UNIVERSIDAD DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Set Packing, Location and Related Problems

Empaquetamiento de Conjuntos, Localización y Problemas Relacionados

Dña. María de las Mercedes Pelegrín García

2019

A Blas

Resumen

Uno de los pasatiempos más divertidos de la infancia es hacer puzles. De pequeña, solía disfrutar intentando completar las imágenes de mis dibujos favoritos, para después enmarcarlos como prueba del reto completado. El juego era como sigue. Tenías un conjunto de piezas de forma irregular que representaban fragmentos de una imagen más grande, y el objetivo consistía en encajarlos unos con otros con el fin de recuperar la imagen original. Solíamos resolverlo en grupos, cada uno se centraba en una parte de la imagen y después juntábamos los fragmentos reconstruidos. Lo que ninguno sabíamos es que estábamos intentando resolver uno de los problemas de optimización combinatoria conocidos como "duros". Y aún menos podíamos imaginar que nuestra estrategia de resolución era una heurística para el problema del *bin packing*, el cual consiste en juntar el mayor número de piezas candidatas posibles dentro de un contenedor dado, el rectángulo de la imagen en nuestro querido juego.

No fue hasta mi época como estudiante de Matemáticas en la universidad cuando me di cuenta de la ubicuidad de los problemas de optimización combinatoria en el día a día. Uno de mis profesores, Alfredo Marín, contribuyó a ello en gran medida. "Piensa en el ascensor de la facultad; para funcionar, ha de resolver problemas combinatorios cada día"— solía decir. Entonces, ¿cómo es posible que los niños den con maneras de resolver uno de estos problemas duros que el profesor Marín encontraba tan fundamentales? Ciertamente, los puzles son instancias sencillas del *bin packing*, pues uno sabe de antemano que todas las piezas encajarán hasta formar el rectángulo de la imagen. Sin embargo, no cabe duda de que los fragmentos de la imagen representados en cada pieza del puzle son de vital importancia para poder completarlo. De hecho, resultan decisivos cuando el tamaño del puzle aumenta y con él el número de piezas y formas de ensamblarlas. En general, los problemas combinatorios se caracterizan por su crecimiento exponencial en complejidad con el aumento de su tamaño. Esto significa que, aunque encontrar una solución para una instancia pequeña pueda resultar fácil (inclusive una que sea óptima), resolver el problema a partir de cierto tamaño puede llevar tiempo ilimitado, incluso usando un ordenador potente.

Existen gran variedad de técnicas para abordar la resolución de problemas de optimización combinatoria, que pueden ser clasificadas *grosso modo* en dos categorías: los métodos exactos y los no exactos, que incluyen los algoritmos de aproximación y los heurísticos. Los métodos exactos buscan obtener la mejor solución posible, mientras que el resto simplemente buscan "buenas" soluciones. Aunque heurísticas es el término más utilizado, podemos distinguir entre los métodos no exactos que proporcionan una estimación de la diferencia entre el valor encontrado y el óptimo, llamados algoritmos de aproximación, y aquellos sin garantía sobre la precisión alcanzada, conocidos como heurísticos. Los métodos exactos están ligados a una disciplina de la Matemática Aplicada, que se conoce como Programación Matemática. Esta estudia los llamados *programas matemáticos* o *formulaciones matemáticas*, los cuales son simplemente un tipo de modelo matemático para problemas de optimización.

Esta tesis es sobre métodos exactos para problemas de optimización combinatoria, consti-

tuyendo la Programación Matemática el marco de trabajo común de los diferentes capítulos. Un programa matemático consiste en una función objetivo y una región factible, normalmente contenida en $\mathbb{R}^n$, donde se busca el máximo o mínimo de la función objetivo. Este escenario da cabida al desarrollo de investigación tanto en la matemática pura como en la aplicada. Los programas matemáticos se relacionan de manera directa con el Análisis y la Geometría; el estudio de programas cuya región factible es discreta interactúa con el Álgebra. El aspecto aplicado de la Programación Matemática es probablemente el más evidente, y hace de esta disciplina un área prometedora también para los ingenieros. Los problemas de optimización combinatoria se modelizan con formulaciones llamadas enteras y sus regiones factibles son discretas. En el caso de $\mathbb{R}^n$, la región factible de un problema combinatorio consiste en puntos cuyas coordenadas son números enteros. El estudio teórico de una formulación entera tiene que ver con su *politopo entero*, es decir, la envolvente convexa de sus puntos factibles. La mayoría de capítulos contenidos en esta tesis se centran en el estudio de hiperplanos para describir politopos enteros, así como de herramientas computacionales que permitan incorporarlos a los algoritmos de optimización existentes.

Este trabajo surge del estudio de distintos problemas de optimización combinatoria, que están relacionados de una forma u otra. Se encuentra dividido en dos partes, precedidas por un capítulo que introduce los conceptos fundamentales que se usarán a lo largo de la tesis. La primera parte se centra en el estudio de formulaciones de *empaquetamiento de conjuntos*, las cuales son de vital importancia en la Programación Entera. El empaquetamiento de conjuntos está estrechamente ligado al problema de *particionamiento de conjuntos*, uno de los ejemplos paradigmáticos en optimización combinatoria. El problema de particionamiento de conjuntos consiste en encontrar una partición de mínimo coste de un conjunto de objetos y de hecho es equivalente al problema de empaquetamiento de conjuntos (ver Balas & Padberg, 1976). Estos sirven de marco común para problemas de optimización como planificación de máquinas (ver Sousa & Wolsey, 1992), subastas combinatorias (ver Escudero et al., 2009) y gestión del tráfico en redes (ver Rossi & Smriglio, 2001). Otros problemas combinatorios relacionados y bien conocidos son el empaquetamiento de nodos, coloreado de grafos y clique máximo. La segunda parte de la tesis se sitúa algo más lejos de la programación entera pura, pero sigue estando relacionada con el empaquetamiento de conjuntos. En ella se estudian distintos problemas de optimización que surgen en una amplia variedad de disciplinas: estimación de poblaciones en Genética, etiquetado de mapas en Cartografía e identificación de grupos de influencia en Análisis de Redes.

La primera parte de esta tesis está compuesta de tres capítulos. El primero se centra en aspectos generales del empaquetamiento de conjuntos, mientras que los otros dos estudian instancias particulares de este problema que surgen en el contexto de la Teoría de la Localización.

En el primer capítulo, se introduce un nuevo procedimiento para obtener *facetas* del politopo de empaquetamiento de conjuntos. Como resultado adicional, se presentan dos nuevas familias de *grafos definidores de facetas*. El citado procedimiento pertenece a la clase de los llamados *levantamientos* de desigualdades. Se trata de una generalización de un teorema de levantamiento que se introdujo en los 80s. Este teorema fue encontrado al revisar las técnicas de levantamiento existentes para el problema de localización de plantas sin capacidades, el cual se quería generalizar. Nos dimos cuenta de que el teorema de levantamiento anterior podía ser extendido para cubrir la nueva variante del problema. El teorema resultante se puede aplicar en general a cualquier empaquetamiento de conjuntos que satisfaga ciertas condiciones necesarias, las cuales son respetadas en particular por nuestro problema de localización. También permite obtener nuevas estructuras dentro del llamado *grafo conflicto* que pasan a formar parte de la

lista de grafos definidores de facetas del empaquetamiento de conjuntos, la cual incluye los conocidos *cliques* y los *agujeros impares*.

El siguiente capítulo está dedicado a la propuesta y estudio de una nueva generalización del problema de localización sin capacidades. Este es un problema conocido de localización que consiste en decidir sobre el número de servicios a instalar y sobre los usuarios asignados a cada uno de estos servicios. En nuestra propuesta, introducimos una componente adicional en el modelo, concretamente incompatibilidades entre algunos pares de usuarios que impiden que se asignen al mismo servicio. Modelizar la nueva variante del problema desemboca en una formulación de empaquetamiento de conjuntos. Nos centramos en el estudio de las caras del politopo de empaquetamiento resultante, derivando varias familias de facetas. El teorema de levantamiento del capítulo anterior resulta de ayuda para obtener algunas de ellas. Asimismo, se desarrollan estrategias para manejar las facetas encontradas en un esquema de ramificación y cortes y se comprueba su rendimiento mediante un estudio computacional.

El último capítulo dentro de la primera parte de la tesis estudia una segunda variante del problema de localización de plantas sin capacidades. En este caso, consideramos que cada usuario se asigna a dos servicios distintos y que algunos pares de usuarios deben compartir al menos uno de ellos. Se proponen dos formulaciones, basadas en el modelado estándar de problemas de doble asignación. Después de compararlas desde el punto de vista teórico, una de ellas, la cual es de empaquetamiento de conjuntos, se estudia más en profundidad. Por un lado, se examinan las facetas de clique de esta formulación y por otro se demuestra la complejidad computacional teórica de incorporarlas al algoritmo de resolución. Finalmente, se propone una heurística para abordar este último problema. El capítulo se cierra con un estudio computacional que corrobora la comparación teórica de las formulaciones y prueba la utilidad de la heurística para manejar las facetas de clique.

La segunda parte de la tesis reúne tres problemas que vienen de dominios aparentemente muy distintos, pero que en realidad guardan relación con el empaquetamiento de conjuntos, la localización o ambos. Se trata del haplotipaje de organismos diploides, el etiquetado óptimo de mapas y la detección de grupos de nodos relevantes en redes sociales. En todos los casos, se proponen formulaciones matemáticas para abordar el problema, se estudian posibles formas de reforzar las formulaciones y se lleva a cabo la implementación y prueba de los modelos.

En el primer capítulo de la segunda parte, se estudia un problema de optimización combinatoria en un grafo, para el que se conocen aplicaciones en el contexto del haplotipaje de organismos diploides. El haplotipaje está relacionado con la estimación de poblaciones: dada la información genética de un conjunto de individuos, se ha de estimar el conjunto mínimo de ancestros que explican la población actual. La reconstrucción del *grafo raíz* a partir del grafo que codifica las relaciones de consistencia entre individuos resulta crucial para abordar el problema. Sin embargo, dicha reconstrucción a veces requiere pasar por alto algunas de las relaciones de consistencia, es decir, eliminar algunas de las aristas del citado grafo. Entonces surge un problema de optimización combinatoria que consiste en decidir qué aristas han de ser eliminadas con el fin de permitir la reconstrucción del grafo raíz al mismo tiempo que se preserva la estructura del grafo original en la mayor medida posible. En este capítulo, se muestra la conexión que presenta este problema con las dos variantes del problema de localización de plantas presentadas en la primera parte de la tesis. Concretamente, cuando se combinan ambas variantes, el modelo obtenido tiene solución factible si y sólo si es posible reconstruir cierto grafo raíz. El problema estudiado en este capítulo no es nuevo y ha sido previamente modelizado mediante una formulación matemática. Nuestra contribución en este caso consiste en proponer varias formulaciones alternativas y diferentes familias de *desigualdades válidas*. Exploramos la relación entre las distintas formulaciones y las comparamos desde el punto de

vista teórico y computacional.

El siguiente capítulo aborda el etiquetado óptimo de mapas, que consiste en asignar etiquetas a un conjunto de puntos o áreas distinguidos en un mapa. La función objetivo varía entre las distintas aplicaciones, pero está relacionada frecuentemente con el número de solapamientos entre etiquetas, la ambigüedad del etiquetado y la distribución equilibrada o jerárquica de las etiquetas. Aunque las heurísticas para el etiquetado de mapas son abundantes, la Programación Matemática también ha contribuido a abordar este problema. Sin embargo, los modelos exactos existentes no tratan de forma efectiva el problema de la ambigüedad. En este capítulo, proponemos varias formulaciones para reducir la ambigüedad en los etiquetados de mapas, algunas de ellas inspiradas en los modelos ordenados concebidos originalmente para problemas de localización. El parecido entre el etiquetado de mapas y los problemas de localización motivó el uso de la misma idea que en los modelos de mediana ordenada. La decisión a efectuar en el etiquetado, dónde localizar una serie de elementos en un espacio plano, es la configuración típica de un problema de localización. De hecho a cada zona del mapa se le asigna una etiqueta de la misma forma que a cada usuario se le asigna un servicio en los problemas de localización. Los modelos ordenados propuestos proporcionan una alternativa flexible para el etiquetado de mapas, que permite que el decisor determine la penalización relativa de la primera, la segunda... etiqueta más ambigua en la solución.

En el último capítulo, se estudia cómo identificar a los miembros más influyentes dentro de una red. Estos se entienden como aquellos nodos que juegan un papel distinguido teniendo además áreas de influencia distintas. Las medidas de centralidad clásicas como *degree*, *closeness*, *betweenness* o *eigenvector* no sirven a la hora de determinar la relevancia conjunta de un grupo de nodos. Mientras que algunas de estas medidas se han adaptado para abordar el problema de la relevancia grupal, *eigenvector* ha permanecido sin ser extendida al nuevo problema. En el modelo propuesto, se adapta *eigenvector* para identificar el grupo de mayor influencia, combinándolo con un particionamiento o *clustering* de la red. Esto permite controlar las áreas de influencia de los individuos identificados como relevantes, las cuales coinciden con los grupos o clústeres y se reinterpretan como comunidades. Los miembros del grupo óptimo son aquellos nodos de mayor componente *eigenvector* dentro de su clúster. Modelizar esta idea a través de una formulación matemática conlleva el manejo de ecuaciones no lineales, las cuales se linearizarán para conseguir una formulación más manejable para el problema. El modelo propuesto finalmente revela el grupo de mayor influencia en la red, formado por los centroides de cada clúster, junto con sus comunidades, que coinciden con los clústeres. Los experimentos realizados con redes reales de tamaño pequeño producen interesantes resultados que revelan nodos previamente desapercibidos como miembros del grupo de influencia. Además, los clústeres identificados concuerdan con el conocimiento existente sobre la estructura de comunidad de las redes de prueba. El estudio realizado con redes sintéticas de mayor tamaño demuestra una escalabilidad adecuada, siendo el modelo capaz de obtener soluciones óptimas para redes con cientos de nodos y miles de aristas.

Preface

One of the amusing pastimes of childhood is solving jigsaw puzzles. When I was a child, I used to have fun trying to complete the images of my favourite cartoons. The game was as follows. You were given a set of pieces with irregular shapes and different fragments of a bigger image printed on them. The goal was to find the pieces that fit together to eventually recover the bigger image. We used to solve it in groups: each focused on one side of the image, and then we merged the resulting compositions. What none of us knew— and most people are not aware of— is that we were trying to solve a hard combinatorial optimization problem. And little idea did we have of the fact that we were implementing a heuristic for the *bin packing* problem, which consists of cramming as many candidate elements as possible into a given container, the rectangle of the image in our cherished game.

It was not until I studied Mathematics that I became aware of the ubiquity of combinatorial optimization in our daily lives. One of my professors, Alfredo Marín, largely contributed to this. "Think about the faculty's elevator; it has to solve combinatorial problems to work everyday"— he used to say. Then, how can children come up with working strategies for one of these hard problems that Prof. Marín found so crucial? Certainly, jigsaw puzzles are easy instances of bin packing in a sense, since one knows beforehand that all the pieces will eventually fit in the rectangle of the image. However, the fragments printed on the pieces are the real reason for children's success. They become decisive as the size of the puzzle increases and with it the number of pieces and possible matches. In general, combinatorial problems are characterized by an exponential growth in complexity with the increase of their size. This means that finding a solution of an small instance can be easy—including an optimal one— but solving the problem in general can take unlimited time, even with a computer.

There are wide ranging strategies to approach combinatorial optimization problems, which can be roughly classified into two categories: exact procedures and non-exact ones, which includes approximation and heuristic algorithms. The former define methods to obtain best possible solution(s), while the latter just seek for good ones. Even if heuristic is the most widespread term, we can distinguish between non-exact approaches that provide an estimation on the difference between the returned value and the optimum— approximation algorithms— and those with no guarantee on accuracy— simply called heuristics. Exact approaches are tied to a discipline within Applied Mathematics, called Mathematical Programming. This discipline studies *mathematical programs*, which are mathematical models for optimization problems.

This thesis is on exact approaches for combinatorial problems, being Mathematical Programming the common working framework throughout the different chapters. The main components of a mathematical program are an objective function and a feasible region, usually contained in $\mathbb{R}^n$, where the maximum or minimum of the objective function is to be found. This setup opens a world of research possibilities for mathematicians from pure to applied interests. Mathematical programs readily concern Analysis and Geometry; the study of programs with discrete feasible region interacts with Algebra. The applied side of Mathematical

Programming is probably the most evident— it was conceived to solve practical problems— and makes this discipline a very promising domain for engineers as well. Combinatorial optimization problems are modeled with integer programs, whose feasible regions are made of integer points. Theoretical study of an integer program concerns its *integer polytope*, that is, the convex hull of its feasible solutions. Hyperplanes to describe integer polytopes and computational tools to incorporate them to optimization algorithms will be the focus of most of the chapters contained herein.

This work emerges from the study of several combinatorial optimization problems, which are related in one way or another. It is divided into two parts, which are preceded by a chapter that introduces the basic concepts that will be used throughout the dissertation. The first part of the thesis is devoted to the study of *set packing* programs, which are of crucial relevance in Integer Programming. Set packing is a close relative of *set partitioning*, one of the paradigmatic examples within combinatorial problems. Set partitioning consists of making a minimum cost partition of a set of objects and is in fact equivalent to set packing (see Balas & Padberg, 1976). They serve as common framework for optimization problems such as scheduling (see Sousa & Wolsey, 1992), combinatorial auctions (see Escudero et al., 2009) and traffic network congestion (see Rossi & Smriglio, 2001). Other related well-known combinatorial problems include node packing, graph coloring and maximum clique. The second part of the thesis is a bit further from pure integer programming, but still close to set packing. We will study different combinatorial problems arising in a wide range of disciplines. These problems are population estimation in Genetics, map labeling in Cartography and key nodes identification in Network Analysis.

The first part of the thesis is made of three chapters. The first of them concerns general aspects of set packing, while the other two study particular instances of this problem arising in the context of Locational Analysis.

In Chapter 1, we introduce a new procedure to obtain *facets* of the set packing polytope, and present two new families of *facet defining graphs* as a byproduct. This procedure belongs to the class of so-called inequality *liftings*. It is a generalization of a previous lifting theorem that was introduced in the 80s. The latter was found when reviewing lifting theorems for the uncapacitated plant facility location problem, which we wanted to generalize. We realized that the old lifting theorem could be extended to cover our problem variant. The resulting theorem is applicable in general to any set packing that satisfies some necessary conditions, which are met in particular by our location problem. It also allows to obtain new structures within the *conflict graph* that are incorporated to the list of known facet defining subgraphs, which includes well-known *cliques* and *odd holes*.

Chapter 2 is devoted to a generalization of the uncapacitated plant facility location problem. This is a well-known problem in locational analysis that consists of deciding on a number of services to be installed and on clients allocation to the facilities. We propose to introduce an additional component in the model, namely incompatibilities between clients that forbid some pairs to be allocated to the same facility. Modeling the new variant of the problem results into a set packing formulation. We focus on the study of the facial structure of the resulting packing polytope, deriving various families of facets. The lifting theorem of previous chapter is used to obtain some of them. Strategies to manage the devised facets within a branch and cut scheme are given and their performance is tested in a computational study.

Chapter 3 is the last in the first part of the dissertation. It presents a second variant of the uncapacitated plant facility location problem. Here, double assignment is consider and some pairs of clients have to share at least one of their two facilities. Two formulations, based on standard modeling of double assignment, are proposed. They are theoretically compared and

one of them, which is a set packing, is further studied. All the clique facets of the formulation are uncovered. The computational complexity of managing them within the solving scheme is proved to be NP-hard and a heuristic algorithm is then proposed. The chapter is closed with computational experience that supports the theoretical comparison between formulations and proves the utility of the clique facets heuristic.

The second part of the thesis gathers three problems from domains that are apparently very different but in fact related to set packing, facility location or both. These include haplotyping of diploid organisms, optimal map design and identification of key members in social networks. In all the cases, mathematical programming formulations of the problems are proposed, possible enhancement are studied and computational tests are conducted.

In Chapter 4, we study a combinatorial optimization problem on a graph that is known to have applications in haplotype phasing of diploid organisms. Haplotype phasing concerns population estimation: given some genetic information of a set of individuals, one has to estimate the minimum set of ancestors that explains the current population. *Root graph* reconstruction from the graph that encodes individuals consistency relations is crucial to approach the problem. However, this reconstruction sometimes requires to overlook some of the consistency relations, that is, deleting some edges of the mentioned graph. The combinatorial problem is then to decide which edges to delete so that the root graph reconstruction is allowed while the graph encoding consistency relations is altered as little as possible. We show interesting connections between this problem and the facility location problems of chapters 2 and 3. When both variants of the uncapacitated plant location problem are put together, the resulting model has a feasible solution if and only if it allows some root graph reconstruction. The problem studied in this chapter is not new and was previously formulated with a mathematical program. Our main contribution here consists of several alternative formulations and different families of *valid inequalities*. We explore the relation between formulations and provide both theoretical and computational comparative analysis.

Chapter 5 addresses optimal map labeling, which consists of optimally attaching a label to every distinguished point or area on a map. Objective functions vary between applications, but often take into account the number of overlaps between labels, the ambiguity of the labeling, balanced distribution or hierarchical representation. Even though heuristics for map labeling abound, some authors have also contributed with mathematical programming models for the problem. However, existing exact approaches do not effectively address ambiguity reduction. In this chapter, we propose several integer programs for unambiguous map labeling. Some of them are inspired in ordering models that were conceived for facility location. This was motivated by the observation that map labeling can be viewed as a location problem. The decision to make in map labeling, where to locate a set of elements on a planar space, is the typical setup of facility location. Also, each region of the map is given one label, while each client is allocated to certain facility. Ordering models provide a flexible alternative, in which decision maker is allowed to decide on the relative importance of the first, the second... most ambiguous label in the solution.

In Chapter 6, we study the problem of identifying a group of key nodes in a network. These are understood as members that play a distinguished role within the network and have disjoint areas of influence. Measures of individuals relevance such as *degree*, *closeness*, *betweenness* or *eigenvector centrality* fail to discern directly the relevance of a group of nodes. While some of these measures have been adapted to address group relevance, eigenvector centrality remains unexplored. We adapt eigenvector centrality to identify the group of most relevant nodes in a network. In our approach, eigenvector computation is embedded in a clustering procedure. This allows to control the spheres of influence of the key members, which are identified with the

clusters and reinterpreted as network communities. Key members are nodes with maximum eigenvector centrality within their clusters. Modeling this idea with mathematical optimization variables involves highly non-linear equations, which are linearized to produce a mixed-integer linear programming formulation for the problem. Our model uncovers the group of most relevant nodes in the network, the clusters centroids, and their communities, identified with the clusters. Experiments on real-life networks of small size show interesting results that reveal previously unnoticed key members. Additionally, clusters are consistent with previous knowledge on the community structure of the networks. Our computational experience on larger synthetic networks demonstrates an adequate scalability of the method, which is able to find optimal solutions for networks of hundreds of nodes and thousands of links.

Publications and collaborations Some of the original content of this thesis has been submitted to international journals for publication. Such works have been written as a result of collaborations with recognized researchers in the field other than Prof. Marín. The following lines are to answer the Four Ws—*who*, *what*, *when*, *where*—of the different topics covered in this thesis— the reader should have found an answer for the *why* above.

1. The study of optimal map labelings, addressed in Chapter 5, started with my Master's thesis under Prof. Marín and culminated within the first years of doctoral studies. In 2018, "Towards unambiguous map labeling: An integer programming approach" was published in *Expert Systems with Applications*, Marín & Pelegrín (2018b).

2. As a PhD student in its early stage, research focused on set packing and its applications to facility location. Jointly with my PhD supervisor Prof. Marín, two works emerged from this phase. First, "A new lifting theorem for vertex packing" was published in *Optimization Letters*, Marín & Pelegrín (2018a). Second, " Adding incompatibilities to the simple plant location problem: Formulation, facets and computational experience" was published in *Computers and Operations Research*, Marín & Pelegrín (2019). Part of the content of these publications is reproduced in Chapter 1 and Chapter 2, respectively.

3. In a next stage, collaborations with Prof. Labbé started. This includes a research visit to the Université Libre the Bruxelles on September 2017 that lasted for three months. Results of the collaboration are collected in Chapter 4. A manuscript was also prepared and submitted to the *European Journal of Operational Research*, which has already received positive reviews.

4. During the last phase of my doctoral studies, a second visit was scheduled, this time to the Instituto de Matemáticas de la Universidad de Sevilla. Last three months of 2018, I worked in collaboration with Prof. Carrizosa. The topic was key nodes identification in social networks, which is addressed in Chapter 6. As an outcome of this phase, a manuscript was submitted to *IEEE Systems Journal*, which has been advised publication subject to changes.

Acknowledgments

This doctoral thesis has been developed thanks to one of the scholarships that the *Ministerio de Educación, Cultura y Deporte* (Spanish Ministry of Education) awards every year with its call *Becas de Formación del Profesorado Universitario*. I do want to acknowledge the *Ministerio de Educación, Cultura y Deporte* for awarding this project one of the scholarships in 2015, ref. number FPU15/05883. I would also like to thank the researchers that supported my application, Alfredo Marín, Emilio Carrizosa and Mercedes Landete.

In addition, this work has been indirectly funded by the *Ministerio de Economía y Competitividad*, project MTM2015-65915-R and the *Fundación Séneca de la Consejería de Educación de la Comunidad Autónoma de la Región de Murcia*, project 19320/PI/14, which provided support to participate in scientific conferences, meetings and courses, being this essential for the dissemination of the results.

I owe my greatest gratitude to my supervisor, Alfredo Marín. He integrated me as part of his team at the *Universidad de Murcia* and introduced me into the world of research. With his trust from the outset I grew both professionally and personally. His technical wisdom and guidance have been crucial to complete this journey; at the same time, his familiarity and sincerity made me feel welcomed all the way through. It has been a privilege learning from him. Another person that supported me from the very beginning is Emilio Carrizosa, who became one of the *key members* in my way. I am indebted to Emilio for his generosity and for expanding my horizons, advising and encouraging me at different steps in my career. Later on, I was fortunate to meet Martine Labbé, who hosted me in the *Graphs and Mathematical Optimization* research group at the *Université Libre the Bruxelles*. There, not only I collaborated with Martine but also enjoyed the nice atmosphere of the group and participated from the activities they organised. I sincerely thank her for the great opportunity.

As an undergraduate, some of my lecturers have helped as a guidance and source of inspiration. I am deeply grateful to José Manuel García for the motivating talks and good advise. Also to Guido Sciavicco, for revealing me the beauty of the Theory of Computation and believing in my aptitudes.

One of the best things of research is the opportunity of meeting and keeping in contact with wonderful people. During these years, I have had the pleasure of being part of the warm and cheerful *IMUS family*, enjoying escape rooms and hiking with my colleagues at *Lázaro Cánovas* and sharing jokes and coffees with the *ULB lunch group*. From conferences and courses, I also take with me good friends. I thank all of them for making these years one of the most enjoyable.

This thesis would have never been possible without the love and encouragement of my family, María Dolores, Blas and Juan Diego. María Dolores is the one that celebrates my achievements as her own and my unconditional support in the toughest moments. Blas was the first who transferred me the taste for mathematics and taught me the value of effort. Juan Diego with his good judgement is my greatest inspiration.

Mercedes Pelegrín García, Murcia 2019

Contents

Concepts

Operations Research is a discipline devoted to improve decision making. One of its branches, Mathematical Programming or Mathematical Optimization, is the common framework to address the full range of problems studied in this thesis. In this preliminary chapter, we describe essential notions inside the field, aiming at providing non-expert audience with a comprehensive conceptual review and introducing the reader to the topics, terminology and mathematical notation in the thesis.

0.1 Integer Programming

A mathematical programming model consists of a finite set of *decision variables* bounded by a finite set of inequalities called *constraints* and an *objective function* of the decision variables to minimize or maximize. Such model is usually called program or formulation. Linear Programming studies those mathematical programs with linear constraints and objective function. A generic linear program (LP) reads

$$\min\{f(t) : At \leq b,\ t \in \mathbb{R}^n\},$$

where n is the number of decision variables, $t \in \mathbb{R}^n$ is the vector of such variables, $f : \mathbb{R}^n \to \mathbb{R}$ is linear, and $At \leq b$ is a system of linear inequalities. The set

$$\{t \in \mathbb{R}^n : At \leq b\}$$

is called the *feasible region* of the LP, which is a polyhedron in general or a polytope when bounded. An LP where some decision variables only take integer values,

$$\min\{f(t^1, t^2) : At^1 + Bt^2 \leq b,\ t^1 \in \mathbb{R}^{n_1}, t^2 \in \mathbb{Z}^{n_2}\},$$

where the number of variables is $n = n_1 + n_2$, is called mixed-integer. When $n_1 = 0$, we speak of an integer linear program (ILP) or simply an integer program (IP).

When all decision variables are integer, the feasible region becomes a discrete set of points in $\mathbb{R}^n$. This fact may suggest that integer programs are a simple subject, but they can turn certifiably hard. Just think of vertex coloring, maximum clique, Hamiltonian circuit, vertex covering or partitioning, all of them members of the IP family and NP-complete problems (see Garey & Johnson, 1979). Not by accident, Integer Programming is frequently identified with Combinatorial Optimization, which concerns the study of optimization problems of combinatorial nature. This reveals the very nature of integer programs, whose feasible regions, even if finite, usually feature combinatorial explosions. Optimizing over integer variables needs for strategies and methods according to the complexity of these problems.

Here, we outline most basic mathematical properties of integer programs and some solving techniques. We extend our review to a particular family of integer programs, namely set packing. More extensive introductions to these topics can be consulted in Balas & Padberg (1976); Wolsey (1998); Conforti et al. (2014).

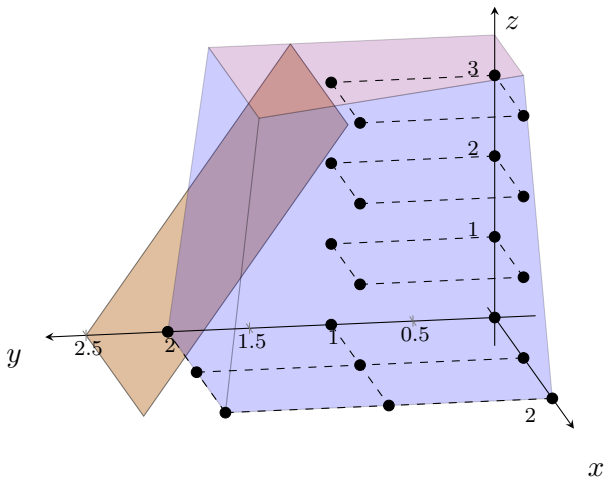


Figure 1: A linear relaxation polytope and a cutting plane

0.1.1 Polyhedral Theory

Most solving strategies for IPs work with the linear relaxation of the problem, which results after removing integrity conditions on the variables. These approaches substantiate on the general conception that solving an LP is a simple task (however, it is still an open question whether there is a polynomial time algorithm to solve linear programs; see Schrijver, 1998).

While the same IP can be described by different systems of inequalities, the feasible regions of the linear relaxations of such systems differ. Moreover, it is usually the case that the optimal objective value, which is a bound on the IP optimal value, is also different for the relaxations. This, of course, has an impact on the solving strategy, which strongly depends on the bounds obtained from the problem relaxation. The tighter the polyhedron of the relaxation is, the better the bounds on the IP objective that are obtained. A common practice to have tighter polyhedra is to add valid inequalities to the IP formulation that remove fractional solutions.

Definition 0.1 (*Valid inequality*)**.** An inequality $\pi t \leq \pi_0$ is valid for an IP $\min\{f(t) : At \leq b,\ t \in \mathbb{Z}^n\}$ if it is satisfied by every point of the feasible region. We will say that the inequality is proper when there exists $\bar{t} \in \mathbb{R}^n$ such that $A\bar{t} \leq b$ and $\pi\bar{t} > \pi_0$. △

Proper valid inequalities are violated by at least one solution of the linear relaxation but do not alter the feasible region of the IP. These inequalities are usually called *cuts*, because they cut the polyhedron of the linear relaxation without changing the IP. Figure 1 shows a cutting plane (in brown) of a linear relaxation polytope (in blue). The feasible solutions correspond to the integer points inside the blue region.

Since proper valid inequalities are actually the interesting ones, they are frequently called just valid. Especial cases are those that cannot be improved, i.e., they are not dominated by other valid inequalities.

Definition 0.2 (*Dominated inequality*)**.** Suppose that $\pi t \leq \pi_0$ and $\rho t \leq \rho_0$ are two valid inequalities for the same IP, $\min\{f(t) : At \leq b,\ t \in \mathbb{Z}^n\}$. We will say that $\pi t \leq \pi_0$ is dominated by $\rho t \leq \rho_0$ if every feasible solution of the linear relaxation, $\bar{t} \in \mathbb{R}^n$ such that $\rho\bar{t} \leq \rho_0$ also satisfies $\pi\bar{t} \leq \pi_0$. △

When one inequality dominates another it is commonly said to be stronger or tighter. The strongest possible valid inequalities are called facets. Given an IP with system of constraints

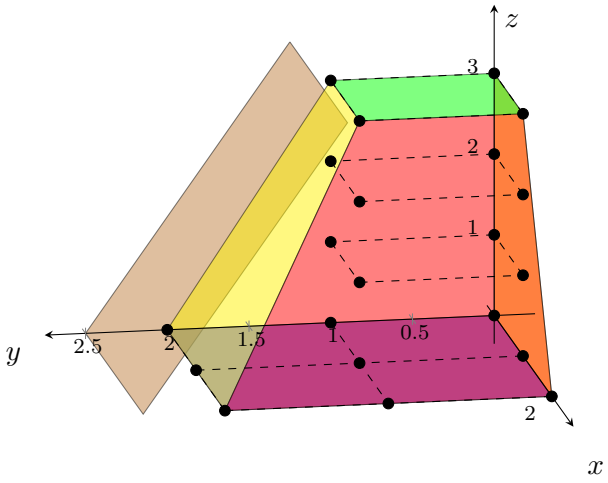


Figure 2: Convex hull of the feasible points in Figure 1 and a valid inequality

defining a bounded polyhedron, we can consider the polytope of the convex hull of all its (integer) feasible points,

$$\mathcal{B} := \text{conv}\{t \in \mathbb{Z}^n : At \leq b\}.$$

The faces of $\mathcal{B}$ of maximal dimension are called facets, and can be formally defined as follows.

Definition 0.3 (*Facet*)**.** An inequality $\pi t \leq \pi_0$ is a facet of a polytope $\mathcal{B}$ of dimension n if:

1. every $t \in \mathcal{B}$ satisfies $\pi t \leq \pi_0$ (validity); and
2. there exist n affinely independent points $t^i \in \mathcal{B}$ satisfying $\pi t^i = \pi_0$, $i = 1, \ldots, n$ (maximality).

△

For the sake of completeness, the following two definitions describe the mathematical notions used in the definition of facet.

Definition 0.4 (*Polytope dimension*)**.** The dimension of $\mathcal{B}$ is n, $\dim(\mathcal{B}) = n$, if the maximum number of affinely independent points in $\mathcal{B}$ is $n + 1$. △

Definition 0.5 (*Affine/linear independence*)**.** Points $t^1, \ldots, t^m$, $t^i \in \mathbb{R}^n \, \forall i = 1, \ldots, m$ are said to be

1. linearly independent, if the unique solution to $\sum_{i=1}^m \lambda_i t^i = \mathbb{0}$ is $\lambda_i = 0 \, \forall i = 1, \ldots, m$;
2. affinely independent, if the unique solution to $\sum_{i=1}^m \lambda_i t^i = \mathbb{0}$, $\sum_{i=1}^m \lambda_i = 0$ is $\lambda_i = 0 \, \forall i = 1, \ldots, m$.

△

Observe that linear independence implies the affine one. The reverse is not true, but the following lemma gives a useful characterization.

Lemma 0.1. *Let $t^1, \ldots, t^m$ be a set of points with $t^i \in \mathbb{R}^n \, \forall i = 1, \ldots, m$. The following statements are equivalent*

(i) $t^1, \ldots, t^m$ *are affinely independent.*

(ii) $t^2 - t^1, \ldots, t^m - t^1$ *are linearly independent.*

Facets are then the best valid inequalities that one can add to a formulation. In an ideal situation, all the constraints in the formulation would be facets. Then, the polytope of the linear relaxation would coincide with the convex hull of all the integer feasible points. Consequently, all its extreme points would have integer components, and the polytope is then said to be integer. When this happens, an optimal solution to the linear relaxation, which coincides with one of these extreme points, would be integer and then optimal for the IP. Figure 2 illustrates such ideal situation, where the feasible points are the same as in Figure 1. The facets of the convex hull of these points are shown in different colors. Observe that the brown plane in the figure does not cut the polytope in this case.

But this scenario is ideal just theoretically. Including all the facets of a polytope into its IP can result completely impractical, since there can be exponentially many— and computational resources are limited. Moving polyhedral theory from principle into practice needs for additional specific strategies and methods. Some of them are *branch and bound*, *cutting planes*, *relaxations* and *decomposition techniques*.

In a branch and bound scheme, subproblems are derived from the original IP by fixing variables to their integer values. These subproblems are hierarchically displayed on a tree, being the original IP at the root. Optimal values of the linear relaxations at the nodes are used as bounds to prune the tree, until a solution is reached. To improve these bounds, it is standard to add valid inequalities at the nodes, originating a *branch and cut* framework. Adding new inequalities within a branch and bound is not trivial, though. In practice, each new inequality is introduced into a pool of constraints, which has to be managed by the solver. The number of inequalities to add can be exponentially many and not carefully designed cutting schemes can easily get out of hand. The whole cutting process— when to cut, how to find the cuts, how many cuts to add, etc.— should follow a set of strategies, which are known as *separation algorithms* (see e.g. Wolsey, 1998; Jünger et al., 2010, for further details).

Valid inequalities can be of general purpose or problem specific. Throughout this thesis, we will speak of valid inequalities meaning constraints tailored to the problem at hand. Conversely, general purpose cutting planes are based on some common mathematical property of the formulations. Some examples are Gomory (see Gomory, 1960), disjunctive (see Balas, 1979), mixed integer rounding (see Nemhauser & Wolsey, 1990) and lift-and-project cuts (see Balas et al., 1993), which are frequently implemented by mathematical programming solvers.

Other general purpose strategies applicable to integer programs include different types of relaxations, such as Lagrangian (Geoffrion, 1974) or others based on reformulation in higher dimensional spaces, e.g. Lovász–Schrijver matrix cone (see Lovász & Schrijver, 1991) and reformulation–linearization techniques (see Sherali & Adams, 1990). Specially popular for mixed-integer programs are decomposition techniques, such as Benders (see Benders, 1962) and cross decomposition (see van Roy, 1986).

0.1.2 Set packing problems

Among all specific configurations in pure Integer Programming, set partitioning probably presents the most widespread applications, see Balas & Padberg (1976). Set partitioning can be stated as: "given a set of objects $S = \{1, \ldots, m\}$ and some subsets of S, $T_1, \ldots, T_n$, with associated costs c_j, $j = 1, \ldots, n$, select a family of subsets in $\{T_j\}_{j=1}^n$ with minimum cost that yields a partition of S". If A is an $m \times n$ matrix of zeros and ones such that $a_{ij} = 1$ iff $i \in T_j$,

$i = 1, \ldots, m$, $j = 1, \ldots, n$, the *set partitioning* problem is

$$(\text{SP}_=) \quad \min\{ct : At = e,\ t \in \{0,1\}^n\},$$

where $e := (1, \ldots, 1)$ is an m-vector and $t_j = 1$ if and only if T_j is selected.

A close relative of set partitioning is *set packing*, which is formulated as

$$(\text{SP}_\leq) \quad \max\{c't : At \leq e,\ t \in \{0,1\}^n\}.$$

The interpretation slightly changes to: "given a set of objects $S = \{1, \ldots, m\}$ and some subsets of S, $T_1, \ldots, T_n$, with associated profits c'_j, $j = 1, \ldots, n$, select a family of disjoint subsets T_j with maximum profit". $(\text{SP}_\leq)$ is an especial case of $(\text{SP}_=)$ with slack variables and costs $c_j = -c'_j\ \forall j = 1, \ldots, n$. Conversely, $(\text{SP}_=)$ can be restated (when feasible) as

$$\max\{(MeA - c)t : At \leq e,\ t \in \{0,1\}^n\},$$

where $M > 0$ is a large enough constant. To arrive at the new statement one just need to observe that $Me(At - e) = 0$ can be added to the packing problem objective and that $At \leq e$ will always attain equality for an optimal solution. The equivalence of $(\text{SP}_=)$ and $(\text{SP}_\leq)$ has been crucial in the literature to gain insight about partitioning problems, see Balas & Padberg (1976).

Another reason for studying set packing is that it serves as common framework for many optimization problems, such as those of finding the maximum clique, Bomze et al. (1999), the maximum edge matching, Balas & Padberg (1976), or some colouring on a graph, Cornaz & Jost (2008); the winner determination problem, Escudero et al. (2009), or facility location problems, Cho et al. (1983b); Cánovas et al. (2002b), among others. Moreover, set packing is strongly related to some especial structures on a graph, called node packings, to the point of allowing a characterization through them. This equivalence offers the opportunity of studying $(\text{SP}_\leq)$ from a different domain.

Node packing

Consider a loopless undirected graph $G = (V, E)$ with no multiple edges. A *node packing* (*stable set*, *independent set*) is a subset of nodes $S \subseteq V$ such that every pair $u, v \in S$ satisfies $(u, v) \notin E$. Suppose that A_G is the incidence matrix of G, i.e., $A_G = (a_{ij})$ is an $m \times n$ matrix— $n := |V|$ and $m := |E|$— such that $a_{ij} = 1$ iff node j is one of the ends of edge i. The problem of finding a node packing of maximum cardinality is

$$(\text{NP}) \quad \max\{et : A_G t \leq e,\ t \in \{0,1\}^n\},$$

which is a particular case of $(\text{SP}_\leq)$. The set of all node packings of a graph G is usually denoted by $\mathcal{P}_G$. A feasible solution to (NP) is the incidence vector of a node packing in G and vice versa, that is, $\mathcal{P}_G$ and the feasible region of (NP) can be identified as follows,

$$\mathcal{P}_G = \{t \in \{0,1\}^n : \{v \in V : t_v = 1\} \text{ is a node packing in } G\}.$$

An important observation is that (NP) admits more than one reformulation. Indeed, $t_u + t_v \leq 1$ for $(u, v) \in E$ can be substituted by stronger $t_u + t_v + t_w \leq 1$ when $(u, w) \in E$ and $(v, w) \in E$, without altering the set of feasible solutions. This fact is the key to reinterpret $(\text{SP}_\leq)$ with matrix $A = (a_{ij})$ as a node packing. Just by considering

$$\begin{aligned} V_A :=& \{1, \ldots, n\}, \text{ and} \\ E_A :=& \{(j, \ell) : j, \ell = 1, \ldots, n, j \neq \ell, a_{ij} = a_{i\ell} = 1 \text{ for some } i = 1, \ldots, m\}, \end{aligned}$$

it immediately follows that ($SP_{\leq}$) is a reformulation of the maximum node packing problem on graph $G_A := (V_A, E_A)$. Consequently, every set packing formulation can be univocally identified with a graph, called the *conflict graph* or *intersection graph*, which has one node per variable and edges between nodes with corresponding variables appearing in the same constraint.

The *packing polytope* or *stable set polytope* is the convex hull of all the feasible points of (NP) for a given graph G,

$$\mathcal{B}_G = \text{conv}\{t \in \{0,1\}^n \ : \ t \in \mathcal{P}_G\}.$$

Note that the definition of the packing polytope does not depend on the matrix that define the system of constraints of (NP). Due to its central role in Integer Programming, the study of the facets of $\mathcal{B}_G$ has received significant attention over the years. Polytope $\mathcal{B}_G$ has two special characteristics. The first one is that $\mathbb{0} \in \mathcal{B}_G$. A second observation is that $\mathcal{B}_G$ has complete dimension, i.e., $\dim(\mathcal{B}_G) = n$. This is a well-known fact, since $\{t^i\}_{i=1}^n$ given by $t^i_i = 1$, $t^i_j = 0 \, \forall j \neq i$ and point $\mathbb{0}$ are $n+1$ affinely independent points in $\mathcal{B}_G$. The next lemma states a characterization of the facets of the set packing polytope with non-zero right-hand side that will be useful in following chapters.

Lemma 0.2. *An inequality $\pi t \leq \pi_0$ with $\pi_0 \neq 0$ is a facet of $\mathcal{B}_G$ if and only if*

1. *(validity) every $t \in \mathcal{B}_G$ satisfies $\pi t \leq \pi_0$ and*
2. *(maximality) there exist n linearly independent points $t^i \in \mathcal{B}_G$ satisfying $\pi t^i = \pi_0$, $i = 1, \ldots, n$.*

Remarkable effort was devoted to identify especial configurations of G that yield facets of the packing polytope, $\mathcal{B}_G$. The two we describe next, cliques and holes, are most certainly the seminal ones. Other kinds of structures that produce facets includes *webs*, Trotter (1975); *wheels*, Cheng & Cunningham (1997) and *hanks*, *fans*, *grilles* and *ranges*, Landete (2001).

In the following, we define cliques and holes as subgraphs of the graph G. Given a subset of nodes, $S \subseteq V$, $G[S]$ denotes the subgraph induced by S, $G[S] := (S, E_S)$ with $E_S := \{(u,v) \in E : u, v \in S\}$.

Definition 0.6 (*Clique*)**.** Given a graph $G = (V, E)$ and a subset of nodes $S \subseteq V$, $G[S]$ is a clique if it is a maximal complete subgraph, i.e., if $(u,v) \in E_S \ \forall u, v \in S, u \neq v$ ($G[S]$ is complete) and $G[S \cup \{w\}]$ is not complete for any $w \in V \setminus S$. In this case, S is said to induce a clique in G. △

Definition 0.7 (*Hole*)**.** Given a graph $G = (V, E)$ and a subset of nodes $S \subseteq V$, $G[S]$ is a hole if it is a cycle with no chords, i.e., S admits an ordering, $S = \{u_1, \ldots, u_{|S|}\}$, such that $E_S = \{(u_i, u_{i+1}) : i = 1, \ldots, |S| - 1\} \cup \{(u_{|S|}, u_1)\}$. In this case, S is said to induce a hole in G. △

Padberg (1973) was the first to notice that cliques and holes define facets. He proved that

$$\sum_{v \in K} t_v \leq 1$$

is a facet of $\mathcal{B}_G$ if and only if $G[K]$ is a clique, whereas

$$\sum_{v \in H} t_v \leq \left\lfloor \frac{|H|}{2} \right\rfloor$$

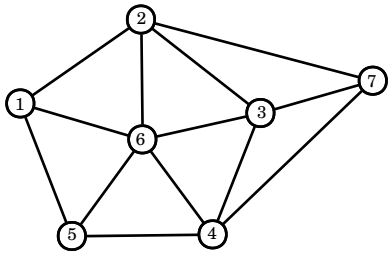


Figure 3: Graph of Example 0.1

is a facet of $\mathcal{B}_{G[H]}$ whenever $G[H]$ is a hole and $|H|$ is odd.

In contrast to cliques, holes only produce facets of the set packing polytope on an induced subgraph of G. In general, if $S \subseteq V$, knowledge of facets of $\mathcal{B}_{G[S]}$ is useful to devise facets of $\mathcal{B}_G$. With this idea, the so-called *lifting* theorems emerged. This name comes from the geometric interpretation of these methods, which consisted of raising up a hyperplane living in $\mathbb{R}^k$ to another hyperplane in a higher dimensional space $\mathbb{R}^n$, $n > k$. The new variables involved are called *lifting variables* and their coefficients in the new hyperplane *lifting coefficients*. The first lifting technique that appeared in the literature is known as *usual lifting* and was simultaneously obtained by several authors, Nemhauser & Trotter (1974); Padberg (1975). It covers the simplest case in which the dimension of the space is augmented by one unit.

Proposition 0.1 (Usual lifting, Nemhauser & Trotter (1974); Padberg (1975))**.** *Let $G = (V, E)$ be a graph and $S \subseteq V$ of cardinality $|V| - 1$ with*

$$\sum_{v \in S} \pi_v t_v \leq \pi_0$$

a facet of $\mathcal{B}_{G[S]}$. Then,

$$\sum_{v \in V} \pi_v t_v \leq \pi_0$$

is a facet of $\mathcal{B}_G$, where the lifting coefficient of $w \in V \setminus S$ is

$$\pi_w := \pi_0 - \max\Big\{ \sum_{v \in S} \pi_v t_v : t \in \mathcal{P}_G, t_w = 1 \Big\}.$$

The coefficient of the new variable is obtained by calculating a node packing in G that maximizes $\sum_{v \in S} \pi_v t_v$ and contains the corresponding new node. Obviously, when $|S| < |V| - 1$ one can always apply Proposition 0.1 several times until obtaining a facet of $\mathcal{B}_G$. The process in which new variables are considered in successive steps and the facet to lift is properly updated every time Proposition 0.1 is applied is usually referred-to as *sequential lifting*. The resulting facet will naturally depend on the order in which the variables are taken, as shown in Example 0.1.

Example 0.1. Consider $G = (V, E)$ the graph depicted in Figure 3 and take $S = \{1, 2, 3, 4, 5\}$. The induced graph $G[S]$ is an odd hole and inequality

$$\sum_{v \in S} \pi_v t_v \leq \left\lfloor \frac{|S|}{2} \right\rfloor = 2$$

is a facet of $\mathcal{B}_{G[S]}$. Proposition 0.1 applied to node 6 yields lifting coefficient 2, since 6 is adjacent to all the nodes of S. If we apply usual lifting to node 7 afterwards, its lifting coefficient is 0, since 7 is not adjacent to 6. In summary,

$$\sum_{v \in S} \pi_v t_v + 2t_6 \leq 2$$

is a facet of $\mathcal{B}_G$. However, if node 7 is lifted in first place, it gets lifting coefficient 1. After that, usual lifting of node 6 will have coefficient 1 as result. Hence, inequality

$$\sum_{v \in S} \pi_v t_v + t_6 + t_7 \leq 2$$

is also a facet of $\mathcal{B}_G$. △

Several lifting techniques appeared soon after usual lifting, see for instance Cho et al. (1983b); Padberg (1977); Wolsey (1976); Barahona & Mahjoub (1994). We still find new ones in more recent literature, e.g. Landete (2001); Galluccio et al. (2008); Xavier & Campêlo (2011) or Chapter 1 of this thesis. However, usual lifting is still the most widespread strategy. Its absolute advantage relies on its practical use; despite most lifting techniques, it does not depend on any especial configuration of the subgraph at hand.

0.2 Location Science

Location Science, also known as Locational Analysis, is a branch of Operations Research concerned with the study of optimal locations for a set of facilities or services in order to satisfy the demand of a set of clients. The field, which comprises a large variety of problems, also interacts with a range of disciplines, such as Economics, Computational Geometry, Geography or Logistics. As a subfield of Location Science, Discrete Location gathers those location problems that can be modeled with integer variables. In this section we briefly introduce those aspects and formulations that will be relevant for the development of this thesis. More thorough reviews on most relevant problems in Location Science can be found in Drezner & Hamacher (2002); ReVelle & Eiselt (2005); ReVelle et al. (2008); Laporte et al. (2015).

There are two decisions involved in almost every discrete location problem. One of them is to select that locations where a service will be installed. The second decision consists of designating the facility that will give service to each client. The set of candidate locations is usually denoted by J and the set of clients by I. Two groups of binary variables are commonly used to model these decisions, namely

$y'_j = 1$ iff some service is installed at candidate location j, $j \in J$ and
$x_{ij} = 1$ iff client i is served by facility at j , $i \in I$, $j \in J$.

Although standard notation for variables y'_j is y_j, here we save y_j for complementary variables,

$y_j = 1$ iff no service is installed at candidate location j, $j \in J$,

which satisfy $y_j = 1 - y'_j$. Variables y'_j and also their complementary y_j are usually called *location variables*, while x_{ij} are usually referred-to as *allocation variables*. These variables and notation will be used for discrete location models of Part I of this thesis. Complementary variables will allow a reformulation of the problem into a set packing in many cases. The common methodology in Part I will be to exploit our knowledge of set packing to improve formulations. Hence, using complementary variables is much more convenient for our exposition, even if not most usual in related literature.

One fundamental problem in Discrete Location that will be studied in this thesis is the *Simple Plant Location Problem* (SPLP), or *Uncapacitated Facility Location Problem*. For its statement, *opening cost* $f_j \geq 0$ of installing a service at j are considered for every candidate location $j \in J$. Once facilities are installed, clients are assigned to those at minimum relative cost — facilities capacities are unlimited for this simplest version. The *assignment cost* of client i and facility j is denoted by $c_{ij} \geq 0$. The first example of the SPLP appeared in the earlies 60s, Stollsteimer (1963), and it has attracted the attention of many researchers ever since. The problem consists of making the two previously stated decisions while incurring minimum *total cost* (opening plus assignment), which formulates as

$$\begin{aligned}
\text{(SPLP)} \quad \min \quad & \sum_{j\in J} f_j y'_j + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in J} x_{ij} = 1 & \forall i \in I & \quad (1)\\
& x_{ij} \leq y'_j & \forall i \in I, \forall j \in J & \quad (2)\\
& x_{ij}, y'_j \in \{0,1\} & \forall i \in I, \forall j \in J. &
\end{aligned}$$

Despite its interpretation, observe that formulation (SPLP) could correspond to a particular combinatorial problem in which some columns from a matrix (c_{ij}) and some of their entries are to be chosen in such a way that there is precisely one entry from each row. Selecting column j or entry (i, j) has a cost f_j and c_{ij}, respectively, and the total cost of the decision made has to be minimized. Applications of the SPLP other than facility location include telecommunication networks design, Gourdin et al. (2002), distributed systems design, Klose & Drexl (2005), and robotics Karch et al. (2002), just to mention some of them.

Many works consider an alternative formulation of the SPLP, which replaces location variables y'_j by their complementary binary variables y_j. This simple idea allows to rewrite (SPLP) into a set packing. On the one hand, the change makes (2) become $x_{ij} + y_j \leq 1\ \forall i \in I, \forall j \in J$. On the other hand, constraints (1) ensure that the product of $\sum_{i=1}^{n}(1 - \sum_{j=1}^{m} x_{ij})$ by an arbitrarily big M is zero. Thus, adding $M\sum_{i=1}^{n}(1 - \sum_{j=1}^{m} x_{ij})$ to the objective in (SPLP) results in the same optimization problem. Grouping terms, the ensuing objective after applying these two transformations would be

$$\min \sum_{j=1}^{m} f_j(1 - y_j) + \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij} - M)x_{ij} + Mn.$$

With this last objective, (1) can be relaxed to less-than-or-equal-to constraints when M is large enough. Thus, formulation (SPLP) is equivalent to the following set packing formulation, a fact that was previously observed by Cho et al. (1983a),

$$\begin{aligned}
\text{(SPLP}_{\leq}\text{)} \max \quad & \sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n}\sum_{j=1}^{m}(M - c_{ij})x_{ij} - Mn - \sum_{j=1}^{m} f_j & & \\
\text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} \leq 1 & \forall i \in I & \quad (3)\\
& x_{ij} + y_j \leq 1 & \forall i \in I, \forall j \in J & \\
& x_{ij}, y_j \in \{0,1\} & \forall i \in I, \forall j \in J. &
\end{aligned}$$

Constraints (3) state that each client must be assigned to one plant at most. However, due to the new objective function, an optimal solution will attain (3) as equalities for M sufficiently

large. Since constraints coefficients in (SPLP$_{\leq}$) are 0/1 and right hand sides are "≤ 1", the formulation is a particular case of (SP$_{\leq}$).

The packing polytope of formulation (SPLP$_{\leq}$) has been extensively studied in the literature. Present knowledge about it includes different families of valid inequalities and facets, including those described in Cho et al. (1983a); Aardal (1998); Cornuéjols et al. (1977); Cornuéjols & Thizy (1982) and Guignard (1980). Some other works also introduced several lifting theorems, such as Cho et al. (1983b). More recently, Landete (2001) studied different graphs that define facets of the set packing polytope in her PhD thesis, and applied them to (SPLP$_{\leq}$) (see also (Cánovas et al., 2000, 2002a,b)). These days, this line is still source of exciting research, see for instance Galli et al. (2015) and Marín & Pelegrín (2015).

Other fundamental models that will be relevant in this thesis are *discrete ordered formulations*. The idea, which was first proposed in Nickel (2001) and further developed by Nickel & Puerto (2005); Marín et al. (2009, 2010), is to assign specific weights to the first, the second... closest allocations client-facility in the solution. This kind of models provided a more flexible alternative than previous ones, which were typically based on minimizing the greatest allocation cost (*centers* problems) or the sum of the allocation costs (*medians* problems).

Discrete ordered formulations incorporate a vector $c_{\leq}$ of ordered costs. This vector contains not repeating assignment costs in $(c_{ij})_{i\in I, j\in J}$ ranked in increasing ordered, i.e.

$$c_{\leq} := (c_{(1)}, \ldots, c_{(\mathcal{K})}); \qquad c_{(1)} < c_{(2)} < \ldots < c_{(\mathcal{K})},$$

where $\mathcal{K}$ is the number of different values in $(c_{ij})_{i\in I, j\in J}$. New auxiliary variables are considered, namely

$s_{ik} = 1$ iff the i-th smallest allocation cost in the solution is at least $c_{(k)}$,

for all $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, \mathcal{K}\}$. Note that i in this case refers to a position in the ranking of assignment costs once the facilities have been installed, which naturally ranges from 1 to n, the number of clients. The new variables have to satisfy

$$\begin{aligned} s_{i,k+1} &\leq s_{i,k} \quad \forall i = 1, \ldots, n, \forall k = 1, \ldots, \mathcal{K} - 1, \\ s_{ik} &\leq s_{i+1,k} \quad \forall i = 1, \ldots, n-1, \forall k = 1, \ldots, \mathcal{K}. \end{aligned}$$

In effect, if the i-th smallest allocation cost is at least $c_{(k+1)}$, it is also greater than previous cost in the ranking, $c_{(k)}$. On the other hand, if the i-th smallest allocation cost is at least $c_{(k)}$, so is the $i+1$-th smallest one. Auxiliary s-variables are linked to allocation variables by the following constraints

$$\sum_{i\in I} \sum_{\substack{j\in J: \\ c_{ij}\geq c_{(k)}}} x_{ij} = \sum_{i=1}^{n} s_{ik} \qquad \forall k = 1, \ldots, \mathcal{K}. \tag{4}$$

For every $k = 1, \ldots, \mathcal{K}$, the left hand side of (4) is equal to the number of clients in the solution whose assignment cost is greater than or equal to $c_{(k)}$. These constraints enforce this same number of clients when ranked by costs. The discrete ordered objective function can be written as follows,

$$\min \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{\mathcal{K}-1} c_{(k)}(s_{ik} - s_{i,k+1}) + c_{(\mathcal{K})} s_{i\mathcal{K}} \right), \tag{5}$$

where $\lambda = (\lambda_1, \ldots, \lambda_n)$ is a vector of positive weights. Note that, if the i-th smallest allocation cost in the solution is $c_{(k')}$, then

$$s_{ik} = 1 \; \forall k \leq k' \qquad \text{and} \qquad s_{ik} = 0 \; \forall k > k'.$$

In this case, $\sum_{k=1}^{\mathcal{K}-1} c_{(k)}(s_{ik} - s_{i,k+1}) = c_{(k')}$ if $k' < \mathcal{K}$, and is zero otherwise. The i-th smallest allocation cost in the solution coincides then with $\sum_{k=1}^{\mathcal{K}-1} c_{(k)}(s_{ik} - s_{i,k+1}) + c_{(\mathcal{K})}s_{i\mathcal{K}}$ and λ_i should be interpreted as the relative weight given to that cost.

Objective (5) generalizes most popular facility location criteria. For instance, with $\lambda = (0, \ldots, 0, 1)$, (5) emulates minimizing the greatest allocation cost; and, when $\lambda = (1, \ldots, 1)$, (5) represents the sum of all allocation costs. Other choices of λ produce multiple models shaped at decision maker's convenience (see Nickel & Puerto, 2005). Particular cases, such as lambda containing many zero entries and with negative values were studied in Marín et al. (2010).

In Chapter 5, we will use the same idea to develop a flexible model for map labeling. The elements to order will be the ambiguity values of the labels, allowing a flexible penalization of ambiguous labels.

Part I

Set packing and facility location

Chapter 1

A new lifting theorem for set packing

This first chapter concerns polyhedral theory in Integer Programming. It presents a new lifting theorem that has application in the family of set packing problems. Its relevance resides in automatic extraction of facets of the set packing polytope. For instance, the new lifting theorem can be applied to obtain new facet defining subgraphs. Identifying one of these subgraphs in the conflict graph of a formulation means finding facets of its polytope. Then, the largest the list of known facet defining subgraphs, the better our chances of uncovering facets of a set packing formulation in the future. Alternatively, the lifting theorem can be directly applied to the conflict graph of some specific formulation, in order to obtain some facets of its polytope. In particular, the theorem will be key for next chapter, where it will be used to obtain facets for a set packing facility location model.

Transforming a facet of some polytope to a facet of another one of higher dimension is a relevant practice in set packing known as lifting. In this chapter, we present a new lifting theorem for the packing polytope, namely

$$\mathcal{B}_G := \text{conv}\{t \in \{0,1\}^n \ : \ t \in \mathcal{P}_G\},$$

where $\mathcal{P}_G$ is the set of all incidence vectors of node packings in G (see Concepts). The theorem can be applied when G has certain structure and we will show that it generalizes an existing lifting result that was introduced by Cho et al., (Cho et al., 1983b, Theorem 2.4). In their paper, these authors give a procedure to transform a facet of $\mathcal{B}_G$ into a facet of $\mathcal{B}_{G^*}$, where G is a subgraph of G^*. The result relies on the following construction. A graph $G = (V, E)$ is given with node set $V = \{1, \ldots, n\}$ having q disjoint subsets $C_1, \ldots, C_q$, $|C_i| \geq 2$, which induce complete subgraphs in G. Then, a graph $G^* = (V^*, E^*)$ is constructed with

$$\begin{aligned} V^* =& V \cup \{n+1,\ n+2,\ \ldots,\ n+q,\ n+q+1\} \\ E^* =& E \cup \{(n+i, j) : j \in C_i, i = 1, \ldots, q\} \cup \{(n+q+1, n+i) : i = 1, \ldots, q\}. \end{aligned}$$

Figure 1.1 shows the construction of the theorem in Cho et al. (1983b). Here, we will follow a similar but more general scheme. First, the configuration we consider allows additional edges between the new nodes $\{n+1, \ldots, n+q\}$ and secondly, C_i are not required to be disjoint. We will demonstrate the applicability of the new result, which served to identify two new families of facet defining graphs for the packing polytope.

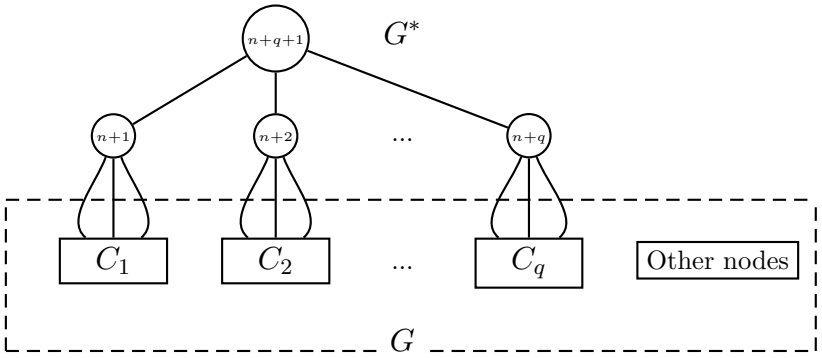


Figure 1.1: Construction of Theorem 2.4 in Cho et al. (1983b)

1.1 The theorem

In addition to basic knowledge about set packing (see Concepts), the following proposition will be useful to prove our lifting theorem.

Proposition 1.1. *Let A be a non-singular matrix of size $r \times r$ and matrices $B_{r\times s}$, $C_{s\times r}$ and $D_{s\times s}$. Then,*

$$det\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = |A| \cdot |-CA^{-1}B + D|.$$

Proof. The well-known fact that

$$det\left(\begin{array}{c|c} A & \mathbb{0} \\ \hline C & D \end{array}\right) = |A| \cdot |D| \qquad \text{and} \qquad det\left(\begin{array}{c|c} A & B \\ \hline \mathbb{0} & D \end{array}\right) = |A| \cdot |D|$$

(see e.g. Silvester, 2000) together with the product decomposition

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)\left(\begin{array}{c|c} I & -A^{-1}B \\ \hline \mathbb{0} & I \end{array}\right) = \left(\begin{array}{c|c} A & \mathbb{0} \\ \hline C & -CA^{-1}B + D \end{array}\right)$$

do the job. □

Let $G = (V, E)$ be a graph with $V = \{1, \ldots, n\}$ and $n \geq 2$. Suppose that $C_1, \ldots, C_q$ are non-empty subsets of V that induce $q \geq 2$ distinct complete subgraphs of G (C_i are not necessarily maximal nor disjoint). Denote by $G^* = (V^*, E^*)$ the graph obtained from G by the construction depicted in Figure 1.2, i.e.,

$$\begin{aligned} V^* =& V \cup \{n+1,\ n+2,\ \ldots,\ n+q,\ n+q+1\} \\ E^* =& E \cup \{(n+i, j) : j \in C_i, i = 1, \ldots, q\} \cup \{(n+q+1, n+i) : i = 1, \ldots, q\} \cup \hat{E}, \end{aligned}$$

where $\hat{E}$ is the set of edges of an arbitrary graph with node set $\{n+1, \ldots, n+q\}$, $\hat{G} := (\{n+1, \ldots, n+q\}, \hat{E})$. The following example illustrates this construction.

Example 1.1. Figure 1.3 shows a graph that fits in the configuration depicted in Figure 1.2. Graph G is in this case an anti-web of 7 nodes and step 3, usually denoted by $W(n = 7, k = 3)$. Graph $\hat{G}$ is a hole of 7 nodes and complete subgraphs $C_1, \ldots, C_7$ are triangles in G. More precisely, the following elements can be identified in the graph depicted on Figure 1.3:

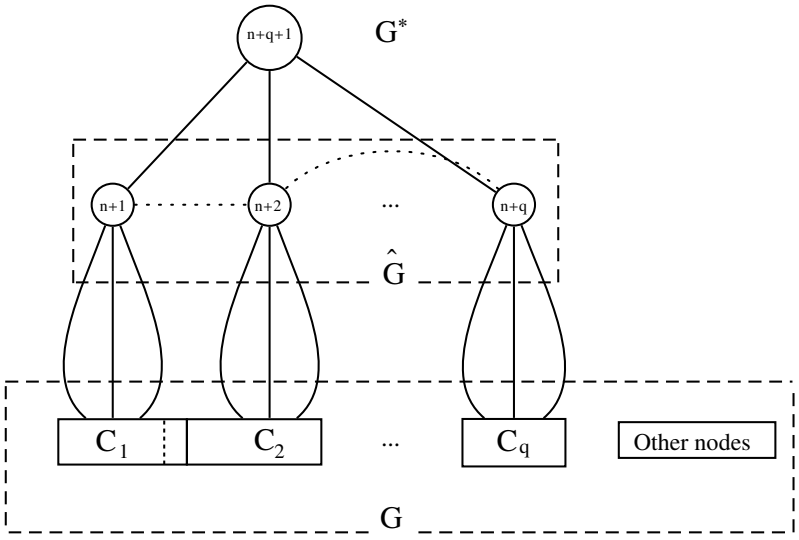


Figure 1.2: Construction of Theorem 1.1

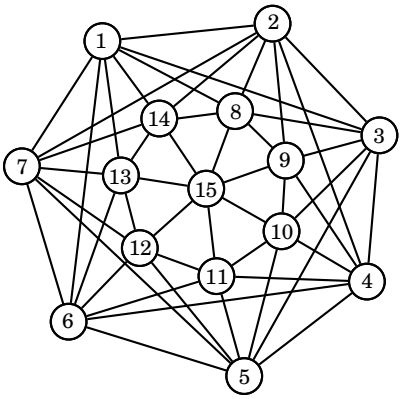


Figure 1.3: Construction of Theorem 1.1 applied to an anti-web

- $n = 7$, $G = (\{1, \ldots, 7\}, E)$, $E = \{(i, (i + 1 + j) \bmod 7) : i = 1, \ldots, 7, j = 0, 1\}$,
- $q = 7$, $C_i = \{i, i + 1, i + 2\}$ for all $i = 1, \ldots, 5$, $C_6 = \{6, 7, 1\}$, $C_7 = \{7, 1, 2\}$,
- $\hat{E} = \{(7 + i, 8 + i), i = 1, \ldots, 6\} \cup \{(8, 14)\}$.

This example is a particular case of the construction depicted in Figure 1.2 with not disjoint complete subgraphs C_i. △

Theorem 1.1. *Suppose that $\pi t \leq 1$ is a facet of $\mathcal{B}_G$ that is not $\sum_{j \in C_i} t_j \leq 1$ for any $i = 1, \ldots, q$. Let $\hat{\alpha}_0$ be the maximum cardinality of a vertex packing in $\hat{G}$ and define the following constants:*

$$\beta_K = \max\{\pi t \ : \ t \in \mathcal{P}_G, t_j = 0 \, \forall j \in \cup_{i \in K} C_i\} \qquad \forall K \subseteq \{1, \ldots, q\},$$
$$\beta_k = \max\{\beta_K \ : \ K \subseteq \{1, \ldots, q\}, \, |K| = k \text{ and } \{t_{n+i} : i \in K\} \in \mathcal{P}_{\hat{G}}\} \qquad \forall k = 1, \ldots, \hat{\alpha}_0.$$

The following inequality:

$$(\hat{\alpha}_0 - 1)\pi t + (1 - \beta_{\hat{\alpha}_0}) \sum_{i=1}^{q+1} t_{n+i} \leq \hat{\alpha}_0 - \beta_{\hat{\alpha}_0} \tag{1.1}$$

is a facet of $\mathcal{B}_{G^*}$ if and only if $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$ and

$$\beta_k \leq 1 - \frac{(k-1)(1-\beta_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1} \qquad \forall k = 2, \ldots, \hat{\alpha}_0 - 1. \tag{1.2}$$

Proof. To begin with, note that $\hat{\alpha}_0 = \beta_{\hat{\alpha}_0}$ iff both are equal to one, since $\hat{\alpha}_0 \geq 1$ and $\beta_{\hat{\alpha}_0} \leq 1$ by definition. If this is the case, all the coefficients of the left-hand side of (1.1) and also its right-hand side are zero. Then, it is clear that $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$ is necessary if (1.1) is a facet.

In order to prove that (1.1) is a facet, we need to prove that it is a valid inequality and that the points satisfying the equality are a face of $\mathcal{B}_{G^*}$ of maximal dimension. We first explore the conditions needed so that (1.1) is valid. We consider $t \in \{0,1\}^n$, $\hat{t} \in \{0,1\}^q$ and $t^* = (t, \hat{t}, t^*_{n+q+1})$ the incidence vector of any node packing in G^*. If $t^*_{n+q+1} = 1$ then $\hat{t}_i = 0$ for all $i = 1, \ldots, q$ and (1.1) becomes:

$$(\hat{\alpha}_0 - 1)\pi t + 1 - \beta_{\hat{\alpha}_0} \leq \hat{\alpha}_0 - \beta_{\hat{\alpha}_0},$$

which is valid because, since $\pi t \leq 1$, $(\hat{\alpha}_0 - 1)\pi t + 1 - \beta_{\hat{\alpha}_0} \leq \hat{\alpha}_0 - 1 + 1 - \beta_{\hat{\alpha}_0} = \hat{\alpha}_0 - \beta_{\hat{\alpha}_0}$. Otherwise suppose that $t^*_{n+q+1} = 0$ and $\hat{t}_i = 1$ for $i \in K$, for a non-empty subset of indices $K \subseteq \{1, \ldots, q\}$ such that $\{n + i : i \in K\} \in \mathcal{P}_{\hat{G}}$. In this case, (1.1) is valid if and only if the maximum value attained at its left-hand side is not greater than its right hand side, i.e., if and only if

$$(\hat{\alpha}_0 - 1)\beta_k + (1 - \beta_{\hat{\alpha}_0})k \leq \hat{\alpha}_0 - \beta_{\hat{\alpha}_0}$$

for all $k = 1, \ldots, \hat{\alpha}_0$. Grouping terms in this inequality we get the equivalent expression

$$\beta_k \leq \frac{\hat{\alpha}_0 - \beta_{\hat{\alpha}_0} - k(1 - \beta_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1},$$

whose right hand side is simplified to

$$\frac{\hat{\alpha}_0 - 1 + 1 - \beta_{\hat{\alpha}_0} - k(1 - \beta_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1} = 1 - \frac{(k-1)(1-\beta_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1}.$$

That is to say, (1.1) is valid if and only if (1.2) holds. Note that condition (1.2) for $k = 1$ becomes $\beta_1 \leq 1$, which always holds since $\pi t \leq 1$, and becomes trivial for $k = \hat{\alpha}_0$.

For the second part of the proof, using Lemma 0.2, we need to find $n + q + 1$ linearly independent feasible points that satisfy (1.1) as equality. We consider the following matrix, whose rows are incidence vectors of vertex packings in G^*

$$A^* = \left(\begin{array}{c|ccc|c} & 0 & \ldots & 0 & 1 \\ A & & \ddots & & \vdots \\ & 0 & \ldots & 0 & 1 \\ \hline & & & & 0 \\ F & & I & & \vdots \\ & & & & 0 \\ \hline t & \hat{t}_1 & \ldots & \hat{t}_q & 0 \end{array}\right).$$

Here A is an $n \times n$ matrix, $|A| \neq 0$, whose rows are n affinely independent incidence vectors of vertex packings in G satisfying $\pi t = 1$. F stands for a $q \times n$ matrix where each row i is an incidence vector of a vertex packing in G with $\sum_{j \in C_i} t_j = 0$ and $\pi t = 1$. We can assure that such vectors exist due to the following reasoning. Suppose that, for some $i \in \{1, \ldots, q\}$, every vertex packing satisfying $\pi x = 1$ necessarily satisfies $\sum_{j \in C_i} t_j = 1$. This would mean that there are $t^1, \ldots, t^n$ affinely independent incidence vectors of vertex packings such that $\pi t^k = 1$ and $\sum_{j \in C_i} t_j^k = 1$, $k = 1, \ldots, n$. Then both $\pi t = 1$ and $\sum_{j \in C_i} t_j = 1$ would be facets in G and they would have to coincide, a possibility that was discarded from our hypothesis. On the other hand, I is the identity matrix of size $q \times q$. Finally, $(t, \hat{t}_1, \ldots, \hat{t}_q)$ is the optimal solution of the optimization problem that defines $\beta_{\hat{\alpha}_0}$. It is easy to check that the rows of A^* are incidence vectors of vertex packings in G^* and satisfy (1.1) as equality. Moreover, applying Proposition 1.1 twice,

$$|A^*| = |A| \cdot \left| \begin{pmatrix} & I & & \mathbb{0} \\ \hat{t}_1 & \ldots & \hat{t}_q & 0 \end{pmatrix} - \begin{pmatrix} F \\ t \end{pmatrix} \cdot A^{-1} \cdot \begin{pmatrix} 0 & & 0 & 1 \\ & \ddots & & \vdots \\ 0 & & 0 & 1 \end{pmatrix}_{n \times (q+1)} \right|$$

$$= |A| \cdot \left| \begin{pmatrix} & I & & \mathbb{0} \\ \hat{t}_1 & \ldots & \hat{t}_q & 0 \end{pmatrix} - \begin{pmatrix} F \\ t \end{pmatrix} \cdot \begin{pmatrix} 0 & & 0 & \pi_1 \\ & \ddots & & \vdots \\ 0 & & 0 & \pi_n \end{pmatrix}_{n \times (q+1)} \right|$$

$$= |A| \cdot \begin{vmatrix} & I & & -\mathbb{1} \\ \hat{t}_1 & \ldots & \hat{t}_q & -\beta_{\hat{\alpha}_0} \end{vmatrix} = |A| \cdot \Big(|I| \cdot \sum_{i=1}^{q} (-\hat{t}_i)(-1) - \beta_{\hat{\alpha}_0} \Big) = |A| \cdot (\hat{\alpha}_0 - \beta_{\hat{\alpha}_0}).$$

This shows that the rows in A^* are linearly independent iff $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$. In conclusion, $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$ and (1.2) are necessary but also sufficient to prove that (1.1) is a facet of $\mathcal{B}_{G^*}$.

□

The interpretation of coefficients β_k is: "among all the packings $\hat{t}$ of cardinality k in $\hat{G}$ and all the possible ways of extending them to a packing in G^* with incidence vector $(t, \hat{t}, 0)$, find the maximum πt". Observe that, if $\hat{\alpha}_0 = 1$ and $\beta_{\hat{\alpha}_0} \neq 1$ then (1.2) holds and (1.1) is the facet defined by a complete subgraph induced by $\{n+1, \ldots, n+q, n+q+1\}$. On the other hand, if $\hat{\alpha}_0 \neq 1$ and $\beta_{\hat{\alpha}_0} = 1$ then (1.2) also holds and (1.1) is the initial facet $\pi t \leq 1$ of $\mathcal{B}_G$, which is in this case also a facet of $\mathcal{B}_{G^*}$.

The following example illustrates the application of Theorem 1.1 to the construction of Example 1.1 and Figure 1.3.

Example 1.2. It is well-known that, when n and k are coprime, $W(n,k)$ defines the facet $\sum_{j=1}^{n} t_j \leq \lfloor n/k \rfloor$ (see Trotter, 1975). Graph $G = W(7,3)$ of Example 1.1 defines then the facet $\frac{1}{2} \sum_{j=1}^{7} t_j \leq 1$. Nodes $\{8, \ldots, 14\}$ induce a hole of length seven, $\hat{G} = (8, \ldots, 14)$, where a maximum node packing has cardinality $\hat{\alpha}_0 = 3$. Since any packing of three nodes in $\hat{G}$ includes all the nodes of G in its neighborhood, it follows that $\beta_{\hat{\alpha}_0} = \beta_3 = 0$. Then, we only have to check that (1.2) holds for $k = 2$. The symmetry of G^* helps to calculate $\beta_2 = 1/2$. Indeed, any packing of cardinality two in $\hat{G}$ only leaves two nodes of G outside its neighborhood. Such nodes are always adjacent, so the maximum value that $\frac{1}{2} \sum_{j=1}^{7} t_j$ can attain is $\frac{1}{2}$. Moreover, condition (1.2) is satisfied:

$$\beta_2 \leq 1 - \frac{(2-1)(1-\beta_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1} = 1 - \frac{1}{3-1} = \frac{1}{2}.$$

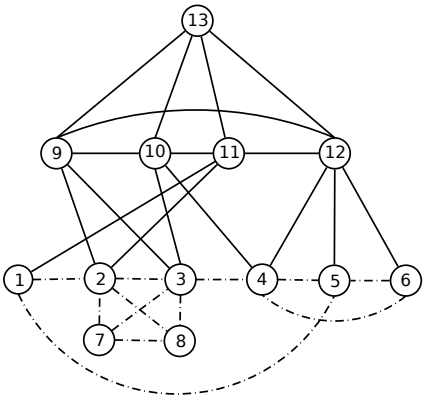


Figure 1.4: A a graph with not disjoint nor maximal complete subgraphs C_i's

We obtain the following facet of $\mathcal{B}_{G^*}$:

$$(3-1)\frac{1}{2}\sum_{j=1}^{7} t_j + (1-0)\sum_{j=8}^{15} t_j \leq 3-0 \quad \Rightarrow \quad \sum_{j=1}^{7} t_j + \sum_{j=8}^{15} t_j \leq 3.$$

△

In a second illustrative example, we show the application of the theorem when C_i's are not maximal nor disjoint.

Example 1.3. Consider the graph shown in Figure 1.4. Dashed-dot lines indicate which edges belong to subgraph G. Hole $(1,2,3,4,5)$ induces a valid inequality, which, after the lifting of t_6, t_7 and t_8 with coefficients 0, turns into the following facet of $\mathcal{B}_G$:

$$\frac{1}{2}\sum_{j=1}^{5} t_j \leq 1.$$

Nodes $9, 10, 11$ and 12 correspond to $n+1, \ldots, n+q$ in Theorem 1.1 (here $n=8$ and $q=4$), and induce a hole, $\hat{G} = (9, 10, 11, 12)$. Since $\hat{G}$ has length four, $\hat{\alpha}_0 = 2$. Finally, one can identify $q = 4$ complete subgraphs in G, $C_1 = \{2,3\}$, $C_2 = \{3,4\}$, $C_3 = \{1,2\}$ and $C_4 = \{4,5,6\}$. When checking the conditions of the theorem, we first observe that $\hat{\alpha}_0 = 2$ and $\beta_{\hat{\alpha}_0} \leq 1$ ensure that $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$. Secondly, due to $\hat{\alpha}_0 = 2$, conditions (1.2) become empty. Consequently, there is a facet for the configuration at hand, whose formula (1.1) involves coefficient $\beta_{\hat{\alpha}_0} = \beta_2$. This coefficient is

$$\begin{aligned}
\beta_2 = \max \quad & \{\max\{\tfrac{1}{2}\textstyle\sum_{j=1}^{5} t_j \ : \ t_j \in \mathcal{P}_G,\, t_j = 0 \,\forall j \in C_i, i \in K\} \\
& : K \subseteq \{1,2,3,4\}, |K| = 2 \text{ and } \{t_{n+i} = 1\}_{i \in K} \in \mathcal{P}_{\hat{G}}\} \\
= \max \quad & \{\max\{\tfrac{1}{2}\textstyle\sum_{j=1}^{5} t_j \ : \ t_j \in \mathcal{P}_G,\, t_j = 0 \,\forall j \in C_1 \cup C_3\}, \\
& \max\{\tfrac{1}{2}\textstyle\sum_{j=1}^{5} t_j \ : \ t_j \in \mathcal{P}_G,\, t_j = 0 \,\forall j \in C_2 \cup C_4\}\} = \quad \frac{1}{2}.
\end{aligned}$$

According to Theorem 1.1 the following is a facet of the packing polytope for graph depicted on Figure 1.4:

$$\frac{1}{2}\sum_{j=1}^{5} t_j + \left(1-\frac{1}{2}\right)\sum_{j=9}^{13} t_j \leq \left(2-\frac{1}{2}\right) \quad \Rightarrow \quad \sum_{j=1}^{5} t_j + \sum_{j=9}^{13} t_j \leq 3.$$

△

It is easy to observe that the facets obtained in examples 1.2 and 1.3 are different from those that one would get by applying sequential lifting. In effect, the right-hand side of the resulting facets is different than that of the original one, something that does not occur in usual lifting.

Remark 1.1. A sufficient condition for the validity of (1.1), which is less restrictive than (1.2), is

$$\gamma_k \leq 1 - \frac{(k-1)(1-\gamma_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1} \qquad \forall k = 1, \ldots, \hat{\alpha}_0, \tag{1.3}$$

where we take

$$\gamma_k \;=\; \max\{\beta_K \,:\, |K| = k\}, \qquad k = 1, \ldots, \hat{\alpha}_0,$$

i.e., we do not require $\{n+i \,:\, i \in K\}$ to be a packing in $\hat{G}$. The proof is exactly the same as in Theorem 1.1, but this time γ_k is a bound on πt that is not tight and thus the condition might not be necessary. Nevertheless, (1.3) can be useful in some cases, since it avoids calculating all possible vertex packings in $\hat{G}$. △

Remark 1.2. In Theorem 2.4, Cho et al. (1983b), $\hat{E} = \emptyset$ and

$$c := \max\{\pi t \,:\, \{j : t_j = 1\} \in \mathcal{P}_G, t_j = 0 \,\forall j \in \cup_{i=1}^{q} C_i\}.$$

They claimed that if $c < 1$ and $\gamma_k \leq 1 - \frac{(k-1)(1-c)}{q-1}$ for all $k = 1, \ldots, q$, then

$$(q-1)\pi t + (1-c)\sum_{i=1}^{q+1} t_{n+i} \leq q - c \tag{1.4}$$

is a facet of $\mathcal{B}_{G^*}$. On the other hand, when $\hat{E} = \emptyset$ we have $\hat{\alpha}_0 = q$ and therefore $\gamma_k = \beta_k$ for all k and $c = \beta_{\hat{\alpha}_0}$. This shows that (1.1) and (1.4) coincide in this particular case. Regarding the hypothesis of (Cho et al., 1983b, Theorem 2.4), $c < 1$ implies $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$; $\gamma_k \leq 1 - \frac{(k-1)(1-c)}{q-1}$ coincides with (1.2); and $\pi_j > 0 \,\forall j$ implies $\pi t \leq 1$ is not $\sum_{j \in C_i} t_j \leq 1$ for any $i = 1, \ldots, q$. This shows that Theorem 1.1 generalizes Theorem 2.4 in Cho et al. (1983b). Moreover, in that work subsets C_i are required to be disjoint and with cardinality at least 2, conditions that vanish in our theorem. △

1.2 New facet-defining graphs

A graph $G = (V, E)$ defines the facet $\pi t \leq \pi_0$ if $\pi t \leq \pi_0$ is a facet of $\mathcal{B}_G$ and $\pi_j > 0 \,\forall j \in V$. We say then that G is facet-defining. In this section we present two new families of facet-defining graphs that are obtained by applying our lifting theorem to odd holes and anti-holes. We name these new families *hyperwheels* and *hyperwebs*, respectively.

1.2.1 Hyperwheels

Let $G = (1, 2, \ldots, n)$ be an odd hole of length $n \geq 7$. Consider the following elements in the construction of Theorem 1.1:

- $q = \frac{n+1}{2}$, $C_i = \{2i-1, 2i\}$ for all $i = 1, \ldots, q-1$, $C_q = \{n\}$,
- $\hat{E} = \{(n+i, n+i+1), i = 1, \ldots, q-1\} \cup \{(n+1, n+q)\}$.

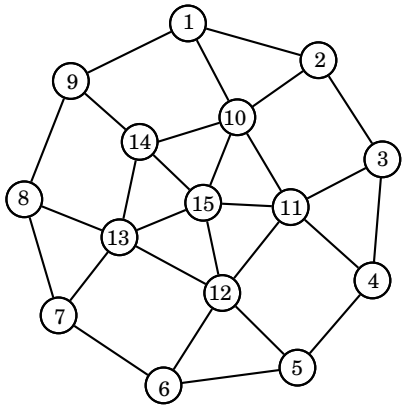


Figure 1.5: Hyperwheel with $n = 9$

The resulting graph from our construction, G^*, consists of a hole of length n and a wheel of $\frac{n+1}{2}+1$ nodes, linked by certain edges. We call this G^* hyperwheel. An hyperwheel with $n = 9$ is shown in Figure 1.5.

The facet that we are going to lift by using Theorem 1.1 is in this case the facet associated to the odd hole G, $\frac{2}{n-1}(t_1+\cdots+t_n) \leq 1$. To begin with, we check the conditions of the theorem. The maximum cardinality of a vertex packing in $\hat{G}$, which is an odd hole, is $\hat{\alpha}_0 = \lfloor \frac{q}{2} \rfloor = \lfloor \frac{n+1}{4} \rfloor$. Observing that a maximum vertex packing in G^* with $\hat{\alpha}_0$ nodes of $\hat{G}$ has $q-\hat{\alpha}_0 = \frac{n+1}{2} - \lfloor \frac{n+1}{4} \rfloor = \lceil \frac{n+1}{4} \rceil$ nodes of G, we get $\beta_{\hat{\alpha}_0} = \frac{2}{n-1} \lceil \frac{n+1}{4} \rceil$. Since $n \geq 7$, $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$. Now observe that, since C_i are disjoint parts of hole $(1, \ldots, n)$, any packing in G^* with $k \geq 2$ nodes of $\hat{G}$ can satisfy $\sum_{j \in C_i} t_j = 1$ for all $i = 1, ..., q$ such that node $n+i$ is not in the packing. Then coefficients $\beta_k = \frac{2}{n-1}(q-k) = \frac{2}{n-1}(\frac{n+1}{2} - k) = \frac{n+1-2k}{n-1}$ are obtained without difficulty. It remains to check condition (1.2). For that purpose we first observe that $\beta_{\hat{\alpha}_0} = \frac{2}{n-1}(\frac{n+1}{2} - \lfloor \frac{n+1}{4} \rfloor)$ because of the general identity $a = \lceil \frac{a}{2} \rceil + \lfloor \frac{a}{2} \rfloor$ for $a \in \mathbb{Z}$. For each $k = 2, \ldots, \hat{\alpha}_0 - 1$, the right-hand side of (1.2) is then:

$$1 - \frac{(k-1)(1-\beta_{\hat{\alpha}_0})}{\hat{\alpha}_0 - 1} = 1 - \frac{(k-1)(1 - \frac{2}{n-1}(\frac{n+1}{2} - \lfloor \frac{n+1}{4} \rfloor))}{\lfloor \frac{n+1}{4} \rfloor - 1}$$

$$= 1 - \frac{\frac{2}{n-1}(k-1)(\lfloor \frac{n+1}{4} \rfloor - 1)}{\lfloor \frac{n+1}{4} \rfloor - 1} = 1 - \frac{2k-2}{n-1} = \frac{n+1-2k}{n-1},$$

which turns to be equal to β_k. Now we can apply Theorem 1.1 to obtain the facet of $\mathcal{B}_{G^*}$:

$$\left(\left\lfloor \frac{n+1}{4} \right\rfloor - 1 \right) \frac{2}{n-1} \sum_{j=1}^{n} t_j + \left(1 - \frac{2}{n-1} \left\lceil \frac{n+1}{4} \right\rceil \right) \sum_{i=1}^{q+1} t_{n+i}$$

$$\leq \left\lfloor \frac{n+1}{4} \right\rfloor - \frac{2}{n-1} \left\lceil \frac{n+1}{4} \right\rceil,$$

which simplifies to

$$\sum_{j=1}^{n+q+1} t_j \leq \frac{n+1}{2}$$

since n is odd.

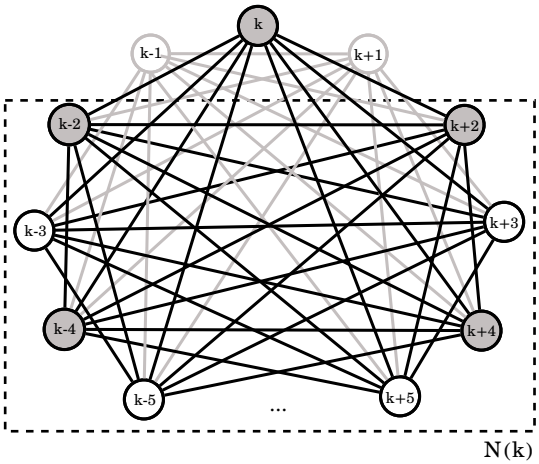


Figure 1.6: Complete subgraph containing a node k of an anti-hole

1.2.2 Hyperwebs

Let $G = (\{1, \ldots, n\}, E)$ be an odd anti-hole of $n \geq 5$ nodes that are numbered according to its formal definition, i.e., G is the complementary graph of the graph induced by the cycle $(1, 2, \ldots, n)$. Before specifying the elements q, C_i, $i = 1, \ldots, q$ and $\hat{G}$ that define this family of graphs, we analyze the structure of G.

First of all, we need to identify q different complete subgraphs in G. Let $k \in \{1, \ldots, n\}$ be a node of G and denote with $N(k)$ the subset of nodes that are adjacent to k. Suppose that k belongs to a subset of nodes $C \subseteq N(k) \cup \{k\}$ that induces a complete subgraph. It is clear that the nodes of C were not consecutive in the cycle $(1, 2, \ldots, n)$. Figure 1.6, where nodes are numbered module n, illustrates this fact. We consider now that the cardinality of C is maximum, i.e., the maximum number of non-consecutive nodes among $N(k)$, $\frac{n-3}{2}$, plus node k. We are interested in calculating how many complete subgraphs of cardinality $\frac{n-3}{2}+1$ contain node k, because they will be our subgraphs $C_1, \ldots, C_q$. For that aim, consider a sequence

$$1, \quad 2, \quad 3, \quad \ldots \quad p,$$

where p is even (this sequence would be the neighbours of k in our case, i.e., $p = n - 3$). How many different ways of taking $p/2$ non-consecutive elements of the sequence are there? This is a small combinatorial problem whose answer is $\frac{p}{2} + 1$. This is easy to see if we start from the sequence of odd numbers $1, 3, 5, \ldots, p-1$ and we generate p sequences more by changing each time one of the elements of the initial sequence and its successors, i.e.:

$$1, \quad 3, \quad 5, \quad \ldots, \quad p-7, \quad p-5, \quad p-3, \quad p,$$

$$1, \quad 3, \quad 5, \quad \ldots, \quad p-7, \quad p-5, \quad p-2, \quad p,$$

$$1, \quad 3, \quad 5, \quad \ldots, \quad p-7, \quad p-4, \quad p-2, \quad p,$$

$$\ldots$$

$$1, \quad 3, \quad 6, \quad \ldots, \quad p-6, \quad p-4, \quad p-2, \quad p,$$

$$1, \quad 4, \quad 6, \quad \ldots, \quad p-6, \quad p-4, \quad p-2, \quad p,$$

$$2, \quad 4, \quad 6, \quad \ldots, \quad p-6, \quad p-4, \quad p-2, \quad p.$$

In our case, k belongs to exactly $\frac{n-3}{2}+1 = \frac{n-1}{2}$ complete subgraphs of size $\frac{n-1}{2}$. Given the symmetry of graph G, this argument is valid independently of the node k taken. We get that, in total, there are n different complete subgraphs of size $\frac{n-1}{2}$ in G and that every node of G belongs exactly to $\frac{n-1}{2}$ of these subgraphs.

We can identify now the elements in the construction of Theorem 1.1 for our particular graph. We will define $m = \frac{n-1}{2}$ and will use a matrix $C_{n\times m} = (c_{ij})$ to arrange the n different complete subgraphs of size $\frac{n-1}{2}$. Each row $c_{i\cdot}$ will represent the subset of nodes C_i, $i = 1, \ldots, q(= n)$. This matrix is defined following the same idea of the previous paragraph:

- For $i = 1, 2, 3$:

  $$c_{i1} = i \text{ and } c_{ij} = c_{i,j-1} + 2 \quad \forall j = 2, \ldots, m.$$

- The following rows $4, 5, 6, \ldots, n-1, n$ are defined in pairs $(2r, 2r+1)$ for $r = 2, \ldots, m$:

  $$\begin{aligned} &c_{2r,1} = 1, \\ &c_{2r,j} = c_{2r,j-1} + 2 \quad \forall j = 2, \ldots, m-r+1, m-r+3, \ldots, m, \\ &c_{2r,m-r+2} = n - 2r + 3 \text{ and} \\ &c_{2r+1,j} = c_{2r,j} + 1 \quad \forall j = 1, \ldots, m. \end{aligned}$$

As indicated in the construction of Theorem 1.1, to extend G to a bigger graph G^*, we add $q(= n)$ nodes to G and consider the following edges between them: $\hat{E} = \{(n+i, n+i+1), i = 1, \ldots, q-1\} \cup \{(n+1, n+q)\}$. As a result, $\hat{G}$ is an odd hole of q nodes. After adding node $n+q+1$ and edges between it and nodes $n+1, \ldots, n+q$, we obtain a graph G^*, which we call hyperweb. Hyperwebs with $n = 5$ and 7, with corresponding matrices

$$C_{5\times 2} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ 1 & 4 \\ 2 & 5 \end{pmatrix} \qquad \text{and} \qquad C_{7\times 3} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 3 & 5 & 7 \\ 1 & 3 & 6 \\ 2 & 4 & 7 \\ 1 & 4 & 6 \\ 2 & 5 & 7 \end{pmatrix},$$

are shown in Figure 1.7, where $\hat{E}$ is depicted with dash-dot lines.

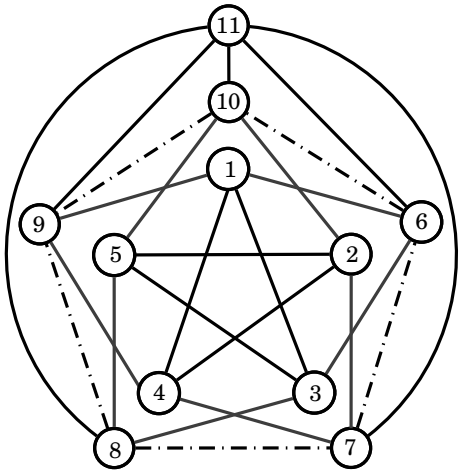
In the following we check that the presented scenario is under the assumptions of Theorem 1.1. It is well-known that G defines the facet $\frac{1}{2}\sum_{j=1}^{n} t_j \leq 1$. Since $\hat{G}$ is an odd hole, $\hat{\alpha}_0 = m$. To calculate β_k, $k = 2, \ldots, \hat{\alpha}_0$, we will need the following lemma.

Lemma 1.1. *For every pair of nodes k, $(k+1) \bmod n$ of G (where we identify 0 with n) that were consecutive in the cycle $(1, 2, \ldots, n)$ and for every two complete subgraphs C_r, C_s, we have that $k \in C_r \cup C_s$ or $((k+1) \bmod n) \in C_r \cup C_s$.*

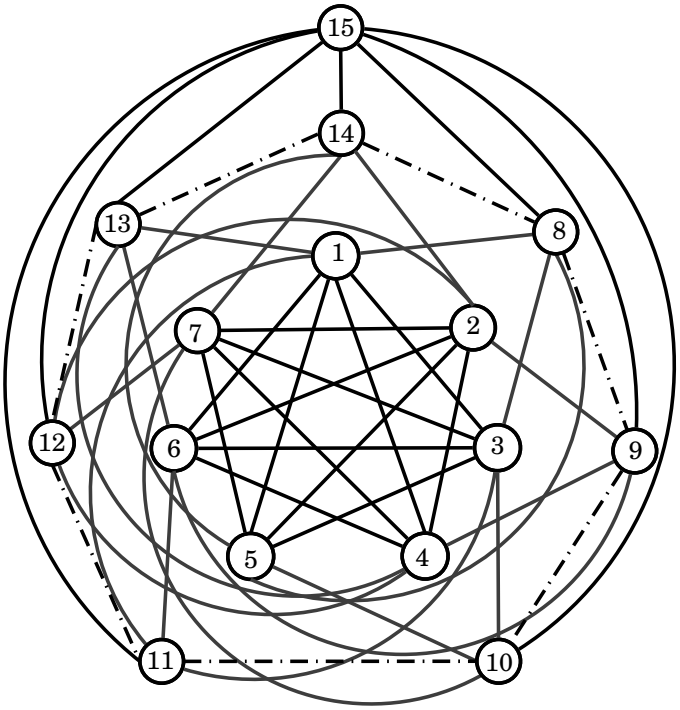
Proof. Suppose that neither k nor $(k+1) \bmod n$ belong to $C_r \cup C_s$. If we consider the sequence

$$1, \quad 2, \quad 3, \quad \ldots, \quad p-1, \quad p,$$

where now p is odd (these would be the $n-2$ remaining nodes), there is only one way of taking $\frac{p+1}{2}$ non-consecutive elements from it (a complete subgraph of $\frac{n-1}{2}$ nodes in our case), which is a contradiction. □



(a) Hyperweb with $n = 5$



(b) Hyperweb with $n = 7$

Figure 1.7: Constructions for hyperwebs

Lemma 1.1 shows that every packing of $k \geq 2$ nodes of $\hat{G}$ can be completed with at most one node from G, yielding $\beta_k \leq \frac{1}{2}$ for all $k = 2, \ldots, \hat{\alpha}_0$. When $k = \hat{\alpha}_0$, it is easy to see that $C_4, C_6, \ldots, C_{n-1}, C_1$ do not contain node 2 and as a consequence $\beta_{\hat{\alpha}_0} = \frac{1}{2}$. Since $\beta_k \geq \beta_{k+1}$ in general, $\beta_k = \frac{1}{2}$ for all $k = 2, \ldots, \hat{\alpha}_0 - 1$. We conclude that for all $k = 2, \ldots, \hat{\alpha}_0 - 1$, $\beta_k = \frac{1}{2}$ verifies condition (1.2). Theorem 1.1 assures then that

$$(m-1)\frac{1}{2}\sum_{j=1}^{n} t_j + \left(1 - \frac{1}{2}\right)\sum_{i=1}^{q+1} t_{n+i} \leq m - \frac{1}{2}$$

is a facet of $\mathcal{B}_{G^*}$, i.e.,

$$(m-1)\sum_{j=1}^{n} t_j + \sum_{i=1}^{q+1} t_{n+i} \leq 2m - 1$$

is a facet of the packing polytope for a hyperweb of $2n + 1$ nodes.

Chapter 2

The Simple Plant Location Poblem with incompatibilities

The Simple Plant Location Problem (SPLP) is one of the seminal problems in Discrete Location that can be formulated as a set packing. In this chapter, we study a modification of the SPLP that has not been considered in previous literature. The novel aspect consists of taking into account constraints due to incompatibilities between users or clients. Users incompatibilities may well be due to technical requirements or competing interests and prevent some pairs of clients from being served by the same facility. The corresponding new mathematical programming constraints are set packing ones. The proposed model for the resulting variant of the SPLP will then be a set packing.

We find some examples of allocating conflicts between users outside the field of location analysis in communication networks. Service management of mobile networks includes channel assignment as a fundamental task. A channel allocation strategy defines the assignment of available channels to calls originated in cells. Due to cochannel interference, the same channel cannot be assigned to certain pairs of cells simultaneously, see Stojmenović (2002). If channels are identified with facilities and cells with clients, the SPLP with incompatibilities simulates these cochannel constraints. Other communication networks consist of a set of terminals (users) and concentrators (switches or multiplexers). A primary problem in its design is to decide how many concentrators are needed and how the terminals should be assigned to the concentrators. The relation of this problem with facility location was observed by Gourdin et al. (2002), who proposed to use the SPLP formulation in one of the phases of an iterative network design process. Considering incompatibilities between the users can be useful to represent competing interests within users in this context.

We explore the implications of adding incompatibility constraints to the set packing formulation of the SPLP. Interestingly, this modification generalizes fault-tolerant location problems and is a close relative of multiproduct facility location. We study the polytope of the new problem, and present different families of facets for the new model. These are based on cliques and odd holes of the corresponding conflict graph. In order to obtain clique facets, we will just need to identify clique subgraphs of the conflict graph, while hole facets demand a little bit more work. Each family of facets is accompanied by a separation algorithm. Our computational experience supports the utility of the devised structures. Experiments on the separation algorithms are reported, together with a comparative analysis with respect to standard clique cuts incorporated by a commercial solver.

2.1 Introducing incompatibilities

The Simple Plant Location Problem with incompatibilities (SPLPI) is a generalization of the SPLP that takes into consideration possible incompatibilities between pairs of clients. Two clients are said to be incompatible if they can not be served by the same facility. Let I be the set of clients and J denotes the set of candidate facilities. To model incompatibility, we define a graph $G_I = (I, E_I)$ that has one node per client and edges $e = (i, k) \in E_I$ if clients i and k are incompatible. The mathematical programming formulation of the SPLPI below uses standard allocation variables x_{ij} and complementary location variables y_j. The latter allow a set packing formulation of the SPLP, as shown in Concepts. Standard notation f_j and c_{ij} is used respectively for opening and assignment costs.

$$
\begin{aligned}
\text{(SPLPI)} \quad \max \quad & \sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n}\sum_{j=1}^{m}(M - c_{ij})x_{ij} - Mn - \sum_{j=1}^{m} f_j & & \\
\text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} \leq 1 & \forall i \in I & \quad (2.1)\\
& x_{ij} + y_j \leq 1 & \forall i \in I, \forall j \in J & \quad (2.2)\\
& x_{ij} + x_{kj} \leq 1 & \forall (i,k) \in E_I, \forall j \in J & \quad (2.3)\\
& x_{ij} \in \{0,1\} & \forall i \in I, \forall j \in J & \\
& y_j \in \{0,1\} & \forall j \in J. &
\end{aligned}
$$

Formulation (SPLPI) is the set packing formulation ($\text{SPLP}_{\leq}$) stated in Concepts plus incompatibility constraints (2.3). This is also a set packing problem, which is closely related to two well-known variants of the SPLP, as shown in next section.

2.1.1 Related problems

The interest of most classical models in Discrete Location (p-median, p-center, covering, hub, among others) has frequently promoted the emergence of a huge range of different variants (see ReVelle & Eiselt, 2005). The SPLP is not an exception. As many problems in location, the SPLP has its capacitated version, which arises when capacities of the facilities are limited and clients demands are quantified. The demand of each client has to be satisfied while capacities are not exceeded, see for instance Revelle & Laporte (1996). A different variant, the Fault-Tolerant Facility Location problem (FTFL), arises when each client has to be assigned to several facilities (see e.g. Swamy & Shmoys, 2008). Its name comes from the interpretation of the multiple facilities as backups. In other cases, modifications concern the objective function. For instance, some authors consider a maximum return-on-investment objective (see Brimberg & ReVelle, 2000), which is more frequent in industrial decision making processes. Other variants include multi-objective (see Current et al., 1990) or multi-product models (see Warszawski, 1973). Interested readers are invited to consult Revelle & Laporte (1996); Drezner & Hamacher (2002); Laporte et al. (2015) for a more extensive review.

This section illustrates the relation between (SPLPI) and some of the variants of the SPLP previously studied in the literature. Concretely, (SPLPI) leaves the FTFL as particular case, and partially describes the feasible region of the multiproduct SPLP.

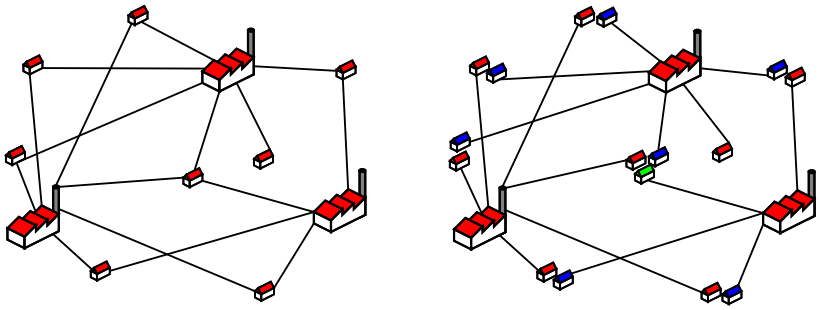


Figure 2.1: Illustration of formulations (FTFL) and (FTFLI)

The Fault-Tolerant Facility Location problem

The FTFL is identical to the SPLP except for the fact that each client $i \in I$ must be allocated to $r_i \geq 1$ facilities. A mathematical programming formulation of the problem is (see Swamy & Shmoys, 2008)

$$\begin{aligned}
\text{(FTFL)} \quad \min \quad & \sum_{j=1}^{m} f_j y'_j + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} & \\
\text{s.t.} \quad & \textstyle\sum_{j=1}^{m} x_{ij} \geq r_i & \forall i \in I \\
& x_{ij} \leq y'_j & \forall i \in I, \forall j \in J \\
& x_{ij} \in \{0,1\} & \forall i \in I, \forall j \in J \\
& y'_j \in \{0,1\} & \forall j \in J.
\end{aligned}$$

Alternatively to (FTFL), the new condition can be stated by replicating client i $r_i - 1$ times, obtaining r_i distinct clients with simple assignment requirements. With this transformation, the allocations in each group of r_i clients must be to r_i different facilities. But this is to say that the r_i copies of client i have to be served by different facilities, i.e., they are pairwise incompatible. With this idea and the following new x-variables

$x^k_{ij} = 1$ iff copy k of client i is served by facility at j , $i \in I$, $j \in J$, $k = 1, \ldots, r_i$,

an alternative formulation of the FTFL is

$$\begin{aligned}
\text{(FTFLI)}\, \min \quad & \sum_{j=1}^{m} f_j y'_j + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} \Big(\sum_{k=1}^{r_i} x^k_{ij}\Big) & \\
\text{s.t.} \quad & \sum_{j=1}^{m} x^k_{ij} = 1 & \forall i \in I, \forall k = 1, \ldots, r_i \\
& \sum_{k=1}^{r_i} x^k_{ij} \leq 1 & \forall i \in I, \forall j \in J \qquad (2.4) \\
& x^k_{ij} \leq y'_j & \forall i \in I, \forall j \in J, \forall k = 1, \ldots, r_i \\
& x^k_{ij} \in \{0,1\} & \forall i \in I, \forall j \in J, \forall k = 1, \ldots, r_i \\
& y'_j \in \{0,1\} & \forall j \in J.
\end{aligned}$$

Constraints (2.4) ensure that copies of one client are not assigned to the same facility. The rest are typical SPLP constraints. Note that (FTFLI) could become a set packing formulation

by using the complementary variables $y_j = 1 - y'_j$. The resulting formulation will clearly correspond to a particular case of (SPLPI), where constraints (2.4) account for an improved version of (2.3). In this particular case, incompatibility constraints are not added by pairs of nodes but by subgroups of r_i incompatible clients, which form a clique in G_I, $i = 1, \ldots, n$.

Figure 2.1 gives a graphical illustration of the differences between formulations (FTFL) and (FTFLI) on its left and right hand sides, respectively. The figure depicts three facilities, eight clients and links representing client-facility allocations. On the left, multiple links to the same client are drawn while, on the right, links are simple and clients are replicated zero, one or two times.

The multiproduct Simple Plant Location Problem

In the multiproduct extension of the SPLP, which we will call MPLP, a firm elaborates different types of products from a set $S = \{1, \ldots, w\}$, which are distributed among its clients. Each client i demands a product type $s_i \in S$ and each facility will be specialized in the production of only one product type. The firm has to decide where to install the facilities and which product type they will produce. Installing the machinery to produce product s at any facility has cost g_s, for each $s \in S$. Depending on the type of product they demand, clients will be assigned to facilities. Using standard allocation variables and the following location variables

$y'_{js} = 1$ if facility j is opened and makes products of type s, $j \in J$, $s \in S$,

a formulation of the multiproduct extension is

$$
\begin{aligned}
\text{(MPLP)} \quad \min \quad & \sum_{j=1}^{m}\sum_{s=1}^{w}(f_j + g_s)y'_{js} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} = 1 & \forall i \in I & \\
& x_{ij} \leq y'_{js_i} & \forall i \in I, \forall j \in J & (2.5)\\
& x_{ij} \leq 1 - y'_{js} & \forall i \in I, \forall j \in J, \forall s \in S, s \neq s_i & (2.6)\\
& \sum_{s=1}^{w} y_{js} \leq 1 & \forall j \in J & (2.7)\\
& x_{ij} \in \{0,1\} & \forall i \in I, \forall j \in J & \\
& y'_{js} \in \{0,1\} & \forall i \in I, \forall s \in S. &
\end{aligned}
$$

Constraints (2.5) ensure that a client can be served by a facility only if the latter is opened and makes the product that the client demands. On the other hand, constraints (2.6) guarantee that facilities only serve clients that demand the type of product they produce. Finally, (2.7) state that each facility is specialized in one product at most.

Since clients that demand different types of product cannot be served by the same facility, an alternative approach to the problem is to consider them as incompatible clients. If so, the second subscript of variables y can be removed, and standard location variables can be considered. Finally, introducing a new variable to keep track of specializing costs,

v_j =cost of specializing facility j if j is open and 0 otherwise,

the following is a valid formulation of the MPLP:

$$
\begin{aligned}
\text{(MPLPI)}\min \quad & \sum_{j=1}^{m}(f_j y'_j + v_j) + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j=1}^{m} x_{ij} = 1 & \forall i \in I & \\
& x_{ij} \leq y'_j & \forall i \in I, \forall j \in J & \\
& x_{ij} + x_{kj} \leq 1 & \forall i,k \in I, \forall j \in J, s_i \neq s_k & (2.8)\\
& g_{s_i} x_{ij} \leq v_j & \forall i \in I, \forall j \in J & (2.9)\\
& x_{ij} \in \{0,1\} & \forall i \in I, \forall j \in J & \\
& y'_j \in \{0,1\} & \forall j \in J. &
\end{aligned}
$$

Incompatibilities between clients are represented by set packing constraints (2.8), while (2.9) ensure that variables v take the desired values. Considering complementary variables $y_j = 1 - y'_j$ allows to transform (MPLPI) into a formulation close to set packing. Note that, except for variables v and associated constraints (2.9), (MPLPI) after the transformation would be a particular case of (SPLPI). The graph of clients incompatibilities would depend on the intersection of the different sets of products they demand.

Some authors consider a different version of the MPLP, in which each client can demand products of more than one type, while every open facility still produces products of only one type, see Warszawski (1973). Consequently, clients can be served by more than one facility here. An approach that uses clients incompatibilities can be used to model the problem, in a very similar way as described for the simpler version of the MPLP.

2.2 The SPLPI polytope

The SPLPI polytope is the convex hull of all the feasible solutions of formulation (SPLPI), that is

$$\mathcal{B}_{splpi} := \text{conv}\{(x,y) \in \{0,1\}^{n\times m} \times \{0,1\}^m : \ (2.1) - (2.3)\}.$$

It is clear that $\mathcal{B}_{splpi}$ is included in the polytope of the set packing formulation of the SPLP, (SPLP$_{\leq}$). Nevertheless, being (SPLPI) a set packing, $\mathcal{B}_{splpi}$ is still of complete dimension.

To study the facial arrangement of $\mathcal{B}_{splpi}$, we will exploit the set packing structure of the problem constraints. Such feature allows identification between (SPLPI) and its conflict graph, which we will call G_{splpi}. As described in Concepts for general set packings, a feasible solution of (SPLPI) corresponds with a node packing of G_{splpi}. In fact, according to Concepts, $\mathcal{B}_{G_{splpi}}$ would be the notation for the packing polytope of (SPLPI), but here we have simplified it to $\mathcal{B}_{splpi}$.

Before investigating the facial structure of $\mathcal{B}_{splpi}$, we need to understand conflict graph G_{splpi}. This graph inherits some characteristic layout from formulation (SPLP$_{\leq}$). Because of constraints (2.1), x-nodes are grouped in n groups of m nodes, each group inducing a complete subgraph. On the other hand, due to constraints (2.2), every y-node is adjacent to only one node of every group. Constraints (2.3) of (SPLPI) disrupt this balanced structure in a way that directly depends on the incompatibility graph G_I. The following example illustrates the relation between G_I and G_{splpi}.

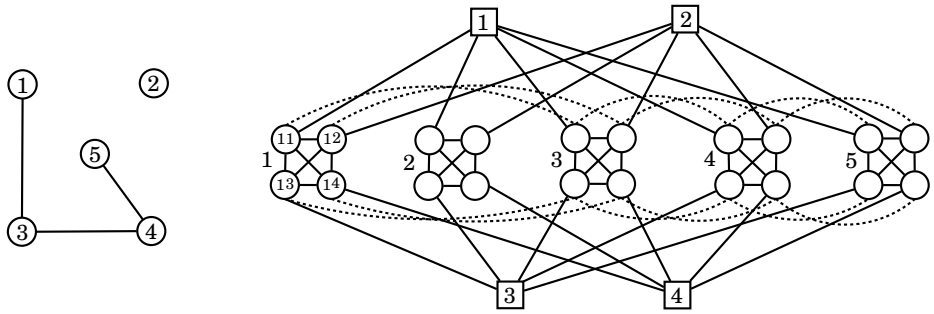


Figure 2.2: Example of graph G of incompatibilities and conflict graph G_{splpi} when $n = 5$ and $m = 4$

Example 2.1. Figure 2.2 shows G_I and G_{splpi} when $m = 4$ and $n = 5$. On the left-hand side, G_I depicts incompatibilities between clients 1 and 3, 3 and 4 and 4 and 5. On the right, square nodes represent variables y and are numbered from 1 to 4, while circular nodes stand for variables x, and are arranged by clients in 5 groups. Nodes of the first group are tagged with their subscripts in (SPLPI). It can be noticed that nodes of a group correspond with x-variables with same index i and different indices j from 1 to m. Continuous edges in G_{splpi} correspond with constraints of the original SPLP, while dashed edges correspond with incompatibility constraints (2.3). It can be easily observed that each edge (i, k) in G_I originates an edge between nodes (x_{ij}, x_{kj}) for each $j \in J$. △

2.3 Facets and separation

In this section, we investigate the facial structure of $\mathcal{B}_{splpi}$, identifying inequalities that were also facets of (SPLP$_{\leq}$) but focusing on those that are facets only of the restricted polytope $\mathcal{B}_{splpi}$. This section also presents separation algorithms to manage the discovered facets inside a branch and cut procedure. Non-expert readers are invited to revisit Concepts for an introduction to Polyhedral Theory, Integer Programming and solving techniques.

Since nodes of x-variables in G_{splpi} can always be grouped in n complete subgraphs, we will name those subgraphs $X_1, \ldots, X_n$. Note that each of them gives the information about a client, therefore it will be natural to identify each client i with X_i throughout the section. We will also call x-nodes and y-nodes those corresponding with variables x_{ij} and y_j, respectively. As it commonly occurs for set packing problems, facets of $\mathcal{B}_{splpi}$ will be identified by means of certain subgraphs of G_{splpi}. Then we will just speak of a facet of a subgraph to refer to a facet of the associated packing polytope.

The core of this section describes the analysis made to obtain facets from two types of subgraphs in G_{splpi}, namely cliques and odd holes. These subgraphs emerge in G_{splpi} as a consequence of similar structures defined by the incompatibilities between clients. As illustrated by Figure 2.2, our conflict graph consists of a fixed layout given by (2.1) and (2.2) plus a set of some additional edges due to (2.3). These edges are incident to nodes of different groups X_i and same subscript j, but we do not know *a priori* which are those groups. This fact will give rise to natural difficulties when exploring the layout of G_{splpi} for a general graph G_I of clients incompatibilities. Nevertheless, some interesting research can certainly be done without imposing additional conditions to G_I. Indeed, we will enumerate all the clique facets of $\mathcal{B}_{splpi}$ in the most general case. In the case of hole inequalities, the theoretical development starts

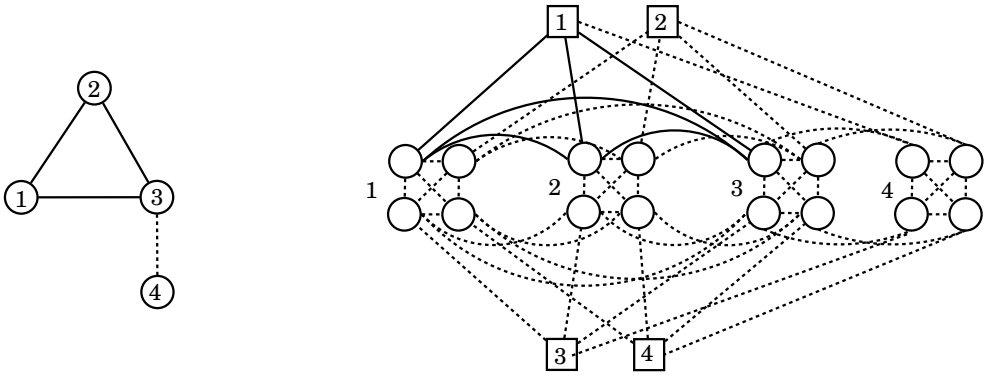


Figure 2.3: Three incompatible clients in G_I and corresponding cliques in G_{splpi}

from a general G_I and will arrive at further conclusions when some particular properties are assumed.

2.3.1 Clique facets

Maximal complete subgraphs induce facets of the conflict graph that are called *clique facets*. It is known that $\sum_{j=1}^{m} x_{ij} \leq 1$ for all $i \in I$ and $x_{ij} + y_j \leq 1$ for all $i \in I$ and $j \in J$ are the only clique facets of the SPLP packing polytope (see Cornuéjols & Thizy, 1982). Starting from this result, we explore whether these inequalities remain facets of (SPLPI) and whether the additional constraints (2.3) introduce new clique facets. Unless stated otherwise, by clique we will refer to a maximal complete subgraph or to the set of nodes which it consists of.

It is easy to observe that a clique in G_I produces m cliques in G_{splpi}. For each plant j and clique $C \subseteq I$ in G_I, the set of nodes corresponding with y_j and x_{ij}, $i \in C$, is a clique. The following example illustrates this connection.

Example 2.2. Figure 2.3 depicts an instance of the SPLPI with three pairwise incompatible clients and four plants. Incompatibility graph G_I and conflict graph G_{splpi} are shown on the left and right hand sides of the figure, respectively. Continuous edges in G_I highlight a clique formed by clients 1, 2 and 3. It induces four cliques in G_{splpi}, each one consisting of nodes y_j, x_{1j}, x_{2j} and x_{3j}, for $j = 1, \ldots, 4$. On the right hand side of the figure, one of these cliques, which corresponds with $j = 1$, is emphasized in continuous trace. △

The following proposition shows that these are the only new cliques that appear in G_{splpi} as a consequence of the inclusion of the incompatibility constraints (2.3).

Proposition 2.1. *The only clique facets in* SPLPI *are:*

(i) $\displaystyle\sum_{j=1}^{m} x_{ij} \leq 1$ *for all* $i \in I$.

(ii) $\displaystyle y_j + \sum_{i \in C} x_{ij} \leq 1$ *for all* $j \in J$ *and for all clique* $C \subseteq I$ *in* G_I.

Proof. Let $\sum_{j=1}^{m} \pi_j y_j + \sum_{j=1}^{m} \sum_{i=1}^{n} s_{ij} x_{ij} \leq 1$ be a clique facet of $\mathcal{B}_{splpi}$, where $\pi = (\pi_j)$ and $S = (s_{ij})$ are both binary. It is clear that either $\pi = \mathbb{0}$ or $\pi_j = 1$ for only one index j, since for every subset of plants $J' \subseteq J$ there is always a feasible solution with $y_j = 1$ for all $j \in J'$. In the latter case, $s_{ih} = 0$ for all $h \neq j$ and for all i due to the problem formulation. Then, let

$C \subseteq I$ be a subset such that $s_{ij} = 1$ iff $i \in C$. Since the initial facet is a clique facet, C induces a clique in G_I and we have an inequality of type (ii). Now, it is also clear from the maximality that if $\pi = \mathbb{0}$, the initial facet is of type (i).

To complete the proof, it would remain to see that (i) and (ii) always define facets. On the one hand, variables in (i) induce the complete subgraph X_i, which is also maximal. On the other hand, variables in (ii) induce a clique in G_{splpi} since C is a clique in G_I and node y_j is adjacent to all nodes $\{x_{ij} : i \in C\}$ due to (2.2). Thus, both (i) and (ii) are clique facets. □

Note that, when $|C| = 1$, (ii) turns into (2.2), i.e., the SPLP clique constraint $x_{ij} + y_j \leq 1$, where $C = \{i\}$. Our interest is then on facets of type (ii) when $|C| > 1$, since they are not explicitly stated in the formulation of the problem. The number of these facets depends on the number of cliques in G_I, which can be large.

2.3.2 Separation of clique facets

Separating clique facets in set packing formulations substantiates on clique identification in the conflict graph, G_{splpi} in our case. Indeed, given a fractional solution, searching for the most violated clique facet is equivalent to identifying the maximum weight clique in a weighted graph. Consequently, clique separation, like most versions of the clique problem, is computationally hard, see Karp (1972). It is worth mentioning, though, that there are some kind of graphs that admit an efficient implementation of the maximum clique search. For instance, Grötschel et al. (1984) showed that, when a graph is perfect, the weighted clique problem is solvable in polynomial time. Perfect graphs are those for which its clique and coloring numbers coincide. That is, the number of nodes of the maximum clique coincides with the minimum number of colors needed to color the nodes in such a way that adjacent nodes have different colors.

Clique inequalities appear in a huge variety of combinatorial optimization problems, such as multi-index assignment (Balas & Saltzman, 1989; Magos & Mourtos, 2009), winner determination (Escudero et al., 2008) or vehicle routing (Spoorendonk, 2008). In most of the cases, there are too many of these inequalities to be treated all at once, and they need to be separated. Although considerable effort has been invested in developing separation procedures for clique inequalities (see e.g. Escudero et al., 2008; Magos & Mourtos, 2009), most of these attempts are particularized to the specific model in study. On the one hand, this kind of approach allows to exploit particular structures of each concrete formulation; on the other hand the possibilities of taking advantage of previous contributions when facing the same problem in a different model are scarce.

Here, we also present an ad-hoc separation strategy. Given that maximum clique is a well studied problem, a natural strategy to approach separation is to make use of the existing knowledge about clique searching in undirected graphs. The hardness of the problem, together with quadratic size of G_{splpi} in m and n, can be discouraging. Of course, expecting an excellent performance in all cases would be unrealistic, but separation can be advantageous as long as it is carefully implemented. In our case, we have considered an efficient algorithm to find all the cliques of a graph and shaped it at our convenience to exploit our problem characteristics. In the following subsections, we introduce the original algorithm first and explain how to adapt it to find violated clique facets in SPLPI afterwards.

Finding all cliques of a graph

The problem of finding all cliques of an undirected graph was effectively tackled and solved in Bron & Kerbosch (1973). They proposed a backtracking algorithm with the advantage that it

does not report duplicated cliques, nor subgraphs of cliques.

The algorithm consists in a main recursive procedure which maintains three subsets of nodes: *compsub*, *candidates* and *not*. The first set, *compsub*, keeps the nodes of the clique in construction; *candidates* is the set of nodes that could potentially extend the current clique (every node in *candidates* is adjacent to every node in *compsub*); finally, *not* contains nodes that, in a previous step of the recursion, served as an extension of the current clique in construction, *compsub*. To obtain all the cliques of the graph, this main procedure is called in first place with arguments $candidates = V$, $compsub = \emptyset$ and $not = \emptyset$. When this first call returns, all cliques of the graph have been enumerated. One recursive call ends when $candidates = \emptyset$ or when the search is pruned. If $candidates = \emptyset$ and *not* is non-empty, *compsub* is part of an already generated clique, and it is discarded. Otherwise, if both *candidates* and *not* are empty sets, *compsub* is a new clique. On the other hand if, at some stage of the algorithm, set *not* contains a point adjacent to all points of *candidates*, it can be predicted that no further extensions of *compsub* will lead to a new clique. If such situation is detected, the recursive call returns (prune).

Algorithm 2.1 shows a scheme of this recursive procedure, together with the auxiliary function that checks if a branch of the clique search has to be pruned. The selection of a candidate in line 8 can be done by following different criteria. Bron & Kerbosch (1973) followed an early pruning strategy, which we have also used in our implementation.

Our clique separation algorithm

The main ingredients in our separation algorithm are the conflict graph G_{splpi} and a fractional solution of $(SPLPI)$ that corresponds to the linear relaxation of the current node of the branch and bound tree. We will identify such solution with weights of the corresponding nodes in G_{splpi}. Then, we will look for cliques in G_{splpi} with total weight greater than one or, equivalently, violated clique facets. We recall that the only clique facets of $\mathcal{B}_{splpi}$ that are not in the formulation, which are the ones to separate, are facets (ii) of Proposition 2.1 with $|C| > 1$.

We incorporated the algorithm proposed by Bron and Kerbosch in the separation process, and we applied it to G_{splpi}. We modified this algorithm considering the following observations. First, due to Proposition 2.1, a violated clique will always include only one y-node. This fact yields immediately from Proposition 2.1. Second, any node with weight equal to 1 will never be in a violated clique. Finally, if the total weight of *compsub* and *candidates* is less than or equal to 1, the corresponding branch can be pruned. Our separation algorithm basically consists in calling the procedure of Bron and Kerbosch with $compsub = \{y_j\}$, $candidates = N(y_j) \setminus \{x_{ij} : w(x_{ij}) = 1\}$ for all plants j such that $w(y_j) \neq 1$ and $not = \emptyset$ (we denote with $w(\cdot)$ the weight function on graph G_{splpi}, which coincides with the solution of the current relaxation). Our implementation of the algorithm includes a slight modification in the pruning criterion (to discard few weighted cliques) and only reports violated cliques. To lighten G_{splpi}, edges between y_j and x_{ij} are not considered if client i has no incompatibilities or that assignment is not allowed (cost c_{ij} is infinite). Despite the modifications included, our implementation is exhaustive: it finds all violated cliques in G_{splpi}.

In Section 2.3.1 we have shown that one clique $C \subseteq I$ in G_I produces m cliques in G_{splpi} as "copies" of C for the different plants. Conversely, the described separation systematically looks for cliques in G_{splpi}, without leveraging its relation with G_I. The reason for this is that working directly on G_{splpi} allows to use information about the solution of the relaxation as node weights and consequently exploits it to prune the search. Such strategy turned out to be

Algorithm 2.1 Finding all cliques of an undirected graph, Bron & Kerbosch (1973)

```
0: global variables
1:     G = (V, E), global var
2:     compsub = ∅, global var
3: end global variables
4: procedure EXTEND(candidates, not)
5:     new_candidates, local var
6:     new_not, local var
7:     while ¬ TestPrune(candidates, not) do
8:         v=SelectNode(candidates)
9:         Add v to compsub
10:        Copy candidates in new_candidates
11:        Copy not in new_not
12:        for all u ∈ new_candidates do
13:            if (u, v) ∉ E then Remove u from new_candidates
14:            end if
15:        end for
16:        for all u ∈ new_not do
17:            if (u, v) ∉ E then
18:                remove u from new_not
19:            end if
20:        end for
21:        if new_not = ∅ and new_candidates = ∅ then
22:            store compsub as new clique
23:        else if new_candidates <> ∅ then
24:            Extend(new_candidates, new_not)
25:        end if
26:        Remove v from compsub
27:        Add v to not.
28:    end while
29: end procedure
30: function TESTPRUNE(candidates, not)
31:    prune=false, local var
32:    allAdj, local var
33:    for all u ∈ not do
34:        allAdj=true
35:        for all v ∈ candidates do
36:            if (u, v) ∉ E then
37:                allAdj=false
38:                stop loop
39:            end if
40:        end for
41:        prune = allAdj
42:        if prune then
43:            stop loop
44:        end if
45:    end for
46: return prune
47: end function
```

more advantageous than a systematic search for cliques on G_I plus violation check out of the copies in G_{splpi}. The performance of our clique separation strategy is extensively analyzed in upcoming Section 2.4, were computational tests on a set of benchmarks are reported.

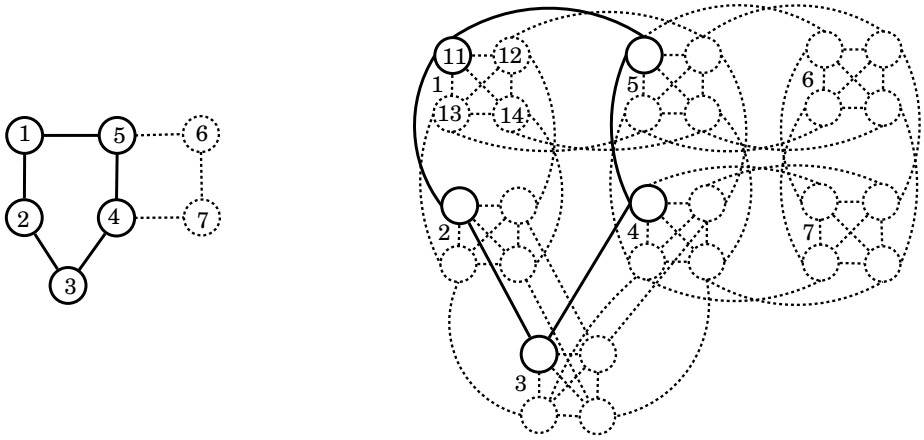
2.3.3 Hole inequalities

An odd hole $(1, \ldots, r)$ defines the facet $\sum_{v=1}^{r} t_v \leq \lfloor \frac{r}{2} \rfloor$. In a similar way as it happened to cliques, holes of G_{splpi} can be obtained from certain structures in G_I. Concretely, in this section we will address three different types of holes in G_{splpi}, which are obtained from holes and paths of G_I. The following example shows these three cases when the length of the hole or path in G_I is as short as possible.

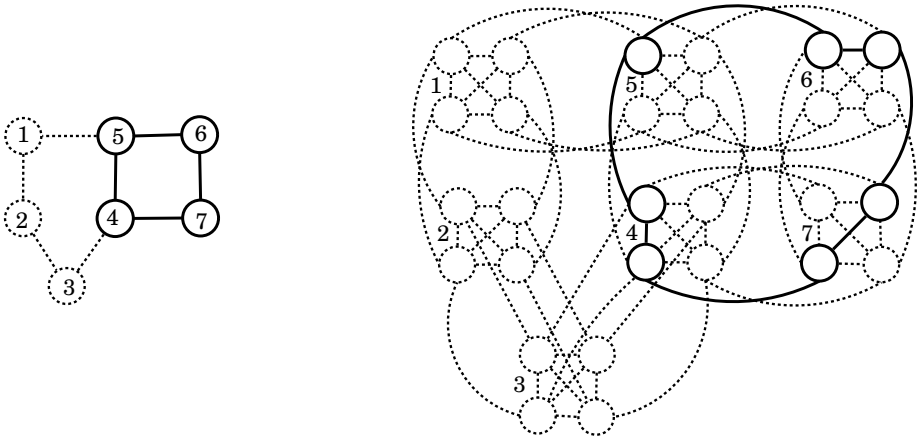
Example 2.3. Figure 2.4 is composed by three subfigures, which depict three pairs of different structures in G_I coupled with their corresponding odd holes in G_{splpi}. The left-hand sides of the three subfigures refer to the same incompatibility graph, but different holes and a path are emphasized in continuous trace in each case. Next to them, odd holes in the corresponding conflict graph of (SPLPI) are shown in each of the three cases. In these examples, we take $m = 4$ and for the sake of clarity we omit y-nodes when they are not relevant. Figure 2.4a shows how an odd hole in G_I produces an odd hole in G_{splpi}. On the left hand side of this subfigure we see an odd hole induced by clients $1, \ldots, 5$, while on its right hand side the same hole is reproduced by taking the first node from each client group $X_1, \ldots, X_5$. Similarly, we can find holes of the same type but choosing the second, the third... node in each group. Figure 2.4b illustrates how we can obtain an odd hole in G_{splpi} from a hole of arbitrary length in G_I. The left hand side of this subfigure shows a hole of length four in G_I, composed by nodes of clients $4, \ldots, 7$. On the right hand side, the incompatibility edges corresponding with the previous hole allow the formation of a hole of length seven in G_{splpi}. This hole is made of two nodes of each subgraph X_4, X_6 and X_7 and one node of X_5. Finally, Figure 2.4c relates non-closed paths of any length in G_I with odd holes in G_{splpi}. Path $\langle 3, 4, 7 \rangle$ in G_I produces an odd hole in G_{splpi} made of two nodes of each X_3, X_4, X_7 plus one y-node. △

In the following, we generalize Example 2.3 and present three families of odd-hole inequalities of G_{splpi} for arbitrary large holes in G_{splpi}. In our analysis, we will use either $H = (i_1, i_2, \ldots, i_\ell)$ to denote a hole in G_I of length ℓ or $P = \langle i_1, i_2, \ldots, i_\ell \rangle$ to denote a path without chords in G_I of length $\ell - 1$, $i_r \in I$, $r = 1, \ldots, \ell$. Linked to H or P, we will take an odd hole in G_{splpi}, which we will name H_{splpi}. We denote by $\Pi_{H_{splpi}}$ the odd-hole inequality produced by H_{splpi}, $x(H_{splpi}) + y(H_{splpi}) \leq \left\lfloor \frac{|H_{splpi}|}{2} \right\rfloor$, where $x(H_{splpi})$ (resp. $y(H_{splpi})$) represents the sum of x-variables (resp. y-variables) corresponding with the nodes of H_{splpi}. These variables will be indexed by the clients in H or P, depending on the case (note that nodes in H are also indices of I, and the same happens for P). Without loss of generality, we will assume from now on that $i_r = r$, $r = 1, \ldots, \ell$, i.e., we consider either a hole $H = (1, \ldots, \ell)$ or a path without chords $P = \langle 1, \ldots, \ell \rangle$ in G_I. To refer to these nodes as clients indices, we will use the notation $I_H = \{1, \ldots, \ell\}$ in the case of H and $I_P = \{1, \ldots, \ell\}$ in the case of P. For the sake of consistency, we will denote the plant indices taking part in $\Pi_{H_{splpi}}$ as J_H or J_P, depending on the case.

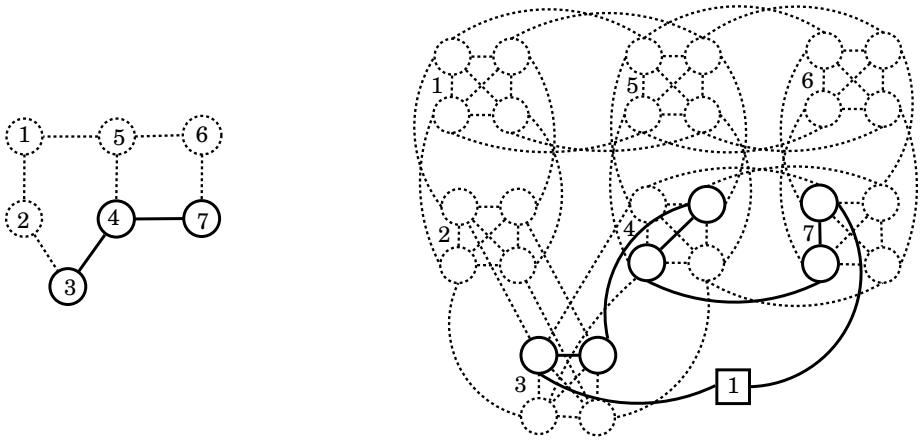
We will study three types of holes in G_{splpi} and their corresponding inequalities. For the sake of simplicity in the exposition, we will use x_{ij} and y_j also to denote the nodes of G_{splpi} that correspond with these variables. We will also use H_{splpi} indistinctly to refer to the nodes of the hole or to the hole itself.



(a) From odd hole of G_I



(b) From hole of G_I



(c) From path of G_I

Figure 2.4: Three ways of extracting holes of G_{splpi}

(a) When $H = (1, \ldots, \ell)$ and ℓ is odd, once an index j is fixed, we take

$$H_{splpi} = (x_{1j}, \ldots, x_{\ell j}),$$

the hole composed of nodes (i.e. variables) x_{ij}, $i \in I_H$ (i.e. one node of X_i, $i \in I_H$). In this case $J_H = \{j\}$. The corresponding valid inequality for the set packing formulation is

$$\sum_{i \in I_H} x_{ij} \leq \frac{\ell - 1}{2}. \tag{2.10}$$

Figure 2.4a shows a hole of this type when $\ell = 5$, which would produce inequality $x_{11} + x_{21} + x_{31} + x_{41} + x_{51} \leq 2$.

(b) When $H = (1, \ldots, \ell)$ (ℓ odd or even), we take

$$H_{splpi} = (x_{1j_1}, x_{2j_1}, x_{2j_2}, x_{3j_2}, x_{3j_3}, \ldots, x_{\ell-1,j_{\ell-1}}, x_{\ell j_{\ell-1}}, x_{\ell j_1}),$$

$j_r \in J$, $j_r \neq j_{r-1}$ for all $r = 2, \ldots, \ell - 1$ (i.e. one node of X_1 and two nodes of every X_i, $i \in I_H \setminus \{1\}$), which gives rise to the valid inequality

$$\sum_{i=1}^{\ell-1} (x_{ij_i} + x_{i+1,j_i}) + x_{\ell j_1} \leq \ell - 1. \tag{2.11}$$

We will refer the set of different plant indices in $\{j_1, j_2, \ldots, j_{\ell-1}\}$ as J_H. It is important to notice that these indices must be chosen so that H_{splpi} does not contain chords (i.e., $j_1 \neq j_3$, $j_2 \neq j_4$, $j_3 \neq j_5 \ldots$). Figure 2.4b depicts this type of hole when $\ell = 4$, producing inequality $x_{51} + x_{61} + x_{62} + x_{72} + x_{73} + x_{43} + x_{41} \leq 3$ in this case.

(c) When $P = \langle 1, \ldots, \ell \rangle$ (ℓ odd or even), we consider

$$H_{splpi} = (y_{j_1}, x_{1j_1}, x_{1j_2}, x_{2j_2}, x_{2j_3}, \ldots, x_{\ell j_\ell}, x_{\ell j_1}),$$

$j_r \in J$, $j_r \neq j_{r-1}$ for all $r = 2, \ldots, \ell$ (i.e. two nodes of X_i, $i \in I_P$, plus one y-node), which gives rise to the valid inequality

$$y_{j_1} + \sum_{i=1}^{\ell-1} (x_{ij_i} + x_{ij_{i+1}}) + x_{\ell j_\ell} + x_{\ell j_1} \leq \ell. \tag{2.12}$$

We will refer the set of different plant indices in $\{j_r : \ r = 1, \ldots, \ell\}$ as J_P. Again, there are some requirements on these indices to avoid chords in the produced hole. Concretely, $j_r \neq j_{r+2}$ $\forall r = 2, \ldots, \ell - 2$ avoids chords between nodes of consecutive subgraphs X_i, $i \in I_P$, and $j_r \neq j_1$ $\forall r = 2, \ldots, \ell$ forbids chords between y_{j_1} and x-nodes of H_{splpi}. Figure 2.4c depicts this type of hole when $\ell = 3$, which has the associated inequality $y_1 + x_{31} + x_{32} + x_{42} + x_{43} + x_{73} + x_{71} \leq 3$.

Expressions (2.10), (2.11) and (2.12) are facets of the set packing restricted to the corresponding H_{splpi} and valid inequalities for (SPLPI). In the following, we show how to lift these three families of inequalities so that they become facets of (SPLPI), i.e., facets of the set packing problem over the whole graph G_{splpi}. The next subsection describes specific lifting procedures for the three cases and concludes with a separation algorithm to manage the resulting three types of facets together.

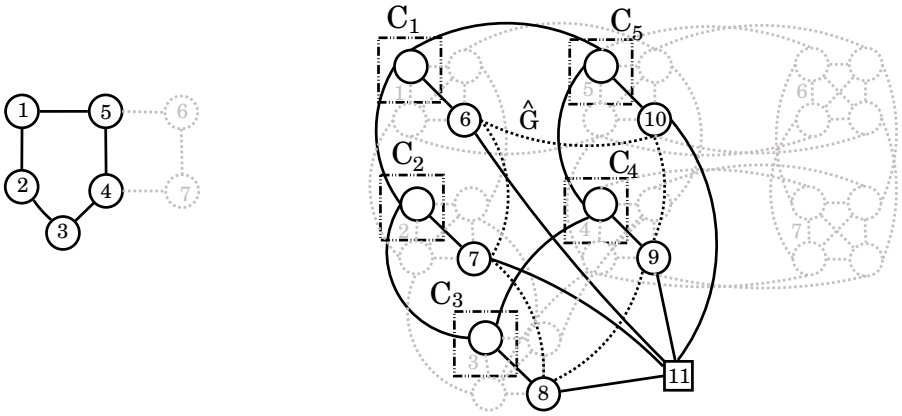


Figure 2.5: Application of Theorem 1.1 to hole of type (a) on Figure 2.4a

2.3.4 Lifting of odd holes

This subsection is devoted to the lifting of (2.10), (2.11) and (2.12). There are some results that will help in the lifting, namely Theorem 1.1 stated and proved in Chapter 1 and sequential lifting. For sequential lifting, we recall that variables are added one by one to the left hand side of the inequality at hand, with some coefficient that needs to be calculated. The right hand side and non-zero-coefficients of the original inequality remain unmodified. We invite the reader to revisit Concepts for an illustrative example of sequential lifting.

The remaining of the subsection is divided in three parts, one devoted to the lifting of each hole inequality (2.10)-(2.12). For the three cases, lifting coefficients will depend to some extent on incompatibilities between clients. In a first part, designated to holes of type (a), (2.10) is transformed into a facet of $\mathcal{B}_{splpi}$ by using just sequential lifting. In this first case, we will show that application of Theorem 1.1 (when appropriate) would have produced the same lifting coefficients. Afterwards, we will pay attention to holes of type (b), which present additional interest. First, these holes are odd by definition, despite the length of the corresponding H in G_I, i.e., despite the fact that H might be even. To our knowledge, there is no precedence of even holes as relevant objects in set packing. Secondly, the lifting of (2.11) we present is far from being a routine task; we will need Theorem 1.1 in addition to sequential lifting. Analysis of holes of type (c) concludes the section with the lifting of (2.12). The proof sketch will be very similar to that of (2.11), based on Theorem 1.1 first and completed with sequential lifting afterwards.

a. Lifting of hole inequality (2.10)

We open this first subdivision with an illustrative example of the application of Theorem 1.1 to a hole of type (a). The conflict graph we use was previously introduced by Example 2.3 and corresponding Figure 2.4a.

Example 2.4. Figure 2.5 illustrates identification of elements of Theorem 1.1 on the hole of Figure 2.4a. This hole of length $\ell = 5$, $H_{splpi} = (x_{11}, x_{21}, x_{31}, x_{41}, x_{51})$, is the graph G in Theorem 1.1 for which a facet is known, $\frac{1}{2}(x_{11} + x_{21} + x_{31} + x_{41} + x_{51}) \leq 1$ in this case. Complete subgraphs C_r are given by nodes of H_{splpi} belonging to the same X_i, i.e., $C_i = \{x_{i1}\}$ for $i = 1, \ldots, 5$. For any plant $j \notin J_H = \{1\}$, nodes x_{ij} with $i \in I_H = \{1, \ldots, 5\}$ and node y_j will play the role of $n + 1, \ldots, n + q$ and $n + q + 1$ of the theorem, respectively. In the example, $n = 5$, q is equal to $\ell = 5$ and $j = 4$. The subgraph $\hat{G}$ of the theorem is then hole

$\hat{G} = (x_{14}, x_{24}, x_{34}, x_{44}, x_{54})$, whose edges are depicted with dashed black lines on Figure 2.5. Once identified the layout described by the theorem, we have that the value of $\hat{\alpha}_0$ is

$$\hat{\alpha}_0 = \max\{|S| \ : \ S \text{ is a vertex packing in } \hat{G}\} = 2.$$

Since $\beta_{\hat{\alpha}_0} \leq 1$ by definition, condition $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$ of Theorem 1.1 holds and family of conditions (1.2) becomes empty. As a consequence, one can derive a facet of the graph induced by $\{x_{i1}\}_{i=1}^5 \cup \{x_{i4}\}_{i=1}^5 \cup \{y_4\}$ (G^* in the theorem). To write such facet, according to the theorem statement, we need to calculate $\beta_{\hat{\alpha}_0}$, that is

$$\begin{aligned}\beta_{\hat{\alpha}_0} = \beta_2 = \max \quad & \{\max\{\tfrac{1}{2}\textstyle\sum_{i=1}^5 x_{i1} \ : \{x_{i1} : x_{i1} = 1\} \in \mathcal{P}_{H_{splpi}},\ x_{i1} = 0 \ \forall i \in K\} \\ & \qquad : K \subseteq \{1, \ldots, 5\}, |K| = 2 \text{ and } \{x_{i4} = 1\}_{i \in K} \in \mathcal{P}_{\hat{G}}\}.\end{aligned}$$

Consider then a packing of cardinality 2 in $\hat{G}$. A maximum vertex packing in G^* that completes the previous packing has 2 nodes of H_{splpi}. Then, $\beta_2 = 1$ and the theorem proves that $\frac{1}{2}(x_{11} + x_{21} + x_{31} + x_{41} + x_{51}) \leq 1$ is still a facet of the augmented graph G^*. △

Given a hole $H_{splpi} = (x_{1j}, \ldots, x_{\ell j})$ with $\ell \geq 5$ and odd, Theorem 1.1 can be applied as illustrated by Example 2.4 to lift the group of variables $\{x_{1h}, \ldots, x_{\ell h}\} \cup \{y_h\}$ for $h \neq j$. Indeed, $\hat{G} = (x_{1h}, \ldots, x_{\ell h})$ would be a hole of length ℓ, which implies $\hat{\alpha}_0 = \lfloor \frac{\ell}{2} \rfloor$. Since $\hat{G}$ and H_{splpi} are in general two odd holes linked by ℓ edges displayed on a "wheel" layout, a maximum packing in $\hat{G}$ of $\lfloor \frac{\ell}{2} \rfloor$ can be completed with $\lfloor \frac{\ell}{2} \rfloor$ "complementary nodes" of H_{splpi}. This fact implies that $\beta_{\hat{\alpha}_0} = 1$, which ensures that the theorem conditions stand. Theorem 1.1 then gives that the original hole inequality $x_{1j} + \ldots + x_{\ell j} \leq \lfloor \frac{\ell}{2} \rfloor$ is a facet of the graph induced by $\{x_{ij}\}_{i=1}^{\ell} \cup \{x_{ih}\}_{i=1}^{\ell} \cup \{y_h\}$. Moreover, a similar argument proves that successive applications of Theorem 1.1 yield the same facet for $\{x_{ij}\}_{i=1}^{\ell} \cup (\cup_{h \neq j}(\{x_{ih}\}_{i=1}^{\ell} \cup \{y_h\}))$.

The following proposition describes a set of variables that have zero coefficients in sequential lifting, including x_{ih} and y_h for any $h \neq j$. In other words, sequential lifting produces the same facet as Theorem 1.1 when applied to hole inequalities of type (a). Starting from (2.10), the proof of the proposition presents a sequential lifting of variables by different groups, getting an inequality of higher dimension in each step. We will denote the neighborhood of a client i in G_I as $N_{G_I}(i) = \{k \in I \ : \ (i,k) \in E_I\}$.

Proposition 2.2. *Every sequential lifting of (2.10) produces coefficients 0 for the following variables (recall that $J_H = \{j\}$):*

(i) y_g *with* $g \neq j$,

(ii) x_{ig} *with* $i \notin I_H$ *and* $g \neq j$,

(iii) x_{ig} *with* $i \in I_H$ *and* $g \neq j$ *and*

(iv) x_{ij} *with* $i \notin I_H$ *and* $N_{G_I}(i) \cap I_H \leq 2$.

Moreover, y_j *has coefficient* $\frac{\ell - 1}{2}$.

Proof.

(i) Despite the order in which we include variables during the lifting process, coefficients of y_g-variables are 0 if $g \neq j$, since y_g has no neighbors in H_{splpi}.

(ii) For the same reason, x_{ig} has coefficient 0 if $i \notin I_H$ and $g \neq j$, in spite of the lifting order.

(iii) Observe now that every x_{ig} with $i \in I_H$ and $g \neq j$ has only one neighbor with its coefficient in the lifted inequality different to 0, which is x_{ij}. This fact is due to (i) and (ii) and thus it is independent of the lifting order. But it results then that coefficients of the mentioned x-variables are also 0.

(iv) Consider the set of variables x_{ij} with $i \notin I_H$ and $N_{G_I}(i) \cap H \leq 2$. Pick one variable of the set, x_{ij}. Despite the state of the lifting process, we can find a set packing of $\ell - 1$ nodes in H_{splpi} that does not include any neighbor of x_{ij}. This ensures that the coefficient of x_{ij} in the lifted inequality is 0, despite the lifting order.

It remains to prove the last assertion of the proposition. To this end, observe that every variable different to y_j and its neighbors, which are x_{ij} for all $i \in I$, has lifting coefficient 0. Therefore, every sequential lifting produces coefficient $\frac{\ell-1}{2}$ for y_j. □

Then, the resulting inequality after applying sequential lifting in arbitrary order would be

$$\frac{\ell-1}{2} y_j + \sum_{i \in I_H} x_{ij} + \sum_{\substack{i \notin I_H: \\ N_{G_I}(i) \cap H > 2}} \gamma_i x_{ij} \leq \frac{\ell-1}{2}. \tag{2.13}$$

Inequality (2.13) is the general form of the facet of G_{splpi} obtained after applying sequential lifting to (2.10). We have omitted in the formula those variables that have lifting coefficient 0 by Proposition 2.2 and included y_j with its lifting coefficient, also given in this result. Coefficients γ_i will depend on the structure of the incompatibilities given by graph G_I. Working out one of these coefficients could be done by sequential lifting, but it will get increasingly harder for the following nodes, since each coefficient depends on all the previous.

We have devised a procedure to calculate coefficients γ_i in (2.13) when $\ell = 5$. To present it, we can assume that $j = 1$ in (2.13) without loss of generality. Consider then L a list that contains all the nodes to be lifted. Initially, $L := \{x_{i1} : i \notin I_H, |N_{G_I}(i) \cap H| > 2\}$. We will remove nodes from L as soon as we calculate its lifting coefficients, in the following order. First, we will consider those nodes with $|N_{G_I}(i) \cap H| = 5$, then those with $|N_{G_I}(i) \cap H| = 4$ and nodes of clients with three neighbors in H will be taken in the last place. We will start from the facet $2y_1 + \sum_{i \in I_H} x_{i1} \leq 2$ of the subgraph induced by those nodes that are not in L.

(i) Nodes $x_{i1} \in L$, $|N_{G_I}(i) \cap H| = 5$. The first node we take in these conditions has coefficient 2. Moreover, we know that every node $x_{t1} \in L$ that is not adjacent to x_{i1} has coefficient 0. Therefore we can update the list of nodes to be lifted: $L = L \setminus (\{x_{t1} : (i,t) \notin E_I\} \cup \{x_{i1}\})$. The next node we take from L fitting in this group will have again coefficient 2, since it will be a neighbor of x_{i1}. Updating L in the same manner and repeating the process as many times as needed, we complete the lifting of this group of nodes.

(ii) Nodes $x_{i1} \in L$, $|N_{G_I}(i) \cap H| = 4$. Nodes previously added to the initial facet will not be relevant to calculate γ_i, since x_{i1} is adjacent to all of them.
Suppose that $x_{i_1 1}$ is the first node of this group that we are going to lift. We have $\gamma_{i_1} = 1$. Let $x_{k_1 1}$ be the only node in H_{splpi} such that $(i_1, k_1) \notin E_I$. Observe that nodes x_{t1} such that client t is not adjacent to i_1 nor k_1 lift with coefficient 0. Remove those nodes from L and take the next node $x_{i_2 1}$ in L such that $|N_{G_I}(i_2) \cap H| = 4$. Such a node will have lifting coefficient 1. Let $x_{k_2 1}$ be the only node in H_{splpi} such that $(i_2, k_2) \notin E_I$. Observe that nodes x_{t1} such that client t is not adjacent to i_2 nor k_2 lift with coefficient 0, and the same happens if t is not adjacent to i_1 nor i_2 and $(i_1, i_2) \notin E$. At step r, let $x_{k_r 1}$ be the only node in H_{splpi} such that $(i_r, k_r) \notin E_I$. Suppose that, in every step r, nodes x_{t1}

such that client t is not adjacent to i_r nor k_r are removed from L.
Statement: Given that $x_{i_1 1}, \ldots, x_{i_{r-1} 1}$ have been lifted with coefficient 1, consider, among them, the subset of nodes that are not adjacent to $x_{i_r 1}$. If the subgraph induced by such subset is a complete subgraph, then $\gamma_{i_r} = 1$.
Proof: A packing with $x_{i_r 1} = 1$ contains one node from $x_{i_1 1}, \ldots, x_{i_{r-1} 1}$ at most, say $x_{i_s 1}$. Such packing could include, at the most, the companion node $x_{k_r 1}$. But, if $(i_s, k_r) \notin E$, $x_{i_s 1}$ would have been deleted from L in a previous step. Then, exactly one of $x_{i_s 1}$ and $x_{k_r 1}$ are in any packing that contains $x_{i_r 1}$, which yields $\gamma_{i_r} = 1$. □
Suppose that the subset of nodes among $x_{i_1 1}, \ldots, x_{i_{r-1} 1}$ that are not adjacent to $x_{i_r 1}$ induce a complete subgraph. To maintain this property in the upcoming steps, we must remove some nodes from L. Concretely, those x_{t1} that are not adjacent to the new pairs of non-adjacent nodes that arise from the inclusion of $x_{i_r 1}$ in the sequence of lifted nodes, i.e., x_{t1} not adjacent to pairs $x_{i_r 1}$ and $x_{i_s 1}$, $s = 1, \ldots, r-1$, $(i_s, i_r \notin E)$. Finally, in step r we do $\gamma_{i_r} = 1$ and successively update:

$$L = L \setminus \{x_{t1} \, : \, (i_r, t) \notin E_I \text{ and } (k_r, t) \notin E_I\},$$
$$L = L \setminus \{x_{t1} \, : \, (i_r, t) \notin E_I \text{ and } (i_s, t) \notin E_I\},$$

for all $s = 1, \ldots, r-1$ such that $x_{i_s 1}$ is not adjacent to $x_{i_r 1}$.

(iii) Nodes $x_{i1} \in L$, $|N_{G_I}(i) \cap H| = 3$. We will only consider nodes with three consecutive neighbors in H (otherwise $\gamma_i = 0$). Given $x_{i_r 1}$ a node in this group, let $x_{k_r,1}$ and $x_{h_r,1}$ be such that $(i_r, k_r) \notin E_I$ and $(i_r, h_r) \notin E_I$. Consider the first node to be lifted, $x_{i_1 1}$. In a packing with $x_{i_1 1} = 1$, at most one from $x_{k_1,1}$ and $x_{h_1,1}$ can be included. Due to the strategy we are following, no other node with non-zero coefficient can be included in the packing. Then, $\gamma_{i_1} = 1$. Next, we update L, removing those nodes that we can assure that have zero coefficient, i.e., $L = L \setminus \big(\{x_{t1} \, : \, (i_1, t) \notin E_I \text{ and } (k_1, t) \notin E_I\} \bigcup \{x_{t1} \, : \, (i_1, t) \notin E_I \text{ and } (h_1, t) \notin E_I\}\big)$. The general case is analogous to that of the previous group of nodes. By induction, assume that $x_{i_1 1}, \ldots, x_{i_{r-1} 1}$ have been lifted with coefficient 1 and that the set of non-adjacent nodes to $x_{i_r 1}$ among that sequence induce a complete subgraph. Then, $\gamma_{i_r} = 1$. Moreover, after the following modifications

$$L = L \setminus \{x_{t1} \, : \, (i_r, t) \notin E_I \text{ and } (k_r, t) \notin E_I\},$$
$$L = L \setminus \{x_{t1} \, : \, (i_r, t) \notin E_I \text{ and } (h_r, t) \notin E_I\},$$
$$L = L \setminus \{x_{t1} \, : \, (i_r, t) \notin E_I \text{ and } (i_s, t) \notin E_I\},$$

for all $s = 1, \ldots, r-1$ such that $x_{i_s 1}$ is not adjacent to $x_{i_r 1}$, the next node taken from L will have lifting coefficient 1.

b. Lifting of hole inequality (2.11)

Here, we lift hole inequalities of type (b) by using Theorem 1.1 several times and sequential lifting afterwards. The following example illustrates the application of Theorem 1.1 to the most basic hole described by Example 2.3 and corresponding Figure 2.4b.

Example 2.5. Figure 2.6 shows a hole of length seven on its right hand side, $H_{splpi} = (x_{51}, x_{61}, x_{62}, x_{72}, x_{73}, x_{43}, x_{41})$, produced by the hole of length $\ell = 4$ depicted on the left. In order to lift the corresponding inequality, $x_{51} + x_{61} + x_{62} + x_{72} + x_{73} + x_{43} + x_{41} \leq 3$, with Theorem 1.1, the following elements are identified. First, $C_1 = \{x_{51}\}$, $C_2 = \{x_{61}, x_{62}\}$, $C_3 = \{x_{72}, x_{73}\}$, $C_4 = \{x_{43}, x_{41}\}$ induce complete subgraphs. Then, x_{5h}, x_{6h}, x_{7h} and x_{4h} with

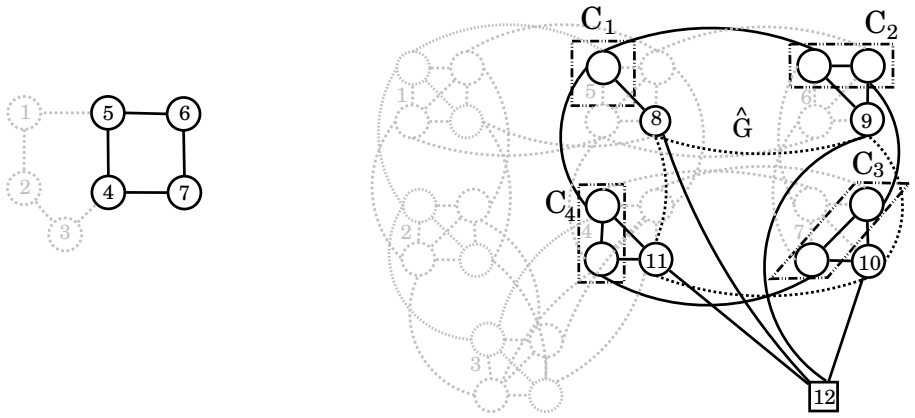


Figure 2.6: Application of Theorem 1.1 to hole of type (b) on Figure 2.4b

$h = 4 \notin J_H = \{1, 2, 3\}$ will play the role of $n + 1, \ldots, n + q$ of the theorem (note that $n = 7$ and q would be equal to $\ell = 4$ in this case). Node y_h is identified with node $n + q + 1$ of the theorem. Observe that then $\hat{G}$ would be a hole of length four, $\hat{G} = (x_{44}, x_{54}, x_{64}, x_{74})$, depicted with dashed black lines on Figure 2.6. Consequently,

$$\hat{\alpha}_0 = \max\{|S| \ : \ S \text{ is a vertex packing in } \hat{G}\} = 2,$$

condition $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$ of Theorem 1.1 holds (since $\beta_{\hat{\alpha}_0} \leq 1$ by definition) and family of conditions (1.2) becomes empty. To obtain the facet given by the theorem, $\beta_{\hat{\alpha}_0} = \beta_2$ has to be computed as

$$\begin{aligned}\beta_2 = \max \quad & \{\max\{\tfrac{1}{3}(x_{51} + x_{61} + x_{62} + x_{72} + x_{73} + x_{43} + x_{41}) \ : \\ & \{x_{ij} \in H_{splpi} : x_{ij} = 1\} \in \mathcal{P}_{H_{splpi}},\ x_{ij} = 0 \ \forall i \in K, x_{ij} \in H_{splpi}\} \\ & : K \subseteq \{4, 5, 6, 7\}, |K| = 2 \text{ and } \{x_{i4} = 1\}_{i \in K} \in \mathcal{P}_{\hat{G}}\}.\end{aligned}$$

Consider a packing of cardinality 2 in the cycle $(x_{44}, x_{54}, x_{64}, x_{74})$. To extend this packing, we build a vertex packing with no nodes of two of the complete subgraphs C_r's, say C_{r_1} and C_{r_2}. We can always include in the packing one node of each C_i at most, $i \neq r_1, r_2$, which yields $\beta_2 = 2/3$. The theorem states that

$$\frac{2-1}{3}(x_{51} + x_{61} + x_{62} + x_{72} + x_{73} + x_{43} + x_{41}) + \Big(1 - \frac{2}{3}\Big)\Big(y_4 + \sum_{i=4}^{7} x_{i4}\Big) \leq 2 - \frac{2}{3},$$

which simplifies to

$$x_{51} + x_{61} + x_{62} + x_{72} + x_{73} + x_{43} + x_{41} + y_4 + \sum_{i=4}^{7} x_{i4} \leq 4,$$

is a facet of the graph induced by $x_{51}, x_{61}, x_{62}, x_{72}, x_{73}, x_{43}, x_{41}, y_4, x_{44}, x_{54}, x_{64}, x_{74}$. △

Let us consider now a general hole of type (b) in the conflict graph, $H_{splpi} = (x_{1j_1}, x_{2j_1}, x_{2j_2}, x_{3j_2}, x_{3j_3}, \ldots, x_{\ell-1,j_{\ell-1}}, x_{\ell j_{\ell-1}}, x_{\ell j_1})$, and recall that I_H and J_H are the different clients and plants indices in H_{splpi}. Consider the associated hole inequality in the form of $\pi t \leq 1$ in the theorem, i.e., with a 1 in its right-hand side:

$$\frac{1}{\ell - 1}\Big(\sum_{i=1}^{\ell-1}(x_{ij_i} + x_{i+1,j_i}) + x_{\ell j_1}\Big) \leq 1. \tag{2.14}$$

Inequality (2.14) defines a facet of the graph induced by H_{splpi}, $G_{splpi}[H_{splpi}]$, Padberg (1973).

To apply Theorem 1.1, we can identify the elements that appear in it in the following way: $q = \ell$, $C_1 = \{x_{1j_1}\}$, $C_i = \{x_{i,j_{i-1}}, x_{ij_i}\}$, $i = 2, \ldots, \ell-1$, $C_\ell = \{x_{\ell,j_{\ell-1}}, x_{\ell j_1}\}$ and $G = G_{splpi}[H_{splpi}]$. Observe now that, when a plant $\jmath_1 \notin J_H$ is fixed, variables $y_{\jmath_1}$ and $x_{i\jmath_1}$ for all $i = 1, \ldots, \ell$ can be identified with x_{n+q+1} and x_{n+i} of Theorem 1.1, $i = 1, \ldots, q$ (see again Figure 2.6 to illustrate case $\ell = 4$). Before applying Theorem 1.1, we have to check that the required conditions are met in this scenario. Since $x_{i\jmath_1}$ with $i = 1, \ldots, \ell$ induce a hole, a maximum packing of these nodes has cardinality $\lfloor \ell/2 \rfloor$. On the other hand, since C_i are disjoint parts of the hole H_{splpi}, any packing in G^* can satisfy $\sum_{j \in C_i} x_j = 1$ for every $i = 2, \ldots, q$ such that node $n+i$ is not in the packing. Coefficients $\hat{\alpha_0}$ and β_k, $k = 2, \ldots, \hat{\alpha_0}$, are then:

$$\hat{\alpha_0} = \left\lfloor \frac{\ell}{2} \right\rfloor, \qquad \beta_k = \frac{\ell - k}{\ell - 1}.$$

The condition of the theorem, $\hat{\alpha_0} \neq \beta_{\hat{\alpha_0}} (= \frac{\ell - \lfloor \ell/2 \rfloor}{\ell - 1})$ is true when $\ell > 3$. Conditions (1.2) are in this case:

$$\frac{\ell - k}{\ell - 1} \leq 1 - \frac{(k-1)(1 - \frac{\ell - \lfloor \ell/2 \rfloor}{\ell - 1})}{\lfloor \frac{\ell}{2} \rfloor - 1} \qquad \forall k = 2, \ldots, \left\lfloor \frac{\ell}{2} \right\rfloor - 1,$$

which, after simplifications, turn to stand as equalities. Therefore, assuming that $\ell \geq 4$, we can apply Theorem 1.1 to lift (2.14) and get the inequality

$$\frac{\hat{\alpha_0} - 1}{\ell - 1} \Big(\sum_{i=1}^{\ell-1} (x_{ij_i} + x_{i+1,j_i}) + x_{\ell j_1} \Big) + (1 - \beta_{\hat{\alpha_0}}) \Big(y_{\jmath_1} + \sum_{i=1}^{\ell} x_{i\jmath_1} \Big) \leq \hat{\alpha_0} - \beta_{\hat{\alpha_0}}.$$

To simplify this last inequality, note that

$$1 - \beta_{\hat{\alpha_0}} = 1 - \frac{\lceil \frac{\ell}{2} \rceil}{\ell - 1} = \frac{\ell - 1 - \lceil \frac{\ell}{2} \rceil}{\ell - 1} = \frac{\lfloor \frac{\ell}{2} \rfloor - 1}{\ell - 1}, \text{ and}$$

$$\hat{\alpha_0} - \beta_{\hat{\alpha_0}} = \left\lfloor \frac{\ell}{2} \right\rfloor - \frac{\lceil \frac{\ell}{2} \rceil}{\ell - 1} = \frac{\ell \cdot \lfloor \frac{\ell}{2} \rfloor - \lfloor \frac{\ell}{2} \rfloor - \lceil \frac{\ell}{2} \rceil}{\ell - 1} = \frac{\ell \cdot \lfloor \frac{\ell}{2} \rfloor - \ell}{\ell - 1} = \ell \cdot \frac{\lfloor \frac{\ell}{2} \rfloor - 1}{\ell - 1}.$$

Then, the lifted inequality of the theorem is

$$\frac{1}{\ell} \Big(\sum_{i=1}^{\ell-1} (x_{ij_i} + x_{i+1,j_i}) + x_{\ell j_1} + y_{\jmath_1} + \sum_{i=1}^{\ell} x_{i\jmath_1} \Big) \leq 1. \tag{2.15}$$

Conflict graph G_{splpi} has a reiterative structure that allows to apply Theorem 1.1 successively. Take now $G = G_{splpi}[H_{splpi} \cup \{y_{\jmath_1}\} \cup \{x_{1\jmath_1}, \ldots, x_{\ell\jmath_1}\}]$ with $C_1 = \{x_{1j_1}, x_{1\jmath_1}\}$, $C_i = \{x_{i,j_{i-1}}, x_{ij_i}, x_{i\jmath_1}\}$, $i = 2, \ldots, \ell-1$, and $C_\ell = \{x_{\ell,j_{\ell-1}}, x_{\ell j_1}, x_{\ell\jmath_1}\}$. Inequality (2.15) defines a facet of such subgraph. Picking $\jmath_2 \notin J_H \cup \{\jmath_1\}$ and nodes $y_{\jmath_2}$ and $x_{i\jmath_2}$ for all $i = 1, \ldots, \ell$, we can repeat the previous steps to get a facet of the induced subgraph $G_{splpi}[H_{splpi} \cup \{y_{\jmath_1}, y_{\jmath_2}\} \cup \{x_{1\jmath_1}, \ldots, x_{\ell\jmath_1}\} \cup \{x_{1\jmath_2}, \ldots, x_{\ell\jmath_2}\}]$. The following result states that this process can be applied as many times as elements are in $J \setminus J_H$.

Proposition 2.3. *Suppose that*

$$H_{splpi} = (x_{1j_1}, x_{2j_1}, x_{2j_2}, x_{3j_2}, x_{3j_3}, \ldots, x_{\ell-1,j_{\ell-1}}, x_{\ell j_{\ell-1}}, x_{\ell j_1})$$

is a hole in G_{splpi} produced by a hole $(1,\dots,\ell)$ in the incompatibility graph G_I with $\ell \geq 4$. Call I_H and J_H the sets of different clients and plants indices in H_{splpi}. Then

$$\sum_{i=1}^{\ell-1}(x_{ij_i}+x_{i+1,j_i})+x_{\ell j_1}+\sum_{j\in J\setminus J_H}\Big(y_j+\sum_{i=1}^{\ell}x_{ij}\Big)\leq \ell+m-|J_H|-1 \tag{2.16}$$

is a facet of $G_{splpi}\big[H_{splpi}\cup\big(\bigcup_{j\in J\setminus J_H}\{y_j,x_{1j},\dots,x_{\ell j}\}\big)\big]$.

Proof. To verify this assertion, we proceed by induction. Suppose that $\jmath_1,\dots,\jmath_r\notin J_H$ are r different plant indices. We want to prove that the following inequality is a facet of $G_{splpi}[H_{splpi}\cup\{y_{\jmath_1},\dots,y_{\jmath_r}\}\cup\{x_{1\jmath_1},\dots,x_{\ell\jmath_1}\}\cup\cdots\cup\{x_{1\jmath_r},\dots,x_{\ell\jmath_r}\}]$:

$$\frac{1}{\ell+r-1}\left(\sum_{i=1}^{\ell-1}(x_{ij_i}+x_{i+1,j_i})+x_{\ell j_1}+\sum_{t=1}^{r}\Big(y_{\jmath_t}+\sum_{i=1}^{\ell}x_{i\jmath_t}\Big)\right)\leq 1. \tag{2.17}$$

Case $r=1$ has been already proved and gives facet (2.15). Suppose then that (2.17) holds, and take a new plant index $\jmath_{r+1}\notin J_H\cup\{\jmath_1,\dots,\jmath_r\}$. Since $x_{i\jmath_{r+1}}$ with $i=1,\dots,\ell$ induce a hole, $\hat{\alpha_0}=\lfloor\ell/2\rfloor$. To calculate $\beta_{\hat{\alpha_0}}$, we follow a similar reasoning as before, taking into account that $y_{\jmath_1},\dots,y_{\jmath_r}$ can also be included in packings. We have

$$\hat{\alpha_0}=\left\lfloor\frac{\ell}{2}\right\rfloor\qquad\text{and}\qquad\beta_k=\frac{\ell-k+r}{\ell+r-1}.$$

Clearly, $\hat{\alpha_0}\neq\beta_{\hat{\alpha_0}}(=\frac{\ell-\lfloor\ell/2\rfloor+r}{\ell+r-1})$. Conditions (1.2) are, for all $k=2,\dots,\lfloor\frac{\ell}{2}\rfloor-1$,

$$\frac{\ell-k+r}{\ell+r-1}\leq 1-\frac{(k-1)(1-\frac{\ell-\lfloor\ell/2\rfloor+r}{\ell+r-1})}{\lfloor\ell/2\rfloor-1},$$

which, after simplifications, turn to stand as equalities. Applying Theorem 1.1 we obtain

$$\frac{\hat{\alpha_0}-1}{\ell+r-1}\left(\sum_{i=1}^{\ell-1}(x_{ij_i}+x_{i+1,j_i})+x_{\ell j_1}+\sum_{t=1}^{r}\Big(y_{\jmath_t}+\sum_{i=1}^{\ell}x_{i\jmath_t}\Big)\right)$$

$$+(1-\beta_{\hat{\alpha_0}})\Big(y_{\jmath_{r+1}}+\sum_{i=1}^{\ell}x_{i\jmath_{r+1}}\Big)\leq\hat{\alpha_0}-\beta_{\hat{\alpha_0}}.$$

Working out the values of the following coefficients

$$1-\beta_{\hat{\alpha_0}}=\frac{\ell+r-1-\lceil\frac{\ell}{2}\rceil-r}{\ell+r-1}=\frac{\lfloor\frac{\ell}{2}\rfloor-1}{\ell+r-1}$$

$$\hat{\alpha_0}-\beta_{\hat{\alpha_0}}=\frac{\ell\cdot\lfloor\frac{\ell}{2}\rfloor+r\cdot\lfloor\frac{\ell}{2}\rfloor-\lfloor\frac{\ell}{2}\rfloor-\lceil\frac{\ell}{2}\rceil-r}{\ell+r-1}=\frac{(\ell+r)\cdot\lfloor\frac{\ell}{2}\rfloor-(\ell+r)}{\ell+r-1}$$

$$=\frac{(\ell+r)\cdot(\lfloor\frac{\ell}{2}\rfloor-1)}{\ell+r-1},$$

the previous inequality simplifies to:

$$\frac{1}{\ell+r}\left(\sum_{i=1}^{\ell-1}(x_{ij_i}+x_{i+1,j_i})+x_{\ell j_1}+\sum_{t=1}^{r+1}\Big(y_{\jmath_t}+\sum_{i=1}^{\ell}x_{i\jmath_t}\Big)\right)\leq 1.$$

□

As we have seen, Theorem 1.1 allows to lift a considerable number of variables in G_{splpi}, but we need to consider every variable in order to have a facet of G_{splpi} instead of a subgraph of it. To complete the lifting of inequality (2.11), we will proceed by case analysis and will distinguish three groups of variables. For the sake of clarity, we will perform the analysis taking $\ell = 4$ but the resulting facet can be also obtained for the general case by a similar reasoning. On the other hand, note that the larger is an odd hole, the less likely is that it can be violated and with $\ell = 4$ we already have a hole of length 7 in G_{splpi}.

Assume that $H_{splpi} = (x_{11}, x_{21}, x_{22}, x_{32}, x_{33}, x_{43}, x_{41})$, where we have taken without loss of generality the hole $H = (1, 2, 3, 4)$ in the incompatibility graph G_I, $I_H = \{1, 2, 3, 4\}$ and $J_H = \{1, 2, 3\}$. By Theorem 2.3, we have that

$$x_{11} + x_{21} + x_{22} + x_{32} + x_{33} + x_{43} + x_{41} + \sum_{j=4}^{m} \Big(y_j + \sum_{i=1}^{4} x_{ij} \Big) \leq m \tag{2.18}$$

is a facet of $G_{splpi}\big[H_{splpi} \bigcup \big(\bigcup_{j=4}^{m} \{y_j, x_{1j}, \ldots, x_{4j}\}\big)\big]$. In the following, we calculate the coefficients of variables in the facet of G_{splpi} that is obtained from (2.18) by sequential lifting. The reader can take holes H and H_{splpi} depicted by Figure 2.4b as reference to illustrate the analysis (if so, they should rename nodes of hole $(5, 6, 7, 4)$ by $(1, 2, 3, 4)$).

(i) We start with variables x_{ij}, $i \in I_H$ and $j \in J_H$, that have not been considered yet: x_{12}, x_{13}, x_{23}, x_{31} and x_{42}. Variables x_{12} and x_{13} have two neighbors in H_{splpi} each, so a packing that includes x_{12} or x_{13} also admits 3 nodes of H_{splpi}. On the other hand, x_{23}, x_{31} and x_{42} have three neighbors in H_{splpi} each, but it is easy to check that we can also get a packing of 3 nodes of H_{splpi} in this case. We show this for x_{23} (the analysis for x_{31} and x_{42} is analogous). Two of the three neighbors of x_{23} in H_{splpi}, x_{21} and x_{22}, are consecutive. The third one, x_{33}, has a neighbor in common with x_{22} in H_{splpi}, x_{32}. Then, $\{x_{23}, x_{32}, x_{43}, x_{11}\}$ is a packing with 3 nodes of H_{splpi} other than x_{23}. In conclusion, for every x_{ij} with $i \in I_H$ and $j \in J_H$, we can construct a packing consisting of x_{ij} and 3 nodes of H_{splpi}. This packing can be extended by adding nodes of variables $y_4, \ldots, y_m$, which have coefficient 1 in (2.18), getting a packing of m nodes other than x_{ij}. Since all coefficients in the left hand side of (2.18) are 1, the existence of a packing with m nodes among the nodes in (2.18) which are not neighbors of a given variable implies lifting it with coefficient 0. Then, variables in this first group are all lifted with 0.

(ii) Variables y_1, y_2 and y_3 lift also with coefficient 0. Nodes corresponding with y_2 and y_3 have two neighbors in H_{splpi} and then the reasoning of the previous point can be applied to them. However, the case of y_1 is different since its corresponding node has three neighbors in H_{splpi}. Include for instance x_{22} and x_{43} in a node packing that initially contains only node y_1. Then, we can also add x_{14} and x_{34}. These two nodes are not adjacent since clients 1 and 3 are not incompatible and are not adjacent to the nodes that were already in the packing since we have taken a different plant index. Finally, we can include y-nodes for the remaining plants that have not been used yet, i.e., $y_5, \ldots, y_m$. We have completed a packing of m nodes, proving that the lifting coefficient of y_1 is zero. Note that we are implicitly assuming that $m > 3 = |J_H|$.

(iii) It remains to consider x-variables corresponding with clients that do not belong to the hole, i.e., x_{ij} such that $i \notin I_H$ $j \in J$. If $j = 1, 2$ or 3, x_{ij} has at most the same neighbors as y_j in H_{splpi} (2 or 3) and we can proceed like in the lifting of y_j to conclude that x_{ij} has coefficient 0. Suppose then that $j \geq 4$. In this case, x_{ij} has no neighbors in H_{splpi},

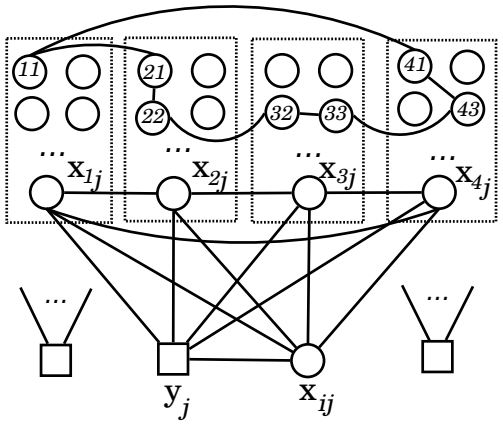


Figure 2.7: Lifting of (2.18), case (iii), x_{ij} with $i \notin I_H$ and $j \geq 4$

so 3 nodes from H_{splpi} can be easily added to the packing. Nodes y_h with $h \geq 4$ but y_j can also be included, getting a packing of $m-1$ nodes (note that $m > 3 = |J_H|$ is also assumed in this case).

This implies that all these variables lift with coefficient 0 or 1. If some node x_{rj}, $r \in \{1,2,3,4\}$, is not adjacent to x_{ij}, it can be added to the packing and the lifting coefficient of x_{ij} is 0. From now on we will only consider those variables that can potentially have coefficient 1, i.e., those x_{ij} with $i \notin I_H$ such that $(i,r) \in E \ \forall r \in \{1,2,3,4\}$. Again, we call this subset L. The first variable in L that we lift will have coefficient 1. Figure 2.7 shows this situation, where it can be noticed that x_{ij} plays exactly the same role as y_j. Note that the argument is independent of the plant index. Then, together with the first lifted variable, say x_{rj}, we can also lift with coefficient 1 all the remaining variables in L that belong to the same subgraph X_r. After that, we can remove from L those x_{ij} such that $(i,r) \notin E$, since they would lift with coefficient 0, for all $j \geq 4$. To complete the lifting, we will continue taking nodes from L that will have lifting coefficient 1 and updating L by removing those that will have coefficient 0, until L becomes empty.

To conclude, after the analysis we can affirm that

$$x_{11} + x_{21} + x_{22} + x_{32} + x_{33} + x_{43} + x_{41} + \sum_{j=4}^{m} \Big(y_j + \sum_{i=1}^{4} x_{ij} \Big) + \sum_{\substack{i \notin I_H: \\ I_H \subseteq N_{G_I}(i)}} \gamma_i \sum_{j=4}^{m} x_{ij} \leq m, \tag{2.19}$$

where $\gamma_i \in \{0,1\}$ are calculated as above, is a facet of the conflict graph of SPLPI, G_{splpi}, when $H = (1,2,3,4)$ is a hole in G_I and $m > 3$.

c. Lifting of hole inequality (2.12)

Due to the similarity of (2.12) with inequality (2.11), this section is organized as the previous one. The following example illustrates application of Theorem 1.1 to an small example and is followed by description of the lifting in the general case by using this theorem and sequential lifting.

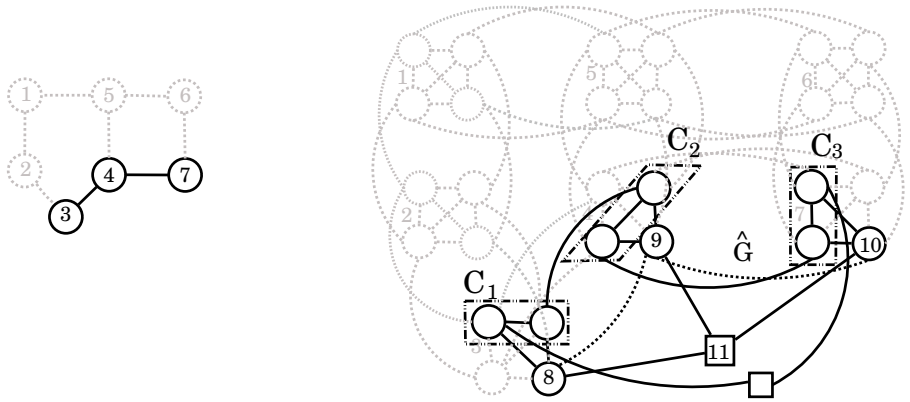


Figure 2.8: Application of Theorem 1.1 to hole of type (c) on Figure 2.4c

Example 2.6. Figure 2.8 shows the same example as Figure 2.4c. The figure depicts a hole in G_{splpi} on its right hand side, $H_{splpi} = (y_1, x_{31}, x_{32}, x_{42}, x_{43}, x_{73}, x_{71})$, which is produced by the path in G_I shown on the left. Nodes of H_{splpi} belonging to same group X_i are identified with complete subgraphs C_r of Theorem 1.1, namely $C_1 = \{x_{31}, x_{32}\}$, $C_2 = \{x_{42}, x_{43}\}$ and $C_3 = \{x_{73}, x_{71}\}$. On the other hand, $n+1, \ldots, n+q$ and $n+q+1$ in the theorem are x_{34}, x_{44}, x_{74} and y_4, respectively (q coincides with $\ell = 3$ and $n = 7$). Subgraph $\hat{G}$ is depicted with dashed black lines and is a path of length three. A packing in $\hat{G}$ is then made of $\hat{\alpha}_0 = 2$ nodes at most. Since $\beta_{\hat{\alpha}_0} \leq 1$ by definition, condition $\hat{\alpha}_0 \neq \beta_{\hat{\alpha}_0}$ of Theorem 1.1 stands and family of conditions (1.2) becomes empty. The theorem gives the following formula for a facet of the graph induced by nodes $1, \ldots, n+q, n+q+1$:

$$(\hat{\alpha}_0 - 1)\pi t + (1 - \beta_{\hat{\alpha}_0}) \sum_{i=1}^{q+1} t_{n+i} \leq \hat{\alpha}_0 - \beta_{\hat{\alpha}_0},$$

where $\pi t \leq 1$ is the hole inequality $\frac{1}{3}(y_1 + x_{31} + x_{32} + x_{42} + x_{43} + x_{73} + x_{71}) \leq 1$ and

$$\begin{aligned} \beta_{\hat{\alpha}_0} = \beta_2 = \max \quad & \{\max\{\tfrac{1}{3}(y_1 + x_{31} + x_{32} + x_{42} + x_{43} + x_{73} + x_{71}) : \\ & \{x_{ij} \in H_{splpi} : x_{ij} = 1\} \in \mathcal{P}_{H_{splpi}},\ x_{ij} = 0\ \forall i \in K, x_{ij} \in H_{splpi}\} \\ & : K \subseteq \{3,4,7\}, |K| = 2 \text{ and } \{x_{i4} = 1\}_{i \in K} \in \mathcal{P}_{\hat{G}}\}. \end{aligned}$$

To compute, β_2, one observes that the only maximum vertex packing in $\hat{G}$ is given by x_{34} and x_{74}. An extended maximum vertex packing containing x_{34} and x_{74} has two more nodes, for instance x_{42} and y_1. We conclude that $\beta_2 = 2/3$ and, applying Theorem 1.1, we obtain the facet

$$\begin{aligned} \frac{2-1}{3}(y_1 + x_{31} + x_{32} + x_{42} + x_{43} + x_{73} + x_{71}) + \\ \Big(1 - \frac{2}{3}\Big)\Big(y_4 + \sum_{i \in \{3,4,7\}} x_{i4}\Big) \leq 2 - \frac{2}{3}, \end{aligned}$$

which simplifies to

$$y_1 + x_{31} + x_{32} + x_{42} + x_{43} + x_{73} + x_{71} + y_4 + \sum_{i \in \{3,4,7\}} x_{i4} \leq 4.$$

△

As we did in Section 2.3.4, we start the lifting of inequalities of general form (2.12) by applying Theorem 1.1 $|J \setminus J_P|$ times. The following proposition states the facet obtained after the first application of Theorem 1.1 to (2.12).

Proposition 2.4. *Let* $\{y_{j_1}, x_{\ell j_1}, x_{\ell j_\ell}\} \cup \{x_{ij_i}, x_{ij_{i+1}}\}_{i=1}^{\ell-1}$ *induce an odd hole in* G_{splpi}, $J_P = \{j_i\}_{i=1}^{\ell}$. *For any plant* $\jmath \in J \setminus J_P$ *inequality*

$$y_{j_1} + \sum_{i=1}^{\ell-1}(x_{ij_i} + x_{ij_{i+1}}) + x_{\ell j_\ell} + x_{\ell j_1} + y_\jmath + \sum_{i=1}^{\ell} x_{i\jmath} \leq \ell + 1 \tag{2.20}$$

is a facet of the subgraph induced by $\{y_{j_1}, x_{\ell j_1}, x_{\ell j_\ell}\} \cup \{x_{ij_i}, x_{ij_{i+1}}\}_{i=1}^{\ell-1} \cup \{x_{i\jmath}\}_{i=1}^{\ell}$.

Proof. First of all, we have to identify graph G, facet $\pi t \leq 1$, cliques $C_1, \ldots, C_q$ and new nodes $x_{n+1}, \ldots, x_{n+q+1}$ of Theorem 1.1 in conflict graph G_{splpi}. We take as G the subgraph induced by $\{y_{j_1}, x_{\ell j_1}, x_{\ell j_\ell}\} \cup \{x_{ij_i}, x_{ij_{i+1}}\}_{i=1}^{\ell-1}$ and inequality (2.12), which is a facet of this subgraph. Let $q = \ell$, $C_i = \{x_{ij_i}, x_{ij_{i+1}}\}$, $i = 1, \ldots, \ell-1$ and $C_\ell = \{x_{\ell j_\ell}, x_{\ell j_1}\}$, $x_{n+i} = x_{i\jmath}$, $i = 1, \ldots, \ell$ and $x_{n+\ell+1} = y_\jmath$. Figure 2.8 illustrates this correspondence between G_{splpi} and Theorem 1.1 when $\ell = 3$. Now we calculate the coefficients of the theorem, $\hat{\alpha_0}$ and $\beta_{\hat{\alpha_0}}$, and check that the required conditions are met:

- Since $\{x_{i\jmath}\}_{i=1}^{\ell}$ induce a chordless path, the maximum cardinality of a set packing in the subgraph induced by $\{x_{i\jmath}\}_{i=1}^{\ell}$ is $\hat{\alpha_0} = \lceil \frac{\ell}{2} \rceil$.
- Suppose a set packing that includes $\lceil \frac{\ell}{2} \rceil$ nodes among $\{x_{i\jmath}\}_{i=1}^{\ell}$, which are alternately chosen from the path $\langle x_{1\jmath}, \ldots, x_{\ell\jmath} \rangle$, starting from $x_{1\jmath}$. For each $i = 1, \ldots, \ell$, if $x_{i\jmath}$ is not in the packing then x_{ij_i} can be included. Finally, y_{j_1} can also be in the packing, yielding the coefficient $\beta_{\hat{\alpha_0}} = \frac{\ell - \lceil \ell/2 \rceil + 1}{\ell}$. Condition $\hat{\alpha_0} \neq \beta_{\hat{\alpha_0}}$ holds since $\ell \geq 3$. Similarly, for $k = 2, \ldots, \lceil \frac{\ell}{2} \rceil - 1$, $\beta_k = \frac{\ell - k + 1}{\ell}$ and condition (1.2) is:

$$\frac{\ell - k + 1}{\ell} \leq 1 - \frac{(k-1)\Big(1 - \frac{\ell - \lceil \frac{\ell}{2} \rceil + 1}{\ell}\Big)}{\lceil \frac{\ell}{2} \rceil - 1},$$

  which is verified as equality.

Inequality (2.20) yields from Theorem 1.1, after simplifying coefficients. □

In a similar way as we showed in Section 2.3.4, it can be proved that

$$y_{j_1} + \sum_{i=1}^{\ell-1}(x_{ij_i} + x_{ij_{i+1}}) + x_{\ell j_\ell} + x_{\ell j_1} + \sum_{\jmath \notin J_P}(y_\jmath + \sum_{i=1}^{\ell} x_{i\jmath}) \leq \ell + m - |J_P| \tag{2.21}$$

is a facet of the induced subgraph $G_{splpi}\big[H_{splpi} \cup \big(\bigcup_{j \in J \setminus J_P}\{y_j, x_{1j}, \ldots, x_{\ell j}\}\big)\big]$. To complete the lifting of the initial inequality (2.12), we will distinguish three groups of variables as we did in Section 2.3.4. In this case, we will perform the analysis taking $\ell = 3$, even though a similar examination could be also done for the general case.

Assume that $H_{splpi} = (y_1, x_{11}, x_{12}, x_{22}, x_{23}, x_{33}, x_{31})$ where we have taken without loss of generality $P = \langle 1, 2, 3 \rangle$ the path in G_I, $I_P = \{1, 2, 3\}$ and $J_P = \{1, 2, 3\}$. By successive applications of Theorem 1.1, we have that

$$y_1 + x_{11} + x_{12} + x_{22} + x_{23} + x_{33} + x_{31} + \sum_{j=4}^{m}\Big(y_j + \sum_{i=1}^{3} x_{ij}\Big) \leq m \tag{2.22}$$

is a facet of $G_{splpi}\big[H_{splpi} \bigcup \big(\bigcup_{j=4}^{m}\{y_j, x_{1j}, \ldots, x_{\ell j}\}\big)\big]$. In the following, we calculate the coefficients of variables in the facet of G_{splpi} that is obtained from (2.22) by sequential lifting. Figure 2.4c can guide the reader through the analysis (nodes of the path $\langle 3,4,7\rangle$ should be renamed with $\langle 1,2,3\rangle$).

(i) We start with variables x_{ij} with $i \in I_P$ and $j \in J_P$ that have not been considered yet: x_{13}, x_{21} and x_{32}. Sets $\{x_{13}, x_{22}, x_{33}, y_1\}$ and $\{x_{32}, x_{23}, x_{12}, y_1\}$ are packings that can be completed with $y_4, \ldots, y_m$ and then x_{13} and x_{32} have lifting coefficient 0. On the other hand, a packing with x_{21} only admits two nodes of H_{splpi}, x_{12} and x_{33}. To complete the packing we can choose for each $j = 4, \ldots, m$ between including x_{2j} or y_j, so we can not get a packing of more than $m-1$ nodes. Then, x_{21} has lifting coefficient 1.

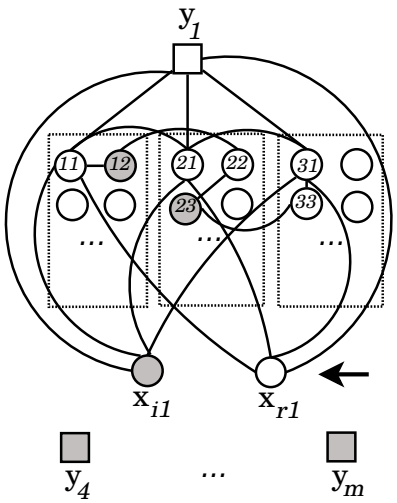
(ii) Variables y_2 and y_3 lift with coefficient 0, since they only have two neighbors in H_{splpi} each.

(iii) It remains to consider x-variables corresponding with clients that do not belong to the path, i.e. x_{ij} such that $i \notin I_P$ $j \in J$. If $j = 2$ or 3, x_{ij} has at most two neighbors in H_{splpi} and has coefficient 0. If $j = 1$ and there is a client $r \in \{1,2,3\}$ such that $(i,r) \notin E_I$, x_{i1} has lifting coefficient 0. For example, if $r = 2$ then $\{x_{i1}, x_{12}, x_{21}, x_{33}, y_4, \ldots, y_m\}$ is a packing containing x_{i1} plus m other nodes. The same is true when $j \geq 4$. Therefore, let us consider only those x_{ij} such that $j = 1$ or $j \geq 4$ and $(i,r) \in E_I$ for all $r \in I_P$. We first lift those nodes of the group with $j = 1$. The first one, say x_{i1}, will have coefficient 1. Consider now a second node of this group, x_{r1}, such that $(i,r) \notin E$. If we consider nodes with coefficient 1 that are not neighbors of x_{r1}, we can construct a packing of two nodes from H_{splpi}, x_{i1} and $y_4, \ldots, y_m$ (see Figure 2.9a). Note that this is not true for x_{rj} such that $(i,r) \notin E$ and $j \geq 4$. Figure 2.9b illustrates this fact. In conclusion, we will remove from the group x_{i1} (lifted with coefficient 1) and x_{r1} for all r such that $(i,r) \notin E$ (lifted with coefficient 0). We repeat the process until there are no more nodes in the group with $j = 1$. Consider now nodes of the group with $j \geq 4$. The procedure to lift these nodes is the same as in the analogous case of Section 2.3.4. Take a first client i such that $(i,r) \in E$ for all $r \in I_P$. All the variables x_{ij} of the group will lift with coefficient 1, for all $j \geq 4$. Again, if we consider x_{rj} in the group such that $(i,r) \notin E$, we can form a packing of m nodes that includes x_{ij} (see Figure 2.9c). Therefore, x_{rj} have lifting coefficient 0, for all $j \geq 4$. We repeat the process until we lift all the nodes of G_{splpi}. If we consider the same order of clients indices when we lift x-nodes with $j = 1$ as for the case $j \geq 4$, we can use a lifting coefficient γ_i that only depends on the client.

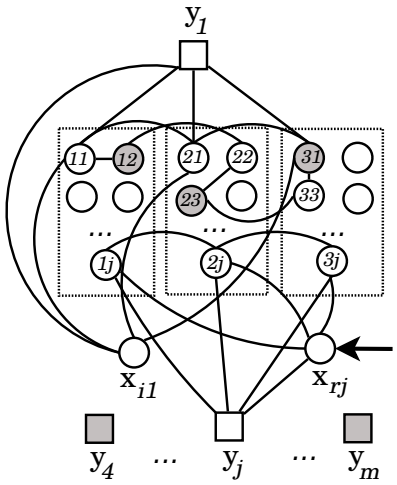
To conclude, after the analysis we can affirm that

$$\begin{aligned} y_1 + x_{11} + x_{12} + x_{22} + x_{23} + x_{33} + x_{31} \quad &+ \quad \sum_{j=4}^{m}\Big(y_j + \sum_{i=1}^{4} x_{ij}\Big) + \\ x_{21} \quad &+ \quad \sum_{\substack{i \notin I_P: \\ I_P \subseteq N_{G_I}(i)}} \gamma_i\Big(x_{i1} + \sum_{j=4}^{m} x_{ij}\Big) \leq m, \end{aligned} \tag{2.23}$$

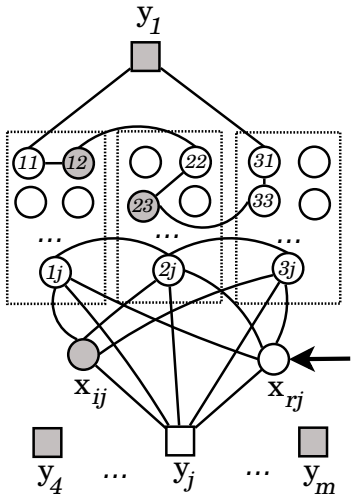
where $\gamma_i \in \{0,1\}$ are obtained as explained above, is a facet of the conflict graph of SPLPI, G_{splpi}, when $P = \langle 1,2,3\rangle$ is a non-closed path in G_I.



(a) Lifting of x_{r1} when x_{i1} has coefficient 1



(b) Lifting of x_{rj} when x_{i1} has coefficient 1



(c) Lifting of x_{rj} when x_{ij} has coefficient 1

Figure 2.9: Lifting of (2.22), case (iii), x_{ij}, $(i, s) \in E\ \forall s \in I_P$ and $j \neq 2, 3$

2.3.5 Separation of hole inequalities

To close the study of hole inequalities, we present a separation algorithm that treats the three types of holes explored. Our hole separation algorithm only considers the three holes of G_{splpi} described in Figure 2.4, which have lengths 5, 7 and 7 respectively. This algorithm uses a modification of a procedure that enumerates all chordless cycles of a graph, proposed in Dias et al. (2013). The key of this procedure is that it finds each chordless cycle only once.

Efficient enumeration of holes

Dias et al. (2013) proposed a labeling of the nodes of a graph that allows to define any cycle in a unique way. Namely, if $(v_1, v_2, \ldots, v_k)$ is a cycle in $G = (V, E)$, a labeling $b : V \to \{1, \ldots, n\}$ with the following properties identifies it uniquely:

(i) $b(v_2) = \min\{b(v_i) \ : \ i = 1, \ldots, k\}$,

(ii) $b(v_1) < b(v_3)$.

The fact that cycles are uniquely defined permits exclusion of duplicated holes. The authors then defined a triplet $\langle x, u, y\rangle$ as a sequence of nodes that can develop into a potential chordless cycle of length greater than three. They denoted by $T(G)$ the set of all triplets in G,

$$T(G) := \{\langle x, u, y\rangle \ : \ x, u, y \in V, x, y \in N(u), b(u) < b(x) < b(y) \text{ and } (x, y) \notin E\}.$$

They used what they called *degree labeling*, which is defined as follows. A sequence of the nodes of the graph $\{u_1, \ldots, u_n\}$ is constructed and the labeling is $b(u_i) = i$, for all $i = 1, \ldots, n$. Node u_i is a node of minimum degree in the induced subgraph $G_i := G[\{u_i, \ldots, u_n\}]$.

Their algorithm consists in taking the triplets in $T(G)$ and expanding them until a chordless cycle is completed or a chord in the current path is found. With the steps of the algorithm, triplets develop into chordless paths. Triplets expansion is implemented using a breadth-first search strategy. This means that all the neighbors of a node candidate for expansion are recursively checked before the next candidate is explored. Those neighbors that could form a chord with some node of the chordless path in construction are discarded during the search or *blocked*. To do so, an auxiliary vector counts the number of times that a node is found as a neighbor of some other in the chordless path in construction. If this counter is greater than one, the node is not considered for further inspection. The counter is incremented before each recursive call of the breadth-first search and decremented when the call returns. A pseudocode of the procedure of Dias et al. (2013) is given in Algorithm 2.2.

Our hole separation algorithm

Let us focus now on our version of the algorithm of Dias et al. applied to G_{splpi}. As we did in clique separation, we identify the solution of the current relaxation with weights of nodes of G_{splpi}, which we will denote by $w(\cdot)$. The objective is then to find an odd hole with total weight greater than the truncated half of its length. Our algorithm works with a subgraph of the weighted version of G_{splpi}. This subgraph is obtained after the elimination of those edges corresponding with constraints of the original SPLP and only preserves those of incompatibilities between clients. The nodes of the so-obtained input graph having non-zero degree are labeled using degree labeling.

The separation starts by finding all triplets of the input graph. Triplets $\langle x, u, y\rangle$ with total weight $w(x) + w(u) + w(y) \leq 1$ are discarded, since they will not lead to violated holes of

Algorithm 2.2 Efficient enumeration of chordless cycles, Dias et al. (2013)

global variables
  $G = (V, E)$, whose nodes are labeled with degree labeling
  $T(G)$, set of triplets of G
end global variables
procedure ChordlessCycles
  global variables
    T, set of expandable chordless paths
    C, set of chordless cycles of G
    $blocked(i)$, number of times i is found as a neighbor of some other in a chordless path
  end global variables
  $T = T(G)$
  $C = \{\langle x, u, y\rangle : x, u, y \in V, x, y \in N(u), b(u) < b(x) < b(y)$ and $(x, y) \in E\}$
  for all $i = 1, \ldots, n$ **do**
    $blocked(i) = 0$
  end for
  while $T \neq \emptyset$ **do**
    $p = \langle x, u, y\rangle \in T$
    $T = T \setminus \{p\}$
    BlockNeighbors(u)
    C=CC_Visit(p, C, $b(u)$, $blocked$)
    UnblockNeighbors(u)
  end while
  return C
end procedure
function CC_Visit($p = \langle u_1, u_2, \ldots, u_t\rangle$, C, key, $blocked$)
  BlockNeighbors(u_t)
  for all $v \in N(u_t)$ **do**
    if $b(v) > key$ and $blocked(v) = 1$ **then**
      $p' = \langle p, v\rangle$
      if $(v, u_1) \in E$ **then**
        $C = C \cup \{p'\}$
      else
        C =CC_Visit(p', C, key, $blocked$)
      end if
    end if
  end for
  UnblockNeighbors(u_t) **return** C
end function
function BlockNeighbors(i, $blocked$)
  for all $v \in N(i)$ **do**
    $blocked(i) = blocked(i) + 1$
  end for
end function
function UnblockNeighbors(i, $blocked$)
  for all $v \in N(i)$ **do**
    if $blocked(i) > 0$ **then**
      $blocked(i) = blocked(i) - 1$
    end if
  end for
end function

length 5 or 7, which are the only cases that we want to address. The remaining triplets are extended to build paths of up to 5 nodes. To extend a triplet or one of these paths in general, we consider the neighbors of its last node that do not form a chord. This extension stops if some of the following situations occur:

(i) A candidate to be included in a triplet $\langle x, u, y\rangle$ closes a chordless cycle of length 4. Note that this is equivalent to finding a hole of length 4 in G_I and that we are therefore in a situation like the one shown in Figure 2.4b. We perform the transformation that is shown in the same figure to obtain a hole of length 7 in G_{splpi}. There are several ways of doing such transformation and nodes will be chosen heuristically with the aim of total weight maximization. Then, the inequality associated with the resulting hole is lifted to a facet of form (2.20), which will be stored in case of violation.

(ii) A candidate to be included in a path $\langle x, u, y, v_4\rangle$ closes a chordless cycle of length 5. In this case, we have a hole of length 5 like illustrated by Figure 2.4a. We check the violation of the corresponding facet, which is obtained using the lifting of hole inequalities of length 5 described in 2.3.4.

(iii) A candidate to be included in a path $\langle x, u, y, v_4\rangle$ does not close a chordless cycle of length 5. In this case we just stop the extension without performing additional actions.

After exploring all the possible extensions of a triplet, we check whether its associated hole of length 7 (see Figure 2.4c) produces a violated facet (after the lifting procedure described in Section 2.3.4). Note that condition $w(x) + w(u) + w(y) > 1$ is necessary so that a found hole, $\langle x, u, y, v_1, v_2\rangle$ or $\langle x, u, y, v_1, v_2, v_3, v_4\rangle$, is violated, but not sufficient. Then, the algorithm can report violated facets coming from the lifting of non-violated hole inequalities.

2.4 Computational study

For validation of the different inequalities presented, we have carried out a computational study. This consists in testing a cut-and-branch procedure that incorporates the separation algorithms described in previous section. We use input data of different sizes and incompatibility graphs of different densities, in order to get representative results. In an initial configuration, the proposed separations called only at the root are tested against clique cuts of a commercial solver. Further experiments were conducted in a second phase, where separation was applied also during the branching and default cuts of the solver were not disabled.

2.4.1 The initial setup

The processor used in the study was an Intel core i7-6700k CPU at 4.0 GHz × 8 with 16 GB of RAM memory. The solver was Cplex v12.6.3 64-bit under operating system Linux Ubuntu 16.04. In our study, we handled 200 files from eleven different data sets, each of them containing from 10 to 30 testing files. These data files are instances of the SPLP taken from the repositories of the Max Planck Institute für Informatik and the Sobolev Institute of Mathematics. For each instance we have considered the same three incompatibility graphs that were randomly generated with densities 2%, 5% and 10%, yielding 600 instances of the SPLPI.

The general scheme of the cut-and-branch procedure that Cplex implements is as follows. At the beginning of the procedure, the linear relaxation of the problem is solved at the root node. The solver allows the user to specify algorithms to be used as *callbacks*. In our case,

implementations of clique and hole separations procedures described in previous sections are provided as argument for these callbacks. Callbacks are applied after solving the linear relaxation. If violated inequalities are incorporated to the problem as a result, the linear relaxation is solved again. After that, callbacks are invoke afresh in case there are more inequalities violated by the new solution. These steps are repeated until no violated inequalities are found. At this point, the branching starts. The user selects whether separation algorithms should be called also during the branching. Note that, even though callbacks are not invoke anymore, the relaxations solved in the branches are tighter due to the inequalities added at the root node.

Each instance in the testbed is solved under three different configurations:

(i) CPLEX cut-and-branch with clique cuts that CPLEX has implemented by default (associated results reported as $cplex_{cli}$ in tables).

(ii) CPLEX branch-and-bound with calls at the root node to the separation procedure for cliques presented in Section 2.3.2 (reported as *cli* in tables).

(iii) CPLEX branch-and-bound with calls at the root node to procedures for clique and hole separation, described in sections 2.3.2 and 2.3.5, respectively (reported as *c+h* in tables).

In all the experiments, the presolver and all cuts (except for clique cuts for case (i)) that CPLEX has incorporated are turned off. Cuts are added with option *purge*, i.e., any cut added will be purged later if considered ineffective. We set a time limit of 30 minutes to solve each instance.

Table 2.1 shows some relevant information of each data set. On the first two columns we see the name of the data sets and the number of instances of the SPLPI in each of them, respectively. Then, the number of instances that were solved to optimality within 30 minutes are depicted on a multicolumn. There are four columns in total, one for each configuration tested and column "all" . The latter represents the number of instances that were solved by the three procedures. Finally, the last column of Table 2.1 shows the number of plants and clients (that coincides) in each case. The last row of the table, which indicates the total number of instances that were solved, shows that the second configuration, *cli*, is the one that solves more instances to optimality. In total, $600(instances) \times 3(procedures) = 1800$ experiments were conducted.

2.4.2 Results and analysis

For a proper analysis, instances are classified into those that were solved by the three configurations tested and those that at least one configuration failed to solve within 30 minutes. Results are displayed in tables that, given the size of the testbed, necessarily display averages within the different data sets.

Table 2.2 shows a summary of the results obtained for those instances that were solved to optimality by the three procedures, i.e., they refer to those instances collected in the fourth column "# Solved" in Table 2.1. Rows in the table, i.e. data sets, are displayed in increasing order in terms of the LP gap. The first three columns of the table report the LP gap in percentage units, the optimal value of the integer program and the optimal value of its linear relaxation. Then, we show in a multicolumn the lower bound (i.e. the objective value of the last relaxation solved) just before the branching, i.e., after the inclusion of the cuts. Each of the three columns in "LB at the root" is dedicated to one of the procedures. The comparison of each one with column "LPOpt" gives an idea of the impact produced by the addition of the cuts in each case. Nevertheless, as already remarked, cuts have also an impact in the improvement of bounds of the rest of the nodes of the B&B tree that is not reported. The next two

Name	#	# Solved				$n = m$
		$cplex_{cli}$	cli	$c + h$	all	
pcodes	90	58	58	56	55	128
unif	90	90	90	90	90	100
D1	30	30	30	30	30	30
B	30	30	30	30	30	50
E1	30	30	30	30	30	50
C	30	30	30	30	30	50
D2	30	30	30	30	30	30
FPP	45	31	30	30	30	133
chess	90	83	90	90	83	144
gapB	45	26	30	29	25	100
gapA	45	29	32	32	27	100
gapC	45	0	0	0	0	100
Total	600	467	480	477	460	

Table 2.1: Information of each data set

multicolumns show the number of nodes that has the B&B tree and the time elapsed when the optimal solution is found. Note that the best result obtained over the three procedures in terms of bounds, nodes and time is emphasized in bold. After this sequence of three multicolumns, we find a fourth, which acts as summary of the previous ones. This multicolumn shows in which relative percentage the best procedure (*cli* or $c + h$) is better than the worse ($cplex_{cli}$) in terms of bounds, nodes of the B&B tree and running time. Finally, *Cuts final* indicates the number of cuts that are active when the optimal solution is found for each procedure.

Results summarized in Table 2.3 correspond with those instances that were not solved to optimality by some of the procedures after 30 minutes (Cplex termination status 11). Column # shows the total number of these instances for each data set. Next multicolumn indicates the concrete number of instances that were not solved by each of the three procedures. Then, we found the average relative gap after 30 minutes of execution in each case. Finally, the last multicolumn shows the number of times that the solution after 30 minutes is in fact the optimum even though the algorithm did not report it yet. For example, procedure $cplex_{cli}$ did not solve to optimality 32 instances of the data set *pcodes*. The average relative gap over those instances was 10.7% but 2 of the 32 solutions after the 30 minutes were in fact optimal (we know it by comparing with procedure *cli*, which solved those 2 instances). Procedures *cli* and *c+h* solved all instances in *chess*, so the corresponding slots in the last two multicolumns are empty.

We can state that the clique cuts that we have identified and implemented (*cli*) lead to improvement compared with those ones that are implemented by default in Cplex ($cplex_{cli}$). On the other hand, incorporating hole cuts to the solving procedure with our clique cuts (*c+h*), improves the simpler version *cli* sometimes, especially when the LP gap is large. In most cases the additional effort is not worth in terms of time, and *c+h* is slower than *cli*. When it comes to the number of nodes of the B&B tree, these two algorithms draw and regarding the lower bound *c+h* (certainly) gets the best one. In any case, the improvement obtained with respect to $cplex_{cli}$ is noticeable. For instance, for data set *FPP* our algorithms reduce the running time and the number of nodes in more than a half. The lower bound also improves dramatically, closing the gap with *IPOpt* by a third. Similarly, a strong performance is reported for data set

File	LPGap	IPOpt	LPOpt	LB at the root (rLB)			nodesBB (nBB)			Time			% Improvement			Cuts final		
				$cplex_{cli}$	cli	$c+h$	$cplex_{cli}$	cli	$c+h$	$cplex_{cli}$	cli	$c+h$	rLB	nBB	time	$cplex_{cli}$	cli	$c+h$
pcodes	9.1	53692	48224	51134	**51135**	**51135**	12394	11422	**11263**	544.0	**511.8**	524.0	0.003	9.1	5.9	84	85	85
unif	10.1	78172	70205	75206	75224	**75224**	4220	3490	**3390**	147.1	**121.1**	122.9	0.024	19.7	17.7	112	156	165
D1	13.2	15095	13071	14513	14518	**14519**	828	**607**	767	9.5	**7.8**	9.0	0.045	26.6	18.3	134	174	181
B	14.7	26176	22168	26048	26065	**26119**	2109	1409	**718**	76.3	57.9	**36.2**	0.270	65.9	52.5	260	435	468
E1	16.0	16147	13521	15307	15315	**15319**	9164	**6527**	7071	262.4	**185.9**	205.7	0.077	28.8	29.2	252	356	378
C	16.1	19440	16252	18548	18564	**18572**	7282	**4288**	4327	209.9	**140.7**	156.8	0.128	41.1	33.0	330	488	509
D2	18.4	21672	17601	20521	20539	**20543**	5613	4286	**3834**	69.3	**57.6**	58.4	0.110	31.7	16.9	265	353	368
FPP	20.6	42227	33515	33565	**36560**	**36560**	41650	**11106**	**11106**	1440.8	556.3	**555.9**	8.191	73.3	61.4	1	164	164
chess	21.8	62122	48250	59357	**59358**	**59358**	1788	**1599**	1673	174.2	**157.4**	176.0	0.001	10.6	9.7	251	252	381
gapB	23.5	45756	34976	37983	**38018**	**38018**	66343	54082	**52998**	842.3	656.7	**622.5**	0.090	20.1	26.1	110	130	131
gapA	30.6	38725	26837	30768	**30800**	**30800**	69885	60326	**59836**	925.9	765.5	**758.0**	0.105	14.4	18.1	146	162	162

Table 2.2: Summary of solved instances (time limit of 30 min)*

File	#	# Status=11			Gap after 30 min (%)			# Opt found		
		$cplex_{cli}$	cli	$c+h$	$cplex_{cli}$	cli	$c+h$	$cplex_{cli}$	cli	$c+h$
pcodes	35	32	32	34	10.7	10.9	10.7	2	2	4
FPP	31	31	30	30	22.1	21.6	21.5	0	0	0
chess	7	7	0	0	26.9	-	-	1	-	-
gapB	20	19	15	16	17.6	17.7	17.0	3	1	1
gapA	18	16	13	13	19.4	19.2	19.0	3	1	1
gapC	45	45	45	45	22.1	22.9	22.7	0	0	0

Table 2.3: Summary of unsolved instances (time limit of 30 min)*

* Results obtained with CPLEX cuts and presolver disabled, only $cplex_{cli}$ allows CPLEX clique cuts

B. The success of algorithms *cli* and *c+h* does not depend on the relative gap of the instances and satisfactory results are obtained also for data sets with largest gaps.

2.4.3 A few more tests

In this second part, we test our separation algorithms under two additional settings, namely when CPLEX cuts are activated and, on the other hand, within a branch and cut scheme. In this section we use 8 instances from each data set of Table 2.1, except for "FPP" and "gapC". We leave them out of the study because the previous computational experience shows little success when solving them — 14 out of 45 were solved in the case of "FPP", most of them for incompatibility density 2%, and none in the case of "gapC". Since this is a reduced study, we have collected in the testbed those instances that were likely to be solved, based on the previous experiments. As before, we use the same incompatibility graphs that were randomly generated with densities 2%, 5% and 10%. Therefore, we will solve 24 instances in total of each data set, which makes a testbed of 240 instances. We set a time limit of 30 minutes to solve each instance. The details of each of the two different settings tested are explained next.

In the first case, the aim is to demonstrate that our cuts are stronger than those identified by CPLEX. Therefore, we run a configuration, $cplex_{def}$, in which we do not disable the cuts that CPLEX uses by default. When we run our separation algorithms, we have to decide whether to let CPLEX add its own clique cuts during the branching or not. Turning CPLEX clique cuts off would make sense to avoid duplicated work. However, if we are to check that our clique cuts represent an improvement when they are used on top of those of CPLEX, we should turn them on. We decided to explore both alternatives. Table 2.4 shows averaged results. Each instance is solved under five configurations in total, which gives five columns for each of the following features: *LB at the root*, *nodesBB* and *Time*. As before, we write in bold those entries of the table corresponding with the best results among the five configurations. We have included a last column, $\%Imp.$, representing the relative reduction in time of the fastest configuration with respect to $cplex_{def}$. We observe that the largest reductions occur when clique cuts are disabled, and are generally due to *cli*. The table shows improvement with respect to CPLEX default settings for all the data sets. Therefore we can say that our inequalities are effective even when other cuts are present, and thus they add value to the default solving procedure. Table 2.5 shows information about those instances that were not solved by some of the configurations in this first experiment, being $c+h$ without CPLEX clique cuts the one that solves more instances. The gap after 30 minutes of execution is similar among the different configurations. Also, none of them obtains better gaps for all the data sets.

Lastly, we present some results about the performance of our inequalities within a branch and cut scheme. Instead of only adding the inequalities at the root, we add them at every node of the B&B tree until a depth limit, which we fix to 3. Regarding CPLEX cuts, we will activate all them (also clique cuts). We slightly modify our separation algorithm. Previously, if there were no violated inequalities found, the separation algorithm was not invoked anymore. Now, we do not keep track of whether the last search of violated inequalities has been successful, since it might have happened in a different branch of the B&B tree. Table 2.6 shows some results. Here, the lower bound is reported after processing the last node at depth 3, i.e., after all the cuts have been added in the case of the separation algorithms *cli* and $c+h$. Comparing these bounds with those of Table 2.4, we can see that the deeper we go into the B&B tree, the more significant is the improvement we get with respect to $cplex_{def}$. Naturally, adding the cuts has a cost in time, which is not always compensated by the improvement of the bound. This is the case of data sets "pcodes" and "chess". On the other hand, we also obtain an improvement

File	LPGap	IPOpt	LPOpt	LB at the root					nodesBB					Time					% Imp.
				all Cplex cuts on			all but cliques		all Cplex cuts on			all but cliques		all Cplex cuts on			all but cliques		
				$cplex_{def}$	cli	$c+h$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	cli	$c+h$	
unif	10.0	77621	69872	74935	**74970**	74960	**74970**	74960	2397	1834	1841	**1820**	1873	90.2	**66.6**	74.7	71.4	73.0	26.2
D1	13.4	15022	13010	14541	14553	**14555**	14554	**14555**	788	**607**	691	629	729	10.0	**8.3**	9.2	8.6	9.4	16.9
pcodes	14.2	56228	48227	52475	**52476**	**52476**	**52476**	**52476**	12534	11555	**11362**	12152	11772	567.5	557.9	**552.2**	556.0	571.7	2.7
B	14.6	25791	22038	25698	25696	**25722**	25696	**25722**	245	136	**122**	136	132	14.9	**12.1**	13.0	12.3	13.5	18.9
E1	16.5	16177	13510	15340	15348	**15349**	15348	**15349**	9201	7610	6884	7502	**6558**	244.8	224.7	192.5	221.5	**175.9**	28.1
C	16.6	19338	16127	18458	18489	**18491**	18489	**18491**	5127	3811	4139	**3535**	3841	154.4	121.3	136.0	**113.7**	131.5	26.3
D2	18.6	21793	17730	20665	20688	**20689**	20688	**20689**	4999	3511	3780	**3466**	3739	71.4	50.1	53.5	**50.0**	56.2	30.0
chess	22.5	62243	48247	59489	59491	**59493**	**59493**	59491	1810	1599	1687	1544	**1518**	177.0	171.9	193.5	**166.2**	179.1	6.1
gapB	23.6	45863	35056	38208	**38284**	**38284**	**38284**	**38284**	59595	61137	**57182**	60404	61621	757.9	716.6	**644.5**	686.2	702.6	15.0
gapA	31.0	38608	26623	**30672**	**30762**	**30762**	30761	30761	61703	47986	47899	**43788**	44691	849.6	622.3	634.0	**577.4**	590.9	32.0

Table 2.4: Summary of solved instances (time limit of 30 min) when Cplex cuts are activated*

File	#	# Status=11					Gap after 30 min (%)					# Opt found				
		all Cplex cuts on			all but cliques		all Cplex cuts on			all but cliques		all Cplex cuts on			all but cliques	
		$cplex_{def}$	cli	$c+h$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	cli	$c+h$
pcodes	8	8	8	8	8	8	10.3	10.8	9.8	10.3	10.8	0	0	0	0	0
gapA	13	11	8	7	6	6	20.4	19.5	19.6	18.7	19.3	6	2	1	1	0
gapB	7	7	6	6	5	4	15.6	17.5	15.8	16.4	16.9	2	0	2	1	0

Table 2.5: Summary of unsolved instances (time limit of 30 min) when Cplex cuts are activated*

*Results obtained with Cplex presolver off. Cuts are enabled (except clique cuts for columns "all but cliques")
$cplex_{def}$ stands for Cplex default configuration and cli and $c+h$ for Cplex with our separation algorithms

File	LPGap	IPOpt	LPOpt	LB at depth 3			nodesBB			Time			%Imp.
				$cplex_{def}$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	
unif	10.0	77621	69872	75984	**76117**	76093	2397	**1476**	1555	90.7	**77.0**	103.6	15.1
pcodes	13.0	55431	48226	52926	53013	**53109**	11529	**10735**	10850	**525.8**	609.0	556.3	-
D1	13.4	15022	13010	14764	14774	**14790**	788	580	**543**	10.0	**9.4**	10.8	6.6
B	15.4	26020	22021	25807	25852	**25883**	268	101	**71**	17.0	**15.1**	20.4	11.1
E1	16.5	16177	13510	15642	15694	**15697**	9201	5351	**5214**	244.7	**168.9**	176.6	31.0
C	16.6	19338	16127	18814	18905	**18916**	5127	2743	**2596**	153.9	**100.3**	110.7	34.8
D2	18.6	21793	17730	21181	**21306**	21290	4999	2586	**2252**	71.6	43.5	**41.6**	41.9
chess	22.5	62243	48247	60703	60804	**60866**	1810	**1700**	1754	**178.0**	219.8	304.8	-
gapB	23.3	45719	35055	39999	**40022**	39981	60438	**50855**	51383	735.9	**578.0**	599.0	21.5
gapA	31.0	38557	26619	33106	33217	**33245**	49499	40019	**39135**	762.4	**608.5**	613.4	20.2

Table 2.6: Summary of solved instances (time limit of 30 min) when all Cplex cuts are activated and cutting depth limit is 3*

File	#	# Status=11			Gap after 30 min (%)			# Opt found		
		$cplex_{def}$	cli	$c+h$	$cplex_{def}$	cli	$c+h$	$cplex_{def}$	cli	$c+h$
pcodes	10	8	9	10	7.8	6.9	7.8	0	1	2
gapA	14	12	4	6	15.9	12.8	15.5	7	1	2
gapB	8	7	5	4	11.5	12.5	11.7	2	2	1

Table 2.7: Summary of unsolved instances (time limit of 30 min) when all Cplex cuts are activated and cutting depth limit is 3*

*Results obtained with Cplex presolver off, $cplex_{def}$ stands for Cplex default configuration and cli and $c+h$ for Cplex with our separation algorithms applied until depth 3 of the B&B tree

of the running time for sets such as “E1”, ’C’ and “D2”, being this improvement even larger than when we only run the separation at the root (see Table 2.4). Finally, Table 2.7 gathers information about those instances that were not solved by some of the three configurations tested.

Chapter 3

The double-assignment plant location problem with twinning

In this chapter, a new variant of the Simple Plant Location Problem is proposed. We consider additional conditions in the classic location-allocation problem for clients and facilities. These new requirements are complementary to those proposed in the previous chapter in a sense. While certain pairs of clients could not be allocated to the same plant there, now some pairs have to be served by a common plant, that is, they have to be "twinned". The resulting problem can be addressed with existing models for the case of single assignment. However, the proposed setting when each client must be assigned to a couple of facilities is still unexplored.

The double-assignment plant location problem with twinning is interesting within the fundamentals of locational analysis and emerged as a natural sequel of Chapter 2. Also, we will see that this problem fits as a particular case of more general scenarios that were previously studied in locational analysis. Other than its interest within the framework of this thesis, double-assignment plant location with twinned clients finds interesting applications in telecommunication networks design. As already described, a generic telecommunication network consists of a set of terminals (users), connected to concentrators (switches or multiplexers) and a backbone network which interconnects the concentrators. A primary problem in network design is to decide how many concentrators are needed and how the terminals should be assigned to the concentrators. Gourdin et al. (2002) observed the relation of this problem with the SPLP, identifying those two decisions with that of facility location and allocation. The modification we propose here corresponds to a configuration in which some users must share a concentrator. This is a realistic assumption, since there could be users with special communication restrictions that want to have a dedicated path to avoid the backbone network.

We examine the implications of adding such "twinning" constraints to standard formulations of the SPLP with double assignment. We compare the resulting formulations from a theoretical point of view. After that, we focus on the study of one of the models, which turned out to be a set packing problem. All the clique facets are identified and a separation algorithm is devised. Although the separation problem is proved to be NP-hard, our computational experience shows that the separation algorithm is effective and efficient, reducing the computational times and the LP bounds for all the instances tested. Our experiments reflect the utility of clique inequalities and support the theoretical comparison of the formulations considered.

3.1 Double assignment with twinning

Consider a simple plant location scenario in which I and J are the sets of clients and candidate facilities. Suppose that some pairs of clients wish to be allocated to the same facility. We will call such pairs *twinned* clients. This could be easily addressed with formulations (SPLP) or (SPLPL$_{\leq}$) described in Concepts just by "merging" variables and costs of each twinned pair. However, sometimes clients must be assigned to a couple of facilities, for instance when a backup service is needed. In such a case, plant location can be modeled as a particular case of the Fault-Tolerant Facility Location problem (see Swamy & Shmoys, 2008), and of the Simple Plant and Warehouse Location problem (see Kaufman et al., 1977). A closely related topic is also hub location, where origin and destination pairs have to be connected by using a couple of hubs (see for instance Campbell et al., 2002). When double assignment is considered, satisfying those pairs of clients that want to be served by at least one common facility becomes a new combinatorial problem that is a variant of the classic SPLP.

The double-assignment plant location problem with twinning, DPLPT, consists of allocating each client to two facilities in such a way that twinned clients share at least one. To model twinning relationships, we consider a graph $G_T = (I, E_T)$ that has one node per client and an edge $e = (i, i') \in E_T$ for every pair (i, i') of twinned clients. Note that one client can be twinned more than once and that twinning is not transitive but reciprocal.

Double assignment can be typically modeled with two alternative approaches. One consists of using standard allocation and location variables and changing the one on the right-hand side of $\sum_{j\in J} x_{ij} = 1$ by a two. The second is to consider new allocation three-indexed variables to represent the pair of facilities that serve each client. The following subsections present both alternative strategies when applied to the DPLPT.

Two-indexed formulation

Consider standard allocation variables and costs, x_{ij} and c_{ij}, together with complementary location variables y_j and opening costs f_j. The formulation of the problem is then

$$
\begin{aligned}
\text{(DPLPT)}\ \min\ & \sum_{j\in J} f_j(1-y_j) + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} & & \\
\text{s.t.}\ & \sum_{j\in J} x_{ij} = 2 & \forall i \in I & \quad (3.1)\\
& x_{ij} + y_j \leq 1 & \forall i \in I, \forall j \in J & \quad (3.2)\\
& x_{i'j} + x_{i'j'} \geq x_{ij} + x_{ij'} - 1 & \forall (i,i') \in E_T, \forall j, j' \in J : j < j' & \quad (3.3)\\
& x_{ij}, y_j \in \{0,1\} & \forall i \in I, \forall j \in J. &
\end{aligned}
$$

Constraints (3.1) stand for double assignment, while (3.2) guarantee that clients are only allocated to open facilities. Constraints (3.3) are the twinning constraints. They are only active when their right hand sides equal one, that is, when a client i is allocated to facilities j and j' ($x_{ij} = x_{ij'} = 1$). In this case, the constraints ensure that i' is allocated to one of these two facilities, j or j', for every twinned client, $(i, i') \in E_T$.

Three-indexed formulation

Alternatively to standard allocation variables, we can use the following ones

$z_{ijk} = 1$ iff client i is served by facilities j and k, for all $i \in I$, $j, k \in J$ such that $j < k$.

With these variables and location y-variables, the problem can be formulated as a set packing in the following way

$$
\begin{aligned}
(\text{DPLPT}_3)\ \min\ & \sum_{j\in J} f_j(1-y_j) + \sum_{i\in I}\sum_{j\in J}\sum_{\substack{k\in J:\\ k>j}} (c_{ij}+c_{ik}-M)z_{ijk} && +Mn && \\
\text{s.t.}\ & \sum_{j}\sum_{k>j} z_{ijk} \le 1 && \forall i\in I && (3.4)\\
& \sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj} + y_j \le 1 && \forall i\in I, \forall j\in J && (3.5)\\
& z_{ijk} + \sum_{\ell} \sum_{\substack{t>\ell:\\ \{j,k\}\cap\{\ell,t\}=\emptyset}} z_{i'\ell t} \le 1 && \forall (i,i')\in E_T, j<k && (3.6)\\
& z_{ijk}, y_j \in \{0,1\} && \forall i\in I, \forall j,k\in J, &&
\end{aligned}
$$

where M is a large enough constant. Constraints (3.4) and (3.5) are typical assignment constraints in facility location. Twinning constraints (3.6) ensure that twinned clients $(i,i')\in E_T$ are not allocated to non-overlapping pairs of facilities. Instead of (3.6), simply imposing $z_{ijk}+z_{i'\ell t}$ for all $(i,i')\in E_T$ and different facility indices j,k,ℓ,t with $j<k$ and $\ell<t$ would have been sufficient. Inequalities (3.6) are just one way of reinforce that simplest translation of twinning requirements into packing constraints. Finally, note that $\sum_j\sum_{k>j} z_{ijk}=1$ for an optimal solution, because otherwise $M\gg 0$ is added to the objective value.

In the following section, we explore the relation between the family of x-variables and z-variables. Using that relation, we compare the constraints of formulations (DPLPT) and (DPLPT_3) and the optimal solutions of their linear relaxations, ultimately concluding which formulation gives better bounds.

3.2 Comparing the formulations

Variables x and z are clearly related to each other. Given a client i, z_{ijk} will be one if and only if x_{ij} and x_{ik} are. This is mathematically written as follows

$$
\begin{aligned}
z_{ijk} &= x_{ij}x_{ik} && \forall i\in I, j,k\in J,\ j<k\\
x_{ij} &= \sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj} && \forall i\in I, j\in J.
\end{aligned}
$$

We use the second formula, which is linear, to replace x in (DPLPT) by the corresponding z-variables. The resulting formulation, which we name (DPLPT'_3), will be written in terms of decision variables y and z. We will use (DPLPT'_3) to compare the objective values of the linear relaxations of (DPLPT) and (DPLPT_3).

Starting with (3.1), we get

$$
\sum_{j\in J} x_{ij} = \sum_{j\in J}\left(\sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj}\right) = \sum_{j\in J}\sum_{k>j} z_{ijk} + \sum_{j\in J}\sum_{k<j} z_{ikj}.
$$

Constraints (3.1) can be written then as $\sum_{j\in J}\sum_{k>j} z_{ijk} + \sum_{j\in J}\sum_{k<j} z_{ikj} = 2$, for all $i\in I$. Second, observe that (3.2) is (3.5) by substitution of x_{ij} as a function of z. We last substitute

in (3.3),

$$\begin{aligned} x_{i'j} + x_{i'j'} &\geq x_{ij} + x_{ij'} - 1 \quad \equiv \\ \sum_{k>j} z_{i'jk} + \sum_{k<j} z_{i'kj} + \sum_{k>j'} z_{i'j'k} + \sum_{k<j'} z_{i'kj'} &\geq \sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj} + \sum_{k>j'} z_{ij'k} + \sum_{k<j'} z_{ikj'} - 1 \equiv \end{aligned}$$

$$\sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj} + \sum_{k>j'} z_{ij'k} + \sum_{k<j'} z_{ikj'} - \sum_{k>j} z_{i'jk} - \sum_{k<j} z_{i'kj} - \sum_{k>j'} z_{i'j'k} - \sum_{k<j'} z_{i'kj'} \leq 1. \quad (3.7)$$

In order to compare (3.7) with (3.6), we unfold the summation in the latter. For every $(i, i') \in E_T$ and $j, k \in J$ such that $j < k$,

$$\sum_{\ell} \sum_{\substack{t>\ell: \\ \{j,k\}\cap\{\ell,t\}=\emptyset}} z_{i'\ell t} = 1 - \Big(\sum_{j'>j} z_{i'jj'} + \sum_{j'<j} z_{i'j'j} + \sum_{j'>k} z_{i'kj'} + \sum_{\substack{j'<k: \\ j'\neq j}} z_{i'j'k} \Big). \quad (3.8)$$

Let

$$Z_{i'jk} := \sum_{j'>j} z_{i'jj'} + \sum_{j'<j} z_{i'j'j} + \sum_{j'>k} z_{i'kj'} + \sum_{\substack{j'<k: \\ j'\neq j}} z_{i'j'k},$$

which will be one if i' is allocated to j, k or both and zero otherwise. Constraints (3.7) are then,

$$Z_{ijk} + z_{ijk} - Z_{i'jk} - z_{i'jk} \leq 1$$

while (3.6) can be rewritten as

$$z_{ijk} + 1 - Z_{i'jk} \leq 1. \quad (3.9)$$

We eventually obtain the following constraints and objective value for (DPLPT'$_3$):

$$\begin{aligned} \text{(DPLPT'}_3\text{)} \quad \min \quad & \sum_{j\in J} f_j(1-y_j) + \sum_{i\in I}\sum_{j\in J} c_{ij}\Big(\sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj}\Big) & & \\ \text{s.t.} \quad & \sum_{j\in J}\sum_{k>j} z_{ijk} + \sum_{j\in J}\sum_{k<j} z_{ikj} = 2 & \forall i \in I & \quad (3.10) \\ & \sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj} + y_j \leq 1 & \forall i \in I, \forall j \in J & \\ & Z_{ijk} + z_{ijk} - Z_{i'jk} - z_{i'jk} \leq 1 & \forall (i,i') \in E_T, j < k & \quad (3.11) \\ & z_{ijk}, y_j \in \{0,1\} & \forall i \in I, \forall j,k \in J, & \end{aligned}$$

The following proposition shows that the LP bound obtained with (DPLPT$_3$) is at least as good as that of (DPLPT). As we already now, having tight bounds is critical for a branch and cut solving approach.

Proposition 3.1. *The optimal value of the linear relaxation of (DPLPT) is smaller than or equal to that of (DPLPT$_3$).*

Proof. Suppose that $(\bar{z}, \bar{y})$ is an optimal solution of the linear relaxation of (DPLPT$_3$) and let $\bar{f}_3$ be its objective value,

$$\bar{f}_3 := \sum_{j\in J} f_j(1-\bar{y}_j) + \sum_{i\in I}\sum_{j\in J}\sum_{\substack{k\in J:\\ k>j}} (c_{ij}+c_{ik}-M)\bar{z}_{ijk} + Mn.$$

We define

$$\bar{x}_{ij} = \sum_{k>j} \bar{z}_{ijk} + \sum_{k<j} \bar{z}_{ikj} \qquad \forall i \in I, j \in J.$$

We will show that $(\bar{x}, \bar{y})$ is a feasible fractional solution of (DPLPT) with objective value $\bar{f}_3$. Since (DPLPT) is a minimization problem, this will prove the proposition.

Given that $(\bar{z}, \bar{y})$ is optimal, we know that $\sum_j \sum_{k>j} \bar{z}_{ijk} = 1$ for all i. Then, $(\bar{z}, \bar{y})$ satisfies (3.10). Moreover, since (3.9) is stronger than (3.11), $(\bar{z}, \bar{y})$ is a feasible solution of the linear relaxation of (DPLPT'$_3$). Since (DPLPT'$_3$) is formulation (DPLPT) when $x_{ij} = \sum_{k>j} z_{ijk} + \sum_{k<j} z_{ikj}$, we conclude that $(\bar{x}, \bar{y})$ is a feasible solution of the linear relaxation of (DPLPT).

The objective value of $(\bar{x}, \bar{y})$ is

$$\bar{f} := \sum_{j\in J} f_j(1-\bar{y}_j) + \sum_{i\in I}\sum_{j\in J} c_{ij}\bar{x}_{ij} = \sum_{j\in J} f_j(1-\bar{y}_j) + \sum_{i\in I}\sum_{j\in J} c_{ij}\Big(\sum_{k>j}\bar{z}_{ijk} + \sum_{k<j}\bar{z}_{ikj}\Big)$$

$$= \sum_{j\in J} f_j(1-\bar{y}_j) + \sum_{i\in I}\sum_{j\in J}\sum_{k>j} c_{ij}\bar{z}_{ijk} + \sum_{i\in I}\sum_{j\in J}\sum_{k<j} c_{ij}\bar{z}_{ikj}.$$

On the other hand, the fact that $\sum_j \sum_{k>j} \bar{z}_{ijk} = 1$ for all $i \in I$ implies

$$\bar{f}_3 = \sum_{j\in J} f_j(1-\bar{y}_j) + \sum_{i\in I}\sum_{j\in J}\sum_{\substack{k\in J:\\ k>j}} (c_{ij}+c_{ik})\bar{z}_{ijk}.$$

After rearranging the summations, we get $\bar{f} = \bar{f}_3$. □

Proposition 3.1 does not ensure that the LP bounds of (DPLPT) and (DPLPT$_3$) do not always coincide. The following example illustrates that the LP bound of (DPLPT$_3$) can be strictly greater than that of (DPLPT).

Example 3.1. Consider an instance with $I = J = \{1, \ldots, 6\}$, opening costs $f_j = 20$ for all $j \in J$ and allocation costs given by the following matrix

$$c = \begin{pmatrix} 0 & 30 & 76 & 25 & 18 & 48 \\ 30 & 0 & 45 & 50 & 14 & 38 \\ 76 & 45 & 0 & 37 & 28 & 53 \\ 25 & 50 & 37 & 0 & 60 & 15 \\ 18 & 14 & 28 & 60 & 0 & 27 \\ 48 & 38 & 53 & 15 & 27 & 0 \end{pmatrix}.$$

Figure 3.1 shows G_T for this example. The optimal solution of this instance is 272, with three facilities opened.

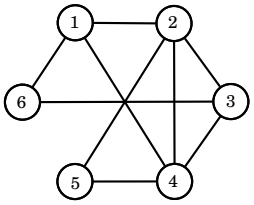


Figure 3.1: Graph G_T of Example 3.1

The optimal solution of the linear relaxation of (DPLPT) is

$$\begin{array}{llllll} x_{11}=0.5 & x_{14}=1 & x_{15}=0.5 & & & \\ x_{21}=0.5 & x_{22}=0.5 & x_{25}=0.5 & x_{26}=0.5 & & \\ x_{33}=1 & x_{34}=1 & & & & \\ x_{44}=1 & x_{46}=1 & & & & \\ x_{52}=0.5 & x_{55}=0.5 & x_{56}=1 & & & \\ x_{64}=1 & x_{66}=1 & & & & \\ y_1=0.5 & y_2=0.5 & y_3=1 & y_4=1 & y_5=0.5 & y_6=1, \end{array}$$

with optimal value 266.

The linear relaxation (DPLPT_3) has optimal solution

$$\begin{array}{llllll} z_{115}=0.6 & z_{145}=0.4 & & & & \\ z_{215}=0.6 & z_{225}=0.4 & & & & \\ z_{334}=0.4 & z_{335}=0.6 & & & & \\ z_{414}=0.2 & z_{436}=0.2 & z_{445}=0.4 & z_{446}=0.2 & & \\ z_{515}=0.4 & z_{525}=0.4 & z_{556}=0.2 & & & \\ z_{645}=0.4 & z_{646}=0.4 & z_{656}=0.2 & & & \\ y_1=0.6 & y_2=0.4 & y_3=1 & y_4=0.8 & y_5=1 & y_6=0.6, \end{array}$$

with optimal value 268.4.

△

3.3 Clique facets

In this section, we identify some facets of the integer polytope of (DPLPT_3),

$$\mathcal{B}_{dplpt} := \text{conv}\{(z,y) \in \{0,1\}^{n\cdot\frac{m(m-1)}{2}} \times \{0,1\}^m : \ (3.4)-(3.6)\}.$$

Like every set packing formulation, (DPLPT_3) has an associated conflict graph, which we name G_{dplpt}. As already mentioned in previous chapters, it is standard to identify facets of a set packing with some structures in its conflict graph. In this section, we will identify the cliques of G_{dplpt}.

The following example illustrates the structure of G_{dplpt} for a small instance of the problem. Given the density of the conflict graph, each family of constraints in (DPLPT_3), namely (3.4), (3.5) and (3.6), is illustrated separately for a restricted number of nodes.

Example 3.2. We consider an instance with five clients and four facilities. The graph describing the twinned pairs, G_T, is shown on Figure 3.2. Figures 3.3-3.5 illustrate constraints of (DPLPT_3) on G_{dplpt}. Circular nodes correspond with z-variables, and are tagged with proper

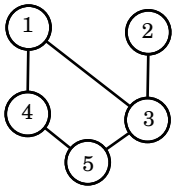


Figure 3.2: A graph G_T of five twinned clients, Example 3.2

subscripts ijk. Square nodes stand for y-variables. For a clear illustration, only a sample of the edges are shown. Circular nodes are arranged in groups forming a matrix, each group corresponding to a pair of facilities j, k with $j < k$ and having five nodes, one for each of the five clients.

Figure 3.3 depicts how edges link nodes that refer to the same client and corresponds to constraints (3.4) of (DPLPT$_3$). Note that they define cliques. In fact, Figure 3.3 shows a clique of six nodes associated to client 2.

Figure 3.4 illustrates that nodes corresponding with the same client and facility are adjacent to the node of that facility. Such subsets of nodes also define cliques in G_{dplpt}. The edges shown in this figure are due to constraints (3.5) in the model.

Finally, Figure 3.5 shows that the groups of nodes displayed in each entry (j, k) are also interconnected by means of the twinning constraints (3.6).

△

A first observation is that (3.6) can be generalized. Take the example depicted in Figure 3.5 and clients $i = 3$ and $i' = 5$, $(i, i') \in E_T$. If we add two new plants with indices 5 and 6, we will have twinning constraints

$$z_{312} + z_{534} + z_{535} + z_{536} + z_{545} + z_{546} + z_{556} \leq 1,$$

$$z_{313} + z_{524} + z_{525} + z_{526} + z_{545} + z_{546} + z_{556} \leq 1$$

for $(j, k) = (1, 2)$ and $(j, k) = (1, 3)$, respectively. However,

$$z_{312} + z_{313} + z_{323} + z_{545} + z_{546} + z_{556} \leq 1,$$

which is illustrated by Figure 3.6, is also valid and not written in the formulation. This corresponds to forbidding the pairs of facilities that serve twinned clients 3 and 5 to be in two non-overlapping subsets of plants $J_1, J_2 \subseteq J$, $J_1 \cap J_2 = \emptyset$. In the example above, $J_1 = \{1, 2, 3\}$ and $J_2 = \{4, 5, 6\}$, while (3.6) stand for the particular case of $J_1 = \{j, \ell\}$ and $J_2 = J \setminus \{j, \ell\}$.

Proposition 3.2. *Let $(i_1, i_2) \in E_T$ and $J_1, J_2 \subset J$ both containing at least two different facilities and such that $J_1 \cap J_2 = \emptyset$. The following inequalities are valid for (DPLPT$_3$)*

$$\sum_{j_1 \in J_1} \sum_{\substack{k_1 \in J_1: \\ k_1 > j_1}} z_{i_1 j_1 k_1} + \sum_{j_2 \in J_2} \sum_{\substack{k_2 \in J_2: \\ k_2 > j_2}} z_{i_2 j_2 k_2} \leq 1. \tag{3.12}$$

Moreover, (3.12) *define facets of $\mathcal{B}_{dplpt}$ if and only if $J_1 \cup J_2 = J$. In particular,* (3.6) *are facets.*

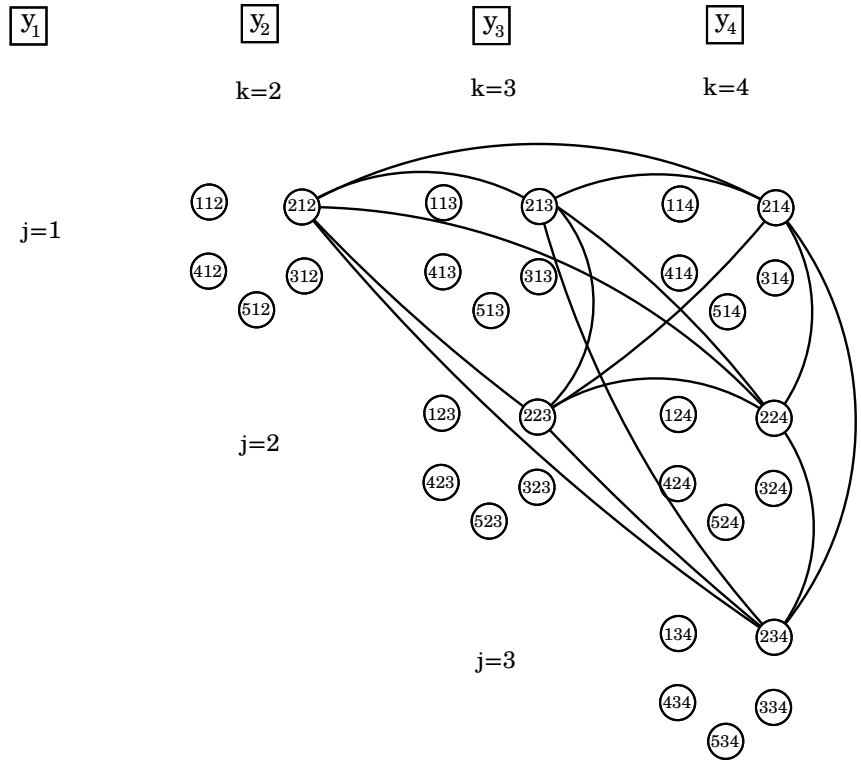


Figure 3.3: Edges in G_{dplpt} corresponding with (3.4) and $i = 2$

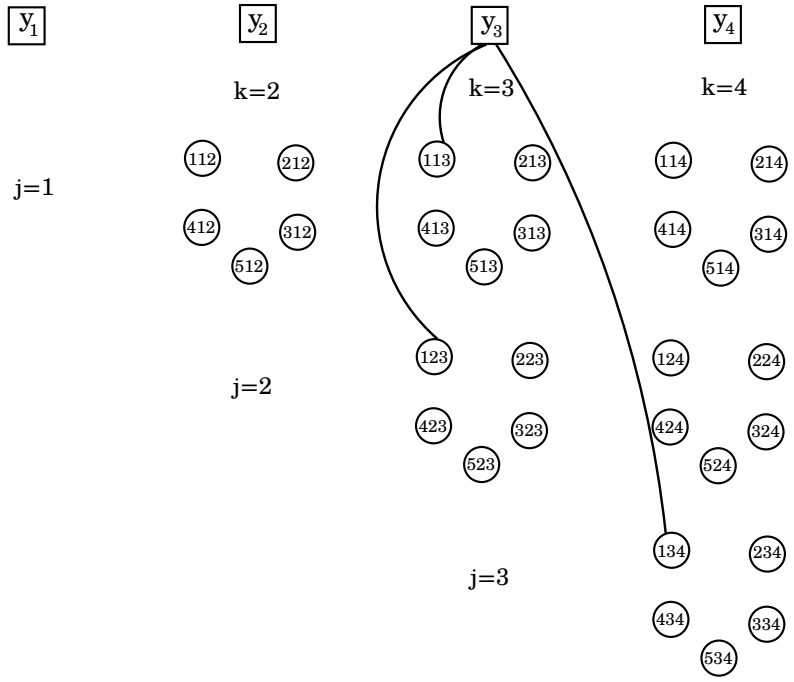


Figure 3.4: Edges in G_{dplpt} corresponding with (3.5) when $i = 1$ and $j = 3$

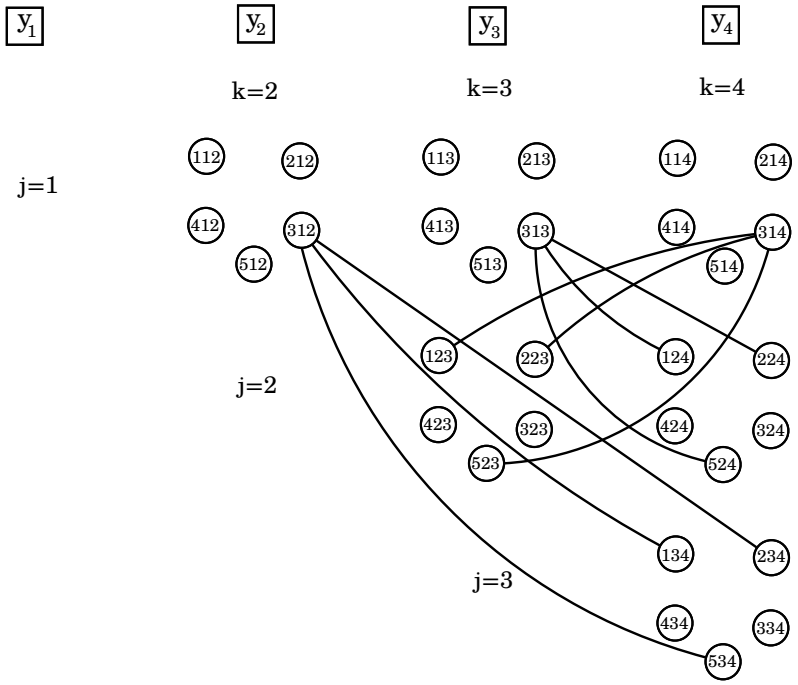


Figure 3.5: Edges in G_{dplpt} corresponding with (3.6), graph in Figure 3.2 and $i = 3$, $i' = 1, 2, 5$, $j = 1$, $k = 2, 3, 4$

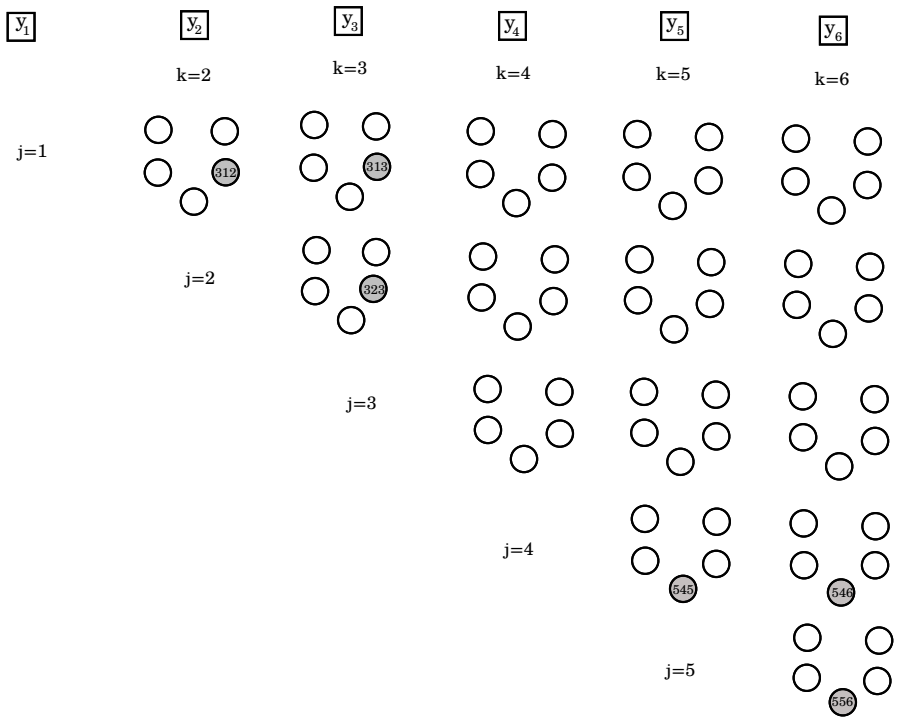


Figure 3.6: Clique in G_{dplpt} (3.12) with $(3, 5) \in E_T$, $J_1 = \{1, 2, 3\}$, $J_2 = \{4, 5, 6\}$

Proof. The fact that (3.12) are valid is clear from the problem definition. For the second statement, we will proof that

$$C := \Big(\bigcup_{j_1 \in J_1} \bigcup_{\substack{k_1 \in J_1: \\ j_1 > k_1}} \{z_{i_1 j_1 k_1}\} \Big) \bigcup \Big(\bigcup_{j_2 \in J_2} \bigcup_{\substack{k_2 \in J_2: \\ j_2 > k_2}} \{z_{i_2 j_2 k_2}\} \Big)$$

define a clique in G_{dplpt} if and only if $J_1 \cup J_2 = J$.

Suppose first that $G_{dplpt}[C]$ is a clique and there is $j' \in J \setminus \{J_1 \cup J_2\}$. In this case

$$\sum_{\substack{k_1 \in J_1: \\ j' > k_1}} z_{i_1 j' k_1} + \sum_{\substack{k_1 \in J_1: \\ j' < k_1}} z_{i_1 k_1 j'} + \sum_{j_1 \in J_1} \sum_{\substack{k_1 \in J_1: \\ j_1 > k_1}} z_{i_1 j_1 k_1} + \sum_{j_2 \in J_2} \sum_{\substack{k_2 \in J_2: \\ j_2 > k_2}} z_{i_2 j_2 k_2} \leq 1 \tag{3.13}$$

would be valid and stronger than (3.12), which is a contradiction.

Conversely, suppose that $J_1 \cup J_2 = J$. From the problem definition, we know that $G_{dplpt}[C]$ is a complete subgraph. Then, we have to prove that $G_{dplpt}[C]$ is a maximal complete subgraph. If there is $v \notin C$ such that $G_{dplpt}[C \cup \{v\}]$ is complete, then v does not correspond with a y-variable. This is clear, since a node y_j is adjacent only to z_{ijk} and z_{ikj} for all i and k. Since $J_1 \cap J_2 = \emptyset$, there cannot be a y-node adjacent to every node in C. Suppose then that v is a z-node, i.e., $G_{dplpt}[C \cup \{z_{i'j'k'}\}]$ is a complete subgraph, for some i', j', k'. We distinguish between two different cases.

1. If $i' = i_1$ (resp. $i' = i_2$), one of the facilities j' or k' is not in J_1 (resp. J_2), otherwise the node was already in C. Without loss of generality, let j' be that facility. Since $J_1 \cup J_2 = J$, $j' \in J_2$ (resp. J_1). But this is a contradiction because $z_{i_1 j' k'}$ and $z_{i_2 j' k_2} \in C$ are not adjacent (resp. $z_{i_2 j' k'}$ and $z_{i_1 j' k_1} \in C$), for any $k_2 \in J_2$, $k_2 > j'$ (resp. $k_1 \in J_1$, $k_1 > j'$).

2. If $i' \neq i_1$ and $i' \neq i_2$, edges between $z_{i'j'k'}$ and nodes in C will all correspond with twinning constraints. In particular, $\{i', i_1, i_2\}$ define a complete subgraph in G_T. Again, due to the fact that $J_1 \cup J_2 = J$, there cannot be such twinning constraints between $z_{i'j'k'}$ and $z_{i_1 j_1 k_1}$ if j' or k' are in J_1, or between $z_{i'j'k'}$ and $z_{i_2 j_2 k_2}$ if j' or k' are in J_2.

□

Following a similar idea, more valid inequalities can be derived for every triangle $\{i, i', i''\} \subseteq I$ in G_T and partition $\{J_1, J_2, J_3\}$ of J with J_i consisting of at least two facilities each. In general, this can be done for every subset of pairwise twinned clients and every partition having the same size. The following proposition states that all the clique facets of $\mathcal{B}_{dplpt}$ can be obtained in this way or are of the form (3.4) or (3.5).

Theorem 3.1. *The only clique facets of $\mathcal{B}_{dplpt}$ are* (3.4), (3.5) *and*

$$\sum_{i \in C_T} \sum_{j \in J_i} \sum_{\substack{k \in J_i: \\ k > j}} z_{ijk} \leq 1, \tag{3.14}$$

for all $C_T \subseteq I$ such that $G_T[C_T]$ is a complete subgraph, and for all $J_i \subseteq J$, $i \in C_T$, such that $|J_i| \geq 2$, $\cup_{i \in C_T} J_i = J$ and $J_i \cap J_{i'} = \emptyset$ for every pair $i \neq' i$.

Proof. Suppose that $\pi z + \rho y \leq 1$ is a clique facet of $\mathcal{B}_{dplpt}$. We will show that it has to be one of the inequalities of the statement.

We analyze first the case in which $\rho_{j'} = 1$ for some $j' \in J$, which readily implies $\rho_j = 0$ for all $j \neq j'$. Variables with positive π-coefficients have to be in the neighborhood of $y_{j'}$ in G_{dplpt}, $N(y_{j'})$. But $N(y_{j'}) = \{z_{ij'k} : i \in I, k > j'\} \cup \{z_{ikj'} : i \in I, k < j'\}$. In particular, z-variables in $N(y_{j'})$ all have one of its plant indices equal to j'. As a consequence, these variables are never adjacent due to a twinning constraint. Due to the problem constraints, they will be adjacent if and only if they have the same client index. In other words, there is a clique containing $y_{j'}$ for each $i \in I$, namely $C_{ij} = \{y_{j'}\} \cup \{z_{ij'k}\}_{j'<k} \cup \{z_{ikj'}\}_{k<j'}$, which corresponds to (3.5).

Suppose now that $\rho_j = 0$ for all $j \in J$. Let

$$C := \{z_{ijk} \ : \ \pi_{ijk} = 1, i \in I, j, k \in J, j < k\}$$

be the nodes of G_{dplpt} corresponding with the clique facet $\pi z \leq 1$. Let

$$C_T := \{i \in I \ : \ z_{ijk} \in C \text{ for some } j, k\}$$

be the subset of clients that appear in the indices of the variables of C. We analyze two cases.

1. $C_T = \{i'\}$. In this case, $C \subseteq \{z_{i'jk} \ : \ j, k \in J, j < k\}$. Due to constraints (3.4), it has to be $C = \{z_{i'jk} \ : \ j, k \in J, j < k\}$. In fact, clique facet $\pi z \leq 1$ corresponds precisely to (3.4).

2. $|C_T| > 1$. In this case, edges of $G_{dplpt}[C]$ between nodes associated to different clients have to be due to (3.6), which are defined for every $(i, i') \in E_T$. As a consequence, $G_T[C_T]$ is also a complete subgraph. For every $i \in C_T$, let $J_i^1 := \{j \in J \ : \ z_{ijk} \in C \text{ for some } k\}$ and $J_i^2 := \{k \in J \ : \ z_{ijk} \in C \text{ for some } j\}$. We also define $J_i := J_i^1 \cup J_i^2$, the set of facilities indices that appear with client i in the variables of C. First, $J_i \cap J_{i'} = \emptyset$ for all $i, i' \in C_T$ since $\pi z \leq 1$ will not be valid otherwise. On the other hand, for all $i \in C_T$, $\{(j, k) \in J \times J \ : \ z_{ijk} \in C\} \subseteq \{(j, k) \in J_i \times J_i \ : \ j < k\}$. If, for some $i' \in C_T$, there was (j', k') contained in the second subset but not in the first, $\pi z \leq 1$ could be improved by adding $z_{i'j'k'}$ to its left-hand side, which contradicts the fact that it is a facet. Hence, $\{(j, k) \in J \times J \ : \ z_{ijk} \in C\} = \{(j, k) \in J_i \times J_i \ : \ j < k\}$. It follows that $\pi z \leq 1$ has the form of (3.14). It remains to prove that $\cup_{i \in C_T} J_i = J$. But now it is easy to see that $\pi z \leq 1$ could be improved if there was $j' \in J \setminus \left(\cup_{i \in C_T} J_i \right)$, just by adding j' to one of the subsets J_i.

□

Corollary 3.1. *All the constraints of (DPLPT$_3$) are facets.*

Remark 3.1. In Theorem 3.1, C_T is not necessarily a clique. Indeed, (3.12) are a particular case of (3.14) when $C_T = \{i_1, i_2\}$ for $(i_1, i_2) \in E_T$, which is not necessarily a maximal complete subgraph. This is interesting because, to the best of our knowledge, there is no precedence in set packing of a clique facet ultimately induced by a not maximal complete subgraph. △

3.4 Clique separation

The only clique facets of $\mathcal{B}_{dplpt}$ that are not included in (DPLPT$_3$) are that of family (3.14) with $|C_T| > 2$ or $|C_T| = 2$ and $|J_i| \geq 2$, $i \in C_T$. This section focuses on the latter, which belong to family (3.12). Given a twinned pair $(i_1, i_2) \in E_T$, (3.12) define exponentially many inequalities, even when $J_1 \cup J_2 = J$. A separation algorithm is then needed to manage these

inequalities within a branch and cut scheme. We will first prove the theoretical computational complexity of this problem and then provide a heuristic algorithm to approach it. These results can be extended to separation of the more general family (3.14). On the one hand, the theoretical complexity does not improve when the size of the partition (which coincides with $|C_T|$) increases. On the other hand, we will see that the proposed heuristic can be easily adapted. However, exhaustive separation of (3.14) is not expected to be rewarding. We will prefer to address the case $|C_T| = 2$ when moving separation from theory to practice (see the computational study of next section).

Given a fractional optimal solution $\bar{z}$, and given an edge $(i_1, i_2) \in E_T$, the *problem of separation* of (3.12) is to find the partition of J in J_1 and J_2 that maximizes

$$\sum_{j_1 \in J_1} \sum_{\substack{k_1 \in J_1: \\ j_1 > k_1}} \bar{z}_{i_1 j_1 k_1} + \sum_{j_2 \in J_2} \sum_{\substack{k_2 \in J_2: \\ j_2 > k_2}} \bar{z}_{i_2 j_2 k_2}. \tag{3.15}$$

This problem can be identified with a max-cut problem in a graph. Given a graph $G = (V, E)$ with weights on its edges, the max-cut is to determine a subset $U \subseteq V$ such that the sum of the weights in the cut— the set of edges with one endnode in U and the other in $V \setminus U$— is maximal. We identify the separation with the following especial case of max-cut, which we name *double-cut*.

Definition 3.1 (*DOUBLE-CUT*)**.** Let $G = (V, E)$ be a graph and let W^1 and W^2 be two matrices of weights for its edges, $W^1 = (w^1_{jk})_{i,k \in V}$, $W^2 = (w^2_{jk})_{i,k \in V}$, $w^1_{jk}, w^2_{jk} \in \mathbb{R}$. Given $W \in \mathbb{R}$, DOUBLE-CUT is to decide whether there is $V_1 \subseteq V$ such that $\sum_{j \in V_1} \sum_{\substack{k \in V_1: \\ k > j}} w^1_{jk} + \sum_{j \notin V_1} \sum_{\substack{k \notin V_1: \\ k > j}} w^2_{jk} \geq W$. △

We will show that the following problem can be reduced to the decision problem of Definition 3.1 in polynomial time.

Definition 3.2 (*PARTITION*)**.** Let $c_1, \ldots, c_n$ be integer, $c_i \in \mathbb{Z}$. PARTITION is to decide whether there is $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$. △

The partition problem is one of the six basic NP-complete problems in Garey & Johnson (1979). Then, giving a polynomial time reduction of PARTITION to DOUBLE-CUT implies that the latter is also NP-complete. The following proposition states and proves such a result.

Proposition 3.3. *DOUBLE-CUT is NP-complete.*

Proof. Suppose that $c_1, \ldots, c_n \in \mathbb{Z}$ is an instance of PARTITION. We will build an instance of DOUBLE-CUT that has positive answer if an only if there is $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$.

Consider an instance of DOUBLE-CUT with $G = K_n$ the complete graph of n nodes and the following weights

$$w^1_{jk} = w^2_{jk} = -c_j \cdot c_k \qquad \forall j, k \in V,$$

$$W = \frac{1}{2} \sum_{j \in V} c_j^2 - \frac{1}{4} \Big(\sum_{j \in V} c_j \Big)^2.$$

DOUBLE-CUT has a positive answer if and only if there is $V_1 \subseteq V$ such that

$$\sum_{j\in V_1}\sum_{\substack{k\in V_1:\\ k>j}} -c_j\cdot c_k + \sum_{j\notin V_1}\sum_{\substack{k\notin V_1:\\ k>j}} -c_j\cdot c_k \geq \frac{1}{2}\sum_{j\in V} c_j^2 - \frac{1}{4}\Big(\sum_{j\in V} c_j\Big)^2. \tag{3.16}$$

We know that

$$\sum_{j\in V_1}\sum_{\substack{k\in V_1:\\ k>j}} -c_j\cdot c_k = -\frac{1}{2}\Big[\big(\sum_{j\in V_1} c_j\big)\cdot\big(\sum_{k\in V_1} c_k\big) - \sum_{j\in V_1} c_j^2\Big].$$

The left-hand side of (3.16) is then

$$-\frac{1}{2}\Big[\big(\sum_{j\in V_1} c_j\big)\cdot\big(\sum_{k\in V_1} c_k\big) + \big(\sum_{j\notin V_1} c_j\big)\cdot\big(\sum_{k\notin V_1} c_k\big) - \sum_{j\in V} c_j^2\Big]$$

$$= \frac{1}{2}\Big[\sum_{j\in V} c_j^2 - \Big(\sum_{j\in V_1} c_j\Big)^2 - \Big(\sum_{j\notin V_1} c_j\Big)^2\Big].$$

In general, function X^2+Y^2 when $X+Y$ is constant attains its minimum at $X=Y$. If we call

$$X = \sum_{j\in V_1} c_j \text{ and}$$
$$Y = \sum_{j\notin V_1} c_j,$$

the left-hand side of (3.16) attains its maximum when X^2+Y^2 attains its minimum, i.e., when $\sum_{j\in V_1} c_j = \sum_{j\notin V_1} c_j = \big(\sum_{j\in V} c_j\big)/2$. In such a case, the left-hand side of (3.16) would be

$$\frac{1}{2}\left[\sum_{j\in V} c_j^2 - 2\cdot\left(\frac{\sum_{j\in V} c_j}{2}\right)^2\right] = \frac{1}{2}\sum_{j\in V} c_j^2 - \frac{1}{4}\Big(\sum_{j\in V} c_j\Big)^2 = W.$$

Then, DOUBLE-CUT can only have positive answer when (3.16) is satisfied as an equality, which happens if and only if there is $V_1 \subseteq V$ such that $\sum_{j\in V_1} c_j = \sum_{j\notin V_1} c_j$. □

Corollary 3.2. *The separation of* (3.12) *when* $J_1 \cup J_2 = J$ *is NP-hard.*

Given the computational complexity of separation, we propose an algorithm that, given an optimal fractional solution $\bar{z}$ and a pair of twinned clients $(i_1, i_2) \in E_T$, tries to maximize (3.15) heuristically. An initial partition $J_1, J_2 \subseteq J$ is considered at first. Facilities are moved from J_1 to J_2 and viceversa whenever this produces an increment of (3.15) and the number of elements is not below three for any of the subsets. If (3.15) is eventually greater than one, the corresponding inequality is added as a cut. Algorithm 3.1 describes the pseudo code of the separation heuristic, which is applied for every pair in E_T in each callback. Note that the idea of the algorithm can be also applied to i_1, i_2, i_3 pairwise adjacent, i.e., inducing a triangle in G_T. Partition will be then made of three subsets, $J_1, J_2, J_3 \subseteq J$, and a larger combinatorial number of possible movements among their components have to be checked.

Algorithm 3.1 Separation heuristic

Input $\bar{z}$: fractional solution of (DPLPT$_3$).
$(i_1, i_2) \in E_T$: pair of twinned clients.
ϵ: tolerance for cut violation.

Variables J_1: first subset of facility indices.
J_2: second subset of facility indices.
$Z_{i_1 j}(J_1)$: the contribution of j to the left-hand side of the cut when $j \in J_1$.
$Z_{i_2 j}(J_2)$: the contribution of j to the left-hand side of the cut when $j \in J_2$.

Output *cut*: a cut in the family (3.12) violated by $\bar{z}$, if exists.

Step 1 $J_1 = \{1, \ldots, \lfloor (m+1)/2 \rfloor\}$, $J_2 = \{\lfloor (m+1)/2 \rfloor + 1, \ldots, m\}$.

Step 2 If $|J_1| \geq 3$, for all $j \in J_1$ do:
2.1 $Z_{i_1 j}(J_1) = \sum_{\substack{k_1 \in J_1: \\ k_1 < j}} \bar{z}_{i_1 k_1 j} + \sum_{\substack{k_1 \in J_1: \\ k_1 > j}} \bar{z}_{i_1 j k_1}$.
2.2 $Z_{i_2 j}(J_2) = \sum_{\substack{k_2 \in J_2: \\ k_2 < j}} \bar{z}_{i_2 k_2 j} + \sum_{\substack{k_2 \in J_2: \\ k_2 > j}} \bar{z}_{i_2 j k_2}$.
2.3 If $Z_{i_1 j}(J_1) < Z_{i_2 j}(J_2)$ then $J_1 = J_1 \setminus \{j\}$; $J_2 = J_2 \cup \{j\}$; go to Step 2.

Step 3 If $|J_2| \geq 3$, for all $j \in J_2$ do:
3.1 $Z_{i_1 j}(J_1) = \sum_{\substack{k_1 \in J_1: \\ k_1 < j}} \bar{z}_{i_1 k_1 j} + \sum_{\substack{k_1 \in J_1: \\ k_1 > j}} \bar{z}_{i_1 j k_1}$.
3.2 $Z_{i_2 j}(J_2) = \sum_{\substack{k_2 \in J_2: \\ k_2 < j}} \bar{z}_{i_2 k_2 j} + \sum_{\substack{k_2 \in J_2: \\ k_2 > j}} \bar{z}_{i_2 j k_2}$.
3.3 If $Z_{i_1 j}(J_1) > Z_{i_2 j}(J_2)$ then $J_1 = J_1 \cup \{j\}$; $J_2 = J_2 \setminus \{j\}$; go to Step 2.

Step 4 If $\sum_{j_1 \in J_1} \sum_{\substack{k_1 \in J_1: \\ j_1 > k_1}} \bar{z}_{i_1 j_1 k_1} + \sum_{j_2 \in J_2} \sum_{\substack{k_2 \in J_2: \\ j_2 > k_2}} \bar{z}_{i_2 j_2 k_2} > 1 + \epsilon$ then
$cut = \sum_{j_1 \in J_1} \sum_{\substack{k_1 \in J_1: \\ j_1 > k_1}} z_{i_1 j_1 k_1} + \sum_{j_2 \in J_2} \sum_{\substack{k_2 \in J_2: \\ j_2 > k_2}} z_{i_2 j_2 k_2} \leq 1$
else $cut = \emptyset$

Step 5 Return *cut*.

3.5 Computational study

The aim of our computational study is twofold. First, we are interested in providing an empirical comparison between the LP bounds of (DPLPT) and (DPLPT$_3$). Second, we intend to validate the separation heuristic applied on (DPLPT$_3$) and test its relative performance with respect to bare formulations (DPLPT) and (DPLPT$_3$).

3.5.1 Experimental setup

The machine used was an Intel Xeon CPU at 3Ghz×8, with 16GB of RAM. Our testbed was based on four instances of the OR-Library, namely pmed1, pmed2, pmed3 and pmed4, see Beasley (1990).

As already mentioned in previous section, the heuristic described on Algorithm 3.1, which address the case $|C_T| = 2$, can be adapted to separate other inequalities of the family (3.14). A preliminary computational study was conducted to check the performance of the heuristic when adapted to $|C_T| \in \{2, 3\}$. Graph G_T was generated having many triangles, so that structures $|C_T| = 3$ were found. For every $i \in \{1, \ldots, \lfloor n/2 \rfloor\}$, E_T had edges $(i, n-i+1)$, $(i, 2i)$, $(2i, n-i+1)$, $(n-i, n-i+1)$ and $(i, n-i)$. The results are shown on Table 3.2. Dual bounds are the same in almost all the cases, except for those depicted in bold. Running times are not always smaller for one of the alternatives. However, we observe that the improvement of the adapted heuristic is not significant in comparison to those cases when it represents the worst alternative. For instance, when $n = 3$ in file pmed3, the adapted heuristic spends more than 100 seconds more than Algorithm 3.1. Conversely, the adapted heuristic saves no more than 30 seconds with respect to Algorithm 3.1 in the best cases. In view of these results, we decided to conduct experiments only with the heuristic of Algorithm 3.1.

For our computational study, we have taken $n = m =$ 10, 15, 20, 25, 30, 35, 40. Since Beasley's instances have 100 clients, we have to trim Beasley's matrices (c_{ij}) to obtain the desired size. A range of constant vectors are considered for the opening costs, for all $j \in J$, $f_j =$ 50, 75, 100, 125, 150, 175, 200. These costs will imply opening from 12.5% to 68% of candidate facilities in the optimal solution. The testbed is then made of 4(Beasley's)×7(size)×7(opening costs)=196 instances in total. Graph G_T is generated depending on the size of the instance, i.e., on n. For every $i \in \{1, \ldots, \lfloor n/2 \rfloor\}$ edges $(i, n-i+1)$ and $(i, 2i)$ are added to E_T, which gives medium density graphs.

Each instance of the testbed is solved under four different configurations in Fico Xpress Mosel 64-bit v4.8.4. Default cutting planes, heuristics and presolving strategies are disabled for all the experiments. The configurations are

- C_1: formulation (DPLPT),
- C_2: formulation (DPLPT$_3$),
- C_3: formulation (DPLPT$_3$) and callbacks to Algorithm 3.1 at the root node of the branching tree,
- C_4: formulation (DPLPT$_3$) and callbacks to Algorithm 3.1 throughout the branching tree.

Time limit is set to half an hour for all the experiments. The two last configurations solved all the instances within the time limit. Formulations (DPLPT) and (DPLPT$_3$) solved 122 and 110 instances, respectively.

File	n	f_j	OPT	LP		Time		NodesBB	
				$\|C_T\| = 2$	$\|C_T\| \in \{2,3\}$	$\|C_T\| = 2$	$\|C_T\| \in \{2,3\}$	$\|C_T\| = 2$	$\|C_T\| \in \{2,3\}$
pmed1	10	50	921	921.0	921.0	0.1	0.3	1	1
pmed1	10	100	1161	1161.0	1161.0	0.1	0.1	1	1
pmed1	20	50	2272	2272.0	2272.0	6.3	5.4	1	1
pmed1	20	100	2688	2688.0	2688.0	5.7	4.7	1	1
pmed1	30	50	3410	3410.0	3410.0	139.9	122.0	1	1
pmed1	30	100	3943	3943.0	3943.0	183.7	149.2	1	1
pmed2	10	50	1340	1340.0	1340.0	0.2	0.1	1	1
pmed2	10	100	1590	1590.0	1590.0	0.2	0.1	1	1
pmed2	20	50	2717	2717.0	2717.0	6.3	4.5	1	1
pmed2	20	100	3175	3175.0	3175.0	7.6	5.1	1	1
pmed2	30	50	3969	3969.0	3969.0	110.6	125.1	1	1
pmed2	30	100	4499	4499.0	4499.0	131.4	126.1	1	1
pmed3	10	50	976	976.0	976.0	0.1	0.2	1	2
pmed3	10	100	1209	1209.0	1209.0	0.1	0.1	1	1
pmed3	20	50	2595	2588.0	2588.0	16.5	20.8	5	21
pmed3	20	100	2983	**2975.2**	**2979.3**	15.4	14.4	13	3
pmed3	30	50	4108	4108.0	4108.0	295.3	379.8	1	1
pmed3	30	100	4687	**4672.4**	**4673.0**	332.4	483.4	21	11
pmed4	10	50	1442	1442.0	1442.0	0.1	0.1	1	1
pmed4	10	100	1701	1701.0	1701.0	0.1	0.1	1	1
pmed4	20	50	3165	**3163.0**	**3165.0**	11.3	11.2	1	1
pmed4	20	100	3627	3627.0	3627.0	10.9	11.7	1	1
pmed4	30	50	4491	4491.0	4491.0	183.5	168.4	1	1
pmed4	30	100	5019	5019.0	5019.0	140.4	154.8	1	1

Table 3.2: Preliminary test to check the performance when adapting the heuristic

Tables 3.3 and 3.4 show the computational results. The LP gap, running time and the number of nodes explored during the branching are displayed in different columns for the different configurations. The LP gap is relative to the optimal value, and is calculated as a percentage. When Algorithm 3.1 is used, the LP gap is calculated using the LP bound after adding the cuts at the root node. Due to the large number of instances, we have to display average results. Table 3.3 shows results on average throughout different costs f_j considered, for every Beasley's file and size. On the other hand, computational results are averaged by input size n on Table 3.4, where a row is displayed for every Beasley's file and opening cost constant vector. The following section discusses the results.

3.5.2 Comparative analysis

From Table 3.3, we can see that (DPLPT) and (DPLPT$_3$) fail to solve the bigger instances, namely those with $n = 35$, 40. Moreover, (DPLPT$_3$) still has difficulties to cope when $n = 30$. Using the separation heuristic with (DPLPT$_3$) can dramatically improve the running time to the point of solving all the instances within 538 seconds. Such an improvement with respect to (DPLPT) and (DPLPT$_3$) begins to appear with $n = 20$. Columns concerning the nodes of the branch and bound tree also reflect a better performance when the separation is incorporated to the solving procedure. Nevertheless, as opposed to running times, (DPLPT$_3$) explores less nodes than (DPLPT) to obtain the optimal solution. This makes very much sense since (DPLPT$_3$) LP bounds are better. The number of variables to manage is probably one of the factors that influence higher running times of (DPLPT$_3$). The fact that many of the instances are solved at the root when cuts are added is consistent with the extremely tight LP gaps that we get in those cases. On the other hand, as expected from the theoretical findings, the LP

File	n	LP GAP			Time				NodesBB			
		C_1	C_2	C_3	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
pmed1	10	4.3	2.2	0.1	0.0	0.3	0.2	0.0	29	19	1	1
	15	14.5	8.8	0.0	0.4	3.5	1.0	0.4	3664	572	1	1
	20	13.5	9.0	0.1	21.3	41.6	4.0	2.4	83068	5408	1	1
	25	12.8	8.5	0.0	267.2	456.9	15.2	9.4	459087	22114	1	1
	30	15.7	9.8	0.1	722.5	1050.3	105.5	50.7	528145	11028	20	7
	35	18.4	11.5	0.0	1088.2	1210.7	293.5	149.6	385108	4265	1	1
	40	21.2	14.0	0.1	1200.2	1213.3	885.1	488.4	303871	1877	4	4
pmed2	10	13.0	6.8	0.0	0.1	0.5	0.1	0.1	218	115	1	1
	15	14.2	8.8	0.0	0.3	3.7	0.5	0.5	1566	955	1	1
	20	16.0	8.7	0.0	3.5	35.9	2.5	2.5	15747	5916	3	2
	25	17.6	8.1	0.0	37.3	165.0	9.2	9.3	61588	5258	2	1
	30	19.9	11.9	0.2	879.9	1203.5	47.4	48.7	600064	14171	18	9
	35	21.0	12.7	0.1	1200.2	1209.4	113.2	130.2	365431	5046	5	4
	40	22.2	13.6	0.1	1200.1	1214.8	483.3	509.7	209979	1660	12	6
pmed3	10	6.4	2.0	0.0	0.0	0.3	0.0	0.1	18	16	1	1
	15	11.5	5.8	0.1	0.3	4.3	0.4	0.4	820	598	1	1
	20	22.2	14.6	0.0	250.1	357.3	3.0	2.9	747206	67113	1	1
	25	20.2	15.0	0.0	514.6	649.6	11.0	11.0	942326	26338	2	1
	30	21.0	14.6	0.0	916.3	1204.2	48.9	48.0	814889	13579	2	1
	35	25.4	17.1	0.2	1200.2	1209.7	216.8	219.7	472411	4249	26	1
	40	24.0	15.7	0.0	1200.3	1214.3	534.6	538.4	238041	1494	2	2
pmed4	10	14.6	6.5	0.0	0.3	0.7	0.1	0.1	180	253	1	1
	15	11.7	6.4	0.0	1.0	3.5	0.4	0.4	1300	560	1	1
	20	13.1	7.0	0.0	8.0	26.3	2.4	2.4	16690	2244	2	1
	25	18.5	10.6	0.2	469.7	675.4	11.6	11.3	713877	29653	8	4
	30	18.8	9.8	0.0	621.9	1116.3	39.2	40.6	514557	7849	2	1
	35	20.8	12.6	0.0	1200.2	1210.1	133.3	138.5	380402	4950	2	2
	40	22.2	14.3	0.1	1200.4	1210.0	460.5	466.6	229642	3005	17	3

Table 3.3: Average computational results as a function of source file and n

File	f_j	LP GAP			Time				NodesBB			
		C_1	C_2	C_3	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
pmed1	50	25.8	16.6	0.0	703.7	667.7	164.0	90.3	750966	15994	2	2
	75	20.0	12.6	0.0	594.9	702.2	182.0	98.4	366724	12417	1	1
	100	15.8	10.1	0.1	525.9	616.7	199.7	104.8	217873	6765	12	5
	125	12.6	8.1	0.0	517.8	551.5	197.9	105.7	214929	3921	9	3
	150	10.1	6.4	0.0	368.9	534.1	186.3	102.2	80211	2830	1	1
	175	8.6	5.3	0.0	351.8	478.7	187.2	102.9	103523	2052	2	3
	200	7.5	4.6	0.1	236.9	425.6	187.4	96.6	28744	1304	3	2
pmed2	50	28.7	16.9	0.1	527.4	554.7	79.0	86.5	377056	8417	4	2
	75	23.7	13.7	0.0	530.3	578.9	85.9	94.1	285738	8135	4	3
	100	19.6	11.3	0.0	521.0	553.1	107.4	112.4	191922	4750	16	4
	125	16.3	9.1	0.0	516.0	542.7	91.6	100.7	165141	3559	1	1
	150	13.9	7.6	0.0	422.5	536.5	92.0	100.9	96590	3260	2	2
	175	11.8	6.5	0.1	412.3	535.3	97.8	105.9	74852	2485	5	4
	200	9.9	5.5	0.1	391.7	531.5	102.6	100.6	63294	2513	12	8
pmed3	50	29.9	19.2	0.1	834.4	825.1	101.7	103.7	1161895	46920	2	2
	75	24.7	15.8	0.0	753.3	816.7	125.6	128.2	794468	33435	9	3
	100	20.4	13.1	0.0	681.9	731.8	124.2	118.7	560321	13682	3	1
	125	17.1	11.1	0.0	540.4	603.0	129.7	124.8	297781	7011	1	1
	150	14.8	9.7	0.0	521.7	560.8	123.1	123.0	216003	4973	3	2
	175	12.8	8.5	0.1	387.8	551.6	117.0	120.3	111752	4010	14	11
	200	11.1	7.5	0.0	362.3	550.6	93.4	101.8	73489	3356	3	2
pmed4	50	28.7	16.9	0.0	690.6	670.9	79.7	82.8	642622	16419	2	2
	75	23.3	13.2	0.0	687.4	686.7	85.1	88.5	548154	12506	2	2
	100	19.1	10.6	0.1	593.0	694.4	91.4	98.4	343380	8601	1	1
	125	15.6	8.5	0.0	432.8	586.2	88.4	92.6	116445	3913	1	1
	150	13.0	7.0	0.1	379.9	593.2	92.9	92.1	72110	3416	5	2
	175	10.8	5.8	0.0	363.3	514.9	114.9	106.7	87922	2167	16	2
	200	9.1	5.0	0.1	354.6	496.2	95.0	98.7	46012	1492	5	4

Table 3.4: Average computational results as a function of source file and costs f_j

gap is always smaller for (DPLPT$_3$), being the difference with (DPLPT) more noticeable with the increase of the problem size.

Table 3.4 allows to compare the formulations when opening costs change. We observe that the instances with tightest gaps are those with highest opening costs, for which fewer facilities are open in an optimal solution. Conversely, running times do not seem to directly depend on f_j in overall terms, nor does the number of nodes explored in the branching.

We further provide several charts that better illustrates the comparison of the four configurations. These depict, for every configuration, an step function to represent the number of instances solved in total after some time or some nodes of the branching tree. LP gaps throughout the testbed are also represented with step functions.

To begin with, a comparison of the LP gaps is given in Figure 3.7. The figure shows relative gaps within the optimum— which is known for all the instances— and the linear relaxations of (DPLPT) and (DPLPT$_3$) when no cut is added and after adding the cuts to the root. On the one hand, experiments support Proposition 3.1, i.e., the LP gap is always tighter for (DPLPT$_3$). On the other hand, applying the separation heuristic at the root surprisingly closes the gap in almost all the cases. This suggests that clique facets (3.12) are very close to cover the side of the polyhedron $\mathcal{B}_{dplpt}$ that faces towards the direction of optimization.

Figure 3.8 gives a comparison of the running times of the four configurations tested. The abscissa axis shows the time in seconds. The ordinate axis shows the percentage of instances that are solved before each time tick mark. According to our experiments, (DPLPT) clearly

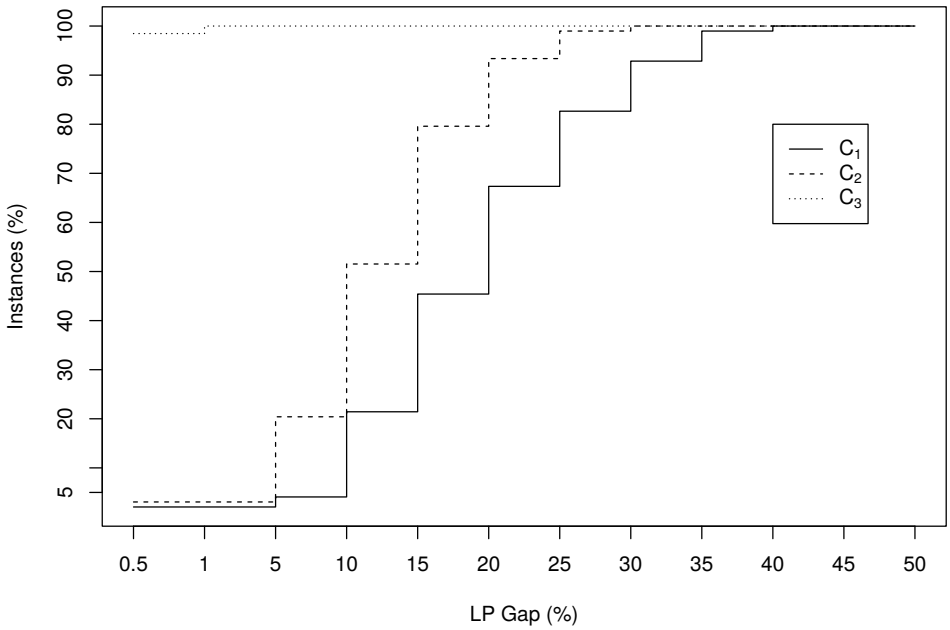


Figure 3.7: Percentage of instances within different LP gaps

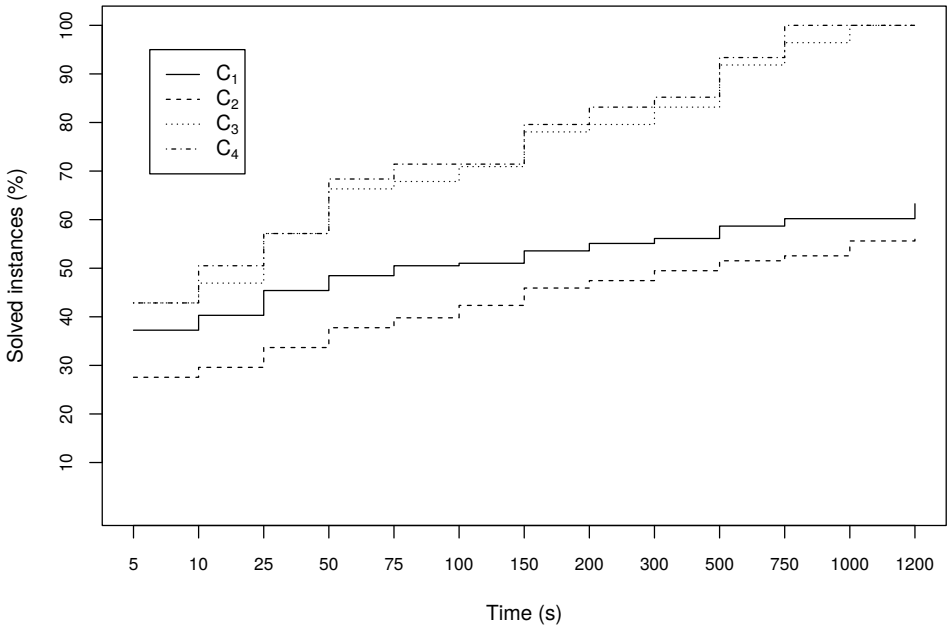


Figure 3.8: Percentage of solved instances as a function of time

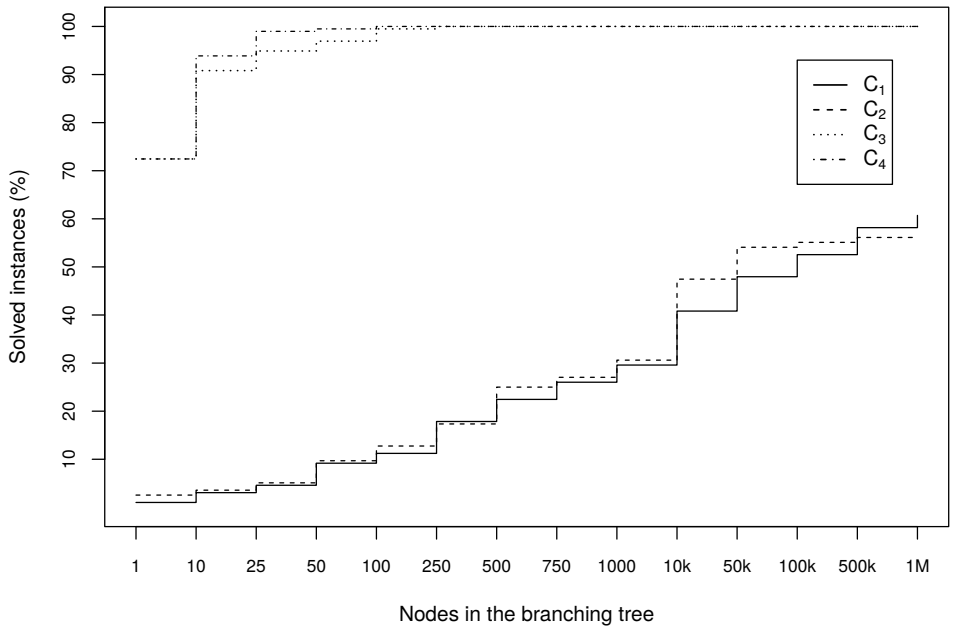


Figure 3.9: Percentage of solved instances as a function of the nodes of the branching tree

outperforms (DPLPT$_3$). However, when the separation heuristic is incorporated to (DPLPT$_3$), things turn out differently. The resulting configurations solve about 70% of the instances in 75 seconds, while (DPLPT) can only solve 62% of the instances within the time limit.

Finally, Figure 3.9 compares the four configurations in terms of the nodes of the branching tree. The abscissa axis shows the number of nodes in the tree, and the ordinate axis depicts the percentage of instances solved up to those number of nodes. Formulations (DPLPT) and (DPLPT$_3$) present a very similar performance in overall terms. The use of the separation heuristic sharply reduces the size of the tree, which consists of only one node for more than 70% of the instances and no more than 50 nodes throughout the testbed.

Part II

Related problems

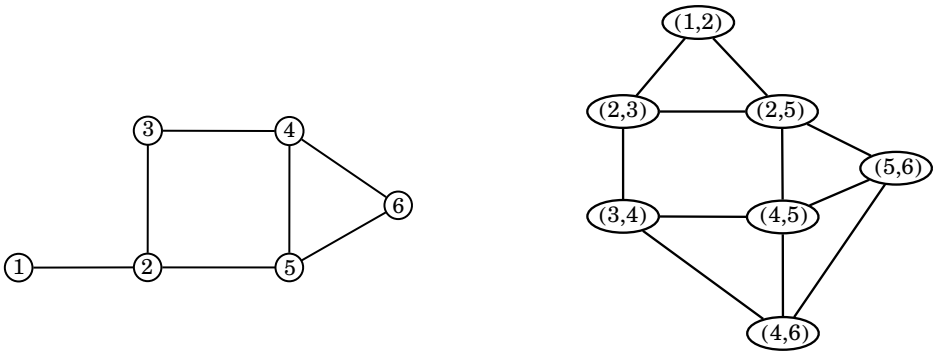
Chapter 4

The edge deletion problem for the property of being line-invertible

Haplotype phasing, which consists of estimating the haplotypes that produced a current population of genotypes, is a primary problem in the analysis of genetic data. In this context, consistency relations between genotypes that could have been originated from a common ancestor are codified by a graph. Root graph reconstruction results useful here to estimate the original population size, that is, the number of generating haplotypes. However, if all the consistency relations are considered, sometimes reconstruction from the graph is not possible. In these cases, one needs to disregard some of these relations, or, in other words, delete some of the edges of the consistency graph. A combinatorial problem then arises, namely which edges to remove so that we retain as much information as possible.

We approach population size estimation when some edges are to be removed from the graph. We propose an integer program to decide which edges to delete. The formulation is compared to that introduced by Halldórsson et al. (2013), which is to our knowledge the only existing integer program for the problem. They interpreted the problem as a graph coloring and so do we. However, our model has a reduced number of variables, which represent color sharing among nodes without explicitly stating which are the colors shared. We devise several families of valid inequalities for our model, including set packing constraints. Symmetry breaking, which is a recurrent problem in Integer Programming that occurs also in graph coloring, is also addressed. Preliminary computational tests showed that dual bounds of our enhanced formulation and that of Halldórsson et al. (2013) were not comparable. In view of these results, we introduce a third approach that combines the previous two formulations and some linking inequalities. We further explore towards removal of some of the constraints of the resulting formulation that do not affect model validity. Our computational experiments allow empirical comparison between the different models and ultimately demonstrate the utility of the proposed alternatives.

Interestingly, the problem studied in this chapter is connected to the variants of the Simple Plant Location Problem of chapters 2 and 3. We will see how it can be interpreted as a facility location problem where incompatibility requirements of Chapter 2 and twinning constraints of Chapter 3 are put together. Clients would correspond to genotypes, while facilities would stand for the generating haplotypes. On the one hand, pairs of incompatible clients will be interpreted as inconsistent genotypes. On the other hand, twinned clients will stand for genotypes generated by the same haplotype.



(a) G (b) $L(G)$

Figure 4.1: A graph and its line graph

4.1 Preliminaries

Line graphs

Given a graph $G = (V, E)$, its *line graph*, $L(G) = (E, F)$, has one node corresponding to each edge $e \in E$ and one edge $f = (e_1, e_2) \in F$ for all $e_1, e_2 \in E$, $e_1 \neq e_2$, that share a node, that is, $e_1 = (u, v)$ and $e_2 = (v, w)$ for some $u, v, w \in V$. Although line graph is the most common term for $L(G)$, it has received many names in the literature, such as *interchange graph*, van Rooij & Wilf (1965), *derived graph*, Beineke (1970) and *dual graph*, Whitney (1992) (note, however, that the dual of $L(G)$ is not G). Conversely, a graph H is *line-invertible* if there exists G such that $H = L(G)$. G is then known as the *root graph* of H. When we speak about the root of a graph H, it is usually assumed that H is connected, since the problem is separable otherwise. As an illustrative example, Figure 4.1 shows a graph and its line graph, whose nodes are labeled with the corresponding edges in the root.

Krausz (1943) proved that a graph H is line-invertible if and only if there is a partition of its edges that induces complete subgraphs such that no node belongs to more than two of the subgraphs. Each of these complete subgraphs corresponds with a node in the root graph, and a node $e = (u, v) \in H$ can belong at most to those of u and v. But not every node in the root has an associated complete subgraph in H. The following example illustrates Krausz's characterization.

Example 4.1. Consider graphs G and $L(G)$ depicted on Figure 4.1. Edges that incide in the same node of G produce complete subgraphs in $L(G)$, like the triangle in Figure 4.1b induced by $(2, 5)$, $(4, 5)$ and $(5, 6)$, which share node 5 in G. Furthermore, we can identify a triangle in $L(G)$ for every node of G with degree 3, i.e., nodes 2, 4 and 5. The partition by Krausz would be $F_2 = \{((1, 2), (2, 3)), ((2, 3), (2, 5)), ((1, 2), (2, 5))\}$, $F_3 = \{((2, 3), (3, 4))\}$, $F_4 = \{((4, 5), (4, 6)), ((3, 4), (4, 6)), ((3, 4), (4, 5))\}$, $F_5 = \{((2, 5), (5, 6)), ((2, 5), (4, 5)), ((4, 5), (5, 6))\}$ and $F_6 = \{((4, 6), (5, 6))\}$. These correspond to nodes 2, 3, 4, 5 and 6 of G, respectively. Node 1 is not associated to any of the complete subgraphs because it has degree 1 in G. △

The characterization given by Krausz comes in a natural way from the definition of a line graph, but there are more. Another characterization, given by van Rooij & Wilf (1965), uses the concept of *odd triangles*. Recall that triangles are complete subgraphs of three nodes. A triangle in a graph is even if every node of the graph is adjacent to either 0 or 2 nodes of the triangle, and it is odd otherwise. The characterization described by van Rooij and Wilf states

(a) $K_{1,3}$ (b) K_3

Figure 4.2: The only two connected graphs having the same line graph, K_3

that a graph is line-invertible if and only if it does not contain $K_{1,3}$ as subgraph and, if $\{a, b, c\}$ and $\{a, b, d\}$ are odd triangles, then either $c = d$ or c and d are adjacent. Figure 4.2a illustrates $K_{1,3}$, which is clearly not a line graph by the characterization of Krausz. Beineke (1970) found a new characterization through nine forbidden subgraphs. Besides these theoretical results, there exist algorithms to find the root graph in linear time, Lehot (1974); Roussopoulos (1973). The root graph, if exists, is unique except for K_3, which has $K_{1,3}$ and also K_3 itself as roots (see Figure 4.2), Whitney (1992).

Line graphs are relevant in many domains other than Graph Theory, such as percolation theory (see Wierman et al., 2007) or community detection (see Ahn et al., 2010; Evans & Lambiotte, 2009; Manka-Krason et al., 2010). The following section explains connections to computational biology and motivates the study of the problem that gives name to this chapter. Links to location models of previous chapters are also described.

Relation with plant location

In previous chapters, we have studied two variants of the Simple Plant Location Problem, which arose when different requirements concerning clients allocation to facilities were considered. Here we show that combining both variants results in a model whose feasible solution can be identified with that of line graph reconstruction.

We consider then a plant location scenario in which each client must be allocated to two different facilities. In addition, suppose that $G_{I,T}$ is a graph having one node per client and edges standing for the following requirements:

1. Two adjacent clients must be allocated to only one facility in common (twinning),

2. Two non-adjacent clients cannot be assign to the same facility (incompatibility).

If location and allocation costs are null, finding a feasible solution to the plant location problem under the above requirements is equivalent to deciding if $G_{I,T}$ is line invertible. Indeed, the nodes of the root can be identified with the facilities needed to serve the clients. According to Krausz's characterization, each of the nodes has an associated complete subgraph in $G_{I,T}$, which will be the clients allocated to the facility in question. Each client in $G_{I,T}$ belong to two of these complete subgraphs, as it is allocated to two facilities. Finally, every twinned pair is allocated to a common facility and incompatible ones do not share any facility. Observe that the number of nodes of the root is a lower bound for the number of open facilities in a feasible solution. Moreover, if $G_{I,T}$ is not line invertible, the location problem above presented is not feasible. We can consider then the option of not satisfying some of the clients allocation requirements so that there exists a feasible solution, that is, deleting some of the edges of $G_{I,T}$ so that it is line invertible, which is the problem addressed in this chapter.

Haplotype phasing

In Genetics, haplotypes codify certain regions of the genome that show a statistically significant variability within a population. It has been observed that such variability plays an important role in human variation and genetic diseases (see Hoehe et al., 2000; Terwilliger & Weiss, 1998). A haplotype of a certain individual is represented by a binary string, 1 codifies the least frequent allele within the population and 0 the most frequent one. The entries of this binary string are usually called sites. Two haplotypes (ancestors) generate a genotype (descendant), codified as a $\{0, 1, 2\}$ string, 0 and 1 encoding homozygous sites and 2 representing heterozygous sites. That is $h_1 \oplus h_2 = g$ where $h_1, h_2 \in \{0, 1\}^n$, $g \in \{0, 1, 2\}^n$ and $1 \oplus 1 = 1$, $0 \oplus 0 = 0$, $1 \oplus 0 = 2$ and $0 \oplus 1 = 2$. Haplotype phasing consists of estimating the haplotypes that resolve a set of genotypes, i.e. a set of ancestors that could produce a current population of genotypes. Two genotypes may have a common ancestor if their $\{0, 1, 2\}$ strings are consistent, i.e., given a site, either some of the strings has code 2 or both strings have the same code (0 or 1).

Consistency relations between genotypes can be represented by a graph called the Clark consistency (CC) graph (see Clark, 1990). Each genotype is represented by a node and adjacent nodes identify consistent genotypes. Figure 4.3a shows the CC graph for a set of five genotypes. Genotypes 210 and 021 are not adjacent in this graph because of their third sites— the former has a 0 and the latter has a 1, so they cannot have a common ancestor. If the CC graph is line-invertible, finding its root gives an estimation on the number of haplotypes needed to explain the genotypes. Indeed, when one checks for line-invertibility, a complete subgraph corresponds to a common node in the root, while here it is potentially interpreted as a common ancestor for the corresponding individuals. When the CC graph is line-invertible it is said to be *allelable*. Further details about equivalence between line-invertible graphs and allelable graphs can be found in Halldórsson et al. (2013). The following example illustrates the relation between the two concepts.

Example 4.2. Consider Figure 4.3b, which depicts the same CC graph as Figure 4.3a. The different line traces give a partition of the edges of the graph in three groups satisfying Krausz's characterization of line graphs, which yields a root graph with three nodes. Nodes within each group are all compatible with each other, and the nodes of the root are identified with their potential ancestors. For the set of genotypes in the example, we found three potential ancestors. The number of nodes of the root graph gives in general a lower bound on the cardinality of the generating population of haplotypes. Figure 4.3b depicts the codes for the haplotypes that would correspond with the three ancestors. An ancestor that generates 202 and 021 must have code 001 and, similarly, 110 is the ancestor of 122 and 210. When it comes to the triangle induced by 102, 122 and 202, it is clear that their ancestor begins with 10. To determine its last site, we need to look at the other just-discovered haplotypes, 001 and 110. The following equations are to be verified:

$$10x \oplus 001 = 202,$$

$$10x \oplus 110 = 122.$$

The former gives $x = 0$ and the latter $x = 1$. This is simply because we need more haplotypes to generate the current genotypes. For instance, 100, 101, 001 and 110 is a feasible set of ancestors. The minimum number of haplotypes needed to resolve the population is in fact 4. The reason why we get the lower bound 3 is that, when obtaining the root, we assume that every pair of genotypes that are neighbors share an ancestor, while edges in the CC graph only represent that the nodes can potentially, but not necessarily, be twinned. △

(a) CC graph G_{cc} (b) Krausz partition for G_{cc}

Figure 4.3: A Clark consistency graph and its Krausz partition

And, what if the CC graph is not allelable? Take the example of $K_{1,3}$ depicted on Figure 4.2a, which we know that is not allelable. The edges here mean that the central node may have an ancestor in common with each of the other three nodes of the graph, which at the same time are incompatible with each other. Since a node only have two ancestors, the central node will share an ancestor with at most two of its three neighbors, i.e., one of the edges is not "used". In general, when the CC graph is not allelable we have edges that do not represent true sharings, i.e., the corresponding genotypes do not have an ancestor in common. One solution to be able to estimate the haplotypes in these cases is to eliminate these false relations within nodes while trying to have the least impact on the graph topology. A combinatorial problem then arises, which is to find the minimum number of edges that we have to delete so that we obtain an allelable graph.

4.2 State of the art

In general, when $\mathbb{P}$ is a property of a graph, we can define the corresponding *edge deletion problem* as follows: "find the minimum number of edges whose deletion produce a graph satisfying $\mathbb{P}$". This terminology gives name to the chapter and problem at hand, the edge-deletion problem for the property of being line-invertible, EDPL from now on. For several common properties, including that of being line-invertible, Yannakakis (1978) proved that edge deletion problems are NP-complete. In fact, the EDPL can be seen as a variant of graph coloring, which is known to be NP-hard (see Garey & Johnson, 1979). In general, a coloring of the nodes or edges of a graph $G = (V, E)$ is a mapping $\varphi : V \to \mathcal{C}$ or $\varphi : E \to \mathcal{C}$ that assigns a color from the set $\mathcal{C}$ to every node or edge in G. Coloring problems can be extended to assigning several colors, arising the so-called k-fold coloring or multi-coloring problems (see for instance Campêlo et al. (2013)).

The following two definitions serve to establish the relation between line-invertible graphs and graph coloring of edges and nodes, which will be ultimately useful to state the EDPL in Lemma 4.1.

Definition 4.1 (*Line-graph edge coloring*)**.** Let $G = (V, E)$ be an undirected graph. We will say that $\varphi : E \to \mathcal{C}$ is an edge coloring of a line-graph if the two following conditions are met:

(E1) Each subset of edges having the same color $c \in \mathcal{C}$, $E_c := \{e \in E : \varphi(e) = c\}$, induces a complete subgraph. That is, $G_c[E_c] := (V_c, E_c)$, with $V_c = \{v \in V : \exists e \in E_c \text{ s.t. } v \in e\}$ is complete.

(E2) For all $v \in V$ there exist at most two colors, c_1 and c_2, such that $v \in V_{c_1} \cap V_{c_2}$. △

If there exists an edge coloring for G that satisfies (E1) and (E2) then G is line-invertible. In effect, it is straightforward to see that these conditions produce an edge partition as that

proposed by Krausz (1943) to characterize line-invertible graphs. Conversely, if G is line-invertible, one can obtain an edge coloring satisfying (E1) and (E2) just by assigning different colors to the groups of edges in Krausz's partition.

A line-graph edge coloring is equivalent to a vertex coloring as follows.

Definition 4.2 (*Line-graph node coloring*)**.** Let $G = (V, E)$ be an undirected graph. We will say that a coloring of the nodes of G with two colors, $\varphi : V \to \mathcal{C} \times \mathcal{C}$, is a node coloring of a line graph if the following conditions are met:

(N1) $\varphi = (\varphi_1, \varphi_2)$ and $\varphi_1(v) \neq \varphi_2(v)\ \forall v \in V$.

(N2) For all $(u, v) \in E$, $|\{\varphi_1(u), \varphi_2(u)\} \cap \{\varphi_1(v), \varphi_2(v)\}| = 1$.

(N3) For all $(u, v) \notin E$, $|\{\varphi_1(u), \varphi_2(u)\} \cap \{\varphi_1(v), \varphi_2(v)\}| = 0$. △

It is easy to check that this node coloring is equivalent to the desired edge coloring. Indeed, (N1) states that the colors given to a node are different; (N2) would allow and edge coloring by assigning to every edge (u, v) the color that u and v have in common; finally, (N3) ensures that such edge coloring induces the complete subgraphs of (E1). Conversely, if an edge coloring satisfies (E1) and (E2), a line-graph node coloring is easily derived by assigning each node the colors of the two complete subgraphs to which it belongs.

As a consequence, Definition 4.2 gives also a characterization of line graphs. This characterization was previously stated in Lemma 2.1 of Halldórsson et al. (2013), by using a different argument. We next formally state the EDPL in terms of the coloring characterization given by definitions 4.1 and 4.2.

Lemma 4.1. *Let $G = (V, E)$ be an undirected graph. The EDPL is equivalent to finding $E' \subseteq E$ such that the induced graph $G[E \setminus E']$ admits a line-graph node/edge coloring and $|E'|$ is minimized.*

Halldórsson et al. (2013) presented an ILP formulation to solve the EDPL. The formulation consists of finding a node coloring that satisfies (N1), (N2) and (N3), minimizing the number of pairs of nodes u and v, $(u, v) \in E$, that do not verify (N2), i.e., the deleted edges.

The authors took inspiration from Campêlo et al. (2007) representative formulation for the classic vertex coloring problem in order to eliminate equivalent (symmetric) solutions. In the classic vertex coloring, each node receives one color. The idea of the representative formulation is to identify colors with nodes so that if node i is colored with i it is said to be a representative. An ordering of the nodes is consider so that every node receives the color of one of its preceding representatives or it is a representative itself. Halldórsson et al. (2013) explained their idea for symmetry breaking as follows. Each node i "owns" two colors, $2i-1$ and $2i$. Every node i can be assigned either some color(s) owned by some neighbor(s) k such that $k < i$, and/or some of its own colors, $2i - 1$ and $2i$. A node will be assigned a color owned by one of its neighbors if and only if this neighbor is using the color in question. This means that k is always the smallest node among those colored with $2k - 1$ or $2k$. On the other hand, if one has to draw on colors $2i$ and $2i - 1$ to color node i, $2i - 1$ will be the first one to be taken and $2i$ will be used only if $2i - 1$ has already been assigned to i.

Before presenting the formulation of Halldórsson et al. (2013), we define the left-neighborhoods of a node i as $N^-(i) = \{k \in V : k < i,\ (i, k) \in E\}$ and $N^-[i] = N^-(i) \cup \{i\}$, for all $i \in V$. The precedence relation between nodes could be any order within V, but the reader can assume

that $V = \{1, \ldots, n\}$ and $<$ is common numerical order. For convenience in the exposition we will use the following notation:

$$\mathcal{C}(i) = \{2k-1, 2k \ : \ k \in N^-[i]\}, \qquad \mathcal{C}(i,j) = \mathcal{C}(i) \cap \mathcal{C}(j),$$

which identify the colors that can be assigned to node i and those that can be shared by i and j, respectively. We will also denote with $\varphi : V \to \mathcal{C} \times \mathcal{C}$ a coloring function, $\varphi = (\varphi_1, \varphi_2)$.

Halldórsson et al. (2013) used three families of variables: y to gather the information about the two colors given to each node; variables s to identify when the two end-nodes of an edge have the same color (they are products of y variables); and variables d that account for the number of edges deleted. These binary variables are formally defined as follows

$y_{ir} = 1$ iff $r \in \{\varphi_1(i), \varphi_2(i)\}$, $i \in V$, $r \in \mathcal{C}(i)$,
$s_{ijr} = 1$ iff $r \in \{\varphi_1(i), \varphi_2(i)\} \cap \{\varphi_1(j), \varphi_2(j)\}$, $(i,j) \in E$, $r \in \mathcal{C}(i,j)$, and
$d_{ij} = 1$ iff edge (i,j) is deleted, $(i,j) \in E$.

The formulation of Halldórsson et al. (2013) reads:

$$\begin{aligned}
\text{(H)} \quad \min \quad & \sum_{(i,j)\in E, i<j} d_{ij} & & \\
\text{s.t.} \quad & \sum_{r\in\mathcal{C}(i)} y_{ir} = 2 & \forall i \in V & (4.1)\\
& y_{ir} + y_{jr} \le 1 & \forall (i,j) \notin E, \forall r \in \mathcal{C}(i,j) & (4.2)\\
& \sum_{r\in\mathcal{C}(i,j)} s_{ijr} = 1 - d_{ij} & \forall (i,j) \in E & (4.3)\\
& y_{ir} + y_{jr} - 1 \le s_{ijr} & \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j) & (4.4)\\
& s_{ijr} \le y_{ir} & \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j) & (4.5)\\
& s_{ijr} \le y_{jr} & \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j) & (4.6)\\
& y_{i,2i} \le y_{i,2i-1} & \forall i \in V & (4.7)\\
& y_{i,2k-1} \le y_{k,2k-1} & \forall i \in V, \forall k \in N^-(i) & (4.8)\\
& y_{i,2k} \le y_{k,2k} & \forall i \in V, \forall k \in N^-(i) & (4.9)\\
& y_{ir}, s_{ijr'}, d_{ij} \in \{0,1\} & \forall i \in V, \forall r \in \mathcal{C}(i), \forall (i,j) \in E, \forall r' \in \mathcal{C}(i,j). &
\end{aligned}$$

Constraints (4.1) impose that each node is assigned two colors, and (4.2) guarantee that non-adjacent nodes are given distinct colors. Constraints (4.3) say that if two adjacent nodes are assigned different colors the edge must be deleted and, at the same time, guarantee that two adjacent nodes share one color at most. Families (4.4)-(4.6) are standard "product constraints" that guarantee that $s_{ijr} = y_{ir}y_{jr}$ for every integer solution. Finally, constraints (4.7)-(4.9) are the symmetry breaking constraints inspired by Campêlo et al. (2007).

4.3 A new model

As opposed to the model of Halldórsson et al. (2013), the formulation we propose is not based on a direct translation of definitions 4.1 or 4.2. However, as we will show later, a line-graph node/edge coloring can be obtained as a byproduct of our solution. We propose to use only two families of variables: x, with four indices, and d, which are identical to that of (H). With variables x, we will represent those pairs of adjacent nodes that share the same color,

distinguishing which function, either φ_1 or φ_2, gives the color to each node. More precisely, our variables are defined as follows:

x_{ijab} = 1 iff $\varphi_a(i) = \varphi_b(j)$, $(i,j) \in E$, $a, b \in \{1,2\}$, and
d_{ij} = 1 iff $(i,j) \in E$ is deleted.

With these variables, colors are not explicitly stated. If for instance $x_{ij21} = 1$ we know that i and j share a color, which corresponds to $\varphi_2(i)$ and $\varphi_1(j)$, but we do not know if such color is blue. Nonetheless, these variables are enough to formulate the problem. Call Γ the set of all the triangles in G, $\Gamma = \{\{i,j,k\} \subseteq V : i < j < k\,;\, (i,j),(i,k),(j,k) \in E\}$. The following mathematical program gives a formulation for the EDPL,

$$\text{(M')} \quad \min \quad \sum_{(i,j)\in E} d_{ij}$$

$$\text{s.t.} \quad x_{ija1} + x_{ija2} + x_{ika1} + x_{ika2} \leq 1 \quad \forall (i,j),(i,k) \in E, \; j \neq k, (j,k) \notin E, \; a \in \{1,2\} \tag{4.10}$$

$$x_{ijba} \geq x_{jkac} + x_{ikbc} - 1 \quad \forall \{i,j,k\} \in \Gamma, \; a,b,c \in \{1,2\} \tag{4.11}$$

$$x_{ikbc} \geq x_{jkac} + x_{ijba} - 1 \quad \forall \{i,j,k\} \in \Gamma, \; a,b,c \in \{1,2\} \tag{4.12}$$

$$x_{jkac} \geq x_{ikbc} + x_{ijba} - 1 \quad \forall \{i,j,k\} \in \Gamma, \; a,b,c \in \{1,2\} \tag{4.13}$$

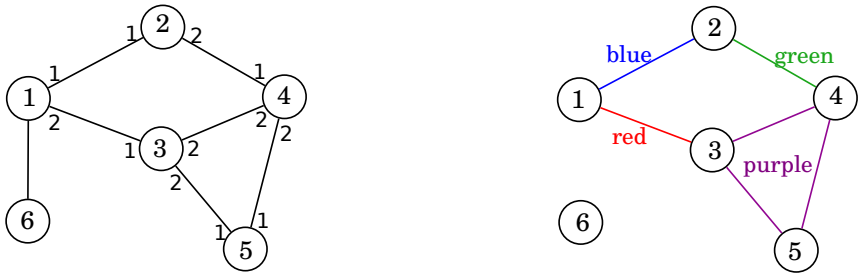
$$\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} = 1 - d_{ij} \quad \forall (i,j) \in E \tag{4.14}$$

$$x_{ijab}, d_{ij} \in \{0,1\} \quad \forall (i,j) \in E, a, b \in \{1,2\}.$$

Constraints (4.10) guarantee that two non-adjacent nodes do not share a color. In effect, these constraints read: if i has two non-adjacent neighbors j and k, it may share a color with one of them, but not the same one with the two. Constraints (4.11)-(4.13) impose the transitivity of color sharing and are defined for every subset of nodes inducing a triangle. Take, for instance, (4.11). These constraints are active when the r.h.s. is 1, that is, when $x_{jkac} = x_{ikbc} = 1$. If this is the case, then (4.11) forces that $x_{ijba} = 1$. Note that $x_{jkac} = 1$ means $\varphi_a(j) = \varphi_c(k)$ and $x_{ikbc} = 1$ means $\varphi_b(i) = \varphi_c(k)$, while $x_{ijba} = 1$ just impose what follows by transitivity, $\varphi_b(i) = \varphi_a(j)$. Constraints (4.12) and (4.13) are analogous. Finally, (4.14) ensure that (i,j) is removed if i and j do not share color and that they can share one color at most.

The following example illustrates the formulation.

Example 4.3. Consider the graph depicted in Figure 4.4a. Every edge (i,j) is labeled with the indices a, b of the corresponding x_{ijab}-variable that takes value 1 in a feasible solution of (M'). For instance, according to the figure, $x_{1211} = 1$ and $x_{1321} = 1$. This means that colors $\varphi_1(1)$ and $\varphi_1(2)$, given to nodes 1 and 2 respectively, coincide and so do $\varphi_2(1)$ and $\varphi_1(3)$, which were given to nodes 1 and 3. Constraints (4.10) when $i = 1$ and $j = 6$ forbid node 1 to share color $\varphi_1(1)$ with 6 when $k = 2$, since $(2,6) \notin E$. Similarly, when $k = 3$, constraints (4.10) avoid that $\varphi_2(1)$ coincides with some of the colors given to node 6. As a result, $d_{16} = 1$ in the depicted solution. Moreover, in this example, 1 is the optimum of the problem. This is, indeed, due to fact that 1 has three neighbors that induce an independent set in the graph (i.e., none of them are adjacent to each other). On the other hand, color sharing between nodes 3, 4 and 5 is consistent thanks to constraints (4.11)-(4.13). Note that, even if not explicitly stated, this feasible solution yields a line-graph node coloring. To obtain this coloring, we start from one



(a) Solution to (M') (b) A line-graph edge coloring for the EDPL

Figure 4.4: A graph and optimal solution with optimal value equal to 1

node, say 1, and suppose that $\varphi_1(1)$ =blue and $\varphi_2(1)$ =red. Then, since $x_{1211} = 1$, node 2 has color blue, $\varphi_1(2)$ =blue, and another color, say green, $\varphi_2(2)$ =green. On the other hand, since $x_{1321} = 1$, $\varphi_1(3)$ =red, and we can assume $\varphi_2(3)$ =purple. Now, $x_{2421} = 1$ and $x_{3422} = 1$ force $\varphi_1(4)$ =green and $\varphi_2(4)$ =purple. Finally, $x_{4521} = 1$ and $x_{3521} = 1$ imply that $\varphi_1(5)$ =purple. Note that $\varphi_2(5)$ is not imposed by any of the neighbors of node 5 and then we can assign any new color to it, for instance, white (in fact this color is not needed if we think about edge coloring). Figure 4.4b illustrates the resulting coloring and edge deletion. △

In general, an algorithm to derive a line-graph node coloring from a solution of (M') will be as follows. For every edge (i, j) such that $d_{ij} = 0$ and for a, b such that $x_{ijab} = 1$: if $\varphi_a(i)$ and $\varphi_b(j)$ are not defined yet then let both be equal to a new color. If $\varphi_a(i)$ is already defined let $\varphi_b(j) = \varphi_a(i)$ and vice versa.

Observe that variables d_{ij} can be eliminated from (M'). In effect, using constraints (4.14), d_{ij} can be removed from the objective, which will be equivalent to maximizing the sum $\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab}$. After that, replacing (4.14) with $\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \leq 1$ will complete the transformation, obtaining the following equivalent formulation without variables d_{ij},

$$
\begin{aligned}
\text{(M)}\ \max \quad & \sum_{(i,j)\in E}\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \\
\text{s.t.} \quad & (4.10) - (4.13) \\
& \sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \leq 1 \qquad \forall (i,j) \in E \\
& x_{ijab} \in \{0,1\} \qquad \forall (i,j) \in E, a, b \in \{1,2\}.
\end{aligned} \tag{4.15}
$$

The formulation can still be improved, for instance, by adding symmetry breaking constraints. But before analyzing this and other enhancements in the next section, we shall compare the size of (H) and (M).

Solutions to (H) provide a line-graph node coloring for the EDPL, while variables of (M) only account for the way colors are shared in such a coloring. Because of this fact, (M) has less variables than (H). If one denotes by t the number of triangles in G, $t = |\Gamma|$, variables in each formulation can be easily compared, as follows. Since color candidates for one node are its two colors and the two colors of its neighbors, (H) has $2n + 2|E|$ y-variables. A similar reasoning serves to count the members of the family of s-variables, which are $2|E| + 2t$ in total. On the

one hand, note that for each edge (i, j) with $i < j$ there exist $s_{ij,2i-1}$ and $s_{ij,2i}$, which accounts for $2|E|$ variables of the family. Other than $2i-1$ and $2i$, nodes i and j could share one of the two colors of a common neighbour, say node k, with $k < i$. That is, $s_{ij,2k-1}$ and $s_{ij,2k}$ exist for each $\{k, i, j\} \in \Gamma$, which accounts for the rest of the variables in the family, $2t$ in total. Finally, we avoid counting d-variables in (H) since they can be eliminated by using a similar argument than that used for (M'). In summary, (H) has $2n + 4|E| + 2t$ variables, while (M) only uses $4|E|$ variables.

As for the number of constraints, let t' be the number of triplets $\{i, j, k\}$ such that $(i, j), (i, k) \in E$ and $(j, k) \notin E$. Regarding (H), it is straightforward that there are n constraints in (4.1). The second family, (4.2), is defined in terms of triplets $\{k, i, j\}$ such that $(k, i), (k, j) \in E$ and $(i, j) \notin E$. In effect, $2k-1$ and $2k$ stand for possible values of r in (4.2) whenever $k < i$ and $k < j$. Using this observation, one can check that there are at most $2t'$ constraints in (4.2). Family (4.3) has $|E|$ members; (4.4)- (4.6) are defined for each valid subscript ijr of variables s, so they account for $6|E| + 6t$ constraints; finally, (4.7), (4.8) and (4.9) gather n, $|E|$ and $|E|$ constraints respectively. Consequently, formulation (H) has $n+2t'+|E|+6|E|+6t+n+2|E| = 2n+2t'+6t+9|E|$ constraints at most (or $n+2t'+6t+7|E|$ if symmetry breaking constraints are excluded). On the other hand, formulation (M) has $2t'$ constraints of type (4.10). Constraints in (4.11) account for 2^3 (the different combinations of a, b and c) times t, and the same happens for (4.12) and (4.13). Finally, (4.15) is composed by $|E|$ constraints. In sum, (M) has $2t' + 24t + |E|$ constraints in total. It is not straightforward to analyze which formulation, (M) or (H), has less constraints. Since they are dependent on the structure of the graph G in question, both situations can occur. Consider for instance a star, that is, a graph consisting of $n + 1$ nodes, one of them adjacent to the rest, which makes $|E| = n$. In this case $t = 0$ and $t' = \binom{|E|}{2}$. This implies that (M) has $2t' + 24t + |E| = n^2$ constraints, while (H) has $n + 2t' + 6t + 7|E| = n^2 + 7n$ if the center of the star corresponds to node 1. Conversely, if one takes the wheel with $n + 1$ nodes, which is made of a cycle of n nodes plus a central node adjacent to the rest, things change. In this case, $|E| = 2n$ and $t = n$. Formulation (M) has then $2t' + 24t + |E| = 2t' + 26n$ constraints, while (H) has $n + 2t' + 6t + 7|E| = 2t' + 21n$ at most.

4.4 Valid inequalities

After studying the structure of (M), we have derived several valid inequalities that we present here in different groups. Even if colors do not appear explicitly in our formulation, (M) has symmetric solutions, since color positions of a node are interchangeable. The first family of inequalities tries to avoid these symmetric solutions. The second one represents the fact that nodes of independent sets can not have the same color. Another family of inequalities considers the case in which a deletion of an edge produces an independent set of 3 nodes. Finally, the last one is related to the transitivity of color sharing.

We will denote $N(i)$ the neighborhood of node i, $N(i) = \{j \in V : (i, j) \in E\}$, and $N[i]$ its closed neighborhood, $N[i] = N(i) \cup \{i\}$.

Symmetry breaking inequalities

Variables x_{ijab} produce symmetric solutions by their very nature. Indeed, we can always swap colors positions of a node and update the variables to be consistent with the change without altering the objective value. One way of forbidding such a change is to enforce the position of the colors for some nodes. More specifically, given a node i, we pay attention to its neighbor

with smallest index, j_1^i. Since i has two colors to share with its neighbors, we can assume without loss of generality that j_1^i is not the one sharing $\varphi_2(i)$. That is, the following variables can be set to zero:

$$x_{ij_1^i 21} = x_{ij_1^i 22} = 0 \qquad \forall i, j_1^i := \min\{j : j \in N(i)\}. \tag{4.16}$$

Nevertheless, if (i, j_1^i) is deleted, $x_{ij_1^i ab} = 0\ \forall a, b \in \{1, 2\}$ and the previous equalities are unnecessary. We can consider now the second neighbor of i with smallest index, say j_2^i. Similarly as before, we can assume that j_2^i does not share $\varphi_2(i)$ if (i, j_1^i) is deleted:

$$x_{ij_2^i 21} + x_{ij_2^i 22} \leq x_{ij_1^i 11} + x_{ij_1^i 12} \qquad \forall i, j_2^i := \min\{j : j_2^i \neq j_1^i,\ (i, j) \in E\}. \tag{4.17}$$

If $x_{ij_1^i 11} = 1$ or $x_{ij_1^i 12} = 1$ (4.17) trivially stands. Otherwise, $x_{ij_1^i 11} + x_{ij_1^i 12} = 0$, which together with (4.16) implies that i and j_1^i do not share any color. Since i and j_1^i are adjacent, this means that the edge will be deleted. Then we can assume that i does not share $\varphi_2(i)$ with j_2^i, i.e., $x_{ij_2^i 21} + x_{ij_2^i 22} = 0$ and hence (4.17) is valid. We can continue the process with the third, fourth... neighbor of i with smallest index. The following set of inequalities gathers all the cases in a compact way:

$$x_{ij21} + x_{ij22} \leq \sum_{k<j:(i,k)\in E} (x_{ik11} + x_{ik12}) \qquad \forall (i, j) \in E. \tag{4.18}$$

These inequalities read: "if i does not share color $\varphi_1(i)$ with any neighbor $k < j$ then it does not share $\varphi_2(i)$ with j either". Inequalities (4.16) and (4.17) are (4.18) with $j = j_1^i$ and j_2^i respectively. Note that in the former case the summation on the right-hand side of (4.18) becomes empty and the inequality then reads $x_{ij_1^i 21} + x_{ij_1^i 22} \leq 0$.

Set packing inequalities

Because of the definition of the problem, two non-adjacent nodes do not share a color. This means that, given a color and two non-adjacent nodes, one of them at most is given the color. More generally, given a color and an independent set of nodes, one of them at most will have the color.

Since variables of (M) are only defined for adjacent nodes, if two non-adjacent nodes were to share a color, this would happen through transitivity. That is, if $(j, k) \notin E$ and i sharing its color in position a with both j and k existed, then j and k would have the same color. This is precisely what (4.10) forbid. To generalize (4.10), we need to look at node packings inside the neighborhood of i, $N(i)$:

$$\sum_{j\in S}(x_{ija1} + x_{ija2}) \leq 1 \qquad \forall i, \forall S \subseteq N(i), S \in \mathcal{P}_G, \forall a \in \{1, 2\}. \tag{4.19}$$

These are set packing constraints —left hand side is a linear combination of the x-variables with coefficients 0 or 1 and the coefficient on the right hand side is 1— that will be stronger with the size of S. Node i and nodes in S induce a star, in which i is the internal node. When $|S| = 3$ we will refer to this induced subgraph (which is in fact $K_{1,3}$) as *fork*.

Quasi-fork inequalities

Edge deletions can yield substructures that were not originally in graph G, like new independent sets. If this occurred, inequalities (4.19) would be valid for the nodes in question. In this section we adapt valid inequalities (4.19) to cover such cases.

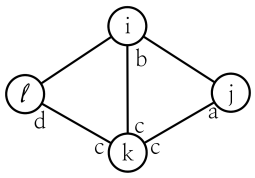


Figure 4.5: Subgraph of inequalities (4.22)

For all $\{i,j,k\} \in \Gamma$ and $\ell \in N(i)$, $\ell \notin N[j]$ and $\ell \notin N[k]$, the following inequalities hold:

$$\sum_{a=1}^{2}\sum_{b=1}^{2}(x_{ijab} + x_{ikab} + x_{i\ell ab} - x_{jkab}) \leq 2, \tag{4.20}$$

$$x_{ija1} + x_{ija2} + x_{ika1} + x_{ika2} + x_{i\ell a1} + x_{i\ell a2} - \sum_{b=1}^{2}\sum_{c=1}^{2} x_{jkbc} \leq 1, \quad \forall a \in \{1,2\}. \tag{4.21}$$

Nodes 1, 3, 4 and 5 of the graph on Figure 4.4a illustrate the situation that we are considering. Note that, if $(4,5)$ is deleted, these nodes induce a fork. In general, we will call the subgraph induced by the triangle $\{i,j,k\}$ and ℓ quasi-fork. Inequalities (4.20) and (4.21) are active when $\sum_{a=1}^{2}\sum_{b=1}^{2} x_{jkab} = 0$, that is, when (j,k) is removed. If this were the case, taking $S = \{j,k,\ell\}$, we have that (4.21) would be (4.19), and (4.20) would be the sum over a of (4.19).

Adjacent triangles inequalities

These inequalities are inspired by transitivity inequalities (4.11)-(4.13) of (M), but here we consider two triangles instead of just one. Suppose that $\{i,j,k\}$ and $\{i,\ell,k\}$ are two elements in Γ such that $(j,\ell) \notin E$. We say that they are adjacent triangles because they share edge (i,k) (see Figure 4.5). The following inequalities are valid for (M):

$$x_{ijba} + x_{i\ell bd} \geq x_{ikbc} + x_{jkac} + x_{k\ell cd} - 1, \quad \forall a,b,c,d \in \{1,2\}, \tag{4.22}$$

$$x_{jkab} + x_{k\ell bd} \geq x_{ikcb} + x_{ijca} + x_{i\ell cd} - 1, \quad \forall a,b,c,d \in \{1,2\}. \tag{4.23}$$

Inequalities (4.22) are illustrated in Figure 4.5. Since $(j,\ell) \notin E$, each instance of (4.22) has right-hand side at most 1. This bound is attained either when $x_{ikbc} + x_{jkac} = 2$ or when $x_{ikbc} + x_{k\ell cd} = 2$. Due to transitivity, this would imply either that $x_{ijba} = 1$ in the first case or $x_{i\ell bd} = 1$ in the second. Then, if the right-hand side of (4.22) is 1 it has to be $x_{ijba} + x_{i\ell bd} \geq 1$ and thus the inequality is valid. Inequalities (4.23) are analogous with the difference that i and k are turned around.

4.5 An alternative approach

After doing some computational experiments, we realized that none of the linear relaxations of (M) and (H) is stronger than the other, even after including inequalities of Section 4.4 in (M) or/and inequalities (4.19) with $|S| = 3$. That is, if v(HLP) and v(MLP) are respectively the optimal values of the corresponding formulations after relaxing the integrality of the variables, we have v(HLP)$<$ v(MLP) for some instances and v(HLP)$>$ v(MLP) for some others. In view of this, we propose a family of formulations, (HM), which combines variables y and s of (H) with variables x of (M). Our intention is to produce formulations with stronger linear

relaxations and, with this, being able to reduce the computational times. Actually, we propose three different versions, (HMa), (HMb) and (HMc), the last two being the result of removing some constraints from (HMa).

Before presenting the family (HM), we introduce some constraints that establish links between the variables from (H) and (M). The aim of these linking constraints is to enhance the family (HM).

4.5.1 Linking constraints

Variables x distinguish between the two colors given to a node, i.e., between the components φ_1 and φ_2 of $\varphi = (\varphi_1, \varphi_2)$. Hence, to be able to state the links between x-variables in (M) and y-variables in (H), we need to establish which color is given by the first and second component of φ. We will assume that if a node i is given colors r_1 and r_2 with $r_1 < r_2$ then $\varphi_1(i) = r_1$ and $\varphi_2(i) = r_2$. In terms of x variables, if $(i,j) \in E$, $y_{i,r_1} = y_{i,r_2} = 1$ and $y_{j,r_1} = y_{j,r_3} = 1$, $r_2 \neq r_3$, (i and j share color r_1) this means that:

(i) $x_{ij11} = 1$ iff $r_1 < r_2$ and $r_1 < r_3$,

(ii) $x_{ij22} = 1$ iff $r_1 > r_2$ and $r_1 > r_3$,

(iii) $x_{ij12} = 1$ iff $r_1 < r_2$ and $r_1 > r_3$,

(iv) $x_{ij21} = 1$ iff $r_1 > r_2$ and $r_1 < r_3$.

Based on this assumption, we develop a set of linking constraints between the variables of (H) and (M). Namely, we devise four families of linking constraints that ensure that the four statements above are satisfied.

To begin with, the following constraints correspond to (i) and (ii)

$$s_{ijr} - \sum_{\substack{r' \in \mathcal{C}(i) \\ r' < r}} y_{ir'} - \sum_{\substack{r' \in \mathcal{C}(j) \\ r' < r}} y_{jr'} \leq x_{ij11} \quad \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j), \tag{4.24}$$

$$s_{ijr} - \sum_{\substack{r' \in \mathcal{C}(i) \\ r' > r}} y_{ir'} - \sum_{\substack{r' \in \mathcal{C}(j) \\ r' > r}} y_{jr'} \leq x_{ij22} \quad \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j). \tag{4.25}$$

A constraint in (4.24) or (4.25) is active when its left-hand side is 1. In the case of (4.24), this occurs when nodes i and j share color r ($s_{ijr} = 1$) and not i nor j are assigned a color smaller than r ($\sum_{r' \in \mathcal{C}(i), r' < r} y_{ir'} = 0$ and $\sum_{r' \in \mathcal{C}(j), r' < r} y_{jr'} = 0$). This would mean that i and j are sharing a color, $\varphi_1(i) = \varphi_1(j)$, that is, we are in case (i). If this happens, (4.24) force $x_{ij11} = 1$. A similar argument proves that constraints (4.25) stand for case (ii).

A slightly different argument serves to model (iii) and (iv) with the following mathematical constraints,

$$\sum_{u \in \mathcal{C}(i,j)} s_{iju} - \sum_{\substack{r' \in \mathcal{C}(i) \\ r' < r}} y_{ir'} - \sum_{\substack{r' \in \mathcal{C}(j) \\ r' > r}} y_{jr'} \leq x_{ij12} \quad \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j), \tag{4.26}$$

$$\sum_{u \in \mathcal{C}(i,j)} s_{iju} - \sum_{\substack{r' \in \mathcal{C}(i) \\ r' > r}} y_{ir'} - \sum_{\substack{r' \in \mathcal{C}(j) \\ r' < r}} y_{jr'} \leq x_{ij21} \quad \forall (i,j) \in E, \forall r \in \mathcal{C}(i,j). \tag{4.27}$$

Again, these constraints are active when their left-hand sides are 1. Take for instance (4.26). If i and j share a color, then one of the terms of the first sum on the left-hand side will be one.

If, for some color r, the colors given to i are greater than or equal to r ($\sum_{r' \in \mathcal{C}(i), r' < r} y_{ir'} = 0$) and those given to j are smaller than or equal to r ($\sum_{r' \in \mathcal{C}(j), r' > r} y_{jr'} = 0$), then i and j share color r. When this happens, r coincides with $\varphi_1(i)$ and $\varphi_2(j)$, that is, we are in case (iii). In this case, (4.26) force $x_{ij12} = 1$. A similar reasoning proves that (4.27) serve to model case (iv).

There are other constraints that can be include to model relations coming from the semantic meaning of the decision variables. These relations can be explicitly written in the mathematical model, with the aim of ultimately producing a stronger polyhedron.

For instance, if i and j do not share a color, that is, $\sum_{a=1}^{2} \sum_{b=1}^{2} x_{ijab} = 0$, then node j can not receive any of the colors of i, $2i-1$ or $2i$. This can be expressed with the following constraints:

$$y_{j,2i-1} + y_{j,2i} \leq \sum_{a=1}^{2} \sum_{b=1}^{2} x_{ijab} \quad \forall j > i : (i,j) \in E. \tag{4.28}$$

On the other hand, if $y_{i,2i} = 0$ then color $2i-1$ is the greatest color that can be assigned to i. Thus, if $x_{ij21} = x_{ij22} = 0$ as well, node j cannot be colored with $2i-1$ nor with $2i$:

$$y_{j,2i-1} + y_{j,2i} \leq x_{ij21} + x_{ij22} + y_{i,2i} \quad \forall j > i : (i,j) \in E. \tag{4.29}$$

Following a similar idea and taking into account that $2i$ is the greatest color that can be assigned to i, we can devise the following constraints,

$$y_{j,2i} \leq x_{ij21} + x_{ij22} \quad \forall j > i : (i,j) \in E. \tag{4.30}$$

Recalling that symmetry breaking constraints in (H) impose that $y_{i,2i-1} = 1$ if $y_{i,2i} = 1$, we observe that if i and j share a color and i receives color $2i$ then j receives $2i-1$ or $2i$:

$$y_{i,2i} + \sum_{a=1}^{2} \sum_{b=1}^{2} x_{ijab} \leq 1 + y_{j,2i-1} + y_{j,2i} \quad \forall j > i : (i,j) \in E. \tag{4.31}$$

Finally, the following equation gathers "incompatible" variables in set packing constraints,

$$y_{j,2i} + y_{j,2j} + x_{ij11} + x_{ij12} \leq 1 \quad \forall j > i : (i,j) \in E. \tag{4.32}$$

4.5.2 A family of hybrid formulations

The first formulation in family (HM), named (HMa), consists of some constraints from (H) and (M) plus the linking constraints above described. The formulation is

$$
\begin{aligned}
\text{(HMa)}\ \max \quad & \sum_{(i,j)\in E}\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \\
\text{s.t.} \quad & (4.1), (4.2), (4.4)-(4.9), \\
& (4.18), (4.24)-(4.32) \\
& \sum_{r\in\mathcal{C}(i,j)} s_{ijr} = \sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \quad \forall (i,j)\in E && (4.33)\\
& y_{j,2j} + \sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \le 1 \quad \forall j > i : (i,j)\in E && (4.34)\\
& \sum_{j\in S}(x_{ija1} + x_{ija2}) \le 1 \quad \forall i, \forall S \subseteq N(i), S\in\mathcal{P}_G, |S|\in\{2,3\} \\
& \qquad\qquad \forall a\in\{1,2\} && (4.35)\\
& y_{ir}, s_{ijr'} \in \{0,1\} \quad \forall i\in V, \forall r\in\mathcal{C}(i), \forall (i,j)\in E, \forall r'\in\mathcal{C}(i,j) \\
& x_{ijab}\in\{0,1\} \quad \forall (i,j)\in E, \forall a,b\in\{1,2\}.
\end{aligned}
$$

Constraints (4.1), (4.2) and (4.4)-(4.9) integrate (H) in the formulation except for the fact that (4.3) are missing. These last constraints are replaced by (4.33), which are the result of combining (4.3) and (4.14).

Constraints (4.34) are a stronger version of (4.15) of (M), since one variable, $y_{j,2j}$, has been added to the left-hand side. Note that, if $y_{j,2j} = 1$, j is colored with $2j$ and $2j-1$. Since $j > i$, i and j will not share any color in this case, so $\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} = 0$. Constraints (4.35) are (4.19) when $|S| = 2$ or $|S| = 3$, which in particular include (4.10). If $|S_1| = 2$, $|S_2| = 3$ and $S_1 \subseteq S_2$, we would only consider (4.35) for S_2. Symmetry breaking constraints (4.18) of formulation (M) are also included in (HMa).

The reader may have noticed that constraints (4.11)-(4.13) of (M) have not been included in (HMa). Nevertheless, the formulation is valid since it somehow includes (H) —more precisely it includes the modification of (H) that results from eliminating variables d. There is room for wondering whether a solution of (HMa) is feasible for (M), though. In Proposition 4.1, we will show that every feasible solution of (HMa) satisfies constraints (4.11)-(4.13). The following lemma gives a partial result that will be crucial in the proof of Proposition 4.1.

Lemma 4.2. *Let $\{i,j,k\}\in\Gamma$ be a triangle in G and let $(\bar{y},\bar{s},\bar{x})$ be a feasible solution of (HMa). If $\bar{x}$ is such that one of the nodes in $\{i,j,k\}$ shares the same color with the other two then there exists $r\in\mathcal{C}$ such that $\bar{y}_{ir} = \bar{y}_{jr} = \bar{y}_{kr} = 1$.*

Proof. Suppose for instance that $\bar{x}_{jkac} = \bar{x}_{ikbc} = 1$, where $a,b,c\in\{1,2\}$. That is, node k shares its color in position c with both i and j. Recalling that nodes in the triangle are ordered, i.e., $i < j < k$, different assumptions can be made other than k sharing a color with i and j. Namely, it could be that $\bar{x}_{ijab} = \bar{x}_{jkbc} = 1$ or $\bar{x}_{ijab} = \bar{x}_{ikac} = 1$. We will prove the lemma when $\bar{x}_{jkac} = \bar{x}_{ikbc} = 1$ and overlook the other cases, which can be proven likewise.

Due to (4.33), $\exists r_1, r_2$ such that $\bar{s}_{ikr_1} = \bar{s}_{jkr_2} = 1$. Because of (4.5) and (4.6), we have that $\bar{y}_{ir_1} = \bar{y}_{kr_1} = \bar{y}_{jr_2} = \bar{y}_{kr_2} = 1$. For contradiction, suppose that $r_1 \neq r_2$. We distinguish the following cases:

1. $c = 1$ and $r_1 < r_2$.
   Take (4.25) applied to indices j, k, r_2:
   $$\bar{s}_{jkr_2} - \sum_{r'>r_2} \bar{y}_{jr'} - \sum_{r'>r_2} \bar{y}_{kr'} \leq \bar{x}_{jk22}.$$
   Since $\bar{s}_{jkr_2} = 1$, $\sum_{r'>r_2} \bar{y}_{kr'} = 0$ because $r_1 < r_2$ and $\bar{x}_{jk22} = 0$ because $c = 1$, the inequality above only stands if $\sum_{r'>r_2} \bar{y}_{jr'} = 1$. If we now consider (4.26) with same indices j, k, r_2:
   $$\sum_u \bar{s}_{jku} \leq \bar{x}_{jk12} + \sum_{r'<r_2} \bar{y}_{jr'} + \sum_{r'>r_2} \bar{y}_{kr'},$$
   it implies $\bar{x}_{jk12} = 1$, which is a contradiction.

2. $c = 1$ and $r_2 < r_1$.
   The proof is analogous to the previous case just by taking (4.25) and (4.26) with indices i, k, r_1 instead of j, k, r_2.

3. $c = 2$ and $r_1 < r_2$.
   Take (4.27) applied to indices i, k, r_1:
   $$\sum_u \bar{s}_{iku} \leq \bar{x}_{ik21} + \sum_{r'>r_1} \bar{y}_{ir'} + \sum_{r'<r_1} \bar{y}_{kr'}.$$
   Since $\sum_u \bar{s}_{iku} = 1$, $\bar{x}_{ik21} = 0$ due to $c = 2$ and $\sum_{r'<r_1} \bar{y}_{kr'} = 0$ because $r_1 < r_2$, the inequality above implies $1 = \sum_{r'>r_1} \bar{y}_{ir'}$. Now take (4.24) again with indices i, k, r_1:
   $$\bar{s}_{ikr_1} - \sum_{r'<r_1} \bar{y}_{ir'} - \sum_{r'<r_1} \bar{y}_{kr'} \leq \bar{x}_{ik11},$$
   where $\bar{s}_{ikr_1} = 1$ and the rest terms on the left-hand side are zero, which yields $\bar{x}_{ik11} = 1$. This contradicts initial assumption $c = 2$.

4. $c = 2$ and $r_2 < r_1$.
   The proof is analogous to the previous case just by taking (4.27) and (4.24) with indices j, k, r_2 instead of i, k, r_1.

All the cases lead to a contradiction and hence we have found $r := r_1 = r_2$ as desired. □

Proposition 4.1. *Let $(\bar{y}, \bar{s}, \bar{x})$ be a feasible solution to (HMa). Then $\bar{x}$ satisfy transitivity constraints (4.11)-(4.13).*

Proof. Let $\{i, j, k\} \in \Gamma$ be a triangle in G. We have to prove that the coloring of nodes i, j, k encoded by $\bar{x}$ is transitive. That is, if, according to the solution, one of the nodes in the triangle $\{i, j, k\}$ shares the same color with the other two, we have to prove that the latter share that color as well. Lemma 4.2 ensures that $\bar{y}_{ir} = \bar{y}_{jr} = \bar{y}_{kr} = 1$ for some color r.

Given that $\bar{y}_{ir} = \bar{y}_{jr} = \bar{y}_{kr} = 1$, $\bar{s}_{ijr} = \bar{s}_{jkr} = \bar{s}_{ikr} = 1$ follows from (4.4). Now, we can define $Y^+_{ir} = \sum_{r'>r} \bar{y}_{ir'}$ and $Y^-_{ir} = \sum_{r'<r} \bar{y}_{ir'}$, for all $i \in V, r \in \mathcal{C}(i)$. Constraints (4.24)- (4.26) for the nodes of our triangle $\{i, j, k\}$ are then:

$$\begin{array}{llll|llll|llll}
1 \leq & x_{ij11} + & Y^-_{ir} + & Y^-_{jr} & 1 \leq & x_{ik11} + & Y^-_{ir} + & Y^-_{kr} & 1 \leq & x_{jk11} + & Y^-_{jr} + & Y^-_{kr} \\
1 \leq & x_{ij22} + & Y^+_{ir} + & Y^+_{jr} & 1 \leq & x_{ik22} + & Y^+_{ir} + & Y^+_{kr} & 1 \leq & x_{jk22} + & Y^+_{jr} + & Y^+_{kr} \\
1 \leq & x_{ij21} + & Y^+_{ir} + & Y^-_{jr} & 1 \leq & x_{ik21} + & Y^+_{ir} + & Y^-_{kr} & 1 \leq & x_{jk21} + & Y^+_{jr} + & Y^-_{kr} \\
1 \leq & x_{ij12} + & Y^-_{ir} + & Y^+_{jr} & 1 \leq & x_{ik12} + & Y^-_{ir} + & Y^+_{kr} & 1 \leq & x_{jk12} + & Y^-_{jr} + & Y^+_{kr}
\end{array}$$

Since $\sum_{r\in\mathcal{C}(i)} \bar{y}_{ir} = 2$, it is straightforward that either $Y^+_{ir} = 1$ (and $Y^-_{ir} = 0$) or $Y^-_{ir} = 1$ (and $Y^+_{ir} = 0$), and the same is true for j and k. Consider the group of rewritten constraints for i, j, which are on the first column. In every solution, only one of the four constraints has null Y-terms, which will indicate which of the x_{ij}-variables takes value one. The same happens for the second and third columns of constraints. It is easy to check that any of the nine possibilities regarding sums $Y^{+,-}_{ir}, Y^{+,-}_{jr}, Y^{+,-}_{kr}$ yield x-variables that respect transitivity in the triangle $\{i, j, k\}$. For instance, suppose that $Y^+_{ir} = 1$, $Y^-_{jr} = 1$ and $Y^-_{kr} = 1$. Then, $\bar{x}_{ij12} = 1$, $\bar{x}_{ik12} = 1$ and $\bar{x}_{jk22} = 1$ yield respectively from the fourth constraints in the first and second columns and the second constraint in the third column. These values respect transitivity constraints (4.11)-(4.13). □

Proposition 4.1 is not true for fractional solutions to (HMa). This means that including (4.11)-(4.13) in (HMa) could produce a formulation with stronger LP bounds than that of (HMa). Nevertheless, computational experience shows that the LP bound of (HMa) is usually not affected when (4.11)-(4.13) are included to the formulation. When this is not the case, the improvement of the bound with respect to plain (HMa) is insignificant.

We noticed during the computational tests that running times of (HMa) can be significantly reduced with little detriment in the LP bound, by removing some constraints. After a few tests, we decided to consider as well the two following modifications of the initial combined formulation.

$$
\begin{aligned}
\text{(HMb)}\ \max\quad & \sum_{(i,j)\in E}\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \\
\text{s.t.}\quad & (4.1), (4.2), (4.4)-(4.9), (4.18), \\
& (4.33)-(4.35), (4.28-4.32) \\
& y_{i,r}, s_{ijr'}, x_{ijab} \in \{0,1\} \qquad \forall i \in V, \forall r \in \mathcal{C}_i, \forall (i,j) \in E, \\
& \qquad\qquad\qquad\qquad\quad \forall r' \in \mathcal{C}(i,j), \forall a, b \in \{1,2\}
\end{aligned}
$$

$$
\begin{aligned}
\text{(HMc)}\ \max\quad & \sum_{(i,j)\in E}\sum_{a=1}^{2}\sum_{b=1}^{2} x_{ijab} \\
\text{s.t.}\quad & (4.1), (4.2), (4.4)-(4.9), (4.18), \\
& (4.33)-(4.35), (4.30) \\
& y_{i,r}, s_{ijr'}, x_{ijab} \in \{0,1\} \qquad \forall i \in V, \forall r \in \mathcal{C}_i, \forall (i,j) \in E, \\
& \qquad\qquad\qquad\qquad\quad \forall r' \in \mathcal{C}(i,j), \forall a, b \in \{1,2\}.
\end{aligned}
$$

Note that, due to the deletion of constraints, we can not guarantee that an optimal solution of (HMb) or (HMc) is going to be feasible for (M). Nevertheless, we can ensure that variables y codify the optimal solution. This is because the formulation that results after removing variables d from (H) is embedded in both (HMb) and (HMc), which is given by constraints (4.1), (4.2), (4.4)-(4.9), (4.33) and 4.34. Thus, in the case of (HMb) and (HMc), variables x are just a tool to produce a tighter and therefore more efficient formulation, as the computational experience reported in the next section proves.

	Fork ineq.	Sim. breaking	Ineq. (4.22) and (4.23)
(M)	No		No
(M1)	No		Yes
(M2)	No	(4.18)	No
(M3)	No	(4.18)	Yes
(M4)	No	(4.16), (4.17)	No
(M5)	No	(4.16), (4.17)	Yes
(M6)	Yes		No
(M7)	Yes		Yes
(M8)	Yes	(4.18)	No
(M9)	Yes	(4.18)	Yes
(M10)	Yes	(4.16), (4.17)	No
(M11)	Yes	(4.16), (4.17)	Yes

Table 4.1: Configurations used in the preliminary study

4.6 Computational experiments

The aim of our computational study is twofold. First, we would like to study the effect of the valid inequalities of Section 4.4 on (M), in order to determine whether it is convenient to add them (or a subset of them) to the model. Second, we will compare the resulting best configurations for (M) with (H), (HMa), (HMb) and (HMc).

The processor used for the tests was an Intel core i7-6700k CPU at 4.0 GHz $\times$ 8 with 16 GB of RAM memory. The solver was Cplex v12.6.3 64-bit under operating system Linux Ubuntu 16.04. The testbed consisted of the following four families of instances, which we have generated ourselves except for the last one.

1. Forbidden graphs: have been generated by combining some of the forbidden subgraphs of Beineke (1970) several times and randomly adding more edges.

2. Random graphs: graphs randomly generated with different number of nodes and densities.

3. Perturbed line graphs: we use a graph as starting point, generate its line graph and then we randomly add edges to it, until a maximum number of extra edges is reached. We use the resulting graph as input of the problem. With this procedure we obtain a bound on the optimal value (the number of edges added).

4. HapMap graphs: these instances have been provided by Hálldorsson and were used in Halldórsson et al. (2013). They codify information from a sample of 77 individuals taken from the populations collected by The International HapMap Consortium (2010).

4.6.1 Preliminary study

In the first part of our computational study, we have used (M) with different combinations of the inequalities of Section 4.4:

- Either with *fork inequalities* (4.19) with $|S| = 3$ or without them.

- Three possibilities regarding symmetry breaking: without constraints, with (4.16) and (4.17) or with (4.18),

- Either with valid inequalities (4.22) and (4.23) or without them.

File	n	m	den.%	OPT	LP OPT												
					H	M	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11
Forb1	20	100	53	43	11.5	0.0	0.0	8.5	8.5	8.5	8.5	32.5	32.5	**34.6**	**34.6**	34.4	34.4
Forb2	18	87	57	33	11.3	0.0	0.0	8.0	8.0	8.0	8.0	26.8	26.8	**27.7**	**27.7**	**27.7**	**27.7**
Forb3	20	110	58	43	12.0	0.0	0.0	9.0	9.0	9.0	9.0	35.5	35.5	**37.2**	**37.2**	**37.2**	**37.2**
Forb4	32	145	29	61	40.0	0.0	0.0	14.5	14.5	14.5	14.5	47.9	47.9	**52.0**	**52.0**	51.6	51.6
Forb5	21	116	55	45	18.4	0.0	0.0	8.8	8.9	8.8	8.9	37.2	37.2	**38.9**	**38.9**	38.8	38.8
Rand1	100	498	10	326	**284.0**	0.0	0.0	47.2	47.2	47.2	47.2	166.0	166.0	197.0	197.0	197.0	197.0
Rand2	20	122	64	42	13.4	0.0	0.0	9.5	9.5	9.5	9.5	34.5	34.5	**35.3**	**35.3**	35.2	35.2
Rand3	40	254	33	148	92.2	0.0	0.0	19.0	19.0	19.0	19.0	84.7	84.7	**96.7**	**96.7**	**96.7**	**96.7**
Rand4	50	256	21	150	**106.0**	0.0	0.0	23.5	23.5	23.5	23.5	85.3	85.3	100.0	100.0	100.0	100.0
Rand5	60	367	21	228	**169.0**	0.0	0.0	28.5	28.5	28.5	28.5	122.0	122.0	141.0	141.0	141.0	141.0
Pert1	86	957	26	45	41.0	0.0	0.0	23.0	25.3	23.0	25.3	**45.0**	**45.0**	**45.0**	**45.0**	**45.0**	**45.0**
Pert2	143	2004	20	72	**72.0**	0.0	0.0	36.0	40.4	36.0	40.2	**72.0**	**72.0**	**72.0**	**72.0**	**72.0**	**72.0**
Pert3	137	1851	20	69	**69.0**	0.0	0.0	34.0	38.6	34.0	38.5	**69.0**	**69.0**	**69.0**	**69.0**	**69.0**	**69.0**
Pert4	191	2461	14	102	99.0	0.0	0.0	48.0	49.2	48.0	49.2	**102.0**	**102.0**	**102.0**	**102.0**	**102.0**	**102.0**
Pert5	271	2550	7	136	135.0	0.0	0.0	74.2	74.8	74.2	74.8	**136.0**	**136.0**	**136.0**	**136.0**	**136.0**	**136.0**
Hap1	77	313	11	161	**114.0**	0.0	0.0	22.2	22.2	22.2	22.2	98.7	98.7	107.0	107.0	107.0	107.0
Hap2	77	253	9	130	83.0	0.0	0.0	21.0	21.0	21.0	21.0	83.5	83.5	**90.8**	**90.8**	90.6	90.6
Hap3	77	357	12	188	99.0	0.0	0.0	22.5	22.5	22.5	22.5	118.0	118.0	**128.0**	**128.0**	**128.0**	**128.0**
Hap4	77	338	12	146	91.0	0.0	0.0	23.7	23.7	23.7	23.7	107.0	107.0	**113.0**	**113.0**	112.0	112.0
Hap5	77	308	11	152	102.0	0.0	0.0	27.8	27.8	27.8	27.8	102.0	102.0	**112.0**	**112.0**	111.0	111.0

Table 4.2: Preliminary study results

After combining the options above, we obtain twelve configurations, which we call (M), (M1),..., (M11) (see Table 4.1). We perform a preliminary study in order to find the best configurations among the twelve. Here we use five instances from each of the four databases, which makes a total of 20 instances. We fix a time limit of 300 seconds, disable Cplex dynamic search to force Cplex to use traditional branch-and-cut strategy, and execute each experiment with and without Cplex cuts.

Table 4.2 summarizes some results of this preliminary study. In the first five columns we can see the name of the instances (File), the number of nodes (n), the number of edges (m), the density as a percentage (den.%) and the optimal value for the EDPL (OPT). The following columns show the optimal value of the linear relaxation of H, M, M1,...,M11. As we mentioned in the previous section, there are instances for which H has the best LP bound, while our formulations are better in this respect for some others (best LP bounds are in bold).

Table 4.3 gathers the running times obtained in this preliminary study when Cplex cuts are activated (default) and when we disable them. It can be observed that the formulations we propose obtain satisfactory results in comparison to (H), except for the third data set. The configurations that obtained the best running times were (M4), (M8) and (M10).

4.6.2 Main computational study

The aim of the main study is to compare Halldórsson et al.'s formulation (H), three variants of our proposed formulation, (M4), (M8) and (M10), and mixed formulations (HMa), (HMb) and (HMc). For the experiments, we set a time limit of 900 seconds and disable Cplex dynamic search and cuts. We use a testbed of 82 files: 23 forbidden graphs, 16 random graphs, 21 perturbed line graphs and 22 HapMap graphs. For the sake of clarity, all the diagrams and charts reported here refer to the best configurations found among our proposals, which turn out to be (M8), (HMb) and (HMc).

Figure 4.6 shows the percentage of instances (ordinate axis) that were solved after a certain number of seconds (abscissa axis). Overall, (HMc) obtains the highest rates. (H) barely solved 60% of the instances within the time limit, while (M8), (HMb) and (HMc) could solve between 80% and 90%. The biggest step that separates (HMc) from the other models on the chart is placed between 30 and 50 seconds, being (HMc) able to solve more than half of the instances within 50 seconds. Figure 4.7 illustrates a similar comparison but in terms of the number of nodes of the branching tree explored instead of the running time. Note that this chart

File	Running time with/without CPLEX cuts																							
	H		M		M1		M2		M3		M4		M5		M6		M7		M8		M9		M10		M11	
Forb1	8.7	60.8	10.4	11.0	175.1	285.4	**3.1**	**5.8**	63.6	132.6	7.9	8.3	118.4	210.7	14.7	7.0	300.0	287.0	5.7	**5.2**	70.5	75.5	5.5	7.0	104.1	260.1
Forb2	4.0	5.0	6.9	4.4	123.4	102.1	**2.7**	**1.8**	23.3	31.6	5.8	**2.2**	38.4	38.3	7.0	3.5	127.6	85.7	**2.4**	**2.1**	31.3	23.6	4.8	**2.3**	48.6	31.3
Forb3	15.5	63.2	14.8	10.4	300.0	300.0	9.6	7.6	160.5	201.9	**8.8**	**6.0**	120.1	119.1	13.7	11.4	300.0	300.0	10.4	**6.4**	96.3	197.3	9.5	9.4	122.2	203.6
Forb4	1.6	1.9	5.5	4.2	89.7	58.4	2.6	**0.5**	12.0	12.3	**0.8**	**0.5**	12.1	7.2	2.9	2.7	34.6	52.0	2.9	1.4	12.9	19.9	1.6	2.0	10.4	16.6
Forb5	11.5	17.3	13.1	9.5	242.4	277.3	4.3	7.2	39.1	75.3	**3.4**	**3.8**	52.5	229.9	11.3	6.2	265.1	240.8	4.2	7.1	22.9	61.7	4.1	**4.1**	25.0	118.4
Rand1	**1.2**	119.8	2.5	6.9	5.8	19.4	1.5	7.7	3.5	19.3	**1.4**	3.4	2.2	3.9	**1.4**	6.7	4.7	5.0	2.0	**2.4**	4.0	11.3	**1.1**	5.4	2.0	2.8
Rand2	66.3	175.6	235.8	65.8	300.0	300.1	69.1	89.9	300.0	300.0	175.6	52.4	300.0	300.0	181.9	**40.8**	300.9	300.0	**44.5**	55.8	300.0	300.2	300.0	51.0	300.0	300.0
Rand3	36.6	301.4	48.2	206.3	300.0	300.0	49.4	141.5	300.1	300.1	**29.7**	300.0	300.0	300.1	56.9	132.1	300.0	300.0	75.5	131.3	300.0	300.0	38.3	**72.5**	300.0	300.0
Rand4	**1.3**	21.1	**1.4**	4.0	8.9	29.1	**1.2**	1.1	4.7	5.5	1.2	**0.3**	2.7	1.1	1.4	**0.7**	5.0	12.5	0.9	**0.5**	2.8	1.1	1.1	**0.4**	5.1	5.2
Rand5	8.8	301.1	16.6	170.7	300.0	300.0	9.5	61.5	34.4	300.0	6.7	71.4	50.6	300.0	32.2	34.8	300.1	300.0	**5.3**	**11.7**	37.2	228.8	8.8	14.7	46.5	300.0
Pert1	**4.4**	**8.3**	300.1	300.0	300.2	300.2	16.5	16.0	100.1	99.8	13.4	13.3	89.8	92.4	300.1	300.0	300.2	300.2	17.0	16.0	94.3	108.1	14.0	13.7	88.7	91.2
Pert2	**12.4**	**16.4**	300.1	300.1	300.3	300.3	108.4	109.8	300.3	300.3	86.0	86.4	300.3	300.3	300.1	300.1	300.3	300.3	109.4	103.6	300.3	300.3	89.1	89.0	300.3	300.3
Pert3	**3.4**	**3.3**	300.1	300.1	300.2	300.3	79.4	80.8	300.3	300.3	66.7	66.9	300.2	300.2	300.1	300.1	300.5	300.5	82.9	82.9	300.3	300.2	69.1	69.2	300.3	300.2
Pert4	**2.7**	**8.6**	300.1	300.1	300.2	300.2	135.8	132.2	300.2	300.2	126.6	127.3	300.2	300.2	300.1	300.1	300.3	300.2	142.1	140.9	300.2	300.2	118.5	118.9	300.2	300.2
Pert5	**1.8**	**3.3**	300.1	300.1	300.2	300.1	81.1	80.9	162.5	163.4	63.6	63.2	138.9	144.8	300.1	300.1	300.2	300.1	86.9	85.2	166.8	166.6	66.0	66.9	139.9	139.7
Hap1	**10.0**	280.1	44.5	50.6	300.0	300.0	18.8	172.2	300.1	300.1	36.4	300.0	300.0	300.1	43.8	69.1	300.0	300.0	29.8	**33.6**	300.1	300.1	46.3	95.2	300.0	300.0
Hap2	4.2	22.7	8.7	9.2	245.6	226.5	**3.5**	8.3	26.8	50.5	4.3	7.4	68.4	168.5	**3.8**	8.0	20.2	209.5	**3.2**	7.4	64.1	35.4	4.0	**5.9**	20.4	116.9
Hap3	**29.7**	301.7	123.8	136.4	300.0	300.0	47.0	92.7	300.1	300.1	300.0	299.2	300.0	300.0	116.6	157.5	300.0	300.1	39.9	**69.0**	300.0	300.1	300.0	300.0	300.0	300.1
Hap4	13.7	120.2	27.6	59.0	300.0	300.0	**10.6**	38.4	227.9	300.0	50.4	133.8	300.0	300.7	89.1	51.2	300.0	300.0	13.9	**19.5**	95.2	300.1	35.5	167.2	300.0	300.1
Hap5	**6.5**	184.3	18.8	25.0	300.0	300.0	**6.4**	21.6	160.3	300.0	10.3	31.7	247.8	300.0	19.8	33.5	300.0	300.0	10.0	**13.2**	283.7	300.0	9.9	**12.8**	129.2	300.0

Table 4.3: Running times of preliminary study

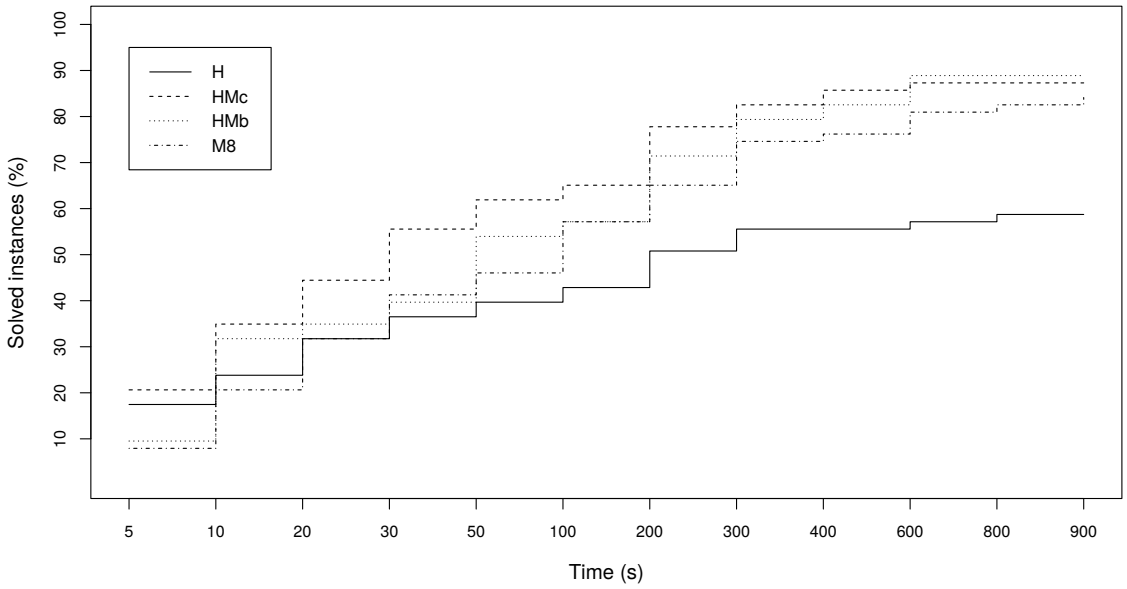


Figure 4.6: Percentage of solved instances after the seconds shown on the abscissa axis (this axis is not scaled)

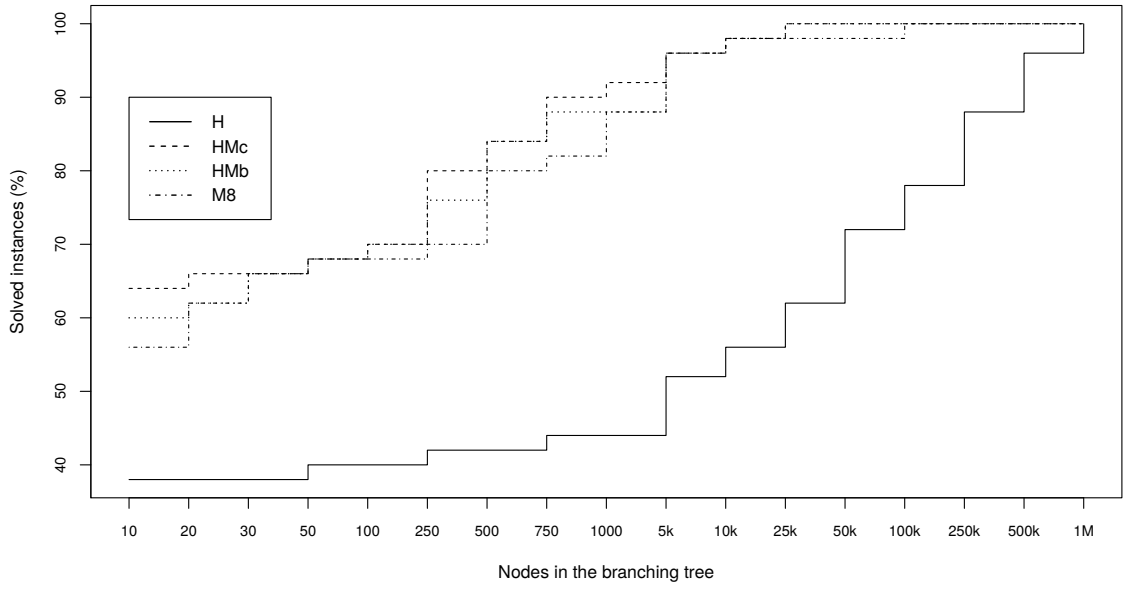


Figure 4.7: Percentage of solved instances after exploring as many nodes as on the abscissa axis (this axis is not scaled)

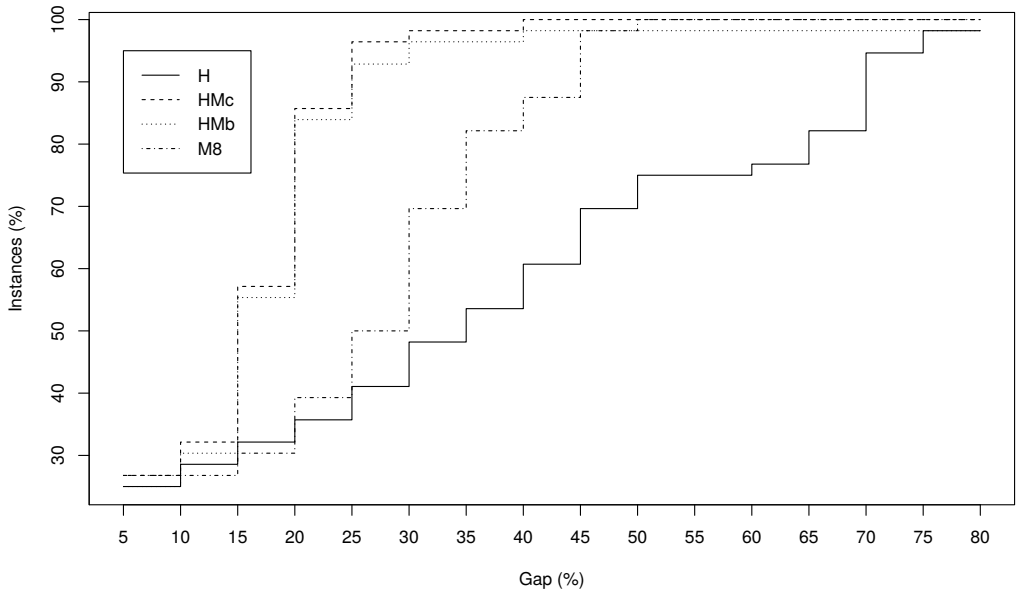


Figure 4.8: Percentage of instances with a gap smaller than or equal to the gap shown on the abscissa axis

only considers instances that all the compared models could solve (the comparison is pointless otherwise). The difference on the overall performance of (H) and the rest of the models is huge. While (H) solves less than half of the instances after exploring 5000 nodes, (HMc) solves more than 90%. Up to 5000 nodes (HMc) is the overall winner, and (M8) and (HMb) reach its performance from 5000 nodes on. Finally, (H) needs more than $500,000$ nodes to solve all the instances. The last chart, depicted on Figure 4.8, shows the percentage of instances (ordinate axis) that has a LP gap smaller than or equal to a certain limit (abscissa axis). The LP gap gives the distance between the optimal value of the EDPL and the optimal value of its relaxation, relative to the optimal value of the problem. Mixed models (HMb) and (HMc) show the best overall trends, as they were meant to. (M8) also shows a good performance; the number of instances within a given gap grow rapidly when the gap limit increases by steps of five units. The chart shows that (HMc) has the overall tightest LP gap, which is smaller than or equal to 45% for all the instances.

Figures 4.9 and 4.10 depict some statistical information about the running times of the experiments. Bars on Figure 4.9 show the average running times for each model grouped by data set. Although we observe that (H) has by far the tallest bars in general, this is not the case for the perturbed line graphs data set, for which it clearly has the smallest running time. As regards the other data sets, (HMc) seems to be a good alternative.

Figure 4.10 is a box-and-whisker diagram, which is also grouped by data sets. The bottom and top of the boxes are the 25th and 75th percentile of the running times obtained, and the band inside them is the median. The end of the whiskers are:

$$\text{upper whisker} = \min(\max_k\{RT_k\}, Q_3 + 1.5 \cdot IQR)$$

$$\text{lower whisker} = \max(\min_k\{RT_k\}, Q_1 - 1.5 \cdot IQR)$$

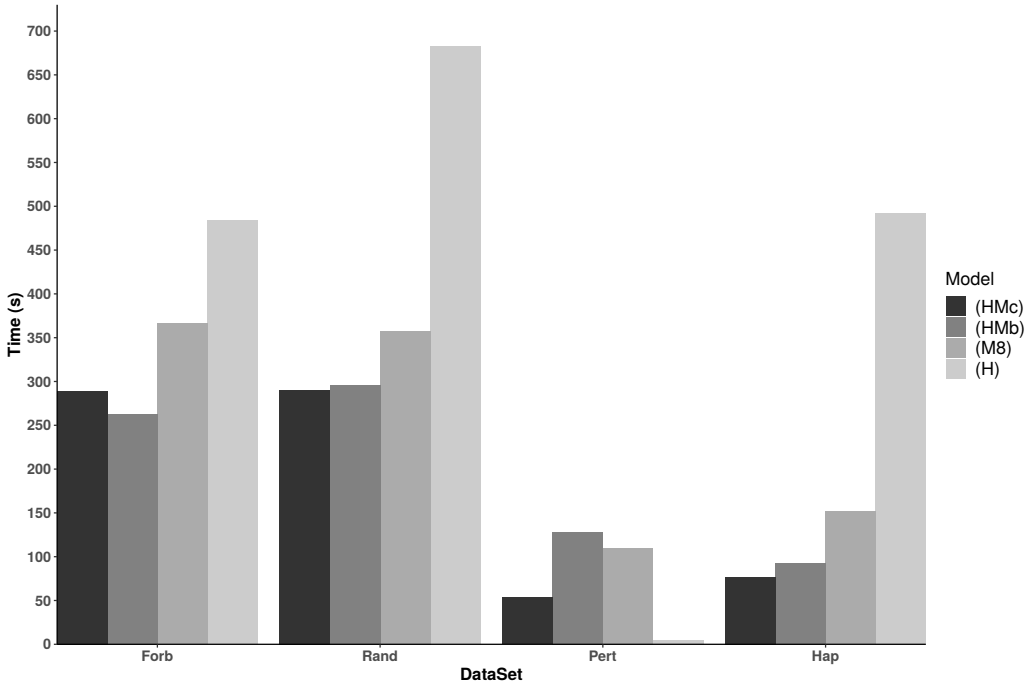


Figure 4.9: Average running time for the different models and data sets

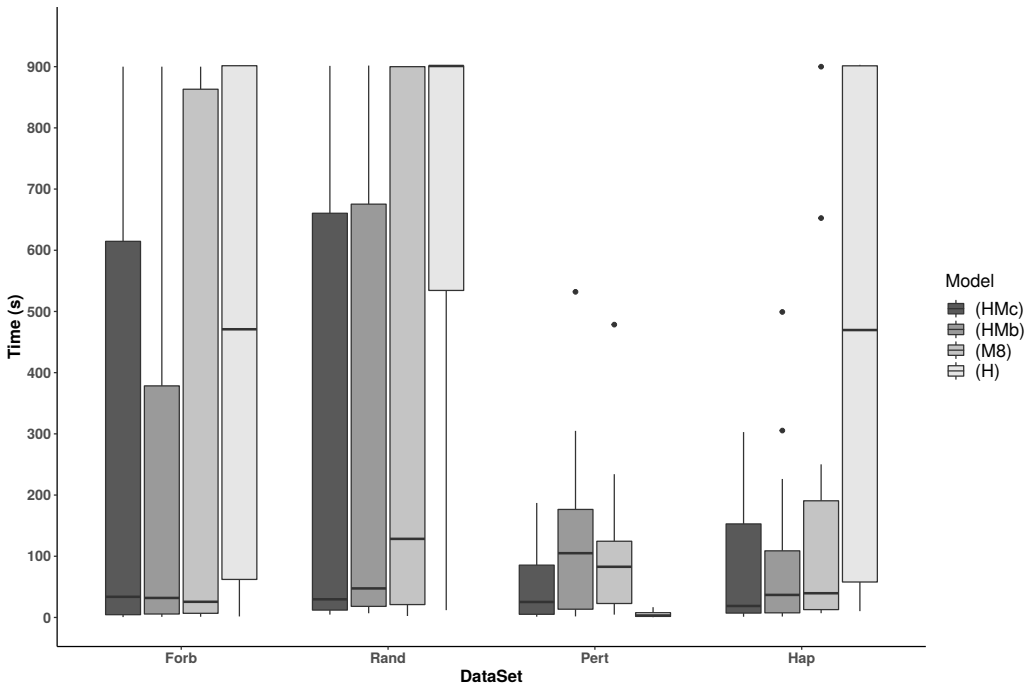


Figure 4.10: Box-and-whisker diagrams representing the running time for the different models and data sets

where $\{RT_k\}$ is the sample of running times, Q_1 and Q_3 are the first and third quartiles and $IQR = Q_3 - Q_1$ is the box length. Finally, dots on the chart are outliers, which fall out of the whiskers limits. As opposed to the average, this kind of diagram allows us to observe the spread of values. For example, (H) presents the largest interquartile range for forbidden and HapMap graphs, which is almost equal to its overall range. This indicates that running times are highly dispersed between 0 and the time limit; the fact that Q_3 is nearly 900 reveals that around 25% of the instances were not solved within the time limit. In the case of random graphs, such percentage increases to 50% at least. Moreover, (H) solved the first 25% of the instances within 550 seconds (Q_1), as opposed to the rest of the formulations which only took few seconds. Conversely, (H) is really effective solving perturbed line graphs. Its tight box-and-whisker shows that it solved all the instances of this dataset within few seconds. As for (M8), it features highly dispersed running times for forbidden and random graphs. Although (M8) is rather fast for half of the complete testbed (see the illustration of the median for each dataset, which remains low), it is never the alternative with lowest measure Q_1, median or Q_3. In general, (M8), (HMb) and (HMc) present medians closest to the bottom of their boxes. This reveals an asymmetry in the distribution of the running times, which is denser for smaller values.

As a conclusion, we can say that (HMb) and (HMc) seem the best alternatives for all the datasets but perturbed line graphs. The reason might be the structure of the target graph, $G = (V, E)$. In these instances, G is built as follows. We start from a graph, obtain its line graph and finally add some edges to the latter. The resulting graph, G, has a strong structure where nodes (edges of the original graph) are clustered according to the end-nodes that they shared, which is not disrupt after adding some edges. This structure is inherited by the variables and constraints of (H), which are defined for each edge of G (except for constraints (4.2)). Conversely, constraints of (M) are defined in terms of two structures: triangles and triplets $\{i, j, k\}$ such that $(i, j), (i, k) \in E$ and $(j, k) \notin E$. This might lead to a model with no structure and therefore harder to decompose and solve. In other words, (H) allows to exploit the structure of $L(G)$ better than (M) in this case. Regarding the other datasets, average running times on Figure 4.10 suggested the use of (HMc), but Figure 4.10 shows that the running times of (HMb) are in general less dispersed.

Chapter 5

Optimal unambiguous map labeling

Map design, which has been performed by cartographers for ages, is laborious and time-consuming. As part of the methodology for map production, several fundamental subtasks take place. First of all, the area to model must be delimited and projected on a flat medium. Then, non-relevant characteristics should be identified according to some criteria, e.g. the targeted audience, to be grouped or removed from the map. Finally, some descriptive information has to be included. Such information is usually associated to the main features depicted and comes in the shape of labels that have to be orchestrated on the map. The problem of map labeling consists of deciding on the location of labels on the map so that some criteria regarding legibility are optimized.

In the Digital Age, there is an increasing development of systems that replace or help humans with hard tasks, including that of map production. In this context, most approaches to map labeling so far developed are heuristics. Among them, simulated annealing is a frequent choice and has been proved to produce better solutions than other heuristic algorithms (see for instance Christensen, Marks & Shieber, 1995; Rylov & Reimer, 2014). Existing mathematical programming approaches range from most simplified models as proposed by Zoraster (1990) to multiobjective in Haunert & Wolff (2016), and include a PhD thesis devoted to orthogonal placement problems, see Klau (2001). An extended review can be found in the survey by Neyer (2003) or in the map-labeling on-line bibliography managed by Wolff & Strijk (2009).

We approach map labeling with especial focus on ambiguity reduction. Classic quality indicators for map labeling are dispersion, reduced ambiguity, features priorities, balanced distribution or consideration of hierarchy of objects. Among them, ambiguity, together with overlapping labels, is one of the main obstacles for a correct map interpretation. To the best of our knowledge, the problem of producing not very ambiguous maps has not been effectively tackled yet. The only models in the literature that consider ambiguity are the heuristic of Rylov & Reimer (2014) and the IP formulation of Haunert & Wolff (2016).

As opposed to previous studies that evaluated labels ambiguity by pairs, we propose to measure the ambiguity of labels individually. We base our approach on the premise that proximity of labels to unrelated features is what makes them ambiguous. Our main contributions are four integer linear programming formulations, which incorporate the proposed ambiguity measure and also address overlaps between labels. As an alternative for the formulations, we develop a heuristic procedure to deal with large instances. Finally, computational experiments made to evaluate the proposed formulations and heuristic are presented. Implications of using the different models as well as parameters tuning recommendations are also discussed.

The problem studied in this chapter is closely related to the topics addressed in the first part of the thesis. First, labels overlaps avoidance is modeled with set packing constraints.

Second, placing labels on point features very much resembles discrete location problems. We will take inspiration from discrete ordered models in locational analysis to propose some of the new models, which penalize ambiguous labels in a flexible manner.

5.1 Label placement

A fundamental problem within Cartography is to associate labels to the main features depicted, usually known as map labeling or label placement. Automation of map labeling consists of finding techniques to produce a good label arrangement, as a cartographer would do (see e.g. Imhof, 1975; Dent, 1996; Robinson, 1958). According to Edmondson et al. (1996), label placement involves three independent subtasks, namely

1. *Candidate-position generation*: Given a feature, identify a set of candidate locations for its label. Feature types were identified by Imhof (1975) as points, lines or areas.

2. *Position selection*: Given the sets of candidate locations identified in previous subtask, select one label position for each feature. The overall quality of the labeling produced, which is determined by next subtask, should be as high as possible.

3. *Position evaluation*: Given an arrangement of the labels, compute a score that indicates its quality.

Depending on the strategy followed to perform each subtask, different models for the automation of label placement may arise.

In the candidate-position generation phase, we find various alternatives regarding some technical modeling aspects. Some authors consider continuous candidate sets, e.g. "sliding" models like the one by Van Kreveld et al. (1999), "elastic labeling" tackled by Iturriaga & Lubiw (1997) or "scaling" models like that proposed by Been et al. (2010). In contrast, there are discrete models or "fixed-position models", see for instance Klau & Mutzel (2003). Different shapes and orientations of the labels might also be considered for candidate sets. In this same phase, there are two alternative approaches concerning geographical aspects. A first strategy is to select candidate locations in a more or less arbitrary way to make a nice map, regardless of the geographical places actually occupied by features. Paradigmatic examples are metro maps (see e.g. Nöllemburg & Wolff, 2011), where stations are organized in straight lines linked by means of 45- or 90-degree angles, and consecutive stations are at the same distance. The elimination of geographical restrictions in this phase overcomes obstacles like label overlaps, making the next subtasks of position selection and evaluation less meaningful. A second option is to choose candidate locations in agreement with the actual geographical position of features. In this case, position selection and evaluation become challenging.

Most of the attempts to address label position selection assume that a discrete set of location candidates has already been produced in the generation phase. Then, position selection reduces to labeling point features, which will be also the starting point for the approaches presented on this chapter. However, placing the label of each feature somewhere around is not an easy task even if the position candidate sets are given. Naively placed labels will overlap in most cases. In addition, they can produce ambiguity due to their location with respect to features. Position selection then requires optimization. Different implementations of this phase include exact methods and heuristic strategies, with the latter being far more common.

Finally, for the definition of quality that guides the position evaluation phase, we find a wide range of alternatives. An initial rough classification of label-placement models in the literature

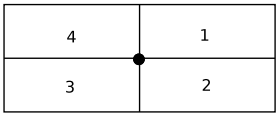


Figure 5.1: Locations of four labels with respect to the point labeled

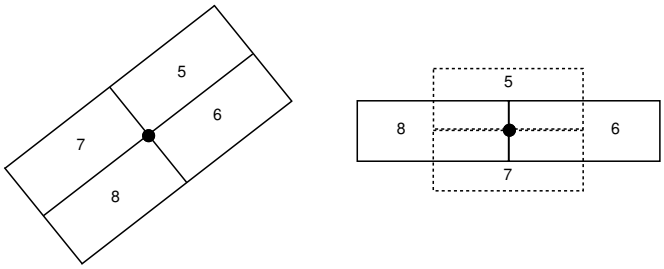


Figure 5.2: Two alternative locations of a label with respect to its point feature

would distinguish between those that allow non-labeled features and those that do not. The former usually forbid overlaps between labels and aim to maximize the number of features labeled, possibly taking into account other quality indicators, such as dispersion, ambiguity, or consideration of hierarchy of objects, see Imhof (1975). The latter allow overlaps between labels and may consider the total overlapping area as an additional quality indicator.

5.2 Problem setup

Consider $I = \{1, \ldots, n\}$ a given set of point features in the plane with coordinates (α_i, β_i), $i \in I$ and let $L = \cup_{i \in I} L_i$ be the discrete set of labels position candidates. Each point feature $i \in I$ has a set of potential labels, L_i, which contains different translations of its name or description. We assume $L_i \cap L_{i'} = \emptyset \ \forall i \neq i'$, i.e., each label corresponds to only one feature. Let $A = (a_{j\ell})_{|L| \times |L|}$ be a binary symmetric matrix with $a_{j\ell} = 1$ iff labels j and ℓ either overlap or belong to the same set L_i. We define

$$E := \{(j, \ell) \in L \times L : \ j < \ell, \ |\{j, \ell\} \cap L_i| \leq 1 \ \forall i \in I, \ a_{j\ell} = 1\},$$

the set of pairs of overlapping labels corresponding with different point features. There will be pairs of features at a relative distance such that their labels will never be in conflict. To focus on the conflicting cases, we also define

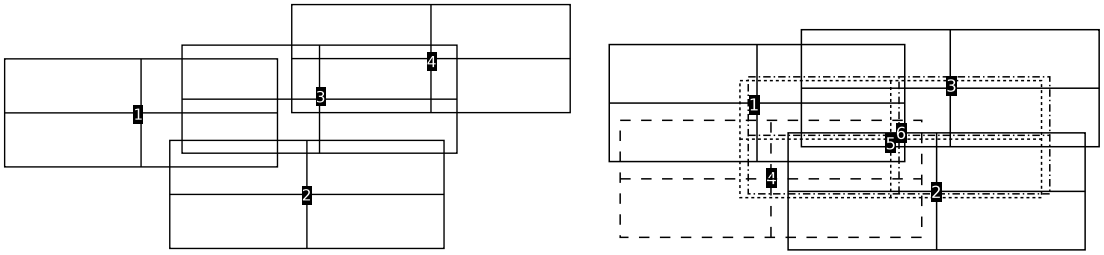
$$H := \{(i, i') \in I \times I : \ i < i', \ \sum_{j \in L_i} \sum_{\ell \in L_{i'}} a_{j\ell} \geq 1\},$$

the set of pairs of point features whose labels could overlap, which will be identified as *conflicting points*.

There are some particular configurations within this general framework that have been frequently addressed in the literature. The most common assumption is that of four rectangular labels per point (see Figure 5.1). In this particular case $L_i = L_i^4$,

$$L_i^4 := \{(i, 1), (i, 2), (i, 3), (i, 4)\} \qquad \forall i \in I,$$

where (i, k), $k = 1, \ldots, 4$, represent four rectangles with opposite vertices $\{(\alpha_i, \beta_i), (\alpha_i + w_i, \beta_i + h_i)\}$, $\{(\alpha_i, \beta_i), (\alpha_i + w_i, \beta_i - h_i)\}$, $\{(\alpha_i, \beta_i), (\alpha_i - w_i, \beta_i - h_i)\}$ and $\{(\alpha_i, \beta_i), (\alpha_i - w_i, \beta_i + h_i)\}$



(a) Four points with their candidate labels

(b) Six points with their candidate labels

Figure 5.3: Two point feature labeling instances with four candidate rectangular labels

respectively, for a given width w_i and a given height h_i associated to each point $i \in I$. We call $L^4 := \cup_{i \in I} L_i^4$. Some settings consider L^4 plus other four rectangular labels, which are usually displayed in one of the two alternative configurations shown by Figure 5.2.

The following example illustrates the configuration described by Figure 5.1 for four and six point features and constant label width and height.

Example 5.1. Figure 5.3a depicts an instance with four point features and four candidate labels each. Point features are numbered from 1 to 4, having coordinates $(\alpha_1, \beta_1) = (10, 10)$, $(\alpha_2, \beta_2) = (22, 4)$, $(\alpha_3, \beta_3) = (23, 11)$ and $(\alpha_4, \beta_4) = (31, 14)$, respectively. Labels are 10×4 rectangles, i.e., $w_i = 10$ and $h_i = 4$ for all $i \in I = \{1, 2, 3, 4\}$. Taking, for instance, $i = 2$, we have that its candidate labels are $L_2^4 = \{(2, 1), \ldots, (2, 4)\}$, with e.g. $(2, 1)$ representing rectangle with opposite vertices in $(\alpha_2, \beta_2) = (22, 4)$ and $(\alpha_i + w_i, \beta_i + h_i) = (32, 8)$, that is to say, the label at the top-right of point 2.

The set of conflicting points is

$$H = \{(1, 2), (1, 3), (2, 3), (3, 4)\},$$

while overlapping labels are

$$\begin{aligned} E = \; & \{((1,1),(3,3)); \quad ((1,1),(3,4)); \quad ((1,2),(2,4)); \quad ((1,2),(3,3)); \\ & ((2,1),(3,2)); \quad ((2,1),(3,3)); \quad ((2,4),(3,3)); \quad ((3,1),(4,1)); \\ & ((3,1),(4,2)); \quad ((3,1),(4,3)); \quad ((3,1),(4,4)); \quad ((3,2),(4,2)); \\ & ((3,2),(4,3)); \quad ((3,3),(4,3)); \quad ((3,4),(4,3)); \quad ((3,4),(4,4))\}. \end{aligned}$$

On the other hand, Figure 5.3b shows an instance with six points. The three first are at the same positions as in Figure 5.3a, and points 4, 5 and 6 have coordinates $(11, 5)$, $(19, 7.5)$ and $(19.5, 7.7)$, respectively. The size of the labels is again 10×4. In this case, every label of points 5 and 6 overlaps all the labels of one of the remaining points in $I \setminus \{5, 6\}$. △

Labeling problems within this context are commonly grouped under the name Point-Feature Label Placement (PFLP). The term does not correspond to a unique problem but is commonly associated to two variants, given by the following definitions.

Definition 5.1 (*PFLP$_{max}$*)**.** Given a set of point features I and a family of candidate labels $\{L_i\}_{i \in I}$, PFLP$_{\max}$ consists of finding a subset of points $I' \subseteq I$ with maximum cardinality such that there is a labeling $\{\ell_i\}_{i \in I'}$ with $\ell_i \in L_i$ and no pair in $\{\ell_i\}_{i \in I'}$ belonging to E. △

Definition 5.2 (*PFLP$_{min}$*)**.** Given a set of point features I and a family of candidate labels $\{L_i\}_{i \in I}$, PFLP$_{\min}$ consists of selecting a set of labels $\{\ell_i\}_{i \in I}$ with $\ell_i \in L_i$ and minimum number of overlaps. △

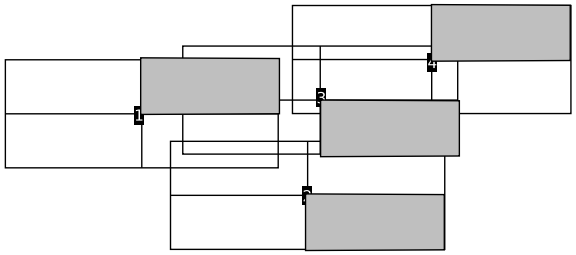


Figure 5.4: Optimal solution to formulation (Z) on Example 5.1

The problem in Definition 5.1 is to look for the most complete labeling without declining legibility, while in Definition 5.2 all points must be labeled and the objective is to preserve readability as much as possible. Different subvariants of the latter include minimizing the pairs of overlapping labels in $\{\ell_i\}_{i\in I}$ or just the number of labels in $\{\ell_i\}_{i\in I}$ that overlap some other label in the solution. More elements regarding other quality indicators could be incorporated to these basic problem statements, which would give rise to more variants of the PFLP. Computational complexity results for both versions of the PFLP, which are NP-hard, can be consulted in Marks & Shieber (1991); Formann & Wagner (1991).

5.3 Previous formulations

Existing formulations for the PFLP rely on a common framework of variables and constraints. Here, we present the first models for the two versions of the problem described in definitions 5.1 and 5.2, which will introduce the reader to the notation and ideas of the formulations presented in the chapter. Finally, two models that are somewhat related to our proposal are also introduced.

Throughout the chapter, we will use the following binary variables, which are usually consider when formulating the PFLP with a mathematical program:

$x_j = 1$ if label $j \in L$ is drawn,
$y_i = 1$ if point $i \in I$ is labeled.

5.3.1 First models

Zoraster (1990) was the first to propose an integer program to address the $\text{PFLP}_{\max}$. He used a large number M to ensure that maximizing the number of labeled points was the more important part of the objective function. Minimizing a penalty f_j, $j \in L$, associated with the chosen labels was of secondary importance. Zoraster's formulation reads

$$
\begin{aligned}
\text{(Z)}\ \min\ & \sum_{j\in L} f_j x_j + \sum_{i\in I} M(1-y_i) & & \\
\text{s.t.}\ & \textstyle\sum_{j\in L_i} x_j = y_i & \forall i \in I & \quad (5.1)\\
& x_j + x_\ell \le 1 & \forall (j,\ell) \in E & \quad (5.2)\\
& x_j \in \{0,1\} & \forall j \in L & \\
& y_i \in \{0,1\} & \forall i \in I. &
\end{aligned}
$$

Example 5.2. Applying formulation (Z) to Example 5.1 (Figure 5.3a) gives the optimal solution of Figure 5.4. Here we have taken $f_j = k-1\ \forall i \in I, \forall j = (i,k)$ and thus have assumed that $(i,1)$ is preferred to $(i,2)$ and so on. Points 1 and 4 are labeled without penalty (with labels at the top-right) and points 2 and 3 are labeled with penalty 1 (with labels at the lower-right part). △

Since $a_{j\ell} = 1\ \forall i \in I$, $\forall j, \ell \in L_i$, (Z) is equivalent to

$$\begin{aligned} \max \quad & \sum_{j \in L} (M - f_j) x_j \\ \text{s.t.} \quad & x_j + x_\ell \leq 1 \qquad \forall (j,\ell) \in L \times L :\ a_{j\ell} = 1 \\ & x_j \in \{0,1\} \qquad \forall j \in L, \end{aligned}$$

which is a particular case of set packing problem. This fact was observed by Verweij (2000) (see also Verweij & Aardal, 1999). In his Ph.D. thesis, he studied the intersection graph of the problem and designed a branch-and-cut solving strategy.

When all points must be labeled, the formulation proposed in Mauri et al. (2010) aims to minimize the number of labels that overlap one or more other labels in the solution. Labels penalties f_j are also considered in the objective function. The authors introduced a new set of binary variables defined as

$z_i = 1$ if a label $j \in L_i^4$ is used and another label ℓ such that $a_{j\ell} = 1$ is also used, $i \in I$.

Their formulation of the $\text{PFLP}_{\max}$ reads

$$\begin{aligned} \text{(MRL)} \ \max \quad & \sum_{j \in L^4} (M - f_j) x_j - \sum_{i \in I} z_i \\ \text{s.t.} \quad & \textstyle\sum_{j \in L_i^4} x_j = 1 && \forall i \in I && (5.3) \\ & x_j + x_\ell \leq 1 + z_i && \forall i \in I,\ j \in L_i^4,\ \ell \in L^4 \setminus L_i^4 :\ a_{j\ell} = 1 && (5.4) \\ & x_j \in \{0,1\} && \forall j \in L^4 \\ & z_i \in \{0,1\} && \forall i \in I. \end{aligned}$$

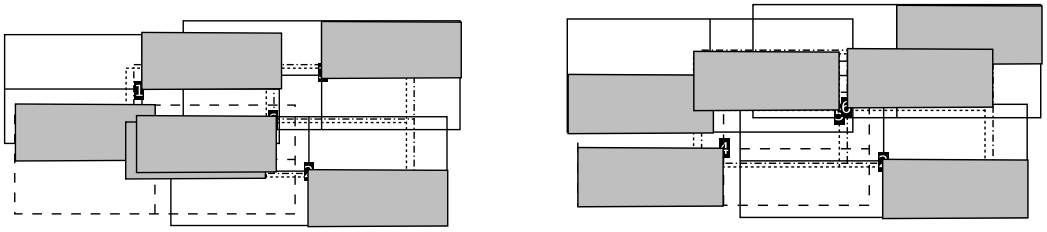
In (MRL), y-variables are not necessary, since all the points must be labeled, as stated by (5.3). New z-variables are included in Zoraster's constraints (5.2) to count how many times overlaps occur, which leads to constraints (5.4).

Instead of minimizing the number of overlapping labels, Ribeiro & Lorena (2008) proposed to minimize the number of overlapping pairs of labels. The variables used for quantifying these overlaps were

$z_{j\ell} = 1$ if a pair of overlapping labels $(j,\ell) \in E$ are both used.

The alternative formulation of the $\text{PFLP}_{\max}$ is

$$\begin{aligned} \text{(RL)} \ \min \quad & \sum_{j \in L^4} f_j x_j + \sum_{j \in L^4} \sum_{\substack{\ell: \\ (j,\ell) \in E}} z_{j\ell} \\ \text{s.t.} \quad & \textstyle\sum_{j \in L_i^4} x_j = 1 && \forall i \in I \\ & x_j + x_\ell \leq 1 + z_{j\ell} && \forall (j,\ell) \in E \\ & x_j \in \{0,1\} && \forall j \in L^4 \\ & z_{j\ell} \in \{0,1\} && \forall (j,\ell) \in E. \end{aligned}$$



(a) Optimal solution of (MRL)

(b) Optimal solution of (RL)

Figure 5.5: Optimal solutions of (MRL) and (RL) to the six-points instance of Figure 5.3b

Example 5.3. Figure 5.5 depicts optimal solutions for the instance of Figure 5.3b obtained with formulations (MRL) and (RL). The number of overlapping labels in the optimal solution of (MRL) is 3, while it is 4 in the solution of (RL). Nevertheless, the number of pairs of overlapping labels with (MRL) is 3, and using formulation (RL) it is only 2. △

Other integer programming formulations of the PFLP include Verweij (2000); Klau (2001); Klau & Mutzel (2003); Verweij & Aardal (1999); Mauri et al. (2010). A concise but descriptive revision of these formulations is provided in Marín & Pelegrín (2018b). In the following section, we only describe the two integer programs that are closer to the approach we propose.

5.3.2 Related models

There are two models in the literature, both proposed by Gomes et al. (2013), oriented towards spreading the labels. They address the second variant of the PFLP, where every point receives a label. However, instead of minimizing the overlaps, they propose to maximize t, the minimum of the distances between the centers of the overlapping labels. The desired effect is to spread the conflicting labels on the map as much as possible, which is related to reducing ambiguity in a way.

The formulation they proposed first is

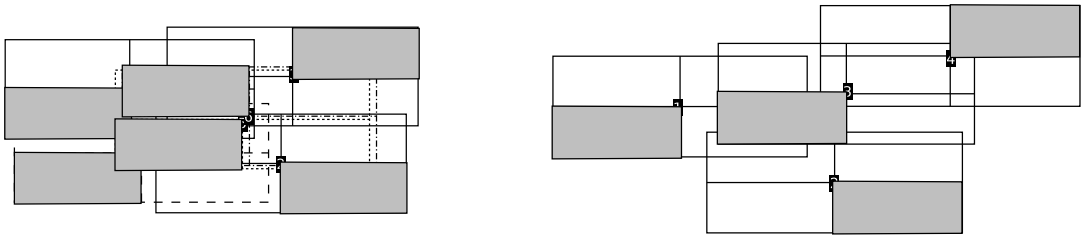
$$\begin{aligned} \text{(GRL1)}\ \max \quad & t & & \\ \text{s.t.} \quad & \textstyle\sum_{j \in L_i^4} x_j = 1 & \forall i \in I & \\ & t \leq M(2 - x_j - x_\ell) + d_{j\ell} & \forall (j,\ell) \in E & \quad (5.5) \\ & x_j \in \{0,1\} & \forall j \in L^4. & \end{aligned}$$

Observe that constraints (5.5) influence the objective only when both x_j and x_ℓ take value 1, in which case t will be less than or equal to $d_{j\ell}$, the Euclidean distance between the centers of labels j and ℓ.

In order to avoid the use of the big M, the authors proposed a second formulation,

$$\begin{aligned} \text{(GRL2)}\ \max \quad & t & & \\ \text{s.t.} \quad & \textstyle\sum_{j \in L_i^4} x_j = 1 & \forall i \in I & \\ & t \leq d_{j\ell}(2 - x_j - x_\ell) & \forall (j,\ell) \in E & \quad (5.6) \\ & x_j \in \{0,1\} & \forall j \in L^4. & \end{aligned}$$

Here, when both x_j and x_ℓ take value 1 the objective value t will take value 0. The authors then explain that "it is advantageous to assign zero to x_j and x_ℓ so that t may increase to $2d_{j\ell}$, avoiding possible conflicts".



(a) Optimal solution of (GRL1) to Figure 5.3b (b) Optimal solution of (GRL2) to Figure 5.3a

Figure 5.6: Optimal solutions of (GRL1) and (GRL2) for two different instances

Example 5.4. Consider first formulation (GRL1). Applying it to the six points instance of Figure 5.3b gives the optimal solution of Figure 5.6a. Since, in this example, some labels necessarily overlap, (GRL1) works properly and a solution with labels as widely spread as possible is given. The optimal value is 8.38, the distance between the centers of labels of points 4 and 5. However, when (GRL1) is applied to the four-point instance given in Figure 5.3a, a solution without overlaps will have the effect of turning constraints (5.5) into $t \leq M + \min\{d_{j\ell} : \ j, \ell \in L^4, \ (j, \ell) \in E, \ x_j = 1\}$. This is an undesired effect since these distances take into consideration labels that were not drawn.

Consider now formulation (GRL2). Applying it to the four-points instance of Figure 5.3a gives the optimal solution of Figure 5.6b. In this example (GRL2) works well, due to the existence of solutions without overlaps, but constraints (5.6) have the effect of bounding t by $d_{j\ell}$ when one of the labels is used but the other is not. The optimal value in the example is 3.16, the distance between the labels of points 2 and 3. When (GRL2) is applied to the six-point instance given in Figure 5.3b, a solution without overlaps does not exist, and then the optimal value of (GRL2) is going to be 0 and all labelings are optimal, even if all labels overlap each other. △

5.4 A new approach

Figure 5.7 shows two different labelings of the same set of points. In both cases the five points are labeled without overlaps. However, the upper labeling is clearly ambiguous, since there are several sets of labels that could correspond to the same point. The reason is that the corners of some labels are too close to points to which they are not assigned. This gives the intuition behind the quantitative estimation of ambiguity presented in the following definition.

Definition 5.3. Let I be a set of point features with candidate labels $\{L_i\}_{i \in I}$. The ambiguity of a given label $j \in L_i$ will be the minimum distance from any of the corners of label j (except the corner placed at point i) to any of the points in $I \setminus \{i\}$. This minimum will be named d_j,

$$d_j := \min\{d(k, i') : \ k \text{ is a corner of } j, \ i' \in I; \ k \neq i, i' \neq i\} \qquad \forall i \in I \, \forall j \in L_i,$$

where $d(\cdot, \cdot)$ stands for the Euclidean distance in the plane. △

Definition 5.3 assumes that labels are rectangles with one of their corners placed on the point to be labeled. Alternative configurations, such as different labels shape, would need some slight modifications of the models but could also be approached without difficulty. As an illustrative example, Figure 5.8 depicts a rectangular label $j \in L_i$ and seven points other than i. Arrows indicate the closest corner to each point. If label j is used, the ambiguity produced by j will be due to the point feature within the minimum distance, marked with a thick arrow.

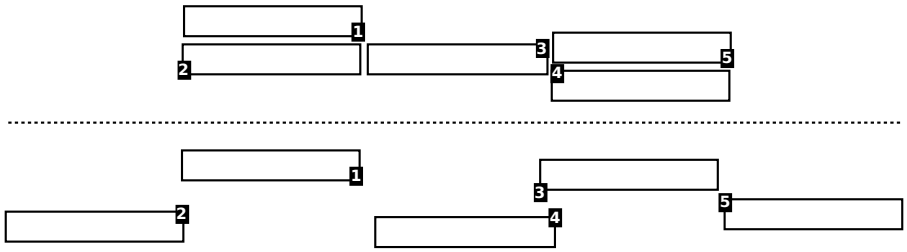


Figure 5.7: Different labelings of the same five points

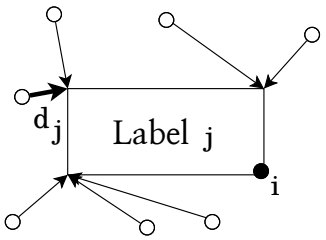


Figure 5.8: Illustration of Definition 5.3

Using Definition 5.3 as starting point, we present a family of integer programming formulations to reduce ambiguity in the PFLP. A first group of formulations, based on labels classification in ambiguity classes, is presented in Section 5.4.1. The limits of these ambiguity classes are an input parameter of the model, i.e., they should be determined by the user. For an adaptive approach, which ranks all the labels by their ambiguity, we propose a second group of formulations in Section 5.4.2, which are inspired on discrete ordered location models, Marín et al. (2009, 2010). Both families consist of formulations for the two versions of the PFLP, namely $\text{PFLP}_{\max}$ and $\text{PFLP}_{\min}$. We will remove the labels $\ell \in L_i$ that overlap a point $i' \neq i$, i.e., we assume $i' \notin \ell \ \forall \ell \in L_i$ with $i' \neq i$. The number of labels in L_i is not relevant in our models, but it will be fixed for the computational experiments in Section 5.6. Finally, ℓ_i will denote the label assigned to i, for all $i \in I$.

5.4.1 Building ambiguity classes

In this first model labels are classified depending on the ambiguity they produce. We define

$$C_0 := 0 < C_1 < \cdots < C_\rho := \infty$$

the endpoints of the ambiguity classes. The set of classes will be indexed by $r \in \{1, \ldots, \rho\}$, with class r containing those labels j such that $d_j \in [C_{r-1}, C_r)$. Given a labeling, we will say that a point i falls into a class r if its label, j_i, is such that $d_{j_i} \in [C_{r-1}, C_r)$. The reader may think of C_1 as a value below which the label is extremely ambiguous, C_2 a value below which the label is very ambiguous, and so on. Apart from classifying labels into groups, this prevents large distances from affecting the model (class $[C_{\rho-1}, C_\rho)$ can be considered as a non-ambiguous labels container).

The global ambiguity of a solution is obtained from the number of points in each ambiguity class. Each class r is given a relative weight, called $\mu_r \geq 0$, which can be tuned according to the decision maker's taste. Concretely, the number of labels in the r-th class, n_r, will be multiplied by μ_r and the linear combination through the classes will be maximized. Then, if the decision maker chooses $\mu_1 = 0$ and the remaining μ_r taking value 1, the model will try to

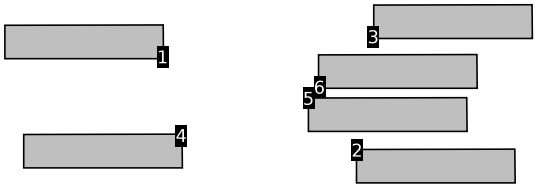


Figure 5.9: Result of Example 5.5

avoid extremely ambiguous labels and will not pay attention to the other classes of ambiguity. If he/she chooses a vector $\mu = (\mu_1, \ldots, \mu_\rho) = (0, \frac{M}{2^{\rho-2}}, \ldots, \frac{M}{2}, M)$, less ambiguous classes are prioritized by a factor of two. We propose two models to address the two variants of the PFLP.

The new IP formulation to address ambiguity reduction in the $\text{PFLP}_{\max}$ is

$$\text{(MP1)} \quad \max \quad \sum_{i \in I} M y_i + \sum_{r=1}^{\rho} \sum_{\substack{j \in L: \\ C_{r-1} \le d_j < C_r}} \mu_r x_j$$

$$\text{s.t.} \quad \sum_{j \in L_i} x_j = y_i \qquad \forall i \in I \tag{5.7}$$

$$x_j + \sum_{\substack{\ell \in L_i: \\ a_{j\ell}=1}} x_\ell \le 1 \qquad \forall (i, i') \in H, j \in L_{i'} \tag{5.8}$$

$$y_i, x_j \in \{0, 1\} \qquad \forall i \in I, \forall j \in L \tag{5.9}$$

Constraints (5.7) and (5.8) keep track of the labels used and forbid overlapping labels. In the objective function, we distinguish two parts. The first part, $\sum_{i \in I} M y_i$, guarantees that the number of labeled points is going to be maximized. If M is sufficiently large, this part of the objective is prioritized. The second part tries to reduce the ambiguity of the solution. The following example illustrates (MP1).

Example 5.5. Consider the six points of Figure 5.3b, and labels of size 10×2 (in order to give points 5 and 6 the possibility of being labeled). We take $\rho = 5$ classes, with $C = (0, 1, 3, 5, 7, \infty)$ and $\mu = (1, 2, 3, 4, 5)$.

The optimal solution is depicted on Figure 5.9. Ambiguities associated to each pair point-label (which are preprocessed data) and optimal values of the x-variables are given in the following matrices:

$$d = \begin{pmatrix} 2.4 & 0.6 & 3.2 & 7.1 \\ 3.0 & 6.2 & 1.4 & 1.4 \\ 6.4 & 3.7 & 3.2 & 3.2 \\ 1.4 & 1.4 & 7.1 & 3.2 \\ - & 2.3 & 2.1 & 1.1 \\ 2.3 & 1.9 & - & 0.6 \end{pmatrix} \quad x^* = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ - & 1 & 0 & 0 \\ 1 & 0 & - & 0 \end{pmatrix}$$

where "$-$" means that the label overlaps a point and cannot be used. From these two matrices, we obtain the ambiguities of each label: $(7.1, 6.2, 6.4, 7.1, 2.3, 2.3)$. The minimum ambiguity is $d_5 = 2.3$, which is the distance from point 6 to the left-bottom corner of ℓ_5. Since 2.3 lies in the second class, $[C_1, C_2) = [1, 3)$, we add 2 units to the objective function. Considering all the labels one by one, the optimal value obtained with this formulation is $6M+2+2+4+4+5+5$. $\triangle$

Formulation (MP1) allows unlabeled points. When all points must be labeled, we present a modification, called (MP1'), that incorporates ambiguity reduction to the objective of minimizing labels overlaps. Additional variables that keep track of the overlaps are needed,

$y'_{ii'} = 1$ if labels assigned to conflicting points $i, i' \in I$ overlap.

The formulation we propose for $\text{PFLP}_{\min}$ is

$$\begin{aligned}
\text{(MP1')}\ \max\ & \sum_{(i,i')\in H} -My'_{ii'} + \sum_{r=1}^{\rho} \sum_{\substack{j\in L:\\ C_{r-1}\le d_j<C_r}} \mu_r x_j \\
\text{s.t.}\ & \sum_{j\in L_i} x_j = 1 && \forall i \in I && (5.10)\\
& x_j + \sum_{\substack{\ell\in L_{i'}:\\ a_{j\ell}=1}} x_\ell \le 1 + y'_{ii'} && \forall (i,i') \in H, \forall j \in L_{i'} && (5.11)\\
& x_j \in \{0,1\} && \forall j \in L\\
& y'_{ii'} \ge 0 && \forall (i,i') \in H.
\end{aligned}$$

Here, $y'_{ii'}$ is used to limit the number of overlapping pairs of labels. Note that $y'_{ii'}$ is not defined as a binary variable, but it will take value 1 or 0 at the optimal solution in any case. To understand this, note that $y'_{ii'}$-variables have negative coefficients in the objective function (to be maximized) and they will take a positive value only if some of the constraints of the problem forces it. However, for a fixed pair $(i,i') \in H$, variable $y'_{ii'}$ only appears in constraints (5.11), which establishes $x_j + \sum_{\substack{\ell\in L_i:\\ a_{j\ell}=1}} x_\ell - 1$ as its lower bound for different values of j. Note that this integer bound is always less than or equal to 1. This is due to the binarity of x_j and the fact that the term $\sum_{\substack{\ell\in L_i:\\ a_{j\ell}=1}} x_\ell$ is upperly bounded by 1 because of constraints (5.10). In fact, $x_j + \sum_{\substack{\ell\in L_i:\\ a_{j\ell}=1}} x_\ell - 1$ only takes values in $\{-1, 0, 1\}$, which ensures that $y'_{ii'}$ is binary in the optimal solution.

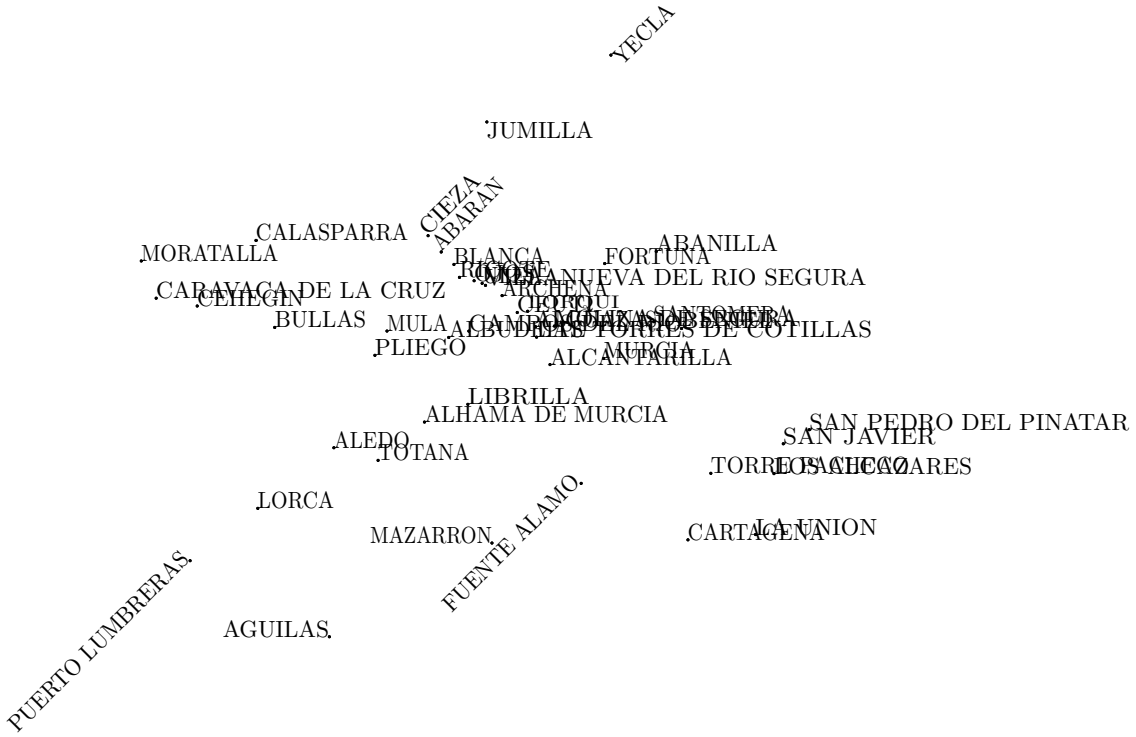
Formulations (MP1) and (MP1') can be seen as variants of formulations (Z) and (RL), respectively. Constraints to control labels overlaps are improved with tighter ones, and weights of the ambiguity classes can be seen as labels priorities f_j.

Example 5.6. Figure 5.10 (resp. 5.11) shows the result of applying formulations (GRL1) and (GRL2) (resp. (MP1) and (MP1')) to 45 towns in the Region of Murcia, in the south-east of Spain. The size of the labels has been chosen to be proportional to the name of the towns, and large enough to make it impossible to label all points without overlaps. Each point has eight candidate labels, the four usual labels plus the four labels obtained from these by means of a 45-degree counterclockwise rotation (see Figure 5.1 and left-hand side of Figure 5.2). Taking the height of the labels as the unit, formulations (MP1) and (MP1') are applied using parameters $C = (0, \frac{0.4}{\sqrt{2}}, 0.4, 0.4\sqrt{2}, 0.8, \infty)$ and $\mu = (1, 2, 3, 4, 5)$ (note that $\sqrt{2}$ is related to the small side of the sloping labels). The readability of the last two maps is much better, not only in their central parts but also in the periphery.

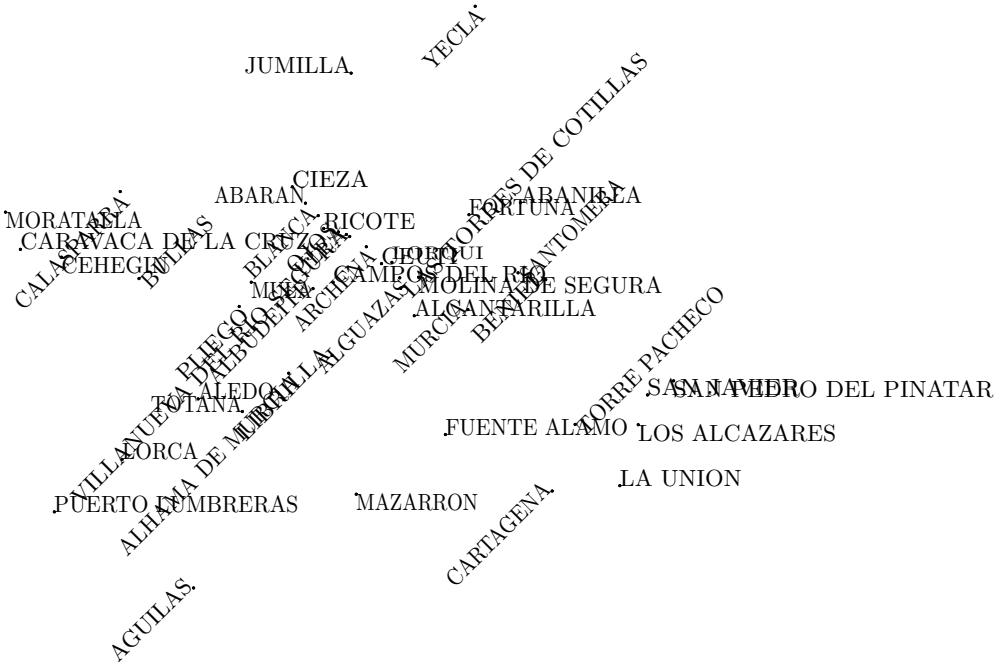
△

5.4.2 An ordered model

The success of the previous model strongly depends on the choice of the class delimiters $C_1, \ldots, C_{\rho-1}$. Once we are given a concrete instance of the PFLP, a good selection crite-

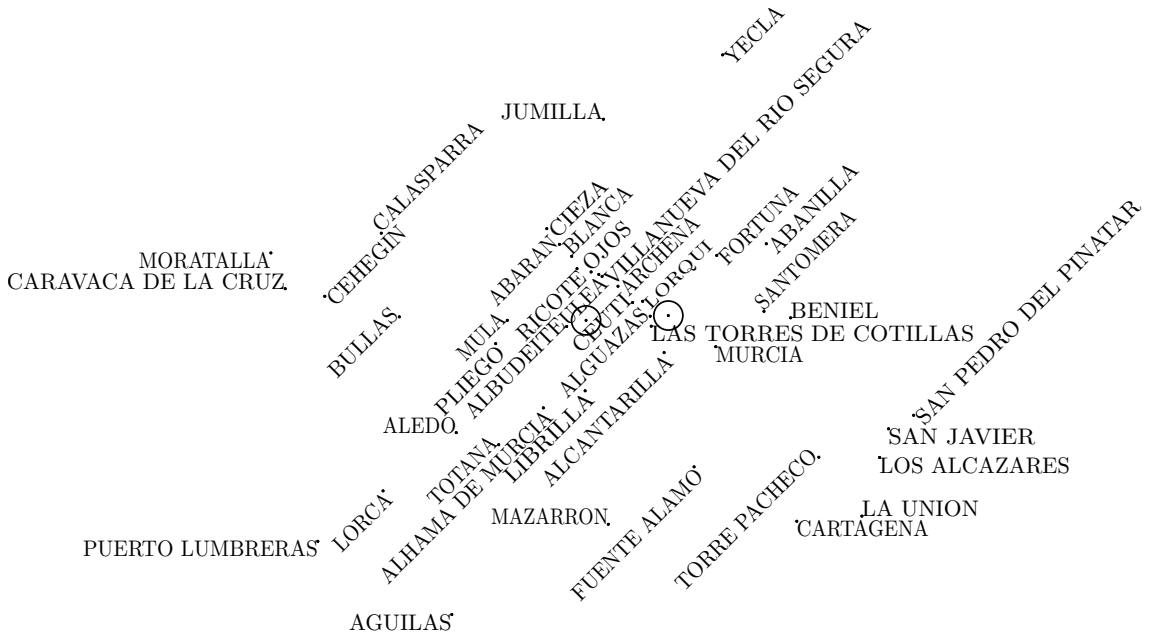


(a) Optimal solution of formulation (GRL1)



(b) Optimal solution of formulation (GRL2)

Figure 5.10: Labeling of 45 towns in the Region of Murcia (Spain) with (GRL1) and (GRL2)



(a) Optimal solution of (MP1)

(b) Optimal solution of (MP1') with $\mu_i = i$

Figure 5.11: Labeling of 45 towns in the Region of Murcia (Spain) with (MP1) and (MP1'). In the case of (MP1) there are two unlabeled points, enclosed with circles

rion to define the classes should be related to distances d_j, $j \in L$. Otherwise, all the labels may fall into the same class and therefore, our model would not address the problem of ambiguity.

In this section we build a more flexible model to tackle the problem of ambiguous labelings. First, we remove from the objective barely ambiguous labels by setting a threshold T and defining new sets of ambiguous labels for each point,

$$L_i^T := \{j \in L_i : \ d_j \leq T\}, \qquad i \in I.$$

Let $L' \subseteq L$ be a feasible labeling and suppose, without loss of generality, that labels $\{j_1, \ldots, j_t\} \subseteq L'$ $(t \leq n)$ are below the threshold. Instead of grouping these labels by classes, consider a permutation σ for which the following inequalities hold:

$$T \geq d_{j_{\sigma(1)}} \geq d_{j_{\sigma(2)}} \geq \cdots \geq d_{j_{\sigma(t)}}.$$

Equivalently,

$$0 \geq d_{j_{\sigma(1)}} - T \geq d_{j_{\sigma(2)}} - T \geq \cdots \geq d_{j_{\sigma(t)}} - T.$$

The objective function of our model will be partially obtained by multiplying these sorted values by a set of parameters $\lambda_i \geq 0$, that is to say, we want to include the non-linear term

$$\lambda_1(d_{j_{\sigma(1)}} - T) + \lambda_2(d_{j_{\sigma(2)}} - T) + \cdots + \lambda_t(d_{j_{\sigma(t)}} - T) \tag{5.12}$$

in the objective. As before, parameters λ can be varied according to the decision maker's taste. Since we will maximize the objective, λ_i, $i = 1, \ldots, t$ should be an increasing sequence. Thus, the more ambiguous a label j_i is (i.e., the shorter its distance d_{j_i} is), the larger its (negative) contribution to the objective becomes. This assures that d_{j_i} is as large as possible in the optimal solution. However, observe that, unlike T and λ, which are problem inputs, $d_{j_{\sigma(\cdot)}}$ values depend on the optimal solution.

In order to incorporate the idea described by (5.12) into an ILP formulation, we first sort the d_j-values of all labels in $L^T := \cup_{i \in I} L_i^T$ in decreasing order (avoiding ties) as

$$(T \geq)\ d'_1 > d'_2 > \cdots > d'_K > 0,$$

where K is the number of labels in L^T with different d_j-values. Since the number of values in the solution falling below T is not previously known, the λ vector must be defined with dimension n. Similarly, we extend σ to all the points given, that is, we will assume that:

$$d_{j_{\sigma(1)}} \geq d_{j_{\sigma(2)}} \geq \cdots \geq d_{j_{\sigma(n)}}.$$

These extensions are necessary in order to define the following variables correctly:

$s_{ik} = 1$ if $d_{j_{\sigma(i)}} \leq d'_k$ $(< d'_{k-1} < \cdots < d'_1)$ and $s_{ik} = 0$ otherwise, $\forall i \in I$, $k \in \{1, \ldots, K\}$.

Assuming that the labels in the solution are ranked by ambiguity, $s_{ik} = 1$ if the ith label in that ranking is at most as ambiguous as the kth ambiguous label in L^T. The resulting matrix of s-variables, (s_{ik}) will be ordered in two different senses. First, since $d'_k < d'_{k-1} < \cdots < d'_1$, $s_{ik} = 1$ implies that $s_{ir} = 1$ for $r < k$. As a consequence, rows in the matrix are made of consecutive ones, followed by consecutive zeros. The transition from 1 (say at column k) to 0 (at column $k+1$) in a row i means that $d_{j_{\sigma(i)}} = d'_k$. Secondly, columns in (s_{ik}) are increasingly ordered by the number of zeros they contain. Note that column k refers the label of L^T whose ambiguity occupies the place kth in the ordering $(T \geq)\ d'_1 > d'_2 > \cdots > d'_K > 0$,. When k increases, d'_k decreases, and there are less labels in the solution with ambiguity smaller than or equal to d'_k.

We consider the two variants of the PFLP and, based on the idea above, we propose two alternative formulations. Using new s-variables along with the previously used x- and y-variables, the first flexible formulation, which address the $\text{PFLP}_{\max}$, is given by

$$\text{(MP2)} \quad \max \quad \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{K-1} (d'_k - T)(s_{ik} - s_{i,k+1}) + (d'_K - T)s_{iK} \right) \quad + \sum_{i \in I} M y_i$$

$$\text{s.t.} \qquad \sum_{j \in L_i} x_j = y_i \qquad \forall i \in I \tag{5.13}$$

$$x_j + \sum_{\substack{\ell \in L_i: \\ a_{j\ell}=1}} x_\ell \leq 1 \qquad \forall (i,i') \in H, j \in L_{i'} \tag{5.14}$$

$$\sum_{i \in I} \sum_{\substack{j \in L_i^T: \\ d_j \leq d'_k}} x_j = \sum_{i=1}^{n} s_{ik} \qquad \forall k = 1, \ldots, K \tag{5.15}$$

$$s_{ik} \leq s_{i+1,k} \qquad \forall i \in I \setminus \{n\}, \forall k = 1, \ldots, K \tag{5.16}$$

$$y_i, x_j, s_{ik} \in \{0,1\} \qquad \forall i \in I, \forall j \in L, \forall k = 1, \ldots, K.$$

Similarly to previous formulations, (5.13) and (5.14) maintain record of the used labels and control overlaps. Constraints (5.15) state that the number of ones in a column k of matrix (s_{ik}) is precisely the number of points that are labeled with ambiguity not greater than d'_k. Constraints (5.16) ensure that the ones in each column are correctly distributed. In effect, if $s_{i+1,k} = 0$ and $d_{j_{\sigma(i+1)}} \leq d_{j_{\sigma(i)}}$ it must be $s_{i+1,k} = 0$. This accumulates the ones at the end of the columns.

We distinguish two parts in the objective function. The second part, $\sum_{i \in I} M y_i$, guarantees that the number of labeled points is going to be maximized. The first part tries to reduce the ambiguity of the solution. We use the fact that the rows of the s-matrix are sorted containing zeros in the right hand side and ones in the left hand side to obtain the sorted values $d_{j_{\sigma(1)}}, \ldots, d_{j_{\sigma(n)}}$.

Note that, were we to include the distances $d_1, \ldots, d_t$ in the objective without taking the order into account, we would have a solution which would minimize the ambiguity in general. This implies that several extremely ambiguous labels could make up for a great number of unambiguous ones. The ordered model we propose allows us to control this effect. Moreover, its flexibility makes it possible to represent specific objectives. Previous models with ordered objectives have been successfully used in the Discrete Location field (Marín, Nickel, Puerto & Velten, 2009; Marín, Nickel & Velten, 2010), which is a closely related issue. The following example illustrates (MP2).

Example 5.7. Consider six points with coordinates $(10,10)$, $(21,16)$, $(10,11)$, $(21,12)$, $(20,9)$, $(10,14)$ and candidates labels in L^4, all of them of size 10×2. The values of ambiguity associated to the 24 possible labels are

$$d = \begin{pmatrix} 1.0 & 1.0 & 3.0 & 1.0 \\ 6.0 & 2.0 & 1.0 & 2.2 \\ 1.0 & 0.0 & 1.0 & 1.0 \\ 2.0 & 1.4 & 1.0 & 1.0 \\ 1.4 & 5.1 & 1.0 & 0.0 \\ 1.0 & 1.0 & 1.0 & 5.0 \end{pmatrix}.$$

If we consider the threshold value $T = 3$, and sort the values of the matrix not greater than T, the result is $d' = (3, 2.2, 2, 1.4, 1, 0)$, with $K = 6$. The optimal solution of (MP2) with $\lambda = (1, 2, 3, 4, 5, 6)$ consists of the following values for x-variables:

$$x^* = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

This matrix yields a labeling $\{\ell_1, \ldots, \ell_6\}$ with associated values of ambiguity $(3, 6, 1, 2, 5.1, 5)$, which produces the ordered vector $d_{j_{\sigma(\cdot)}} = (6, 5, 5.1, 3, 2, 1)$. The optimal values of the s-variables are given in the following matrix:

$$s^* = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Note that there are no 1's in the first three rows because three labels have ambiguity above the threshold, namely $d_{j_{\sigma(1)}} = 6$, $d_{j_{\sigma(2)}} = 5$ and $d_{j_{\sigma(3)}} = 5.1$. Since $d_{j_{\sigma(4)}} = 3$, which coincides with $T = d'_1$, the fourth row contains one 1. The three first entries of the fifth row contain ones since $d_{j_{\sigma(5)}} = d'_3$. Finally, $s_{6k} = 1$ for all $k = 1, \ldots, 5$ represents that $d_{j_{\sigma(6)}} = d'_5$. The second part of the objective function evaluated for this solution is

$$\sum_{i=1}^{6} \lambda_i \Big(\sum_{k=1}^{5} (d'_k - 3)(s_{ik} - s_{i,k+1}) + (d'_6 - 3)s_{i6}\Big) =$$

$$1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot (d'_1 - 3) + 5 \cdot (d'_3 - 3) + 6 \cdot (d'_5 - 3) =$$

$$1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot (3 - 3) + 5 \cdot (2 - 3) + 6 \cdot (1 - 3),$$

which penalize those labels that exceed T according to the weights in λ. △

As for the PFLP$_{\min}$, we present an alternative flexible model,

$$\begin{aligned}
\text{(MP2')} \quad \max \quad & \sum_{i=1}^{n} \lambda_i \Big(\sum_{k=1}^{K-1} (d'_k - T)(s_{ik} - s_{i,k+1}) + (d'_K - T)s_{iK}\Big) && - \sum_{(i,i')\in H} M y'_{ii'} \\
\text{s.t.} \quad & \sum_{j \in L_i} x_j = 1 && \forall i \in I \\
& x_j + \sum_{\substack{\ell \in L_i: \\ a_{j\ell}=1}} x_\ell \le 1 + y'_{ii'} && \forall (i,i') \in H, j \in L_{i'} \\
& \sum_{i \in I} \sum_{\substack{j \in L_i^T: \\ d_j \le d'_k}} x_j = \sum_{i=1}^{n} s_{ik} && \forall k = 1, \ldots, K \\
& s_{ik} \le s_{i+1,k} && \forall i = 1, \ldots, n-1, \ \forall k = 1, \ldots, K \\
& x_j, s_{ik} \in \{0, 1\} && \forall j \in L, \forall i \in I, \ k = 1, \ldots, K \\
& y'_{ii'} \ge 0 && \forall (i,i') \in H.
\end{aligned}$$

Again, $y'_{ii'}$ is used to limit the number of overlapping pairs of labels. Similarly to (MP1'), the formulation implicitly makes these variables to take value 0 or 1 in any optimal solution.

Example 5.8. Figure 5.12 shows optimal solutions of formulations (MP2) and (MP2') to the instance consisting of 45 towns in the Region of Murcia. Model parameters are set to $\lambda = (1, 2, \ldots, K)$ and the threshold value T coincides with the height of the labels. The solutions are very similar to that obtained with (MP1) and (MP1') and depicted by Figure 5.11. △

5.4.3 Some practical observations

Since d'_j-values are decreasingly ordered ($d'_1 > d'_2 > \ldots > d'_K$), if $d_{j_{\sigma(i)}}$ is less than d'_k, then it is also less than d'_i for $i = 1, \ldots, k-1$. Thus, the following inequalities are valid for formulations (MP2) and (MP2'):

$$s_{ik} \geq s_{i,k+1} \quad \forall i \in I,\ \forall k = 1, \ldots, K-1.$$

On the other hand, note that variables y_i can be omitted from models (MP1) and (MP2). In the objective, $\sum_{i \in I} My_i$ can be replaced by $\sum_{i \in I} \sum_{j \in L_i} Mx_j$. Then, constraints (5.7) and (5.13) turn into the following: $\sum_{j \in L_i} x_j \leq 1\ \forall i \in I$.

When it comes to making the process of labeling simpler, the problem can be divided into several subproblems. Consider, for example, a map depicting the most important cities in a country or state. Less populated areas separating huge cities may naturally avoid potential conflicts between labels. We aim to identify these independent areas in a general case, in order to solve simpler independent subproblems instead of a bigger one. We will consider $\mathcal{F} \subseteq \mathcal{P}(I)$ the family of these areas. An element of the family, $I' \in \mathcal{F}$, will represent a subset of point features on one independent area of the map. Labels of points belonging to different sets in $\mathcal{F}$ must not overlap and $\mathcal{F}$ must encompass every point in I. Concretely, the following requirements are met:

1. Given a set in the family $\mathcal{F}$, any two complementary subsets must contain respective overlapping points, i.e.:

$$\forall I' \in \mathcal{F}, \forall J \subset I',\ \exists i \in J,\ \exists i' \in I' \setminus J\ :\ L_i \cap L_{i'} \neq \emptyset.$$

2. The candidate labels of two points belonging to different sets in $\mathcal{F}$ do not overlap:

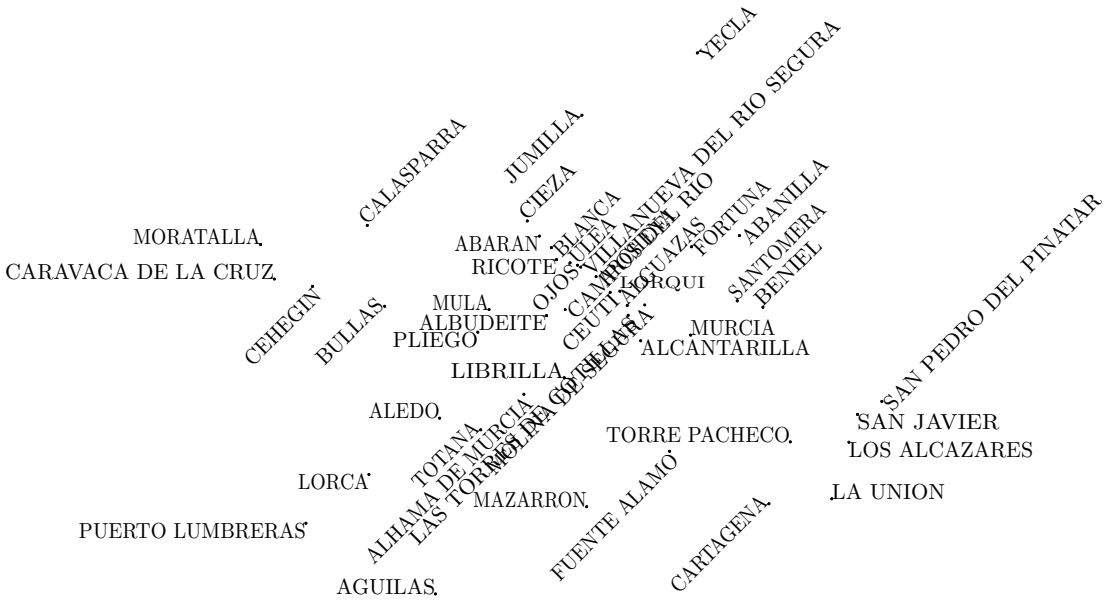
$$\forall I', I'' \in \mathcal{F}, I' \neq I'',\ \forall i \in I',\ \forall i' \in I'',\ L_i \cap L_{i'} = \emptyset.$$

3. The family $\mathcal{F}$ is a partition of I:

$$I' \cap I'' = \emptyset\ \forall I', I'' \in \mathcal{F},\ I' \neq I'', \quad \cup_{I' \in \mathcal{F}} I' = I.$$

Once the previous division has been made, we can apply any of the proposed formulations to each of the sets in $\mathcal{F}$. In this context, we recall that we consider two objectives in our models: avoidance or minimization of overlaps and reduction of ambiguity. Regarding overlaps, the previous conditions divide the map into self-reliant areas. Conversely, the ambiguity of a label may not be related to any of the points of the area that it belongs to. However, this fact does not introduce any further complexity, since ambiguities are preprocessed data that can be calculated before dividing the map into areas and solving each of the resulting subproblems. Thus, each area $I' \in \mathcal{F}$ will be self-reliant even if for some label $j \in L_i$ of a point $i \in I'$, its distance d_j is related to a point in another area.

(a) Optimal solution of (MP2)



(b) Optimal solution of (MP2') with $\lambda_i = i$

Figure 5.12: Labeling of 45 towns in the Region of Murcia (Spain) with formulations (MP2) and (MP2'). In the case of (MP2) there are again two unlabeled points, encircled

Finally, it would be interesting to have an index of ambiguity between 0 and 1 corresponding to every labeling in order to allow possible comparisons. We obtain this index from the optimal value. First, we remove the term relating to overlaps. Then, we normalize the remaining part of the objective, called a, that accounts for the ambiguity of the labeling, in the following way:

- For each $i \in I$, let $d_{max_i} = \max\{d_j \ : \ j \in L_i\}$ and $d_{min_i} = \min\{d_j \ : \ j \in L_i\}$.
- Obtain a_{max} and a_{min} as the ambiguity part of the objective function when points are labeled with labels corresponding with d_{max_i} and d_{min_i} respectively, $i \in I$.
- The normalized ambiguity value of the problem is

$$\Big(1 - \frac{a}{a_{max}}\Big)\Big(\frac{a_{max}}{a_{max} - a_{min}}\Big).$$

A resulting value of 0 would mean that the labeling is not at all ambiguous, while a value of 1 represents maximum ambiguity.

5.5 Heuristic method

In this section, we describe a heuristic procedure to find good solutions of the proposed formulations. This is to some extent naive and only intends to offer an alternative procedure for those cases in which the exact solver fails to scale as were desirable. The use of more elaborated metaheuristics to frame the proposed formulations might produce better results than those reported here, but would be worth a separated study.

Our heuristic also covers the two variants of the PFLP. The key idea, sequenced in Algorithm 5.1, is in both cases the same. The procedure starts by building an initial (potentially non-feasible) solution. Such solution assigns to each point i the less ambiguous of its candidate labels, $j \in L_i$, with high probability (see Step 1 in Algorithm 5.1). To randomize this step, function $argmax70\%$ is used. It works as a classic argmax function except that the current maximum is updated with 70% probability. Should the resulting initial solution be feasible (condition of Step 3.4 is true for each label in the solution), we consider its fitness and update the current best solution if pertinent (Step 4). Otherwise, labels placement is rearranged. For the new placement, points are first ordered in a list (Step 2 of Algorithm 5.1) and examined one by one. The aim is to go through the points from the most crowded part of the map (in terms of nearby features) to the outskirts. Each point $i \in I$ will be given a rate τ_i, which represents the sum of the number of overlaps that their labels can potentially produce, i.e.

$$\tau_i = \#\{(j, \ell) : a_{j\ell} = 1 \text{ for some } j \in L_i \text{ and } \ell \in L \setminus L_i\}.$$

In each iteration of Step 3, a non-checked point i in the ordered list will be selected (or visited) with a probability that is proportional to τ_i, so that the desired effect will be caused (see steps 3.1-3.3). Suppose that point i is selected from the list and that j is its label. If j does not overlap any other label of the current solution, it remains and the analysis of the points from the list continues (Step 3.4). When overlaps occur, we define C_i^1 as the set of candidate labels for i such that they do not overlap any label of the current solution (Step 3.5). If there are labels in this set, the least ambiguous one is assigned to i (Step 3.6). Otherwise, C_i^2 is defined as those labels in L_i that are in conflict with labels of visited points (Step 3.6). The least ambiguous label in this set will be placed at i, unless the set is empty (see Step 3.7). In this case, the point will be left unlabeled when overlaps are not allowed (models (MP1) and

Algorithm 5.1 Heuristic labeling procedure

Input d: vector containing the value of ambiguity of each label.
τ: vector containing the number of potential overlaps corresponding to each point.
r: number used to generate the probability of visiting a point.
a: matrix $A = (a_{ij})_{|L|\times|L|}$.
$maxIte$: maximum number of iterations.

Variables it: number of iterations performed so far.
$\mathcal{L}$: ordered list of points in each iteration.
$index$: list index of the selected point from $\mathcal{L}$ in each iteration of Step 3.
x_{ini}: initial solution in each iteration, $x_{ini}[i]$ is the label placed at i (potentially unfeasible).
$fitness$: quality value of the best solution found.

Output x: vector containing the label of each point, or -1 if the point is not labeled.

Step 0 $fitness := -\infty$, $it := 0$.

Step 1 $\forall i \in I\ x_{ini}[i] := argmax70\%(d_k : k \in L_i)$.

Step 2 $\mathcal{L}$:=list of points decreasingly ordered by τ, $\mathcal{L} = \{\mathcal{L}(1), \mathcal{L}(2), \ldots, \mathcal{L}(n)\}$,
$\tau[\mathcal{L}(i)] > \tau[\mathcal{L}(j)]\ \forall i, j = \{1, \ldots, n\}$, $i < j$.

Step 3 While there is a non-visited point in $\mathcal{L}$ do:

3.0 $index = 1$.

3.1 Find the next smallest $index$ verifying that $\mathcal{L}(index)$ is non-visited.

3.2 Select point $\mathcal{L}(index)$ with probability $p = 0.5(1 - e^{-\tau[index]^{\frac{1}{r}}}) + 0.5$.

3.3 If $\mathcal{L}(index)$ is not selected: $index$=$(index + 1) \mod n$, go to Step 3.1,
else $i = \mathcal{L}(index)$.

3.4 If $x_{ini}[i]$ does not overlap any other label in x_{ini}, go to Step 3.

3.5 $Lab = \{\ell \in L : \ell = x_{ini}[j]$, for some $j = 1, \ldots, n\}$,
$C_i^1 = \{k \in L_i : a_{k\ell} = 0\ \forall \ell \in Lab \setminus L_i\}$.

3.6 If $C_i^1 \neq \emptyset$: $x_{ini}[i] = \text{argmax}\ \{d_k : k \in C_i^1\}$, go to Step 3,
else $C_i^2 = \{k \in L_i : \forall \ell \in Lab\ a_{k\ell} = 1 \Rightarrow r$ is non-visited, where $x_{ini}[r] = \ell\}$.

3.7 If $C_i^2 \neq \emptyset$: $x_{ini}[i] = \text{argmax}\ \{d_k : k \in C_i^2\}$, go to Step 3,
else $x_{ini}[i] = -1$ or $x_{ini}[i] = k$, where k is a label with minimum number of overlapping labels of visited points (depending on the objective).

3.8 Go to Step 3.

Step 4 If $fitness < evaluate(x_{ini})$ then $x = x_{ini}$, $fitness = evaluate(x_{ini})$.

Step 5 $it = it + 1$.

Step 6 If $it < maxIte$ go to Step 1.

Step 7 Return x.

(MP2)) and a label producing as few overlaps as possible will be chosen when all points must be labeled (models (MP1') and (MP2')). When all the points of the list have been visited (end of Step 3), a feasible solution has been produced. In Step 4 the current best solution found and its fitness is updated, using the function *evaluate*. This function quantifies the fitness of a solution in terms of an objective function. This objective function will be taken from models (MP1), (MP1'), (MP2) or (MP2'). The process is repeated from Step 1 to generate another feasible solution, up to a maximum number of iterations.

Example 5.9. At the top of Figure 5.13 we find the solution obtained by Algorithm 5.1 for (MP1), while the solution for (MP1') is depicted at the bottom. These two labelings coincide with the heuristic labeling obtained by the same algorithm for (MP2) and (MP2'), respectively. Comparing with Figures 5.10 and 5.11, we observe that the heuristic places one label fewer than the exact formulations (MP1) and (MP2). Regarding (MP1') and (MP2') the difference in the solution quality can not be appreciated at sight (but the objective value of the heuristic solution is not optimal).

△

5.6 Computational study

In order to validate the different formulations and the heuristic proposed, we have conducted a computational study. We formed a testbed made of input data of different size. Every instance is solved under different values of the model parameters. The processor used for the experiments was an Intel core i7-6700k CPU at 4.0 GHz × 8 with 16 GB of RAM memory. The solver was Cplex v12.6.3 64-bit under operating system Linux Ubuntu 16.04.

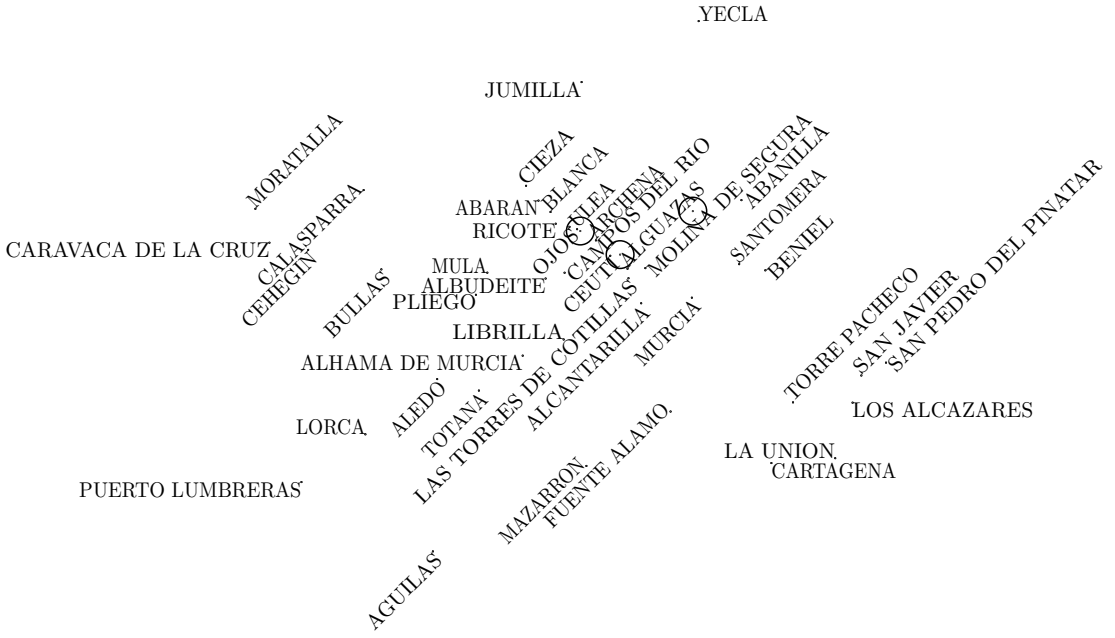
5.6.1 Test design and output format

Out testbed consists of six different data files. These files contain information about the names and the coordinates of several municipalities in Spain. Concretely, each file refers municipalities with more than 75,000; 50,000; 25,000; 10,000; 7,500 and 5,000 inhabitants. This corresponds, respectively, to 88, 137, 274, 692, 872 and 1,213 municipalities (point features) for each file. The information in our data files was taken from the website of the Coventaria (2012) consulting group. They extracted population data from the official census of 2011 in the Spanish National Institute of Statistics website, and they took coordinates from Google Maps. These coordinates are latitudes and longitudes in decimal degrees for the different municipalities in Spain. To obtain coordinates in the plane we make the following transformation:

$$x_i = \frac{longitud_i \cdot 10^6}{84750}, \qquad y_i = \frac{latitude_i \cdot 10^5}{11111}.$$

The resulting point features (x_i, y_i), $i \in I$, would lie in a plane of dimensions 120×142. Labels are placed on this flat mean with scale 0.1.

We have performed several experiments with each data file. For each point we considered eight possible labels, the four usual labels (see Figure 5.1) plus the four labels obtained from them by means of a 45-degree counterclockwise rotation (see left hand-side of Figure 5.2). First, we tested each one of our models: (MP1), (MP1'), (MP2) and (MP2'). For each model and data file tested, we varied different parameters. On the one hand, some parameters are involved in all the models. These parameters are: M, the size of the labels, and the parameters of the heuristic r and $maxIte$. We took $M = 10000$, $r = 10$ and $maxIte = 500$. We fixed a common



(a) Solution of the heuristic for (MP1) and (MP2)

(b) Solution of the heuristic for (MP1') with $\mu_i = i$ and (MP2') when $\lambda_i = i$

Figure 5.13: Labeling of 45 towns in the Region of Murcia, Spain, obtained by the heuristic for (MP1) and (MP1') with $\mu_i = i$ (these are also the solutions found for (MP2) and (MP2') when $\lambda_i = i$). In the case of (MP1) (or (MP2)) there are three unlabeled points, rounded with circles

$\mu^1 =$	(1,2,3,4,5)	$\lambda^1 =$	(1,2,3,4,5,0,. . .,0)
$\mu^2 =$	(1,2,4,8,16)	$\lambda^2 =$	(1,1,1,1,1,0,. . .,0)
$\mu^3 =$	(1,4,8,9,10)	$\lambda^3 =$	(10,0,. . .,0)
$\mu^4 =$	(0.01,0.1,1,10,100)	$\lambda^4 =$	(1,0,0,0,0,0,0,0,0,1,0,. . .,0)

Table 5.2: Vectors of weights that were used in the experiments

	$> 75,000$	$> 50,000$	$> 25,000$	$> 10,000$	$> 7,500$	$> 5,000$
$h = 0.1$	13	17	35	94	117	269
$h = 0.5$	17	25	112	685	868	1212
$h = 1$	45	118	272	691	871	1212

Table 5.3: Subproblem maximum number of points

height for every label of the same experiment. We used three different heights: 0.1, 0.5 and 1. The width varied depending on the length of the name of the specific town, which was scaled in order to be proportional with the selected height in the different cases. On the other hand, each model has its own parameters. Models (MP1) and (MP1') are characterized by the class limits $C_0 := 0, C_1, \ldots, C_\rho := \infty$ and the vector of weights $\mu = (\mu_1, \ldots, \mu_\rho)$. The former were always taken following a fixed strategy, which depends on the label height, h. Specifically, we took $\rho = 5$ and $C_0 := 0$, $C_1 = \frac{c \cdot h}{\sqrt{2}}, C_2 = c \cdot h, C_3 = c \cdot h \cdot \sqrt{2}, C_4 = c \cdot h \cdot 2$, $C_5 := \infty$, where c is a constant that we fixed to 0.4. Models (MP2) and (MP2') are characterized by the vector of weights $\lambda = (\lambda_1, \ldots, \lambda_n)$ and the threshold T. We took $T = 1.5 \cdot h$. To generate our test battery, we considered four possible choices of the vectors of weights (μ and λ respectively), displayed on Table 5.2. As a result, we performed $6(files) \times 4(models) \times 3(sizes) \times 4(\mu \; or \; \lambda) = 288$ experiments.

Recall that, to label a map, we divide it into several self-sufficient areas, in the sense explained in Subsection 5.4.3. As a consequence, we are not actually solving labeling instances of size 88, 137, 274, 692, 872 and 1213, but subproblems of these. Table 5.3 shows the number of points of the biggest subproblem processed, which depends on the data file and the size of labels.

In all the experiments both CPLEX solver and our heuristic were applied. When summarizing the running time results, we distinguish between two categories of instances: those where CPLEX was not capable of finding the optimum within a time limit and those where an optimal solution was found. One instance will belong to the first category if some of its subproblems is not solved within one hour. We analyze running results from three different perspectives: fixing the model (Table 5.4), fixing the size of the labels and data file (Table 5.7) and fixing the model and the vector of weights (Table 5.9). These tables summarize the average results obtained for each case. Some of their entries are double, because they contain the same information, but referring to the exact and the heuristic solution (upper and lower subentries, respectively). In the following, we outline the meaning of the abbreviations in tables 5.4, 5.7 and 5.9.

LP: optimal value of the linear relaxation.

LB: lower bound on the objective value.

OPT: objective value of the solution found by CPLEX/our heuristic (OPT_h).

$Norm$: normalized objective value of the solution found by CPLEX/our heuristic.

Gap: relative gap between LP and OPT/between OPT and OPT_h (Gap_h).

Nod: number of branching nodes that were processed in the CPLEX solution.

$\%L$: percentage of labeled points in the solution found by CPLEX/our heuristic.

$\%O$: percentage of overlaps in the solution found by CPLEX/our heuristic.

$Time$: running time in seconds of CPLEX/our heuristic.

Ci: number of labels in the optimal/heuristic solution in class of ambiguity i.

$< T$: number of labels in the optimal/heuristic solution below the threshold T.

$\%NS$: percentage of instances that CPLEX could not solve to optimality.

The strategy followed to generate a proper lower bound, LB, was different depending on the formulation. For (MP1) and (MP2) setting all variables equal to zero is clearly a feasible solution. Zero is then a valid lower bound. Formulations (MP1') and (MP2') force the labeling of every point. Now, to get a feasible solution, we chose for each point the least ambiguous label and calculated the objective value of the resulting solution. Note that this number is also a lower bound on the objective value of the heuristic solution (in fact, it is the objective value of the initial solution of the heuristic). These lower bounds are especially useful to get the relative gaps when the objective function can be negative, which is the case of models (MP1') and (MP2'). We quantify the relative gaps as

$$Gap = \frac{LP - OPT}{LP - LB} \cdot 100, \quad Gap_h = \frac{OPT - OPT_h}{OPT - LB} \cdot 100.$$

Note that, when the objective function is positive, $LB = 0$ is implicitly taken. Nevertheless, when we are maximizing an objective function which takes negative values, we need a lower bound in order to get a relative gap between 0% and 100%.

On the other hand, $\%O$ is related to the number of points, n, instead of the potential number of overlaps $\binom{n}{2}$, which was considered too large. This might lead to values of $\%O$ greater than 100. The normalized ambiguity of an instance, $Norm$, is obtained by averaging the normalized values of the labelings of its subproblem. Labelings of areas with few points barely produce ambivalence, thus diminishing the global ambiguity of the instance. Due to this, $Norm$ usually shows low values in the summary tables we report. Finally, since the running time for a given instance is the sum of the running times of the corresponding subproblems, $Time$ values may be greater than the time limit of the solver (one hour).

5.6.2 Overall results

Tables 5.4 and 5.5 show the average experimental results obtained for each of the proposed models, namely (MP1), (MP1'), (MP2) and (MP2'). Table 5.4 refers to the cases where an optimal solution was found, while Table 5.5 below collects information about the rest of the executions. Let us first focus on the upper subtable. The relative gaps are reasonably small in all cases. This means that the heuristic objective value is close to the optimal value, which at the same time is not far from the optimal value of the linear relaxation. The number of nodes that are processed in the branch-and-bound algorithm increases from the top to the bottom of the subtable. This seems reasonable, since the list of models in the subtable, (MP1), (MP1'), (MP2) and (MP2'), is ordered increasingly with the number of variables. The percentage of labeled points is not 100% even for models (MP1') and (MP2'), since labels containing points are discarded from our models. With respect to time, the heuristic algorithm is faster on average. Model (MP2') is the most time-consuming. Regarding Table 5.5, we can see an increase in the difficulty of solving the model when we go from the top to the bottom. First, model (MP1) does not appear in this subtable since it is optimally solved for all the testing instances. Thus, with model (MP1') we fail to solve to optimality 5.6% of the

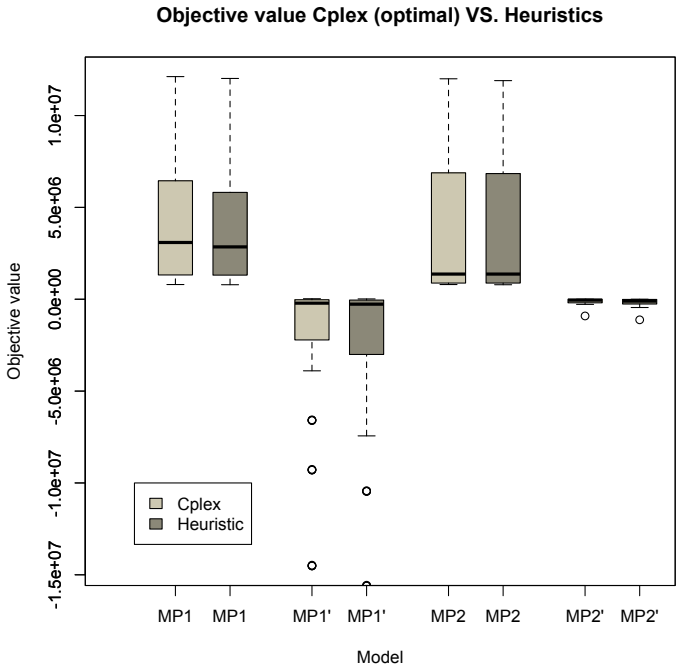
Model	LP	LB	OPT	Norm	Gap	Nod	% L	% O	Time	C1	C2	C3	C4	C5	$< T$
(MP1)	4,420,689	0	4,034,943	0.0115	7.4	2,578	82.1	0	690	5	6	10	16	366	-
			3,776,278	0.0107	5.7		78.7	0	8.2	6	8	10	14	339	
(MP1')	-1,920,043	-14,153,608	-2,257,847	0.0143	3.5	3,161	92.8	28.2	192.4	11	11	18	27	379	-
			-2,578,602	0.0174	3.5		92.8	32.5	47.5	12	13	20	28	374	
(MP2)	3,590,734	0	3,558,177	0.0103	1.5	4,490	96.4	0	396.6	-	-	-	-	-	20
			3,527,267	0.0115	1		95.4	0	0.34						18
(MP2')	-77,930	-980,238	-115,122	0.0116	3.3	6,364	98.9	5.1	881.4	-	-	-	-	-	26
			-155,820	0.0130	3.1		98.9	6.2	2.1						25

Table 5.4: Average experimental results for solved instances as a function of the models

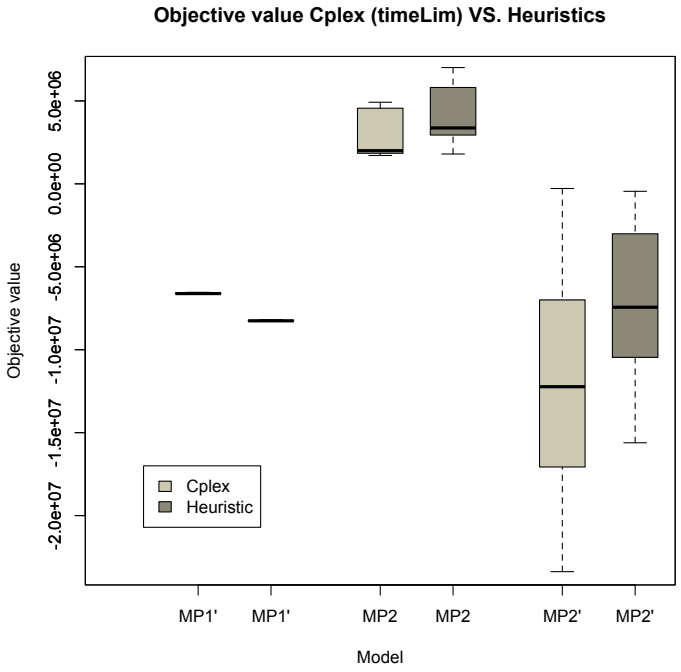
Model	LP	LB	OPT	Norm	Gap	Nod	% L	% O	Time	C1	C2	C3	C4	C5	$< T$	% NS
(MP1')	-5,400,223	-39,209,938	-6,598,044	0.0394	3.5	89,325	86	54.6	26,021.2	26	50	77	105	785	-	5.6
			-8,243,765	0.0503	5		86	68.2	318.7	34	56	84	108	761		
(MP2)	5,691,516	0	3,017,497	0	44.9	18,139	41.2	0	13,735.3	-	-	-	-	-	20	38.9
			4,138,563	0	-		52.4	0	21.3						189	
(MP2')	-5,164,972	-37,178,285	-11,780,008	0.0001	23.6	24,748	82.9	127.0	12,761.7	-	-	-	-	-	311	40.3
			-6,985,526	0.0001	-		82.9	76.6	157.2						388	

Table 5.5: Average experimental results for unsolved* instances as a function of the models

* Unsolved instances are those that were not solved to optimality within the time limit, which was one hour for each of their subproblems



(a) Instances optimally solved by CPLEX



(b) Instances not optimally solved by CPLEX

Figure 5.14: Comparison of objectives values from CPLEX and our heuristic over the models

	$n = 88$	$n = 137$	$n = 274$	$n = 692$	$n = 872$	$n = 1,213$
$h = 0.1$	2.3	2.8	6.8	8.9	10.5	11.1
$h = 0.5$	15.2	20.3	33.2	47.3	51	57.1
$h = 1$	29.8	36.8	52.4	69.5	73.3	80.1

Table 5.6: Subproblem percentage of discarded labels

instances. Nevertheless, in those cases the solution found by CPLEX is better on average than the heuristic solution. Finally, for models (MP2) and (MP2'), the heuristic procedure finds a better solution than CPLEX for 38.9% and 40.3% of the instances, respectively. In these latter cases we observe a significant relative gap, which confirms that the objective value of the solution found by CPLEX is far from the bound. The relative gap between the solution found by CPLEX and the heuristic one is omitted (set to '-'), since the latter is better this time. Results in both tables confirm that models (MP2) and (MP2') are harder to solve. For these models, even when instances are solved to optimality, the heuristic algorithm represents a good trade-off compared to CPLEX (see double entries of columns $Time$, $\%L$ for (MP2) and $\%O$ for (MP2') in the first subtable). We conclude that using the heuristic to solve models (MP2) and (MP2') is, on average, rewarding and supposes a good compromise between solution quality and running time.

Figure 5.14 depicts box-and-whisker plots for the objective function values over the different models. According with Figure 5.14a, (MP1') is the model for which the heuristic worst approaches the optimal solution found by CPLEX. On the other hand, Figure 5.14b confirms that the heuristic overcomes CPLEX when it fails to find the optimum of (MP2) and (MP2') within the time limit.

5.6.3 Results as a function of instance size

The instance size varies with the number of points, i.e., the data file, and with the size of the labels, which are both directly related with the number of potential overlaps. Thus, a larger size of the instance corresponds in general to models with more variables, but also more constraints. However, since labels that overlap an unrelated point are removed from the set of candidates, larger instances can also lead to simpler models. The portion of labels that are discarded under each configuration is shown in Table 5.6.

Tables 5.7 and 5.8 are devoted to the comparison of the average results produced when the size of the instances is varied.Table 5.7 refers to the instances solved to optimality, while Table 5.8 collects information of the rest of the cases. In both tables, the first column shows the height of the labels: 0.1, 0.5 and 1. For each height, the second column indicates the number of point features, n: 88, 137, 274, 692, 872 and $1,213$.

As we can see from Table 5.7, when the height is fixed and the number of points increases, the quality of the solution decreases and the running time is longer in general. However, when the size of the instance increases the running time does not always follow its lead. An increase of the instance size involves more potential overlaps between labels, but also between labels and points. Therefore, more labels are potentially discarded for the model, which can lead in extreme cases to easier instances. On the other hand, relative gaps are reasonably small, always below 15%.

Moving on to Table 5.8, we see the information about the instances for which CPLEX did not find the optimum. The number of such instances grows as the size of the instance increases. From the table, one could deduce that the problem of labeling towns with more than 5,000 inhabitants is harder when $h = 0.5$ than when $h = 1$. $\%NS$ is greater as are the number of

Size	n	LP	LB	OPT	Norm	Gap	Nod	% L	% O	Time
$h = 0.1$	88	441,441	-9,280	441,441	0	0	0	100	0	0
				441,441	0	0		100	0	0
	137	687,243	-18,878	687,243	0	0	0	100	0	0
				687,243	0	0		100	0	0
	274	1,374,487	-182,757	1,374,487	0.0003	0	0	100	0	0.2
				1,364,478	0.0003	1.5		99.8	0.2	0.1
	692	3,456,704	-544,341	3,426,252	0.0014	1.7	184	99.7	0.4	5
				3,376,172	0.0022	3.2		99.4	0.8	0.9
	872	4,319,564	-882,897	4,244,121	0.0010	3.2	809	99.4	1	43.9
				4,154,034	0.0010	3.8		99.0	1.6	1.7
	1,213	6,864,399	-1,107,263	6,759,572	0.0005	2.8	20,888	99.4	1	2,209.5
				6,629,489	0.0011	3.6		98.9	1.6	8.4
$h = 0.5$	88	428,722	-119,293	421,394	0.0075	1.9	0	98.9	1.1	0.2
				411,383	0.0081	2.9		98.3	1.7	0
	137	659,270	-313,908	637,121	0.0078	2.5	63	97.8	1.8	0.9
				627,098	0.0091	1.2		97.4	2.2	0.1
	274	1,167,724	-1,147,924	1,098,864	0.0159	3.0	3,942	92.5	4.7	106.2
				1,033,826	0.0179	3.0		91.6	6.2	1.0
	692	2,191,108	-6,895,253	1,621,902	0.0094	7.0	232	85.4	16.0	44.9
				930,631	0.0108	8.8		81.1	21.7	56.6
	872	2,055,931	-10,613,262	1,290,173	0.0116	7.4	24,280	82.3	22.4	1,085.9
				528,809	0.0140	7.5		78.7	27.5	92.9
	1,213	9,288,893	0	8,082,697	0.0319	13	44,080	66.4	0	11,757.2
				7,029,496	0.0262	13		57.8	0	52.9
$h = 1$	88	391,261	-214,339	351,248	0.0330	7.8	74	94.9	5.7	1.5
				346,204	0.0384	0.6		94.3	5.7	0.1
	137	534,398	-498,992	496,888	0.0071	3.5	2,975	92.7	8.0	79.0
				466,828	0.0088	2.9		91.2	8.8	0.9
	274	595,687	-2,935,596	397,828	0.0292	6.5	5,611	82.6	18.5	2,317.8
				183,122	0.0250	7.5		79.1	22.7	8
	692	-529,801	-16,857,912	-1,584,226	0.0403	13.4	1,508	63.9	47.7	258.7
				-2,240,348	0.0432	8.4		60.5	53.8	51.2
	872	-1,533,510	-27,566,838	-2,722,948	0.0417	13.5	253	58.4	53.3	81.3
				-3,619,318	0.0472	9.5		54.8	60	80.4
	1,213	-3,636,645	-50,535,710	-5,207,194	0.0448	15	1,696	48.4	59.9	467.7
				-6,108,731	0.0478	9.3		45.4	64.3	150.5

Table 5.7: Average results for solved instances as a function of the instance size

Size	File	LP	LB	OPT	Norm	Gap	Nod	% L	% O	Time	%NS
$h = 0.1$	1,213	-141,145	-2,610,002	-280,002	0.0012	5.6	230,875	99.9	2.3	21,821.7	12.5
				-450,002	0.0021	7.3		99.9	3.7	18	
$h = 0.5$	692	2,173,628	-6,905,003	-1,472,503	0	38	1,748	75.6	50.7	12,771.7	50
				914,994	0	-		81.1	21.7	57.1	
	872	2,034,856	-10,625,003	-2,755,003	0	37.3	1,396	71.8	58	14,578.4	50
				509,995	0	-		78.7	27.5	95.8	
	1,213	-521,368	-26,149,981	-6,275,183	0.0131	28.5	29,775	70.7	65.1	14,384.3	75
				-3,167,926	0.0168	-		76.6	45.5	234	
$h = 1$	274	965,791	-2,267,149	738,561	0	8.4	103,497	79.7	14.2	24,216.8	43.8
				548,560	0	7.1		76	17.5	6.6	
	692	-541,503	-16,865,006	-4,472,507	0	36.9	1,432	53.9	79.2	12,932.1	50
				-2,250,011	0	-		60.5	53.8	52.1	
	872	-1,546,661	-27,575,006	-7,271,257	0	40.5	341	47.1	94	11,028.6	50
				-3,630,011	0	-		54.8	60	82	
	1,213	-3,650,877	-50,545,006	-10,632,506	0	40.3	0	38.7	94.8	7,813.1	50
				-6,120,011	0	-		45.4	64.3	155.5	

Table 5.8: Average results for unsolved instances as a function of the instance size

Model	W	LP	LB	OPT	Norm Gap	Nod	% L	% O	Time	C1	C2	C3	C4	C5	$< T$
(MP1)	μ^1	4,409,804	0	4,024,716	0.0092 7.4	2,254	82.2	0	520.4	5	6	10	16	366	-
				3,766,803	0.0087 5.7		78.7	0	8.2	6	8	10	14	339	
	μ^2	4,414,182	0	4,028,813	0.0133 7.4	2,303	82.2	0	755.9	5	6	9	15	366	-
				3,770,601	0.0122 5.7		78.7	0	8.2	6	8	10	14	339	
	μ^3	4,411,945	0	4,026,685	0.0066 7.4	2,419	82.2	0	589.2	5	6	11	16	365	-
				3,768,634	0.0066 5.7		78.7	0	8.2	6	8	10	14	339	
	μ^4	4,446,828	0	4,059,558	0.0173 7.4	3,338	82.2	0	894.9	5	6	9	15	366	-
				3,799,075	0.0154 5.8		78.7	0	8.2	6	8	10	14	339	
(MP1')	μ^1	-1,930,646	-14,164,851	-2,268,495	0.0117 3.5	1,865	92.8	28.2	137.5	11	12	18	27	379	-
				-2,589,097	0.0141 3.5		92.8	32.6	47.5	12	13	20	28	374	
	μ^2	-1,926,363	-14,160,330	-2,264,196	0.0165 3.5	2,778	92.8	28.2	199.8	11	12	18	27	379	-
				-2,584,854	0.0198 3.5		92.8	32.6	47.6	12	13	20	28	374	
	μ^3	-1,928,503	-14,162,667	-2,266,348	0.0084 3.5	3,248	92.8	28.2	213.1	11	11	18	27	379	-
				-2,586,961	0.0105 3.5		92.8	32.6	47.6	12	13	20	28	374	
	μ^4	-1,894,660	-14,126,584	-2,232,349	0.0209 3.5	4,752	92.8	28.2	219.2	11	11	18	27	380	-
				-2,553,496	0.0252 3.5		92.8	32.6	47.5	12	13	20	28	374	
(MP2)	λ^1	3,590,733	0	3,558,173	0.0141 1.5	2,712	96.4	0	146.9	-	-	-	-	-	19
				3,527,263	0.0151 1		95.4	0	0.3						17
	λ^2	3,590,734	0	3,558,179	0.0125 1.5	4,330	96.4	0	363.5	-	-	-	-	-	19
				3,527,270	0.0136 1		95.4	0	0.3						17
	λ^3	3,590,734	0	3,558,174	0.0085 1.5	3,915	96.4	0	176.2	-	-	-	-	-	22
				3,527,265	0.0099 1		95.4	0	0.3						18
	λ^4	3,590,735	0	3,558,181	0.0060 1.5	7,002	96.4	0	899.7	-	-	-	-	-	21
				3,527,272	0.0076 1		95.4	0	0.3						19
(MP2')	λ^1	-58,573	-791,010	-87,010	0.0137 3.2	838	99.1	4.6	39.2	-	-	-	-	-	22
				-117,011	0.0176 2.8		99.1	5.4	1						21
	λ^2	-119,230	-1,200,003	-161,822	0.0131 3.3	5,216	98.2	7.2	1,923.1	-	-	-	-	-	31
				-208,185	0.0148 3.0		98.2	8.6	2.2						31
	λ^3	-72,687	-1,015,009	-115,009	0.0117 3.7	14,540	99.1	4.8	1,127.9	-	-	-	-	-	26
				-161,009	0.0138 3.3		99.1	5.8	2.8						27
	λ^4	-66,078	-956,365	-104,546	0.0093 3.4	5,679	99.1	4.3	461.5	-	-	-	-	-	27
				-147,274	0.0074 3.2		99.1	5.3	2.6						25

Table 5.9: Average results for solved instances as a function of the models and the vectors of weights

nodes and the running time. This could be explained by the fact that, while the number of points of the largest subproblem to solve is 1,212 in both cases (see Table 5.3), when $h = 0.5$, 57.1% of the labels are discarded and when $h = 1$, discarded labels account for 80.1% of the total. This is a practical example of how constantly increasing the size of the instances ceases to make sense at some point. It also justifies our choice of the testing instances size.

5.6.4 Results as a function of model parameters

Tables 5.9 and 5.10 are a refinement of tables 5.4 and 5.5 respectively, where we differentiate the vector of weights used with each model (recall Table 5.2 for the description of λ's and μ's). Looking at Table 5.9, we observe that, once the model is fixed, $\%L$ and $\%O$ remain the same as the vector of weights changes. This was the desired effect: varying the weights should change the measure of ambiguity but not the main objective, which is to maximize the labels placed or minimize the overlaps produced. It also proves that the choice of M is good in relation to the dimension of the instances. However, the effect apparently does not appear for model (MP2'). Note that for this model the percentage of non-solved instances, $\%NS$ changes when λ varies. Even when the percentage coincides for different λ's, the instances solved are not the same. This justifies the different values obtained for $\%L$ and $\%O$ by model (MP2'), since

Model	W	LP	LB	OPT	Norm	Gap	Nod	% L	% O	Time	C1	C2	C3	C4	C5	$< T$	%NS
(MP1')	μ^1	-5,422,560	-39,234,902	-6,615,300	0.0313	3.5	93,388	86	54.6	25,831.5	26	50	78	105	784	-	5.6
				-8,265,365	0.0403	5.1		86	68.2	318.7	34	56	84	108	761		
	μ^2	-5,413,411	-39,224,783	-6,616,146	0.0453	3.6	91,314	86	54.7	26,154.8	26	50	76	104	787	-	5.6
				-8,256,478	0.0580	5		86	68.2	318.5	34	56	84	108	761		
	μ^3	-5,417,650	-39,229,833	-6,620,352	0.0209	3.6	89,280	86	54.7	25,794.5	25	49	78	107	784	-	5.6
				-8,260,488	0.0290	5		86	68.2	318.9	34	56	84	108	761		
	μ^4	-5,347,270	-39,150,234	-6,540,378	0.0600	3.5	83,316	86	54.6	26,304	26	51	77	104	785	-	5.6
				-8,192,730	0.0739	5.1		86	68.2	318.7	34	56	84	108	761		
(MP2)	λ^1	5,691,515	0	2,974,283	0	45.8	15,326	40.7	0	13,089.2	-	-	-	-	-	17	38.9
				4,138,555	0	-		52.4	0	20.9						189	
	λ^2	5,691,517	0	2,944,285	0	46.4	22,267	40.1	0	15,325.4	-	-	-	-	-	16	38.9
				4,138,566	0	-		52.4	0	21.2						189	
	λ^3	5,691,516	0	3,005,708	0	45	12,879	41.1	0	13,383.6	-	-	-	-	-	17	38.9
				4,138,560	0	-		52.4	0	21.8						189	
	λ^4	5,691,517	0	3,145,713	0	42.3	22,083	43.1	0	13,143.2	-	-	-	-	-	32	38.9
				4,138,569	0	-		52.4	0	21.4						189	
(MP2')	λ^1	-4,712,260	-34,021,266	-10,782,514	0.0002	22.2	43,966	84.2	117.5	13,615.5	-	-	-	-	-	287	44.4
				-6,393,765	0.0003	-		84.2	70.9	144.8						358	
	λ^2	-5,281,751	-38,125,719	-12,458,576	0.0002	25.6	28,333	83.4	132.5	11,788	-	-	-	-	-	308	38.9
				-7,147,148	0.0003	-		83.4	75.2	160.7						391	
	λ^3	-5,365,276	-38,508,583	-11,548,583	0	22.1	8,316	81.9	123.1	12,937.4	-	-	-	-	-	331	38.9
				-7,242,869	0	-		81.9	80.5	162.2						403	
	λ^4	-5,365,273	-38,508,574	-12,472,859	0	24.9	15,630	81.9	135.9	12,584	-	-	-	-	-	321	38.9
				-7,242,859	0	-		81.9	80.5	163						402	

Table 5.10: Average results for unsolved instances as a function of the models and the vectors of weights

these percentages are not actually different if we compare them instance by instance. We have to look at column *Norm* to appreciate the effect of changing the vector of weights in each model, since it contains the information related to the ambiguity of the solution. Conversely, in Table 5.10, the solution found by CPLEX is not optimal and therefore $\%L$ and $\%O$ could vary when the vector of weights changes. Many cells in *Norm* are zero in this subtable. Note that low ambiguity is achieved here in detriment of more overlaps/ less labeled points. For models (MP1) and (MP1'), the number of labels in each class of ambiguity changes when μ varies, as was expected. For models (MP2) and (MP2'), the number of labels with ambiguity value less than T also varies when λ does. Figures 5.15 and 5.16 give more detailed information of this effect.

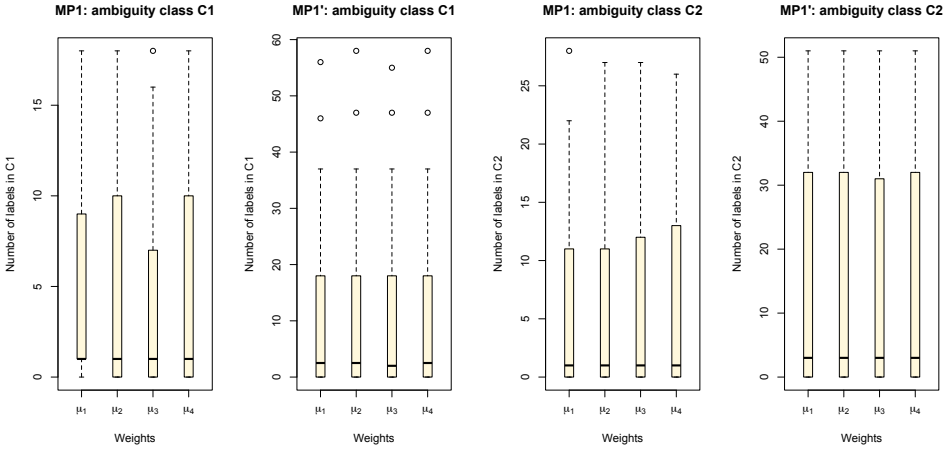
Figure 5.15 depicts how many labels in the solutions of (MP1) and (MP1') are in each ambiguity class, $C_1, \ldots, C_5$. Data on this figure refer to solutions found by CPLEX, regardless of whether they are optimal or reported after the time limit. We shall now look at the left-hand side of each subfigure, i.e., at the number of labels in each class for model (MP1). Such diagrams suggest that μ^3 is a good choice for the vector of weights. It not only produces fewer labels belonging to C_1 (the class with most ambiguous labels), but also more labels in classes C_3 and C_4 (which collect moderate and little ambiguous labels). Regarding the class of non-ambiguous labels, C_5, most of the labels produced with μ^3 belong to it, similarly as for μ^1, μ^2, μ^4 and μ^5. To understand why μ^3 produces well balanced labelings, we shall give some insight into the interpretation of the chosen values for μ. First, $\mu^1 = (1, 2, 3, 4, 5)$ gives weight i to class C_i, $i = 1, \ldots, 5$. Since we are maximizing the objective function, these weights favor labels in the last classes. However, due to the monotonous increment of weights in one unit over the classes, moving one label from class C_1 to C_2 has the same benefit as changing it from C_4 to C_5. Conversely, for $\mu^2 = (1, 2, 4, 8, 16)$ rearrangement between last classes are considered more

profitable than between the first ones (compare for instance the difference of 8 units between μ_4^2 and μ_5^2 with the difference of 1 between μ_1^2 and μ_2^2). Observe as well that, in this case, two labels in opposite extreme classes C_1 and C_5, with profit $1+16 = 17$, are preferable to a more balanced distribution among classes C_3 and C_4, which would increase the objective in $4 + 8 = 12$. The choice of weights for vector $\mu^3 = (1, 4, 8, 9, 10)$ tries to avoid unbalanced labelings. In this case, for instance, $\mu_1^3 + \mu_4^3 = 10 < \mu_2^3 + \mu_3^3 = 12$ and $\mu_1^3 + \mu_5^3 = 11 < \mu_3^3 + \mu_4^3 = 17$. At the same time, this third choice of μ gives preference to rearrangement of labels from the most ambiguous classes forward (note high relative increments among the first weights in μ^3 compared with the unitary increment between its last components). Finally, $\mu_4 = (0.01, 0.1, 1, 10, 100)$ is just a geometric sequence with ratio 10. This vector of weights produces the same effect as μ_2 but labels in the last classes are more strongly prioritized in this case. We shall move on to the right-hand side of each subfigure, i.e., the results obtained for model (MP1'). Regarding the number of labels in each class, no significant differences are found this time when μ varies. Vector μ_3 could be considered slightly better, since it presents fewer labels on average in class C_1 and a shorter deviation for class C_2. Model (MP1') forces the labeling of every point feature, trying to get the minimum number of overlaps as first objective and the minimum ambiguity as the second. Once the number of overlaps in a solution has been determined, overlapping labels would belong to class C_1. Therefore, there are some labels that must belong to this class, regardless of the choice of μ. According to the diagrams, this precondition seems not to give room for a wide variety of distributions of the remaining labels, which must not produce new overlaps, among the different classes.

Figure 5.16 shows how many labels in the solution found by CPLEX are ambiguous (i.e., have ambiguity value below the threshold T) for models (MP2) and (MP2'). This information is represented for each of the four different vectors of weights used in the experiments. When using λ^1 and λ^2 only the five most ambiguous labels are considered, with different weights in the first case and same weights in the latter. Vector λ^3 concerns only the most ambiguous label in the solution. Finally, when λ^4 is the vector of weights, the total ambiguity of the 1st and 10th most ambiguous labels in the solution is minimized. For (MP2), λ^3 is the vector of weights producing fewest ambiguous labels, while λ^4 is the vector that produces the most number of ambiguous labels. This is reasonable since, with λ^3, the most ambiguous label in the solution is as unambiguous as possible, so setting a bound on the ambiguity of the rest of the labels. Conversely, λ^4 stands for a tradeoff between the ambiguity of the 1st and 10th most ambiguous labels. Taking now the results for (MP2') depicted in the right-hand side of the figure, we observe that the choice of λ does not have a significant impact on the number of unambiguous labels in the solution. This also happens when we compare (MP1) and (MP1') in Figure 5.15 and is related with the fact that models (MP1) and (MP2) allow unlabeled points while (MP1') and (MP2') do not.

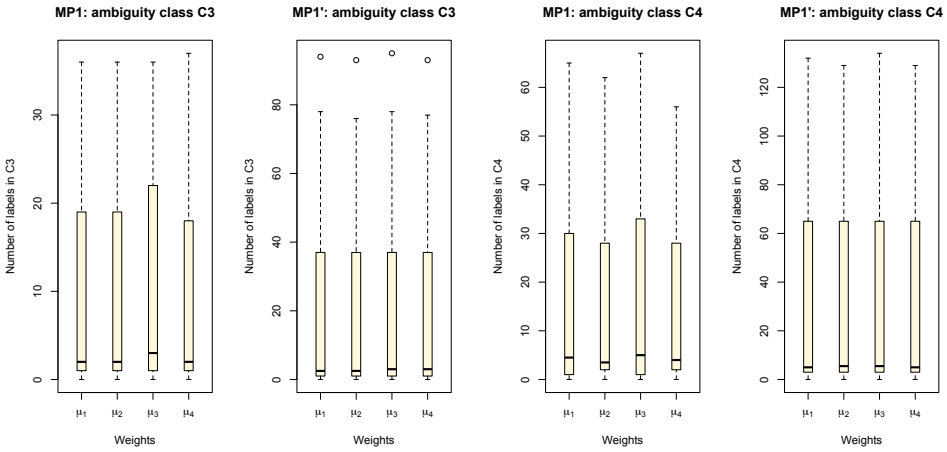
5.6.5 Average running times

Finally, Table 5.11 shows the time needed to solve each model to optimality, depending on the number of point features of the instances. A rigorous comparison with previous models tested on instances of similar sizes is not possible due to the different settings of the experiments. Nevertheless, these tables may provide the reader with an idea of the relative performance of our approach. Model (MP1) can be seen as a particular case of formulation (Z) by Zoraster (1990) with improved constraints and labels priorities based on ambiguity. Similarly, (MP1') is an improvement of (RL) (Ribeiro & Lorena, 2008), where labels priorities depend on their ambiguity. Therefore, (MP1) and (MP1') should perform at least as well as these previous



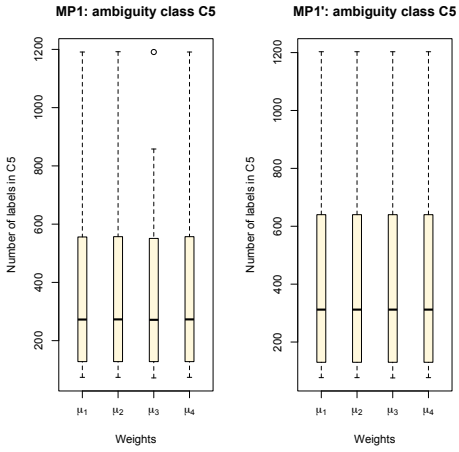
(a) Number of labels in class C_1

(b) Number of labels in class C_2



(c) Number of labels in class C_3

(d) Number of labels in class C_4



(e) Number of labels in class C_5

Figure 5.15: Number of labels in each ambiguity class in the solution found by CPLEX

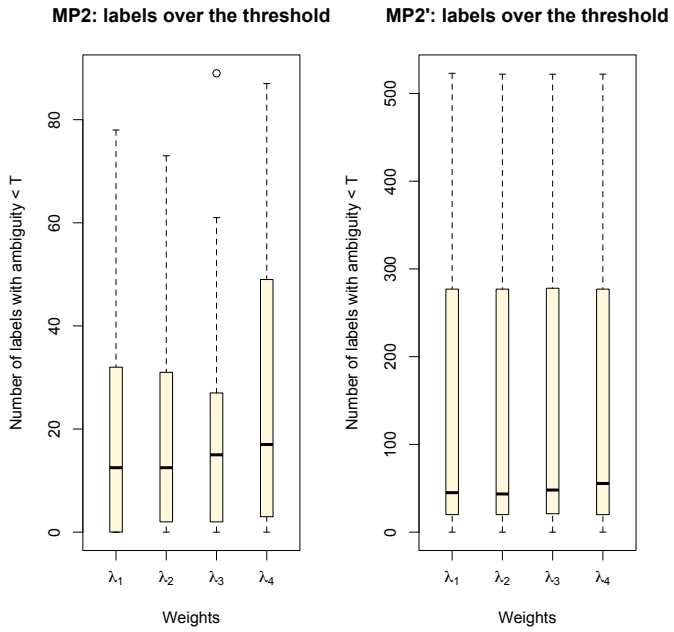


Figure 5.16: Number of labels below the threshold T in the solution found by CPLEX

$h = 0.1$	n	(MP1)	(MP1')	(MP2)	(MP2')
	88	0.0138	0.0129	0.0135	0.0019
	137	0.0285	0.0255	0.0308	0.0038
	274	0.2525	0.0943	0.2060	0.0961
	692	0.8005	1.1003	10.4792	7.4827
	872	3.1823	10.7719	78.4389	83.2927
	1,213	13.0936	16.4320	3764.3517	7878.6107
$h = 0.5$	n	(MP1)	(MP1')	(MP2)	(MP2')
	88	0.0577	0.0839	0.3518	0.3522
	137	0.1312	0.3975	1.4610	1.6346
	274	0.3831	1.0605	283.6772	139.5103
	692	23.20	66.6595	-	-
	872	63.96	2107.8344	-	-
	1,213	11757.23	-	-	-
$h = 1$	n	(MP1)	(MP1')	(MP2)	(MP2')
	88	0.0906	0.1264	2.8169	2.7792
	137	0.1771	0.7282	220.4016	94.6540
	274	1.9728	7.0889	-	20824.3579
	692	75.4400	441.9159	-	-
	872	139.4286	23.2473	-	-
	1,213	342.2344	593.1018	-	-

Table 5.11: Average CPU time (s) for instances solved by CPLEX to optimality

$h = 0.1$	n	(MP1)	(MP1')	(MP2)	(MP2')
	88	0.0054	0.0151	0.0057	0.0151
	137	0.0106	0.0328	0.0114	0.0338
	274	0.0452	0.1467	0.0471	0.1552
	692	0.2439	1.2426	0.2561	1.7467
	872	0.4320	2.5323	0.4536	3.5231
	1,213	2.1241	15.5419	2.5021	18.2219
$h = 0.5$	n	(MP1)	(MP1')	(MP2)	(MP2')
	88	0.0142	0.0315	0.0149	0.0329
	137	0.0298	0.0855	0.0313	0.0903
	274	0.2168	1.4056	0.2258	2.1565
	692	14.7862	98.4754	13.5544	100.6715
	872	25.6473	160.2071	25.4116	166.2849
	1,213	52.9478	318.6947	56.8880	326.3519
$h = 1$	n	(MP1)	(MP1')	(MP2)	(MP2')
	88	0.0342	0.1704	0.0358	0.1800
	137	0.1594	1.3383	0.1652	2.1307
	274	1.4488	12.9805	1.3396	13.7875
	692	10.1916	92.2951	9.7540	94.5399
	872	15.0327	145.8170	15.3891	148.6846
	1,213	24.8154	276.1217	26.9102	284.0492

Table 5.12: Average CPU time (s) to obtain a heuristic solution

ILP formulations, and differences will be due only to the implementation and machine used. Conversely, models (MP2) and (MP2') are more complex than previous ILP formulations, not only due to the number of variables and constraints (which is comparable to the size of (KM)) but also because of the idea they enclose. This increase of the complexity is clearly reflected in the CPU times reported in Table 5.11, where we observe that the formulation is not even able to scale for the largest instances. The difference is not so dramatic when we compare CPU times of the heuristic for (MP1) and (MP1') with the same times for (MP2) and (MP2') (Table 5.12). The reason is that, unlike the exact method, the heuristic does not vary its computational complexity when we change from the first models to the last.

Chapter 6

Spotting key members in networks

The interest towards key nodes in networks emerged in the past century as a subject of Mathematical Sociology and Graph Theory, and has grown to the point of becoming one of the most noteworthy challenges towards understanding social systems. State-of-the-art of network analysis includes discerning the relevance of a group of nodes as network representatives. Many sophisticated approaches based on techniques borrowed from disciplines such as Physics and Statistics have been developed. However, only a few works use mathematical optimization to address group relevance.

This chapter presents a mathematical programming formulation for identification of the group of most relevant nodes and their communities. The initial idea was to explore the use of eigenvector centrality, a well-known measure for individual nodes, to spot the group of key members of a network. We realized that real networks are usually a combination of functional subunits, known by social scientists as *communities*. Aiming at coverage, it appeared natural to assume that targeted key members will belong to different communities. Our approach emerged then as a combination of clustering, which uncovered the communities, and eigenvector centrality, which quantified group relevance. Namely, a representative is chosen for each cluster to be in the group of key members. The network clustering yielding maximum overall relevance of the representatives, which is calculated as an eigenvector centrality, is selected.

Several interesting questions concerning the field of Operations Research arose when modeling eigenvector centrality with clustering. The first one was to write a mathematical program that we could manage. Naive attempts to model eigenvector centrality over the clusters lead to highly nonlinear programs. Solving them by a sequential linearization approach turned out inoperative. In the end, a more elaborated modeling of eigenvector calculation over the clusters, which includes additional decision variables and constraints, resulted in a mixed-integer linear program for the problem. Furthermore, different ways of improving the linear formulation as well as the computational times were studied. First, variables reduction was investigated for undirected networks. A second difficulty concerned symmetry breaking in integer programming. Symmetry arose from the decision variables and constraints that model network clustering, which produced a particular case of the set partitioning problem. Rediscovering the facets of the partitioning orbitope introduced by Kaibel & Pfetsch (2008) allowed us to break the symmetry and enhance our formulation.

This chapter addresses the technical details of the modeling process and ultimately focuses on the applicability of the resulting formulation. Discussion includes experiments on small real-life networks that reveal previously unnoticed key members. Additionally, clusters will prove to be consistent with previous knowledge on the community structure of the networks. Computational experience on larger synthetic networks demonstrates an adequate scalability

of the method, which will be able to find optimal solutions for networks of hundreds of nodes and thousands of links.

The main contributions in this chapter can be summarized as follows. First, an innovative model that combines optimization of group relevance and community discovery in the same process is presented. Second, our exact approach shows the potential of mathematical programming to uncover complex network structures, a context where heuristics abound. Finally, the proposed model serves as a suitable adaptation of widespread PageRank to the problem of group centrality.

6.1 Introduction

Identifying key members in a social network is critical to understand the underlying system behaviour. When social networks analysis was still in its infancy, different strategies were explored in order to determine the relevance of a single node, which gave rise to the so-called centrality measures. Some of the classics are based on local criteria such as the number of connections with other nodes of the network (*degree centrality*), the number of shortest paths that contain the node (*betweenness centrality*, Freeman, 1977; Newman, 2005)) or the distance between the node in question and the rest of the nodes in the network (*closeness centrality*, Freeman, 1978; Sabidussi, 1966). A different approach assumes that one node's importance not only depends on its connections to the rest of the network, but also on the importance of its neighbors. Translating such recursive definition into a mathematical formula yields the search of the eigenvectors of a matrix, the *matrix of relationships* in Sociology and the *adjacency matrix* in Graph Theory. The result is a decentralized measure that has been known as *eigenvector centrality* (see Bonacich, 1972; Katz, 1953) and inspired popular Google's method for rating web pages, PageRank (see Page et al., 1999).

A first thought to identify a group of key members was to leverage previous knowledge of the problem on a single node. But measures designed to discern the most central member fail to identify a central group. Indeed, computing some of the above-mentioned measures for every node gives a ranking from where selecting those with highest individual relevance, but says nothing about their potential influence power as a group, though. As an illustrative example, Figure 6.1 depicts the seven characters with highest eigenvector centrality in *Les Misérables* network. This network is a popular benchmark for social science algorithms that represents characters in the novel *Les Misérables*. Links connect any pair of characters that coappear in the novel— weights to quantify the number of such coappearances are also considered, as compiled by Knuth (1993). Evidence that eigenvector centrality fails to comply with network coverage is noticeable from the figure; most of the top-7 characters fall in the same area. One of the aspects that become crucial when determining joint relevance of multiple members is their relative distance, see Kitsak et al. (2010); Zhang et al. (2016). Think about a marketer who wants to sell a product. Targeting the two top influencers among its potential clients seems a sensible strategy. However, if these celebrities happen to have lots of followers in common, it seems more reasonable to replace them by a pair whose influence covers a bigger portion of clients, even if they are less popular. While classical degree, closeness and betweenness centralities have already been adapted to quantify group relevance (see Borgatti, 2006; Everett & Borgatti, 1999), eigenvector centrality has been not.

We explore the use of eigenvector centrality to spot the key members of a network, accounting for those fragments covered by their influence. The latter will form a partition of the network and ultimately reveal its underlying organization. In fact, real networks usually display a modular structure that emanates from the combination of compartments or func-

tional subunits, identified by social scientists as *communities*, see Lancichinetti et al. (2008). Our approach assumes that targeted key members will tend to be in different communities, something that was already suggested in Li et al. (2016). Figure 6.2 shows the group of seven most relevant characters in *Les Misérables* network (diamond nodes) together with their communities (in different colors), according to our approach. The solution found exhibits a more balanced distribution of key nodes than that of Figure 6.1 and further reveals the structural relations between the characters of the novel. Valjean, the protagonist, has been identified as a member of the top group. However, Javert, the antagonist, has been not. A community only admits one leader; given two competing nodes, the model automatically discerns whether the sphere of influence of one of them should absorb the other candidate or, conversely, there is room for two leaderships. In this case, Javert is eclipsed by Valjean and embedded in his community. Many other major characters are covered by Valjean's influence sphere. In their place, our method prefers secondary characters such as bishop Myriel or Courfeyrac to build a more balanced and covering group of leaders. Case of Child1 is specially remarkable. The character, of little significance for the novel, is, according to our solution, among the seven most relevant ones. Indeed, Child1 and Child2 are to some extent isolated from the rest, and it is precisely this distinct role what the model detects. Gueulemer is identified as the leader of the group consisting of the gang of criminals made of Montparnasse, Claquesous, Babet and himself, and Brujon, another criminal in the novel. Our formulation uses a coefficient to control how cohesive clusters in the solution are, which is named *mixing parameter* and denoted by μ. Solution depicted on Figure 6.2 has been obtained under mixing parameter $\mu = 0.46$. This means that node relations within a cluster represents, for each node, at least a 54% of its relations. The mixing parameter is an input of our model that can be tuned to control the cohesiveness of the spheres of influence (clusters) in the solution.

Approaches to address group relevance have traditionally overlooked network modularity, while community detection algorithms have ignored key nodes. Only in recent years has community discovery led by key nodes become a hot research topic in network analysis. In Li et al. (2015), key nodes are first identified in accordance to their relevance and relative dispersion to be then used as seeds for a k-means clustering. A different strategy also uses the key nodes as seeds but interprets clustering as the result of a non-cooperative game where every node tries to maximize its own social identity, see Bu et al. (2018). Other approaches use statistical models, such as kernel functions in Li et al. (2016); Zhang et al. (2009) or unbiased random walks in Stanoev et al. (2011), to represent relevance and select a number of centers. In a second step, community memberships are determined by evolving a dynamical system where centers memberships are immutable, see Li et al. (2016); Stanoev et al. (2011). Our approach is conceptually different from the previous ones because community discovery and key nodes identification go hand in hand.

For previous mathematical programming formulations to approach group relevance, we can cite Arulselvan et al. (2009), who presented an integer linear program that determines which nodes to remove so that the remaining ones has minimum pair-wise connectivity. More recently, Furini et al. (2019) studied the clique interdiction problem, which consists in removing at most k nodes from a graph in such a way that the size of the maximum clique in the remaining graph is minimized. Fischetti et al. (2018) looked at the problem from a different perspective. They proposed several integer linear programming formulations for least cost influence propagation. They incorporated activation functions to represent which nodes are reached by influence propagation. Based on the formulations, the authors developed an exact method and also a heuristic algorithm that hinges on column generation.

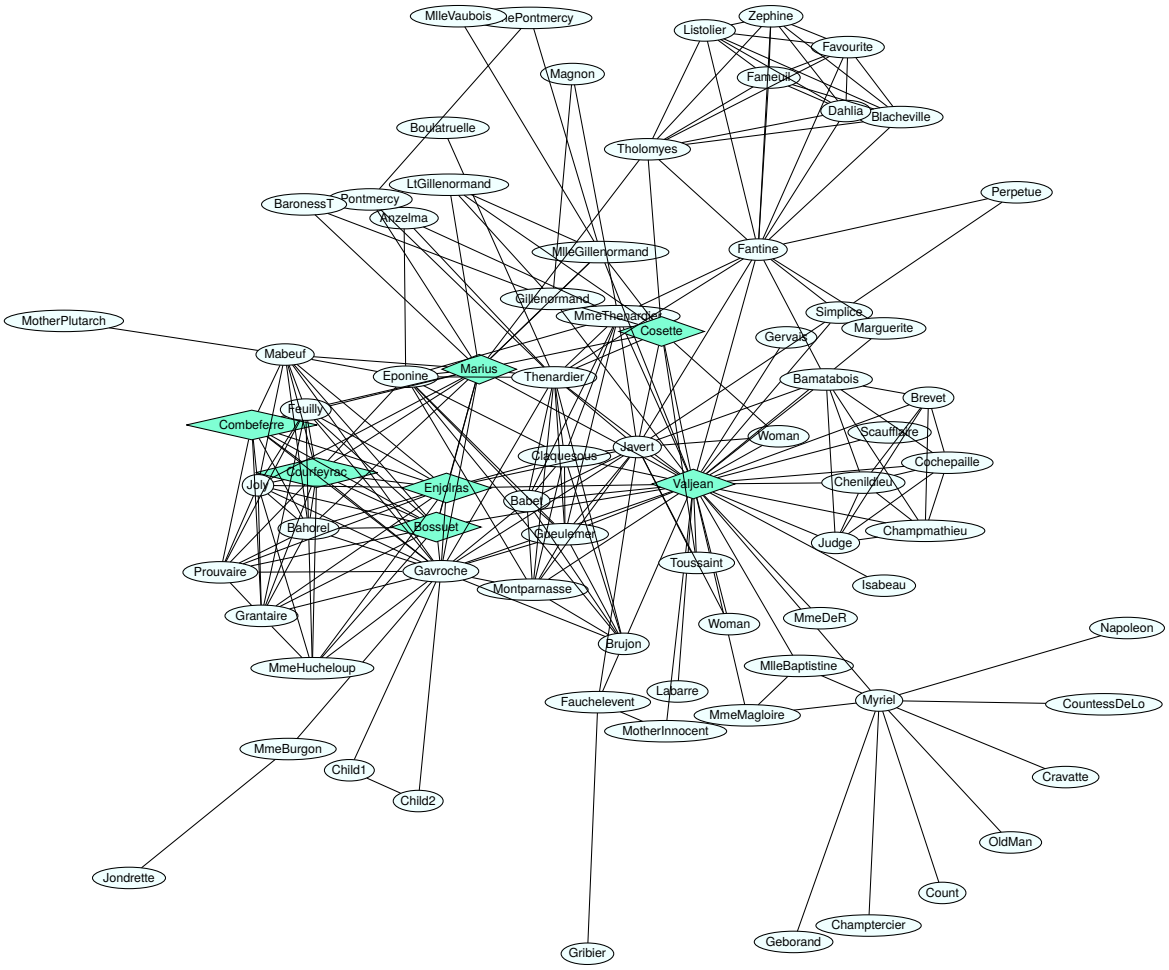


Figure 6.1: Top-7 nodes for *Les Misérables* network, according to eigenvector centrality

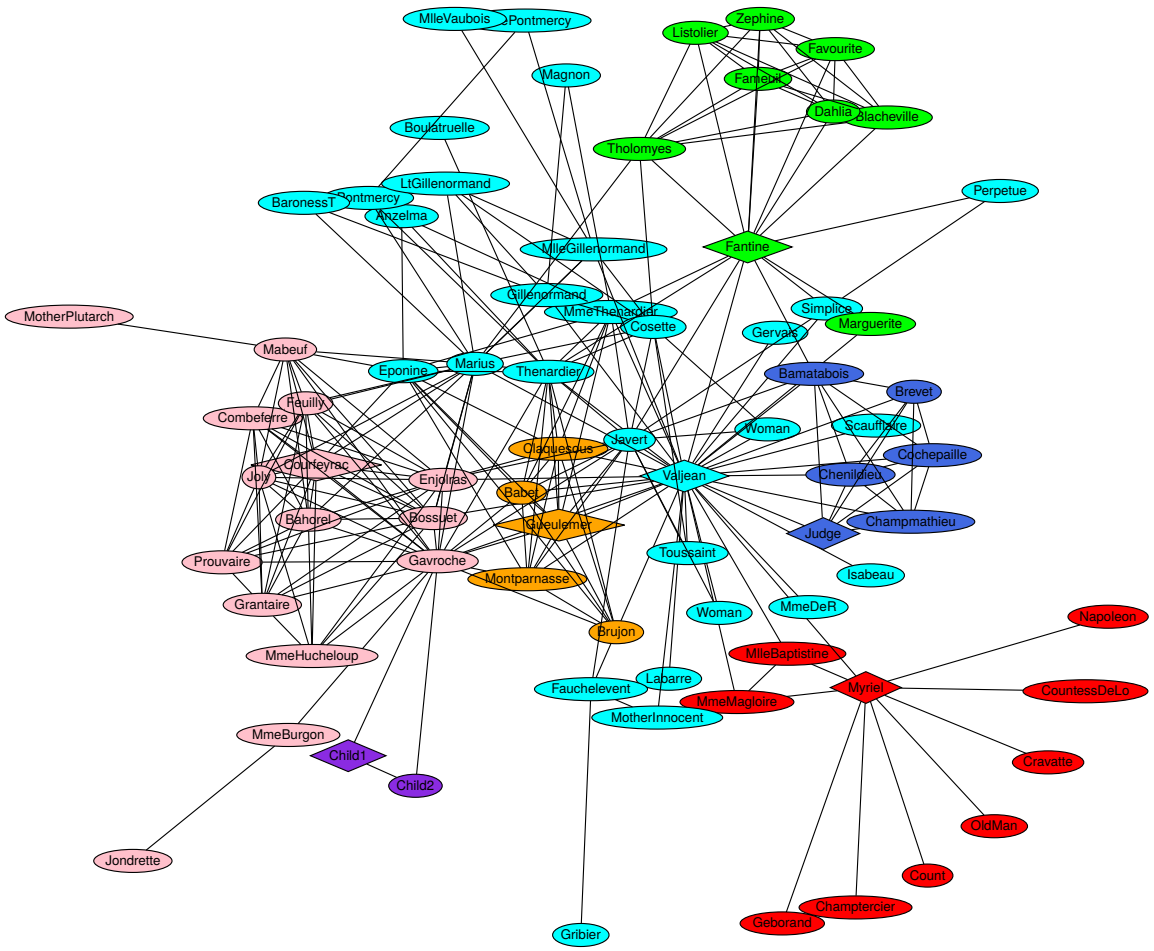


Figure 6.2: Optimal solution for *Les Misérables* network with $p = 7$ clusters and mixing parameter $\mu = 0.46$

6.2 Modeling node relevance

Let (V, E, W) be a weighted undirected graph with n nodes, $V = \{1, \ldots, n\}$, and matrix of weights W symmetric and positive: $W = (w_{ij})$ with $w_{ij} = w_{ji}$ and $w_{ij} \geq 0$ for every pair $i, j \in V$. Weight w_{ij} represents the strength of edge (i, j), if it exists, and it is 0 otherwise. To present node relevance estimation, which will ultimately determine the clusters centroids, we first introduce the concept of eigenvector centrality. Even though we will only consider undirected graphs during our exposition, eigenvector centrality for the directed case is analogous. Our proposal is also easy to adapt for directed graphs.

Eigenvector centrality

Eigenvector centrality of one node $j \in V$, π_j, is defined as a weighted sum of the eigenvector centralities of its neighbors Page et al. (1999); Hubbell (1965):

$$\lambda \pi_j = \sum_{i=1}^{n} w_{ij} \pi_i. \tag{6.1}$$

Equation (6.1) indicates that one node's importance depends on how notably it is linked to other nodes and how important are such. This equation is equivalent to determining some of the left eigenvectors of W and its associated eigenvalue. Indeed, (6.1) yields the eigenvector equation $\lambda \pi = \pi W$. Solutions for such equation are only guaranteed to exist in $\mathbb{R}^n$ under some especial conditions. Particularly, when all the entries of W are strictly positive and the sum of any of its rows equals one— W is row stochastic— , W can be reinterpreted as the *transition matrix* of an irreducible, aperiodic and positive recurrent Markov chain, see Serfozo (2009). This fact ensures that $\lambda = 1$ is the largest eigenvalue of W with associated eigenvector π unique and such that $\pi_i > 0$ and $\sum_{i=1}^{n} \pi_i = 1$. Eigenvector π is then called a *stationary distribution* of the Markov chain.

To guarantee solution existence, a fixed score e_j is usually added to the eigenvector centrality of j:

$$\lambda \pi_j = \sum_{i=1}^{n} w_{ij} \pi_i + e_j, \tag{6.2}$$

that is, $\lambda \pi = \pi W + e$, where $e = (e_j)$, $e_j > 0$. Note that, if $\sum_{i=1}^{n} \pi_i = 1$, we can rewrite previous equation as $\lambda \pi = \pi(W + \mathbb{1}^T \times e)$ where $\mathbb{1} = (1, \ldots, 1)$ is the row vector of n ones, i.e., π is a left eigenvector of $W + \mathbb{1}^T \times e$, whose entries are strictly positive. If $W + \mathbb{1}^T \times e$ was row stochastic, $\lambda \pi = \pi(W + \mathbb{1}^T \times e)$ would have a solution with $\lambda = 1$ and associated eigenvector π in $(\mathbb{R}^+)^n$ such that $\sum_{i=1}^{n} \pi_i = 1$, which would be also a solution to (6.2). Vector e might have different interpretations, including the "exogenous contribution" to a person's status, see Hubbell (1965), and the probability that an Internet surfer clicks on links at random, see Page et al. (1999).

Clustering embedded eigenvector centrality

To introduce eigenvector computation for uncovering the groups of key nodes, let $\{C_k\}_{k=1}^{p}$ denote a partition of V in p clusters, i.e., $\dot{\cup}_{k=1}^{p} C_k = V$. We consider $W^\epsilon = (w_{ij}^\epsilon)$, where $w_{ij}^\epsilon = w_{ij} + \epsilon$ and $\epsilon > 0$. For each node i belonging to C_k, we denote with π_{ik} the eigenvector

centrality of i within C_k. That is, node eigenvector centrality within cluster C_k, $\pi_{\cdot k}$, is computed as a solution of the following system:

$$\pi_{jk} = \sum_{i \in C_k} \pi_{ik} \frac{w^{\epsilon}_{ij}}{\sum_{\ell \in C_k} w^{\epsilon}_{i\ell}} \quad \forall j \in C_k \tag{6.3}$$

$$0 < \pi_{ik} < 1 \quad \forall i \in C_k \tag{6.4}$$

$$\sum_{i \in C_k} \pi_{ik} = 1. \tag{6.5}$$

Equation (6.3) is equivalent to $\pi_{\cdot k} = \pi_{\cdot k} W^{\epsilon,k}$, where $W^{\epsilon,k} := \left(\frac{w^{\epsilon}_{ij}}{\sum_{\ell \in C_k} w^{\epsilon}_{i\ell}}\right)_{i,j \in C_k}$ is the submatrix of W^{ϵ} induced by C_k and normalized by rows. Normalization conceptually means that the strength of one link ij is relative to the total influence power exerted from i, computed as the sum of the corresponding row. Since $W^{\epsilon,k}$ has positive entries and is row stochastic ($\sum_{j \in C_k} \frac{w^{\epsilon}_{ij}}{\sum_{\ell \in C_k} w^{\epsilon}_{i\ell}} = 1$ for all $i \in C_k$), the system (6.3)-(6.5) has a unique solution $\pi_{\cdot k}$ for each C_k. Note that the introduction of parameter ϵ is the equivalent of taking uniform vector $e = (\epsilon)$. Despite its possible interpretations, Page et al. (1999); Hubbell (1965), adding ϵ responds to a technical need: solution existence cannot be guaranteed otherwise. Therefore, we advise to set ϵ sufficiently small to prevent network nature from disruption.

6.3 Mathematical programming formulation

In order to model both, node-cluster assignment and eigenvector calculation, we define the following mathematical programming variables for all $i \in V$ and $k = 1, \ldots, p$:

$x_{ik} \in \{0,1\}$,	$x_{ik} = 1$ iff $i \in C_k$;
$\Pi_{ik} \in [0,1]$,	the eigenvector centrality of i within C_k if $i \in C_k$, 0 otherwise;
$y_{ik} \in \{0,1\}$,	$y_{ik} = 1$ iff i is the node with maximum eigenvector centrality within C_k;
$z_k \in [0,1]$,	the maximum eigenvector centrality in C_k, i.e., $z_k = \sum_{i=1}^{n} \Pi_{ik} y_{ik}$.

Variables Π encode p eigenvectors, one per cluster. For each k, positive components of the vector of variables $\Pi_{\cdot k} = (\Pi_{1k}, \ldots, \Pi_{nk})$ correspond with node eigenvector centralities within C_k. Observe that we model eigenvector centralities as normalized vectors, i.e., $\Pi_{ik} \in [0,1]$. Variables y represent clusters centroids, i.e., $y_{ik} = 1$ if and only if $\Pi_{ik} = \max\{\Pi_{jk} : j \in C_k\}$.

Given a network partition $\{C_k\}_{k=1}^{p}$, our vectors of variables $\Pi_{\cdot k} = (\Pi_{1k}, \ldots, \Pi_{nk})$, $k = 1, \ldots, p$, must satisfy the following constraints (stationary distribution):

(i) $\Pi_{jk} = x_{jk} \sum_{i=1}^{n} x_{ik} \Pi_{ik} \frac{w^{\epsilon}_{ij}}{\sum_{\ell=1}^{n} w^{\epsilon}_{i\ell} x_{\ell k}}$, for all $j = 1, \ldots, n$.

(ii) $0 \leq \Pi_{ik} \leq 1$ for all $i = 1, \ldots, n$.

(iii) $\sum_{i=1}^{n} \Pi_{ik} = 1$.

These conditions are just a translation of system (6.3)-(6.5) into our modeling language with decision variables. Note that condition (i) not only indicates eigenvector computation inside C_k, but also ensures that $\Pi_{\cdot k}$ components are zero for nodes outside C_k.

Any solution must additionally fulfill some constraints regarding clustering cohesion. Such constraints restrict the cluster-external relations of one node to represent at most a fraction μ

of its relations within the entire network. The incorporation of this parameter to the model avoids dummy solutions in which clusters have disconnected components or are singletons. But, more interestingly, μ allows us to reinterpret clusters as network communities. A community is usually defined as a group of nodes such that the density of links between nodes of the group is higher than the average link density in the network, Fortunato & Castellano (2012). In fact, $1-\mu$ determines community cohesion and μ has been known as the *mixing parameter*, Lancichinetti et al. (2008).

The group of top relevance nodes— the cluster centroids— is made of those with the highest eigenvector component in each cluster. The objective function is to find the partition in clusters that maximizes the sum of centroids' eigenvector components. Note that the fact that eigenvectors are normalized prevents the optimization to produce biased solutions.

We have now the ingredients to present the following mixed-integer non-linear formulation for the problem of group centrality, where $i, j \in V$ and $k = 1, \ldots, p$:

$$\max \quad \sum_{k=1}^{p} z_k \tag{6.6}$$

$$\text{s.t.} \quad \sum_{k=1}^{p} x_{ik} = 1 \qquad \forall i \tag{6.7}$$

$$\sum_{i=1}^{n} y_{ik} = 1 \qquad \forall k \tag{6.8}$$

$$\sum_{i=1}^{n} \Pi_{ik} = 1 \qquad \forall k \tag{6.9}$$

$$\Pi_{jk} = x_{jk} \sum_{i=1}^{n} x_{ik} \Pi_{ik} \frac{w^{\epsilon}_{ij}}{\sum_{\ell=1}^{n} w^{\epsilon}_{i\ell} x_{\ell k}} \qquad \forall j, k \tag{6.10}$$

$$\Pi_{ik} \leq x_{ik} \qquad \forall i, k \tag{6.11}$$

$$y_{ik} \leq x_{ik} \qquad \forall i, k \tag{6.12}$$

$$z_k \leq \Pi_{ik} + 1 - y_{ik} \qquad \forall i, k \tag{6.13}$$

$$y_{ik} \leq 1 + \Pi_{ik} - \Pi_{jk} \qquad \forall i, j, k \tag{6.14}$$

$$\sum_{j=1}^{n} (w_{ij} + w_{ji}) x_{jk} \geq (1-\mu) x_{ik} \sum_{j=1}^{n} (w_{ij} + w_{ji}) \qquad \forall i, k \tag{6.15}$$

$$\Pi_{ik}, z_k \geq 0 \qquad \forall i, k$$

$$x_{ik}, y_{ik} \in \{0, 1\} \qquad \forall i, k.$$

The objective function (6.6) is the sum of the greatest eigenvector components in each cluster. Constraints (6.7) impose that each node is assigned to only one cluster, and (6.8) guarantee that every cluster has only one centroid. Regarding eigenvectors, constraints (6.9) and (6.10) ensure that stationary distribution equations (iii) and (i) are respectively satisfied. On the other hand, (6.11) and (6.12) prevent Π_{ik} and y_{ik} to be positive when i is not in cluster C_k. Constraints (6.13) have no effect when i is not a centroid and upperly bound z_k by Π_{ik} when it is; since (6.6) is to be maximized, z_k will attain this bound in the optimal solution. Constraints (6.14) ensure that binary variable y_{ik} is zero if there is a node with eigenvector component greater than Π_{ik}. Note that they are not necessary to have a valid formulation because, in an optimal solution, y_{ik} will always be one for the node with maximum centrality in the cluster. Nevertheless, they serve to enhance the formulation. Finally, (6.15) guarantee clusters cohesion as follows. Given that $i \in C_k$ (i.e., $x_{ik} = 1$ on the right hand side of (6.15)), the overall relations of i with nodes inside C_k (on the left hand side) must be greater than a fraction $1-\mu$ of its overall relations within the network, $\sum_{j=1}^{n}(w_{ij} + w_{ji})$. On the other hand,

observe that cluster cohesion is imposed using W— and not W^ϵ— as a reference. Last lines in the formulation determine the type of the variables— positive (Π and z) and binary (x and y).

Even though the previous is a valid formulation, it has a main drawback, namely that (6.10) is highly non-linear. In order to linearize it, we introduce two families of continuous variables, T and F, for all $i, j \in V$ and $k = 1, \ldots, p$:

$$T_{ik} := \frac{x_{ik}\Pi_{ik}}{\sum_{\ell=1}^{n} w^\epsilon_{i\ell} x_{\ell k}} \tag{6.16}$$

$$F_{ijk} := T_{ik} x_{jk}. \tag{6.17}$$

Using T variables, eigenvector equation (6.10) can be rewritten as

$$\Pi_{jk} = x_{jk} \sum_{i=1}^{n} w^\epsilon_{ij} T_{ik} \qquad \forall i, k, \tag{6.18}$$

while, using F variables, equation (6.16) reads

$$\sum_{\ell=1}^{n} w^\epsilon_{i\ell} F_{i\ell k} = x_{ik}\Pi_{ik} \qquad \forall i, k. \tag{6.19}$$

With the new variables, eigenvector computation is given by a system of equations consisting of (6.17), (6.18) and (6.19). Note that all of them involve products between binary variables and continuous ones. However, they can be linearized to rewrite previous non-linear model into the following mixed-integer linear formulation:

$$\begin{aligned}
(p\text{-leaders}) \quad \max \quad & \sum_{k=1}^{p} z_k \\
\text{s.t.} \quad & (6.7) - (6.9), (6.11) - (6.15) \\
& F_{ijk} \le T_{ik} && \forall i, j, k && (6.20)\\
& F_{ijk} \le M x_{jk} && \forall i, j, k && (6.21)\\
& F_{ijk} \ge T_{ik} - M(1 - x_{jk}) && \forall i, j, k && (6.22)\\
& \sum_{\ell=1}^{n} w^\epsilon_{i\ell} F_{i\ell k} = \Pi_{ik} && \forall i, k && (6.23)\\
& T_{ik} \le M x_{ik} && \forall i, k && (6.24)\\
& \sum_{i=1}^{n} w^\epsilon_{ij} T_{ik} \le \Pi_{jk} + M'(1 - x_{jk}) && \forall i, k && (6.25)\\
& \sum_{i=1}^{n} w^\epsilon_{ij} T_{ik} \ge \Pi_{jk} - 1 + x_{jk} && \forall i, k && (6.26)\\
& \Pi_{ik}, z_k, T_{ik}, F_{ijk} \ge 0 && \forall i, j, k \\
& x_{ik}, y_{ik} \in \{0, 1\} && \forall i, k.
\end{aligned}$$

Here, (6.20)-(6.22) serve to linearize equation (6.17). In effect, when $x_{jk} = 1$, (6.20) and (6.22) guarantee that $F_{ijk} = T_{ik}$, and $F_{ijk} = 0$ otherwise due to (6.21). Equations (6.19) stand thanks to (6.23) and (6.24). On the one hand, (6.23) are precisely (6.19) when $x_{ik} = 1$. On the other hand, if $x_{ik} = 0$, the right hand side of (6.23) is zero because of (6.11), and so is its left hand side due to (6.24) together with (6.20). Finally, (6.25) and (6.26) account for eigenvector

equation (6.18) when $x_{jk} = 1$. Otherwise, (6.25) and (6.26) have no effect and (6.11) ensure that (6.18) stand.

Constants M and M' should be large enough so that (p-leaders) is valid. Tight values for them can be calculated in terms of W, as follows. If we consider inequalities (6.24) first, M has to satisfy $T_{ik} \leq M$ for all nodes $i \in C_k$ and for all clusters C_k. But, if $i \in C_k$, we have that:

$$T_{ik} = \frac{\Pi_{ik}}{\sum_{\ell=1}^{n} w_{i\ell}^{\epsilon} x_{\ell k}} \leq \frac{1}{\sum_{\ell=1}^{n} w_{i\ell}^{\epsilon} x_{\ell k}} \leq \frac{1}{\min_{\ell:(\ell,i)\in E}\{w_{i\ell}\}}.$$

Therefore, setting $M := (\min_{\ell:(\ell,i)\in E}\{w_{i\ell}\})^{-1}$ will do the work. Now we check that the chosen value is adequate for (6.21) and (6.22). According to (6.21), if $j \in C_k$, M should satisfy $F_{ijk} \leq M$. We distinguish two cases. If $i \in C_k$, constraints (6.20) and (6.24) imply that $F_{ijk} \leq T_{ik} \leq M$. On the other hand, if $i \notin C_k$, $F_{ijk} = 0$ again due to (6.20) and (6.24) and the inequality $F_{ijk} \leq M$ also stands. Secondly, according to (6.22), if $j \notin C_k$, it has to be $F_{ijk} \geq T_{ik} - M$. But this is always true because $T_{ik} - M$ is negative or zero. Finally, we have to give a value to M', which only appears in constraints (6.25). In this case, if $x_{jk} = 0$, we have to ensure that (6.25) are not too restrictive. If $x_{jk} = 0$, also $\Pi_{jk} = 0$, so M' should be an upper bound for the left-hand side of (6.25),

$$\sum_{i=1}^{n} w_{ij}^{\epsilon} T_{ik} = \sum_{i\in C_k} w_{ij}^{\epsilon} T_{ik} + \sum_{i\notin C_k} w_{ij}^{\epsilon} T_{ik} \leq \sum_{i\in C_k} w_{ij}^{\epsilon} M.$$

Then, we set $M' := \sum_{i\in C_k} w_{ij}^{\epsilon} M$.

Formulation (p-leaders) can be now implemented in a computer and solved with any optimizer for mixed-integer linear programming such as CPLEX, GUROBI or XPRESS. All these solvers feature exact procedures, i.e., the solution obtained is guaranteed to be optimal. Nevertheless, a careful design of formulations is crucial for solvers success. In the following section, we explore some modifications to improve (p-leaders).

6.4 Formulation improvements

Even though we have restricted to the undirected case, (p-leaders) is thought to be applied to directed or undirected graphs— note how (6.15) is written without assuming that W is symmetric. However, the following proposition shows that the size of the formulation can be reduced when $w_{ij} = w_{ji}$ for all $(i, j) \in E$.

Proposition 6.1. *If G is undirected, a feasible solution of (p-leaders) satisfies that $T_{ik} = T_{jk}$ for all $i, j \in C_k$ and $k = 1, \ldots, p$.*

Proof. Let $(\bar{x}, \bar{y}, \bar{z}, \bar{\Pi}, \bar{T}, \bar{F})$ $i, j \in C_k$ be a feasible solution of (p-leaders). Suppose that i and j belong to the same cluster C_k in the solution, that is, $\bar{x}_{ik} = \bar{x}_{jk} = 1$. Then, $\bar{T}_{ik} = \bar{T}_{jk}$ if and only if

$$\frac{\bar{\Pi}_{ik}}{\sum_{\ell=1}^{n} w_{i\ell}^{\epsilon} \bar{x}_{\ell k}} = \frac{\bar{\Pi}_{jk}}{\sum_{\ell=1}^{n} w_{j\ell}^{\epsilon} \bar{x}_{\ell k}},$$

or, equivalently,

$$\bar{\Pi}_{ik} \sum_{\ell=1}^{n} w^{\epsilon}_{j\ell} \bar{x}_{\ell k} = \bar{\Pi}_{jk} \sum_{\ell=1}^{n} w^{\epsilon}_{i\ell} \bar{x}_{\ell k}.$$

Since $(\bar{x}, \bar{\Pi})$ satisfies (6.10), we can replace $\bar{\Pi}_{ik}$ and $\bar{\Pi}_{jk}$ using that relation. Previous equality would be then

$$\Big(\sum_{s=1}^{n} \bar{x}_{sk} \bar{\Pi}_{sk} \frac{w^{\epsilon}_{si}}{\sum_{r=1}^{n} w^{\epsilon}_{sr} \bar{x}_{rk}} \Big) \sum_{\ell=1}^{n} w^{\epsilon}_{j\ell} \bar{x}_{\ell k} = \Big(\sum_{s=1}^{n} \bar{x}_{sk} \bar{\Pi}_{sk} \frac{w^{\epsilon}_{sj}}{\sum_{r=1}^{n} w^{\epsilon}_{sr} \bar{x}_{rk}} \Big) \sum_{\ell=1}^{n} w^{\epsilon}_{i\ell} \bar{x}_{\ell k}.$$

Now, including all the terms in the same summation and applying the distributive property, we have that

$$\sum_{s=1}^{n} \Big(\frac{\bar{x}_{sk} \bar{\Pi}_{sk}}{\sum_{r=1}^{n} w^{\epsilon}_{sr} \bar{x}_{rk}} \sum_{\ell=1}^{n} w^{\epsilon}_{si} w^{\epsilon}_{j\ell} \bar{x}_{\ell k} \Big) = \sum_{s=1}^{n} \Big(\frac{\bar{x}_{sk} \bar{\Pi}_{sk}}{\sum_{r=1}^{n} w^{\epsilon}_{sr} \bar{x}_{rk}} \sum_{\ell=1}^{n} w^{\epsilon}_{sj} w^{\epsilon}_{i\ell} \bar{x}_{\ell k} \Big);$$

$$\sum_{s=1}^{n} \sum_{\ell=1}^{n} \Big(\frac{\bar{x}_{sk} \bar{\Pi}_{sk}}{\sum_{r=1}^{n} w^{\epsilon}_{sr} \bar{x}_{rk}} w^{\epsilon}_{si} w^{\epsilon}_{j\ell} \bar{x}_{\ell k} \Big) = \sum_{s=1}^{n} \sum_{\ell=1}^{n} \Big(\frac{\bar{x}_{sk} \bar{\Pi}_{sk}}{\sum_{r=1}^{n} w^{\epsilon}_{sr} \bar{x}_{rk}} w^{\epsilon}_{sj} w^{\epsilon}_{i\ell} \bar{x}_{\ell k} \Big).$$

The last equality stands because W is symmetric and completes the proof. □

Based on clustering, our approach involves symmetry issues regarding node-group assignments, an so does (p-leaders). Given a feasible solution to (p-leaders), several (symmetric) solutions can be produced by permuting clusters members, all them having the same objective value. This represents a serious hindrance for the solving procedure, which is based on branching on the binary variables: the branching tree grows exponentially towards the different equivalent feasible solutions. A natural approach to avoid this is to eliminate such symmetric solutions obtained by permutations, i.e., to make them unfeasible for the model. In our case, imposing some constraints that rule node-cluster assignment will cause the desired effect. For each node i and cluster C_k, we consider the following mathematical programming constraints:

$$x_{ik} = 0 \qquad \forall i < k, \forall j \in V \tag{6.27}$$

$$\sum_{s=1}^{k} x_{is} + \sum_{j=k}^{i-1} x_{jk} \geq 1 \qquad \forall i > k. \tag{6.28}$$

Constraints (6.27) state that node 1 is assigned to C_1, node 2 is assigned either to C_1 or C_2 and, in general, i is not assigned to C_k if $k > i$. These constraints also imply $y_{ik} = \Pi_{ik} = T_{ik} = F_{ijk} = 0 \ \forall i < k, \forall j \in V$. For an interpretation of (6.28), assume without loss of generality that nodes are assigned to their clusters in order by index, i.e., node 1 is assigned first, node 2 is assigned in second place, and so on. If i is the next node to assign and C_k is empty, i.e. $\sum_{j=k}^{i-1} x_{jk} = 0$, then i will not be assigned to a cluster with index greater than k, i.e. $\sum_{s=1}^{k} x_{is} = 1$. The following example demonstrates the effect of (6.28) on a feasible solution.

Example 6.1. Figure 6.3 illustrates (6.28) when $k = 5$, $i = 16$ and $p = 7$. In this example, nodes from 1 to 15 have been placed inside clusters $C_1, \ldots, C_4$. Constraints (6.28) forbid that the next node, 16, is placed into clusters C_6 or C_7, because C_5 is empty. Constraints (6.28) together with (6.27) indeed constitute the following rule for node-cluster assignment: "every node is assigned either to some of the already used clusters or to the first empty one". △

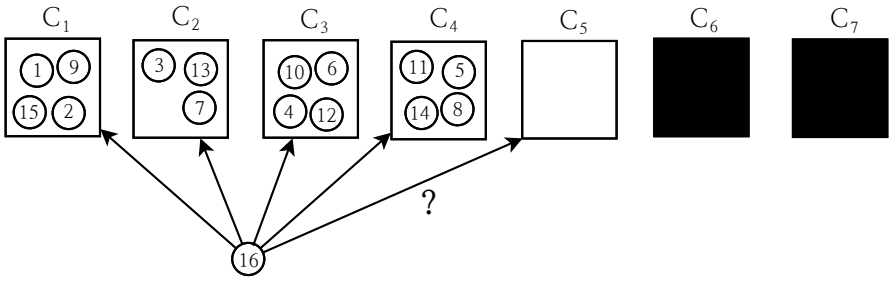


Figure 6.3: Realization of (6.28) when $k = 5$ and $i = 16$

Symmetry breaking constraints (6.28) were first introduced in Kaibel & Pfetsch (2008) and are known as *column inequalities*. They remove symmetric solutions of the partition problem, which is one of the seminal problems of Integer Programming. It consists in finding a classification of all the elements of a set into disjoint subsets, which matches our network clustering. Indeed, our binary variables x_{ij} serves to model a general partitioning problem; for each element i and subset j, $x_{ij} = 1$ if and only if i is assigned to subset j. Let

$$\mathcal{M}^{=}_{n,p} := \Big\{ X = (x_{ij})_{n\times p} \,:\, x_{ij} \in \{0,1\},\, \sum_{j=1}^{p} x_{ij} = 1 \,\forall i \Big\}$$

be the set of 0/1 matrices of size $n \times p$ whose rows sum one, and let $\prec$ be such that $X \prec X'$ with $X = (x_{ij}), X' = (x'_{ij}) \in \mathcal{M}^{=}_{n,p}$ if and only if $x_{k\ell} < x'_{k\ell}$, being (k, ℓ) the first position where X and X' differ, with respect to the ordering

$$(1,1) < (1,2) < \ldots < (1,p) < (2,1) < \ldots < (2,p) < \ldots < (n,1) < \ldots < (n,p).$$

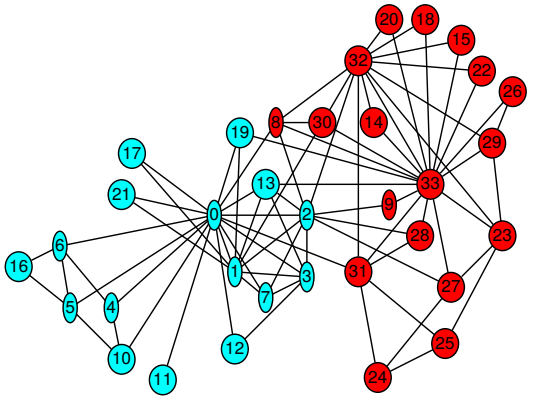
Let $\mathcal{M}^{max}_{n,p}$ be the set of 0/1 matrices of size $n \times p$ that are $\prec$-maximal within their orbits under the group that permutes its columns. Note that columns permutations are clusters permutations in our model. The *partitioning orbitope* is defined in Kaibel & Pfetsch (2008) as follows

$$O^{=}_{n,p} := \text{conv}\,\big(\mathcal{M}^{max}_{n,p} \cap \mathcal{M}^{=}_{n,p}\big).$$

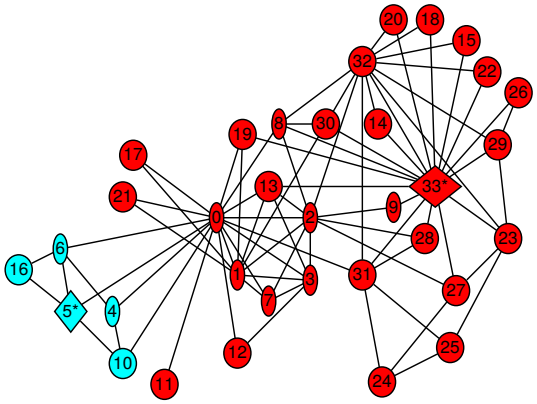
The partitioning orbitope describes a feasible set for the partitioning problem with no symmetric solutions. The authors proved that column inequalities, together with $\sum_{j=1}^{p} x_{ij} = 1$ for all i, characterized the integer points of the partitioning orbitope. That is, these inequalities completely remove the symmetry of the formulation with x-variables. They also provided a complete description of the facets of the partitioning orbitope by generalizing the concept of column inequalities.

6.5 Computational study

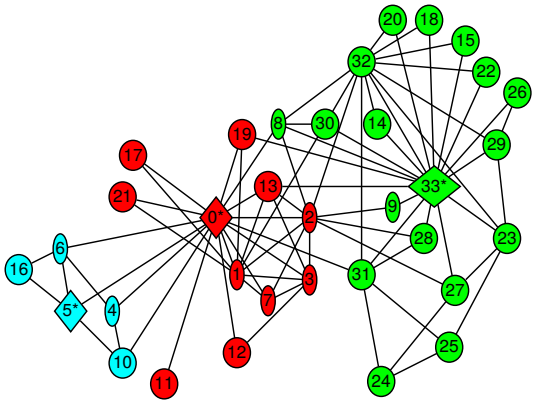
This section discusses the computational performance of the proposed approach. We conducted experiments on two real-life networks and twenty synthetically generated networks. Our experiments were run on an i7-6700k 4.0 GHz × 8 machine with 16 GB memory and the solver used was Cplex v12.6.3.



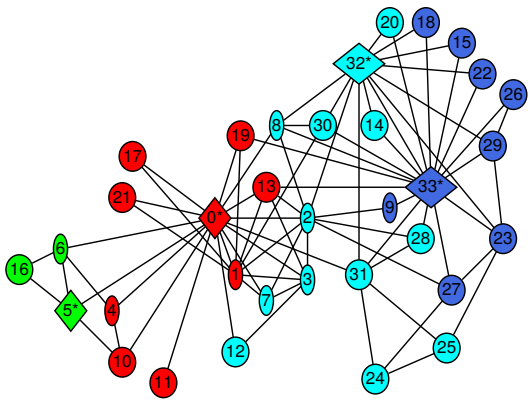
(a) Ford-Fulkerson binary community detection algorithm applied by Zachary (1977)



(b) Our method with $p = 2$, $\mu = 0.38$

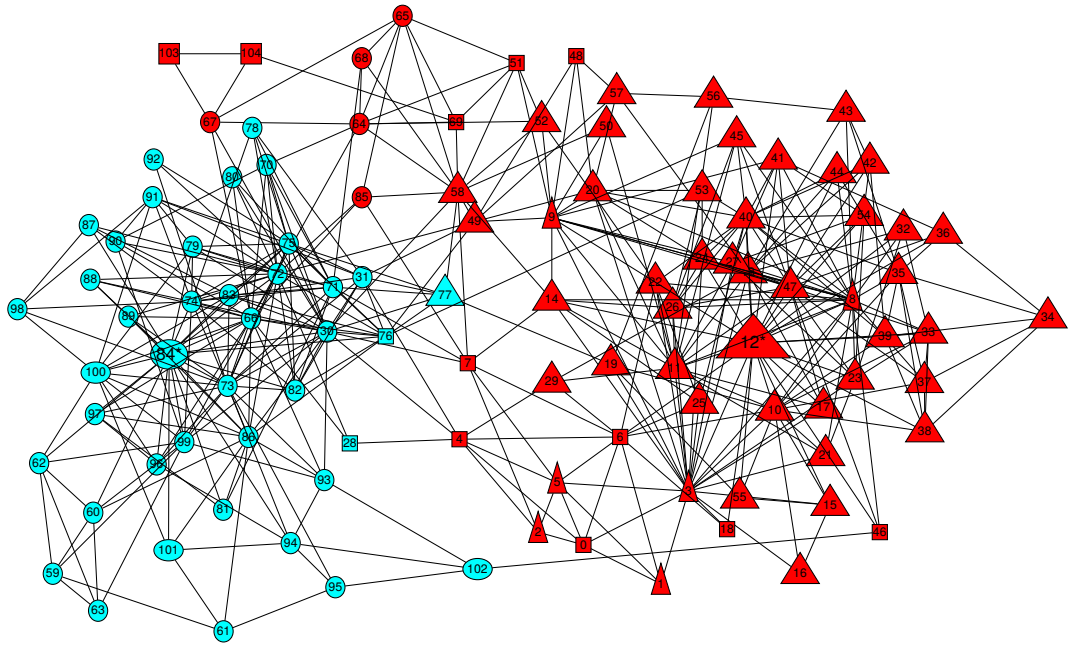


(c) Our method with $p = 3$, $\mu = 0.42$

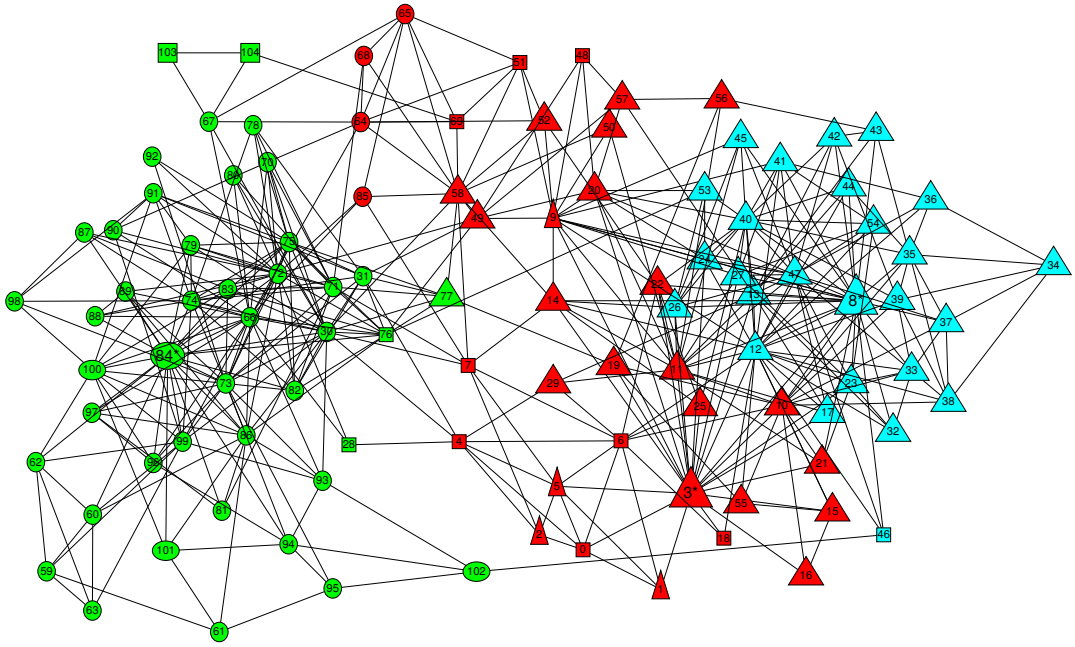


(d) Our method with $p = 4$, $\mu = 0.53$

Figure 6.4: Social structures in the Zachary's Karate Club weighted network Zachary (1977) revealed by different methods. Edge weights are omitted in the illustration



(a) Our method with $p = 2$, $\mu = 0.4$. Nodes 84 and 12 are an optimal group of key members



(b) Our method with $p = 3$, $\mu = 0.48$. Nodes 84, 8 and 3 are an optimal group of key members

Figure 6.5: Key members and communities in the American Political Books (unweighted) network Krebs. Circle, triangle and square nodes represent, respectively, liberal, conservative and neutral books

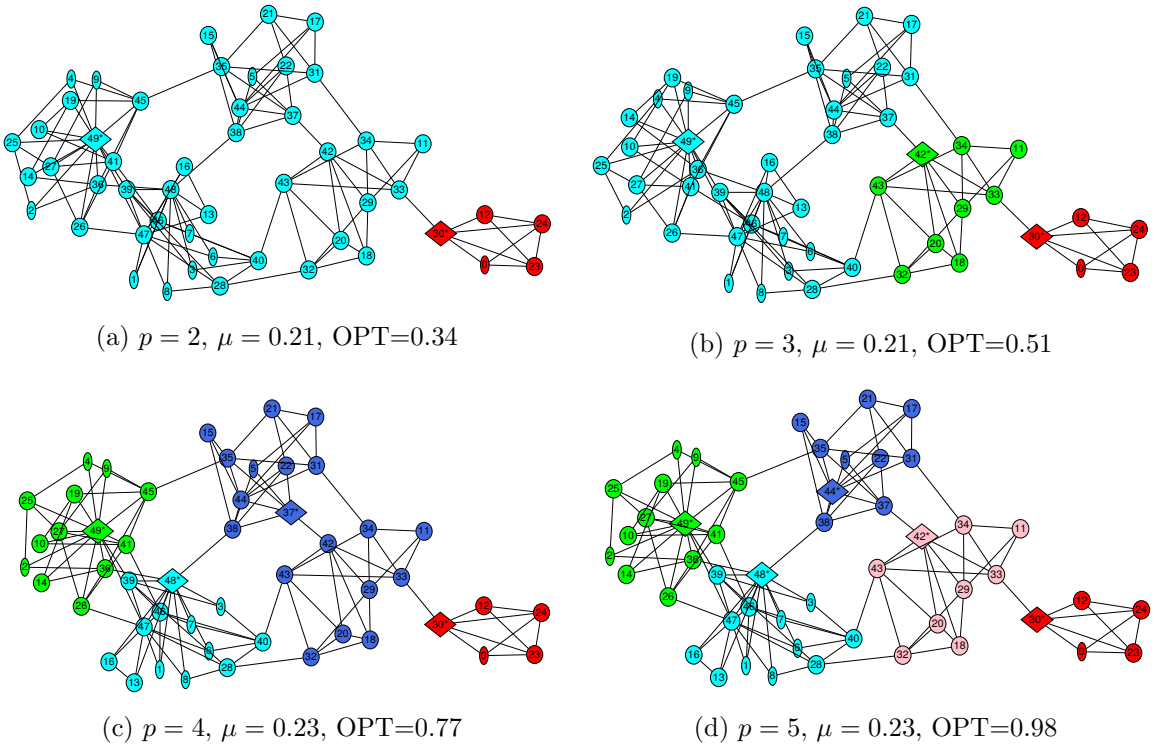
6.5.1 Real-life networks

As first real-word example, we use the Karate Club weighted network of Zachary (1977), which is a standard for testing community detection algorithms. This network collects weights relative to social interactions between the members of a university-based karate club. Figure 6.4 shows several illustrations that emphasize different social structures within the club, according to different methods. Uncovered key nodes are represented with diamonds and labeled with an asterisk. Nodes are labeled with numbers from 0 to 33 that represent anonymous club members, except for the instructor Mr. Hi (node 0) and the administrator John A. (node 33). After a conflict, the club separated into those members that formed a club around John A. and those who remained with Mr. Hi. Figure 6.4a shows the community structure observed by Zachary, which only misclassified node 8 with respect to the actual split. Regarding members relevance, John A. and Mr. Hi occupy the first and second place respectively when nodes are ranked by eigenvector centrality applied to the whole network. Conversely, (p-leaders) reveals previously unnoticed node 5 as a key member for a choice of two candidates with minimum cluster cohesion $1 - \mu = 0.62$, see Figure 6.4b. This exemplifies the formulation preference for small communities, given that they satisfy constraints regarding cohesion. When a set of three key members is selected, node 5 together with "ground truth" leaders John A. and Mr. Hi are nominated. The increment entails a loss of cluster cohesion, which drops from 0.62 to 0.58. On the optimal solution, illustrated in Figure 6.4c, communities frontiers originally observed by Zachary (Figure 6.4a) remain unaltered. However, Mr. Hi's faction is divided into two subunits, one led by member 5 and other led by himself. Finally, Figure 6.4d illustrates the optimal group of four key members and their relative communities, which inevitably present significant interrelations ($\mu = 0.53$). Spotted key members are 0, 5, 32 and 33, as opposed to the top four 33, 0, 32 and 2 according to a ranking by eigenvector centrality.

For a second test with real-word data, we use the American Political Books network compiled by Krebs (see Figure 6.5). Here, nodes are 105 books about US politics sold on Amazon, and edges represent frequent co-purchasing of two books by the same buyers. The books of the network were classified by Newman into three types, namely liberal, conservative or neutral Newman (2006). These categories are represented with different node shapes, namely circles, triangles and squares, respectively. Uncovered key nodes are displayed on a larger scale. When the books are ranked by eigenvector centrality, books 8 and 12 draw as top ones, followed by 3 and 84, which also end in a tie. Conversely, our formulation identifies 84 (on the center of the blue community in Figure 6.5a) and 12 (red community on the right part of Figure 6.5a) as optimal group of two most influential books. This provides a more balanced solution in terms of network coverage than that made of 8 and 12. Alternatively, 84 and 8 (which is next to 12 in Figure 6.5a) are also an optimal solution of size two. For the three most popular books, (p-leaders) finds 84, 3 and 8, respectively in the green, red and blue communities depicted by Figure 6.5b. This time ties between 8 and 12 are broken, and 12 is absorbed by the community of 8, which is selected as one of the key books. Further, nodes 3 and 84 are identified as leading books of distinct communities. Finally, one can observe from Figure 6.5 that network partitions in our solutions fairly respect the categories identified by Newman.

6.5.2 Synthetic networks

The use of the generator of synthetic networks featuring modular structures in Lancichinetti et al. (2008) allows us to test the scalability of the model while validating community discovery. The resulting testbed gathers weighted networks of 50, 100, 150, 200 and 500 nodes, with links densities ranging from 1.5 to 16 (%). The benchmarks feature between 3 and 13 communities



(a) $p = 2$, $\mu = 0.21$, OPT=0.34

(b) $p = 3$, $\mu = 0.21$, OPT=0.51

(c) $p = 4$, $\mu = 0.23$, OPT=0.77

(d) $p = 5$, $\mu = 0.23$, OPT=0.98

Figure 6.6: Optimal solutions to instance 50_1 for different p

whose interrelation accounts for at most a 58% of the links of one node in the less cohesive network and a 8% in the most cohesive one. We compiled in total 20 benchmarks, four of each size. Time limit was set to 5000 seconds.

For evaluating consistency of the method, for each of the 20 instances, we solve the model for different values of p without exceeding the number of communities in the network according to the benchmark generator of Lancichinetti et al. (2008). We observe whether the group of key members is stable when p increases. Parameter μ, the maximum mixing allowed between clusters, have to be tuned in each experiment. For instance, with $p = 2$, we begin with a small μ of 0.1 or 0.2 and increment it until the model is feasible. Feasibility is rapidly discarded, which allows us to tune by inspection. The number of clusters is increased at the cost of losing cohesiveness, and μ has to be consequently updated with the increase of p. As an illustrative example, Figure 6.6 shows the optimal solutions on a network of 50 nodes with $p = 2, 3, 4$ and 5. Different clusters are depicted in different colors and key members are marked with an asterisk and displayed as diamond nodes. Figure 6.6d illustrates the optimal solution for $p = 5$, which perfectly matches the community structure produced by the benchmark generator. Taking then this figure as reference, one can observe that communities frontiers are maintained for $p < 5$. Indeed, when p is decreased, the original five communities are placed one inside another, like matryoshkas. Inspecting now how the optimal group of members varies when p decreases, we find a singularity when $p = 4$. Member 37 is in the optimal group of four leaders depicted on Figure 6.6c, while it is not among the optimal five key members in Figure 6.6d. Despite the apparent contradiction, both solutions seem natural, since 37 is linked to members 42 and 44, the leaders of the original communities on Figure 6.6d that are merged to form a bigger community on Figure 6.6c whose leader is now 37. These fluctuations in leaderships can account for quite of the experiments made on one instance and may not occur for another. On

	p												
	1	2	3	4	5	6	7	8	9	10	11	12	13
50_1	0	21	21	23	**23**								
50_2	0	41	41	44	48	48	**48**						
50_3		0	**17**										
50_4	0	36	**36**										
100_2		00	29	29	30	34	34	**37**					
100_3							0	25	26	**-**			
100_5	0	22	22	23	23	**27**							
100_6	0	27	27	28	30	35	35	**-**					
150_1	0	27	27	28	33	-	-	**-**					
150_4	0	22	23	24	24	25	25	27	30	**30**			
150_5	0	26	26	28	29	**31**							
150_7	0	26	26	27	-	**-**							
200_1	0	21	23	24	26	29	**32**						
200_2			0	17	17	18	18	19	19	20	**26**		
200_3	0	-	-	-	-	-	-	21	-	**23**			
200_4		0	14	14	15	15	15	16	16	17	**18**		
500_1							0	5	6	6	**8**		
500_2					0	7	7	7	**11**				
500_3			0	19	19	**20**							
500_4						0	19	24	30	30	-	-	**44**

Table 6.1: Mixing parameter μ (in%) for different p throughout the benchmarks. Entries are in bold when p equals the number of communities indicated by the benchmark generator

average, we have found that such changes occur in a 4.7% of our experiments. This tendency may suggest the use of some ad hoc strategy to approximate a solution by successive steps.

Table 6.1 summarizes the experiments compiled. The benchmarks, whose nodes range from 50 to 500, are displayed on different rows. For each of them, we have executed our method with p ranging from the number of connected components of the network to the number of communities according to the benchmark generator Lancichinetti et al. (2008). The table shows, for each instance and p, the minimum mixing parameter under which an optimal solution is obtained. In other words, there is not a partition of the given network in p clusters with mixing parameter smaller than that indicated in Table 6.1. Zero entries appear when p equals the number of connected components of the network. In this case, our method simply computes eigenvector centrality in each connected component. Bold entries correspond to experiments in which p matches the number of communities identified by the benchmark generator. In those experiments, the community structure produced by the generator was perfectly uncovered, i.e., it coincides with the optimal cluster partition identified by our method. Finally, dash entries indicate unsolved configurations.

Table 6.1 shows that, when $n = 50$, the method is able to find the optimal key members for all the experiments run, even when the mixing parameter is close to 0.5. The underlying communities according to the benchmark generator are discovered in all the cases. When $n = 100$ or $n = 150$, the method also finds optimal solutions for wide ranging sizes of the key members group. However, sometimes p does not reach the number of underlying communities when they are not well-cohered (a mixing parameter close to 0.5 or exceeding 0.5 has been observed in those cases). For the last group of benchmarks, those with $n = 200$ and $n = 500$, community structures produced by the generator are always uncovered. On the other hand, Figure 6.7 gives quantitative information about the performance of (p-leaders). Namely, it shows the percentage of instances (ordinate axis) that were solved after a certain number of seconds (abscissa axis). Close to 90 % of the instances were solved, more than a half of them

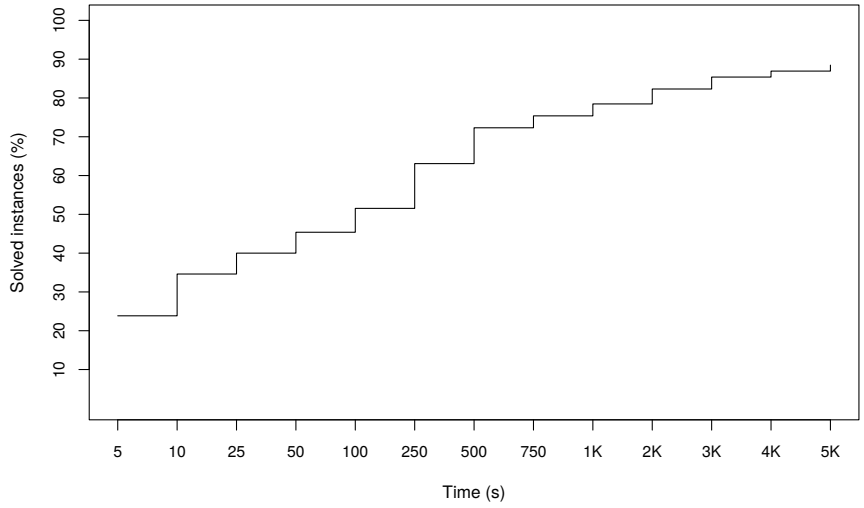


Figure 6.7: Percentage of solved instances after the seconds shown on the abscissa axis

within few seconds.

Despite comparison with underlying structures for testing purposes, one should keep in mind that parameters p and μ are not known *a priori* for a given network. Moreover, these parameters present some degree of uncertainty in real applications, especially for large networks. In this vein, difficulties in finding an optimal solution to, for instance, network 500_4 and $p = 12$, are palliated by the discovery of 13 key members and their communities, which, as a matter of fact, fit better with the actual network layout (see Figure 6.8). Indeed, if the model struggles to find a solution, it is probably the case that the structures that are being searched for (which are determined by p and μ) are far from the intrinsic network organization.

Figure 6.9 shows the average relevance (OPT) of the optimal key members by benchmarks size. Each group of bars displays different values of p. Bar heights are averages of the optimal values for the networks of n nodes solved with the same p Ticks on the bars represent lower bounds on the objective function. Since p normalized eigenvectors are computed— one per cluster— the sum of node relevance throughout the network equals p, and considering an equitable distribution of such overall relevance serves as a lower bound on the optimal value. The larger the optimal value is compared with that bound, the more outstanding the key nodes are compared with the rest. The lower bound is frequently doubled by the optimal value. In some extreme cases, e.g. $n = 500$ and $p = 13$, the optimal value surpasses by far the bound due to the existence of several small connected components (see Figure 6.8).

Unlike the vast majority of existing approaches, which are of heuristic nature, results reported here provide the best group of key members among all possible candidates. Finding such optimum obviously has a computational cost, which can turn expensive depending on the network at hand. Our computational experience on the proposed model demonstrates extremely good performance for the smallest instances and high flexibility for the largest ones, even if sometimes optimal solutions are not achieved within a reasonable time limit through the full range of possible values of p.

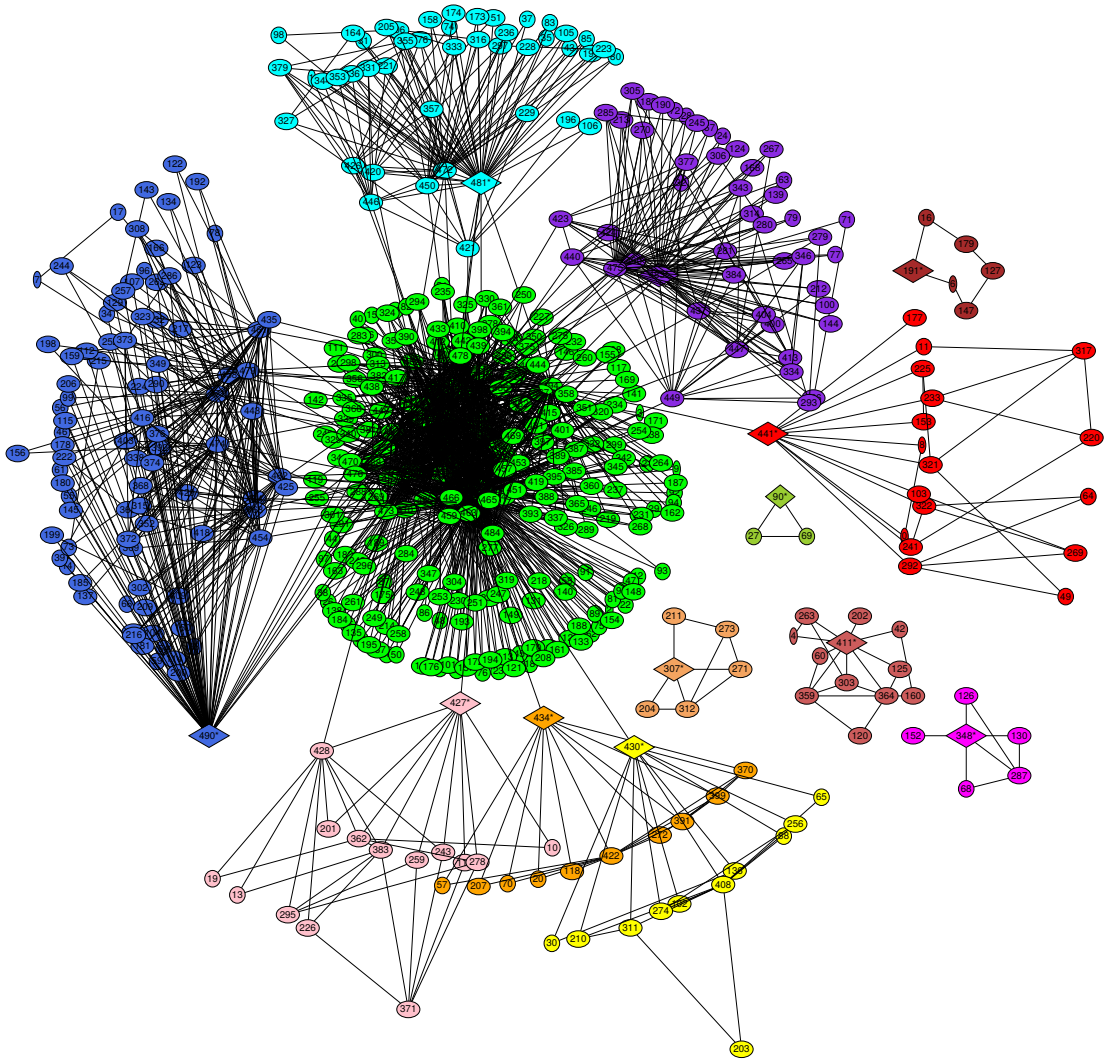


Figure 6.8: Optimal solution for synthetic benchmark 500_4 with $n = 500$, $p = 13$, $\mu = 0.44$

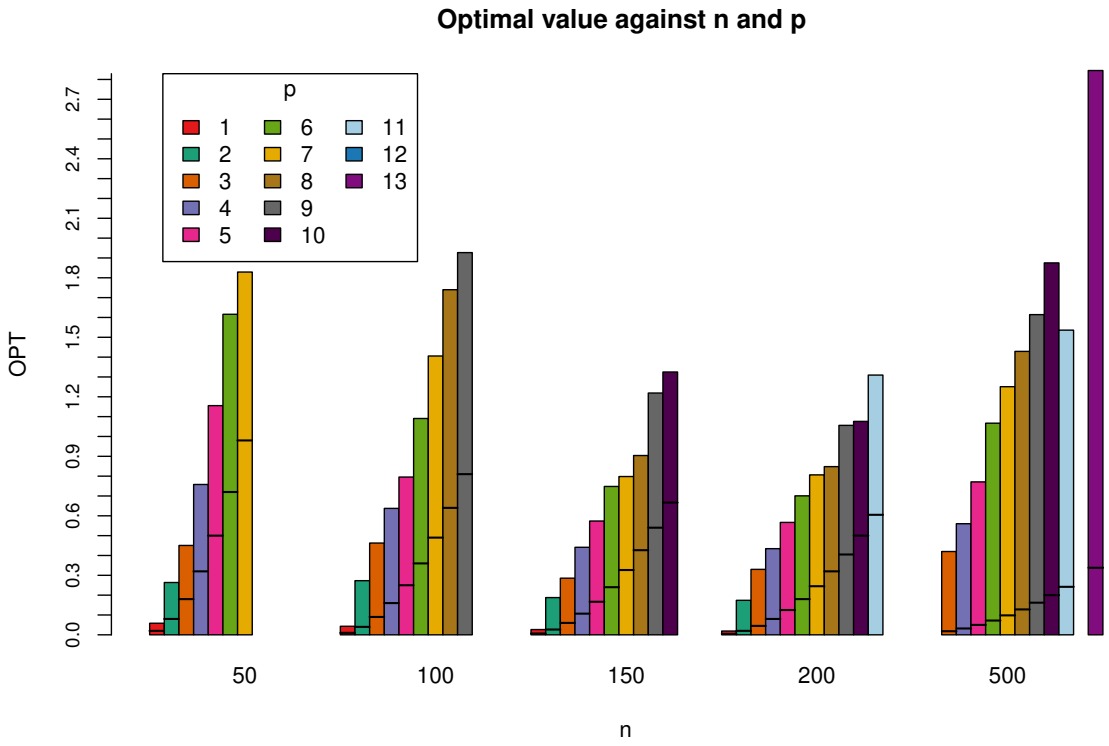


Figure 6.9: Optimal value and lower bounds for different n and p

Bibliography

Aardal, K. (1998). Capacitated facility location: separation algorithms and computational experience. *Mathematical Programming*, *81*, 149–175.

Ahn, Y., Bagrow, J., & Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature*, *466*, 761–764.

Arulselvan, A., Commander, C., Elefteriadou, L., & Pardalos, P. (2009). Detecting critical nodes in sparse graphs. *Computers & Operations Research*, *36*, 2193–2200.

Balas, E. (1979). Disjunctive programming. *Annals of Discrete Mathematics*, *5*, 3–51.

Balas, E., Ceria, S., & Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, *58*, 295–324.

Balas, E., & Padberg, M. W. (1976). Set partitioning: A survey. *SIAM Review*, *18*, 710–760.

Balas, E., & Saltzman, M. (1989). Facets of the three index assignment polytope. *Discrete Applied Mathematics*, *23*, 201–229.

Barahona, F., & Mahjoub, A. (1994). Compositions of graphs and polyhedra ii: stable sets. *SIAM Journal on Discrete Mathematics*, *7*, 359–371.

Beasley, J. (1990). Or-library. Available at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html`.

Been, K., Nöollenburg, M., Poon, S., & Wolff, A. (2010). Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry: Theory and Applications*, *43*, 312–328.

Beineke, L. (1970). Characterizations of derived graphs. *Journal of Combinatorial Theory*, *9*, 129–135.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, *4*, 238–252.

Bomze, I., Budinich, M., Pardalos, P., & Pelillo, M. (1999). The maximum clique problem. In P. Pardalos, D. Du, & R. Graham (Eds.), *Handbook of Combinatorial Optimization* (pp. 1–74). Springer, US.

Bonacich, P. (1972). Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, *2*, 113–120.

Borgatti, S. (2006). Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory*, *12*, 21–34.

Brimberg, J., & ReVelle, C. (2000). The maximum return-on-investment plant location problem. *Journal of the Operational Research Society*, *51*, 729–735.

Bron, C., & Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, *16*, 575–577.

Bu, Z., Cao, J., Li, H., Gao, G., & Tao, H. (2018). Gleam: a graph clustering framework based on potential game optimization for large-scale social networks. *Knowledge and Information Systems*, *55*, 741–770.

Campbell, J., Ernst, A., & Krishnamoorthy, M. (2002). Hub location problems. In Z. Drezner, & H. Hamacher (Eds.), *Facility Location: Applications and Theory* chapter 9. (pp. 373–407). Springer volume 45.

Campêlo, M., Campos, V., & Corrêa, R. (2007). On the asymmetric representatives formulation fot the vertex coloring problem. *Discrete Applied Mathematics*, *156*, 1097–1111.

Campêlo, M., Moura, P., & Santos, M. (2013). On the representatives k-fold coloring polytope. *Electronic Notes in Discrete Mathematics*, *44*, 239–244.

Cánovas, L., Landete, M., & Marín, A. (2000). New facets for the set packing polytope. *Operations Research Letters*, *27*, 153–161.

Cánovas, L., Landete, M., & Marín, A. (2002a). Facet obtaining procedures for set packing problems. *SIAM Journal on Discrete Mathematics*, *16*, 127–155.

Cánovas, L., Landete, M., & Marín, A. (2002b). On the facets of the simple plant location packing polytope. *Discrete Applied Mathematics*, *124*, 27–53.

Cheng, E., & Cunningham, W. (1997). Wheel inequalities for stable set polytopes. *Mathematical Programming*, *77*, 389–421.

Cho, D., Johnson, E., Padberg, W., & Rao, M. (1983a). On the uncapacitated plant location problem i: valid inequalities and facets. *Mathematics of Operations Research*, *8*, 579–589.

Cho, D., Padberg, W., & Rao, M. (1983b). On the uncapacitated plant location problem ii: facets and lifting theorems. *Mathematics of Operations Research*, *8*, 590–612.

Christensen, J., Marks, J., & Shieber, S. (1995). An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics (TOG)*, *14*, 203–232.

Clark, A. (1990). Inference of haplotypes from pcr-amplified samples of diploid populations. *Molecular Biology and Evolution*, *7*, 111–122.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer Programming*. Graduate Texts in Mathematics (GTM). Springer.

Cornaz, D., & Jost, V. (2008). A one-to-one correspondence between colorings and stable sets. *Operations Research Letters*, *36*, 673–676.

Cornuéjols, G., Fisher, M., & Nemhauser, G. (1977). On the uncapacitated location problem. *Annals of Discrete Mathematics*, *1*, 163–177.

Cornuéjols, G., & Thizy, J. (1982). Some facets of the simple plant location polytope. *Mathematical Programming*, *23*, 50–74.

Coventaria (2012). Data of population and coordinates of Spanish settled areas. Avaliable at `http://coventaria.es`. Accessed on 15 May 2015.

Current, J., Min, H., & Schilling, D. (1990). Multiobjective analysis of facility location decisions. *European Journal of Operational Research*, *49*, 295–307.

Dent, B. (1996). *Cartography*. Wm. C. Brown Publishers.

Dias, E., Castonguay, D., Longo, H., & Jradi, W. (2013). Efficient enumeration of chordless cycles. ArXiv preprint arXiv:1309.1051.

Drezner, Z., & Hamacher, H. (Eds.) (2002). *Facility Location: Applications and Theory*. Springer Science & Business Media.

Edmondson, S., Christensen, J., Marks, J., & Shieber, S. (1996). A general cartographic labelling algorithm. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *33*, 13–24.

Escudero, L., Landete, M., & Marín, A. (2008). A branch-and-cut algorithm for the winner determination problem. *Decision Support Systems*, *46*, 649–659.

Escudero, L., Landete, M., & Marín, A. (2009). A branch-and-cut algorithm for the winner determination problem. *Decision Support Systems*, *46*, 649–659.

Evans, T., & Lambiotte, R. (2009). Line graphs, link partitions, and overlapping communities. *Physical Review E*, *80*, 016105.

Everett, M., & Borgatti, S. (1999). The centrality of groups and classes. *Journal of Mathematical Sociology*, *23*, 181–201.

Fischetti, M., Kahr, M., Leitner, M., Monaci, M., & Ruthmair, M. (2018). Least cost influence propagation in (social) networks. *Mathematical Programming*, *170*, 293–325.

Formann, M., & Wagner, F. (1991). A packing problem with applications to lettering of maps. In *7th Annual ACM Symposium on Computational Geometry* (pp. 281–288).

Fortunato, S., & Castellano, C. (2012). Community structure in graphs. In *Computational Complexity* (pp. 490–512). Springer, New York.

Freeman, L. (1977). A set of measures of centrality based on betweenness. *Sociometry*, *40*, 35–41.

Freeman, L. (1978). Centrality in social networks conceptual clarification. *Social Networks*, *1*, 215–239.

Furini, F., Ljubić, I., Martin, S., & Segundo, P. S. (2019). The maximum clique interdiction problem. *European Journal of Operational Research*, *277*, 112–127.

Galli, L., Letchford, A., & Miller, S. (2015). New valid inequalities and facets for the simple plant location problem. Work document.

Galluccio, A., Gentile, C., & Ventura, P. (2008). Gear composition and the stable set polytope. *Operations Research Letters*, *36*, 419–423.

Garey, M., & Johnson, D. (1979). *Computers and intractability : a guide to the theory of NP-completeness*. New York: W.H. Freeman & Co.

Geoffrion, A. (1974). Lagrangean relaxation for integer programming. In M. Balinski (Ed.), *Approaches to integer programming* (pp. 82–114). Springer, Berlin, Heidelberg.

Gomes, S., Ribeiro, G., & Lorena, L. (2013). Dispersion for the point-feature cartographic label placement problem. *Expert Systems with Applications*, *40*, 5878–5883.

Gomory, R. (1960). *An algorithm for the mixed integer problem*. Technical Report RM-2597 The RAND corporation.

Gourdin, E., Labbé, M., & Yaman, H. (2002). Telecommunication and location. In Z. Drezner, & H. Hamacher (Eds.), *Facility Location: Applications and Theory* chapter 9. (pp. 275–305). Springer volume 45.

Grötschel, M., Lovász, L., & Schrijver, A. (1984). Polynomial algorithms for perfect graphs. *Annals of Discrete Mathematics*, *21*, 325–356.

Guignard, M. (1980). Fractional vertices, cuts and facets of the simple plant location problem. *Mathematical Programming*, *12*, 150–162.

Halldórsson, B., Blokh, D., & Sharan, R. (2013). Estimating population size via line graph reconstruction. *Algorithms for Molecular Biology*, (pp. 8–17).

Haunert, J., & Wolff, A. (2016). Beyond maximum independent set: an extended model for point-feature label placement. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, *41*.

Hoehe, M., Kopke, K., Wendel, B., Rohde, K., Flachmeier, C., Kidd, K., Berrettini, W., & Church, G. (2000). Sequence variability and candidate gene analysis in complex disease: Association of μ opioid receptor gene variation with substance dependence. *Human Molecular Genetics*, *9*, 2895–2908.

Hubbell, C. (1965). An input-output approach to clique identification. *Sociometry*, *28*, 377–399.

Imhof, E. (1975). Positioning names on maps. *The American Cartographer*, *2*, 128–144.

Iturriaga, C., & Lubiw, A. (1997). Elastic labels: the two-axis case. In *Graph Drawing* (pp. 181–192). Springer.

Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G., Rinaldi, G., & Wolsey, L. (Eds.) (2010). *50 Years of Integer Programming 1958-2008*. Springer.

Kaibel, V., & Pfetsch, M. (2008). Packing and partitioning orbitopes. *Mathematical Programming*, *114*, 1–36.

Karch, O., Noltemeier, H., & Wahl, T. (2002). Location and robotics. In Z. Drezner, & H. Hamacher (Eds.), *Facility Location: Applications and Theory* chapter 13. (pp. 409–438). Springer volume 45.

Karp, R. (1972). Reducibility among combinatorial problems. In R. Miller, J. Thatcher, & J. Bohlinger (Eds.), *Complexity of Computer Computations* (pp. 85–103). New York: Plenum.

Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika*, *18*, 39–43.

Kaufman, L., Eede, M., & Hansen, P. (1977). A plant and warehouse location problem. *Journal of the Operational Research Society*, *28*, 547–554.

Kitsak, M., Gallos, L., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H., & Makse, H. (2010). Identification of influential spreaders in complex networks. *Nature Physics*, *6*, 888–893.

Klau, G. (2001). *A Combinatorial Approach to Orthogonal Placement Problems*. Ph.D. thesis Universidad de Saarlandes, Germany.

Klau, G., & Mutzel, P. (2003). Optimal labeling of point features in rectangular labeling models. *Mathematical Programming*, *94*, 435–458.

Klose, A., & Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, *162*, 4–29.

Knuth, D. (1993). The stanford graphbase: A platform for combinatorial computing.

Krausz, J. (1943). Démonstration nouvelle d'une théoreme de Whitney sur les réseaux. *Matematikai és Fizikai Lapok*, *50*, 75–85.

Krebs, V. (). Books about us politics. Avaliable at `http://www.orgnet.com/`.

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, *78*.

Landete, M. (2001). *Obtención de facetas de poliedros asociados a problemas de empaquetamiento*. Ph.D. thesis University of Murcia.

Laporte, G., Nickel, S., & da Gama, F. S. (Eds.) (2015). *Location Science*. Springer International Publishing.

Lehot, P. (1974). An optimal algorithm to detect a line graph and output its root graph. *Journal of the ACM*, *21*, 569–575.

Li, H., Bu, Z., Li, A., Liu, Z., & Shi, Y. (2016). Fast and accurate mining the community structure: integrating center locating and membership optimization. *IEEE Transactions on Knowledge and Data Engineering*, *28*, 2349–2362.

Li, Y., Jia, C., & Yu, J. (2015). A parameter-free community detection method based on centrality and dispersion of nodes in complex networks. *Physica A*, *438*, 321–334.

Lovász, L., & Schrijver, A. (1991). Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, *1*, 166–190.

Magos, D., & Mourtos, I. (2009). Clique facets of the axial and planar assignment polytopes. *Discrete Optimization*, *6*, 394–413.

Manka-Krason, A., Mwijage, A., & Kulakowski, K. (2010). Clustering in random line graphs. *Computer Physics Communications*, *181*, 118–121.

Marín, A., Nickel, S., Puerto, J., & Velten, S. (2009). A flexible model and efficient solution strategies for discrete location problems. *Discrete Applied Mathematics*, *157*, 1128–1145.

Marín, A., Nickel, S., & Velten, S. (2010). An extended covering model for flexible discrete and equity location problems. *Mathematical Methods of Operations Research*, *71*, 125–163.

Marín, A., & Pelegrín, M. (2015). Set-packing problems in discrete location. In *VI International Workshop on Location Analysis and Related Problems* (p. 49). Barcelona, Spain, November 25-27.

Marín, A., & Pelegrín, M. (2018a). A new lifting theorem for vertex packing. *Optimization Letters*, *doi:10.1007/s11590-018-1312-4*.

Marín, A., & Pelegrín, M. (2018b). Towards unambiguous map labeling: An integer programming approach. *Expert Systems with Applications*, *98*, 221–241.

Marín, A., & Pelegrín, M. (2019). Adding incompatibilities to the simple plant location problem: Formulation, facets and computational experience. *Computers and Operations Research*, *104*, 174–190.

Marks, J., & Shieber, S. (1991). *The computational complexity of cartographic label placement*. Technical Report Harvard Computer Science Group.

Mauri, G., Ribeiro, G., & Lorena, L. (2010). A new mathematical model and a Lagrangean decomposition for point-feature cartographic label placement problem. *Computers & Operations Research*, *37*, 2164–2172.

Nemhauser, G., & Trotter, L. (1974). Poperties of vertex packing and independence system polyhedra. *Mathematical Programming*, *6*, 48–61.

Nemhauser, G., & Wolsey, L. (1990). A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming*, *46*, 379–390.

Newman, M. (2005). A measure of betweenness centrality based on random walks. *Social Networks*, *27*, 39–54.

Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, *103*, 8577–8582.

Neyer, G. (2003). *Map labeling with application to graph labeling*. John Wiley & Sons, Inc.

Nickel, S. (2001). Discrete ordered weber problems. In B. Fleischmann, R. Lasch, U. Derigs, W. Domschke, & U. Rieder (Eds.), *Operations Research Proceedings 2000* (pp. 71–76). Berlin, Heidelberg: Springer.

Nickel, S., & Puerto, J. (2005). *Facility Location - A Unified Approach*. Springer Verlag.

Nöllemburg, M., & Wolff, A. (2011). Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, *17*, 626–641.

Padberg, M. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming*, *5*, 199–215.

Padberg, M. (1975). A note on zero-one programming. *Operations Research*, *23*, 833–837.

Padberg, M. (1977). On the complexity of set packing polyhedra. *Annals of Discrete Mathematics*, *1*, 421–434.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web.

ReVelle, C., & Eiselt, H. (2005). Location analysis: A synthesis and survey. *European Journal of Operational Research*, *165*, 1–19.

ReVelle, C., Eiselt, H., & Daskin, M. (2008). A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, *184*, 817–848.

Revelle, C., & Laporte, G. (1996). The plant location problem: New models and research prospects. *Operations Research*, *44*, 864–874.

Ribeiro, G., & Lorena, L. (2008). Lagrangean relaxation with clusters for the point-feature cartographic label placement problem. *Computers & Operations Research*, *35*, 2129–2140.

Robinson, H. (1958). *Elements of cartography*. John Wiley & Sons, Inc.

van Rooij, A., & Wilf, H. (1965). The interchange graph of a finite graph. *Acta Mathematica Academiae Scientiarum Hungarica*, *16*, 263–269.

Rossi, F., & Smriglio, S. (2001). A set packing model for the ground holding problem in congested networks. *European Journal of Operational Research*, *131*, 400–416.

Roussopoulos, N. (1973). A $\max\{m, n\}$ algorithm for determining the graph h from its line graph g. *Information Processing Letters*, *2*, 108–112.

van Roy, T. (1986). A cross decomposition algorithm for capacitated facility location. *Operations Research*, *34*, 145–163.

Rylov, M., & Reimer, A. (2014). A comprehensive multi-criteria model for high cartographic quality point-feature label placement. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *49*, 52–68.

Sabidussi, G. (1966). The centrality index of a graph. *Psychometrika*, *31*, 581–603.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

Serfozo, R. (2009). *Basics of applied stochastic processes*. Springer Science & Business Media.

Sherali, H., & Adams, W. (1990). A hierarchy of relaxations between continuous and convex hull representations for zero one programming problems. *SIAM Journal of Discrete Mathematics*, *3*, 411–430.

Silvester, J. (2000). Determinants of block matrices. *The Mathematical Gazette*, *84*, 460–467.

Sousa, J., & Wolsey, L. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical programming*, *54*, 353–367.

Spoorendonk, S. (2008). *Cut and Column Generation*. Ph.D. thesis University of Copenhagen.

Stanoev, A., Smilkov, D., & Kocarev, L. (2011). Identifying communities by influence dynamics in social networks. *Physical Review E*, *84*, 046102.

Stojmenović, I. (Ed.) (2002). *Handbook of Wireless Networks and Mobile Computing*. John Wiley & Sons, Inc.

Stollsteimer, J. (1963). A working model for plant numbers and locations. *Journal of Farm Economics*, *45*, 631–645.

Swamy, C., & Shmoys, D. B. (2008). Fault-tolerant facility location. *ACM Transactions on Algorithms*, *4*, 51.

Terwilliger, J., & Weiss, K. (1998). Linkage disequilibrium mapping of complex disease: Fantasy and reality? Current opinions. *Biotechnology*, *9*, 579–594.

The International HapMap Consortium (2010). Integrating common and rare genetic variation in diverse human populations. *Nature*, *467*, 52–58.

Trotter, L. (1975). A class of facet-producing graphs for vertex packing polyhedra. *Discrete Mathematics*, *12*, 373–388.

Van Kreveld, M., Strijk, T., & Wolff, A. (1999). Point labeling with sliding labels. *Computational Geometry*, *13*, 21–47.

Verweij, A. M. (2000). *Selected Applications of Integer Programming: A Computational Study*. Ph.D. thesis Universidad de Utrecht.

Verweij, B., & Aardal, K. (1999). An optimisation algorithm for maximum independent set with applications in map labelling. In *7th Annu. Europ. Symp. Algorithms (ESA 99)* (pp. 426–437). Prague, Czech Rep.: Springer volume 1643 of LNCS.

Warszawski, A. (1973). Multi-dimensional location problems. *Operational Research Quarterly*, *24*, 165–179.

Whitney, H. (1992). Congruent graphs and the connectivity of graphs. In J. Eells, & D. Toledo (Eds.), *Hassler Whitney Collected Papers* (pp. 61–79). Birkhäuser Boston.

Wierman, J., Naor, D., & Smalletz, J. (2007). Incorporating variability into an approximation formula for bond percolation thresholds of planar periodic lattices. *Physical Review E*, *75*, 011114.

Wolff, A., & Strijk, T. (2009). The map labeling bibliography. Avaliable at `http://i11www.iti.kit.edu/~awolff/map-labeling/bibliography/`. Accessed on 27 Sep 2017.

Wolsey, L. (1998). *Integer Programming*. John Wiley & Sons.

Wolsey, L. A. (1976). Further facet generating procedures for vertex packing polytopes. *Mathematical Programming*, *11*, 158–163.

Xavier, A. S., & Campêlo, M. (2011). A new facet generating procedure for the stable set polytope. *Electronic Notes in Discrete Mathematics*, *37*, 183–188.

Yannakakis, M. (1978). Node and edge deletion np-complete problems. In 10^{th} *Annual ACM Symposium on Theory of Computing* (pp. 253–264). New York.

Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropology Research*, *33*, 452–473.

Zhang, J., Chen, D., Dong, Q., & Zhao, Z. (2016). Identifying a set of influential spreaders in complex networks. *Scientific Reports-UK*, *6*.

Zhang, J., Zhang, K., Xu, X., Chi, K., & Small, M. (2009). Seeding the kernels in graphs: Toward multi-resolution community analysis. *New Journal of Physics*, *11*, 113003.

Zoraster, S. (1990). The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, *38*, 752–759.