

Université Catholique de Louvain

Faculté des Sciences Appliquées

Unité de Gestion Industrielle

**THE GRAPH EQUIPARTITION PROBLEM: OPTIMAL SOLUTIONS,
EXTENSIONS AND APPLICATIONS**

Thèse présentée en vue de l'obtention du Grade de Docteur en Sciences Appliquées

par

Cid Carvalho de Souza

Promoteur: Professeur Laurence A. Wolsey

Louvain-La-Neuve, November 1993.

Université Catholique de Louvain

Faculté des Sciences Appliquées

Unité de Gestion Industrielle

**THE GRAPH EQUIPARTITION PROBLEM: OPTIMAL SOLUTIONS,
EXTENSIONS AND APPLICATIONS**

Thèse présentée en vue de l'obtention du Grade de Docteur en Sciences Appliquées

par

Cid Carvalho de Souza

Promoteur: Professeur Laurence A. Wolsey

Jury: Prof. G. de Ghellinck

Prof. M. Grötschel

Prof. R. Keunings

Prof. E. Loute

Prof. L. A. Wolsey

Louvain-La-Neuve, November 1993.

To *Tania*
and *Lucas*

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to my supervisor Prof. *Laurence A. Wolsey*, for his attention to my research in the last four years. His constant guidance and encouragement have been essential for the accomplishment of this work. It has been a pleasure and a privilege to have him as supervisor.

I am grateful to the members of my committee for their helpful comments and suggestions: Prof. *Guy de Ghellinck* (UCL), Prof. *Martin Grötschel* (Konrad-Zuse Zentrum, Berlin), Prof. *Roland Keunings* (UCL) and Prof. *Etienne Loute* (Facultés St. Louis and UCL). I have tried to incorporate all of their contributions to the final text of the thesis.

At different stages of this work, I have benefited from the collaboration of many people.

I would like to express my gratitude to Prof. *Monique Laurent* who has made possible the extension of several of my old results in Chapter 2, either by combining them with her own results or by suggesting me the good directions to follow. I also thank her for helping me to keep my list of references on graph partitioning problems up to date.

I am very grateful to *Robert Weismantel*, *Alexander Martin* and to *Carlos Ferreira* from the Konrad-Zuse Zentrum (Berlin). Not only they have kindly provided me with their own implementation of a branch-and-cut code for graph partitioning problems, but also they have given me a valuable assistance in many computational matters of Chapter 4. Some of the results in Chapter 3 are part of our joint research project which has been a very rewarding work.

I would like to thank Prof. *Roland Keunings* for introducing me to the combinatorial problem treated in Chapter 5 which is at the origin of our joint research. I have deeply appreciated his support, as well as that of other researchers of the Division of Applied Mechanics (MEMA, UCL), along the project. Among the researchers of MEMA, I would like to mention *Olivier Zone*, *Alain Couniot* and *Denis VanderStraeten*.

I am greatly indebted to Prof. *Celso Ribeiro* (PUC-RJ, Brazil) for his most special friendship and continuous incentive. He has introduced me to Integer Programming and Combinatorial Optimization and made possible my return to the academic life. Especially

in the hard moments, the support I had from him has gone far beyond I could have expected and, perhaps, deserved.

I also thank Prof. *Nelson Maculan* (UFRJ, Brazil) and Prof. *Yves Pochet* (UCL) for their support.

I would like to thank my friends and fellows at CORE: *El-Houssaine Aghezzaf*, *Miguel Constantino*, *Karen Aardal*, *Francisco Ramos*, *Peng Kuang-Kai* and *Patrick Watteyne* for stimulating discussions on various topics and for their constant encouragement.

I wish to thank all the secretaries at CORE who have helped me in many different ways.

My stay at CORE was financed by a fellowship from the Conselho Nacional de Pesquisa e Desenvolvimento (CNPq) of Brazil. The financial support is greatly acknowledged and certainly recognized as the enabling condition for all the rest.

Some of my travel expenses in Europe have been financed by the Science Training Program (SC1-CT91-0620) of the European Economic Community and by the Fond National pour la Recherche Scientifique (Belgium). These financial supports are also acknowledged.

I would like to take the opportunity to thank some friends with whom I shared so many good moments in Louvain-La-Neuve: *José Salim*, *Joseneide* and their wonderfully funny children; *Márcio* and *Rosângela*; *Luiz Gutierrez* and *Hideko*; *João* and *Lili*. Though the enormous physical distance between us, other people have also given me an important support. I thank them all and, in particular, I would like to mention *Airton* and *Nadiméa*.

I wish to express my gratitude to my parents *Altair* and *Ingrid*, my sister *Clarisse* and my brother *Ibá* whose love have given me the strength to overcome so many obstacles in my life.

Last, but not least, I would like to thank my wife *Tania* for her great love and unlimited patience. I would have never been able to go so far in this adventure without her. I am sorry for the high bill she had to pay. On the other hand, I am extremely happy for having her as my coauthor in the (undoubtedly) most beautiful work I (we) have ever done: our lovely son *Lucas* !

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 Scope	1
1.2 Graph Theory Background	5
1.3 Integer Programming Formulations and Algorithms	9
1.4 Polyhedral Theory Background	17
1.4.1 Basic Definitions	17
1.4.2 Characterizing Facets	19
1.4.3 Separation and Optimization Problems	23
1.5 Literature Survey	25
1.6 Thesis Outline	31
2. VALID INEQUALITIES FOR THE EQUIPARTITION PROBLEM	35
2.1 Introduction	35
2.2 Models for Equipartition and Clustering Problems	37
2.3 Preliminary Results for the Equipartition and Cut Polytopes	42
2.4 The <i>Path-Block Cycle</i> Inequalities	46
2.5 The <i>Suspended Tree</i> Inequalities	64
2.6 Extensions of PBC and Suspended Tree Inequalities	89
2.7 Further Remarks	95

3. VALID INEQUALITIES FOR SOME GRAPH PARTITIONING PROBLEMS	97
3.1 Introduction	97
3.2 The Single Cluster Problem	100
3.3 The Graph Partitioning Problem with Cardinality Capacity Constraints	108
3.4 The Graph Partitioning Problem with Knapsack Capacity Constraints	126
 4. SEPARATION ROUTINES AND COMPUTATIONAL RESULTS FOR GRAPH PARTITIONING PROBLEMS	 132
4.1 Introduction	132
4.2 Separation Routines	134
4.2.1 Separation Routine for the Cycle Inequalities	136
4.2.2 Separation Routine for the PBC Inequalities	143
4.2.3 Separation Routine for the Tree Inequalities	150
4.2.4 Separation Routine for the Knapsack Tree Inequalities	155
4.3 About the Branch-and-Cut Code	159
4.4 Problem Instances	165
4.5 Computational Results and Discussion	174

5. THE FRONTWIDTH REDUCTION PROBLEM IN FINITE ELEMENT COMPUTATIONS	190
5.1 Introduction	190
5.2 The Finite Element Concept and the Frontal Method	193
5.2.1 The Finite Element Concept	193
5.2.2 The Frontal Method	196
5.2.3 Existing Heuristic Algorithms for the FRP	200
5.3 Standard and Modified Frontwidth Problems	203
5.4 A Divide-and-Conquer Heuristic Algorithm for $\overline{\text{FRP}}$	207
5.5 Heuristics for Minimum Weight Equipartition	213
5.5.1 Kernighan and Lin (KL)	214
5.5.2 Simulated Annealing (SA)	216
5.5.3 Stochastic Evolution (SE)	218
5.6 Initial Solution for Equipartition	221
5.7 Computational Results and Discussion	222
 6. CONCLUSION	 232
 REFERENCES	 238
 NOTATION	 247

1. Introduction

1.1 Scope

In this thesis we consider three different graph optimization problems: the equipartition problem, the clustering problem and the cutwidth problem.

In the equipartition problem we want to find a partition of the nodes of a graph into two subsets of equal size such that the sum of the costs of the edges with endnodes in different subsets is minimized.

In the clustering problem weights are assigned to each node of the graph and we are interested in partitions of the node set into a fixed number of subsets such that the sum of the node weights in each subset of the partition is bounded by a constant. The goal is again to minimize the sum of the costs of the edges joining nodes in different subsets of the partition. This problem generalizes the equipartition problem and is also known in the literature as the graph partitioning or multicut problem.

Let V be the set of nodes in a graph. An ordering of V is a one-to-one function that assigns labels in the set $\{1, 2, \dots, |V|\}$ to the nodes in V . In the cutwidth problem an ordering of V has to be found that optimizes a min-max cost function where the cost of an ordering is computed as follows. Given an integer k in $\{1, 2, \dots, |V|\}$, consider the set of nodes with label greater than k and count the number of edges going from this set to its complement in V . The maximum of such values computed for all possible values of k is defined to be the cost of the ordering. So, the cutwidth problem is the problem of finding an ordering of minimum cost.

The graph optimization problems described above can be used to model a large variety of practical optimization problems. Applications are found in VLSI design, compiler design, finite element computations and physics.

These three problems belong to the class of NP-hard problems and two different approaches can be used to tackle them. In the first we look for optimal solutions and for this exact algorithms are necessary. In the second we look for good, but not necessarily optimal, solutions. In the latter case, heuristic algorithms have to be developed.

In this thesis our goal is to develop exact algorithms for the equipartition and clustering problems and to develop a high performance heuristic for the cutwidth problem on graphs arising from Finite Element meshes.

We choose an algorithm for solving problems exactly which is based on cutting planes. This leads us to look for strong valid inequalities for the convex hull (polytope) of feasible solutions of the integer programming formulations of the equipartition and clustering problems. Several classes of facet defining inequalities for the equipartition polytope are introduced in Chapter 2. In Chapter 3, we derive facet defining inequalities for the polytopes related to three clustering problems: the single cluster polytope, the graph partitioning polytope with cardinality capacity constraints and the graph partitioning polytope with knapsack capacity constraints. In Chapter 4, we develop routines that can automatically generate some of the inequalities introduced in Chapters 2 and 3 and we give the computational results we have obtained with our exact algorithm. Optimal solutions for small instances of practical problems are reported.

One way to solve a graph optimization problem to optimality is to formulate it as a 0-1 Integer Programming problem (0-1 IP problem). Then, in principle, exact algorithms for general IP problems, such as branch-and-bound, can be used to find an optimal solution. However, this approach is often not effective in practice since the branch-and-bound algorithm is too slow.

In the last decade cutting plane algorithms for IP have been used successfully to solve reasonably sized instances of hard combinatorial optimization problems such as the Traveling Salesman Problem (TSP). The design of a cutting plane algorithm is characterized

by two phases. The first phase is devoted to the study of the convex hull (polyhedron) of the feasible solutions of the IP problem. The idea is to determine strong valid inequalities for this polyhedron in order to improve the Linear Programming (LP) formulation of the problem (which is obtained from the IP formulation by relaxing the integrality constraints). Typically, the number of such inequalities grows exponentially with the size of the input data and it becomes impractical to add them all to the initial LP formulation. So, a second phase in the design of a cutting plane algorithm is necessary where we consider the problem of generating valid inequalities as they are needed. The cutting plane algorithm can be applied in each node of a branch-and-bound tree in which case we speak of a branch-and-cut algorithm.

The success obtained by cutting plane (branch-and-cut) algorithms in solving practical instances of the Traveling Salesman Problem, one of the most challenging combinatorial optimization problems, was a motivation for the study that is carried on in this thesis for the equipartition and clustering problems.

The cutwidth problem is used here as an approximation for the problem of reducing the frontwidth of Finite Element meshes. The choice for a heuristic approach to tackle it is initially motivated by the large sizes of the practical instances we have to treat. Another reason to opt for a heuristic approach is that the IP formulation of the cutwidth problem has far too many variables.

In Chapter 5 we describe the heuristics we propose for the cutwidth problem. These heuristics are based on local search strategies which have been implemented by deterministic and nondeterministic algorithms. In the computational experiments we have carried out, our heuristic has outperformed the standard Reverse Cuthill-McKee algorithm for frontwidth reduction in finite element meshes and, in most cases, the relative improvements in frontwidth are in the range 25-50%.

In the implementation of our heuristic, we have used two nondeterministic local search algorithms, namely, the Simulated Annealing and the Stochastic Evolution algorithms.

The use of nondeterministic algorithms in combinatorial optimization started in the early eighties with the paper by Kirkpatrick et al. (1982) on simulated annealing. Since then, numerous applications of the simulated annealing algorithm in a large spectrum of combinatorial optimization problems have appeared in the literature (see, van Laarhoven and Aarts, 1987). The flexibility of the simulated annealing algorithm, which allows us to adapt it easily to many different problems, but specially the high quality of the solutions it produces when compared to other deterministic heuristics can be pointed out as the two main reasons why the algorithm has been so popular. Other nondeterministic algorithms, like stochastic evolution (see, e.g., Saab and Rao, 1991) and tabu search (see, e.g., Glover and Laguna, 1992) came later and they usually present alternative ways to improve on the simulated annealing algorithm.

The present chapter is organized as follows. In the next section we formalize the definitions of the equipartition, clustering and cutwidth problems in terms of graph optimization problems. Some definitions and terminology from graph theory that are used in the text are also given. In Section 3 we show how graph optimization problems can be formulated in general as 0-1 IP problems and, in particular, we give an IP formulation for the clustering problem. Methods for solving IP problems to optimality including the branch-and-bound and branch-and-cut algorithms are also discussed in Section 3. The effectiveness of a cutting plane algorithm heavily depends on the strength of the inequalities it adds to the LP formulation (relaxation) of the problem. Thus, in Section 4 we introduce some basic concepts of polyhedral theory that provide us with the tools to characterize how strong an inequality is. Section 5 contains a literature survey on the polyhedral investigations that have been carried out for equipartition, clustering and related problems. The survey also includes a brief literature review relative to the cutwidth problem. Finally, in Section 6 we give the thesis outline.

1.2 Graph Theory Background

A graph $G = (V, E)$ consists of a finite set V of nodes (or vertices) and a set E of edges joining different pairs of distinct nodes. When the edges are defined by unordered pairs of nodes the graph is *undirected*, otherwise it is a *directed* graph. All graphs treated here are undirected. In the graphical representation of a graph, each node is indicated by a point (or a circle) and each edge by a line linking the points that represent its ends.

Let V_1 and V_2 be two disjoint subsets of V . The *cutset* or *cut* of V_1 and V_2 , denoted by $\delta(V_1, V_2)$, is the set of edges with one end in V_1 and the other in V_2 . If $V_2 = V \setminus V_1$, the notation is abbreviated to $\delta(V_1)$ ($=\delta(V_2)$) and we refer simply to the cutset of V_1 . This definition can be extended naturally for more general partitions. If V_1, V_2, \dots, V_k is a partition of V , $\delta(V_1, V_2, \dots, V_k)$ denotes the cutset of the partition, i.e., the set of edges with ends in different subsets of the partition. When V is partitioned into more than 2 subsets, the resulting cutset is also called a *multicut*.

With the definitions above we can formalize the equipartition, the clustering and the cutwidth problems for graph G as follows.

Equipartition Problem: Suppose that a cost c_e is given to every edge $e \in E$. Find a subset $U \subset V$ with $|U| = \lceil \frac{|V|}{2} \rceil$ that minimizes $\sum_{e \in \delta(U)} c_e$.

Clustering Problem: Suppose that costs c_e are associated to every edge $e \in E$ and that weights w_u are associated to every node $u \in V$. Given an integer K and values F_1, F_2, \dots, F_K (cluster capacities), let $\Pi(F_1, \dots, F_K)$ denote the family of partitions of graph G into K subsets of nodes V_1, \dots, V_K (clusters) such that $\sum_{u \in V_i} w_u \leq F_i$ for all $i \in \{1, \dots, K\}$. Find a partition of G in $\Pi(F_1, \dots, F_K)$ that minimizes $\sum_{e \in \delta(V_1, \dots, V_K)} c_e$.

Cutwidth Problem: Suppose that an ordering is given to the node set V that makes a one-to-one correspondence between the labels in $\{1, \dots, |V|\}$ and the nodes of the graph. For $i \in \{1, \dots, n\}$, let V^i denote set of nodes with label at most i . The

cutwidth of an ordering is computed as $\max\{|\delta(V^i)| : i \in \{1, \dots, n\}\}$. Find an ordering of V of minimum cutwidth.

It can be seen from these definitions that the equipartition problem is a particular case of the clustering problem where the constant K and the F_i are taken as $K = 2$, $F_1 = \lceil \frac{|V|}{2} \rceil$ and $F_2 = \lfloor \frac{|V|}{2} \rfloor$, and all node weights are set to 1. The cutset corresponding to an equipartition of the nodes in V is called an *equicut*. So, the equipartition problem is also called the equicut problem.

Consider the graph in Figure 1.1. Assume that all edge costs are taken to be 1. The nodes $\{v_1, v_2, v_3, v_4, v_5\}$ define a feasible equipartition of the graph with a cost of 4. Suppose that an ordering is given to the nodes of the graph such that the label of node v_i is equal to i . The cutwidth of such an ordering is 5 since $|\delta(\{v_1, \dots, v_6\})| = 5$ and $|\delta(\{v_1, \dots, v_i\})| \leq 5$ for all $i = 1, \dots, 9$.

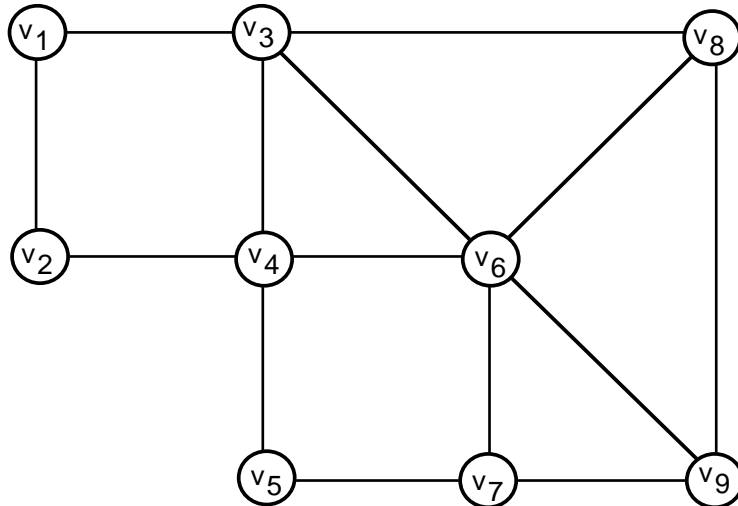


Figure 1.1:

Like many other graph optimization problems, the equipartition, clustering and cutwidth problems are NP-hard. In broad terms, this means that any exact algorithm known to solve one of these problems needs CPU time that grows exponentially with the number of nodes in the graph. Thus, one can only expect to solve exactly equipartition, clustering and cutwidth problems for graphs with a limited number of nodes. The goal of using a cutting plane approach for solving these problems is precisely to push up these limits. For a detailed introduction to complexity theory we refer to Garey and Johnson (1979).

In the remainder of this section we give some basic definitions from graph theory that we use throughout the text. Additional definitions will be given as they are needed. For more on graph theory we refer to Bondy and Murty (1976).

A graph is called *complete* if there exists an edge joining each pair of distinct nodes. The complete undirected graph on n nodes is denoted by \mathcal{K}_n . A graph is said to be *planar* if it can be drawn on a plane without edges crossing.

Two nodes i and j are *adjacent* if the edge $e = (i, j)$ is in E . On the other hand, if a node i is one of the ends of an edge e , e is said to be *incident* with i . The number of incident edges with a node i is the *degree* of i .

A graph $H = (V(H), E(H))$ is a *subgraph* of G if $V(H) \subseteq V$ and $E(H) \subseteq E$. Let V' be a nonempty subset of V and E' the subset of edges in E with both ends in V' . The subgraph $G' = (V', E')$ is called the subgraph of G *induced* by V' ; we say that G' is an induced subgraph of G .

A *path* is a sequence of nodes (v_1, v_2, \dots, v_k) such that v_i and v_{i+1} are adjacent for $i \in \{1, \dots, k-1\}$. Clearly, a path can also be defined in terms of its edges. A *simple* (*elementary*) path is a path which does not use the same edge (node) more than once. All paths treated here are simple and elementary.

A *cycle* is a path (v_1, v_2, \dots, v_k) in which the initial node v_1 coincides with the final node v_k . A simple cycle is a cycle that does not use the same edge more than once. Elementary cycles are those in which the initial node is used exactly twice and all remaining nodes are used exactly once. Unless the contrary is specified, all cycles we treat here are simple and elementary.

Two vertices u and v of G are said to be *connected* if there exists a path in G with initial node u and final node v . The subgraph induced by a subset of nodes V' of V such that all pair of nodes in V' are connected and is maximal with respect to this property forms a (connected) *component* of G . The graph is connected if it has exactly one component, otherwise it is *disconnected*.

An *acyclic* graph is one that contains no cycle. A *tree* is a connected acyclic graph. The nodes of the tree with degree one are called the *leaves* of the tree. A *star* is a tree in which all nodes are leaves except one node called the *center* of the star. A graph made of the union of disconnected trees is a *forest*.

1.3 Integer Programming Formulations and Algorithms

A Linear Programming (LP) problem can be defined as the problem of optimizing a linear function over a region described by a set of linear inequality and equality constraints. The points in this region form the set of feasible solutions of the LP problem. Such a problem can be written in matrix form as:

$$\min\{cx : A^{(<)}x \leq b^{(<)} , A^{(=)}x = b^{(=)} , x \in \mathbb{R}_+^n\}$$

where $c \in \mathbb{R}^n$, $A^{(<)}$ is a $m^{(<)} \times n$ matrix, $A^{(=)}$ is a $m^{(=)} \times n$ matrix, $m = m^{(=)} + m^{(<)}$ and $b \in \mathbb{R}^m$. If the variables x are constrained to be integer ($x \in \mathbb{Z}^n$ instead of $x \in \mathbb{R}^n$), the problem is called an Integer Programming (IP) problem. Moreover, if the variables are constrained to take values in $\{0, 1\}$, we have a 0-1 IP problem. The feasible solutions of the IP (0-1 IP) problem are the integer (0-1) points satisfying the linear system. Sometimes it is convenient to tackle a graph optimization problem by formulating it as a 0-1 Integer Programming problem. Below we discuss how to obtain such a formulation.

Consider a graph $G = (V, E)$ where $|V| = n$ and $|E| = m$. Let \mathcal{B}^d denote the set of 0-1 vectors of dimension d (vectors with all d components equal to either 0 or 1).

Suppose that x is a vector in \mathcal{B}^m and let E' be the subset of edges $e \in E$ such that the e -th component of x is equal to 1 ($x_e = 1$). Then x is called the *incidence* or *characteristic* vector of E' . Conversely, given an edge subset E' , the incidence vector of E' is obtained by setting x_e to 1 if edge e is in E' and to 0 if not. Equivalently, a vector y in \mathcal{B}^n is the incidence vector of a subset of nodes in V . The components of vector x are called the *edge variables* and those of the vector y are the *node variables*.

Typically in a graph optimization problem, one has to find a subgraph of G satisfying a set of properties which minimizes a given objective function. Representing the subgraphs of G by their incidence vectors $(x, y) \in \mathcal{B}^{m+n}$, the set of properties that characterize the feasible subgraphs can usually be expressed in terms of a system of linear inequalities

$Ax + By \leq b$ and the objective function in terms of a linear function $cx + wy$. In this case, the graph optimization problem has a 0-1 IP formulation of the form:

$$\min\{cx + wy : Ax + By \leq b, (x, y) \in \mathbb{B}^{m+n}\}$$

Consider, for instance, the 0-1 IP formulation of the clustering problem given below, where the x and y variables are interpreted as follows:

$$x_{uv} = \begin{cases} 1 & \text{if edge } (u, v) \text{ is in the multicut defined by the partition} \\ & \text{(that is, nodes } u \text{ and } v \text{ are in different clusters)} \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_u^k = \begin{cases} 1 & \text{if node } u \text{ belongs to the } k\text{-th cluster of the partition} \\ 0 & \text{otherwise} \end{cases}$$

An IP formulation of the clustering problem is then:

$$\min \quad \sum_{e \in E} c_e x_e$$

$$\text{Subject to } \sum_{\substack{u \in V \\ K}} w_u y_u^k \leq F_k \quad \forall k \in \{1, \dots, K\} \quad (\text{I})$$

$$\sum_{k=1}^K y_u^k = 1 \quad \forall u \in V \quad (\text{II})$$

$$y_u^k + y_v^\ell - x_{uv} \leq 1 \quad \forall k \neq \ell \in \{1, \dots, K\} \\ \forall (u, v) \in E \quad (\text{III})$$

$$y_u^k + y_v^k + x_{uv} \leq 2 \quad \forall k \in \{1, \dots, K\} \\ \forall (u, v) \in E \quad (\text{IV})$$

$$y_u^k \in \{0, 1\} \quad \forall u \in V, \forall k \in \{1, \dots, K\} \quad (\text{V})$$

$$x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E \quad (\text{VI})$$

When K is the number of clusters in the partition, this formulation contains $n \times K$ node variables and m edge variables. Constraint (I) says that all cluster capacities must be satisfied; constraint (II) makes sure that every node of G is assigned to a cluster; constraint (III) forces an edge to be in the multicut if its endnodes are in different clusters; constraint (IV) forces an edge to be out of the multicut if its endnodes are in the same cluster, and

constraints (V) and (VI) are the 0-1 integer constraints. In the LP relaxation of this problem, the last two constraints are replaced by the constraints below:

$$\begin{aligned} 0 \leq y_u^k \leq 1 \quad & \forall u \in V, \forall k \in \{1, \dots, K\} \\ 0 \leq x_{uv} \leq 1 \quad & \forall (u, v) \in E \end{aligned}$$

The IP formulation given above for the clustering problem has two sets of variables: one representing the nodes (the y variables) and the other representing the edges (the x variables). Thus, we call this formulation the *node-edge model*. Noting that the cost function only has terms in edge variables, one can think of working only with edge variables. This gives rise to the *edge model* for the clustering problem.

We define S as the set of feasible solutions of the node-edge model and the projection of S into the space of edge variables is denoted by X , that is:

$$X = \{x \in \mathbb{B}^m : \exists y \text{ with } (x, y) \in S\}$$

Thus, the clustering problem can be written as $\min\{\sum_{e \in E} c_e x_e : x \in X\}$. The advantage of the edge model when compared to the node-edge model is the fact that it has $n \times K$ less variables. However, one of the disadvantages of using the edge model is that it may be difficult to find a compact linear system $Ax \leq b$ for which the set of integer solutions is precisely X .

The theoretical results for the equipartition (Chapter 2) and clustering (Chapter 3) problems in this thesis have been obtained for edge models, while for the computational results (Chapter 4) we have used the node-edge model. The distinctions between the models, and some reasons for such a choice will be discussed later in Chapter 2. For the moment, we focus our attention on the methods that are available for solving 0-1 IP problems.

Usually 0-1 IP problems are NP-hard. One way to tackle such a problem is to solve linear relaxations of it. As remarked above, in the LP relaxation the integer constraints $(x, y) \in \mathbb{B}^{m+n}$ are replaced by linear constraints of the type $0 \leq (x, y) \leq 1$ and $(x, y) \in$

\mathbb{R}^{m+n} . There are two classical approaches to solving 0-1 IPs using linear approximations: the fractional cutting plane algorithm (FCPA) and the branch-and-bound or implicit enumeration algorithm. These algorithms are briefly described below. For the remainder of this section, S denotes the set of feasible solutions of the 0-1 IP problem.

The FCPA algorithm is based upon the strengthening of the LP relaxation with the addition of valid inequalities to the formulation. An inequality is said to be *valid* with respect to S if it is satisfied by all points in S .

At each iteration i of FCPA, a linear relaxation LP^i of the IP problem is solved. Let (x^i, y^i) be an optimal solution obtained for the linear relaxation LP^i . If (x^i, y^i) is in S , the algorithm stops, returning (x^i, y^i) as an optimal solution of the IP problem. Otherwise, the relaxation has to be strengthened. For this, a valid inequality $\pi x + \mu y \leq \pi_0$ for S has to be found that is violated by (x^i, y^i) . A new iteration has to be executed where the relaxation LP^{i+1} is obtained from LP^i by adding the inequality $\pi x + \mu y \leq \pi_0$ to the current set of linear constraints. Let z^i and z^{i+1} denote the value of the optimal solutions of LP^i and LP^{i+1} respectively, that is, $z^i = cx^i + wy^i$ and $z^{i+1} = cx^{i+1} + wy^{i+1}$. Assuming that the IP is a minimization problem, then $z^{i+1} \geq z^i$ or, in other words, the lower bound on the optimal solution of the IP problem monotonically increases as the iterations go on.

The idea of the FCPA is illustrated in Figure 1.2 for an IP (maximization) problem on 2 variables. The region limited by the black lines represents the set of feasible solutions of the LP relaxation. The dots indicate the integer points. The points in S are denoted by the black dots. The arrow indicates the direction of maximization and its tail is located at the optimal LP (fractional) solution. The dashed line represents a valid inequality that cuts off this fractional solution.

The initial study of adding valid inequalities for general IP was carried out by Gomory in the late fifties. Unfortunately, the inequalities he proposed to add to the formulation did not prove to be efficient in practice since the algorithm becomes too slow.

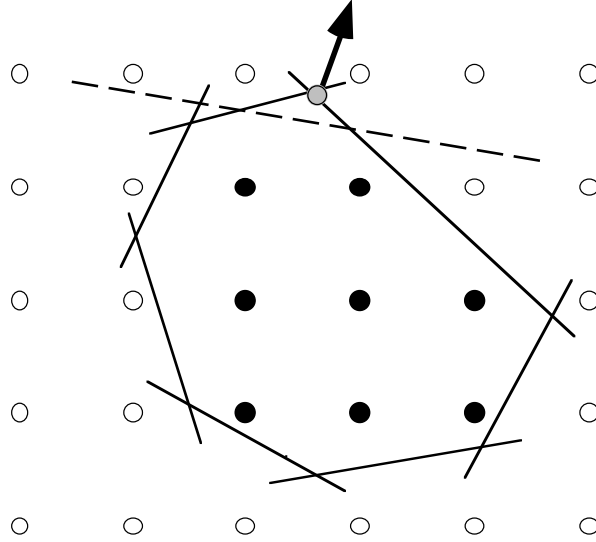


Figure 1.2:

The branch-and-bound approach is based on the general principle of *divide-and-conquer*, i.e., if it is too hard to optimize over the set S , the set S is partitioned into smaller subsets with the hope that optimizing is possible on these subsets, and then the results can be merged to get the optimal solution in S .

Frequently, the partition is constructed recursively. This allows a graphical representation of the whole process in terms of a tree: the *enumeration tree*. In this representation, the sons of a given node form the partition of the feasible region of their father.

In general, for 0-1 IP problems, the enumeration tree is a binary tree. Each node i of the tree corresponds to a linear relaxation LP^i of the IP problem defined on a subset S^i of S . Let (x^i, y^i) be the optimal solution found for LP^i and $z^i = cx^i + wy^i$. Depending on the value of z^i , node i can give rise to two other nodes (its sons) or can be *pruned*, i.e., the subset of feasible solutions of node i is either partitioned into two new subsets or it will not be partitioned any more during the remaining steps of the algorithm.

Suppose that we decide for the partition at a node i . Then, a variable ξ_j with a fractional value in (x^i, y^i) is chosen. This variable is called the *branching variable*. The linear relaxation for the first (second) son of node i is obtained by adding the constraint $\xi_j = 0$ ($\xi_j = 1$) to LP^i . Therefore, in a given node of the enumeration tree, the set of feasible solutions is a subset of S with some variables fixed to 1 and some others fixed to zero. Clearly, if no nodes are pruned, we have a complete enumeration of the solutions in S .

Some criteria are used to avoid branching at a node of the tree (the *bounding phase*). Clearly, in a minimization problem, the optimal value in a given node can never be greater than the optimal value of its sons. Therefore, if we know an upper bound for the optimal value of the original IP problem, we can prune all nodes whose linear relaxations give optimal values that are greater than that bound. Moreover, the nodes for which the optimal solution is integral can also be pruned. The nodes of the tree that are not pruned are said to be *active nodes*. The algorithm stops when there are no more active nodes.

Another method for solving 0-1 IP problems embeds a *cutting plane* phase in the *branch-and-bound* algorithm. This gives rise to a *branch-and-cut* algorithm. To aid in describing such an algorithm, we first consider how an integer program can be tackled, at least partially, by linear programming.

Let $\text{conv}(S)$ denote the convex hull of the feasible set S . The convex hull of S is a polyhedron and therefore can be described as a system of linear inequalities and equations. If this linear system is available, the IP problem can, in principle, be solved by linear programming since all extreme points are integer feasible solutions in S . Unfortunately, the number of inequalities in the system is exponential for NP-hard problems and usually only few inequalities of the description of $\text{conv}(S)$ are known. However, assume that a certain class \mathcal{F} of strong valid inequalities for $\text{conv}(S)$ are known. Moreover, given any point $(x, y) \in \mathbb{R}_+^n$, assume that an algorithm is available that looks for an inequality in \mathcal{F} violated by the point (x, y) . Such an algorithm is called a separation routine (this

nomenclature will become clear in the next section). The branch-and-cut algorithm can now be described.

In a typical iteration of the algorithm we are at a node i of the enumeration tree and $P^i = \{(x, y) \in \mathbb{R}^{m+n} : A^i x + B^i y \leq b, 0 \leq (x, y) \leq 1\}$ is the polytope corresponding to the linear relaxation LP^i . If (x^i, y^i) is the optimal solution of this linear relaxation and it has fractional components (variables), the separation routine is called to look for a violated inequality in \mathcal{F} . If the separation routine returns an inequality $\pi x + \mu y \leq \pi_0$, this inequality is added to the system of inequalities defining P^i and LP^i is solved again. We keep doing this until: either (x^i, y^i) is integral, or z^i is greater than the current upper bound available, or the separation routine fails to produce a new violated inequality cutting the point (x^i, y^i) . In this last case, a variable is to be chosen for branching.

Successful applications of branch-and-cut algorithms for the solution of difficult 0-1 IP problems of large practical size have been reported. Examples can be found in: Padberg and Grötschel (1985) and Padberg and Rinaldi (1987) for the Traveling Salesman Problem; Crowder et al. (1983) for general large-scale 0-1 IP problems; Grötschel et al. (1984) for the Linear Ordering problem; Barahona et al. (1988) for max cut problems arising from circuit layout design and statistical physics or in Grötschel et al. (1992) for the design of communication networks. Among others, these results have stimulated many authors to design exact algorithms for other NP-hard problems within the same framework, as is the case in this thesis.

The implementation of a branch-and-cut code is a major software development effort (see for instance Padberg and Rinaldi, 1987). The efficiency of the code depends on many factors ranging from data management to algorithmic aspects. Some of these algorithmic aspects are discussed below.

First, the cutting phase is only attractive if a family \mathcal{F} of "strong" valid inequalities is known. Finding such a family \mathcal{F} implies a theoretical study of the polyhedron describing

the convex hull of the feasible solutions. On the other hand, it also requires the design of an algorithm to generate violated inequalities in \mathcal{F} . This algorithm, called a separation routine for \mathcal{F} , has to be fast and must have a high probability of returning a violated inequality in \mathcal{F} when one exists.

But finding strong valid inequalities and separation routines is not enough to have a good branch-and-cut code.

There are other sensitive parts of the algorithm that are intrinsic to the branch-and-bound algorithm. Thus, it is necessary to have heuristics to generate integer feasible solutions from the fractional optimal solutions of the relaxed problems (which can produce upper bounds that will prune some nodes of the enumeration tree) and a good branching criterion. Further improvements include a preprocessing phase which, for instance, can eliminate variables a priori (by fixing them to 0 or 1).

For more on enhancements and different possible strategies that can be used in implementing branch-and-bound and branch-and-cut algorithms we refer to the book of Nemhauser and Wolsey (1988).

We have mentioned that the branch-and-cut algorithm is only useful when \mathcal{F} is a class of strong valid inequalities for $\text{conv}(S)$. Thus, we would like to characterize the strength of a valid inequality. In the next section, we introduce some basic concepts of Polyhedral Theory that allow us to discuss such questions.

1.4 Polyhedral Theory Background

In this section we summarize the definitions and fundamental results in polyhedral combinatorics that are used in this thesis. We start with some basic concepts like polyhedron, valid inequality, face and facets. Afterwards, we present two ways to characterize facets. We close the section by introducing the Separation and optimization Problems, together with the fundamental result establishing their equivalence (in the sense of computational complexity).

1.4.1 Basic Definitions

Definition 1.1 A set of points $x^1, \dots, x^k \in \mathbb{R}^n$ is linearly independent if the unique solution to $\sum_{i=1}^k \lambda_i x^i = 0$ is $\lambda_i = 0$ for all $i \in \{1, \dots, k\}$.

Definition 1.2 A set of points $x^1, \dots, x^k \in \mathbb{R}^n$ is affinely independent if the unique solution to $\sum_{i=1}^k \lambda_i x^i = 0$, $\sum_{i=1}^k \lambda_i = 0$ is $\lambda_i = 0$ for all $i \in \{1, \dots, k\}$.

Definition 1.3 Let $S = \{x^1, \dots, x^k\}$ be a set of points in \mathbb{R}^n . The convex hull of S is the set of points given by

$$\text{conv}(S) = \left\{ \sum_{i=1}^k \lambda_i x^i : \sum_{i=1}^k \lambda_i = 1, x^i \in S, \lambda_i \in \mathbb{R}, \lambda_i \geq 0, i = 1, \dots, k \right\}$$

The affine hull of S is the set of points given by

$$\text{aff}(S) = \left\{ \sum_{i=1}^k \lambda_i x^i : \sum_{i=1}^k \lambda_i = 1, x^i \in S, \lambda_i \in \mathbb{R}, i = 1, \dots, k \right\}$$

Definition 1.4 A set $S \subseteq \mathbb{R}^n$ is of dimension k , denoted by $\dim(S) = k$, if the maximum number of affinely independent points in S is $k + 1$. S is said to be full-dimensional if $\dim(S) = \dim(\mathbb{R}^n) = n$.

Definition 1.5 A polyhedron $P \subseteq \mathbb{R}^n$ is a set of points that satisfy a finite number of linear inequalities, i.e., $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. If a polyhedron P is bounded, i.e., $P \subseteq \{x \in \mathbb{R}^n : -w \leq x_j \leq w, \text{ for all } j = 1, \dots, n\}$ for some $w \in \mathbb{R}_+$, P is called a polytope.

If S is a finite set of points in \mathbb{R}^n , the convex hull of S is a polytope and, moreover, each vertex of $\text{conv}(S)$ is in S . For instance, consider a graph $G = (V, E)$ where $|E| = m$. Let S be the set of points in \mathbb{R}^m corresponding to the incidence vectors of equicuts of G . Then, $\text{conv}(S)$ is called the equicut (or equipartition) polytope.

Definition 1.6 *The cone generated by a set $S \subseteq \mathbb{R}^n$, denoted by $\text{cone}(S)$, is the set of points given by*

$$\text{cone}(S) = \{y \in \mathbb{R}^n : y = \lambda x, \text{ for some } x \in \text{conv}(S) \text{ and some } \lambda \in \mathbb{R}_+\}$$

Definition 1.7 *The rank of a matrix A , denoted by $\text{rank}(A)$, is the maximum number of linearly independent rows (columns) of A .*

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ where (A, b) is an $m \times (n + 1)$ matrix. Let $M^- = \{i : a^i x = b_i \text{ for all } x \in P\}$. Let (A^-, b^-) be the rows of (A, b) corresponding to M^- .

Proposition 1.8 *If $P \subseteq \mathbb{R}^n$, then $\dim(P) = n - \text{rank}(A^-, b^-)$.*

This means that if P is not full-dimensional, at least one of the inequalities in the description of P is satisfied at equality by all points in P .

Definition 1.9 *An inequality $\pi x \leq \pi_0$ is valid for a polyhedron P if it is satisfied by all points in P .*

Definition 1.10 *Let $\pi x \leq \pi_0$ be a valid inequality for a polyhedron P . The set $F = \{x \in P : \pi x = \pi_0\}$ is called a face of P (F is the face defined in P by the inequality $\pi x \leq \pi_0$). A face is said to be proper if $F \neq \emptyset$ and $F \neq P$.*

Definition 1.11 *Let F be a face of a polyhedron $P \subseteq \mathbb{R}^n$. If $\dim(F) = \dim(P) - 1$, then F is a facet of P .*

Definition 1.12 *The support of an inequality $\pi x \leq \pi_0$ is given by the set $\{j \in \{1, \dots, n\} : \pi_j \neq 0\}$.*

Suppose that the x variables are in one-to-one correspondence with the edges of a graph G . Given the inequality $\pi x \leq \pi_0$, let S be the support of this inequality. The subgraph of G whose edges are indexed by the elements of S is called the *support graph* of the inequality $\pi x \leq \pi_0$. The graphical representation of the support graph is usually helpful for the understanding of an inequality.

We now explain what we meant by *lifting*. Given an inequality $\pi x \leq \pi_0$ valid with respect to a polyhedron $P \subseteq \mathbb{R}^n$, let F_π denote the face of P given by $\{x \in P : \pi x = \pi_0\}$. Suppose that there exists another inequality $\mu x \leq \mu_0$, valid with respect to P , which defines the face F_μ in P . Then, the inequality $\mu x \leq \mu_0$ is said to be a lifting of the inequality $\pi x \leq \pi_0$ if the following holds:

- (i) $F_\pi \subset F_\mu$
- (ii) $\dim(F_\pi) < \dim(F_\mu) \leq \dim(P) - 1$

Clearly, if an inequality is facet defining, then it cannot be lifted.

1.4.2 Characterizing Facets

There are two methods commonly used to characterize facets of a polyhedron. Given a valid inequality $\pi x \leq \pi_0$ of a polyhedron $P \subseteq \mathbb{R}^n$, the first method (direct), consists to exhibit $\dim(P)$ affinely independent points in the set $\{x \in P : \pi x = \pi_0\}$. A second alternative (indirect method) comes up from the following theorem:

Theorem 1.13 *Let (A^-, b^-) be the equality set of $P \subseteq \mathbb{R}^n$ and let $F = \{x \in P : \pi x = \pi_0\}$ be a proper face of P . The following statements are equivalent:*

- i) F is a facet of P .
- ii) If $\lambda x = \lambda_0$ for all $x \in F$, then

$$(\lambda, \lambda_0) = (\alpha\pi + uA^-, \alpha\pi_0 + ub^-)$$

for some $\alpha \in \mathbb{R}_+$ and some $u \in \mathbb{R}^{|A^-|}$.

The characterization given in the above theorem is used for most of our proofs that valid inequalities are facet defining. The search for facets of a polyhedron is justified in the next paragraphs.

Consider the IP problem $\min\{cx : x \in S\}$ where S is a finite set of integer points in \mathbb{R}^n (i.e., $S \subset \mathbb{Z}^n$). This problem can be solved by any linear programming algorithm, if we know a system of linear inequalities $Ax \leq b$ that describes $\text{conv}(S)$ (since the extreme points of $\text{conv}(S)$ are in S). If every inequality in $Ax \leq b$ is facet defining for $\text{conv}(S)$ and, conversely, every facet F of $\text{conv}(S)$ is defined by exactly one inequality in $Ax \leq b$, then $Ax \leq b$ is a minimal system for $\text{conv}(S)$. The previous statement summarizes the importance of the concept of facet.

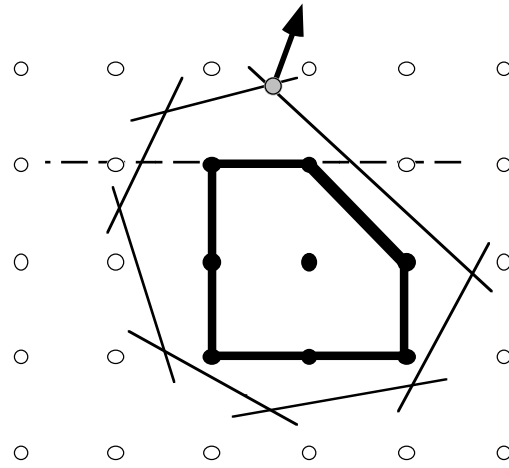
The use of a minimal system describing $\text{conv}(S)$ is usually not possible in practice. There are two main reasons for this. The first is that the number of inequalities in the minimal system is frequently exponential. This is true not only for NP-hard problems, but also for some polynomially solvable problems. The second reason is that, in general, we do not know all the inequalities that describe $\text{conv}(S)$, i.e., there are families of facet defining inequalities that are unknown. This is often the case for NP-hard problems.

However, for a given objective function, the IP problem can be solved by linear programming if the system of linear inequalities corresponding to a relaxation of $\text{conv}(S)$ contains the inequalities defining an optimal solution. This suggests the use of an algorithm for the solution of the IP problem that recursively improves the linear relaxation of $\text{conv}(S)$ by adding new inequalities (preferably facets) with the hope that, at some point, the optimal solution of the relaxation will be in S . This is the way how the fractional cutting plane algorithm (FCPA) of Section 1.3 works. But here there is an attempt to make the algorithm faster by adding preferably inequalities that are facet defining for $\text{conv}(S)$.

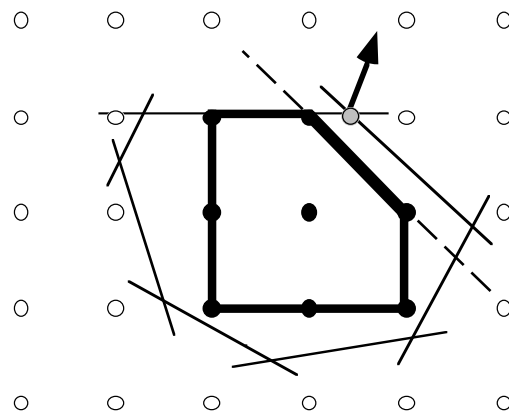
Figure 1.3 illustrates the idea of a cutting plane algorithm based on the use of facet defining inequalities. The example given in this figure is the same as that in Figure 1.2.

The region limited by the bold lines represents the convex hull of S . In Figure 1.3(a) the optimal LP (fractional) solution is cut off by the facet defining inequality represented by the dashed line. The optimal solution of the new (strengthened) formulation is again fractional but can be cut off by the facet defining inequality indicated in Figure 1.3(b). Although the next LP formulation (Figure 1.3(c)) does not describe $\text{conv}(S)$, the optimal LP solution is in S and therefore it is an optimal solution for the IP problem.

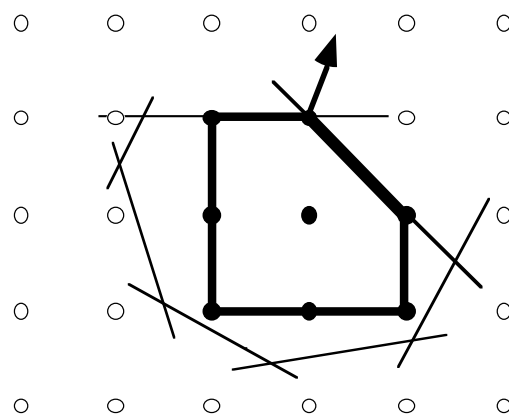
As stated above, it is crucial in a cutting plane algorithm to be able to generate violated inequalities. In the next subsection the problem of the identification of violated inequalities is addressed. This problem is known as the separation problem and we will see that, at least in theory, it is as difficult to solve as the optimization problem.



(a)



(b)



(c)

Figure 1.3:

1.4.3 Separation and Optimization Problems

Consider the following problems:

Separation Problem for a Family of Polyhedra:

Given a point $y \in \mathbb{R}^n$ and a polyhedron P in the family, decide whether or not y belongs to P and, if not, find an inequality $\pi x \leq \pi_0$ that is valid for P and such that $\pi y > \pi_0$.

Optimization Problem for a Family of Polyhedra:

Given a vector $c \in \mathbb{R}^n$ and a polyhedron P in the family, assume that $P \neq \emptyset$ and that cx is bounded for all $x \in P$. Find a solution $x^* \in P$ such that $cx^* \leq cx$ for all $x \in P$.

Separation Problem for a Family \mathcal{F} of Inequalities:

Given a point $y \in \mathbb{R}^n$, show that y satisfies all the inequalities in \mathcal{F} , or find one or more inequalities $\pi x \leq \pi_0$ in \mathcal{F} such that $\pi y > \pi_0$.

Clearly, if we consider the family of polyhedra that are completely described by the family \mathcal{F} of inequalities, the first and the third problems are equivalent.

The next theorem, due to Grötschel, Lovasz and Schrijver (1981), shows that the complexities of the separation and optimization problems are equivalent or, in other words, these two problems are equally difficult to solve.

Theorem 1.14 *For a family of polyhedra, there exists a polynomial algorithm for the separation problem if and only if there exists a polynomial time algorithm for the optimization problem.*

Let $\text{conv}(S)$ be the convex hull of feasible solutions of a 0-1 IP problem that we want to solve using a cutting plane algorithm. At each iteration of the algorithm $\text{conv}(S)$ is represented by a linear relaxation $\text{LP}(\mathcal{F})$, where \mathcal{F} is a family of valid inequalities for $\text{conv}(S)$. The number of inequalities in \mathcal{F} is typically exponential and, therefore, $\text{LP}(\mathcal{F})$ only contains a few of them.

If $\text{conv}(S)$ can be fully described using only inequalities in \mathcal{F} and the IP problem is polynomially solvable, then Theorem 1.14 implies that the separation problem is polynomially solvable. This means that, during the cutting plane algorithm, we can always find in polynomial time an inequality in \mathcal{F} that cuts the fractional solution.

On the other hand, if the IP problem is NP-hard and \mathcal{F} represents one or more families of strong valid inequalities for $\text{conv}(S)$, there may be some families of inequalities in \mathcal{F} for which the separation problem is polynomial, but, from Theorem 1.14, there will be other families for which the separation problem is NP-hard. Therefore, there will be some instances of the IP problem for which the cutting plane algorithm will terminate with a fractional solution.

The next section is devoted to a literature survey on the problems that we tackle in this thesis.

1.5 Literature Survey

There has been a large research effort on clustering (or graph partitioning) problems for many years, but many of the problem variants still remain among the most difficult and challenging combinatorial optimization problems. In this survey, we particularly focus our attention on works involving polyhedral studies of clustering problems. We divide the survey into four parts. The first is devoted to the max cut problem which is the most studied case of graph partitioning problems. The second part of the survey refers to the equipartition problem, while in the third part we review the literature for more general graph partitioning problems. Finally, the fourth part refers to the cutwidth problem.

Max Cut Problem

In the classical cut problem we look for a partition of V into at most two subsets. The cut problem has many applications such as in network flow design. The cut polytope has been studied in Barahona and Mahjoub (1986). There the authors identify several classes of facet defining inequalities and establish many fundamental results for the cut polytope. One of these results is that all facets of the cut polytope can be obtained by knowing only the facets that define one of its vertices. The cone generated by the points in the cut polytope is the cut cone. The previous result implies that getting a characterization of the cut cone is as hard as getting a characterization of the cut polytope itself.

In the light of the previous remark, the same authors have started investigating the polyhedral properties of the cut cone. Important works that follow this research direction are: Deza and Laurent (1992a, 1992b, 1992c), Deza, Grishukin and Laurent (1991), Deza, Laurent and Poljak (1992), De Simone, Deza and Laurent (1989).

A cut is defined to be even (odd) if it is of even (odd) size. Deza and Laurent (1989) study the polytope defined by the convex hull of the incidence vectors of even (odd) cuts of a complete graph. They introduce some classes of facet defining inequalities for these polytopes and give the relationship between these inequalities and facet defining

inequalities for the cut polytope.

A graph is called *bipartite* if its node set V can be partitioned into two nonempty disjoint sets V_1 and V_2 such that no two nodes of V_1 and no two nodes of V_2 define an edge. The convex hull of the incidence vectors of all edge sets of bipartite subgraphs of a graph G is the bipartite subgraph polytope of G . Given any cut $\delta(W)$ of G , it is clear that the subgraph $H = (V, \delta(W))$ is bipartite. Therefore, polyhedral results concerning the bipartite subgraph polytope can be useful for the cut polytope. Grötschel and Pulleyblank (1981) use these ideas to design polynomial time algorithms for special cases of the max cut problem. These algorithms have been implemented by Barahona and Maccioni (1982) for the solution of medium-sized real-world problems. Further investigations of the bipartite subgraph polytope can be found in Barahona, Grötschel and Mahjoub (1985) and Gerards (1985).

A cutting plane algorithm for the max cut problem, with applications to statistical physics and circuit layout design, is given in Barahona et al. (1988). The authors report that the max cut problem has been solved for graphs with up to 1600 nodes.

Boros and Hammer (1993) present a one-to-one correspondence between the valid inequalities for the cut polytope of the complete graph on $n + 1$ nodes (\mathcal{K}_{n+1}) and the set of nonnegative quadratic functions on n Boolean variables. They obtain a large class of valid and facet defining inequalities that includes many of the previously known inequalities for the cut polytope. In fact, the correspondence mentioned above is given for the Boolean Quadric polytope of the complete graph on n nodes. Then, the result is extended to the cut polytope using the fact that the valid inequalities for the Boolean Quadric Polytope of \mathcal{K}_n are in a one-to-one correspondence with the valid inequalities of the cut polytope of \mathcal{K}_{n+1} (Padberg, 1989).

Recently Deza and Laurent (1992d) list some applications of the cut polytope and also of related polyhedra, namely, the boolean quadric, the hypermetric and metric polytopes.

Applications are reported in the fields of functional analysis, geometry of numbers, quantum mechanics and multicommodity flow problems.

Algorithmic approaches can be found for some polynomially solvable versions of the max cut problem. Hadlock (1975) gives a polynomial time algorithm for the planar version of the problem, and in Hochbaum and Shmoys (1985) an efficient algorithm for the problem of partitioning a graph into three connected components is proposed.

Equipartition Problem

The equipartition polytope is studied in Conforti et al. (1990a,b). The dimension as well as basic and more complex classes of facet defining inequalities are introduced. Some of their results are generalized in this thesis.

Deza, Fukuda and Laurent (1989) investigate the relationship between the cut cone, the equipartition polytope and the inequicut cone (i.e. the cone generated by all incidence vectors of cuts that do not form equipartitions). They show that the inequicut cone and the equipartition polytope "inherit" all facets of the cut cone.

Nonpolyhedral approaches to the equipartition problem can be found in Liebling and Vaca (1991), where a dynamic programming based algorithm that polynomially solves the problem for trees, and in Arbib (1988) where a polynomial algorithm is given for partitioning a line-graph.

Different local search heuristics have been proposed for the equipartition problem. The most well known such heuristic is due to Kernighan and Lin (1970). This heuristic is discussed in detail in Chapter 5. Johnson et al. (1989) develop an algorithm for the equipartition problem based on simulated annealing. In the latter paper the authors compare the computational results produced by pure local search, Kernighan-Lin's heuristic and simulated annealing. The simulated annealing algorithm is also discussed in Chapter 5 of this thesis.

Graph Partitioning Problem

Chopra and Rao (1989a,b) study three different formulations of the clustering problem. For the problems they consider, a fixed integer number K , with $1 \leq K \leq n$, is given. There are no node weights and capacity constraints. In these formulations the nodes of V are to be partitioned into r subsets where either $r \leq K$, or $r \geq K$ or $r = K$. Results relative to the dimension, basic and more complex facet defining inequalities for the corresponding polytopes are presented. These models cover a large spectrum of graph partitioning problems.

Chopra and Rao (1989c) introduce more facets of the partitioning polytope for the case where the number of subsets in the partition is at most K . For this problem, Chopra (1991) gives a system of linear inequalities that completely describes the convex hull of the incidence vectors of all feasible partitions of the node set V when G is a series-parallel, or 4-wheel free graph.

Deza, Grötschel and Laurent (1991), using a computer program, give a complete linear description of many different graph partitioning polytopes defined on complete graphs of small size ($n \leq 5$). The same authors (1992) study a class of *clique web inequalities* for various graph partitioning polytopes, characterizing the conditions under which they induce facets and exhibiting a subclass of them for which the separation problem can be solved in polynomial time.

Grötschel and Wakabayashi (1987, 1989, 1990) investigate the polytope associated to the clustering problem when there are no restrictions on the number of subsets in the partition and on the number of nodes in each subset. They call this problem the *clique partitioning* problem.

Johnson et al. (1991) consider the clustering problem with no restriction on the number of subsets that form the partition but with capacity constraints associated to the subsets (consequently, node weights are assigned to the nodes of the graph). They suggest a

column generation scheme to solve the problem where the choice of the column entering the basis is made by solving an optimization problem in which we look for a subset of the node set V such that the sum of the weights of the nodes in it is not greater than a constant and the sum of the edges in the induced subgraph is minimized. This subproblem is a NP-hard problem. The authors investigate the facial structure of the corresponding polytope. Classes of valid and facet defining inequalities are presented together with heuristic separation routines for some of them. Computational results for problems arising in compiler design are reported.

Weismantel (1992) studies a variant of the clustering problem in which each node is not necessarily assigned to a cluster. The number of clusters in this variant is fixed and the cluster capacities are all the same. The author presents several classes of facet defining inequalities for the corresponding polytope.

Aghezzaf (1992) studies the polytope associated to the problem of partitioning a tree into subtrees, where weights are assigned to the nodes and the subtrees have weight capacities.

Other approaches can be found in the literature. Barnes (1982) considers the problem of partitioning a graph into at most k subsets and suggests a heuristic which involves the solution of a linear programming transportation problem. Bui and Peck (1992) consider the problem of partitioning a planar graph into two subsets of given size and cutting a minimum number of edges. They obtain an $O(b^2 n^3 2^{4.5b})$ for this problem where b is the value of the optimal solution. Feo et al. (1992) considered the problem of partitioning a graph into exactly K subsets of $\frac{n}{K}$ nodes each. The objective is to maximize the total weight of the edges not cut by the partition (within the clusters). For the cases where K equals 3 and 4, the authors proposed a heuristic which yields solutions that are at least one-half the weight of the optimal solution.

Another rather different approach to tackle graph partitioning and other graph optimization problems makes use of eigenvalues. A survey on the use of eigenvalues in combi-

natorial optimization can be found in Mohar and Poljak (1992). A specific application of that technique to the max cut problem with computational results and comparisons with other methods is reported in Poljak and Rendl (1991). Rendl and Wolkowicz (1991) apply eigenvalue techniques to graph partitioning problems and a computational study for the cases of bipartition and tripartition of graphs is carried out in Falkner et al. (1992).

Cutwidth Problem

Yannakakis (1985) studies the cutwidth problem on tree graphs. He gives an $O(n \log n)$ algorithm that solves the problem, when n is the number of nodes in the tree. An $O(n)$ algorithm for a planar version of the problem on trees (improving a previous algorithm of Dovlev and Trickey, 1982) and applications of this special version of the to VLSI layout design are given.

Another interesting application of the cutwidth problem is given in Andreatta et al. (1989) for a problem arising in Flexible Manufacturing System design. The problem can be briefly described as follows. Large rectangular panels are to be cut into smaller (rectangular) boards of standard sizes. There is a fixed number of cutting schemes (for the panels) and of board standard sizes. The production of one sort of board is activated (deactivated) when the first (last) cutting scheme producing it is executed. The problem is to minimize the number of different productions that are active simultaneously. Clearly, this depends on the execution ordering of the cutting schemes. The authors give a formulation that generalizes the cutwidth problem to hypergraphs and propose some heuristics to the problem using Simulated Annealing.

1.6 Thesis Outline

The remaining chapters of the thesis are organized as follows.

In Chapter 2, we investigate the facial structure of the convex hull of equicuts of a graph which is called the equipartition or equicut polytope. The goal is to find class of strong valid inequalities to be used in a branch-and-cut algorithm for solving the equipartition problem.

In Section 2.2 we discuss the different models that can be used for the equipartition and the clustering problems. Section 2.3 contains some preliminary results about the equipartition and cut polytopes.

In many practical applications, the graph for which the problem is defined is sparse. It is reasonable then to look for inequalities having sparse support graphs. Therefore, we are initially interested in finding inequalities with planar support graphs. The departure point of our research is the work of Conforti et al. (1990a,b). Very few of the facet defining inequalities introduced by these authors have planar support graphs. However, one of them has a cycle as support graph and we generalize this inequality in different ways.

The first generalization is given in Section 2.4 by the *path-block cycle* inequalities. Basically, the support graph of this inequality can be viewed as a sum of cycles. We give necessary and sufficient conditions for a *path-block cycle* inequality to be valid for the equipartition polytope and we show some special cases where it can be facet defining. Moreover, we show how these inequalities can be transformed in order to obtain also a new class of facet defining inequalities for the cut polytope.

A second generalization is given by the *suspended tree* inequalities of Section 2.5 which we prove to be facet defining for the equipartition polytope. Here the support graph is given by a tree and an additional node joined to some appropriate nodes in the tree.

The two generalizations of the cycle inequality mentioned above have connected support graphs. In Section 2.5, another class of facet defining inequalities with support composed by one *suspended tree* component and by an odd complete subgraph component. Again, we are able to prove that these inequalities are facet defining for the equipartition polytope and that they can be transformed so as to obtain new classes of facets for the cut polytope.

Section 2.6 contains a series of remarks concerning the path-block cycle and suspended tree inequalities suggesting that all the new facet defining inequalities introduced here can perhaps be put together into a unique large class of inequalities. The results of Chapter 2 are given for complete graphs. In Section 2.7 we discuss the extension of these results to incomplete graphs.

Part of the results in Chapter 2 have been introduced in de Souza and Laurent (1991).

In Chapter 3, we use the inequalities of the equipartition polytope given in Chapter 2 to find new inequalities for some models of graph partitioning problems that appear in the literature. In Section 3.2 we show that a class of facet defining inequalities introduced by Johnson et al. (1991) which have trees as support graphs can be generalized to obtain a larger class of facet defining inequalities. The support of an inequality in that class can be viewed as a sum of trees which is similar to the view, taken in Chapter 2, of the support of a path-block cycle inequality as a sum of cycles.

In Section 3.3, we study the clustering (graph partitioning or multicut) polytope when the cluster capacity constraints reduce to cardinality constraints that limit the number of nodes in a cluster. The dimension of the polytope is given and also conditions under which some facet defining inequalities of the equipartition polytope can be directly adapted to produce facet defining inequalities for this polytope.

Section 3.4 closes this chapter. There we consider the clustering polytope when weights are assigned to the nodes of the graph and the cluster capacities are all equal to a fixed constant. In this case, it is usually hard to determine the dimension of the polytope.

However, we derive some valid inequalities whose supports are of the same type as those for inequalities in Section 3.3. Another valid inequality is given which has been obtained in a different way. The support of this inequality is a tree and, for the edges in the tree, the coefficients of the associated variables are computed using information from the (knapsack) cluster capacity constraints. We call such an inequality a *knapsack tree* inequality.

The results in Sections 3.3 and 3.4 have been obtained in a joint research project with R. Weismantel, A. Martin, C. Ferreira and L. Wolsey.

In Chapter 4, we test the use of the inequalities introduced in Chapters 2 and 3 when generated within a branch-and-cut framework. Section 4.2 is devoted to the presentation of heuristics for solving the separation problem for cycle, path-block cycle, tree and knapsack tree inequalities. These routines have been added to a branch-and-cut code developed by Ferreira, Martin and Weismantel (1992) for the graph partitioning problem. Their code offers an easy way to insert separation routines for new classes of valid inequalities. A small number of implementations details are discussed in Section 4.3. In Section 4.4, we describe the problem instances we have used in the numerical experiments. Computational results and the conclusions drawn from these runs are presented in Section 4.5.

In Chapter 5 we tackle a combinatorial optimization problem that comes from the frontal approach to finite element computations. The Finite Element concepts and the Frontal Method for solving linear systems in finite element computations are presented in Section 5.2. For the Frontal Method to be efficient, a preprocessing phase is necessary in which we look for a good solution to the *frontwidth reduction problem*. The description of this problem and a review of the existing heuristics to solve it are presented at the end of Section 5.2. An alternative formulation of the frontwidth reduction problem in terms of a cutwidth problem on a graph is suggested in Section 5.3.

To solve the cutwidth problem, in Section 5.4, we propose a heuristic approach based on a divide and conquer strategy. This algorithm recursively solves equipartition subproblems

and different versions of it arise from the fact that the equipartition problems are solved by different local search heuristics.

The local search heuristics we choose are the Kernighan and Lin (1970), the Simulated Annealing and the Stochastic Evolution algorithms. These heuristics are discussed in detail in Section 5.5.

Since the quality of the solutions produced by local search algorithms heavily depends on the initial solutions, in Section 5.6, we propose alternative methods to find one such solution. Section 5.7 closes the chapter with the presentation and the analysis of the computational results we obtained from a small sample of two and three dimensional finite element meshes.

The results in Chapter 5 are taken from de Souza, Keunings, Wolsey and Zone (1992).

Finally, Chapter 6 contains our conclusions and suggestions for future research directions.

2. Valid Inequalities for the Equipartition Problem

2.1 Introduction

This chapter is devoted to the study of the facial structure of the Equipartition Polytope. As mentioned before, the effectiveness of a branch-and-cut scheme depends on the strength of the inequalities that are added during the algorithm. One measure of the strength of an inequality is the dimension of the face it defines in the polyhedron corresponding to the convex hull of the (integer) feasible points.

The higher the dimension of the face, the stronger is the inequality defining it. In this sense, the strongest inequalities are those that define facets of the polyhedron, i.e., faces whose dimension is equal to the dimension of the polyhedron minus one.

In this chapter we introduce classes of valid inequalities for the equipartition polytope. We prove that the inequalities in these classes, or in certain subclasses, define facets of the polytope. Therefore, in the forthcoming sections, we deal with the theoretical question of finding strong valid inequalities. The practical question of how to use these inequalities in a branch-and-cut algorithm depends on our ability to solve the corresponding separation problem. This question is addressed in Chapter 4.

The material of this chapter is divided as follows. In Section 2.2, different models for the equipartition and clustering problems are discussed. The discussion of these models will appear in a forthcoming paper by Wolsey, Weismantel, Martin, Ferreira and de Souza.

Section 2.3 contains some fundamental results about the cut polytope and the equipartition polytope. The dimension of these polytopes and some facet defining inequalities are presented. For the equipartition polytope, special attention is given to an inequality that has a cycle as its support graph.

In Section 2.4, we introduce a new class of valid inequalities for the equipartition polytope called *path-block cycle* inequalities and prove them to be facet defining in some special cases. The support graph of these inequalities can be interpreted as a combination of cycles. This allows us to view them as generalizations of the cycle inequality of Section 2.3. We also show that these inequalities can be transformed so as to give valid and facet defining inequalities for the cut polytope.

The class of *suspended tree inequalities* is introduced in Section 2.5. The cycle inequality of Section 2.3 is again a member of this more general class of inequalities which we show to be facet defining for the equipartition polytope. Further classes of facet defining inequalities are derived where the support graph typically contains two connected components: one which is a suspended tree and the other which is an odd complete subgraph. For each of these classes of inequalities, we show the transformations that produce facet defining inequalities for the cut polytope.

Sections 2.3, 2.4 and 2.5 include the results that appear in de Souza and Laurent (1991).

In Section 2.6, we show some examples that suggest that the path-block cycle inequalities of Section 2.4 and the suspended tree inequalities of Section 2.5 can be combined into a unique larger class of valid inequalities for the equipartition polytope.

The results in this chapter are given for complete graphs. Since in many applications the graphs considered are not complete, in Section 2.7 we discuss the extension of these results to incomplete graphs.

2.2 Models for Equipartition and Clustering Problems

In Section 1.3 two different IP (Integer Programming) formulations that can be used to tackle graph partitioning problems are presented. One of these formulations contains variables corresponding to nodes and edges of the graph, the node-edge model, and the other only contains variables corresponding to edges, the edge model. Before applying a polyhedral approach to the graph partitioning (or equipartition) problem, we have to choose a model to work with. In this section, we discuss these models, the links between them and the choices we have made in this thesis.

Let us start by recalling the node-edge model given in Section 1.3 for the graph partitioning problem (as defined in Section 1.2). This model is as follows:

Variables:

$$x_{uv} = \begin{cases} 1 & \text{if edge } (u, v) \text{ is in the multicut defined by the partition} \\ & \text{(that is, nodes } u \text{ and } v \text{ are in different clusters)} \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_u^k = \begin{cases} 1 & \text{if node } u \text{ belongs to the } k\text{-th cluster of the partition} \\ 0 & \text{otherwise} \end{cases}$$

IP formulation:

$$\min \quad \sum_{e \in E} c_e x_e$$

$$\text{Subject to} \quad \sum_{u \in V} w_u y_u^k \leq F_k \quad \forall k \in \{1, \dots, K\} \quad (\text{I})$$

$$\sum_{k=1}^K y_u^k = 1 \quad \forall u \in V \quad (\text{II})$$

$$y_u^k + y_v^\ell - x_{uv} \leq 1 \quad \forall k \neq \ell \in \{1, \dots, K\} \\ \forall (u, v) \in E \quad (\text{III})$$

$$y_u^k + y_v^k + x_{uv} \leq 2 \quad \forall k \in \{1, \dots, K\} \\ \forall (u, v) \in E \quad (\text{IV})$$

$$y_u^k \in \{0, 1\} \quad \forall u \in V, \forall k \in \{1, \dots, K\} \quad (\text{V})$$

$$x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E \quad (\text{VI})$$

For the special case of the equipartition problem (as defined in Section 1.2) this model is given below:

Variables:

$$x_{uv} = \begin{cases} 1 & \text{if edge } (u, v) \text{ is in the equicut defined by the partition} \\ & \text{(that is, node } u \in U \text{ and node } v \in V \setminus U) \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_u^k = \begin{cases} 1 & \text{if node } u \text{ belongs to the } k\text{-th cluster of the partition} \\ 0 & \text{otherwise} \end{cases}$$

IP formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{Subject to} \quad & \sum_{u \in V} y_u^k \leq \lceil \frac{|V|}{2} \rceil \quad k = 1, 2 \\ & y_u^1 + y_u^2 = 1 \quad \forall u \in V \\ & y_u^1 + y_v^2 - x_{uv} \leq 1 \\ & y_u^2 + y_v^1 - x_{uv} \leq 1 \quad \forall (u, v) \in E \\ & y_u^1 + y_v^2 + x_{uv} \leq 2 \\ & y_u^2 + y_v^1 + x_{uv} \leq 2 \quad \forall (u, v) \in E \\ & y_u^k \in \{0, 1\} \quad \forall u \in V, k = 1, 2 \\ & x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E \end{aligned}$$

We define the following sets:

$$S = \{(x, y) \in \mathbb{R}^{m+n \times K} : (x, y) \text{ satisfies (I)-(VI)}\}$$

and

$$X = \{x \in \mathbb{R}^m : \exists y \text{ with } (x, y) \in S\}$$

Thus, X represents the set of incidence vectors of feasible multicuts (or equicuts) of the graph G . In fact, X is the projection of the set S into the space of x variables. Since there are no costs associated to the node variables in the node-edge model, it is reasonable to

look for an alternative model that only contains edge variables. An edge model of that type is at hand if we can find a $q \times m$ matrix A and a vector $b \in \mathbb{R}^q$ such that:

$$X = \{x \in \mathbb{B}^m : Ax \leq b\}$$

An obvious advantage of the edge model compared to the node-edge model is that it has less variables.

However, the edge model presents some shortcomings. The first is that in most cases it is not known how to find a matrix A and a vector b (as defined above) which give a compact IP formulation for the problem of minimizing cx over X . This is the case, for instance, when X represents the set of incidence vectors of equicuts in an arbitrary incomplete graph (a compact formulation of the problem when the graph is complete is given at the end of this section).

A second shortcoming presented by the edge model is the following. Consider the graph partitioning problem defined on a graph G as in Section 1.2. Assume that we have an algorithm that can find an optimal solution x^* for the minimization problem defined over the set X .

Now, suppose that we want to find a partition that corresponds to this optimal multicut. If G is a complete graph, this is an easy task. It suffices to remove from G all the edges e with $x_e^* = 1$, and the resulting graph will have at most K connected components, each of them corresponding to the nodes forming one cluster of the partition. The problem arises when G is not complete and, after removing the edges satisfying $x_e^* = 1$, the resulting graph has more than K connected components. Again, we know that the nodes within a component belong to a same cluster. The problem, which may be hard to solve, is to decide which components come together in each cluster.

The question raised in the previous paragraph is still more delicate when a cutting plane algorithm is used to solve the graph partitioning problem on the edge model. Recall that a cutting plane algorithm solves Linear Programming problems on approximations of the

set X . Thus, it may be the case that the algorithm stops with an integer solution x^* . Since an approximation of X is used, we have to decide whether or not x^* is a feasible multicut. If the answer to this question cannot be given, the algorithm fails to solve the problem.

Nevertheless, the edge model presents interesting features. The value of the optimal solutions in X and S are the same. Because X is a projection of S , any valid inequality for X is also valid for S . Moreover, it is technically easier to work with X than with S , since X lies in a lower dimensional space (less variables).

Due to the points raised above, we decide to use the edge model to apply the polyhedral approach, that is, we seek valid inequalities for the convex hull of X . In this chapter, X is the set of incidence vectors of equicuts of graph G , while in Chapter 3, X represents the set of incidence vectors of multicuts of more general partitions.

In Chapter 4, the node-edge model is used for the computational experiments. The advantage of the node-edge model for the computation is that the feasibility of an integer feasible solution can be checked simply by substituting the value of the variables in the system (I)-(VI).

To end this section, we give one case for the equipartition problem where the edge model is convenient both for polyhedral study and for computation. Consider the equipartition problem defined on $\mathcal{K}_n = (V, E)$ (the complete graph on n nodes). Then, X is the set of incidence vectors of equicuts in \mathcal{K}_n and it has a compact formulation given by:

$$\begin{aligned} X = \{x \in \mathbb{R}^{|E|} : & \quad x_{ij} + x_{jk} + x_{ki} \leq 2, \quad \text{for all } i \neq j \neq k, i, j, k \in V, \\ & \quad x_{ij} - x_{jk} - x_{ki} \leq 0, \quad \text{for all } i \neq j \neq k, i, j, k \in V, \\ & \quad x(E) = \lceil \frac{n}{2} \rceil \lfloor \frac{n}{2} \rfloor, \\ & \quad x_e \in \{0, 1\}, \forall e \in E\} \end{aligned}$$

The first two set of inequalities and the integrality constraints describe the set of incidence vectors of cuts in \mathcal{K}_n (see, for instance, Barahona, Grötschel and Mahjoub, 1985), while

the equation restricts the solution set to the cuts containing $\lceil \frac{n}{2} \rceil \lfloor \frac{n}{2} \rfloor$ edges (the equicuts). It can be noticed that this edge model for the equipartition problem on \mathcal{K}_n has less variables than the corresponding node-edge model. Therefore, the edge model is suitable for computation in the case of equipartition problem on a complete graph. But, this is one of the few cases we have found where such a situation occurs.

2.3 Preliminary Results for the Equipartition and Cut Polytopes

We first recall the definition of the equipartition problem. Consider a graph $G = (V, E)$ with costs c_e associated to the edges $e \in E$. For $U \subseteq V$, we denote by $\delta(U)$ the set of edges with one endnode in U and the other endnode in $V \setminus U$. The set $\delta(U)$ is called a *cut* of G . If $|U| = \lceil \frac{|V|}{2} \rceil$, then $\delta(U)$ is said to be an *equicut* and $(U, V \setminus U)$ is an *equipartition* of G . The *minimum (maximum) equipartition problem* is the problem of finding an equicut $\delta(U)$ minimizing (maximizing) $c(\delta(U)) = \sum_{e \in \delta(U)} c_e$ over all equicuts of G .

The incidence vector of a subset $E' \subseteq E$ is the vector $x^{E'} \in \mathbb{R}^{|E|}$ defined by:

$$x_e^{E'} = \begin{cases} 1 & \text{if } e \in E' \\ 0 & \text{otherwise} \end{cases}$$

The convex hull of all equicuts in G is the equipartition polytope of G denoted by $P_{EC}(G)$; similarly the cut polytope of G is the convex hull of all cuts of G and is denoted by $P_C(G)$.

The facial structure of the equipartition polytope was first investigated in Conforti et al. (1990a, 1990b). Most of the results in this section were obtained by these authors. Further polyhedral investigations of the equipartition problem are due to Deza et al. (1989).

The cut polytope has been extensively studied (see Section 1.5). This study was initiated by a paper of Barahona and Mahjoub (1986) from which we extract the following two results.

Theorem 2.1 *The cut polytope is full dimensional, i.e., $\dim(P_C(G)) = |E|$.*

Theorem 2.2 *Let $G = (V, E)$ be a graph and $P_C(G)$ the corresponding cut polytope. If $G' = (V', E')$ is an induced complete subgraph of G with $|V'|$ odd, then*

$$x(E') \leq \lceil \frac{|V'|}{2} \rceil \lfloor \frac{|V'|}{2} \rfloor$$

is facet defining for $P_C(G)$.

The importance of this result lies in the following fact. Assume that G is the complete graph on $2p + 1$ nodes for some positive integer p . Now, in Theorem 2.2, take $V' = V$ (and consequently $E' = E$). We have that $x(E) \leq \lceil \frac{|V|}{2} \rceil \lfloor \frac{|V|}{2} \rfloor = p(p + 1)$ is facet defining for $P_C(G) = P_C(\mathcal{K}_{2p+1})$. This facet contains all points that are incidence vectors of cuts of size $p(p + 1)$ which are nothing but the equicuts of G . Thus, the equipartition polytope of an odd complete graph is a facet of the cut polytope of the same graph. From Theorem 2.2, this implies that the dimension of $P_{EC}(\mathcal{K}_{2p+1})$ is $|E| - 1$. This remark will be used below to obtain new classes of facets for $P_C(G)$ from those we introduce for $P_{EC}(G)$.

Another interesting result shown by Deza et al. (1989) is that, if the inequality $ax \leq 0$ defines a facet of $P_C(\mathcal{K}_n)$, then it also defines a facet of $P_{EC}(\mathcal{K}_m)$ for any odd m , $m \geq 2n + 1$; therefore, in some sense, the equicut polytope contains all facets of the cut polytope and much more.

The dimension of the equipartition polytope is given in the theorem below.

Theorem 2.3 (Conforti et al., 1990a) *Consider the graph $G = (V, E)$ and let p be some positive integer. Then,*

- (i) *If G is of odd size and $|V| = 2p + 1$, then $P_{EC}(G)$ is full dimensional if and only if G is not a complete graph. If $G = \mathcal{K}_{2p+1}$, then $\dim(P_{EC}(G)) = |E| - 1$ and $\text{aff}(P_{EC}(G)) = \{x \in \mathbb{R}^{|E|} : x(E) = p(p + 1)\}$.*
- (ii) *If G is of even size and $|V| = 2p$, define \overline{G} to be the partial graph of \mathcal{K}_{2p} induced by $\overline{E} = E(\mathcal{K}_{2p}) - E$. Let q be the number of bipartite connected components of \overline{G} . Then, the dimension of $P_{EC}(G)$ is $|E| - q$. If $G = \mathcal{K}_{2p}$, then $\dim(P_{EC}(G)) = |E| - 2p$ and $\text{aff}(P_{EC}(G)) = \{x \in \mathbb{R}^{|E|} : x(\delta(v)) = p, \forall v \in V\}$.*

Conforti et al. pointed out that from a complete linear description of $P_{EC}(\mathcal{K}_{2p-1})$ one can always deduce a complete linear description of $P_{EC}(\mathcal{K}_{2p})$. The reason for that is the following. Suppose that \mathcal{K}_{2p-1} is the complete subgraph of \mathcal{K}_{2p} obtained by removing a node u and its incidence edges from \mathcal{K}_{2p} . Note that the dimension of $P_{EC}(\mathcal{K}_{2p-1})$ is equal

to that of $P_{EC}(\mathcal{K}_{2p})$ and that, for every equipartition of \mathcal{K}_{2p-1} , if node u is added to the smallest subset of the partition, then an equipartition of \mathcal{K}_{2p} is created. So, take any facet defining inequality of $P_{EC}(\mathcal{K}_{2p-1})$, say $\pi x \leq \pi_0$, and let S be a maximal set of affinely independent points in that facet. From every equipartition corresponding to a point in S , we can obtain the incidence vector of an equicut in \mathcal{K}_{2p} in the way mentioned above. All these incidence vectors clearly satisfy $\pi x = \pi_0$ and are affinely independent. Since $\dim(P_{EC}(\mathcal{K}_{2p-1})) = \dim(P_{EC}(\mathcal{K}_{2p}))$, the set of these incidence vectors is maximal and, therefore, $\pi x \leq \pi_0$ is also facet defining for $P_{EC}(\mathcal{K}_{2p})$.

Therefore, we can restrict ourselves to the study of the odd case or the even case. From the above theorem, the difference between the dimension of the space and that of the polytope is smaller for the odd case. Thus, it is technically easier to work with $P_{EC}(\mathcal{K}_{2p+1})$ than with $P_{EC}(\mathcal{K}_{2p})$.

To simplify the proofs, we only deal with complete graphs. However, the next lemma shows how the facet defining property is preserved when G is a subgraph of the complete graph \mathcal{K}_{2p+1} .

Lemma 2.4 (*Conforti et al., 1990a*) *Let $ax \leq a_0$ be a facet inducing inequality for $P_{EC}(\mathcal{K}_{2p+1})$. Then, for any graph G' obtained from \mathcal{K}_{2p+1} by removing a subset of its edges not belonging to the support of the vector a , the inequality $ax \leq a_0$ is a facet inducing inequality for $P_{EC}(G')$.*

In many practical applications the graph G to be equipartitioned is sparse ($|E| \ll \frac{n(n-1)}{2}$) and, as a consequence, the support graphs of strong valid inequalities of $P_{EC}(G)$ are also sparse. Thus, if G is a subgraph of \mathcal{K}_{2p+1} , by Lemma 2.4, we have that the face defining inequalities of $P_{EC}(\mathcal{K}_{2p+1})$ that are inherited by $P_{EC}(G)$ must have sparse support graphs. This was a motivation for us to look for facets of $P_{EC}(\mathcal{K}_{2p+1})$ with planar supports. As we will see later on, the path-block cycle inequalities of Section 2.3 and the suspended tree inequalities of section 2.4 satisfy this property. In some other classes of inequalities introduced in Section 2.4 planarity is lost and the support graphs

are disconnected. This is in contrast to the cut polytope where the support graphs of all facets are 2-connected (De Simone, 1990).

Many of the facet defining inequalities for the equipartition polytope introduced by Conforti et al. have dense support graphs. Among those having sparse supports, special attention is paid here to the one given in Theorem 2.5 below. This inequality has a cycle as its support graph and is a particular case of the more general classes of inequalities given in Sections 2.3 and 2.4.

Theorem 2.5 *Let C be a cycle of \mathcal{K}_{2p+1} with $|V(C)| = p + 2$ and $p \geq 3$. Then the inequality*

$$x(E(C)) \geq 2 \tag{1}$$

is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

To close this section we present a lemma that is used repeatedly in our proofs that valid inequalities are facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

Lemma 2.6 *Let $G = (V, E)$ be a graph with $|V| = 2p + 1$ nodes and let $ax \leq a_0$ be a valid inequality for $\mathcal{P}_{EC}(G)$. Let S_1, S_2, T_1 and T_2 be four mutually disjoint subsets of V such that $|S_1| = |S_2|$, $|T_1| = |T_2|$ and $|S_1 \cup S_2 \cup T_1 \cup T_2| = 2p$. Finally, let $\{w\} = V \setminus (S_1 \cup S_2 \cup T_1 \cup T_2)$ and suppose that the incidence vectors of the following equicuts of G :*

$$\Gamma_1 = \delta(S_1 \cup T_1)$$

$$\Gamma_2 = \delta(S_2 \cup T_2)$$

$$\Gamma_3 = \delta(S_1 \cup T_2)$$

$$\Gamma_4 = \delta(S_2 \cup T_1)$$

satisfy $ax = a_0$. Then,

$$\sum_{z \in S_1} a_{wz} = \sum_{z \in S_2} a_{wz}$$

2.4 The Path-Block Cycle Inequalities

We start this section by introducing a new graph structure that we call a path-block cycle. Next, we define the path-block cycle inequalities which have that structure as support graph. Necessary and sufficient conditions are given for a path-block cycle (PBC) inequality to be valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. We then prove that a subclass of the PBC inequalities define facets of the equipartition polytope. Finally, we show how to extend this result to obtain new classes of valid and facet defining inequalities for the cut polytope $P_C(\mathcal{K}_{2p+1})$.

Let $\mathcal{C} = ((V(C_1), C_1), (V(C_2), C_2), \dots, (V(C_r), C_r))$ be a collection of r cycles in a graph $G = (V, E)$. Assume that the cycles in \mathcal{C} are such that if $u \in V(C_i) \cap V(C_j)$ for some $1 \leq i \neq j \leq r$, then $u \in V(C_\ell)$ for all $1 \leq \ell \leq r$. In other words, if a node that is common to a pair of cycles in \mathcal{C} , it must be common to all cycles. Moreover, assume that the common nodes are visited in the same sequence whatever the cycle we traverse in \mathcal{C} . The subgraph of G with node set given by $\bigcup_{i=1}^r V(C_i)$ and the edge set given by $\bigcup_{i=1}^r C_i$ defines a *path-block cycle* on G . Figure 2.1 shows an example of a PBC graph composed of three cycles. Note that, in this example, nodes 1, 2, 3, 4 and 5 are common to all cycles and the edge (3,4) is common to cycles C_1 and C_2 .

In order to describe the PBC inequality associated to a given PBC support graph, we have to introduce more definitions. Let N be the set of nodes that are common to all cycles in \mathcal{C} and $t = |N|$. Let (s_1, s_2, \dots, s_r) be the sequence in which the nodes in N are visited if we traverse any of the cycles in \mathcal{C} in a fixed direction starting at node $s_1 \in N$. We define $((V(P_{ij}), P_{ij})$ to be the path in cycle C_j joining the nodes s_i and s_{i+1} (indices are taken modulo t) and $q_{ij} = |V(P_{ij})| - 2$. In fact, q_{ij} is the number of nodes in $V(P_{ij}) \setminus \{s_i, s_{i+1}\}$.

A *path-block* is the set of paths P_{ij} , $1 \leq j \leq r$, for any fixed i in $\{1, \dots, t\}$. A path-block, or simply block, is said to be degenerate if the number of distinct paths in it is less than r . The nodes s_i and s_{i+1} are said to be respectively the source and the destination nodes of the i -th block.

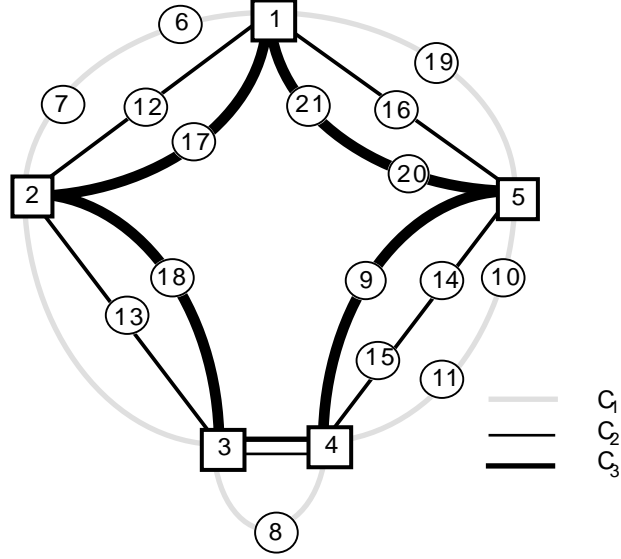


Figure 2.1:

For every edge $e \in C_1 \cup \dots \cup C_r$, let a_e be the number of cycles that contain edge e ($a_e = |\{j : e \in C_j \in \mathcal{C}\}|$). The path-block cycle inequality associated to the collection \mathcal{C} of cycles is given by:

$$\sum_{e \in \bigcup_{i=1}^r C_i} a_e x_e \geq 2r \quad (2)$$

We investigate the validity of a PBC inequality with respect to the equipartition polytope. For this, assume that $n_o = |V \setminus V(\bigcup_{i=1}^r C_i)|$ and that \bar{q}_1 is the largest of the values q_{ij} for $1 \leq j \leq r$, \bar{q}_2 is the second largest among those values and so on. Let $\bar{Q} = \sum_{i=1}^{r-1} \bar{q}_i$. The following theorem gives the necessary and sufficient conditions for the PBC inequality to be valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

Theorem 2.7 *The PBC inequality (2) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ if and only if $\bar{Q} + n_o < p$.*

Proof:

Necessity: We prove that, if $\bar{Q} + n_o \geq p$, then (2) is not valid. Let \bar{P}_i be the path corresponding to the value \bar{q}_i and $\bar{V}_i = V(\bar{P}_i) \setminus N$, where $i \in \{1, \dots, r-1\}$. By definition, we have that $|\bar{V}_i| = \bar{q}_i$. Now, we construct a set $U \subseteq V$ by successively adding nodes

to it until we reach $|U| = p$. This is done in the following way. If $n_o \geq p$, then take U as any subset of $V \setminus V(\bigcup_{i=1}^r C_i)$ with $|U| = p$ and the proof is complete because $\sum_{e \in E} a_e x_e^{\delta(U)} = 0$ and (2) is not valid since $\delta(U)$ is an equicut. If $n_o < p$, we initialize U by assigning all nodes in $V \setminus V(\bigcup_{i=1}^r C_i)$ to it and then $|U| = n_o$. Starting with i set to one, if $|U| - p \geq \bar{q}_i$ we add the nodes of \bar{V}_i to U , we increment i and we repeat the process. Because $\sum_{i=1}^{r-1} \bar{q}_i + n_o = \bar{Q} + n_o \geq p$, we will stop either with $|U| = p$, or with $i = k$ such that $|U| - p < \bar{q}_k$. In the last case, we assign exactly $|U| - p$ nodes of $V(\bar{P}_k)$ that form a subpath of \bar{P}_k to the set U . Constructed in this way, $\delta(U)$ is an equicut. Moreover, $|\delta(U) \cap (\bigcup_{i=1}^r C_i)| \leq 2r - 2$ and $a_e = 1$ for all edges in $\delta(U) \cap (\bigcup_{i=1}^r C_i)$. Thus, $\sum_{e \in E} a_e x_e < 2r$ and (2) is not valid. Necessity is proved.

Sufficiency: The proof is by contradiction. We assume that $\bar{Q} + n_o < p$ and that there exists an equicut $\Gamma = \delta(U)$ for which $\sum_{e \in E} a_e x_e < 2r$. Our goal is to find such an equicut Γ .

Clearly, if $x^\Gamma(C_i) > 0$ for all $C_i \in \mathcal{C}$, inequality (2) is satisfied ($x^\Gamma(C_i) > 0 \Rightarrow x^\Gamma(C_i) > 2$). Therefore, Γ must be a cut in which at least one of the cycles in \mathcal{C} is not cut. In this case, all nodes in N belong to the same shore of the equipartition (i.e., $N \subseteq U$ or $N \subseteq V \setminus U$). This implies that the paths P_{ij} , for all $1 \leq i \leq r$, $1 \leq j \leq t$ are cut in an even number of edges and therefore $\sum_{e \in E} a_e x_e = 2k$ for some integer $k \leq r - 1$ (otherwise we have no violation). Suppose that $N \subseteq V \setminus U$. If path P_{ij} contains a node in U , then $x^\Gamma(P_{ij}) > 2$. Thus, for a violation of (2), we may have at most $r - 1$ paths with nodes in U .

Let $\tilde{P}_1, \tilde{P}_2, \dots, \tilde{P}_s$ be the set of paths having a nonempty intersection with U and define $\tilde{q}_i = |\tilde{P}_i \cap U|$ for $1 \leq i \leq s \leq r - 1$. We have that: $\tilde{Q} = \sum_{i=1}^s \tilde{q}_i = |U \cap V(\bigcup_{j=1}^r C_j)|$.

Let q_o be the size of the set of nodes given by $(V \setminus V(\bigcup_{j=1}^r C_j)) \cap U$. This yields:

$$|U| = \tilde{Q} + q_o \leq \bar{Q} + q_o \leq \bar{Q} + n_o < p$$

where the first inequality comes from the definition of \bar{Q} (and the fact that $s \leq r - 1$), the second comes from the definition of n_o and the third by hypothesis. Therefore, $(U, V \setminus U)$ does not form an equipartition and we conclude that there is no equicut Γ violating (2). The proof is complete. \square

The next lemma characterizes the equicuts whose incidence vectors satisfy the PBC inequality at equality.

Lemma 2.8 *Let $x^{\delta(U)}$ be the incidence vector of an equicut satisfying (2) at equality. Then $\delta(U)$ is such that:*

- (I) *either there are two blocks of the PBC which have all paths cut only in one edge and the remaining blocks do not have edges in $\delta(U)$.*
- (II) *or there are r paths P_{ij} in the PBC that are cut exactly in two edges and the remaining paths are not cut.*

Proof: Suppose first that the path P_{ij} is cut in an odd number of edges of $\delta(U)$, i.e., $x^{\delta(U)}(P_{ij}) = 2k_{ij} + 1$ for some integer $k_{ij} \geq 0$. This implies that the endnodes s_i and s_{i+1} of the i -th path-block are in different shores of the partition. Thus, all the r cycles of \mathcal{C} are cut in an odd number of edges both inside and outside the i -th block. This yields:

$$\begin{aligned}
\sum_{\ell=1}^r \sum_{e \in C_\ell} a_e x_e &= \sum_{\ell=1}^r \sum_{e \in P_{i\ell}} a_e x_e^{\delta(U)} + \\
&\quad \sum_{\ell=1}^r \sum_{h=1, h \neq i}^t \sum_{e \in P_{h\ell}} a_e x_e^{\delta(U)} \\
&\geq \sum_{\ell=1}^r (2k_{i\ell} + 1) + \sum_{\ell=1}^r (2k_\ell + 1) \\
&\geq 2r + 2 \sum_{i=1}^r (k_{i\ell} + k_\ell)
\end{aligned}$$

where $k_{i\ell}$ and k_ℓ are nonnegative integers for $1 \leq \ell \leq r$. But, since $x^{\delta(U)}$ lies in the face defined by (2), we have that all $k_{i\ell}$'s and k_ℓ 's are zero. This implies that all paths in block i are cut exactly once. Symmetric arguments show that the r paths P_{ij} that contain one edge in $\delta(U)$ are concentrated on a single block (other than i). We conclude that $x^{\delta(U)}$ must be in the incidence vector of an equicut satisfying I .

A second possibility is that all paths P_{ij} in the PBC are cut in an even number of edges. Using cyclic arguments it is easy to see that in this case all nodes of N must be in the same shore of the partition, say $V \setminus U$. So, only the paths that are cut contain nodes in

U . Let P_1, \dots, P_s be such paths and $q_i = |V(P_i)|$, for $1 \leq i \leq s$. Since $\delta(U)$ is an equicut for which $x^{\delta(U)}$ satisfies (2) at equality, s must be less than or equal to r . If s is strictly less than r , we have that: $|U| \leq \sum_{i=1}^s q_i + n_o < \overline{Q} + n_o \leq p$. Therefore, $(U, V \setminus U)$ cannot form an equipartition if $s < r$. On the other hand, if $s = r$ and $x^{\delta(U)}$ belongs to the face defined by (2), then the r paths that are cut contain precisely two edges in $\delta(U)$ and so we are in case *II*. \square

Lemma 2.8 together with Lemma 2.6 will provide the necessary amount of affinely independent points needed to prove the facet defining property for a subclass of the PBC inequalities. In fact, it does not seem to be an easy task to obtain the necessary and sufficient conditions for PBC inequalities to define facets of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ in general. However, if we restrict ourselves to PBCs that have some symmetry a proof can be given that they define facets. Before presenting one such case, we note that the cycle inequality is a member of the class of PBC inequalities. For this, we restrict \mathcal{C} to a single cycle C_1 which implies that $r = 1$, $N = V(C_1)$, $q_{ij} = 0$ and $a_e = 1$ for all $e \in C_1$.

Consider now the class of PBC inequalities for which the collection of cycles \mathcal{C} satisfies the conditions below.

- (i) All r cycles in \mathcal{C} are of the same size, i.e., $|V(C_i)| = |V(C_j)|$ for $C_i \neq C_j \in \mathcal{C}$ and $r \geq 2$;
- (ii) The number of blocks in the PBC is even, i.e., $|N| = 2t$, $t \in \mathbb{Z}$, $t \geq 2$;
- (iii) All blocks indexed by an odd integer $i \in \{1, \dots, 2t\}$ are such that $P_{ij} = \{s_i, s_{i+1}\}$ for all $j \in \{1, \dots, r\}$ and consequently, $q_{ij} = 0$ and $a_{(s_i, s_{i+1})} = r$;
- (iv) All blocks indexed by an even integer $i \in \{1, \dots, 2t\}$ are such that $P_{i\ell} \cap P_{ik} = \emptyset$ for all $1 \leq \ell \neq k \leq r$ and $q_{ij} = q > 0$. In other words, those blocks are formed by r disjoint paths going from s_i to s_{i+1} and their lengths are all equal to a constant $q + 1$. So, $a_e = 1$ for all edges e in P_{ij} .

An example of such PBC is shown in Figure 2.2. In this example, $|V(C_i)| = 15$, $r = t = q = 3$. The edges represented by thick lines belong simultaneously to the three cycles that form the PBC.

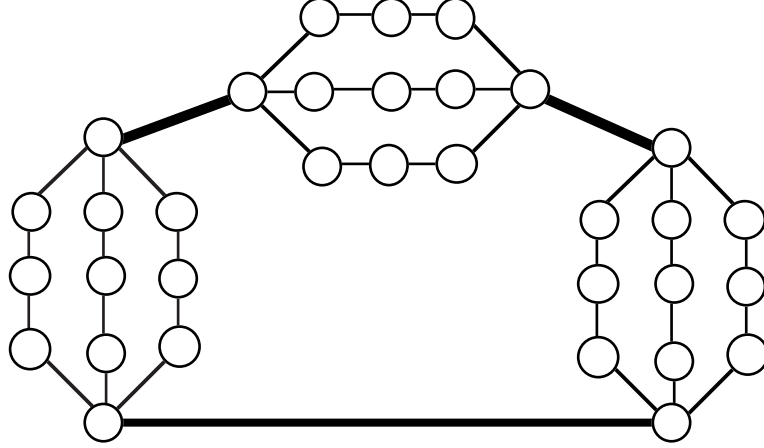


Figure 2.2:

Let V_{in} denote the set of nodes in the PBC and E_{in} the set of its edges. If V_{out} is the set of nodes in $V \setminus V_{in}$, then $n_o = |V_{out}|$. Now, consider a PBC satisfying the conditions (i)-(iv). If n_i is the size of V_{in} , then $n_i = t(rq + 2)$. Moreover, if we partition the edge set E_{in} into two sets E_1 and E_r such that E_1 is the set of edges in nondegenerate blocks (indexed by even numbers) and E_r is the set of edges in degenerate blocks (indexed by odd numbers), inequality (2) can be written as:

$$x(E_1) + rx(E_r) \geq 2r \quad (3)$$

Theorem 2.9 *Suppose that a PBC satisfying conditions (i)-(iv) is a subgraph of \mathcal{K}_{2p+1} and that $n_o = p - (r - 1)q - 1$. Then, the PBC inequality (3) is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.*

Before we prove the theorem, we introduce some notation. The name block is used to refer to the nondegenerate blocks (even ones) and PB_k is in fact the $2k$ -th block of the

PBC and its endnodes are given by s_{2k} and s_{2k+1} . To simplify notation, we define the following node sets:

- $V_{kj} = V(P_{2k,j}) \setminus \{s_{2k}, s_{2k+1}\}$ (nodes of the j -th path of the $2k$ -th path-block PB_k excluding nodes s_{2k} and s_{2k+1});
- $V_k = \bigcup_{j=1}^r V_{kj}$ (nodes of the $2k$ -th path-block PB_k excluding nodes s_{2k} and s_{2k+1});

Proof of Theorem 2.9: Validity comes directly from Theorem 2.7 since $\overline{Q} = (r-1)q$ and, therefore, $n_o + \overline{Q} = p - (r-1)q - 1 + (r-1)q = p - 1 \leq p$.

Let $\pi x \geq \pi_0$ be a facet defining inequality for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. We define:

$$F = \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : x(E_1) + rx(E_r) = 2r\}$$

$$F_\pi = \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \pi x = \pi_0\}$$

Assume that $F \subseteq F_\pi$. We want to prove that $\pi x \geq \pi_0$ can be written as a linear combination of (3) and the equation $x(E) = p(p+1)$, *i.e.*, there exists $\lambda_1 \in \mathbb{R}^+$ and $\lambda_2 \in \mathbb{R}$ such that:

$$(\pi x \geq \pi_0) = \lambda_1(x(E_1) + rx(E_r) \geq 2r) + \lambda_2(x(E) = p(p+1))$$

The proof is divided into two major parts. In the first one, using Lemma 2.6, we show that all components of the vector π associated to edges not in E_{in} take the same value. In the second part, we compute the values of the components of π corresponding to edges in E_{in} .

Usually, when Lemma 2.6 is applied, the choice of the node w and that of the set S_1 and S_2 arises in a natural way from what is being proved. The difficult task is to build the sets T_1 and T_2 . One way to do so makes use of a **balancing criterion** which works as follows. Suppose that the nodes of a given set U are to be assigned to T_1 and T_2 . According to the balancing criterion, we choose the assignment that minimizes $||U \cap T_1| - |U \cap T_2||$. This is necessary to guarantee that the proof is correct for any triple of parameters (r, t, q) , where $r, t, q \geq 2$.

The incidence vectors of the cuts Γ_1 - Γ_4 as in Lemma 2.6 must lie on the face defined by (3) in $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. For the equicuts Γ_1 - Γ_4 that we construct in this proof, this fact can be easily verified and we limit ourselves to indicating the type of root they define in (3), i.e., to which case of Lemma 2.8 correspond the incidence vectors $x^{\Gamma_1}, \dots, x^{\Gamma_4}$.

First we prove that the components of π corresponding to edges incident to any node in V_{out} take all the same constant value.

(a) : Let u be a node in V_{in} and u', v' any two nodes in V_{out} . Let $w = u'$, $S_1 = \{u\}$, $S_2 = \{v'\}$ and C be a cycle in PBC containing u .

Now starting from u we move clockwise around the cycle until we reach the $\lceil \frac{t(q+2)}{2} \rceil$ -th node, say v . We define A_1 as the set of all nodes visited up to v excluding u . The set A_2 is then the set of nodes visited after v . Clearly, $V(C) = A_1 \cup A_2 \cup \{u\}$.

Let PB_k and PB_i be the blocks containing nodes u and v respectively. The procedure **BuildSets** described below can be applied to correctly build the sets T_1 and T_2 :

Step 1: Assign the nodes of A_1 (respectively, A_2) to T_1 (respectively, T_2).

Step 2: For blocks other than PB_k and PB_i , if both endnodes are in T_1 (respectively, T_2), assign its remaining unassigned nodes to T_1 (respectively, T_2).

Step 3: For PB_k (respectively, PB_i), if u (respectively, v) corresponds to one of the endnodes, say the source, the remaining unassigned nodes of it are assigned to the same set as the other endnode, in that case, the destination.

Step 4: If block PB_k (or PB_i) still has unassigned nodes, they are assigned to T_1 and T_2 according to the Balancing Criterion (applied to V_{in}) in such a way that $\delta(T_1, T_2) \cap E_{in}$ has one edge intersecting each of its internal paths.

Step 5: The sets T_1 and T_2 are filled up to size $p - 1$ with the unassigned nodes of V_{out} .

By construction PB_k is different from PB_i and, thus, $\Gamma_1, \dots, \Gamma_4$ are equicuts of type I . We conclude that $\pi_{u'u} = \pi_{u'v'} = \gamma$ for all $u \in V_{in}$, $u', v' \in V_{out}$. Note that, since u' and v' were taken arbitrarily, π_e takes the value γ for any edge e with both endnodes in V_{out} .

Below, we prove that $\pi_e = \gamma$ for all edges e not in E_{in} but joining two nodes in different blocks of PBC .

(b) : Let u, v be two nodes in different blocks of PBC such that $e = (uv) \notin E_{in}$ and u' a node in V_{out} . Let $w = u$, $S_1 = \{v\}$, $S_2 = \{u'\}$ and C be a cycle in PBC with edge set given by:

- The edges of a path arbitrarily chosen for each block except the ones containing u and v , say PB_k and PB_i , respectively;
- The edges of the path in PB_k that contains u ;
- The edges of the path in PB_i that contains v ;
- The edges in E_r .

Defining the sets A_1 and A_2 exactly as we did in (a) and applying the procedure **Build-Sets**, we conclude that $\pi_{uv} = \pi_{uu'} = \gamma$ for all nodes $u, v \in V_{in}$ ($u \neq v$ and $(uv) \notin E_{in}$) and $u' \in V_{out}$. Again we have that $\Gamma_1, \dots, \Gamma_4$ are equicuts of type I .

Next we investigate the case when we have two nodes in a same block of PBC and besides this they belong to a common path of the block.

(c) : Let u and v be two nonadjacent nodes of a block PB_k in PBC and u' a node of V_{out} . Suppose that there are paths (one or r) in PB_k containing both u and v . Let $w = u$, $S_1 = \{v\}$ and $S_2 = \{u'\}$. To build T_1 and T_2 , we apply the following procedure:

Step 1: For all internal paths of PB_k containing both u and v , assign the nodes between u and v to T_1 .

Step 2: For the remaining paths of PB_k (if any), assign its intermediate nodes to T_1 such that; (i) $|\delta(T_1, T_2)|$ (calculated on these paths) is equal to two if both u and v are not endnodes of PB_k or equal to one otherwise and (ii) the current size of T_1 attains $q(r - 1) + 1$.

Step 3: Assign the nodes of V_{out} to T_1 until $|T_1| = p - 1$. The remaining nodes of V_{out} are then assigned to T_2 .

If both u and v are endnodes of PB_k , then $\Gamma_1, \dots, \Gamma_3$ are roots of type *I* and Γ_4 is a root of type *II*. In contrary, if both u and v are not endnodes of PB_k , then all four equicuts give rise to roots of type *II*. If U is an endnode of PB_k but not v , then Γ_1 and Γ_3 are type *I* while Γ_2 and Γ_4 are type *II*. Finally, if v is an endnode of PB_k but not u Γ_1 and Γ_2 are type *I* while Γ_3 and Γ_4 are type *II*.

We conclude that $\pi_{uv} = \pi_{uu'} = \gamma$ for all nodes u, v both belonging to a same block in V_{in} and to at least one internal path of it and all nodes u' in V_{out} .

In (d) we deal with the case not treated in (c) when the nodes belong to a same block but are not in a same path.

(d) : Let u and z be two nodes of a block PB_k in PBC and u' a node of V_{out} . Suppose that there are no paths in PB_k containing both u and z . Let $w = u$, $S_1 = \{z\}$ and $S_2 = \{u'\}$. Let C , v , A_1 and A_2 be defined as in (a).

To correctly construct the sets T_1 and T_2 , we apply the same procedure **BuildSets** from (a) but with a slight change in Step 4. Suppose that z is in the j -th path of PB_k (necessarily different from the one containing u). For path j , in Step 4, the nodes between the source (respectively, destination) and z are forced to be in the same set of the source (respectively, destination). After doing this, the nodes from other paths of PB_k are assigned following the rules expressed in Step 4.

Note that Step 3 is never executed because u and z cannot be endnodes of PB_k . All the four equicuts of Lemma 2.6 are of type I .

This shows that $\pi_{uz} = \pi_{uu'} = \gamma$ for all nodes u and z belonging to a same block in PBC (but not to a same path) and all u' in V_{out} .

We already proved that all edges not belonging to the support graph of (6) are associated to components of π with value γ . From (e) to (g) we show that the components of π associated to edges in E_1 have a constant value α .

(e) : Let u be a node of V_k for some block PB_k of PBC and j be the path containing u in PB_k with v and z its two adjacent nodes in it. We then build the following sets: $w = u$, $S_1 = \{v\}$, $S_2 = \{z\}$, $T_1 = \cup_{i \neq j} (V_{ik}) \cup V_{out}$ and $T_2 = V \setminus (S_1 \cup S_2 \cup T_1 \cup \{w\})$.

In this case, if both v and z are endnodes of PB_k , we have that $\Gamma_1, \dots, \Gamma_4$ are of type I . On the other hand, if both v and z are not endnodes of PB_k , $\Gamma_1, \dots, \Gamma_4$ are of type II . If v (z) is an endnode of PB_k but not z (v), then Γ_1 and Γ_2 (Γ_3 and Γ_4) are of type I and Γ_3 and Γ_4 (Γ_1 and Γ_2) are of type II .

From Lemma 2.6, we deduce that $\pi_{uv} = \pi_{uz} = \alpha_{jk}$, i.e., $\pi_e = \alpha_{jk}$ for all edges e in the j -th path of PB_k .

In the remaining of the proof, we do not use Lemma 2.6. We simply exhibit two equicuts satisfying (3) at equality and, since they must satisfy $\pi x = \pi_0$, we obtain further relations between the components of π .

(f) : Let i and j be two distinct paths of some block PB_k and u a node in V_{k-1} . We define the following subsets of V :

$$\begin{aligned} U_1 &= \bigcup_{\ell=1}^r (V_{\ell k} : \ell \neq i) \cup V_{out} \cup \{u\} \quad (\text{type } II) \\ U_2 &= \bigcup_{\ell=1}^r (V_{\ell k} : \ell \neq j) \cup V_{out} \cup \{u\} \quad (\text{type } II) \end{aligned}$$

The incidence vectors of both $\delta(U_1)$ and $\delta(U_2)$ satisfy (3) at equality. Hence, $\pi x^{\delta(U_1)} = \pi x^{\delta(U_2)}$ which implies that $\alpha_{ik} = \alpha_{jk} := \alpha_k$. Hence, for all $k \in \{1, \dots, t\}$, if e is an edge of PB_k then $\pi_e = \alpha_k$.

Now, we prove that α_k is a constant not depending on the block k , i.e., $\alpha_k := \alpha$.

(g) : Let v be a node in the r -th path of PB_k and $U_3 = \bigcup_{\ell=1}^{r-1} (V_{\ell k}) \cup V_{out} \cup \{v\}$. The incidence vector of $\delta(U_3)$ satisfies (3) at equality and, therefore, $\pi x^{\delta(U_1)} = \pi x^{\delta(U_3)} = \pi_0$ which shows that $\alpha_k = \alpha_{k-1} := \alpha$. By symmetry, we have proved that, for all edges E in E_1 , $\pi_e = \alpha$. Note that U_3 is of type *II*.

All the edges in E_r are associated to components in π that take the same value as we prove below.

(h) : Consider the following subsets of V :

$$\begin{aligned} U_4 &= \bigcup_{\ell=1}^{r-1} (V_{\ell k}) \cup V_{out} \cup \{s_{2k}\} \quad (\text{type } I) \\ U_5 &= \bigcup_{\ell=1}^{r-1} (V_{\ell k}) \cup V_{out} \cup \{s_{2k+1}\} \quad (\text{type } I) \end{aligned}$$

The equicuts $\delta(U_4)$ and $\delta(U_5)$ both satisfy (6) at equality. Comparing the values of $\pi x^{\delta(U_4)}$ and $\pi x^{\delta(U_5)}$ (which are equal to π_0), we get that $\pi_{s_{2k}s_{2k-1}} = \pi_{s_{2k+1}s_{2k+2}} = \beta$. Again, by using symmetric arguments, we deduce that for all edges e in E_r , $\pi_e = \beta$.

It remains to investigate how α , β and γ are related. This is easily done by noting that both $x^{\delta(U_3)}$ and $x^{\delta(U_4)}$ are in F . Hence, $\pi x^{\delta(U_3)} = \pi x^{\delta(U_4)} = \pi_0$ implying that:

$$\beta = r\alpha - (r-1)\gamma.$$

The results we proved so far show that $\pi x \geq \pi_0$ can actually be written as a linear combination of (3) and $x(E) = p(p+1)$ as stated before. In order to do so we take:

- $\lambda_1 = \alpha - \gamma$;
- $\lambda_2 = \gamma$.

The proof is complete. \square

For a PBC satisfying conditions (i)-(iv), we have that $\overline{Q} + n_o = p - 1$. In the next theorem, we see that this is a necessary condition for a PBC inequality to be facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

Theorem 2.10 *Consider a PBC which is a subgraph of \mathcal{K}_{2p+1} . If the corresponding PBC inequality is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, then $\overline{Q} + n_o = p - 1$.*

Proof: By contradiction. Assume that $\overline{Q} + n_o \leq p - 2$. From Theorem 2.7 the PBC inequality is valid. Suppose first that P_{ij} is a path in the PBC such that $q_{ij} = \min\{q_{ab} > 0, \forall a, b \text{ such that } 1 \leq a \leq t \text{ and } 1 \leq b \leq r\}$.

Let u, v, z be three nodes in $V(P_{ij})$ with $(u, v), (u, z) \in P_{ij}$ (u is not an endnode of block i). We claim that there exists no vector in $F = \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : x \text{ satisfies (2) at equality}\}$ such that $x_{uv} = x_{uz} = 1$. In fact, if $\delta(U)$ is an equicut corresponding to a root of (2) and it cuts the edges (u, v) and (u, z) , according to Lemma 2.8, it must cut $(r - 1)$ different paths in two edges. Suppose that all the nodes between the two edges that are cut in these $(r - 1)$ paths and the node u are in set U . All the other nodes in the PBC are in $V \setminus U$. Thus, even if all nodes not in the PBC are in U , we have that $|U| \leq \sum_{i=1}^{r-1} \overline{q}_i + n_o + 1 = \overline{Q} + n_o + 1 = p - 1$ and $(U, V \setminus U)$ cannot be an equipartition. Thus, all vectors in F satisfy $x_{uv} + x_{uz} = x_{vz}$ and (2) can be lifted by reducing the coefficient a_{uv} and a_{uz} from 1 to 0 and increasing a_{vz} from 0 to 1.

Now assume that $q_{ij} = 0$ for all paths in the PBC. This implies that $\overline{Q} = 0$ and the support graph of the PBC inequality reduces to a cycle on more than $p + 2$ nodes (since $n_o \leq p - 2$). If (u, v) and (u, z) are two adjacent edges in this cycle, it is easy to see that $x_{uv} + x_{uz} = x_{vz}$ for all incidence vectors in F (defined as above). By repeating the same operation described in this previous case, we obtain a stronger inequality that proves that the original one does not define a facet of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. This completes the proof. \square

We have seen in Theorem 2.2 that the equipartition polytope of \mathcal{K}_{2p+1} is a facet of the cut polytope of \mathcal{K}_{2p+1} . Therefore, given a facet defining inequality of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, there exists a suitable linear combination of this inequality and the equation $x(E) = p(p+1)$ which produces a facet defining inequality for $P_C(\mathcal{K}_{2p+1})$. In the next theorem we introduce an inequality which arises from such a linear combination and we give the necessary and sufficient conditions for this inequality to be valid with respect to $P_C(\mathcal{K}_{2p+1})$. Moreover, if the PBC inequality is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ and the linear combination results in a valid inequality for $P_C(\mathcal{K}_{2p+1})$, Theorem 2.10 above is used to prove that the former defines a facet for $P_C(\mathcal{K}_{2p+1})$.

Theorem 2.11 *Given a PBC inequality that is a subgraph of \mathcal{K}_{2p+1} , let $ax \geq 2r$ be the corresponding inequality. Consider the following inequality:*

$$\begin{aligned} (\pi x \leq \pi_0) &= (x(E) = p(p+1)) - (ax \geq 2r) \\ &= (x(E) - ax \leq p(p+1) - 2r) \end{aligned} \quad (4)$$

Then,

(i) For all $k \in \mathbb{Z}_+$ with $k(k+1) \leq 2(r-1)$, let \overline{Q}_k be such that

$$\overline{Q}_k = \sum_{i=1}^{r-1-\epsilon(k)} \overline{q}_i$$

where $\epsilon(k) = \frac{k(k+1)}{2}$ (according to this definition, we have that $\overline{Q} = \overline{Q}_0$). Then,

(4) is valid for $P_C(\mathcal{K}_{2p+1})$ if and only if $n_0 + \overline{Q}_k \leq p - 1 - k$ for all $k \in \mathbb{Z}_+$ with $k(k+1) \leq 2(r-1)$.

(ii) If (4) is valid for $P_C(\mathcal{K}_{2p+1})$ and $ax \geq 2r$ is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, then (4) is facet defining for $P_C(\mathcal{K}_{2p+1})$.

Proof:

(i) : As in Theorem 2.7, we first prove necessity. For this, assume that $\exists j \in \mathbb{Z}_+$ such that $j(j+1) \leq 2(r-1)$ and $n_0 + \overline{Q}_j \geq p - j$. This means that there exists a node set $U \subseteq V$ with $|U| = p - j$ such that the nodes in U are all internal nodes of the paths

corresponding to the $r - 1 - \frac{j(j+1)}{2}$ largest \bar{q}_i values and all the remaining nodes are not in the PBC. For such a set U , we have that:

$$\begin{aligned} x^{\delta(U)}(E) - ax^{\delta(U)} &= p(p+1) - j(j+1) - 2(r - 1 - \frac{j(j+1)}{2}) \\ &= p(p+1) - 2r + 2 - j(j+1) + j(j+1) \\ &= p(p+1) - 2r + 2 \end{aligned}$$

Therefore (4) is not valid for $P_C(\mathcal{K}_{2p+1})$.

To prove sufficiency, we assume that $n_0 + \bar{Q}_k \geq p - k - 1$ for all $k \in \mathbb{Z}_+$ with $k(k+1) \leq 2(r-1)$ and that (4) is not valid. If we end up with a contradiction the result is proved.

Since (4) is not valid for $P_C(\mathcal{K}_{2p+1})$ there exists at least one cut $\delta(U)$ such that $x^{\delta(U)}(E) - ax^{\delta(U)} > p(p+1) - 2r$. As in the proof of Theorem 2.7, we know that $\delta(U)$ cannot intersect all the r cycles of the PBC and that it can only intersect $r - \lambda$ paths P_{ij} for some $\lambda \geq 1 \in \mathbb{Z}_+$. Our goal is to find a cut $\delta(U)$ corresponding to a violation of (4) and we can eliminate all equicuts from our search.

Suppose that $|U| = p - k$ and $|V \setminus U| = p + k + 1$. For the left-hand side of (4) we have:

$$\begin{aligned} x^{\delta(U)}(E) - ax^{\delta(U)} &= p(p+1) - k(k+1) - ax^{\delta(U)} \\ &= p(p+1) - k(k+1) - 2(r - \lambda) \\ &= p(p+1) - 2r + 2\lambda - k(k+1) \end{aligned}$$

Violation of (4) implies that $\lambda > \frac{k(k+1)}{2}$ or, because the quantities in this inequality are integer, $\lambda \geq \frac{k(k+1)}{2} + 1$. Thus, the nodes of U are distributed in such a way that α nodes are not in the PBC and β nodes are concentrated in at most $r - 1 - \frac{k(k+1)}{2}$ paths of the PBC. We have that:

$$\alpha + \beta \leq n_0 + \beta \leq n_0 + \bar{Q}_k \leq p - k - 1$$

where the second inequality comes from the definition of \bar{Q}_k and the last one comes from the assumption. But since $\alpha + \beta = p - k$, we get a contradiction.

Note that, using the definition of \overline{Q}_k , we implicitly have assumed that $\frac{k(k+1)}{2} \leq r-1$. To complete the proof, consider now that $\frac{k(k+1)}{2} \geq r$. For the left-hand side of (4) we get:

$$\begin{aligned} x^{\delta(U)}(E) - ax^{\delta(U)} &= p(p+1) - k(k+1) - ax^{\delta(U)} \\ &\leq p(p+1) - 2r - ax^{\delta(U)} \\ &\leq p(p+1) - 2r \end{aligned}$$

and the inequality is trivially satisfied. This completes the proof of (i).

(ii) : Let U be the set of all nodes not in the PBC and the internal nodes of the paths $\overline{P}_1, \dots, \overline{P}_{r-1}$ (recall that $|V(\overline{P}_i)| - 2 = \overline{q}_i$, $1 \leq i \leq r-1$). From Theorem 2.10, since $ax \geq 2r$ is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ we have that $|U| = n_0 + \overline{Q}_0 = n_0 + \overline{Q} = p-1$. So, $(U, V \setminus U)$ does not form an equipartition of V . Let X be a maximal affinely independent set of incidence vectors of equicuts satisfying (4) at equality, i.e., $|X| = |E| - 1$. The set $X \cup \{x^{\delta(U)}\}$ is affine independent and if we prove that $x^{\delta(U)}$ is on the face defined by (4) in $P_C(\mathcal{K}_{2p+1})$ we are done.

The left-hand side of (4) is given by:

$$x^{\delta(U)}(E) - ax^{\delta(U)} = p(p+1) - 2 - ax^{\delta(U)}$$

So, $x^{\delta(U)}$ satisfies (4) at equality if and only if $ax^{\delta(U)} = 2(r-1)$. This implies that $\overline{P}_1, \dots, \overline{P}_{r-1}$ are all cut twice, which is true if and only if $\overline{q}_i > 0$ for all $i = 1, \dots, r-1$.

The validity of (4) for $P_C(\mathcal{K}_{2p+1})$ implies that $\overline{Q}_1 + n_0 \leq p-2$. By definition, $\overline{Q}_0 = \overline{Q}_1 + \overline{q}_{r-1}$ and $\overline{q}_{r-1} \leq \dots \leq \overline{q}_1$. From Theorem 2.10, $\overline{Q}_0 + n_0 = p-1$. Combining these results we have:

$$\begin{aligned} p-1 &= \overline{Q}_0 + n_0 \\ &= \overline{Q}_1 + \overline{q}_{r-1} + n_0 \\ &\leq p-2 + \overline{q}_{r-1} \leq \dots \leq p-2 + \overline{q}_1 \\ \Rightarrow 1 &\leq \overline{q}_{r-1} \leq \dots \leq \overline{q}_1 \end{aligned}$$

The proof is complete. □

The requirements on the \overline{Q}_k in part (i) of Theorem 2.11 can be relaxed if the collection \mathcal{C} of cycles forming the PBC does not contain two or more cycles that are identical. Figure 2.3 illustrates a PBC in \mathcal{K}_{15} which is the support of a valid inequality for $P_{EC}(\mathcal{K}_{15})$. The numbers indicated in parenthesis refer to the coefficients a_e in the PBC inequality and are omitted if $a_e = 1$. The value of r is 3 and $\overline{Q} + n_0 = 6 \leq p - 1 = 6$. However, the inequality $x(E) - ax \leq 50$ is not valid for $P_C(\mathcal{K}_{15})$ and the cut that violates the inequality is indicated in the figure. For this cut, $x(E) - ax$ is equal to 52.

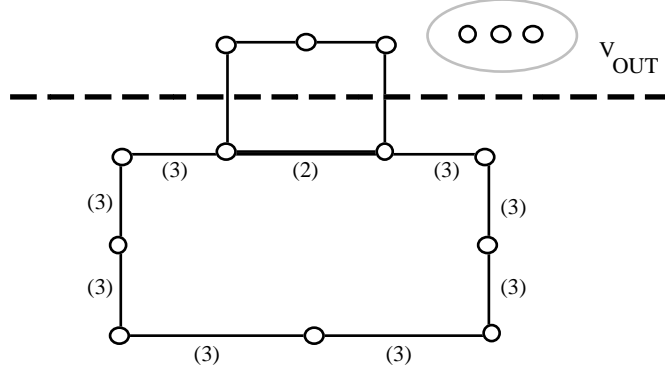


Figure 2.3:

We show now that this situation occurs because $\overline{q}_{r-1} = 0$. Indeed, if $\overline{q}_{r-1} = 0$, then all blocks of the PBC have at least two paths which reduce to the edge joining the source and the destination nodes. Without loss of generality, we can suppose that these paths form the cycles C_r and C_{r-1} of \mathcal{C} and, clearly, $C_r = C_{r-1}$. All the roots of the associated PBC inequality are of type *I* (following the notation of Lemma 2.8) the reason being that, for $\overline{q}_{r-1} = 0$, at most $r - 2$ paths can be cut twice. Therefore, removing C_r from \mathcal{C} and decreasing r by one, we obtain a new inequality defining the same face of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

This cycle elimination can be repeated until we obtain $\overline{q}_{r-1} \geq 1$. This implies that $\overline{q}_1 \geq \dots \geq \overline{q}_{r-1} \geq 1$. If $\overline{Q} + n_0 = \overline{Q}_0 + n_0 \leq p - 1$, we can use the recurrence formula:

$$\overline{Q}_k + n_0 = \overline{Q}_0 + n_0 - \sum_{i=r-\frac{k(k+1)}{2}}^{r-1} \overline{q}_i$$

$$\begin{aligned}
&\leq p - 1 - (r - 1 - r + \frac{k(k+1)}{2} + 1) \\
&\leq p - 1 - \frac{k(k+1)}{2} \\
&\Rightarrow \overline{Q}_k + n_0 \leq p - 1 - k
\end{aligned}$$

The results in the preceeding paragraph, together with Theorem 2.11, can be used to prove the following theorem.

Theorem 2.12 *Consider a PBC which is a subgraph of \mathcal{K}_{2p+1} and such that \mathcal{C} does not contain two or more cycles that are identical. Then, inequality (2) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ if and only if inequality (4) is valid for $P_C(\mathcal{K}_{2p+1})$. Moreover, if inequality (2) defines a facet of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, then inequality (4) defines a facet of $P_C(\mathcal{K}_{2p+1})$.*

An immediate corollary of Theorem 2.12 is the following (see Theorem 2.9).

Corollary 2.13 *Let $ax \geq 2r$ be the PBC inequality corresponding to a PBC subgraph of \mathcal{K}_{2p+1} satisfying conditions (i)-(iv) and such that $n_0 = p - (r - 1)q - 1$. Then, the inequality*

$$(\pi x \leq \pi_0) = (x(E) = p(p+1)) - (ax \geq 2r)$$

is facet defining for $P_C(\mathcal{K}_{2p+1})$.

In the next section we introduce another class of facet defining inequalities for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ having sparse support graphs.

2.5 The Suspended Tree Inequalities

In this section, we describe another generalization of the cycle inequality obtained by replacing the cycle structure by a more complicated graph, namely a (suitably weighted) suspension of a tree. This idea comes from Boros and Hammer (1993) who applied it to construct a class of facets for the cut polytope (see also Remark 1).

Moreover, we shall see that we can relax the conditions on the number of nodes on which the cycle is defined. Clearly, if we take a cycle C which is defined on more than $p+2$ nodes in \mathcal{K}_{2p+1} , inequality (1) remains valid, but is not facet defining; for example, if C is the cycle $(u_1, u_2, \dots, u_{p+3})$, then $x(E(C)) \geq 2$ is the sum of two valid inequalities for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$: $x(E(C')) \geq 2$ and $x_{u_1, u_{p+3}} + x_{u_{p+2}, u_{p+3}} - x_{u_1, u_{p+2}} \geq 0$ where C' is the cycle $(u_1, u_2, \dots, u_{p+2})$. However, if we take a cycle C which is defined on less than $p+2$ nodes, *e.g.* on $p+1$ or p nodes, by suitably modifying the original inequality we will produce some facet defining inequality for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

A nice feature of the inequality (1) is the planarity of its support graph. The generalization we proposed in the previous section and most of those we present in this section still possess the planarity property. This is indeed the case for the inequalities (5) and (6). From Lemma 2.4, all these inequalities provide facets for the equicut polytope of planar graphs.

Let $(V(T), T)$ be a tree in \mathcal{K}_{2p+1} and u_0 be a node in $V \setminus V(T)$. For every node $u \in V(T)$, we denote by d_u the degree of u in T . We consider the following inequality:

$$\omega(T, u_0)x := \sum_{u \in V(T)} (2 - d_u)x_{u_0u} + \sum_{uv \in T} x_{uv} \geq 2 \quad (5)$$

Note that, if T is the path (u_1, u_2, \dots, u_k) , then $\omega(T, u_0)x = x(E(C))$, where C is the cycle $(u_0, u_1, u_2, \dots, u_k)$. Therefore, inequality (5) generalizes the cycle inequality. The support graph of the inequality (5) is a suspension of the tree T with apex u_0 , whose edges are the pairs (u_0, u) for $u \in V(T)$ such that $d_u \neq 2$ and $(u, v) \in T$. An example

of a suspended tree is shown in Figure 2.4. The dashed lines correspond to the edges having negative coefficients in (5). Coefficients with absolute values different from one are indicated in parenthesis.

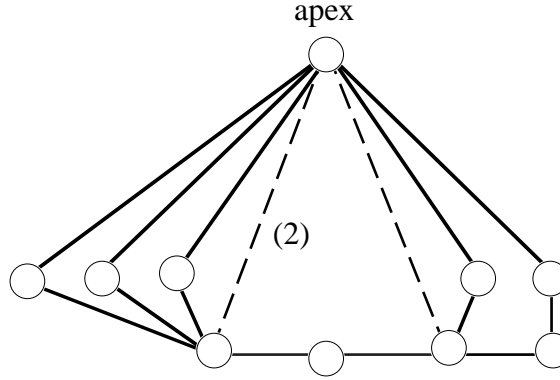


Figure 2.4:

Lemma 2.14 *Let U be a subset of $V(T)$ and consider the complete graph with node-set $V(T) \cup \{u_0\}$. Then $\omega(T, u_0)x^{\delta(U)} = 2c(U)$, where $c(U)$ is the number of connected components of the subgraph $(U, T(U))$ of $(V(T), T)$ induced by U .*

Proof: We compute:

$$\begin{aligned}
 \omega(T, u_0)x^{\delta(U)} &= \sum_{u \in U} (2 - d_u) + |T \cap \delta(U)| \\
 &= 2|U| - \sum_{u \in U} |\{uv \in T : v \in V(T)\}| + \\
 &\quad \sum_{u \in U} |\{uv \in T : v \in V(T) \setminus U\}| \\
 &= 2|U| - \sum_{u \in U} |\{uv \in T : v \in U\}| \\
 &= 2|U| - 2(|U| - c(U)) = 2c(U)
 \end{aligned}$$

□

An immediate corollary is:

Corollary 2.15 *Let T be a tree of \mathcal{K}_{2p+1} with $|V(T)| = p + 1$ and u_0 be a node of $V(\mathcal{K}_{2p+1}) \setminus V(T)$, then inequality (5) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. Moreover, the equicuts of*

\mathcal{K}_{2p+1} satisfying the equality $\omega(T, u_0) x = 2$ are of the form $\delta(U)$ with $|U| = p, p + 1$ and $T(U \cap V(T))$ is connected.

A connected graph $G = (V, E)$ is called a *star* if there exists $u \in V$ (the center of the star) such that $G - \{u\}$ consists only of isolated nodes, i.e., $E = \{(v, u) : v \in V \setminus \{u\}\}$.

Theorem 2.16 *Let T be a tree of \mathcal{K}_{2p+1} with $|V(T)| = p + 1$ and u_0 be a node of $V(\mathcal{K}_{2p+1}) \setminus V(T)$. Then, inequality (5) defines a facet of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ if and only if T is not a star.*

Proof: We first prove necessity. For this, suppose that T is a star and $p \geq 2$. We want to prove that (5) cannot be a facet defining inequality for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. The case $p = 1$ will be treated separately.

Let v be the center of the star T and consider the complete subgraph \mathcal{K}_{2p-1} defined on the nodes in $V \setminus \{u_0, v\}$. It is not difficult to see that the incidence vector $x^{\delta(U)}$ of any equicut $\delta(U)$ that satisfies (5) at equality also satisfies $x(E(\mathcal{K}_{2p-1})) = p(p-1)$. Thus, (5) cannot define a facet of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

If $p = 1$ any tree on $p + 1$ nodes is a star composed by a single edge and the suspended tree support graph is the complete graph \mathcal{K}_3 . Thus, inequality (5) does not induce a proper face of $\mathcal{P}_{EC}(\mathcal{K}_3)$ and necessity is proved.

To prove sufficiency we apply Lemma 2.6 and, once more, we restrict ourselves to show how the sets S_1, S_2, T_1 and T_2 are constructed. The resulting equicuts can be easily seen to define roots of (5) by applying Lemma 2.14.

Again, we use the following characterization for proving that inequality (5) is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. Let $\pi x \geq \pi_0$ be a valid inequality for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ such that:

$$\begin{aligned} F &= \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \omega(T, u_0) x = 2\} \\ F_\pi &= \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \pi x = \pi_0\} \end{aligned}$$

and $F \subseteq F_\pi$. We show that there exist some scalars λ_1, λ_2 such that:

$$\begin{aligned}\pi x &= \lambda_1 \omega(T, u_0) x + \lambda_2 \sum_{e \in E(\mathcal{K}_{2p+1})} x_e \quad \text{and} \\ \pi_0 &= 2\lambda_1 + \lambda_2 p(p+1)\end{aligned}$$

This amounts to check the existence of some scalars α, β such that the following conditions hold:

- $\pi_{u,v} = \alpha$ for all edges $(u, v) \in T$;
- $\pi_e = \beta$ for all edges $e \in E(\mathcal{K}_{2p+1}) \setminus (T \cup \{(u_0, u) : u \in V(T)\})$
- $\pi_{u_0, u} = (\alpha - \beta)(2 - d_u) + \beta$ for all $u \in V(T)$;
- $\pi_0 = 2(\alpha - \beta) + \beta p(p+1)$

We set $V = V(\mathcal{K}_{2p+1}) = V(T) \cup \{u_0\} \cup V'$, implying that $|V'| = p-1$. We just indicate how to build the sets W, S_1, S_2, T_1 and T_2 that suggests the use of Lemma 2.6 leading to the results below.

(a) : Take any three distinct nodes u', v' and w' of V' . Set $A' = V' \setminus \{u', v', w'\}$ and take a subset A of three nodes in $V(T)$ that induces a connected subtree of T . Now, let $w = u'$, $S_1 = \{v'\}$, $S_2 = \{w'\}$, $T_1 = A \cup A'$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. From Lemma 2.6, we deduce that $\pi_{u'v'} = \pi_{u'w'}$ and, therefore, there exists a scalar β such that $\pi_{u'v'} = \beta$ for all u' and v' in V' .

(b) : Take $u \in V(T)$ and two distinct nodes $u', v' \in V'$. Since T is not a star, we can find an edge (v, z) ($v, z \neq u$) such that $T(\{u, v, z\})$ is connected. Now, we set $w = u'$, $S_1 = \{v'\}$, $S_2 = \{u\}$, $T_1 = \{v, z\} \cup (V' \setminus \{u', v'\})$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. From Lemma 2.6, conclude that:

$$\pi_{u'u} = \pi_{u'v'}$$

Hence,

$$\pi_{uu'} = \beta \quad \text{for all } u \in V(T), u' \in V'$$

(c) : Let A be a subset of $V(T)$ such that $2 \leq |A| \leq p-1$ and both $T(A)$ and $T(V(T) \setminus A)$ are connected (to find such set A , take a node u of $V(T)$ of degree at least 2 in T ; since T is not a star, some component of $T - \{u\}$ is not reduced to an isolated node; chose its nodeset for A). Take distinct nodes $u', v' \in V'$ and $A' \subseteq V' \setminus \{u', v'\}$ with $|A'| + |A| = p-1$. Now, we set: $w = u'$, $S_1 = \{v'\}$, $S_2 = \{u_0\}$, $T_1 = A \cup A'$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. From Lemma 2.6, we have:

$$\pi_{u'u_0} = \pi_{u'v'} = \beta$$

Hence,

$$\pi_{u_0u'} = \beta \quad \text{for all } v \in V'$$

(d) : Take two distinct nodes u and v in $V(T)$ that are not adjacent in T . Let P_{uv} be the path in T joining u to v ; set $A = V(P_{uv}) \setminus \{u, v\}$ (clearly, $1 \leq |A| \leq p-1$). Take $u' \in V'$ and $A' \subseteq V' \setminus \{u'\}$ with $|A| + |A'| = p-1$. Setting: $w = u$, $S_1 = \{v\}$, $S_2 = \{u'\}$, $T_1 = A \cup A'$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. From Lemma 2.6, we deduce that:

$$\pi_{uv} = \pi_{uu'} = \beta$$

Or, more generally:

$$\pi_{uv} = \beta \quad \text{for all } u \in V(T), v \in V(T) \text{ with } (uv) \notin T$$

(e) : Assume that u, v and z are in $V(T)$ and that u is adjacent to both v and z . Let $w = u$, $S_1 = \{v\}$, $S_2 = \{z\}$, $T_1 = V'$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. Using Lemma 2.6, we prove that:

$$\pi_{uv} = \pi_{uz}$$

By connectivity of the tree T we can deduce that, for some scalar α , $\pi_e = \alpha$ for all edges $e \in T$.

Finally, we compute the value of π_{u_0u} for $u \in V(T)$ and of π_0 . Take an edge (u, v) of T ; both sets $A = \{u, v\} \cup V'$ and $A' = \{v\} \cup V'$ define roots of (5), so we deduce that:

$$\begin{aligned} 0 &= \pi x^{\delta(A)} - \pi x^{\delta(A')} && \Rightarrow \\ 0 &= \pi_{u_0u} + \alpha(d_u - 1) + \beta(p - d_u) - \beta(p - 1) - \alpha && \Rightarrow \\ \pi_{u_0u} &= (2 - d_u)(\alpha - \beta) + \beta \end{aligned}$$

Moreover, for $A'' = V' \cup \{u\}$:

$$\begin{aligned} \pi_0 &= \pi x^{\delta(A'')} && \Rightarrow \\ \pi_0 &= \beta(p - 1)(p + 1) + \pi_{u_0u} + \alpha d_u + \beta(p - d_u) && \Rightarrow \\ \pi_0 &= \beta p(p + 1) + 2(\alpha - \beta) \end{aligned}$$

□

Remark 1 : If we relax the connectivity condition and suppose that T is a forest with $|V(T)| = p + 1$, then inequality (5) remains valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, however, the facet property is lost if T is not a tree. Indeed, suppose T has k connected components T_1, \dots, T_k ; choose two leaves ℓ_i and ℓ'_i in each tree T_i and construct the tree T' obtained by adding to T the edges (ℓ'_{i+1}, ℓ_i) for $i = 1, 2, \dots, k - 1$. Then, we have that:

$$\omega(T, u_0)x = \omega(T', u_0)x + \sum_{i=1}^{k-1} (x_{u_0, \ell'_i} + x_{u_0, \ell_{i+1}} - x_{\ell'_i, \ell_{i+1}})$$

and, thus, the inequality $\omega(T, u_0)x \geq 2$ is the sum of the valid inequalities: $\omega(T', u_0)x \geq 2$ and the inequalities $x_{u_0, \ell'_i} + x_{u_0, \ell_{i+1}} - x_{\ell'_i, \ell_{i+1}} \geq 0$ for $i = 1, \dots, k - 1$, and hence is not facet defining.

We now see how to modify inequality (5) (or inequality (1)) when the tree T is defined on p nodes only. Let T be a tree of \mathcal{K}_{2p+1} with p nodes and u_1, u_2, u_3 be distinct nodes of $V(\mathcal{K}_{2p+1}) \setminus V(T)$. Let Δ denote the clique (triangle) with nodes u_1, u_2 and u_3 . We set $\Delta x := x_{u_1, u_2} + x_{u_1, u_3} + x_{u_2, u_3}$. We consider the inequality:

$$\omega(T, \Delta, u_0)x = \sum_{u \in V(T)} (2 - d_u)x_{u_0, u} + \sum_{e \in T} x_e - \sum_{e \in \Delta} x_e \geq 0 \quad (6)$$

Thus, $\omega(T, \Delta, u_0)x$ is given by $\omega(T, u_0)x - \Delta x$. Figure 2.5 shows the support graph of such an inequality.

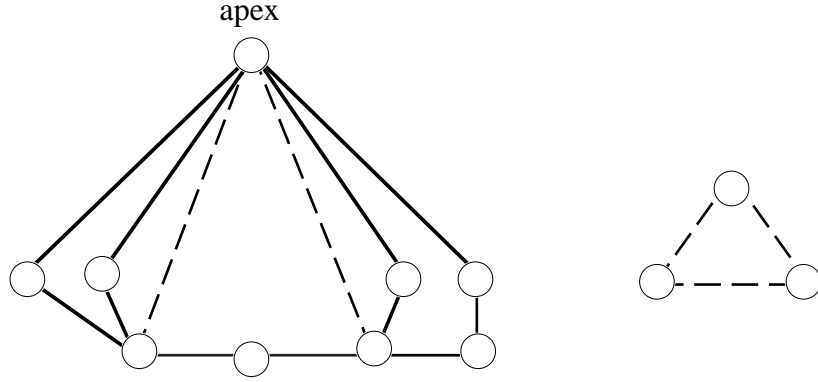


Figure 2.5:

Proposition 2.17 *Let T be a tree of \mathcal{K}_{2p+1} with $|V(T)| = p$, Δ be a triangle with $V(\Delta) \subseteq V(\mathcal{K}_{2p+1}) \setminus V(T)$ and u_0 be a node in $V(\mathcal{K}_{2p+1}) \setminus (V(T) \cup V(\Delta))$. Then, inequality (6) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.*

Proof : Take an equicut $\delta(U)$ of \mathcal{K}_{2p+1} with $|U| = p, p+1$ and we can suppose that $u_0 \notin U$. Then:

$$\begin{aligned} \omega(T, \Delta, u_0)x^{\delta(U)} &= \omega(T, u_0)x^{\delta(U)} - \Delta x^{\delta(U)} \\ &= 2c(U \cap V(T)) - \Delta x^{\delta(U)} \quad (\text{by Lemma 2.14}) \end{aligned}$$

If $\Delta x^{\delta(U)} = 0$, then clearly $\omega(T, \Delta, u_0)x^{\delta(U)} \geq 0$; else, $\Delta x^{\delta(U)} = 2$, implying that $|U \cap V(\Delta)| = 1$ or 2 and thus, since $|U| = p$ or $p+1$, $|U \cap V(\Delta)| \geq 1$ and so $c(U \cap V(T)) \geq 1$, implying that $\omega(T, \Delta, u_0)x^{\delta(U)} \geq 0$ and this completes the proof. \square

As a consequence of the previous proof, the equicuts of \mathcal{K}_{2p+1} satisfying (6) at equality are of the form $\delta(U)$ with $U = V(\mathcal{K}_{2p+1}) \setminus (V(T) \cup \{u_0\})$ or, $|U| = p, p+1$ and $|U \cap V(\Delta)| = 1, 2$, $T(U \cap V(T))$ is connected.

Theorem 2.18 *Let T be a tree of \mathcal{K}_{2p+1} with $|V(T)| = p$, u_0, u_1, u_2, u_3 be nodes of $V(\mathcal{K}_{2p+1}) \setminus V(T)$ and Δ denote the triangle with nodes u_1, u_2, u_3 . If $p \geq 5$ and T is not a star, then inequality (6) defines a facet of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$,*

Proof : Take a valid inequality $\pi x = \pi_0$ for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ and let:

$$\begin{aligned} F &= \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \omega(T, \Delta, u_0) x = 0\} \\ F_\pi &= \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \pi x = \pi_0\} \end{aligned}$$

and $F \subseteq F_\pi$. We show below that there exists some scalars α and β such that:

- $\pi_{uv} = \alpha$ for all edges $(uv) \in T$;
- $\pi_{uv} = \beta$ for all $(uv) \in E(\mathcal{K}_{2p+1}) \setminus (T \cup \Delta \cup E_0)$ where $E_0 = \{(u_0, u) : u \in V(T)\}$;
- $\pi_{u_0 u} = (\alpha - \beta)(2 - d_u) + \beta$ for all $u \in V(T)$;
- $\pi_{uv} = 2\beta - \alpha$ for all $u, v \in V(\Delta)$, $u \neq v$;
- $\pi_0 = \beta p(p + 1)$

We set $V' = V(\mathcal{K}_{2p+1}) - (V(T) \cup \{u_0\} \cup V(\Delta))$, so the size of V' is $p - 3$.

The idea of the proof is basically the same of the one for Theorem 2.16. We want to use Lemma 2.6 and, for this, we need to define the sets W , S_1 , S_2 , T_1 and T_2 such that the incidence vectors of the equicuts defined in the lemma are in F . The former can be checked easily from the result contained in the paragraph just preceding the theorem.

(a) : Let A be a subset of three nodes of $V(T)$ such that $T(A)$ is connected. Take z, u, v in V' , $A' = V' \setminus \{u, v, z\}$ and set: $w = z$, $S_1 = \{u\}$, $S_2 = \{v\}$, $T_1 = A \cup A' \cup \{u_1, u_2\}$ and $T_2 = V \setminus (\{w\} \cup S_1 \cup S_2 \cup T_1)$. Using Lemma 2.6, we have that: $\pi_{zu} = \pi_{zv}$. Hence, for some scalar β :

$$\pi_{zu} = \beta \quad \text{for all } z, u \in V' \text{ and } z \neq u$$

Take now two distinct nodes z, u in V' , $A' = V' \setminus \{z, u\}$ and set: $w = z$, $S_1 = \{u\}$, $S_2 = \{u_1\}$, $T_1 = A \cup A' \cup \{u_2\}$ and $T_2 = V \setminus (\{w\} \cup S_1 \cup S_2 \cup T_1)$. Lemma 2.6 implies that: $\pi_{z, u_1} = \pi_{zu} = \beta$. Similarly, one can prove that: $\pi_{z, u_1} = \pi_{z, u_2} = \pi_{z, u_3} = \beta$.

(b) : Given an edge (uv) of T , set: $W = \{u_1\}$, $S_1 = \{u_2\}$, $S_2 = \{u_3\}$, $T_1 = V' \cup \{u, v\}$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. Lemma 2.6 implies that:

$$\pi_{u_1 u_2} = \pi_{u_1 u_3}$$

Or similarly,

$$\pi_{u_i u_j} = \beta' \quad \text{for all } i, j \in \{1, 2, 3\}, i \neq j$$

(c) : Given $u \in V(T)$, let A be a subset of $V(T) \setminus \{u\}$ such that $2 \leq |A| \leq p-2$ and $T(A)$, $T(A \cup \{u\})$ and $T(V(T) \setminus A)$ be connected subtrees of T (this is possible, since T is not a star). Take $w \in V'$, $A' \subseteq V' \setminus \{w\}$ such that $|A| + |A'| = p-2$. With $W = \{w\}$, $S_1 = \{u\}$, $S_2 = \{u_1\}$, $T_1 = A \cup A' \cup \{u_2\}$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$ we build the roots of (6) which, from Lemma 2.6, imply that: $\pi_{wu} = \pi_{wu_1} = \beta$. Hence:

$$\pi_{wu} = \beta \quad \text{for all } u \in V(T), w \in V'$$

On the other hand, setting: $W = \{u_1\}$, $S_1 = \{u\}$, $S_2 = \{w\}$ and keeping T_1 and T_2 as before, we obtain different roots of (6), implying that: $\pi_{u_1 u} = \pi_{u_1 w} = \beta$. Hence:

$$\pi_{uu_1} = \pi_{uu_2} = \pi_{uu_3} = \beta \quad \text{for all } u \in V(T)$$

Also for the same sets T_1 and T_2 , setting $W = \{w\}$, $S_1 = \{u_0\}$ and $S_2 = \{u_1\}$, will define new roots of (6) leading to the conclusion that: $\pi_{wu_0} = \pi_{wu_1} = \beta$; and for $W = \{u_1\}$, $S_1 = \{u_0\}$ and $S_2 = \{w\}$ the roots obtained for (6) imply that: $\pi_{u_1 u_0} = \pi_{u_1 w} = \beta$. Therefore,

$$\pi_{u_0 v} = \beta \quad \text{for all } v \in V(\Delta) \cup V'$$

(d) : Take two distinct nodes $w, u \in V(T)$ that are not adjacent in T . Let $P = (v_1 = w, v_2, \dots, v_k, v_{k+1} = u)$ be a path in T joining u to w . Let $A = \{v_2, \dots, v_k\}$, so that $1 \leq |A| \leq p-2$ and take $A' \subseteq V'$ with $|A| + |A'| = p-2$. Set $W = \{w\}$, $S_1 = \{u\}$, $S_2 = \{u_2\}$, $T_1 = A \cup A' \cup \{u_1\}$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. From Lemma 2.6, we deduce that $\pi_{wu} = \pi_{wu_2} = \beta$.

(e) : Take distinct nodes $w, u, v \in V(T)$ such that w is adjacent to both u and v in T .
Let : $W = \{w\}$, $S_1 = \{u\}$, $S_2 = \{v\}$, $T_1 = V' \cup \{u_1, u_2\}$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$.
The usual argument shows that: $\pi_{wu} = \pi_{wv}$. By connectivity of T , we deduce that, for some scalar α :

$$\pi_{uv} = \alpha \quad \text{for all edges } (uv) \in T$$

(f) : It only remains to check that $\beta' = 2\beta - \alpha$, and to compute the values of π_{u_0u} for $u \in V(T)$ and of π_0 . Take $u \in V(T)$, then $\delta(V' \cup \{u, u_1, u_2\})$ is a root of (6), implying that:

$$\pi_0 = \pi x^{\delta(V' \cup \{u, u_1, u_2\})}$$

and, thus:

$$\pi_0 = \pi_{u_0u} + \alpha d_u + 2\beta' + \beta(p^2 + p - 3 - d_u)$$

(g) : Now, if $(uv) \in T$, then $\delta(V' \cup \{u, v, u_1, u_2\})$ is a root of (6), implying that:

$$\pi_0 = \pi_{u_0u} + \pi_{u_0v} + 2\beta' + \alpha(d_u + d_v - 2) + \beta(p^2 + p - 2 - d_u - d_v)$$

(h) : Comparing the equations obtained in (e) and (f), we get:

$$\pi_0 = 2\beta' + 2\alpha + \beta(p^2 + p - 4)$$

Using the equation in (h), we deduce from (f) that:

$$\pi_{u_0,u} = (\alpha - \beta)(2 - d_u) + \beta$$

Since $\delta(V' \cup V(\Delta))$ is in F , we can deduce that:

$$\pi_0 = \pi x^{\delta(V' \cup V(\Delta))} = \beta p(p + 1)$$

Finally, comparing this value of π_0 with the one given in (h), we show that: $\beta' = 2\beta - \alpha$.
This concludes the proof. \square

We now describe a second extension for the inequalities (5) and (6) for the case when the tree T is defined on $p - 1$ nodes instead of $p + 1$ or p nodes.

Given five nodes u_1, \dots, u_5 , consider the cycles $C = (u_1, u_2, u_3, u_4, u_5)$ and $C' = (u_1, u_3, u_5, u_2, u_4)$. Let us call *chorded pentagon*, denoted by CP , the weighted graph on the nodes u_1, \dots, u_5 whose edges are those of C with weights $+1$ and those of C' with weights -1 . We set $CP.x = \sum_{e \in E(C)} x_e - \sum_{e \in E(C')} x_e$. Let T be a tree of \mathcal{K}_{2p+1} , u_0, u_1, \dots, u_5 be nodes of $V(\mathcal{K}_{2p+1}) \setminus V(T)$ and CP be the chorded pentagon defined on nodes u_1, \dots, u_5 . We consider the inequality:

$$\omega(T, CP, u_0)x = \sum_{u \in V(T)} (2 - d_u)x_{u_0, u} + \sum_{e \in T} x_e + \sum_{e \in C} x_e - \sum_{e \in C'} x_e \geq 0 \quad (7)$$

So, $\omega(T, CP, u_0)x$ is given by $\omega(T, u_0)x + CP.x$. The support graph of such an inequality is shown in Figure 2.6.

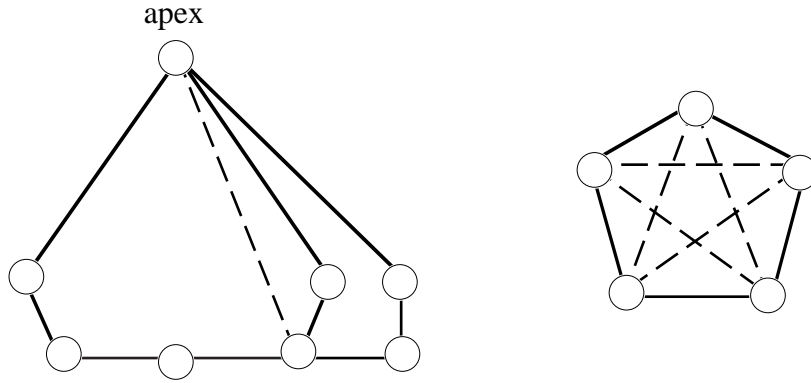


Figure 2.6:

Proposition 2.19 *Let T be a tree of \mathcal{K}_{2p+1} with $|V(T)| = p-1$, CP be a chorded pentagon with $V(CP) = \{u_1, \dots, u_5\} \subseteq V(\mathcal{K}_{2p+1}) \setminus V(T)$ and $u_0 \in V(\mathcal{K}_{2p+1}) \setminus (V(T) \cup V(CP))$. Then, the inequality (7) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.*

Proof : Take an equicut $\delta(U)$ of \mathcal{K}_{2p+1} with $u_0 \notin U$. One checks easily that, if $|U \cap V(CP)| = 1$ or 4 , then $CP.x^{\delta(U)} = 0$; if $U \cap V(CP)$ is of the form $\{u_1, u_2\}$ or $\{u_1, u_2, u_3\}$ (i.e., a circular interval of size 2, 3 on the cycle C), then $CP.x^{\delta(U)} = -2$ and $CP.x^{\delta(U)} = 2$ otherwise. But, if $|U \cap V(CP)| \leq 3$, then, since $|U| = p$ or $p+1$, $|U \cap V(T)| \geq 1$ and, thus, by Lemma 2.14, $\omega(T, CP, u_0)x^{\delta(U)} \geq 2$. Therefore, $\omega(T, CP, u_0).x^{\delta(U)} \geq 0$ for all equicuts $\delta(U)$. \square

Let us set $V' = V(\mathcal{K}_{2p+1}) \setminus (V(T) \cup V(CP) \cup \{u_0\})$, so $|V'| = p - 4$. We see from the last proof that the equicuts whose incidence vectors satisfy equality $\omega(T, CP, u_0).x = 0$ are of the form $\delta(U)$ with $U = V(CP) \cup V'$, $V(CP) \setminus \{u_h\} \cup V'$ for $u_h \in V(CP)$, $V(CP) \cup V' \setminus \{v\}$ for $v \in V'$ and U such that $T(U \cap V(T))$ is connected, $U \cap V(CP) = \{u_h, u_{h+1}\}$ or $\{u_h, u_{h+1}, u_{h+2}\}$ (indices taken modulo 5) and $|U| = p$ or $p + 1$.

Given an integer $k \geq 1$, a connected graph $G = (V, E)$ is called a k -star with center v_1 if $v_1 \in V$ and all the connected components of $G - v_1$ are trees on at most k nodes. In particular, a 1-star is a usual star and, in a 2-star with center v_1 , all components of $G - v_1$ are isolated nodes or edges.

Theorem 2.20 *Let T be a tree of \mathcal{K}_{2p+1} with $|V(T)| = p - 1$, CP be a chorded pentagon of \mathcal{K}_{2p+1} with $V(CP) \subseteq V(\mathcal{K}_{2p+1}) \setminus V(T)$ and $u_0 \in V(\mathcal{K}_{2p+1}) \setminus (V(T) \cup V(CP))$. If $p \geq 6$ and T is not a 2-star, then inequality (7) is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.*

Proof: Let us consider a valid inequality $\pi x \geq \pi_0$ for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ such that:

$$\begin{aligned} F &= \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \omega(T, CP, u_0).x = 0\} \\ F_\pi &= \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \pi x = \pi_0\} \end{aligned}$$

and $F \subseteq F_\pi$. We show below that there exist some scalars α and β such that:

- $\pi_{uv} = \alpha$ for all edges $(uv) \in T$;
- $\pi_{uv} = \beta$ for all $(uv) \in E(\mathcal{K}_{2p+1}) \setminus (T \cup E(CP) \cup E_0)$ where $E_0 = \{(u_0, u) : u \in V(T)\}$;
- $\pi_{u_0 u} = (\alpha - \beta)(2 - d_u) + \beta$ for all $u \in V(T)$;
- $\pi_{u_i u_j} = \alpha$ for all $(u_i, u_j) \in C$, $i \neq j$;
- $\pi_{u_i u_j} = 2\beta - \alpha$ for all $(u_i, u_j) \in C'$, $i \neq j$;
- $\pi_0 = \beta p(p + 1)$

The proof is similar to those for Theorems 2.16 and 2.18: we give the sets S_1 , S_2 , T_1 and T_2 and the node w used to construct the equicuts which can be easily check to be roots of (7) (see Proposition 2.19).

(a) : For some scalar β , $\pi_{uv} = \beta$ for $u, v \in V'$ and $\pi_{wu} = \beta$ for all $u \in V'$ and all $w \in V(CP)$ (the details are omitted, the proof is similar to that of (a) in Theorem 2.16).

(b) : Since, by assumption, T is not a 2-star, given $u \in V(T)$, we find $A \subseteq V(T) \setminus \{u\}$ such that $3 \leq |A| \leq p-3$ and $T(A)$, $T(A \cup \{u\})$, $T(V(T) \setminus A)$ are all connected.

For $v, z \in V'$, take $A' \subseteq V' \setminus \{v, z\}$ such that $|A| + |A'| = p-3$ and set: $w = z$, $S_1 = \{u\}$, $S_2 = \{v\}$, $T_1 = A \cup A' \cup \{u_1, u_2\}$ and $T_2 = V \setminus (\{w\} \cup S_1 \cup S_2 \cup T_1)$. We obtain roots of (7) which from Lemma 2.6 imply: $\pi_{zu} = \pi_{zv} = \beta$. Keeping T_1 and T_2 as above and taking $w = u_0$, $S_1 = \{z\}$, $S_2 = \{v\}$, we obtain:

$$\pi_{u_0 z} = \beta \quad \text{for all } z \in V'$$

For $v \in V'$ and $u \in V(T)$, take $A' \subseteq V' \setminus \{v\}$ such that $|A| + |A'| = p-3$ and set: $w = u_1$, $S_1 = \{u\}$, $S_2 = \{v\}$, $T_1 = A \cup A' \cup \{u_2, u_3\}$ and $T_2 = V \setminus (\{w\} \cup S_1 \cup S_2 \cup T_1)$. We then build the roots of (7) leading to:

$$\pi_{u_1 u} = \pi_{u_1 v} = \beta$$

By changing the sets S_1 and S_2 to: $S_1 = \{u_0\}$, $S_2 = \{v\}$, we define new roots of (7) implying that:

$$\pi_{u_1 u_0} = \pi_{u_1 v} = \beta$$

Therefore,

$$\pi_{u_0 u_h} = \pi_{u_h u} = \beta \quad \text{for all } u \in V(T) \text{ and } u_h \in V(CP)$$

(c) : If $w, u \in V(T)$ and are not adjacent in T , take a path $P = \{v_1 = w, v_2, \dots, v_k, v_{k+1} = u\}$ joining w to u . Let $A = \{v_2, \dots, v_k\}$, $A' \subseteq V'$ such that $|A| + |A'| = p-3$ and set: $W = \{w\}$, $S_1 = \{u\}$, $S_2 = \{u_1\}$, $T_1 = A \cup A' \cup \{u_2, u_3\}$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. We build the roots of (7) and deduce that:

$$\pi_{wu} = \pi_{wu_1} = \beta$$

(d) : If $w, u \in V(T)$ and are adjacent in T , then $\pi_{wu} = \alpha$ for some constant α . Indeed, if w is adjacent to u and v in T , set $W = \{w\}$, $S_1 = \{u\}$, $S_2 = \{v\}$, $T_1 = V' \cup \{u_1, u_2, u_3\}$ and $T_2 = V \setminus (W \cup S_1 \cup S_2 \cup T_1)$. We build the roots of (7) and deduce that:

$$\pi_{wu} = \pi_{wv} = \alpha$$

The result follows by the connectivity of T .

(e) : Since $\delta(V(CP) \cup V')$ is a root of (7), we deduce that:

$$\pi_0 = \pi_x^{\delta(V(CP) \cup V')} = \beta p(p+1)$$

Given an edge $(uv) \in T$, set $U = \{u, v\} \cup V'$. The equicuts $U_1 = U \cup \{u_2, u_3\}$, $U_2 = U \cup \{u_4, u_5\}$, $U_3 = U \cup \{u_1, u_2, u_3\}$ and $U_4 = U \cup \{u_1, u_4, u_5\}$ are in F , implying the following relation:

$$\pi_{u_1 u_2} + \pi_{u_1 u_3} = \pi_{u_1 u_4} + \pi_{u_1 u_5}$$

(f) : Since $\delta((V(CP) \setminus \{u_5\}) \cup V')$ is a root of (7), we deduce that:

$$\begin{aligned} \pi_0 &= \pi_x^{\delta((V(CP) \setminus \{u_5\}) \cup V')} \\ &= \beta(p+1)(p-4) + 4\beta p + \pi_{u_1 u_5} + \pi_{u_2 u_5} + \pi_{u_3 u_5} + \pi_{u_4 u_5} \\ &= \beta(p+1)(p-4) + 4\beta p + 2(\pi_{u_1 u_5} + \pi_{u_2 u_5}) \end{aligned}$$

Therefore, we deduce that:

$$\pi_{u_1 u_5} + \pi_{u_2 u_5} = 2\beta$$

(g) : Let:

$$\gamma = \pi_{u_1 u_4} = \pi_{u_1 u_3} = \pi_{u_2 u_4} = \pi_{u_2 u_5} = \pi_{u_3 u_5}$$

then,

$$\pi_{u_1 u_2} = \pi_{u_2 u_3} = \pi_{u_3 u_4} = \pi_{u_4 u_5} = \pi_{u_1 u_5} = 2\beta - \gamma$$

For $u \in V(T)$ and $A = \{u_1, u_2, u_3, u\} \cup V'$, $x^{\delta(A)}$ is a root of (7), implying that:

$$\begin{aligned} \pi_0 &= \pi_x^{\delta(A)} \\ &= \pi_{u_0 u} + (\alpha - \beta)d_u + \beta(p^2 + p - 3) + \pi_{u_4 u_2} + \pi_{u_4 u_1} \end{aligned}$$

Therefore,

$$\begin{aligned}\pi_{u_0u} &= \beta p(p+1) - (\alpha - \beta)d_u - \beta(p^2 + p - 3) - 2\gamma \quad i.e. \\ \pi_{u_0u} &= (\beta - \alpha)d_u + 3\beta - 2\gamma\end{aligned}$$

Given an edge $(uv) \in T$ and $A = \{u_1, u_2, u, v\} \cup V'$, $x^{\delta(A)}$ is a root of (7), implying that: $\pi_0 = \pi x^{\delta(A)}$, from which we deduce that:

$$0 = \pi_{u_0u} + \pi_{u_0v} + (\alpha - \beta)(d_u + d_v) - 2\alpha - 2\beta + 2\gamma$$

Using the last relation in (g), we deduce that: $\gamma = 2\beta - \alpha$. Hence, from (g),

$$\pi_{u_0u} = (\alpha - \beta)(2 - d_u) + \beta$$

Also,

$$\pi_{u_1u_2} = 2\beta - \gamma = \alpha$$

and

$$\pi_{u_1u_3} = \gamma = 2\beta - \alpha$$

This concludes the proof. \square

We have seen in the previous section that an appropriate linear combination of the equation $x(E) = p(p+1)$ and a facet defining inequality $\pi x \leq \pi_0$ of $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ produces a facet defining inequality for $P_C(\mathcal{K}_{2p+1})$, i.e., for the cut polytope. Consider the following three inequalities obtained by subtracting inequalities (5), (6) and (7), respectively, from the equation $x(E) = p(p+1)$:

$$x(E) - \omega(T, u_0)x \leq p(p+1) - 2 \tag{8}$$

$$x(E) - \omega(T, \Delta, u_0)x \leq p(p+1) \tag{9}$$

$$x(E) - \omega(T, CP, u_0)x \leq p(p+1) \tag{10}$$

To prove that these inequalities are facet defining for $P_C(\mathcal{K}_{2p+1})$, we have to show that they are valid and to exhibit a cut which is not an equicut and which has the incidence vector on the face defining in $P_C(\mathcal{K}_{2p+1})$ for each of the inequalities (8)-(10).

Theorem 2.21 *Suppose that the inequalities $\omega(T, u_0)x \geq 2$, $\omega(T, \Delta, u_0)x \geq 0$ and $\omega(T, CP, u_0)x \geq 0$ are facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$. Then, inequalities (8), (9) and (10) are facet defining for $P_C(\mathcal{K}_{2p+1})$.*

Proof:

Validity: Let $\delta(U)$ be a cut and $|U| = p - k$. This implies that $x^{\delta(U)}(E) = p(p+1) - k(k+1)$ which is decreasing with k . Inequalities (8)-(10) are valid for equicuts and therefore we only have to check validity when $k \geq 1$. From Propositions 2.15, 2.17 and 2.19, it is easy to see that $-\omega(T, u_0)x$, $-\omega(T, \Delta, u_0)x$ and $-\omega(T, CP, u_0)x$ are bounded respectively by the values 0, +2 and +2. Since $k \geq 1$, we have that:

$$\begin{aligned}
\text{(i)} \quad x^{\delta(U)}(E) - \omega(T, u_0)x^{\delta(U)} &\leq (p(p+1) - 2) + 0 \\
&= p(p+1) - 2 \\
\text{(ii)} \quad x^{\delta(U)}(E) - \omega(T, \Delta, u_0)x^{\delta(U)} &\leq p(p+1) - 2 + 2 \\
&= p(p+1) \\
\text{(iii)} \quad x^{\delta(U)}(E) - \omega(T, CP, u_0)x^{\delta(U)} &\leq p(p+1) - 2 + 2 \\
&= p(p+1)
\end{aligned}$$

Thus the validity of (8) comes from (i) and those of (9) and (10) from (ii) and (iii) respectively.

Facet: Consider the cuts:

$$\begin{aligned}
U_1 &= V(T) \cup \{u_0\} \\
U_2 &= V(T) \cup \{u_0, u_1\} \quad (V(\Delta) = \{u_1, u_2, u_3\}) \\
U_3 &= V(T) \cup \{u_0, u_1, u_3\} \quad (V(CP) = \{u_1, u_2, u_3, u_4, u_5\})
\end{aligned}$$

We have that $|U_1| = |U_2| = |U_3| = p + 2$. Thus, $\delta(U_1)$, $\delta(U_2)$ and $\delta(U_3)$ are not equicuts and $x^{\delta(U_1)}(E) = x^{\delta(U_2)}(E) = x^{\delta(U_3)}(E) = p(p+1) - 2$. Moreover, $\omega(T, u_0)x^{\delta(U_1)} = 0$ and $\omega(T, \Delta, u_0)x^{\delta(U_2)} = \omega(T, CP, u_0)x^{\delta(U_3)} = 2$. Thus, $x^{\delta(U_1)}$ is a root of (8), $x^{\delta(U_2)}$ is a root of (9) and $x^{\delta(U_3)}$ is a root of (10). The proof is complete. \square

The inequalities (9) and (10) give some new facets of the cut polytope but inequality (8) corresponds, via the switching operation (Barahona and Mahjoub, 1986), to a facet introduced by Boros and Hammer (1993).

Note that, when we pass from inequality (5) to inequality (6) and from inequality (6) to inequality (7), the size of the suspended tree component in the support graph decreases by one. We would like to know if we can still reduce the size of this component to get a facet defining inequality for the equipartition polytope and, if so, what is the other component of the support graph. It is natural to conjecture that this second component is nothing but \mathcal{K}_7 with some weights suitably chosen. We now examine this idea and to keep the analogy with the \mathcal{K}_3 and \mathcal{K}_5 components in inequalities (6) and (7) respectively, we assume that the 21 edges of \mathcal{K}_7 are partitioned into three cycles in which the edges have equal weights. These cycles are shown in Figure 2.7 below.

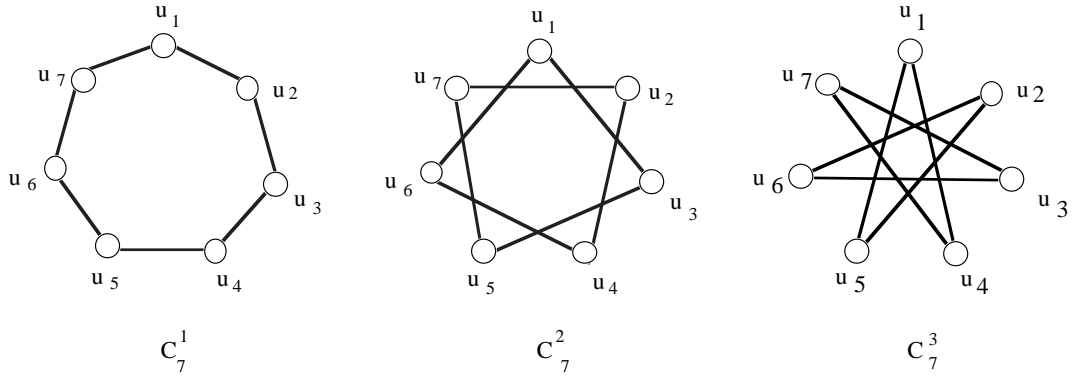


Figure 2.7:

Suppose that: T is a tree on $p - 2$ nodes in \mathcal{K}_{2p+1} , \mathcal{K}_7 is a complete subgraph of \mathcal{K}_{2p+1} node disjoint from T , (C_7^1, C_7^2, C_7^3) is a partition of the edges of \mathcal{K}_7 into three edge disjoint cycles (see Figure 2.7) and u_0 is a node in $V(\mathcal{K}_{2p+1}) \setminus V(\mathcal{K}_7 \cup T)$. We will look for weights α, β and γ such that the inequality

$$\omega(T, u_0)x \geq \sum_{e \in C_7^1} \alpha_e x_e - \sum_{e \in C_7^2} \beta_e x_e - \sum_{e \in C_7^3} \gamma_e x_e = W\mathcal{K}_7(\alpha, -\beta, -\gamma)x$$

is, hopefully, facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

To have simpler proofs, we consider the case where the suspended tree component is a cycle on $p - 1$ nodes, that is, when the tree T reduces to a path on $p - 2$ nodes. So, the previous inequality reduces to

$$x(C_{p-1}) \geq W\mathcal{K}_7(\alpha, -\beta, -\gamma)x \quad (11)$$

where C_{p-1} denotes a cycle on $p - 1$ nodes. We first find values α, β and γ for which (11) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

Let $\delta(U)$ be an equicut of \mathcal{K}_{2p+1} . If $W\mathcal{K}_7(\alpha, -\beta, -\gamma)x$ is nonpositive, then (11) is trivially valid. On the other hand, $x(C_{p-1})$ is positive if and only if not all nodes of C_{p-1} are in U (or $V \setminus U$) and this implies that U and $V \setminus U$ both contain at least three nodes not in C_{p-1} . Thus α, β and γ must be chosen in such a way that $W\mathcal{K}_7(\alpha, -\beta, -\gamma)x$ is always less than or equal to 2 and is nonpositive whenever the nodes of \mathcal{K}_7 are not equipartitioned between U and $V \setminus U$.

Figure 2.8 shows all possible configurations of partitions of \mathcal{K}_7 and the inequalities that must be satisfied by α, β and γ according to what is stated in the previous paragraph.

The analogy with inequalities (6) and (7) suggests us to set the coefficients of edges in one of the cycles of \mathcal{K}_7 to 1. Therefore, if α takes the value 1, then the validity of (11) implies that β and γ are such that:

$$(\beta, \gamma) \in S_{\beta\gamma} = \{(\beta, \gamma) \in \mathbb{R}^2 : \begin{array}{rcl} \beta & + & \gamma \geq 1 \\ \beta & + & 2\gamma \geq 2 \\ 2\beta & + & \gamma \geq 2 \\ 2\beta & + & 2\gamma \geq 1 \\ 2\beta & + & 3\gamma \geq 0 \\ 3\beta & + & \gamma \geq 1 \end{array} \}$$

The interesting values of β and γ are those corresponding to the extreme points of the polyhedron $S_{\beta\gamma}$, since they correspond to tight solutions for inequality (11).

Figure 2.8:

Consider the extreme point of $S_{\beta\gamma}$ given by $\beta = \gamma = \frac{2}{3}$. For this point, inequality (11) can be written as:

$$x(C_{p-1}) \geq x(C_7^1) - \frac{2}{3}x(C_7^2) - \frac{2}{3}x(C_7^3)$$

or equivalently,

$$3x(C_{p-1}) \geq 3x(C_7^1) - 2x(C_7^2 \cup C_7^3) \quad (12)$$

If $\delta(U)$ is an equicut whose incidence vector satisfies (12) at equality, one of the three conditions below holds:

- (I) $V(C_{p-1}) \subset U$ and $V(\mathcal{K}_7) = \{u_1, \dots, u_7\} \subset V \setminus U$;
- (II) $U = V(C_{p-1}) \cup \{u_i, u_j\}$ where u_i and u_j form one of the following pairs of nodes in \mathcal{K}_7 (see Figure 2.8): (u_1, u_3) , (u_3, u_5) , (u_5, u_7) , (u_2, u_7) , (u_2, u_4) , (u_4, u_6) , (u_1, u_6) , (u_1, u_4) , (u_4, u_7) , (u_3, u_7) , (u_3, u_6) , (u_2, u_6) , (u_2, u_5) or (u_1, u_5) ;
- (III) Let (V_1, V_2) be a partition of the nodes in C_{p-1} such that all nodes in V_1 (and V_2) appear consecutively in the cycle C_{p-1} . Then U is given by:

$$U = V_1 \cup \{u_i, u_j, u_k\}$$

where u_i, u_j and u_k form one of the following triples of nodes in \mathcal{K}_7 (see Figure 2.8): (u_1, u_3, u_5) , (u_2, u_4, u_6) , (u_3, u_5, u_7) , (u_4, u_6, u_1) , (u_5, u_7, u_2) , (u_6, u_1, u_3) or (u_7, u_2, u_4) ;

We have then the following theorem.

Theorem 2.22 *Let C_{p-1} be a cycle of \mathcal{K}_{2p+1} with $|V(C_{p-1})| = p - 1$ and \mathcal{K}_7 a complete subgraph of \mathcal{K}_{2p+1} node disjoint from C_{p-1} . Then, for $p \geq 5$, inequality (12) is facet defining for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.*

Proof: Validity of (12) comes from our construction. The usual technique is used to prove it to be facet defining, i.e., if $F = \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : \pi x = \pi_0\}$, $\pi x \geq \pi_0$ is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ and $F' = \{x \in \mathcal{P}_{EC}(\mathcal{K}_{2p+1}) : x \text{ satisfies (12) at equality}\}$ and $F' \subseteq F$, then

we have to prove that $(\pi x \geq \pi_0)$ is a linear combination of (12) and $x(E) = p(p+1)$. For this we apply Lemma 2.6. The equicuts obtained by means of the sets S_1, S_2, T_1 and T_2 that we give along the proof can be easily seen to satisfy one of the three conditions of the paragraph preceding the theorem to be a root of (12).

(a) : Let $v \in V(C_{p-1})$ and $w, z \in V_0 = V \setminus V(C_{p-1} \cup \mathcal{K}_7)$, $w \neq z$. Choose one direction to traverse C_{p-1} starting from node v . Define T'_1 to be the set of $\lceil \frac{p-2}{2} \rceil$ first nodes visited after v and T'_2 to be the set of $\lfloor \frac{p-2}{2} \rfloor$ last nodes visited before reaching node v again. Now, setting $S_1 = \{v\}$, $S_2 = \{z\}$, $T_1 = T'_1 \cup \{u_1, u_3, u_5\} \cup V_{01}$ and $T_2 = T'_2 \cup \{u_2, u_4, u_6, u_7\} \cup V_{02}$ where (V_{01}, V_{02}) is a partition of $V_0 \setminus \{v, z\}$ and V_{01} and V_{02} have appropriate sizes that makes $|T_1| = |T_2|$, Lemma 2.6 implies that:

$$\pi_{wv} = \pi_{wz} = \gamma \quad \forall w, z \in V_0, v \in V(C_{p-1})$$

(b) : Let $w \in V(C_{p-1})$ and $v \in V_0$. We construct the sets T'_1 and T'_2 as we did in (a) but using w as the reference node (instead of v). Define: $S_1 = \{u_1\}$, $S_2 = \{v\}$, $T_1 = T'_1 \cup \{u_2, u_4, u_6\} \cup V_{01}$ and $T_2 = T'_2 \cup \{u_3, u_5, u_7\} \cup V_{02}$ where (V_{01}, V_{02}) is a partition of $V_0 \setminus \{v\}$ and are chosen such that $|T_1| = |T_2|$. From Lemma 2.6, we obtain:

$$\pi_{wv} = \pi_{wu_1} = \gamma$$

and by symmetry:

$$\pi_{wv} = \pi_{wu_i} = \gamma \quad \forall v \in V_0, \forall w \in V(C_{p-1}), \forall i \in \{1, \dots, 7\}$$

Changing the roles played by nodes w and u_i , we have:

$$\pi_{u_i v} = \pi_{wu_i} = \gamma \quad \forall v \in V_0, \forall w \in V(C_{p-1}), \forall i \in \{1, \dots, 7\}$$

(c) : Let $w, u \in V(C_{p-1})$, (w, u) be a chord of C_{p-1} and $v \in V_0$. Suppose we traverse the cycle C_{p-1} starting at node w . We define T'_1 to be the set of nodes visited before u (excluding w) and T'_2 to be the set of nodes visited after node u . Assume that $|T'_1| \geq |T'_2|$

(if this is not the case, inverse the roles played by u and w). If $S_1 = \{u\}$, $S_2 = \{v\}$ and the sets T_1 and T_2 are built exactly as in (a) with T'_1 and T'_2 defined as above, we get that:

$$\pi_{wu} = \pi_{wv} = \gamma \quad \forall (w, u) \text{ chord of } C_{p-1}$$

Up to now, we have proved that the coefficients of all edges not belonging to the support graph of (12) are equal to a constant γ .

(d) : Let $w, u, v \in V(C_{p-1})$ and (w, u) and (w, v) be two consecutive edges of C_{p-1} . Suppose that we traverse the cycle starting from u in the direction such that w is the last node visited. Define T'_1 (T'_2) to be the first $\lceil \frac{p-4}{2} \rceil$ (last $\lfloor \frac{p-4}{2} \rfloor$) nodes visited. With S_1 given by $\{u\}$, S_2 by $\{v\}$ and T_1 and T_2 defined as in (a) (for T'_1 and T'_2 as above), we have:

$$\pi_{wu} = \pi_{wv} = \alpha \quad \forall (w, u) \in C_{p-1} \text{ (using cyclic arguments)}$$

(e) : Suppose that $(u, v) \in C_{p-1}$. Define: $w = u_i$, $S_1 = \{u_{i+1}, u_{i+3}, u_{i+5}\}$, $S_2 = \{u_{i+2}, u_{i+4}, u_{i+6}\}$, $T_1 = V(C_{p-1}) \setminus \{u, v\}$ and $T_2 = \{u, v\} \cup V_0$ (where the indices for the u_j 's are taken modulo 7). By Lemma 2.6 we have:

$$\pi_{u_i u_{i+1}} + \pi_{u_i u_{i+3}} + \pi_{u_i u_{i+5}} = \pi_{u_i u_{i+2}} + \pi_{u_i u_{i+4}} + \pi_{u_i u_{i+6}} \quad \forall i \in \{1, \dots, 7\}$$

(f) : Consider the set $U^i \subseteq V$ given by:

$$U^i = V(C_{p-1}) \cup \{u_i, u_{i+2}\} \quad \forall i \in \{1, \dots, 7\}$$

The incidence vectors of $\delta(U^i)$ and $\delta(U^i \cup \{u_{i+4}\})$ are both roots of (12). Using the results from (a) to (d) we have that:

$$\pi_{u_i u_{i+4}} + \pi_{u_{i+2} u_{i+4}} + 4\gamma = \sum_{j=1, j \neq 2, 4}^6 (\pi_{u_{i+4} u_{i+j}}) + 2\alpha$$

Comparing the equation obtained in (e) for a given $i \in \{1, \dots, 7\}$ and the one obtained in (f) for $i - 4$, we get:

$$\pi_{u_i u_{i+1}} = 2\gamma - \alpha$$

So, the coefficients of all edges in C_7^1 (Figure 2.8) are given by a constant $\alpha_1 = 2\gamma - \alpha$.

(g) : Let U^i be given by $V(C_{p-1}) \cup \{u_i\}$ for all $i \in \{1, \dots, 7\}$. The two following equicuts correspond to roots of (12): $\delta(U^i \cup \{u_{i+2}\})$ and $\delta(U^i \cup \{u_{i+4}\})$. Thus, for all $i \in \{1, \dots, 7\}$, we conclude that:

$$\pi_{u_i u_{i+4}} + \pi_{u_{i+2} u_{i+6}} + \pi_{u_{i+2} u_{i+5}} = \pi_{u_i u_{i+2}} + \pi_{u_{i+1} u_{i+4}} + \pi_{u_{i+4} u_{i+6}}$$

Comparing this result with the equality obtained in (e) for $i+4$ and using the fact that $\pi_{u_i, u_{i+1}} = \alpha_1$ for all $i \in \{1, \dots, 7\}$, yields:

$$\pi_{u_i u_{i+4}} + \pi_{u_{i+2} u_{i+6}} + \pi_{u_{i+2} u_{i+5}} = \pi_{u_i u_{i+2}} + \pi_{u_{i+4} u_i} + \pi_{u_{i+4} u_{i+2}}$$

or

$$\pi_{u_{i+2} u_{i+6}} + \pi_{u_{i+2} u_{i+5}} = \pi_{u_i u_{i+2}} + \pi_{u_{i+4} u_{i+2}}$$

Since the indices are taken modulo 7, the equation above can be rewritten as:

$$\pi_{u_i u_{i+4}} + \pi_{u_i u_{i+3}} = \pi_{u_i u_{i+5}} + \pi_{u_i u_{i+2}}$$

which, when compared to the equation in (e), gives:

$$\pi_{u_i u_{i+3}} = \pi_{u_i u_{i+2}} = \alpha_2$$

This argument applied repeatedly leads to the conclusion that $\pi_e = \alpha_2$ for all edges in $C_7^2 \cup C_7^3$ (see Figure 2.8).

Consider now the two following equicuts that are roots of (12): $\delta(V(C_{p-1}) \cup \{v\})$ and $\delta(V(C_{p-1}) \cup \{u_1, u_3\})$ where $v \in V_0$. Comparing the values of πx for the incidence vectors of these equicuts, we obtain the following relation between α_1, α_2 and γ :

$$2\alpha_1 + 3\alpha_2 = 5\gamma$$

Thus, if $\lambda_1 = \frac{\alpha_2 - \gamma}{2}$ and $\lambda_2 = \gamma$:

$$(\pi x \geq \pi_0) = \lambda_1((12)) + \lambda_2(x(E) = p(p+1))$$

□

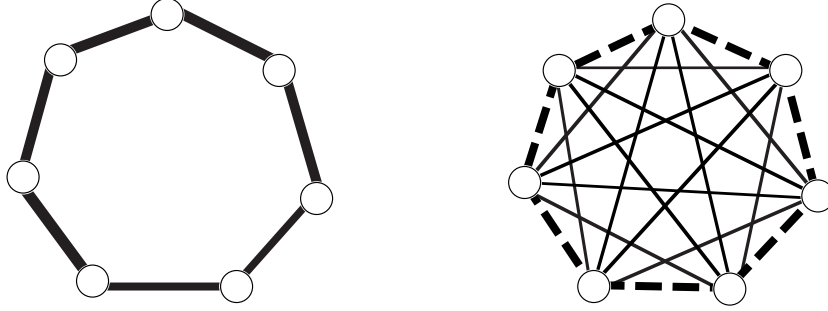


Figure 2.9:

In Figure 2.9, we show the support of an inequality given as in (11) (or equivalently, in (12)).

The left-hand side of inequality (12) has a cycle on $p - 1$ nodes as support graph. In inequalities (6) and (7) the cycles were replaced by suspended trees. Also, in inequality (12), a more general inequality can probably be obtained in that way and this suggests us that there exists a class of facet defining inequalities for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ which support graphs have two connected components. The first is a suspended tree T on $p - q$ nodes with apex node u_0 and the second component is a complete graph on $2q + 1$ nodes. The values of the coefficients corresponding to edges in the suspended tree component are computed in the usual way, while those of edges in the component \mathcal{K}_{2q+1} have a much more involved computation.

2.6 Extensions of PBC and Suspended Tree Inequalities

We have seen in the previous sections that both the PBC and the suspended tree inequalities ((2) and (5)) generalize the cycle inequality (1). In this section we investigate whether the intersection between the families of PBC inequalities and suspended tree (ST) inequalities is restricted to the cycle inequality.

Recall that the PBCs are composed of a collection of cycles. Since the suspended trees are, in some sense, generalizations of cycles, let us consider that we have now a collection $\mathcal{T} = ((T_1, u_0^1), \dots, (T_r, u_0^r))$ of suspended trees. As for PBCs, we require that \mathcal{T} has the property that, if a node v is in the intersection of any pair of two suspended trees in \mathcal{T} , then v belongs to all suspended trees in the collection. Consider the following inequality:

$$\sum_{i=1}^r \omega(T_i, u_0^i)x \geq 2r \quad (13)$$

where $\omega(T_i, u_0^i)x$ is defined as in (5).

An example of a support graph for an inequality of type (13) is shown in Figure 2.10.

If we define N to be the set of nodes that are common to all suspended trees in \mathcal{T} , we have for Figure 2.10 that $N = \{2, 3, 6, 7\}$ (the nodes indicated by squares).

To study the validity of (13) with respect to $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, we split this class of inequalities into 2 subclasses. In the first, we assume that the apex node of any suspended tree in \mathcal{T} belongs to N . In the second case, we assume that there are suspended trees in \mathcal{T} with apex not in N . For the first case, we can establish necessary and sufficient conditions for validity while for the second case we only have a necessary condition for validity.

Consider the first case where $u_0^i \in N$ for all $1 \leq i \leq r$. Removing all nodes in N from the structure, we get a forest formed by connected subtrees of the trees T_1, \dots, T_r . Let $((V(A_1), A_1), (V(A_2), A_2), \dots)$ be the ordered sequence of those subtrees such that $|V(A_1)| \geq |V(A_2)| \geq \dots$. Moreover, define: $\overline{Q} = \sum_{i=1}^{r-1} |V(A_i)|$, $V_C = \bigcup_{i=1}^r V(T_i, u_0^i)$, $V_0 = V \setminus V_C$ and $n_0 = |V_0|$.

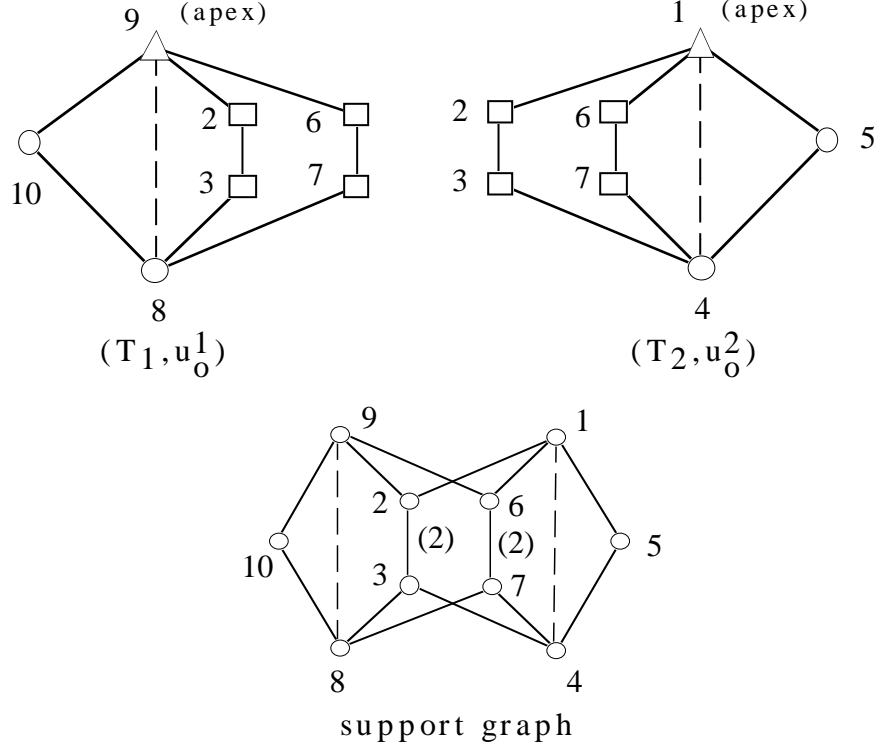


Figure 2.10:

Theorem 2.23 *If $u_0^i \in N$ for all $1 \leq i \leq r$, then inequality (13) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$ if and only if $\overline{Q} + n_0 \leq p - 1$.*

Proof: Similar to that of Theorem 2.7.

Necessity: By contradiction, assume that $\overline{Q} + n_0 \geq p$. Then, if we take $U = (\bigcup_{i=1}^{r-1} V(A_i)) \cup V'_0$ where $V'_0 \subseteq V_0$ is chosen such as to obtain $|U| = p$, then Lemma 2.14 can be used to show that $\sum_{i=1}^r \omega(T_i, u_0^i)x = 2(r-1)$ and we are done.

Sufficiency: As in Theorem 2.7, if validity is not satisfied, then the equicut $\delta(U)$ corresponding to the violation must satisfy $w(T_j, u_0^j)x^{\delta(U)} = 0$ for some $j \in \{1, \dots, r\}$. This implies that all nodes in N are in a same shore, say $V \setminus U$, of that equicut. To have a violation of (13) there must be at most $r - 1$ subtrees in $\{A_1, A_2, \dots\}$ with nodes in U (since all nodes u_0^i are in $V \setminus U$ and, from Lemma 2.14, each subtree with nodes in U contributes

with 2 unities to the left-hand side of (13)). Even if we take the nodes in the $r - 1$ largest subtrees and those of V_0 to be in U , we get: $|U| = \overline{Q} + n_0 \leq p - 1$. Thus, $(U, V \setminus U)$ does not form an equipartition and sufficiency is proved. \square

Theorem 2.23 does not hold when there exists one suspended tree in \mathcal{T} with the apex node not in N . To see this, consider the example shown in Figure 2.11.

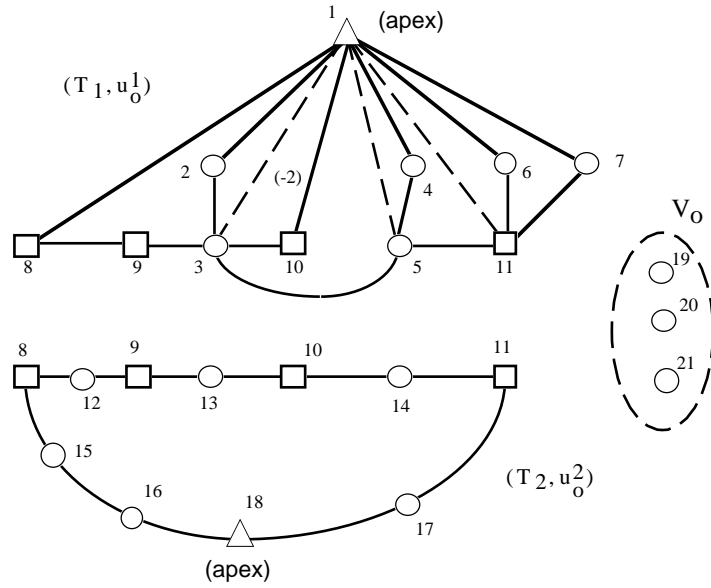


Figure 2.11:

In this example we have: $\overline{Q} = 7$ (corresponding to the subtree with nodes $1, \dots, 7$), $r = 2$, $N = \{8, 9, 10, 11\}$, $n_0 = 3$ and $p = 10$. Note that the apex of both trees (nodes 1 and 18) are not in N . Moreover, $\overline{Q} + n_0 = 10 = p$. However, the inequality $\sum_{i=1}^2 \omega(T_i, u_0^i) x \geq 4$ can be easily checked to be valid for $P_{EC}(\mathcal{K}_{21})$.

If there exists $j \in \{1, \dots, r\}$ such that $u_0^j \notin N$, we can redefine the subtrees $(V(A_i), A_i)$ by considering that they are obtained by removing all nodes of $N \cup \{u_0^1, \dots, u_0^r\}$ from the structure. Keeping the definitions of \overline{Q} , V_C , V_0 and n_0 as before, we have the following result.

Theorem 2.24 Suppose that there exists $j \in \{1, \dots, r\}$ such that $w_0^j \notin N$. If inequality (13) is valid for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$, then $\overline{Q} + n_0 \leq p - 1$.

Proof: same as in Theorem 2.23. □

However, $\overline{Q} + n_0 \leq p - 1$ is not a sufficient condition for the validity of (13) when there exists $j \in \{1, \dots, r\}$ with $w_0^j \notin N$. This can be seen in Figure 2.12, where it is shown the support graph of an inequality as in (13) which is not valid for $\mathcal{P}_{EC}(\mathcal{K}_{19})$. In this case, we have $\overline{Q} = 5$ (subtree with nodes 2, 3, 4, 10, 11), $n_0 = 3$, $p = 9$ and the equicut $\delta(1, \dots, 9)$ violates the inequality.

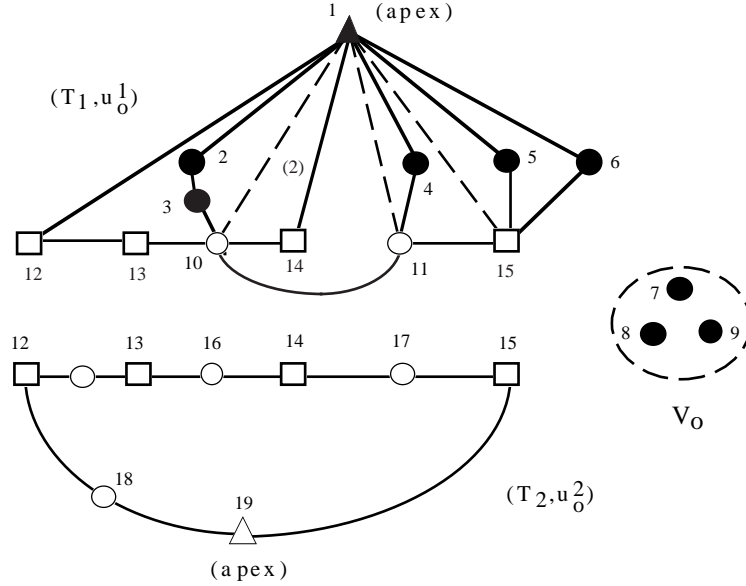


Figure 2.12:

We did not investigate the properties of the suspended tree combination (STC) inequalities any further. Nevertheless, the result given below indicates that the study of STC inequalities is perhaps a promising way to find new facets for $\mathcal{P}_{EC}(\mathcal{K}_{2p+1})$.

Theorem 2.25 *Consider the inequality below corresponding to the support graph shown in Figure 2.10:*

$$\begin{aligned}
& x_{1,5} + x_{4,5} + x_{4,7} + x_{1,6} + x_{3,4} + x_{1,2} + x_{9,10} + \\
& x_{8,10} + x_{2,9} + x_{3,8} + x_{7,8} + x_{6,9} + 2x_{2,3} + 2x_{6,7} - \\
& x_{8,9} - x_{1,4} \geq 4
\end{aligned}$$

This inequality is facet defining for $P_{EC}(\mathcal{K}_{13})$.

Another possibility that may be exploited is the use of PBCs combined with complete odd subgraphs as in inequalities (6), (7) and (12). Two results indicating that this can be a direction for investigation are given below.

Theorem 2.26 (i) *The PBC + triangle inequality corresponding to the support graph of Figure 2.13(a), i.e.,*

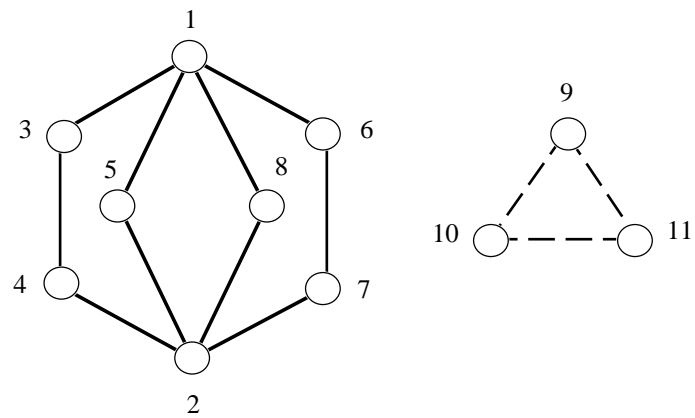
$$\begin{aligned}
& (x_{1,3} + x_{3,4} + x_{2,4} + x_{1,6} + x_{6,7} + x_{2,7} + x_{1,5} \\
& + x_{2,5} + x_{1,8} + x_{2,8}) - (x_{9,10} + x_{10,11} + x_{9,11}) \geq 2
\end{aligned}$$

is facet defining for $P_{EC}(\mathcal{K}_{11})$.

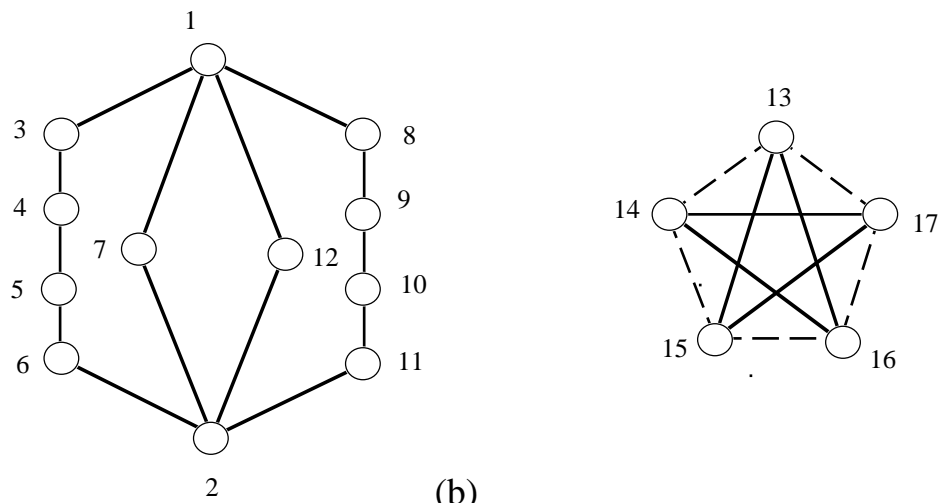
(ii) *The PBC + chorded-pentagon inequality corresponding to the support graph of Figure 2.13(b), i.e.,*

$$\begin{aligned}
& (x_{1,3} + x_{3,4} + x_{4,5} + x_{5,6} + x_{2,6} + x_{1,8} + x_{8,9} + \\
& x_{9,10} + x_{10,11} + x_{2,11} + x_{1,7} + x_{2,7} + x_{1,12} + x_{1,12}) - \\
& (x_{13,14} + x_{14,15} + x_{15,16} + x_{16,17} + x_{17,13}) + \\
& (x_{13,15} + x_{17,15} + x_{17,14} + x_{14,16} + x_{16,13}) \geq 2
\end{aligned}$$

is facet defining for $P_{EC}(\mathcal{K}_{17})$.



(a)



(b)

Figure 2.13:

2.7 Further Remarks

We have introduced here large classes of facet defining inequalities of the equicut polytope for complete graphs. In practice, one does not always deal with complete graphs and, when this is the case, the natural question that arises is how useful it is to have results for complete graphs. Some comments on this question have been made earlier in this chapter (see Lemma 2.4). In the next paragraphs we discuss further the interest of studying classes of facet defining inequalities for complete graphs.

There are some technical reasons to work with complete graphs. Specifically the proofs concerning the dimension and facets are easier for complete graphs than for incomplete graphs, for which the proofs are often case dependent.

Another reason for studying complete graphs is the following. Consider the equipartition problem defined for a graph $G = (V, H)$ where G is a subgraph of $\mathcal{K}_n = (V, E)$ (that is, $H \subseteq E$). Define the sets X and X_E by:

$$X = \{x \in \mathbb{B}^{|H|} : x \text{ is the incidence vector of an equicut of } G\}$$

and

$$X_E = \{\tilde{x} \in \mathbb{B}^{|E|} : \tilde{x} \text{ is the incidence vector of an equicut of } \mathcal{K}_n\}$$

Note that X is the projection of X_E in the space of the variables corresponding to the edges in H . Moreover, $P_{EC}(G)$ is the convex hull of X and $P_{EC}(\mathcal{K}_n)$ is the convex hull of X_E .

If the vector $\tilde{c} \in \mathbb{R}^{|E|}$ is defined such that:

$$\tilde{c}_e = \begin{cases} c_e & \text{if } e \in H \\ 0 & \text{if } e \in E \setminus H \end{cases}$$

then, $\min\{cx : x \in X\} = \min\{\tilde{c}\tilde{x} : \tilde{x} \in X_E\}$. This means that the equipartition problem defined on any arbitrary graph G can always be transformed into an equipartition problem defined on the minimal complete graph containing G . Thus, at least theoretically, it is

enough to study the facial structure of the equipartition polytope for complete graphs, although this implies working in a higher dimensional space (with more variables).

To remain in a lower dimensional space, we have to study the polytope given by the convex hull of X (instead of X_E). In this case, we may have to be satisfied to find valid inequalities since the facet property is very dependent on the structure of the subgraph G . However, valid inequalities for $P_{EC}(G)$ can be obtained from valid inequalities for $P_{EC}(\mathcal{K}_n)$.

Consider a valid inequality $\sum_{e \in E} a_e x_e \leq a_0$ for $P_{EC}(\mathcal{K}_n)$ and let \tilde{G} be the subgraph of \mathcal{K}_n with edge set given by the support of the inequality and node set given by the nodes incident to these edges. Then $\sum_{e \in H} a_e x_e \leq a_0$ can also be proved to be valid for $P_{EC}(G)$ if \tilde{G} is subgraph of G . This means that the study of valid inequalities for the case of complete graphs is useful to obtain valid inequalities for the case of incomplete graphs. In some cases, a "weak" valid inequality for $P_{EC}(\mathcal{K}_n)$ can be a strong valid inequality for $P_{EC}(G)$ (see the discussion on tree inequalities in Chapter 4).

For more general multicut, where the facet property is difficult to prove, the argument in the previous paragraph establishing the links between valid inequalities for the complete graph problem and valid inequalities for a subgraph problem remains true. For this reason, in Chapter 3, where we tackle more general graph partitioning problems, the results are also given in terms of complete graphs.

3. Valid Inequalities for some Graph Partitioning Problems

3.1 Introduction

In this chapter we investigate the polytopes associated to graph partitioning problems. Some of these problems generalize the equipartition problem. It is natural then to try to use the same structures that led to strong valid inequalities to the equipartition problem to derive valid inequalities for these more general problems.

The graph partitioning problems considered in this chapter are defined as follows. We are given an integer K and a graph $G = (V, E)$ with weights w_i associated to each node i in V . A partition of G is feasible if the nodes in V are divided into K clusters such that the sum of the weights of the nodes in each cluster k , for $k = 1, \dots, K$, does not exceed the cluster capacity F_k . There are costs c_{ij} associated to every edge (i, j) in E and costs h_{ik} associated to every node i and every cluster k . The cost of a feasible partition is then obtained by adding two terms. The first is computed by the sum of the h_{ik} for all pairs (i, k) such that node i is in cluster k . The second is the sum of the c_{ij} such that nodes i and j are in different clusters (or in the same cluster). The goal is to find a feasible partition of V that minimizes (or maximizes) the cost function.

The equipartition problem of Chapter 2 is obtained from the general problem above by fixing the number of clusters to two, the node costs to zero, the node weights are 1 and the cluster capacities to $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$ respectively.

In Johnson et al. (1991), the authors propose an algorithm based on column generation for a graph partitioning problem. The subproblem to generate additional columns is a single cluster problem in which we are asked to find a subset of nodes in the graph whose sum of weights is bounded by the cluster capacity $F_k = W$ and such that it minimizes a linear cost function. The model used is a node-edge model slightly different from the one we have

discussed in Section 2.2 since the authors are interested in characterizing the edges that are within the cluster instead of those that are in the cutset.

The polytope corresponding to the feasible solutions of the single cluster problem has been studied in the latter paper. One of the inequalities introduced there has a tree as its support. In the next section, we show that this tree inequality can be generalized by summing nonvalid tree inequalities appropriately so as to generate a strong valid inequality. The support of this new inequality is the union of the individual supports of the tree inequalities that participate in the summation. This idea is similar to the one used in Chapter 2 to obtain the PBC inequality from nonvalid cycle inequalities.

The results in Sections 3.3 and 3.4 have been obtained in a joint research project with R. Weismantel, A. Martin, C. Ferreira and L. Wolsey.

In Section 3.3, we consider the (multicut) polytope corresponding to the convex hull of (K, F) multicuts of a graph G . A (K, F) multicut of a graph G is a multicut that partitions G into K clusters of size at most F . In this case, the cluster capacity constraints are simply cardinality constraints. Results concerning the dimension of the polytope are given. The arguments used to show the validity of inequalities for the equicut polytope are extended to the case of multicuts and permit us to derive inequalities that define facets for the multicut polytope. We also discuss the possibility of obtaining strong inequalities by lifting valid inequalities whose support graphs are cycles covering multiple clusters.

Section 3.4 is devoted to the study of valid inequalities for the convex hull of the incidence vectors of (K, W) multicuts of a graph G . A (K, W) multicut of a graph G is a multicut that partitions the graph into K clusters within which the sum of the node weights is bounded by W . The node weights are assumed to take arbitrary positive values and, in general, we are not able to compute the dimension of the polytope. However, we give valid inequalities that essentially have the same supports as the inequalities for the polytope of Section 3.3. An additional inequality is introduced which we call the knapsack tree inequality. We shall see

in Chapter 4 that these inequalities have been crucial for solving some test problems in our computational experiments.

In Sections 3.3 and 3.4, we have used the edge model discussed in Section 2.2. The results are given in terms of complete graphs and can be extended to incomplete graphs as we have pointed out in Section 2.7.

3.2 The Single Cluster Problem

Consider the graph partitioning problem as defined in the previous section and suppose that a cluster k of the partition is fixed. In the single cluster problem, we look for a node set that satisfies the capacity (knapsack) constraint of cluster k and that optimizes a linear cost function. An Integer Programming formulation for this problem is given below.

Given a graph $G = (V, E)$, for all $ij \in E$ let the edge variable z_{ij} be 1 if nodes i and j are both in the cluster and 0 otherwise. For all $i \in V$, define the node variable y_i which takes the value 1 if i is in the cluster and 0 otherwise. Let B be the capacity of the fixed cluster k and w_i the weight of node i for each $i \in V$. The single cluster problem is formulated as follows:

$$\begin{aligned}
& \max && \sum_{i \in V} h_i y_i + \sum_{e \in E} c_e z_e \\
& \text{Subject to} && z_{ij} \leq y_i, \quad z_{ij} \leq y_j \quad \forall (i, j) \in E \quad (1) \\
& && y_i + y_j - z_{ij} \leq 1 \quad \forall (i, j) \in E \quad (2) \\
& && \sum_{u \in V} w_u y_u \leq B \quad (3) \\
& && y_i \in \{0, 1\} \quad \forall i \in V \\
& && z_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \quad (4)
\end{aligned}$$

The $(0, 1)$ integer solutions of the system above are the incidence vectors of the feasible solutions to the single cluster problem. This model is very similar to the node-edge model discussed in Section 2.2. Note however that the edge variables here indicate which edges are within the cluster and not those that are in the cutset.

Denote by P_{CL} (single cluster polytope) the convex hull of the integer points satisfying inequalities (1) to (4) and by P_{Kn} the convex hull of the $(0, 1)$ integer points describing the knapsack polytope defined by inequality (3).

The single cluster polytope P_{CL} is studied in Johnson et al. (1991). It appears in the optimization of the subproblems in a column generation algorithm for a graph partitioning problem originally used to describe a compiler design problem (see reference for details). The

authors proposed to solve this problem by a cutting plane algorithm and introduced some strong valid inequalities that we have generalized

Before presenting the inequalities given in Johnson et al. (1991) a few more definitions are introduced.

A subset S of V is said to be independent for P_{Kn} if $\sum_{i \in S} w_i \leq B$ (in this case y^S , the incidence vector of S , is in P_{Kn}); otherwise ($y^S \notin P_{Kn}$) S is said to be dependent or to be a *cover*. A dependent set (or a *cover*) is minimal if all its subsets are independent.

Theorem 3.1 (*Johnson et al., 1991*)

Let S be a dependent set for P_{Kn} and let T be a tree spanning the nodes in S . For each $i \in S$, let d_i be the degree of node i in T . Then, the tree inequality

$$\omega(T)(z, y) = \sum_{e \in T} z_e - \sum_{i \in S} (d_i - 1)y_i \leq 0 \quad (5)$$

is valid for $P_{CL}(G)$.

When $S \setminus \{i\}$ is an independent set for every end node i of T , inequality (5) defines a facet of the polytope $P_{CL}(G_S)$, where $G_S = (S, E(S))$ is the subgraph induced by S in G .

An immediate corollary is obtained when the tree in inequality (5) is a star.

Corollary 3.2 (*Johnson et al., 1991*)

Let S be a dependent set for P_{Kn} and T the set of edges of a star spanning S and centered at a node i . Then, the star inequality

$$\sum_{e \in T} z_e - (|S| - 2)y_i \leq 0 \quad (6)$$

is valid for $P_{CL}(G)$.

The next lemma ensures the validity of inequality (5) and characterizes the points that lie on the face it defines on P_{CL} . We point out to the similarities between this lemma and Lemma 2.14 of Section 2.5.

Lemma 3.3 *Let $G = (V, E)$ be a graph, T a tree in G and $\omega(T)(z, y) \leq 0$ its corresponding inequality. Given a feasible solution (z, y) of P_{CL} , let A be the subset of nodes in V defined as: $A = \{u \in V(T) : y_u = 1\}$. If $c(A)$ is the number of connected components induced by A in T and $\delta_T(A)$ is the cutset of A in $(V(T), T)$, then $\omega(T)(z, y) = c(A) - |\delta_T(A)|$.*

Proof: We compute:

$$\begin{aligned} \omega(T)(z, y) &= \sum_{(ij) \in T} z_{ij} + \sum_{u \in V(T)} (1 - d_u) y_u \\ &= \sum_{i, j \in A} z_{ij} + \sum_{u \in A} (1 - d_u) \\ &= \sum_{i, j \in A} z_{ij} + |A| - \sum_{u \in A} d_u \end{aligned}$$

The first summation corresponds to the number of edges in the forest induced by A in T , therefore: $\sum_{i, j \in A} z_{ij} = |A| - c(A)$. In the second summation each edge joining two nodes i and j in A is counted twice (in d_i and d_j) and the edges going from A to $V(T) \setminus A$ (those in $\delta_T(A)$) appear only once. This yields: $\sum_{u \in A} d_u = 2(|A| - c(A)) + |\delta_T(A)|$.

Therefore:

$$\begin{aligned} \omega(T)(z, y) &= |A| - c(A) - 2|A| + 2c(A) - |\delta_T(A)| \\ &= c(A) - |\delta_T(A)| \end{aligned}$$

□

From the lemma above, if $S \subseteq V$ is a feasible cluster in G and $A = S \cap V(T)$, the incidence vector of S is a root of (5) if $c(A) = |\delta_T(A)|$. This means that each of the subtrees in the forest induced by A in T can be disconnected from the remaining nodes of T by removing a single edge. Moreover, the nodes in $V(T) \setminus A$ are all contained in a single subtree. The tree obtained by contracting the nodes of A that are in a same subtree and by contracting the nodes of $(V(T) \setminus A)$ is a star (see Figure 3.1).

Clearly, $c(A)$ is always less than or equal to $|\delta_T(A)|$ except if $A = V(T)$ in which case $c(A) = 1$ and $|\delta_T(A)| = 0$. Therefore, inequality (5) is valid for P_{CL} if and only if $\sum_{u \in V(T)} w_u > B$, that is, $V(T)$ is a dependent set for P_{Kn} . A similar argument has been used in Chapter

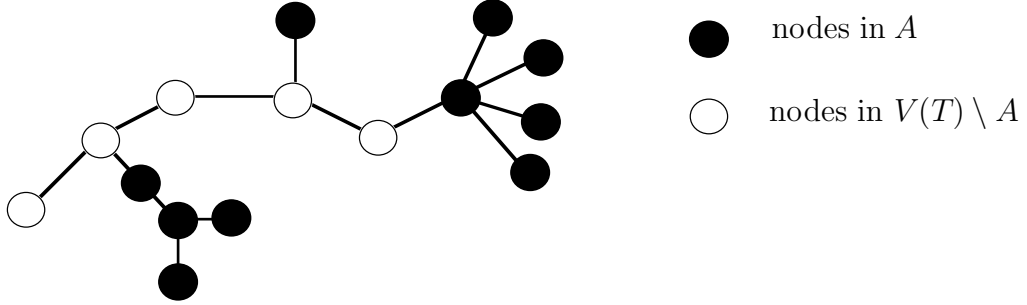


Figure 3.1:

2 to prove that the cycle inequality is valid for the equipartition polytope. When suitable conditions are satisfied, the sum of nonvalid cycle inequalities also gives rise to strong valid inequalities for the equipartition polytope, namely the PBC inequalities. We now apply the same idea to find inequalities for P_{CL} that come from the sum of nonvalid tree inequalities.

Let Υ be a collection of trees $((V(T_1), T_1), \dots, (V(T_t), T_t))$ of a graph G satisfying the following property: if $u \in V(T_i) \cap V(T_j)$ for some $i, j \in \{1, \dots, t\}$, then $u \in V(T_k) \cap V(T_\ell)$ for all $k, \ell \in \{1, \dots, t\}$.

For each tree T_i of Υ , we can define the tree inequality as before, i.e., $\omega(T_i)(z, y) \leq 0$. The tree combination inequality for Υ is given by the sum of these inequalities, that is:

$$\omega(\Upsilon)(z, y) = \sum_{T_i \in \Upsilon} \omega(T_i)(z, y) \leq 0 \quad (7)$$

Consider the subgraph of G formed by the edges and nodes in Υ . Let N be defined as the set of nodes that are common to all trees in Υ . If the nodes in N are removed from this subgraph, what remains is a forest of subtrees. Let (S_1, S_2, \dots) be the ordered set of these subtrees where $q_1 \geq q_2 \geq \dots$ and $q_i = \sum_{u \in S_i} w_u$. We define $\overline{Q} = \sum_{i=1}^{t-1} q_i$. In Figure 3.2 it is shown the support of a tree combination inequality for which $t = 3$ and $\overline{Q} = 6$.

Theorem 3.4 *Let Υ be a tree combination in $G = (V, E)$ such that $V(\Upsilon) = \cup_{i=1}^t V(T_i)$. Suppose that $\sum_{u \in V(\Upsilon)} w_u = B + W$, where $W \geq 0$. Then, inequality (7) is valid for $P_{CL}(G)$ if and only if $\overline{Q} < W$.*

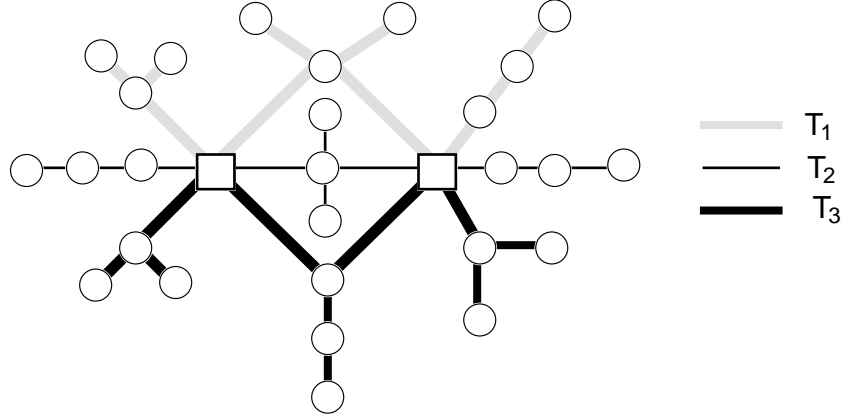


Figure 3.2:

Proof: Necessity: Suppose that $\overline{Q} \geq W$. This implies that there exists a set Λ of $\lambda \leq t - 1$ subtrees such that the sum of the node weights for all subtrees in Λ is greater than or equal to W .

Define S to be the set of all nodes of $V(\Upsilon)$ that are not in $V(\Lambda)$ ($= \{ \text{nodes of subtrees in } \Lambda \}$). The incidence vector (z^S, y^S) of S can be easily checked to be in $P_{CL}(G)$.

From Lemma 3.3, $\omega(T_i)(z^S, y^S) = c(S \cap V(T_i)) - |\delta_T(S \cap V(T_i))|$. However, $|\delta_T(S \cap V(T_i))|$ can be computed as $c(S \cap V(T_i)) + c(V(T_i) \setminus S) - 1$ which yields $\omega(T_i)(z^S, y^S) = 1 - c(V(T_i) \setminus S)$. Thus,

$$\begin{aligned}
 \omega(\Upsilon)(z^S, y^S) &= \sum_{i=1}^t \omega(T_i)(z^S, y^S) \\
 &= \sum_{i=1}^t (1 - c(V(T_i) \setminus S)) \\
 &= t - \sum_{i=1}^t c(V(T_i) \setminus S) \\
 &= t - \lambda \\
 &\geq t - (t - 1) \geq 1
 \end{aligned}$$

The left-hand side of inequality (7) is positive for this feasible solution and therefore the inequality is not valid. Necessity is proved.

Sufficiency: We assume that (7) is not valid and we end up with the conclusion that \overline{Q} must be greater than or equal to W . For this, let S be a subset of V such that (z^S, y^S) violates (7), that is:

$$\omega(\Upsilon)(z^S, y^S) = t - \sum_{i=1}^t c(V(T_i) \setminus S) \geq 1$$

which implies that:

$$\sum_{i=1}^t c(V(T_i) \setminus S) \leq t - 1$$

Now, since (z^S, y^S) violates (7), there exists at least one tree in Υ , say T_j , such that $\omega(T_j)(z^S, y^S) = 1$. This implies that all nodes in $V(T_j)$ and, consequently, all nodes in N are in S . Thus, from the last expression above, we can conclude that there are at most $t - 1$ subtrees in Υ (obtained by removing the nodes in N) that do not contain nodes in S .

If $\overline{Q} < W$, the previous observation implies that $\sum_{u \in S} w_u > B$ and, therefore, S cannot be a feasible cluster. We conclude that, if (7) is not valid, then $\overline{Q} \geq W$ and this completes the proof. \square

It is hard to find necessary and sufficient conditions for inequality (7) to be facet defining for P_{CL} , specially when the nodes weights are taken arbitrarily. Nevertheless, we have found one case for which facet defining inequalities can be obtained by combining nonvalid tree inequalities. This case is described below.

Suppose that the knapsack constraint in the system defining P_{CL} is replaced by a cardinality constraint of the type $\sum_{i \in V} y_i \leq B$ (this amounts to setting all node weights to 1). Suppose that Υ is a collection of two trees T_1 and T_2 that have one node v_c in common ($N = \{v_c\}$) and the degree of any node i in trees T_1 and T_2 is not greater than 2. In this case, each of the trees of the combination reduces to a path and the support graph of the tree combination inequality looks like a cross centered at node v_c (see Figure 3.3).

Removing node v_c from the support graph, what remains is a forest composed of 4 paths. Let $(V(P_1), P_1), \dots, (V(P_4), P_4)$ be those paths, $q_1 \geq q_2 \geq q_3 \geq q_4$ ($q_i = |V(P_i)|$) and

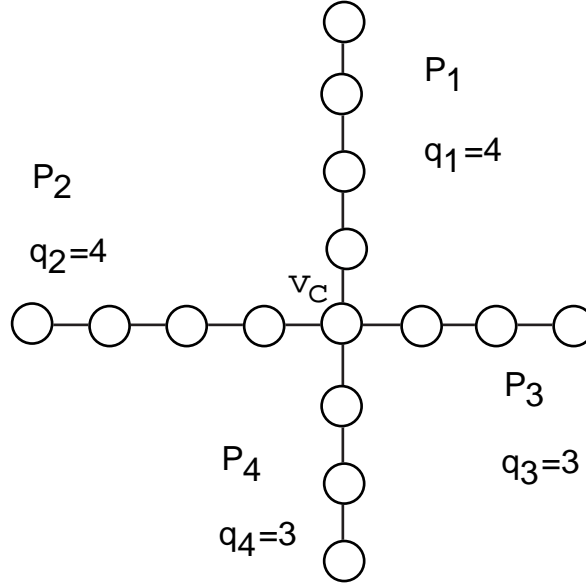


Figure 3.3:

$v_1^i, \dots, v_{q_i}^i$ denote the nodes of $V(P_i)$. Moreover, define p_j to be the minimum of $q_k + q_\ell$ for k and ℓ in $\{1, 2, 3, 4\} \setminus \{j\}$. The tree combination inequality corresponds to:

$$\sum_{e \in P_1 \cup P_2 \cup P_3 \cup P_4} z_e - 2y_{v_c} - \sum_{i=1}^4 \sum_{j=1}^{q_j-1} y_{v_j^i} \leq 0 \quad (8)$$

The following result can be proved.

Theorem 3.5 *Let Υ be cross centered at a node v_c and spanning the graph $G = (V, E)$. Suppose that $q_2 + q_3 + q_4 \geq B$ and $|V(\Upsilon)| = B + W$. Then, for $B > W \geq 3$, the following holds:*

- (i) *Inequality (8) is valid for $P_{CL}(G)$.*
- (ii) *Inequality (8) is facet defining for $P_{CL}(G)$ if and only if $q_j + p_j = B$ for all $j = 1, 2, 3, 4$.*

The following remarks concern Theorem 3.5. Since $q_2 + q_3 + q_4 \geq B$ and $q_1 + q_2 + q_3 + q_4 + 1 = B + W$, we have that $W \geq q_1 + 1$. Therefore, for $W = 1$, we must have $q_1 = q_2 = q_3 = q_4 = 0$ and there is no tree combination. On the other hand, if $W = 2$, we must have $q_1 = q_2 = q_3 = q_4 = 1$ which implies $B = 3$. In this case, the tree combination inequality is dominated by

the four possible different star inequalities (see Corollary 3.2) centered at node v_c . Note also that, when (8) is facet defining for P_{CL} , we have that $q_1 + p_1 = q_2 + p_2 = B$, $p_1 = p_2 = q_3 + q_4$ and $q_1 + q_2 + q_3 + q_4 + 1 = B + W$. This implies that $q_1 = q_2 = W - 1$.

The results in this section illustrate how to apply the ideas developed in Chapter 2 to derive valid inequalities for P_{CL} . These inequalities can define facets for P_{CL} as it is the case in Theorem 3.5. In the next section we investigate again the possibility of using some of the ideas of Chapter 2 to obtain valid and facet defining inequalities for polyhedra related to graph partitioning problems.

3.3 The Graph Partitioning Problem with Cardinality Capacity Constraints

In this section we consider the problem of partitioning the nodes of a graph $G = (V, E)$ into at most K subsets of size bounded by a constant F . Such a partition is called a (K, F) partition of V and the corresponding cutset (the set of edges joining two nodes in different clusters of the partition) is called a (K, F) multicut.

The (K, F) partition polytope denoted by $P_{MC}^{K,F}(G)$ is the convex hull of incidence vectors of (K, F) multicuts in G . Properties of the polytope $P_{MC}^{K,F}(G)$ are discussed below. We start by computing the dimension of the polytope and then we consider the possibility of using facet defining inequalities of the equipartition polytope ($P_{EC}(G)$) to derive facet defining inequalities for $P_{MC}^{K,F}(G)$. We show that inequalities having cycles, PBCs and suspended trees (see Chapter 2) as support graphs define facets for $P_{EC}(G)$ when the supports form covers for a single cluster of the partition. For $r \geq 2$, we show that when a cycle covers $r - 1$ clusters of the partition the natural extension of the usual cycle inequality given by $x(C) \geq r$ can be lifted in many different ways to generate stronger inequalities.

The dimension of $P_{MC}^{K,F}(\mathcal{K}_n)$ is given in the following result:

Theorem 3.6 *Consider the complete graph $\mathcal{K}_n = (V, E)$ with $|V| = n \geq 2$. Then,*

- (i) *If $K \geq 2$, $F \geq 2$ and $n \leq KF - 2$, then $P_{MC}^{K,F}(\mathcal{K}_n)$ is full-dimensional.*
- (ii) *If $n = KF - 1$, then $\dim(P_{MC}^{K,F}(\mathcal{K}_n)) = |E| - 1$ and $P_{MC}^{K,F}(\mathcal{K}_n) \subset \{x \in \mathbb{R}^{|E|} : x(E) = \frac{K(K-1)}{2}F^2 - (K-1)F\}$.*
- (iii) *If $n = KF$, then $\dim(P_{MC}^{K,F}(\mathcal{K}_n)) = |E| - n$ and $P_{MC}^{K,F}(\mathcal{K}_n) \subset \{x \in \mathbb{R}^{|E|} : x(\delta(v)) = (K-1)F, \forall v \in V\}$.*

Proof: (i) : Assume that $P_{MC}^{K,F}(\mathcal{K}_n) \subset \{x \in \mathbb{R}^{|E|} : \pi x = \pi_0\}$. If we prove that $\pi_e = \pi_0 = 0$ for all edges $e \in E$, we are done.

case (i.1) : $2 \leq n \leq KF - 2$, $K \geq 3$ and $F \geq 3$.

For $n = 2$ the proof is trivial. So, assume that $n \geq 3$. Let:

- $w \neq u \neq v \in V$;
- T_1 and T_2 be two disjoint node subsets in $V \setminus \{w, u, v\}$ such that $|T_1| = |T_2| = \min\{F - 2, \lfloor \frac{n-3}{2} \rfloor\}$ if $n \geq 5$ and $T_1 = T_2 = \emptyset$ otherwise.
- L_1, \dots, L_{K-2} be $K - 2$ disjoint subsets in $V \setminus (\{w, u, v\} \cup T_1 \cup T_2)$ such that $\sum_{i=1}^{K-2} |L_i| = n - 2|T_1| - 3$ and $|L_i| \leq F$ for all $i = 1, \dots, K - 2$.

One can see that the number of nodes contained in clusters L_1, \dots, L_{K-2} is $n - 2F + 1$ if $F - 2 \leq \lfloor \frac{n-3}{2} \rfloor$, and otherwise it is 1 for n even or 0 for n odd. In any case, it is clear that these nodes can be distributed among the $K - 2$ subsets without violating their capacities.

Consider the following feasible (K, F) partitions:

$$\begin{aligned} \delta_1 &= \delta(\{w, u\} \cup T_1, \{v\} \cup T_2, L_1, \dots, L_{K-2}) \\ \delta_2 &= \delta(\{w, v\} \cup T_1, \{u\} \cup T_2, L_1, \dots, L_{K-2}) \\ \delta_3 &= \delta(\{w, u\} \cup T_2, \{v\} \cup T_1, L_1, \dots, L_{K-2}) \\ \delta_4 &= \delta(\{w, v\} \cup T_2, \{u\} \cup T_1, L_1, \dots, L_{K-2}) \end{aligned}$$

Comparing the expressions obtained for πx for the four incidence vectors of the (K, F) multicuts above we get:

$$\pi_{wu} = \pi_{wv}$$

Since (u, v, w) can be any triple of nodes in V , we conclude that:

$$\pi_e = \alpha \quad \forall e \in E$$

Suppose that $n = qF + r$, where $r \leq F - 1$. If $r < F - 1$, then there exists a feasible (K, F) partition of V with q clusters containing F nodes, one cluster containing r nodes and the remaining clusters are empty. When $r < F - 1$, another feasible partition has $q - 1$ clusters of size F , one cluster of size $F - 1$, one cluster of size $r + 1$ and all remaining clusters are empty. The cutsets of these two partitions are of different sizes and, because the edge coefficients

have been proved to be all equal to α , the comparison between the expressions of πx for these two different solutions leads to the conclusion that:

$$(F - r - 1)\alpha = 0 \Rightarrow \alpha = 0 \Rightarrow \pi_0 = 0$$

and this completes the proof for when $r < F - 1$.

On the other hand, if $r = F - 1$, one can apply the same rationale as above for the two following feasible solutions. The first one contains q clusters of size F , one cluster of size $F - 1$ and all other clusters are empty. The second solution contains q clusters of size F , one cluster of size $F - 2$, one cluster containing a single node and all remaining clusters are empty. The difference between the size of the cutsets corresponding to these two solutions is $F - 2$, which implies again that $\alpha = \pi_0 = 0$ and the proof is complete for the case (i.1).

case (i.2) : $F = 2$, $K \geq 2$ and $2 \leq n \leq 2K - 2$.

The proof for $n = 2$ is trivial and we assume that $n = 2q + r \geq 3$ with $r \in \{0, 1\}$. For $r = 0$ (n is even), let u and v be two nodes in V and $(L_1, \dots, L_q, L_{q+1}, \dots, L_K)$ and $(S_1, \dots, S_q, S_{q+1}, \dots, S_K)$ be two feasible partitions such that:

- $L_i = S_i \subset V \setminus \{u, v\}$ and $|L_i| = 2$ for all $i = 1, \dots, q - 1$;
- $L_i = S_i = \emptyset$ for all $i = (q + 2), \dots, K$;
- $L_q = \{u, v\}$ and $L_{q+1} = \emptyset$;
- $S_q = \{u\}$ and $S_{q+1} = \{v\}$

Comparing the expressions of πx for the incidence vectors of the two (K, F) multicuts defined above yields:

$$\pi_e = \pi_0 = 0 \quad \forall e \in E$$

A proof still has to be given for the case $r = 1$ (n odd). For this, let u, v and w be three nodes in V and consider the two feasible partitions $(L_1, \dots, L_q, L_{q+1}, \dots, L_K)$ and $(S_1, \dots, S_q, S_{q+1}, \dots, S_K)$ such that:

- $L_i = S_i \subset V \setminus \{u, v\}$ and $|L_i| = 2$ for all $i = 1, \dots, q-1$;
- $L_i = S_i = \emptyset$ for all $i = (q+2), \dots, K$;
- $L_q = \{u, v\}$ and $L_{q+1} = \{w\}$;
- $S_q = \{v\}$ and $S_{q+1} = \{u, w\}$

Since w, u, v were chosen arbitrarily, the comparison between the values of πx for the incidence vectors of such partitions leads to the conclusion that:

$$\pi_e = \alpha \quad \forall e \in E$$

Since $n = 2q + 1 \leq 2K - 2$ we conclude that $q \leq K - 2$. Therefore, there exists a feasible solution where $q - 1$ clusters have 2 nodes, three clusters have 1 node and all other clusters are empty. Note that the two solutions presented before have q clusters of size 2, one cluster of size one and all other clusters are empty. Thus, there are cutsets of different sizes and, because the coefficients of π all take the same value, this implies that:

$$\pi_e = \pi_0 = 0 \quad \forall e \in E$$

which completes the proof of (i.2).

case (i.3) : $K = 2$, $F \geq 3$ and $2 \leq n \leq 2F - 2$.

The proof for $n = 2$ is trivial and therefore we assume that $n \geq 3$. Consider the following sets:

- L_1 with $|L_1| = \lceil \frac{n}{2} \rceil - 1$, $v \in L_1$ and $u \notin L_1$;
- L_2 with $|L_2| = \lfloor \frac{n}{2} \rfloor$, $w \in L_2$ and $u \notin L_2$;
- $L_1 \cap L_2 = \emptyset$;
- $S_1 = (L_1 \setminus \{v\}) \cup \{w\}$;
- $S_2 = (L_2 \setminus \{w\}) \cup \{v\}$;

The following multicuts are feasible:

$$\begin{aligned}\delta_1 &= \delta(\{u\} \cup L_1, L_2) \\ \delta_2 &= \delta(\{u\} \cup L_2, L_1) \\ \delta_3 &= \delta(\{u\} \cup S_1, S_2) \\ \delta_4 &= \delta(\{u\} \cup S_2, S_1)\end{aligned}$$

Because $\pi x^{\delta_1} = \pi x^{\delta_2} = \pi x^{\delta_3} = \pi x^{\delta_4}$, we conclude that:

$$\pi_{uv} = \pi_{uw}$$

And since u, v and w can be chosen arbitrarily:

$$\pi_e = \alpha \quad \forall e \in E$$

The multicut given by $\delta(\{u, w\} \cup L_1, L_2 \setminus \{w\})$ can be easily verified to be feasible and its size differs from that of δ_1 . Because we have concluded that all coefficients in vector π have the same value α , this implies that:

$$\pi_e = \pi_0 = \alpha = 0 \quad \forall e \in E$$

Hence (i.3) is proved which completes the proof of (i).

(ii) : Assume that $P_{MC}^{K,F}(\mathcal{K}_n) \subset \{x \in \mathbb{R}^{|E|} : \pi x = \pi_0\}$. If we prove that $\pi x = \pi_0$ is a scalar multiple of $x(E) = \frac{K(K-1)}{2}F^2 - (K-1)F$, we are done.

Let u be a node in V , $(L_1, \dots, L_{K-2}, L_{K-1}, L_K)$ and $(S_1, \dots, S_{K-2}, S_{K-1}, S_K)$ be two feasible partitions of V where:

- $L_i = S_i$ and $|L_i| = F$ for all $i = 1, \dots, K-2$
- $L_{K-1} = S_{K-1} \setminus \{u\}$ and $|L_{K-1}| = F$
- $L_K \cup \{u\} = S_K$ and $|L_K| = F-1$

Comparing the expressions of πx for the incidence vectors of the two (K, F) multicuts defined above yields:

$$\sum_{z \in L_K} \pi_{uz} = \sum_{z \in S_{K-1}} \pi_{uz}$$

Suppose that w is a node in L_K and v is a node in S_{K-1} . The following two partitions are also feasible: $(L_1, \dots, L_{K-2}, (L_{K-1} \setminus \{v\}) \cup \{w\}, (L_K \setminus \{w\}) \cup \{v\})$ and $(S_1, \dots, S_{K-2}, (S_{K-1} \setminus \{v\}) \cup \{w\}, (S_K \setminus \{w\}) \cup \{v\})$. Comparing the expressions of πx for the incidence vectors of these two multicut gives:

$$\sum_{z \in L_K} \pi_{uz} - \pi_{uw} + \pi_{uv} = \sum_{z \in S_{K-1}} \pi_{uz} - \pi_{uv} + \pi_{uw}$$

Subtracting this equation from the one obtained above yields:

$$\pi_{uv} = \pi_{uw}$$

Again, since u , v and w can be chosen arbitrarily, we conclude that $\pi_e = \alpha$ for all $e \in E$.

In every feasible partition of V there are $K - 1$ clusters with maximum size F and one cluster with $F - 1$ nodes. Thus, the total number of edges in a (K, F) multicut is given by $\beta = \frac{K(K-1)}{2}F^2 - (K - 1)F$. So, we conclude that $\pi_0 = \beta\alpha$ and the proof of (ii) is complete.

(iii) : All feasible solutions here have the K clusters filled up to their maximum capacities, i.e., the size of any cluster in a feasible solution is always F . This implies that the number of edges leaving a node u of V in a feasible solution is given by $(K - 1)F$ and the hyperplane defined by the equation $x(\delta(v)) = (K - 1)F$ contains the polytope $P_{MC}^{K,F}(\mathcal{K}_n)$. There are n such hyperplanes and therefore $\dim(P_{MC}^{K,F}(\mathcal{K}_n)) \leq |E| - n$. The result in (iii) holds if we can exhibit a set of $|E| - n + 1$ affine independent points in $P_{MC}^{K,F}(\mathcal{K}_n)$.

Pick one node $u \in V$ and let \mathcal{K}_{n-1} denote the complete graph obtained from \mathcal{K}_n by removing node u from V and all its incident edges from E . From the result in (ii), the size of any maximal set of affine independent points in the polytope $P_{MC}^{K,F}(\mathcal{K}_{n-1})$ is $\frac{(KF-1)(KF-2)}{2}$. Let X be one such set.

If we add node u to the unique set containing $F - 1$ nodes in all partitions corresponding to incidence vectors in X , the partitions thus obtained are feasible for the problem with $n = KF$. Moreover, the incidence vectors of these (K, F) multicuts remain affinely independent.

The size of this set is:

$$\begin{aligned}
\frac{(KF-1)(KF-2)}{2} &= \frac{(KF-1)(KF)}{2} - (KF-1) \\
&= |E| - KF + 1 \\
&= |E| - n + 1
\end{aligned}$$

Therefore, this is a maximal affine independent set and the proof is complete. \square

The next theorem shows how a facet defining inequality for $P_{MC}^{K,F}(\mathcal{K}_n)$ is inherited by $P_{MC}^{K,F}(\mathcal{K}_{n-1})$. That is, we give some necessary conditions for when the facet property remains if a single node is removed from \mathcal{K}_n .

Theorem 3.7 *Let $\mathcal{K}_{n-1} = (V_{n-1}, E_{n-1})$ denote the complete complete subgraph induced in \mathcal{K}_n by the nodes in $V_n \setminus \{u\}$. Suppose that $P_{MC}^{K,F}(\mathcal{K}_n)$ and $P_{MC}^{K,F}(\mathcal{K}_{n-1})$ are both full-dimensional. Let π and x be two vectors in $\mathbb{R}^{|E|-n}$, x^u and $\mathbf{0}$ be two vectors in \mathbb{R}^n and assume that the components of x^u correspond to the edges in $\delta(u)$. Moreover, suppose that $(\pi, \mathbf{0})(x, x^u) \leq \pi_0$ is a facet defining inequality for $P_{MC}^{K,F}(\mathcal{K}_n)$ and that $\pi x \leq \pi_0$ is valid with respect to $P_{MC}^{K,F}(\mathcal{K}_{n-1})$. Then, $\pi x \leq \pi_0$ is facet defining for $P_{MC}^{K,F}(\mathcal{K}_{n-1})$.*

Proof: Suppose the contrary, i.e., $\pi x \leq \pi_0$ is not facet defining for $P_{MC}^{K,F}(\mathcal{K}_{n-1})$. This implies that there exists a vector $a \in \mathbb{R}^{|E|-n}$ and a scalar $b \in \mathbb{R}$ such that: $ax \leq b$ is valid for $P_{MC}^{K,F}(\mathcal{K}_{n-1})$, $\{x \in P_{MC}^{K,F}(\mathcal{K}_{n-1}) : \pi x = \pi_0\} \subset \{x \in P_{MC}^{K,F}(\mathcal{K}_{n-1}) : ax = b\}$ and there exists no scalar λ such that $(\pi, \pi_0) = \lambda(a, b)$ (note that the polytopes are full-dimensional). Define F_π and F_a as follows:

$$\begin{aligned}
F_\pi &= \{(x, x^u) \in P_{MC}^{K,F}(\mathcal{K}_n) : (\pi, \mathbf{0})(x, x^u) = \pi_0\} \\
F_a &= \{(x, x^u) \in P_{MC}^{K,F}(\mathcal{K}_n) : (a, \mathbf{0})(x, x^u) = b\}
\end{aligned}$$

By assumption, there exists no scalar λ such that $(\pi, \mathbf{0}, \pi_0) = \lambda(a, \mathbf{0}, b)$ and, since $(\pi, \mathbf{0})(x, x^u) \leq \pi_0$ is facet defining for $P_{MC}^{K,F}(\mathcal{K}_n)$, there exists a $(0, 1)$ vector (\bar{x}, \bar{x}^u) in $P_{MC}^{K,F}(\mathcal{K}_n)$ such that $(\bar{x}, \bar{x}^u) \in F_\pi$ and $(\bar{x}, \bar{x}^u) \notin F_a$ (otherwise $\dim(F_\pi) \leq |E| - 2$ and F_π cannot be a facet).

Now, (\bar{x}, \bar{x}^u) is the incidence vector of a feasible partitioning of \mathcal{K}_n . The partitioning obtained by the removal of node u from \mathcal{K}_n is clearly feasible for \mathcal{K}_{n-1} and \bar{x} is the incidence vector of the multicut corresponding to such partition. Therefore,

$$(\pi, 0)(\bar{x}, \bar{x}^u) = \pi\bar{x} = \pi_0$$

and

$$(a, 0)(\bar{x}, \bar{x}^u) = a\bar{x} \neq b$$

This contradicts our initial assumption that $\{x \in P_{MC}^{K,F}(\mathcal{K}_{n-1}) : \pi x = \pi_0\} \subset \{x \in P_{MC}^{K,F}(\mathcal{K}_{n-1}) : ax = b\}$ and this concludes the proof. \square

In the theorem below we show that facet defining inequalities from the equipartition polytope can be used to derive facets for the (K, F) partition polytope for the complete graph on $KF - 2$ nodes. If the conditions in Theorem 3.7 hold, these inequalities remain facet defining for the (K, F) partition polytope defined on complete graphs of smaller size.

Theorem 3.8 *Let \mathcal{K}_{2F} be a complete subgraph of $\mathcal{K}_n = (V, E)$ with $n = KF - 2$. Let π and x be two vectors in $\mathbb{R}^{F(2F-1)}$, \tilde{x} and $\mathbf{0}$ be two vectors in $\mathbb{R}^{|E|-F(2F-1)}$ and the components of vectors x and \tilde{x} correspond to the edges in \mathcal{K}_{2F} and in $\mathcal{K}_n - \mathcal{K}_{2F}$ respectively. Suppose that $\pi x \geq \pi_0$ is a facet defining inequality of $P_{EC}(\mathcal{K}_{2F})$ (the equipartition polytope defined on \mathcal{K}_{2F}) and that $(\pi, \mathbf{0})(x, \tilde{x}) \geq \pi_0$ is valid for $P_{MC}^{K,F}(\mathcal{K}_n)$. If $K \geq 4$, $F \geq 3$ and the support graph of $\pi x \geq \pi_0$ contains no more than $2F - 1$ nodes, then $(\pi, \mathbf{0})(x, \tilde{x}) \geq \pi_0$ is facet defining for $P_{MC}^{K,F}(\mathcal{K}_n)$.*

Proof: Define $G_\pi = (V_\pi, E_\pi)$ to be the support graph of the inequality $\pi x \geq \pi_0$.

Suppose that $\{(x, \tilde{x}) \in P_{MC}^{K,F}(\mathcal{K}_n) : (\pi, \mathbf{0})(x, \tilde{x}) = \pi_0\} \subset \{(x, \tilde{x}) \in P_{MC}^{K,F}(\mathcal{K}_n) : a(x, \tilde{x}) = b\}$. From Theorem 3.6, $P_{MC}^{K,F}(\mathcal{K}_n)$ is a full-dimensional polytope and, therefore, if we prove that (a, b) is a scalar multiple of $((\pi, \mathbf{0}), \pi_0)$ we are done.

Let z be a node in $V(\mathcal{K}_{2F}) \setminus V_\pi$ and $(L_1, L_2 \cup \{z\})$ be an equipartition of the nodes in \mathcal{K}_{2F} such that $x^{\delta(L_1)} \in \mathbb{R}^{F(2F-1)}$ satisfies $\pi x \geq \pi_0$ at equality. Let w , u and v be three

distinct nodes in $V \setminus V(\mathcal{K}_{2F})$ and define the partition $(L_3, \dots, L_{K-1}, L_K)$ of the nodes in $V \setminus (L_1 \cup L_2 \cup \{w, u, v\})$ such that:

- $|L_i| = F$ for all $i = 3, \dots, K-3$
- $|L_{K-1}| = |L_K| = F-2$
- $z \in L_K$

The multicuts described below are (K, F) multicuts that satisfy $(\pi, 0)(x, \tilde{x}) = \pi_0$:

$$\begin{aligned} &\delta(L_1, L_2, \dots, L_{K-1} \cup \{w, u\}, L_K \cup \{v\}) \\ &\delta(L_1, L_2, \dots, L_{K-1} \cup \{w, v\}, L_K \cup \{u\}) \\ &\delta(L_1, L_2, \dots, L_{K-1} \cup \{u\}, L_K \cup \{w, v\}) \\ &\delta(L_1, L_2, \dots, L_{K-1} \cup \{v\}, L_K \cup \{w, u\}) \end{aligned}$$

Comparing the expressions of $(a, \tilde{a})(x, \tilde{x})$ for the incidence vectors of the multicuts above we obtain that $a_{wu} = a_{wv}$.

Note that $V_\pi \subset V(\mathcal{K}_{2F})$ and, therefore, \mathcal{K}_{2F} can be defined in different ways depending on the nodes of $V \setminus V_\pi$ taken to be in $V(\mathcal{K}_{2F})$. This implies that, in fact, the nodes u , v and w in the previous result can be taken arbitrarily in $V \setminus V_\pi$ which leads to the conclusion that:

$$a_{uv} = \gamma \quad \forall u \text{ and } v \in V \setminus V_\pi, u \neq v$$

Consider the two following multicuts satisfying $(\pi, \mathbf{0})(x, \tilde{x}) = \pi_0$:

$$\begin{aligned} &\delta(L_1, L_2 \cup \{z\}, \dots, L_{K-1} \cup \{w, u\}, L_K \cup \{v\} \setminus \{z\}) \\ &\delta(L_1, L_2 \cup \{z\}, \dots, L_{K-1} \cup \{w\}, L_K \cup \{u, v\} \setminus \{z\}) \end{aligned}$$

Comparing the expressions of $a(x, \tilde{x})$ for these two vectors we get that

$$a_{uv} = \gamma = 0 \quad \forall u \text{ and } v \in V \setminus V_\pi, u \neq v$$

So far, we proved that the coefficients of vector a for edges not containing nodes in V_π are all null. We still have to find relations involving: (i) edges that have one endnode in V_π and the other endnode not in V_π and (ii) edges that have both endnodes in V_π .

From the remark following Theorem 2.3 in Chapter 2, if \mathcal{K}_{2F-1} is the complete subgraph of \mathcal{K}_{2F} obtained by removing any node from $V(\mathcal{K}_{2F})$, then $\pi x \leq \pi_0$ is facet defining for $P_{EC}(\mathcal{K}_{2F-1})$.

Let X be a maximal set of affinely independent points in $P_{EC}(\mathcal{K}_{2F-1})$. Any incidence vector of an equicut in X can be extended to an incidence vector of a (K, F) multicut of \mathcal{K}_n . For this, let (L_3, \dots, L_K) be any feasible $(K-2, F)$ partition of $V \setminus V(\mathcal{K}_{2F-1})$. If L_1 and L_2 are the two sets in the equipartition of \mathcal{K}_{2F-1} defined by any incidence vector in X , $(L_1, L_2, L_3, \dots, L_K)$ is a feasible (K, F) partition of \mathcal{K}_n . The incidence vectors of these (K, F) multicuts can be used (as in the proof that $\pi x \geq \pi_0$ is facet defining for $P_{EC}(\mathcal{K}_{2F-1})$) to prove that:

$$a_e = \alpha_1 \pi_e + \alpha_2 \quad \forall e \in E_\pi$$

and

$$a_e = \alpha_2 \quad \forall e \in E(\mathcal{K}_{2F-1}) \setminus E_\pi$$

where $\alpha_1 \in \mathbb{R}_+$ and $\alpha_2 \in \mathbb{R}$.

If $|V_\pi| \leq 2F-2$, then there exists at least one edge $e \in E(\mathcal{K}_{2F-1}) \setminus E_\pi$ and, from the results obtained so far, a_e is zero. Therefore, in this case, α_2 is zero and the proof is complete.

Suppose then that $|V_\pi| = 2F-1$ and u is the node in $V \setminus V_\pi$ that is added to V_π to form the set $V(\mathcal{K}_{2F})$. We can use similar arguments as before, to prove that:

$$a_e = \alpha_u \quad \forall e \in E(\mathcal{K}_{2F}) \setminus E_\pi$$

The notation α_u is used here to indicate that this result can be proved whatever is the node u assigned to \mathcal{K}_{2F} . However, we still have no proof that the constants α_u are the same for all possible choices of u . Below we prove that α_u is indeed zero for any node u .

The set V_π can be partitioned into two sets L_1 and L_2 such that $|L_1| = F$ and $|L_2| = F-1$ and such that $\pi x^{\delta(L_1)} = \pi_0$. Define the partition $(L_3, \dots, L_{K-1}, L_K)$ of the nodes in $V \setminus (L_1 \cup L_2)$ such that:

- $|L_i| = F$ for all $i = 3, \dots, K - 3$
- $|L_{K-1}| = |L_K| = F - 1$

The multicuts described below are (K, F) multicuts that satisfy $(\pi, \mathbf{0})(x, \tilde{x}) = \pi_0$:

$$\begin{aligned} \delta(L_1, L_2 \cup \{u\}, \dots, L_{K-1}, L_K) \\ \delta(L_1, L_2, \dots, L_{K-1}, L_K \cup \{u\}) \end{aligned}$$

Comparing the expressions of $(a, \tilde{a})(x, \tilde{x})$ for the incidence vectors of the multicuts above we obtain that

$$\sum_{z \in L_K} a_{uz} = \sum_{z \in L_2} a_{uz}$$

which implies that:

$$(F - 1)\alpha_u = 0 \quad \Rightarrow \quad \alpha_u = 0$$

and this completes the proof. \square

It is possible to give a proof of Theorem 3.8 with $K = 3$ and $|V_\pi| \leq 2F - 2$ using similar arguments as above.

The results of Chapter 2 can now be extended to the (K, F) partition polytope. If K and F are taken appropriately, it is easy to check the validity of cycle, PBC and suspended tree inequalities when supports are covers for a single cluster of $\mathcal{K}_n = \mathcal{K}_{KF-2}$ and from Theorem 3.8 these inequalities must define facets of $P_{MC}^{K,F}(\mathcal{K}_n)$. Examples of the support graphs of such inequalities are shown in Figure 3.4 below, where we assume that: $n = 38$, $K = 4$ and $F = 10$. Note also that Theorem 3.7 can be used to prove that these inequalities still remain facet defining when the graph loses some of its nodes.

Now we address the following question. The cycle inequality defines a facet of the (K, F) polytope if the nodes of the cycle form a minimal cover of a single cluster of a feasible partition. If we take a cycle $(V(C), C)$ with $|V(C)| = 2F + 1$, then the inequality $x(C) \geq 3$ is trivially valid. More generally, we are given a cycle C that is a minimal $(r - 1)$ -cover (i.e., $|V(C)| = (r - 1)F + 1$) and the valid inequality:

$$x(C) \geq r \tag{9}$$

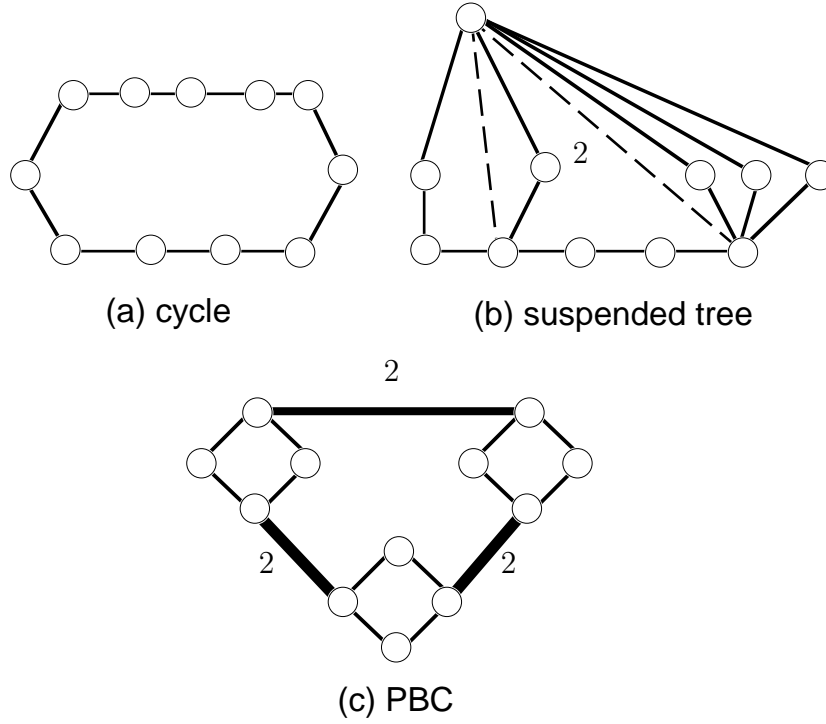


Figure 3.4:

We would like to know how strong such an inequality is.

Consider the example of Figure 3.5 below, where $r = F = 6$, and assume that n and K are large enough so that the cycle shown is a subgraph of \mathcal{K}_n . Inequality (9) becomes $x(C) \geq 6$. Given a (K, F) partition of \mathcal{K}_n , a set of consecutive nodes in the cycle that are assigned to a same cluster and which is maximal with respect to this property is called a *sector*. In this example, any feasible solution defines at least 6 sectors in the cycle. Clearly, the multicuts that satisfy $x(C) = 6$ are those that partition the nodes in C into 6 sectors.

We claim that all these multicuts contain the edge $(2, 11)$. To see this, assume that edge $(2, 11)$ is not in a multicut. Then nodes 2 and 11 are in the same cluster ($x_{2,11} = 0$), but since these nodes are too far apart, they cannot be in the same sector of any (K, F) multicut, which implies that at least 7 sectors are defined by the solution. Therefore, we can lift the coefficient of edge $(2, 11)$ to one and the new inequality becomes $x(C) + x_{2,11} \geq 7$.

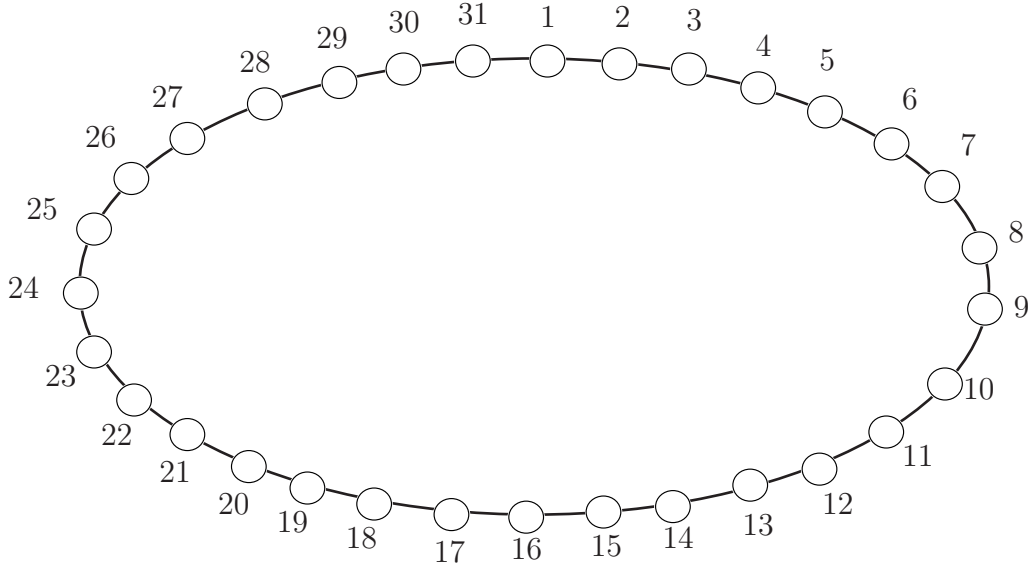


Figure 3.5:

All points satisfying the new inequality at equality are either the ones that already satisfied $x(C) = 6$ or the incidence vectors of (K, F) partitions that define 7 sectors in the cycle and such that nodes 2 and 11 are in different sectors of nodes belonging to a same cluster. Figure 3.6 shows the support graph of the new inequality.

We want to derive a general procedure for lifting the coefficients of chords of the cycle $(V(C), C)$. For this, we introduce some additional notation. Let \overline{S} denote the set of all chords of C not yet tested for lifting and S the set of all chords of C that have been lifted. Given a chord (i, j) in \overline{S} , let P_{ij} denote the shortest path from node i to node j in the graph $(V(C), C \cup S)$ and C_{ij} be the cycle formed by the edges in $\{(i, j)\} \cup P_{ij}$.

The **lifting procedure** works as follows. It initializes the set \overline{S} with all the chords of C and the set S as empty. Then, it seeks an edge (i, j) in \overline{S} for which $|C_{ij}| \geq F + 1$. If no such edge exists, the procedure stops. Otherwise, the procedure continues for another iteration after the following updates have been executed: (i) the coefficient of (i, j) is lifted from zero to one; (ii) the RHS of the current inequality is incremented by one; (iii) edge (i, j) is inserted

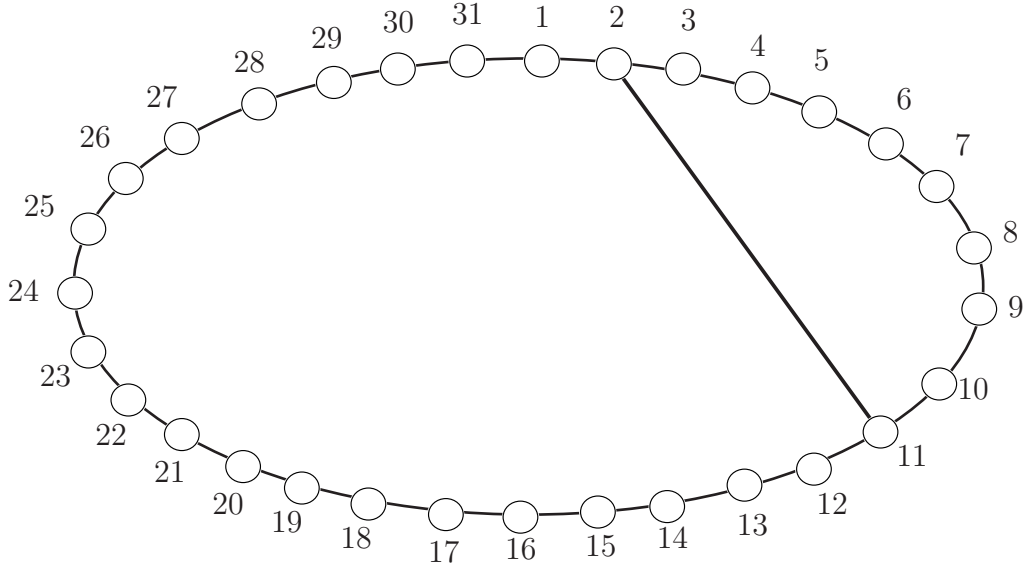


Figure 3.6:

in the set S and (iv) edge (i, j) is removed from the set \overline{S} .

Theorem 3.9 justifies the operations that are carried out in the lifting procedure.

Theorem 3.9 *Let $(V(C), C)$ be a cycle of \mathcal{K}_n such that $|C| = (r-1)F+1$. Suppose that at a some iteration of the lifting procedure, the current inequality is given by $x(C) + x(S) \geq r + s$, where s is the size of set S . If (i, j) is an edge in \overline{S} and $|C_{ij}| \geq F + 1$, then the inequality $x(C) + x(S) + x_{ij} \geq r + s + 1$ is valid for $P_{MC}^{K,F}(\mathcal{K}_n)$ and, moreover, $\{x \in P_{MC}^{K,F}(\mathcal{K}_n) : x(C) + x(S) = r + s\} \subset \{x \in P_{MC}^{K,F}(\mathcal{K}_n) : x(C) + x(S) + x_{ij} = r + s + 1\}$*

Proof: Define:

- $F_\ell = \{x \in P_{MC}^{K,F}(\mathcal{K}_n) : x(C) + x(S) = r + s\}$
- $F_{\ell+1} = \{x \in P_{MC}^{K,F}(\mathcal{K}_n) : x(C) + x(S) + x_{ij} = r + s + 1\}$

Let us compute $\alpha = \min\{x(C) + x(S) : x \in \overline{P}\}$, where $\overline{P} = \{x \in P_{MC}^{K,F}(\mathcal{K}_n) : x_{ij} = 0\}$.

Let δ be the cutset of a feasible partition of \mathcal{K}_n in which the nodes i and j are in the same cluster. Thus, if x^δ is the incidence vector of this cutset, we have that x^δ is in \overline{P} . Define q to be the number of sectors defined in C by the cutset δ and p be the number of clusters having nodes in $V(C)$. For i varying from 1 to p , denote by q_i the number of sectors whose nodes are in the i -th cluster that has a nonempty intersection with C in the partition δ . By definition, the following relations hold: $\sum_{i=1}^p q_i = q$, $p \geq r$ and $x^\delta(C) = q$.

Now, we know that all the chords in S have been added by the lifting procedure. So, consider any subset of $V(C)$ with at most F nodes and which defines t different sectors in C . The number of chords with both endnodes in these t sectors is at most $t - 1$ (otherwise there will be a cycle with less than F nodes in the subgraph $(V(C), C \cup S)$).

Let β be the number of chords in S whose endnodes are in two sectors in C of a same cluster of the partition with cutset δ . These are the chords of S that are not in the cutset δ . Therefore, a lower bound for $x^\delta(C) + x^\delta(S)$ is computed by taking β at its maximum value β_{\max} , that is:

$$x^\delta(C) + x^\delta(S) = q + s - \beta \geq q + s - \beta_{\max}$$

Since edge (i, j) is not in S , β_{\max} is computed as $\sum_{i=1}^p (q_i - 1) - 1$. Thus,

$$\begin{aligned} x^\delta(C) + x^\delta(S) &\geq q + s - \beta_{\max} \\ &= q + s - \left(\sum_{i=1}^p (q_i - 1) - 1 \right) \\ &= p + s + 1 \end{aligned}$$

Because $p \geq r$, we conclude that $x^\delta(C) + x^\delta(S) \geq r + s + 1$. Finally, since the multicut δ is taken arbitrarily so that $x_{ij}^\delta = 0$, the following holds:

$$\begin{aligned} \min\{x(C) + x(S) : x \in \overline{P}\} &\geq r + s + 1 \\ &\Downarrow \\ \alpha &\geq r + s + 1 \end{aligned}$$

Therefore, $x(C) + x(S) + x_{ij} \geq r + s + 1$ is valid for $P_{MC}^{K,F}(\mathcal{K}_n)$ and there exists no $x \in F_\ell$ such that $x_{ij} = 0$. Since every point with $x_{ij} = 1$ is trivially in $F_{\ell+1}$, we conclude that $F_\ell \subset F_{\ell+1}$. \square

Usually, if the cycle C_{ij} is of size F or smaller, the chord (i, j) cannot be lifted. This is always true if all the edges in C_{ij} except (i, j) are in the original cycle C . In this case, the nodes of $V(C)$ can be easily partitioned into r sectors such that all nodes in C_{ij} are contained in the same sector. Thus, the incidence vector of this multicut satisfies $x(C) = r$ and it also satisfies $x(C) + x(S) = r + s$. For instance, in the example in Figure 3.5, the chord $(1, 6)$ cannot be lifted because, as it is illustrated in Figure 3.7, there exists a solution satisfying $x(C) = 6$ and not cutting $(1, 6)$.

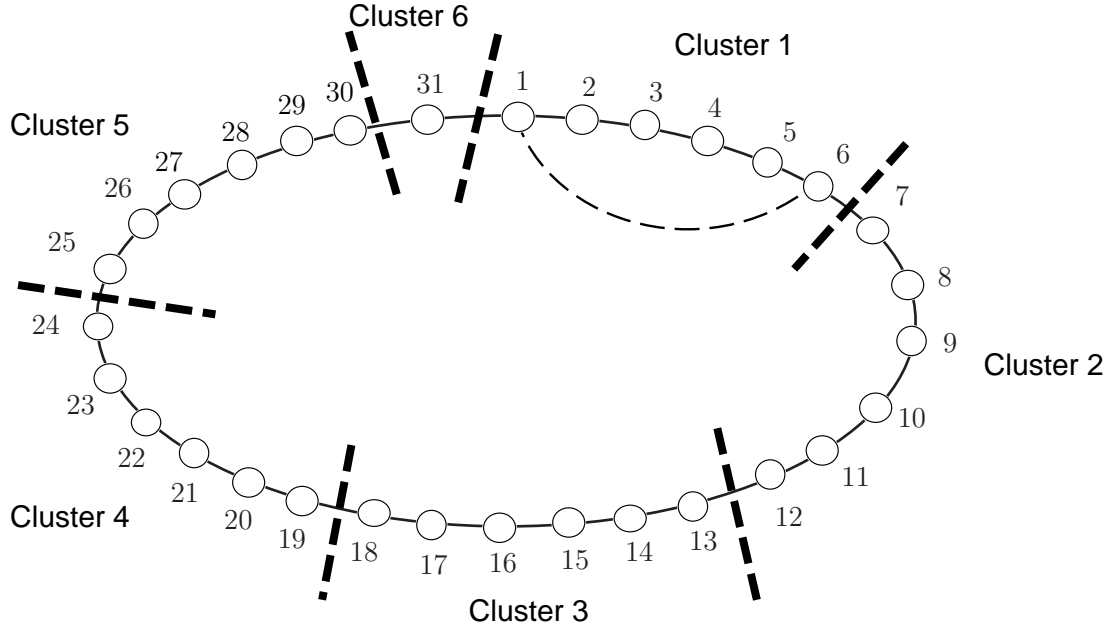


Figure 3.7:

Another situation in which the chord cannot be lifted is when C_{ij} is of size at most F and it contains exactly one chord e that was previously lifted. If this is the case, one can show that there exists a (K, F) partition of \mathcal{K}_n such that the cycle C is partitioned into $r + 1$

sectors where the nodes of C_{ij} are contained in two different sectors of a same cluster and the other sectors are each in a different cluster. Thus, the incidence vector of this multicut satisfies $x(C) = r + 1$ and $x(S) = s - 1$ (e is not cut). This can be seen in Figure 3.8 where we consider the possibility of lifting the chord $(4, 13)$ after lifting edge $(2, 11)$ in Figure 3.6.

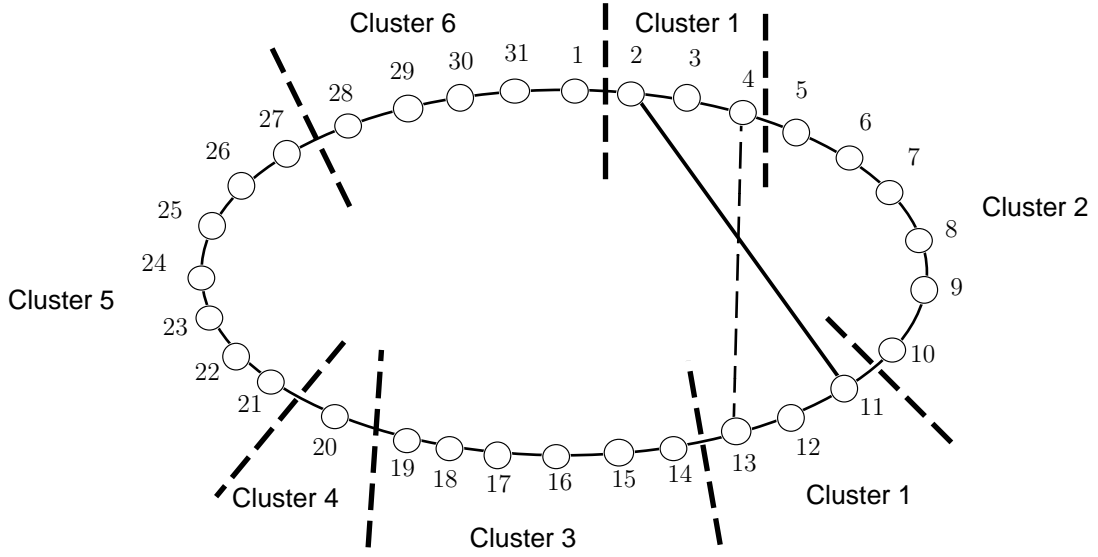


Figure 3.8:

When C_{ij} is of size at most F and it contains at least two chords that have been previously lifted, it may be the case that the lifting for edge (i, j) is possible. To see this, consider the following example. Suppose that we start with the inequality with support graph shown in Figure 3.5 and that the chords $(2, 11)$ and $(12, 22)$ have been the first two chords lifted. Edge $(1, 23)$ will not be a candidate for lifting in the next iteration of our lifting procedure because the cycle $C_{ij} = \{(1, 2), (2, 11), (11, 12), (12, 22), (22, 23), (23, 1)\}$ has size $F = 6$. However, looking more carefully at this example, we conclude that $x(C) + x(S)$ is greater than or equal to 9 whenever $x_{1,23}$ is not cut and, therefore, $(1, 23)$ should be considered for lifting.

Thus, we have seen that the multicover cycle inequality $x(C) \geq r$ can be strengthened by lifting some chords in C . For this, one can use the lifting procedure suggested above. However, as we have just shown, it is possible that the inequality produced by the procedure

can be further strengthened but, for this, a different lifting strategy has to be devised.

In the next section we investigate the multicut polyhedron when the nodes have weights and the cardinality constraints are replaced by knapsack constraints.

3.4 The Graph Partitioning Problem with Knapsack Capacity Constraints

Consider the graph $G = (V, E)$ and assume that there is a weight w_i associated to every node i in V . We are interested in the partitions of V into K subsets with weight capacity given by a constant W , i.e., the sum of the node weights in each cluster is at most W . Analogously to the previous section, let us call such a partition a (K, W) partition of V and the corresponding cutset a (K, W) multicut of G .

Define $P_{MC}^{K,W}(G)$ to be the convex hull of incidence vectors of (K, W) multicuts in G . The dimension of the (K, W) multicut polytope depends on the node weights and on the cluster capacity W . So, no general result concerning the dimension of $P_{MC}^{K,W}(G)$ is given. Nevertheless, it is possible to obtain valid inequalities for this polytope with supports given by cycles, suspended trees, PBCs and multicover cycles (as in Section 3.3).

The next theorem gives the conditions for a suspended tree inequality to be valid with respect to $P_{MC}^{K,W}(G)$.

Theorem 3.10 *Let $(V(T), T)$ be a suspended tree in G with apex at node u_0 and satisfying $W < w_{u_0} + \sum_{u \in V(T)} w_u < 2W$. Then, the suspended tree inequality*

$$\omega(T, u_0)x \geq 2$$

is valid with respect to $P_{MC}^{K,W}(G)$, where $\omega(T, u_0)$ is defined as in inequality (5) of Section 2.5.

Note that if the tree T is a path the inequality in Theorem 3.10 reduces to a cycle inequality. The proof is omitted since it is very similar to that of Proposition 2.15 of Section 2.5.

Let $\mathcal{C} = ((V(C_1), C_1), (V(C_2), C_2), \dots, (V(C_r), C_r))$ be a PBC in G and N , with $|N| = t$, be the set of nodes that are common to all cycles in \mathcal{C} . As in Section 2.4, we can define $((V(P_{ij}), P_{ij})$ to be the path in cycle C_j joining the nodes s_i and s_{i+1} (indices are taken

modulo t). But here we generalize the definition of the q_{ij} to allow for the node weights:

$$q_{ij} = \sum_{u \in V(P_{ij})} w_u - w_{s_i} - w_{s_{i+1}}$$

The value of \overline{Q} is still defined as the sum of the largest $(r-1)$ values of the q_{ij} . Theorem 3.11 gives the conditions for a PBC inequality to be valid with respect to $P_{MC}^{K,W}(G)$.

Theorem 3.11 *Let $(V(C), C)$ be a PBC subgraph of G with r cycles. Assume that $W < \sum_{u \in V(C)} w_u < 2W$. If $\sum_{u \in V(C)} w_u - \overline{Q} > W$, then the PBC inequality $\sum_{e \in C} a_e x_e \geq 2r$ is valid for $P_{MC}^{K,W}(G)$, where the coefficients a_e are computed as in inequality (3) of Section 2.4.*

Again we omit the proof since it follows the same steps as in the proof of Theorem 2.7 of Section 2.4.

If C is a cycle in G with $|C| > (r-1)W$, then the multicover cycle inequality $x(C) \geq r$ is valid for $P_{MC}^{K,W}(G)$. The lifting procedure described in Section 3.3 can be used here. As before, let S denote the set of chords already taken by the procedure. Given a chord (i, j) not in S , C_{ij} is the cycle formed by (i, j) and the edges in the path from i to j in $C \cup S$ whose sum of node weights is minimum. The chords chosen to be lifted are those such that $\sum_{u \in C_{ij}} w_u > W$. The proof that these chords can be lifted is as in Theorem 3.9.

In the inequalities presented so far, the role played by the node weights is restricted to the fact that the support graph should form a cover (or some generalization of a cover) for the clusters in the partition. In the valid inequality given in Theorem 3.12 below, the node weights are considered in a somewhat different way. For this, valid inequalities for $P_{MC}^{K,W}(G)$ are derived from valid inequalities for the knapsack polytope defined by the cluster capacity constraint. The support of such an inequality is a tree that spans the nodes whose coefficients are positive in the original knapsack inequality.

Theorem 3.12 *Let P_{Kn} be the knapsack polytope defined by the cluster capacity constraint of the graph partitioning problem, that is:*

$$P_{Kn} = \{y \in \mathbb{B}_+^n : \sum_{u \in V} w_u y_u \leq W\}$$

Suppose that $\pi y \leq \pi_0$ is a valid inequality for P_{Kn} and let $V(\pi)$ be the set of nodes u such that $\pi_u \neq 0$. Moreover, let T be a tree in G rooted at a node r that spans all nodes in $V(\pi)$. Let $P(u)$ denote the path in T from r to u . Then,

$$\sum_{u \in V(T)} \pi_u (1 - \sum_{e \in P(u)} x_e) \leq \pi_0 \quad (10)$$

is valid for $P_{MC}^{K,W}(G)$.

Inequality (10) is called the *knapsack tree inequality*. Let $S(e)$ denote the subtree of T not containing the root and obtained from T by removing e . Inequality (10) can be written as:

$$\sum_{e \in T} a_e x_e \geq \sum_{u \in V(T)} \pi_u - \pi_0 \quad (11)$$

where $a_e = \sum_{u \in V(S(e))} \pi_u$.

Before giving to the proof of Theorem 3.12, consider the following example. Suppose that $W = 8$ and that T is the tree rooted at node 1 as shown in Figure 3.9. The node weights are given by: $w_1 = 2$ and $w_2 = \dots = w_9 = 3$. The inequality $\sum_{u=1}^9 x_u \leq 3$ is valid for the knapsack polytope $P_{Kn} = \{y \in \mathbb{IB}_+^n : \sum_{u \in V} w_u y_u \leq W\}$. The corresponding knapsack tree inequality is then:

$$1 + (1 - x_{13} - x_{23}) + (1 - x_{13}) + (1 - x_{15} - x_{45}) + (1 - x_{15}) + (1 - x_{16}) + \\ + (1 - x_{16} - x_{67}) + (1 - x_{16} - x_{67} - x_{78}) + (1 - x_{16} - x_{67} - x_{79}) \geq 3$$

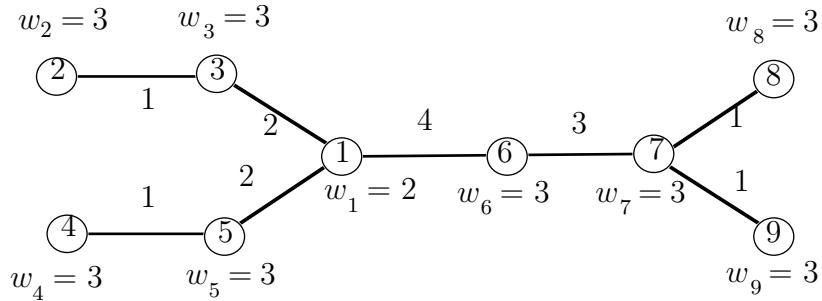


Figure 3.9:

Proof of Theorem 3.12: Suppose that there exists a vector x^* in $P_{MC}^{K,W}(G)$ not satisfying inequality (10). Let S be the node set given by $\{u \in V(T) : \sum_{e \in P(u)} x_e^* = 0\}$. Therefore, the root node r is in S . Since $1 - \sum_{e \in P(u)} x_e^* \leq 0$ for nodes u in $V(T) \setminus S$, we have that:

$$\sum_{u \in S} \pi_u (1 - \sum_{e \in P(u)} x_e^*) \geq \sum_{u \in V(T)} \pi_u (1 - \sum_{e \in P(u)} x_e^*)$$

or

$$\sum_{u \in S} \pi_u \geq \sum_{u \in V(T)} \pi_u (1 - \sum_{e \in P(u)} x_e^*)$$

Because x^* violates (10), the last inequality implies that:

$$\sum_{u \in S} \pi_u > \pi_0$$

The nodes in S are all in the same cluster. Thus, the incidence vector of S must be in P_{Kn} .

But, this contradicts the fact that $\pi y \leq \pi_0$ is valid for P_{Kn} . \square

Given any feasible partition of G , let S be the set of nodes in T defined as in the previous proof. The incidence vector of the corresponding multicut lies on the face defined by inequality (10) in $P_{MC}^{K,W}(G)$ if the sum of the π_u values for $u \in S$ is equal to π_0 . This implies that, in any solution satisfying (10) at equality, the path from the root to any node in T is cut exactly once.

Below we investigate how to strengthen the knapsack tree inequality. One obvious situation in which a strengthening is possible is when a coefficient a_e for some edge e in T is larger than $\sum_{u \in T} \pi_u - \pi_0$. Clearly, a_e can be reduced to at least the same value as the right-hand side of inequality (10) since no point in the face defined by the knapsack tree inequality can cut edge e .

Now suppose that a_{vw} is larger than π_0 for some edge (v, w) in T . Let T_v and T_w be the two trees obtained from T when edge (v, w) is removed. Assume, without loss in generality, that v and the root node r are in T_v and w is in T_w . Let ℓ_v and ℓ_w be lower bounds for $\sum_{e \in T_v} a_e x_e$ and $\sum_{e \in T_w} a_e x_e$ respectively. Clearly, any solution over the $P_{MC}^{K,W}(G)$ polytope cutting edge (v, w) satisfies $\sum_{e \in T} a_e x_e \geq a_{vw} + \ell_v + \ell_w$. If α_{vw} is defined to be equal to

$a_{vw} + \ell_v + \ell_w - (\sum_{u \in T} \pi_u - \pi_0)$, then the coefficient of (v, w) can be reduced of $\max\{\alpha_{vw}, 0\}$. Simple values for the lower bounds ℓ_v and ℓ_w can be obtained as follows.

The inequality $\sum_{u \in V(T_v)} \pi_u \leq \pi_0$ is trivially valid for P_{Kn} because $V(T_v) \subset V(T)$. The knapsack tree inequality corresponding to tree T_v is given by $\sum_{e \in T_v} a_e^v x_e \geq \sum_{u \in V(T_v)} \pi_u - \pi_0$. The coefficients a_e^v are given by:

$$a_e^v = \begin{cases} a_e & \text{if } e \notin P(v) \\ a_e - \sum_{u \in V(T_w)} \pi_u & \text{if } e \in P(v) \end{cases}$$

From the expression above, $a_e^v \leq a_e$ for all e in T_v . So, for any vector x in $P_{MC}^{K,W}(G)$, we have that:

$$\sum_{e \in T} a_e x_e \geq \sum_{e \in T_v} a_e x_e \geq \sum_{u \in V(T_v)} \pi_u - \pi_0 \quad (12)$$

Thus, ℓ_v can be taken as $\sum_{u \in V(T_v)} \pi_u - \pi_0$.

Analogously, the inequality $\sum_{u \in V(T_w)} \pi_u \leq \pi_0$ is trivially valid for P_{Kn} . The knapsack tree inequality corresponding to tree T_w , when w is taken as the root, is given by $\sum_{e \in T_w} a_e^w x_e \geq \sum_{u \in V(T_w)} \pi_u - \pi_0$ where $a_e^w = a_e$ for all e in T_w . Thus, ℓ_w can be taken as $\sum_{u \in V(T_w)} \pi_u - \pi_0$.

With the values of ℓ_v and ℓ_w given as before, we have:

$$\alpha_{vw} = a_{vw} + \left(\sum_{u \in V(T_v)} \pi_u - \pi_0 \right) + \left(\sum_{u \in V(T_w)} \pi_u - \pi_0 \right) - \left(\sum_{u \in V(T)} \pi_u - \pi_0 \right)$$

or

$$\alpha_{vw} = a_{vw} - \pi_0 > 0$$

Thus, after lifting, the coefficient of edge (v, w) is π_0 .

As an example, consider again the knapsack tree inequality with support given by Figure 3.9. Edge $(1, 6)$ has a coefficient of 4 which is larger than π_0 ($= 3$). The knapsack tree inequality corresponding to T_v is given by

$$x_{23} + x_{45} + 2x_{13} + 2x_{15} \geq 2$$

and the one corresponding to T_w by

$$x_{78} + x_{89} + 3x_{67} \geq 1$$

Therefore, any solution cutting edge $(1, 6)$ satisfies $\sum_{e \in T} a_e x_e \geq 4 + 2 + 1 = 7$ and $\alpha_{16} = 7 - 6 = 1$. This implies that the coefficient of edge $(1, 6)$ can be reduced to 3 ($= \pi_0$).

The liftings discussed above are resumed in the following proposition.

Proposition 3.13 *Let $\sum_{e \in T} a_e x_e \geq \sum_{u \in V(T)} \pi_u - \pi_0$ be a valid knapsack tree inequality defined as in Theorem 3.12. Suppose that F_a is the face of $P_{MC}^{K,W}(G)$ whose points satisfy $\sum_{e \in T} a_e x_e = \sum_{u \in V(T)} \pi_u - \pi_0$. Then, the inequality $\sum_{e \in T} \tilde{a}_e x_e \geq \sum_{u \in V(T)} \pi_u - \pi_0$, with \tilde{a}_e computed as the minimum among $\{a_e, \pi_0, \sum_{u \in V(T)} \pi_u - \pi_0\}$, is also valid for $P_{MC}^{K,W}(G)$ and the face it defines in this polytope contains F_a .*

The liftings suggested in Proposition 3.13 may not be maximal. To see this, consider again the example of Figure 3.9 and suppose that edge $(1, 6)$ is cut. We would like to know what is the minimum value of $\sum_{e \in T \setminus \{(1,6)\}} a_e x_e$ when x is a feasible vector satisfying $x_{16} = 1$. By inspection, one can see that this minimum is attained by cutting edges $(2, 3)$, $(4, 5)$, $(7, 8)$ and $(7, 9)$ which gives a value of 4. Thus, $\min\{\sum_{e \in T} a_e x_e : x_{16} = 1\} = 8$ and the coefficient of $(1, 6)$ can be reduced by 2. Arguing in the same way, the coefficient of edge $(6, 7)$ can also be reduced by 2. The new support graph is shown in Figure 3.10.

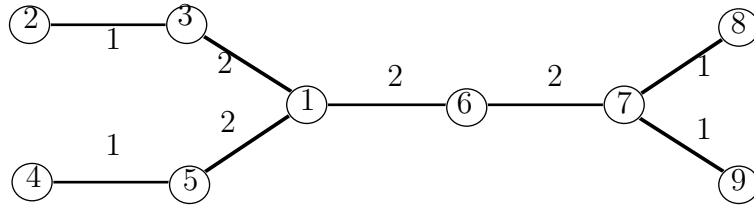


Figure 3.10:

Unfortunately, to compute these maximal liftings we have to solve knapsack problems to optimality and this is known to be a NP-hard problem.

4. Separation Routines and Computational Results for Graph Partitioning Problems

4.1 Introduction

In the last two chapters we have presented classes of strong valid inequalities for the equipartition polytope, and for the polytopes of the more general graph partitioning problem. Efficient algorithms for identifying violated inequalities in these classes have to be developed if one wants to use them in a cutting plane framework. An algorithm designed to find violated inequalities in a given class of valid inequalities is a separation routine for that class.

In this chapter we develop separation routines for some classes of valid inequalities introduced in Chapters 2 and 3. For the development of the separation routines, we have considered the graph partitioning problem with knapsack capacity constraints (Section 3.4). This is a natural choice since this problem generalizes the equipartition problem (Chapter 2) and the graph partitioning problem with cardinality capacity constraints (Section 3.3).

We then discuss the insertion of these separation routines in a branch-and-cut code developed by Ferreira, Martin and Weismantel (1992), and the numerical experiments that have been carried out in collaboration with the developers of the code.

The material of the chapter is organized as follows.

In Section 4.2, separation routines are given for the cycle and the PBC inequalities introduced in Chapter 2. A third separation routine, which seeks violated inequalities with support graphs given by trees, is then presented. Although tree inequalities usually do not define facets they are easily seen to be valid and may be strong if the graph under

consideration is very sparse. For completeness, the separation routine for the knapsack tree inequalities developed by Ferreira, Martin and Weismantel (1993) is also discussed.

The separation problems for the cycle and the tree inequalities are NP-hard. For both cases this can be shown using a "proof by restriction" (see Garey and Johnson, 1979). For cycles the transformation is from the Traveling Salesman Problem, while for trees the transformation is from the Steiner Tree Problem. The separation problem for the PBC inequalities is probably also NP-hard. Therefore, the separation routines presented here are heuristic algorithms. This implies that, for a given a class of valid inequalities, our separation routine may end up with an inequality that is not the most violated one in that class. It is also possible that the routine fails to produce a violated inequality even when one exists.

In Section 4.3 we focus our attention on a small number of selected implementation details of the branch-and-cut code which are necessary for the analysis of the computational results. Section 4.4 is devoted to the description of the different classes of problems chosen for numerical testing of the code. In Section 4.5, we describe the numerical experiments that have been carried out and discuss the results that have been obtained.

4.2 Separation Routines

The separation routines for the cycle, the PBC, the tree and the knapsack tree inequalities are now discussed. Suppose that S is the set of feasible solutions of the problem we want to solve and $\text{conv}(S)$ is the convex hull of S (which can be one of the polytopes introduced in Chapters 2 and 3). The separation routines have as input the fractional solution, say x^* , of the current linear relaxation of $\text{conv}(S)$. The separation routine for a given class \mathcal{F} of valid inequalities is an algorithm that checks if there is an inequality $\pi x \geq \pi_0$ in \mathcal{F} which is not satisfied by x^* . If such an inequality is found, then we add it to the linear relaxation.

For the first separation routine $\pi x \geq \pi_0$ is the cycle inequality $x(C) \geq 2$ discussed in Chapters 2 and 3. The second separation routine is designed to look for violated PBC inequalities (Chapters 2 and 3) where the PBC support graph is made of two cycles C_1 and C_2 . In this case $\pi x \geq \pi_0$ can be written as $x(C_1) + x(C_2) \geq 4$. Finally, the third separation routine looks for violated inequalities of the type $x(T) \geq 1$ where T is a tree whose nodes form a cover for any of the clusters in the partition. This last inequality was not discussed in the previous chapters but it can be trivially checked to be valid for the polytopes considered here.

For the descriptions of the algorithms below, we consider that a graph $G = (V, E)$ is given and that a weight w_u is assigned to each node $u \in V$. The current fractional solution is given by x^* and the maximum of the cluster capacities is given by W . Thus we can define the polytope $P_{MC}^{K,W}(G)$ from Chapter 3 and, as we mentioned there, this includes the equipartition polytope.

Throughout this section, the following representation is used for the edges in E . If an edge belongs to some tree (or some forest) that we want to characterize, then it is represented by a thick grey line. When the tree (forest) is rooted, the root node (nodes) is (are) represented by a square(s). If an edge is in the support of the current inequality,

it is represented by a thick black line. The remaining edges of the graph are represented by thin black lines.

The basic object we deal with in each inequality is a set of nodes $V' \subseteq V$ with $\sum_{u \in V'} w_u > W$ called a *cover*. We shall speak of a *cover cycle* (or tree) to mean that the sum of the weights of the nodes in the cycle (tree) is larger than W .

Given a subgraph $G' = (V', E')$ of G , we define the length of G' as the sum of the x^* variables associated to the edges in E' and, analogously, the weight of G' as the sum of weights of the nodes in V' . If G'' is another subgraph of G , we say that G' is heavier than G'' if the weight of G' is larger than that of G'' . According to the preceding paragraph, G' is said to be a cover if its weight is larger than W .

4.2.1 Separation routine for the Cycle Inequalities

This routine looks for a cover cycle C in G such that $\sum_{e \in C} x_e^* < 2$. Clearly if such a cycle is found, then a violated cycle inequality is available.

The routine is executed once for every node r of V for which the degree of r is larger than one. The node r is called the root node or just root for simplicity. The idea is that, at the output, the cycle will contain the root. Below, we describe the basic steps that finds the cycle C for a given root r .

In the **first step** of the separation routine, we construct a spanning tree $T(r)$ rooted at r which contains only two edges that are incident with r . For this, we use a greedy strategy very similar to Prim's algorithm. The tree $T(r)$ is initialized with the two least cost edges (in terms of the x^* variables) leaving r . At each iteration, a new node v is added to $T(r)$ such that $\frac{x_{uv}^*}{w_v}$ is minimum among all possible edges (u, v) where u is in the current tree $T(r)$ and v is not in it. This construction implies that there are two subtrees in $T(r)$ hanging from r .

To illustrate the algorithm, consider the graph shown in Figure 4.1 below. Assume that an equipartition problem is defined for this graph and, therefore, we have: $w_u = 1$ for all u in V , $n = 31$ and $W = \lceil \frac{n}{2} \rceil = 16$. Figure 4.1 shows a possible tree rooted at node 19.

In the **second step** of the routine, we try to find an edge which, together with some edges in $T(r)$, forms a cycle containing r . Thus, we consider the edges having each endnode in one of the different subtrees that hang from r . These are the edges that induce cycles in $T(r)$ containing the root r .

Only edges inducing cycles with length less than two are of interest since they can potentially lead to a violated inequality. If one such edge is available and it induces a cycle C in $T(r)$ satisfying $\sum_{u \in C} w_u > W$, then a violated cycle inequality is found. When many edges correspond to violated cycle inequalities, define e_I to be the edge corresponding to

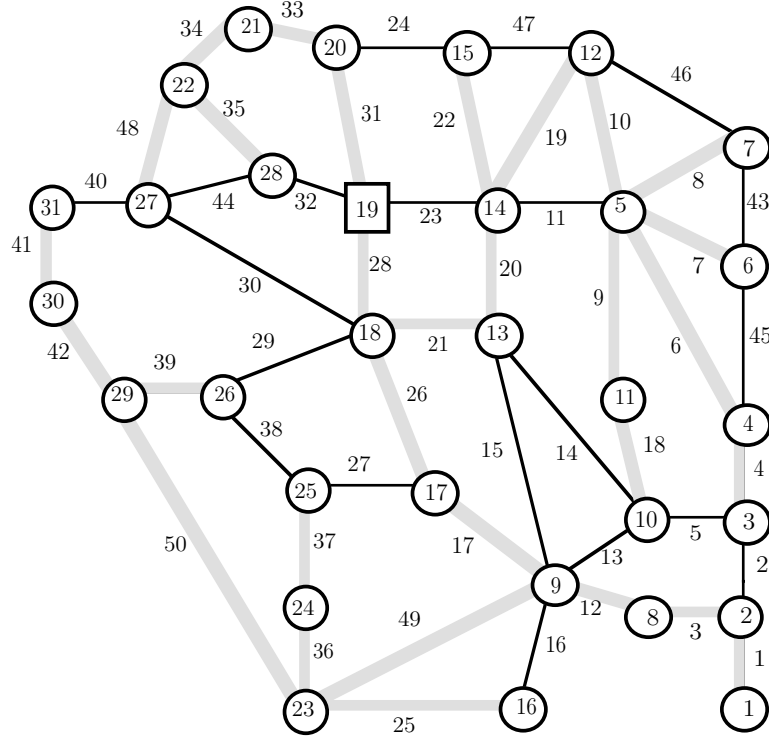


Figure 4.1:

the cycle of smallest weight. On the contrary, if there are edges inducing cycles with length less than two but none of them is a cover, then let e_I be the edge inducing the heaviest such cycle.

After this second step, either a violated cycle inequality is at hand or the heaviest cycle found does not constitute a cover. In the first case, we go to the lifting step to be described below. Otherwise, a **third step** has to be executed to produce a new cycle heavier than the present cycle. This step is repeated until the current cycle becomes a cover and has length less than 2 or until all the cycles that we can find have length at least 2.

To illustrate the operations that are executed in this third step, we continue with the example of Figure 4.1. For this, suppose that edge 40 is the edge e_I as defined above. The cycle induced by e_I in $T(r) = T(19)$ is shown in Figure 4.2 and has weight 12. We still

[illegible]

Consider the set \mathcal{L} of edges whose induced cycles are covers. If \mathcal{L} is not empty, define e_D^* to be the edge that minimizes the weight of $C(e_D)$ among all edges in \mathcal{L} . Otherwise, if \mathcal{L} is empty, define e_D^* to be the edge that maximizes the weight of $C(e_D)$. Let e_I^* be the unique edge in $C(e_D^*) \setminus T(r, e_D^*)$.

We go to the lifting step if $C(e_D^*)$ is a cover cycle since, in this case, a violated cycle inequality is available. If $C(e_D^*)$ is not a cover but it is heavier than C , then the following updates take place: T is modified by adding e_I and removing e_D^* , C is replaced by $C(e_D^*)$ and e_I is replaced by e_I^* . In the last case, the inequality associated to the new cycle C is not valid and the third step is repeated. The routine stops and no cycle inequality (for the root node r) is produced when $C(e_D^*)$ has the same weight as C .

Consider the example of Figures 4.1 and 4.2 and suppose that e_D^* is the edge 26. The tree $T(r, e_D) = T(19, 26)$ is obtained from the tree $T(19)$ (Figure 4.1) by removing edge 26 and adding edge 40 since $e_I = 40$ (see Figure 4.2) and is shown in Figure 4.3. Assuming that edge 13 induces a cycle of length less than two in $T(19, 26)$, no additional iteration is necessary to find a heavier cycle since this one has size 17 and is a cover.

Actually, edge e_I can be chosen such that one of its endnodes is the root node r . In other words, we do not have to restrict the set of candidate edges for inducing cycles to the ones having endnodes in the different subtrees of $T(r)$. Clearly, any edge not in $T(r)$ but with r as an endnode also induces a cycle in $T(r)$ that contains r . To see this, consider the example of Figure 4.1 where the root is now defined as being node 2. Because one of the subtrees only contains node 1 and the degree of that node is one, the routine fails to build the very first cycle if the original set of candidate edges is maintained as before.

Assume then that edges not in $T(r)$ and incident with the root are accepted in the set of candidate edges to form cycles. If e_I is one such edge, it is clear that the edge joining r to the subtree containing the other endnode of e_I is the only edge in the current cycle that can be deleted from the $T(r)$. The reason is that this edge has to be deleted so as to

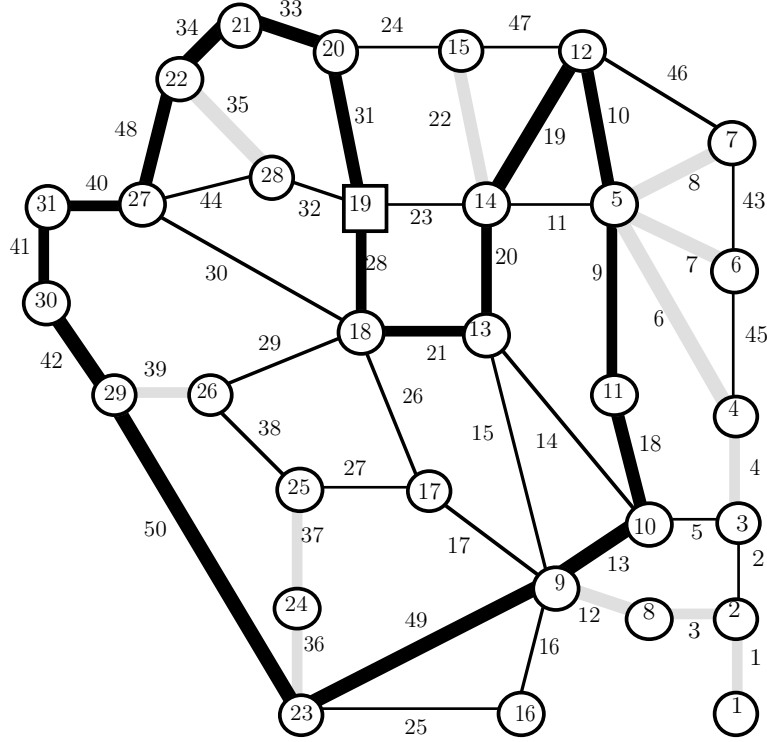


Figure 4.3:

keep the degree of r equal to 2 in the new tree. Such a situation is illustrated in Figure 4.4 where the graph of Figure 4.1 is considered but node 2 is now defined to be the root. Edge 2 (e_I) induces a cycle of weight 11 and edge 3 (e_D) must be removed (otherwise the degree of node 2 in the new tree will be larger than 2).

Once a violated cycle inequality is available, we go to a **fourth step** where we try to strengthen the inequalities by lifting. After the three first steps have been completed, the routine may end up with a cycle C corresponding to a violated cycle inequality for which $\sum_{u \in C} w_u > W + \alpha$ with $\alpha > 0$. Let (z, v) be a chord of C . There are two simple paths in C going from z to v . Suppose that P_{zv} is the path for which the sum of the node weights is minimum and let β be given by $\sum_{u \in P_{zv}} w_u$. If $\alpha - \beta > 0$, the inequality can be lifted in such a way that the support of the new inequality is the cycle C' arising from C by removing the edges in P_{zv} and adding the chord (z, v) . This operation is repeated until

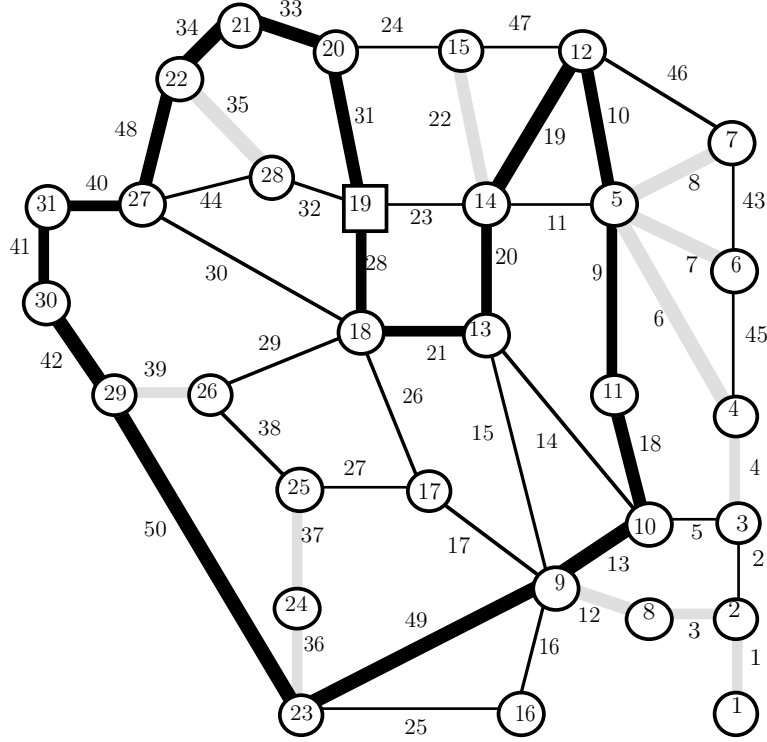


Figure 4.4:

no more chords are available for lifting.

Another improvement added to the routine goes as follows. Consider the example in Figure 4.4. As we have seen before, edge 3 is the unique edge that can be removed from the current cycle. At this point, the routine gets stuck since no cycle can be found whose weight is larger than 11 (the weight of the current cycle). When this is the case, the routine starts looking for more general subgraphs in G which are called *ears*.

An ear is defined constructively as follows:

- (i) a cycle is an ear;
- (ii) Let $(V(E_a), E_a)$ be an ear in G and $(V(P_{ij}), P_{ij})$ be a path in G such that $V(E_a) \cap V(P_{ij}) = \{i, j\}$ and $E_a \cap P_{ij} = \emptyset$. Then $(V(E_a \cup P_{ij}), E_a \cup P_{ij})$ is an ear in G .

From the definition above, one can see that an ear is at least a 2-connected graph. Thus, if $\sum_{u \in V(E_a)} w_u > W$, the inequality $x(E_a) \geq 2$ is valid for the polytopes we consider here. In general, such an inequality is not facet defining and its strength increases as the graph becomes sparser.

During the execution of the separation routine, suppose that a cycle is available at the end of the third step for which the corresponding inequality is not valid. We use this cycle as the initial ear and at each succeeding iteration a new ear is created by appending a path to the current one. Since the weight of the ear increases at each iteration, we continue iterating until a cover is found.

The path P_{ij} that is added to the current ear at each iteration is found in the following way. Let $(V(E_a), E_a)$ be the current ear. Consider the nodes in $V(E_a)$ as roots and construct the rooted spanning forest $T(V(E_a))$ in the same way we did before for tree $T(r)$. In the case of Figure 4.4, the current ear is the cycle with nodes 2, 3, 4, 5, 12, 13, 14, 18, 17, 9, 8 and Figure 4.5 shows a possible configuration for $T(V(E_a))$.

Consider an edge e_I not in $T(V(E_a)) \cup E_a$ with one endnode in one subtree of the forest and the other endnode in a different subtree (or in a root node). Together with a subset of edges in $T(V(E_a))$, e_I forms a path $P(e_I)$ that can be added to the current ear so as to obtain a new one. Only the ears that are supports of violated inequalities are of interest and, therefore, only the edges e_I such that $x(E_a) + x(P(e_I)) < 2$ are taken into account. The strategy used here to construct the new ear is the same used as that for the cycles in the sense that: if there are new ears that are covers then the one with the smallest weight is taken, otherwise no possible new ear is a cover and the heaviest one is taken.

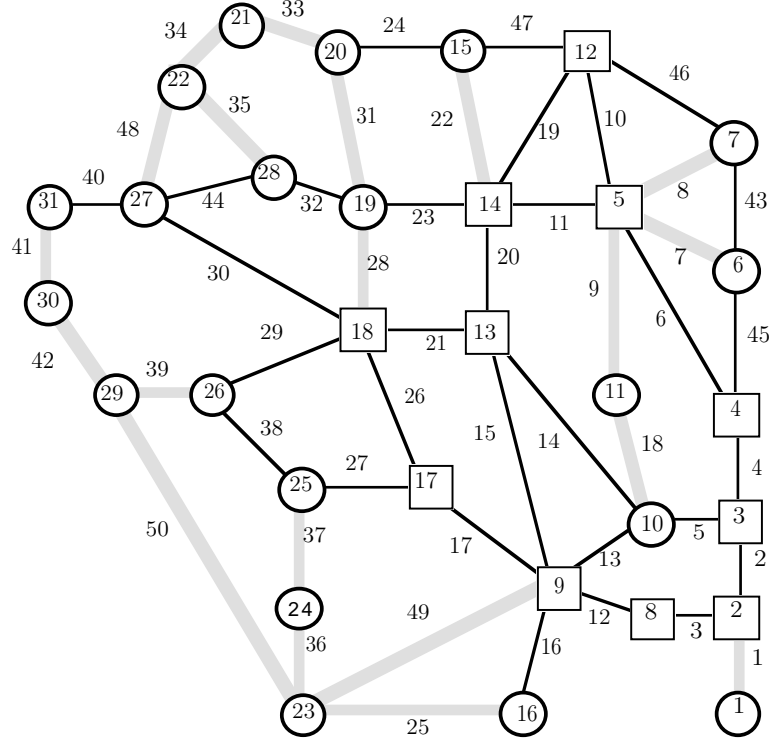


Figure 4.5:

4.2.2 Separation Routine for the PBC Inequalities

As mentioned before, this routine looks for PBCs in G composed of 2 cycles. Thus, the PBC inequalities generated are of the type $x(C_1) + x(C_2) \geq 4$ and the edges in the support can be classified into two sets according to their coefficients. We define a 2-path as a maximal set of consecutive edges in C_1 that are also in C_2 . The paths obtained in C_1 and C_2 by removing all the edges that are in 2-paths are called 1-paths. Clearly, the coefficients in the PBC inequality are 2 for edges in 2-paths, 1 for edges in 1-paths and 0 otherwise.

From the definitions of Chapter 2 and 3, \overline{Q} is the maximum of the sum of the node weights among all 1-paths. The validity of the PBC inequality is achieved when $\sum_{u \in V(C_1 \cup C_2)} w_u > W + \overline{Q}$. Note that, when C_1 and C_2 coincide, $\overline{Q} = 0$ and the validity of the inequality implies that C_1 (or C_2) is a cover cycle.

The routine starts with a cycle $C_1 = C_2$ satisfying $x^*(C_1) = x^*(C_2) < 2 - \epsilon$ where ϵ is a small positive number (taken as .20 in our experiments). Usually the inequality $x(C_1) + x(C_2) = 2x(C_1) \geq 4$ is not valid because C_1 is not a cover. The idea of the routine is to keep the cycle C_1 unchanged, while cycle C_2 is changed so as to achieve validity of the PBC inequality. Thus, suppose that two nodes u and v in $V(C_1)$ are in a same 2-path P_{uv} of the current PBC and that there exists a path \tilde{P}_{uv} with no edges in $C_1 \cup C_2$ and such that $x(C_1) + x(C_2) - x(P_{uv}) + x(\tilde{P}_{uv}) < 4$. Then, C_2 can be updated with $C_2 \cup \tilde{P}_{uv} \setminus P_{uv}$. In fact, this change is allowed provided that the variation in the value of $\sum_{u \in V(C_1 \cup C_2)} w_u - \overline{Q}$ is nonnegative. This point will be addressed later.

We now discuss steps of the routine in more detail. The routine is run for every node in G with degree at least 3. These nodes form the set of possible roots. Once the root node r is fixed, the **first step** involves the construction of the initial cycle C_1 and for this we make use of the algorithm described in the previous subsection. At this stage, \overline{Q} is zero.

The **second step** constructs the first block of the PBC. We want the first block to be nondegenerate and to have the root r as one of its ends. This is the reason why we only select nodes with degree larger than 2 to be the root node.

So let $T(r)$ be the last spanning tree coming from the cycle routine. For all nodes u adjacent to r in the subgraph $(V, E \setminus T(r))$, compute the value of x_{uv}^* plus the length of the path going from u to r in $T(r)$. Let v be the node for which the minimum of such values is achieved. Denote by w the first node visited in the cycle C_1 when we go from node v to node r using the edges in $T(r)$. Moreover, let $P(r, w)$ denote the path of minimum weight in C_1 that goes from r to w . We update cycle C_2 by removing the edges in $P(r, w)$, adding edge (rv) and adding the edges in the path going from v to t in $T(r)$.

We now illustrate the construction of the first block of the PBC. In Figure 4.1, suppose that node 19 is the root and that the cycle separation routine has found the cycle shown in Figure 4.6, where we also indicate the edges that are in the last tree found in the cycle

separation routine. According to the notation in the preceding paragraph, if v is node 28, then w is node 22 and, since the node weights are all equal to one, $P(r, w)$ is formed by the edges 31, 33 and 34. Figure 4.7 shows the new PBC obtained in this way. It satisfies $\sum_{u \in V(C_1 \cup C_2)} w_u = 15$ and $\overline{Q} = 2$. In the new inequality, the coefficients of the edges in the first block are 1, while those of the edges in the unique 2-path of the PBC are 2 (indicated in Figure 4.7 by the double thick black lines).

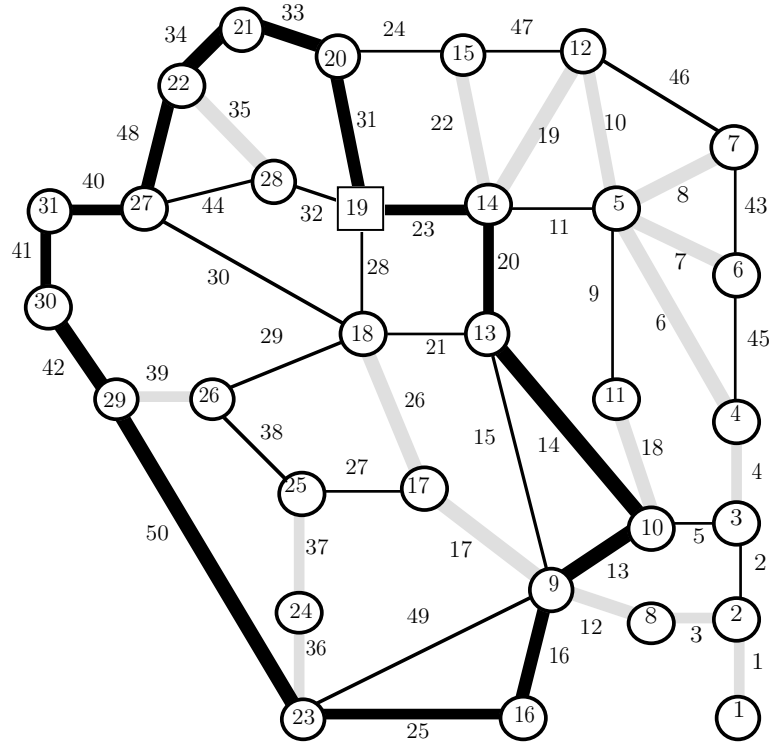


Figure 4.6:

The routine stops if it fails to produce the first block. Otherwise, if the current PBC is not the support of a valid inequality, a **third step** is executed that adds new blocks to the PBC. New blocks are added as many times as necessary until a valid PBC inequality is found. Figure 4.7 is used to show how a new block is constructed.

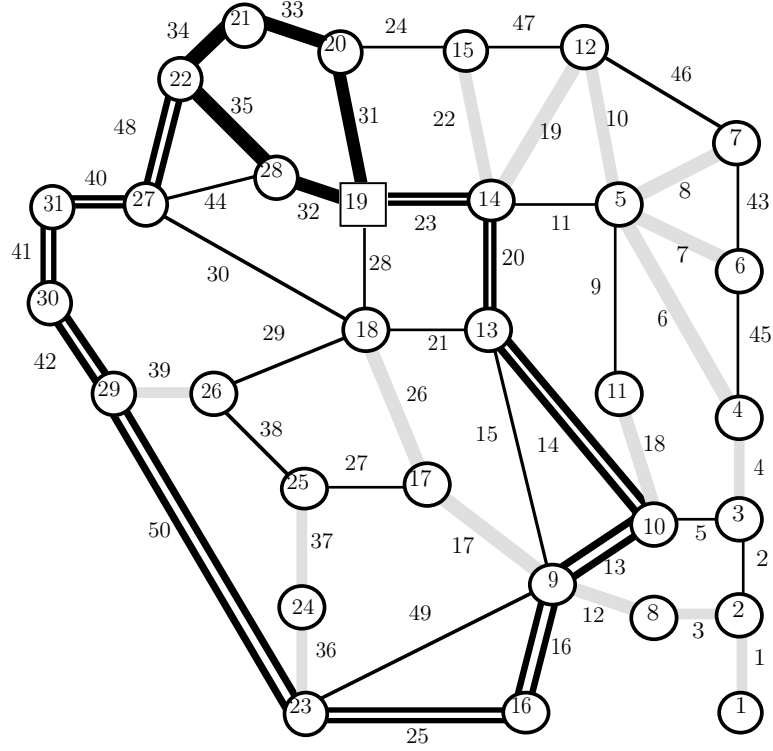


Figure 4.7:

We first find a rooted spanning forest T of G such that all nodes of the PBC are roots and those that are not in any 2-path form singletons in T (for instance, node 20 in Figure 4.7). Forest T is constructed by the usual greedy strategy.

Secondly, we find the set S of edges which are candidates to be in a 1-path of the new PBC. In principle, e is a candidate edge if there is a 2-path in the current PBC containing two nodes u and v such that there exists a path from u to v in the subgraph $(V, T \cup \{e\})$. Thus, an edge e satisfying one of the conditions below is discarded S :

- both endnodes of e are in the current PBC. Example: edges 15, 44 and 49 in Figure 4.7.
- both endnodes of e are in the same subtree of T . Example: edges 43, 45 and 46 in Figure 4.7.

- the endnodes of e are in different subtrees of T but these subtrees have the same root node. Example: edge 47 in Figure 4.7.
- the endnodes of e are in different subtrees of T with different roots but the roots are in different 2-paths of the current PBC. Example: edge 21 in Figure 4.8.
- one endnode of e is internal to some 1-path of the current PBC. Example: edges 24 and 44 in Figure 4.7.
- one endnode of e is the root of the subtree that contains the other endnode. Example: edge 11 in Figure 4.7.

Note that, if e satisfies one of the last five conditions above, then e cannot be used to form a 1-path of a new PBC. This remains true if e satisfies the first of these conditions and e is not a chord of a 2-path in the current PBC. When e is a chord of a 2-path, it could be used to form a 1-path of the new PBC but these edges are preserved to be used in the lifting step.

In Figure 4.7, considering the conditions stated above, S is contained in the set of edges given by $\{2, 5, 9, 21, 27, 28, 29, 30, 38\}$. The current PBC satisfies $\sum_{u \in V(C_1 \cup C_2)} w_u = 15$ and $\overline{Q} = 2$. Therefore

$$\sum_{u \in V(C_1 \cup C_2)} w_u - \overline{Q} = 13 < W = 16$$

and, if the value of \overline{Q} remains unchanged, the weight of the PBC must be increased to 19 to have a valid inequality (recall that the node weights and the cluster capacity are integers). Suppose that edge 30 is used to form a new block in the PBC. Two 1-paths are created: the path with edges (30, 26, 17) and the path with edges (16, 25, 50, 42, 41, 40). The new PBC has weight 17 and \overline{Q} is 5. The value of $\sum_{u \in V(C_1 \cup C_2)} w_u - \overline{Q}$ decreases by one unity comparing to the previous PBC. In some sense, we have moved away from validity since this value has to be larger than W in a valid PBC inequality. Thus, the edges for which the difference between the new PBC weight and \overline{Q} decreases are also removed from S . In the example of Figure 4.7, edge 30 is the only such edge.

Every edge e in S , together with some edges of the forest T , constitutes a path $P(e)$ joining two nodes u and v that are in a same 2-path of the PBC. Define $P(u, v)$ to be the set of edges between nodes u and v in this 2-path. If S is empty, the routine does not produce any violated inequality. Otherwise, let e^* be the edge that minimizes $\alpha(e) = x(C_1) + x(C_2) + x(P(e)) - x(P(u, v))$ for all edges e in S . If $\alpha(e^*)$ is less than 4, then the new PBC is accepted, and otherwise the routine fails to produce a violated inequality.

In the example of Figure 4.7, suppose that edge 27 corresponds to e^* . The new PBC is shown in Figure 4.8. It has weight 18 and \overline{Q} is 3. Therefore, validity is not reached and we have to go for another iteration. With the spanning forest shown in Figure 4.8, edge 29 can be seen to be in S . Suppose that edge 29 is selected to form the new PBC shown in Figure 4.9 and which has weight 20. Now, since $\overline{Q} = 3$, validity is reached and a violated PBC inequality is available.

Two different types of lifting have been implemented in the **fourth step** of the separation routine. In the first, assume that $\sum_{u \in V(C_1 \cup C_2)} w_u > W + \overline{Q} + \alpha$ for some number $\alpha > 0$. Let (z, v) be a chord of a 2-path in the current PBC and $P(z, v)$ be the edges between z and v in this 2-path. If $\sum_{u \in P(z, v)} w_u - \alpha \geq 0$, then both cycles C_1 and C_2 are changed by removing the edges in $P(z, v)$ and adding the edge (z, v) .

In the second lifting, we also deal with chords of 2-paths in the PBC. However, in contrast with the first case, the resulting PBC contains the same nodes as the previous one. Let (z, v) and $P(z, v)$ be defined as before. If $\sum_{u \in P(z, v)} w_u \leq \overline{Q}$, edge (z, v) can be used to create a new block in the PBC. For this, we keep cycle C_1 unchanged and we change cycle C_2 by removing the edges in $P(z, v)$ and adding edge (z, v) . In the example of Figure 4.9, edge 15 is a chord that allows us to make such a lifting.

Other liftings exist that have not been implemented. These liftings involve the change of one or more existing blocks of the PBC. As an illustration, consider again the example in

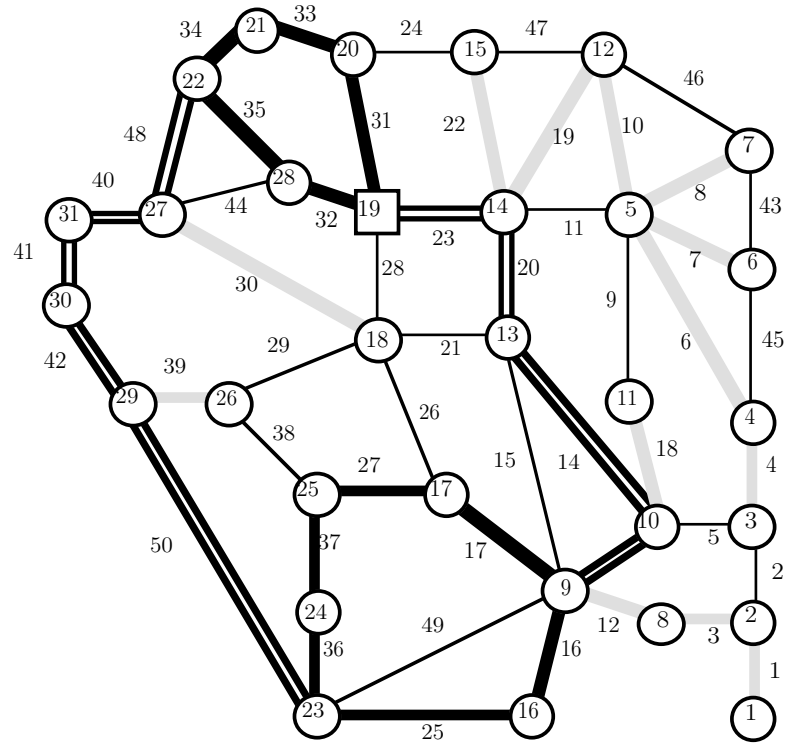


Figure 4.8:

Figure 4.9. The coefficients of edges 35, 44 and 48 in the current inequality are respectively 1, 0 and 2. This inequality can be easily seen to be dominated by the inequality where these coefficients are changed to 0, 1 and 1 respectively, while the coefficients of the remaining variables remain unchanged.

the inequality $2x(T \setminus C) + x(C) \geq 2$ is valid and that all points satisfying $2x(T) = 2$ also satisfy $2x(T \setminus C) + x(C) = 2$, although the inverse is not true.

To illustrate the contents of the paragraph above, consider the example of Figure 4.10 where the tree T rooted at node 18 is shown. Edge 37 is not in T and together with edges 17, 19, 26, 29, 36 and 38 it forms the cycle C . Any solution satisfying $2x(T) = 2$ and not intersecting an edge of $T \setminus C$ also satisfies $2x(T \setminus C) + x(C) = 2$. If the solution intersects an edge in $T \cap C$, then edge 37 must be intersected.

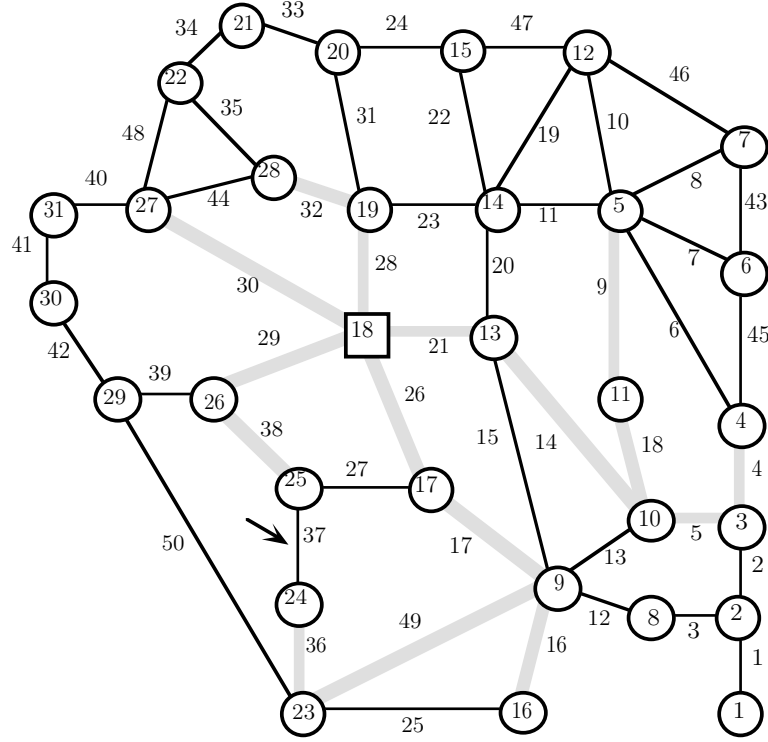


Figure 4.10:

The routine continues by transforming all nodes in $V(C)$ into root nodes and removing all edges in C from T . By doing this, T becomes a forest. There are tree types of root nodes in forest T :

- those that are singletons (nodes 23, 24, 25, 26 and 17 in Figure 4.10).

- those that are roots of a single subtree of T (node 9 in Figure 4.10).
- those that are roots of multiple subtrees of T (node 18 in Figure 4.10).

This is typically the situation at the beginning of a new iteration of the algorithm. At each iteration, we try to obtain a stronger inequality by finding a cycle C in the forest T that contains one of the root nodes. In terms of the inequality, this corresponds to reducing the coefficients of the edges in $C \cap T$ from 2 to 1 and to increasing the coefficient of the edge that induces the cycle from 0 to 1. It is an easy task to prove that the resulting inequality is valid and that the face it defines in the polytope contains the face defined by the preceding inequality. The nodes in C become roots and the routine goes for a new iteration. This is repeated until no further cycle is found.

In the example of Figure 4.10, edge 44 creates a new cycle and the nodes in this cycle (19, 28 and 27) become roots of the new tree T . This operation is shown in Figure 4.11.

Assume that r_u and r_v are respectively the roots of the subtrees containing nodes u and v where the edge (u, v) is in $E \setminus T$. If the edge (r_u, r_v) is in the support graph of the current inequality, then let C be the cycle induced by (u, v) and (r_u, r_v) in T . Consider the new inequality obtained from the current one by increasing the coefficient of edge (u, v) from 0 to 1, decreasing the coefficients of edge (r_u, r_v) from 1 to 0 and decreasing the coefficient of the remaining edges of C from 2 to 1. This inequality can be easily seen to be valid and stronger than the original one. For instance, consider the example of Figure 4.11. If (u, v) is the edge 25, then (r_u, r_v) is the edge 49. The support graph of the new inequality is represented in Figure 4.12.

The liftings given so far only involve nonforest edges that join different subtrees. After some iterations, no more such liftings are possible. This is the case in Figure 4.12 where there is only one subtree in T (the one rooted at node 18) that is not a singleton. Further lifting is still possible using the procedure explained below.

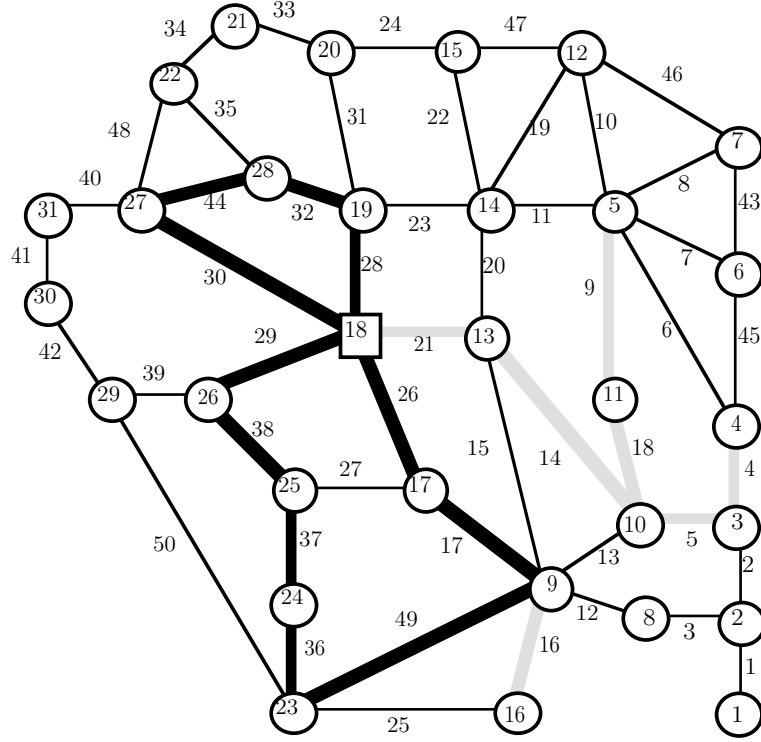


Figure 4.11:

First visit all root nodes which do not form singletons and mark those that are roots of a single subtree. In Figure 4.12 only node 18 is marked. For each marked node r and starting from the edge incident with r , the edges of the corresponding subtree are traversed. For every edge that is traversed, the endnode not in the root set becomes a root. This is repeated until one of the new root nodes has degree larger than 2 in T . The idea is to alternate a lifting phase with an expansion phase that enlarges the set of root nodes. We continue with the example of Figure 4.12 to see how this works.

In Figure 4.12, the subtree rooted at node 18 is traversed. The first edge traversed is the edge 21 and node 13 becomes a root. The next edge traversed is edge 14 and node 10 becomes a root. Since node 10 has degree 3 in T the routine stops expanding the set of root nodes. The lifting phase uses edge 6 as in Figure 4.13 (the coefficients of the edges 4, 5, 9 and 18 decrease to 1 and the one of edge 6 increases to 1). Since no further lifting

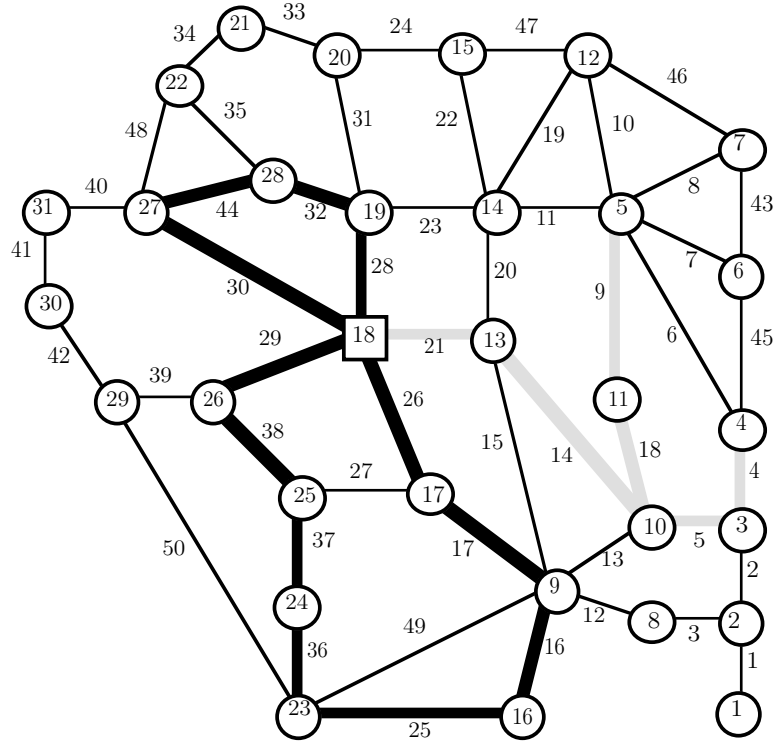


Figure 4.12:

is found, the routine switches to the expansion of the set of roots. However, since all root nodes are singletons, no expansion is possible and, in this is the case, the routine stops.

Unfortunately, the routine does not capture some other type of liftings. For example, if a solution is on the face defined by the support graph represented in Figure 4.13 and it cuts only edge 21, then edges 15 and 13 are cut. On the other hand, if the solution cuts only edge 14, then either edge 13 or edge 15 is cut. Consequently, a stronger inequality can be obtained from the current one by changing the coefficients of edges 21, 14, 15 and 13 respectively from 2, 2, 2 and 0 to 0, 1, 1 and 1, while keeping the coefficients of the remaining variables unchanged. This new inequality can be proved to be facet defining for the equipartition polytope of the graph shown in Figure 4.13.

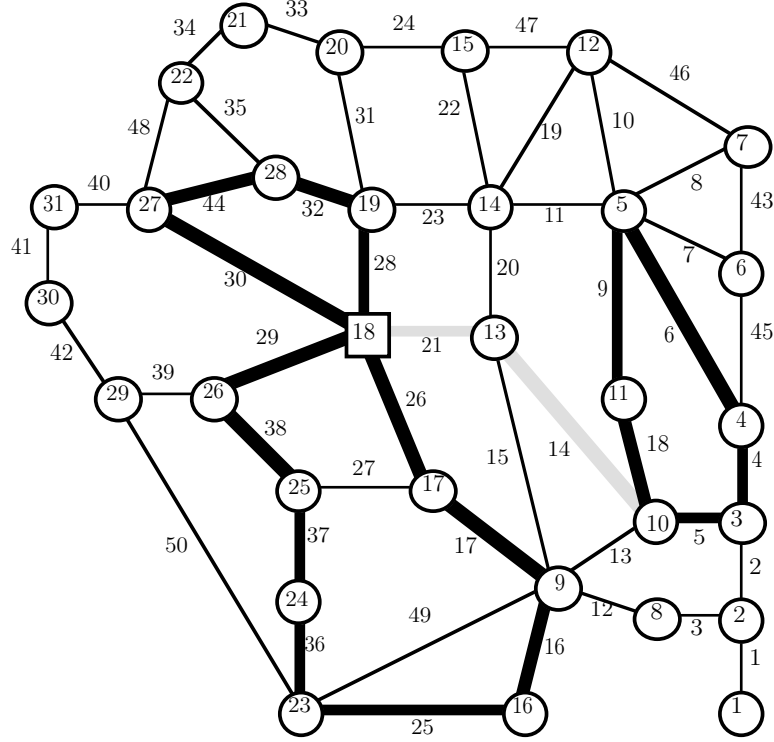


Figure 4.13:

4.2.4 Separation Routine for the Knapsack Tree Inequalities

We now describe the separation routine that has been implemented by Ferreira, Martin and Weismantel (1993) to identify violated knapsack tree inequalities.

Let $(V(T), T)$ be a tree in G rooted at node r and consider the knapsack polytope P_{Kn} given by:

$$P_{Kn} = \{y \in \mathbb{B}_+^n : \sum_{u \in V} w_u y_u \leq W\}$$

If $\pi y \leq \pi_0$ is a valid inequality for P_{Kn} such that $V(\pi) = \{u \in V : \pi_u > 0\}$ and $V(\pi) \subseteq V(T)$, then the knapsack tree inequality is written as:

$$\sum_{u \in V(\pi)} \pi_u (1 - \sum_{e \in P(u)} x_e) \leq \pi_0$$

where $P(u)$ is the path in T that joins node u to the root node r for all $u \in V(\pi)$. Given a fractional solution x^* , it is easy to see that, if the nodes u in T such that $\sum_{e \in P(u)} x_e^* \geq 1$

are removed from T , then the new knapsack tree inequality is still valid and its violation is larger than the violation of the starting inequality.

Thus, our separation routine is divided into two main steps. In the **first step**, we try to find a good choice for the tree T and, in the **second step**, we look for the coefficients π_u for the nodes $u \in T$ which give rise to a valid knapsack tree inequality that is violated. The choice of a good tree T depends on the root node and, since every node in turn is considered as the root, the separation routine repeats these two steps n times.

Let us consider initially the problem of finding a good choice for tree T and, for this, suppose that a node r has been fixed to be the root. A tree is considered to be a good choice if we are likely to find a knapsack tree inequality that has this tree as its support. We have seen that the smaller the values of the terms $1 - \sum_{e \in P(u)} x_e$, the higher is the violation (if so) of the inequality. Now with edge distances x_e , $\sum_{e \in P(u)} x_e$ is the distance in the tree from the root r to node u . Therefore, it is reasonable to construct a tree T rooted at r such that, the distance from r to any other node in T is minimum. So, the separation routine starts by constructing the shortest path tree and, for this, Dijkstra's algorithm is used.

As discussed earlier, any node in T whose distance to r is at least 1 can be deleted from T since this will increase the chance of finding a violated inequality. After doing these node deletions, the separation routine goes to the second step in which the coefficients π_u in the knapsack tree inequality are computed.

Initially, define variables y_u for all $u \in V(T)$ as follows:

$$y_u = \begin{cases} 1 & \text{if } u \text{ and } r \text{ are in the same cluster} \\ 0 & \text{otherwise} \end{cases}$$

Now, the knapsack polytope P_{Kn} is given by $P_{Kn} = \{y \in B^{|V(T)|} : \sum_{u \in V(T)} w_u y_u \leq W\}$.

By construction of the tree T , we have that $0 \leq \sum_{e \in P(u)} x_e^* \leq 1$ for all $u \in V(T)$. Thus, a fractional point $y^* \in \mathbb{B}^{|V(T)|}$ can be defined with components given by $y_u^* = 1 - \sum_{e \in P(u)} x_e^*$, which clearly satisfy $0 \leq y_u^* \leq 1$. Finally, using the separation routines for the knapsack polytope, it can be checked if there is an inequality $\mu y \leq \mu_0$ valid with respect to P_{Kn} such that $\mu y^* > \mu_0$. If this is the case, we are done because, by taken $\pi = \mu$ and $\pi_0 = \mu_0$, a violated knapsack tree inequality for $P_{MC}^{K,W}(G)$ is obtained since:

$$\begin{aligned} \sum_{u \in V(T)} \mu_u y_u^* &> \mu_0 \Leftrightarrow \\ \sum_{u \in V(T)} \pi_u y_u^* &> \pi_0 \Leftrightarrow \\ \sum_{u \in V(T)} \pi_u (1 - \sum_{e \in P(u)} x_e^*) &> \pi_0 \end{aligned}$$

The second step of the separation routine terminates by tightening the inequality $\sum_{u \in V(T)} \pi_u (1 - \sum_{e \in P(u)} x_e^*) \leq \pi_0$ as described in Proposition 3.13.

The number of knapsack tree inequalities generated for a given root node depends on the number of violated inequalities that are found for the knapsack polytope P_{Kn} .

Computational Complexity of the Separation Routines

The main purpose of implementing these separation routines was to test the strength of the inequalities introduced in the thesis. There was no major concern about the efficiency of the routines in terms of computational complexity. Nevertheless, a brief discussion about the complexity of the separation routines we have implemented is presented below. In the analysis that follows m is the size of the edge set E and n is the size of the node set V .

Consider initially the cycle separation routine and the step of the routine in which a heavier cycle is obtained. For a fixed edge e_D in the current cycle C , a new tree $T(r, e_D)$ is defined. To find the best cycle induced in $T(r, e_D)$ (in the sense described in Subsection 4.2.1), all edges in $E \setminus T(r, e_D)$ have to be tested which gives a complexity of $O(m)$. These

operations are repeated for all edges e_D in C and the maximum size of C is n , then the overall complexity of this step is $O(mn)$.

The difficulty here is to determine how many times a new cycle has to be obtained in the worst case. If the node weights and the capacity W are all positive integers a trivial bound for this value is W because the weight of C strictly increases from one iteration to another (otherwise the routine stops). Therefore, in the worst case, the complexity of the routine is $O(Wmn^2)$ since this step can be seen to dominate the remaining steps of the algorithm and it is repeated once for each root node. Tests run for some examples of our sample indicate that the number of new cycles generated is usually small (less than 10) and independent of the size of the graph. So, the expected CPU time of the cycle routine is $O(mn^2)$.

It can be observed that this complexity also holds for the PBC separation routine. At most n new blocks can be created by the algorithm because the number of such blocks is bounded by the number of edges in the initial cycle C_1 . The same argument also shows that the number of liftings is bounded by n . The creation of a new block and the lifting involve the visiting of all edges in the graph and therefore a complexity of $O(nm)$ is achieved for every root node fixed in V . Therefore, the complexity of these steps are dominated by the complexity of constructing the initial cycle C_1 which, as discussed above, is $O(Wmn)$. When the routine is executed for all root nodes, the complexity goes to $O(Wmn^2)$ as before.

In the tree separation routine, there are two phases: the lifting phase and the root set expansion phase. Every lifting phase is $O(m)$ because it implies the visiting of all edges not in the current support graph. The number of lifting phases that are executed is bounded by the number of expansion phases that are executed. In the worst case, every expansion phase adds a new node to the set of roots and this yields a complexity of $O(mn)$ for every choice of the initial root. Thus, the overall complexity of the routine is $O(mn^2)$.

4.3 About the Branch-and-Cut Code

The branch-and-cut code we have used was designed and implemented by Ferreira, Martin and Weismantel (1992). Their code, called CLOP (for CLustering OPTimization), has been made available to us and is particularly interesting for our purposes since it provides an easy way to insert new separation routines. This has given us the possibility both to test the strength of various families of valid inequalities, and to evaluate the performance of our separation routines.

Therefore, our contribution has been to add to CLOP the implementations of the separation routines for cycle, PBC and tree inequalities as described in Section 4.2. CLOP is written in C and runs on Sun Sparc Workstations. The Linear Programming package used is CPLEX (1990).

The goal of this section is to give a few details about CLOP that will be helpful in analysing the computational results. A detailed discussion about the CLOP code and the implementation issues will appear in forthcoming work of Ferreira, Martin and Weismantel.

Problem Formulation

As discussed in Chapter 2 the node-edge model is a suitable formulation of the graph partitioning problem for computational purposes. This model, given in Section 1.3 and reproduced below, is used in CLOP.

Variables:

$$x_{uv} = \begin{cases} 1 & \text{if edge } (u, v) \text{ is in the multicut defined by the partition} \\ & \text{(that is, nodes } u \text{ and } v \text{ are in different clusters)} \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_u^k = \begin{cases} 1 & \text{if node } u \text{ belongs to the } k\text{-th cluster of the partition} \\ 0 & \text{otherwise} \end{cases}$$

IP formulation:

$$\begin{aligned}
& \min && \sum_{e \in E} c_e x_e \\
& \text{Subject to} && \sum_{u \in V} w_u y_u^k \leq F_k \quad \forall k \in \{1, \dots, K\} & (I) \\
& && \sum_{k=1}^K y_u^k = 1 \quad \forall u \in V & (II) \\
& && y_u^k + y_v^\ell - x_{uv} \leq 1 \quad \forall k \neq \ell \in \{1, \dots, K\} \\
& && \quad \quad \quad \forall (u, v) \in E & (III) \\
& && y_u^k + y_v^k + x_{uv} \leq 2 \quad \forall k \in \{1, \dots, K\} \\
& && \quad \quad \quad \forall (u, v) \in E & (IV) \\
& && y_u^k \in \{0, 1\} \quad \forall u \in V, \forall k \in \{1, \dots, K\} & (V) \\
& && x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E & (VI)
\end{aligned}$$

Constraints (III) and (IV) are initially omitted from the formulation. For nonnegative edge costs, constraints (IV) can be eliminated from the formulation since we have a minimization problem. Constraints (III) are added as they are needed by means of a polynomial time separation routine. The idea is to avoid unnecessarily large LPs. In fact, constraints (III) can be strengthened and the inequalities that are added in CLOP are not of the form appearing in the formulation above. Instead, they are added in the strengthened form given below:

$$\sum_{k \in S} y_u^k + \sum_{k \notin S} y_v^k - x_{uv} \leq 1$$

where $S \subset \{1, \dots, K\}$.

The program offers the possibility of choosing different strategies for separating valid inequalities. Separation routines are available for several classes of valid inequalities. Besides the separation routines for cycle, PBC and tree inequalities that we have implemented, other separation routines are available such as the ones for knapsack tree and star inequalities which have been implemented by Ferreira, Martin and Weismantel (1993). The star inequality is a particular case of the knapsack tree inequalities which we describe briefly in the next paragraph.

Consider the set $N(v)$ of nodes adjacent to a given node v in graph G and suppose that $\sum_{u \in N(v)} w_u + w_v > W$ (v and its adjacent nodes form a cover). The simple tree inequality $\sum_{e \in T} x_e \geq 1$, where $T = \delta(v)$, is valid but it is possible to strengthen this inequality in the following way. Let $N'(v)$ be the largest set in $N(v)$ whose nodes fit together with v in a cluster. Then, a stronger inequality is obtained by increasing the current right-hand side to $|N(v) \setminus N'(v)|$. Sometimes, additional changes are possible that makes the inequality stronger. In these changes, either an edge is removed from the star, or both the coefficient of an edge and the right-hand side of the inequality are increased by the same amount. A separation routine has been implemented in CLOP (Weismantel, 1993) that looks for violated star inequalities.

Upper Bounds

As the branch-and-cut algorithm runs, the lower bound for the optimal integer solution provided by the current optimal fractional solution increases. Heuristics for obtaining good upper bounds for the optimal integer solution are also available. The idea of these heuristics is to find a feasible solution for the problem from the information provided by the current LP (fractional) solution. The existence of a good upper bounds allows us to fix some variables to their integer values (see Nemhauser and Wolsey, 1988) which, in turn, may allow us to terminate the computation earlier. This is the case, for instance, when we have an integer cost function and the difference between the lower and the upper bounds is less than one.

Two such heuristics are available in CLOP. The first heuristic (Ferreira, 1993) uses only the information provided by the edge variables to generate an upper bound. We call it the *EDGE* heuristic and it works as follows.

Suppose that x^* is the current LP fractional solution. The rationale behind the *EDGE* heuristic is based upon the fact that, if x_{ij}^* is close to 0, the nodes i and j will probably be together in a same cluster of the partition. Thus, given a positive (small) value ϵ , let E' be the set of edges in E satisfying $x_e^* < \epsilon$. Consider now the problem of finding

a minimum cost forest T in (V, E') such that: (i) the number of components in T is at most K (the size of a feasible partition) and (ii) the sum of the node weights in each component in T is at most W (the cluster capacities). Clearly, any solution of this problem immediately converts into a solution of the graph partitioning problem by making a one-to-one correspondence between the components of the forest and the clusters in the partition.

Therefore, a modified version of Kruskal's algorithm for the minimum spanning tree problem is used to find the forest T . The aim of these modifications is to avoid the creation of components for which the sum of the node weights is larger than W . This modified version of Kruskal's algorithm may terminate with a forest that has more than K components. If the forest has at most K components, a feasible solution for the graph partitioning problem is available. On the other hand, if the number of components in T exceeds K , then the EDGE heuristic looks for a feasible solution for the bin packing problem in which the components in T are interpreted as items that have to be packed into bins (cluster) of size W . If a solution is found for the bin packing problem with cost at most K , then a feasible solution for the partitioning problem is available. Otherwise, the heuristic fails to produce a feasible partition of G and is unable to give an upper bound.

The second heuristic (Martin, 1993) uses the node variable information to find an upper bound and it is called the *RANDOM* heuristic. It is based upon a heuristic for the Unconstrained Global Routing Problem in VLSI design (see Lengauer, 1990). Since we have the equality constraints $\sum_{k=1}^K y_u^k = 1$ in our model, we can interpret the y_u^k variables as a probability distribution for node u . For instance, for an equipartition problem, suppose that we have a fractional solution such that $y_u^1 = .6$ and $y_u^2 = .4$. In this case, we assume that node u has 60% probability of being in cluster 1 and 40% of being in cluster 2.

The heuristic simulates a series of experiments in which the nodes are assigned to the clusters according to these probability distributions and, among the solutions produced by

the experiments, it picks the feasible solution of minimum cost as the upper bound. At the beginning of an experiment, the n nodes of the graph are unassigned and the experiment consists in executing n iterations in the following way. First choose an unassigned node u in the graph and assign it to a cluster k with probability y_u^k . If the capacity remaining in cluster k is less than w_u , then a new trial has to be made for node u since this assignment is not possible. In the new trial, the probability distribution is adjusted so as to prevent node u from going to cluster k again. These operations are repeated until either there are no more clusters available in which u fits and the experiment is aborted, or a cluster is found that can accept node u and a new iteration executed.

Parameters

The maximum number of cuts added at each iteration is passed to the program as parameter. In tests we have carried out, this parameter has been set to 1000. In the current version of the code, this parameter has to be taken into account when choosing the order in which we separate the inequalities. For instance, suppose that the separation routines for cycles, trees and PBCs, in that order, have been chosen for generating violated inequalities in a given run. If the number of violated cycle and tree inequalities found in an iteration is larger than 1000, then no violated PBC inequality will be added in this iteration, even if one exists that can be found with the separation routine.

A second parameter passed to the program is the CPU time limit which works as follows. Suppose that the CPU time limit is set to 120 minutes. After the execution of every LP, the program checks if the CPU time of the run exceeds 120 minutes and, if this is the case, it stops. Note that, in that way, it is possible that the total CPU time of a given run passes the limit. In fact, the idea is to avoid CPU times that are too large, but is also to have the information provided by the LP relaxation that is being solved when the time limit expires.

Three other parameters are used in the program to decide when and how to branch in the enumeration tree. These parameters are used to evaluate the lower bound improve-

ment (LP progress) during the execution and if the latter is not considered to be large enough, the program selects a variable for branching. The current version of the code gives preference to branch on edge variables. A new branch is made if the LP progress after 10 (first parameter) iterations is less than 0.10 (second parameter) and the largest *violation ratio* among the cuts found by the separation routines in the last iteration is at most 0.05 (third parameter). Given a fractional solution x^* and an inequality $\pi x \leq \pi_0$, the *violation ratio* is computed as $|\frac{\pi_0 - \pi x^*}{\pi_0}|$ if π_0 is not null and, otherwise, it is computed as $\pi_0 - \pi x^*$.

Pool of Inequalities

The program keeps track of a pool of inequalities. Some inequalities that have been generated previously in the algorithm are stored in the pool. An inequality generated in an iteration can be dropped from the LP relaxation in some other iteration and, after that, become violated again. The pool of inequalities provides a cheap way of searching for violated inequalities since one can check very rapidly if a given inequality is violated.

4.4 Problem Instances

The algorithm has been tested on three different classes of problem. The characteristics of the test graphs are now discussed.

The Class of Mesh Problems

The first type of problem arises in finite element computations. This problem is described in more detail in the next chapter. For the present, we restrict ourselves to the following brief description of the problem.

Suppose that a planar region is given and that this region is partitioned into smaller pieces called the elements. Typically the elements are rectangles or triangles. The set of elements defines a mesh. There is a set of variables associated to every element and if two elements touch each other then they have variables in common. A system of linear equations is defined over these variables. Assume that K processors are available to solve this system. It is well-known that the amount of communication and the workload balance among the processors are crucial for the performance of parallel algorithms.

Assume that two elements touch each other only if they have a side in common and that the number of variables associated with one element is given by a constant. The first assumption is usually not true but, as we will see in the next chapter, it leads to a reasonable relaxation of the problem. So, take any partition of the mesh into K subsets of elements and assign each of these subsets to the different processors. The workload at each processor depends essentially on the number of elements (variables) it has to deal with, and the communication cost depends on the number of variables shared by elements in different sets of the partition.

Consider the dual graph G_M of a planar mesh M . Let $G = (V, E)$ be the partial subgraph of G_M induced by the nodes that correspond to an element of M (for more details see Chapter 5). Suppose that all edge costs are set to one. From what was stated in the previous paragraph, solving the mesh partitioning problem is equivalent to solving

the graph partitioning problem in G where the number of sets in the partition is K and the size of each of these sets is bounded by $\lceil \frac{|V|}{K} \rceil$. When $K = 2$, the mesh partitioning problem gives rise to a graph equipartition problem.

Another problem that can be defined on the graph G associated to a finite element mesh is the cutwidth problem. In Chapter 5, we will see that orderings with small cutwidth in G provide good solutions for the Frontwidth Reduction Problem of finite element meshes. The heuristic we propose in Chapter 5 for the cutwidth problem in G hierarchically solves equipartition problems in graphs obtained from G by contracting some of its edges.

Since the elements are rectangles or triangles in the meshes considered in our tests, the maximum degree of a node in the resulting graphs (unless some edges have been contracted) never exceeds 4. The graphs are usually sparse with density of about 2.5 on average and the graphs are 2-connected.

Note that the mesh problems are essentially graph partitioning problems with cardinality capacity constraints (see Section 3.3).

The Class of Compiler Design Problems

The second class of problems arise from compiler design problems (see, Johnson et al., 1992). Here the nodes are in one-to-one correspondence with the routines of a compiler, and the node weights represent the amount of memory used to store each routine. If two routines communicate with each other, there is an edge between the two corresponding nodes and a positive cost is associated to that edge. This is the cost of communication between the two routines if they are not in the same page of memory (the cost is assumed to be zero otherwise). So, the problem is to find a partition of the routines (nodes) into pages of memory (clusters) so that the total communication cost is minimized.

The node weights for graphs arising from compiler design problems vary considerably. In the instances of Johnson et al., there are typically two nodes of large weight and the

remaining nodes have small weights. None of the graphs are 2-connected and most of their edges are incident with one of the two nodes of large weight. The density of the graphs tested ranges from 1.56 to 3.07. No special structure is apparent in the edge costs.

The compiler design problems are then modeled as graph partitioning problems with knapsack cardinality constraints (see Section 3.4).

The Class of VLSI Design Problems

The third class of problems arises in VLSI placement design (Weismantel, 1992). Each node of the graph corresponds to a chip in the circuit and its weight is given by the area of the chip. An edge is placed between two nodes (chips) if the two chips are in a same net of the circuit. Its cost is given by the number of such nets. The circuit area is divided into small rectangles. Thus, the problem is to assign the chips (nodes) to the rectangles (clusters) such that the number of times the nets have to cross the boundary of the rectangles (sum of edge weights) is minimized. An assignment (partition) is feasible only if, for every rectangle (cluster), the sum of the chip areas (node weights) assigned to the rectangle does not exceed the rectangle area (cluster capacity).

The edge costs can be defined differently leading to different cost functions. For a detailed discussion on this topic, we refer to Weismantel (1992).

The variance in the node weights for the graphs arising from VLSI placement design is usually small. In fact the node weights typically take three to four different possible values. The graphs are a little denser than in the compiler design problems but are still sparse. Densities vary from 1.68 to 3.03. The edge costs also vary considerably and typically assume one of five different values. The graphs can be disconnected.

Like the compiler design problems, the VLSI design problems are modeled as graph partitioning problems with knapsack cardinality constraints.

The Test Problems

The following notation is used to identify the problem instances:

$$name.n.K.m$$

where: *name* is either *mesh*, or *cb450* or *vlsi* depending whether the problem belongs, respectively, to the mesh, compiler design or VLSI design class; *n* (*m*) is the number of nodes (edges) of the graph and *K* is the number of clusters allowed in the feasible partitions.

The test problems from each of the three classes of problems are presented below.

Mesh Problems

The list of the six mesh problems on which our experiments have been carried out is the following:

$$\begin{array}{lll} mesh.274.2.469 & mesh.148.2.265 & mesh.148.4.265 \\ mesh.138.2.232 & mesh.70.2.120 & mesh.31.2.50 \end{array}$$

The mesh corresponding to problem *mesh.274.2.469* is shown in Figure 4.14 (de Souza et al., 1992). Denote by G_{274} the graph associated to that mesh. The graph of problem *mesh.138.2.232* is the induced subgraph of G_{274} corresponding to the 138 leftmost elements of the mesh in Figure 4.14, while the graph of problem *mesh.70.2.120* is the subgraph arising from the contraction of the 205 rightmost nodes of G_{274} into a single node. The mesh corresponding to problems *mesh.148.2.265* and *mesh.148.4.265* is shown in Figure 4.15 and it is taken from Pina (1981). The graph of problem *mesh.31.2.50* is the one used in Section 4.2 to illustrate the execution of our separation routines (Figure 4.1). It corresponds to a modification of a mesh also taken from the paper of Pina.

Note that, since all cluster capacities for mesh problems are equal to $\lceil \frac{|V|}{K} \rceil$, and all node weights are 1, all mesh instances except *mesh.148.4.265*, are graph equipartition problems.

Figure 4.14:

Figure 4.15:

Compiler Design Problems

Six of the Compiler Design problems on which our experiments have been carried out are listed below:

cb450.30.6.47 *cb450.30.6.56* *cb450.45.8.98*
cb450.47.8.99 *cb450.47.9.101* *cb450.61.9.187*

These examples have been taken from Johnson et al. (1991). Figure 4.16 shows the graph for the problem *cb450.30.6.47*.

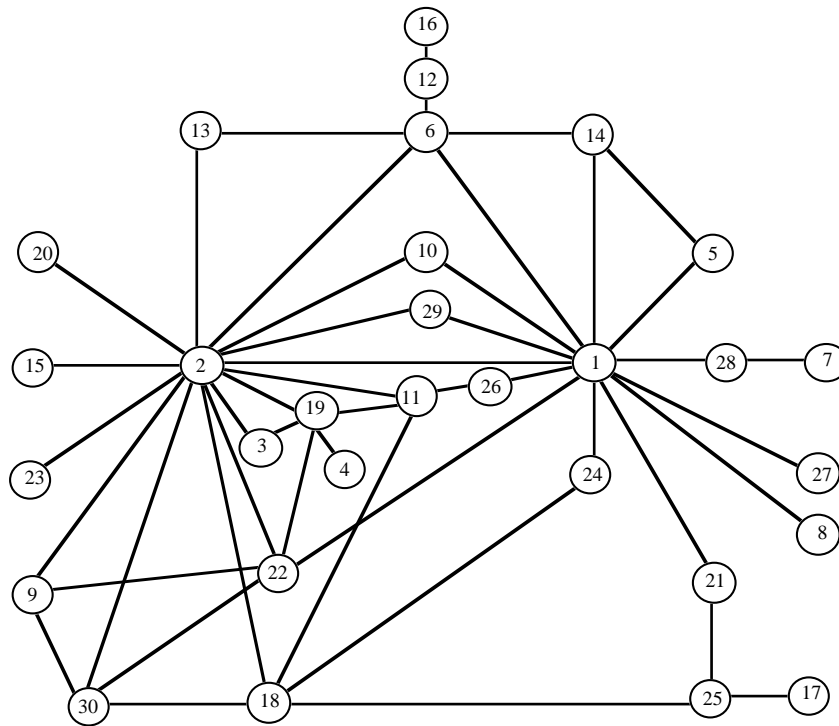


Figure 4.16:

In the compiler design problems, all cluster capacities (memory page sizes) are equal to 450 (kilobytes). We have also modified the data for the smallest five of these problems so as to convert them into bipartition problems. The capacities of the two clusters in the bipartition versions of the problem have been adapted so that in a feasible solution the weights of the two clusters are approximately the same. The aim is to have a "weighted"

version of the equipartition problem. The problem instances generated in this way are denoted by:

$$\begin{array}{l} weq.30.6.47 \quad weq.30.6.56 \quad weq.45.8.98 \\ weq.47.8.99 \quad weq.47.9.101 \end{array}$$

The interest of studying such bipartition problems comes from the fact that, in practice (see, for instance, Lengauer, 1990) solutions for partitioning problems are often given by heuristics based upon a divide-and-conquer strategy that hierarchically split the graph into two (balanced) parts. The point here is to test if our inequalities are more effective in solving the bipartition problems arising from the above approach, or in solving the original partition problem itself.

VLSI Design Problems

The list of the six VLSI problems used in our experiments is given below.

$$\begin{array}{l} vlsi.166.4.504 \quad vlsi.37.4.92 \quad vlsi.38.4.105 \\ vlsi.43.4.105 \quad vlsi.48.4.81 \quad vlsi.17.4.39 \end{array}$$

The data for these problems have been given to us by the developers of CLOP. Figure 4.17 shows the graph for the problem *vlsi.48.4.81*. The graphs in problems *vlsi.37.4.92*, *vlsi.38.4.105*, *vlsi.43.4.105* and *vlsi.48.4.81* have been obtained by splitting the graph of problem *vlsi.166.4.504*.

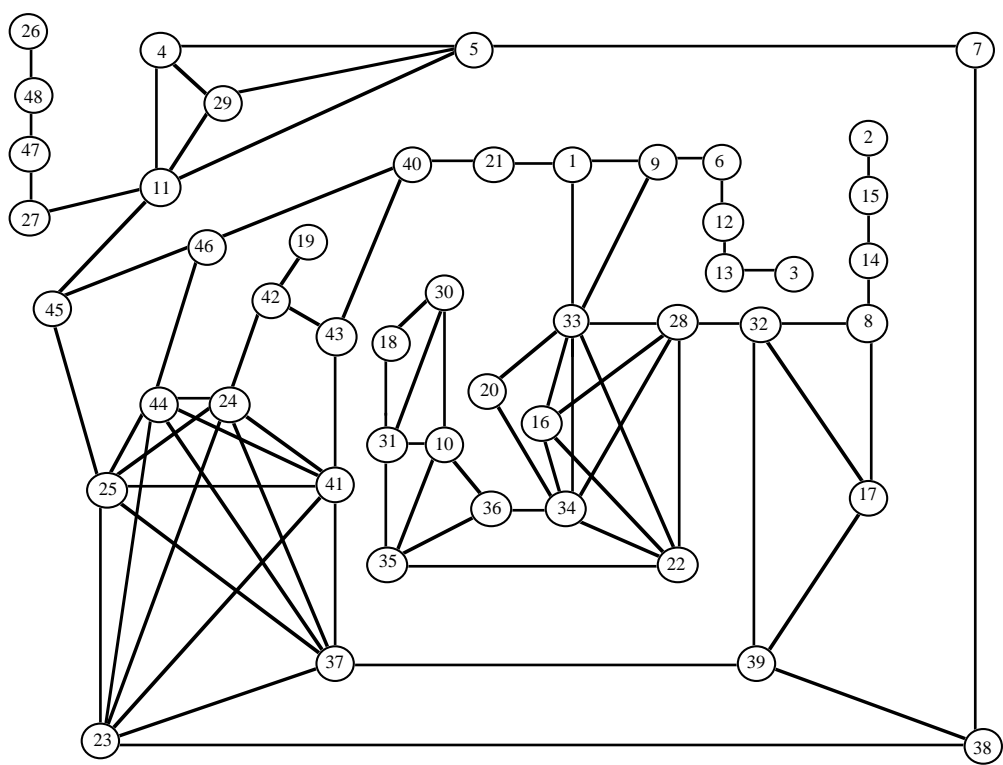


Figure 4.17:

4.5 Computational Results and Discussion

Below we present the computational results that have been obtained by CLOP.

In the tables with the computational results, the following notation is used to characterize the different separation routines: **P** for the PBC inequalities; **C** for the cycle and tree inequalities; **K** for the knapsack tree inequalities; **S** for the star inequalities and **E** for the edge inequalities (see the comments following the model given in Section 4.4).

Two types of tables summarize the computational results for each of the three classes of problems tested. The meanings of the columns in these tables are now explained. The first type of table (Tables 4.1, 4.3 and 4.5 for mesh, compiler design and VLSI design, respectively) contains general information about the runs. It has eight columns meaning:

Column 1 (Problem): the problem name;

Column 2 (Separation Strategy): The separation routines that have been selected during the run (see the notation given in the beginning of this section);

Column 3 (LP value): the optimal value of the LP relaxation in the first node of the enumeration tree;

Column 4 (% GAP): the percentage gap between the upper bound (found by heuristics EDGE or RANDOM) and the optimal value of the LP relaxation in the first node of the enumeration tree;

Column 5 (UB): the best upper bound found by the heuristic EDGE or RANDOM;

Column 6 (OPT): If known, the cost of an optimal solution or of a better upper bound than the one computed by the heuristics EDGE and RANDOM. An asterisk appears in this column if the problem was solved to optimality;

Column 7 (B & C Nodes): the number of nodes in the enumeration tree;

Column 8 (CPU time): the total CPU time in minutes. The numbers that appears in parenthesis are the CPU times (in minutes) after the execution of the first node of the enumeration tree, that is, before branching the first variable. If nothing appears, then no branching has been done for the run.

The second type of table (Tables 4.2, 4.4 and 4.6 for mesh, compiler design and VLSI design, respectively) gives detailed information about the number of inequalities generated in each run as well the percentage of the overall CPU time spent in solving LPs and in separating inequalities. The meanings of the columns in this type of table are:

Column 1 (Problem): the problem name;

Column 2 (C): Number of cycle and tree inequalities generated;

Column 3 (P): Number of PBC inequalities generated;

Column 4 (K): Number of knapsack tree inequalities generated;

Column 5 (S): Number of star inequalities generated;

Column 6 (E): Number of edge inequalities generated (see Section 4.3);

Column 7 (% LP): percentage of the CPU time spent in solving LPs;

Column 8 (% SEP): percentage of the CPU time spent in separation;

Mesh Problems

The computational results for the six mesh problems are summarized in Tables 4.1 and 4.2.

For all mesh problems, the upper bounds have been computed by the EDGE heuristic (RANDOM was turned off) and the CPU time limit was fixed to 120 minutes. The decision on which separation strategy to apply for a given problem was taken by running each problem with different separation strategies for a few iterations. We kept the strategies that we hoped would solve the problem within the time limit that had been fixed.

Three of the mesh problems tested have been solved to optimality: *mesh.148.2.265*, *mesh.70.2.120* and *mesh.31.2.50*. For problems *mesh.274.2.469* and *mesh.148.4.265* the LP-value is far from the best upper bound available which, in turn, is also far from the upper bounds computed by the heuristic EDGE. In the third unsolved problem, *mesh.138.2.232*, the gap between the LP-value and the optimal solution (given in column titled OPT) is relatively small since, with this lower bound, we are able to prove that any feasible solution with cost 8 is optimal (recall that the edge costs are all 1 for mesh problems). The fractional solution available after the program passes the CPU time limit was not structured enough to allow the EDGE heuristic to reduce the upper bound from 8 to 9, which would have solved the problem.

Consider now the results in Tables 4.1 and 4.2 for the multiple runs of problems *mesh.148.2.265*, *mesh.70.2.120* and *mesh.31.2.50*. These problems have been solved to optimality when only PBC inequalities or only knapsack tree inequalities have been added. An exception for the PBC inequalities is the problem *mesh.70.2.120* for which a single fractional solution was chopped off by executing the cycle and tree separation routines. We observe that the number of PBC inequalities (or PBC, cycle and tree inequalities in problem *mesh.70.2.120*) is much smaller (from 2 to 8 times smaller) than the number of knapsack tree inequalities that have been added to prove optimality. On the other hand, the CPU time spent when the PBC inequalities are separated can be much larger than the time spent when the knapsack tree inequalities are used (see problem *mesh.148.2.265*). The ratio between the LP and the separation times is at least 4 for PBCs, while it does not exceed 2 for the knapsack trees.

From the discussion above, it is reasonable to say that the PBC inequalities are important to define an optimal solution but that a faster separation routine is needed to generate these inequalities.

Note that less than 20 minutes (using knapsack tree inequalities) were necessary to find an optimal solution for the problem *mesh.148.2.265*. However, the problem of partitioning

the same mesh into 4 equal parts (*mesh.148.4.265*) is a rather difficult example for our algorithm since, after more than 2 hours of CPU time, we are, at least in terms of the best known upper bound, far from solving the problem.

Compiler Design Problems

Tables 4.3 and 4.4 give the results for the compiler problems and their bipartition versions. For the original problem versions, the CPU time limit was set to 120 minutes and, for the bipartition versions, this limit was set to 60 minutes. The upper bound heuristics EDGE and RANDOM have been both turned on for all runs. The optimal solutions for the original problem versions (column titled OPT in Table 4.3), where known, are taken from Johnson et al. (1991).

In their original versions, 4 of the 6 problems have been solved to optimality and, among these, for two of them (*cb450.45.8.98* and *cb450.47.8.99*) the program went into a branching phase. The two unsolved problems are: *cb450.47.9.101* and *cb450.61.9.187*. For the first of them, the algorithm explored 78 nodes of the enumeration tree and was unable to give the optimal solution although the gap between the lower (LP) and upper bounds in the first node of the enumeration tree was relatively small.

From Table 4.4, we notice that the number of PBC inequalities added during the execution of the algorithm is small when compared to the number of other inequalities. This was expected for two reasons. First, in the compiler design problems, the graphs are usually not 2-connected and most of their edges are incident to one of the two nodes of large weight. Therefore, the chance of finding a PBC subgraph composed of two cycles that is the support of a valid inequality is very small. The second reason is that, at each iteration, our implementation of the PBC (and also of the cycle and tree) separation routine generates at most n inequalities, while the maximum number of knapsack tree and star inequalities that is generated is much larger.

The results in Table 4.3 for the original and bipartition versions of the five smallest compiler design problems indicate that, for these instances, a heuristic which hierarchically solves bipartition problems to optimality is not a good method to tackle the original problem. In fact, except for problem *cb450.47.9.101*, we spent more time in solving the bipartition versions than the original problems. Even worse, we could not solve

problems *weq.45.2.98* and *weq.47.2.99*, whereas their original versions *cb450.45.8.98* and *cb450.47.9.99*, respectively, have been both solved to optimality.

VLSI Design Problems

The results obtained for the third class of problems, namely the VLSI placement problems, are presented in Tables 4.5 and 4.6. The CPU time limit has been set to 60 minutes for all problems except problem *vlsi.166.4.504* where we double the time limit (120 minutes). All problems have been run with both the upper bound heuristics EDGE and RANDOM active.

Only problem *vlsi.37.4.92* has been solved at the first node of the enumeration tree. The gap between the lower and upper bounds for the two unsolved problems, *vlsi.166.4.504* and *vlsi.48.4.81*, is still very large.

Compared to the compiler design instances, the number of PBC inequalities generated has increased but it is still small when compared to the number of other inequalities generated.

Further Tests and Concluding Remarks

We also have tested the behavior of the algorithm when the cluster capacities become larger. For this, three runs of problem *eq.31.2.50* have been executed with the cluster capacities of 16, 21, and 26, respectively. Moreover, four runs of problem *cb450.30.6.47* have been executed with the cluster capacities of 450, 500, 450 and 1024. The results of these tests, presented in Table 4.7, do not allow us to derive any reasonable conclusion. The runs have been made with the separation routines of PBC, cycles, trees, knapsack trees, stars and edges turned on, and with the upper bound heuristics EDGE and RANDOM activated.

Both problems have been solved to optimality for all choices of cluster capacities. For some cases, when the capacity grows, both the CPU time and the number of inequalities increase. For other cases, the opposite behavior is observed, that is, the cluster capacities increased but the CPU time and the number of inequalities decreased. Thus, we cannot conclude that the performance of our branch-and-cut algorithm depends on the cluster capacities, as is sometimes the case for other branch-and-cut algorithms designed for capacitated problems (for instance, the Capacited Facility Location Problem studied in Aardal, 1992).

As a general remark, we observe that the number of inequalities generated to find an optimal solution in our test problems is very large. This is an indication that we are missing important inequalities in the LP relaxations. The LP relaxations could be improved by strengthening some of the valid inequalities already known. The knapsack tree inequality seems to be an attractive inequality for strengthening. In fact, the addition of the knapsack tree inequalities to our relaxations has been determinant for solving some of the instances in our sample, which characterizes the importance of these inequalities in tightening the formulations. But, possibly, we are not exploiting all the potential of the knapsack tree inequality since there are examples where the lifting suggested in Proposition 3.13 is weak.

Another alternative to obtain better LP relaxations is to improve the separation routines used in the code to make them generate other valid inequalities. For instance, we are not exploiting the whole class of PBC inequalities but only a subclass of it. Therefore, we could change the PBC separation routine to make it look for more general violated PBC inequalities.

We also point out to the fact that, up to now, in all the inequalities added by the program (except the constraints (III) of the original model) the coefficients of the variables are positive. In some sense, this means that we are working with the dominant of the polytope $P_{MC}^{K,W}(G)$ and so, we are not using some inequalities such as the triangle inequalities (Barahona and Mahjoub, 1986) and the suspended tree inequalities of Chapter 2. Because the edge costs are all nonnegative and there are no node costs in our examples, these inequalities may not produce any significant change in the lower bounds (LP values), but they may give more structure to the fractional solutions that will allow the upper bound heuristics to work better.

Finally, a third alternative to improve the LP relaxations is to look for new classes of strong valid inequalities for the graph partitioning polytopes.

5. The Frontwidth Reduction Problem in Finite Elements Computations

5.1 Introduction

In this chapter we describe an application of the equipartition problem arising in Finite Elements. Finite Elements is a numerical method used to approximate functions in a given domain. Typically, the final step of such a method involves the solution of a system of linear equations. The variables in this system represent the approximation of the function at some points of the domain fixed a priori.

Several direct methods have been proposed to solve linear systems. The common goal of the different methods is the reduction of the CPU and storage requirements during the variable elimination process. In this chapter, we focus our attention to a method called *frontal* (Irons, 1970) and, more precisely, we are interested in the *frontwidth reduction problem*, that is, the combinatorial optimization problem that is solved in the preprocessing phase of the *frontal* method.

In the next section we introduce briefly the finite element concept and some terminology such as *mesh*, *element* and *nodes*. We show how the solution of a problem by finite elements leads to a system of linear equations and we describe how this system is solved with the frontal method.

To reduce the CPU time and the memory storage, the frontal method needs a preprocessing phase in which one has to find a good ordering for the elements in the mesh. The problem of finding such an ordering is called the *frontwidth reduction problem* (FRP). A short literature review on the heuristics developed for solving the FRP is presented at the end of Section 5.2.

In Section 5.3, a more formal definition of the frontwidth reduction problem is given. We introduce a graph model for the mesh that allows us to reformulate the FRP as a *cutwidth problem*.

A divide-and-conquer algorithm for the cutwidth problem is developed in Section 5.4. This algorithm solves graph equipartition problems recursively and their solutions are combined so as to obtain a good ordering for the cutwidth problem and, hopefully, for FRP.

We have chosen a heuristic approach to solve the equipartition problems within the main divide-and-conquer procedure for the cutwidth problem. Two main reasons are at the origin of our decision. First, the intrinsic difficulty of the equipartition problem which is NP-hard. Secondly, the graphs arising from finite element meshes are very large. Therefore, Section 5.5 is devoted to the presentation of local search heuristics for graph equipartition. In Subsection 5.5.1 we describe the deterministic heuristic of Kernighan and Lin (1970). The next two subsections present the nondeterministic heuristics that specialize the simulated annealing and stochastic evolution algorithms, respectively, to the graph equipartition problem.

One of the key questions in a local search framework is the choice of a starting solution for the algorithm. This point is addressed in Section 5.6 where we suggest two different methods to generate an initial solution for equipartition.

We close the chapter with a discussion of the computational results obtained by our divide-and-conquer algorithm for FRP. A small sample of two and three dimensional meshes has been tested and the results are compared to those obtained by the Reverse Cuthill-McKee algorithm that is a standard greedy algorithm. In most of the cases, the improvements observed ranged from 25 to 50%.

For a thorough presentation of the finite element method, we refer to Zienkiewicz and Morgan (1983) and Zienkiewicz and Taylor (1989). For direct methods for sparse matrices, we refer to Pissanetsky (1984) and Duff et al. (1986).

Most of the material in this chapter appears in de Souza, Keunings, Wolsey and Zone (1992).

5.2 The Finite Element Concept and the Frontal Method

The goal of this section is to give an insight into the context in which the combinatorial problem of frontwidth reduction occurs. For this, a brief introduction to the finite element concept is given and we show that the final step of the method involves the solution of a system of linear equations. Then, we describe how to solve this system with the frontal method, and we show that the efficiency of the method depends on our ability to find a good solution to a combinatorial optimization problem called the Frontwidth Reduction Problem (FRP). A short literature review on the heuristics that have been proposed for solving the FRP closes this section.

5.2.1 The Finite Element Concept

To determine the solution of differential equations numerically it is necessary to develop accurate function approximation methods. The finite element approach described below is one such methods.

Given a function ϕ to be approximated in some region Ω bounded by a closed curve Γ , the approximation function $\tilde{\phi}$ can be written as:

$$\tilde{\phi} = \sum_{i=1}^m a_i N_i \quad (1)$$

where the N_i 's are called the trial (shape or basis) functions and the a_i 's are constants to be computed.

Suppose that a physical phenomenon is governed by a linear differential equation in ϕ in the region Ω bounded by the curve Γ . This differential equation can be written in the quite general form below:

$$A(\phi) = \mathcal{L}\phi + p = 0 \quad \text{in } \Omega \quad (2)$$

where \mathcal{L} is an appropriate linear differential operator and p is independent of ϕ . Usually, boundary conditions have also to be satisfied and can be expressed in a general form as:

$$B(\phi) = \mathcal{M}\phi + r = 0 \quad \text{on } \Gamma \quad (3)$$

where \mathcal{M} is an appropriate linear differential operator and r is independent of ϕ .

Assume that a set (N_1, N_2, \dots, N_m) of trial functions has been chosen. To completely characterize the approximation $\tilde{\phi}$, the values of the constants a_i for $i \in \{1, \dots, m\}$ must also be determined. These values are computed by the *weighted residual method*. To explain this method, we first define the errors R_Ω and R_Γ in the approximation. The residual R_Ω in the domain Ω is given by:

$$R_\Omega = A(\tilde{\phi}) - A(\phi) = A(\tilde{\phi}) = \mathcal{L}\tilde{\phi} + p \quad \text{in } \Omega \quad (4)$$

The residual R_Γ on the boundary Γ is given by:

$$R_\Gamma = B(\tilde{\phi}) - B(\phi) = B(\tilde{\phi}) = \mathcal{M}\tilde{\phi} + r \quad \text{on } \Gamma \quad (5)$$

The weighted residual method is an attempt to reduce the sum of the residuals (errors) in the domain and on the boundary in some overall manner. For this, we require that a certain number k of integrals of the residual over Ω and Γ , weighted in different ways, be zero, i.e.:

$$\int_{\Omega} W_j R_\Omega d\Omega + \int_{\Gamma} \overline{W}_j R_\Gamma d\Gamma = 0 \quad \forall j \in \{1, \dots, k\} \quad (6)$$

Assume that the number of independent weight functions W_j is chosen to be equal to the number of trial functions N_i ($k = m$). Substituting (1), (3) and (4) in the equations (6) yields a linear system on m equations and m variables which can be written in matricial form as:

$$\mathcal{K}a = q \quad (7)$$

where

$$k_{ji} = \int_{\Omega} W_j \mathcal{L}N_i d\Omega + \int_{\Gamma} \overline{W}_j \mathcal{M}N_i d\Gamma \quad (8)$$

$$q_j = - \int_{\Omega} W_j p d\Omega - \int_{\Gamma} \overline{W}_j r d\Gamma \quad (9)$$

The development given so far applies to any approximation method of the function ϕ . In the sequel we specialize the above formulas to the case of the finite element method.

The finite element method to approximate ϕ in the bounded region Ω starts by partitioning the region Ω into smaller regions Ω_e called the *elements*. The set of all elements in the partition defines a *mesh* on Ω . Each element Ω_e has nodes on its boundary and possibly in its interior. Clearly, a node can belong simultaneously to many different elements.

For each node i , a trial function N_i is defined in a piecewise manner such that it takes a value zero on all elements of the mesh except those containing node i . With the N_i functions defined as above, the definite integrals in (6), (8) and (9) can be computed by summing the individual contributions of each element. For (8) and (9), this yields:

$$\begin{aligned} k_{ji} &= \sum_{e=1}^n \left(\int_{\Omega_e} W_j \mathcal{L} N_i d\Omega + \int_{\Gamma_e} \overline{W}_j \mathcal{M} N_i d\Gamma \right) \\ &= \sum_{e=1}^n k_{ji}^{\Omega_e} \end{aligned} \quad (10)$$

$$\begin{aligned} q_j &= - \sum_{e=1}^n \left(\int_{\Omega_e} W_j p d\Omega + \int_{\Gamma_e} \overline{W}_j r d\Gamma \right) \\ &= \sum_{e=1}^n q_j^{\Omega_e} \end{aligned} \quad (11)$$

where n is the number of elements in the mesh and Γ_e denotes the portion of the boundary of element Ω_e that lies on Γ .

Let \mathcal{K}^e be the $m \times m$ matrix associated to element Ω_e whose coefficients are given by the $k_{ji}^{\Omega_e}$ values in (10). Note that these values are 0 if node i or node j are not in Ω_e . We have that matrix \mathcal{K} (see (7)) is given by the sum of the \mathcal{K}^e matrices for $e = 1, \dots, n$, i.e.:

$$\mathcal{K} = \sum_{e=1}^n \mathcal{K}^e \quad (12)$$

Analogously, if q^e is the vector with components given by the $q_j^{\Omega_e}$ in (11), we have that:

$$q = \sum_{e=1}^n q^e \quad (13)$$

From (12) we can see that the element matrices \mathcal{K}^e can be computed and added sequentially so as to obtain the global matrix \mathcal{K} . From (13), we can see that the same is

true for the element vectors q^e and the vector q . This process is called the *assembly* of \mathcal{K} and q and, at some iteration, an element is said to be assembled if its associated matrix and its associated vector have already been summed.

In practice, the trial and weighting functions are chosen appropriately such that \mathcal{K} is sparse and often it is a symmetric matrix. Thus, \mathcal{K} can be put into a band form. In such a situation, the system $\mathcal{K}a = q$ can be solved efficiently using a band method in which the variable elimination begins only after the matrix \mathcal{K} has been fully assembled. An alternative technique to solve the linear system is provided by the frontal method (Irons, 1970) in which the variable elimination starts before the summations in (12) and (13) have been completed. The Frontal Method is presented below.

5.2.2 The Frontal Method

Suppose that the elements are assembled in the conventional order defined by the index $e \in \{1, \dots, n\}$. Let C_k be the set of assembled elements at the k -th step. The node set is partitioned into four subsets defined below:

- $S_1^k = \{i \in M : E_i \subseteq C_{k-1}\}$
(set of nodes fully assembled and eliminated);
- $S_2^k = \{i \in M : E_i \subseteq C_k \text{ and } E_i \not\subseteq C_{k-1}\}$
(set of nodes fully assembled but not eliminated);
- $S_3^k = \{i \in M : C_k \cap E_i \neq \emptyset \text{ and } E_i \not\subseteq C_k\}$
(set of nodes partially assembled);
- $S_4^k = \{i \in M : C_k \cap E_i = \emptyset\}$
(set of nodes not assembled);

where M is the set of nodes in the mesh, E is the set of elements in the mesh and E_i is the set of elements that contain node i .

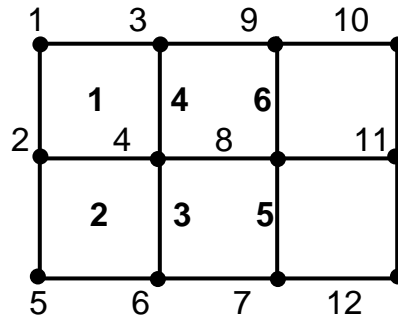
Clearly, if Ω_e is an element not in C_k , then $k_{ij}^{\Omega_e} = 0$ if either i , or j , or both, are in S_1^k . This means that all rows and columns of \mathbb{K} corresponding to nodes in S_1^k are fully assembled. Note also that $k_{ij} = 0$ if $i \in (S_1^k \cup S_2^k)$ and j is in S_4^k , because, in this case, $E_i \cap E_j = \emptyset$.

The variables corresponding to nodes in S_2^k can be eliminated and this affects only the submatrix associated to the nodes in $S_2^k \cup S_3^k$. This means that only this submatrix needs to be available in memory at this stage for the elimination to be processed. In other words, we must be able in step k to store a $f_k \times f_k$ submatrix where $f_k = |S_2^k| + |S_3^k|$ is called the k -th frontwidth and $F_k = S_2^k \cup S_3^k$ is the k -th front.

Therefore, the memory storage necessary to run the frontal method until the matrix \mathbb{K} has been completely assembled is proportional to f^2 , where f is computed as $\max\{f_k : k = 1, \dots, n\}$. It can be proved that the CPU time needed to solve the linear system $\mathbb{K}a = q$ is also proportional to f^2 . Thus, the performance of the method is closely related to our ability to keep the frontwidth small.

The frontal method is applied to the simple mesh of Figure 5.1(a). In Figure 5.1(b), the matrix \mathcal{K} associated to this mesh is represented at each step and the shaded area corresponds to the submatrix that has to be available in memory for the elimination of the variables in the set S_2^k . We assume that \mathcal{K} is symmetric and we denote the nonzero elements in \mathcal{K} by an 'X'. The rows and columns have been permutated at each step for the clarity of the presentation.

It can be seen from this figure that the elimination of a row (and column) corresponding to a mesh node is processed as soon as all elements containing that node have been assembled. To have small frontwidths, a node should stay as few steps as possible in the front and this depends upon the order in which the elements are assembled. The problem of finding the ordering to assemble the \mathcal{K}^e matrices that minimizes the maximum of the frontwidths is called the Frontwidth Reduction Problem (FRP). In the next sections of this chapter we develop a heuristic to produce good solutions for the FRP. But, before, we present a literature review on heuristic algorithms for the FRP.



(a)

Figure 5.1:

Figure 5.1:

5.2.3 Existing Heuristic Algorithms for FRP

Though the FRP is commonly cited in many works as a difficult problem, we did not find any formal proof in the literature that it belongs to the class of NP-hard problems. However, the difficulty of the problem and the necessity to find solutions to practical applications have pushed many authors to develop heuristics to obtain approximate solutions for FRP. Some authors, like Sloan and Randolph (1983), have classified the heuristics into *direct* and *indirect*.

Typically, an indirect heuristic has two phases. The first phase gives a solution to the Bandwidth Problem (BP), that is, the problem of finding an ordering (labelling) of the variables of the mesh which minimizes the bandwidth. The bandwidth of an ordering is the maximum, taken over all elements in the mesh, of the difference between the largest and the smallest variable labels.

The second phase of an indirect heuristic uses this new variable ordering to produce an element ordering (solution for FRP). In this element ordering, the elements are labelled by increasing order of the least label of their variables (given by the solution of the Bandwidth Problem). The fundamental reason for this strategy is that the frontwidth obtained in this way cannot be larger than the corresponding bandwidth (see, for instance, Sloan and Randolph, 1983).

Because the indirect heuristics initially solve the BP, we include in our literature review some of the heuristics proposed for this problem.

The most well-known heuristic for BP is due to Cuthill and McKee (1969). They use a graph model to represent a mesh in which the mesh variables are in one-to-one correspondence with the graph nodes. This model is sometimes called the *node connectivity graph*. A greedy strategy based on the node degrees is then used to generate the variable ordering.

Empirical observations of George (1971) originated in a new version of the heuristic of Cuthill and McKee. In this version, an additional step is executed at the end of the original algorithm that simply reverses the element ordering. It is reported that this often produces better orderings than the original Cuthill-McKee algorithm. This heuristic is known in the literature as the Reverse Cuthill-McKee algorithm.

Some shortcomings of the Cuthill-McKee algorithm for the BP have been pointed out in Gibbs et al. (1976). There another greedy heuristic is proposed that uses more involved graph concepts. The results reported show that the solutions are comparable in quality to the ones obtained with the Reverse Cuthill-McKee algorithm. However, the CPU times reported indicate that the algorithm of Gibbs et al. runs faster than the Reverse Cuthill-McKee algorithm.

Examples of indirect heuristics for FRP are found in Razzaque (1980), Akin and Purdue (1976) and Sloan and Randolph (1983).

Direct heuristics do not reorder the variables of the mesh and, therefore, do not have to solve the BP. They work directly with the element labels but the main strategy remains of greedy type. Some of these heuristics, as in Pina (1981), use the same graph model described before. Others have a different graph model where the mesh elements are in one-to-one correspondence with the graph nodes. This model is sometimes called the *element connectivity graph* and has been used in Bykat (1977) and in Fenves and Law (1983).

Another direct heuristic commonly used in practice is the Reverse Cuthill-McKee algorithm. Although, this algorithm has been originally developed for the BP, it can be applied to the element connectivity graph and, in this case, the algorithm generates an element ordering which, hopefully, gives a good solution to the FRP. This heuristic has been used in our computational experiments.

The literature on heuristics for the FRP and the BP is extensive. We do not go further in there, but we point out that the existing heuristics are essentially based upon greedy strategies.

5.3 Standard and Modified Frontwidth Problems

In the previous section we have seen that, to solve the linear system $Ka = q$ efficiently, the frontal method requires that the elements of the mesh are ordered appropriately. Such an ordering can be obtained by solving the frontwidth reduction problem (FRP) which is formally defined in this section. We propose to tackle a modified frontwidth problem (\overline{FRP}), which is related, though different, from the standard problem. As we shall see below, this modified problem leads itself more readily to a solution by means of graph theoretic concepts. The numerical results of Section 5.7 show that, in practice, a good ordering for the modified problem is also a good ordering for the original problem.

We consider general, structured or unstructured finite element meshes in two or three dimensions. For two-dimensional meshes, the elements have the shape of a rectangle or of a triangle and the nodes are located at the corners and at the midsides of the elements (and also in the center of gravity for rectangular elements). On the other hand, three-dimensional meshes are composed of elements with the shape of a tetrahedron with the nodes only at their corners. We shall assume for the sake of illustration that a single nodal value (variable) is associated to each node of the mesh, in other words, we are interested in computing the value of a single function ϕ at each node (for example, ϕ may represent the temperature).

Throughout the remainder of this chapter, the terminology vertex is used to designate a node of a graph. Our intention is to avoid any confusion between the nodes of a graph and the nodes of a mesh.

Given an ordering of the elements of a mesh, the standard front F_i and the modified front H_i are defined as follows:

Initialization: $F'_0 \leftarrow \phi$. Set $i \leftarrow 1$

Step 1: $F_i \leftarrow F'_{i-1} \cup \{\text{all nodes of element } i\}$.

$H_i \leftarrow \{\text{sides common to one element with label } \leq i \text{ and}$
 $\text{one with label } > i\}$.

Set $f_i \leftarrow |F_i|$ and $h_i \leftarrow |H_i|$.

Step 2: $F'_i \leftarrow F_i \setminus \{\text{nodes only belonging to elements } 1, \dots, i\}$.

Set $i \leftarrow i + 1$.

These definitions are illustrated in Figure 5.2, where we present a simple mesh composed of 8-node rectangular elements. With the ordering shown, $f_4 = 10$ while $h_4 = 2$.

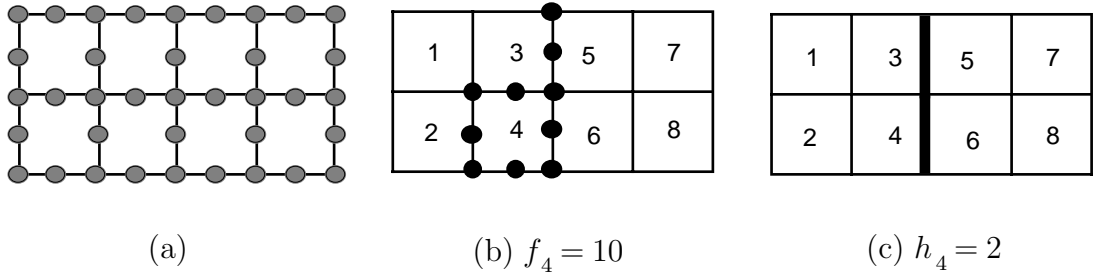


Figure 5.2:

The *standard frontwidth problem* FRP is to find an ordering of the elements minimizing $f = \max_i f_i$. The *modified frontwidth problem* \overline{FRP} tackled below is to find an ordering minimizing $h = \max_i h_i$.

In order to tackle the modified frontwidth problem, we model the problem as follows. Let us construct a graph $G = (V, E)$ in which each vertex $i \in V$ corresponds to an element of the mesh, and two vertices are joined by an edge $e = (i, j) \in E$ if the elements

$$n = |V|.$$


GRAPH $G=(V,E)$

for element orderings (see Fenves and Law, 1983).

problem as the cutwidth problem on graph G .

it can be far from optimal as it is illustrated by the example of Figure 5.4. The ordering

indicated in Figure 5.4(b) is optimal for \overline{FRP} , while the one of Figure 5.4(c) is optimal for FRP. But, the (standard) frontwidth obtained with the optimal ordering of \overline{FRP} is 22% larger than the optimal value of the frontwidth for FRP. Despite this negative example, the numerical results of Section 5.7 show that our alternative model is effective in practice.

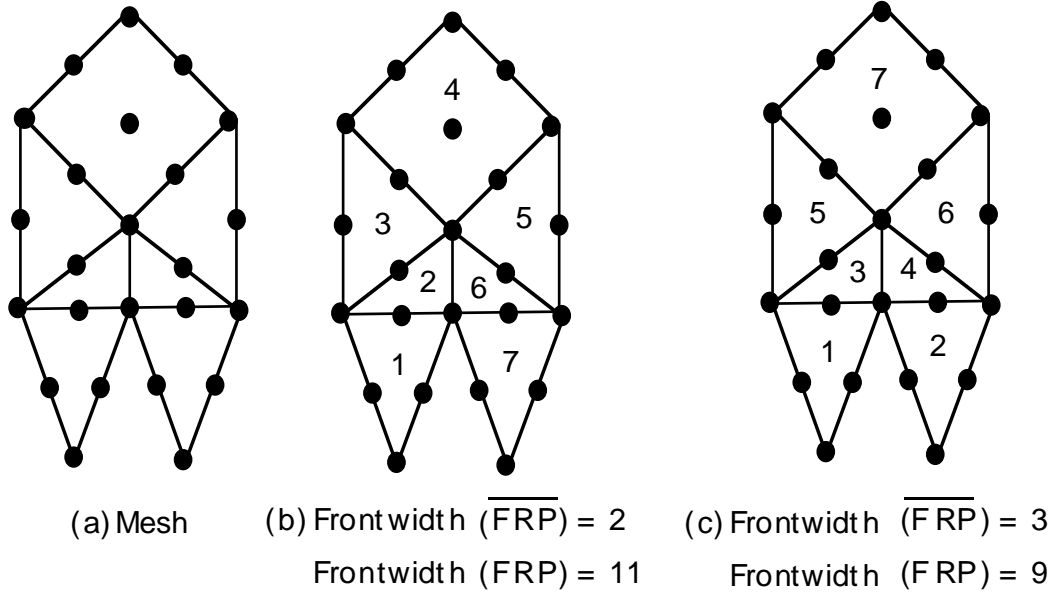


Figure 5.4:

For what comes next, it is useful to think of an equipartition $(U, V \setminus U)$ of a graph $G = (V, E)$ as a bicoloring of the vertices in V , i.e., all vertices in U have the same color and all vertices not in U have also the same color but different from that of vertices in U .

5.4 A Divide-and-Conquer Heuristic Algorithm for \overline{FRP}

In one way or another, the existing heuristics for FRP are based on a greedy strategy in which the elements of the mesh are ordered sequentially and the next element chosen at each iteration is the one that causes the smallest increase in the current frontwidth. However, this strategy can produce solutions of very poor quality. To see this, consider the example shown in Figure 5.5 and suppose that ℓ is the number of square elements in the top row.

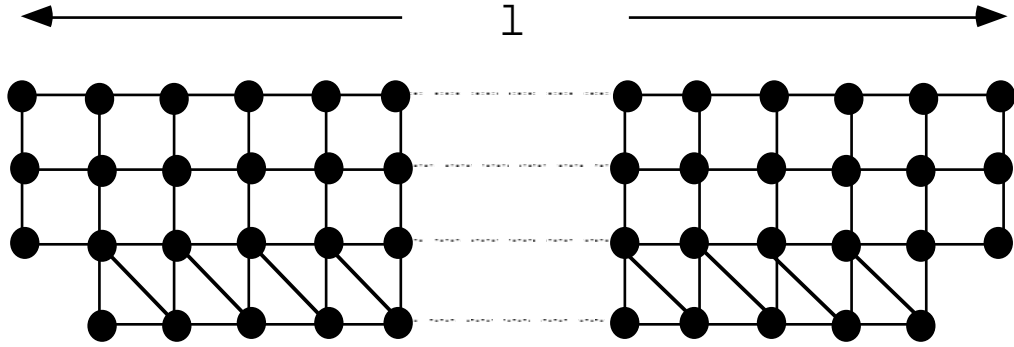


Figure 5.5:

If we apply the Cuthill-McKee (1969) algorithm to this mesh, the resulting ordering has both standard and modified frontwidths of size $O(\ell)$ since the heuristic labels all triangular elements before it starts labelling the square elements. It is easy to see that an optimal ordering has standard frontwidth of size 6 and modified frontwidth of size 4.

From this example, we see that greedy (local) heuristics cannot always work well in solving \overline{FRP} (FRP) because the modified (standard) frontwidth clearly depends on the global structure of the graph (mesh).

This is the first observation that motivates the approach we use in the heuristic algorithm for \overline{FRP} that is described in this section.

The second observation comes from an attempt to find a good *lower bound* on the value h of the optimal (modified) frontwidth based upon a more global view of the graph. Note that $h = \min_{\pi} \max_i |\delta(U_i)| \geq \max_i \min_{\pi} |\delta(U_i)| \geq \min_{\pi} |\delta(U_i)|$ for all $i = 1, \dots, n$. Taking the view that $i = \lfloor \frac{n}{2} \rfloor$ is a priori a choice that will give a strong bound, we have that $h \geq \min_{\pi} |\delta(U_{\lfloor \frac{n}{2} \rfloor})|$. However finding $\min_{\pi} |\delta(U_{\lfloor \frac{n}{2} \rfloor})|$ is the problem of choosing a partition π such that the $(\lfloor \frac{n}{2} \rfloor)$ th cutset is minimized, and this is nothing but the Equipartition problem with unit edge weights. Thus a solution to the Equipartition problem gives us sets $V_b = U_{\lfloor \frac{n}{2} \rfloor}$ and $V_g = V \setminus U_{\lfloor \frac{n}{2} \rfloor}$ such that the vertices (elements) of V_b are numbered $1, \dots, \lfloor \frac{n}{2} \rfloor$ and those of V_g are numbered $\lfloor \frac{n}{2} \rfloor + 1, \dots, n$. Now it is natural to again use equipartition to divide up V_b into two sets $(V_b)_b$ and $(V_b)_g$ so that the vertices of $(V_b)_b$ are numbered $1, \dots, \lfloor \frac{n}{4} \rfloor$ and those of $(V_b)_g$ are numbered $\lfloor \frac{n}{4} \rfloor + 1, \dots, \lfloor \frac{n}{2} \rfloor$, etc.

Thus we arrive naturally at the idea of a divide-and-conquer approach based on solving $n - 1$ equipartition subproblems. The description of the algorithm is given below, while Figures 5.6 and 5.7 give illustrative examples.

Let us assume that, in a typical subproblem, a given set of p unordered vertices of V has been assigned the first p positions in the ordering, and a different set of $n - q$ unordered vertices has been assigned the last $n - q$ positions. For this subproblem, we first construct a new graph $G' = (V', E')$ from $G = (V, E)$ as follows.

If $p > 0$, V' contains a black pseudovortex b representing the first set of p vertices, and if $q < n$, V' contains a grey pseudovortex g representing the latter set of $n - q$ vertices. In addition V' contains the $q - p$ unassigned (white) vertices of V . The edges in E' and their weights are defined as follows. Given a vertex $j \in V' \setminus \{b, g\}$, let J_b (respectively J_g) be the subset of vertices represented by b (respectively g) adjacent to j in G . When J_b (respectively J_g) is nonempty, the edge (b, j) ((g, j)) is in E with weight $w_{bj} = |J_b|$ ($w_{gj} = |J_g|$). Let $E(b, g) \subset E$ be the set of edges going from vertices represented by b to vertices represented by g . For $E(b, g)$ nonempty, (b, g) is in E' and its weight $w_{bg} = |E(b, g)|$. The remaining edges of E' are the edges $(i, j) \in E$, where i and $j \in V' \setminus \{b, g\}$, and their weights are set

to one.

Once $G' = (V', E')$ has been constructed, the equipartition subproblem finds a partition $(U', V' \setminus U')$, with $b \in U'$ and $g \in V' \setminus U'$, of the $q - p + 2$ vertices of G' in which the white vertices are distributed in a balanced way. The outcome is a decision that the white vertices of U' will all precede the white vertices of $V' \setminus U'$ in the final ordering.

The procedure then continues with the creation of two new subproblems. The first takes b as the new black pseudovortex, $V' \setminus U'$ as the new grey pseudovortex and the vertices of $U' \setminus \{b\}$ are white. The second takes U' as the new black pseudovortex, g as the new grey pseudovortex and the vertices of $V' \setminus (U' \cup \{g\})$ are white.

We start the algorithm with $G' \leftarrow G$, and all vertices colored white. The equipartition procedure for a given subproblem terminates when the corresponding U' (or $V' \setminus U'$) contains a single white vertex. The position of that white vertex is then uniquely determined: it is assigned the first position after that of the black vertices (or the last position prior to that of the grey vertices).

Figure 5.6 illustrates the above operations for a particular subproblem. Figure 5.6(a) shows the graph of Figure 5.3 and the associated graph $G' = (V', E')$ for the present subproblem. Here, the unordered black vertices have been assigned positions $\{1, \dots, 4\}$ from previous steps, while the white vertices have no assigned positions as yet. The optimal equipartition of $G' = (V', E')$ leads to $U' = \{b, v_6, v_9\}$ and cutset $\delta(U') = \{(b, v_5), (v_9, v_5), (v_6, v_7)\}$. It follows that, in the final ordering, vertices $\{v_6, v_9\}$ will occupy positions $\{5, 6\}$, while $\{v_5, v_7, v_8\}$ are assigned positions $\{7, 8, 9\}$. Note, however, that the actual internal ordering remains to be determined for both sets $\{v_6, v_9\}$ and $\{v_5, v_7, v_8\}$. The two new subproblems originating from that of Figure 5.6(a) are shown in Figures 5.6(b)-(c). In the first subproblem, vertices $\{v_5, v_7, v_8\}$ form the grey pseudovortex, and only the white nodes v_6, v_9 are unassigned. In the second subproblem, vertices $\{v_1, \dots, v_4, v_6, v_9\}$ form the black pseudovortex. Note that the edge (b, v_5) has a weight of 2.

In Figure 5.7, we apply the complete *DC* algorithm to the mesh of Figure 5.3. Assuming 4-node quadrilateral elements and one scalar unknown per node, the resulting ordering yields a frontwidth $f = 6$. This is a 33% improvement relative to the initial ordering of Figure 5.3, for which $f = 9$.

Figure 5.7:

The use of equipartition of the element connectivity graph is not a novelty in the finite element field. Williams (1990) has studied the Load Balancing problem for parallel mesh computations. In the Load Balancing problem, one has to partition the vertices of the graph into 2^p subsets such as to minimize a given function $g(.)$, where 2^p is the number of processors available. Each of the subsets in the partition corresponds to the portion of the mesh that has to be treated by one of the processors. The objective function $g(.)$ is composed of two terms. The first penalizes unbalanced partitions in an attempt to reach a uniformly distributed workload among the processors. The second term favours the partitions that have few edges cut, since each of these edges somehow express the needs for communication between two processors.

Williams proposed two methods that recursively apply equipartition to obtain the desired 2^p subsets. The first called orthogonal bisection is a simple method which cuts the graph in two parts by a vertical line, then cut each half in two by an horizontal line, each quarter in two by a vertical line and so on. This method is suitable for planar meshes and the choice of the lines location depends on the coordinates of the elements. The second method is more complicated and the successive equipartitions problems are solved using the eigenvectors of a matrix related to the adjacency matrix of the graph.

VanderStraeten (1992) also considered the load balancing problem. In his work, which finds some of its roots in this thesis, he used nondeterministic combinatorial optimization techniques such as simulated annealing and stochastic evolution to partition the mesh into 2^p parts.

We now describe three different heuristics used to solve the graph equipartition problem.

5.5 Heuristics for Minimum Weight Equipartition

The three heuristics we have used for the equipartition problems in the Divide-and-Conquer algorithm are all neighborhood search techniques.

The first, due to Kernighan and Lin (1970), is deterministic. The other two, Simulated Annealing and Stochastic Evolution, are stochastic. The first two heuristics are well-known and a recent study of Johnson et al. (1989) contains extensive results comparing the two algorithms for the equipartition problem. Our implementation is thus largely based on their report, though the class of graphs we work on differs considerably. The third approach, Stochastic Evolution, is more recent. The basic reference is Saab and Rao (1990). Below we briefly describe the three heuristics and discuss the main implementation issues. Although the heuristics are actually applied to the G' graphs, we describe them for a general graph $G = (V, E)$.

We wish to partition the vertex set V into two sets of, say, blue and red vertices. Since a partition is fully characterized by the set B of blue vertices, we use B to denote the partition.

Associated with a given partition B is a set of neighbouring partitions $N(B)$ and a cost $f(B)$. The way the neighborhood B is defined depends on the method chosen to perturb B to obtain a new partition. The goal is to find a partition B^* that minimizes the cost function f .

All three heuristics used here contain a major loop, called a pass, consisting of a specified number of iterations. At the beginning of each pass, an initial partition B is given. In each iteration of the pass, one starts with the current partition B' (which is equal to B in the first iteration) and chooses a neighbour \tilde{B} . Then, depending on the value of $f(B') - f(\tilde{B})$, \tilde{B} is either accepted, in which case the iteration terminates with $B' = \tilde{B}$, or \tilde{B} is rejected and B' remains unchanged. The best partition found by the algorithm

is updated at the end of the each iteration or after the pass. The actual definition of the neighborhood depends on the chosen heuristics.

5.5.1 Kernighan and Lin (KL)

The neighborhood $N(B)$ is the set of partitions obtainable from B by simultaneously making one blue vertex red and vice-versa. Thus, taking the initial solution as an equipartition ensures that all succeeding solutions are equipartitions. The objective function is $f(B) = \sum_{e \in \delta(B)} w_e$.

A pass consists of $\frac{n}{2}$ iterations, where $n = |V|$. The candidate partition \tilde{B} is chosen as the best possible neighbour of the current partition B' , subject to the additional constraint that no vertex is moved that has already been transferred during the pass. The candidate is always accepted, i.e., a pass iteration terminates with $B' = \tilde{B}$. The pass finishes by taking the best of the $\frac{n}{2}$ equipartitions that have been generated. If the chosen equipartition improves the objective function, it is taken as the initial solution for a new pass; otherwise the algorithm terminates.

Given the fact that in our graphs the vertex degree is at most 4, it is possible, using appropriate data structures in the way suggested by Fiduccia and Matheyses (1982), to implement the algorithm so that a pass takes $O(n)$ operations. As reported by Johnson et al. (1989), 3 to 5 passes are typically required for the algorithm to terminate.

The basic steps of the algorithm are given below.

Kernighan-Lin Heuristic:

Step 0: (initializations)

Generate an initial equipartition B and set:

$B^* \leftarrow B$ (B^* is the best solution available)

Step 1: (Pass initializations)

improvement \leftarrow FALSE;

$B' \leftarrow B^*$ (B' is the current partition)

Step 2: (Pass)

For $t = 1$ **to** $\frac{n}{2}$ **do**

{ Choose \tilde{B} in the neighborhood of B' }

· Find $\tilde{B} = B' \cup \{v^t\} \setminus \{u^t\}$

minimizing the value of $f(\cdot)$ in $N(B')$ restricted

to $u^t \in B' \setminus \{u^1, \dots, u^{t-1}\}$ and

to $v^t \in (V \setminus B') \setminus \{v^1, \dots, v^{t-1}\}$;

{ Acceptance of \tilde{B} : accepted with probability one }

· $B' \leftarrow \tilde{B}$

{ Update of the best solution found }

· **If** $f(B^*) > f(B')$ **then**

$B^* \leftarrow B'$;

improvement \leftarrow TRUE;

end if

end for

Step 3: (Stopping criterion)

If improvement=TRUE **then**

go to Step 1;

return B^* ;

stop.

end if

In the next two stochastic heuristics, a slightly different neighborhood is taken. Here $N(B)$ is the set of partitions obtained from B by changing the color of a single vertex. The result is that B is not always an equipartition. To handle this, the objective function is taken to be $f(B) = \sum_{e \in \delta(B)} w_e + \alpha(2|B| - n)^2$ where the second term is used to penalize unbalanced partitions and α is a parameter to be chosen. If either of the two algorithms given below terminates with a partition that is not an equipartition, we convert it into

one using a greedy algorithm.

5.5.2 Simulated Annealing (SA)

After each pass, a parameter T known as the temperature is decreased. The pass consists of a given number L of the following iterations. Given the current partition B' , randomly choose one of the n neighbouring partitions $\tilde{B} \in N(B')$. If the objective function decreases, i.e., $f(\tilde{B}) < f(B')$, replace B' by \tilde{B} . However, if $f(\tilde{B}) \geq f(B')$, replace B' by \tilde{B} with probability $e^{-\frac{f(\tilde{B}) - f(B')}{T}}$, and otherwise B' remains unchanged.

As T decreases from pass to pass, the probability of accepting a move to a worse partition is higher early in the algorithm and eventually decreases almost to zero. Thus, in some sense, simulated annealing first behaves somewhat randomly, but finishes with behaviour close to that of a deterministic descent heuristic.

The algorithm involves four parameters: T_0 (the initial temperature), L (the number of iterations during each pass), r (the cooling parameter, with $0 < r < 1$, that reduces T to rT after each pass) and α (the penalty used in the objective function). In the experiments described below we took $T_0 = 0.9$, and the values suggested by Johnson et al. (1989) for the other parameters: $L = 16n$, $r = 0.9$ and $\alpha = .05$. Finally, the algorithm was stopped after five successive passes in which the number of accepted changes was below 2% (the algorithm is then said to be frozen).

Our implementation also included two other modifications suggested in Johnson et al. (1989). To ensure that some vertex was not chosen L times during a pass, we imposed that each vertex was chosen randomly and exactly once in the first n iterations. For the remaining $(L - n)$ iterations, the vertex was chosen as above. In addition, we implemented a table look-up scheme for the costly calculation of the exponential $e^{-\frac{f(\tilde{B}) - f(B')}{T}}$. For an overview of simulated annealing, see Laarhoven and Aarts (1987).

The steps in the SA algorithm for equipartition are the following:

Simulated Annealing Heuristic:

Step 0: (initializations)

Generate an initial equipartition B and set:

$B^* \leftarrow B$ (B^* is the best solution available)

$B' \leftarrow B$ (B' is the current partition)

$T \leftarrow T_0$ (initial temperature)

Step 1: (Pass)

For $t = 1$ **to** L **do**

{ Choose \tilde{B} in the neighborhood of B' }

· Pick $u \in V$ randomly.

If $u \in B'$ **then**

$\tilde{B} \leftarrow B' \cup \{u\}$

else $\tilde{B} \leftarrow B' \setminus \{u\}$

{ Acceptance of \tilde{B} }

· $B' \leftarrow \tilde{B}$ with probability given by:

$\min\{1, e^{-\frac{f(\tilde{B}) - f(B')}{T}}\};$

{ Update of the best solution found }

· **If** $f(B^*) > f(B')$ **then** $B^* \leftarrow B'$;

end for

Step 2: (Parameters update)

· $T \leftarrow \rho T$;

update *frozen* { see text };

Step 3: (Stopping criterion)

If *frozen* = *FALSE* **then go to step 1**;

else If $(B^*, V \setminus B^*)$ **is not an equipartition**

then update $(B^*, V \setminus B^*)$ using a greedy

procedure to restore feasibility;

Return $(B^*, V \setminus B^*)$ and **stop**.

end if.

5.5.3 Stochastic Evolution (SE)

At each pass, we update an acceptance parameter p and a parameter ρ measuring improvements of the solution. At the beginning, we define a randomly chosen sequence of all the vertices $\{v_1, v_2, \dots, v_n\}$. A pass consists of n iterations. The i -th iteration starts with a partition B' (for the first iteration of the first pass B' is equal to a given partition B). We then consider the neighbour \tilde{B} obtained from B' by changing the color of v_i . When $f(\tilde{B}) \leq f(B')$, B' is replaced by \tilde{B} with probability one. If the candidate partition \tilde{B} is worse than B' , we replace B' by \tilde{B} with probability $\max\{0, 1 - \frac{f(\tilde{B}) - f(B')}{p}\}$.

Otherwise, B' remains unchanged for the next iteration. The equipartition coming out from the n -th iteration is the starting solution for the next pass. If this solution is the best found so far, ρ is decreased by a parameter R otherwise it is increased by one unity.

The parameter R can be seen as a credit that is given to the maximum number of allowed pass iterations without finding any improvement. At the end of a pass, if no improvement was found with respect to the solution with which the pass has been initiated, the acceptance parameter p is increased to $g(p)$ otherwise it is set back to the base (small) value p_0 .

Note that when p increases, the probability of accepting a worse partition increases. Thus, in contrast to simulated annealing, stochastic evolution will start off by only taking descent steps, and only when no improvement is found will increase the probability of accepting worse partitions so as to escape from a local optimum. Once an improvement is found, it immediately switches back to deterministic descent mode.

The algorithm involves the parameters p_0 , R , α and an increasing function $g(p)$. The algorithm terminates when $\rho = R$, so the number of consecutive passes without improvement is at least R . The values used in the implementation were: $p_0 = 0.2$, $R = \max\{20, \frac{n}{15}\}$ for 2D meshes and $R = \max\{20, 4(\lceil \log(n) \rceil + 1)\}$ for 3D meshes, $\alpha = 0.05$ and $g(p) = \frac{3}{4}(p+1)$.

With p_0 and $g(p)$ as above, p never exceeds 3 and $g(p)$ is indeed an increasing function. This can be seen by writing p with the recurrence formula (defined on $g(p)$) given by:

$$p_i = \frac{3^i}{4} p_0 + 3(1 - \frac{3^i}{4}) = (\frac{3}{4})p_{i-1} + \frac{3}{4}$$

Thus, since the maximum degree of a vertex is 4 for the graphs treated here, the choices for the value of p_0 and the function $g(.)$ will cause the rejection of all moves producing an increase of at least 3 unities in $f(.)$ and moves increasing $f(.)$ of at most 3 unities but leading to highly unbalanced partitions.

The basic steps of the SE procedure are presented below.

Stochastic evolution Heuristic:

Step 0: (initializations)

Generate an initial equipartition B and set:

$B^* \leftarrow B$ (B^* is the best solution available)

$B' \leftarrow B$ (B' is the current partition)

$p \leftarrow p_0$ (acceptance parameter is set to a small value)

$\rho \leftarrow 1$ (is the counter of pass iterations)

Choose randomly a sequence $S = \{v^1, \dots, v^n\}$ of vertices in V

Step 1: (Pass initialization)

$f' \leftarrow f(B')$;

Step 2: (Pass)

For $t = 1$ **to** nL **do**

{ Choose \tilde{B} in the neighborhood of B' }

· **If** $v^i \notin B'$ **then**

$\tilde{B} \leftarrow B' \cup \{v^i\}$

else $\tilde{B} \leftarrow B' \setminus \{v^i\}$

{ Acceptance of \tilde{B} }

· $B' \leftarrow \tilde{B}$ with probability given by:

$\min\{1, \max\{0, 1 - \frac{f(\tilde{B}) - f(B')}{p}\}\}$;

end for

Step 3: { Update of the best solution found }

If $f(B') < f(B^*)$

then $B^* \leftarrow B'$ **and** $\rho \leftarrow \rho - R$

else $\rho \leftarrow \rho - 1$;

Step 4: (Acceptance parameters update)

If $f' < f(B')$

then $p \leftarrow g(p)$ { **stochastic behavior is promoted** }

else $p \leftarrow p_0$ { **deterministic behavior is promoted** } ;

Step 5: (Stopping criterion)

If $\rho < R$ **then go to step 1;**

else

If $(B^*, V \setminus B^*)$ **is not an equipartition**

then update $(B^*, V \setminus B^*)$ using a greedy
procedure to restore feasibility;

Return $(B^*, V \setminus B^*)$ and **stop.**

end if.

5.6 Initial Solution for Equipartition

As mentioned above, all three heuristics need an initial partition for each of the equipartition subproblems. Our first approach, denoted *ICM*, was to make use of the reverse Cuthill-McKee ordering. For each equipartition problem with n' vertices in the graph G' , we take B to be the $\frac{n'}{2}$ vertices that appear first in the CM ordering.

Inspection of the resulting initial solutions led us to devise an alternative procedure, denoted *ISG*. In order to describe it, we first introduce some standard graph theoretic terminology.

Let $G = (V, E)$ be a connected graph. The distance between two vertices in G is the size of the shortest path connecting them. A diameter of G is a shortest path connecting two vertices of maximal distance in G . A pseudo diameter is a path with size close to that of the diameter.

The *ISG* starting heuristic is motivated by the idea that during the *DC* steps, the partitions of the G' graphs should, if possible, form connected subgraphs. This is often not the case for the CM ordering.

The idea of the algorithm is simple. For a graph $G' = (V', E')$, a pseudo diameter with end vertices d_1 and d_2 is calculated using an algorithm from Gibbs, Poole and Stockmeyer (1976). If pseudoverties b and g exist, they are used in place of d_1 and d_2 . Starting from the cut $\delta(U_1)$ with $U_1 = \{d_1\}$, we successively add vertices to U_1 where, at each step, the vertex v to be added is chosen among the closest available vertices to d_1 so as to maximize the decrease in the cut size $\delta(U_1 \cup \{v\}) - \delta(U_1)$. We stop when U_1 forms an equipartition of V' . The same procedure is then repeated starting from d_2 , and the better of the two partitions is chosen as initial solution.

5.7 Computational Results and Discussion

A computational study was conducted to test whether a global approach as exemplified by the divide-and-conquer *DC* heuristic could produce significantly smaller frontwidths than the reverse Cuthill-McKee (*CM*) approach. Two associated questions that arise concern (i) the effectiveness of using the ordering obtained by minimizing the modified frontwidth \overline{FRP} to calculate the standard frontwidth FRP , and (ii) the relative effectiveness with respect to solution quality and running time of the six different variants of the *DC* algorithm, obtained by combining the three different equipartition heuristics *KL*, *SE* and *SA*, and the two initial solution strategies *ICM* and *ISG*.

Each of the six variants is denoted here by an ordered pair (α, β) where α is the equipartition heuristic and β is the initial solution strategy.

The reverse Cuthill-McKee algorithm has been programmed by A. Couniot and is based upon the implementation given in George and Liu (1983).

The tests were carried out on a set of two and three dimensional meshes. The elements of the 2D meshes consisted of six-node triangles and nine-node quadrilaterals, and those of the 3D meshes were four-node tetrahedra. One scalar nodal value was assumed per node. The name of each mesh indicates its dimension and number of elements. Meshes *2D274a*, *2D562*, *2D608*, *2D666*, *2D1699*, *2D2175* and *3D3689* are shown in the Figure 5.8. Mesh *2D274b* has the same structure as mesh *2D274a* but the order in which the elements are stored in the input file is different. In mesh *2D274a*, the elements are in the *CM* ordering, while in *2D274b* the element ordering was optimized by hand. Mesh *2D210* discretizes fluid flow through a 4:1 abrupt contraction. Mesh *2D460* is a regular T-shape. Mesh *2D805* discretizes fluid flow between a single triangular cam and a cylindrical barrel. Mesh *2D918* is a minor modification of mesh *2D666*. Meshes *2D664* and *2D2169* are refinements of mesh *2D274a*, while mesh *3D973* was refined to obtain meshes *3D3689* (shown in Figure 5.8(g)) and *3D14392*.

Figure 5.8:

Figure 5.8:

Results are shown in Table 5.1a-b, respectively for 2D and 3D meshes. Here, we give the size of the meshes, the frontwidth FRP obtained with the reverse Cuthill-McKee heuristic (except for meshes $2D274b$ and $2D918$ for which the orderings have been improved "by hand"), the *best* frontwidth FRP obtained from the six variants of the Divide-and-Conquer heuristic, and the percentage decrease (if any) relative to the CM solution. We see that significant improvements of between 20% and 40% are obtained for about 3/4 of the meshes. The more than 40% improvement levels obtained with the complex 3D meshes are quite remarkable.

Below we discuss the results in more detail. First, we consider the two-dimensional meshes, and the use of \overline{FRP} to approximate FRP . In Table 5.2, we compare the values of the standard frontwidth FRP and the modified frontwidth \overline{FRP} obtained with CM and the three DC heuristics (using ICM to obtain the initial equipartitions).

We observe that the percentage reductions of FRP and \overline{FRP} are close for each variant of the DC algorithm. Our use of the modified frontwidth to derive an ordering of the elements thus appears justified.

Let us now consider whether using the starting procedure *ISG* developed in Section 5.6 produces better results than the initial ordering *ICM*. We show in Table 5.3 the values of \overline{FRP} and the percentage reductions of the six *DC* variants. It appears that whatever the equipartition heuristic used, neither *ICM* nor *ISG* dominates the other.

Table 5.3 also allows us to compare the equipartition heuristics *KL*, *SA* or *SE*. It appears that *SE* produces better solutions than *SA*, which in turn produces better solutions than *KL*. Note that Table 5.1 was constructed from Table 5.3 by choosing the best of the six results for each mesh.

A comparison of the *DC* heuristics should include considerations on *CPU* time and robustness. In Table 5.4, we show typical timings as observed on a Data General Aviion Server. The actual absolute importance of these timings can only be considered relative to the overall gains obtained from a reduction in frontwidth. The latter depend on the type of finite element problem (e.g. linear steady, nonlinear steady, transient) that one wishes to solve with a particular mesh. It is clear, however, that the *DC* heuristic using *KL* is an order of magnitude faster than the others. Also, *DC* using *SE* is generally faster than *DC* using *SA*.

As *SA* and *SE* are stochastic algorithms, we ran the corresponding heuristics ten times on the two problems *2D562* and *2D664* in order to assess their robustness. For problem *2D562*, the values of \overline{FRP} obtained with *SA* varied in the range $[35, 49]$ with mean 40.1, whereas with *SE* the range was $[35, 43]$ with mean 35.8. The corresponding values for *2D664* were range $[20, 25]$ and mean 23.4 for *SA*, and $[20, 24]$ with mean 21.3 for *SE*. The *CPU* times never varied by more than about 20%. Thus, if anything, *SE* appears to be somewhat better, faster and more stable than *SA*.

Finally, we present the results obtained for three medium-to-large 3D meshes. Here again the use of the modified frontwidth \overline{FRP} appears to be justified by the results. In Table 5.5 we present the values of the standard frontwidths obtained with CM and with the six variants of the DC heuristic.

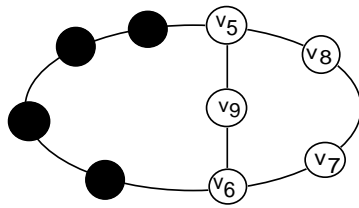
The CPU times for KL varied between 2 and 90 seconds, while for SE and SA they varied between 0.5 and 40 minutes. The improvements with all six DC heuristics are significant. We need, however, a larger mesh sample to distinguish between the different variants using SE and SA .

In conclusion, the various DC heuristics proposed in this thesis produce significantly smaller frontwidths than the reverse Cuthill-McKee algorithm for a variety of unstructured 2D and 3D meshes. Our results indicate that if one wants an ordering rapidly, one should use the two DC variants (KL, ICM) , (KL, ISG) based on the Kernighan-Lin algorithm. On the other hand, if solution quality is of primary concern, one should apply the two DC variants (SE, ICM) , (SE, ISG) based on Stochastic Evolution. The results also suggest that refinement of a mesh does not lead to deterioration in the quality of the DC heuristics (e.g., meshes 2D274a, 2D664 and 2D2169 are successive refinements of one another; the same comment holds for the 3D meshes used here).

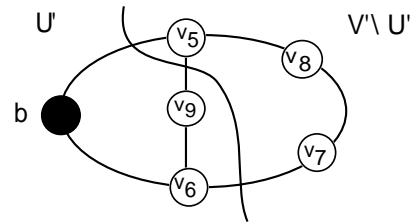
To verify that important savings in CPU time are obtained in the frontal method with a good element ordering, the following test has been carried out (Zone, 1993). Mesh 2D2169 has been used to discretize the flow of a polymeric fluid. This mesh has 47862 nodal variables and the (standard) frontwidths found by the Reverse Cuthill-McKee algorithm and the heuristic (KL, ICM) are 859 and 578, respectively. The CPU time necessary to solve this problem by Finite Elements on a SG IRIS Workstation was 19144 seconds using the Reverse Cuthill McKee ordering and 8028 seconds using the (KL, ICM) ordering. Therefore, the problem has been solved 2.38 times faster with the ordering produced by our heuristic. This result is close to the theoretical result that the CPU time is proportional to the square of the frontwidth since

$$\left(\frac{859}{578}\right)^2 = 2.21 \approx \frac{19144}{8028} = 2.38$$

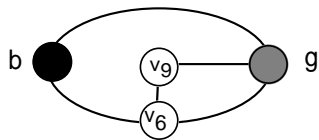
(a)



black: assigned to positions 1-4.
grey : no vertices assigned

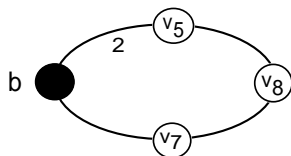


Outcome:
 v_6, v_9 in positions 5,6
 v_5, v_7, v_8 in positions 7-9



(b)

SUBPROBLEM 1
Black: assigned to positions 1-4
Grey: assigned to positions 7-9



(c)

SUBPROBLEM 2
Black: assigned to positions 1-6
Grey: no vertices assigned

Figure 5.6:

6. Conclusion

In this thesis we have studied three combinatorial optimization problems defined on graphs: the equipartition problem, the clustering or graph partitioning problem with capacity constraints and the cutwidth problem.

To tackle the equipartition problem we have chosen an exact approach based upon a cutting plane algorithm. Thus, we have investigated the facial structure of the equipartition polytope. We have introduced new classes of facet defining inequalities for this polytope. Two of them, namely the classes of PBC and suspended tree inequalities, generalize the cycle inequalities given in Conforti et al. (1990). We have found three other inequalities that define facets whose support graphs are composed of two connected components: a suspended tree and a complete odd subgraph on 3, 5 and 7 nodes, respectively. The computation of the coefficients of the variables in these inequalities is well defined for edges in the suspended tree component, but is more involved for edges in the second component. However, this suggests that the class of suspended tree inequalities can be further enlarged to a class of facet defining inequalities with supports given by two connected components as described above. We have shown how to transform all these inequalities to generate new classes of facet defining inequalities for the cut polytope.

We also present some results that suggest that the classes of PBC and suspended tree inequalities can probably be viewed as subclasses of a larger class of inequalities. We could not completely determine the necessary and sufficient conditions for an inequality in this large class to be valid for the equipartition polytope. But, since we have been able to find examples of such inequalities in which they define facets of the polytope, we believe that further investigations in this direction could be useful for better describing the equipartition polytope.

The exact approach based on cutting planes is taken again to tackle three different versions of the graph partitioning problem we considered. These problems are: the single

cluster problem, the graph partitioning problem with cardinality constraints and the graph partitioning problem with capacity constraints.

For the single cluster polytope, we generalize the class of tree inequalities given in Johnson et al. (1991). For this, we use arguments very similar to those that allowed us to generalize cycles with PBCs in the equipartition case.

For the graph partitioning problem with cardinality capacity constraints, we consider the case of equal cluster capacities. We give results on the dimension of the polytope and on facet defining inequalities. The arguments given in the equipartition case are repeated here to prove that inequalities with supports given by PBC, cycles and suspended trees define facets of the polytope. In each case the nodes of the supports of the inequalities form a cover (or some extension of a cover) for a single cluster. We have studied another inequality whose support is a cycle that covers multiple clusters. A procedure is given to strengthen this inequality which explores the chords of the cycle.

The third polytope we have studied is that of the graph partitioning problem where the cluster capacity constraints are given by knapsack constraints. Again, the clusters all have equal capacities. Valid inequalities are given whose supports derive from those of the inequalities introduced for the cardinality case. Another inequality has been presented, called the knapsack tree inequality, which is generated from valid inequalities for the knapsack polytope defined by the cluster capacity constraints.

The strength of the inequalities introduced for the equipartition and the graph partitioning polytopes has been tested computationally. For this we have designed and implemented heuristic separation routines for three classes of inequalities: the PBC, the cycle and the tree inequalities. The separation routine for PBC inequalities is able to identify violated inequalities only in a subclass of the entire class of PBC inequalities. These routines have been added to a branch-and-cut code called CLOP.

Three classes of problems have been used in our tests: mesh problems, compiler design problems and VLSI design problems. The mesh problems come from ordering and decomposition problems of Finite Elements meshes. The sample is composed of 23 instances (6 mesh, 11 compiler design and 6 VLSI design problems).

For the mesh and specially for the VLSI design problems, the sizes of the instances on which we have run the code are, up to now, of very small size compared to the instances arising in practice. The largest problems we have solved in each class are: (i) an equipartition problem on a graph with 148 nodes and 265 edges in the mesh class; (ii) a graph partitioning problem on a graph with 47 nodes and 99 edges (8 clusters) in the compiler design class and (iii) a graph partitioning problem on a graph with 43 nodes and 105 edges (4 clusters) in the VLSI design class.

Although we have been able to solve most of the instances in our sample, the number of inequalities generated to prove optimality is large and the CPU times are important considering the small sizes of the instances we have tested. The PBC and knapsack tree inequalities appeared as the most effective inequalities for tightening the formulation.

The results indicate that, for mesh problems, the PBC inequalities contribute significantly to improve the LP relaxations. Since the PBCs are at least 2-connected subgraphs, the PBC inequalities are difficult to separate in very sparse graphs such as those in the classes of compiler and VLSI design problems. This could explain why the PBC inequalities have not been of much use in solving problems in these two latter classes. The separation routine that we have implemented for PBCs suffers from two major drawbacks. The first is that it is too time consuming and the second is that it can only separate a subclass of the PBC inequalities. Therefore, it would be interesting to improve the separation routine so as to overcome these problems.

We have observed that the knapsack tree inequalities are crucial to solve partitioning problems on very sparse graphs with arbitrary (positive) node weights. However, the

number of these inequalities that is usually generated to prove optimality is large. We know that, in practice, the changes we have introduced to strengthen the knapsack tree inequalities are sometimes not sufficient to generate a violated inequality that cuts off a fractional point. Other strengthenings we have discussed in Chapter 3, though expensive in terms of processing time, should be added to the separation routine for knapsack tree inequalities.

The results we have obtained by varying the cluster capacities are inconclusive. It would be interesting to study the case when we have different cluster capacities and to make more tests for the case when the cluster capacities are equal, but large enough to allow a great number of feasible solutions.

It is clear from our computational experiments that we are still at a significant distance away from solving real world instances of graph partitioning problems with our branch-and-cut code. Several important valid inequalities are still missing and much more theoretical effort has to be made in finding new strong valid inequalities.

The cutwidth problem has been considered here as a relaxation of the frontwidth reduction problem (FRP) of Finite Elements meshes. We have shown how to formulate the FRP as a cutwidth problem and the graphs arising in this model are usually of very large size. Consequently, the Integer Programming formulation of the problem has a huge number of variables and this has discouraged us from any attempt to apply the cutting plane approach to tackle the cutwidth problem.

Thus, we have decided to use a heuristic method based upon a divide-and-conquer strategy in which equipartition subproblems are solved repeatedly. The algorithm resulting from this strategy is called the DC algorithm. Our goal was to produce better solutions to the FRP than those produced by the classical greedy algorithms. Different variants of the DC algorithm have been implemented. Each of these variants is characterised by the local search algorithm that is used to solve the equipartition subproblems and by the solution with which the local search starts.

Three heuristics have been implemented that solve equipartition problems. The first is the classical deterministic heuristic of Kernighan and Lin (KL) and the two others, namely the Simulated Annealing (SA) and Stochastic Evolution (SE) heuristics, are nondeterministic. Two different starting solutions have been tried in our computational experiments. The first, denoted by ICM, uses directly the ordering generated by the classical Reverse Cuthill-McKee algorithm, while the second, denoted by ISG, is produced by an algorithm we proposed in Section 5.6. By combining each local search heuristic with each of the two starting solution algorithms, we have six different variants of the DC algorithm: (KL,ICM), (KL,ISG), (SA,ICM), (SA,ISG), (SE,ICM) and (SE,ISG).

These variants of the DC algorithm have been tested on a sample composed of 13 two-dimensional meshes and 3 three-dimensional meshes. The largest of the two-dimensional meshes had 2175 elements and the largest of the three-dimensional meshes had 14392 elements. The results obtained with our DC algorithm have been compared to those produced by the standard Reverse Cuthill-McKee algorithm. Relative improvements in frontwidth are in the range 25-50% in most cases, which represents a significant 2-4 speedup of the Finite Element solver phase. Our results indicate that if one wants an ordering rapidly, one should use the two DC variants (KL,ICM) and (KL,ISG) based on the Kernighan and Lin heuristic for equipartition. On the other hand, if solution quality is of primary concern, one should apply the two DC variants (SE,ICM) and (SE,ISG) based on Stochastic Evolution. No conclusive results were obtained that allow us to conclude that one of the two starting solutions is better than the other.

For a Finite Element problem on a mesh containing 2169 elements and 47682 variables, we observed a saving of approximately 3 hours out of 5.5 hours in computation on a SG IRIS Workstation when the (KL,ICM) ordering is used instead of the Reverse Cuthill-McKee ordering.

The DC algorithm was used here for solving the cutwidth problem on a very special class of graphs arising from Finite Elements meshes. We have not verified if our algorithm

is effective in solving cutwidth problems for more general graphs. For the graphs we have treated, it was possible to devise routines to generate potentially good starting solutions that we believe had improved considerably the performance of the local search heuristics. Another relevant property of these graphs is that the nodes have degrees bounded by a small constant, and important savings in processing time can be achieved by exploiting this fact. These are two points to be taken into account if one is considering to apply the DC algorithm for other classes of graphs.

References

- K. Aardal (1992). On the Solution of One and Two-Level Capacitated Facility Location Problem by the Cutting Plane Approach, Doctoral Thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.
- E.H. Aghezzaf (1992). Optimal Constrained Rooted Subtrees and Partitioning Problems on Tree Graphs, Doctoral Thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.
- J.E. Akin and R.M. Purdue (1976). Element Resequencing for Frontal Solutions, in *The Mathematics of Finite Elements and Applications II* MAFELAP 1975, J.R. Whiteman editor, Academic Press.
- G. Andreatta, A. Basso, A. Caumo, and L. Deserti (1989). Un Problema di "Min-Cutwidth" Generalizzato e sue Applicazioni ad un FMS, paper presented at *Giornate di Lavoro A.I.R.O.*, Udine, Italy.
- C. Arbib (1988). A Polynomial Algorithm for Line-Graph Partitioning, *Information Processing Letters* **26**, 223-230.
- F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt (1988). An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design, *Operations Research* **36**, 493-513.
- F. Barahona, M. Grötschel, and A.R. Mahjoub (1985). Facets of the Bipartite Subgraph Polytope, *Mathematics of Operations Research* **10**, 340-358.
- F. Barahona and E. Maccioni (1982). On the Exact Ground States of Three-Dimensional Ising Spin Glasses, *Journal Phys* **A15**, L611-L615.

- F. Barahona and A.R. Mahjoub (1986). On the Cut Polytope, *Mathematical Programming* **36**, 157-173.
- E.R. Barnes (1982). An Algorithm for Partitioning the nodes of a Graph, *SIAM J. Alg. Disc. Meth.* **3(4)**, 541-550.
- J.A. Bondy and U.S.R. Murty (1976). *Graph Theory with Applications*, Macmillan.
- E. Boros and P.L. Hammer (1993). Cut Polytopes, Boolean Quadric Polytopes and Non-negative Quadratic Pseudo-Boolean Functions, *Mathematics of Operations Research* **18**, 245-253.
- T.N. Bui and A. Peck (1992). Partitioning Planar Graphs, *SIAM J. Computing* **21(2)**, 203-215.
- A. Bykat (1977). A Note on an Element Ordering Scheme, *Int. J. Num. Meth. Eng.* **11**, 194-198.
- S. Chopra (1991). The Graph Partition Polytope on Series Parallel and 4-Wheel Free Graphs, Kellogg Graduate School of Management, Northwestern University, Preprint.
- S. Chopra and M.R. Rao (1989a). The Partition Problem I: Formulations, Dimensions and Basic Facets, Working Paper No. 89-27, Stern School of Business, New York University.
- S. Chopra and M.R. Rao (1989b). The Partition Problem II: Valid Inequalities and Facets, Working Paper No. 89-26, Stern School of Business, New York University.
- S. Chopra and M.R. Rao. Facets of the K-Partition Polytope, Stern School of Business, New York University, Preprint.
- M. Conforti, M. Rao, and A. Sassano (1990a). The Equipartition Polytope I, *Mathematical Programming* **49**, 49-70.

- M. Conforti, M. Rao, and A. Sassano (1990b), The Equipartition Polytope II, *Mathematical Programming* **49**, 71-90.
- CPLEX Optimization, INC. (1990). Using the CPLEX Linear Optimizer.
- H.P. Crowder, E.L. Johnson, and M.W. Padberg (1983). Solving Large-Scale Zero-One Linear Programming Problems *Operations Research* **31**, 803-834.
- E. Cuthill and J. McKee (1969). Reducing Bandwidth of Sparse Symmetric Matrices, in *Proceedings of the ACM National Conference*, Association of Computer Machinery, New York, pp. 157-172.
- C. De Simone (1990). Lifting facets of the cut polytope, *Operations Research Letters* **9**, 341-344.
- C. De Simone, M. Deza, and M. Laurent (1989). Collapsing and Lifting for the Cut Cone, Report No. 265, IASI-CNR, Roma.
- C.C. de Souza, R. Keunings, L.A. Wolsey, and O. Zone (1992). A New Approach to Minimising the Frontwidth in Finite Element Calculations, CORE Discussion Paper No. 9255, Université Catholique de Louvain, Louvain-la-Neuve, Belgium. To appear in *Computer Methods in Applied Mechanics and Engineering*.
- C.C de Souza and M. Laurent (1991). Some New Classes of Facets for the Equicut Polytope, CORE Discussion Paper No. 9157, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.
- M. Deza, F. Fukuda, and M. Laurent (1989). The Inequicut Cone, Report No.89-04, GSSM, The University of Tsukuba, Tokyo.

- M. Deza, V.P. Grishukhin, and M. Laurent (1991). The Symmetries of the Cut Polytope and Some Relatives, in P. Gritzman and B. Sturmfels, editors, *Applied Geometry And Discrete Mathematics*, The Victor Klee Festschrift, volume 4 of DIMACS series in Discrete Mathematics and Theoretical Computer Science, pp. 205-220.
- M. Deza, M. Grötschel, and M. Laurent (1991). Complete Descriptions of Small Multicut Polytopes, in P. Gritzman and B. Sturmfels, editors, *Applied geometry And Discrete Mathematics*, The Victor Klee Festschrift, volume 4 of DIMACS series in Discrete Mathematics and Theoretical Computer Science, pp. 221- 252.
- M. Deza, M. Grötschel, and M. Laurent (1992). Clique Web Facets for Multicut Polytopes, *Mathematics of Operations Research* **17(3)**.
- M. Deza and M. Laurent (1989). The Even and Odd Cut Polytopes, Research Report B-231, Tokyo Institute of Technology.
- M. Deza and M. Laurent (1992a). Facets for the Cut Cone I, *Mathematical Programming* **56(2)**, 121-160.
- M. Deza and M. Laurent (1992b). Facets for the Cut Cone II, *Mathematical Programming* **56(2)**, 161-188.
- M. Deza and M. Laurent (1992c). New Results on Facets of the Cut Cone, *Journal of Combinatorics* **8**, 125-142.
- M. Deza and M. Laurent (1992d). Applications of Cut Polyhedra, Report BS-R9221, Centrum voor Wiskunde en Informatica, Department of Operations Research, Statistics and System Theory, Amsterdam.
- M. Deza, M. Laurent, and S. Poljak (1992). The Cut Cone III: On the Role of Triangle Facets, *Graphs and Combinatorics* **8**, 125-142.

- D. Dovlev and H. Trickey (1982). Embedding a Tree on a Line, IBM Technical Report RJ3368, San José, California.
- Duff, Erisman, and Reid (1986). *Direct Methods for Sparse Matrices*, Oxford Science Publications.
- J. Falkner, F. Rendl, and H. Wolkowicz (1992). A Computational Study of Graph Partitioning, Report 228, Technische Universität Graz, Austria.
- S.J. Fenves and K.H. Law (1983). A Two-Step Approach to Finite Element Ordering, *Int. J. Num. Meth. Eng.* **19**, 891-911.
- T. Feo, O. Goldschmidt, and M. Khellaf (1992). One-half approximation algorithms for the K-partition problem, *Operations Research* **40**, Supp., No. 1, S-170-S-173.
- C. Ferreira (1993). Private communication.
- C. Ferreira, A. Martin, and R. Weismantel (1992). CLOP: A Branch-and-Cut Code for Clustering Optimisation, unpublished manuscript.
- C. Ferreira, A. Martin, and R. Weismantel (1993). Private communication.
- C.M. Fiduccia and R.M. Mattheyses (1982). A Linear Time Heuristic for Improving Network Partitionings, in *Proceedings of the 19th Design Automation Conference*, Las Vegas, 175-181.
- M.R. Garey and D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman.
- A. George (1971). Computer Implementation of the Finite Element Method, STAN-CS-71-208, Computer Science Department, Stanford, CA.
- A. George and J.W.H. Liu (1983), *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall Series in Computational Mathematics, Prentice-Hall.

- A.M.H. Gerards (1985). Testing the Odd Bicycle Wheel Inequalities for the Bipartite Subgraph Polytope, *Mathematics of Operations Research*, **10**, 359-360.
- N.E. Gibbs, W.G. Poole, and P.K. Stockmeyer (1976). An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix, *SIAM J. Numerical Analysis*, Vol.**13**, No. 2, 236-250.
- F. Glover and M. Laguna (1992). Tabu Search, in *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Publishing.
- R.E. Gomory (1958). Outline of an Algorithm for Integer Solutions to Linear Programs, *Bull. of the American Math. Society* **64**, 275-278.
- M. Grötschel, M. Jünger, and G. Reinelt (1984). A Cutting Plane Algorithm for the Linear Ordering Problem, *Operations Research* **32**, 1195-1220.
- M. Grötschel, L. Lovasz, and A. Schrijver (1981). The ellipsoid Method and Its Consequences in Combinatorial Optimization, *Combinatorica* **1**, 169-197.
- M. Grötschel, C.L. Monma, and M. Stoer (1992). Computational Results with a Cutting Plane Algorithm for Designing Communication Networks with Low Connectivity Constraints, *Operations Research* **40**, 309-330.
- M. Grötschel and W.R. Pulleyblank (1981). Weakly Bipartite Graphs and the Max-Cut Problem, *Operations Research Letters* **1**, 23-77.
- M. Grötschel and Y. Wakabayashi (1987). Compositions of Facets of the Clique Partitioning Polytope, Report No. 18, Institut für Mathematik, Universität Augsburg.
- M. Grötschel and Y. Wakabayashi (1989). A Cutting Plane Algorithm for a Clustering Problem, *Mathematical Programming* **45**, 59-96.
- M. Grötschel and Y. Wakabayashi (1990). Facets of the Clique Partitioning Polytope, *Mathematical Programming* **47**, 367-387.

- F. Hadlock (1985). Finding a Maximum Cut of Planar Graph in Polynomial Time, *SIAM Journal of Computing* **4**, 221-225.
- D.S. Hochbaum and D.B. Shmoys (1985). An $O(|V^2|)$ Algorithm for the Planar 3-Cut Problem, *SIAM J. Alg. Disc. Meth.* **6**, 707-712.
- B.M. Irons (1970). A Frontal Solution Program for Finite Element Analysis, *Int. J. Num. Meth. Eng.* **2**, 5-32.
- D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon (1989). Optimization by Simulated Annealing: An Experimental Evaluation: Part I, Graph Partitioning, *Operations Research* Vol.**37**, No. 6, 865-892.
- D.S. Johnson, A. Mehrotra, and G.L. Nemhauser (1991). Min-Cut Clustering, Research Report COC-91-10, Georgia Institute of Technology.
- W. Kernighan and S. Lin (1970). An Efficient Heuristic Procedure for Partitioning Graphs, *Bell Systems Technical Journal* **49(2)**, 291-307.
- S. Kirkpatrick, C.D. Gellat, and M.P. Vecchi (1982). Optimization by Simulated Annealing, IBM Research Report RC 9359.
- P.J.M. van Laarhoven and E.H.L. Aarts (1987). *Simulated Annealing: Theory and Applications*, D. Reidel Publishing Company.
- T. Lengauer (1990). *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons.
- T. Liebling and P. Vaca (1991). Un Algorithme Polynomial Pour le Problème de l'Equi-partition Dans un Arbre, preprint.
- A. Martin (1993). Private communication.

- B. Mohar and S. Poljak (1992). Eigenvalues in Combinatorial Optimization, Preprint Series vol. 30, Department of Mathematics, University of Ljubljana, Slovenia.
- G.L. Nemhauser and L.A. Wolsey (1988). *Integer and combinatorial Optimization*, John Wiley & Sons.
- M.W. Padberg (1989). The Boolean Quadratic Polytope: Some Characteristics, Facets and Relatives, *Mathematical Programming* **45**, 139-172.
- M.W. Padberg and M. Grötschel (1985). Polyhedral Computations, in Lawler, Lenstra et al., pp. 307-360.
- M.W. Padberg and G. Rinaldi (1987). Optimization of a 532-City Traveling Salesman Problem by Branch and Cut, *Operations Research Letters* **6**, 1-8.
- H. Pina (1981). An Algorithm for Frontwidth Reduction, *Int. J. Num. Meth. Eng.* **17**, 1539-1546.
- S. Pissanetsky (1984). *Sparse Matrix Technology*, Academic Press.
- S. Poljak and F. Rendl (1991). Solving the Max-Cut Problem Using Eigenvalues, Report No. 199, Technische Universität Graz, Austria.
- A. Razzaque (1980). Automatic Reduction of Frontwidth for Finite Element Analysis, *Int. J. Num. Meth. Eng.* **15**, 1315-1324.
- F. Rendl and H. Wolkowicz (1991). Applications of Parametric Programming and Eigenvalue Maximization to the Quadratic Assignment Problem, Technical Report, University of Waterloo
- Y.G. Saab and V.B. Rao (1991). Combinatorial Optimization by Stochastic Evolution, *IEEE Transactions on Computer-Aided Design*, Vol.. **10**, No. 4, 525-535.

- S.W. Sloan and M.F. Randolph (1983). Automatic Element Reordering for Finite Element Analysis with Frontal Solution Schemes, *Int. J. Num. Meth. Eng.* **19**, 1153-1181.
- D. Vanderstraeten (1992). Génération Automatique d'une Décomposition de Domaine pour le Calcul par Éléments Finis sur Ordinateur Parallèle, Mémoire d'Ingénieur, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.
- D. Vanderstraeten and R. Keunings (1993). Optimized Partitioning of Unstructured Finite Element Meshes, Technical Report 93.32, Centre for Systems Engineering and Applied Mechanics, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.
- R. Weismantel (1993). Plazieren von Zellen: Theorie and Lösung eines quadratischen 0-1 Optimierungsproblem, Technical Report TR 92-3, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.
- R. Weismantel (1993). Private communication.
- R.D. Williams (1990). Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations, Report C3P913, California Institute of Technology.
- M. Yannakakis (1985). A Polynomial Algorithm for the Min-Cut Linear Arrangement of Trees, *Journal of the ACM* **32**(4), 950-988.
- O.C. Zienkiewicz and Morgan (1983). *Finite Elements and Approximation*, John Wiley & Sons.
- O.C. Zienkiewicz and R.L.T. Taylor (1989). *The Finite Element Method*, 2 Vols., McGraw-Hill.
- O. Zone (1993). Private communication.

NOTATION

$G = (V, E)$: graph with node set V and edge set E ;

\mathcal{K}_n : the complete graph on n nodes;

$V(S)$: the set of nodes spanned by the edges in S ;

$\delta(S)$: the set of edges with one endnode in S and the other endnode not in S ;

$|S|$: the number of elements in the set S ;

$\lceil x \rceil$: the smallest integer number larger than $x \in \mathbb{R}$;

$\lfloor x \rfloor$: the largest integer number smaller than $x \in \mathbb{R}$;

$\text{conv}(S)$: convex hull of the points in the set S ;

$\text{aff}(S)$: affine hull of the points in the set S ;

$\dim(S)$: dimension of the set S ;

$\mathbb{R}^{|S|}$: set of $|S|$ -dimensional real vectors whose components are indexed by the elements in the set S ;

$\mathbb{Z}^{|S|}$: set of $|S|$ -dimensional integer vectors whose components are indexed by the elements in the set S ;

$\mathbb{B}^{|S|}$: set of $|S|$ -dimensional 0-1 vectors whose components are indexed by the elements in the set S ;

$x(S)$: $\sum_{i \in S} x_i$;

$P_C(G)$: polytope describing the convex hull of the incidence vectors of cuts in the graph G ;

$P_{EC}(G)$: polytope describing the convex hull of the incidence vectors of equicuts in the graph G ;

$P_{MC}^{K,F}(G)$: polytope describing the convex hull of the incidence vectors of (K, F) multicuts in the graph G , that is, the multicuts corresponding to the partitions of G into K clusters of size at most F ;

$P_{MC}^{K,W}(G)$: polytope describing the convex hull of the incidence vectors of (K, W) multicuts in the graph G , that is, the multicuts corresponding to the partitions of G into K clusters such that the sum of the node weights within each cluster does not exceed W ;

LP: Linear Program;

IP: Integer Program;

0-1 IP: Integer Program in which all variables assume values in $\{0, 1\}$;

FCPA: fractional cutting plane algorithm;

PBC: path-block cycle;

\overline{Q} : see definitions in pages 47, 89, 103 and 127.

\overline{Q}_k : see definitions in page 59.

B & C: branch-and-cut;

CLOP: (CLustering OPTimization) the branch-and-cut code used in Chapter 4;

FRP : frontwidth reduction problem;

\overline{FRP} : modified frontwidth reduction problem;

DC: Divide-and-Conquer algorithm for the FRP ;

CM: Cuthill-McKee algorithm for the FRP ;

KL: Kernighan and Lin heuristic for the FRP ;

SA: Simulated Annealing heuristic for the FRP ;

SE: Stochastic Evolution heuristic for the FRP ;