\mathbf{F}, p)$. in case that the set $\mathbf{F}$ of feasible blockmodels in the set $\mathbf{B}$ of all blockmodels. The reason is that an ideal structural partition stays ideal structural if the group identification numbers are permuted; the same holds for regular partitions. We now discuss the possibilities in case that the set $\mathbf{F}$ is restricted.

**Group size constraints.** Assume $\mathbf{F}$ is restricted by group size constraints. That is, there is a maximum size $s_k$ specified for every vertex group $V_k$. Given a feasible vector, a permutation of the group ID numbers could then turn the vector infeasible. In this case, the above constraints cannot be used for symmetry breaking, unless all groups have the same requirement. We propose the following alternative. Assume w.l.o.g. that the groups are non-increasingly sorted by their maximum size limits: $s_1 \geq s_2 \geq \cdots \geq s_c$. The following constraint models that group $k$ must be at

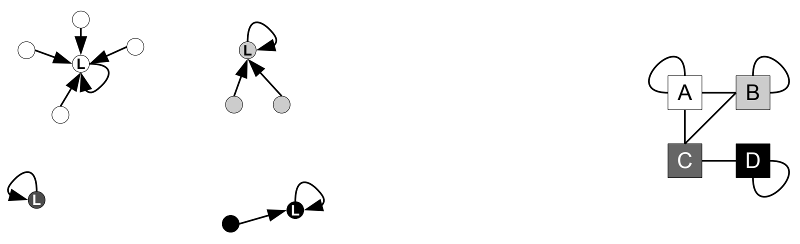


Figure 3.1: Left: Four groups defined by 4 leader vertices. Right: Interchanging the labels of groups $V_A$ and $V_B$ does not affect the feasibility of a solution, as they have exactly the same neighbor groups $V_A, V_B$, and $V_C$ in the image graph.

least as large as group $k+1$:

$$\sum_{v=1}^{|V|} x_{vk} \geq \sum_{v=1}^{|V|} x_{v(k+1)} \text{ for all } k = 1, \ldots, c-1. \tag{3.12}$$

If an $x$-vector describing a partition $P$ violates the constraint, all permutation vectors in $X_P$ violate it as well. For every feasible partition $P$, there is hence a vector in $X_P$ satisfying the constraint. Note however that the constraint is not symmetry breaking since there can be several vectors in $X_P$ satisfying the constraint. This is due to the fact that *same-sized* groups can still be permuted without violating the constraint.

**Fixed image matrix.** Consider the case where the image matrix is fixed a priorily, but no further restrictions are imposed on $\mathbf{F}$. If the vertices in group 1 are required to have at least one neighbor in group 3, whereas the vertices in 2 are required to have no neighbors in 3, then the interchanging of groups IDs 1 and 2 can turn a feasible vector into an infeasible one. Clearly, only those groups $V_{k_1}$ and $V_{k_2}$ may be permuted which have the same requirements concerning the other groups. That is, $k_1$ and $k_2$ need to have exactly the same neighbors the image graph. Figure 3.1 (right) gives an example. Let $P_1, P_2, \ldots, P_s$ denote a partition of the set $P$ of all groups into classes, such that the groups in $P_i$ pairwisely have exactly the same neighbors in the image graph. Denote by $P_{i1}, P_{i2}, \ldots$ the elements of $P_i$. The following constraints can be used to reduce the amount of symmetry:

$$\sum_{v=1}^{|V|} x_{u,ik} \geq \sum_{v=1}^{|V|} x_{v,i(k+1)} \text{ for all } i = 1, \ldots, s, k = 1, \ldots, |P_i| - 1. \tag{3.13}$$

The constraints models that the groups in $P_i$ need to be ordered by size, for every $i = 1, \ldots, s$. It is again not symmetry breaking as it allows same-sized groups to be permuted.

## 3.2 EDGE ASSIGNMENT VARIABLES $y$

Objective functions of pattern search problems usually model statements on edges (or non-edges) of the graph. For example, it is necessary to count the number of edges between certain groups in case that density patterns are searched for. For patterns of link existence, constraints demanding the existence of certain edges must be modeled. In these cases, so-called edge assignment variables are used in the integer programming model.
For two distinct vertices $u, v$ and two vertex groups $V_A, V_B$, we set the variable $y_{uA,vB}$ to 1 if both $u$ is in $V_A$ and $v$ is in $V_B$:

$$y_{uA,vB} = \begin{cases} 1 & \text{if } P(u) = A \text{ and } P(v) = B, \\ 0 & \text{otherwise,} \end{cases}$$

For ease of notation, we use both $y_{uA,vB}$ and $y_{vB,uA}$ to denote the same variable in the remainder of the thesis. In practical integer programming formulations, often only a subset of the $y$-variables are actually used. That is, they are introduced not for *all* vertex pairs, but only for a subset $E_Y$. Let $K_n = (V_n, E_n)$ denote the complete graph on $n$ vertices. For a subset $E_Y \subset E_n$ of its edges, we define the set $Y(E_Y) := \{y_{uA,vB} \mid uv \in E_Y, A, B \in [c]\}$.

Consider a graph $G = (V,E)$ to be searched for patterns. We will show in this section that it is sufficient to define the $y$-variables only on the edge set $E$ to define a partition. They are furthermore sufficient to formulate most of the common constraints. In this case, $E_Y = E$ holds for undirected graphs. In case of a directed graph $D = (V,A)$, the $y$ variables need to be defined on the edge set of the underlying graph of $D$. It is the undirected graph which evolves from $D$ by ignoring the directions of its arcs:

**Definition 8.** Given a directed graph $D = (V,A)$, its undirected *underlying graph* $G_D = (V,E)$ is defined by $E := \{uv \mid (u,v) \in A \text{ or } (v,u) \in A\}$.

An example for an underlying graph is given in Figure 3.2. In our survey, $y$-variables are only used in combination with $x$ variables. However, we present a way to use $y$-variables exclusively. In this formulation, the variables do not need to be defined for every vertex pair, but only for those on a given set $E_Y$ of vertex pairs. To prove the validity of the subsequent model, we need to define *partition vectors* first.

**Definition 9.** Given the complete graph $K_n = (V_n,E_n)$, an edge set $E_Y \subseteq E_n$, and a number $c$ of groups, a binary vector $\{x_{uA}\}_{u\in V,A\in[c]}$ is called a *partition vector* if it satisfies Constraint (3.1). A binary vector $\{y_{uA,vB}\}_{uv\in E_Y,A,B\in[c]}$ is called a *partition vector* if there is exactly one partition vector $\{x_{uC}\}_{u\in V,C\in[c]}$ with $x_{uA}x_{vB} = y_{uA,vB}$ for all $uv \in E_Y, A,B \in [c]$. This vector $x$ is called the *corresponding partition vector* of $y$. We call $(x,y)$ a *partition vector* if $y$ is a partition vector and $x$ is its corresponding partition vector.

**Proposition 14.** Given the complete graph $K_n = (V_n,E_n)$, an edge set $E_Y \subseteq E_n$, and a number $c$ of groups, such that the graph $(V,E_Y)$ has no isolated vertices, then a vector $\{y_{uA,vB}\}_{uv\in E_Y,A,B\in[c]}$ is a partition vector if and only if it satisfies the following constraints:

$$\sum_{A,B\in[c]} y_{uA,vB} = 1 \text{ for all } uv \in E_Y, \tag{3.14}$$

$$\sum_{B\in[c]} y_{uA,vB} = \sum_{B\in[c]} y_{uA,wB} \text{ for all } uv,uw \in E_Y, v \neq w, A \in [c], \tag{3.15}$$

$$y_{uA,vB} \in \{0,1\} \text{ for all } uv \in E_Y, A,B \in [c]. \tag{3.16}$$

*Proof.* Clearly, every partition vector $y$ satisfies the above constraints. To prove the opposite direction, consider a vector $y$ satisfying the constraints. We need to show that there is a unique partition vector $x$ corresponding to $y$. We construct $x$ by the following algorithm. Set $x$ to the zero vector. Then, for each $y_{uA,vB}$ that equals 1, set $x_{uA} = 1$ and $x_{vB} = 1$. We now show that $x$

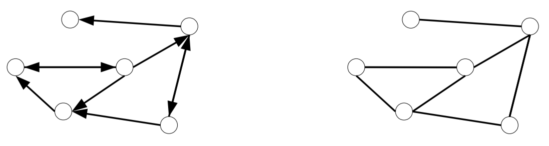


Figure 3.2: A directed graph (left) and its undirected underlying graph.

is a partition vector. Assume $\sum_{A\in[c]} x_{uA} = 0$ for a $u \in V$. Then, $\sum_{A,B\in[c]} y_{uA,vB} = 0$ for a $v$, which contradicts Constraint (3.14). Now assume $\sum_{A\in[c]} x_{uA} \geq 2$ for a $u \in V$. Then, both $x_{uA} = 1$ and $x_{uA'} = 1$ for two different groups $V_A$ and $V_{A'}$. Hence, $y_{uA,vB} = 1$ (*) and $y_{uA',wC} = 1$ (**) for two $y$-variables. By subtracting (**) from Constraint (3.14), we conclude that $y_{uA,wD} = 0$ (***) for all $D \in [c]$. We obtain the contradiction

$$1 \leq \sum_{D\in[c]} y_{uA,vD} = \sum_{D\in[c]} y_{uA,wD} = 0.$$

Here, the inequality holds because of (*), the first equation because of Constraint (3.15), and the second one because of (***). Hence, $y_{uA,vB}$ and $y_{uA',wF}$ cannot both be 1 at the same time. Thus, $\sum_{A\in[c]} x_{uA} = 1$ holds for all $u$. Hence, $x$ is a partition vector.
Secondly, we need to show that $x_{uA}x_{vB} = y_{uA,vB}$ holds. For $y_{uA,vB} = 1$, it holds due to the construction of $x$. For $y_{uA,vB} = 0$, there must be a variable $y_{uA',vB'} = 1$ with $A' \neq A$ because of Constraint (3.14). The latter one implies $x_{uA'} = 1$. As $x$ is a partition vector, $x_{uA} = 0$ follows, which proves the required relation. Thirdly, we need to show that $x$ is the only partition vector for which this relation holds: Assigning $u$ to $V_{A'}$ instead of $V_A$ would violate the constraint $x_{uA}x_{vB} = y_{uA,vB}$ for a $v$ in $V_B$. □

The number of constraints is in $O(c|E_Y|^2)$, as it is dominated by Constraint (3.15). This constraint is an equation for each pair $(uv, uw)$ of edges incident to the same vertex $u$. Let $E(u) = \{e_1, e_2, \ldots, e_{N(u)}\}$ denote the set of all edges incident to $u$. It is clearly sufficient to state the equality for the pairs $(e_1,e_2),(e_2,e_3),\ldots,(e_{N(u)-1},e_{N(u)})$, as the remaining ones are implied. The total number of constraints is then in $O(c|E_Y|)$.
Being a special case of Proposition 14, we conclude that it is sufficient to introduce the $y$-variables only on the edges of the graph (or its underlying graph) in case that it does not contain isolated vertices. Even a subset of the edge set can be sufficient to model the partition itself. Note however, that the modeling of the objective function can require more variables.

### 3.2.1 **Group Size and Symmetry Breaking Constraints**

All constraints which can be expressed in $x$-variables can be expressed in $y$-variables as well. If $y$ is a partition vector, then the sum

$$\sum_{B\in[c]} y_{uA,vB} \tag{3.17}$$

is 1 if and only if vertex $u$ is in $V_A$ (for any choice of $v \neq u$). It equals 0 otherwise. Hence, the term $x_{uA}$ can be replaced by the above sum to obtain a corresponding constraint in $y$-variables. Especially, group size and symmetry breaking constraints can be expressed in $y$-variables as well.

### 3.2.2 **Strengthening the Formulation**

We show that the following constraints are valid in any formulation using $y$-variables. They are adopted from constraints for the *quadratic assignment problem (QAP)* and the *maximum cut problem (MCP)*. The first constraint is adopted from Barvinok [Bar92], who introduced it for a

QAP formulation. Let again $V$ denote the vertex set of the input graph $D = (V, A(D))$ and $E_Y$ denote the set of vertex pairs on which $y$-variables are defined.

**Proposition 15.** For any 3-cycle $vw, wv, vu$ in $(V, E_Y)$, two not necessarily distinct vertex groups $V_A, V_B$ of $P$, and a group subset $L \subseteq \{V_1, \ldots, V_c\}$ of $P$, the following inequality is valid for every partition vector $y$:

$$\sum_{V_C \in L} y_{uA,wC} + \sum_{V_C \in \{V_1,\ldots,V_c\} \setminus L} y_{vB,wC} \geq y_{uA,vB}. \tag{3.18}$$

*Proof.* Assume that the right-hand side is equal to 1, as the constraint is trivially satisfied otherwise. That is, $u$ is in $V_A$ and $v$ is in $V_B$. By using this information and by substituting each $y$-variable on the left-hand side by the product of its two $x$-variables, the left-hand side reduces to $\sum_{V_C \in L} x_{wC} + \sum_{V_C \in \{V_1,\ldots,V_c\} \setminus L} x_{wC}$. This equals $\sum_{V_C \in P} x_{wC}$, which equals 1 because of Constraint 3.1. □

Furthermore, several constraints from the *maximum cut problem* can be transferred. For a vertex pair subset $F \subseteq E_Y$ and a group subset $L \subseteq \{V_1, \ldots, V_c\}$ of $P$, let $y(F, L)$ denote the sum

$$y(F, L) := \sum_{uv \in F} \sum_{V_A \in L, V_B \in \{V_1,\ldots,V_c\} \setminus L} (y_{uA,vB} + y_{uB,vA}). \tag{3.19}$$

If $E_Y$ is interpreted as the edge set of the graph $(V, E_Y)$, then $y(F, L)$ is the sum over all edges in $F$ leaving the group $L$. See Figure 3.3 for an example.

**Proposition 16.** Let $L \subset \{V_1, \ldots, V_c\}$, $\emptyset \neq L \neq \{V_1, \ldots, V_c\}$, be a proper subset of the group set. Let $Q$ be a cycle in $(V, E_Y)$ and $F \subseteq Q$ a subset of the cycle with $|F|$ odd. Then,

$$y(F, L) - y(Q \setminus F, L) \leq |F| - 1 \tag{3.20}$$

is valid for every partition vector $y$.

*Proof.* The expression $y(\{uv\}, L)$ is in $\{0, 1\}$ for every partition vector $y$, every $uv \in E_Y$, and every $L \subset \{V_1, \ldots, V_c\}$ with $\emptyset \neq L \neq \{V_1, \ldots, V_c\}$. It is 1 if and only if exactly one vertex in $\{u, v\}$ is in a group in $L$. The remainder of the proof follows from the fact that the cycle inequalities are valid for the max-cut problem, by considering the cut with shores $L$ and $\{V_1, \ldots, V_c\} \setminus L$. □

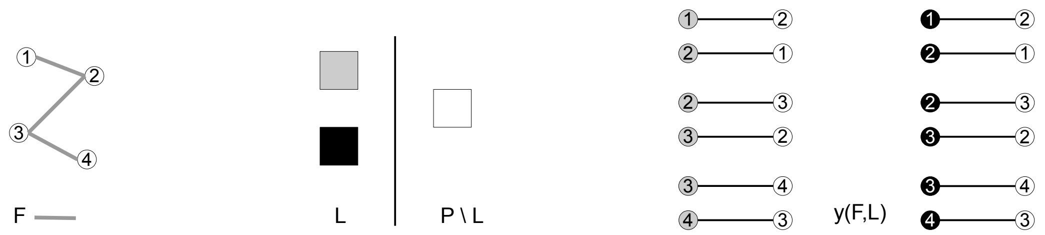


Figure 3.3: The sets $F$ (left) and $L$ (center). $y(F, L)$ is the sum over all $y$ variables depicted on the right side.

Similarly, further subgraph based constraints can be transferred:

**Proposition 17.** Let $L \subset \{V_1,\dots,V_c\}$, $\emptyset \neq L \neq \{V_1,\dots,V_c\}$, be a subset of the group set. Let $W \subseteq V$ with $|W| \geq 3$ be a vertex set which induces a clique $(W,F)$ in $(V,E_Y)$. Then,

$$y(F,L) \leq \left\lceil \frac{|W|}{2} \right\rceil \cdot \left\lfloor \frac{|W|}{2} \right\rfloor \tag{3.21}$$

is valid for every partition vector $y$.

A graph is called a *bicycle p-wheel* if it consists of a cycle of length $p$ as well as two additional vertices $s$ and $t$, which are adjacent to each other and to every vertex in the cycle. The bicyle 5-wheel is depicted in Figure 3.4.

**Proposition 18.** Let $L \subset \{V_1,\dots,V_c\}$, $\emptyset \neq L \neq \{V_1,\dots,V_c\}$, be a subset of the group set. Let $(W,F)$ denote a bicycle-$p$-wheel in $(V,E_Y)$ with $p \geq 1$. Then,

$$y(F,L) \leq 2p \tag{3.22}$$

is valid for every partition vector $y$.

## 3.2.3 Combining $x$ and $y$ Variables

Both $x$- and $y$-variables are co-used within a single model by [JMN93], [XTP07], and [BS09]. We have seen that every linear term in $x$-variables can be replaced by a linear term in $y$-variables. Even though this allows for pure $y$-variable models, the authors prefer to use the $x$-variables to define the partition, then link the $y$-variables to them. We will compare these two approaches in Section 4.6. To link the $y$ to the $x$-variables, the following products need to be linearized:

$$y_{uA,vB} = x_{uA} \cdot x_{vB} \qquad \text{for all } u,v \in V, u \neq v, A,B \in [c]. \tag{3.23}$$

The linearization is achieved in the following way by all authors of our survey. We assume again the variable $y_{uA,vB}$ to exist only for $\{u,v\} \in E_Y$, where $E_Y \subseteq V \times V$ is a subset of vertex pairs.

$$y_{uA,vB} \leq x_{uA} \qquad \text{for all } uv \in E_Y, A,B \in [c], \tag{3.24}$$

$$y_{uA,vB} \leq x_{vB} \qquad \text{for all } uv \in E_Y, A,B \in [c], \tag{3.25}$$

$$y_{uA,vB} \geq x_{uA} + x_{vB} - 1 \qquad \text{for all } uv \in E_Y, A,B \in [c], \tag{3.26}$$

$$y_{uA,vB} \in \{0,1\} \qquad \text{for all } uv \in E, A,B \in [c]. \tag{3.27}$$

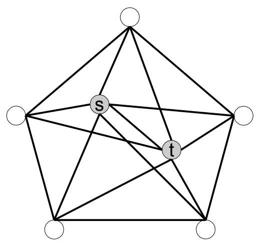


Figure 3.4: The bicycle 5-wheel.

This is the standard linearization technique for binary programs. It is introduced by Balas [Bal64] in 1964 and generally applicable to polynomials of any degree. An alternative is proposed by Frieze and Yadegar [FY83] in an integer programming formulation for the Quadratic Assignment Problem. As pointed out by Liberti [Lib07], it can be applied to a wide class of problems. It can be easily verified that the pattern search problems which we deal with are part of this class. The linearization constraints are the following ones.

$$x_{uA} = \sum_{B \in [c]} y_{uA,vB} \qquad \text{for all } (u,v,k) \text{ with } \{u,v\} \in E_Y, A \in [c]. \tag{3.28}$$

**Proposition 19.** For all binary vectors $\{x_{uA}\}_{u\in V, A\in[c]}$ and all vectors $\{y_{uA,vB}\}_{(u,v)\in E_Y, A,B\in[c]}$ with fractional non-negative entries, which satisfy Constraint (3.1) and (3.28), the relation $y_{uA,vB} = x_{uA} \cdot x_{vB}$ holds for all $(u,v) \in E_Y$ and all $A, B \in [c]$.

The proof follows from Liberti [Lib07]. Note that the relation $y_{uA,vB} = x_{uA} \cdot x_{vB}$ does not hold in LP relaxations of the model. If $x$ is not binary, but can contain any fractional values from the interval $[0,1]$, the following vector satisfies Constraint (3.1) and (3.28): Set $x \equiv 1/c$, where $c$ is the number of vertex groups, $y_{uA,vB} = 1/c$ if $A = B$ and $y_{uA,vB} = 0$ otherwise. However, the relation $y_{uA,vB} = x_{uA} \cdot x_{vB}$ does not hold if $A \neq B$.
Concerning the redundancy of the equations in Constraint (3.28), we can transfer a result by Kaibel [Kai97] from his thesis on the Quadratic Assignment Polytope:

**Proposition 20.** The linear equation system consisting of constraints (3.1) and (3.28) can be turned into a minimal one in the following way: For every $uv \in E_Y$, remove exactly one equation (3.28) of the form $(u,v,k)$.

## 3.3 EQUIVALENCE RELATION VARIABLES $s$, VERTEX SUBSET VARIABLES $w$

In clustering problems, it is relevant whether a certain edge is within a cluster or in-between two clusters. If the constraints on all clusters (vertex groups) are identical, it is not necessary to know *which* clusters the edge connects. In this case, a third kind of basic variable can be used. It exists for each vertex pair and indicates whether the two vertices are in the same group:

$$s_{uv} = \begin{cases} 1 & \text{if } P(u) = P(v), \\ 0 & \text{otherwise,} \end{cases}$$

for all unordered vertex pairs $\{u,v\} \subseteq V$. Again, only the variables $s_{uv}$ with $u < v$ are introduced for practical computations. In the following, we will identify $s_{uv}$ with $s_{vu}$ for ease of notation. The variable defining constraints for $s$ are well-known from models for the clique partitioning problem:

$$s_{uv} + s_{vw} - s_{uw} \leq 1 \text{ for all distinct } u,v,w \in V, \tag{3.29}$$

$$s_{uv} \in \{0,1\} \text{ for all distinct } u,v \in V. \tag{3.30}$$

These variables are used by Grötschel and Wakabayashi [GW89], Mehrotra and Trick [MT98], and Agarwal and Kempe [AK08].

**Constraints on Group Sizes and Numbers of Groups.** Constraints on the group sizes can be modeled with $s$-variables as well. At least, if the number $k$ of vertices requried in a group is at least 2 and the same for all groups. We can then require each vertex $v$ to share its color with at least $k-1$ other vertices. The demand for groups of almost equal size can be modeled this way.

$$\sum_{u \in V \setminus \{v\}} s_{uv} \geq k-1 \text{ for all } v \in V. \tag{3.31}$$

Let us now consider constraints on the number of groups. The following constraint (3.32) demands at least two groups. However, there is no possibility to demand at least $k$ groups for general $k$, as the number of groups is not directly related to the number of same grouped pairs. It is not true that a lower number of groups implies a larger number of same grouped pairs. For example, consider a graph with 8 nodes (Figure 3.5). A partition into 2 groups of sizes 4 and 4 yields $\sum_{u,v \in V} s_{uv} = 12$, but a partition into 3 groups of sizes 1, 1 and 6 yields a sum of $15 > 12$. Nevertheless, we present a possibility to restrain the number of groups from above. For the typical case $k \ll |V|$, we can achieve this by adding Constraint (3.33).

$$\sum_{u,v \in V} s_{uv} \geq 1, \tag{3.32}$$

$$\sum_{\substack{u,v \in S,\\ u \neq v}} s_{uv} \geq 1 \text{ for all subsets } S \subseteq V \text{ with } |S| = k+1. \tag{3.33}$$

This latter constraint only allows solutions with at most $k$ groups. In this case, there are two vertices $u, v$ with $s_{uv} = 1$ in every vertex subset $S$ of cardinality $k+1$. Now assume the model had a solution $s$ with more than $k$ groups. Let $S$ denote the set that includes an arbitrary representative node from each group. Choose a subset $S' \subseteq S$ with exactly $k+1$ elements. Then, $s_{uv} = 0$ holds for all distinct nodes $u, v \in S'$. Hence, Constraint (3.33) is violated by $S'$. Unfortunately, the number of constraints grows exponentially in the number of vertices. On the other hand, there is a linear time separation procedure.

**Vertex Subset Variables $w$.** A fourth type of variable is used by Mehrotra and Trick [MT98] in a column generation approach. They define a variable $w_S$ for each vertex subset $S \subseteq V$. It takes the value 1 if $S$ is a group in the final partition, and 0 otherwise. One has to make sure that the final groups contain all vertices and are non-overlapping. This can be achieved by the following

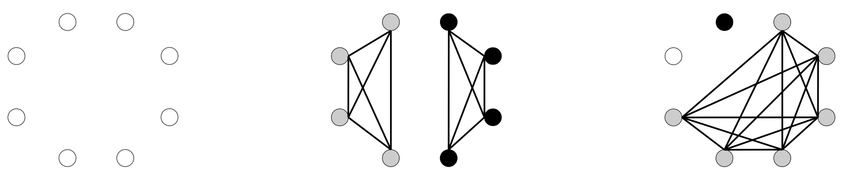


Figure 3.5: The set of 8 vertices (left) is partitioned into two groups (center) and three groups (right). Edges represent $s$-variables.

constraint. It states that every vertex $v$ must be included in exactly one group:

$$\sum_{S:v\in S} w_S = 1 \text{ for all } v \in V, \tag{3.34}$$

$$x_S \in \{0,1\} \text{ for all } S \subseteq V. \tag{3.35}$$

To have exactly $k$ groups, one needs to add the equation $\sum_{S\subseteq V} w_S = k$ to the model. To allow only certain group sizes, one simply sets $w_S = 0$ for all vertex sets $S$ of inappropriate size.

## 3.4 MINIMUM BLOCKMODEL ERRORS BY POLYNOMIAL LINEARIZATION

In this chapter, we treat a variable that is needed in models for patterns of link existence. We introduce it for directed graphs $D = (V, A(D))$, as the undirected case can be modeled analogously. Recall that the image matrix entry $I_{AB} = 1$ requires many vertices in group $V_A$ to have at least one successor vertex in group $V_B$. The penalty variable $\alpha_{v,AB}$ is used to indicate a violation against this requirement. That is, its value is 1 if vertex $v$ is in group $V_A$, but does *not* have any successor in group $V_B$. Similarily, $\beta$ variables are defined to indicate a violation against the *predecessor* requirement. Recall that $I_{AB} = 1$ also requires many vertices in $V_B$ to have at least one predecessor in group $V_A$. That is,

$$\alpha_{v,AB} = \begin{cases} 1 & \text{if } P(v) = A \text{ and there is no } w \in V \text{ with } (v,w) \in A(D) \text{ and } P(w) = B, \\ 0 & \text{otherwise.} \end{cases}$$

$$\beta_{v,AB} = \begin{cases} 1 & \text{if } P(v) = B \text{ and there is no } w \in V \text{ with } (w,v) \in A(D) \text{ and } P(w) = A, \\ 0 & \text{otherwise.} \end{cases}$$

To our knowledge, Brusco and Steinley [BS09] are the only authors to use this variable in an integer programming model. They use a static constraint to define it. In this chapter, we derive for each variable an exponential-size family of constraints, where each member of the family can be used to define the variable. The family evolves through different ways of linearization of the so-called $\alpha$-defining polynomial. In Section 4.3, a polynomial time separation procedure to be used in cutting plane algorithms is described. We furthermore show that the constraint of Brusco and Steinley is a member of this family.

### 3.4.1 **Linearization of Error Defining Polynomials**

We begin with some general assumptions. We assume that the input to our integer programming model contains a directed graph $D = (V, A(D))$ to be searched for patterns, as well as a number $c$ of groups we want to partition the vertex set $V$ into. For ease of notation, let us always consider the penalty variable $\alpha_{0,AB}$. I.e., the penalty inducing vertex is 0, and the image matrix entry to be 1 is $I_{AB}$. We thus omit the indices wherever possible. Furthermore, let us assume that the successor vertices $N^+(0)$ of vertex 0 are $\{1, 2, \ldots, N := \deg^+(0)\}$. All assumptions can be made without loss of generality. Moreover, all results can be easily transferred to $\beta$ variables as well.

We will now see how the variable $\alpha$ can be defined by using vertex assignment variables $x$ and edge assignment variables $y$ from the preceding chapter.
Generally, the variable $\alpha$ can be defined by a constraint

$$\alpha = p(x,y),$$

where $p(x,y)$ is a polynomial function in the vertex assignment variables $x$ and the edge assignment variables $y$. In order to state a possible realization of $p(x,y)$, some further definitions are required. Recall that $x_{uA} = 1$ if $u$ is assigned to group $A$. We define $x_{\overline{uA}} = 1$ if $u$ is *not* in group $V_A$. Furthermore, we use a similar notation for the edge assignment variables $y$. The variable $y_{\overline{uA},vB}$ is 1 if $u$ is *not* in group $V_A$, but $v$ is in group $V_B$. Similarily, $y_{\overline{uA},\overline{vB}}$ is 1 if neither $u$ is in group $V_A$ nor $v$ is in group $V_B$:

$$\begin{aligned}
x_{\overline{vA}} &:= 1 - x_{vA},\\
y_{\overline{uA},vB} &:= x_{uA} - y_{uA,vB},\\
y_{uA,\overline{vB}} &:= x_{vB} - y_{uA,vB},\\
y_{\overline{uA},\overline{vB}} &:= y_{uA,vB} - x_{uA} - x_{vB} + 1.
\end{aligned}$$

Using this notation, the polynomial $p(x,y)$ can be realized as follows.

$$p(x,y) := x_{0A} \prod_{v=1}^{N} x_{\overline{vB}}. \tag{3.36}$$

It takes a value of 1 if and only if vertex 0 is in group $V_A$, but no successor is in group $V_B$. Hence, the constraint $\alpha = p(x,y)$ defines $\alpha$. To avoid the distinction between 0 and its successors in more complex formulae, we introduce

$$\tilde{x}_v = \begin{cases} x_{0A} & \text{if } v = 0,\\ x_{\overline{vB}} & \text{otherwise,}\end{cases} \qquad \tilde{y}_{uv} = \begin{cases} y_{0A,\overline{vB}} & \text{if } u = 0,\, v \neq 0,\\ y_{\overline{uB},0A} & \text{if } v = 0,\, u \neq 0,\\ y_{\overline{uB},\overline{vB}} & \text{if } u,v \neq 0,\end{cases} \tag{3.37}$$

and sometimes write $\tilde{y}_{uu}$ instead of $\tilde{x}_u$. The polynomial defined by Equation (3.36) can now be re-written as

$$p(x,y) := \prod_{v=0}^{N} \tilde{x}_v. \tag{3.38}$$

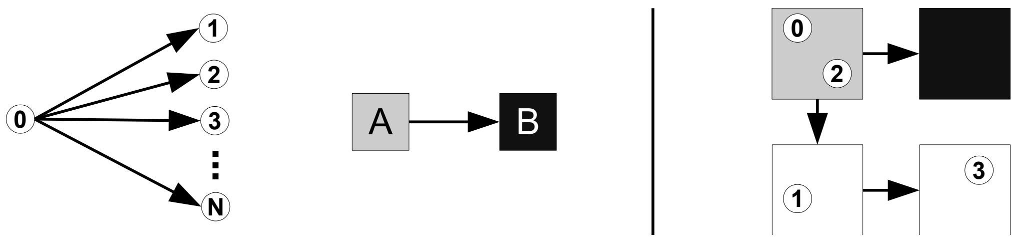


Figure 3.6: Left: The standard setting (vertex 0 and its successors 1 to $N$, groups $V_A$ and $V_B$). Right: Example for $\alpha_{0,AB} = 1$ (vertex 0 is in $V_A$, but none of its three successors is in $V_B$).

$p(x,y)$, as realized in Formula (3.38), is a polynomial function of degree $N+1$. If $N \leq 1$ holds, i.e., if vertex 0 has either no or one successor, $\alpha$ can be defined by a linear constraint respectively:

$$\begin{aligned} \alpha &= x_{0A}, \\ \alpha &= y_{0A,\overline{1B}}. \end{aligned} \tag{3.39}$$

For a greater value of $N$, there is also a way to define $\alpha$ by linear constraints.

### 3.4.2 Bounding the Error From Below

As a starting point, the following constraint bounds $\alpha$ from below for all $N \geq 2$:

$$\sum_{v=0}^{N} \tilde{x}_v \leq \alpha + N. \tag{3.40}$$

Only if all summands on the left-hand side are 1, the penalty $\alpha$ is forced to be 1 as well. On the left-hand side of this inequality, the product in Equation (3.38) was replaced by the sum of the very same variables. More generally, it can be replaced by any sum of products of these variables. To this end, let $S_0, S_1, \ldots, S_k$ denote a covering of $\{0, \ldots, N\}$ with non-empty, possibly overlapping subsets $S_i$. Then, the following inequality defines $\alpha$:

$$\sum_{i=0}^{k} \prod_{v \in S_i} \tilde{x}_v \leq \alpha + k. \tag{3.41}$$

Constraint (3.40) is a special case that is obtained by setting $S_i = \{i\}$. Again, $\alpha$ is forced to be 1 only if all variables on the left-hand side are 1 as well. In this case, vertex 0 is in group $V_A$, but none of its successors is in group $V_B$. Assuming that all possible $y$-variables exist in the model, the constraint is linear in $x$ and $y$ if and only if $|S_i| \leq 2$ for all $i = 0, \ldots, k$. In this case, the sets $S_i$ are of the form $S_i = \{u_i, v_i\}$ for not neccessarily distinct $u_i$ and $v_i$. Constraint (3.41) then has the following linear form:

$$\sum_{i=0}^{k} \tilde{y}_{u_i v_i} \leq \alpha + k. \tag{3.42}$$

We present two ways to strengthen the linearization in Constraint (3.42). The first linearization technique to be presented replaces the constant $k$ on the right-hand side by a sum of variables, which has a value of at most $k$. To see which variables are used in the replacement, we introduce an undirected graph $G_\alpha = (V_\alpha, E_\alpha)$. As on Page 35, let $E_Y$ denote the set of all vertex pairs for which all $y$-variables exist in the model. That is, $uv \in E_Y$ if and only if $y_{uC,vD}$ is in the model for all $C, D \in [c]$. The graph $G_\alpha$ has the vertex set $V_\alpha := \{0, \ldots, N\}$. Its edge set $E_\alpha$ is $E_Y$ restricted on the vertices in $V_\alpha$ together with all loop edges $vv$ on all vertices $v \in \{0, \ldots, N\}$ (see Figure 3.7). Colorings of the edges of $G_\alpha$ are now considered. We call a coloring a *blue coloring*, if every edge is either blue or uncolored and the blue edges form an edge cover of $G_\alpha$. Recall that an edge cover is a set of edges such that every vertex is incident to at least one edge of the set (see Figure 3.7). Note that this requirement is equivalent to the covering condition for the sets $S_i = \{u_i, v_i\}$, if the $u_i v_i$ are interpreted as blue edges. Using this notation, Constraint (3.42) can be re-written in the following way. There, $B \subseteq E(G_\alpha)$ denotes the set of blue edges in a blue coloring of $G_\alpha$.

$$\sum_{uv \in B} \tilde{y}_{uv} \leq \alpha + |B| - 1. \tag{3.43}$$

Given a blue edge coloring, we call a function $r$ a *blue endnode mapping* if it maps every blue edge onto exactly one of its end nodes. Let $|r^{-1}(v)|$ denote the number of blue edges which are mapped onto vertex $v$ (see Figure 3.7).

**Proposition 21.** Constraint (3.43) can be strengthened to the following one, where $B$ are the blue edges in a blue coloring of $G_\alpha$ and $r$ is a blue endnode mapping.

$$\sum_{uv \in B} \tilde{y}_{uv} + \prod_{\substack{v=0,\dots,N: \\ |r^{-1}(v)| \neq 0}} \tilde{x}_v \leq \alpha + \sum_{v=0,\dots,N} |r^{-1}(v)| \tilde{x}_v. \tag{3.44}$$

*Proof.* Consider a binary partition vector $(x^*, y^*)$. In case that the product equals 1, the sum on the right-hand side equals $|B|$ and the inequality is hence the same as in Constraint (3.43). Now consider the case in which the product equals 0. That is, there is a non-empty set $X_0$ of all vertices $v$ with $\tilde{x}^*_v = 0$ and $|r^{-1}(v)| \neq 0$. For every $w \in X_0$, there are $|r^{-1}(w)|$ variables $\tilde{y}^*$ on the left-hand side that equal 0. These are those $\tilde{y}$ variables that correspond to the blue edges which are mapped onto $w$ by the function $r$. All in all, at least $\sum_{w \in X_0} |r^{-1}(w)|$ of the $\tilde{y}^*$ variables equal 0. The left-hand side value $lhs(x^*, y^*)$ is hence at most $|B| - \sum_{w \in X_0} |r^{-1}(w)|$. The right-hand side is $\alpha + rhs(x^*, y^*)$, where $rhs(x^*, y^*)$ is at least $|B| - \sum_{w \in X_0} |r^{-1}(w)|$. The constraint is thus satisfied for any binary value of $\alpha$. $\square$

To our knowledge, $\alpha$-variables are only used by Brusco and Steinley [BS09] in literature. Let us

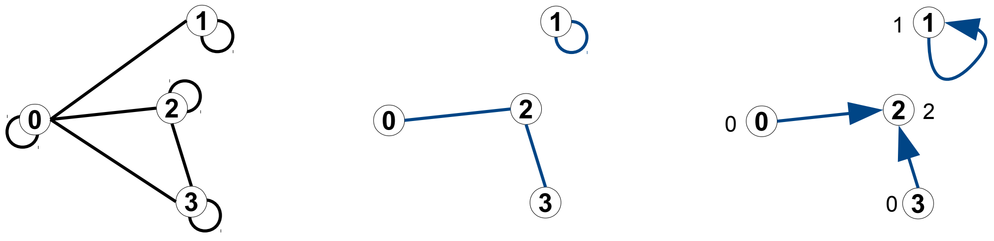


Figure 3.7: Graph $G_\alpha$ (left), a blue coloring of $G_\alpha$ (center), and a blue endnode mapping of the blue coloring (right) with values $|r^{-1}(v)|$ at the vertices.

show that their $\alpha$ defining constraints are of the form described in Constraint (3.44).

$$\begin{aligned}
&\sum_{v=1}^{N} y_{0A,vB} + \alpha \geq x_{0A} \text{ (Brusco \& Steinley)} \qquad (3.45)\\
\Leftrightarrow &\sum_{v=1}^{N} y_{0A,vB} \geq -\alpha + 1 - x_{\overline{0A}}\\
\Leftrightarrow &\sum_{v=1}^{N} (y_{0A,vB} + x_{\overline{0A}}) \geq -\alpha + 1 + (N-1)x_{\overline{0A}}\\
\Leftrightarrow &\sum_{v=1}^{N} (1 - y_{0A,\overline{vB}}) \geq 1 - \alpha + (N-1)(1 - x_{0A})\\
\Leftrightarrow &\sum_{v=1}^{N} y_{0A,\overline{vB}} \leq \alpha + (N-1)x_{0A}\\
\Leftrightarrow &\sum_{v=1}^{N} \tilde{y}_{0v} \leq \alpha + (N-1)\tilde{x}_{0}
\end{aligned}$$

The final inequality is of our type, where the blue edges are the star graph centered at 0 and the function $r$ maps all blue edges onto vertex 0 (see Figure 3.8).
Constraint (3.44) is not linear in $(x,y)$ if the blue edges are mapped onto 3 or more distinct vertices. One possibility is to introduce a new variable to represent the product. In this case, at most $2|V|$ additional variables are needed to define all $\alpha$ and $\beta$ variables. This number is especially independent of the number of partitions and the vertex degrees. A second possibility is to use a weaker, but linear version of the constraint. As an example, define $\tilde{r}(v) = |r^{-1}(v)| - 1$, if $|r^{-1}(v)| \neq 0$, and $\tilde{r}(v) = 0$ otherwise. The correctness of the following constraint can be easily shown.

$$\sum_{uv \in B} \tilde{y}_{uv} \leq \alpha + |B| - 1 + \sum_{v=0}^{N} \tilde{r}(v)(\tilde{x}_v - 1) \qquad (3.46)$$

We now present a second possibility to strengthen Constraint (3.43). Here, further $y$-variables are added to the left-hand side of the inequality. They are denoted by $\hat{y}$.

$$\hat{y}_{uv} = \begin{cases} y_{\overline{0A},vB} & \text{if } u = 0,\, v \neq 0,\\ y_{uB,\overline{0A}} & \text{if } v = 0,\, u \neq 0,\\ y_{uB,vB} & \text{if } u,v \neq 0. \end{cases}$$

Again, the variables to be added can be interpreted as edges in $G_\alpha$. We call them the *red* edges. An edge coloring of $G_\alpha$ is called a *blue-red coloring*, if every edge is either blue, red, or uncolored and

1. the blue edges form an edge cover of $G_\alpha$,
2. every set of $h$ red edges is incident to at least $h+1$ blue edges, for all $h \in \mathbb{N}_+$.

An example for a blue-red coloring is depicted in Figure 3.8.

**Proposition 22.** Constraint (3.43) can be strengthened to the following one, where $B$ and $R$ are the blue and red edges in a blue-red coloring of $G_\alpha$:

$$\sum_{uv\in B} \tilde{y}_{uv} + \sum_{uv\in R} \hat{y}_{uv} \leq \alpha + |B| - 1. \tag{3.47}$$

*Proof.* If $R$ is empty, the inequality reduces to Constraint (3.43), the validity of which has already been proven. Assume $R$ to be non-empty. Consider a binary partition vector $y^*$ and note that all entries of $\tilde{y}^*$ and $\hat{y}^*$ are binary. Let $R_1$ denote the subset of $R$ containing those edges $uv \in R$ with $y^*_{uv} = 1$. Define $B_0$ analogously. By Condition 2, there are at least $|R_1| + 1$ blue edges incident to $R_1$. These incident edges are all in $B_0$ by the definition of $\tilde{y}$ and $\hat{y}$. It follows that $|B_0| > |R_1|$. Thus, the inequality remains valid if $R$ is non-empty. □

### 3.4.3 Bounding the Error From Above

Even though the values of the $\alpha$-variables are usually minimized, as $\alpha = 1$ indicates a deviation from a pattern of link existence, there are models which require $\alpha$ to take an exact value. In these cases, $\alpha$ needs additionally to be bounded from above. This can be achieved in a straightforward way by linearizing the constraint

$$\alpha \leq \prod_{v=0}^{N} \tilde{x}_v \tag{3.48}$$

with the following set of linear constraints:

$$\tilde{x}_v \geq \alpha \text{ for all } v = 0, \ldots, N. \tag{3.49}$$

This set of constraints already suffices to set $\alpha$ to 0 in all partitions that are not penalty inducing. Again, the model formulation can be improved by interpreting these constraints as a subset of a larger family of constraints. The general form of the member constraints is the following one. It depends on a set $S$ of indices:

$$\sum_{(u,C,v,D)\in S} y_{uC,vD} \geq \alpha. \tag{3.50}$$

We now examine for which choices of $S$ the above inequality is valid. To this end, we introduce for each vertex $v$ the set $Pen_v(\alpha_{0,AB})$ of penalty inducing groups:

$$Pen_v(\alpha_{0,AB}) = \begin{cases} \{A\} & \text{if } v = 0, \\ [c] \setminus \{B\} & \text{otherwise.} \end{cases}$$

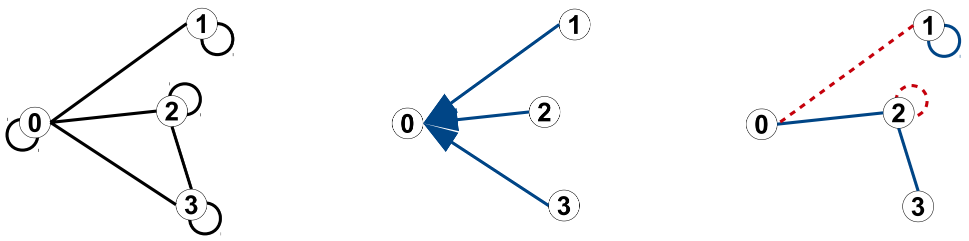


Figure 3.8: A graph $G_\alpha$ (left), blue endnode mapping used by Brusco and Steinley (center), and an example for a blue-red coloring (right) with red (dashed) and blue (solid) edges.

That is, $\alpha_{0,AB} = 1$ if and only if every vertex $v$ for $v = 0, \ldots, N$ is assigned to a group in $Pen_v(\alpha_{0,AB})$. We will now show that Constraint (3.50) is valid if the set $S$ is *above bounding*:

**Definition 10.** A set $S \subseteq V_\alpha \times [c] \times V_\alpha \times [c]$ is called *above bounding for* $\alpha$, if for each $c$-partition $P$ with $P(v) \in Pen_v(\alpha)$ for all $v = 0, \ldots, N$, there are two distinct indices $i, j \in \{0, \ldots, N\}$ such that $(i, P(i), j, P(j)) \in S$.

**Proposition 23.** Constraint (3.50) is a valid inequality if and only if the set $S$ is above bounding for $\alpha$.

*Proof.* Assume the constraint is valid. Then, for every partition $P$ which is error inducing for $\alpha$, i. e., $\alpha = 1$, at least one $y$-variable on the left-hand side takes a value of 1. This variable has the form $y_{uP(u)vP(v)}$ with $P(u) \in Pen_u(\alpha)$ and $P(v) \in Pen_v(\alpha)$. Hence, $(u, P(u), v, P(v)) \in S$, which implies that $S$ is above bounding. Now assume that $S$ is above bounding. Then, for every partition $P$ which is error inducing, there are two indices $i, j$ with $(i, P(i), j, P(j)) \in S$. Hence, the variable $y_{iP(i)jP(j)}$ exists on the left-hand side of the constraint and takes the value 1. The constraint is hence satisfied. □

We will show that a set $S$ is above bounding if and only if the following graph $G^S_\alpha = (V^S, E^S)$ does not contain a generalized hamiltonian cycle. The vertex set $V^S$ is the union of $N+1$ disjoint vertex sets $V^S_i$:

$$V^S_i = \{(i, X) \mid X \in Pen_i(\alpha)\}, \tag{3.51}$$

$$V^S = \bigcup_{i=0}^{N} V^S_i. \tag{3.52}$$

The vertices in $V^S$ are depicted in Figure 3.9. The edge set $E^S$ is the complete $(N+1)$-partite graph with shores $V^S_i$. together with loop edges on all vertices, where all edges in $S$ are removed.

$$\begin{aligned} E^S = &\{uv \mid u \in V^S_i, v \in V^S_j, i \neq j\} \cup \{uu \mid u \in V^S\} \\ &\setminus \{uv \mid (u, X, v, Y) \in S\}. \end{aligned} \tag{3.53}$$

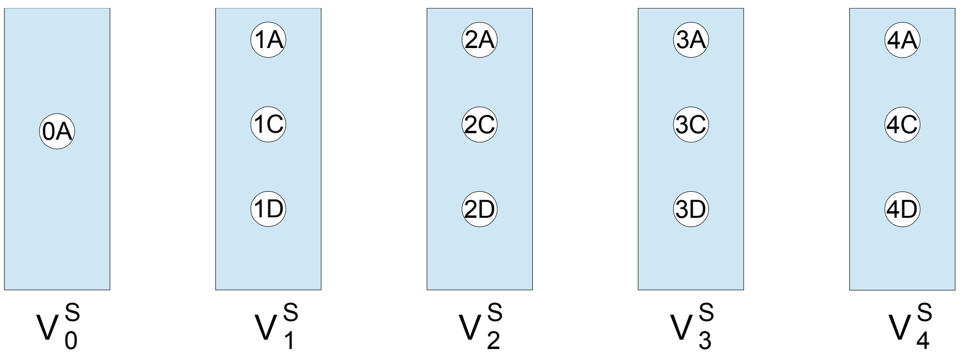


Figure 3.9: The sets $V^S_i$ for $N = 4$ and $c = 4$.

**Definition 11.** A clique in $G^S_\alpha$ is called a *partition spanning clique*, if it contains exactly one vertex of every set $V^S_i$ for $i = 0,\dots,N$.

**Proposition 24.** A set $S \subseteq V_\alpha \times [c] \times V_\alpha \times [c]$ is *above bounding for* $\alpha$, if and only if the graph $G^S_\alpha$ does not contain a partition spanning clique.

*Proof.* Consider an above bounding set $S$. Assume that there was a partition spanning clique $C$ in $G^S_\alpha$. Let $(0,X_0),\dots,(N,X_N)$ denote the vertices of the clique. Then, the partition $P$ defined by $P(i) = X_i$ for $i = 0,\dots,N$ satisfies $P(v) \in Pen_v(\alpha)$. Hence, as $S$ is above bounding, there are two indices $i, j$ with $(i,X_i, j,X_j) \in S$. This implies that the edge $(i,X_i)(j,X_j)$ is not in $G^S_\alpha$, which contradicts the assumption that $C$ was a clique.
Consider the case that $G^S_\alpha$ does not contain a partition spanning clique. Assume that $S$ is not above bounding. Then, there is a partition $P$ with $P(v) \in Pen_v(\alpha)$ for all $v$, such that for all index pairs $i, j$ holds that $(i,P(i), j,P(j)) \notin S$. It follows that the vertices $(0,P(0)),\dots,(N,P(N))$ induce a partition spanning clique, which contradicts the assumption. □

We now give two examples of above bounding sets which can be used for any number $N$ of successor vertices. The two sets are visualized in Figures 3.10 and 3.11. In these figures, the edge $(u,X)(v,Y)$ is drawn if $(u,X,v,Y) \in S$. To verify that $S$ is indeed above bounding, Proposition 24 can be used: In the figure, check that there is no stable set containing exactly one vertex from each partition $V^S_i$.

**Proposition 25.**

a) If $u, v \in \{0,\dots,N\}$ are two distinct vertices, then the following set $S$ is above bounding for $\alpha$:

$$S = \{uHvK \mid H \in Pen_u(\alpha), K \in Pen_w(\alpha)\}. \tag{3.54}$$

b) Let $\pi$ denote a permutation of the vertices in $\{0,\dots,N\}$ with $\pi(0) = 0$. Let $C^* : \{0,\dots,N\} \to Pen$ denote a function which assigns every vertex $v$ an element in $Pen_v(\alpha)$. If the number $p$ of partitions is at least 3, the following set $S$ is above bounding for $\alpha$:

$$\begin{aligned} S = \{(\pi(v),C^*(\pi(v)),\pi(v+1),D) \mid D \in Pen_{\pi(v+1)}(\alpha) \setminus \{C^*(\pi(v+1))\}, v = 0,\dots,N-1\} \\ \cup \{(\pi(N),C^*(\pi(N)),0,A)\}. \end{aligned} \tag{3.55}$$

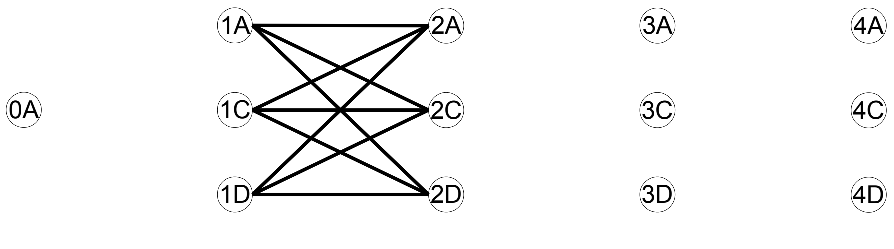


Figure 3.10: The set $S$ from Proposition 25 a) with $c = 4$, $N = 4$, $u = 1$, and $v = 2$.

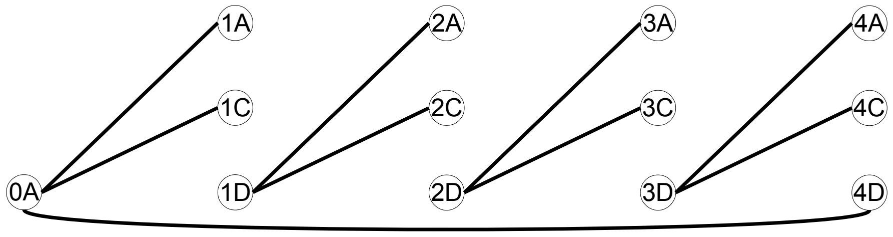


Figure 3.11: The set $S$ from Proposition 25 b) with $c = 4$, $N = 4$, $\pi = id$ (identity mapping), and $C^*(i) = D$ for $i = 1, \ldots, N$, $C^*(0) = A$.

*Proof.*

a) The resulting inequalities can be written in the following way. As they are obviously valid, $S$ is above bounding according to Proposition 23.

$$y_{\overline{uD},\overline{vD}} \geq \alpha \text{ if } u \neq 0 \neq v,$$
$$y_{\overline{uC},\overline{vD}} \geq \alpha \text{ if } u = 0.$$

b) Consider a partition $P$ with $P(v) \in Pen_v(\alpha)$ respectively. Assume $(\pi(v), P(\pi(v)), \pi(v+1), P(\pi(v+1))) \notin S$ for all $v = 0, \ldots, N-1$. Then, $P(\pi(N)) = C^*(\pi(N))$ and $P(0) = C^*(0) = A$, thus $(\pi(N), P(\pi(N)), 0, P(0)) = (\pi(N), C*(\pi(N)), 0, A) \in S$. Hence, $S$ is above bounding.

□

# Chapter 4

# BRANCH-AND-CUT ALGORITHMS FOR A PATTERN SEARCH PROBLEM

In this chapter, we investigate a specific pattern search problem. We are the first authors to consider it. In our opinion, the problem is interesting for three reasons. First, it can be seen as the most straightforward model to measure the deviation of a given network from a given pattern, as it counts the number of non-fitting arcs. Still, we show that this problem is already NP-hard. Second, we show that the problem is the generalization of several well-studied combinatorial optimization problems, such as the traveling salesman, the linear ordering, and the quadratic assignment problem. We show how to exploit these relations algorithmically. Third, the problem can be solved exactly for real-world problems with more than 100 vertices.
We develop a branch-and-cut algorithm to solve the problem to optimality, whose strength lies both in the exploitation of the results of Chapter 3, as well as the development of problem-specific separation routines, as well as primal and dual heuristics. We show that our algorithm is up to $10,000$ times faster than comparable models from literature.
Section 4.1 introduces the problem and shows its NP-hardness as well as its relations to its well-known special cases. Section 4.2 explains the network instances which are used to test all components of the branch-and-cut algorithm. In Section 4.3, new cutting plane separation algorithms for the constraints introduced in the preceding chapter are discussed. Sections 4.4 and 4.5 introduce new primal and dual heuristics and show their suitability in computational tests respectively. We finally present computational results for the complete algorithm in Section 4.6.

## 4.1 THE OPTIMIZATION PROBLEM

In this section, we introduce a combinatorial optimization problem. Even though it is one of the most simple approaches to search for patterns of link existence, it has not been studied so far. According to our classification scheme in Chapter 2, it can be characterized as the subgraph relaxation method with both the most basic distance function and penalty combination function:

- Ideal graphs are used,
- $d(G_{P,A,B}, H_{P,A,B}) = \sum_{u,v\in V, u\neq v} |Adj(G_{P,A,B})_{u,v} - Adj(H_{P,A,B})_{u,v}|$,

- $p(P) = \sum_{A,B \in [c], A \leq B} p_{AB}(P)$.

To make this chapter self-contained, we shortly motivate and explain this model again. We have seen in Section 2.2.1 that many real-world networks do not show a clear link pattern in the case that a small number of vertex groups is demanded. It is thus necessary to relax the strict definition of ideal regular vertex partitions (Definition 1 on Page 14). If subgraph relaxations are chosen, we measure for each pattern (image graph) the amount of link changes in the network which are necessary to obtain a network which perfectly shows the given pattern. The smaller the amount of necessary changes, the more suitable is the pattern to describe the network's structure. Possible "changes" are the addition and deletion of arcs from the network graph. I. e., entries in the adjacency matrix of the network graph can be changed. The lower the number of changes, the better the image graph. To sum up, the optimization problem to find the best image graph can be formulated as follows:

**Definition 12.** Let $D$ denote a digraph and $I$ a $c \times c$ binary matrix for a $c \geq 1$. $REG(D,I)$ is the set of all pairs $(H,P)$, such that $H$ is a graph with the same vertex set as $D$ such that the vertex partition $P$ is an ideal regular $c$-partition on $H$ with image matrix $I$.

**Problem 1.** Given a digraph $D = (V,A)$ and a number $c \geq 1$ of vertex groups, solve

$$\min_{I \in \{0,1\}^{c \times c}} \min_{(H,P) \in REG(D,I)} \sum_{u,v \in V, u \neq v} |Adj(D)_{u,v} - Adj(H)_{u,v}|. \tag{4.1}$$

The solution of the inner minimization hence gives the number of adjacency matrix changes which are necessary to obtain a graph which perfectly has the pattern described by image matrix $I$. The outer minimization finds the best out of all image matrices. A weighted version of this problem can be formulated by introducing two weight matrices: A matrix $W \in \mathbb{R}^{|V| \times |V|}$ to specify individual costs for adding or deleting an arc from $D$ and a matrix $B \in \mathbb{R}^{c \times c}$ to model that the cost for adding an arc can depend on the groups the arc is added between. In the weighted problem, Formula (4.1) thus needs to be replaced by

$$\min_{I \in \{0,1\}^{c \times c}} \min_{(H,P) \in REG(D,I)} \sum_{u,v \in V, u \neq v} W_{uv} B_{P(u)P(v)} |Adj(D)_{u,v} - Adj(H)_{u,v}|. \tag{4.2}$$

### 4.1.1 **Hypothesis Test**

Let us now consider the inner minimization in Problem 1. That is, Problem 1 with the additional constraint that an image matrix $I$ is fixed:

**Problem 2.** Given a digraph $D = (V,A)$, a number $c \geq 1$ of vertex groups, and a $c \times c$ binary matrix $I$, solve

$$\min_{(H,P) \in REG(D,I)} \sum_{u,v \in V, u \neq v} |Adj(D)_{u,v} - Adj(H)_{u,v}|. \tag{4.3}$$

That is, given a pattern represented by an image matrix $I$, decide how well the network $D$ matches the pattern. An example is given in Figure 4.1. Given the image graph in a), whose adjacency matrix is an image matrix denoted by $I$, and the digraph $D$ in b), the optimum solution value to

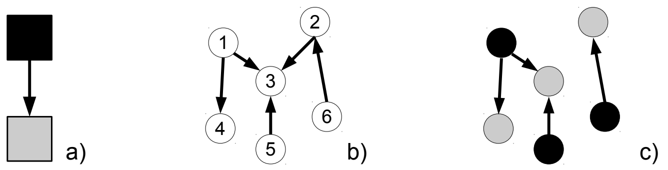


Figure 4.1: Example for Problem 2.

Problem 2 for $D$ and $I$ is 1, as c) shows that only one arc (from 2 to 3) needs to be deleted such that $I$ is an image matrix for $D$.
This problem is a so-called *hypothesis test*. The hypothesis that the given network $D$ has a certain link pattern can be compared to alternate hypotheses assuming distinct patterns. We have seen in Section 1.4 that these hypothesis are sometimes formulated by experts in the field of application. In the case of the world trade network, Wallerstein states the hypothesis that the world trade system has a core-semiperiphery-periphery rather than a core-periphery structure, which is later confirmed by Smith and White [SW92]. The two hypotheses can thus be compared by solving Problem 2 for both corresponding image matrices. The hypotheses can also be formulated by heuristics in the case of the absence of an expert. Brusco and Steinley [BS09] propose a 2-step procedure in this case: First, a short list $I_1, \ldots, I_k$ of potentially good image matrices is heuristically obtained. Second, Problem 2 is solved for all image matrices $I_i$. The matrix $I_i^*$ with the lowest optimum value represents the actual link pattern in the best way.

### 4.1.2 NP-Hardness and Relation to Combinatorial Problems

Both Problem 1 and Problem 2 are NP-hard. The proof follows from the fact that the penalty function (4.1) is regularly sensitive (Definition 7 on Page 27), as the four conditions stated on Page 27 are all met. The NP-hardness of Problem 1 then follows from Proposition 7 on Page 28, the NP-hardness of Problem 2 from Proposition 8 on Page 28.
We are the first authors to consider the two optimization problems above. We now show that they are not only interesting from the viewpoint of network analysis, but also from a combinatorial optimization perspective. Problem 2 is in fact a generalization of several well-known combinatorial optimization problems. Some of them, especially the Quadratic Assignment Problem, are known to be difficult to solve even for small instance sizes.

**Quadratic Assignment.** Input of the problem is a number $k$ of facilities (or locations), a $k \times k$ weight matrix $W^{QAP}$, and a $k \times k$ distance matrix $B^{QAP}$. The problem is to find an assignment $f$ of facilities to locations such that $\sum_{i,j \in [k], i \neq k} W_{i,j}^{QAP} B_{f(i)f(j)}^{QAP}$ is minimized.
The problem can be solved by solving Problem 2 with the following input data. Set $D$ to the complete graph $K_k$, $I$ to the $k \times k$ zero matrix (see Fig. 4.2 a)), $W = W^{QAP}$, and $B = B^{QAP}$.
Consider a solution $(H^*, P^*)$ which takes the optimum value. As all image matrix entries are 0, all arcs in $K_k$ have been deleted, that is, $H^*$ is an empty graph. The cost for deleting arc $uv$ is $W_{uv} B_{P(u)P(v)} = W_{uv}^{QAP} B_{P(u)P(v)}^{QAP}$. It is hence the cost for placing facilities $u$ and $v$ at locations $P(u)$ and $P(v)$ respectively. $P^*$ is thus the optimum assignment of the facilities to the locations.
Note that because $K_k$ is directed, both of the anti-parallel arcs $uv$ and $vu$ need to be deleted. The

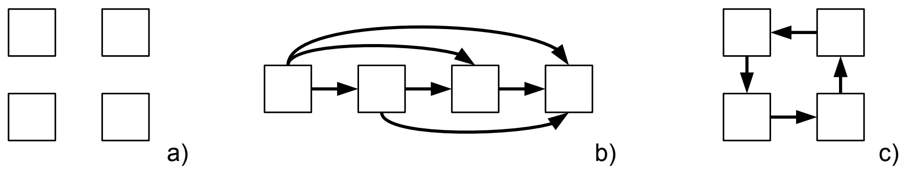


Figure 4.2: Image graphs used in the transformations from the Quadratic Assignment (a), Linear Ordering (b), and Asymmetric Traveling Salesman Problem (c).

optimum value to Problem 2 is hence twice the cost of the optimum quadratic assignment. Alternatively, one could set the matrices $W = \frac{1}{2}W^{QAP}$ and $B = \frac{1}{2}B^{QAP}$.

**Linear Ordering.** Input of the problem is the complete digraph $K_k$, with vertex set $V_k$, and edge weights $p_{uv}$ (preferences). The problem is to find a linear ordering $\pi : V \to [k]$ of the vertices such that $\sum_{u,v \in V:\pi(u)<\pi(v)} p_{uv}$ is maximized.

Set $D = K_k$, $I$ to the $|V_k| \times |V_k|$ upper triangular matrix, $W_{uv} = p_{uv}$, and $B \equiv 1$.

Consider a solution $(H^*, P^*)$ which takes the optimum value. $P^*$ assigns every vertex to a distinct group, hence to a distinct integer in $[k]$ and hence defines an ordering of the vertices. As the image matrix is upper triangular, the image graph contains exactly all *forward arcs*, that is, all arcs from $i$ to $j > i$. $H^*$ contains thus only forward arcs. The backward arcs must be deleted from $D$ with the cost $W_{uv}B_{P(u)P(v)} = p_{uv}$ for the arc $uv$. The total cost is hence the sum of all *backward* preferences. Its minimization is equivalent to the maximization of the *forward* preferences, as it is done in the Linear Ordering Problem. The ordering induced by $P^*$ is thus the optimum linear ordering.

The transformations for the following four problems are analogous to the two examples above.

**Asymmetric Traveling Salesman.** Input of the problem is the complete digraph $K_k$ and arc weights $d_{uv}$ (distances). The problem is to find a directed Hamiltonian cycle with the lowest sum of distances on its arcs. Set $D = K_k$, $I$ to a directed cycle graph of length $k$, $W_{uv} = -d_{uv}$, and $B \equiv 1$.

**Minimum k-Cut.** Input of the problem is a number $k$ of shores and an undirected graph $G = (V, E)$ with edge weights $c_{uv}$. The problem is to find a partition $P$ of $V$ into $k$ groups such that the sum of edges in-between the groups is minimized: $\min \sum_{u,v \in V:P(u) \neq P(v)} c_{uv}$.

Set $D$ to $G$, $I$ to the $k \times k$ zero matrix, $W_{uv} = c_{uv}$ for all $uv \in E$, $W_{uv} = 0$ otherwise, and $B_{ij} = 1$ for all $i \neq j$ $B_{ij} = 0$ otherwise.

**Minimum Edge Cover.** Input of the problem is a graph $G = (V, E)$ with edge weights $c_{uv}$. The problem is to find an edge set $E' \subseteq E$ with minimum total weight such that every vertex in $V$ is incident to at least one edge in $E'$.

Set $D = G$, $I$ to a $2 \times 2$ matrix with $I_{ij} = 0$ if and only if $i = j$, $W_{uv} = -1$ for $uv \in E$, $W_{uv} = M$ otherwise with a large positive number $M$, and $B \equiv 1$.

**Newman-Girvan Modularity.** Input for the problem of maximizing the Newman-Girvan modularity [NG04] is a graph $D_M = (V_M, A_M)$ and a number $k$ of clusters. The problem is defined on Page 23.
Set $D = D_M$, $I$ to the $k \times k$ zero matrix, $W_{uv} = |N^+(u)||N^-(v)|/|A_M|^2$ for all $u = v$, $W_{uv} = 1 - |N^+(u)||N^-(v)/|A_M|^2$ otherwise, and $B \equiv 1$.

### 4.1.3 **Integer Quadratic Programming Formulations**

We model Problem 1 and 2 as integer quadratic programs. The models are based on the variable types $x$, $y$, $s$, and $\alpha$ as defined in Chapter 3. To state the model for Problem 1, an additional variable type is needed. It indicates whether an edge $uv$ exists in the ideal graph $H$ that the original network $D = (V, A)$ is being compared to.

$$h_{uv} = \begin{cases} 1 & \text{if arc } (u,v) \text{ exists in the modified network } H, \\ 0 & \text{otherwise,} \end{cases}$$

for all distinct $u, v \in V$. The model can then be formulated as follows.

**Model 1.**

$$\min_{h,x} |A| + \sum_{(u,v)\notin A} h_{uv} - \sum_{(u,v)\in A} h_{uv}$$

$$\sum_{b\in V\setminus\{a,v\}} s_{ab} \cdot h_{vb} \geq s_{uv} \cdot h_{ua} \quad \text{for all } u,v,a \in V, a \neq u \neq v, \tag{4.4}$$

$$\sum_{b\in V\setminus\{a,v\}} s_{ab} \cdot h_{bv} \geq s_{uv} \cdot h_{au} \quad \text{for all } u,v,a \in V, a \neq u \neq v, \tag{4.5}$$

$$\sum_{u,v\in C} s_{uv} \geq 1 \quad \text{for all } C \subseteq V \text{ with } |C| = c+1, \tag{4.6}$$

$$s_{uv} + s_{vw} - s_{uw} \leq 1 \quad \text{for all pairwisely distinct } u,v,w \in V, \tag{4.7}$$

$$s_{uv}, h_{uv} \in \{0,1\} \quad \text{for all } u,v \in V. \tag{4.8}$$

The objective function minimizes the number of added arcs ($\sum_{(u,v)\notin A} h_{uv}$) and the number of removed arcs ($|A| - \sum_{(u,v)\in A} h_{uv}$). Constraints (4.4) and (4.5) assure that the partition is ideal regular on the digraph $H$: Consider Constraint (4.4). In the case that the right-hand side equals 0, the constraint is satisfied. If it equals 1, we obtain a situation as depicted in Figure 4.3a): The arc $ua$ exists in $H$, $u$ and $v$ are in the same group of the partition. According to Definition 3(ii) on Page 15 for ideal regular partitions, at least one successor vertex $b_i$ of $v$ must be in the same group as $a$, thus

$$\sum_{b\in V, b\neq a} s_{ab} \cdot h_{vb} \geq 1.$$

The constraints (4.6) limit the maximum number of groups to $c$. The transitivity constraints (4.7) demand the $s$-variables to take consistent values.
The model for Problem 2 also requires the introduction of an additional variable type $p$. For each pair $(A, B)$ of vertex groups, it specifies the number of arcs from $A$ to $B$ that need to be

added. Denote by $a$ the number of vertices in $A$ missing an arc into $B$ and by $b$ the number of vertices in $B$ missing an arc from $A$. Clearly, $\max\{a,b\}$ arcs need be inserted to obtain the ideal situation in which all vertices in $A$ have an arc into $B$ and vice versa. Hence, $p_{AB}$ can be defined as $p_{AB} := \max\{\sum_{u\in V}\alpha_{uAB}, \sum_{u\in V}\beta_{uAB}\}$. Recall that $N^+(u)$ ($N^-(u)$, resp.) denote the set of successor (predecessor, resp.) vertices of $u$ in $D$. We now state a model for Problem 2 without arc weights ($W \equiv 1$) and with positive group pair weights ($B > 0$):

**Model 2.**

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} B_{ab}x_{ua}x_{vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} B_{ab}p_{ab} \tag{4.9}$$

$$\sum_{a\in[c]} x_{ua} = 1 \qquad \text{for } u \in V, \tag{4.10}$$

$$\sum_{u\in V} x_{ua} \geq 1 \qquad \text{for } a \in [c], \tag{4.11}$$

$$x_{ua} + \sum_{v\in N^+(u)} (1 - x_{vb}) \leq \alpha_{uab} + |N^+(u)| \qquad \text{for } u \in V, a,b \in [c], I_{ab} = 1, \tag{4.12}$$

$$x_{ub} + \sum_{v\in N^-(u)} (1 - x_{va}) \leq \beta_{uab} + |N^-(u)| \qquad \text{for } u \in V, a,b \in [c], I_{ab} = 1, \tag{4.13}$$

$$p_{ab} \geq \sum_{u\in V} \alpha_{uab} \qquad \text{for } a,b \in [c], I_{ab} = 1, \tag{4.14}$$

$$p_{ab} \geq \sum_{u\in V} \beta_{uab} \qquad \text{for } a,b \in [c], I_{ab} = 1, \tag{4.15}$$

$$x_{ua}, \alpha_{uab}, \beta_{uab} \text{ binary, } p_{ab} \text{ integer} \qquad \text{for} \ldots \text{(see above).} \tag{4.16}$$

We now show the correctness of the formulation. If $I_{ab} = 0$, then there are no arcs from group $a$ to group $b$ in an ideal regular partition. The double sum of the objective function hence counts the number of such arcs, as they need to be deleted to obtain a ideal regular partition. If $I_{ab} = 1$, then $p_{ab}$ expresses the minimum number of arcs to be added in order to obtain an ideal regular partition between groups $a$ and $b$. Constraints (4.10) and (4.11) assure that $P$ is a partition without empty groups. Constraints (4.14) and (4.15) define $p_{ab} = \max\{\sum_{u\in V}\alpha_{uab}, \sum_{u\in V}\beta_{uab}\}$ in every optimum solution. Constraint (4.12) defines the $\alpha$-variables. If $u$ is in group $a$, but none of its successors in $b$, then the penalty variable $\alpha_{uab}$ is forced to be 1. In any other case, it may take both of the values 0 and 1. Note that $p_{ab}$ will nevertheless have a correct value in every optimum solution due to the minimization of the objective function. The $\beta$ variables are defined analogously in (4.13).
In order to linearize Model 2, we introduce edge assignment variables $y$ as in Section 3.2. Recall that $y_{ua,vb}$ models the product $x_{ua} \cdot x_{vb}$. Furthermore, $E_A$ denotes the edge set of the underlying

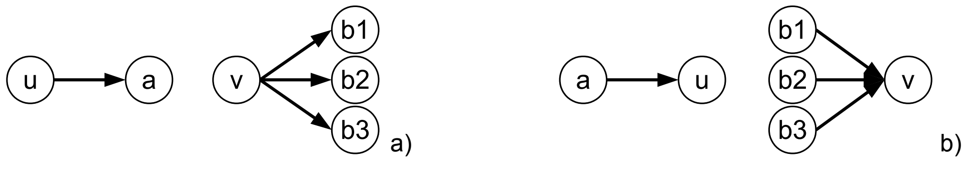


Figure 4.3: Explaining Constraints (4.4) and (4.5).

undirected graph of $D$ (Page 35). The correctness of the following linearized model follows from the proofs in Section 3.2.

**Model 2-lin.**

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} B_{ab} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} B_{ab} p_{ab} \tag{4.17}$$

$$\text{Constraints (4.10) - (4.16),}$$

$$x_{ua} = \textstyle\sum_{b\in[c]} y_{ua,vb} \text{ for all } uv \in E_A, a \in [c], \tag{4.18}$$

$$y_{ua,vb} \geq 0 \text{ for } uv \in E_A, a,b \in [c]. \tag{4.19}$$

## 4.2 COMPUTATIONAL SETTING AND TEST INSTANCES

In the following sections, we present primal and dual heuristics, separation routines, and a branch-and-cut algorithm for the problems above. All of these components are tested separately in order to measure their individual strength. However, we use a common test environment as well as common test graph instances, which are introduced in this section.

**Computational setting.** We implemented all tested models and separation routines in the branch-and-cut framework SCIP 3.1.0 [Ach09] in C++. In SCIP, we used the default values for all 1628 standard parameters. SCIP uses CPLEX 12.6 as its LP solver. The experiments were carried out on a desktop computer with 1.9 GHz clock speed and 4 GB random access memory under Ubuntu Linux 14.04 LTS. A single processor core was used during the computations.

**Watts-Strogatz Random Graphs (ws).** The Watts-Strogatz algorithm [WS98] is used to generate random digraphs. It produces digraphs that are similar to social networks with respect to the properties short average path lengths, high clustering coefficient, and a homogeneous degree distribution. The digraph $D$ is produced as follows. In a first step, $n$ vertices are created and numbered $0,\dots,n-1$. Then, each vertex $i$ is linked to its three neighbor vertices $i+1, i+2, i+3$ (*mod n* respectively). We give each link a random direction to obtain a digraph. In a second step, we consider each arc and rewire it with a probability of 10%. To rewire arc $(u,v)$ means that $v$ is replaced by a random vertex in $V \setminus N^+(u)$, where $N^+(u)$ are the *current* successors of $u$ in order to avoid duplicate links. See Figure 4.4 for an example showing Step 1 (left) and Step 2 (right) without arc directions. $4\times 4$ image matrices are computed randomly according to a uniform distribution.

As the tested image graph is computed randomly, we expect a large optimum value; the random network is unlikely to have the particular structure indicated by the image graph. Moreover, as the network itself is random, a good description by *any* image graph is unlikely. For this reason, we use a second random digraph generator. In a first step, it randomly generates an image graph. Secondly, it constructs a network that has a perfect regular coloring with the given image graph.

The optimum value would hence be 0, if it was not for a third step: The random network is distorted by adding and deleting edges. The amount of distortion can be specified by the user. This pair of network and image graph models the realistic scenario in which a heuristic or an expert already provided a good image graph description of the network.

**Distorted Random Graphs (rd).** Four parameters $n$, $c$, $d$ and $k$ are required. First, a number $o$ of one-entries in the image matrix is randomly chosen from $\{0,\dots,c^2\}$. Among all $c \times c$ image matrices with $o$ one entries, one matrix $I$ is randomly chosen. Now, $c-1$ distinct integers $n_1,\dots,n_{c-1}$ are randomly selected from $[1,n]$. The vertices $n_i+1$ to $n_{i+1}$ are colored with $i+1$ for $i=0,\dots,c-1$, with $n_0=0$ and $n_c=n$. Third, arcs are inserted such that the coloring is regular with respect to $I$ and the density $d$ is approximately reached. Finally, the resulting digraph is distorted by adding $k$ arcs and deleting from $k$ vertices either all incoming or all outgoing arcs. Let $D$ denote the distorted digraph. The test is then to solve Problem 2 for digraph $D$ and image matrix $I$. We set $c=4$ in all tests and generate one set of instances for $k=10$ and $k=100$ respectively.

**German Photo Trading Network (p).** We use a real-world network on the trading between German photo agencies in 2005. It has 70 nodes, but only 87 arcs. As image matrices, we take the three market structures depicted in Fig. 4.5. The instances are hence named *ph_A* to *ph_C*.

## 4.3 SEPARATION OF CUTTING PLANES

All constraints in Model 2-lin are polynomial in size and can be added to the initial integer programming formulation in a branch-and-cut algorithm. However, we have discussed further constraints to strengthen the model formulation, which are exponential in size. It seems reasonable not to add them to the initial formulation, but to add only a subset of them during the cutting phase of the algorithm. Given an optimum solution $u^* = (x^*, y^*, \alpha^*, \beta^*, p^*)$ to the current LP relaxation, the following heuristics try to find a constraint $a^T u \leq b$ which maximizes the amount $a^T u^* - b$ of violation by the LP optimum. We will see that the heuristics improve the LP bounds for 74% of our test instances.

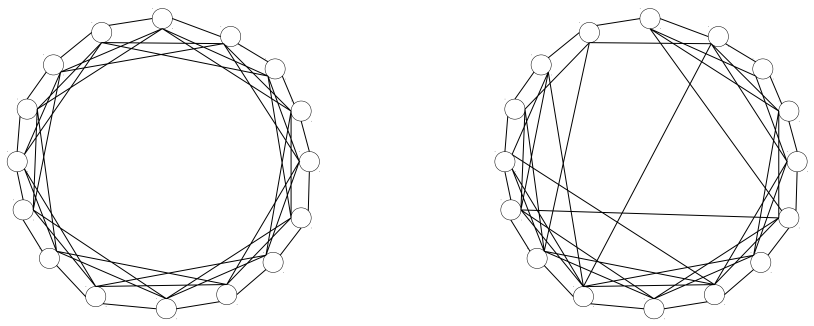


Figure 4.4: Steps 1 (left) and 2 (right) of the Watts-Strogatz graph generation.

### 4.3.1 **Error Defining Constraints**

In Section 3.4.2, we have seen that there are factorially many ways to define each single variable $\alpha_{u,AB}$. We hence describe a heuristic separation algorithm, which runs in polynomial time. It separates Constraint (3.47). Let us again assume that we consider those constraints which define $\alpha_{0,AB}$ and the successor vertices of vertex 0 are $\{1,\dots,N\}$. All other cases can be dealt with in a perfectly analogous way.

We are given a graph $G_\alpha$ with two types of weights on each edge $vw$: A blue weight $1-\tilde{y}^*_{vw}$ and a red weight $\hat{y}^*_{vw}$. We search for a coloring of the edges, where each edge must be either blue, red, or uncolored. Such a coloring is feasible, if the blue edges form an edge cover and every set of $k$ red edges is incident to at least $k+1$ blue edges (for all $k \geq 1$). Among all feasible colorings, we search for the one with the largest total weight of the colored edges. The computational complexity of this maximization problem is yet unknown. We describe a greedy separation heuristic based on the following algorithm, which constructs feasible colorings on $G_\alpha$. Starting with a single blue edge, it iteratively colors two incident edges red and blue until feasibility is obtained.

**Algorithm 1:** Red-Blue Tree

```
  Data: Graph $G_\alpha = (V_\alpha, E_\alpha)$
  Result: Red-blue coloring of $E_\alpha$
1 All vertices in $G_\alpha$ are unmarked, all edges are uncolored.
2 Choose an edge $uv \in E_\alpha$, color it blue, and mark the vertices $u$ and $v$.
3 while there are two unmarked vertices $b,c \in V_\alpha$ and a marked vertex $a$ with
  $(a,b),(b,c) \in E_\alpha$ do
4     Color $(a,b)$ red and $(b,c)$ blue.
5     Mark $b$ and $c$.
6 end
7 For all unmarked vertices $a \in V_\alpha$: Color $(a,a)$ blue.
```

After Step 2, the condition that all $k$ red edges are incident to at least $k+1$ blue edges is satisfied, as there are no red edges. It is obvious that each iteration of Step 3 preserves this property. Furthermore, Step 4 preserves it, as no further red edges are added. After Step 2, the blue edges form an edge cover of the graph induced by all *marked* vertices. This property is preserved by Step 3 and 4. After Step 4, all vertices are marked. The resulting coloring is hence feasible. However, there are feasible colorings which cannot be constructed by the algorithm. An example is a triangle graph with blue loops on all three vertices and red triangle edges. Figure 4.6 visualizes the steps of the algorithm from left to right.

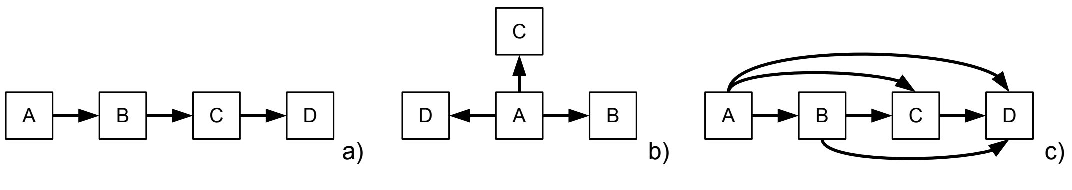


Figure 4.5: Possible market structures of the photo trading network.

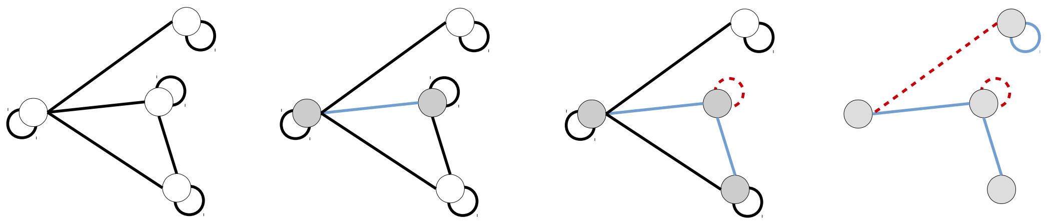


Figure 4.6: Red-Blue Tree: Edges are iteratively colored blue (solid light) and red (dashed).

In the separation algorithm, we use a greedy strategy: The edge $uv$ in Step 1 is chosen randomly. In Step 3, the vertices $a,b,c$ are chosen such that the total weight is increased by the maximum possible amount. That is, such that the sum of the red weight of edge $(a,b)$ and the blue weight of edge $(b,c)$ is maximized. The running time of a trivial implementation of this strategy is in $O(d^4)$, where $d$ is the degree of vertex $u$. This is still acceptable in practice, since in social networks, the average node degree is not only a small number, but also constant with respect to $|V|$ [Mel06].

**Computational Test Table.** We performed computational tests to investigate on the strength of the presented separation algorithm in practice. The results are given in Table 4.1. In this table, every row corresponds to an instance, and every instance is characterized by the first five column entries. $T$ indicates the type of network introduced in the previous section: Watts-Strogatz (ws), distorted random (rd), or pA, pB, pC (German Photo Trading) with image matrices a) to c) from Fig. 4.5. $V$ gives the number of vertices, $A$ the number of arcs. The number of one entries in the image matrix is given in the column $I_1$. *Opt* gives the optimum solution value if known, otherwise an upper bound marked by an asterisk $(*)$. All instances are unweighted, that is, $W \equiv 1$ and $B \equiv 1$. By *LP Gap*, we denote the relative difference between the value *opt* in column *Opt* and the optimum value $lp$ of the integer program's LP relaxation. The gap is hence $(Opt - lp)/lp$. Note that *Opt* does not always contain the optimum solution of the integer program; in the case that it is unknown, the best known primal heuristic value (marked by an asterisk) is taken from column *Opt*. In the case $lp = 0$, we denote the gap by "$\infty$". The value $lp$ for the column *M2lin* is the LP relaxation value for Model 2-lin. For the column *M2lin-*$\alpha$, we compute $lp$ by the following procedure: Compute the LP relaxation optimum of Model 2-lin. Iteratively separate violated constraints of type (3.47) until no violation is found. The resulting optimum value is then a bound on the LP relaxation value for Model 2-lin with all constraints of type (3.47) added. Analogously, the values $lp$ for all Barvinok constraints (3.18) are computed. The Barvinok constraints are separated by complete inspection. The resulting gap is given in column *M2lin-Bar*.

We report the CPU times for solving Problem 2 to optimality for two algorithms. *M2lin* and *M2lin-sep* are the SCIP implementations of Model 2-lin, where in *M2lin-sep* $\alpha-,\beta-$defining and Barvinok constraints are separated. The former ones are separated as explained in this section, the latter ones by enumeration. Separation is only performed in the root node of the branch-and-cut tree. The CPU times are given in seconds (bold numbers). If an instance could not be solved within a 1 hour time limit, we report the remaining gap $(ub - lb)/lb$ between the current upper and lower bound on the optimum value (non-bold numbers).

From the table we conclude the following. The additional $\alpha-$ and $\beta-$defining constraints can improve the LP Gap for only 44% of the tested instances; the Barvinok constraints for 74%. In the remaining cases, they are not violated by the LP optimum solution. The separation cannot improve the total running times. A possible explanation is that the time needed for the separation is relatively large for instances which can be solved by *M2lin* within a few seconds. If we however consider instances which are solved by *M2lin* in the magnitude of minutes or hours and in which the LP gap *M2lin-*$\alpha$ is tighter than the one of *M2lin*, an improvement can be seen. See for example the *ws* instance with 24 vertices, where the running time is improved from 1200 to 517 seconds by the use of separation. To make practical use of this effect, the separation should be started not from the beginning of the branch-and-cut algorithm, but after a certain time threshold in which the problem remains unsolved. Furthermore, the separation of $\alpha-$ and $\beta-$defining constraints should be immediately aborted if the first call does not find any violated cut.

### 4.3.2 **Cycle Inequalities**

The cycle inequalities presented in Proposition 16 on Page 37 can be exactly separated in time complexity $O(V^3 2^{c-1})$. Note that the number of vertex groups $c$ is a small number in most applications. For every partition of the vertex groups $\{V_1,\dots,V_c\}$ into two non-empty groups $L$ and $\{V_1,\dots,V_c\}\setminus L$, call the separation algorithm by Barahona and Mahjoub [BM86] for the max-cut problem. In our adaption of the algorithm, the graph $(V,E_Y)$ needs to be transformed into a graph $G'$ in the following way. For every vertex $u \in V$, $G'$ contains two vertices $u',u''$. For every edge $uv \in E_Y$, $G'$ contains the edges $u'v'$ and $u''v''$ with weight $y^*(\{uv\},L)$, and the edges $u'v''$ and $u''v'$ with weight $1-y^*(\{uv\},L)$. The search for a most violated cycle in the original graph $(V,E_Y)$ is now replaced by the search for a shortest path in the modified graph $G'$, which starts and ends at two representatives $u',u''$ of one vertex $u$ and can thus be re-transformed into a cycle. The fact that the shortest path switches an odd number of times between vertices marked with ' and " leads to an odd number of edges in $F$ after the re-transformation.

**Algorithm 2:** Cycle Separation

```
   Data: $c \geq 2$, $V$, $G'$
   Result: Path in $G'$, $L^*$
1  forall the proper subsets $L_1 \subset [c]$ do
2      Construct $G'$ with $L = \{V_i\}_{i \in L_1}$.
3      forall the $u \in V$ do
4          Compute a shortest path from $u'$ to $u''$ in $G'$.
5          Set $l(L_1,u)$ to its length.
6      end
7  end
8  Set $(u^*,L_1^*) := \arg\min_{u \in V, L_1 \subset [c]} l(L_1,u)$.
9  return Shortest path from $(u^*)'$ to $(u^*)''$ and $L^* = \{V_i\}_{i \in L_1^*}$.
```

The vertices in the shortest path (without ' or ") are the vertices in the cycle. An edge $uv$ is in $F$, if either $u'v''$ or $u''v'$ is in the shortest path. The corresponding cycle inequality is violated by the LP optimum if for the length of the shortest path holds $l(L_1^*,u^*) < 1$.

## 4.4 PRIMAL HEURISTICS

In this section, we introduce primal local search heuristics for Problems 1 and 2. They are generalizations of the Kernighan-Lin heuristic [KL70] for the Minimum-$k$-Cut problem with equal shore sizes. First, we briefly introduce local search heuristics. Then, we explain our new heuristics for Problems of the $\textsc{Block}(\mathbf{F},p)$ type. We show how to implement this heuristic specifically for Problems 1 and 2 in a way which was developed together with Rube [Rub13].
Finally, we test the heuristics for both quality and scalability. The test shows that the heuristics provide near-optimum solutions for graphs with about 100 vertices. For more than 50% percent of the tested instances, even the optimum solution is found. Testing the scalability, we find that the heuristics can be applied to instances with a million arcs within one minute on a non-parallel desktop computer.

**Local Search Heuristics.** *Local Search* is a metaheuristic concept for the solution of combinatorial optimization problems. Starting from a good (or random) feasible solution, a better feasible solution is obtained through a local change in the original solution. This process is iteratively repeated in a usually greedy manner, until no local improvements are possible anymore, the solution quality increases by very small amounts, or a time limit is reached.
Local search heuristics are applicable to link pattern search problems in a natural way. As every feasible solution to the corresponding optimization problem $\textsc{Block}(\mathbf{F},p)$ is a blockmodel (P,I), consisting of a partition $P$ of the vertices and an image matrix $I$, at least three local changes are natural:

1. Change an image matrix entry in $I$ (*image flip*),
2. Move a vertex from one group in $P$ to another (*vertex move*),
3. Choose two vertices in different groups of $P$ and exchange their group memberships (*vertex swap*).

Even though the vertex swap can be simulated by two successive move operations, it is listed separately, as it is considered a single step in the local search heuristic. Larger subsets of vertices are usually not reassigned simultaneously for running time reasons.
Well-known metaheuristics which are considered special cases of the local search metaheuristic are *hill climbing*, *simulated annealing*, and *tabu search*. The latter is known to be implemented in the network analysis software UCINET 6 [BEF02] for use in most link pattern search algorithms. However, no implementation details are given.

**A Kernighan-Lin Heuristic.** We are interested in a heuristic to provide good primal bounds on the optimum values of Problem 1 and 2. The Kernighan-Lin approach is a local search heuristic developed by Kernighan and Lin [KL70] for the minimum $k$-cut problem with vertex groups of equal size. Later, it was successfully adopted by Bonato [Bon11] for the maximum cut problem. Both problems can be seen as special cases of the general link pattern search problem $\textsc{Block}(\mathbf{F},p)$.
The following algorithm is our adaption of the Kernighan-Lin metaheuristic for Problem 1 and 2. It is based on vertex moves and implicit image flips. The heuristic starts with a blockmodel $(P_0,I_0)$

for the input graph $G = (V,E)$. In every step, it searches for a better blockmodel $(P_{i+1}, I_{i+1})$, starting from the current blockmodel $(P_i, I_i)$, by performing a sequence of vertex moves. In this sequence, every vertex of the graph is moved once, unless its move would create an empty group. After the move, the image matrix is updated as the one minimizing the objective function for the given partition. Here, the best image matrix is chosen either from the set of all $c \times c$ image matrices (Problem 1) or is constantly the one image matrix which is given in the formulation of Problem 2. The new blockmodel $(P_{i+1}, I_{i+1})$ is finally chosen to be the best one along the sequence of vertex moves. Note that the procedure makes a random choice wherever there are several alternate possibilities of the same quality. The procedure terminates as soon as the best solution $(P_{best}, I_{best})$ cannot be improved anymore.

The set of candidates for $I_{best}$ is either the set of all image matrices (Problem 1) or only one image matrix (Problem 2). The algorithm can be extended to the intermediate case, where *some* entries of the image matrix are fixed, whereas the others are being optimized upon. The input to the algorithm is hence a matrix $F \in \{0,1,*\}^{c \times c}$, where an asterisk (*) indicates that the corresponding entry is not fixed to 0 or 1. Given such a matrix $F$, we denote by $I(F)$ the set of all realizations in $\{0,1\}^{c \times c}$ of $F$.

Algorithm 3, the following pseudo code, explains the procedure in greater detail.

**Implementation.** We implemented the heuristic in `C++`. The code is attached to the print version of this thesis. Nevertheless, we give brief descriptions on the following implementation choices in this paragraph: The construction of the initial partition $P_0$ in Line 1, the evaluation of the objective function $f$ in Line 8, and the computation of the best image matrix in Line 8.

The initial partition $P_0$ is constructed randomly: For every vertex $u \in V$, a random group is chosen. In the unlikely case that a group remains empty, a random vertex from another non-empty group is moved into the empty group.

In Line 8, the objective function $f$ needs to be evaluated for every possible move of $u$ and every feasible image matrix $I_{i+1}$. The following observations are crucial for a reduction of the running time. Consider Problem 1. By its construction as a subgraph relaxation approach, the objective function $f$ can be decomposed over the set of all group pairs, that is, $f$ can be written as $f(P) = \sum_{A,B \in [c]} p_{AB}(P)$ , see the paragraph "Combining Subgraph Penalties" on Page 21. Assume that vertex $u$ is moved from group $C$ to $D$. In this case, only the values of the form $p_{C\cdot}$, $p_{\cdot C}$, $p_{D\cdot}$, and $p_{\cdot D}$ may possibly change. The same holds for the image matrix entries $I_{C\cdot}$ and $I_{\cdot D}$. Instead of re-evaluating all $c^2$ values, it is hence sufficient to update $4c - 4$ values. For example, consider Figure 4.7, where only the penalties for the gray image matrix entries need to be re-evaluated in the case that a vertex is moved from group $C$ to $D$. First, consider $p_{CX}$ and $I_{CX}$ for a group $V_X$. In the case that $I_{CX} = 0$, the penalty value $p_{CX}$ needs to be decreased by the number of arcs from $u$ into group $X$. These arcs, which needed to be deleted in order to obtain an ideal regular partition, are now not existent anymore due to the move of $u$; hence, the penalty value decreases. On the same time, the penalty values $p_{XD}$ need to be increased by the number of arcs going from group $V_X$ to $u$, in the case that $I_{XD} = 0$. For group pairs with image matrix value 1, the penalty value can only increase or decrease by 1. In order to compute the changes in the $p$-values more quickly,

**Algorithm 3:** Adapted Kernighan-Lin Heuristic

```
Data: Digraph $D = (V,A)$, number $c$ of vertex groups, objective function $f : \mathbf{B}_c(G) \to \mathbb{R}$, a
      matrix $F \in \{0,1,*\}^{c \times c}$ denoting the feasible image matrices.
Result: Best found blockmodel $(P_{best}, I_{best})$
1  Construct a partition $P_0$ of $V$.
2  Set $P_{best} = P_0$.
3  Set $I_{best}$ to $\arg\min_{I \in I(F)} f(P_{best}, I)$.
4  Set $i = 0$ and all vertices $v \in V$ unmarked.
5  while there are unmarked vertices do
6      Choose an unmarked vertex $u$ and mark it.
7      if $u$ is not the only vertex in its group then
8          Apply a vertex move for $u$ from $P_i$ which minimizes $f(P_{i+1}, I_{i+1})$ over all moves of
           $u$ and all $I_{i+1} \in I(F)$, where $P_{i+1}$ is the respective partition after the move.
9      end
10     if $f(P_{i+1}, I_{i+1}) < f(P_{best}, I_{best})$ then
11         Set $P_{best} = P_{i+1}$, $I_{best} = I_{i+1}$.
12     end
13     Set $i = i + 1$.
14 end
15 if $f(P_{best}, I_{best}) = f(P_0, I_0)$ then
16     STOP( Return $(P_{best}, I_{best})$ ).
17 end
18 else
19     Set $i = 0$, $P_0 = P_{best}$. Goto Line 4.
20 end
```

we store for each vertex $u$ and each class $A$ the number of arcs that $u$ has from and into $A$. These values are updated with every vertex move.

Let us now turn to the search for the best image matrix over all $I_{i+1} \in F_I$ in Line 8. In fact, we do not optimize over the image matrices in our implementation. Instead, for each group pair $V_A, V_B$, two penalty values are constantly stored and updated: A value $p_{AB0}$ and a value $p_{AB1}$ for $I_{AB} = 0$ and $I_{AB} = 1$ respectively. Only in Line 16, where the best blockmodel is returned, the image matrix $I_{best}$ is actually created: For each pair $A, B \in [c]$, the values $p_{AB0}$ and $p_{AB1}$ are compared. The entry $AB$ of $I_{best}$ is 0 if the former is lower, 1 if the latter is lower, and a random binary value if both are equal. In this process, the feasibility constraints for the possible image matrices are taken into account.

The implemented algorithm also supports both weight types $W$ and $B$. In the case that a weight matrix $W$ is used, the update of the values $p_{AB1}$ however requires the solution of an edge cover problem. See Rube [Rub13] for details.

**Computational Tests.** We test Algorithm 3 in its implementation explained in the previous paragraph. The first test examines the quality of the heuristic solution by a comparison to exact

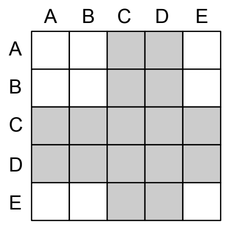


Figure 4.7: Penalties to be re-evaluated.

solution values. Table 4.2 shows the results. The test instances are the ones used for the cutting plane test in Table 4.1. The columns $T$, $V$, $A$, and $I_1$ hence have the same meaning. Column *Opt* contains the optimum solution value when the instance is used as an input to the unweighted Problem 2. Column *Heu* gives the best heuristic value after 10 runs of Algorithm 3.

We see that the heuristic found the optimum solution for 11 out of the 20 instances. The average gap $(Heu - Opt)/Opt$ is 10%. Note however that only those instances could be used for the test for which the optimum solution was found within 1 hour. Their accessibility to the branch-and-cut algorithm might indicate an instance structure which is also more accessible to the heuristic, even though there is no direct relation between the two algorithms.

To test the running time of the heuristic, we generated larger instances for which the optimum solution is unknown. Table 4.3 gives the results. The instances of type *rd* are generated as explained in Section 4.2 with $k$ set to 1000 and $d$ to be 20% of the maximum possible number of arcs given the vertex partition and the image matrix. The graphs of type *ws* are created as explained in Section 4.2. Column *Problem 2* gives the results for the heuristic solving Problem 2, where a fixed image matrix is generated as explained in Section 4.2. Column *Problem 1* gives the results for the heuristic solving this problem, i. e., , all image matrices are feasible and part of the optimization $(F = \{*\}^{c \times c})$. Column *#rounds* gives the number of times Line 4 is executed. One round consists hence of (at most) $|V|$ vertex moves. Recall that a new round is started until the objective value cannot be improved anymore within a round. Column *time / round* gives the total CPU time of the algorithm divided by the number of rounds.

We observe that the running time increases with the number of arcs rather than the number of vertices. The number of rounds is lower than 10 for all but two instances. The value of 1 minutes per round is reached for about 1 million arcs for *rd* and $100,000$ arcs for *ws* instances. The running time for Problem 1 is about twice the running time for Problem 2. This is expected since for every vertex move and every image matrix entry, both possible entries 0 and 1 need to be evaluated in the former case. However, the total running time is still lower for Problem 1, as the average number of rounds is only 2.6, compared to 6.8 rounds for Problem 2.

## 4.5 DUAL HEURISTICS

In this section, we discuss ways to obtain lower bounds on the optimum solution value of Problem 2. The idea of the approach is the decomposition of the original problem into several smaller

ones. More precisely, the input digraph is decomposed into overlapping subgraphs, where the problem is solved on each subgraph separately. The respective solution values are combined to obtain a lower bound on the solution of the original problem. Details of this procedure as well as its implementation were developed together with Haas [Haa13].
We will see that for graphs representing social networks (generated by the Watts-Strogatz procedure), the heuristic improves the dual bound significantly. On average, the gap between dual and primal bound, which lies in $[0,1]$, could be reduced from 0.93 for the LP relaxation to 0.44 for our heuristic.
In Section 4.5.1, we present the overall method. Sections 4.5.2 and 4.5.3 give details on algorithmic choices and subroutines. Finally, Section 4.5.4 presents computational results on tested instances.

### 4.5.1 Decomposition into Overlapping Subgraphs

In the original formulation of Problem 2, the following objective function needs to be minimized, following the notation of Model 2-lin:

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} B_{ab} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} B_{ab} p_{ab}. \tag{4.20}$$

Recall that the penalty value $p_{ab}$ is the maximum of the total $\alpha$ and the total $\beta$ penalty for the vertex group pair $(V_a, V_b)$, such that we can equivalently rewrite this function as

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} B_{ab} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} B_{ab} \max\{\sum_{u\in V} \alpha_{u,ab}, \sum_{u\in V} \beta_{u,ab}\}. \tag{4.21}$$

Before this objective function can be decomposed, it needs to be relaxed. That is, we replace it by the following function, whose values are lower or equal to the above function for any argument. The optimum value of the latter function is hence a lower bound on the optimum value of the former, when being optimized over the constraints of Model 2-lin.

$$\min f_t = \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} B_{ab} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} B_{ab}(t \sum_{u\in V} \alpha_{u,ab} + (1-t) \sum_{u\in V} \beta_{u,ab}) \tag{4.22}$$

for $t \in [0,1]$. This is due to the simple observation that $\max\{a,b\} \geq ta + (1-t)b$ generally holds for any two real numbers $a, b$ and any $t \in [0,1]$. In the remainder of this section, we will always use $t = 0.5$, even though the following argumentation holds for any other choice of $t$. The objective function is hence

$$\min f_{0.5} = \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} B_{ab} y_{ua,vb} + \sum_{\substack{a,b\in[c]:\\ I_{ab}=1}} (0.5 B_{ab} \sum_{u\in V} (\alpha_{u,ab} + \beta_{u,ab})). \tag{4.23}$$

It is now possible to rewrite $f := f_{0.5}$ in the decomposed form $f = \sum_{i=1}^{k} g_i$, with

$$g_i := \sum_{\substack{(u,v)\in A\\ u,v\in\overline{V}_i}} \sum_{\substack{a,b\in[c]:\\ I_{ab}=0}} B_{ab} y_{ua,vb} + 0.5 \sum_{\substack{uv\in A:\\ uv\in\delta(\overline{V}_i)}} \sum_{\substack{a,b\in[c]:\\ I_{ab}=0}} B_{ab} y_{ua,vb} + \sum_{\substack{a,b\in[c]:\\ I_{ab}=1}} (0.5 B_{ab} \sum_{u\in\overline{V}_i} (\alpha_{u,ab} + \beta_{u,ab})), \tag{4.24}$$

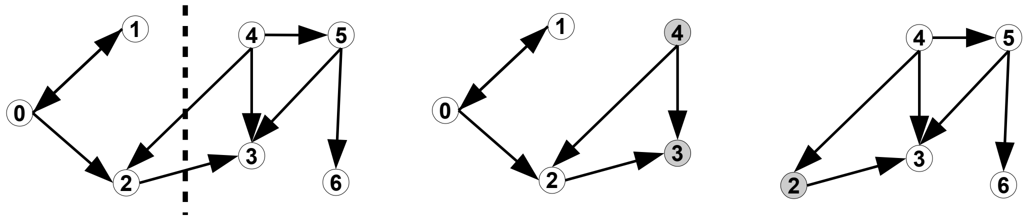


Figure 4.8: The partition (left) induces two subgraphs (center, right).

where $\overline{V}_i \subseteq V$ is a subset of the vertex set $V$ of the input graph $D = (V,A)$ and $\delta(U)$ is the set of all arcs leaving or entering a vertex set $U \subseteq V$, that is,

$$\delta(U) := \{uv \in A \mid (u \in U \wedge v \notin U) \vee (u \notin U \wedge v \in U)\}.$$

Using the above transformation, we deduce Algorithm 4 to obtain a lower bound on the optimum solution of Problem 2.

```
Algorithm 4: DualBound
   Data: Input to Problem 2
   Result: Dual bound $L$
 1 Compute a partition $\{\overline{V}_1,\dots,\overline{V}_k\}$ of $V$.
 2 Set $L = 0$.
 3 for $i = i,\dots,k$ do
 4     Solve Model 2-lin with objective function $g_i$.
 5     Increase $L$ by the optimum solution value.
 6 end
 7 return $L$
```

Note that the vertex groups $\overline{V}_i$ are not related to the vertex groups $V_i$ as part of the optimum blockmodel for Problem 2. The groups $\overline{V}_i$ are merely used *within* the lower bound computation algorithm in order to divide the original problem into smaller ones. However, they do not represent good solutions to Problem 2.

The vertices and arcs appearing in the definition of $g_i$ form a subgraph $D_i$ of the original input graph $D = (V,A)$. The vertex set of $D_i$ consists of the vertices $\overline{V}_i$ (the *core vertices*) and all vertices in $V \setminus \overline{V}_i$ which have at least one arc from or to a core vertex (the *periphery vertices*). The arcs of $D_i$ are all arcs in $A$ induced by the vertex set of $D_i$, except for those arcs connecting two periphery vertices. See Figure 4.8 for an example: The graph on the left is partitioned into two vertex sets by the dashed line. The two resulting subgraphs are depicted in the center and on the right, with core vertices and periphery vertices colored in white and gray respectively.

To show the correctness of the algorithm, it remains to be proven that $\sum_{i=1}^{k} g_i$ is indeed a decomposition of $f$:

**Proposition 26.** Let $\{\overline{V}_1,\dots,\overline{V}_k\}$ denote a partition of $V$ into disjoint non-empty subsets. Then, $f = \sum_{i=1}^{k} g_i$ holds.

*Proof.* The definition (4.24) of the function $g_i$ consists of three double sums. Obviously, the third double sums add up to the second double sum in the definition (4.23) of $f_{0.5}$, as the sets $\overline{V}_i$ form a disjoint partition. It remains to be shown that the first two double sums for $g_i$ add up to the first double sum for $f_{0.5}$. The term $B_{ab}y_{ua,vb}$ is summed up for all arcs $(u,v)$ within the groups, that is, between core vertices. Furthermore, for all arcs from a core vertex to a periphery vertex, the term $0.5B_{ab}y_{ua,vb}$ is summed up two times: One time $u$ is the core and $v$ is the periphery vertex, one time vice versa. All in all, the term $B_{ab}y_{ua,vb}$ is summed up for all $(u,v) \in A$, as in the first double sum in the formula for $f_{0.5}$. □

**Use within Branch-and-Bound.** Consider a branch-and-bound algorithm for Problem 2. The branching rule is the following one: Given the root node in the branch-and-bound tree, choose a vertex $v_0 \in V$ and create $c$ child nodes. In each child node, $v_0$ is fixed to a different vertex group. That is, in child $i$, $x_{v_0 i}$ is fixed to 1. For the third level of the tree, choose another vertex $v_1 \in V$ and create $c$ children for every node in the second level by fixing $x_{v_1 1}$ to $x_{v_1 c}$ respectively. Continue this process analogously. Consider two nodes $H$ and $L$ in the tree, where $H$ is the parent of $L$. Let $v_j$ denote the vertex which was fixed to obtain $L$ from $H$. To calculate the dual bound for $L$ with Algorithm 4, it is sufficient to solve the problem in Line 4 only once. Namely, for the objective function $g_i$ such that $v_j$ is in $V(D_i)$. The objective values of all other problems with objective $g_{i'}$ ($i' \neq i$) remain unchanged in comparison to $H$.

### 4.5.2 Solution of the Subproblems

In the following sense, the optimization problem in Line 4 of Algorithm 4 is usually easier to solve than the overall Problem 2: On the one hand, it is defined on a lower number of vertices and arcs. In fact, only the subgraph $D_i$ of $D$, which was defined in the previous subsection, needs to be considered. On the other hand, the constraints of the integer program *Model 2-lin* can be simplified if its objective function is replaced by $g_i$. The model to be solved in Line 4 can be stated as

**Model 3.**

$$\min g_i$$

$$\begin{aligned}
\sum_{a\in[c]} x_{ua} &= 1 && \text{for } u \in V(D_i),\\
\sum_{u\in V} x_{ua} &\geq 1 && \text{for } a \in [c], \qquad (4.25)\\
x_{ua} + \sum_{v\in N^+(u)} (1-x_{vb}) &\leq \alpha_{uab} + |N^+(u)| && \text{for } u \in \overline{V}_i, a,b \in [c], I_{ab}=1,\\
x_{ub} + \sum_{v\in N^-(u)} (1-x_{va}) &\leq \beta_{uab} + |N^-(u)| && \text{for } u \in \overline{V}_i, a,b \in [c], I_{ab}=1,\\
x_{ua} &= \sum_{b\in[c]} y_{ua,vb} && \text{for all } uv \in E_{A(D_i)}, a \in [c],\\
y_{ua,vb} &\geq 0 && \text{for } E_{A(D_i)}, a,b \in [c],\\
x_{ua}, \alpha_{uab}, \beta_{uab} &\text{ binary,} && \text{for} \ldots \text{(see above)},
\end{aligned}$$

where $E_{A(D_i)}$ denotes the underlying graph as defined on Page 35. Note that instead of summing over all $u \in V$ in Constraint (4.25), it is sufficient to sum over all vertices in $V(D_i)$ together with $\min\{c, |V| - |V(D_i)|\}$ dummy vertices. If $c$ is the minimum, as it is the case in most practical applications of the heuristic, the constraint is satisfied by the dummy vertices and does not hold for the original vertices. Hence, empty vertex groups are usually allowed in the subproblems, leading to two interesting consequences. First, the heuristic provides the same dual bound in the following two cases:

1. There is only 1 subproblem (all vertices of the input graph are core vertices of this subgraph),
2. There are $k$ subproblems, where $k$ is the number of weakly connected components of the input graph (the vertices of a component are the core vertices of the corresponding subgraph).

This means that the heuristic can be applied to each weakly connected component separately, as the total dual bound is the sum of the $k$ component bounds. Second, the heuristic should not be applied if the image matrix has a 1-entry on its diagonal. In the case that each vertex in the input graph has at least one successor and at least one predecessor, the dual bound will be 0, since all vertices can be assigned to the group $i$ with image matrix diagonal entry $I_{ii} = 1$.

### 4.5.3 **Determination of the Subproblems**

In Line 1 of Algorithm 4, a partition $\{\overline{V}_1, \dots, \overline{V}_k\}$ of the vertex set $V$ needs to be determined. The computational experiments of Haas [Haa13] indicate that the quality of the dual bound increases with the number of arcs which are within the subsets $\overline{V}_i$. Hence, the computation of the partition results in a clustering problem. Formally, the problem is to find a number $k$ and a partition $\{\overline{V}_1, \dots, \overline{V}_k\}$ which maximizes

$$\sum_{i=1}^{k} |\{uv \in A \mid u, v \in \overline{V}_i\}|.$$

There is a heuristic by Karger and Stein [KS96] for this problem, which successively contracts arcs. The contraction $contract(u, v)$ of an arc $uv$ is the replacement the two vertices $u$ and $v$ by a new (super-)vertex $w$, where $w$ has an arc to (resp. from) a vertex $x$ (distinct from $u, v, , w$) if $u$ or $v$ had an arc to (resp. from) $x$ prior to the contraction. The Karger algorithm iteratively chooses a random arc and contracts it, until only $k$ vertices are left. Those $k$ super-vertices determine the $k$ groups $\overline{V}_i$ by the vertices they include. It has been shown that the probability to find the optimum solution grows quickly with the number of runs of this randomized algorithm, see Karger and Stein [KS96] for details.

We adopt this idea to construct a heuristic for the problem in Line 1. For our purposes, it is important to limit the maximum number $s_{max}$ of vertices per group $\overline{V}_i$, to reduce the running time of the subproblem optimization. On the other hand, we need to avoid trivially small groups, as the resulting contribution to the dual bound $L$ is probably weak in this case.

Our heuristic is a 2-step procedure. Its input is the maximum number $s_{max}$ of vertices per group, whereas the number $k$ of groups remains unspecified. In the first step, randomly chosen arcs are again iteratively contracted. However, there is the additional rule that a contraction may only be executed if the resulting super-vertex has less than $s_{max}/2$ vertices. We call these arcs *contractable* with respect to the current graph and partition. The procedure stops as soon as there is no contractable arc anymore. In the second step, the resulting groups $\overline{V}_i$ are pairwisely considered. Two groups are united, if the resulting group has less than $s_{max}$ vertices. The procedure *union*$(\overline{V}_i, \overline{V}_j)$ updates the current partition by replacing $\overline{V}_i$ with $\overline{V}_i \cup \overline{V}_j$ and deleting $\overline{V}_j$. Thus, the first step avoids large groups, whereas the second step is designed to avoid small ones. The precise heuristic is given as a pseudo code in Algorithm 5.

**Algorithm 5:** Contraction

```
   Data: $D = (V,A)$, $s_{max}$
   Result: Partition $\overline{P} = \{\overline{V}_1, \dots, \overline{V}_k\}$
1  Set $P$ as $\overline{V}_u = u$ for all $u \in V$.
2  Compute the set $C \subseteq A$ of contractable arcs w.r.t. $P$.
3  while $C \neq \emptyset$ do
4      Randomly choose $(u,v) \in C$.
5      contract$(u,v)$
6      Update $P$ and $C$.
7  end
8  for $i,j = 1,\dots,|P|, i < j$ do
9      if $|\overline{V}_i| + |\overline{V}_j| \leq s_{max}$ then
10         union($\overline{V}_i, \overline{V}_j$)
11     end
12 end
13 return $P$
```

### 4.5.4 Computational Tests

We perform computational tests to examine the quality and speed of the proposed dual bound procedure. Both quality and running times are compared to the LP relaxation value; the standard dual bound in branch-and-cut approaches. The LP relaxation value can be computed quicker, since our procedure requires the solution of several integer programs with binary variables. We will see, however, that the gap between primal and dual bound can be more than halved by the heuristic, even though only 8-times more computational time is being used (note that most bounding procedures do not increase the absolute value of the dual bound linearly in time).

Table 4.4 on Page 79 contains the results of our tests. Its entries have the following meaning. Column $T$ specifies the type of random network, $V$ and $A$ the number of vertices and arcs in the network. Column $I_1$ gives the number of 1-entries in the image matrix, which is randomly computed as a $4 \times 4$ binary matrix with a diagonal forced to be 0 (see Section 4.5.2). Column *opt-sol* gives the optimum solution value for each instance being an input to the unweighted Problem 2. Column *LP* and *bound* gives the values of the LP relaxation of Model 2-lin and our dual bound, respectively. Columns ending with *-t* give the CPU times for computing the respective bound or optimum solution. Finally, Column # *sub* gives the number of subproblems generated by our heuristic.

Before the heuristic can divide the original problem into smaller subproblems, the objective function needs to be relaxed. The gap between the optimum values of the original problem and the relaxed problem can hence never be bridged. It is hence a necessary requirement for the applicability of the heuristic that this gap is small. The results of the corresponding test can be obtained from the left-hand half of Table 4.4. Column *opt-sol* gives the optimum solution to the original problem, whereas Column *bound* gives the optimum solution to the relaxed problem; that is, the heuristic bound when only 1 subproblem is used. With the exception of the outlier instance *ws25*, the gap between the two optima is 4% on average.

Despite of this moderately sized minimum gap, he second test shows that the heuristic can significantly improve the LP relaxation bound in an acceptable amount of time. To this end, consider the right-hand side of Table 4.4. For the networks of type *ws*, we set the maximum size $s_{max}$ of the vertex subsets $\overline{V}_i$ to 15. That is, every subinstance consists of at most 15 core vertices and their neighbors. For instances of type *rand*, a hybrid strategy is used: If the LP relaxation is solved within less than 10 seconds, $s_{max}$ is set to $|V|$ (marked by an asterisk in Column # *sub*; the number of subproblems can still be greater than 1 as connected graph components are treated separately). Otherwise, it is set to 15. Using this setting, the computation of the heuristic dual bound takes 8.5-times as long as the computation of the LP relaxation on average. On the other hand, we see that the dual bound can be significantly improved for all instances of type *ws*. As the instances are too large to be solved exactly, we use the primal heuristic from Section 4.4 to compute the gap $pb - db/pb$ between dual and primal bound. For instances of type *rand*, the reduction is only from 54.9% to 51.3%, such that the usage of the heuristic is not recommended for this instance type. However, for instances of type *ws*, this gap can be significantly reduced from a 93% to a 44% average.

## 4.6 COMPUTATIONAL RESULTS

In this section, we test and compare complete branch-and-cut solvers for the unweighted Problem 2. They combine the components introduced in the previous sections as well as the modeling choices introduced in Chapter 3.

In Section 4.6.1, we will see that our solver is up to $10,000$ times faster than comparable models from literature when applied to our test instances. Section 4.6.2 examines how to predict the hardness of an instance to our solver, as the number of network vertices turns out to be an unsuitable indicator.

All models, separation routines and heuristics to be tested were implemented in the branch-and-cut framework SCIP 3.1.0 [Ach09] in C++. In SCIP, we used the default values for all 1628 standard parameters. SCIP uses CPLEX 12.6 as its LP solver. See Section 4.2 for details on the computer system.

### 4.6.1 **Overall Performance of the Solver**

We test the performance of several branch-and-cut solvers for Problem 2. First, a model from literature is introduced and modified for comparison to our models. Then, the running time table is explained and discussed.

**Model by Brusco and Steinley.** To our knowledge, we are the first authors to consider Problems 1 and 2. In order to examine the quality of our integer programming formulation, we consider a model by Brusco and Steinley [BS09] for a problem which slightly differs from Problem 2. Instead of using our objective function

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} p_{ab}, \tag{4.26}$$

they minimize

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} \Big(\sum_{u\in V} \alpha_{uab} + \beta_{uab}\Big). \tag{4.27}$$

That is, they minimize the sum of all $\alpha$ and $\beta$ variables, instead of their maximum values $p_{ij} = \max\{\sum_{u\in V}\alpha_{uij}, \sum_{u\in V}\beta_{uij}\}$. Their model can be easily transformed into a model for Problem 2. To this end, the objective function needs to be replaced and all constraints defining the $p$-variables need to be added. We obtain the following modified Brusco-Steinley model for Problem 2. The difference to Model 2 lies in the way the $\alpha$- and $\beta$-variables are defined by (4.32), (4.33) and that the Balas linearization (4.29)–(4.31) is used.

**Model BS.**

$$\min \sum_{(u,v)\in A} \sum_{\substack{a,b\in[c]\\ I_{ab}=0}} y_{ua,vb} + \sum_{\substack{a,b\in[c]\\ I_{ab}=1}} p_{ab} \tag{4.28}$$

s.t. (4.10), (4.11), (4.14), (4.15), (4.16), and

$$y_{ua,vb} \le x_{ua} \text{ for all } u,v\in V, u\neq v, a,b\in[c], \tag{4.29}$$

$$y_{ua,vb} \le x_{vb} \text{ for all } u,v\in V, u\neq v, a,b\in[c], \tag{4.30}$$

$$y_{ua,vb} \ge x_{ua} + x_{vb} - 1 \text{ for all } u,v\in V, u\neq v, A,B\in[c], \tag{4.31}$$

$$\alpha_{uab} + \sum_{v\in N^+(u)} y_{ua,vb} \le x_{ua} \text{ for } u\in V, a,b\in[c], I_{ab}=1, \tag{4.32}$$

$$\beta_{uab} + \sum_{v\in N^-(u)} y_{ub,va} \le x_{ub} \text{ for } u\in V, a,b\in[c], I_{ab}=1. \tag{4.33}$$

**Computational Test.** The following models are compared to each other. They form a sequence of improvements from the basic BS model to the fully improved model M2-BC:

| | |
|---|---|
| BS. | *Model BS*, the model by Brusco and Steinley transferred to our objective function. |
| BS-y. | *Model BS*, where $y$-variables are only used on edges, that is, $y_{ua,vb}$ is only introduced for $(u,v) \in A$ or $(v,u) \in A$. |
| BS-FY. | Model BS-y, where Balas linearization is replaced by Frieze-Yadegar linearization. |
| M2. | *Model2-lin*, our model formulation. |
| M2-BC. | *Model 2-lin*, where Barvinok and $\alpha-, \beta-$defining constraints are separated in the root node and the Kernighan-lin heuristic is called once in the preprocessing phase. |

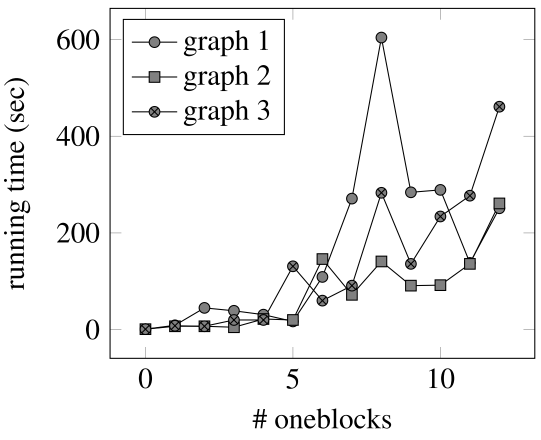
The test instances are described as in Table 4.1 for the separation heuristic, see Page 60. By *LP gap*, we denote the relative difference between the optimum solution value *opt* of the integer program and the optimum value $lp$ of its LP relaxation. The gap is hence $(opt - lp)/lp$. Note that *opt* may be unknown; in this case, it is replaced by the best-known primal value (marked by an asterisk). In the case $lp = 0$, we denote the gap by $\infty$. We report the CPU times for solving Problem 2 to optimality. The times are given in seconds (bold numbers). If an instance could not be solved within a 1 hour time limit, we report the remaining gap $(ub - lb)/lb$ between the current upper and lower bound on the optimum value (non-bold numbers).

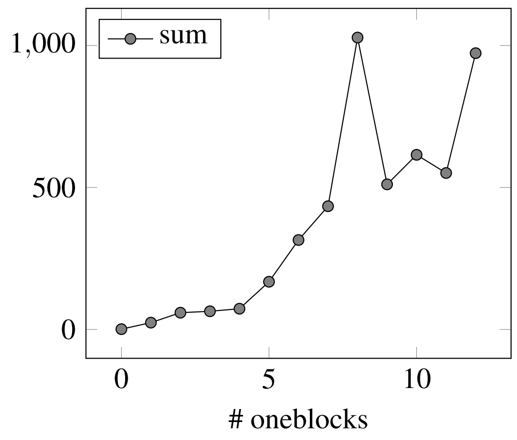
**Results.** Concerning the running times, the new model formulation *M2* leads to better results than the improved model *BS-y* from literature for all instances. Either the problem is solved faster or the gap after one hour is reduced. The maximum improvement is achieved on the *rd* instance with 150 vertices, where the solution time could be reduced from 46 hours to 4 seconds by the use of the new *Model 2* in combination with the separation of the new constraints, reducing the running time by a factor of 40,000.
Concerning the LP gap, the replacement of the Balas by the Frieze-Yadegar linearization (columns BS/BS-y and BS-FY) leads to a significant improvement of the LP gap in all instances, except for the ones where the dual bound equals 0 (entry "$\infty$") in column BS/BS-y. The bound can be further improved for 74% of the instances by the addition of all $\alpha-, \beta-$defining and Barvinok constraints.

### 4.6.2 Hardness Dependence on Input Data

In this section, we investigate the observation that instances of approximately the same size lead to significantly different running times. We observe in Table 4.5 that the running times of instances with approximately the same number of vertices vary to a great extent: The *rd* instance with 190 vertices can be solved within a second, whereas the dual bound of the instance with 200 vertices is still 0.0 after one hour. Although we cannot explain the latter effect, it seems that the number $I_1$ of arcs in the image graph is a suitable indicator for the difficulty of the *rd* instances to our method: If the number of arcs is less or equal to 6, the running time is below 30 seconds. If it is larger or equal to 12, the instance could not be solved within one hour. If it is in-between, the running time ranges from 31 to 258 seconds.
For the test, we use the Watts Strogatz model (ws) for directed graphs. We fix the number of vertices to 20, the number of arcs to 160, and the number of vertex groups to 4. We create three such random graphs named graph1 to graph3. The rewire probability in the Watts-Strogatz algorithm is set to 10%. For each graph and all $i = 0, \ldots, 12$, we randomly generate an image $4 \times 4$ image

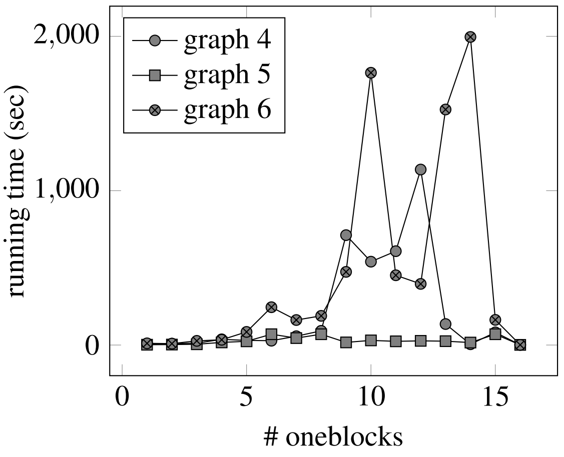
matrix with zeros on the main diagonal and exactly *i* one entries off-diagonally. The following figure shows the running times of our solver to optimize the resulting problems.





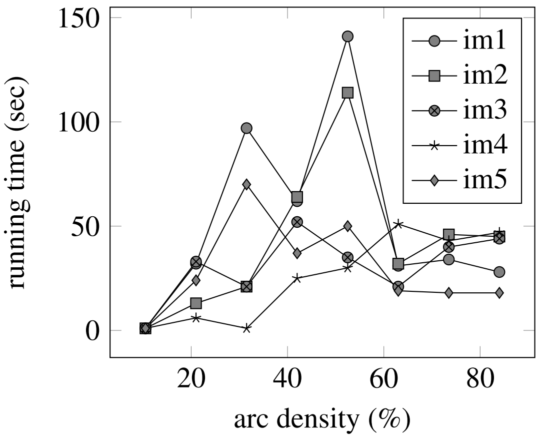
We see that the running time indeed increases tendentiously with the number of arcs in the image graph.

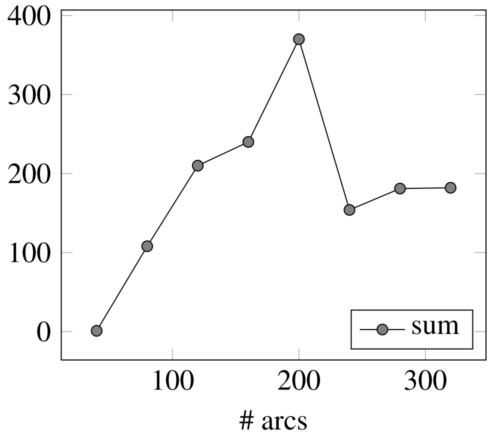
To test whether a one entry on the diagonal has an effect on the solution times, we repeat the test with three graphs named *graph4* to *graph6*. Again, $4 \times 4$ image matrices are randomly created with a uniform distribution. However, only image matrices with a non-zero diagonal are accepted. The resulting running times are displayed in the following figure.



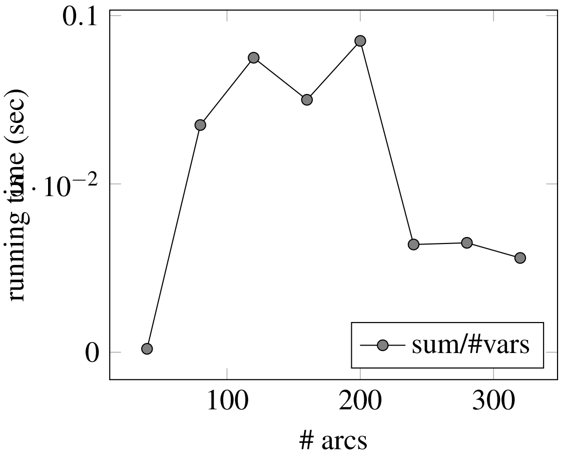
We observe that the running times increase for 1 to 10 one entries in the image matrix. However, it decreases between 10 and 12 such entries, i. e., if the image graph is (almost) complete. Furthermore, we do not see any clear running time difference to the zero-diagonal instances. We conclude that the diagonal entries are not of special importance to the performance of our solver.

We now test how the number of arcs influences the running times of the solver. To this end, we compute five $4 \times 4$ image matrices *im1* to *im5* uniformly at random. For each image matrix, we use the Watts-Strogatz model to generate random graphs with 20 vertices and 40, 80, 120, 160, 200, 240, 280, and 320 arcs. The rewire probability is set to 10%. The running times to solve these instances are displayed in the following figure.





We see that the hardest instances are those with a density of about 50%. Both sparser and denser instances are easier to solve. However, the increase in running time is moderate: The following figure shows that the time per variable in the model is almost constant in the interesting hard mid section. Recall that the number of variables is approximately the number of vertex groups times the number of arcs.



| Instance | | | | | LP Gap (%) | | | **Time (s)** / Gap (%) | |
|---|---|---|---|---|---|---|---|---|---|
| T | V | A | $I_1$ | Opt | M2lin | M2lin-$\alpha$ | M2lin-Bar | M2lin | M2lin-sep |
| ws | 23 | 276 | 8 | 102* | ∞ | ∞ | 52 | 43 | 13 |
| ws | 24 | 288 | 5 | 120 | ∞ | 545 | 22 | **1200** | **517** |
| ws | 25 | 300 | 3 | 199 | 45 | 45 | 10 | **791** | **813** |
| ws | 26 | 312 | 8 | 92 | ∞ | ∞ | 106 | **956** | **2399** |
| ws | 27 | 324 | 7 | 98* | ∞ | ∞ | 42 | 27 | 18 |
| ws | 28 | 336 | 3 | 227 | 35 | 35 | 12 | **986** | 3 |
| ws | 29 | 348 | 3 | 232 | 40 | 39 | 11 | **3170** | 2 |
| ws | 30 | 360 | 5 | 147 | ∞ | 461 | 21 | 30 | **2162** |
| rd | 130 | 883 | 4 | 16 | 64 | 7 | 7 | **10** | **25** |
| rd | 140 | 1147 | 3 | 6 | 0 | 0 | 0 | **1** | **1** |
| rd | 150 | 1725 | 7 | 18 | 260 | 125 | 0 | **258** | **4** |
| rd | 160 | 2128 | 12 | 16* | ∞ | ∞ | ∞ | ∞ | ∞ |
| rd | 170 | 1866 | 7 | 12 | 36 | 8 | 4 | **53** | **92** |
| rd | 180 | 1504 | 4 | 17 | 75 | 75 | 36 | **18** | **23** |
| rd | 190 | 1260 | 5 | 13 | 0 | 0 | 0 | **2** | **2** |
| rd | 200 | 4390 | 14 | 15* | ∞ | ∞ | ∞ | ∞ | ∞ |
| rd | 40 | 250 | 7 | 25 | 373 | 299 | 194 | **31** | **22** |
| rd | 42 | 383 | 16 | 32* | 166 | 166 | 166 | 128 | 142 |
| rd | 44 | 233 | 2 | 59 | 1 | 1 | 0 | **1** | **1** |
| rd | 46 | 468 | 15 | 34* | 82 | 82 | 82 | 51 | 61 |
| rd | 48 | 403 | 15 | 17* | 112 | 112 | 112 | 82 | 122 |
| rd | 50 | 303 | 8 | 45 | 246 | 246 | 47 | **244** | **415** |
| rd | 52 | 517 | 5 | 49 | 262 | 68 | 18 | **19** | **23** |
| rd | 54 | 398 | 13 | 41 | 530 | 528 | 529 | 139 | 191 |
| pA | 70 | 87 | 3 | 33 | 2 | 1 | 1 | **1** | **1** |
| pB | 70 | 87 | 6 | 101 | 8 | 6 | 6 | **8** | **9** |
| pC | 70 | 87 | 6 | 63 | 15 | 10 | 10 | **4** | **21** |

Table 4.1: Test of the separation routines.

| T | V | A | $I_1$ | Opt | Heu |
|---|---|---|---|---|---|
| ws | 24 | 288 | 5 | 120 | 120 |
| ws | 25 | 300 | 3 | 199 | 199 |
| ws | 26 | 312 | 8 | 92 | 92 |
| ws | 28 | 336 | 3 | 227 | 227 |
| ws | 29 | 348 | 3 | 232 | 233 |
| ws | 30 | 360 | 5 | 147 | 147 |
| rd | 130 | 883 | 4 | 16 | 16 |
| rd | 140 | 1147 | 3 | 6 | 16 |
| rd | 150 | 1725 | 7 | 18 | 18 |
| rd | 170 | 1866 | 7 | 12 | 12 |
| rd | 180 | 1504 | 4 | 17 | 17 |
| rd | 190 | 1260 | 5 | 13 | 13 |
| rd | 40 | 250 | 7 | 25 | 28 |
| rd | 44 | 233 | 2 | 59 | 59 |
| rd | 50 | 303 | 8 | 45 | 47 |
| rd | 52 | 517 | 5 | 49 | 53 |
| rd | 54 | 398 | 13 | 41 | 42 |
| pA | 70 | 87 | 3 | 33 | 34 |
| pB | 70 | 87 | 6 | 101 | 102 |
| pC | 70 | 87 | 6 | 63 | 65 |

Table 4.2: Results of the quality test of the primal heuristic.

| Instance | | | | Problem 2 | | Problem 1 | |
|---|---|---|---|---|---|---|---|
| T | A | V | $I_1$ | # rounds | time / round | # rounds | time / round |
| rd | 149,722 | 2000 | 1 | 2 | 1s | 2 | 5s |
| rd | 161,634 | 1000 | 13 | 4 | 7s | 2 | 8s |
| rd | 291,530 | 3500 | 2 | 5 | 8s | 2 | 44s |
| rd | 456,460 | 1500 | 16 | 1 | 37s | 2 | 37s |
| rd | 771,887 | 2500 | 9 | 6 | 56s | 2 | 109s |
| rd | 1,594,178 | 3000 | 12 | 7 | 167s | 2 | 287s |
| rd | 3,113,115 | 4000 | 13 | 9 | 457s | 1 | 673s |
| rd | 3,415,494 | 4500 | 13 | 3 | 620s | 1 | 975s |
| rd | 4,624,531 | 5000 | 13 | 5 | 767s | 2 | 1461s |
| ws | 12,000 | 1,000 | 6 | 7 | 1s | 3 | 1s |
| ws | 24,000 | 2,000 | 7 | 7 | 3s | 3 | 6s |
| ws | 36,000 | 3,000 | 6 | 5 | 7s | 4 | 15s |
| ws | 48,000 | 4,000 | 6 | 14 | 13s | 3 | 27s |
| ws | 60,000 | 5,000 | 5 | 8 | 17s | 3 | 43s |
| ws | 72,000 | 6,000 | 5 | 9 | 29s | 3 | 64s |
| ws | 84,000 | 7,000 | 6 | 9 | 42s | 5 | 84s |
| ws | 96,000 | 8,000 | 5 | 9 | 49s | 3 | 112s |
| ws | 108,000 | 9,000 | 7 | 11 | 78s | 4 | 143s |
| ws | 120,000 | 10,000 | 5 | 9 | 82s | 3 | 177s |

Table 4.3: Results of the running time tests for the primal heuristic.

| T | V | A | $I_1$ | opt-sol | bound | opt-t | bound-t | T | V | A | $I_1$ | LP | bound | LP-t | bound-t | # sub |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ws | 20 | 80 | 5 | 39 | 38.5 | 3s | 3s | ws | 200 | 2000 | 5 | 82.1 | 447.5 | 9s | 40s | 22 |
| ws | 25 | 100 | 9 | 184 | 17 | 81s | 47s | ws | 250 | 2500 | 8 | 0 | 338.5 | 3s | 27s | 30 |
| ws | 30 | 120 | 6 | 53 | 52.5 | 7s | 6s | ws | 300 | 3000 | 5 | 144.4 | 724 | 17s | 46s | 37 |
| ws | 35 | 140 | 8 | 30 | 28.5 | 697s | 596s | ws | 350 | 3500 | 9 | 0 | 502.5 | 5s | 101s | 42 |
| ws | 40 | 160 | 9 | 25 | 24 | 1630s | 1082s | ws | 400 | 4000 | 6 | 0 | 851.5 | 3s | 124s | 48 |
| ws | 45 | 180 | 9 | $\leq 30$ | 23 | $>3600$s | 1283s | ws | 450 | 4500 | 7 | 188.8 | 1054.5 | 91s | 131s | 52 |
| ws | 50 | 200 | 6 | 85 | 84 | 49s | 32s | ws | 500 | 5000 | 7 | 203.9 | 791.5 | 49s | 97s | 62 |
| rd | 45 | 237 | 7 | 51 | 49 | 417s | 403s | rd | 200 | 2117 | 3 | 101.4 | 100 | 1s | 2s | 11* |
| rd | 50 | 185 | 5 | 71 | 68.5 | 84s | 34s | rd | 210 | 7887 | 9 | 0 | 139 | 67s | 456s | 17 |
| rd | 55 | 462 | 10 | 52 | 46 | 21s | 15s | rd | 220 | 3758 | 5 | 171.6 | 172.5 | 4s | 7s | 1* |
| rd | 60 | 208 | 4 | 74 | 72 | 102s | 42s | rd | 230 | 4615 | 4 | 1003.1 | 1023 | 39s | 152s | 10 |
| rd | 75 | 304 | 2 | 143 | 139.5 | 45s | 79s | rd | 240 | 1812 | 2 | 73 | 62 | 1s | 2s | 75* |
| rd | 80 | 287 | 5 | 56 | 44 | 17s | 2s | rd | 250 | 8413 | 7 | 3.41 | 642 | 20s | 492s | 20 |
| rd | 85 | 139 | 0 | 139 | 139 | 1s | 1s | rd | 260 | 14602 | 12 | 0 | 0 | 7s | 3600s | 1* |
| rd | 90 | 383 | 1 | 155 | 153 | 2s | 1s | rd | 270 | 15258 | 11 | 0 | 82.5 | 1035s | 1417s | 22 |
| rd | 95 | 485 | 6 | 181 | 179.5 | 429s | 314s | rd | 280 | 9184 | 7 | 43.7 | 213.5 | 83s | 371s | 22 |

Table 4.4: Results of the computational tests of the dual heuristic.

| Instance | | | | | LP Gap (%) | | | **Time (s)** / Gap (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| T | V | A | $I_1$ | Opt | BS / BS-y | BS-FY | M2-sep | BS-y | M2 | M2-sep |
| ws | 23 | 276 | 8 | 102* | ∞ | ∞ | 52 | 129 | 43 | 13 |
| ws | 24 | 288 | 5 | 120 | ∞ | ∞ | 22 | 83 | **1200** | **517** |
| ws | 25 | 300 | 3 | 199 | ∞ | 45 | 10 | 123 | **791** | **813** |
| ws | 26 | 312 | 8 | 92 | ∞ | ∞ | 106 | 176 | **956** | **2399** |
| ws | 27 | 324 | 7 | 98* | ∞ | ∞ | 42 | 206 | 27 | 18 |
| ws | 28 | 336 | 3 | 227 | ∞ | 35 | 12 | 174 | **986** | 3 |
| ws | 29 | 348 | 3 | 232 | ∞ | 40 | 11 | 167 | **3170** | 2 |
| ws | 30 | 360 | 5 | 147 | ∞ | ∞ | 21 | 159 | 30 | **2162** |
| rd | 130 | 883 | 4 | 16 | 966 | 64 | 7 | **473** | **10** | **25** |
| rd | 140 | 1147 | 3 | 6 | 0 | 0 | 0 | **150** | **1** | **1** |
| rd | 150 | 1725 | 7 | 18 | 350 | 260 | 0 | 8322 | **258** | **4** |
| rd | 160 | 2128 | 12 | 16* | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| rd | 170 | 1866 | 7 | 12 | 298 | 36 | 4 | 4900 | **53** | **92** |
| rd | 180 | 1504 | 4 | 17 | 1600 | 75 | 36 | **846** | **18** | **23** |
| rd | 190 | 1260 | 5 | 13 | 1705 | 0 | 0 | **848** | **2** | **2** |
| rd | 200 | 4390 | 14 | 15* | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| rd | 40 | 250 | 7 | 25 | 733 | 373 | 194 | **2806** | **31** | **22** |
| rd | 42 | 383 | 16 | 32* | ∞ | 166 | 166 | 314 | 128 | 142 |
| rd | 44 | 233 | 2 | 59 | ∞ | 1 | 0 | 47 | **1** | **1** |
| rd | 46 | 468 | 15 | 34* | 403 | 82 | 82 | 119 | 51 | 61 |
| rd | 48 | 403 | 15 | 17* | 457 | 112 | 112 | 233 | 82 | 122 |
| rd | 50 | 303 | 8 | 45 | 542 | 246 | 47 | 110 | **244** | **415** |
| rd | 52 | 517 | 5 | 49 | 4102 | 262 | 18 | 69 | **19** | **23** |
| rd | 54 | 398 | 13 | 41 | 1266 | 530 | 528 | 733 | 139 | 191 |
| pA | 70 | 87 | 3 | 33 | 388 | 2 | 1 | **19** | **1** | **1** |
| pB | 70 | 87 | 6 | 101 | 56 | 8 | 6 | **259** | **8** | **9** |
| pC | 70 | 87 | 6 | 63 | 377 | 15 | 10 | **95** | **4** | **21** |

Table 4.5: Results of the computational tests of the complete branch-and-cut solver.

# Chapter 5

# LINK PATTERNS IN THE WORLD TRADE NETWORK

In this chapter, we show that the heuristics and integer programming-based algorithms developed in Chapter 4 can be used in practical network analysis.

To this end, we analyze a world trade network provided by the United Nations, which comprises the pairwise amount of trade between 229 countries.

In Section 5.1, we explain the data, related work and the goals of our analysis. Section 5.2 reports our search for a pattern which underlies the world trade network. We will see that the network deviates by only 0.14% from the optimum pattern we have computed. In Section 5.3, we are the first authors to classify trade networks. That is, for a collection of trade networks for different commodities, we find common patterns which underlie all of the markets.

## 5.1 DATA AND GOALS

**The Comtrade Database.** The *United Nations Commodity Trade Statistics Database (COMTRADE)* [Com15] provides freely available data on international trade. The trade partners are divided into 288 countries or trade zones, existing currently or in the past. Data on the trade between these partners are reported by the partners themselves, both on import and export. The reports were submitted year-wise from 1962 to 2014. For each pair of partners, the user of the database can choose to download either the total trade or the trade of a single commodity. Commodities are thereby defined by different classification schemes. In the remainder of this chapter, we always use the *Standard International Trade Classification, Revision 3 (SITC)*. The amount of trade can be either obtained in natural units (kilograms, liters, pieces, etc.) or by its total value in US-dollars. See [Nat98] for detailed information on the data and its collection.

In this chapter, we examine the trade data report for the year 2010, as extracted in May 2015 from the database. We did not choose the latest data as it might still be incomplete [Nat]. We consider only export reports (the sum of exports might not equal the sum of imports for various reasons, see [Nat]). As a measure of trade amount, we always choose the value of the trade in US dollars.

**Preprocessing the Data.** For every commodity (or the "total trade"), we are given a file containing a table in the form of a comma separated list. Every row of the table describes the export from a country $u$ to another country $v$. Given $n$ countries and assuming total trade, there are hence $n(n-1)$ rows in each file. We transform the table into a weighted digraph $D = (V,A)$, where the vertices $V$ are the participating countries and the weight $w_{uv}$ on arc $(u,v) \in A$ gives the amount of trade from country $u$ to $u$ in US dollars. In a preprocessing step, we remove some vertices from the digraph as they do not represent single countries, but accumulations ("Europe EU", "Eastern Europe", "The World"), whose member countries are already represented by separate vertices. Thus, vertices with the following trade partner IDs are deleted: AFR, CAC, EEU, EFT, ESH, EU, EU2, LAI, NAF, NCA, NEU, OAF, OAM, OAS, OCE, OEU, UMI, USP, WAS, WLD, XXX, XXY, ZAF, and ZON. These abbreviations are explained in the *Table of Country Names* on Page 105.

**Goals.** For the world trade digraph and some of its sub digraphs, we solve Problem 1: Find the pattern (image matrix) which describes the digraph's structure in the most accurate way. That is, a minimum number of entries in the digraph's adjacency matrix need to be changed in order to obtain the pattern. Which number of vertex groups and which pattern describes the trade network most accurately?

Note that we do not directly answer the question whether the world trade digraph *has* a clear link pattern. We merely compute the best description out of all possible patterns. However, we examine to which extent the found solution depends on the parameter choices in our analysis algorithm.

In Section 5.3, we search for common patterns in 9 trade digraphs for different product groups. Is there a small number of patterns such that each network can be assigned to one of these patterns? Does trade in general follow a small number of reoccurring patterns?

**Related Usage of the Database.** Lloyd et al. [LML09] recently give a survey on publications on world trade analysis. According to their analysis, the majority of the published approaches follow a certain scheme. First, for each pair $u, v$ of countries, a similarity value $s_{uv}$ is computed. A large value indicates that $u$ and $v$ should be placed in the same vertex group, whereas a low value indicates the opposite. The partition $P$ is then obtained by a clustering (clusters contain large similarity values) or multi-dimensional scaling.

The corresponding image graph $I$ itself is either obtained through a human interpretation of $P$ or is a priorily assumed. Srholec [Srh06] assumes a core-periphery trade pattern a priorily and uses a *coreness measure* to determine which of the groups in $P$ is the core group. Mahutga and Smith [MS06] determine the core intuitively. Smith and White [SW92] assume a core-semi-periphery-periphery pattern and find that there are in fact two semi-peripheries. The reason for this a priori assumption is the work of earlier authors, stating a core-periphery trade structure (see also Dasandi [Das14]).

In this chapter, our focus lies on the link pattern rather than on the partition of the vertices. We do not assume any pattern (or collection of possible patterns) a priori. Instead, we optimize over all patterns to find the most suitable description of the world trade network. The actual partition $P$ of the countries is stated, but not taken into account.

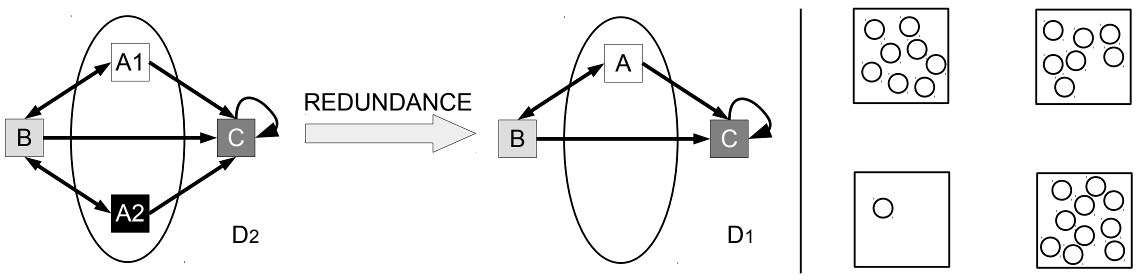


Figure 5.1: Left: Redundant groups $A_1$ and $A_2$. Right: A degenerated image graph with optimum vertex partition.

## 5.2 ANALYSIS OF THE WORLD TRADE NETWORK

We analyze the Comtrade network on total trade as described in the previous section. That is, we analyze digraphs, where the vertices represent the world's countries and the arcs give the amount of trade in the year 2010 quantified in US dollars. Our algorithms search for patterns of link existence in the given digraphs. The output of the algorithms are hence blockmodels $(P, I)$, consisting of a partition $P$ of the countries and a link pattern represented by an image matrix $I$. The penalty value associated with a blockmodel is thereby measured as in Problem 2 treated in Chapter 4. That is, we count the number of arcs in the digraph which do not fit the pattern. The lower this number, the lower the penalty, and thus the higher the quality of the pattern.

In order to exclude trivial and unnecessarily complex solutions from the output of the algorithm, we define two kinds of unsuitable image matrices: Redundant and degenerated image matrices.

**Redundant Image Matrices.** Consider two image graphs $D_1$ and $D_2$, where $D_2$ evolves from $D_1$ by applying the SPLIT operation (Page 24) on a vertex $A$ in $D_1$, where $A$ does not have a self-loop in $D_1$. The operation splits vertex $A$ into two subvertices $A_1$ and $A_2$. Recall that both subvertices inherit from $A$ its relations to the other vertices. See Figure 5.1 for an example. $D_2$ now contains redundant information: The network vertices in both vertex groups $A_1$ and $A_2$ do not trade within the group nor with the other group. Furthermore, they have the same relations to all other groups. For the analysis of the trade market, both $D_1$ and $D_2$ thus provide exactly the same information. Clearly, $D_1$ is the preferred representation as it contains no redundant information. The procedure REDUNDANCE($I$), which will be used in our analysis algorithm, searches for all redundant pairs of vertices in the image graph of $I$ and deletes one vertex in each pair. Note that the deletion of one vertex of a pair is equivalent to a "merger" of the pair. The procedure returns the resulting image graph, which is non-redundant in the above sense. For a multi-digraph, i. e., a digraph $D = (V, A_1, \ldots, A_r)$ with several arc sets, the definition is related. A pair $\{i, j\} \subseteq [c]$ of image graph vertices is redundant w.r.t. a set $\{I_1, \ldots, I_r\}$ of $r$ image matrices, if it is redundant w.r.t. $I_i$ for all $i = 1, \ldots, r$. Not only does the reduction lead to a pattern with the same interpretation, it may also improve the optimum solution value in Problem 2. At least, the optimum cannot become worse due to the reduction, as the following proposition shows.

**Proposition 27.** The optimum value to Problem 2 for input $(D,I)$ is larger or equal to the optimum value for input $(D,\text{REDUNDANCE}(I))$.

*Proof.* Let $D(I)$ denote the digraph with adjacency matrix $I$. Consider the optimum solution $(H^*,P^*)$ to Problem 2 with input $(D,I)$. Let $opt$ denote its solution value. In a first step, assume that REDUNDANCE($I$) unions two vertices $A_1$ and $A_2$ in $D(I)$ into a new vertex $A$. From Proposition 4 on Page 24 follows that the solution $(H^*,P^*)$ is feasible to Problem 2 with input graph $D$ and input image matrix REDUNDANCE($I$). As the value of this feasible solution is again $opt$, the proposition follows from the observation that the REDUNDANCE operation is a sequence of vertex pair unions. □

Note that the proof does not make use of the precise form of the objective function in Problem 2. The proposition is hence true for *any* subgraph relaxation approach.

**Degenerated Image Matrices.** We call a partition $P$ in an optimum solution $(P,H)$ to Problem 2 *degenerated*, if one of the vertex groups in $P$ contains the minimum number of vertices demanded by the model. In our experiments, this minimum group size equals 1. A solution is hence degenerated if there is a vertex group which contains a single vertex. If only degenerated solutions are found for a given image matrix with $c$ groups, it is likely that the market has a link pattern with $c-1$ groups which is at least as good as the current pattern. In this case, the only reason for one vertex to be placed in the $c$-th group is probably the minimum group size constraint. The procedure DEGENERATION($I,P$) thus removes all minimum size groups from the image graph.

In the following two subsections, we present two algorithms for the analysis of the COMTRADE data set. In the single arc type approach, the data are converted into a digraph $D=(V,A)$. In the multi arc type approach, we distinguish arcs of different weight. The data is hence converted into a multi-graph $D=(V,A_1,\dots,A_r)$, where $A_0$ contains the lightest and $A_r$ the heaviest arcs.

### 5.2.1 A Single Arc Type Approach

In the single arc type approach, we create and analyze a digraph $D=(V,A)$ representing the world trade network. Output of the analysis algorithm is a blockmodel $(P,I)$, consisting of a partition $P$ of the countries and a trade pattern encoded as an image matrix $I$.

**Trade Digraph.** The complete COMTRADE network, including all countries and all trade relations, has an arc density of 25.6%. Since it is impossible to find a non-trivial link pattern in such a dense network, a preprocessing step is required in this approach. This step is called *dichotomization*, as it decides for each arc with continuous trade volume whether there exists a substantial amount of trade or not.
First, we limit the number of vertices to the 50 most-trading countries. To measure the amount of trade for each country, we compute the maximum of the value of all exports and the value of all imports. Second, we limit the number of arcs to 400. That is, we delete all but the arcs with the 400 largest trade volumes from the network. Note that the remaining arcs still carry more than

80% of the total trade volume. As this choice is however arbitrary, we verify its suitability in a post-processing step to be explained later.

**Trade Analysis Algorithm.** The analysis algorithm for $D = (V,A)$ follows a 2-step procedure proposed by Brusco and Steinley [BS09]. In the first step, a heuristic is used to find several image matrices $I_1,\dots,I_k$ of high quality for the given digraph. Since a heuristic is being used, the penalty values $p_{heu}(I_i)$ might not be equal to the real quality values $p(I_i)$. In fact, $p_{heu}(I_i) \geq p(I_i)$ holds. Hence, in the second step, the exact values $p(I_i)$ are computed for all proposed image matrices $I_i$. This computation is NP-hard as shown in Section 4.1.2. We solve it with the branch-and-cut algorithm developed in Chapter 4. Not only does it compute the penalty value $p(I_i)$, but also a partition $P_i$ of the countries such that the blockmodel $(P_i,I_i)$ takes the penalty value $p(I_i)$.

The input to the analysis algorithm is a digraph $D = (V,A)$ and a maximum number $c_{max}$ of vertex groups to be searched for.

**Algorithm 6:** Trade Analysis (Single Arc)

```
Data: Digraph $D = (V,A)$, maximum number $c_{max}$ of vertex groups, number $k$ of heuristic runs
Result: Blockmodel $(P,I)$
1  Set $p_{best} = \infty$.
2  Set imagelist$_2 = \cdots =$ imagelist$_{c_{max}} = \{\}$.
3  for $c = c_{max},\dots,2$ do
4      for $i = 1,\dots,k$ do
5          Obtain blockmodel $(P,I)$ by applying the Kernighan-Lin Algorithm 3 to Problem 1 with $c$ groups.
6          Set $I =$ REDUNDANCE$(I)$.
7          Add $I$ to imagelist$_{dim(I)}$.
8  for $c = c_{max},\dots,2$ do
9      for $I \in imagelist_c$ do
10         Compute exact optima to Problem 2 for $I$ by branch-and-cut (Section 4.6).
11         if all optima degenerated then
12             Add $I' =$ DEGENERATION$(I,P)$ to imagelist$_{dim(I')}$, where $(P,H)$ is an arbitrary solution.
13         else
14             Set $(P,H)$ to an non-degenerated solution.
15             if $p(P,I) < p_{best}$ then
16                 $p_{best} = p(P,I)$, $P_{best} = P$, $I_{best} = I$.
17 return $(P_{best},I_{best})$
```

In Algorithm 6, Lines 3 to 7 correspond to the first step of the algorithm. For every number of vertex groups within a meaningful range, image matrices are determined by the Kernighan-Lin heuristic. In the second step (Lines 8 to 16), the exact quality of these candidate image matrices is computed. Note that in both steps, the found image matrices are tested for redundancy and degeneration. Afterwards, they are possibly replaced by other matrices of lower dimension.

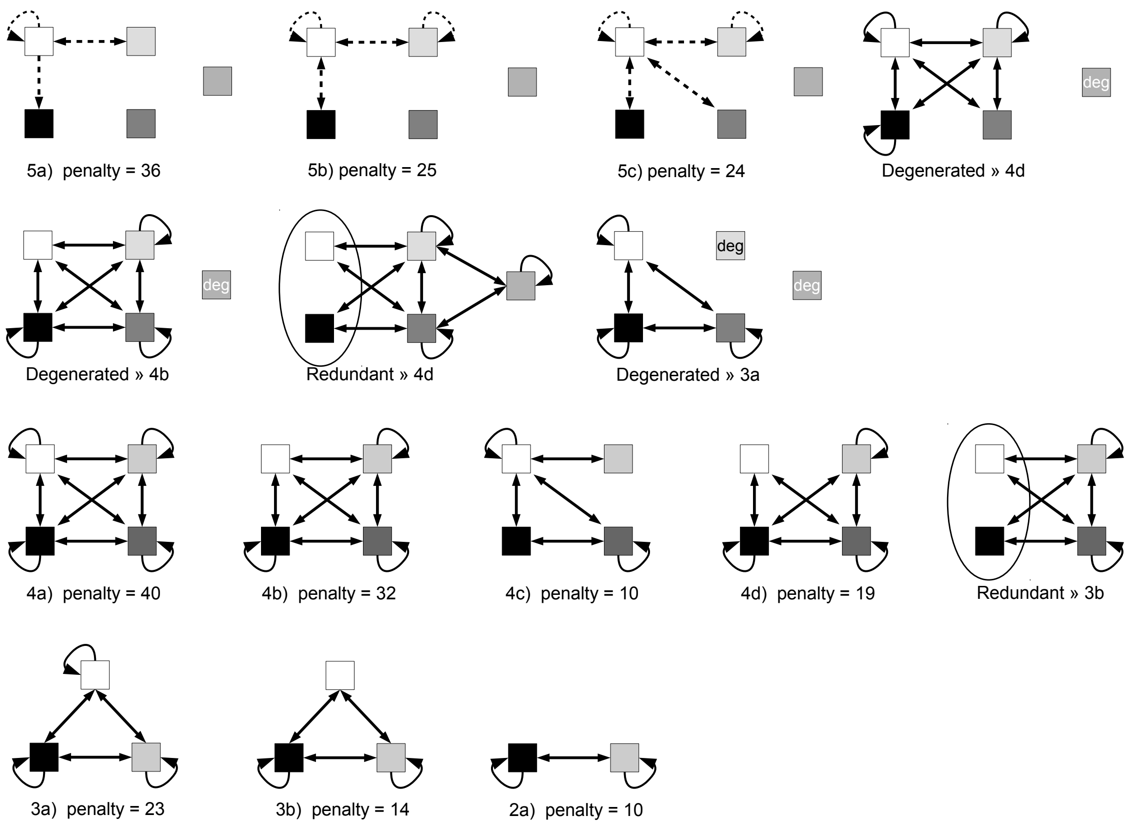
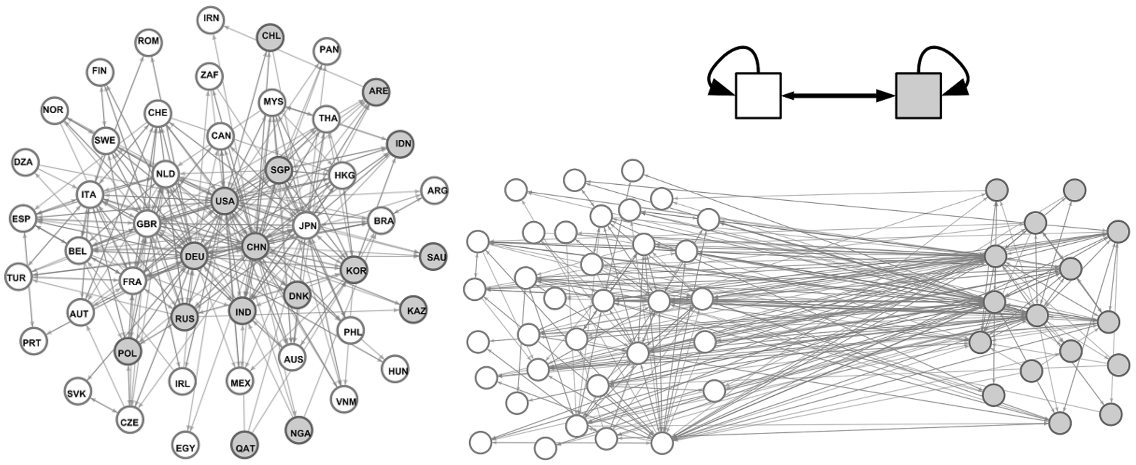


Figure 5.2: Image graphs found by Algorithm 6 in Line 5. Either the existing (bold) or the non-existing arcs (dashed) are given. The value *penalty* gives the objective value computed in Line 10. Best image graphs are 4c and 2a with a penalty of 10.

**Results.** Figure 5.2 shows the image graphs found by the Kernighan-Lin heuristic in Algorithm 6 for maximally $c_{max} = 5$ vertex groups and $k = 10$ runs of the heuristic for each fixed number of groups. Below each image graph, the exact penalty value computed by the exact branch-and-cut algorithm in Line 10 is given. In case that the image graph is redundant or degenerated, the ID number of the simplified image graph is given instead.
We see that there are two image graphs with the best penalty value of 10 respectively: Graphs 2a and 4c. The penalty value of 10 means that 10 out of 2450 entries (0.4%) in the adjacency matrix of the trade digraph need to be changed in order to perfectly obtain the trade structure of depicted image graphs.

> *At most* 0.4% *of the world trade network's adjacency matrix needs to be modified to obtain the trade pattern depicted in Figure 5.4.*



| Stability of the image graph | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # arcs | 300 | 320 | 340 | 360 | 380 | **400** | 420 | 440 | 460 | 480 | 500 |
| penalty | 18 | 15 | 14 | 12 | 10 | **10** | 9 | 8 | 8 | 8 | 7 |

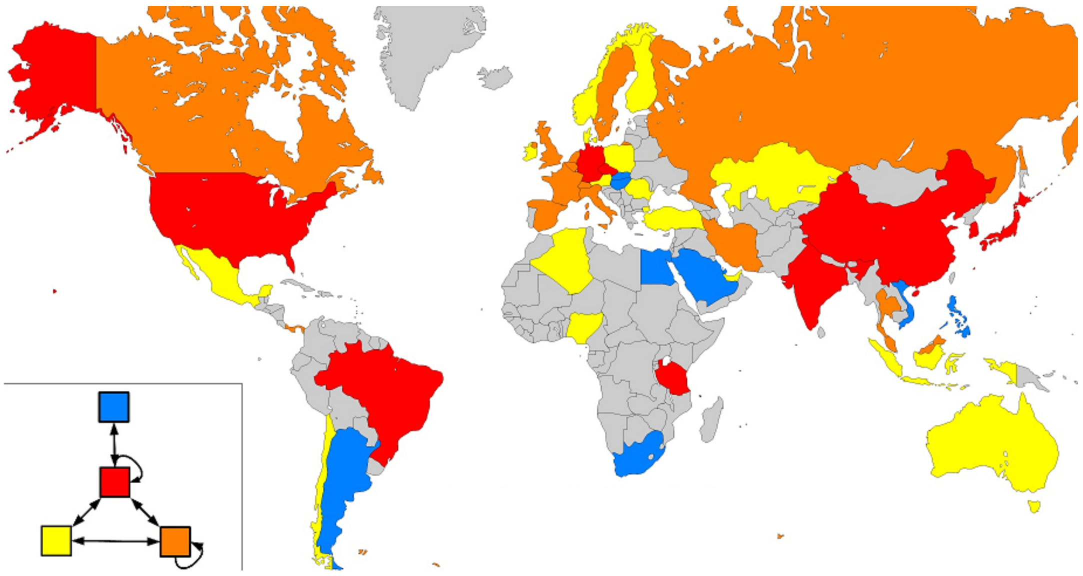
Figure 5.3: Optimum partition of the 50 most trading countries into 2 groups. The country names are given in the Appendix on Page 112.

The corresponding country partitions are depicted in Figures 5.4 and 5.3 respectively.

**Discussion.** As the number 400 of trade arcs under consideration was chosen in a suitable magnitude, but arbitrarily, the tables below Figures 5.3 and 5.4 prove the stability of the our solutions a posteriori. If the number of arcs is lowered to 300 or raised to 500, the number of changed adjacency entries stays within the range of $[0.36\%, 0.66\%]$ for the 4 groups solution and $[0.28\%, 0.74\%]$ for the 2 groups solution.

We venture a brief interpretation of the 4 groups solution in Figure 5.4. There is a core group (center) of 9 countries which trade among each other and with all other groups, concerning both imports and exports. Furthermore, there is a group of 10 periphery countries (top) which are all specialized on trade with the center group. Hence, they trade neither with other groups nor among each other. The large left and right groups trade both with the center and with each other. The difference between the two lies in the observation that the right group additionally trades within itself.

The interpretation of the 2 groups solution (Figure 5.3) is as follows: It is possible to divide the countries into two groups such that every country has a trade partner in both its own and in the other group. Note that this observation is problematic for two reasons. First, it holds for any partition of the complete graph and often for any partition of dense graphs. The optimality of this image graph might hence indicate that the choice of 400 arcs is too large for 2 vertex groups only. Second, the solution is unstable from the vertex perspective: Every vertex could change its



| Stability of the image graph | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # arcs | 300 | 320 | 340 | 360 | 380 | **400** | 420 | 440 | 460 | 480 | 500 |
| penalty | 16 | 15 | 14 | 11 | 10 | **10** | 11 | 11 | 9 | 10 | 9 |

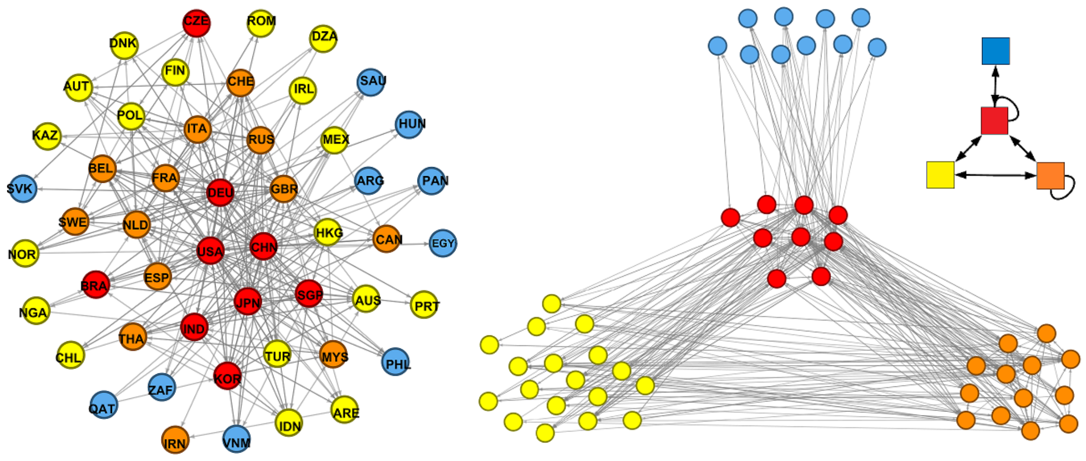


Figure 5.4: Partition of the 50 most trading countries into 4 groups (bottom left). If 0.4% of the trade adjacency matrix is changed, the depicted trade pattern (bottom right) is obtained. Country names are given in the Appendix on Page 112 and are visualized in the world map (top).

group without violating the property that every vertex has a trade partner both in its own and in the other group. We thus prefer the 4 groups solution as the result of the analysis. Still, we do not consider the image graph in Figure 5.3 being logically *redundant*, as its interpretation ("a trade partner exists in both the own and the other group") is logically stronger than the interpretation of the simplified graph consisting of a single vertex with self-loop ("there exists one trade partner").

### 5.2.2 **A Multiple Arc Type Approach**

As the world trade digraph is dense, it hardly contains any link pattern. The only exceptions are the trivial patterns, in which all country groups trade with each other. In the previous subsection, the single arc type approach addressed this problem by considering only the $m$ heaviest trade arcs. In this subsection, we present an alternate approach which considers all trade arcs. To this end, the arcs are divided into $r$ weight classes. For $r = 3$, these classes could be labeled *low, medium,* and *high trade volume*. The output of the analysis algorithm is then a *multi-relation* blockmodel as introduced on Page 8. It hence consists of one *single* partition of the countries and $r$ trade patterns (image graphs); one for each weight class. To our knowledge, we are the first authors to compute multi-arc blockmodels for trade networks. We now show how Problems 1 and 2 can be formulated for multi-relation graphs and how the solution algorithms need to be adjusted. Afterwards, we show how the arc weight classes are computed.

**Multi-Relation Problems 1 and 2.** In Section 4.1, we introduced Problem 1: For a given digraph, find a pattern (image matrix $I$) such that the arc set of $D$ needs to be changed to the lowest possible amount in order to meet the pattern. The problem was formulated for digraphs with a single arc set. Given a multi-graph $D = (V, A_1, \ldots, A_r)$ with several arc sets $A_i$, the definition can be easily adjusted. To this end, define $D_i := (V, A_i)$ the digraph which only contains the arcs of type $i$.

**Problem 1-multi.** Given a multi-digraph $D = (V, A_1, \ldots, A_r)$ and a number $c \geq 1$ of vertex groups, solve

$$\min_{P \in \mathbf{P}_c(D)} \sum_{i=1}^{r} \left( \min_{\substack{I_i, H_i: \\ (H_i, P) \in REG(D_i, I_i)}} \sum_{u,v \in V, u \neq v} |Adj(D_i)_{u,v} - Adj(H_i)_{u,v}| \right). \tag{5.1}$$

Similarly, Problem 2-multi can be defined by fixing the image matrices $I_1, \ldots, I_r$ in the inner minimization problem.

**Multi-Arc Algorithms.** The adaption of the Kernighan-Lin heuristic to the multi-arc case is straight-forward. For every vertex move or swap operation, the error update needs to be computed for all $r$ arc types successively, but independently. The total running time is hence $r$-times as large as in the single arc case. The adaption of the integer programming formulation is unproblematic as well. The variables of types $\alpha$, $\beta$, and $p$ need to be copied $r$ times; once for each arc type. They are hence provided with an additional index $i$ indicating the arc type: $\alpha_{i,v,AB}$, $\beta_{i,v,AB}$, $p_{i,AB}$. Analogously, some constraints need to be copied $r$-times. The $\alpha$- and $\beta$-defining constraints from Section 3.4 need to be stated for all new $\alpha$ and $\beta$ variables. Furthermore, $p$ defining constraints (4.14) and (4.15) needs to be stated $r$-times. That is, all of the constraints and separation routines derived in Chapters 3 and 4 can be used in the multi-arc case as well. In fact, it is not necessary to link the copies of a variable to each other or to add any further constraints beyond those. All in all, the resulting integer programming model can be stated as follows.

**Model 2-multi.**

$$\min \sum_{i=1}^{r} \left( \sum_{(u,v)\in A_i} \sum_{\substack{a,b\in[c] \\ I_i(a,b)=0}} B_{ab} x_{ua} x_{vb} + \sum_{\substack{a,b\in[c] \\ I_i(a,b)=1}} B_{ab} p_{i,ab} \right)$$

$$\begin{aligned}
\sum_{a\in[c]} x_{ua} &= 1 && \text{for } u \in V, \\
\sum_{u\in V} x_{ua} &\geq 1 && \text{for } a \in [c], \\
x_{ua} + \sum_{v\in N_i^+(u)} (1 - x_{vb}) &\leq \alpha_{i,u,ab} + |N_i^+(u)| && \text{for } i \in [r], u \in V, a, b \in [c] \text{ with } I_i(a,b) = 1, \\
x_{ub} + \sum_{v\in N_i^-(u)} (1 - x_{va}) &\leq \beta_{i,u,ab} + |N_i^-(u)| && \text{for } i \in [r], u \in V, a, b \in [c] \text{ with } I_i(a,b) = 1, \\
p_{i,ab} &\geq \sum_{u\in V} \alpha_{i,u,ab} && \text{for } i \in [r], a, b \in [c] \text{ with } I_i(a,b) = 1, \\
p_{i,ab} &\geq \sum_{u\in V} \beta_{i,u,ab} && \text{for } i \in [r], a, b \in [c] \text{ with } I_i(a,b) = 1, \\
x_{ua}, \alpha_{i,u,ab}, \beta_{i,u,ab} \text{ binary}, &\ p_{i,ab} \text{ integer} && \text{for} \ldots \text{(see above).}
\end{aligned}$$

**Trade Multi-Digraph.** We use the complete trade digraph with all 229 vertices and all 13398 arcs (after preprocessing). In a multi-arc approach, all arcs in the trade network are considered. They are however partitioned into $r$ non-overlapping sets according to their weight. The network is hence represented by a multi-digraph $D = (V, A_1, \ldots, A_r)$, containing arc sets from light trade volume arcs $A_1$ to heavy weight trade volume arcs $A_r$.

The choice of equally sized weight intervals would classify 99% of the arcs as $A_1$, as Figure 5.2.2 shows that the weight distribution is at least exponential. We thus choose equally sized intervals on the logarithmically scaled axis. Figure 5.2.2 shows the arc classification for $r = 3$ arc types. To describe the classification mathematically, let $a_{min}$ and $a_{max}$ denote the minimum and maximum trade value in the trade network. Define $l := \ln a_{min}$ and $u := \ln a_{max}$. The step length $s$ is computed as $(u - l)/r$. We then use the following weight intervals for classification:

$$\begin{aligned}
\text{Class } A_1 :& [e^l, e^{l+s}) \\
\text{Class } A_2 :& [e^{l+s}, e^{l+2s}) \\
\text{Class } A_3 :& [e^{l+2s}, e^{l+3s}) \\
&\vdots \\
\text{Class } A_r :& [e^{l+(r-1)s}, e^{l+rs}).
\end{aligned}$$

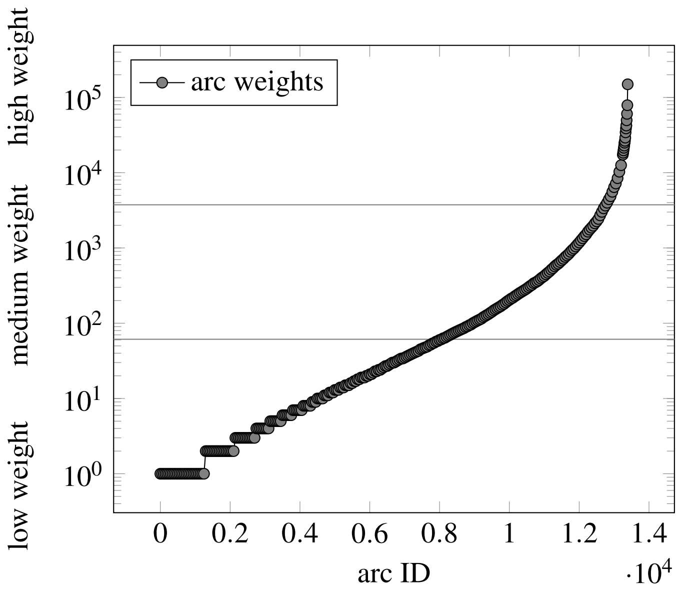


Figure 5.5: Weight distribution of the arcs in the complete trade network.

**Trade Analysis Algorithm.** Algorithm 7, the multi arc trade analysis algorithm, is similar to the single-arc algorithm. As additional input parameters, it has bounds $r_{min}$ and $r_{max}$ on the number $r$ of arc types. Furthermore, all digraphs and blockmodels are of the multi type. Note the following differences to the single arc case. The exact optimization in Line 10 has to be replaced by the Kernighan-Lin heuristic for Problem 2 (see Section 4.4), because of the large size of the digraph. Furthermore, the REDUNDANCY routine is removed, since this effect is unlikely in the case of multiple arc sets. The Kernighan-Lin heuristic tries to find solutions with low penalty value. As the maximal possible penalty value grows linearly in the number $r$ of arc types, we compensate this effect by a normalization in Lines 14 and 15. This way, we can compare the quality of solutions with different $r$-values.

**Results.** We run Algorithm 7 with the following parameters: $r_{min} = 3$, $r_{max} = 5$, and $k = 10$. The algorithm is executed two times, for $c = 4$ and $c = 6$ country groups respectively.
The optimum solution for $c = 4$ country groups uses $r = 4$ arc types. The optimum image graphs for *high*, *upper-medium*, *lower-medium*, and *low trade* are depicted in Figure 5.6. The penalty value of this solution is at most 307. This means that at most 0.14% of all possible adjacency matrix entries need to be changed in the world trade network in order to perfectly obtain the depicted trade structure. The member countries for each group are given in a table on Page 111.
The optimum solution for $c = 6$ country groups uses $r = 3$ arc types. The optimum image graphs for *high*, *medium*, and *low trade* are depicted in Figure 5.7. Its penalty value is at most 251 (0.16%). The member countries for each group are given in a table on Page 113.

*At most* 0.16% *of the world trade network's adjacency matrix needs to be modified to obtain the trade pattern depicted in Figure 5.7.*

**Algorithm 7:** Trade Analysis (Multi Arc)

```
Data: Multi-digraph $D = (V,A)$, bounds $r_{min}$ and $r_{max}$ for the number of arc types,
      number $c$ of vertex groups, number $k$ of heuristic runs
Result: Multi-blockmodel $(P, I_1, \dots, I_r)$
1  Set $p_{best} = \infty$. Set $D_{r_{min}} = \dots = D_{r_{max}} = \{\}$. Set imagelist= $\{\}$.
2  for $r = r_{min}, \dots, r_{max}$ do
3      Set $D_r = (V, A_1, \dots, A_r)$, where $(A_1, \dots, A_r)$ is a partition of $A$ into $r$ weight classes.
4      for $i = 1, \dots, k$ do
5          Obtain multi-blockmodel $(P, I_1, \dots, I_r)$ by applying the Kernighan-Lin Algorithm 3
           to Problem 1 with $D_r$ and $c$.
6          Add $(I_1, \dots, I_r)$ to imagelist.
7  for $I = (I_1, \dots, I_r) \in$ imagelist do
8      for $i = 1, \dots, k$ do
9          Apply Kernighan-Lin Algorithm to Problem 2 with $I$, $D_r$ and $c$.
10     if all optima degenerated then
11         Add $I' = $ DEGENERATION$(I, P)$ to imagelist, where $(P, H)$ is an arbitrary solution.
12     else
13         Set $(P, H)$ to a non-degenerated solution.
14         if $p(P, I)/r < p_{best}$ then
15             $p_{best} = p(P, I)/r$, $P_{best} = P$, $I_{best} = I$.
16 return $(P_{best}, I_{best})$
```

**Discussion.** We briefly interpret the 4 groups solution in Figure 5.6. On a low trade level, we see that the four groups almost completely trade with each other. The only exception is group $D$, which imports from, but does not export to other countries. On the next higher level, we see that the imports of group $D$ originating in group $C$ vanish. On the high trade level, we see that only the members of group $A$ trade with each other. The members of this group moreover export to all other groups on a upper-medium trade level. All of the import is provided by group $B$. All in all, we observe a trade power hierarchy of the groups $A$,$B$,$C$, and $D$ in precisely this order.

We now turn to the 6 groups solution in Figure 5.7 on Page 94. On the low trade level, we observe again almost complete trade relations. The only exceptions are the black and white groups (top center, bottom center), which both do not export and thus do not trade with each other. The high trade level view identifies the blue group (bottom right) as the main core group, having strong relations to the second core group yellow (bottom left). The three groups on the top are periphery to the core groups (see medium trade level), that is, they do not trade with each other. They can however be distinguished by their different relations to the core: red (right) has complete relations to the core and trades among itself, orange (left) has complete relations to the core and does not trade among itself. The black group (center) only imports from the core, but does not export at any level.

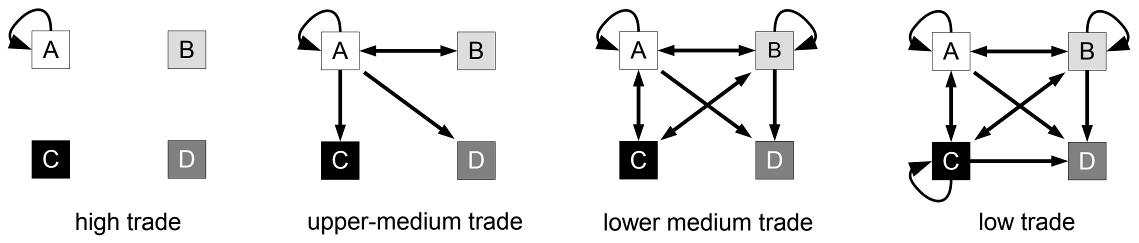


Figure 5.6: The world trade network has the above structure if 0.14% of its adjacency matrix entries are changed.

## 5.3 CLASSIFICATION OF TRADE NETWORKS

In the previous section, we were given a single trade network and searched for patterns to describe it. In this section, we are given several trade networks; one for each group of products. The problem is to find a small collection of patterns such that every network is well-described by one of those patterns. That is, we are looking for the most common trade market structures. This assignment $\phi$ is visualized in Figure 5.8 (left). Before we introduce the complete analysis algorithm, we first explain how a representative subset is chosen from a list of patterns.

**Choosing Representative Patterns.** Suppose we are given $n$ trade digraphs and $m$ image matrices. For each pair $(i, j)$ of a trade digraph $i$ and an image matrix $j$, we can compute the penalty $M_{ij}$ as the optimum value to Problem 2. That is, $M_{ij}$ states how many adjacency matrix entries of $i$ need to be changed in order to obtain the pattern represented by $j$. As this value can be computed for all pairs $(i, j)$, we obtain an $n \times m$ matrix $M$ containing them. For a given number $k$, we define the $k$ image matrices best representing the $n$ trade networks as the solution to the following optimization problem:

**Problem 3.** Find an index set $S \subseteq \{1, \dots, m\}$ with $|S| \leq k$ and a mapping $\phi : [n] \to S$, such that $\max\{M_{\phi(i)j} \mid i = 1, \dots, n, j \in S\}$ is minimized.

The solution $(S^*, \phi^*)$ with value $s^*$ thus allows a statement of the following form: In each network, at most $s^*$ adjacency matrix entries need to be changed such that its pattern is included in $S^*$.
The problem can be modeled as an integer program in the following way. Using the software CPLEX 12.6, it could be solved in less than one second in our practical experiments. The program uses two kinds of binary variables:

$$s_j = \begin{cases} 1 & \text{if index } j \text{ is in } S, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if network } i \text{ is assigned to pattern } j, \\ 0 & \text{otherwise,} \end{cases}$$

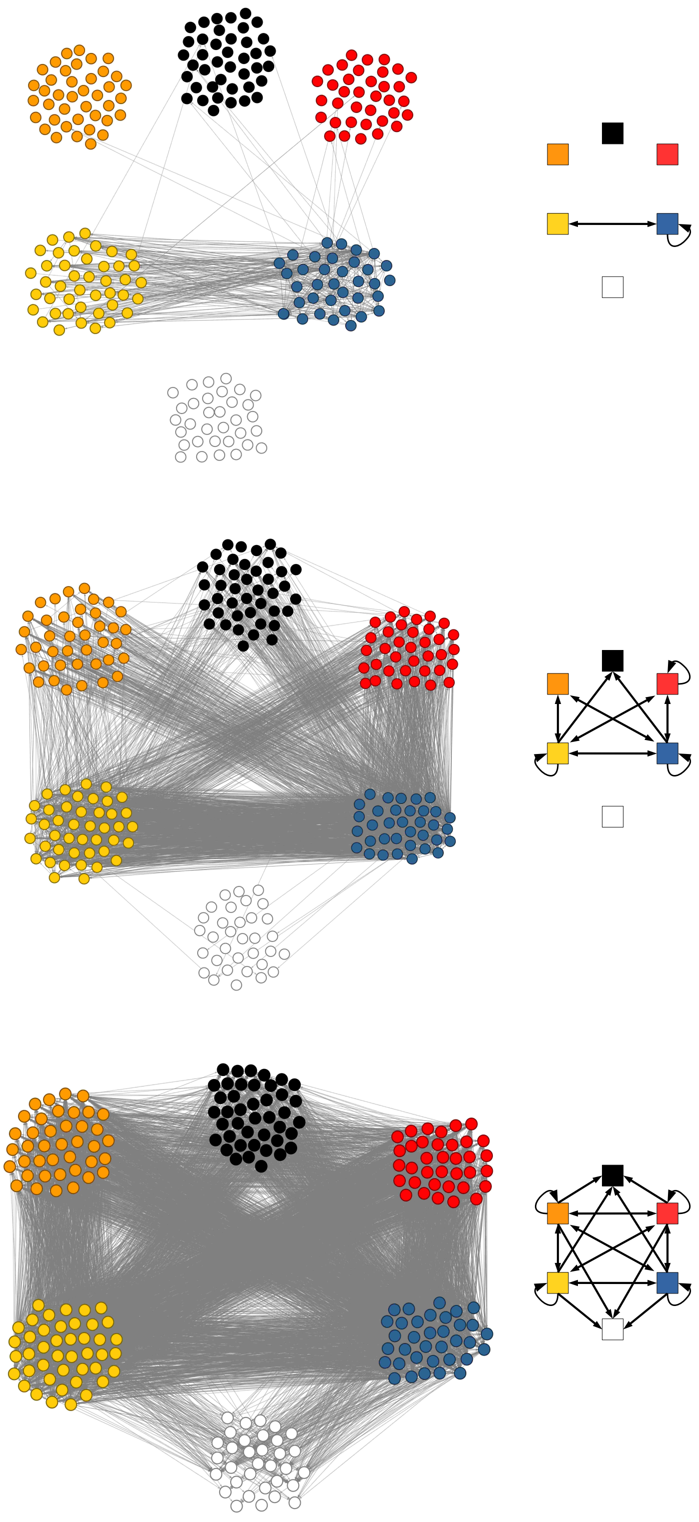


Figure 5.7: Optimum solution for 6 country groups and high (top), medium (center), and low (bottom) trade volume. 0.16% of the world trade network's adjacency matrix entries need to be changed to perfectly obtain the depicted structures. In the top figure, we can see some vertical arcs causing a penalty.

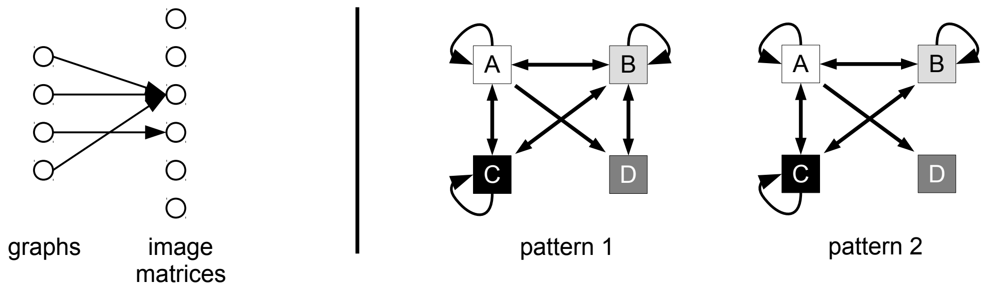


Figure 5.8: Left: Assignment of trade graphs to image matrices. Right: Two patterns which were frequently found in trade digraphs.

**ASSIGN**(*M*,*k*)

$$\min \quad L$$

$$\begin{aligned}
M_{ij}x_{ij} &\leq L \quad \text{for } i = 1,\ldots,n, j1,\ldots,m \\
\sum_{j=1}^{m} s_j &\leq k \\
\sum_{j=1}^{m} x_{ij} &= 1 \quad \text{for } i = 1,\ldots,n \\
x_{ij} &\leq s_j \quad \text{for } i = 1,\ldots,n, j = 1,\ldots,m \\
s_i, x_{ij} &\in \{0,1\} \quad \text{for } i = 1,\ldots,n, j = 1,\ldots,m
\end{aligned}$$

**Classification Algorithm.** The analysis of the $n$ trade networks for different product groups is performed as described in Algorithm 8 below. It consists of four steps. In the first step, the data is normalized and dichotomized. That is, the arc-weighted digraphs are transformed into unweighted digraphs by keeping only the $nV$ most-trading vertices and the $nA$ largest arcs. In the second step, a list **I** of good image matrices is collected. To this end, the $k$ best image matrices are stored in **I** for all networks respectively. Recall that two image matrices are *isomorphic* if their image graphs are identical if their vertices are renamed. Before we add a new image matrix to **I**, we therefore check whether **I** already contains an image matrix isomorphic to it. In the third step, we compute how well the collected image matrices describe the networks. We compute the corresponding penalty value $M_{ij}$ for *all* pairs of networks $i$ and collected image matrices $j$. In the last step, the networks are assigned to the image matrices as described in the preceding paragraph.

**Data.** The network data is again obtained from the COMTRADE database. All settings and references are identical to the ones stated in the previous section. In this section, however, we did not request data on the total trade ("all commodities"), but for the 9 major trade sectors according to the SITC Rev. 3 classification. These sectors are named in the following table. We hence obtained 9 arc-weighted digraphs.

**Algorithm 8:** Trade Classification

```
Data: Set $\mathbf{D} = (D_1, \ldots, D_n)$ of arc-weighted digraphs, numbers $nV$ and $nA$ of vertices and
      arcs, number $k$ of candidate search runs per graph, number $c$ of vertex groups
Result: Set $\mathbf{I}$ of image matrices, assignment $\phi : D \to I$

2  /* preprocess digraphs */
3  for $i = 1, \ldots, n$ do
4      Delete all but the $nV$ most-trading vertices from $D_i$.
5      Delete all but the $nA$ largest arcs from $D_i$.

7  /* compute candidate image matrices */
8  Set $\mathbf{I}$ to an empty list.
9  for $i = 1, \ldots, n$ do
10     for $j = 1, \ldots, k$ do
11         Obtain blockmodel $(P, I)$ by applying the Kernighan-Lin Algorithm 3 to Problem 1
           with $D_i$ and $c$.
12         if I is non-isomorphic to all elements of $\mathbf{I}$ then
13             Add $I$ to $\mathbf{I}$.

15 /* compute assignment matrix */
16 Initialize $M$ as an $n \times |I|$ integer matrix.
17 for $d = 1, \ldots, n$ do
18     for $i = 1, \ldots, |\mathbf{I}|$ do
19         Set $M(d, i)$ to the optimum value to Problem 2 with inputs $D_d$ and $\mathbf{I}_i$ by
           branch-and-cut (Section 4.6).
20         if All optima degenerated then
21             Set $M(d, i) = \infty$.

23 /* compute assignments */
24 for $i = 1, \ldots, |\mathbf{I}|$ do
25     Set $\phi_i$ to the optimum assignment by solving ASSIGN($M$,$i$) with CPLEX.

27 return $(\phi_1, \ldots, \phi_{|\mathbf{I}|})$
```

| Sector ID | Description |
|---|---|
| 0 | Food and live animals |
| 1 | Beverages and tobacco |
| 2 | Crude materials, inedible, except fuels |
| 3 | Mineral fuels, lubricants and related materials |
| 4 | Animal and vegetable oils, fats and waxes |
| 5 | Chemicals and related products, n.e.s. |
| 6 | Manufactured goods classified chiefly by material |
| 7 | Machinery and transport equipment |
| 8 | Miscellaneous manufactured articles |

**Settings and Results.** We apply Algorithm 6 to the $n = 9$ sector trade digraphs. We choose the parameters $nV = 50$, $nA = 400$, $k = 10$, $c = 4$ and comment on their stability in the next paragraph. The solution time for the exact optimization in Line 19 is limited to 1000 seconds. In case that the limit is reached, the best known primal solution is returned. The total running time is dominated by the exact optimization and took 3 hours of CPU time on the computer described in Section 4.2. The algorithm finds 8 candidate image matrices for the 9 networks. The penalty values $M_{ij}$ are given by the following matrix.

$$M = \begin{pmatrix} 29 & 26 & 23 & 29 & 40 & 68 & 53 & 49 \\ 19 & 29 & 24 & 23 & 43 & 72 & 54 & 51 \\ 22 & 22 & 24 & 26 & 30 & 49 & 39 & 37 \\ 29 & 19 & 22 & 33 & \infty & 43 & 31 & 30 \\ 30 & 17 & 19 & 34 & 24 & 46 & 33 & 32 \\ 34 & 27 & 29 & 40 & \infty & 40 & 34 & 32 \\ 28 & 23 & 25 & 29 & 31 & 59 & 45 & 42 \\ 23 & 23 & 25 & 23 & \infty & 59 & 44 & 41 \\ 19 & 22 & 26 & 20 & 41 & 73 & 55 & 51 \end{pmatrix}$$

If only one image matrix can be chosen, there are two optimum solutions. The two optimum image graphs are depicted in Figure 5.8 (right). For both of the two image graphs holds: If each out of the 9 digraphs is changed by at most 29 adjacency matrix entries (1.2%) respectively, all digraphs perfectly have the structure of the depicted image graph.

If the number of image graphs which can be chosen lies between 2 and 8, there is an optimum solution which uses 2 image matrices only. It has an even better optimum value of 27 (1.1%). The statement is as follows.

*If each out of the* 9 *digraphs is changed by at most* 1.1% *respectively, they perfectly have one of the two patterns depicted in Figure 5.8.*

**Verification.** We show that the two patterns in Figure 5.8 cannot be found in a *random* collection of graphs with 50 vertices and 400 arcs. To this end, we computed 9 random digraphs, where each possible arc had the same probability to be one of the 400 created arcs. We then solve Problem 3 for these digraphs and the two image graphs in Figure 5.8. For the random graphs, we

| # arcs | 300 | 350 | 400 | 450 | 500 |
|---|---|---|---|---|---|
| Max. adj. changes | 1.5% | 1.3% | 1.1% | 1.1% | 1.0% |

Table 5.1: The solution quality remains if the number of arcs is slightly increased or decreased.

need to change at least 6.2% of the adjacency matrix entries to obtain the shown patterns. For the trade graphs, at most 1.1% need to be changed.

We finally try to justify the choice of the parameters and the quality of the solution a posteriori. The choice of $k = 10$ candidate images per digraph seems to be sufficient, since out of 90 candidates found, only a subset of 8 candidates was pairwisely non-isomorphic. The choice of $c = 4$ vertex groups is not too large, since otherwise one would expect a larger number of degenerate solutions. One could however compute solutions with more than 4 groups in future research. The choice of exactly $nA = 400$ arcs was arbitrary, but from a range allowing for non-trivial patterns (no complete and no empty trade pattern). To show that our 2-pattern solution in Figure 5.8 is non-sensitive to changes in $nA$, we computed its value to Problem 3 from $nA = 300$ to $nA = 500$. The results are given in Table 5.1.

# Chapter 6

# DISCUSSION AND FUTURE RESEARCH

In this chapter, we briefly discuss the obtained results with a strong emphasis on open questions and future research topics.

## 6.1 CLASSIFICATION OF APPROACHES

In Chapter 2, we presented a classification for clustering and blockmodeling approaches and showed that these approaches are based on relaxations of graph theoretical partitioning definitions. Basically, there are only three types of relaxations: Single node, node pair, and subgraph relaxations. The classification unifies link density pattern (including clustering) and link existence pattern approaches and shows the connections between them.

A drawback of a theory about approaches in literature is clearly its incompleteness as soon as new kinds of approaches are developed. Furthermore, it does not yet cover approaches in which group size restrictions are not posed as constraints, but are included in the penalty function $p$. An example is the *conductance approach* for clusterings. To classify this approach, the requirement for same group sizes needs to be added to the ideality definitions, such that a deviation can be penalized. We did not include it as most approaches include size restrictions into the constraint set. However, we also see two kinds of practical benefits to our theory:

**Design Of New Approaches.** First, the classification allows to generate new approaches by a series of design choices, which include the relaxation type, the choice between ideal and average graphs, the combination formula for the penalties, and so on. Not all approaches which can be generated are actually considered in literature. For example, approaches which use average graphs usually compare $G$ to a single average graph $H$, whose edge weights are fractional. This choice seems to be arbitrary, as one could also use a whole set $\mathbf{H}_{P,A,B}$ of unweighted average graphs for the comparison to $G$. The classification can hence be used to generate and examine new approaches. Moreover, it can provide reasons for the non-existence of possible approaches.

**Methodological Guidance.** The classification provides a guidance in methodological questions. Given a network, the analyst needs to choose from a variety of pattern analyzation approaches. In literature, the choice of the penalty function is most often reasoned by properties of the function itself, not by its suitability for the given network type and the research question posed. However, we can derive network-specific rules from the classification:

- The question whether vertices or edges should be relaxed depends on the input data. For example, consider a trade network: the existence of the nodes (countries) is not misled by measuring errors, whereas the link weights (trade amounts) might be errorenous due to incomplete or inaccurate reports. Approaches which relax edges are stable against these deviations and should thus be preferred.

- Maximizing the distance to average graphs is preferrable in case that it is import to show that the computed link pattern is not a random phenomenon. Minimizing the distance to ideal graphs is the better choice if an almost ideal pattern is assumed to exist in the network.

- Pairwise relaxations should only be used if patterns of link *quantity* are searched for and if the partition of the vertices is more important than the pattern itself. For patterns of link quantity, the pairwise Definition 3 states that two vertices in the same group have exactly the same neighbor vertices. In the pairwise relaxation, it is hence possible to quantify the amount of identical neighbors by a constant value $c_{uv}$ for every pair $\{u,v\}$ of vertices. For patterns of link *existence*, however, the pairwise definition states that two vertices in the same group have neighbors in the same groups. This value is however non-constant, as it depends on the partition of the vertices. Still, many authors compute the similarities $c_{uv}$ for link existence patterns as well. For example, the earlier the partition algortihm REGE decides that $u$ and $v$ should belong to distinct groups, the larger the similarity value $c_{uv}$ is set. That is, for every pair $\{u,v\}$, the similarity value $c_{uv}$ is computed with respect to a different grouping. When the values $c_{uv}$ are finally clustered into groups and the link pattern is obtained by a rounding procedure, the pattern is hence not guaranteed to be good by any measurement. We therefore recommend to use the pairwise relaxations only for patterns of link quantity.

Using the classification, the decision-making process can be performed stepwise: Are ideal or average graphs more suitable, should edges or vertices be relaxed, should node pairs or subgraphs be relaxed, how should subgraph penalties be combined, and so on.

## 6.2 ALGORITHMIC IMPROVEMENTS

In Chapter 4, we developed an exact algorithm for the solution of Problem 2: Given a digraph and a link existence pattern, compute how well the pattern describes the network. The resulting branch-and-cut algorithm turned out to be 100 to 10,000 times faster than comparable approaches from the literature. The greatest portion of the speedup is due to two major reasons: First, the dimension of the solution space could be significantly reduced. Second, the primal Kernighan-Lin heuristic turned out to produces near-optimum solutions on our test instances. However, further research should consider the dual bound. Even though the dual bound given by the optimum to the LP-relaxation could be improved by our dual heuristic, the resulting gap between primal and dual bound remains large for many test instances. Further standard approaches such as the Lagrangian relaxation failed in our tests and are thus not reported in this thesis. A further research direction lies in the application of non-linear integer programming techniques. Instead of using linearizations, which increase the dimension of the solution space to a large extent, the problem could be solved directly. However, as the integer program is non-binary and the objective function is non-convex in general, current direct approaches are still slower by several orders of magnitude.

## 6.3 WORLD TRADE ANALYSIS

In Chapter 5, we analyzed a world trade network provided by the United Nations. We found patterns of link existence which describe the network well in an absolute sense. That is, the networks need to be adjusted by a small amount ($< 1\%$) to perfectly match the patterns. Still, the analysis methods were heuristical: In a first step, a collection of good candidate patterns was computed. In the second step, their quality was exactly computed. Even though the heuristic was shown to produce near-optimal solutions on our test instances in Section 4.4, it will likely miss good candidate patterns in the first step. We thus do not know whether the absolutely good solutions are also relatively good with respect to other solutions. Future research should focus on the exact solution of Problem 1 in a single step. To this end, a branch-and-cut algorithm based on the integer program *Model 1* on Page 55 could be developed.

In Section 5.3, we developed a classification algorithm for trade networks and applied it to 9 subnetworks of the world trade network, based on 9 groups of commodities. As these groups are still quite extensive, future research could investigate on even more specialized sub groups, such as “sugar” and “mushrooms” instead of “food and live animals”, and classify hundreds of them.

The world trade analysis was conducted to demonstrate the practical usability of the algorithms developed in the preceding chapters. In future research, we will discuss the computed optimum patterns with economists in order to investigate on their real-world meanings.

# Appendices

# Table of Country Names

| | |
|---|---|
| AFG | Afghanistan |
| AFR | Africa CAMEU region, nes |
| AGO | Angola |
| AIA | Anguilla |
| ALB | Albania |
| AND | Andorra |
| ANT | Antarctica |
| ARB | Aruba |
| ARE | United Arab Emirates |
| ARG | Argentina |
| ARM | Armenia |
| ASM | American Samoa |
| ATF | Fr. South Antarctic Terr. |
| ATG | Antigua and Barbuda |
| AUS | Australia |
| AUT | Austria |
| AZE | Azerbaijan |
| BAN | Br. Antarctic Terr. |
| BDI | Burundi |
| BEL | Belgium |
| BEN | Benin |
| BFA | Burkina Faso |
| BGD | Bangladesh |
| BGR | Bulgaria |
| BHR | Bahrain |
| BHS | Bahamas |
| BIH | Bosnia Herzegovina |
| BLR | Belarus |
| BLU | Belgium-Luxembourg |
| BLZ | Belize |
| BMU | Bermuda |
| BOL | Bolivia |
| BRA | Brazil |
| BRB | Barbados |
| BRN | Brunei Darussalam |
| BTN | Bhutan |
| BUN | Bunkers |
| BVT | Bouvet Island |
| BWA | Botswana |
| CAC | CACM, nes |
| CAF | Central African Rep. |
| CAN | Canada |
| CAR | Caribbean, nes |
| CCK | Cocos Isds |
| CHE | Switzerland |
| CHL | Chile |
| CHN | China |
| CIV | Cote d'Ivoire |
| CMR | Cameroon |
| COD | Dem. Rep. of the Congo |
| COG | Congo |
| COK | Cook Isds |
| COL | Colombia |
| COM | Comoros |
| CPV | Cape Verde |
| CRI | Costa Rica |
| CSK | Czechoslovakia |
| CUB | Cuba |
| CXR | Christmas Isds |
| CYM | Cayman Isds |

| | |
|---|---|
| CYP | Cyprus |
| CZE | Czech Rep. |
| DDR | Fmr Dem. Rep. of Germany |
| DEU | Fmr Fed. Rep. of Germany |
| DEU | Germany |
| DJI | Djibouti |
| DMA | Dominica |
| DNK | Denmark |
| DOM | Dominican Rep. |
| DZA | Algeria |
| ECU | Ecuador |
| EEU | Eastern Europe, nes |
| EFT | Europe EFTA, nes |
| EGY | Egypt |
| ERI | Eritrea |
| ESH | Western Sahara |
| ESP | Spain |
| EST | Estonia |
| ETH | Ethiopia |
| ETH | Fmr Ethiopia |
| EU | Europe EU, nes |
| EU2 | EU-27 |
| FIN | Finland |
| FJI | Fiji |
| FLK | Falkland Isds (Malvinas) |
| FRA | France |
| FRO | Faeroe Isds |
| FSM | FS Micronesia |
| GAB | Gabon |
| GBR | United Kingdom |
| GEO | Georgia |
| GHA | Ghana |
| GIB | Gibraltar |
| GIN | Guinea |
| GLP | Guadeloupe |
| GMB | Gambia |
| GNB | Guinea-Bissau |
| GNQ | Equatorial Guinea |
| GRC | Greece |
| GRD | Grenada |
| GRL | Greenland |
| GTM | Guatemala |
| GUF | French Guiana |
| GUM | Guam |
| GUY | Guyana |
| HKG | China, Hong Kong SAR |
| HMD | Heard Island and McDonald Islands |
| HND | Honduras |
| HRV | Croatia |
| HTI | Haiti |
| HUN | Hungary |
| IDN | Indonesia |
| IND | India |
| IND | India, excl. Sikkim |
| IOT | Br. Indian Ocean Terr. |
| IRL | Ireland |
| IRN | Iran |
| IRQ | Iraq |
| ISL | Iceland |
| ISR | Israel |
| ITA | Italy |
| JAM | Jamaica |
| JOR | Jordan |
| JPN | Japan |
| KAZ | Kazakhstan |
| KEN | Kenya |
| KGZ | Kyrgyzstan |
| KHM | Cambodia |
| KIR | Kiribati |
| KNA | Saint Kitts and Nevis |
| KNA | Saint Kitts, Nevis and Anguilla |
| KOR | Rep. of Korea |
| KWT | Kuwait |
| LAI | LAIA, nes |
| LAO | Lao People’s Dem. Rep. |
| LBN | Lebanon |

| | |
|---|---|
| LBR | Liberia |
| LBY | Libya |
| LCA | Saint Lucia |
| LKA | Sri Lanka |
| LSO | Lesotho |
| LTU | Lithuania |
| LUX | Luxembourg |
| LVA | Latvia |
| MAC | China, Macao SAR |
| MAR | Morocco |
| MDA | Rep. of Moldova |
| MDG | Madagascar |
| MDV | Maldives |
| MEX | Mexico |
| MHL | Marshall Isds |
| MKD | TFYR of Macedonia |
| MLI | Mali |
| MLT | Malta |
| MMR | Myanmar |
| MNE | Montenegro |
| MNG | Mongolia |
| MNP | N. Mariana Isds |
| MOZ | Mozambique |
| MRT | Mauritania |
| MSR | Montserrat |
| MTQ | Martinique |
| MUS | Mauritius |
| MWI | Malawi |
| MYS | Malaysia |
| MYT | Mayotte |
| NAF | Northern Africa, nes |
| NAM | Namibia |
| NAN | Neth. Antilles |
| NAN | Neth. Antilles and Aruba |
| NCA | North America and Central America, nes |
| NCL | New Caledonia |
| NER | Niger |
| NEU | Neutral Zone |
| NFK | Norfolk Isds |
| NGA | Nigeria |
| NIC | Nicaragua |
| NIU | Niue |
| NLD | Netherlands |
| NOR | Norway |
| NPL | Nepal |
| NRU | Nauru |
| NZL | New Zealand |
| OAF | Other Africa, nes |
| OAM | Rest of America, nes |
| OAS | Other Asia, nes |
| OCE | Oceania, nes |
| OEU | Other Europe, nes |
| OMN | Oman |
| PAK | East and West Pakistan |
| PAK | Pakistan |
| PAN | Fmr Panama, excl.Canal Zone |
| PAN | Panama |
| PCI | Fmr Pacific Isds |
| PCN | Pitcairn |
| PCZ | Fmr Panama-Canal-Zone |
| PER | Peru |
| PHL | Philippines |
| PLW | Palau |
| PMA | Peninsula Malaysia |
| PNG | Papua New Guinea |
| POL | Poland |
| PRK | Dem. People's Rep. of Korea |
| PRT | Portugal |
| PRY | Paraguay |
| PSE | Occ. Palestinian Terr. |
| PYF | French Polynesia |
| QAT | Qatar |
| REU | Reunion |
| RHO | Fmr Rhodesia Nyas |
| ROM | Romania |

| | |
|---|---|
| RUS | Russian Federation |
| RWA | Rwanda |
| RYU | Ryukyu Isd |
| SAB | Sabah |
| SAR | Sarawak |
| SAU | Saudi Arabia |
| SCG | Serbia and Montenegro |
| SDN | Sudan |
| SEN | Senegal |
| SGP | Singapore |
| SGS | South Georgia and the South Sandwich Islands |
| SHN | Saint Helena |
| SIK | Sikkim |
| SLB | Solomon Isds |
| SLE | Sierra Leone |
| SLV | El Salvador |
| SMR | San Marino |
| SOM | Somalia |
| SPM | Saint Pierre and Miquelon |
| SRB | Serbia |
| STP | Sao Tome and Principe |
| SUN | Fmr USSR |
| SUR | Suriname |
| SVK | Slovakia |
| SVN | Slovenia |
| SWE | Sweden |
| SWZ | Swaziland |
| SYC | Seychelles |
| SYR | Syria |
| TAN | Fmr Tanganyika |
| TCA | Turks and Caicos Isds |
| TCD | Chad |
| TGO | Togo |
| THA | Thailand |
| TJK | Tajikistan |
| TKL | Tokelau |
| TKM | Turkmenistan |
| TMP | Timor-Leste |
| TON | Tonga |
| TTO | Trinidad and Tobago |
| TUN | Tunisia |
| TUR | Turkey |
| TUV | Tuvalu |
| TZA | United Rep. of Tanzania |
| UGA | Uganda |
| UKR | Ukraine |
| UMI | United States Minor Outlying Islands |
| URY | Uruguay |
| USA | USA |
| USA | USA (before 1981) |
| USP | US Misc. Pacific Isds |
| UZB | Uzbekistan |
| VAT | Holy See (Vatican City State) |
| VCT | Saint Vincent and the Grenadines |
| VDR | Fmr Dem. Rep. of Vietnam |
| VEN | Venezuela |
| VGB | Br. Virgin Isds |
| VIR | US Virgin Isds |
| VNM | Fmr Rep. of Vietnam |
| VNM | Viet Nam |
| VUT | Vanuatu |
| WAS | Western Asia, nes |
| WLD | World |
| WLF | Wallis and Futuna Isds |
| WSM | Samoa |
| XXX | Special Categories |
| XXY | Areas, nes |
| YEM | Fmr Arab Rep. of Yemen |
| YEM | Yemen |
| YMD | Fmr Dem. Yemen |
| YUG | Fmr Yugoslavia |
| ZAF | So. African Customs Union |
| ZAF | South Africa |
| ZAN | Fmr Zanzibar and Pemba Isd |
| ZMB | Zambia |

| | |
|---|---|
| ZON | Free Zones |
| ZWE | Zimbabwe |

# Country Partitions

| Multiple Arc Type, 4 groups, 4 arc types | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | AUS | ARE | ARG | AUT | BEL | BGR | BLR | BRA | CAN | CHE |
| | CHL | CHN | COL | CZE | DEU | DNK | DZA | ECU | EGY | ESP |
| | EST | FRA | GBR | GUY | HKG | HRV | HUN | IDN | IND | IRL |
| | IRN | ISR | ITA | JPN | KOR | LBY | LVA | MEX | MYS | NGA |
| | NLD | NOR | PAN | POL | PRT | ROM | RUS | SGP | SLV | SRB |
| | SWE | THA | TUR | UGA | UKR | USA | VNM | ZAF | | |
| B | ALB | AZE | BFA | BGD | BIH | BOL | BWA | CIV | CMR | COG |
| | CRI | CYP | DOM | ETH | FIN | FJI | GEO | GHA | GRC | GTM |
| | HND | ISL | JOR | KAZ | KEN | KHM | LKA | LTU | LUX | MAR |
| | MMR | MOZ | MRT | NAM | NZL | OMN | PAK | PER | PHL | PRY |
| | QAT | SDN | SEN | SUR | SVK | SVN | SYR | TGO | TTO | TUN |
| | TZA | URY | YEM | ZMB | ZWE | | | | | |
| C | AFG | ARB | ARM | ATG | BDI | BEN | BHR | BHS | BLZ | BRB |
| | BTN | CAF | DMA | GMB | GRL | JAM | KGZ | LBN | MAC | MDA |
| | MDG | MDV | MKD | MLI | MLT | MNE | MUS | MWI | NCL | NER |
| | NIC | NPL | PSE | PYF | RWA | SAU | SLB | STP | TON | VCT |
| | VEN | VUT | WSM | | | | | | | |
| D | CYM | AGO | AIA | AND | ANT | ASM | ATF | BMU | BRN | BUN |
| | BVT | CCK | COD | COK | COM | CPV | CUB | CXR | DJI | ERI |
| | FLK | FRO | FSM | GAB | GIB | GIN | GNB | GNQ | GRD | GUM |
| | HMD | HTI | IOT | IRQ | KIR | KNA | KWT | LAO | LBR | LCA |
| | LSO | MHL | MNG | MNP | MSR | MYT | NAN | NFK | NIU | NRU |
| | PCN | PLW | PNG | PRK | SGS | SHN | SLE | SMR | SOM | SPM |
| | SWZ | SYC | TCA | TCD | TJK | TKL | TKM | TMP | TUV | UZB |
| | VAT | VGB | WLF | | | | | | | |

| Single Arc Type, 2 groups | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Left | AUS | AUT | BEL | BRA | CZE | FRA | HKG | HUN | IRN | IRL | ITA |
| | JPN | MEX | MYS | NLD | NOR | PAN | PHL | PRT | ROM | SVK | ZAF |
| | ESP | SWE | CHE | THA | TUR | EGY | GBR | DZA | ARG | VNM | |
| Right | CHL | CHN | DNK | DEU | IDN | KAZ | KOR | NGA | POL | RUS | SAU |
| | IND | SGP | ARE | USA | QAT | | | | | | |

| Single Arc Type, 4 groups | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Center | BRA | CHN | CZE | DEU | IND | JPN | KOR | SGP | USA | |
| Top | ARG | EGY | HUN | PAN | PHIL | QAT | SAU | SVK | VNM | ZAF |
| Left | ARE | AUT | AUS | CHL | DNK | DZA | FIN | HKG | IDN | IRL |
| | KAZ | MEX | NGA | NOR | POL | ROM | TUR | | | |
| Right | BEL | CAN | CHE | ESP | FRA | GBR | IRN | ITA | MYS | NLD |
| | RUS | SWE | THA | | | | | | | |

| Multiple Arc Type, 6 groups, 3 arc types | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Upper left (orange) | AFG | BHR | BOL | BLZ | KHM | CAF | CYP | LUX | MDA | NPL |
| | UGA | CMR | BEN | ETH | GEO | GMB | PSE | MRT | NER | TGO |
| | BFA | YEM | KGZ | NAM | ARB | ATG | BHS | ARM | SLB | DMA |
| | JAM | MAC | MWI | MDV | MNE | VUT | BTN | BDI | PYF | GRL |
| Upper center (black) | CYM | IRQ | KWT | LBR | MHL | SWZ | TJK | AGO | CPV | TCD |
| | CUB | GNQ | DJI | GAB | GIN | BMU | BRN | COD | MNG | NAN |
| | TKM | UZB | ASM | AND | GIB | LAO | LSO | PNG | GNB | KNA |
| | LCA | VGB | ERI | GUM | NIU | NFK | TMP | SOM | BUN | FRO |
| | PRK | TCA | | | | | | | | |
| Upper right (red) | ALB | EST | GTM | ISL | JOR | LVA | LBY | LTU | MDG | MLI |
| | RWA | SRB | MKD | TZA | LKA | DOM | SLV | GHA | LBN | MOZ |
| | NZL | ZWE | URY | BGD | BRB | BWA | MMR | FJI | GUY | HND |
| | MUS | PRY | VCT | SUR | NCL | WSM | PRK | TCA | | |
| Lower left (yellow) | AUS | BGR | CAN | CHL | HRV | DNK | FIN | GRC | HKG | IRN |
| | ISR | KAZ | MYS | MAR | OMN | NGA | PAN | PHL | POL | PRT |
| | ROM | SAU | SEN | SVN | ZAF | CHE | TUR | UKR | EGY | ZMB |
| | DZA | ECU | CIV | KEN | VNM | SDN | TUN | VEN | AZE | COL |
| | TTO | CRI | PER | | | | | | | |
| Lower center (white) | COM | SLE | HTI | STP | SYC | GRD | SMR | TON | ANT | IOT |
| | CXR | CCK | COK | FLK | KIR | NRU | MNP | FSM | PLW | PCN |
| | SHN | AIA | SPM | TUV | WLF | MYT | ATF | VAT | MSR | TKL |
| | BVT | SGS | HMD | | | | | | | |
| Lower right (blue) | AUT | BEL | BIH | BRA | BLR | CHN | COG | CZE | FRA | DEU |
| | HUN | IDN | IRL | ITA | JPN | KOR | MLT | MEX | NLD | NIC |
| | NOR | RUS | IND | SGP | SVK | ESP | SWE | SYR | THA | ARE |
| | GBR | USA | ARG | PAK | QAT | | | | | |

# Index

# Bibliography

[ABL78] Phipps Arabie, Scott A Boorman, and Paul R Levitt, *Constructing blockmodels: How and why*, Journal of Mathematical Psychology **17** (1978), no. 1, 21–63.

[Ach09] Tobias Achterberg, *Scip: Solving constraint integer programs*, Math. Prog. Comp. **1** (2009), no. 1, 1–41.

[AK08] Gaurav Agarwal and David Kempe, *Modularity-maximizing graph communities via mathematical programming*, The European Physical Journal B **66** (2008), no. 3, 409–418.

[Alb73] Richard D Alba, *A graph-theoretic definition of a sociometric clique*, Journal of Mathematical Sociology **3** (1973), no. 1, 113–126.

[Bal64] Egon Balas, *Extension de l'algorithme additif a la programmation en nombres entiers et a la programmation non lineaire*, CR Acad. Sci. Paris **258** (1964), 5136–5139.

[Bar92] AI Barvinok, *Combinatorial complexity of orbits in representations of the symmetric group*, Advances in Soviet Mathematics **9** (1992), 161–182.

[BDF92] Vladimir Batagelj, Patrick Doreian, and Anuška Ferligoj, *An optimizational approach to regular equivalence*, Social Networks **14** (1992), no. 1, 121–135.

[BE89] Stephen P Borgatti and Martin G Everett, *The class of all regular equivalences: Algebraic structure and computation*, Social Networks **11** (1989), no. 1, 65–88.

[BE92] ———, *Regular blockmodels of multiway, multimode matrices*, Social Networks **14** (1992), no. 1, 91–120.

[BE93] ———, *Two algorithms for computing regular equivalence*, Social Networks **15** (1993), no. 4, 361–376.

[BE99] John P Boyd and Martin G Everett, *Relations, residuals, regular interiors, and relative regular equivalence*, Social Networks **21** (1999), no. 2, 147–165.

[BE00] Stephen P Borgatti and Martin G Everett, *Models of core/periphery structures*, Social Networks **21** (2000), no. 4, 375–395.

[BEF02] Stephen P Borgatti, Martin G Everett, and Linton C Freeman, *Ucinet 6 for windows: Software for social network analysis*, Harvard, MA: Analytic Technologies (2002).

[BHL$^+$07] Jonathan W Berry, Bruce Hendrickson, Randall A LaViolette, Vitus J Leung, and Cynthia A Phillips, *Community detection via facility location*, arXiv preprint arXiv:0710.3800 (2007).

[BL10] Ulrik Brandes and Jürgen Lerner, *Structural similarity: spectral methods for relaxed blockmodeling*, Journal of classification **27** (2010), no. 3, 279–306.

[BM86] Francisco Barahona and Ali Ridha Mahjoub, *On the cut polytope*, Mathematical programming **36** (1986), no. 2, 157–173.

[Bon11] Thorsten Bonato, *Contraction-based separation and lifting for solving the max-cut problem*, Ph.D. thesis, Universitaet Heidelberg, 2011.

[BS09] Michael J Brusco and Douglas Steinley, *Integer programs for one-and two-mode blockmodeling based on prespecified image matrices for structural and regular equivalence*, Journal of Mathematical Psychology **53** (2009), no. 6, 577–585.

[Bur76] Ronald S Burt, *Positions in networks*, Social forces **55** (1976), no. 1, 93–122.

[CDW08] William YC Chen, Andreas WM Dress, and Q Yu Winking, *Community structures of networks*, Mathematics in Computer Science **1** (2008), no. 3, 441–457.

[CHB80] Peter J Carrington, Greg H Heil, and Stephen D Berkowitz, *A goodness-of-fit index for blockmodels*, Social Networks **2** (1980), no. 3, 219–234.

[Com15] U.N. Comtrade, *United nations commodity trade statistics database*, URL: http://comtrade.un.org (2015).

[Das14] Niheer Dasandi, *International inequality and world poverty: A quantitative structural analysis*, New Political Economy **19 (2)** (2014), 201–226.

[DBF05] Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj, *Generalized blockmodeling*, vol. 25, Cambridge University Press, 2005.

[FMdS$^+$96] Carlos E Ferreira, Alexander Martin, Cid C de SOUZA, Robert Weismantel, and Laurence A. Wolsey, *Formulations and valid inequalities for the node capacitated graph partitioning problem*, Mathematical Programming **74** (1996), no. 3, 247–266.

[For09] Santo Fortunato, *Community detection in graphs*, Physics Reports **486** (2009), no. 3, 75–174.

[FP03] Jiří Fiala and Daniël Paulusma, *The computational complexity of the role assignment problem*, Springer, 2003.

[FY83] AM Frieze and J Yadegar, *On the quadratic assignment problem*, Discrete applied mathematics **5** (1983), no. 1, 89–98.

[GW89] Martin Grötschel and Yoshiko Wakabayashi, *A cutting plane algorithm for a clustering problem*, Mathematical Programming **45** (1989), no. 1-3, 59–96.

[Haa13] Dominik Haas, *Relaxierungen des Reguläre-Äquivalenz-Problems*, Master's thesis, Universität Heidelberg, 2013.

[Ji04] Xiaoyun Ji, *Graph partition problems with minimum size constraints*, Ph.D. thesis, Rensselaer Polytechnic Institute, 2004.

[JMN93] Ellis L Johnson, Anuj Mehrotra, and George L Nemhauser, *Min-cut clustering*, Mathematical Programming **62** (1993), no. 1-3, 133–151.

[KA91] Gary Klein and Jay E Aronson, *Optimal clustering: A model and method*, Naval Research Logistics (NRL) **38** (1991), no. 3, 447–461.

[Kai97] Volker Kaibel, *Polyhedral combinatorics of the quadratic assignment problem*, Ph.D. thesis, Universitaet Koeln, 1997.

[Kar72] Richard M Karp, *Reducibility among combinatorial problems*, Springer, 1972.

[KL70] Brian W Kernighan and Shen Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Technical Journal **49** (1970), no. 2, 291–307.

[KS96] David R Karger and Clifford Stein, *A new approach to the minimum cut problem*, Journal of the ACM (JACM) **43** (1996), no. 4, 601–640.

[LBJE03] Joseph J Luczkovich, Stephen P Borgatti, Jeffrey C Johnson, and Martin G Everett, *Defining and measuring trophic role similarity in food webs using regular equivalence*, Journal of Theoretical Biology **220** (2003), no. 3, 303–321.

[Ler05] Jürgen Lerner, *Role assignments*, Network analysis, Springer, 2005, pp. 216–252.

[Lib07] Leo Liberti, *Compact linearization for binary quadratic problems*, 4OR **5** (2007), no. 3, 231–245.

[LML09] Paulette Lloyd, Matthew C Mahutga, and Jan De Leeuw, *Looking back and forging ahead: Thirty years of social network research on the world-system*, American Sociological Association **XV (1)** (2009), 48–85.

[LW71] Francois Lorrain and Harrison C White, *Structural equivalence of individuals in social networks*, The Journal of Mathematical Sociology **1** (1971), no. 1, 49–80.

[Mel06] Guy Melancon, *Just how dense are dense graphs in the real world?: a methodological note*, Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization, ACM, 2006, pp. 1–7.

[MKTT06] Zeev Maoz, Ranan D Kuperman, Lesley Terris, and Ilan Talmud, *Structural equivalence and international conflict a social networks analysis*, Journal of Conflict Resolution **50** (2006), no. 5, 664–689.

[Mok79] Robert J. Mokken, *Cliques, clubs and clans*, Quality and Quantity **13** (1979), 161–173.

[MS06] Matthew C Mahutga and David A Smith, *Globalization, the structure of the world economy and economic development*, IROWS Working Paper **#52** (2006).

[MT98] Anuj Mehrotra and Michael A Trick, *Cliques and clustering: A combinatorial approach*, Operations Research Letters **22** (1998), no. 1, 1–12.

[Nat] United Nations, *http://comtrade.un.org/db/help/ureadmefirst.aspx*, May 12th 2015.

[Nat98] ______, *International merchandise trade statistics: Concepts and definitions*, Studies in Methods Series M, No.52, Rev.2, 1998.

[NG04] Mark EJ Newman and Michelle Girvan, *Finding and evaluating community structure in networks*, Physical review E **69** (2004), no. 2, 026113.

[Nor07] Carl Nordlund, *Identifying regular blocks in valued networks: A heuristic applied to the st. marks carbon flow data, and international trade in cereal products*, Social Networks **29** (2007), no. 1, 59–69.

[NS05] Marc Nunkesser and Daniel Sawitzki, *Blockmodels*, Network Analysis, Springer, 2005, pp. 253–292.

[RS01] Fred S Roberts and Li Sheng, *How hard is it to determine if a graph has a 2-role assignment?*, Networks **37** (2001), no. 2, 67–73.

[Rub13] Simon Rube, *Eine Heuristik zur Bestimmung regulärer Äquivalenzklassen*, Master's thesis, Universität Heidelberg, 2013.

[RW07] Jörg Reichardt and Douglas R White, *Role models for complex networks*, The European Physical Journal B-Condensed Matter and Complex Systems **60** (2007), no. 2, 217–224.

[Sco02] John Scott, *Social networks: Critical concepts in sociology*, vol. 2, Taylor & Francis, 2002.

[SF78] S.B. Seidman and B.L. Foster, *A graph-theoretic generalization of the clique concept*, Journal of Mathematical Sociology **6** (1978), 139–154.

[Srh06] Martin Srholec, *Fragmentation and trade: A network perspective*, ETSG Annual Conference in Vienna (2006).

[SW92] David A Smith and Douglas R White, *Structure and dynamics of the global economy: Network analysis of international trade 1965–1980*, Social Forces **70** (1992), no. 4, 857–893.

[Vin69] Hrishikesh D Vinod, *Integer programming and the theory of grouping*, Journal of the American Statistical Association **64** (1969), no. 326, 506–519.

[WCHC10] Calvin S Weng, Wan-Yu Chen, Hui-Ying Hsu, and Shih-Hung Chien, *To study the technological network by structural equivalence*, The Journal of High Technology Management Research **21** (2010), no. 1, 52–63.

[WF94] Stanley Wasserman and Katherine Faust, *Social network analysis: Methods and applications*, vol. 8, Cambridge University Press, 1994.

[WS98] DJ Watts and SH Strogatz, *Collective dynamics of small-world networks*, Nature **393 (6684)** (1998), 440–442.

[XTP07] G Xu, S Tsoka, and LG Papageorgiou, *Finding community structures in complex networks using mixed integer optimisation*, The European Physical Journal B **60** (2007), no. 2, 231–239.