Mixed-Integer Programming for Clustering in Non-reversible Markov Processes

Masterarbeit bei
Prof. Dr. Thorsten Koch

Zweitkorrektor: PD Dr. Marcus Weber

vorgelegt von
Leon Eifler
Technische Universität Berlin
Fachbereich Mathematik

4. January, 2018

Eigeständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Hereby I confirm that the work contained in this thesis is my own unless otherwise stated. All adoptions of literature have been referenced as such and are listed in the References section.

Berlin, den 4. January, 2018

_____

Leon Eifler

Abstract

The topic of this thesis is the examination of an optimization model which stems from the clustering process of non-reversible markov processes. We introduce the *cycle clustering problem* und formulate it as a mixed integer program (MIP).

We prove that this problem is $\mathcal{NP}$-hard and discuss polytopal aspects such as facets and dimension. The focus of this thesis is the development of solving methods for this clustering problem. We develop problem specific primal heuristics, as well as separation methods and an approximation algorithm. These techniques are implemented in practice as an application for the MIP solver SCIP.

Our computational experiments show that these solving methods result in an average speedup of $\times 4$ compared to generic solvers and that our application is able to solve more instances to optimality within the given time limit of one hour.

Zusammenfassung

Inhalt dieser Arbeit ist die Untersuchung eines Optimierungsmodells, welches zum Clustering von nicht-reversiblen Markow-Prozessen genutzt wird. Das sogenannte *cycle clustering problem* wird motiviert und als gemischt ganzzahliges Programm formuliert. Wir beweisen, dass dieses Problem $\mathcal{NP}$-schwer ist und untersuchen die Dimension und Facetten des zugehörigen Polytops. Schwerpunkt der Arbeit ist die Entwicklung von Lösungsmethoden für dieses Problem. Wir entwickeln primale Heuristiken, Separierungsmethoden sowie einen Approximationsalgorithmus. Diese Methoden werden als eine Erweiterung des MIP Lösers SCIP implementiert.

Unsere rechnerischen Experimente zeigen, dass diese Lösungsmethoden eine durchschnittliche Beschleunigung um einen Faktor $\times 4$ im Vergleich mit generischen Lösern zeigen. Des weiteren können innerhalb des Zeitlimits von einer Stunde mehr Instanzen optimal gelöst werden.

Contents

Chapter 1

Introduction and Preliminaries

1.1 Introduction

Graphs are one of the most commonly used structures when mathematically modeling any kind of application problem. The study of large graphs inherently leads to graph partitioning problems. Any time a graph has to be decomposed into smaller parts, a graph partitioning problem has to be solved. Example applications are VLSI layout design [24] and GSM frequency planning [14].

Already the case where the graph is only split in two parts, the *max-cut problem* is $\mathcal{NP}$-complete [25]. However, several researchers have studied graph-partitioning problems with a greater number of partitions. Grötschel and Wakabayashi have studied the *clique partitioning polytope* [22] in order to solve problems in data analysis. Later, Chopra and Rao have introduced the *k-partition problem* [8] and a formulation that is beneficial if the studied graph is not complete. Ferreira et al. have studied the *node capacitated graph partitioning problem* [17] that imposes capacity restrictions on the partitions.

At the center of many of these publications is the search for valid inequalities in order to produce tighter relaxation bounds. Other approaches to solve the graph-partitioning problem involve semidefinite programming [3], continuous non-convex quadratic programming [23] or combining different relaxations [28].

In this thesis, we investigate a specific graph partitioning problem and attempt to solve it using mixed integer programing. The problem, called *cycle clustering*, stems from the coarse graining process of markov state models

1.2 Motivation

In the computational modeling of biological and chemical processes as markov state models, one obtains a stochastic transition matrix that represents the time evolution of the system. As a direct interpretation of this high-dimensional matrix is difficult, clustering methods are used to extract insight from the data. In [4], a MIP-based clustering approach called *cycle clustering* was developed for non-reversible processes such as catalytic cycles that do not meet the requirements of previous clustering methods.

The idea of this method is to identify a cycle of clusters such that the directed cut between consecutive clusters is maximal. It was proven in [4] that the resulting optimization problem is $\mathcal{NP}$-hard.

The aim of this thesis is to develop solving techniques for the cycle clustering problem.

1.3 Outline

In the remaining part of Chapter 1, we present basics of mixed integer programming, as well as a short introduction to markov state models.

In Chapter 2, we formally introduce the cycle clustering problem and discuss MIP formulations for it. Moreover, we discuss the complexity in detail and investigate polytopal aspects of the MIP formulation. In Chapter 3, we present problem specific primal heuristics and valid inequalities. As the transition matrices obtained from molecular modeling are often not sparse, we discuss possibilities to obtain approximate solutions by reducing the instance size.

In Chapter 4, we conduct computational experiments, testing the overall runtime and the impact of the various problem specific extensions presented in Chapter 3. We compare our results to those of generic MIP solvers. Finally we draw conclusions and give an outlook in Chapter 5.

1.4 Preliminaries

1.4.1 Mixed Integer Programming

Let $m, n \in \mathbb{N}$. Let $A \in \mathbb{R}^{n \times m}$ be a matrix of coefficients, $c \in \mathbb{R}^n, b \in \mathbb{R}^m$ and $\mathcal{I} \subseteq \{1, \dots, n\}$ be the index-set of integer variables.

A mixed integer program is an optimization problem of the form

$$\begin{aligned} \text{(MIP)} \qquad & \min c^T x \\ & \text{s.t.} \quad Ax \leq b \\ & \qquad\quad x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I} \\ & \qquad\quad x_i \in \mathbb{R} \quad \forall i \in \{1, \dots, n\} \setminus \mathcal{I}. \end{aligned}$$

Any $x \in \mathbb{R}^n$ that fulfills the restrictions above is called *feasible solution* of the MIP. Special cases of MIPs that will be of interest in this thesis are *integer programs* (IPs), where $\mathcal{I} = \{1, \dots, n\}$, *binary programs* (BPs), where $\mathcal{I} = \{1, \dots, n\}$ and all variables are bounded between 0 and 1, as well as *linear programs* (LPs), where $\mathcal{I} = \emptyset$.

Solving MIPs is $\mathcal{NP}$-hard in general [19]. Omitting the integrality requirement in the above problem is called the *LP-relaxation* of a MIP

$$\begin{aligned} \text{(LP-relaxation)} \qquad & \min c^T x \\ & \text{s.t.} \quad Ax \leq b \\ & \qquad\quad x_i \in \mathbb{R} \quad \forall i \in \{1, \dots, n\}. \end{aligned}$$

In order to shorten notation, we will refer to the optimal solution of the LP-relaxation as the *LP-solution*.

An optimal solution of the LP-relaxation provides a lower bound on the optimal value of the MIP. The most common way to solve MIPs, the *LP-based branch-and-bound*, exploits this property.

1.4.2 Branch-and-Bound

We illustrate the branch-and-bound technique at the example of LP-based branch-and-bound, as this is the problem we focus on in this thesis. As pointed out previously, the LP-solution x^* of an LP-relaxation provides a lower bound on the optimal solution of the MIP. If x^* satisfies all of the integrality conditions, then it is also a feasible solution of the MIP and therefore the global optimal solution.

If this is not the case, we can select a variable x_i, with $\lfloor x_i^* \rfloor < x_i^* < \lceil x_i^* \rceil$.

We introduce two new subproblems that both consist of the original MIP, with the added constraints $x_i \leq \lfloor x_i^* \rfloor$ and $x_i \geq \lceil x_i^* \rceil$, respectively. We select one of them, compute the LP-relaxation and proceed to create further subproblems in the same way. This is called the branching step. If the solution x^* of one of the LP-relaxations of a subproblem satisfies all integrality conditions, then it is a feasible solution to the original MIP and thus provides an upper bound on the optimal solution. If the LP-relaxation of any subproblem has a solution value greater than that of the best known integer-feasible solution $\hat{x}$, then this subtree can be discarded. This is called the bounding step. A subtree can also be discarded if the LP-relaxation of a subproblem has no feasible solution.

The whole concept can be visualized as a *branch-and-bound tree*, see Figure 1.1.

1.4.3 Introduction to Markov State Models

We give a brief overview over the topic of markov state models [12, 33, 34]. A more extensive introduction for non-experts in this field can be found in [32].

Generally speaking, a markov state model can be used as a way to describe and analyze data from a simulation in a meaningful way as a discrete system of states. Having a set of data (e.g., trajectories from a molecular dynamics simulation), we define a set of n microstates and then assign each data point from our simulation to one of these microstates. It is possible to assign membership to a microstate on a continuous basis (see [37]), i.e., every data point is assigned fractionally to different microstates, using a membership function.

The number of transitions between any two microstates i and j that occur within some defined lag time τ is counted, i.e., if the system is in microstate i at time t, how often will it go to state j at time $t + \tau$.

The probability p_{ij} of a transition from i to j can be estimated by dividing the count of those transitions by the overall number of transition

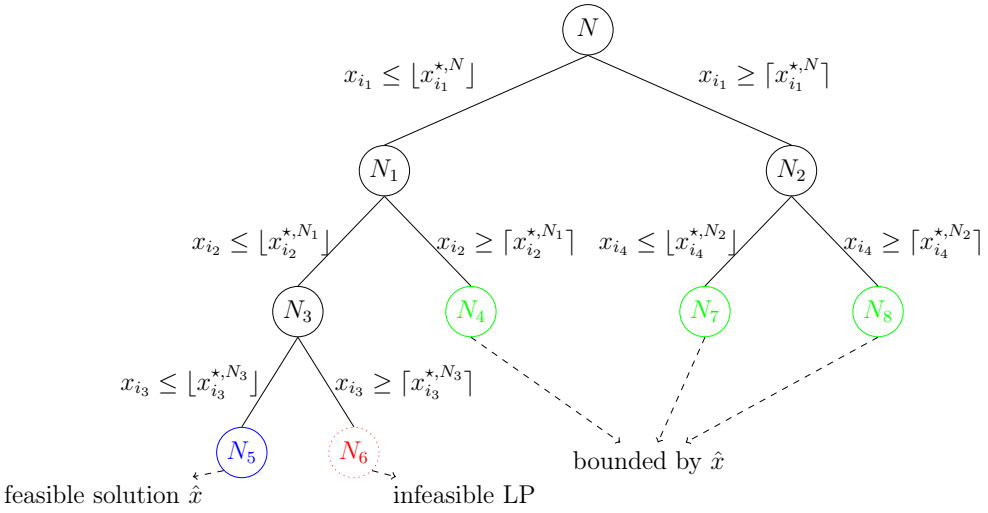


FIGURE 1.1: Illustration of a branch-and-bound tree. We solve the LP-relaxation of the root problem N and branch on the fractional variable x_{i_1}, creating the subproblems N_1 and N_2. We select N_1 as the active node and branch again. In this example, the LP-relaxation of N_5 is feasible for the original MIP. N_6 can be discarded because the LP-relaxation is infeasible. The other nodes N_4, N_7, and N_8 can be discarded due to bounding as their LP-relaxation has a larger objective value than $\hat{x}$.

starting at i. We will call the matrix of conditional probabilities obtained this way the *transition matrix* $P \in [0,1]^{n \times n}$.

The markov state model is usually a high-resolution model, i.e., it will have a large number of states n. Thus, it is often hard to gain insight from the data or visualize the process in an intuitive way.

For that reason, clustering or coarse graining methods are used to create models with a lower resolution, i.e., a lower number of states. There are several ways to do this. Spectral clustering methods can be used to cluster states that are kinetically related into *metastabilities* [11], by taking into account the m leading eigenvalues of P. If the dynamics of the process can be characterized by *dominant cycles*, then eigenvalues of P that are close to the complex unit circle can be identified and used to create a clustering [9]. These methods are well-studied and fast due to the well-developed linear algebra solving techniques for eigenvalue decompositions. However, if one wants to investigate general cyclic behavior of a process that does not have a dominant cycle, a different approach is needed.

We will present such a clustering method that can be used to identify and evaluate cycles in a markov state model.

Chapter 2

Cycle Clustering

In this chapter, we formally introduce and discuss the cycle clustering problem. First, we formulate a high-level description of the problem and prove that it is $\mathcal{NP}$-hard. Afterwards, we construct an IP-formulation which we obtain by linearizing a binary quadratic program. Moreover, we investigate the dimension and facets of the polytope corresponding to this formulation. Finally, we present an alternative formulation for cycle clustering, using semidefinite programming.

2.1 The Cycle Clustering Model

As we briefly illustrated in the previous section, clustering methods are needed for the analysis and understanding of markov state models. In [4] a new clustering approach called *cycle clustering* was developed that identifies an ordered set of clusters.

We construct the clustering model in the same way as in [4], starting from a markov process with a finite set of states $\mathcal{S} = \{1, \dots, n\}$ and a fixed number of clusters $m \geq 3$. A *clustering* of $\mathcal{S}$ has to contain all states i.e., $\bigcup_{i=1}^{m} C_i = \mathcal{S}$, and for each $t \neq t' \in \{1, \dots, m\}$ the clusters have to be disjoint, i.e., $C_t \cap C_{t'} = \emptyset$. Furthermore, all clusters have to be non-empty, i.e., $C_t \neq \emptyset$, for all $t \in \{1, \dots, m\}$.

This markov process has a matrix of conditional transition probabilities $P \in \mathbb{R}^{n \times n}$, where each entry p_{ij} is the probability of moving from one state i to another state j in one time step. We assume that P is stochastic, i.e., that the row-sum of each row is equal to one. We also assume that there exists a stationary distribution vector $\pi \in [0, 1]^n$ such that $\pi^T P = \pi$ and $\sum_{i=1}^{n} \pi_i = 1$. A sufficient condition for this is if the markov process is irreducible [13].

We define the unconditional transition matrix $Q \in \mathbb{R}^{n \times n}$ by setting $q_{ij} \coloneqq \pi_i p_{ij}$ for all $i, j \in \mathcal{S}$. Each entry q_{ij} is the unconditional probability of a transition from i to j among all transitions. The sum of all entries in Q equals one, as

$$\sum_{i,j=1}^{n} q_{ij} = \sum_{i=1}^{n} \pi_i \sum_{j=1}^{n} p_{ij} = \sum_{i=1}^{n} \pi_i = 1.$$

The goal of our clustering approach consists of two parts. The first part is to identify cycles in this markov process, i.e., we want to partition the states $\mathcal{S}$ into an ordered set of clusters $\mathcal{C} = (C_1, \ldots, C_m)$, such that the probability of moving from one cluster to the next is higher than the probability of going backwards. At the same time, we do not want to cluster states together that rarely interact with each other, i.e., have a low transition probability. To measure these two objectives we define the *net flow* between two clusters, as well as the *coherence* within a cluster.

Definition 1. *Let $A, B \subset \mathcal{S}$ be two disjoint sets of states. Then the net flow from A to B is defined as*

$$f(A, B) = \sum_{i \in A} \sum_{j \in B} (q_{ij} - q_{ji}). \tag{2.1}$$

Definition 2. *Let $A \subseteq \mathcal{S}$. Then the coherence of A is defined as*

$$g(A) = \sum_{\substack{i,j \in A \\ i<j}} (q_{ij} + q_{ji}). \tag{2.2}$$

The coherence of a set of states is the probability that a transition takes place inside this set.

We define a function for denoting consecutive clusters as

$$\phi : \{1, \ldots, m\} \to \{1, \ldots, m\}, \quad \phi(t) = \begin{cases} t+1, & \text{if } t < m, \\ 1, & \text{if } t = m. \end{cases}$$

Our goal is to find a clustering that maximizes the overall net flow while still preserving some structural integrity, measured by the coherence. The high-level description of this optimization problem is

$$\begin{aligned} \max \sum_{t=1}^{m} f(C_t, C_{\phi(t)}) + \alpha \sum_{t=1}^{m} g(C_t) \\ \text{s.t.} \quad \bigcup_{t=1}^{m} C_t = \mathcal{S} \\ C_t \neq \emptyset \quad \forall t \in \{1, \ldots, m\} \\ C_t \cap C_{t'} = \emptyset \quad \forall t \neq t' \in \{1, \ldots, m\}. \end{aligned}$$

Here, $\alpha > 0$ is a scaling parameter that determines the influence of the coherence. A low value $\alpha << 1$ is usually used to find a clustering with high net flow, whereas the coherence is needed to cluster states that have no net flow between them. For an example that illustrates the importance of coherence, we refer to [4].

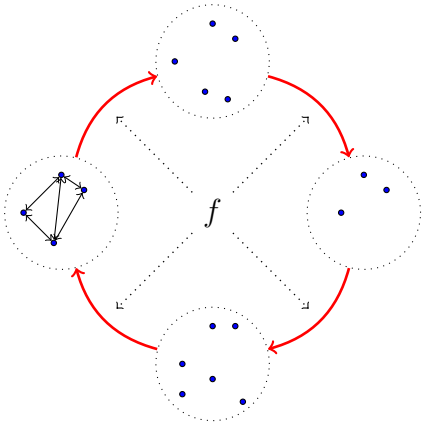


FIGURE 2.1: Illustration of the clustering model. The net flow from one cluster to the next along the cycle is marked by the one-sided arrows. The transition probabilities between all the states in one cluster form the coherence. These transitions are shown as the two-sided arrows in the first cluster.

2.2 Complexity of Cycle Clustering

It was proven in [4] that the cycle clustering problem is $\mathcal{NP}$-hard. In this section, we give a more detailed proof using the same reduction technique from the *multiway cut* problem, which is $\mathcal{NP}$-hard for any fixed $m \geq 3$ [10].

Definition 3 (multiway cut)**.** *Let $G = (V, A)$ be an undirected graph with non-negative edge weights $c(a) \geq 0$ for all $a \in A$. Let $T = \{t_1, \ldots, t_m\} \subseteq V$ be a subset of specified vertices, called terminals. A multiway cut is a subset of edges $A' \subseteq A$ that separates the terminals $t_1, \ldots, t_m$ in the sense that there exists no path from any terminal to any other terminal in $(V, A \setminus A')$. The multiway cut problem is finding a weight-minimal multiway cut.*

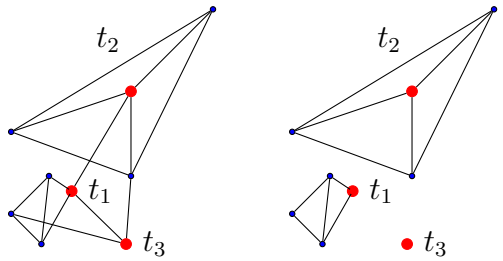


FIGURE 2.2: Illustration of the multiway cut problem. On the left is a graph with three terminals t_1, t_2, t_3. On the right side a subset of edges A' was removed such that all terminals are separated. Therefore A' forms a multiway cut

Theorem 4. *The cycle clustering problem defined in Section 2.1 is $\mathcal{NP}$-hard for any number of clusters $m \geq 3$ and $0 < \alpha < 1$.*

Proof. We consider an instance of the multiway cut problem, i.e., an undirected graph $G = (V, A)$ with a set of nodes $V = \{1, \dots, n\}$, and a set of edges $A \subseteq \binom{V}{2}$ with edge weights $c(a) \geq 0$ for all $a \in A$. Let $T = \{t_1, \dots, t_m\} \subseteq V$ be the set of terminals with $m \geq 3$. We construct an instance of the cycle clustering problem whose optimal solution corresponds to a weight-minimal multiway cut.

By definition, any edge $a = \{t_{i_1}, t_{i_2}\}$ between two terminal nodes has to be in every multiway cut $A' \subseteq A$, or the terminals would not be separated. Therefore, the weight-sum over all edges connecting terminals provides a constant offset to the objective value. W.l.o.g. let $A \cap \binom{T}{2} = \emptyset$.

Furthermore, we may assume that every non-terminal node has at least one edge of positive weight connected to it, i.e., $\sum_{v \in V \colon \{u,v\} \in A} c\{u, v\} > 0$ for all $u \in V \setminus T$. If this is not the case, the node can be assigned arbitrarily.

We construct a cycle clustering instance with a set of states $\mathcal{S} = V$ and m clusters, where we define the probabilities in such a way that

- the net flow forces the terminals into different clusters,
- the coherence represents the objective of the multiway cut,
- the weights are scaled so that the resulting transition matrix P is stochastic.

Set for all $u, v \in V$

$$d_{u,v} = \begin{cases} M, & \text{if there exists } t_i \in T : u = t_i, v = t_{\phi(i)}, \\ c\{u, v\}/2, & \text{if } \{u, v\} \in A, \\ 0, & \text{else.} \end{cases}$$

The constant M is chosen sufficiently large such that each terminal is forced into a different cluster. We will specify M later in the proof.

These weights are scaled in order to construct a stochastic matrix $P \in \mathbb{R}^{n \times n}$. Denote the row-sum for one state u by $\|D_{u\cdot}\|_1 = \sum_{u' \in V} d(u, u')$. Let the entries of P be defined as

$$p_{u,v} = \frac{d_{u,v}}{\|D_{u\cdot}\|_1} \qquad \text{for all } u, v \in V.$$

This transition matrix P has a stationary distribution $\pi \in \mathbb{R}^n$ given by

$$\pi_u = \frac{\|D_{u\cdot}\|_1}{\sum_{u' \in V} \|D_{u'\cdot}\|_1} \qquad \text{for all } u \in V.$$

P is stochastic and the sum over all entries of π is one, since

$$\sum_{v \in V} p_{u,v} = \sum_{v \in V} \frac{d_{u,v}}{\|D_{u\cdot}\|_1} = 1,$$

$$\sum_{v \in V} \pi_v = \sum_{v \in V} \frac{\|D_{v\cdot}\|_1}{\sum_{u' \in V} \|D_{u'\cdot}\|_1} = 1.$$

To prove that π is a stationary distribution, we need to show $\pi^T P = \pi^T$. Denote the v-th column of P by P_v, then

$$\begin{aligned}\pi^T P_v &= \sum_{u \in V} \pi_u p_{u,v} = \sum_{u \in V} \frac{\|D_{u\cdot}\|_1}{\sum_{u' \in V} \|D_{u'\cdot}\|_1} \frac{d_{u,v}}{\|D_{u\cdot}\|_1} \\ &= \sum_{u \in V} \frac{d_{u,v}}{\sum_{u' \in V} \|D_{u'\cdot}\|_1} = \sum_{u \in V} \frac{d_{v,u}}{\sum_{u' \in V} \|D_{u'\cdot}\|_1} = \pi_v.\end{aligned}$$

In the last equation, we exploited the fact that $\sum_{u \in V} d_{v,u} = \sum_{u \in V} d_{u,v}$. For all $v \notin T$, this holds because $d_{u,v} = d_{v,u}$ for all $(u, v) \in (V \times V) \setminus (T \times T)$.

If $v = t_i \in T$, then

$$d_{t_i,t_j} = \begin{cases} M, \text{ if } j = \phi(i), \\ 0, \text{ else.} \end{cases}$$

Therefore, it holds that

$$\begin{aligned}\sum_{u \in V} d_{t_i,u} &= \sum_{u \in V \setminus T} d_{t_i,u} + \sum_{t_j \in T} d_{t_i,t_j} = \sum_{u \in V \setminus T} d_{u,t_i} + d_{t_i,t_{\phi(i)}} \\ &= \sum_{u \in V \setminus T} d_{u,t_i} + d_{t_{\phi^{-1}(i)},t_i} = \sum_{u \in V \setminus T} d_{u,t_i} + \sum_{t_j \in T} d_{t_j,t_i} = \sum_{u \in V} d_{u,t_i}.\end{aligned}$$

For $u, v \in V$, the corresponding matrix Q of unconditional transition probabilities has entries

$$q_{u,v} = \pi_u p_{u,v} = \frac{\|D_{u\cdot}\|_1}{\sum_{u' \in V} \|D_{u'\cdot}\|_1} \frac{d_{u,v}}{\|D_{u\cdot}\|_1} = \frac{d_{u,v}}{\sum_{u' \in V} \|D_{u'\cdot}\|_1}.$$

Thus, only the terminals contribute to the net flow, since

$$q_{u,v} - q_{v,u} = \begin{cases} \frac{M}{\sum_{u \in v} \|D(u)\|_1}, & \text{for } u = t_i, v = t_{\phi(i)}, \\ -\frac{M}{\sum_{u \in v} \|D(u)\|_1}, & \text{for } v = t_{\phi(i)}, u = t_i, \\ 0, & \text{otherwise.} \end{cases}$$

We set

$$M > \frac{\alpha \sum_{a \in A} c(a)}{1 - \alpha},$$

and prove that any optimal solution to the cycle clustering problem cannot have two terminals in the same cluster. Any solution to the cycle clustering problem with $t_i \in C_i$ for all $i = 1, \ldots, m$ has an objective value obj_1 of at least

$$obj_1 \geq \frac{mM}{\sum_{u \in v} \|D(u)\|_1}.$$

Assume that two terminals are in the same cluster in a clustering. Then we can formulate an upper bound on the objective value obj_2 as

$$obj_2 \leq \frac{1}{\sum_{u \in v} \|D(u)\|_1} \left((m-1)M + \alpha(\sum_{a \in A} c(a) + M) \right).$$

Due to the choice of M, it holds that $obj_1 > obj_2$, so the terminals have to be arranged in a cycle in any optimal solution.

Let $(C_1, \ldots, C_m)$ be an optimal solution of the cycle clustering problem w.r.t. the constructed matrix Q.

The assignment of non-terminal nodes does not affect the net flow, as they always have the same forward and backward probability. Therefore, they are assigned in order to maximize coherence. The following calculation shows that maximizing this remaining part of the objective function is equivalent to minimizing the weight of the edges in the corresponding multiway cut.

$$\begin{aligned}
\alpha \sum_{t=1}^{m} g(\mathcal{K}_t) &= \alpha \sum_{t=1}^{m} \sum_{u,v \in \mathcal{K}_t} q_{u,v} = \alpha \sum_{t=1}^{m} \sum_{u,v \in \mathcal{K}_t} \frac{d_{u,v}}{\sum_{u' \in V} \|D_{u\cdot}\|_1} \\
&= \frac{\alpha}{\sum_{u' \in V} \|D_{u\cdot}\|_1} \sum_{t=1}^{m} \sum_{u,v \in \mathcal{K}_t} d_{u,v} = \frac{\alpha}{\sum_{u' \in V} \|D_{u\cdot}\|_1} \sum_{t=1}^{m} \sum_{a \in A \cap \binom{\mathcal{K}_t}{2}} c(a) \\
&= \underbrace{\frac{\alpha}{\sum_{u' \in V} \|D_{u\cdot}\|_1}}_{\text{constant} > 0} \Big(\underbrace{\sum_{a \in A} c(a)}_{\text{constant}} - \underbrace{\sum_{a \in A \setminus \bigcup_{t=1}^{m} \binom{\mathcal{K}_t}{2}} c(e)}_{\text{multiway cut weight}} \Big)
\end{aligned}$$

To summarize, we gave a polynomial reduction of the multiway cut problem to cycle clustering, proving that cycle clustering is $\mathcal{NP}$-hard. □

2.3 MIP-Formulation

We formulate an integer programn in order to solve the cycle clustering problem. First, we present an intuitive formulation as a quadratic binary program and then linearize it.

We denote the index-set of the clusters by $\mathcal{K} \coloneqq \{1, \ldots, m\}$. For each state $i \in \mathcal{S} = \{1, \ldots, n\}$, and each cluster $C_t, t \in \mathcal{K}$, we introduce a binary variable $x_{it} \in \{0, 1\}$ such that

$$x_{it} = 1 \Longleftrightarrow \text{State } i \text{ belongs to cluster } C_t.$$

We also introduce continuous variables f_t for the net flow from C_t to the next cluster $C_{\phi(t)}$, as well as variables g_t for the coherence in cluster C_t.

A straightforward non-linear formulation of the cycle clustering problem is given by

$$
\begin{aligned}
\text{(Bilin)} \qquad & \max \sum_{t\in\mathcal{K}} f_t + \alpha \cdot \sum_{t\in\mathcal{K}} g_t \\
& \text{s.t.} \quad \sum_{t\in\mathcal{K}} x_{it} = 1 && \forall i \in \mathcal{S} && (2.3)\\
& \qquad \sum_{i\in\mathcal{S}} x_{it} \geq 1 && \forall t \in \mathcal{K} && (2.4)\\
& \qquad g_t = \sum_{\substack{i,j\in\mathcal{S}\\ i<j}} (q_{ij}+q_{ji}) x_{it} x_{jt} && \forall t \in \mathcal{K} && (2.5)\\
& \qquad f_t = \sum_{\substack{i,j\in\mathcal{S},\\ i\neq j}} (q_{ij}-q_{ji}) x_{it} x_{j\phi(t)} && \forall t \in \mathcal{K} && (2.6)\\
& \qquad x_{it} \in \{0,1\} && \forall t \in \mathcal{K}, i \in \mathcal{S}\\
& \qquad f_t, g_t \in \mathbb{R}_{\geq 0} && \forall t \in \mathcal{K}.
\end{aligned}
$$

Constraints (2.3) ensure that each state i is assigned to exactly one cluster, while constraints (2.4) ensure that no cluster is empty.

Constraints (2.5) and (2.6) correspond to the coherence and net flow, respectively.

There are numerous possible ways to linearize the above non-linear program. Following [31], we could introduce a binary variable and linearization constraints for each occurring bilinear term.

A more sophisticated and compact method, which exploits the set-partitioning constraints (2.3) is proposed in [27]. This method introduces binary variables for each product $x_{it}x_{jt'}$ and solves a MIP to find the formulation with the least amount of necessary constraints. However, the number of linearization variables is still unnecessarily large with $\mathcal{O}(n^2m^2)$.

Since we know the underlying problem, only three cases need to be distinguished for each pair of states. Either i and j are in the same cluster or they are in consecutive clusters or they are more than one cluster apart. All of these cases can be covered with a fixed number of variables.

In fact, if there exist pairs of states $i, j \in \mathcal{S}$ with transition probability equal to zero, then we do not need extra variables for this pair. Therefore, we define the set of *relevant transitions*

$$E = \{(i,j) \in \mathcal{S}\times\mathcal{S} \mid i \neq j, q_{ij}+q_{ji} > 0\}. \tag{2.7}$$

The linearization that is used for our specific problem is to introduce binary variables y_{ij} and z_{ij} with the following meaning.

$$
\begin{aligned}
y_{ij} = 1 \quad &\Leftrightarrow \quad \text{i and j are in the same cluster} && \forall (i,j) \in E, i<j\\
z_{ij} = 1 \quad &\Leftrightarrow \quad \text{i is one cluster before j along the cycle} && \forall (i,j) \in E
\end{aligned}
$$

If $i > j$, we will use y_{ij} as a notation to donate y_{ji} in order to keep our formulation as short as possible. A linear reformulation of the non-linear program (Bilin) is given by

$$\text{(CC-MIP)} \quad \max \sum_{(i,j)\in E} z_{ij}(q_{ij} - q_{ji}) + \alpha \cdot \Big(\sum_{\substack{(i,j)\in E \\ i<j}} y_{ij}(q_{ij} + q_{ji}) \Big)$$

$$s.t. \quad \sum_{t\in\mathcal{K}} x_{it} = 1 \qquad \forall i \in \mathcal{S} \tag{2.8}$$

$$\sum_{i\in\mathcal{S}} x_{it} \geq 1 \qquad \forall t \in \mathcal{K} \tag{2.9}$$

$$x_{it} + x_{jt} - y_{ij} + z_{ij} - x_{j\phi(t)} - x_{i\phi^{-1}(t)} \leq 1 \qquad \forall t \in \mathcal{K},\ (i,j) \in E \tag{2.10}$$

$$x_{it} + x_{j\phi(t)} - z_{ij} + y_{ij} - x_{jt} - x_{i\phi(t)} \leq 1 \qquad \forall t \in \mathcal{K},\ (i,j) \in E \tag{2.11}$$

$$y_{ij} + z_{ij} + z_{ji} \leq 1 \qquad \forall (i,j) \in E,\ i < j \tag{2.12}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in \mathcal{S},\ \forall t \in \mathcal{K}$$

$$y_{ij}, z_{ij}, z_{ji} \in \{0,1\} \qquad \forall (i,j) \in E,\ i < j.$$

The first sum in the objective function represents the net flow between consecutive clusters, while the second sum is the coherence within all clusters.

Constraints (2.10) are best explained by examining several weaker constraints. Let $(i,j) \in E$, $t \in \mathcal{K}$, and consider the constraints

$$x_{it} + x_{jt} - y_{ij} \leq 1, \tag{2.13}$$

$$x_{it} + z_{ij} - x_{j\phi(t)} \leq 1, \tag{2.14}$$

$$x_{jt} + z_{ij} - x_{i\phi^{-1}(t)} \leq 1. \tag{2.15}$$

The reasoning behind (2.13) is that if i and j are in the same cluster, then y_{ij} has to be equal to one. If i is in some cluster t and $z_{ij} = 1$, then (2.14) forces j to be in the next cluster $\phi(t)$. In the same way, if j is in cluster t and $z_{ij} = 1$, then (2.15) ensures that i is in the cluster preceeding t. All of these three cases are covered by (2.10) since y_{ij}, z_{ij} are binary and at most one of the two can be non-zero at the same time.

In the same way, constraints (2.11) cover the functionality of the weaker constraints

$$x_{it} + x_{j\phi(t)} - z_{ij} \leq 1,$$

$$x_{it} + y_{ij} - x_{jt} \leq 1,$$

$$x_{j\phi(t)} + y_{ij} - x_{i\phi(t)} \leq 1.$$

Since (2.10) and (2.11) are all defined for all $t \in \mathcal{K}$ and all $(i,j) \in E$, the following holds.

- If i and j are in the same cluster, then $y_{ij} = 1$. (2.10)

- If i and j are in consecutive clusters, then $z_{ij} = 1$. (2.11)
- If $y_{ij} = 1$, then there has to exist some $t \in \{1, \dots, m\}$ such that $x_{it} = x_{jt} = 1$. (2.11)
- If $z_{ij} = 1$, then there has to exist some $t \in \{1, \dots, m\}$ such that $x_{it} = x_{j\phi(t)} = 1$. (2.10)

Constraints (2.12) ensure that two states cannot be in the same cluster and in consecutive clusters at the same time. All of this taken together ensures that the linear formulation is equivalent to the non-linear one. Given $n \in \mathbb{N}$ states and $m \in \mathbb{N}$ clusters, this formulation uses $\frac{3}{2}|E| + nm$ variables and $(2m + 0.5)|E| + n + m$ constraints.

In the special case that $m = 3$, we can simplify this model significantly as the y-variables become obsolete. If $m = 3$, then $y_{ij} + z_{ij} + z_{ji} = 1$ for all $(i, j) \in E$. So we can express $y_{ij} = 1 - z_{ij} - z_{ji}$. It is also possible to use less constraints for the model if $m = 3$. The following MIP is a simplified version that is equivalent to CC-MIP if $m = 3$.

$$
\begin{aligned}
\max \sum_{(i,j)\in E} z_{ij}(q_{ij} - q_{ji}) + \alpha \cdot \Big(\sum_{\substack{(i,j)\in E \\ i<j}} (1 - z_{ij} - z_{ji})\,(q_{ij} + q_{ji}) \Big) & \\
s.t. \quad \sum_{t\in\mathcal{K}} x_{it} = 1 \quad & \forall i \in \mathcal{S} \\
\sum_{i\in\mathcal{S}} x_{it} \geq 1 \quad & \forall t \in \mathcal{K} \\
x_{it} + x_{j\phi(t)} - z_{ij} + z_{ji} - x_{j\phi^{-1}(t)} - x_{i\phi^2(t)} \leq 1 \quad & \forall t \in \mathcal{K},\ (i,j) \in E \qquad (2.16) \\
z_{ij} + z_{ji} \leq 1 \quad & \forall (i,j) \in E,\ i < j \\
x_{it} \in \{0,1\} \quad & \forall i \in \mathcal{S},\ \forall t \in \mathcal{K} \\
z_{ij} \in \{0,1\} \quad & \forall (i,j) \in E
\end{aligned}
$$

As before, we illustrate the reasoning behind constraints (2.16) by pointing out weaker constraints. The purpose of constraints

$$
\begin{aligned}
x_{it} + x_{j\phi(t)} - z_{ij} \leq 1, \\
x_{it} + z_{ji} - x_{j\phi^{-1}(t)} \leq 1, \\
x_{j\phi(t)} + z_{ji} - x_{i\phi^2(t)} \leq 1
\end{aligned}
$$

are all covered by (2.16). In the special case of $m = 3$, the MIP formulation uses $|E| + nm$ variables and $(m + 0.5)|E| + n + m$ constraints.

2.4 Polytopal Aspects

In this section, we consider the polytope defined by the CC-MIP presented in Section 2.3. Throughout the section, let $\mathcal{S} = \{1, \dots, n\}$ be the number of states, $m \in \mathbb{N}, m \geq 3$ be the number of clusters, and let $E \subset \mathcal{S} \times \mathcal{S}$

be the edge-set of relevant transitions, as defined in (2.7). We define the *cycle clustering polytope* (CCP) as the convex hull of all incident vectors of clusterings.

$$CCP \coloneqq \operatorname{conv}\left(\{(x,y,z) \in \mathbb{R}^{nm+1.5|E|} \mid (x,y,z) \text{ is a cycle clustering}\}\right)$$

First, we introduce some notation to facilitate the discussion of this polytope. We reference the components of a given vector $(x,y,z) \in \mathbb{R}^{nm+1.5|E|}$ as in previous sections with x_{it}, y_{ij}, and z_{ij}. We introduce characteristic vectors for these components as

$$X_{it} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix} \leftarrow \text{ component } it \quad .$$

Analogously, we define characteristic vectors for all the y-and z-variables, denoted by Y and Z, respectively. For any $A \subseteq \mathcal{S}$, the characteristic vectors are defined as

$$\begin{aligned} Y_{i,A} &\coloneqq \sum_{j \in A} Y_{ij}, \\ Z_{i,A} &\coloneqq \sum_{j \in A} Z_{ij}, \\ Z_{A,i} &\coloneqq \sum_{j \in A} Z_{ji}. \end{aligned}$$

2.4.1 Dimension of the Polytope

We recall the dimension of a polytope as defined in [30].

Definition 5. *A polytope P is of dimension k, denoted by $\dim(P) = k$, if the maximum number of affinely independant points in P is $k+1$.*

A different formulation uses the affine hull of a polyope.

Definition 6. *Let $x_1, \ldots, x_n \in \mathbb{R}^d$. An affine combination of $x_1, \ldots, x_n$ is defined as a linear combination $\sum_{i=1}^n \lambda_i x_i$ such that $\sum_{i=1}^n \lambda_i = 1$. The affine hull of a set is defined the set of all affine combinations of points within that set.*

Lemma 7. *The dimension of a polytope P is the dimension of its affine hull* $\operatorname{aff}(P)$.

Proof. Since $P \subset \operatorname{aff}(P)$, it is clear that $\dim(P) \leq \dim(\operatorname{aff}(P))$. Assume that $\dim(P) < \dim(\operatorname{aff}(P))$. Denote by $\delta \coloneqq \dim(P)$. Then there exist at least $\delta + 2$ affinely independant points in $\operatorname{aff}(P)$. Each of those points can be described as an affine combination of $\delta + 1$ points in P. This is a contradiction. □

The dimension of the cycle clustering polytope CCP is described in the following theorem.

Theorem 8. *Let $G = (V, E)$ be an undirected graph with $V = \{1, \dots, n\}$ and let $3 \leq m \leq n-2$ be the number of clusters. Then the dimension of the corresponding cycle clustering polytope is*

$$\begin{aligned} &(m-1)n + 1.5|E|, && \text{if } m \geq 4, \\ &(m-1)n + |E|, && \text{if } m = 3. \end{aligned}$$

Proof. We assume that $m \geq 4$ and discuss the special case $m = 3$ later. A brief outline of the proof can be described as follows.

We construct a finite set $M \subset \mathbb{R}^{nm+1.5|E|}$ such that for every $p \in CCP$ it holds that $\operatorname{span}(M) + p \subseteq \operatorname{aff}(CCP)$. It immediately follows that $\dim(CCP) \geq \dim(\operatorname{span}(M))$.

Then we show that $CCP \subseteq \operatorname{span}(M) + p$ for any $p \in CCP$ and therefore $\dim(CCP) \leq \dim(\operatorname{span}(M))$.

Construction of M: We construct M in such a way that for every $m \in M$ there exist points $p_1, \dots, p_k \in CCP$ and $p'_1, \dots, p'_k \in CCP$ such that $m = \sum_{i=1}^{k} \lambda_i (p_i - p'_i)$. In other words, every $m \in M$ is a linear combination of differences of points in CCP. That means that for every $p \in CCP$, $p + m$ is an affine combination of elements in CCP.

Let $i \in \mathcal{S}, t \in \{2, \dots, m\}$. First, we set $(X_{i1} - X_{it}) \in M$ and show that the difference condition for M is fulfilled. W.l.o.g. we assume that $t = 2$.

Choose any clustering $(A_1, \dots, A_m)$ of $\mathcal{S} \setminus \{i\}$. Let $s \in \mathcal{K}$ be arbitrary and define A_s as the first cluster of the cycle, i.e., consider the incidence vectors of the two clusterings

$$\begin{aligned} P_s &= \left(A_s \cup \{i\}, A_{\phi(s)}, \dots A_{\phi^{-1}(s)}\right), \\ P'_s &= \left(A_s, A_{\phi(s)} \cup \{i\}, \dots A_{\phi^{-1}(s)}\right). \end{aligned}$$

The only difference between P_s and P'_s is in the assignment of i.

We denote the incidence vectors of P_s and P'_s by $(x, y, z)^{P_s}$ and $(x, y, z)^{P'_s}$, respectively. Subtracting one from the other yields

$$\begin{aligned} (x, y, z)^{P_s} - (x, y, z)^{P'_s} &= X_{i1} - X_{i2} && (2.17) \\ &\quad + Y_{i,A_s} - Y_{i,A_{\phi(s)}} && (2.18) \\ &\quad + Z_{i,A_{\phi(s)}} + Z_{A_{\phi^{-1}(s)},i} - Z_{A_s,i} - Z_{i,A_{\phi(\phi(s))}}. && (2.19) \end{aligned}$$

Both incidence vectors have the same x-part, except in x_{i1} and x_{i2}, so (2.17) is clear. In the y-parts, P_s and P'_s differ in the interaction with A_s and $A_{\phi(s)}$, which yields (2.18). For the z-parts, both the previous as well as the successive cluster is relevant, resulting in the term (2.19).

As s is arbitrary, we can take the sum over all $s = 1, \dots, m$. Then the Y and the Z parts each sum up to zero and we have shown that

$$\frac{1}{m} \sum_{s=1}^{m} (x, y, z)^{P_s} - (x, y, z)^{P'_s} = X_{i1} - X_{it}.$$

These $(m-1)n$ vectors are linearly independent as $X_{i1} - X_{it}$ has a non-zero entry in the component x_{it}. Therefore

$$\dim\left(\operatorname{span}\left(\{X_{i1} - X_{it} \mid 2 \leq t \leq m\}\right)\right) = (m-1)n. \tag{2.20}$$

Let $(i,j) \in E$ be any transition. We now set $Y_{ij}, Z_{ij}, Z_{ji} \in M$ and show that these vectors fulfill the difference condition for M.

Take any $k \neq l \in \mathcal{S} \setminus \{i,j\}$ and consider the clusterings

$$\begin{aligned}
P_1 &= (\{i,j\}, \{k\}, A_1, \ldots, A_{m-2}),\\
P_1' &= (\{i\}, \{j,k\}, A_1, \ldots, A_{m-2}),\\
P_2 &= (\{i,j,l\}, \{k\}, B_1, \ldots, B_{m-2}),\\
P_2' &= (\{i,l\}, \{j,k\}, B_1, \ldots, B_{m-2}),
\end{aligned}$$

where $(A_1, \ldots, A_{m-2})$ is an arbitrary partition of $\mathcal{S} \setminus \{i,j,k\}$ into $m-2$ clusters and $(B_1, \ldots, B_{m-2})$ is an arbitrary partition of $\mathcal{S} \setminus \{i,j,k,l\}$ into $m-2$ clusters. An illustration of this idea is Figure 2.3.

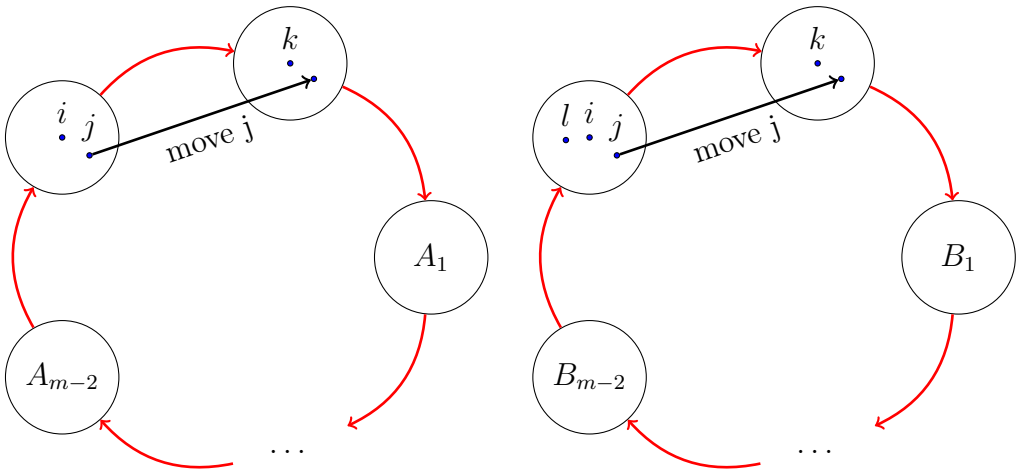


FIGURE 2.3: Illustration of the idea behind the proof. On the left side is the change from P_1 to P_1' and on the right P_2 to P_2'.

In both P_1, P_1' as well as P_2, P_2' we move j from the first to the second cluster. The only difference is that the first cluster contains one more element in the case of P_2.

Similar as before, we consider the difference between their incidence vectors.

$$\begin{aligned}(x,y,z)^{P_1} - (x,y,z)^{P_1'} &= X_{j1} - X_{j2} \\ &\quad + Y_{ij} - Y_{jk} \\ &\quad + Z_{A_{m-2},j} + Z_{jk} - Z_{j,A_1} - Z_{ij} \\ (x,y,z)^{P_2} - (x,y,z)^{P_2'} &= X_{j1} - X_{j2} \\ &\quad + Y_{ij} + Y_{jl} - Y_{jk} \\ &\quad + Z_{B_{m-2,j}} + Z_{jk} - Z_{j,B_1} - Z_{ij} - Z_{lj}\end{aligned}$$

$A_1, \ldots, A_{m-2}$ and $B_1, \ldots, B_{m-2}$ were arbitrary up to this point. Now, we choose $A_{m-2} = B_{m-2}, l \in A_1, B_1 = A_1 \setminus \{l\}$ and get

$$\left((x,y,z)^{P_2} - (x,y,z)^{P_2'}\right) - \left((x,y,z)^{P_1} - (x,y,z)^{P_1'}\right) = Y_{jl} + Z_{jl} - Z_{lj}.$$

Exploiting the symmetry in the y-Variables, we switch the role of j and l. This yields the vector

$$Y_{jl} + Z_{lj} - Z_{jl}.$$

If we add both of them, we have shown that Y_{jl} can be constructed as a linear combination of differences of points in CCP.

Conversely, if we set $l \in A_{m-2}, B_{m-2} = A_{m-2} \setminus \{l\}, A_1 = B_1$ and do the same we get the vector

$$Y_{jl} - 2Z_{lj}.$$

Since we have already proven that Y_{jl} can be put in M, it immediately follows that Z_{lj} and Z_{jl} can also be constructed as a linear combination of differences of points in CCP.

To summarize, we have constructed M such that the condition $p + \operatorname{span}(M) \subseteq \operatorname{aff}(CCP)$ is satisfied and M consists of the vectors

$$\begin{aligned} X_{i1} - X_{it} &\quad \forall i \in \mathcal{S}, t \in \{2, \ldots, m\}, \\ Y_{ij} &\quad \forall (i,j) \in E, i < j, \\ Z_{ij} &\quad \forall (i,j) \in E. \end{aligned}$$

The dimension of the linear space spanned by these vectors is

$$\dim(\operatorname{span}(M)) = (m-1)n + 1.5|E|.$$

As explained at the beginning of this proof, it follows that

$$\dim(CCP) \geq (m-1)n + 1.5|E|.$$

$CCP \subset \operatorname{span}(M) + P$: Take any two clusterings P_1, P_2 of $\mathcal{S}$. Then we can express P_2 as a linear combination of $P_1, X_{i1} - X_{it}, Y_{ij}$, and Z_{ij}. These are $1 + (m-1)n + 1.5|E|$ affinely independent vectors.

This proves that $\dim(CCP) \leq (m-1)n + 1.5|E|$ and we have shown that $\dim(CCP) = (m-1)n + 1.5|E|$.

In the case of $m = 3$, we can omit the y-variables, since

$$y_{ij} = 1 - z_{ij} - z_{ji}.$$

Other than that, we can do the same proof as for $m \geq 4$, with the difference that $A_1, \ldots, A_{m-2}$ consists of only one set A'_1, and therefore $B'_1 = A'_1 \setminus \{l\}$. Subtracting the incidence vectors as before yields

$$\left((x,y,z)^{P_2} - (x,y,z)^{P'_2}\right) - \left((x,y,z)^{P_1} - (x,y,z)^{P'_1}\right) = 2Z_{lj} - Z_{jl}.$$

We switch the roles of j, l and get

$$2Z_{jl} - Z_{lj}$$

as a linear combination of elements in CCP. By combining these, it follows that

$$2\left(2Z_{lj} - Z_{jl}\right) + 2Z_{jl} - Z_{lj} = 3Z_{lj}.$$

So with the same argumentation as before we have shown that the dimension of CCP is $(m-1)n + |E|$, in case $m = 3$. □

2.4.2 Facets of the Problem Formulation

We recall the definition of a facet, as defined in [30]. Let $ax+by+cz \leq \delta$ be a *valid inequality* for CCP, i.e., the inequality is satisfied by all $(x, y, z) \in CCP$. Then a *face* of CCP is the set

$$F = \{(x, y, z) \in CCP \mid ax + by + cz = \delta\}.$$

We call F *proper*, if $F \neq \emptyset$ and $F \neq CCP$.

A *facet* is defined as a face with $\dim(F) = \dim(CCP) - 1$. Therefore a facet is a maximal proper face.

Similar to the notation in the previous section, we use the notation a_{it} for the component of the a-vector belonging to x_{it} and the same for b, c. Additionally, for $A \subseteq \mathcal{S}$ we define

$$b_{i,A} \coloneqq \sum_{j \in A} b_{ij},$$
$$c_{i,A} \coloneqq \sum_{j \in A} c_{ij},$$
$$c_{A,i} \coloneqq \sum_{j \in A} c_{ji}.$$

The following theorem addresses which variable bounds are facet-defining.

Theorem 9. *Let $i, j \in \mathcal{S}, t \in \mathcal{K}$. Then the following holds for the variable bounds.*

- *The bounds* $x_{it} \leq 1, y_{ij} \leq 1, z_{ij} \leq 1$ *do not define facets.*
- $x_{it} \geq 0$ *defines a facet.*
- $y_{ij} \geq 0, z_{ij} \geq 0$ *defines a facet.*

Proof. First we will prove that the upper bounds never define a facet, as

$$\{(x,y,z) \mid x_{it} = 1\} \subsetneq \{(x,y,z) \mid x_{i\phi(t)} = 0\},$$
$$\{(x,y,z) \mid y_{ij} = 1\} \subsetneq \{(x,y,z) \mid z_{ij} = 0\},$$
$$\{(x,y,z) \mid z_{ij} = 1\} \subsetneq \{(x,y,z) \mid z_{ji} = 0\}.$$

For the lower bounds, consider $F \coloneqq \{(x,y,z) \mid x_{it} = 0\}$. Using the same steps as in the proof of Theorem 8, we can prove that the dimension of F is exactly one less than the dimension of CCP. The vector that is missing is $X_{i1} - X_{it}$. This proves that F is a facet of CCP. The same reasoning can be applied to the lower bounds of the y, z-variables. □

For the other model inequalities we need a more sophisticated approach. First we illustrate the general strategy that is used in the proofs.

Let $ax + by + cz \leq \delta$ be a valid inequality for CCP. A facet is defined as a maximal face, i.e., there can exist no other proper face that contains the facet. Thus, in order to prove that $ax + by + cz \leq \delta$ is facet-defining, we will assume that there exists a facet-defining inequality $\hat{a}x + \hat{b}y + \hat{c}z \leq \hat{\delta}$ such that

$$\{(x,y,z) \in CCP \mid ax+by+cz = \delta\} \subseteq \{(x,y,z) \in CCP \mid \hat{a}x+\hat{b}y+\hat{c}z = \hat{\delta}\}. \tag{2.21}$$

If we can prove that the inequalities have to be scalar multiple of each other, it follows that $ax + by + cz \leq \delta$ is facet-defining. To illustrate the usefulness of this approach, consider a clustering

$$P = (A_1, \ldots, A_m) \tag{2.22}$$

whose incidence vector (x, y, z) fulfills $ax + by + cz = \delta$. Then, by (2.21) it follows that $\hat{a}x + \hat{b}y + \hat{c}z = \hat{\delta}$. Let (x', y', z') be a different incidence vector that also satisfies $ax' + by' + cz' = \delta$. Then we can compare the coefficients for both of the incidence vectors because

$$\hat{a}x + \hat{b}y + \hat{c}z = \hat{a}x' + \hat{b}y' + \hat{c}z'. \tag{2.23}$$

If the incidence vectors are similar to each other, most of the coefficients will cancel each other out and we will be able to make statements about individual coefficients.

We formulate two lemmas which are used in the proofs of facet-defining inequalities. We will only prove the first, the second proof follows analogously.

Lemma 10. *Let $m \geq 4$ and $t \in \mathcal{C} \setminus \{2, m\}$. Let $ax + by + cz \leq \delta$ be a valid inequality for CCP that is satisfied at equality by the clusterings*

$$\begin{aligned} P_1 &= (A_1 \cup \{l\}, \ldots, A_t, \ldots, A_m), & P_1' &= (A_1, \ldots, A_t \cup \{l\}, \ldots, A_m), \\ P_2 &= (A_t \cup \{l\}, \ldots, A_1, \ldots, A_m), & P_2' &= (A_t, \ldots, A_1 \cup \{l\}, \ldots, A_m). \end{aligned}$$

Then

$$b_{l,A_1} = b_{l,A_t}.$$

Proof. If we put in the incidence vectors for P_1, P_1', following (2.23) we get

$$a_{l1} + b_{l,A_1} + c_{l,A_2} + c_{A_m,l} = a_{lt} + b_{l,A_t} + c_{l,A_{\phi(t)}} + c_{A_{\phi^{-1}(t)},l},$$

and doing the same for P_2, P_2' yields

$$a_{l1} + b_{l,A_t} + c_{l,A_2} + c_{A_m,l} = a_{lt} + b_{l,A_1} + c_{l,A_{\phi(t)}} + c_{A_{\phi^{-1}(t)},l}.$$

If we subtract both of these equations, everything except the b-parts is zero and the claim follows immediately. □

The intuition behind this lemma is that l has the same influence on the net flow for both P_1 and P_2. In both cases, we move l to a different cluster and in the new clustering l has the same influence on the net flow in P_1' and P_2'.

We have done the same exchange two times, both times with the same net flow. Therefore, the only thing that changes is the part concerning the coherence for l. The second lemma makes the connection between the net flow coefficients and the coherence coefficients.

Lemma 11. *Let $m \geq 4$ and $ax + by + cz \leq \delta$ be a valid inequality that is satisfied at equality by the clusterings*

$$\begin{aligned} P_1 &= (A_1 \cup \{l\}, A_2, A_3, \ldots, A_m), & P_1' &= (A_1, A_2 \cup \{l\}, A_3, \ldots, A_m), \\ P_2 &= (A_1 \cup \{l\}, A_3, A_2, \ldots, A_m), & P_2' &= (A_1, A_3 \cup \{l\}, A_2, \ldots, A_m). \end{aligned}$$

Then

$$2c_{l,A_2} + b_{l,A_3} = 2c_{l,A_3} + b_{l,A_2}. \tag{2.24}$$

Using these lemmas, we can prove that the other model-inequalities define facets. The lemmas are also used later in Chapter 3 and in Appendix A.

Theorem 12. *Let $(i, j) \in E$, $t \in \mathcal{C}$, and $m \geq 4$. Then the model inequalities (2.10) and (2.11), i.e.,*

$$\begin{aligned} x_{it} + x_{jt} - y_{ij} + z_{ij} - x_{j\phi(t)} - x_{i\phi^{-1}(t)} &\leq 1, \\ x_{it} + x_{j\phi(t)} - z_{ij} + y_{ij} - x_{jt} - x_{i\phi(t)} &\leq 1 \end{aligned} \tag{2.25}$$

define facets of CCP.

Proof. We will only prove that the first of the inequalities defines a facet, the proof for the second can be done analogously. W.l.o.g. we assume that $t = 1$ and define

$$F \coloneqq \{(x, y, z) \in CCP \mid (x, y, z) \text{ satisfies (2.25) at equality }\}. \tag{2.26}$$

First off, we have already explained why the inequality is valid for CCP when we formulated the MIP. Furthermore, it is easy to see that it is indeed a proper face. The incidence vector of any clustering with i, j in the cluster t fulfills the inequality at equality, so $F \neq \emptyset$. The incidence of any clustering with i and j in cluster $\phi(t)$ is not in F, so $F \neq CCP$.

Step 0: Strategy

Assume there exists a valid inequality $ax + by + cz \leq \alpha$ that defines a proper face of CCP such that

$$F \subseteq \{(x, y, z) \in CCP \mid ax + by + cz = \alpha\}. \tag{2.27}$$

Following the strategy outlined at the start of this section, we will consider all coefficients of this inequality and prove that

$$F = \{(x, y, z) \in CCP \mid ax + by + cz = \alpha\}.$$

Step 1: y-variables

First, we will prove that $b_{lu} = 0$, for all $l, u \in \mathcal{S} \setminus \{i, j\}$.

Let $l \in \mathcal{S} \setminus \{i, j\}$ and $(A_1, \ldots, A_m)$ be any clustering of $\mathcal{S} \setminus \{l\}$ with states $i \in A_1, j, u \in A_3$. We set the clusterings as in Lemma 10:

$$\begin{aligned} P_1 &= (A_1 \cup \{l\}, A_2, A_3, \ldots, A_m), & P_1' &= (A_1, A_2, A_3 \cup \{l\}, \ldots, A_m), \\ P_2 &= (A_3 \cup \{l\}, A_2, A_1, \ldots, A_m), & P_2' &= (A_3, A_2, A_1 \cup \{l\}, \ldots, A_m). \end{aligned}$$

All of these clusterings satisfy (2.25) at equality since $i \in A_1$ and therefore

$$b_{l,A_1} = b_{l,A_3}.$$

We can do the same thing again with u in A_2 instead of A_3, i.e., we apply Lemma 10 to

$$\begin{aligned} P_1 &= (A_1 \cup \{l\}, A_2 \cup \{u\}, A_3 \setminus \{u\}, \ldots, A_m), \\ P_2 &= (A_3 \setminus \{u\} \cup \{l\}, A_2 \cup \{u\}, A_1, \ldots, A_m), \end{aligned}$$

and P_1', P_2' accordingly. It follows that

$$b_{l,A_1} = b_{l,A_3 \setminus \{u\}}.$$

Thus, as both equalities have the same left hand side

$$b_{l,A_3 \setminus \{u\}} = b_{l,A_3} \quad \Rightarrow \quad b_{lu} = 0 \quad \forall l, u \notin \{i, j\}.$$

We will use this trick often i.e., applying the same lemma twice and just switching one state into a cluster that does not appear in the lemma, and

therefore call it switch-trick.

Next, we prove that $b_{lj} = 0$ for $l \in \mathcal{S} \setminus \{i\}$. We consider the clusterings

$$P_1 = (\{i,j,l\}, A_2, A_3, \dots, A_m), \qquad P_1' = (\{i,j\}, A_2, A_3 \cup \{l\}, \dots, A_m),$$
$$P_2 = (\{i,l\}, A_2, A_3 \cup \{j\}, \dots, A_m), \qquad P_2' = (\{i\}, A_2, A_3 \cup \{j,l\}, \dots, A_m).$$

If we compare the incidence vectors of these, applied to $ax + by + cz = \delta$, then we get from P_1, P_1'

$$a_{l1} + b_{il} + b_{jl} + c_{l,A_2} + c_{A_m,l} = a_{l3} + c_{l,A_4} + c_{A_2,l},$$

and from P_2, P_2'

$$a_{l1} + b_{il} + c_{l,A_2} + c_{A_m,l} = a_{l3} + b_{jl} + c_{l,A_4} + c_{A_2,l}.$$

Substracting both equalities yields $b_{jl} = 0$. Denote $\beta = b_{ij}$, then we have proven so far that the inequality $ax + by + cz \leq \alpha$ is of the form

$$ax + \beta y_{ij} + cz \leq \alpha.$$

Step 2: x, z-variables:

In the same way as for the y-variables, we can use Lemma 11 and the switch-trick to prove that

$$c_{lu} = 0 \qquad \text{for all } l, u \in \mathcal{S} \setminus \{i,j\}.$$

Let $l \in \mathcal{S} \setminus \{i,j\}$. Consider the clusterings

$$\begin{aligned} P_1 &= (\{i,j,l\}, A_2, A_3, \dots, A_m), \\ P_2 &= (\{i,j\}, A_2 \cup \{l\}, A_3, \dots, A_m), \\ &\dots \\ P_m &= (\{i,j\}, A_2, A_3, \dots, A_m \cup \{l\}). \end{aligned}$$

In each of these, i, j are both in the first cluster and l is rotated through all other clusters. Because i and j are in the first cluster, (2.25) is satisfied at equality and from the incidence vectors we can derive the equalities

$$a_{l1} = a_{l2} + c_{il} + c_{jl} = a_{l3} = \dots = a_{lm} + c_{li} + c_{lj}.$$

If we do the same thing, i.e., rotating l through all the clusters but this time with i in the first and j in the second cluster, we get

$$a_{l1} + c_{lj} = a_{l2} + c_{il} = a_{l3} + c_{jl} = \dots = a_{lm} + c_{li}.$$

If we subtract these two equality-chains, we get

$$-c_{lj} = c_{jl} = -c_{jl} = c_{lj} = 0.$$

Switching the role of i and j, it follows that $c_{li} = c_{il} = 0$.

Furthermore, it follows that $a_{lt} := a_l$ is constant for all $t \in \mathcal{K}$. Since any incidence vector of a clustering in CPP has to fulfill (2.3), i.e.,

$$\sum_{t=1}^{m} x_{it} = 1,$$

we can assume $a_{lt} = 0$. To prove this, consider any incidence vector (x, y, z) applied to the inequality, i.e.,

$$\begin{aligned} ax + \beta y_{ij} + cz &= \sum_{l=1}^{n}\sum_{t=1}^{m} a_{lt}x_{lt} + \beta y_{ij} + cz \\ &= \sum_{l=1,l\neq i,j}^{n} a_l + \sum_{t=1}^{m}(a_{it}x_{it} + a_{jt}x_{jt}) + \beta y_{ij} + cz \leq \delta \end{aligned}$$

We can include the constant $\sum_{l=1,l\neq i,j}^{n} a_l$ in the right hand side.

To summarize, we have proven so far that the inequality $ax+by+cz \leq \delta$ has to be of the form

$$\sum_{t=1}^{m}(a_{it}x_{it} + a_{jt}x_{jt}) + b_{ij}y_{ij} + c_{ij}z_{ij} + c_{ji}z_{ji} \leq \delta.$$

In order to prove that the remaining coefficients are as claimed, we examine all of the different clusterings that satisfy (2.25) at equality. We denote by $(A_1, \ldots, A_m)$ a clustering of $\mathcal{S}$. Then all of the following clusterings satisfy (2.25) at equality:

a) $i \in A_1, j \in A_t, t = 1, \ldots, m,$

b) $i \in A_t, j \in A_1, t = 1, \ldots, m,$

c) $i \in A_t, j \in A_{\phi(t)}, t = 1, \ldots, m.$

As before, we can assume that $\sum_{t=1}^{m} a_{it} = 0$, due to (2.3). If we compare the incidence vectors for all of the above cases a)-c), we get the system of $3m - 1$ equations

$$\begin{aligned} &a_{i1} + a_{j2} = a_{i2} + a_{j3} = \ldots = a_{im} + a_{j1}, \\ &a_{i1} + b_{ij} = a_{i2} + c_{ji} = a_{i3} = \ldots = a_{im} + c_{ij}, \\ &a_{j1} + b_{ij} = a_{j2} + c_{ij} = a_{j3} = \ldots = a_{jm} + c_{ji}, \\ &a_{i1} + a_{j1} + b_{ij} = a_{i2} + a_{j3} + c_{ij} = \ldots = a_{im-1} + a_{jm}, \ +c_{ij}, \\ &\sum_{t=1}^{m} a_{it} = \sum_{t=1}^{m} a_{jt} = 0. \end{aligned}$$

We can reduce this to the case of $m = 4$, as $a_{i3} = \ldots = a_{im-1}$ and the same holds for j.

Using Gaussian elimination, it can be checked that any solution to this system is of the form $\beta = b_{ij} = -c_{ij} = a_{i1} = a_{j1} = -a_{im} = -a_{j2}$ and the

rest of the coefficients zero. Therefore the inequality is of the form

$$\beta \left(x_{it} + x_{jt} - y_{ij} + z_{ij} - x_{j\phi(t)} - x_{i\phi^{-1}(t)} \right) \leq \delta.$$

It is clear that in order to fulfill (2.27), it has to hold that $\beta = \delta$. If $\beta < 0$, then the inequality is not valid for CCP. If $\beta = 0$, then $F = CCP$ and therefore not a proper face. This concludes the proof. □

In the case of $m = 3$ we use slightly different inequalities. The proof that these define facets follows the same strategy as before.

Theorem 13. *In the case $m = 3$, the reduced model constraints*

$$x_{it} + x_{j\phi(t)} - z_{ij} + z_{ji} - x_{j\phi^{-1}(t)} - x_{i\phi^2(t)} \leq 1 \tag{2.28}$$

are facet-defining.

Proof. Let $F \coloneqq \{(x, y, z) \in CCP \mid (x, y, z) \text{ satisfies (2.28)}\}$ and w.l.o.g. let $t = 1$. As in the proof of Therom 12, assume there exists a valid inequality $ax + by + cz \leq \alpha$ that defines a proper face of CCP such that

$$F \subseteq \{(x, y, z) \in CCP \mid ax + by + cz = \alpha\}. \tag{2.29}$$

Step 1: x-variables

Let $l \in \mathcal{S} \setminus \{i, j\}$ and let (A_1, A_2, A_3) be a clustering with $i, j \in A_1$. Then the clusterings

$$\begin{aligned} P_1 &= (A_1 \cup \{l\}, A_2, A_3), & P_1' &= (A_1, A_2 \cup \{l\}, A_3), \\ P_2 &= (A_3, A_1 \cup \{l\}, A_2), & P_2' &= (A_3, A_1, A_2 \cup \{l\}) \end{aligned}$$

satisfy (2.28) at equality. Comparing the coefficients for P_1, P_1' and P_2, P_2' yields

$$\begin{aligned} a_{l1} + c_{l,A_2} + c_{A_3,l} &= a_{l2} + c_{l,A_3} + c_{A_1,l}, \\ a_{l2} + c_{l,A_2} + c_{A_3,l} &= a_{l3} + c_{l,A_3} + c_{A_1,l}. \end{aligned}$$

It follows that $2a_{l2} = a_{l1} + a_{l3}$. In the same way, by starting with l in A_2 it can be proven that $2a_{l3} = a_{l1} + a_{l2}$. Therefore, we have shown that $a_{l1} = a_{l2} = a_{l3}$ and can assume that $a_{lt} = 0$ for all $l \neq \{i, j\}, t = 1, 2, 3$.

Step 2: z-variables:

Let $v \in \mathcal{S}, l, u \in \mathcal{S} \setminus \{i, j, v\}$ and let $(A_1, \{l, u\}, A_3)$ be a clustering of $\mathcal{S}$ with $i, j, v \in A_1$. Then the clusterings

$$\begin{aligned} P_1 &= (A_1, \{l, u\}, A_2), & P_1' &= (A_1, \{u\}, A_2 \cup \{l\}), \\ P_2 &= (A_1 \setminus \{v\}, \{l, u, v\}, A_2), & P_2' &= (A_1 \setminus \{v\}, \{u, v\}, A_2 \cup \{l\}) \end{aligned}$$

satisfy (2.28) at equality. It follows that

$$\begin{aligned} c_{A_1,l} + c_{l,A_2} &= c_{ul} + c_{l,A_1}, \\ c_{A_1\setminus\{v\},l} + c_{l,A_2} &= c_{ul} + c_{vl} + c_{l,A_1\setminus\{v\}}. \end{aligned}$$

Subtracting the second from the first inequality yields

$$c_{vl} = c_{lv} - c_{vl} \Rightarrow 2c_{vl} = c_{lv}.$$

Doing the same thing again, but moving l to A_1 instead of A_2, i.e., setting

$$P_1' = (A_1 \cup \{l\}, \{u\}, A_2), \qquad P_2' = ((A_1 \cup \{l\}) \setminus \{v\}, \{u, v\}, A_2)$$

yields

$$c_{vl} = 2_{lv}.$$

This proves that $c_{lv} = c_{vl} = 0$ for all $v \in \mathcal{S}, l \in \mathcal{S} \setminus \{i, j, v\}$.
Step 3: i, j

If (A_1, A_2, A_3) is a clustering, then it fulfills (2.28) at equality if $i \in A_1$ or $j \in A_2$ or both. By comparing coefficients, we get the following set of equations.

$$\begin{aligned}
i \in A_1 \text{ move j from } A_2 \text{ to } A_3 &\Rightarrow a_{j2} + c_{ij} = a_{j3} + c_{ji} \\
i \in A_1 \text{ move j from } A_1 \text{ to } A_2 &\Rightarrow a_{j1} = a_{j2} + c_{ij} \\
i \in A_1 \text{ move j from } A_1 \text{ to } A_3 &\Rightarrow a_{j1} = a_{j3} + c_{ji} \\
j \in A_2 \text{ move i from } A_1 \text{ to } A_2 &\Rightarrow a_{i1} + c_{ij} = a_{i2} \\
j \in A_2 \text{ move i from } A_1 \text{ to } A_3 &\Rightarrow a_{i1} + c_{ij} = a_{i3} + c_{ji} \\
j \in A_2 \text{ move i from } A_2 \text{ to } A_3 &\Rightarrow a_{i2} = a_{i3} + c_{ji} \\
j \in A_1 \text{ move i from } A_1 \text{ to } A_2 &\Rightarrow a_{i1} = a_{i2} + c_{ji}
\end{aligned}$$

As in the proof of Theorem 12, we can assume

$$\begin{aligned}
a_{i1} + a_{i2} + a_{i3} = 0, \\
a_{j1} + a_{j2} + a_{j3} = 0.
\end{aligned}$$

This system of linear equations has solutions of the kind

$$\beta = c_{ji} = a_{i1} = a_{j2} = -a_{i3} = -a_{j3} = -c_{ij},$$

with $0 = a_{i2} = a_{j1}$. Therefore, we have proven that $ax + by + cz \leq \delta$ is of the form

$$\beta \left(x_{it} + x_{j\phi(t)} - z_{ij} + z_{ji} - x_{j\phi^{-1}(t)} - x_{i\phi^2(t)} \right) \leq \delta.$$

With the same argument as in the proof of Theorem 12, the only choice of β and δ that produces a valid inequality and a proper face F is $\beta = \delta > 0$. □

2.5 Semidefinite Relaxation of Quadratic Programs

A different approach for graph partitioning problems with the aim of obtaining tighter relaxations is to formulate a semidefinite integer program.

We describe such a formulation for the *minimum k-partitioning problem* (MkP) and then modify it in order to describe the cycle clustering problem.

In the minimum k-partitioning problem, the task is to partition a graph into k disjoint subsets, such that the total weight of edges inside the same set is minimized. A semidefinte programming formulation for this problem is given in [14] as

$$\begin{aligned}
\text{(MkP)} \qquad \min \quad & \sum_{1\le i\neq j\le n} c_{ij}\frac{(k-1)Y_{ij}+1}{k} \\
\text{s.t.} \quad & Y \succeq 0 && && (2.30)\\
& Y_{ii} = 1 && \forall i \in \mathcal{S} && (2.31)\\
& Y_{ij} \in \left\{\frac{-1}{k-1}, 1\right\} && \forall i,j \in \mathcal{S}.
\end{aligned}$$

Here, c_{ij} are the edge weights, constraints (2.30) and (2.31) ensure that Y is positive semidefinite and has diagonal one, respectively. If Y_{ij} is set to one, then i and j are in the same partition and if it is set to $\frac{-1}{k-1}$ then they are in different partitions. In other words, Y_{ij} has the same function as the variables y_{ij} from Section 2.3.

Cycle clustering can bee seen as an extension of MkP. Instead of just considering the edges inside each cluster, we also have to take into account the edges that lie between consecutive clusters along the cycle. We do this by adding new variables to the problem that model the net flow.

Let $\mathcal{S} = \{1, \dots, n\}$ be the set of states, $m \geq 3$ the amount of clusters. Then the cycle clustering problem is modeled by the semidefinite integer program

$$\begin{aligned}
\text{(CC-SDP)} \quad \min \quad & \sum_{1\le i\neq j\le n} (-\alpha q_{ij})\frac{(m-1)Y_{ij}+1}{m} + \sum_{1\le i\neq j\le n} (q_{ij}-q_{ji})\, Z_{ij} \\
\text{s.t.} \quad & Y \succeq 0 && && \\
& Y_{ii} = 1 && \forall i \in \mathcal{S} && \\
& \frac{m}{m-1}Z_{ij} + Y_{ij} \leq 1 && \forall i,j \in \mathcal{S} && (2.32)\\
& Y_{ij} + Z_{ik} - Z_{jk} \leq 1 && \forall i,j,k \in \mathcal{S} && (2.33)\\
& Z_{ij} + Z_{ik} - Y_{jk} \leq 1 + \frac{1}{m-1} && \forall i,j,k \in \mathcal{S} && (2.34)\\
& Y_{ij} \in \left\{\frac{-1}{m-1}, 1\right\} && \forall i,j \in \mathcal{S} && \\
& Z_{ij} \in \{0,1\} && \forall i,j \in \mathcal{S}. &&
\end{aligned}$$

The matrix Y has the same function as in (MkP), while the matrix Z describes which edges are in consecutive clusters. Constraints (2.32), (2.33), and (2.34) couple the variables Y and Z and ensure that a clustering

is created. Constraints (2.33) ensure that if state i and j are in the same cluster and state k is in the successor of the cluster of i then k is also in the successor of the cluster of j. Similarly, constraints (2.34) ensures that if state i is in predecessor of the cluster of j and k, then j and k have to be in the same cluster.

The issue with this formulation is that a semidefinite integer program with $3n^3$ constraints has to be solved. A framework that allows the separation of violated model constraints as well as the generation of additional cutting planes is necessary to tackle this kind of problem. With frameworks for solving integer semidefinite programs such as SCIP-SDP [18] this is not yet possible.

To gain insight if this formulation provides a stronger relaxation for our specific problem, we will compare the strength of the LP-relaxation for the CC-MIP presented in Section 2.3 to the relaxation obtained by omitting the integrality requirements in the above integer semidefinite program.

Chapter 3

Solving Methods for Cycle Clustering

Modeling an optimization problem as a mixed integer program makes it possible to use one of the powerful generic MIP solvers. In addition, the underlying structure of the specific problem can be exploited to extend a generic solver and improve it for this class of problem.

In this chapter, we introduce three problem-specific primal heuristics and three types of valid inequalities that are used for the cycle clustering MIP (CC-MIP), presented in Section 2.3. Throughout the whole chapter, we consider a set of states $\mathcal{S} = \{1, \dots, n\}$, a set of clusters $\mathcal{K} = \{1, \dots, m\}$, and a matrix of transition probabilities $Q \in \mathbb{R}^{n \times n}$. Denote by $E \subset S \times S$ the set of relevant transitions according to (2.7).

3.1 Primal Heuristics

As explained in Section 1.4.2, good incumbent solutions can greatly reduce the overall tree-size in LP-based branch-and-bound. These solutions originate either from feasible LP-solutions or from *primal heuristics*.

Primal heuristics are methods without any success or quality guarantee that aim at constructing solutions for an optimization problem. They can be grouped into the categories start heuristics and improvement heuristics. Start heuristics attempt to find feasible solutions without requiring a solution as input. Improvement heuristics require a feasible solution as input and attempt to construct solutions with a better objective value.

The heuristics presented in this section all exploit the fact that an integer solution $(x, y, z) \in \mathbb{R}^{nm+3|E|}$ for the CC-MIP is completely defined by the x-variables. If the x-variables are known and define a clustering, then the rest of the variables can be inferred by

$$y_{ij} \coloneqq \begin{cases} 1, \text{ if } \max_{k=1,\dots,m}(x_{ik}x_{jk}) = 1 \text{ and } (i,j) \in E, \\ 0, \text{ else} \end{cases} \tag{3.1}$$

and

$$z_{ij} \coloneqq \begin{cases} 1, \text{ if } \max_{k=1,\dots,m}(x_{ik}x_{j\phi(k)}) = 1 \text{ and } (i,j) \in E, \\ 0, \text{ else.} \end{cases} \tag{3.2}$$

Thus, a primal heuristic that attempts to construct a feasible solution for the CC-MIP only has to find a valid assignment for the x-variables.

3.1.1 Greedy Heuristic

This start heuristic constructs a feasible solution by iteratively assigning states to clusters in a greedy fashion. It consists of two stages.

In the first stage, each empty cluster gets assigned one state, as constraints (2.4) require all clusters to be non-empty. In the second stage, the remaining unassigned states are assigned iteratively.

Stage 1: Let $\mathcal{X} = (X_1, \dots, X_m)$ be an empty clustering, i.e., $X_t = \emptyset$ for all $t \in \mathcal{K}$ and set $U = \mathcal{S}$ as the set of unassigned states. Select $i_1, i_2 \in \mathcal{S}$ such that

$$(i_1, i_2) = \underset{(i,j)\in E}{\arg\max} \left(\max\{q_{ij} - q_{ji}, \alpha(q_{ij} + q_{ji})\}\right). \tag{3.3}$$

Assign i_1 to X_1, i_2 to X_2 and remove them from U. To finish stage 1, states $i_3, \dots, i_m$ are selected iteratively such that

$$i_{t+1} = \underset{j\in U}{\arg\max} \left(\max\{q_{i_t j} - q_{j i_t}, \alpha(q_{i_t j} + q_{j i_t})\}\right).$$

Then they are assigned to clusters $X_3, \dots, X_m$ and removed from U.

Stage 2: The new objective value v_{new} that results from assigning $i \in U$ to one of the sets $X_t, t \in \mathcal{K}$ can be computed using the old objective value v_{old} in $\mathcal{O}(n)$ operations. The coherence in cluster t has to be updated, as well as the net flow from i to $X_{\phi(t)}$ and from $X_{\phi^{-1}(t)}$ to i, respectively. It holds that

$$v^{new} = v^{old} + \eta(\mathcal{X}, i, t),$$

with

$$\eta(\mathcal{X}, i, t) = \alpha \left(\sum_{j\in X_t} q_{ij} + q_{ji} \right) + \sum_{j\in X_{\phi^{-1}(t)}} c_{ji} - c_{ij} + \sum_{j\in X_{\phi(t)}} c_{ij} - c_{ji}.$$

This value $\eta(\mathcal{X}, i, t)$ is computed for all $i \in U$ and all $t = 1, \dots, m$. Then the best state j and cluster u are selected as

$$(j, u) = \underset{(i,t)\in U\times\mathcal{K}}{\arg\max}\, \eta(\mathcal{X}, i, t).$$

The state j is removed from U and added to cluster u. This process is repeated until $U = \emptyset$. The heuristic in pseudo code is described in Algorithm 1.

The algorithm ensures that each cluster contains at least one state and that all states are assigned. Therefore it is clear that the method always produces a clustering.

As it always holds that $|U| < n$, and $\eta(\mathcal{X}, i, t)$ can be computed in $\mathcal{O}(n)$ operations, the run-time for this algorithm is in $\mathcal{O}(mn^3)$.

Algorithm 1 Greedy Heuristic

Input: Weight matrix Q
Output: Complete clustering $\mathcal{X} = (X_1, \dots, X_m)$
Initialize $\mathcal{X} = (X_1, \dots, X_m)$ with $X_t = \emptyset$ for all $t \in \mathcal{K}$. $U = \mathcal{S}$
Select i_1, i_2 according to (3.3).
$X_1 \leftarrow \{i_1\}, X_2 \leftarrow \{i_2\}, U \leftarrow U \setminus \{i_1, i_2\}$
for $t = 3, \dots, m$ **do**
    $X_t \leftarrow j$, with $j = \arg\max_{i \in U} \eta(\mathcal{X}, i, t)$
    $U \leftarrow U \setminus \{j\}$
end for
while $U \neq \emptyset$ **do**
    Compute $(j, u) = \arg\max_{(i,t) \in U \times \mathcal{K}} \eta(\mathcal{X}, i, t)$
    $X_u \leftarrow X_u \cup \{j\}$
    $U \leftarrow U \setminus \{j\}$
end while
return $\mathcal{X}$

3.1.2 Exchange Heuristic

This improvement heuristic starts with a clustering $\mathcal{X} = (X_1, \dots, X_m)$. Then a series of exchanges is computed, where states are transferred to different clusters. The heuristic is inspired by the famous graph-partitioning heuristic [26].

If a clustering $\mathcal{X} = (X_1, \dots, X_m)$ with objective value v_{old} is known, the objective value v_{new} that results from removing state i from cluster X_t and adding it to a different cluster $X_{t'}$ can be computed in $\mathcal{O}(n)$ by

$$v_{new} = v_{old} + \psi(\mathcal{X}, i, t, t'),$$

with

$$\begin{aligned}\psi(\mathcal{X}, i, t, t') \coloneqq &- \left(\alpha \sum_{j \in X_t} (c_{ij} + c_{ji}) + \sum_{j \in X_{\phi^{-1}(t)}} (q_{ji} - q_{ij}) + \sum_{j \in X_{\phi(t)}} (q_{ij} - q_{ji}) \right) \\ &+ \left(\alpha \sum_{j \in X_{t'}} (c_{ij} + c_{ji}) + \sum_{j \in X_{\phi^{-1}(t')}} (q_{ji} - q_{ij}) + \sum_{j \in X_{\phi(t')}} (q_{ij} - q_{ji}) \right).\end{aligned}$$

Let $\mathcal{X} = (X_1, \dots, X_m)$ be a clustering, and denote the set of states that were not exchanged yet by $U \subset \mathcal{S}$. For each $i \in U$, let $t(i)$ be the cluster that i is currently in. At each iteration of the heuristic, the best possible exchange of any state $i \in U$ to any other cluster is computed as

$$(j, v) = \arg\max_{(i,t) \in U \times \mathcal{K}} \psi(\mathcal{X}, i, t(i), t).$$

Then that state j is transferred to the new cluster v and removed from U, even if $\psi(\mathcal{X}, j, t(j), v) < 0$.

Repeating this until all states have been exchanged once produces n clusterings. The clustering with the best objective value is selected as the solution for the heuristic. The condition that no state can be exchanged twice is important to prevent the algorithm from cycling after an exchange that decreases the objective value.

Algorithm 2 describes this heuristic in pseudo code.

Algorithm 2 Exchange Heuristic

Input: Feasible starting solution $\mathcal{A} = (A_1, \dots, A_m)$, weight matrix Q
Output: Clustering $\mathcal{X} = (X_1, \dots, X_m)$
Initialize $\mathcal{X}_0 \leftarrow \mathcal{A}$, Set of unmarked states $U \leftarrow \mathcal{S}$, initial objective value v_0.
for $c = 1, \dots, n$ **do**
    Compute $(j, v) = \arg\max_{(i,t) \in U \times \mathcal{K}} \psi(\mathcal{X}_{c-1}, i, t(i), t)$
    $v_c \leftarrow v_{c-1} + \psi(\mathcal{X}_{c-1}, j, t(j), v)$
    $\mathcal{X}_c \leftarrow \mathcal{X}_{c-1}$, with j removed from $t(j)$ and added to v
    $U \leftarrow U \setminus \{j\}$
end for
return $\mathcal{X}_c$ with $c = \arg\max_{0,\dots,n} v_c$

The heuristic can be executed repeatedly, with the solution of the previous run as the input, until no more improvement is found. If the heuristic cannot improve upon a starting solution, then that means that the solution is locally optimal in the sense that it cannot be improved by a series of single exchanges.

Furthermore, if the found solution is not globally optimal, it might be useful to start from a significantly different solution. Therefore, a random subset $F_t \subset A_t$ with $|F_t| = \lfloor |A_t|/2 \rfloor$ is selected from each cluster in the original clustering $\mathcal{A}$. Then all the states in F_t are moved to a different cluster. While this does not guarantee to find the optimal solution either, it is highly likely that a different solution is found.

3.1.3 Rounding Heuristic

Rounding heuristics aim at recovering an integer-feasible solution from a given LP-solution. They are part of all state-of-the-art MIP solvers.

We present a rounding heuristic for the CC-MIP that only rounds on the x-variables and then tries to recover an integer solution according to (3.1) and (3.2).

Let $(x^f, y^f, z^f) \in \mathbb{R}^{nm+3|E|}$ be an LP-solution for the CC-MIP. The x-variables are rounded to binary values with

$$x_{it} := \begin{cases} 1, & \text{if } x^f_{it} = \max_{s=1,\dots,m} x^f_{is}, \\ 0, & \text{else.} \end{cases}$$

The rest of the variables are set according to (3.1) and (3.2). If there exists no empty cluster after rounding the x-variables, i.e., $\sum_{i \in \mathcal{S}} x_{it} \geq 1$ for all

$t \in \mathcal{K}$, then the heuristic produces a feasible solution. For each variable, m cases have to be considered. Therefore, the heuristic runs in $\mathcal{O}(n^2 m)$. It is described in Algorithm 3.

Algorithm 3 Rounding Heuristic

Input: LP-solution (x^f, y^f, z^f), Set of relevant transitions $E \subset \mathcal{S} \times \mathcal{S}$
Output: Integer solution (x, y, z)
for $i = 1, \ldots, n$ **do**
    Compute $t' = \arg\max_{t \in \mathcal{K}} x^f_{it}$
    for $t = 1, \ldots, m$ **do**
        if $t = t'$ **then**
            Set $x_{it} = 1$
        else
            Set $x_{it} = 0$
        end if
    end for
end for
for $(i, j) \in E$ **do**
    Set $y_{ij} = \max_{k=1,\ldots,m}(x_{ik} x_{jk})$
    Set $z_{ij} = \max_{k=1,\ldots,m}(x_{ik} x_{j\phi(k)})$
end for
return $(x, y.z)$

3.2 Valid Inequalities

Adding valid inequalities in order to separate solutions that are optimal for the LP-relaxation but not integer-feasible is an important part of any state-of-the-art MIP solver. This produces tighter relaxations, and therefore reduces the overall size of the branch-and-bound tree. We describe three kinds of problem-specific inequalities that are used as cutting planes for the cycle clustering problem. For any $i \in \mathcal{S}$, we reuse the notation $t(i)$ as the cluster that i is currently in. For a cluster t, we refer to the cluster $\phi(t)$ that is directly after t as the *successor* and to the cluster $\phi^{-1}(t)$ as the *predecessor* of t.

3.2.1 Triangle Inequalities

We call inequalities that involve exactly three states *triangle-inequalities*. In Section 2.3, we showed that the MIP can be simplified in case of exactly three clusters, e.g., the y-variables can be omitted. Therefore, we briefly consider the case of three clusters and then move to the general case of $m \geq 4$.

Let $m = 3$. If there exists a transition from state i to state j, as well as a transition from state j to state k, then i has to be in the successor of

$t(k)$, since there are exactly three clusters. This yields the inequality

$$z_{ij} + z_{jk} - z_{ki} \leq 1 \qquad \forall (i,j),(j,k),(k,i) \in E$$

and is illutrated in Figure 3.1.

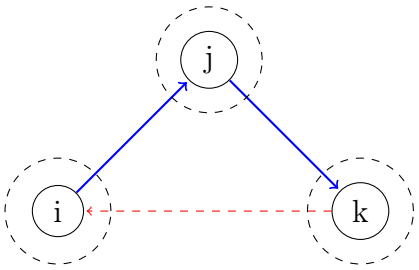


FIGURE 3.1: Triangle for three clusters. If both z_{ij} and z_{jk} are set to one (indicated by the non-dashed arrows), then z_{ki} has to be one as well. The three clusters are indicated by the dashed circles.

For the rest of this section, assume that $m \geq 4$. The first kind of inequality only incorporates y-variables and is also used in several other graph-partitioning publications, e.g., [8, 22]. The reasoning is that if state i is in the same cluster as state j and j is in the same cluster as state k, then i and k have to be in the same cluster. This can be formulated as

$$y_{ij} + y_{jk} - y_{ik} \leq 1 \qquad \forall (i,j),(j,k),(k,i) \in E \tag{3.4}$$

and is illustrated in Figure 3.2. Due to symmetry, (3.4) remains valid no matter where the minus sign is.

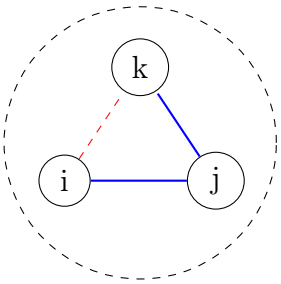


FIGURE 3.2: Triangle within one cluster. The y-variables are indicated by the lines between i, j, and k. The fact that all three states are in the same cluster is indicated by the dashed circle.

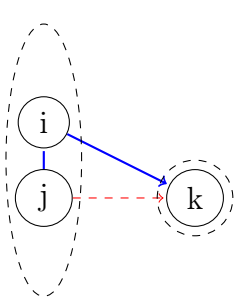
The following observation yields another kind of triangle inequality. If i and j are in the same cluster and k is in the successor of $t(i)$, then k also has to be in the successor of $t(j)$. The corresponding inequality is

$$y_{ij} + z_{ik} - z_{jk} \leq 1 \qquad \forall (i,j),(j,k),(k,i) \in E. \tag{3.5}$$

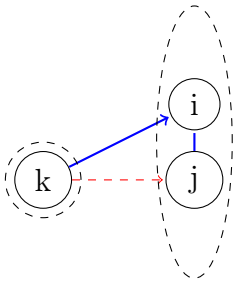
Of course, the same argument is true if we use the predecessor of $t(i)$ instead, yielding the inequality

$$y_{ij} + z_{ki} - z_{kj} \leq 1 \qquad \forall (i,j),(j,k),(k,i) \in E. \tag{3.6}$$

The inequality (3.5) is illustrated in Figure 3.3a and (3.6) is illustrated in Figure 3.3b.



(A) State k is in the successor of $t(i)$.



(B) State k is in the predecessor of $t(i)$.

FIGURE 3.3: Second kind of triangle.

In order to strengthen these inequalities, we introduce a facet-defining inequality.

Theorem 14. *Let $i,j,k \in \mathcal{S}$ with $(i,j),(j,k),(i,k) \in E$. Then the triangle inequality*

$$y_{ij} + y_{jk} - y_{ik} + 0.5\,(z_{ij} + z_{ji} + z_{jk} + z_{kj} - z_{ik} - z_{ki}) \leq 1 \tag{3.7}$$

defines a facet of the cycle clustering polytope.

Proof. Similar to the argumentation for the model-inequalities of CC-MIP in Section 2.3, the validity of inequality (3.7) is explained by the observations of the inequalities (3.4), (3.5), and (3.6).

The proof that this inequality defines a facet is very similar to the proofs in Section 2.4. Therefore, we skip the proof here and refer to Appendix A, Theorem 20. □

Another kind of triangle-inequality is derived from the following observation. If state i is in the predecessor of $t(j)$ as well as in the predecessor of $t(k)$, then j and k must be in the same cluster. This yields the inequality

$$z_{ij} + z_{ik} - y_{jk} \leq 1 \qquad \forall (i,j),(i,k),(j,k) \in E. \tag{3.8}$$

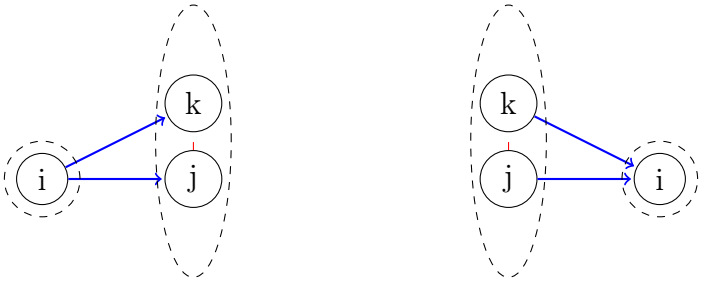
Conversely, if i is the successor of $t(j)$ and also in the successor of $t(k)$ then j and k are in the same cluster. This results in the inequality

$$z_{ji} + z_{ki} - y_{jk} \leq 1 \qquad \forall (i,j),(i,k),(j,k) \in E. \tag{3.9}$$

The inequalities (3.8) and (3.9) are illustrated in Figure 3.4a and 3.4b, respectively.

In the case that $m = 4$ we can strengthen the inequality by using

$$z_{ij} + z_{ik} - 2y_{jk} - (z_{jk} + z_{kj} + z_{ji} + z_{ki}) \leq 0. \tag{3.10}$$



(A) State i is in the predecessor of $t(k)$ and $t(j)$

(B) State i is in the successor of $t(k)$ and $t(j)$

FIGURE 3.4: Third kind of triangle.

Theorem 15. *Let $i, j, k \in \mathcal{S}$ with $(i, j), (j, k), (i, k) \in E$.*

- *If $m > 4$, then (3.8) and (3.9) are facet-defining for the cycle clustering polytope.*
- *If $m = 4$, then (3.10) is facet-defining*

Proof. We have already explained why the inequality is valid if $m > 4$. For the special inequality if $m = 4$, consider the following.

If $m = 4$ and j is in the successor of $t(i)$ but k is not in the successor of $t(i)$, then k has to be in one of the other three clusters. In each of those three cases, one of the variables z_{jk}, z_{kj}, z_{ki} has to be set to one. If k is in the same cluster as i, then z_{kj} has to be one. If k is in the predecessor of $t(i)$, then z_{ki} has to be one. Finally, if k is in the successor of $t(j)$, then z_{jk} has to be one. The same argumentation holds if z_{ik} is one but z_{ij} is not.

For the proof that the inequalities are facet-defining, we refer to Appendix A, Theorem 21. □

3.2.2 Subtour and Path Inequalities

In the cycle clustering model, we consider a fixed number of clusters. This means that if there is a transition between clusters from a state i to a state j, there must be exactly $m - 1$ further transitions along the cycle before we can reach i again. This observation yields the following inequality.

Let $K = \{(i_1, i_2), (i_2, i_3), \ldots, (i_{s-1}, i_s), (i_s, i_1)\} \subset E$ be any cycle of length $1 < |K| < m$. Then the *subtour-elimination inequality*

$$\sum_{(i,j)\in K} z_{ij} \leq |K| - 1 \tag{3.11}$$

is valid for any solution of the cycle clustering MIP.

The inequality can be strengthened by adding variables for transitions inside a cluster. Let $U \subset K$ be any subset of the edges in the cycle with $|U| \leq |K| - 1$. Then the *extended subtour-elimination inequality*

$$\sum_{(i,j)\in K} z_{ij} + \sum_{(i,j)\in U} y_{ij} \leq |K| - 1 \tag{3.12}$$

is valid for any solution of the cycle clustering MIP. The inequality is valid because there cannot exists a cycle with a number of forward transitions that is not a multiple of the amount of clusters m. If (3.12) is violated, then there exists a cycle with a number of forward transitions greater than zero but smaller than the amount of clusters. The inequality (3.12) is stronger than (3.11) in the sense that it defines a higer dimensional face of CCP. It is evident that any $(x, y, z) \in CCP$ that satisfies (3.11) at equality also satisfies (3.12) at equality, since we added a number of non-negative variables. An example for a clustering that satifies (3.12) but not (3.11) at equality is $(A_1, \ldots, A_m)$ with $i_t \in A_1$ for all $t = 1, \ldots, s$.

Figure 3.5 illustrates a violated subtour of length 3 for a problem with 5 clusters. Figure 3.6 illustrates a violated extended subtour.

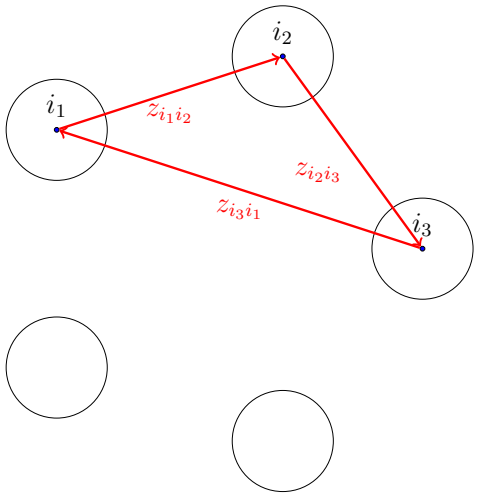


FIGURE 3.5: Invalid subtour of length 3 in a problem with 5 clusters. State i_1 is reached again after three forward transitions.

These extended subtour-elimination inequalities do not define facets but proved effective in practice and can be separated effectively.

We give an example of an inequality that defines a higher-dimensional face than (3.12) for $|K| = 4$. Let $K = \{(i_1, i_2), (i_2, i_3), (i_3, i_4), (i_4, i_1)\} \subset E$ be any cycle of length 4 and let $m \geq 5$. Then the inequality

$$\sum_{(i,j)\in K} z_{ij} + y_{ij} - y_{i_1,i_3} \leq 3 \tag{3.13}$$

is satisfied at equality by more points in CCP than (3.12).

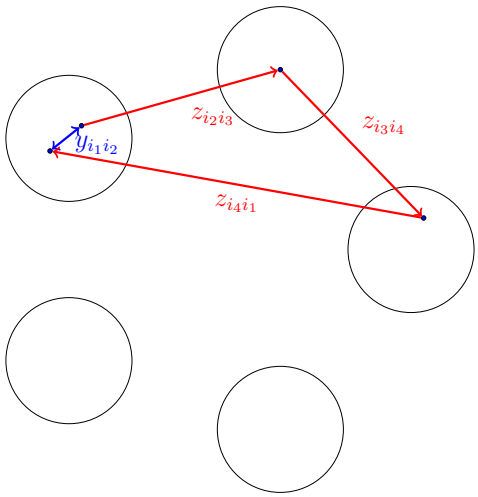


FIGURE 3.6: Invalid extended subtour of length 4 in a 5-cluster problem. One y-variable and 3 z-variables are set to one, resulting in an invalid cycle with 3 forward transitions.

Let $P = (A_1, \ldots, A_m)$ be any clustering of $\mathcal{S}$ that satisfies (3.12) at equality. If i_1 and i_3 are in the same cluster A_t in P, then the only possibility that (3.12) is satisfied at equality is if i_2 and i_4 are also in A_t. In that case (3.13) is tight. If i_1 and i_3 are in different clusters then $y_{i_1,i_3} = 0$ and it is obvious that (3.13) is satisfied at equality. Any clustering $(A_1, \ldots, A_m)$ with $i_1, i_2 \in A_1$ and $i_3, i_4 \in A_2$ satisfies (3.13) at equality but not (3.12).

Theorem 16. *The extended subtour inequality* (3.12) *can be separated in polynomial time.*

Proof. We assume w.l.o.g. that $U = K \setminus \{(i_1, i_2)\}$, since we can choose any edge of a cycle as the first edge.

Let (x^f, y^f, z^f) be a given LP-solution. We choose one $u \in \mathcal{S}$ as the start node and define a directed graph $G = (\mathcal{S}, E)$, with the edge set $E = \{(i, j) \mid z^f_{ij} + y^f_{ij} > 0\}$. We define arc weights a_{ij} as

$$a_{ij} = \begin{cases} z^f_{ij}, & \text{if } i = u, \\ z^f_{ij} + y^f_{ij}, & \text{else.} \end{cases} \tag{3.14}$$

Then a cycle K in E of length $s < m$ that has a weight greater than $|K| - 1$ corresponds to a violated extended subtour. Therefore, the inequality (3.12) can be separated by computing longest paths with a fixed number of arcs. As all arc weights in the graph are non-negative, this is possible in polynomial time. Algorithm 4 detects for all $s = 2, ..., m - 1$ if there exists a cycle of length s that violates (3.12). □

Algorithm 4 Subtour-Elimination

1: **Input:** LP-solution (x^f, y^f, z^f), number of clusters m, set of states $\mathcal{S}$
2: **Output:** Violated extended subtour, if there exists one
3: **for all** $u \in \mathcal{S}$ **do**
4: Initialize a^1_{ij} as described in (3.14)
5: **for** $s = 2, \dots, m-1$ **do**
6: **for** $(i,j) \in \mathcal{S} \times \mathcal{S}$ **do**
7: $a^s_{i,j} = \max_{l \in \mathcal{S} \setminus \{i,j\}} a^{s-1}_{i,l} + a_{l,j}$
8: **end for**
9: **if** $(a^s_{i,i} > s-1)$ **then**
10: Found violated subtour. The cycle can be found by backtracking.
11: **end if**
12: **end for**
13: **end for**

The following *path inequality* stems from a similar observation. Let m be the number of clusters, $i_1 \neq i_m \in \mathcal{S}$, $P = \{(i_1, i_2,), \dots, (i_{m-1}, i_m)\}$ be a path from i_1 to i_m of length $m-1$, and let $U \subset P$ with $0 \leq |U| \leq |P| - 1$. Then the path inequality

$$\sum_{(i,j)\in P} z_{ij} + \sum_{(i,j)\in U} y_{ij} + y_{i_1,i_m} \leq m - 1 \tag{3.15}$$

is valid for CCP. If $\sum_{(i,j)\in P} z_{ij} + \sum_{(i,j)\in U} y_{ij} = m-1$, then there are between one and $m-1$ forward transitions from i_1 to i_m. Therefore, i_1 and i_m cannot be in the same cluster and y_{i_1,i_m} has to be zero.

Figure 3.7 illustrates this type of inequality. These path inequalities can be separated in the same way as the subtour inequalities. We simply check in Step 10 of Algorithm 4 whether $a^{m-1}_{i_1,i_m} + y^f_{i_1,i_m} > m - 1$.

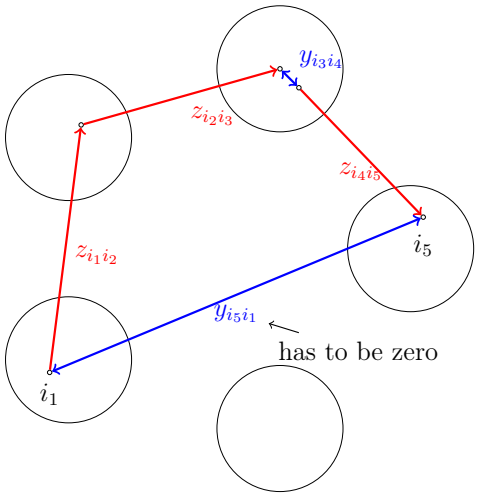


FIGURE 3.7: Invalid path of length 4 in a 5-cluster problem. The path from i_1 to i_5 has 3 forward transitions. Therefore i_1 and i_5 cannot be in the same cluster.

3.2.3 Partition Inequalities

We introduce one more kind of valid inequality that is a generalization of a triangle inequality. It is motivated by the partition inequalities in [22] and illustrated in Figure 3.8.

Theorem 17. *Let $A, B \subset \mathcal{S}, A \cap B = \emptyset$. Then the partition inequality*

$$\sum_{i\in A, j\in B} z_{ij} - \sum_{i,j\in A, i<j} y_{ij} - \sum_{i,j\in B, i<j} y_{ij} \leq \min\{|A|, |B|\} \tag{3.16}$$

is valid for the cycle clustering polytope.

Proof. We show this by induction over $|A| + |B|$.

Induction Base: Let $A, B \subset \mathcal{S}, A \cap B = \emptyset, |A| = 1$, *i.e.*, $A = \{u\}$. Then the inequality is of the form

$$\sum_{i\in B} z_{ui} - \sum_{i,j\in B, i<j} y_{ij} \leq 1.$$

If none of the z_{ui} is set to one, then the inequality is obviously satisfied. Denote by $B_o \subseteq B$ the subset of B such that $z_{ui} = 1$ for all $i \in B_o$. Then $y_{ij} = 1$ for all $i, j \in B_o$, because they are all one cluster after $t(u)$. So it holds that

$$\sum_{i\in B} z_{ui} - \sum_{i,j\in B, i<j} y_{ij} = |B_o| - \sum_{l=1}^{|B_o|-1} l \leq 1. \tag{3.17}$$

Analogously, it follows that the inequality is valid for any A, B with $|B| = 1$.

Inductive Step: Assume validity was proven for A, B with $|A|+|B| \leq k$. Assume w.l.o.g. that $|A| \leq |B|$. We have already proven the claim in the case of $|A| = 1$, so assume further that $|A| > 1$. Select any $u \in A$. Then, by induction hypothesis the inequality is valid for $A \setminus \{u\}$ and B, i.e., it holds that

$$\sum_{i\in A\setminus\{u\}, j\in B} z_{ij} - \sum_{i,j\in A\setminus\{u\}\setminus, i<j} y_{ij} - \sum_{i,j\in B, i<j} y_{ij} \leq |A| - 1. \tag{3.18}$$

Conversely, the same holds if we remove any $v \in B$, i.e.,

$$\sum_{i\in A, j\in B\setminus\{v\}} z_{ij} - \sum_{i,j\in A, i<j} y_{ij} - \sum_{i,j\in B\setminus\{v\}, i<j} y_{ij} \leq \min\{|A|, |B| - 1\}. \tag{3.19}$$

We sum up (3.18) for all $u \in A$. It follows

$$\begin{aligned}
&\sum_{u\in A} \left(\sum_{i\in A\setminus\{u\}, j\in B} z_{ij} - \sum_{i,j\in A\setminus\{u\}, i<j} y_{ij} - \sum_{i,j\in B, i<j} y_{ij} \right) \\
&= (|A| - 1) \left(\sum_{i\in A, j\in B} z_{ij} - \sum_{i,j\in A, i<j} y_{ij} \right) - |A| \left(\sum_{i,j\in B, i<j} y_{ij} \right).
\end{aligned}$$

The same holds if we sum up (3.19) for all $v \in B$. Summing up (3.18) for all $u \in A$ and (3.19) for all $v \in B$ yields the inequality

$$(|A| + |B| - 2)\left(\sum_{i\in A, j\in B} z_{ij} - \sum_{i,j\in A, i\neq j} y_{ij} - \sum_{i,j\in B, i\neq j} y_{ij}\right) \leq |A|(|A|-1) + |B|\min\{|A|, |B|-1\}.$$

Since it holds that $\min\{|A|, |B|-1\} \leq |A|$, dividing by $(|A|+|B|-2)$ yields

$$\sum_{i\in A, j\in B} z_{ij} - \sum_{i,j\in A, i\neq j} y_{ij} - \sum_{i,j\in B, i\neq j} y_{ij} \leq \left\lfloor \frac{|A|(|A|+|B|-1)}{(|A|+|B|-2)} \right\rfloor = |A|.$$

Rounding down the right hand side is allowed here, since the left hand side has an integer value for all points in CCP. This concludes the proof. □

Theorem 18. *The partition inequalities 3.16 are facet-defining if $m > 4$ and $||A| - |B|| = 1$.*

For the proof we refer to Appendix A, Theorem 22.

Separation for these inequalities is done heuristically and only violated partition inequalities with $|A| + |B| \leq 5$ are searched. Let (x^f, y^f, z^f) be an LP-solution of CC-MIP. We enumerate all triangle-inequalities, and consider the active ones i.e., $A = \{i\}, B = \{j, k\}$ such that

$$z^f_{ij} + z^f_{ik} - y^f_{jk} = 1.$$

Then we add one more state l to B with

$$l = \arg\max_{v\in\mathcal{S}} \left(z^f_{iv} - y^f_{vj} - y^f_{vk}\right). \tag{3.20}$$

If the corresponding partition inequality is violated, i.e., with $A = \{i\}$ and $B = \{j, k, l\}$, it is added to the MIP. Otherwise, we add one more state u to A with

$$u = \arg\max_{v\in\mathcal{S}} \left(\sum_{w\in B} z^f_{vw} - y^f_{vi}\right).$$

The partition inequality with $A = \{u, i\}$ and $B = \{j, k, l\}$ is added to the MIP, if it is violated by the LP-solution.

Selecting states in this way and only for active triangle inequalities keeps the computational effort to an acceptable amount for the test set that we considered. We enumerate $\mathcal{O}(n^3)$ triangle inequalities and for each of the active ones, we consider $2n$ states.

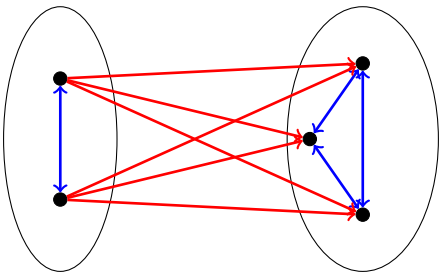


FIGURE 3.8: Illustration of a partition inequality with $|A| = 2, |B| = 3$. If all the z-variables are one (represented by the arcs between the clusters) then all the y-variables also have to be set to one (edges inside clusters) and the inequality is satisfied at equality.

3.3 Multinode Branching

Our experiments suggest that standard two-node branching is not the best strategy for our problem instances. Examine the constraints

$$\sum_{t=1}^{m} x_{it} = 1.$$

Setting one of the x_{it}-variables to one implicates that all other $x_{it'}$ for this state have to be set to zero. In contrast, fixing x_{it} to zero has no direct implications on its own. This is the motivation for a branching rule that only sets x-variables to one.

There exist sophisticated branching rules, specifically for set-partitioning problems [7], but as the set-partitioning constraints in our problem have a very simple structure, we use a simple branching rule called multinode branching, also presented in [7].

Let (x^f, y^f, z^f) be the LP-Solution at a node in the branch-and-bound tree. One state i and a subset of clusters $F \subset \mathcal{K}$ is selected, with

$$0 < x_{it}^f < 1 \qquad \text{for all } t \in F.$$

One child node is created for each $t \in F$, with $x_{it} = 1$ as the branching decision, as well as one child node with $\sum_{t \in F} x_{it} = 0$.

The branching rule is illustrated in Figure 3.9. In order to keep the tree from getting very broad, the branching rule is restricted to $|F| \leq 3$ so no more than 4 children can be created at each node.

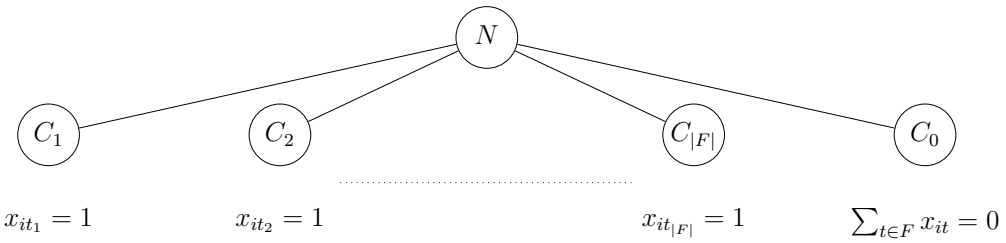


FIGURE 3.9: Illustration of multinode branching. $|F| + 1$ child nodes are created. At the first $|F|$ child nodes, one of the variables is set to one, in the last node all variables in F are set to zero.

3.4 Approximation by Reducing Instance Size

The adjacency matrix Q determines how many variables and constraints the formulation has. The denser Q is, the more variables and constraints are needed. This matrix Q stems from a markov state model. In practice, these matrices are often dense as the membership functions used when assigning microstates do not have a local support, see e.g., [16]. This means that between many of the states i and j, the transition probability q_{ij} might be very small compared to a few transitions with higher probability.

For two states $i, j \in \mathcal{S}$, define the *impact* of the transition ij as

$$\theta_{ij} = \max(\alpha(q_{ij} + q_{ji}), |q_{ij} - q_{ji}|).$$

In order to reduce the size of the MIP, we define a percentile of transitions, according to their impact. Then that percentile of transitions with the smallest impact is set to zero. Denote the set of transitions discarded this way by $D \subset \mathcal{S} \times \mathcal{S}$. We call the MIP resulting from omitting all of these transitions CC-MIP$_{red}$. Let x^{red} be the optimal solution of CC-MIP$_{red}$ with objective value v^{red}. Then we know that the distance to the optimal objective value v^{orig} of the original CC-MIP can be at most the sum of all the discarded transitions, i.e.,

$$\delta := \sum_{(i,j) \in D} \theta_{ij} \geq |v^{red} - v^{orig}|. \tag{3.21}$$

The reason this holds is that for any transition $(i, j) \in \mathcal{S} \times \mathcal{S}$, the impact on the objective function is limited by θ_{ij}. At most one of y_{ij}, z_{ij} and z_{ji} can be set to one and each has an objective coefficient that is bounded by θ_{ij}.

This can be used as a simple approximation algorithm. The reduced MIP is solved and provides an approximate solution of the original MIP, with a maximal approximation error of δ. Even if the reduced MIP cannot be solved to optimality, any dual bound of the reduced MIP can be turned into a dual bound for the original MIP, by adding δ.

Chapter 4

Computational Experiments

In this chapter, we evaluate the performance of the solving methods presented in Chapter 3. To do this, we developed a SCIP application for the cycle clustering problem called CC-SCIP.

SCIP is a framework for solving constrained integer programs and the fastest non-commercial solver for mixed integer programs. Originally developed by Achterberg [1, 2], SCIP is continuously improved and extended by numerous researchers at the Zuse-Institute Berlin [29].

The application CC-SCIP reads a matrix of unconditional transition probabilities Q and creates from it the model described as CC-MIP in Section 2.3. In order to extend SCIP for cycle clustering, we implemented the heuristics (cf. 3.1) and separation routines (cf. 3.2) described in the previous chapter, as well as the multinode branching rule (cf. 3.3).

The following is a short outline of the structure of this chapter. We first describe the test set and then evaluate the overall performance of CC-SCIP by comparing it to the default version of SCIP, as well as to the commercial solver Gurobi. Furthermore, we evaluate all the presented solving methods in detail, measuring their individual impact and the benefit of combining them. In addition, we evaluate the approximation method presented in Section 3.4 in terms of speedup and approximation error. Finally, we compare three different relaxations for the cycle clustering problem.

We will focus strictly on computational performance, for experiments evaluating the quality of the model see [4].

All experiments were run on 2.7 GHz Intel Xeon E5-2680 0 CPUs with 64 GB main memory. The time limit was set at $3600s$, with only one job running per node at a time.

4.1 Test Set

In our computational experiments, we consider a test set with data from three different types of simulations. The first type are catalytic cycle instances and they constitute the largest part of the test set with 32 instances. The second kind are repressilator simulations with 6 instances and the third kind are Hindmarsh-Rose simulations, also with 6 instances. Moreover, for each of these 44 instances, we created 3 instances of reduced size by applying the method described in Section 3.4.

The instance sizes range from 20 to 250 states. For detailed information on the MIP-size of all the instances, we refer to Appendix B.

The following is a brief description of the three different kinds of simulations. We will describe the catalytic cycle in more detail and refer to the literature for more in-depth descriptions of the rest.

Catalytic cycle

The first part of the test set is comprised of artificial catalytic cycle instances that were created using a hybrid Monte-Carlo method (HMC) [6]. A potential $\Omega : \mathbb{R}^2 \to \mathbb{R}$ with several minima, and a drift $d \in \mathbb{R}^2$, that propagates the simulation towards the next minimum is used. When the system enters a predefined set, the drift is changed so that the propagation turns toward the next minimum. Figure 4.1 illustrates a potential with four minima.

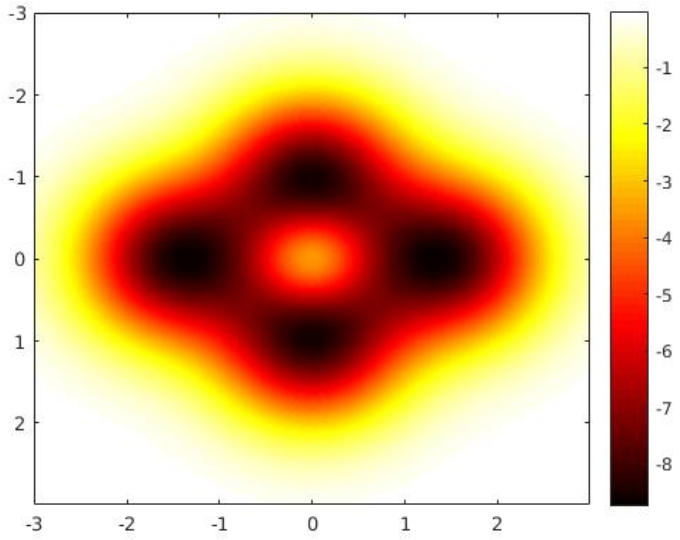


FIGURE 4.1: Heat map of a potential with four minima that are plotted as the darker areas.

For further details on the creation of the instances, we refer to [4]. Algorithm 5 describes the creation of the trajectory.

Mircostates $x_1, \ldots, x_n$ are created such that the fill distance

$$h = \max_{j=1,\ldots,N} \min_{i=1,\ldots,n} \|u_j - x_i\|_2$$

is minimized.

It is noteworthy that non-local membership functions were used when defining the microstates, i.e., each trajectory-point u is assigned fractionally to the microstates according to

$$\psi_i(u) = \frac{\exp(-\|u - x_i\|_2^2)}{\sum_{j=1}^{n} \exp(-\|u - x_j\|_2^2)}.$$

The transition matrix P is defined by

$$p_{ij} = \frac{\sum_{k=0}^{N} \psi_i(u_k)\psi_j(\tilde{u}_k)}{\sum_{k=0}^{N} \psi_i(u_k)},$$

where $\tilde{u}_k$ is the state u_k after one time step. Thus, the resulting transition matrices are dense.

The test set is comprised of instances with $3, 4$, and 6 minima. For each of those, transition matrices with $20, 30, 50, 100$, and 200 states were created.

Algorithm 5 HMC with drift

Input: Start vector u_0, inverse temperature β, N, drift d, random vectors $r_1, \ldots, r_N$, uniformly distributed numbers $v_1, \ldots, v_N \in [0, 1]$
Output: trajectory $u_0, \ldots, u_{N-1}$
for $i = 1, \ldots, N-1$ **do**
    $u_{new} \leftarrow u_{i-1} + r_i + d$
    if $\exp(-\beta(\Omega(u_{new}) - \Omega(u_{i-1}))) < v_i$ **then**
        $u_i \leftarrow u_{new}$
        $d \leftarrow \text{update}(d)$
    else
        $u_i \leftarrow u_{i-1}$
    end if
end for

Repressilator

The repressilator is a very prominent example of a synthetic genetic regulatory network [15]. Three genes TetR, λcI, and LacI each produce a protein that represses the production of the mRNA of one of the other two genes. Therefore, the whole system expresses a cyclic behavior. This system can be described by a system of six ordinary differential equations, one for each gene and each protein.

$$\frac{\mathrm{d}\, m_A}{\mathrm{d}\, t} = -m_A + \frac{v}{1 + p_C^h} + v_0 \qquad \frac{\mathrm{d}\, p_A}{\mathrm{d}\, t} = -\beta(p_A - m_A)$$
$$\frac{\mathrm{d}\, m_B}{\mathrm{d}\, t} = -m_B + \frac{v}{1 + p_A^h} + v_0 \qquad \frac{\mathrm{d}\, p_B}{\mathrm{d}\, t} = -\beta(p_B - m_B)$$
$$\frac{\mathrm{d}\, m_C}{\mathrm{d}\, t} = -m_C + \frac{v}{1 + p_B^h} + v_0 \qquad \frac{\mathrm{d}\, p_C}{\mathrm{d}\, t} = -\beta(p_C - m_C)$$

The constants were set according to [4] with $v = 298.2, \beta = 0.2, v_0 = 0.03$, and $h = 2$. The test set is comprised of instances with $40, 80$, and 200 microstates.

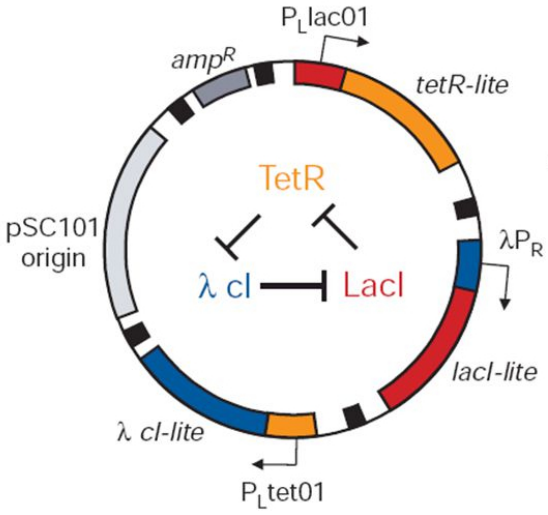


FIGURE 4.2: Illustration of the repressilator network. Image from [15]

Hindmarsh-Rose

The Hindmarsh-Rose model is used to study neuronal activity and is formulated as a system of nonlinear ordinary differential equations [35]. The equations read

$$\begin{aligned}\frac{\mathrm{d}\,x}{\mathrm{d}\,t} &= y - ax^3 + bx^2 - z + I,\\ \frac{\mathrm{d}\,y}{\mathrm{d}\,t} &= c - dx^2 - y,\\ \frac{\mathrm{d}\,z}{\mathrm{d}\,t} &= \epsilon(s(x - x_0) - z).\end{aligned}$$

In our test set, this system was used to simulate the membrane potential of cells in a human heart. Test instances with 50 and 250 microstates were created.

4.2 Overall Performance

In order to evaluate the methods presented in this thesis, we compare CC-SCIP to SCIP 4.0.0, as well as to Gurobi 7.5.1. For all SCIP variants, we used CPLEX 12.7.1.0 as the LP-solver. We ran Gurobi with default settings and made two runs for SCIP, one with default settings and one using the multinode branching rule presented in Section 3.3. The number of instances that were solved within the time limit is compared, as well as the average solving time. Furthermore, the primal integral as well as the dual integral are computed in order to gain insight into the solving process. Since the instances vary greatly in size, we use the shifted geometric mean as defined in [1] with a shift of 10 to measure average time, primal integral and dual integral.

The primal integral is defined according to [5]. Let $\tilde{x}$ be any solution of a MIP, $\tilde{x}_{opt}$ be an optimal solution and $c^T\tilde{x}$ be the the objective value. The primal gap $\gamma \in [0,1]$ of $\tilde{x}$ is defined as

$$\gamma(\tilde{x}) := \begin{cases} 0, & \text{if } |c^T\tilde{x}_{opt}| = |c^T\tilde{x}| = 0, \\ 1, & \text{if } c^T\tilde{x}_{opt} \cdot c^T\tilde{x} < 0, \\ \frac{|c^T\tilde{x}_{opt} - c^T\tilde{x}|}{\max\{|c^T\tilde{x}_{opt}|, |c^T\tilde{x}|\}}, & \text{else.} \end{cases}$$

The primal gap function $p : [0, t_{max}] \longrightarrow [0,1]$ is then defined as the primal gap over time

$$p(t) := \begin{cases} 1, & \text{if no incumbent solution until point } t, \\ \gamma(\tilde{x}(t)), & \text{with } \tilde{x}(t) \text{ incumbent solution at point } t. \end{cases}$$

Since the primal gap function is piecewise constant, the primal integral $P(T)$ with $T \in [0, t_{max}]$ is given by

$$P(T) := \int_{t=0}^{T} p(t)dt = \sum_{i=1}^{l} p(t_{i-1}) \cdot (t_i - t_{i-1}).$$

The primal integral is useful to measure the impact of primal heuristics and can be used to indicate how early good solutions are found during the solving process [5].

The dual integral is defined in the same way, the gap is computed as the difference between the current dual bound and the best known primal solution. This introduces a constant offset if the optimal solution is unknown. The dual integral is useful to measure the impact of separation routines and gives insight how quickly good bounds for the optimization problem can be proven.

Table 4.1 shows a comparison between the different solvers. CC-SCIP solves 11 more instances than Multinode-SCIP, 12 more than Gurobi and 17 more than SCIP within the given time limit. On average, CC-SCIP performs best in all three categories. Multinode-SCIP produces the second best results in terms of solving time and dual integral, while Gurobi produces the second smallest primal integral. Compared to the second best in each category, CC-SCIP has less than half the solving time, with a primal integral that is a factor of $\times 14.9$ and a dual integral that is a factor of $\times 3.7$ smaller than for the other solvers.

It is interesting to note that Multinode-SCIP has a smaller dual integral than Gurobi and is actually able to solve one more instance to optimality.

Solver	solved	time [s]	primal int	dual int
CC-SCIP	130	64.7	82.9	327.1
Multinode-SCIP	119	130.6	2160.9	1213.3
SCIP	113	182.4	3274.8	1794.0
Gurobi	118	153.1	1236.6	1524.9

TABLE 4.1: Overview of the performance of the cycle clustering application CC-SCIP in comparison to SCIP, Multinode-SCIP, and Gurobi. Column 'solved' gives the number of instances solved to optimality out of the total 176 instances. All other columns are geometric means with a shift of 10.

The very small primal integral of CC-SCIP can be explained by the fact that the best known reference value for the unsolved instances is the best solution found by the heuristics in CC-SCIP. For this reason, Table 4.2 shows the performance on the 112 instances that can be solved to optimality within the given time limit by all four solvers. The general performance is similar. CC-SCIP shows the best performance in all three categories, with Multinode-SCIP being second in time and dual integral, while Gurobi comes in second in terms of the primal integral. We discuss the average improvement that CC-SCIP presents, compared to the second best solver in the different categories. The speedup in solving time is of a factor $\times 4.1$, the improvement in the primal integral is of a factor $\times 6.4$ and the improvement in the dual integral is of a factor $\times 4.8$.

This indicates that overall, the methods developed in this thesis are effective, with a speedup of at least $\times 4$, compared to the second fastest solver. Comparing the different values for SCIP and Multinode-SCIP, the non-standard branching rule seems to be a good choice for this type of problem.

Solver	time [s]	primal int	dual int
CC-SCIP	3.3	15.8	19.1
Multinode-SCIP	13.6	206.9	91.9
Default-SCIP	26.6	324.9	167.3
Gurobi	19.3	101.7	138.2

TABLE 4.2: Comparison on the 112 instances that were solved to optimality by all four solvers.

Figures 4.3 and 4.4 show the average primal and dual gap function over time, respectively.

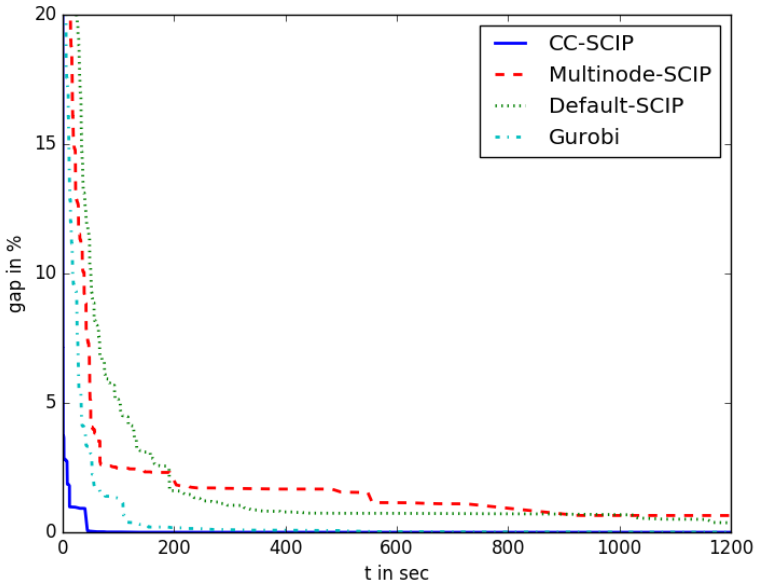


FIGURE 4.3: Average primal gap function over time for instances that could be solved to optimality by all solvers.

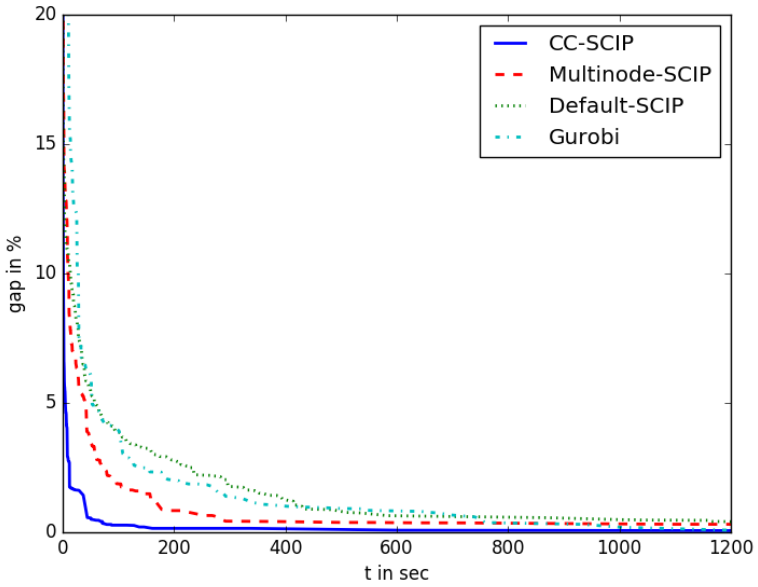


FIGURE 4.4: Average dual gap function over time for instances that could be solved to optimality by all solvers.

4.3 Evaluation of Primal Heuristics

In order to evaluate the impact of the three heuristics presented in Chapter 3, we tested each heuristic on its own, all other problem specific heuristics and problem specific separation routines were turned off. In addition, we

ran all heuristics together, all problem specific separation was turned off. This is compared to Multinode-SCIP as a baseline.

We run the greedy heuristic only at the start of the solving process. The exchange heuristic is run every 10 nodes, and only when a new primal feasible solution is found. The rounding heuristic is run after every node.

The average solving time, number of nodes in the branch-and-bound tree, and primal integral are compared, using the geometric mean with a shift of 10 for all averages. Table 4.3 shows the results of this experiment. Each individual heuristic shows a slight improvement compared to Multinode-SCIP. The exchange heuristic has the highest individual impact, which leads to a reduction of 7% in solving time, 33% in the number of nodes, and 53% in the primal integral. All heuristics run together provide a higher impact than each heuristic on its own, with a reduction in the primal integral of 96%.

The decrease in the average solving time is relatively small at 11%, compared to the decrease in the primal integral. One reason for this is that without the problem specific separation, the LP-relaxation does not provide tight bounds and the proof of optimality takes a long time. Another reason is that some hard instances cannot be solved within the time limit by any variant and therefore contribute the same to the average solving time for all tests.

Heuristic	time[s]		nodes		primal integral	
default	130.6	-	256.6	-	2160.9	-
greedy	128.4	0.98	260.2	1.01	1478.8	**0.68**
exchange	120.9	0.93	171.4	**0.67**	1009.6	**0.47**
rounding	125.8	0.96	188.9	**0.74**	1475.1	**0.68**
all	116.4	**0.89**	140.2	**0.55**	86.5	**0.04**

TABLE 4.3: Comparison of solving time, number of nodes, and primal integral for the different heuristics. In each category, the value as well as the ratio compared to the default (Multinode-SCIP) is displayed.

Table 4.4 shows the same experiment on the part of the test set that could be solved optimally within the time limit by the default. As expected, the decrease in solving time and number of nodes is larger than on the whole test set, while the decrease in terms of primal integral is smaller.

Figure 4.5 shows the course of the average primal gap function over time for the different heuristics and illustrates the point that the heuristics run together have a higher impact than each heuristic on its own.

Heuristic	time[s]		nodes		primal integral	
default	19.7	-	211.9	-	255.9	
greedy	19.0	0.96	190.9	0.90	182.3	**0.71**
exchange	16.8	**0.85**	117.9	**0.56**	115.3	**0.45**
rounding	18.2	0.92	144.8	**0.68**	166.9	**0.65**
all	15.5	**0.79**	82.7	**0.39**	21.1	**0.08**

TABLE 4.4: Comparison of heuristics on the part of the test set that could be solved optimally by the default (Multinode-SCIP).

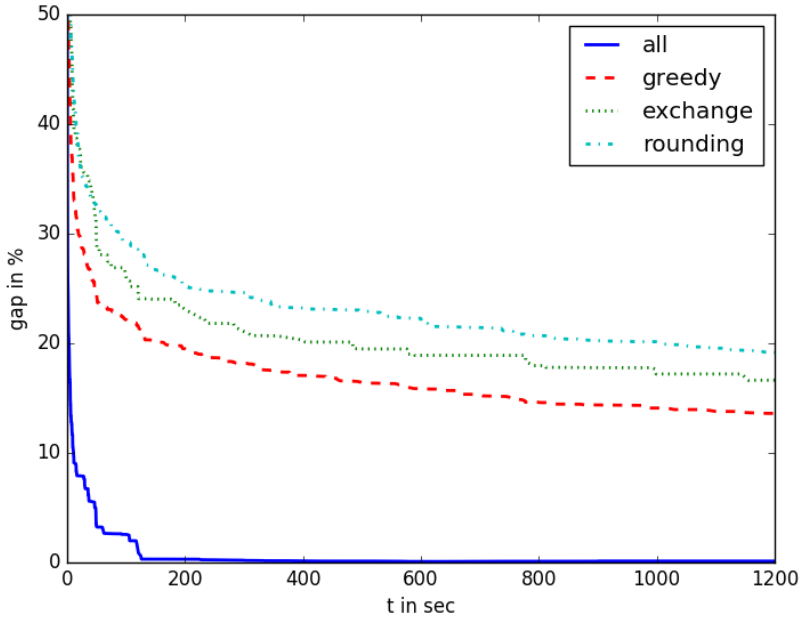


FIGURE 4.5: Average primal gap function over time for different heuristics.

4.4 Evaluation of Valid Inequalities

To evaluate the valid inequalities presented in Section 3.2, we ran the same test setup as for the heuristics. Each type of inequality is tested on its own, all problem-specific heuristics and other problem-specific separation routines turned off. Additionally, all separation routines are run together, all problem-specific heuristics turned off. We limited the number of separated inequalities per call to 500 for all three types of inequalities.

The results are compared to the Multinode-SCIP as default in Table 4.5. All averages are calculated using the geometric mean with a shift of 10. Each type of inequality shows a decrease in average time, nodes and the dual integral compared to the default. The highest individual impact comes from the triangle inequalities presented in Section 3.2.1 with a 40%

decrease in solving time and a 89% decrease in the number of nodes. As with the primal heuristics, the strongest results are achieved when running all separation routines together.

Separator	time[s]		nodes		dual integral	
default	130.6	-	256.6	-	1213.3	-
triangle	77.6	**0.59**	28.8	**0.11**	474.1	**0.39**
subtour	122.6	0.94	174.3	**0.68**	1036.4	**0.85**
partition	122.6	0.94	121.6	**0.47**	933.1	**0.77**
all	68.3	**0.52**	15.4	**0.06**	396.6	**0.33**

TABLE 4.5: Comparison of solving time, number of nodes, and dual integral for the different separators. In each category, the value as well as the ratio compared to the default (Multinode-SCIP) is displayed.

Table 4.6 shows the same comparison on the part of the test set that was solved to optimality within the timelimit by the default. The general behavior is the same, all separation routines present an improvement compared to the default, with the highest impact achieved by the triangle-inequalities.

Separator	time[s]		nodes		dual integral	
default	19.7	-	211.9	-	126.4	-
triangle	9.3	**0.47**	20.7	**0.10**	40.6	**0.32**
subtour	17.2	**0.87**	136.3	**0.64**	102.1	**0.81**
partition	18.2	0.92	87.2	**0.41**	90.4	**0.72**
all	6.2	**0.31**	10.2	**0.05**	31.4	**0.25**

TABLE 4.6: Comparison of separators on the part of the test set that was solved optimally by the default (Multinode-SCIP).

Figure 4.6 shows the course of the average dual gap function over time for the different separation routines.

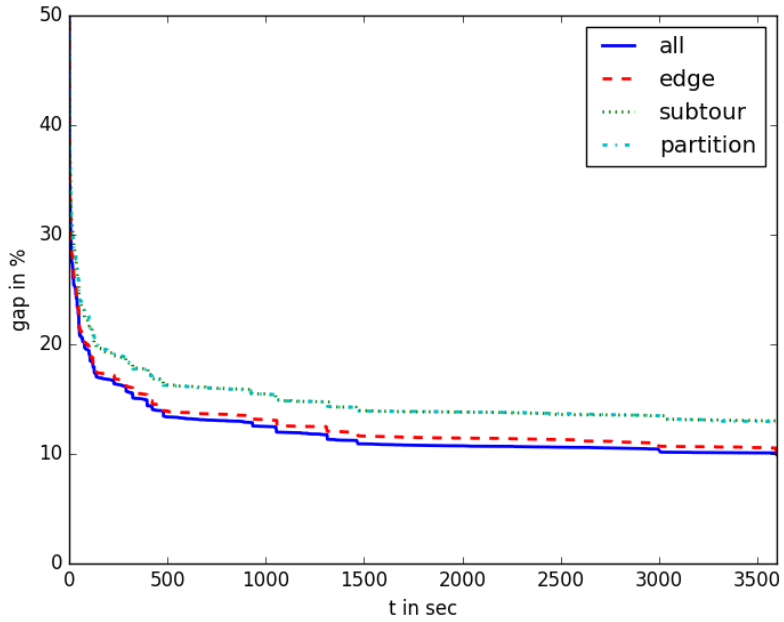


FIGURE 4.6: Average dual gap function over time for different separation routines.

4.5 Evaluation of Approximation Method

In this section, we evaluate the impact of the approximation method presented in Section 3.4. For each of the original instances in the test set, we have created three reduced instances with 25%, 50%, and 75% of transitions omitted, respectively.

Table 4.7 presents an overview of the results. The approximation gap is defined as $\frac{\delta}{v_{app}}$, where δ is the total approximation error defined as in equation (3.21) and v_{app} is the best known objective value for the approximate problem. The average solving time decreases with reducing the amount of transitions, however the average worst-case approximation gap also becomes substantial if 75% of the transitions are omitted.

Instances	solved	time[s]		approx gap
normal	30	96.2	-	-
reduced25	31	81.8	**0.85**	5.6%
reduced50	33	57.0	**0.59**	22.9%
reduced75	36	37.6	**0.39**	75.3%

TABLE 4.7: Evaluation of the approximation method. Column 'solved' shows how many of the 44 instances were solved within the time limit. Column 'time' sows the average solving time, as well as the ratio to the non-reduced case. The column labeled 'approx gap' displays the average approximation gap using Equation (3.21).

Better insight into the effectiveness of the method can be given, if we consider instances that can be solved optimally within the given timelimit but are not easy. On the smallest instances, omitting part of the transitions does not provide a large benefit as these instances are already solved very quickly. Restricting the test on the instances that can be solved optimally makes it possible to evaluate the actual reduction in solving time, without the constant offset provided by the instances that ran into the timelimit. Since the optimal value v_{opt} of the non-reduced problem is known for this part of the test set, we can compare the actual gap g between the approximate objective value v_{app} and v_{opt} as

$$g = \frac{|v_{opt} - v_{app}|}{\max\{|v_{opt}|, |v_{app}|\}}. \tag{4.1}$$

Table 4.8 shows results for instances that can be solved to optimality without reductions, but have a solving time greater than five seconds. When comparing the actual gap and the worst case approximation gap, we see that on this test set the actual gap is roughly one third of the size of the approximation gap in all three reduction cases.

Instances	time[s]		approx gap	actual gap
normal	41.9	-	-	-
reduced25	23.6	**0.56**	7.1%	2.1%
reduced50	7.8	**0.19**	28.2%	10.5%
reduced75	2.2	**0.05**	91.4%	30.4%

TABLE 4.8: Evaluation of the approximation methods on instances that were solved optimally with solving time of more than five seconds. Column 'actual gap' shows is the relative gap between reduced and non-reduced instances according to (4.1).

The approximate method can also be used to achieve better solutions of the original problem. Evaluating the best known approximate solution with respect to the objective function of the original non-reduced problem can be used as a primal heuristic.

We illustrate this at the example of a Hindmarsh-Rose instance with 250 states and 5 clusters. The best known solution after 3600s obtained by normal solving has an objective value of 0.195. If we use the best known solution after 3600s obtained by the 75% reduced instance as a heuristic, we get a feasible solution with objective value of 0.199. Furthermore, we can use the dual bound of the approximate solution together with the approximation error δ from Equation (3.21) to produce a dual bound for the original problem. In this instance, the best known dual bound for the original instance is 0.338. If we add the approximation error to the dual bound of the reduced instance, we get a dual bound of 0.235. The primal-dual gap using the original best solution and dual bound is 73.7%. If the solution obtained with the reduced instance and the new dual bound is

used, the gap is 18%. We chose to illustrate this method on an example since it is beneficial only if

- the problem cannot be solved optimally,
- the approximation error is small compared to the approximate solution,
- the dual bound for the reduced instance is tighter than for the original instance.

4.6 Comparing different Relaxations

In this last section, we compare three different relaxations for the cycle clustering problem. The first is the linear relaxation of the CC-MIP from Section 2.3. The second is the semidefinite programming relaxation (SDP), obtained by omitting the integrality requirements of the semidefinite integer program presented in Section 2.5. For the third relaxation, we use the automatic linearization generated by SCIP for the nonlinear program (Bilin) formulated in Section 2.3.

We used CVX, a package for solving convex programs [20, 21], to solve the SDP. Because no cutting plane framework for semidefinite programs was available and the semidefinite formulation has $\mathcal{O}(n^3)$ constraints, we only compare the relaxations on smaller instances with a number of states $n = 20, 30$.

Table 4.9 shows the dual gap for the three different relaxations. On average, the dual gap for the LP-relaxation of CC-SCIP is 2.5 times larger than for the semidefinite formulation. On the other hand, the average gap for the automatic linearization is roughly 2 times larger than the gap for CC-SCIP.

This suggests that the semidefinite formulation results in tighter relaxations than the linear formulation, at the cost of harder subproblems. We also conclude that the chosen problem-specific linearization for CC-SCIP seems effective for this type of problem.

Instance	Dual Gap		
	CC-SCIP	SDP	Bilinear
Pot3_30	35.7%	15.0%	77.5%
Pot_30	33.4%	15.3%	76.5%
Pot3NonCycle_20	34.8%	19.0%	77.4%
Pot3_20	26.5%	9.8%	72.6%
Pot4f_20	37.8%	13.0%	82.5%
Pot4s_20	39.4%	17.1%	83.3%
Pot4f_30	38.4%	13.7%	83.4%
Pot4s_30	39.5%	16.3%	83.8%
Pot4NonCycle_20	38.0%	17.9%	82.8%
Pot4NonCycle_30	39.5%	19.5%	83.9%
Pot4_20	32.4%	6.9%	32.1%
Pot4asym_20	34.4%	12.3%	33.7%
Pot4asym_30	35.5%	12.8%	82.5%
Pot6_20	30.7%	10.9%	30.3%
average	35.4%	**14.3%**	**70.2%**

TABLE 4.9: Comparison of the dual gap for different relaxations. Column 'CC-SCIP' is the gap for the LP-relaxation of CC-SCIP, 'SDP' for the relaxation from Section 2.5 and 'Bilinear' for the LP-relaxation of the nonlinear formulation from Section 2.3

Chapter 5

Conclusion and Outlook

In this thesis, we have introduced and studied cycle clustering, a graph partitioning problem which is an extension of the minimum k-partitioning problem. We have proven that this problem is $\mathcal{NP}$-hard, introduced a mixed integer programming formulation, and discussed the corresponding polytope.

We have introduced solving techniques for this problem and implemented them as a SCIP application called CC-SCIP. When comparing CC-SCIP to generic MIP solvers, we measured a speedup of factor $\times 4$ compared to the second fastest variant.

Three primal heuristics were presented that proved effective at obtaining feasible solutions for the clustering problem. Each individual heuristic improved the performance, reducing the average solving time and the average primal integral. Especially when run together these heuristics quickly found solutions close to the optimum or provided the best known solution in hard problem instances. When running these heuristics, the average primal integral decreased by over 90% compared to the default. We introduced three different types of valid inequalities. In practice, separating these inequalities reduced the average solving time by 68.5% compared to the default.

Furthermore, we discussed an approximation algorithm that reduces the instance size of the MIP by discarding entries in the transition matrix with small probabilities. Our tests have shown that this approximation method can greatly reduce solving time, with an average 80% reduction if half of the transitions are omitted. Since the approximation gap can only be computed after the test is run, it is best to do multiple tests with different reduction percentages. Usefulness of this approximation technique varies greatly among different types of instances.

Finally, we compared three relaxations on a test set of small instances. We have observed that the LP-relaxation of the problem-specific linearization gives a tighter dual bound than the automatic linearization. On the other hand, the SDP-relaxation provided a much tighter dual bound than the LP-relaxation, however at a higher computational cost.

In conclusion, the methods and algorithms presented in this thesis provided a significant improvement in the ability to solve the cycle clustering problem. However, large instances with 200 or more states could not be solved to optimality within the given time limit. Using parallelization in the branch-and-bound tree, e.g., with Para-SCIP [36], would be one way

to tackle larger instances. A different possibility is to use a branch-and-cut approach that uses semidefinte programming relaxations instead of linear relaxations. We have shown that this might lead to tighter relaxations and therefore a smaller branch-and-bound tree. More sophisticated inequalities with a richer structure and more advanced separation routines could lead to even tighter bounds and might be useful to improve the SCIP application further.

In order to tackle transition matrices with thousands of microstates, it is not realistic to expect a globally optimal MIP-solution. However, the presented heuristics can still generate locally optimal solutions and the LP-solution can provide a dual bound. A useful future extension of CC-SCIP in this direction might be to adapt other heuristic or approximate graph partitioning methods, such as coarsening or more general local search methods.

Bibliography

[1] T. Achterberg. *Constraint Integer Programming.* PhD thesis, 2007.

[2] T. Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, Jul 2009.

[3] M. F. Anjos, B. Ghaddar, L. Hupp, F. Liers, and A. Wiegele. *Solving k-Way Graph Partitioning Problems to Optimality: The Impact of Semidefinite Relaxations and the Bundle Method*, pages 355–386. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] I. Beckenbach, L. Eifler, K. Fackeldey, A. Gleixner, A. Grever, M. Weber, and J. Witzig. Mixed-integer programming for cycle detection in non-reversible markov processes. *Multiscale Modeling and Simulation*, 2016. under review.

[5] T. Berthold. *Heuristic algorithms in global MINLP solvers.* PhD thesis, 2014.

[6] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo.* CRC press, 2011.

[7] N. Cho and J. Linderoth. Row-partition branching for set partitioning problems. 2014.

[8] S. Chopra and M. R. Rao. The partition problem. *Mathematical Programming*, 59(1):87–115, 1993.

[9] N. D. Conrad, M. Weber, and C. Schütte. Finding dominant structures of nonreversible markov processes. *Multiscale Modeling and Simulation*, 14(4):1319 – 1340, 2016.

[10] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251. ACM, 1992.

[11] P. Deuflhard, W. Huisinga, A. Fischer, and C. Schütte. Identification of almost invariant aggregates in reversible nearly uncoupled markov chains. *Linear Algebra and its Applications*, 315(1):39 – 59, 2000.

[12] P. Deuflhard and M. Weber. Robust perron cluster analysis in conformation dynamics. *Linear Algebra and its Applications*, 398:161 – 184, 2005. Special Issue on Matrices and Mathematical Biology.

[13] R. Durrett. *Essentials of Stochastic Processes*. Springer Texts in Statistics. Springer New York, 2012.

[14] A. Eisenblätter. The semidefinite relaxation of the k-partition polytope is strong. In W. J. Cook and A. S. Schulz, editors, *Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO'02)*, volume 2337 of *Lecture Notes in Computer Science*, pages 273–290, Berlin Heidelberg, 2002. Springer-Verlag.

[15] Elowitz Michael B. and Leibler Stanislas. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, jan 2000. 10.1038/35002125.

[16] K. Fackeldey, A. Bujotzek, and M. Weber. A meshless discretization method for markov state models applied to explicit water peptide folding simulations. In *Meshfree Methods for Partial Differential Equations VI*, volume 89, pages 141 – 154, 2012.

[17] C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel, and L. A. Wolsey. The node capacitated graph partitioning problem: A computational study. *Mathematical Programming*, 81(2):229–256, Apr 1998.

[18] T. Gally, M. E. Pfetsch, and S. Ulbrich. A framework for solving mixed-integer semidefinite programs. *Optimization Methods and Software*, 0(0):1–39, 0.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[20] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[21] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar. 2014.

[22] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1):367–387, 1990.

[23] W. W. Hager, D. T. Phan, and H. Zhang. An exact algorithm for graph partitioning. *Mathematical Programming*, 137(1):531–556, Feb 2013.

[24] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[25] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[26] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[27] L. Liberti. Compact linearization for binary quadratic problems. *4OR*, 5(3):231–245, 2007.

[28] A. Lisser and F. Rendl. Graph partitioning using linear and semidefinite programming. *Mathematical Programming*, 95(1):91–101, Jan 2003.

[29] S. J. Maher, T. Fischer, T. Gally, G. Gamrath, A. Gleixner, R. L. Gottwald, G. Hendel, T. Koch, M. E. Lübbecke, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, D. Weninger, J. T. Witt, and J. Witzig. The scip optimization suite 4.0. Technical Report 17-12, ZIB, Takustr.7, 14195 Berlin, 2017.

[30] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[31] M. Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45(1):139–172, Aug 1989.

[32] V. S. Pande, K. Beauchamp, and G. R. Bowman. Everything you wanted to know about Markov State Models but were afraid to ask. *Methods*, 52:99–105, 2010.

[33] J.-H. Prinz, H. Wu, M. Sarich, B. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte, and F. Noe. Markov models of molecular kinetics: Generation and validation. *The Journal of Chemical Physics*, 134(17):174105, 2011.

[34] M. Sarich and C. Schütte. Markov model theory. In G. R. Bowman and V. S. P. and, editors, *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*, volume 797, pages 23 – 44. 2014.

[35] A. Shilnikov and M. Kolomiets. Methods of the qualitative theory for the hindmarsh-rose model: a case study - a tutorial. 18:2141–2168, 08 2008.

[36] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch. Parascip: a parallel extension of scip. In C. Bischof, H.-G. Hegering, W. Nagel, and G. Wittum, editors, *Competence in High Performance Computing 2010*, pages 135 – 148, 2012.

[37] M. Weber. A subspace approach to molecular markov state models via a new infinitesimal generator, 2011.

Appendix A

Auxiliary Proofs

The following Lemma is used in the proofs in this Appendix. It is symmetrical to Lemma 11, so we will state it without proof.

Lemma 19. *Let $m \geq 4$ and $ax + by + cz \leq \delta$ be a valid inequality that is satisfied at equality by the clusterings*

$$\begin{aligned} P_1 &= (A_1, A_2, A_3 \cup \{l\}, \ldots, A_m), & P_1' &= (A_1, A_2 \cup \{l\}, A_3, \ldots, A_m), \\ P_2 &= (A_2, A_1, A_3 \cup \{l\}, \ldots, A_m), & P_2' &= (A_2, A_1 \cup \{l\}, A_3, \ldots, A_m). \end{aligned}$$

Then

$$2c_{A_2,l} + b_{l,A_1} = 2c_{A_1,l} + b_{l,A_2}.$$

Theorem 20. *The inequality*

$$y_{ij} + y_{jk} - y_{ik} + 0.5\,(z_{ij} + z_{ji} + z_{jk} + z_{kj} - z_{ik} - z_{ki}) \leq 1 \tag{A.1}$$

defines a facet for CCP.

Proof. Assume there exists an inequality $ax + by + cz \leq \alpha$ such that

$$\begin{aligned} &\{(x, y, z) \in CCP \mid (x, y, z) \text{ satisfies (A.1) at equality }\} \\ &\qquad \subseteq \{(x, y, z) \in CCP \mid ax + by + cz = \alpha\}. \end{aligned}$$

We follow the same strategy as in the proof of Theorem 12 and use the same switch-trick.

Step 1: y-variables:

Let $l \in \mathcal{S}$ be any state. Let $(A_1, \ldots, A_m)$ be any clustering of $\mathcal{S} \setminus \{l\}$ with

- If $l \neq i$, we set $i, j \in A_2$.
- If $l = i$, we set $j, k \in A_2$.

We set the ordered clusterings as

$$\begin{aligned} P_1 &= (A_1 \cup \{l\}, A_2, A_3, \ldots, A_m), & P_1' &= (A_1, A_2, A_3 \cup \{l\}, \ldots, A_m), \\ P_2 &= (A_3 \cup \{l\}, A_2, A_1, \ldots, A_m), & P_2' &= (A_3, A_2, A_1 \cup \{l\}, \ldots, A_m). \end{aligned}$$

All of these clusterings satisfy (A.1) at equality and therefore we can apply Lemma 10 and it follows that

$$b_{l,A_1} = b_{l,A_3}.$$

Select any $u \in A_3, u \notin \{i,j,k\}$ and move u to A_2, i.e., we appyly Lemma 10 to

$$P_1 = (A_1 \cup \{l\}, A_2 \cup \{u\}, A_3 \setminus \{u\}, \dots, A_m),$$

and P_1', P_2, P_2' accordingly. It follows that

$$b_{l,A_1} = b_{l,A_3 \setminus \{u\}}.$$

Thus, as both equalities have the same left hand side

$$b_{l,A_3\setminus\{u\}} = b_{l,A_3} \quad \Rightarrow \quad b_{l,u} = 0 \quad \forall l \in \mathcal{S}, u \notin \{i,j,k\}.$$

Set $l = j$ and take any clustering $(A_1, \dots, A_m)$ of $\mathcal{S} \setminus \{l\}$ with $i \in A_1$ and $k \in A_3$. Then we can apply Lemma 10 once more and since we have already proven that all other coefficients are zero, we get

$$b_{ij} = b_{jk}. \tag{A.2}$$

We do the same with $l = i$ and $j, k \in A_1$ and get

$$b_{ij} + b_{ik} = 0. \tag{A.3}$$

Define $\beta \coloneqq b_{ij}$, then we proven so far that the inequality $ax + by + cz \leq \alpha$ has to be of the form

$$ax + \beta yij + \beta y_{jk} - \beta y_{ik} + cz \leq \alpha.$$

Step 2: z-variables

We use the switch-trick once more and use it together with Lemma 11 to prove that $c_{lu} = 0$ for $l \in \mathcal{S}, u \in \mathcal{S} \setminus \{i,j,k\}$. It works the same way as above, so we will not spell it out in detail again.

Let $l = i, A_2 = \{j,k\}$ and assign the rest of the states so that $(A_1, \dots, A_m)$ is a clustering of $\mathcal{S} \setminus \{i\}$. Then, using Lemma 11, we get

$$2(c_{ij} + c_{ik}) = b_{ik} + b_{ij} = \beta - \beta = 0.$$

If we do the same for $l = k, A_2 = \{i,j\}$, we get

$$c_{kj} + c_{ki} = 0.$$

If we use Lemma 19 in the same way we get

$$c_{ji} + c_{ki} = 0,$$
$$c_{jk} + c_{ik} = 0.$$

Setting $l = j, A_1 = \{i\} A_2 = \{k\}$ and using Lemma 11 gives us

$$c_{jk} = \frac{b_{jk}}{2} = \frac{\beta}{2},$$

and with $l = j, A_1 = \{k\}, A_2 = \{i\}$ we get

$$c_{ji} = \frac{b_{ji}}{2} = \frac{\beta}{2}.$$

So we now know that the inequality has to be of the form

$$ax + \beta y_{ij} + \beta y_{jk} - \beta y_{ij} + \frac{\beta}{2}(z_{ij} + z_{jk} + z_{ji} + z_{kj} - z_{ki} - z_{ik}) \leq \alpha. \quad \text{(A.4)}$$

Step 3: x-variables
Let $l \in \mathcal{S} \setminus \{i, j, k\}$. Let $(A_1 \cup \{l\}, \ldots, A_m)$ be a clustering with $i, j, k \in A_1$. The corresponding incidence-vector fulfills (A.1) at equality and if we switch l to a different cluster the inequality remains active. We know that all b, c-coefficients that contain l are zero. Thus, by comparing the coefficients for the incidence vectors of

$$P_1 = (A_1 \cup \{l\}, \ldots, A_m), \qquad P_2 = (A_1, \ldots, A_t \cup \{l\}, \ldots, A_m),$$

we immediately get

$$a_{l1} = a_{lt} \quad \forall t = 1, \ldots, m. \quad \text{(A.5)}$$

Let $l = i$, let j, k be in A_1 and set P_1 and P_2 as before. We compare the incidence vectors and get

$$a_{i1} + b_{ij} + b_{ik} = a_{i2} + c_{ji} + c_{ki} = a_{i3} = \ldots = a_{im-1} = a_{im} + c_{ij} + c_{ik}.$$

We know that $b_{ij} + b_{ik} = c_{ji} + c_{ki} = c_{ij} + c_{ik} = 0$. As i and k have the same role in the inequality, we get $a_{k1} = a_{kt}$ for all $t = 1, \ldots, m$. If $l = j$, we put $i \in A_1, k \in A_t$ and do the same as before. Once more, all the coefficients cancel each other and thus we have proven that for each $l \in \mathcal{S}$ there exists a constant α_l such that

$$a_{lt} = \alpha_l \quad \forall t = 1, \ldots, m.$$

Analogously to the proof of Theorem 12, we can assume that $a_{lt} = 0$ for all $l \in \mathcal{S}, t \in \mathcal{K}$. The only choice of δ that produces a valid inequality for CCP that defines a face is $\delta = \beta > 0$. This concludes the proof. □

Theorem 21. *Let $i, j, k \in \mathcal{S}$ with $(i, j), (j, k), (i, k) \in E$.*

- *If $m > 4$, then the inequalities*

$$z_{ij} + z_{ik} - y_{jk} \leq 1 \qquad \forall (i, j), (i, k), (j, k) \in E,$$
$$z_{ji} + z_{ki} - y_{jk} \leq 1 \qquad \forall (i, j), (i, k), (j, k) \in E$$

  are facet-defining for the cycle clustering polytope.

- *If $m = 4$, then*

$$z_{ij} + z_{ik} - 2y_{jk} - (z_{jk} + z_{kj} + z_{ji} + z_{ki}) \leq 0 \quad \forall (i, j), (i, k), (j, k) \in E$$

  define facets.

Proof. We prove the first point, i.e., let m > 4. Large parts of the proof are done similar to Theorem 20 and we will only explain the parts that differ in detail. Again we assume $ax + by + cz \leq \delta$ is a valid inequality for CCP such that

$$\{(x, y, z) \in CCP \mid z_{ij} + z_{ik} - y_{jk} = 1\} \subseteq \{(x, y, z) \in CCP | ax + by + cz = \delta\}.$$

As in Theorem 20, using the switch-trick we prove that

$$c_{lu} = c_{ul} = b_{lu} = 0 \text{ for all } u \in \mathcal{S}, l \in \mathcal{S} \setminus \{i, j, k\}, u \neq l.$$

Setting $i \in A_1, k \in A_2, l = j$ and using Lemma 10 yields

$$b_{ij} = 0,$$

and switching the roles of j, k yields

$$b_{ik} = 0.$$

Let $l = i, t \in \{1, \ldots, m\}$. Take any clustering $(A_1, \ldots, A_m)$ with $j \in A_2$ and $k \in A_{\phi(t)}$. Then both $(A_1 \cup \{i\}, \ldots, A_m)$ as well as $(A_1, \ldots, A_t \cup \{i\}, \ldots, A_m)$ fulfill the inequality at equality and we get

$$a_{i1} + c_{ij} = a_{it} + c_{ik}.$$

Switching the roles of j, k immediately yields

$$a_{i1} + c_{ik} = a_{it} + c_{ij}.$$

If we add both these inequalities, it follows that a_{it} is constant for all t and we can assume it to be zero. From the above equations it immediately follows that $c_{ij} = c_{ik}$. In the same way, it can be proven that $a_{lt} = 0$ for all $l \in \mathcal{S}, t \in \mathcal{K}$.

Consider the Clusterings

$$P_1 = (A_1 \cup \{i\}, A_2 \cup \{j\}, A_3 \cup \{k\}, \ldots, A_m),$$
$$P_2 = (A_1 \cup \{i\}, A_2 \cup \{j\}, A_3, A_4 \cup \{k\}, \ldots, A_m).$$

The incidence vectors of both satisfy (3.8) at equality and if we compare the coefficients we get

$$c_{jk} = 0.$$

Analogously, it can be proven that

$$c_{kj} = c_{ji} = c_{ki} = 0.$$

From

$$P_1 = (A_1 \cup \{i\}, A_2 \cup \{k, j\}, A_3, \ldots, A_m),$$
$$P_2 = (A_1 \cup \{i\}, A_2 \cup \{k\}, A_3 \cup \{j\}, \ldots, A_m)$$

follows

$$b_{jk} + c_{ik} = 0.$$

We define $\beta = c_{ij}$ and have now proven that the inequality has the form

$$-\beta y_{jk} + \beta(z_{ij} + z_{ik}) \leq \delta.$$

The only choice of β and δ that results in a valid inequality which defines a proper face is $\beta = \delta > 0$. The proof for the case that $m = 4$ can be done analogously. □

Theorem 22. *Let $A, B \subset \mathcal{S}, A \cap B = \emptyset, m > 4$ and let $||A| - |B|| = 1$. Then the partition inequality*

$$\sum_{i \in A, j \in B} z_{ij} - \sum_{i,j \in A, i<j} y_{ij} - \sum_{i,j \in B, i<j} y_{ij} \leq \min\{|A|, |B|\} \tag{A.6}$$

is facet-defining.

Proof. The proof that the inequalities are not-facet defining if $m = 4$ is given by the triangle inequality (3.8). It is a partition inequality with $|A| = 1$, $|B| = 2$. If $m = 4$, we have proven that (3.10) is of higher dimension. W.l.o.g. let $|A| = h, |B| = h + 1, h > 1$ and let $A = \{f_1, \ldots, f_h\}$, $B = \{g_1, \ldots, g_{h+1}\}$. We follow the same strategy as in all the proofs. Let $ax + by + cz \leq \delta$ be a valid inequality for CCP such that

$$\begin{aligned}\{(x, y, z) \in CCP | (x, y, z) \text{ satisfy (3.16) at equality}\} \\ \subseteq \{(x, y, z) \in CCP \mid ax + by + cz = \delta\}.\end{aligned}$$

The partition inequality is satisfied at equality by the clustering $(P_1, \ldots, P_m)$ if

a) $A \subseteq P_1, B \subseteq P_2$,

b) $A \subseteq P_1, B \setminus \{g_k\} \subseteq P_2$,

c) $A \setminus \{f_j\} \subseteq P_1, B \setminus \{g_k\} \subseteq P_2, f_j \in P_3, g_k \in P_4$,

d) $A \setminus \{f_j\} \subseteq P_1, B \setminus \{g_k, g_u\} \subseteq P_2, f_j \in P_3, g_k, g_u \in P_4$.

Using case a) and b), the switch-trick and Lemmas 10 and 11, it is straightforward to prove that $b_{lu} = c_{lu} = 0$ for all $l \in \mathcal{S}, u \in \mathcal{S} \setminus (A \cup B)$. In the same way as in the previous facet-defining proofs, it holds that $a_{it} = 0$ for all $i \in \mathcal{S}, t \in \mathcal{K}$.

We explicitly state the rest of the proof. To shorten notation, we always take case a) as the first clustering and then simply state what changes for the second clustering. For example, case c) is described as: move f_j to P_3, g_k to P_4.

Moving g_k to P_3, P_4 and P_m, respectively, we derive the equations

$$\sum_{i=1}^{h} c_{f_i, g_k} + \sum_{i \neq k} b_{g_i, g_k} = \sum_{i \neq k} c_{g_i, g_k} = 0 = \sum_{i=1}^{h} c_{g_k, f_i}. \tag{A.7}$$

Moving f_j to P_3, g_k to P_4 and f_j to P_{m-1}, g_k to P_m yields the equations

$$\sum_{i\neq j} b_{f_i,f_j} + \sum_{i\neq k} b_{g_i,g_k} + \sum_{i\neq j} c_{f_i,g_k} + \sum_{i\neq k} c_{f_j,g_i} = \sum_{i\neq k} c_{g_i,f_j} = \sum_{i\neq j} c_{g_k,f_i}. \quad \text{(A.8)}$$

We move f_j to A_3, g_k, g_u to A_4 as well as f_j to A_{m-1}, g_k, g_u to A_m and get

$$\sum_{i\neq j} b_{f_i,f_j} + \sum_{i\neq k,u} (b_{g_i,g_k} + b_{g_i,g_u}) + \sum_{i\neq j} (c_{f_i,g_k} + c_{f_i,g_u}) + \sum_{i\neq u,k} c_{f_j,g_i} = \sum_{i\neq k,u} c_{g_i,f_j} = \sum_{i\neq j} (c_{g_k,f_i} + c_{g_u,f_i}). \quad \text{(A.9)}$$

Moving f_j, g_u to A_3, g_k to A_4 and f_j, g_u to A_{m-1}, g_k to A_m yields

$$\sum_{i\neq k,u} c_{g_i,f_j} + c_{g_i,g_u} = \sum_{i\neq j} c_{g_k,f_i}. \quad \text{(A.10)}$$

Subtracting (A.9) from (A.10) proves that

$$-c_{g_u,f_j} + \sum_{i\neq k,u} c_{g_i,g_u} = 0 \Rightarrow c_{g_u,f_j} = \sum_{i\neq k,u} c_{g_i,g_u}.$$

This holds for all $j = 1, \ldots, h$. Therefore c_{g_u,f_j} is constant for all j and since $\sum_{i=1}^{h} c_{g_k,f_i} = 0$ it follows that

$$c_{g_i,f_j} = 0 \quad \text{for all } g_i \in B, f_j \in A.$$

We now look at the left-side equations. The equations (A.9) and (A.8) simplify to

$$\sum_{i\neq j} b_{f_i,f_j} + \sum_{i\neq k} b_{g_i,g_k} + \sum_{i\neq j} c_{f_i,g_k} + \sum_{i\neq k} c_{f_j,g_i} = 0, \quad \text{(A.11)}$$

$$\sum_{i\neq j} b_{f_i,f_j} + \sum_{i\neq u,k} (b_{g_i,g_u} + b_{g_i,g_k}) + \sum_{i\neq j} (c_{f_i,g_k} + c_{f_i,g_u}) + \sum_{i\neq u,k} c_{f_j,g_i} = 0. \quad \text{(A.12)}$$

Subtracting (A.11) from (A.12) yields

$$-b_{g_u,g_k} + \sum_{i\neq u,k} b_{g_i,g_u} + \sum_{i\neq j} c_{f_i,g_u} - c_{f_j,g_u} = 0. \quad \text{(A.13)}$$

We examine (A.7) once more.

$$\sum_{i=1}^{h} c_{f_i,g_u} + \sum_{i\neq k} b_{g_i,g_u} = 0 \Rightarrow \sum_{i\neq u,k} b_{g_i,g_u} + \sum_{i\neq j} c_{f_i,g_u} = -b_{g_u,g_k} - c_{f_j,g_u}$$

Putting this into (A.13) proves that

$$-2b_{g_1,g_2} - 2c_{f_1,g_2} = 0 \Rightarrow b_{g_i,g_j} = -c_{f_k,g_j}.$$

Since this holds for all i, j, and k, it follows that $c_{f_j,g_u} = -b_{g_u,g_k} = \beta$ is a constant for al k, j, and u.

If $|A| = 2$, then we can directly infer from (A.8) that $b_{f_1,f_2} = -b_{g_1,g_2}$. If $|A| \geq 3$, we move f_j, f_v to A_4, g_k, g_u to A_5. This yields the equation

$$\sum_{i\neq j,v}(b_{f_i,f_j}+b_{f_i,f_v})+\sum_{i\neq k,u}(b_{g_i,g_k}+b_{g_i,g_u})+\sum_{i\neq u,k}(c_{f_j,g_i}+c_{f_v,g_i})+\sum_{i\neq j,v}(c_{f_i,g_k}+c_{f_i,g_u}) = 0.$$

We subtract from this the equation (A.12). This yields

$$-b_{f_j,f_v} + \underbrace{\sum_{i\neq j,v} b_{f_i,f_v}}_{-b_{f_j,f_v}-(h-1)\beta} + \underbrace{\sum_{i\neq u,k} c_{f_v,g_i}}_{(h-1)\beta} \underbrace{-c_{f_v,g_k} - c_{f_v,g_u}}_{-2\beta} = 0.$$

It follows that

$$b_{f_j,f_v} = -\beta \quad \text{for all } f_j, f_v \in A.$$

The only thing that remains to be proven is that $b_{f_j,g_k} = c_{g_k,g_u} = 0$. Moving f_j, g_k to P_3, g_u to P_4 yields

$$b_{f_j,g_k} + c_{g_k,g_u} = 0.$$

Moving f_j to P_3, g_k to P_4 and g_u to P_5 yields

$$c_{g_k,g_u} = 0.$$

To summarize, we have proven that $ax + by + cz \leq \delta$ is of the form

$$\beta\left(\sum_{i\in A,j\in B} z_{ij} - \sum_{i,j\in A,i\neq j} y_{ij} - \sum_{i,j\in B,i\neq j} y_{ij} \leq \min\{|A|,|B|\}\right) \leq \delta.$$

This concludes the proof, as the only choice for δ, β that results in a valid inequality which defines a proper face is $\beta = \delta > 0$.

□

Appendix B

Test set

Instance	Vars	Cons	Instance	Vars	Cons
HindRose_250_13cl	21427	321390	Pot4Cycle_20_sym_4cl	650	3254
HindRose_250_5cl	19427	127494	Pot4Cycle_30_f_4cl	1425	7429
HindRose_250_7cl	19927	175968	Pot4Cycle_30_s_4cl	1425	7429
HindRose_50_13cl	1352	12465	Pot4Cycle_30_sym_4cl	1425	7429
HindRose_50_5cl	952	4969	Pot4NonCycle_20_4cl	650	3254
HindRose_50_7cl	1052	6843	Pot4NonCycle_30_4cl	1425	7429
Pot3Cycle_20_f_3cl	630	1923	Pot4_T_100_4cl	15250	84254
Pot3Cycle_20_s_3cl	630	1923	Pot4_T_20_4cl	650	3254
Pot3Cycle_20_sym_3cl	630	1923	Pot4_T_200_4cl	60440	338164
Pot3Cycle_30_f_3cl	1395	4383	Pot4_T_50_4cl	3875	20879
Pot3Cycle_30_s_3cl	1395	4383	Pot4asym_20_4cl	650	3254
Pot3Cycle_30_sym_3cl	1395	4383	Pot4asym_30_4cl	1425	7429
Pot3NonCycle_20_3cl	630	1923	Pot6_T_100_6cl	14850	118856
Pot3NonCycle_T_20_1t_3cl	630	1923	Pot6_T_20_6cl	684	4726
Pot3NonCycle_T_30_1t_3cl	1395	4383	Pot6_T_200_6cl	56619	462031
Pot3NonCycle_T_40_1t_3cl	2457	7833	Pot6_T_50_6cl	3891	29981
Pot3_T_100_3cl	15144	49583	rep_40_P_3cl	2460	7843
Pot3_T_20_3cl	630	1923	rep_40_P_6cl	2580	19546
Pot3_T_200_3cl	60216	198923	rep_200_P_3cl	60297	199193
Pot3_T_50_3cl	3825	12303	rep_200_P_6cl	60897	497681
Pot4Cycle_20_f_4cl	650	3254	rep_80_P_3cl	9714	31663
Pot4Cycle_20_s_4cl	650	3254	rep_80_P_6cl	9954	79036

TABLE B.1: Test set of all original instances with number of variables and number of constraints before presolving. The first number in the instance name is the number of states, the second number is the number of clusters for that instance.

instance	25		50		75	
	vars	cons	vars	cons	vars	cons
HindRose_250_13cl	17032	243707	12532	164207	8098	85873
HindRose_250_5cl	15032	96691	10532	65191	6098	34153
HindRose_250_7cl	15532	133445	11032	89945	6598	47083
HindRose_50_13cl	1193	9651	1037	6895	866	3874
HindRose_50_5cl	793	3851	637	2759	466	1562
HindRose_50_7cl	893	5301	737	3793	566	2140
Pot3Cycle_20_f_3cl	516	1543	399	1153	258	683
Pot3Cycle_20_s_3cl	498	1483	384	1103	258	683
Pot3Cycle_20_sym_3cl	489	1882	351	1284	228	751
Pot3Cycle_30_f_3cl	1113	3443	822	2473	504	1413
Pot3Cycle_30_s_3cl	1089	3363	804	2413	492	1373
Pot3Cycle_30_sym_3cl	1065	4258	765	2958	450	1593
Pot3NonCycle_20_3cl	486	1443	345	973	207	513
Pot3NonCycle_T_20_1t_3cl	486	1443	360	1023	237	613
Pot3NonCycle_T_30_1t_3cl	1068	3293	765	2283	465	1283
Pot3NonCycle_T_40_1t_3cl	1872	5883	1314	4023	771	2213
Pot3_T_100_3cl	11736	38223	8025	25853	4311	13473
Pot3_T_20_3cl	537	1613	405	1173	261	693
Pot3_T_200_3cl	45975	151453	31050	101703	16125	51953
Pot3_T_50_3cl	3051	9723	2136	6673	1218	3613
Pot4Cycle_20_f_4cl	542	2642	422	1962	281	1163
Pot4Cycle_20_s_4cl	530	2574	419	1945	281	1163
Pot4Cycle_20_sym_4cl	527	2557	407	1877	260	1044
Pot4Cycle_30_f_4cl	1155	5899	861	4233	534	2380
Pot4Cycle_30_s_4cl	1128	5746	861	4233	534	2380
Pot4Cycle_30_sym_4cl	1122	5712	840	4114	501	2193
Pot4NonCycle_20_4cl	515	2489	410	1894	281	1163
Pot4NonCycle_30_4cl	1125	5729	834	4080	525	2329
Pot4_T_100_4cl	11836	64908	8125	43879	4411	22833
Pot4_T_20_4cl	560	2744	425	1979	281	1163
Pot4_T_200_4cl	46175	257329	31250	172754	16325	88179
Pot4_T_50_4cl	3104	16510	2186	11308	1268	6106
Pot4asym_20_4cl	524	2540	416	1928	281	1163
Pot4asym_30_4cl	1131	5763	855	4199	534	2380
Pot6_T_100_6cl	11328	89506	7788	60006	4188	30006
Pot6_T_20_6cl	543	3551	423	2551	312	1626
Pot6_T_200_6cl	43377	351681	28830	230456	14157	108181
Pot6_T_50_6cl	2991	22481	2145	15431	1299	8381
rep_40_P_3cl	1974	6223	1404	4323	825	2393
rep_40_P_6cl	2094	15496	1524	10746	945	5921
rep_200_P_3cl	45975	151453	31050	101703	16125	51953
rep_200_P_6cl	45975	151453	31050	101703	16125	51953
rep_80_P_3cl	7542	24423	5208	16643	2850	8783
rep_80_P_6cl	7782	60936	5448	41486	3090	21836

TABLE B.2: Test set of all reduced instances before presolving, using the approximation scheme form Section 3.4. Columns '25', '50', and '75' denote which percentile of transitions was removed.