SOME COMBINATORIAL OPTIMIZATION PROBLEMS IN GRAPHS WITH APPLICATIONS IN TELECOMMUNICATIONS AND TOMOGRAPHY

THÈSE N^{O} 3082 (2004)

PRÉSENTÉE À LA FACULTÉ SCIENCES DE BASE

Institut de mathématiques

SECTION DE MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

David SCHINDL

ingénieur mathématicien diplômé EPF
et de nationalité autrichiennne

acceptée sur proposition du jury:

Prof. D. de Werra, directeur de thèse
Prof. H. Groeflin, rapporteur
Prof. A. Hertz, rapporteur
Prof. J.-Y. Le Boudec, rapporteur

Lausanne, EPFL
2004

Acknowledgements

I would like to thank the following people for having supported me during my thesis:
First of all, my supervisor Prof. Dominique de Werra, for the nice atmosphere of his working group ROSE where he received me, for the numerous contacts he introduced me to, for the time and energy he spent for making my thesis more clear, and for the useful and realistic advice he gave me many times;
Prof. Alain Hertz, for putting trust in me from the beginning, for his enthusiasm and also the careful and critical reading of the present report; Prof. Martine Labbé whose help and investment were essential in studying the covering, packing and "tartitioning" polytopes of graph coloring in Chapter 5; Prof. Pierre Hansen for pointing out to me this unexplored area and providing me with the necessary tools for studying it; Michael Gerber for leading me to the boundary between easy and hard problems (Chapter 1), and Vadim Lozin whom I met there, and who is an inexhaustible source of inspiration; Professors Marie-Christine Costa and Christophe Picouleau for suggesting to color lines and squares instead of points (Chapter 3);
Teha for the amount of work she did for Chapter 2; all other past and actual members making the group ROSE as it is, especially Nicolas for the good times we spent in Quebec, Annecy and Marseille, or in Lausanne after having worked until 10 PM, Tinaz for being such a kind and pleasant colleague and Ivo for his patience in showing me again and again how to speak with computers.
Finally, I would like to thank my family and friends, for supporting me directly or indirectly but efficiently too, by playing coinche every day, drinking beers and/or playing rock twice a week with me.

Résumé

Le point commun entre les différents chapitres de ce travail est la théorie des graphes. Certains problèmes abordés sont bien connus, d'autres moins et découlent plus directement d'applications.

Le premier chapitre traite du problème du stable maximum, et on y propose de nouvelles classes de graphes où il se résout en temps polynomial. Chacune de ces classes est héréditaire, donc caractérisée par une liste de graphes induits interdits. Les algorithmes proposés sont purement combinatoires.

Le deuxième chapitre est consacré à l'étude d'un problème lié à des questions de sécurité dans des réseaux mobiles. La particularité est qu'il n'y a pas d'autorité centrale permettant de garantir la sécurité, mais elle est gérée par les utilisateurs eux-mêmes. Le réseau se modélise par un graphe orienté, où les sommets sont les utilisateurs, et les arcs représentent des *certificats de clé publique*. On associe à chaque sommet du graphe un sous-graphe, de manière à respecter certaines contraintes limitant la taille des sous-graphes, limitant le nombre de fois qu'un sommet apparaît dans le sous-graphe d'un autre sommet, et imposant qu'au moins un certain nombre de paires de sommets soient mutuellement connectées quand on met leurs sous-graphes en commun. Des heuristiques constructives sont proposées, des bornes aux solutions optimales sont données, des cas particuliers sont étudiés et un algorithme tabou est adapté et testé.

Dans le troisième chapitre, on aborde un problème de reconstruction d'image à partir de ses projections en termes de fréquence d'apparition de chaque couleur dans chaque ligne et chaque colonne. Le cas à deux couleurs se résout à l'aide de techniques de flots, le cas à quatre couleurs est $\mathcal{NP}$-complet, et la complexité du cas à trois couleurs est ouverte. Un cas intermédiaire entre deux et trois couleurs est considéré et prouvé soluble en temps polynomial.

Les deux derniers chapitres sont consacrés à la coloration des sommets d'un graphe. Dans le quatrième, on prouve un résultat livrant toute une collection de cas particuliers $\mathcal{NP}$-difficiles, sous la forme de classes héréditaires de graphes. Dans le chapitre 5 enfin, c'est une approche par la programmation linéaire de la coloration de graphes qui est étudiée. Les liens entre différentes formulations sont d'abord mis en évidence, puis certaines familles de facettes sont caractérisées. Dans la dernière section on étudie un algorithme branch and price utilisant la relaxation linéaire de l'une des formulations exposées pour le calcul de bornes inférieures au nombre chromatique. Un preprocessing est proposé et son efficacité testée.

Abstract

The common point between the different chapters of the present work is graph theory. We investigate some well known graph theory problems, and some which arise from more specific applications.

In the first chapter, we deal with the maximum stable set problem, and provide some new graph classes, where it can be solved in polynomial time. Those classes are hereditary, i.e. characterized by a list of forbidden induced subgraphs. The algorithms proposed are purely combinatorial.

The second chapter is devoted to the study of a problem linked to security purposes in mobile telecommunication networks. The particularity is that there is no central authority guaranteeing security, but it is actually managed by the users themselves. The network is modelled by an oriented graph, whose vertices represent the users, and whose arcs represent *public key certificates*. The problem is to associate to each vertex a subgraph with some requirements on the size of the subgraphs, the number of times a vertex is taken in a subgraph and the connectivity between any two users as they put their subgraphs together. Constructive heuristics are proposed, bounds on the optimal solution and a tabu search are described and tested.

The third chapter is on the problem of reconstructing an image, given its projections in terms of the number of occurrences of each color in each row and each column. The case of two colors is known to be polynomially solvable, it is $\mathcal{NP}$-complete with four or more colors, and the complexity status of the problem with three colors is open. An intermediate case between two and three colors is shown to be solvable in polynomial time.

The last two chapters are about graph (vertex-)coloring. In the fourth, we prove a result which brings a large collection of $\mathcal{NP}$-hard subcases, characterized by forbidden induced subgraphs. In the fifth chapter, we approach the problem with the use of linear programming. Links between different formulations are pointed out, and some families of facets are characterized. In the last section, we study a branch and bound algorithm, whose lower bounds are given by the optimal value of the linear relaxation of one of the exposed formulations. A preprocessing procedure is proposed and tested.

Contents

Introduction

In combinatorial optimization, there are several problems (usually called *easy* problems) which have been solved with efficient algorithms. On the other hand, there is a continuously growing collection of problems (called *hard* problems) for which this task is notoriously difficult, maybe impossible.

There are mainly three kinds of approaches to a hard problem. The first and obvious way is simply to design an algorithm that, implicitly or not, scans all solutions and returns the best one. Unfortunately, the limitations of such an algorithm appear in most cases even for medium-sized instances; its execution time seems to be infinite. The second approach consists in searching for a good solution, without warranty of optimality. The algorithms designed for this sake are called *heuristics*, and have the advantage of running in a reasonable amount of time. The third kind occurs when one wants to learn more about the difficulty of the problem, and explore the world of its subproblems, i.e. find out which ones are easy, and which ones are still difficult to solve. The goal is then to localize as precisely as possible the borderline between easy and difficult cases.

The first of these approaches is used in Chapter 5, for the graph coloring problem, where a branch and price algorithm is studied. Examples of the second approach can be found in Chapter 2 for a security problem in telecommunication networks. The third approach is taken in the three remaining chapters: in Chapter 1 for the maximum stable set problem, in Chapter 3, for an image reconstruction problem in discrete tomography and in Chapter 4 for graph coloring.

The set of hard problems itself has some subclasses of similar problems. Some are *covering* problems and consist in covering a given ground set by a minimum number of its subsets verifying some properties; another kind are *packing* problems, and roughly consist in finding a subset of maximum cardinality or weight of a given set, subject to some incompatibility constraints.

Based on this classification, the structure of the text is the following. A packing problem, namely the maximum stable set problem, is investigated in Chapter 1. It consists in finding in a graph a subset of vertices of maximum size, and containing no two adjacent vertices. Some new cases are proven polynomially solvable. All these cases are expressed as hereditary classes of graphs; the algorithms are combinatorial for these classes.

Covering type problems are studied in Chapters 3, 4, and 5. In Chapter 3, one wants to cover the squares of a 2 dimensional grid with a set of colors, while having given numbers of occurrences of each color in each row and in each column. The case of three colors is studied. In Chapters 4 and 5, the problem of interest is the graph coloring problem, whose aim is to cover the set of vertices of a graph with a minimum number of stable sets. In Chapter 4, two polynomial problem reductions are combined to provide a family of hard subproblems, Chapter 5 is devoted to linear programming approaches and contains some links between different formulations, polyhedral studies and the development of a branch and price algorithm permitting one to solve the problem on medium sized instances.

Chapter 2 deals with a problem which can be classified as being "between" packing and covering: it consists, given an oriented graph, in associating a subgraph to each vertex, subject to three types of constraints. Two of them can be seen as packing constraints, since they impose some limitations on the sizes and intersections of the subgraphs; constraints of the third type can be seen as covering constraints since they express that some paths between two vertices must be covered when they merge their respective subgraphs. Finally, the special case of complete graphs is also investigated, a few constructive heuristics are proposed as well as a tabu search.

Preliminaries

We first define some basic notions which are common to most chapters of this work, and more specific definitions will appear in concerned chapters. Definitions (or equivalent ones) about graphs can also be found in [Ber70] and those about complexity can be found in [GJ99].

Graphs

A simple, undirected *graph* G is a pair (V, E), such that V is a set, and E is composed of distinct unordered pairs of distinct elements of V. If those pairs are ordered, G is called a *directed* graph. If they are not distinct, the graph is not simple anymore, and may be called *multigraph*. In the sequel, if nothing is specified the graph will be simple and undirected. V is called the set of *vertices* of G and E its set of *edges*, or *arcs* in the directed case. If confusion is possible, $V(G)$ will denote the vertex set of the graph G, end $E(G)$ its edge set. Two vertices x and y such that $(x, y) \in E$ are called *adjacent*. The set $N(x) := \{y \in V : (x, y) \in E\}$ is called the *neighborhood* of x. The *degree* of a vertex is the number of its neighbours: $d(x) = |N(x)|$, and the maximum degree among all vertices of G is denoted $\Delta(G)$. A graph may be drawn as in Figure 1, where vertices are points and an edge (x, y) is represented by a line between x and y. In the directed case, an arc (x, y) is represented by an arrow going from x to y. A *subgraph* G *induced* by the vertex set A is the graph $G[A] := (A, \{(x, y) \in E \mid x \in A, y \in A\})$. A *partial subgraph* of G is a graph $G' = (V', E')$, such that $V' \subseteq V$ and $E' \subseteq \{(x, y) \in E \mid x \in V', y \in V'\}$. The *complementary* graph of $G = (V, E)$ is the graph $\overline{G} := (V, \{x \in E, y \in E, x \neq y \mid (x, y) \notin E\})$. Given any graph $G = (V, E)$, its *line graph* is the graph $L(G) = (E, \{(v, w) : v \text{ and } w \text{ have a common vertex in } G\})$. A set of non-adjacent vertices is called a *stable set* (or *independent set*), and its complement, i.e. a set of pairwise adjacent vertices, is a *clique*. The maximum size of a stable set in G is called the *stability number* of G and is denoted $\alpha(G)$, while the maximum size of a clique in G, called the *clique number*, is denoted $\omega(G)$.

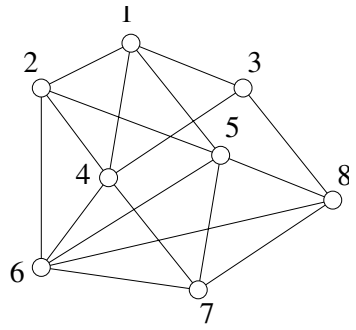

Figure 1: $G = (\{1, 2, 3, 4, 5, 6, 7, 8\}, \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 4), (2, 5), (2, 6), (3, 4), (3, 8), (4, 6), (4, 7), (5, 6), (5, 7), (5, 8), (6, 7), (6, 8), (7, 8)\})$

We define the following simple graphs which often appear in graph theory. In the sequel, all graph definitions are given up to isomorphism, i.e. two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $|V_1| = |V_2|$ are considered as equal, if G_1 can be obtained from G_2 by renaming its vertices. A graph on n vertices is *complete* and denoted K_n, if its set of vertices induce a clique. It is *empty* and denoted O_n if its vertex set is stable. A *bipartite* graph $B = (S, T, E)$ is a graph $(S \cup T, E)$, with S and T being stable sets. A bipartite graph $B = (S, T, E)$ is said *complete* if $E = \{(x, y) \mid x \in S, y \in T\}$; $K_{r,s}$ denotes a complete bipartite graph whose parts have respectively r and s vertices. A *path* P_k denotes the graph $(\{a_1, \ldots, a_k\}, \{(a_1, a_2), (a_2, a_3), \ldots, (a_{k-1}, a_k)\})$. A *cycle* (or *hole*) C_k is the graph $(\{a_1, \ldots, a_k\}, \{(a_1, a_2), (a_2, a_3), \ldots, (a_{k-1}, a_k), (a_k, a_1)\})$, and a *tree* is a graph containing no cycle as a (induced or partial) subgraph.

Among problems related to graphs, the two following ones are of particular interest here. The first one is called the *Maximum Stable set Problem* (MSP), and consists in finding a stable set of maximum size in a given graph. The second problem is the one of assigning a color to each vertex of V, such that any two adjacent vertices have different colors, and the total number of colors used is minimized. Of course, colors can be replaced by numbers or letters, or anything, but the total number of different things used has to be minimized. This problem is called the *graph Minimum (vertex-)Coloring Problem* (MCP) of G, and its optimal value is called the *chromatic number* of G, and denoted $\chi(G)$. Those two problems, without being equivalent, are correlated. If we notice, for instance, that a set of same-colored vertices constitutes a stable set of G, the graph coloring problem of G can be viewed as the one of covering V with a minimum number of stable sets. See Chapters 4 and 5 for relationships between both problems.

Complexity

Dealing with a combinatorial optimization problem usually means in fact dealing with a *set* of questions, parametrized by an *instance*, which is a data of a certain type. In the sequel, we will then speak of a "problem" to refer to the whole set, and of an "instance" if the data is specified. For example, MSP is a class of problems, and an instance of them is a graph. Solving a combinatorial optimization problem C consists then of giving an *algorithm* (sequence of operations) which, for any instance of C, gives its solution in finite time. Of course, if the set of solutions of C is finite or can be bounded (what is almost always the case), such an algorithm exists, since the one consisting of scanning each solution and keeping the best one always runs out in finite time. The difficulty is finding an algorithm which lasts not too long, even for large instances.

More formally, let A be an algorithm for solving the combinatorial optimization problem C, for each instance $\pi \in \Pi$, the set of instances of C. Denote $s(\pi)$, the size of π ($s(\pi)$ may be the number of vertices of the graph if C is MSP), and $t(A, \pi)$, the number of operations of algorithm A for solving C with the instance π; $t(A, \pi)$ is called the *complexity* of A. For the sake of completeness, we provide a formalism allowing to value the efficiency of algorithms, by

analysing their complexity, and then to value the difficulty of a problem.

Let $f : \mathbb{N} \to \mathbb{N}$. The *order* of $f(n)$, defined as

$$O(f(n)) := \{t : \mathbb{N} \to \mathbb{R}_+ \mid \exists\, c \in \mathbb{R}_+^* \text{ and } n_0 \in \mathbb{N} \text{ s.t. } t(n) \leq cf(n)\ \forall\, n \geq n_0\}$$

is the set of functions which can be bounded by $cf(n)$ ($c \in \mathbb{R}_+^*$), if n is large enough. The set of algorithms for solving combinatorial problems can be partitioned in two parts: those whose complexity is in $O(p(s(\pi)))$, where $p(n)$ is a polynomial, and those for which such a polynomial does not exist. For the first algorithms, we will say that they run in *polynomial time*. So the primary task, when studying a combinatorial optimization problem, is to try to determine whether it can be solved with a polynomial time algorithm. We next define four classes of problems, roughly characterizing their difficulty.

A *decision* problem is just a problem whose answer is "yes" or "no". Any minimization or maximization problem has a version of type decision. For example, MSP's decision version, denoted MSPD is:

MSPD
Instance: A graph G and an integer k.
Question: Does there exist a stable set of size k in G ?

A necessary condition for a decision problem to admit a polynomial time algorithm, is the existence of a polynomial time algorithm permitting one to *check* that an answer "yes" is correct, given a certificate (of course an algorithm that solves the problem guarantees the correctness of the answer). Denote $\mathcal{NP}$ this class of problems. MSPD is in $\mathcal{NP}$, since there exists an algorithm for verifying that a set of size k (certificate) is a stable set, and the function giving the number of operations (depending on the instance, here G and k) of this algorithm belongs to $O(|V|^2)$. More briefly, we will say that a solution of MSPD can be verified in $O(|V|^2)$. By $\mathcal{P}$, we denote the class of problems admitting a polynomial time algorithm for solving it. From the remark above, we have that $\mathcal{P} \subseteq \mathcal{NP}$. There is an important collection of very difficult decision problems in $\mathcal{NP}$, which is called $\mathcal{NP}$-*complete*. It is defined as the set of problems A in $\mathcal{NP}$, such that any other problem in $\mathcal{NP}$ can be solved by calling a polynomial number of times an algorithm solving A, and doing a polynomial number of elementary operations. Consequently, if we would find a polynomial algorithm for a problem in $\mathcal{NP}$-complete, one would have a polynomial algorithm for any problem in $\mathcal{NP}$, which would imply that $\mathcal{P} = \mathcal{NP}$. However, most researchers conjecture that such a polynomial algorithm does not exist, i.e. that $\mathcal{P} \neq \mathcal{NP}$. This open problem is one of the most famous in the field of mathematics. In the sequel, we will deal with optimization problems, rather than decision problems. Such a problem whose decision version is in $\mathcal{NP}$-complete is called $\mathcal{NP}$-*hard*[1]. MSP and MCP are examples of $\mathcal{NP}$-hard problems. Further, in this work the set $\mathcal{P}$ will also denote the optimization problems whose decision version is in $\mathcal{P}$.

[1]The terms "$\mathcal{NP}$-complete" and "$\mathcal{NP}$-hard" can usually be used for denoting a set of problems, or as an adjective like here.

Chapter 1

Polynomial cases for the maximum stable set problem

1.1 Introduction

Recall that a *stable* set S in a graph G is a set of pairwise non-adjacent vertices. S is *maximum* if its cardinality $|S|$ is maximum, while it is *maximal* if it is not contained in another stable set of G. The maximum cardinality of a stable set in G is denoted $\alpha(G)$ and is called the *stability number* of G. The problem of finding a maximum stable set in a graph is called the *Maximum Stable set Problem* (MSP), which is $\mathcal{NP}$-hard, as mentioned in the preliminaries. For $A \subseteq V(G)$, we denote $N_A(x) = N(x) \cap A$ the neighbourhood of x in $G[A]$. For two subsets A and B of vertices, we use the notation $N_A(B) = \cup_{b \in B} N_A(b)$ for the set of vertices in B which have a neighbour in A. If a graph G contains a graph H as an induced subgraph, we simply say that G contains H, and that H is a subgraph of G or is contained in G.

We define here some notations used to depict some specific small graphs. By $S_{i,j,k}$ we denote a tree composed of a central vertex from which start at most three pending paths of respective lengths i, j and k. In particular, $S_{1,1,1}$ is a *claw*, $S_{1,1,2}$ is a *fork*, and $S_{2,2,0}$ is a P_5. A *banner* is the graph with vertices a, b, c, d, e and edges (a, b), (b, c), (c, d), (d, e) and (e, b). Some of those graphs are displayed in Figure 1.1. A graph mG is the graph obtained by copying G m times, without additional edge between different copies.

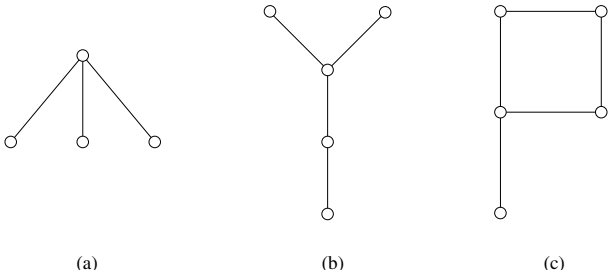

Figure 1.1: (a) Claw (b) Fork (c) Banner

1.1.1 Hereditary classes of graphs

In this chapter, we are interested in the complexity status of some special cases of MSP. A convenient and common way of defining special cases is to restrict MSP to graphs verifying a certain property. A class of graphs $\mathbf{X}$ is *hereditary*, if any subgraph of a graph in $\mathbf{X}$ is also in $\mathbf{X}$. We follow the notations and definitions of [Ale03], where further properties of hereditary classes, related to MSP, can be found.

It is not difficult to see that for a hereditary class of graphs $\mathbf{X}$, there is a unique minimal set of graphs $\mathbf{Y}$, such that $\mathbf{X}$ is exactly the set of graphs containing no graph from $\mathbf{Y}$. We will then write $\mathbf{Y} = Forb(\mathbf{X})$, to say that $\mathbf{Y}$ is the unique minimal set of graphs which have to be forbidden in order to characterize $\mathbf{X}$. Conversely, we will write $\mathbf{X} = Free(\mathbf{Y})$, to depict the class of graphs containing no graph from $\mathbf{Y}$. Alternatively, we will sometimes just call $\mathbf{Y}$-*free* a graph belonging to $Free(\mathbf{Y})$. For instance, the class of stable sets is simply $Free(P_2)$ and the class of bipartite graphs is $Free(\{C_{2k+1} \mid k \geq 1\})$. This characterization is found for lots of hereditary classes of graphs, although it is not always a trivial task. A well-known example is the class of *perfect* graphs. A graph G is perfect if $\omega(G') = \chi(G')\ \forall\ G'$ contained in G. In 1961, C. Berge announced the conjecture that a graph is perfect if (and only if) it contains neither an odd cycle of at least size 5 nor the complement of such an odd cycle, i.e. that the set of perfect graphs is $Free(\{C_{2k+1}, \overline{C_{2k+1}} \mid k \geq 2\})$. This conjecture has been proved in 2002 by M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas [CRST].

Notice finally that if G' is a subgraph of G, then $Free(G') \subseteq Free(G)$, and that if $\mathbf{Y}' \subseteq \mathbf{Y}$ are two sets of graphs, then $Free(\mathbf{Y}) \subseteq Free(\mathbf{Y}')$. Moreover, if a problem is $\mathcal{NP}$-hard, any of its generalizations which still belong to $\mathcal{NP}$ are as well. This implies that if MSP is $\mathcal{NP}$-hard in $Free(\mathbf{Y})$, it is $\mathcal{NP}$-hard in $Free(\mathbf{Y}')$. Conversely, if it admits a polynomial time algorithm in $Free(\mathbf{Y}')$, this algorithm also works for $Free(\mathbf{Y})$.

1.1.2 State of the art

The most useful fact in the study of subcases of MSP, is certainly the one resulting from the following simple observation [Ale83]. Given a graph $G = (V, E)$ and a vertex $x \in V$, partition arbitrarily its neighborhood into three parts $N(x) = N_a \cup N_b \cup N_c$, replace x by a $P_3 = (\{x_1, x_2, x_3\}, \{(x_1, x_2), (x_2, x_3)\})$, and add edges so that $N(x_1) = N_a \cup N_b \cup \{x_2\}$, $N(x_2) = \{x_1, x_3\}$ and $N(x_3) = N_b \cup N_c \cup \{x_2\}$. This graph transformation, sometimes called *vertex splitting*, is depicted in Figure 1.2. It is easy to show that after this transformation, the stability number of a graph G increases by one. Notice that if a K_3 is induced by x, a vertex in N_a and a vertex in N_c, it disappears after the transformation, and if N_b is empty no new K_3 is created. Thus, using this transformation at most as many times as the number of K_3 in G, one can remove all these structures, while increasing the size of G by a polynomial in $|V|$. Thus, if we have a polynomial algorithm A for the class $Free(K_3)$, then we have a polynomial algorithm

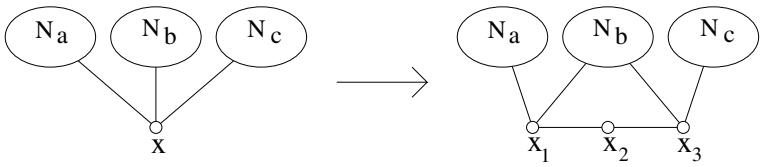

Figure 1.2: Vertex splitting.

$i+j+k$	$S_{i,j,k}$	Graph Class	MSP Status
2	P_3	Union of disjoint cliques	Polynomial
3	$S_{0,0,3}$	Cographs	Polynomial[CPS85]
	$S_{1,1,1}$	$Free(claw) \supseteq$ line-graphs	Polynomial[Min80, Sbi80]
4	$S_{1,1,2}$	$Free(fork)$	Polynomial[Ale04]
	$S_{0,2,2}$	$Free(P_5)$	???

Table 1.1: Complexity status of MSP in $Free(S_{i,j,k})$, for $i+j+k \leq 4$.

for the set of all graphs, consisting first in removing all K_3 from G (call the graph obtained G'), then applying algorithm A for $G' \in Free(K_3)$. The stability number $\alpha(G)$ is then obtained by removing from $\alpha(G')$ the number of times vertex splitting has been applied. This proves that MSP is $\mathcal{NP}$-hard in $Free(K_3)$. By using the same transformation, one can remove from G each cycle of a given size, reduce to 3 the degree of any vertex, and put a path of arbitrary length between any two vertices of degree 3. Since those transformations can be made consecutively without creating new structures which have been removed before, we obtain the following result.

Theorem 1.1 *[Ale83] MSP is $\mathcal{NP}$-hard in $Free(G_1, G_2, \ldots, G_p)$, where p is finite and no G_i has all its connected components of the form $S_{i,j,k}$.* ■

This result substantially restricts the set of graph classes for which the complexity status is unknown. Among those classes, research has naturally focused on classes where only one connected subgraph is forbidden. From what precedes, we know that such a graph has to be of the form $S_{i,j,k}$. Table 1.1 shows, for increasing values of $i+j+k$, the complexity status of MSP in $Free(S_{i,j,k})$. As can be seen, P_5 is the only connected graph G on 5 vertices such that the complexity of MSP is still unknown in the class $Free(G)$. Moreover, P_5 is contained in any $S_{i,j,k}$ with $i+j+k=5$, hence MSP in the class $Free(S_{i,j,k})$, for any such $S_{i,j,k}$, is a more general problem than MSP in $Free(P_5)$. Hence there is no hope of solving MSP in $Free(S_{i,j,k})$ with $i+j+k \geq 5$ before having solved it in $Free(P_5)$. This is what makes this case an interesting and central open problem. In Section 1.2.1 we shall report some partial results in this direction.

Among classes $Free(G)$, where G is disconnected and has all its connected components of the form $S_{i,j,k}$, all cases are in $\mathcal{P}$ if $|V(G)| \leq 4$, the only non trivial case being $Free(2K_2)$, which is a special case of $Free(mK_2)$ for which a polynomial algorithm has been found [Ale91]. The

only open case with $|V(G)| = 5$ is $Free(P_2 \oplus P_3)$, where $G_1 \oplus G_2$ denotes the disjoint union of G_1 and G_2. As for $Free(P_5)$ and connected graphs, all classes with open complexity status which are not superclasses of $Free(P_5)$ and are characterized by a single disconnected forbidden graph of 6 or more vertices contain $Free(P_2 \oplus P_3)$.

Furthermore, subcases of $\mathcal{NP}$-hard cases can also be explored. For instance, Alekseev and Lozin found polynomial algorithms for any class of the form $Free(C_4, C_5, \ldots, C_{k-1}, P_k)$ $(k > 4)$[AL04], while the cases $Free(C_4, C_5, \ldots, C_{k-1})$ are all $\mathcal{NP}$-hard. $Free(banner)$ constitute such a class, and in Section 1.3, a new polynomial algorithm for a quite large subclass of $Free(banner)$ is proposed.

1.1.3 The augmenting graph technique

Numerous polynomial algorithms for special cases of MSP have been designed, and various tehniques have been invented. However, most algorithms rely on two techniques. The first one consists in transforming the original graph in another graph, in such a way that the incidence on its stablility number is known. For instance, removing an isolated vertex from G decreases $\alpha(G)$ by one. If one can sufficiently transform G so that the final graph G' belongs to a class where a polynomial algorithm is known, then $\alpha(G')$ may be computed and $\alpha(G)$ deduced from the transformations made on G. An example [BL01] of a successful application of this technique uses the fact that in $Free(P_5, banner)$, each non-extremal vertex of a $P4$ can be removed from G, without modifying $\alpha(G)$. Consequently, $\alpha(G)$ can be computed in a graph in $Free(P_5, banner)$ by first removing the second vertex of each P_4, which can be done in $O(|V|^4)$, and then applying a polynomial algorithm valid for graphs in $Free(P_4)$. Graphs transformations have been extensively studied, and even general methods to design such transformations have been formulated, like the *struction* method, invented in 1984 and described in [EHdW84], which permits to replace a vertex and its neighborhood by another subgraph, so that $\alpha(G)$ decreases by exactly one. More generally, Alekseev and Lozin propose in [AL04] a framework which permits, for an arbitrary graph transformation, to find an hereditary class of graphs such that the stability number of graphs in this class remains unchanged after the transformation.

Probably the technique which has been most successful is the following. A *matching* is a set of edges, no two of which have a vertex in common. The problem of finding a matching of maximum cardinality is a special case of MSP, when restricted to the hereditary class of line graphs. The first polynomial time algorithm to find a maximum matching has been proposed by Edmonds [Edm65]. The algorithm exploits the idea of Berge that a matching M in a graph is maximum if and only if there are no augmenting (alternating) chains for M [Ber57]. Finding augmenting chains is a special case of a more general approach to solve MSP, known as the *augmenting graph technique*.

A bipartite graph $H = (V_1, V_2, E)$ with parts V_1 and V_2 is called *augmenting* for S if $|V_2| > |V_1|$, $V_1 \subseteq S$, $V_2 \subseteq V(G) \backslash S$, and $N(b) \cap S \subseteq V_1$ for each vertex $b \in V_2$. The *increment* of H is

defined as $\Delta(H) = \mid V_2 \mid - \mid V_1 \mid$. Clearly, if H is augmenting for S, then S is not of maximum cardinality, since $S' = (S \backslash V_1) \cup V_2$ is a larger stable set. The converse is also true: if S is not a maximum stable set, and S' is a stable set with $|S'| > |S|$, then the subgraph of G induced by the set $(S \backslash S') \cup (S' \backslash S)$ is augmenting for S. Thus, MSP is polynomially equivalent to detecting augmenting graphs. However, if for a certain class of graphs, we have

(a) a complete list of augmenting graphs,

(b) a polynomial time algorithm for detecting each augmenting graph in the list,

then MSP can be solved efficiently with this approach. Notice that it is even sufficient to find augmenting graphs which are minimal under inclusion, i.e. connected augmenting graphs which do not strictly contain another augmenting graph as an induced subgraph. Clearly those augmenting graphs are connected and have an increment of one.

For instance, for the class of *claw*-free graphs, question (a) has a simple answer. Indeed, by definition, augmenting graphs are bipartite, and each vertex in a *claw*-free bipartite graph clearly has degree at most two. Hence, every connected *claw*-free bipartite graph is either an even cycle or a chain. Cycles of even length and chains of odd length cannot be augmenting, since they have equal number of vertices in both parts. Thus, every connected *claw*-free augmenting graph is a chain of even length. However, finding augmenting chains is not a trivial task. In 1980, Minty proposed a way to determine whether a *claw*-free graph contains an augmenting chain by reducing the problem to the class of line graphs, i.e. to the maximum matching problem. In 1999, Alekseev extended the result of Minty to the class of $fork$-free graphs. This important result has been translated in english and published in 2004 [Ale04]. He has shown that every connected $fork$-free augmenting graph is either a chain or an almost complete bipartite graph (i.e. a graph in which every vertex has at most one non-neighbor in the opposite part), and has proven that both types of augmenting graphs can be found in polynomial time in $fork$-free graphs. Many other classes have been recently studied for possible application of the augmenting graph technique (see e.g., [AL03, BL01, GHL03, GL03, GHS03, Loz00, Mos97, Mos99, Mos03]). For many of them, polynomial algorithms have been designed.

In Section 1.2, the augmenting graph technique is applied for finding new subclasses of P_5-free graphs where MSP has a polynomial time solution, and in Section 1.3 to find augmenting chains in extensions of *claw*-free graphs, and provide polynomial algorithms for some of those extensions, which are special cases of *banner*-free graphs.

1.2 Subclasses of P_5-free graphs

Our developments are based on a characterization of all connected bipartite P_5-free graphs. This characterization allows us to detect new families of subclasses of P_5-free graphs where MSP has

a polynomial time solution. These new families extend several previously studied classes.

As mentioned in last section, the class of P_5-free graphs is of special interest since it is the only minimal class defined by a single connected forbidden induced subgraph where the complexity status of MSP is unknown. Polynomial algorithms have been developed for several subclasses of P_5-free graphs [BH99, BL01, GR97, Loz00, Mos97].

We first give a characterization of all connected P_5-free augmenting graphs and then use it in Sections 1.2.2 and 1.2.3 to determine subclasses of P_5-free graphs where MSP can be solved in polynomial time. Most of what follows can also be found in [GHS03].

1.2.1 P_5-free augmenting graphs

A bipartite graph H is said to be *chain bipartite* [Yan82] if either $N(x) \subseteq N(y)$ or $N(y) \subset N(x)$ for any choice of two vertices x and y in the same part of H. It is easy to prove (see for example [Mos97]) that every connected bipartite P_5-free graph is chain bipartite. To every integer vector $(d_1, \ldots, d_n)$ such that $d_1 \geq d_2 \geq \ldots \geq d_n$, we associate the chain bipartite graph denoted $B_n(d_1, \ldots, d_n)$ with parts $V_1 = \{a_1, \ldots, a_n\}$ and $V_2 = \{b_1, \ldots, b_{d_1}\}$, and in which there is an edge linking a vertex $a_i \in V_1$ to a vertex $b_j \in V_2$ if and only if $j \leq d_i$. Notice that a_1 is adjacent to all b_j $(j = 1, \ldots d_1)$, and b_1 is adjacent to all a_i $(i = 1, \ldots, n)$. We say that the pair (a_1, b_1) is a *dominating pair* in $B_n(d_1, \ldots, d_n)$. In particular, $B_n(d, \ldots, d)$ is the complete bipartite graph $K_{n,d}$. We can now characterize connected augmenting P_5-free graphs.

Theorem 1.2 *Let H be a connected augmenting graph for a stable set S in a graph G. Then H is P_5-free if and only if it is isomorphic to a $B_n(d_1, \ldots, d_n)$ with $n < d_1$ and $d_n > 0$.*

Proof: It follows from the above definition that each $B_n(d_1, \ldots, d_n)$ is P_5-free. So assume that H is P_5-free, and let $V_1 = \{a_1, \ldots, a_{|V_1|}\}$ and $V_2 = \{b_1, \ldots, b_{|V_2|}\}$ denote its S-part and $\bar{S}$-part, respectively. Notice that H is chain bipartite since it is a connected bipartite P_5-free graph. Hence, we may assume that the vertices in V_1 and V_2 are ordered in such a way that $N(a_j) \subseteq N(a_i)$ and $N(b_j) \subseteq N(b_i)$ for all $i < j$. In other words, G is isomorphic to $B_n(d_1, \ldots, d_n)$ with $n =\mid V_1 \mid$ and $d_i =\mid N(a_i) \mid$ $(i = 1, \ldots, n)$. We have $n =\mid V_1 \mid < \mid V_2 \mid = d_1$ since H is augmenting, and $d_n > 0$ else H is not connected. ■

As an illustration of the above theorem, we now know that there are only three non-isomorphic connected P_5-free augmenting graphs $H = (V_1, V_2, E)$ with $\mid V_1 \mid = 2$ and $\mid V_2 \mid = 3$: $B_2(3,1)$ (also called a *chair*), $B_2(3,2)$ (also called a *banner*) and $B_2(3,3)$ (the complete bipartite graph $K_{2,3}$) (see Figure 1.3).

The following two lemmas provide additional useful information on connected augmenting graphs.

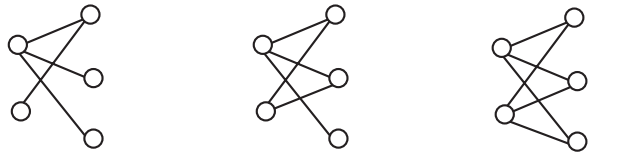

Figure 1.3: The three non-isomorphic connected P_5-free augmenting graphs with 2 vertices in the S-part and 3 in the $\bar{S}$-part.

Lemma 1.3 *Let H be a minimal connected augmenting graph for a stable set S, with S-part parts V_1 and $\bar{S}$-part V_2 ($|V_1| < |V_2|$). Then each vertex in V_1 has at least two neighbours in V_2.*

Proof: Notice first that each vertex in V_1 has at least one neighbour, else H is not connected. Assume now that V_1 contains a vertex x with a unique neighbour y in V_2. Then the graph H' obtained from H by removing vertices x and y is also augmenting with $\Delta(H') = \Delta(H)$, which contradicts the minimality of H. ■

Lemma 1.4 *Let S be a stable set in a P_5-free graph G, and let $B_n(d_1, \ldots, d_n)$ be an augmenting graph for S. If G does not contain any augmenting $K_{1,2}$, then $n > 1$ and $d_2 \geq d_1 - 1$.*

Proof: Let $V_1 = \{a_1, \ldots, a_n\}$ and $V_2 = \{b_1, \ldots, b_{d_1}\}$ be the two parts of $B_n(d_1, \ldots, d_n)$. If $n = 1$, then vertices a_1, b_1 and b_2 induce an augmenting $K_{1,2}$ for S in G, a contradiction. Similarly, if $d_2 < d_1 - 1$, then a_1, b_{d_1} and b_{d_1-1} induce an augmenting $K_{1,2}$ for S in G, a contradiction. ■

1.2.2 Stable sets in $(P_5, K_{3,3} - e)$-free graphs

Let $K_{3,3} - e$ denote the graph obtained by deleting an edge in the complete bipartite graph $K_{3,3}$. The next theorem characterizes connected $(P_5, K_{3,3} - e)$-free augmenting graphs.

Theorem 1.5 *Let S be a maximal stable set in a $(P_5, K_{3,3} - e)$-free graph G, and assume that G does not contain any augmenting $K_{1,2}$ for S. Then each connected minimal augmenting graph H for S is either a $B_n(d, \ldots, d)$ or a $B_n(d, d-1, \ldots, d-1)$ with $1 < n < d$.*

Proof: Consider any connected minimal augmenting graph H for S in G. By Theorem 1.2 and Lemma 1.3, we know that H is isomorphic to a $B_n(d_1, \ldots, d_n)$ with $d_n > 1$. If there exists an index $i > 2$ such that $2 \leq d_i < d_2$, then vertices a_1, a_2, a_i, b_1, b_2 and b_{d_i+1} induce a $K_{3,3} - e$ in G, a contradiction. Hence $d_i = d_2$ for each index $i > 2$ such that $d_i > 1$. It follows from Lemma 1.4 that $n > 1$ and $d_1 - 1 \leq d_2 = \ldots = d_n$. Hence, H is either a $B_n(d, \ldots, d)$ or a $B_n(d, d-1, \ldots, d-1)$ with $1 < n < d$. ■

Notice that $B_n(d,\ldots,d)$ is a $K_{n,d}$ while $B_n(d,d-1,\ldots,d-1)$ is the graph obtained by adding a pending edge to one vertex of degree $d-1$ in a $K_{n,d-1}$. The latter graph is denoted $K^+_{n,d-1}$. The following result is a direct corollary of Theorem 1.5.

Corollary 1.6 *Let S be a maximal but non-maximum stable set in a $(P_5, K_{3,3}-e)$-free graph G, and assume that G does not contain any augmenting $K_{1,2}$ for S. Then there exists an augmenting graph H for S such that*

- $\Delta(H) = \alpha(G) - \mid S \mid$, *and*

- *each connected component of H is either a $K_{n,d}$ or a $K^+_{n,d-1}$ with $1 < n < d$.*

In order to solve MSP in polynomial time in $(P_5, K_{3,3}-e)$-free graphs, it is sufficient to design a polynomial algorithm that finds augmenting $K_{n,d}$ and $K^+_{n,d-1}$ in $(P_5, K_{3,3}-e)$-free graphs. Such an algorithm is not yet available. Brandstädt and Lozin [BL01] have proposed a polynomial algorithm that solves MSP in $(P_5, K_{3,3}-e, TH)$-free graphs, where TH (also called $twin-house$) is a particular graph with 6 vertices. We show in this section that MSP has a polynomial time solution in the class of $(P_5, K_{3,3}-e, K^+_{m,m})$-free graphs, with fixed m. Such a result is already known for $m=1$ and $m=2$. Indeed, $K^+_{1,1}$ is a $K_{1,2}$ and $K^+_{2,2}$ is a *banner*, and the stability number of a $K_{1,2}$-free graph G is its number of connected components, while Lozin [Loz00] has designed a polynomial algorithm that solves MSP in $(P_5, banner)$-free graphs.

Let S be a maximal stable set in a $(P_5, K_{3,3}-e, K^+_{m,m})$-free graph G, with fixed m. Assume there is no augmenting $K^+_{r,r}$ for S with $r < m$. Then there is no augmenting $K^+_{r,s-1}$ for S with $1 < r < s$ and $r < m$ since by removing $s-r-1$ vertices in the $\bar{S}$-part one would get an augmenting $K^+_{r,r}$ with $r < m$. Moreover, there is no augmenting $K^+_{r,s-1}$ for S with $1 < r < s$ and $r \geq m$ since G is $K^+_{m,m}$-free. Hence, it follows from Corollary 1.6 that if S is not maximum, then there exists an augmenting graph H for S such that $\Delta(H) = \alpha(G) - \mid S \mid$, and each connected component of H is an augmenting complete bipartite graph.

Let S be a stable set in G and let x and y be two vertices outside S. Vertices x and y are said *similar* if $N_S(x) = N_S(y)$. Clearly, the similarity is an equivalence relation, and we denote $Q_1,\ldots,Q_k$ the similarity classes. It follows from the definitions that if $K_{r,s}$ $(1 < r < s)$ is an augmenting graph for a stable set S, then its S-part is a $N_S(Q_i)$ for some similarity class Q_i with $\mid N_S(Q_i) \mid > 1$, while its $\bar{S}$-part is a stable set in $G[Q_i]$. A similarity class Q_i is said *interesting* if $\mid N_S(Q_i) \mid > 1$ and $\alpha(G[Q_i]) > \mid N_S(Q_i) \mid$. A vertex $q_i \in Q_i$ is said to be *non-dominating* in Q_i if there exists a vertex $q_j \neq q_i$ in Q_i which is not adjacent to q_i in G. Notice that every interesting similarity class contains at least $\alpha(G[Q_i]) > 1$ non-dominating vertices.

Lemma 1.7 *Let S be a stable set in a $(P_5, K_{3,3}-e)$-free graph G, and let Q_i and Q_j be two interesting similarity classes such that G contains at least one edge linking a non-dominating*

vertex in Q_i to a non-dominating vertex in Q_j. Then either $N_S(Q_i) \subseteq N_S(Q_j)$ or $N_S(Q_j) \subset N_S(Q_i)$.

Proof: Assume G contains an edge between a non-dominating vertex $q_i \in Q_i$ and a non-dominating vertex $q_j \in Q_j$. If neither $N_S(Q_i) \subseteq N_S(Q_j)$ nor $N_S(Q_j) \subset N_S(Q_i)$, then there exists a vertex $x_i \in N_S(Q_i)$ and a vertex $x_j \in N_S(Q_j)$ such that x_i is not linked to q_j and x_j is not linked to q_i is G. Consider any vertex $y_i \in Q_i$ which is not adjacent to q_i, and any vertex $y_j \in Q_j$ which is not adjacent to q_j. Vertex q_i is adjacent to y_j else vertices x_i, q_i, q_j, x_j and y_j induce a P_5 in G, a contradiction. Similarly, q_j is adjacent to y_i. Hence, y_i is adjacent to y_j else vertices x_i, y_i, q_j, x_j and y_j induce a P_5 in G, a contradiction. But now, vertices x_i, y_i, q_i, x_j, y_j and q_j induce a $K_{3,3} - e$ in G, a contradiction. ■

Corollary 1.8 *Let S be a stable set in a $(P_5, K_{3,3} - e)$-free graph G. Let Q_i and Q_j be two interesting similarity classes such that $N_S(Q_i) \cap N_S(Q_j) = \emptyset$, and let S_i and S_j be two maximum stable sets in $G[Q_i]$ and $G[Q_j]$, respectively. Then $S_i \cup S_j$ is a stable set in G.*

Proof: Notice first that $\mid S_i \mid > 1$ and $\mid S_j \mid > 1$ since Q_i and Q_j are interesting similarity classes. Hence, all vertices in S_i are non-dominating in Q_i and all vertices in S_j are non-dominating in Q_j. Since $N_S(Q_i) \cap N_S(Q_j) = \emptyset$, we know by Lemma 1.7, that there is no edge linking a vertex in S_i to a vertex in S_j. ■

Lemma 1.9 *Let S be a stable set in a $(P_5, K_{3,3} - e)$-free graph G, and let Q_i and Q_j be two interesting similarity classes such that $N_S(Q_i) \cap N_S(Q_j) \neq \emptyset$. Then either $N_S(Q_i) \subseteq N_S(Q_j)$ or $N_S(Q_j) \subset N_S(Q_i)$.*

Proof: Consider any non-dominating vertices $q_i \in Q_i$ and $q_j \in Q_j$, and let x be any vertex in $N_S(Q_i) \cap N_S(Q_j)$. If neither $N_S(Q_i) \subseteq N_S(Q_j)$ nor $N_S(Q_j) \subset N_S(Q_i)$, then S contains two vertices y_i and y_j such that y_i is adjacent to q_i but not to q_j, and y_j is adjacent to q_j but not to q_i in G. Moreover, it follows from Lemma 1.7 that q_i is not adjacent to q_j. Hence, vertices y_i, q_i, x, q_j and y_j induce a P_5 in G, a contradiction. ■

In summary, we have proved that if S is a stable set in a $(P_5, K_{3,3} - e, K^+_{m,m})$-free graph G with fixed m, and if there is no augmenting $K^+_{r,r}$ for S with $r < m$, then determining an augmenting graph H for S in G with maximum increment $\Delta(H) = \alpha(G) - \mid S \mid$ reduces to determining a subset $\mathcal{Q}$ of interesting similarity classes such that $N_S(Q_i) \cap N_S(Q_j) = \emptyset$ for each pair (Q_i, Q_j) of elements in $\mathcal{Q}$ and with $\sum_{Q_i \in \mathcal{Q}} \alpha(G[Q_i]) - \mid N_S(Q_i) \mid = \alpha(G) - \mid S \mid$. This is done as in [Loz00]. More precisely, let $\mathcal{I}$ denote the set of interesting similarity classes. We define a graph, denoted $F(S)$, with vertex set $\mathcal{I}$ and in which two vertices Q_i and Q_j are linked by an edge if and only if $N_S(Q_i) \cap N_S(Q_j) \neq \emptyset$. With each vertex Q_i in $F(S)$ we associate a weight equal to $\alpha(G[Q_i]) - \mid N_S(Q_i) \mid$. The weight of a subset of vertices is the sum of weights of its elements. It

is now sufficient to determine a stable set $\mathcal{S}$ with maximum weight in $F(S)$. We then associate a connected augmenting graph H_i for S with each vertex $Q_i \in \mathcal{S}$, the S-part of H_i being equal to $N_S(Q_i)$ while its $\bar{S}$-part is any stable set of maximum size in $G[Q_i]$. The disjoint union of all these augmenting graphs H_i is an augmenting graph H for S with maximum increment. The proposed algorithm for the solution of MSP in the class of $(P_5, K_{3,3}-e, K^+_{m,m})$-free graphs, with fixed m, is summarized below.

Procedure ALPHA(G)

Input: a $(P_5, K_{3,3}-e, K^+_{m,m})$-free graph G with fixed m.

Output: a maximum stable set S in G.

1. Find an arbitrary maximal stable set S in G.
2. If G contains an augmenting $H = K^+_{r,r}$ for S with $r < m$, then replace the S-part of H in S by its $\bar{S}$-part, and repeat Step 2.
3. Partition the vertices of $V(G)\backslash S$ into similarity classes $Q_1, \ldots, Q_k$, and remove the classes Q_i with $\mid N_S(Q_i) \mid < 2$.
4. For each remaining class Q_i, determine a maximum stable set S_i in $G[Q_i]$ by calling **ALPHA($G[Q_i]$)**.
5. Remove all similarity classes Q_i with $\mid S_i \mid \leq \mid N_S(Q_i) \mid$.
6. Construct graph $F(S)$ and find a stable set $\mathcal{S}$ of maximum weight in it.
7. Exchange $N_S(Q_i)$ with S_i for each Q_i in $\mathcal{S}$.
8. Return S and stop.

In order to find a stable set of maximum weight in $F(S)$, it is sufficient to observe (as was done in [AL03]) that $F(S)$ is (P_4, C_4)-free (where a P_4 is a chordless chain on 4 vertices and a C_4 is a chordless cycle on 4 vertices).

Lemma 1.10 *[AL03] Graph $F(S)$ is (P_4, C_4)-free.*

Proof: Assume $F(S)$ is not (P_4, C_4)-free. Consider four vertices Q_1, Q_2, Q_3, Q_4 in $F(S)$ such that Q_2 is adjacent to Q_1 and Q_3 but not to Q_4, and Q_3 is adjacent to Q_2 and Q_4 but not to Q_1 in $F(S)$. Hence, vertices Q_1, Q_2, Q_3 and Q_4 induce a P_4 (if Q_1 is not adjacent to Q_4) or a C_4 in $F(S)$. Since $N_S(Q_2) \cap N_S(Q_3) \neq \emptyset$, we may assume by Lemma 1.9 that $N_S(Q_2) \subseteq N_S(Q_3)$ in G. Hence, $N_S(Q_1) \cap N_S(Q_3) = \emptyset$ implies $N_S(Q_1) \cap N_S(Q_2) = \emptyset$ which contradicts the fact that there exists an edge between Q_1 and Q_2 in $F(S)$. ■

The graphs containing no P_4 and no C_4 as induced subgraphs have been extensively studied in the literature under different names, like trivially perfect graphs [Gol78] and quasi-threshold graphs [CCY96]. The problem of finding a stable set of maximum weight can be solved in that class in linear time using modular decomposition [MS99].

Theorem 1.11 *The stability number of a $(P_5, K_{3,3,} - e, K^+_{m,m})$-free graph with n vertices and fixed $m > 1$ can be determined in $O(n^{m+2})$.*

Proof: Correctness of algorithm **ALPHA** follows from the theorems proved in this section. To estimate the time complexity, we note that Steps 1, 3, 5, 6, 7 and 8 take in the worst case $O(n^3)$ time. An augmenting $K^+_{r,r}$ for S with $r < m$ can be found in $O(n^m)$ time. Since Step 2 is repeated at most n times, the total time complexity of this step is $O(n^{m+1})$. The graph G' obtained by making the disjoint union of all $G[Q_i]$ with $\mid N_S(Q_i) \mid > 1$ has strictly less vertices than G since graphs $G[Q_1], \ldots, G[Q_k]$ are vertex disjoint while G' does not contain any vertex from S. But Step 4 reduces to finding a maximum stable set in G'. Hence, the recursion in Step 4 results in the total time $O(n^{m+2})$. ■

Lozin [Loz00] and Mosca [Mos97] have proposed polynomial algorithms for the solution of MSP in $(P_5, banner)$-free and $(P_5, K_{2,3})$-free graphs, respectively. The above theorem extends both results since $K_{3,3} - e$ and $K^+_{3,3}$ contain an induced *banner* and an induced $K_{2,3}$. Notice also that if p and q are two fixed integers, then MSP has a polynomial solution in the class of $(P_5, K_{3,3,} - e, K^+_{p,q})$-free graphs since these graphs do not contain any induced $K^+_{m,m}$ with $m \geq \max\{p, q\}$.

1.2.3 An infinite family of subclasses of P_5-free graphs

In this section we illustrate the use of the characterization of all connected P_5-free augmenting graphs by identifying an infinite family of subclasses of P_5-free graphs for which MSP has a polynomial time solution. Given a graph H and an integer $t \geq 0$, we denote $A(t, H)$ the graph obtained by adding a clique $K = \{k_1, \ldots, k_t\}$ and a stable set $L = \{l_1, \ldots, l_t\}$ to H, by linking each vertex of K to each vertex of H, and by linking a vertex k_i to a vertex l_j if and only if $i \geq j$. As an illustration, graphs $A(t, H)$ are depicted in Figure 1.4 for various graphs H and for various values of t. We prove in this section that if MSP can be solved in polynomial time in the class of (P_5, H)-free graphs, then MSP can also be solved in polynomial time in the class of $(P_5, A(t, H))$-free graphs, for any fixed t.

Theorem 1.12 *Let H be any graph. If one can solve MSP in a (P_5, H)-free graph G in time $O(\mid V(G) \mid^p)$, then one can solve MSP in a $(P_5, A(1, H))$-free graphs G in time $O(\mid V(G) \mid^{p+1} \cdot \mid E(G) \mid)$.*

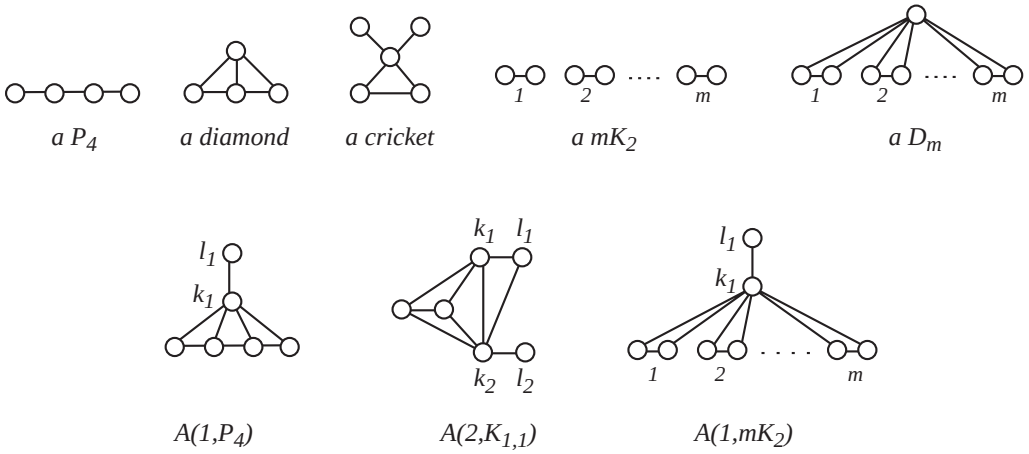

Figure 1.4: Special graphs and illustration of the construction of $A(t, H)$ graphs.

Proof: Let G be a $(P_5, A(1, H))$-free graph. Consider any stable set S in G as well as two adjacent vertices $x \in S$ and $y \notin S$. Let R denote the subset of vertices z in $V(G)\backslash(S \cup \{y\})$ which are adjacent to x but not to y, and such that $N_S(z) \subseteq N_S(y)$. There exists an augmenting $B_n(d_1, \ldots, d_n)$ for S with dominating pair (x, y) if and only if R contains a stable set with $d_1 - 1$ vertices. Hence, to determine whether (x, y) is a dominating pair in an augmenting graph for S, it is sufficient to determine a maximum stable set S' in $G[R]$: $\mid S' \mid \geq \mid N_S(y) \mid$ if and only if $N_S(y) \cup (S' \cup \{y\})$ induces an augmenting $B_n(d_1, \ldots, d_n)$ with $n = \mid N_S(y) \mid$, $d_1 = \mid S' \mid +1$, and with dominating pair (x, y). But $G[R]$ is H-free, else $G[R \cup \{x, y\}]$ contains an $A(1, H)$. Hence $\alpha(G[R])$ can be determined in polynomial time.

Now, one can determine whether G contains an augmenting graph for S by considering all pairs (x, y) of adjacent vertices with $x \in S$ and $y \notin S$, and by checking whether (x, y) is a dominating pair in an augmenting graph for S. Since a maximum stable set in G is necessarily reached after at most $\mid V(G) \mid$ augmentations, one can solve MSP in G by running $O(\mid V(G) \mid \cdot \mid E(G) \mid)$ times the polynomial algorithm which solves MSP in the class of (P_5, H)-free graphs. ■

The following stronger result was proved independently by Mosca [Mos03]. Let WMSP denote the problem of finding a stable set of maximum weight in a graph, and let H be any graph. If one can solve the WMSP in a (P_5, H)-free graph G in time $O(\mid V(G) \mid^p)$, then one can solve the WMSP in a $(P_5, A(1, H))$-free graph G in time $O(\mid V(G) \mid^{p+2})$.

Since $A(t, H) = A(1, A(t-1, H))$, we can state the following corollary.

Corollary 1.13 *Let H be any graph. If MSP has a polynomial time solution in the class of (P_5, H)-free graphs, then it also has a polynomial time solution in the class of $(P_5, A(t, H))$-free graphs G, for any positive integer t.*

As a first illustration of the above result, consider the graph $H = K_{1,1}$ (i.e., H contains only two vertices linked by an edge). MSP is particularly easy to solve in the class of $K_{1,1}$-free graphs since the stability number of such a graph $G = (V, E)$ is equal to $\mid V \mid$. As a consequence, for any fixed integer t, MSP has an $O(\mid E \mid^t \cdot \mid V \mid^{t+1})$ time solution in the class of $(P_5, A(t, K_{1,1}))$-free graphs. But $A(t, K_{1,1})$ contains an induced clique with $t + 2$ vertices. Hence, if the size of the largest clique in a P_5-free graph $G = (V, E)$ is bounded by some fixed number m, then the stability number of G can be determined in $O(\mid E \mid^{m-1} \cdot \mid V \mid^m)$ time. Notice also that $A(2, K_{1,1})$ contains a *diamond* and a *cricket* (see Figure 1.4). It is proved in [AM99] and [Mos97], respectively, that MSP has a polynomial time solution in the classes of $(P_5, \textit{diamond})$-free and $(P_5, \textit{cricket})$-free graphs. Corollary 1.13 therefore generalizes these two results.

As a second illustration, consider $H = P_4$. Obviously, a graph is (P_5, P_4)-free if and only if it is P_4-free. Moreover, it is well known that MSP has a linear time solution in the class of P_4-free graphs [CPS85, MS99]. Hence, Theorem 1.12 and Corollary 1.13 show that MSP can be solved in $O(\mid E \mid^{t+1} \cdot \mid V \mid^t + \mid E \mid^t \cdot \mid V \mid^{t+1})$ time in the class of $(P_5, A(t, P_4))$-free graphs, for any fixed t. Notice that $A(1, P_4)$ contains a *diamond* and a *cricket* (see Figure 1.4). We therefore get a second generalization of the results contained in [AM99] and [Mos97].

As a third illustration, consider the class of $(P_5, K_{1,m})$-free graphs with fixed $m > 1$. Mosca [Mos97] has shown that MSP has an $O(\mid V(G) \mid^{m+1})$ time solution in this class of graphs. This result is in fact a simple corollary of Theorem 1.12. Indeed, define H as the graph made of $m-1$ isolated vertices. MSP can obviously be solved in H-free graphs in $O(\mid V(G) \mid^{m-2})$ time. Since $A(1, H)$ is a $K_{1,m}$, Theorem 1.12 shows that MSP has an $O(\mid E(G) \mid \cdot \mid V(G) \mid^{m-1})$ time solution in $(P_5, K_{1,m})$-free graphs.

Finally, let mK_2 denote the graph made of m disjoint edges. Alekseev [Ale91] has proved that the number of maximal stable sets in mK_2-free graphs is bounded by a polynomial for any fixed m. In combination with the algorithm of Tsukiyama et al. [AIST77] that generates all maximal stable sets, this leads to a polynomial algorithm for MSP in mK_2-free graphs with a fixed m. It follows from Theorem 1.12 that MSP has a polynomial time solution in the class of $(P_5, A(1, mK_2))$-free graphs. But $A(1, mK_2)$ contains a cricket for $m \geq 2$. Hence, Theorem 1.12 provides a third generalization of Mosca's result on $(P_5, \textit{cricket})$-free graphs. Now let D_m denote the graph obtained from mK_2 by adding a vertex linked to all vertices in mK_2 (see Figure 1.4). Notice that D_{m+1} contains $A(1, mK_2)$ which contains D_m. Gerber and Lozin [GL03] have proved recently that MSP has a polynomial solution in the class of (P_5, D_m)-free graphs, for any fixed m. Theorem 1.12 provides another simple proof of this result.

1.2.4 Minimal augmenting P_5-free graphs

Theorem 1.2 gives a characterization of all connected P_5-free augmenting graphs. However, it does not distinguish minimal augmenting graphs from not minimal ones. The following proposition gives a necessary and sufficient condition on values $d_1, \ldots, d_n$ for $B_n(d_1, \ldots, d_n)$ to be

minimal.

Proposition 1.14 *A P_5-free augmenting graph $B_n(d_1, \ldots, d_n)$ is minimal if and only if $d_1 = n+1$ and $d_i \geq n+2-i$, for any $1 < i \leq n$.*

Proof: An augmenting graph $(S, \bar{S}, E)$ is minimal if and only if each proper subset $\bar{S}'$ of $\bar{S}$ verifies property P: $|N_S(\bar{S}')| \geq |\bar{S}'|$. Denote $S = \{a_1, \ldots, a_n\}$ and $\bar{S} = b_1, \ldots, b_{d_1}$, like in the proof of Theorem 1.2. Clearly, if $d_1 > n+1$, then $\bar{S} \backslash \{b_1\}$ does not verify property P. If $d_i \leq n+1-i$, then $N_S(\{b_1, \ldots, b_i\}) \subseteq \{a_1, \ldots, a_{i-1}\}$ and $\{b_1, \ldots, b_i\}$ does not verify property P. Thus both conditions are necessary.

For the sufficiency, we may assume w.l.o.g. that $\bar{S}'$ is of the form $\{b_1, \ldots, b_k\}$, since any subset $\bar{S}'' = \{b_{i_1}, b_{i_2}, \ldots, b_{i_l}\}$ of $\bar{S}$ with $i_1 < i_2 < \ldots < i_l$ verifies $N_S(\bar{S}'') = N_S(\{b_{i_l}\}) = N_S(\{b_1, b_2, \ldots, b_{i_l}\})$ and $\bar{S}'' \subseteq \{b_1, b_2, \ldots, b_{i_l}\}$. So let $\bar{S}' = \{b_1, \ldots, b_k\}$, and denote $N_S(\bar{S}') = \{a_1, \ldots, a_p\}$. We need to prove that $k \leq p$. We have $N_{\bar{S}}(S \backslash \{a_1, \ldots, a_p\}) \subseteq (\bar{S} \backslash \bar{S}')$. Using this and both hypothesis we have $n+2-(p+1) \leq d_{p+1} \leq d_1 - k = n+1-k \Rightarrow k \leq p$. ■

Notice in particular that $d_n \geq 2$. Another simple consequence is that minimal augmenting $(P_5, K_{m,m})$-free graphs have a size bounded by a fonction of m, but independent from $|V|$. Indeed, $B_n(d_1, \ldots, d_n)$ contains a $K_{m,m}$ if and only if $d_m < m$. But from Proposition 1.14, $n+2-m \leq d_m < m$, so $n < 2m-2$. In other words minimal augmenting $(P_5, K_{m,m})$-free graphs have at most $2m-3+2m-2 = 4m-5$ vertices. Hence detecting whether a $(P_5, K_{m,m})$-free graph contains a minimal augmenting graph for a given stable set can roughly be achieved in $O(|V|^{2m})$, which implies polynomial solvability of MSP in $Free(P_5, K_{m,m})$. This is just an alternative method for this class, since in [GL03] Gerber and Lozin propose a completely different algorithm whose complexity is in $O(|V|^{2m})$.

1.3 Subclasses of *banner*-free graphs

We concentrate now on the class *banner*-free graphs, where MSP is itself an $\mathcal{NP}$-hard problem. This class contains the well-studied class of line graphs, since the *claw* is one among the 9 forbidden graphs characterizing them. As mentioned in Section 1.1, MSP in line graphs is exactly the problem of finding a maximum sized matching in an arbitrary graph. This has been solved by Edmonds in 1965, and then extended to the weighted MSP in *claw*-free graphs [Min80, Sbi80, NT01], and to MSP in *chair*-free graphs [Ale04]. We provide here another generalization of this result. This may alternatively be read in [HLS03].

Since we use the augmenting graph technique, our development consists, for a given class of graphs, in first characterizing augmenting graphs and then designing a polynomial algorithm for detecting them. For certain classes, partial information has already been obtained. For instance, for the class of $(S_{1,2,4}, banner)$-free graphs (an extension of claw-free graphs) question

(a) has been completely solved [GHL03], while for (b), only partial solution is available: the only open problem is how to find augmenting chains in polynomial time in that class of graphs. In this section we settle this problem even for more general graphs by reducing it to the class of claw-free graphs.

For S a stable set of a graph G, we call the vertices of S *white* and the remaining vertices of the graph *black*. In order to determine whether S admits an augmenting chain, we consider two black non-adjacent vertices, denoted x_0 and x_k, each of which has exactly one white neighbor. If G contains no such vertices, then obviously there is no augmenting chain for S. Having found such a pair of vertices, we determine whether there exists an augmenting chain with x_0 and x_k being the endpoints. Without loss of generality we assume that

(1) each white vertex has at least two black neighbors,

(2) each black vertex different from x_0 and x_k has exactly two white neighbors,

(3) no black vertex is adjacent to x_0 or x_k.

The vertices not satisfying these assumptions can be simply removed from the graph, since they cannot occur in any augmenting chain connecting x_0 to x_k.

1.3.1 Augmenting chains in $(S_{1,2,i}, banner)$-free graphs

Let $G = (V, E)$ be a $(S_{1,2,i}, banner)$-free graph, and S a maximal stable set in G. We look for an augmenting chain of the form $P = (x_0, x_1, x_2, \ldots, x_{k-1}, x_k)$ (k is even) where the even-indexed vertices of P are black, and the odd-indexed vertices are white. To simplify the proof we start with a preprocessing consisting in detecting augmenting chains with at most $i + 3$ vertices. In order to determine whether S admits an augmenting chain with at least $i + 4$ vertices (i.e., $k \geq i + 3$), we first find two black non-adjacent vertices x_0 to x_k, as suggested above, and then two disjoint chordless alternating chains $L = (x_0, x_1, x_2)$ and $R = (x_{k-m}, x_{k-m+1}, \ldots, x_{k-1}, x_k)$ such that no vertex of L is adjacent to any vertex of R, and where $m = 2\lfloor \frac{i}{2} \rfloor$ and each vertex x_j is black if and only if j is even. Such a pair (L, R) of alternating chains is said *candidate*. Our purpose is to find an augmenting chain containing L and R as subchains. Evidently, if there are no such chains, then there is no augmenting chain with at least $i+4$ vertices between x_0 and x_k. Having found a candidate pair (L, R) of alternating chains, we may furthermore assume that

(4) no black vertex outside L and R has a neighbor in L or R.

Again, the vertices not satisfying the assumption can be removed from the graph, as the desired chain cannot contain them.

Lemma 1.15 *Let $G = (V, E)$ be a $(S_{1,2,i}, banner)$-free graph, and S a maximal stable set in G. Let (L, R) be a candidate pair of alternating chains with $L = (x_0, x_1, x_2)$ and $R = (x_{k-m}, x_{k-m+1}, \ldots, x_{k-1}, x_k)$, and assume that the vertices of G satisfy (1)-(4). If S admits an augmenting chain $P = (x_0, \ldots, x_k)$, then no vertex of P is the center of an induced claw.*

Proof: By contradiction, assume that G contains a claw $C(a; b, c, d)$ whose center a is a vertex x_j on P. Notice that since each black vertex of P has all its white neighbors defined, each vertex of $C \backslash P$ is black. We shall use the following convention: for a black vertex $v \in \{b, c, d\}$, if only one of the two white neighbors of v is defined explicitly, then the other is denoted $\overline{v}$. Also, for a vertex v belonging to $C \backslash P$, we denote by $r(v)$ the largest index in $\{3, 4, \ldots, k-m-1\}$ such that v is adjacent to $x_{r(v)}$. We now analyze three cases: exactly one (C1), two (C2) or three (C3) vertices in $\{b, c, d\}$ do not belong to P.

Case (C1). We may assume $b = x_{j-1}$ and $c = x_{j+1}$. Then d is adjacent neither to x_{j-2} nor to x_{j+2}, else there is a $Banner(c, a, b, x_{j-2}, d)$ or a $Banner(b, a, c, x_{j+2}, d)$, respectively. But then we have either a $S_{1,2,i}(x_{j+i}, \ldots, x_{j-2}, d)$ if $r(d) = j$, or a $S_{1,2,i}(x_{r(d)+i-2}, \ldots, x_{r(d)}, d, a, b, x_{j-2}, c)$ if $r(d) > j$, a contradiction.

Case (C2). Assume that b belongs to P while c and d are outside P. Then vertex b is either equal to x_{j-1} or to x_{j+1}. If $b = x_{j-1}$, then x_{j+1} is adjacent both to c and d to avoid (C1), and x_{j-2} is adjacent neither to c nor to d, else there is a $Banner(b, x_{j-2}, c, x_{j+1}, d)$, a $Banner(c, a, b, x_{j-2}, d)$ or a $Banner(d, a, b, x_{j-2}, c)$. By symmetry, if $b = x_{j+1}$, then x_{j-1} is adjacent both to c and d, while x_{j+2} is adjacent neither to c nor to d. In both cases we have $\overline{c} \neq \overline{d}$, else there is a $Banner(b, a, c, \overline{c}, d)$. Moreover, $r(c) \neq r(d)$, since otherwise there is either a $Banner(b, a, c, x_{r(c)}, d)$ (if $r(c) > j+1$) or a $S_{1,2,i}(x_{j+i+1}, \ldots, x_{j+1}, c, \overline{c}, d)$. But we may then assume $r(c) > r(d)$, and we therefore have a $S_{1,2,i}(x_{r(c)+i-2}, \ldots, x_{r(c)}, c, a, d, \overline{d}, b)$ (if $\overline{d} \neq x_{r(c)-1}$) or a $S_{1,2,i}(x_{r(c)+i}, \ldots, x_{r(c)}, \overline{d}, d, c)$, a contradiction.

Case (C3). Without loss of generality suppose that the claw $C(a; b, c, d)$ with center $a = x_j$ minimizes j. Notice first that $r(b)$, $r(c)$ and $r(d)$ are three different integers, else we may assume $r(b) = r(c)$ and the claw $C(x_{r(c)}; x_{r(c)+1}, b, c)$ contradicts (C2). Moreover, by minimality of j and by (C2), we know that x_{j-1} has exactly two neighbors in $\{b, c, d\}$, say b and c. To avoid (C1) and (C2) we conclude that x_{j+1} is adjacent to d and has at least one neighbor in $\{b, c\}$, say c. Then x_{j+1} is not adjacent to b to avoid a $Banner(d, x_{j+1}, b, x_{j-1}, c)$. We prove now that each white vertex $w \notin \{x_{j-1}, x_j, x_{j+1}\}$ is adjacent to at most one vertex in $\{b, c, d\}$. If this is not the case, then such a white vertex w is adjacent to b, c and d, otherwise a banner appears. Consequently, a is a white vertex, since otherwise c would have three white neighbors x_{j-1}, x_{j+1} and w; but we know by minimality of j that $w \neq x_{j-2}$, and therefore there exists a claw $C(x_{j-1}; x_{j-2}, b, c)$, which contradicts (C2). Now let v_1 and v_2 be the vertices in $\{b, d\}$ renamed in such a way that $r(v_1) > r(v_2)$, and let $\overline{v_2}$ denote the white neighbor of v_2 which is not in $\{x_{j-1}, x_j, x_{j+1}\}$. If $r(c) > r(v_1)$, then there is a $S_{1,2,i}(x_{r(c)+i-2}, \ldots, x_{r(c)}, c, a, v_2, \overline{v_2}, v_1)$. Otherwise, there is a $S_{1,2,i}(x_{r(v_1)+i-2}, \ldots, x_{r(v_1)}, v_1, a, v_2, \overline{v_2}, c)$ (if $\overline{v_2} \neq$

$x_{r(v_1)-1}$) or a $S_{1,2,i}(x_{r(v_1)+i}, \ldots, x_{r(v_1)}, \overline{v_2}, v_2, v_1)$, a contradiction. ■

Theorem 1.16 *Given a $(S_{1,2,i}, banner)$-free graph G, and a stable set S in G, one can determine whether S admits an augmenting chain in time $O(n^{\frac{i+14}{2}})$.*

Proof: Augmenting chains of small length $k < i + 3$ can be found in a trivial way in time $O(n^{\frac{i+4}{2}})$ by inspecting all subsets of black vertices of cardinality at most $\frac{i+4}{2}$. To detect a larger augmenting chain, we first find a pair of black vertices x_0 and x_k as described above, and then remove from G all the black vertices not satisfying (2) and (3). For the given pair x_0 and x_k, we do the following:

> Find all candidate pairs (L, R) of alternating chains, and for each such pair, do steps (a) through (d):
>
> (a) remove all black vertices that have a neighbor in L or in R,
>
> (b) remove the vertices of L and R except for x_2 and x_{k-m},
>
> (c) remove all the vertices that are the center of a claw in the remaining graph,
>
> (d) in the resulting claw-free graph, determine whether there exists an augmenting chain between x_2 and x_{k-m}.

With an exhaustive search all candidate pairs (L, R) of alternating chains can be found in time $O(n^{\frac{i+2}{2}})$. For each such pair, steps (a) through (d) can be implemented in time $O(n^4)$. So, the total time for finding an augmenting chain between x_0 and x_k is $O(n^{\frac{i+10}{2}})$. In the worst case, we have to check $O(n^2)$ pairs of black vertices as possible endpoints of an augmenting chain. Hence the conclusion. ■

1.3.2 Application to $(S_{1,2,4}, banner)$-free graphs

The complete description of minimal (inclusionwise) augmenting graphs in the class of $(S_{1,2,4}, banner)$-free graphs has been found in [GHL03].

Theorem 1.17 *A minimal augmenting $(S_{1,2,4}, banner)$-free graph is one of the following graphs (see Fig. 1)*

- *a complete bipartite graph,*
- *a chain,*
- *a simple augmenting tree,*
- *a plant,*

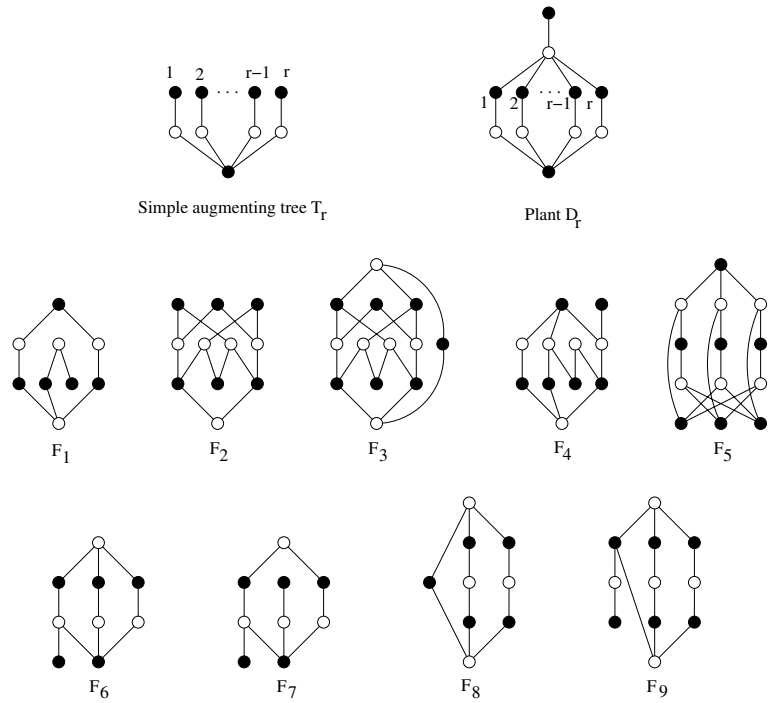

Figure 1.5: Augmenting tree T_r, plant D_r and graphs $F_1, \ldots, F_9$.

- *one of the graphs* $F_1, \ldots, F_9$.

Every graph in the list $F_1, \ldots, F_9$ has at most 7 black vertices. So, these graphs can be detected in time $O(n^7)$. An $O(n^6)$ algorithm is described in [AL03] for finding simple augmenting trees and plants in $(S_{1,2,4}, banner)$-free graphs. By Theorem 1.16, an augmenting chain in that class can be determined in time $O(n^9)$. Thus, in at most $O(n^{10})$ steps we can find a stable set in a $(S_{1,2,4}, banner)$-free graph that admits no augmenting graph except possibly complete bipartite graphs. Let C be a subclass of *banner*-free graphs. It is proved in [AL03] that if for every graph in C one can determine in time $O(n^k)$ a stable set that admits no augmenting graph except possibly complete bipartite graphs, then one can solve the maximum stable set problem in C in time $O(n^{\max\{4,k+1\}})$. Summarizing the above arguments, we conclude that

Theorem 1.18 *Given a* $(S_{1,2,4}, banner)$*-free graph* G *with* n *vertices, one can find a stable set of maximum cardinality in time* $O(n^{11})$.

We hence have proved in this section, that augmenting chains can be found in polynomial time in the class of $(S_{1,2,i}, banner)$-free graphs, for any value of i. Together with the results in [AL03] and [GHL03] this leads to the conclusion that the maximum stable set problem is polynomially solvable in the class of $(S_{1,2,4}, banner)$-free graphs, while it is $\mathcal{NP}$-hard for *banner*-free graphs [Ale83, Mah90]. Our result generalizes polynomial time algorithms for claw-free graphs [Min80, Sbi80], (P_6, C_4)-free graphs [Mos99], and $(P_7, banner)$-free graphs [AL03].

Chapter 2

Optimization techniques for self-organized public-key management in mobile ad hoc networks

2.1 Introduction

In this chapter, we are interested in a type of problem arising from a telecommunication application. A part of it can be found as a technical report in [JSdW03].

Mobile ad hoc networks, unlike conventional networks, do not rely on any fixed infrastructure and do not provide access to a trusted authority; instead all networking functions are performed by the users themselves in a self-organized manner. As the security system is based on public-keys, the main problem is to make the public-key of each user available to others in such a way that its authenticity can be verified. The most famous approach to the public-key management problem is based on public-key certificates. Such a certificate is a data structure in which a public-key is bound to an identity (and possibly to some other attributes) by the digital signature of the issuer of the certificate.

A self-organized public-key management system was proposed by the Laboratory for Computer Communications and Applications (LCA of EPFL) in [BCH03], where public and private keys of users are created by the users themselves (for simplicity, it is assumed that each honest user owns a single vertex, so we will use the same identifier for the user and his representative vertex in the network to be considered). In this system, certificates are mainly stored and distributed by the vertices in an entirely self-organized process, and each certificate is issued with a limited validity period.

Key authentication is performed the following way. When a user u wants to obtain the public-key of another user v, he acquires a sequence of valid public-key certificates such that:

1. The first certificate of the sequence can be directly verified by u, by using a key that u holds and trusts (e.g. his own public-key).
2. Each remaining certificate can be verified using the public-key contained in the previous certificate of the sequence.
3. The last certificate contains the public-key of the target user v.

Such sequences will be called *certificate paths*[1]. To find appropriate certificate paths linking a vertex to other users, each vertex maintains in principle two local certificate repositories: the *nonupdated certificate repository* and the *updated certificate repository*. The nonupdated repository of a vertex contains expired certificates that the vertex does not keep updated, and its updated certificate repository contains a subset of certificates that the vertex keeps updated. The selection of certificates into the vertex's updated repository is performed according to an appropriate algorithm.

Several points have been studied by the LCA concerning this self-organized public-key management system (see [BCH03, BBC+01, BCH01]). Our interest concentrates on the authentication problem, i.e. finding an appropriate algorithm to construct the vertices' updated repositories. For simplicity purposes, we will consider only updated certificates in the graph formulation given below.

2.2 Definitions and basic model

In this chapter, all graphs $G = (V, E)$ are directed, which means that their edges are in fact arcs. Let us introduce some definitions related to such graphs. We denote by $d^+(v)$ the number of arcs starting from v and by $d^-(v)$ the number of arcs ending at v. A *path* from v_1 to v_k is a sequence of arcs of the form $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$. If there is a path from vertex u to vertex v, we will say that v is *reachable* from u in G (denoted in the sequel by $u \rightarrow_G v$). A *root* is a node from which each $x \in V$ is reachable; an *anti-root* is a node reachable from each $x \in V$. A graph is *strongly connected*, if any vertex is reachable from any other vertex. A *chain* is a sequence $e_1, \ldots, e_k$ of arcs such that for $1 < i < k$, e_i shares exactly one vertex with e_{i-1}, and its other vertex with e_{i+1}; a cycle is a chain such that the vertex not shared with e_2 by e_1 is the vertex not shared with e_{k-1} by e_k. The *length* of a path (chain) is the number of its arcs. An *out-tree* is a directed connected graph without cycle, containing a root. An *in-tree* is a directed connected graph without cycle, containing an anti-root. In an in-tree or an out-tree, a *leaf* is

[1] They were called *certificate chains* in [BCH03], but the word "path" is more accurate according to standard definitions in graph theory.

a vertex with degree 1. A *circuit* is a path where the first vertex and the last one coincide. A circuit is elementary if it passes at most once by each vertex of G, and it is called *hamiltonian* if it passes exactly once by each vertex of G. A graph is *hamiltonian* if it contains a hamiltonian circuit. A graph $G' = (V', E')$ such that $E' \subseteq \{(u, v) \in E | u \in V', v \in V'\}$ is a *partial* subgraph of G. Notice that an induced subgraph is a partial subgraph, while the converse is not always true. Unless the contrary is explicitly mentioned, all subgraphs in the sequel will be partial. Finally, a graph is complete if for any pair $\{i, j\}$ of vertices, either $(i, j) \in E$ or $(j, i) \in E$.

In what follows, it will be useful to consider the scheme in terms of an abstract model. In this model, the public-keys and the certificates of the system are represented as a directed graph $G = (V, E)$[2], called the *certificate graph*. Its vertex set V represents the set of public-keys[3] and its arc set E represents the set of certificates. The authentication is performed the following way. When a user u wants to verify the authenticity of the public-key of another user v, they merge their certificate repositories and u tries to find a certificate path to v in the merged repository. If u finds a path to v, then u authenticates the public-key of v.

A certificate path from the public-key of a vertex u to the public-key of another vertex v is represented by a path from vertex u to vertex v in G, which means that the vertex v is reachable from the vertex u in G. In the model proposed in [BCH03] the updated and the nonupdated certificate repositories of user u are represented by the updated and nonupdated certificate graphs G_u and G_u^N, respectively. Therefore, for any u, G_u is a subgraph of G, but G_u^N is not necessarily a subgraph of G, as it may also contain some implicitly revoked certificates[4]. As mentioned we will consider only graphs G_u.

Denote by $G_1 \cup G_2$ the graph $G = (V, E)$ defined by $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$. The authentication is then performed as follows: u tries to find a path from u to v in $G_u \cup G_v$, and uses the certificates on this path to authenticate the public-key of v. If such a path exists, then u performs authentication, otherwise u fails to authenticate v's public-key.

The problem is to associate a good repository to each user. A solution to our problem is thus a collection $F = \{G_v : v \in V\}$ of $|V(G)|$ subgraphs G_v associated to each vertex $v \in V$. The following three criteria are defined in [BCH03] in order to quantify the quality of a solution and to formulate different versions of our problem in terms of graphs.

We define the *basic performance* $p(F, G)$ of a solution F with the certificate graph G as the ratio between the number of vertex pairs $\{u, v\}$ for which there is a path from u to v in $G_u \cup G_v$, and the total number of pairs $\{u, v\}$ for which there is a path from u to v in G. Formally, the basic performance is defined as follows:

$$p(F, G) = \frac{|\{\{u, v\} \in V \times V : u \rightarrow_{G_u \cup G_v} v\}|}{|\{\{u, v\} \in V \times V : u \rightarrow_G v\}|}. \tag{2.1}$$

[2]for simplicity, we assume that each user generates a single (public, private) key pair and, therefore, is represented by a single vertex in the graph

[3]we will denote by u the vertex corresponding to the public-key of u

[4]implicitly revoked certificates are certificates that the corresponding user did not update

If the performance is close to one, the solution provides essentially the same service as if the whole certificate graph were available to each user.

The *size* $s_F(u)$ *of u's certificate repository* of a solution F is defined as the number of arcs of G_u: $s_F(u) = |E(G_u)|$, where $E(G_u)$ denotes the set of arcs of the subgraph G_u. Since it is a measure of the amount of memory needed by u to store its certificate repository, a solution with smaller sizes will be of better practical interest. In the model we consider, all users have the same characteristics, so the maximum repository size

$$s(F) = \max_{u \in V} s_F(u) \tag{2.2}$$

should be as small as possible.

The *usage* $U_F(v)$ of a given vertex (i.e. of the public-key of a given vertex) v is defined as:

$$U_F(u) = |\{v \in V : u \in V(G_v)\}|, \tag{2.3}$$

where $V(G_u)$ denotes the set of vertices of the subgraph G_u. This value indicates how many times the key of u could, in the worst case, be used for authentication (i.e. could be in a certificate path that is used for authentication). On account of robustness, a solution should not contain vertices with high usage. Indeed, the security of the network would then rely on a small set of users, what is undesirable. To avoid this, the maximum usage

$$U(F) = \max_{v \in V} U_F(v)$$

should also be minimized.

Let $G = (V, E)$ be a certificate graph, $U_0 \in \mathbb{N}$ and $0 \leq p_0 \leq 1$. Depending on the importance attached to the above criteria, we formulate the five following variant formulation of problem.

$$(1) \begin{cases} \min\ s(F) \\ \text{s.c.} \quad p(F,G) \geq p_0 \end{cases} \qquad (2) \begin{cases} \min\ s(F) \\ \text{s.c.} \quad p(F,G) \geq p_0 \\ \qquad U(F) \leq U_0 \end{cases}$$

$$(A) \begin{cases} \max\ p(F,G) \\ \text{s.c.} \quad s(F) \leq s_0 \end{cases}$$

$$(B) \begin{cases} \min U(F) \\ \text{s.c.} \quad p(F,G) \geq p_0 \\ \qquad s(F) \leq s_0 \end{cases} \qquad (C) \begin{cases} \max p(F,G) \\ \text{s.c.} \quad s(F) \leq s_0 \\ \qquad U(F) \leq U_0 \end{cases}$$

The complexity status of those problems are unknown. A basic formulation of the decision problem associated to problem (A) with $p_0 = 1$ could be the following:

" Given a strongly connected graph $G = (V, E)$ and a positive integer s_0, does there exist a family of subsets $E(v) \subseteq E$ associated to all vertices $v \in V$ such that $|E(u)| \leq s_0$ and for each pair u, v of vertices, the subgraph $G_u \cup G_v$ (as defined above) contains a path from u to v and a path from v to u ? "

2.3 Some basic properties

In the remainder of the paper, we assume that the instance graph is strongly connected, so that a solution with performance 1 always exists. Analysing problem (1), an upper bound on its optimal value $s_{opt}(G)$ for a given graph G was given in [BCH03]:

$$s_{opt}(G) \leq \min_{x \in V} \max_{v \in V} (d(x, v) + d(v, x)) \tag{2.4}$$

where $d(x, v)$ is the length of a shortest path between v and x in G. It is obtained by algorithm *Construction I*. It consists in selecting for each vertex u a subgraph composed of the union of a shortest path from x to u and from u to x, where x is the vertex that has the smallest maximal distance to and from all other vertices in V (i.e. the vertex that minimizes $\max_{v \in V}(d(x, v)+d(v, x))$). It is easy to see that this solution has $p(F, G) = 1$, and hence *Construction I* provides an admissible solution to problem (1) for any value $p_0 \in [0, 1]$. Next we exhibit some cases where this bound is not reached.

In Figure 2.1, an elementary circuit is displayed. We have $\min_{x \in V} \max_{v \in V} (d(x, v) + d(v, x)) = |V|$; but for the set F of subgraphs G_i, with G_i containing all arcs except the one entering i, we get $s(F) = |V| - 1$. Since each arc $(i, i + 1)$ is missing in exactly one subgraph (namely subgraph G_{i+1}), the whole circuit will be in any union $G_i \cup G_j$ $(i \neq j)$ and hence $p(F, G) = 1$. Therefore,

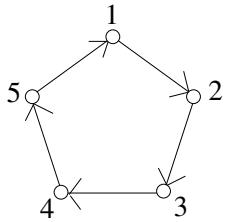

Figure 2.1: Elementary circuit

F is also an admissible solution to problem (1) for any value of p_0. In [Jac03], it was shown that this solution is actually optimal if $p_0 = 1$.

Circuits provide a class of graphs for which the above bound is not tight. This class can be extended in the following way: start with an elementary circuit $(1, 2), (2, 3), \ldots, (|V| - 1, |V|), (|V|, 1)$, and for any vertex i, $1 < i \leq |V|$, add in an arbitrary way arcs (i, j), with

$1 < j < i$. In Figure 2.2, such a graph on 8 vertices is represented. On one hand by construction this graph is hamiltonian, so the same solution as for the circuit is admissible (because $p(F, G) = 1$) and has value $s(F) = |V| - 1$. On the other hand, since the only circuit leading from any vertex i, $1 < i \leq |V|$ to 1 and from 1 to i passes by all vertices of G, $\min_{x \in V} \max_{v \in V}(d(x, v) + d(v, x)) = \min_{x \in V}(d(x, 1) + d(1, x)) = |V|$.

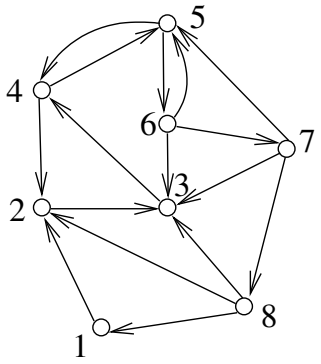

Figure 2.2: Hamiltonian graph s.t. $\min_{x \in V} \max_{v \in V}(d(x, v) + d(v, x)) = |V|$

Figure 2.3 shows a set of graphs where this bound is not attained, and where the slack between the bound and the optimal solution is arbitrarily large. Indeed, we have

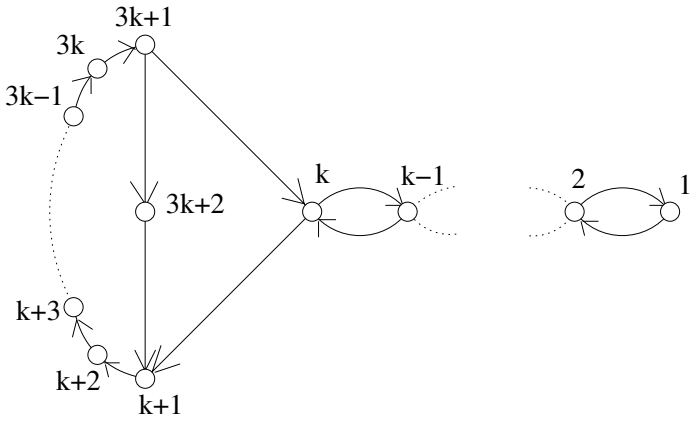

Figure 2.3: Graph for which the bound is not sharp

$$\min_{x \in V} \max_{v \in V}(d(x, v) + d(v, x))$$

$$= \min\{6k + 2, 6k, \ldots, 4k + 6, 4k + 4, 4k, 4k, 4k, \ldots, 4k, 4k, 4k, 6k + 2\} = 4k$$

but for the following set F of subgraphs

$$E(G_i) = \begin{cases} \{(1,2),(2,1),\ldots,(k-1,k),(k,k-1),(k,k+1),(k+1,k+2), & \\ (3k+1,k)\} & 1 \leq i \leq k \\ \{(k,k+1),\ldots,(3k,3k+1)\} & k+1 \leq i \leq 3k+1 \\ \{(k+2,k+3),\ldots,(3k,3k+1),(3k+1,3k+2), & \\ (3k+2,k+1)\} & i = 3k+2 \end{cases}$$

we get $s(F) = |E(G_{3k+2})| = 2k+1$. So the slack between the bound and $s(F)$ is $2k-1$. Since it depends on the value of k, this slack may be arbitrarily large.

In [BCH03], another result points out that the requirement of an equal key usage is a severe design criterion, in the sense that if it is respected, then the size of the updated certificate repositories, and therefore the communication costs, must be high. Other properties can be found in [Jac03]. Notice that the complexity of problems (1), (2), (A), (B), (C) has not been determined yet.

2.4 The case of complete graphs

Let us consider the case where G is a (strongly connected) complete graph, i.e. such that there is exactly one arc between each pair of vertices. In the next proposition, we provide an upper bound on $s(F)$ for such graphs, when a performance of 1 is required.

Proposition 2.1 *If $G = (V, E)$ is a strongly connected complete graph on n vertices, then there is a set of subgraphs F, such that $p(F, G) = 1$ and*

$$s(F) \leq \left\lceil \frac{n+2}{2} \right\rceil \tag{2.5}$$

Proof: It is known (see [Ber70]) that a complete graph is strongly connected if and only if it has a Hamiltonian circuit C. Let $0, 1, \ldots, n-1$ be the vertices consecutively visited when following the circuit C; so its arcs are $(0,1,,)(1,2),\ldots,(n-2,n-1),(n-1,0)$. In the rest of the proof, all numbers of vertices are taken modulo n. We distinguish the cases when n is even and when n is odd.

n **even:** Let us call *opposite* two vertices of the form $(i, i+\frac{n}{2})$, and *consecutive* two pairs of opposite vertices of the form $(i, i+\frac{n}{2})$, $(i+1, i+\frac{n}{2}+1)$. We show that there is always a pair of consecutive opposite vertices (i,j) and $(i+1,j+1)$, such that $(i,j) \in E$ and $(j+1,i+1) \in E$, or $(j,i) \in E$ and $(i+1,j+1) \in E$.
Start with $i = 0$ and $j = \frac{n}{2}$ and assume w.l.o.g that $(0, \frac{n}{2}) \in E$. If $(\frac{n}{2}+1, 1) \in E$, we are done. Otherwise, consider the pair $(2, \frac{n}{2}+2)$. By the same argument as before, we may

assume that $(2, \frac{n}{2}+2) \in E$. Going on this way, we may assume that $(\frac{n}{2}-1, n-1) \in E$. But then the opposite pairs of vertices $(\frac{n}{2}-1, n-1)$ and $(0, \frac{n}{2})$ have the required property. So let $(i, i+\frac{n}{2})$ and $(i+\frac{n}{2}+1, i+1)$ be in E; the following set F of subgraphs verifies $s(F) = \frac{n+2}{2}$ and $p(F,G) = 1$:

$$E(G_k) = \begin{cases} \{(i+\frac{n}{2}+1, i+1), (i+1, i+2), \ldots, \\ \quad (i+\frac{n}{2}, i+\frac{n}{2}+1)\} & i+1 \leq k \leq i+\frac{n}{2}+1 \\ \\ \{(i, i+\frac{n}{2}), (i+\frac{n}{2}, i+\frac{n}{2}+1), \ldots \\ \quad (i-2, i-1), (i-1, i)\} & i+\frac{n}{2}+2 \leq k \leq i+n \end{cases}$$

n odd: Consider the (partial) subgraph G' of G composed of all vertices of G, and only the arcs between vertices i and $i + \lfloor \frac{n}{2} \rfloor$ for all i. In G', all vertices have degree two and G' is an odd cycle. Now if we look at the orientations, at least one of its vertices has an ingoing arc and an outgoing arc. Using this last property, a similar construction as in the even case can be made, and the maximum repository size of the obtained solution F is at most $s(F) = \frac{n+3}{2}$.

■

Notice that this bound is not valid in the general case, as we have seen with elementary circuits. Moreover it is reached by at least one type of complete graphs, as for instance $G = (V, E)$, with $n = |V|$ even.

$$V = \{1, 2, \ldots, n\}$$

$$E = \{(i, i+1) : i = 1, \ldots, n-1\} \cup \{(j, i) : 1 \leq i < j - 1 < n\}$$

Proposition 2.2 *For G, any solution F with $p(F,G) = 1$ verifies $s(F) \geq \frac{n+2}{2}$.*

Proof: Assume to the contrary that there exists F with $s(F) \leq \frac{n}{2}$, i.e. $|E(G_u)| \leq \frac{n}{2}\ \forall\ u \in V$. First notice that in G, the only way to go from vertex 1 to vertex n is through the path $(1,2), (2,3), \ldots, (n-1, n)$. Since a performance of 1 is required, this path must be in $G_1 \cup G_n$, and hence $E(G_1)$ and $E(G_n)$ form a partition of $\{(1,2), (2,3), \ldots, (n-1, n), (n, 1)\}$.
Assume $(n, 1) \in E(G_1)$, thus $|\{(1,2), (2,3), \ldots, (n-1, n)\} \cap E(G_1)| = \frac{n}{2} - 1$. If $(n-1, n)$ is also in $E(G_1)$, then $E(G_n) \subseteq \{(1,2), (2,3), \ldots, (n-2, n-1)\}$. Since the path $\{(1,2), (2,3), \ldots, (n-2, n-1)\}$ has to be in $E(G_1) \cup E(G_{n-1})$ and $|\{(1,2), (2,3), \ldots, (n-2, n-1)\} \backslash E(G_1)| = \frac{n}{2}$, then $E(G_{n-1}) \subseteq \{(1,2), (2,3), \ldots, (n-2, n-1)\}$, and no arc leaves n in $G_n \cup G_{n-1}$, which is impossible. So $(n-1, n) \notin E(G_1)$, thus $(n-1, n) \in E(G_n)$. Consider now $G_1 \cup G_{n-1}$. It must contain either $\{(1,2), (2,3), \ldots, (n-1, n), (n, 1)\}$ or $\{(1,2), (2,3), \ldots, (n-2, n-1), (n-1, 1)\}$. Since $(n, 1) \in G_1$, we have in both cases $|E(G_{n-1}) \backslash \{(1,2), (2,3), \ldots, (n-2, n-1)\}| = 1$. In the first case, the remaining arc in $E(G_{n-1})$ is $(n-1, n)$ and in the second case it is $(n-1, 1)$. In both cases, no arc leaves n in $G_n \cup G_{n-1}$, which contradicts $(n, 1) \in E(G_1)$.

The case $(n,1) \in E(G_n)$, is treated in a similar way, by replacing $n-1$ by 2 and inverting 1 and n. ■

It seems that one can do much better in most strongly connected complete graphs, the best case being when one can reach $s(F) = 3$. Consider for instance the graph, on n vertices, whose arcs are all (i,j) with $i < j$, except for $(n,1)$; the hamiltonian circuit being $(1,2),(2,3),\ldots,(n-1,n),(n,1)$. Then the solution

$$E(G_k) = \begin{cases} \{(1,2),(2,n),(n,1)\} & k = 1 \text{ or } n \\ \{(1,k),(k,n),(n,1)\} & \text{otherwise} \end{cases}$$

is of the appropriate size. Next proposition shows that it is impossible, in complete graphs with at least four vertices, to have $s(F) = 2$, and $p(F,G) = 1$.

Proposition 2.3 *For a complete strongly connected graph $G = (V,E)$ with $n = |V| > 3$, for any solution F such that $p(F,G) = 1$, we have $s(F) \geq 3$.*

Proof: Assume $s(F) \leq 2$. We need to show that for at least one pair $\{u,v\}$ of vertices, $u \not\to_{G_u \cup G_v} v$ or $v \not\to_{G_u \cup G_v} u$. We may assume that $|G_u| = 2 \ \forall\ u \in V$, since adding an arc to a repository will not decrease the performance.
We will call *self-ingoing* an arc of G_u of the form (x,u) and *self-outgoing* an arc of G_u of the form (u,x). Notice first that if G_u has no self-ingoing arc, then any other repository G_v must contain an arc of the form (x,u), so that $v \to_{G_u \cup G_v} u$. If two repositories G_u and $G_{u'}$ have no self-ingoing arc, any other repository must be of the form $\{(x,u),(y,u')\}$. But then for v, v' different from u and u', $v \not\to_{G_v \cup G_{v'}} v'$.
So there is at most one vertex u such that G_u has no self-ingoing arc. Assume next that exactly one such vertex exists. Then any repository G_v $(v \neq u)$ is composed of one arc of the form (x,u) and one self-ingoing arc (y,v). Notice that x may be equal to v, or y may be equal to v, but both can not happen since there is exactly one arc linking each pair of vertices in G. Hence for any pair $\{v,v'\}$ of vertices $v \neq u \neq v'$, $v \not\to_{G_v \cup G_{v'}} v'$, or $v' \not\to_{G_v \cup G_{v'}} v$.
So each repository has a self-ingoing arc, and symmetrically a self-outgoing arc. Consider now

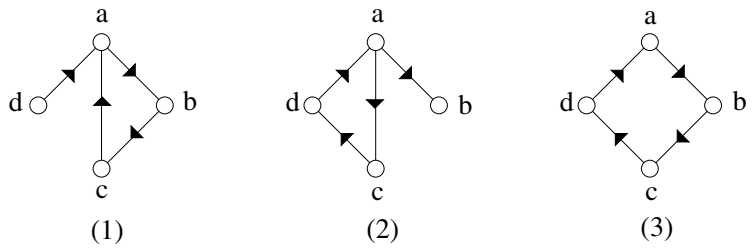

Figure 2.4: Repositories G_a and G_c.

repositories G_a and G_c. In Figure 2.4 are displayed the three possible cases (where b and d can

be any other vertices of G): if $(c,a) \in G_c$ we are in case (1), if $(a,c) \in G_c$ we are in case (2) (cases $(c,a) \in G_a$ and $(a,c) \in G_a$ are symmetrical), if $(c,a) \notin G_a \cup G_c$ and $(a,c) \notin G_a \cup G_c$ we are in case (3). In case (1), in order to have $c \rightarrow_{G_c \cup G_d} d$ and $d \rightarrow_{G_c \cup G_d} c$, we must necessarily have $G_d = \{(c,d),(d,b)\}$, but then $a \not\rightarrow_{G_a \cup G_d} d$. In case (2), in order to have $b \rightarrow_{G_b \cup G_c} c$ and $c \rightarrow_{G_b \cup G_c} b$, we must necessarily have $G_b = \{(d,b),(b,c)\}$, but then $b \not\rightarrow_{G_a \cup G_d} a$. In case (3), to have $c \rightarrow_{G_b \cup G_c} b$, the self-ingoing arc of G_b must be (d,b), but then $b \not\rightarrow_{G_a \cup G_b} a$ (since $(b,a) \notin E$). ■

Finally, notice that in Section 2.3, all examples of graphs where *Construction I* fails to give an optimal solution are not complete. Moreover, for all examples of graphs seen in this section, the solution proposed minimizing $s(F)$ could have been obtained with it. Although we conjectured first that it is always the case, we found a complete graph where the solution provided by *Construction I* is not optimal. It is displayed in Figure 2.5.

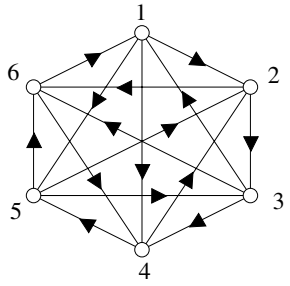

Figure 2.5: Complete graph where *Construction I* fails to give an optimal solution to Problem (1), with $p_0 = 1$.

For this graph, one can see that $d(4,1) = d(2,5) = d(6,3) = 3$, so the solution F given by *Construction I* verifies $s(F) \geq 4$, while solution F'

$$E(G_k) = \begin{cases} \{(1,2),(2,6),(6,1)\} & k = 1 \text{ or } 2 \\ \{(2,3),(3,4),(4,2)\} & k = 3 \text{ or } 4 \\ \{(4,5),(5,6),(6,4)\} & k = 5 \text{ or } 6 \end{cases}$$

has $s(F') = 3$, with performance still at 1.

2.5 Heuristic procedures

A general formulation of a combinatorial optimization problem could be the following: given a (discrete) set of solutions S (also called *solution space*) and an objective function f (that associates to each solution $s \in S$ a real number $f(s)$), find a solution s_{opt} minimizing (or maximizing) the value of $f(s)$. In our case, a solution s is a set of subgraphs of G denoted by

F. A *heuristic* is an algorithm which constructs a "good" solution, i.e. not necessarily optimal. It is most often used to find such solutions in reasonable time when the problem is known or suspected to be $\mathcal{NP}$-hard. One distinguishes generally three basic types of heuristics: local search heuristics, constructive heuristics, and evolutive heuristics [Cos95].

An evolutive heuristic involves a population of several solutions (or parts of solutions) that will cooperate, combine and improve themselves. In our case, a solution is a set of subgraphs of G, but since the graphs under consideration have more than 1000 vertices, dealing with a dozen of such solutions would lead to a prohibitive use of memory. So we did not concentrate on this kind of heuristics for our problem.

A constructive heuristic constructs a complete solution by increasing step by step a partial solution. Several constructive heuristics have been developed. *Construction I* is an example of such an algorithm. Another example described in [BCH03] is the *Maximum Degree algorithm*, which constructs the repositories of the vertices such that each local repository has a predefined size.

A local search algorithm starts from any solution and tries to improve it step by step. In such heuristics, a set called *neighbourhood of* s, $N(s) \subseteq S$ associated to each solution $s \in S$ is used. The solutions in $N(s)$ are called *neighbour solutions* of s, and are obtained applying local modifications on s according to precise rules that depend on the problem considered. These modifications are called *movements*. From an initial solution $s_0 \in S$, a local search method generates some solutions $s_1, s_2, \ldots \in S$ such that $s_{i+1} \in N(s_i)$. The difference between different local search algorithms lies essentially in the way of defining the neighbourhood $N(s)$ associated to a solution s and the way of choosing a solution in $N(s)$. As far as we know, local search heuristics have not been used systematically for problems as considered here.

Construction I provides solutions with performance 1, such that each subgraph is composed of two shortest paths. Those solutions are in most cases good for problems not involving $U(F)$ (especially for problem (A) with $p_0 = 1$). The main drawback of this algorithm lies in that it produces the worst possible $U(F)$, and hence gives bad results for problem (B). Our goal is to design a construction preserving the advantages of *Construction I*, while trying to distribute in a better way the usage between the vertices.

We propose a variant, which we call *Construction k among l*. The idea is as follows: choose l vertices ($l << |V|$) and set $k = \lceil \frac{l+1}{2} \rceil$. For each vertex $v \in V(G)$, construct the subgraph G_v as a circuit passing by at least k of the l vertices and by v. Since two subsets of size k of a set of size stricly lower than $2k$ have a non-empty intersection, it follows that for any pair of vertices (u, v), G_u and G_v are two intersecting circuits. As a consequence, there is always a path from u to v and from v to u in $G_u \cup G_v$, so $p(F, G) = 1$.

This construction may be seen as a generalization of *Construction I*, which is obtained by choosing $l = 1$. In comparison with *Construction I*, performance will be maintained at 1, while

the maximum repository size will increase and the maximum usage will decrease. More precisely, the maximum usage may decrease a little, but will be at least $\frac{k}{l}|V| = \lceil\frac{l+1}{2}\rceil\frac{|V|}{l} \geq \frac{l}{2}\frac{|V|}{l} = \frac{|V|}{2}$, and when we pass from l to $l+2$ (l odd), the bound decreases by at most

$$\begin{aligned}\lceil\frac{l+1}{2}\rceil\frac{|V|}{l} - \lceil\frac{l+3}{2}\rceil\frac{|V|}{l+2} &= \lceil\frac{l+1}{2}\rceil(\frac{|V|}{l} - \frac{|V|}{l+2}) - \frac{|V|}{l+2} \\ &= \frac{l+1}{2}(\frac{|V|(l+2)-|V|l}{l(l+2)}) - \frac{|V|}{l+2} \qquad (l \text{ is odd}) \\ &= \frac{|V|}{l(l+2)}\end{aligned}$$

Hence the gain is at most $\frac{|V|}{3}$ if l goes from 1 to 3, then it is at most $\frac{|V|}{15}$ if l goes from 3 to 5, then $\frac{|V|}{35}$ and so on. From this observation and for the sake of simplicity, we decided to focus on construction *2 among 3*.

Here is the sketch of the algorithm (details will follow):

Input: Graph G, integer number l
Output: F with $p(F,G) = 1$, $\lceil\frac{l+1}{2}\rceil\frac{|V|}{l} \leq U(F) \leq |V|$ and reasonable $s(F)$.

1: Choose a triplet of vertices v_1, v_2, v_3;
2: **for** each $v \in V$ **do**
3: Choose two vertices x and y among v_1, v_2 and v_3;
4: G_v is either the circuits composed of the shortest paths from v to x then from x to y and from y to v, or from v to y, from y to x and from x to v.
5: **end for**

In an analogous way to *Construction I* where the maximum usage is reached by the vertex minimizing the total distance, in *Construction 2 among 3* it is likely to be attained by a vertex belonging to the triplet chosen in Step 1. Since we want a maximum usage as low as possible, the triplet is chosen, if possible, among those such that no vertex, like v_1, belongs to the[5] shortest path from v_2 to v_3 or from v_3 to v_2. Among those triplets, the one chosen will minimize the sum (over v_1, v_2 and v_3) of the total distance to and from all vertices. In Steps 3 and 4, both vertices and the direction of the circuit are chosen such that the resulting subgraph is as small as possible, but each vertex in $\{v_1, v_2, v_3\}$ has a selection quota of $\frac{2}{3}|V|$ (so that the usage is well distributed among v_1, v_2 and v_3).

Table 2.1 summarizes the results of the tests comparing this new constructive heuristic with *Construction I*. The graphs were provided by the LCA and correspond to certificate graphs of mobile ad hoc networks (real or generated, with the good properties).
Those graphs are quite sparse, since the smallest one has 1345 vertices, and those vertices have on average only about 10 ingoing and 10 outgoing arcs. With *Construction I* we always get $U(F) = |V| - 1$, and with both constructions we obtain $p(F,G) = 1$. As we can easily deduce these data from $|V|$, they are not in the table.

[5]Several shortest paths may exist, but only one is computed.

$\|V\|$ (nb)	$s(F_I)$	$s(F_{2/3})$	$U(F_{2/3})$
2516 (9)	15.9	18.9	1717
4000 (7)	10.1	11.7	2693
4578(10)	11.4	14.9	3583
5174(10)	12.0	12.7	4027
5765(10)	12.2	14.7	4687
6466(10)	13.6	16.0	5092
7278(10)	14.7	17.5	5490
7775(10)	13.5	16.7	6140
8444(10)	15.5	18.2	6779
9180(10)	16.4	18.9	6962
9765(10)	17.6	21.5	7680

Table 2.1: Comparison between *Construction I* and *Construction 2 among 3*. All values are averages on nb graphs.

In [BCH03] it was observed that *Construction I* gives subgraphs with size $\sim \log |V|$, since each repository consists of two shortest paths. Here we have the same property, since each repository consists only of three such paths. The results show that both constructions provide values of $s(F)$ which are close to each other. Moreover, the maximum usage is significantly decreased with *Construction 2 among 3* and is not far from $\frac{2}{3}|V|$.

The lower bound $\lceil \frac{|V|}{2} \rceil$ on the maximum usage provided by this last construction can be further improved, while keeping the performance value at one. The following construction is called *Construction IC* (for intersecting circuits).

Input: Graph G, integer number l
Output: F with $p(F,G) = 1$, $\lfloor \frac{|V|}{l} \rfloor + \lceil \frac{|V|}{l} \rceil \leq U(F) \leq |V|$ and reasonable $s(F)$.

1: Set $L = \frac{l(l-1)}{2}$;
2: Choose L vertices and rename them $v_{1,2}, v_{1,3}, \ldots, v_{1,l}, v_{2,3}, \ldots, v_{2,l}, \ldots, v_{l-1,l}$;
3: For all $k = 1, \ldots, l$, let A_k be those $l-1$ vertices $v_{i,j}$, such that $i = k$ or $j = k$;
4: Partition V into l subsets $V_1, \ldots, V_l$ of sizes $\lfloor \frac{|V|}{l} \rfloor$ and $\lceil \frac{|V|}{l} \rceil$;
5: **for** each $v \in V$ **do**
6: Let k be such that $v \in V_k$;
7: Set G_v as a circuit passing by v and all vertices in A_k;
8: **end for**

To see that the lower bound on the maximum usage is now $\lfloor \frac{|V|}{l} \rfloor + \lceil \frac{|V|}{l} \rceil$, one just needs to remark that each subgraph G_v with $v \in V_i \cup V_j$ will contain a vertex of the form $v_{i,j}$, and that

at least one set V_i has size $\lceil \frac{|V|}{l} \rceil$. So we have

$$U(F) \geq \max_{i,j}(|V_i| + |V_j|) \geq \lfloor \frac{|V|}{l} \rfloor + \lceil \frac{|V|}{l} \rceil$$

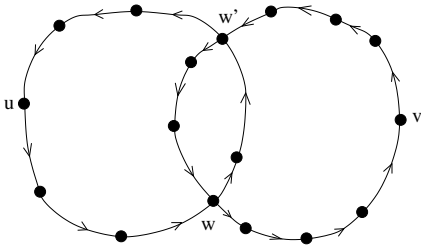

Figure 2.6: Two circuits intersecting

To see that the performance of the solution obtained is one, consider two vertices u and v in G. Notice first that if a circuit passing by u and a circuit passing by v both pass by at least one common vertex w, there will be a path from u to v and a path from v to u using only arcs from the above circuits. In Figure 2.6, we have for instance $u \rightarrow_{G_u \cup G_v} w \rightarrow_{G_u \cup G_v} v \rightarrow_{G_u \cup G_v} w' \rightarrow_{G_u \cup G_v} u$.

Now if u and v belong to the same subset of the partition defined at Step 4, say V_k, both G_u and G_v will have all vertices from A_k in common. If u and v belong to different sets, say V_k and $V_{k'}$ with $k < k'$, then $v_{k,k'} \in V(G_u \cup G_v)$.

For the choice of the vertices (Step 2), one should apply the same type of criterion as for *Construction I* and *2 among 3*.

For the construction of the circuits G_v (Step 7), a way would be to first define a selection quota of $2\lceil \frac{|V|}{l} \rceil$ for each vertex and to initialize the number of selections at 0 for the vertices which have not been chosen in Step 2, and at $|V_i| + |V_j|$ for each vertex $v_{i,j}$, since it already belongs to each subgraph in the set $\{G_m, m \in A_i \cup A_j\}$. Then starting from v connect the vertices of the corresponding set A_k in a greedy way (nearest first), constructing at each step a shortest path in the graph in which the vertices having reached their quota are temporarily removed. If no vertex can be connected, put back all removed vertices and increase the quota (by adding a constant for instance). Each time a subgraph G_v is constructed, update the number of selections.

This construction has not been tested in general, but we notice that *Construction 2 among 3* may also be considered as *Construction IC* for $l = 3$, and the results ($U(F)$ close to $\frac{2}{3}|V|$ and relatively small $s(F)$) obtained for it are promising.

Finally, though it seems to provide larger values for $s(F)$ than *Construction I*, we can see that in the graph of Figure 2.7 *Construction 2 among 3* provides arbitrarily smaller $s(F)$. Indeed, in this example *Construction I* gives $s(F) = 12$, and more generally $2\lceil \frac{|V|}{2} \rceil$ for such graphs, while with *Construction 2 among 3*, one gets $2\lceil \frac{|V|}{3} \rceil$ in general. In the example of Figure 2.7, a solution with $s(F) = 8$ would be

$$E(G_k) = \begin{cases} \{(1,2),(2,1),(2,3),(3,2),\ldots,(4,5),(5,4)\} & 1 \leq k \leq 5 \\ \{(5,6),(6,5),(6,7),(7,6),\ldots,(8,9),(9,8)\} & 6 \leq k \leq 9 \\ \{(9,10),(10,9),\ldots,(11,12),(12,11),(12,1),(1,12)\} & 10 \leq k \leq 12 \end{cases}$$

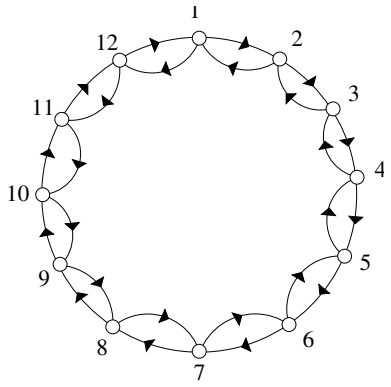

Figure 2.7: Graph where *Construction 2 among 3* provides a smaller $s(F)$ value than *Construction I.*

2.6 Tabu search

Although the above simple procedures seem to provide reasonably good solutions for the problems considered here, we will nevertheless describe an adaptation of the general technique of tabu search to our context. This procedure may be useful for other situations where additional requirements or different objectives may be present. Moreover the reliability of the procedure (as shown by the many situations where it has been used successfully) makes it an easy tool for evaluating the performance of other techniques, by comparison.

The tabu algorithm is a local search heuristic developed by Glover [Glo86] and Hansen [Han86], which can be described as follows. In each step, a solution $s_{i+1} \in N(s_i)$ is chosen. When a movement from s_i to s_{i+1} is made, the reverse movement is introduced in a finite list called *tabu list* (TL), and it is forbidden to perform this movement during t steps (t being a parameter fixed by the user). The forbidden movements are called *tabu movements.* The stopping criterion may be the total running time, the total number of steps or the number of steps without improving the best solution encountered so far.

In order to allow comparison with *Construction 2 among 3*, we decided to apply tabu search to problem (B). We define the solution space as the set of subgraphs F such that each subgraph G_u is the union of an in-tree and an out-tree of root, respectively anti-root u. A neighbour solution of F is defined as a solution (i.e. a set of subgraphs of G with the above property) obtained from F by adding or removing one or several arcs in some of the subgraphs composing F. The objective function is:

$$f(F) = c_u \cdot U(F) + c_0 \cdot \sum_{v \in V(G)} (s_F(v) - s_0)_+ + c_1 \cdot (p_0 - p(F,G))_+ \tag{2.6}$$

where c_u, c_0 and c_1 are arbitrary constant coefficients (we choose $c_u = 1$, $c_0 = c_1 = |V|$), and x_+ stands for x if $x > 0$ and 0 otherwise. These values for the coefficients ensure that any solution satisfying the constraints on the performance and the repository size will have a

better (lower in our case) objective value than a solution violating one of them. We use three different neighbourhoods, depending on the quality of the current solution (which constraints are violated). They are given in pseudocode in Figure 2.8 and we give here some explanations.

If both constraints are satisfied, we are in Case 1, and the solution will be transformed so as to have a lower maximum usage. This is achieved by randomly choosing a vertex among those with usage equal to $U(F)$, and a graph G_j containing an arc incident to i, such that vertex j is not tabu. If no such non tabu j exists, nothing is done during this iteration. Otherwise, the current solution is modified by removing from G_j all arcs incident to i, and try to recover the structure of a union of an in-tree and an out-tree of G_j by adding some arcs not incident to i. This can be done for instance by finding a path in $G\backslash\{i\}$ connecting each (ordered) pair $\{u, v\}$ of vertices such that v is not reachable by u anymore after the transformation. If no such path exists, G_j is reconstructed by inserting a limited number of those arcs incident to i.

If the constraint on the repository size is violated but not the performance constraint, or if there is no vertex i with $s_F(i) < s_0$, we are in Case 2. Neighbourhood solutions will be constructed so as to have a lower maximum repository size, but about the same performance. They are obtained by first randomly choosing a vertex i maximizing $s_F(i)$ among the non tabu vertices. Then one arc e incident to a leaf is chosen in G_i, so that its removal from G_i minimizes the size of the set $D\backslash D_e$ of vertices v for which there is no path anymore to or from i in $G_i \cup G_v$. Since e is chosen incident to a leaf, $G_i \setminus \{e\}$ is still the union of an in-tree and an out-tree. At step (iii) of Case 2, if there are some vertices $v \in D\backslash D_e$, whose subgraph can be increased without violating the corresponding constraint, e is inserted into each such G_v.

If we are neither in Case 1 nor in Case 2, we are in Case 3: the performance constraint is violated and there is at least one vertex i with $s_F(i) < s_0$. The neighbourhood will then consist of solutions with possibly better performance. For their construction, choose first a non tabu vertex i such that $s_F(i) < s_0$. If no such vertex exists, nothing is done during this iteration. If such an i exist, choose one for which few nodes v have a path from or to i in $G_v \cup G_i$ (so an augmentation of G_i may easily increase $p(F, G)$), while $|s_0 - s_F(i)|$ is large (so that many arcs can be added to G_i without violating the constraint on the repository size). Then insert in G_i as many arcs as possible (but at most $|s_0 - s_F(i)|$), so that G_i remains the union of an in-tree and an out-tree, and so that $U(F)$ (the objective function) is not increased. Among the candidate arcs, those with many connections outside G_i will be preferred.

An important point is that at each iteration only a subset of the neighbourhood is considered, since a complete exploration may be too time consuming. When the neighbourhood has been chosen, the best neighbour (i.e. the one minimizing the objective function) is selected and the current solution, and if necessary the best solution found, are updated. At the end of each iteration, the tabu list is also updated by removing the eldest vertex from it and inserting a new vertex into it: if we are in Cases 1 or 3, it is the only vertex whose subgraph has been modified (if no modification has been made, no vertex is inserted in the list); in Case 2, and

- **Case 1 :** $s(F) \leq s_0$ and $p(F,G) \geq p_0$:
  - (i) choose $i \in V(G)$ maximizing $U_F(i)$
  - (ii) choose $j \in V(G)$ not tabu (if possible) s.t. $i \in V(G_j)$
  - (iii) set $G'_j = G_j \setminus \{e \in E(G_j) \,|\, e \text{ incident to } i\}$,
  - (iv) add arcs not incident to i (if possible, otherwise limit the use of such arcs) to G'_j so as to get $\forall\, u \in V(G'_j)$: $\quad (j \rightarrow_{G_j} u \Rightarrow j \rightarrow_{G'_j} u)$ and $(u \rightarrow_{G_j} j \Rightarrow u \rightarrow_{G'_j} j)$
  - (v) set $G_j = G'_j$.
- **Case 2 :** $(p(F,G) \geq p_0$ and $s(F) > s_0)$ or $(s_F(i) \geq s_0 \;\forall\, i \in V(G))$:
  - (i) choose $i \in V(G)$ not tabu maximizing $s_F(i)$
  - (ii) choose $e \in E(G_i)$, e incident to a leaf, minimizing $|D| - |D_e|$ where:
    $D = \{\, v \in V \,|\, v \rightarrow_{G_v \cup G_i} i \text{ or } i \rightarrow_{G_v \cup G_i} v\}$,
    $D_e = \big\{\, v \in V \,|\, v \rightarrow_{G_v \cup G_i \setminus \{e\}} i \text{ or } i \rightarrow_{G_v \cup G_i \setminus \{e\}} v\big\}$
  - (iii) set $G_i = G_i \setminus \{e\}$ and $G_v = G_v \cup \{e\}\; \forall\, v \in D \setminus D_e$ and s.t. $s_F(v) < s_0$.
- **Case 3 :** $p(F,G) < p_0$ and there is at least one vertex i s.t. $s_F(i) < s_0$
  - (i) choose $i \in V(G)$ not tabu (if possible), s.t. $s_F(i) < s_0$, and minimizing $\dfrac{|D|}{s_0 - s_F(i)}$
    where $D = \{\, v \in V(G) \,|\, v \rightarrow_{G_v \cup G_i} i \text{ or } i \rightarrow_{G_v \cup G_i} v\}$
  - (ii) choose $k \leq s_0 - s_F(i)$ arcs $(x,y)^a \in B$ where B is:
    $B = \{(x,y) \in E(G) | (x \in G_i,\; y \notin G_i \text{ and } U_F(y) < U(F))$,
    or $(y \in G_i,\; x \notin G_i \text{ and } U_F(x) < U(F))\}$,
  - (iii) insert the chosen arcs in G_i.

[a] choose $(x,y) \in B$ maximizing $d^+_G(y)$ if $i \rightarrow_{G_i} y$, or maximizing $d^-_G(x)$ if $x \rightarrow_{G_i} i$.

Figure 2.8: Neighbourhoods considered in the tabu method

only if $s_F(i) \leq s_0$ (otherwise no vertex is inserted), it is the one corresponding to the subgraph from which some arcs were removed.

2.7 Computational experiments

We use the same set of graphs for our computational experiments as those used to compare *Construction I* with *Construction 2 among 3*, but our simulations are done only on graphs with up to 4000 vertices, for time and memory capacity reasons. Simulations are run on a machine with a processor 2.8GHz, and 3GB of main memory.

The constraints are fixed in order to allow comparison with the Maximum Degree Algorithm previously designed by the LCA: $s_0 = 60$, 80 and $p_0 = 0.9$, 0.95. Previous experiments made us fix the neighbourhood size at 20 and the stopping criterion at 10'000 iterations without improvement, but at most 100'000 iterations. Computations times are not written, but they vary from half an hour to 35 hours for the graphs for which the limit of 100'000 was reached. In order to point out the benefit of using a tabu method, two different lengths of the tabu list are tested: 0 and $\frac{1}{4} \cdot |V|$.

The initial solution is obtained in a constructive manner we fixed after several attempts. We first tried the set of subgraphs G_i consisting in the outgoing and ingoing arcs of the vertex i. This solution is of course not admissible for sparse graphs if one wants a performance close to 1. Results were not satisfying, since it appeared that the tabu method spent most of the time trying to increase the performance (neighbourhood of Case 3), but could not reach p_0. The second construction we tried was the set of subgraphs G_i consisting of outgoing and ingoing paths (from and to i) randomly chosen, but we had the same trouble as for the previous construction. Then we tried to run our algorithm starting with *Construction I*. In that case the initial solution satisfied the performance constraint, but the repository size constraint was not. So the first iterations of the tabu search used the neighbourhood of Case 2, and a lot of time was spent to repair the repository size constraint, before using the neighbourhood of Case 1. Finally we tried *Construction 2 among 3* for the initial solution, and this gave us even better results, since the initial solution had already a good maximum usage. This can be due to the lack of symmetry among the vertices in solutions provided by *Construction I*, as compared to *Construction 2 among 3*. We decided to use this last one for our simulations.

We point out that the experiments we made with $p_0 = 1$ and *Construction I* or *Construction 2 among 3* for the initial solution gave somewhat disappointing results. This can be explained by observing that the removal of any arc from a repository in a solution obtained with *Construction I* or *Construction 2 among 3* will imply a big loss of performance, which will be difficult to repair without just putting back in the repository the arc which was removed $|TL|$ iterations before, even if $|TL|$ is large.

$\|V\|(nb)$	p_0	s_0	$s(F)$	$U(F)$	$p(F,G)$
2216 (7)	0.9	60	60	1376	0.9
		80	80	1373	0.9
	0.95	60	60	1429	0.95
		80	80	1428	0.95
4000 (7)	0.9	60	47	2520	0.9
		80	61	2518	0.9
	0.95	60	53	2597	0.95
		80	71	2596	0.95

Table 2.2: $|TL| = 0$ (values are averages over nb graphs)

$\|V\|(nb)$	p_0	s_0	$s(F)$	$U(F)$	$p(F,G)$
2216 (7)	0.9	60	53	1359	0.9
		80	64	1350	0.9
	0.95	60	50	1422	0.95
		80	60	1417	0.95
4000 (7)	0.9	60	21	2498	0.9
		80	21	2498	0.9
	0.95	60	21	2593	0.95
		80	21	2593	0.95

Table 2.3: $|TL| = \frac{|V|}{4}$ (values are averages over nb graphs)

The simulations show that tabu search (i.e. $|TL| > 0$) gives on average better solutions than a simple local search. It often occurs (especially for large graphs, and with the tabu search) that the solution found has a maximum subgraph size $s(F)$ much lower than s_0, while such a difference never occurs for the performance. This is not surprising, since minimizing $s(F)$ and minimizing $U(F)$ both tend to decrease the overall size of the solution, while maximizing the performance tends to increase it.

2.8 Conclusion

In this chapter, we provided various examples where the bound given by (2.4) is not sharp. Indeed, in each graph class we studied, we could find an example where *Construction I* does not give an optimal solution. We presented then the case of complete graphs with the constraint $p_0 = 1$, and proved that there is always a solution with maximum repository size of about the half of the total number of vertices. We also showed that the maximum repository size has a lower bound of three, which is attained for very specific complete graphs. We then focused on problem (B), and proposed two ways of generalizing *Construction I* with several central vertices, in order to diminish the maximum usage, while maintaining the performance at level 1. Implementation of this construction with 3 central vertices provided satisfactory results. Next, with the tabu search approach, we provided a general framework to solve problem (B), for any values of p_0 and s_0. The flexibility of this approach permits an easy adaptation for solving the four other problems or even some variations.

However, if we focus on the cases where the performance must be close to 1, the tabu search seems less accurate than *Construction I* and *Construction 2 among 3*. Indeed, the maximum usage has a little decreased with the tabu approach, but it is at the cost of a slightly decreased performance, and much higher computation time. This shows that, although this approach could most probably be reasonably improved, the above constructive heuristics seem to provide solutions of reasonable quality, at least for problems (1), (A) and (B).

Finally it would be useful to establish the complexity status of the above problems, so that the use of heuristic procedures could be justified. To end up, we formulate an alternative decision problem, similar to problem (A), but where the subgraphs are not partial but induced subgraphs.

" Given a strongly connected graph $G = (V, E)$, does there exist a family of subsets $V(u) \subseteq V$ associated to all vertices $u \in V$, such that $|V(u)| \leq k$ for each vertex u, and for each pair u, v of vertices, the subgraph induced by $V(u) \cup V(v)$ contains a path from u to v and a path from v to u ? "

To our knowledge the complexity status of this problem is not known.

Chapter 3

A solvable case of image reconstruction in discrete tomography

In this chapter, we study a special case of the problem of reconstructing an image given the number of occurrences of each color in each row or column. The formulation will be based on graph theory concepts and this will allow us to show that this case can be solved in polynomial time; it generalizes earlier known solvable cases.

3.1 Introduction

The aim of tomography is to reconstruct an object from its projections. Indeed, while the projections of an object in given directions are uniquely defined, the converse is in general not true. It is thus natural to ask whether, given such projections, it is possible to construct with certainty the underlying object, and whether such an object even exists.

The most frequent application of tomography is in medical imagery. In this continuous case, a projection of a 3-dimensional object in a given direction is a 2-dimensional function giving the width with respect to a given direction. Given three (or more) of those projections, one wants to reconstruct the whole object. Discrete tomography is when the object can be expressed with a discrete set of data. An application in physics is the reconstruction of a picture taken by an electron microscope, which measures the number of atoms in each line of some direction [BLNP95, GG97, CD01]. Discrete tomography also has many applications in computer science, for instance in pattern recognition, image processing and data compression. The most theoretically studied discrete cases are when the objects are 2-dimensional, and two projections are given. For instance one may ask how to (and whether it is possible to) reconstruct a binary

matrix, given the number of occurrences of 1 in each line and in each column [Rys63]. Another problem which has been investigated is the one of reconstructing a convex polyomino (object drawn on a 2-dimensional grid), given its horizontal and vertical projections, where projections represent the total length of its contour line on each (horizontal or vertical) coordinate [Pic02], see Figure 3.1. A further problem consists in reconstructing a domino tiling of a rectangle, given

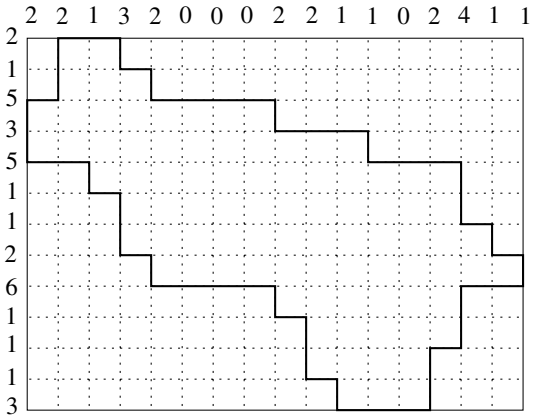

Figure 3.1: A convex polyomino and its projections.

for each line (row or column) the number of domino covering at least one cell on that line. This problem is open and in [Pic01], it is shown that it can be solved in polynomial time when some convexity requirements on the given projections are added. For more theoretical and practical aspects, see [HK99].

The object we aim at reconstructing in this chapter is an image, defined by a matrix containing a color in each cell. The projections are the number of occurrences of each color in each row and each column. We consider a special case and show that it can be solved in polynomial time. The complexity status of a slight extension of this solvable case is still open; so our result is a step towards the boundary between easy and difficult problems in image reconstruction. What follows is based on [CPSdW03].

3.2 Problem formulation

We shall define the general image reconstruction problem as follows: an image of $(m \times n)$ pixels of p different colors has to be reconstructed. For convenience we consider that there is in addition a color $p+1$ which is the ground color. We are given the number $\alpha(i, s)$ of pixels of each color s in each row i and also the number $\beta(j, s)$ of pixels of each color s in each column j; is it possible to reconstruct an image, i.e. can one assign a color s to each entry $[i, j]$ of the image in such a way that there are $\alpha(i, s)$ occurrences of color s in each row i and $\beta(j, s)$ occurrences of color s in each column j, for all i, j, s?

For a solution to exist we must necessarily have

$$\begin{array}{lcll} \sum_{s=1}^{p+1} \alpha(i,s) & = & n & (i = 1, \ldots, m) \\ \sum_{s=1}^{p+1} \beta(j,s) & = & m & (j = 1, \ldots, n) \\ \sum_{i=1}^{m} \alpha(i,s) & = & \sum_{j=1}^{n} \beta(j,s) & (s = 1, \ldots, p+1) \end{array}$$

These conditions are necessary but not sufficient for the existence of a solution. The smallest example where these conditions are not sufficient is displayed in Figure 3.1.

	$\beta(1,1) = 2$ $\downarrow$	$\beta(2,1) = 0$ $\downarrow$
$\alpha(1,1) = 2,\ \alpha(1,2) = 0 \rightarrow$		
$\alpha(2,1) = 0,\ \alpha(2,2) = 2 \rightarrow$		

Table 3.1: An example where the conditions are not sufficient.

This simplified version of image reconstruction problems occurring in discrete tomography is denoted by $R(m, n, p)$; it is a combinatorial problem whose complexity status is unknown for $p = 2$ colors (i.e. when we have $p + 1 = 3$ colors including the ground color). It is $\mathcal{NP}$-complete for $p \geq 3$ (see [CD01, HK99]). In [CPdW03] some special cases solvable in polynomial time have been presented. Notice that it is solvable if $p + 1 = 2$ (see [Rys63]).

3.3 Graph theoretical formulation

We associate with the problem a complete bipartite graph $G = K_{m,n}$ on two sets of vertices R, S with sizes m and n. Each edge (i, j) of $K_{m,n}$ corresponds to entry $[i, j]$ in row i and column j of the $(m \times n)$ array.

The image reconstruction problem can be interpreted as follows: the entries of color s in the array correspond to a subset B_s of edges (a partial subgraph of $K_{m,n}$) such that B_s has $\alpha(i, s)$ edges adjacent to vertex i of R and $\beta(j, s)$ edges adjacent to vertex j of S. We have to find a partition $B_1, B_2, \ldots, B_{p+1}$ of the edge set of $K_{m,n}$ where each B_s satisfies the above degree requirements.

As mentioned above, for the case $p+1 = 3$, the complexity is unknown. The problem is solvable in polynomial time with $p + 1 = 4$ colors (see [CPdW03]) if $p = 3$ colors, say colors 1, 2 and 3, are *unary*, i.e. $\alpha(i, j) \leq 1$ and $\beta(i, j) \leq 1$ for all i and $j \leq 3$. In the same paper, it is shown that it is solvable with $p + 1 = 3$ colors if two colors, say colors 1 and 2, are *semi-unary*, i.e.

$\alpha(i,1) \leq 1 \; \forall \; i$ or $\beta(i,1) \leq 1 \; \forall \; i$, and $\alpha(i,2) \leq 1 \; \forall \; i$ or $\beta(i,2) \leq 1 \; \forall \; i$. Our purpose is to consider a case which lies between the general cases $p+1=2$ and $p+1=3$ colors, and to show that this case is still solvable in polynomial time.

3.4 A special case of $RP(m,n,p+1=3)$

$RP3(m,n;q,r)$ will denote the problem with $p+1=3$ colors without restrictions on colors 2 and 3, and where there is a fixed number q of rows and columns in which color 1 may have several occurrences but not more than r.

We shall consider the graph-theoretical formulation of the problem; so we have a complete bipartite graph $K_{m,n}$ and we assume that the vertices i in R (corresponding to the rows of the array) are ordered according to *non increasing* values of $\alpha(i,2)$ and the vertices j in S (corresponding to the columns j of the array) are ordered according to *non decreasing* values of $\beta(j,2)$. If we merge colors 1 and 2, our problem amounts to finding in $K_{m,n}$ a partial graph H where each vertex i in R has degree $d_H(i) = \alpha(i,1) + \alpha(i,2)$ and each vertex j in S has degree $d_H(j) = \beta(j,1) + \beta(j,2)$. Such an H will be called *12-feasible*. In addition H must contain in its edge set $E(H)$ a partial graph M with degree $\alpha(i,1)$ for each vertex i in R and degree $\beta(j,1)$ for each vertex j in S. Such an M will be called *1-feasible*. We shall write $a < b$ if a comes before b in the ordering of the vertices (in R or in S). We shall say that two edges $(a,b),(c,d)$ of M form a *crossing* if $a < c,\ d < b$ and $\alpha(a,1) = \alpha(c,1) = \beta(d,1) = \beta(b,1) = 1$.

Lemma 3.1 *If $RP3(m,n;q,r)$ has a solution, then there is also a solution associated to a 1-feasible M^* which has no crossing.*

Proof: Let us assume that there is a crossing $(a,b),(c,d)$ in a 1-feasible M contained in a 12-feasible H.

Notice that if $(a,d),(c,b)$ are both in $H \backslash M$ (they are not in M by definition of a crossing), then we may replace $(a,b),(c,d)$ in M by $(a,d),(c,b)$; we get a 1-feasible M' (which is still contained in H) and where the number of crossings has decreased. Also if $(a,d),(c,b)$ are both out of H, we may again replace $(a,b),(c,d)$ in M and in H by $(a,d),(c,b)$ and we get a new 12-feasible H' containing a 1-feasible M' where the number of crossings has decreased.

Let us consider now the case where exactly one of the edges $(a,d),(c,b)$ is in $H \backslash M$; w.l.o.g we may assume $(a,d) \in H \backslash M$ and $(c,b) \notin H$. Since the vertices j in S are ordered according to the non decreasing values of $\beta(j,2)$ there must be a vertex e in R such that $(e,b) \in H \backslash M$ and $(e,d) \notin H \backslash M$. Notice that (e,d) cannot be in M since $\alpha(d,1) = 1$. So we have $(e,d) \notin H$. We replace (c,d) and (e,b) in H by (c,b) and (e,d). Now we replace $(a,b),(c,d)$ in M by $(a,d),(c,b)$ and we get a 1-feasible M^* contained in a 12-feasible H^*; furthermore the number of crossings in M^* is smaller than in M. ■

We now define for each instance of a problem $RP3(m,n;q,r)$ the class $\mathcal{C}$ of all 1-feasible partial graphs M which have no crossings. $\mathcal{C}$ can be generated as follows: let Q be the set of lines (rows and columns) which have more than one occurrence of color 1; assume $m \geq n$. We first enumerate all partial graphs $M^{'}$ which satisfy the following:

$$\begin{array}{lllll} \text{a)} & d_{M'}(i) & = & \alpha(i,1) & \text{for all} \quad i \in Q \bigcap R \\ & & \leq & \alpha(i,1) & \text{for all} \quad i \in R \backslash Q \\ \text{b)} & d_{M'}(j) & = & \beta(j,1) & \text{for all} \quad j \in Q \bigcap S \\ & & \leq & \beta(j,1) & \text{for all} \quad j \in S \backslash Q \\ \text{c)} & \text{each edge of } M^{'} & \text{has at least one vertex in } Q. & & \end{array}$$

For each $i \in Q \cap R$ we have at most n^r ways of choosing the assignment of color 1 in row i (i.e., the $\alpha(i,1)$ edges of color 1 adjacent to vertex i); for each $j \in Q \cap S$, we have at most m^r. So globally we have at most $(m^r)^q = m^{r.q}$ possible assignments of color 1 to edges adjacent to vertices in Q.

We will enumerate all these partial assignments $M^{'}$. For each such $M^{'}$, we have to determine the remaining edges to be added in order to obtain a 1-feasible M. Let $V^{'}(M^{'})$ be the set of vertices which are not saturated yet (i.e., where some edges of color 1 should be added). We notice that each vertex in $V^{'}(M^{'})$ has to receive exactly one edge of color 1. If the basic conditions $\sum(\alpha(i,1) \mid i = 1,\ldots,m) = \sum(\beta(j,1) \mid j = 1,\ldots,n)$ are satisfied we will have $\mid R \bigcap V^{'}(M^{'}) \mid = \mid S \bigcap V^{'}(M^{'}) \mid$. So there will exist an assignment of color 1 for edges between $R \bigcap V^{'}(M^{'})$ and $S \bigcap V^{'}(M^{'})$ (because we still have a complete bipartite graph between these two sets of vertices) and there is a unique way of choosing those edges without introducing crossings. So we can obtain a 1-feasible M from each $M^{'}$.

Observe furthermore that any 1-feasible M^* which has no crossing is in $\mathcal{C}$; this can be seen as follows. We remove from M^* all edges which are adjacent to some vertex $i \in R$ (with $\alpha(i,1) > 1$) or $j \in S$ (with $\beta(j,1) > 1$). We also remove those vertices; this subset of edges removed has been considered as an $M^{'}$ in the enumeration process. Now the edges remaining in M^* do not have any crossing; so they are uniquely defined in the complete bipartite subgraph constructed on the remaining vertices. Hence this set M^* has been constructed in $\mathcal{C}$.

We can now state:

Proposition 3.2 *$RP3(m,n;q,r)$ can be solved in polynomial time.*

Proof: We construct each 1-feasible M^* in $\mathcal{C}$; there are at most $m^{r.q}$ such partial graphs. Each one is constructed in $O(m)$. Then for each M^* we do the following: remove M^* from $K_{m,n}$; in the remaining graph G^* examine if there exists a partial graph I with $d_I(i) = \alpha(i,2)$ for each $i \in R$ and $d_I(j) = \beta(j,2)$ for each $j \in R$. If there is such an I, it gives the assignment of color 2 while M^* gives the assignment of color 1. So we have found a solution. If for every M^* in $\mathcal{C}$, no I

can be found, the problem has no solution: according to Lemma 3.1, if there is a solution, there is one where M^* has no crossing. Since $\mathcal{C}$ contains all 1-feasible M^* which have no crossing, we are done. The construction of I is a flow problem; so it can be performed in polynomial time ($O(m^3)$ for instance); see [AMO93]. ■

3.5 Concluding remarks

Removing the assumption on color 1 would give us the general case $R(m, n, p+1=3)$ whose complexity status is open. In the case where $\alpha(i,1) \leq 1$ and $\beta(j,1) \leq 1$ for all i, j, we have some structure in M^* (no crossings) and an easy construction procedure can be derived.

Our algorithm is based on the fact that if the problem admits a solution, then there is one without crossing. This property can not be generalized to the case where color 1 is semi-unary. Figure 3.2 provides a counterexample. As can be seen, there is only one way (up to permutations of the three similar vertices) of ordering the vertices on the left in non increasing values of $\alpha(i,2)$, and one way of ordering the vertices on the right in non decreasing values of $\beta(j,2)$. With this ordering, no solution exists if color 1 is assigned without crossing, while three solutions exist if crossings are allowed.

$\alpha(i,s)$

i \ s	1	2	3
1	1	1	0
2	1	1	0
3	1	1	0
4	1	0	1

$\beta(j,s)$

i \ s	1	2	3
1	3	1	0
2	1	2	1

$\longrightarrow$ Solutions:

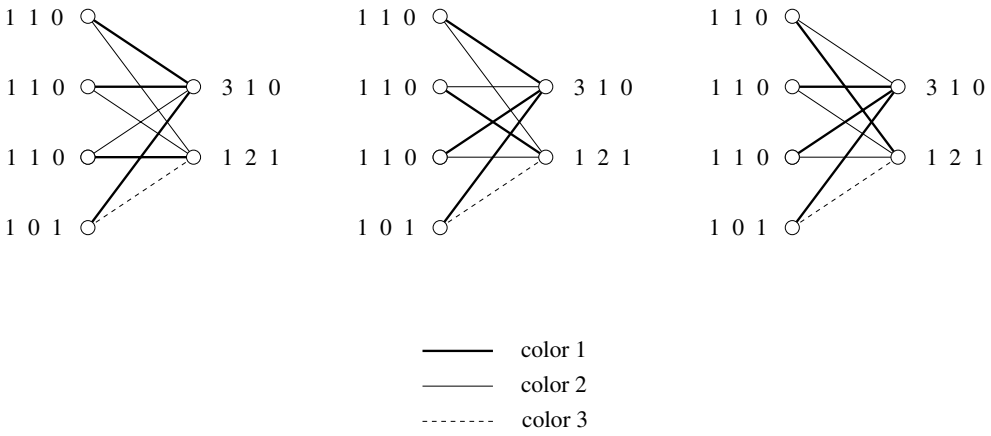

Figure 3.2: An example where a generalized crossing cannot be avoided

Chapter 4

$\mathcal{NP}$-hard hereditary classes for graph coloring

4.1 Introduction

We shall consider here the two classical problems of minimum graph coloring (MCP) and maximum stable set (MSP), as defined in the Preliminaries. This chapter is devoted to a complexity result for many subcases of MCP. It is obtained by combining two polynomial reductions. The first one leads from MSP in general graphs to MSP in some specific hereditary graph classes, and the second leads from MSP in such classes to MCP in hereditary classes of another kind. What follows can also be found in [Sch04].

There are close links between MCP and MSP; Chvátal has shown in [Chv73] that deciding whether the vertex set V of a graph $G = (V, E)$ can be covered by k stable sets is equivalent to finding whether an associated graph $G'(k)$ has a stable set of size $|V|$; $G'(k)$ is obtained by taking k copies of G and forming cliques by joining the k copies of the same vertex.

Conversely, S. Poljak has shown in [Pol74] that deciding whether a triangle-free graph $G = (V, E)$ has a stable set of size k is equivalent to finding whether the vertex set V'' of an associated graph G'' can be covered by a number $f(k)$ of stable sets. In this paper we intend to exploit this construction and combine it with some tranformations of graphs which increase the stability number by one. This will allow us to derive some complexity results which strengthen some properties related to the complexity of coloring problems.

We use the notations for graphs defined in the Preliminaries; further, a $S_{i,j,k}$ is a tree composed of a central vertex from which start at most three pending paths of respective lengths i, j and k. The graph $A_{i,j,k}$ is just $L(S_{i,j,k})$, see Figure 4.1. Given graphs G and G', we denote $G \oplus G'$ the one obtained by putting G side by side with G', without link between them, and kG is $G \oplus (k-1)G$.

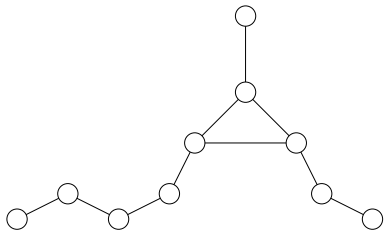

Figure 4.1: The graph $A_{2,3,5} = L(S_{2,3,5})$.

A graph which contains no graph from a (possibly infinite) list $\{H_1, \ldots, H_p\}$ as an induced subgraph is said to be $\{H_1, \ldots, H_p\}$-*free*. Those classes $\mathcal{C}$ of graphs are called *hereditary*, in the sense that any induced subgraph of a graph in $\mathcal{C}$ is also in $\mathcal{C}$.

Graphs $S_{i,j,k}$ are of particular interest with respect to MSP. Indeed, by using a simple graph transformation increasing the stability number by one (see Section 1.1.2), Alekseev has obtained the following.

Theorem 4.1 *[Ale83] MSP is $\mathcal{NP}$-hard in $\{G_1, G_2, \ldots, G_p\}$-free graphs, where p is finite and each G_i has at least one connected component which is not of the form $S_{i,j,k}$.*

In other words, for the class of $\{G_1, G_2, \ldots, G_p\}$-free graphs to have a chance (provided $\mathcal{P} \neq \mathcal{NP}$) to be a polynomial case of MSP, at least one graph among $G_1, \ldots, G_p$ must have all its connected components of the form $S_{i,j,k}$. In what follows, we combine this reduction with the one depicted in [Pol74] to provide a similar result for MCP.

4.2 Toward some new $\mathcal{NP}$-hard cases

The complexity status of MCP in hereditary classes has deserved only little interest, though it is widely studied for MSP (see among other papers, [Ale83, Ale91, Ale04, BH99, GHL03, HLS03, Mos99]). The most complete paper on this topic is probably [KKTW01], written in 2001 by Král, Kratochvíl, Tuza and Woeginger, where the complexity status of MCP is given, for any class defined by a single forbidden induced subgraph.

Theorem 4.2 *[KKTW01] MCP in H-free graphs is in $\mathcal{P}$ if H is an induced subgraph of P_4 or $P_3 \oplus K_1$, and is in $\mathcal{NP}$-hard otherwise.*

In the same paper, the authors study cases where two induced graphs are forbidden and characterize many such cases. The reduction we propose next will permit to prove $\mathcal{NP}$-hardness of some additional cases, with two and more forbidden induced subgraphs.

We first depict in more details the problem reduction mentioned in last Section, which leads from MSP in triangle-free graphs (which is $\mathcal{NP}$-hard) to MCP in complements of line graphs of triangle-free graphs [Pol74]. A *vertex cover* denotes a set $V' \subseteq V$ such that each edge has at least one endpoint in V'. It follows that if V' is a vertex cover, then $V \backslash V'$ is a stable set. By MVC we denote the problem of finding a vertex cover of minimum cardinality. Further, by $\overline{\text{MCP}}$ we denote the problem of covering the set of vertices of a graph by a minimum number of cliques, which may alternatively be viewed as the problem of finding a minimum coloring in the complementary graph.

From Alekseev's work described in Section 1.1.2, and leading to Theorem 4.1, we know that MSP in general graphs can be reduced to MSP in triangle-free graphs. Let $G = (V, E)$ be a triangle-free graph and consider its line graph $L(G)$. Of course the set of edges incident to a vertex x of G form a clique in $L(G)$. Three adjacent vertices in $L(G)$ correspond either to a triangle in G, or to three edges incident to some vertex of G. So if G is triangle-free, there will be a one-to-one correspondence between maximal cliques in $L(G)$ and bundles of edges in G. Since $\overline{\text{MCP}}$ amounts to covering the vertices with a minimum number of maximal cliques, we have the following reductions.

$$\begin{gathered}\text{MSP} \propto \text{MSP in triangle-free graphs} \propto \text{MVC in triangle-free graphs} \\ \propto \overline{\text{MCP}} \text{ in line graphs of triangle-free graphs} \\ \propto \text{MCP in complements of line graphs of triangle-free graphs}\end{gathered}$$

We now show that Alekseev's result can be combined with this last one. Denote by $L^{-1}(H)$ the set of graphs having no isolated vertex, and whose line graph is H.

$$L^{-1}(H) = \{G = (V, E) : L(G) = H, N(v) \neq \emptyset \; \forall \; v \in V\}$$

For instance, $L^{-1}(P_4) = \{P_5\}$, $L^{-1}(K_3) = \{K_{1,3}, K_3\}$ and $L^{-1}(K_{1,3}) = \emptyset$, since $K_{1,3}$ is not a line graph. We extend this definition to several graphs the following way: $L^{-1}(\{H_1, \ldots, H_p\}) = L^{-1}(H_1) \cup \ldots \cup L^{-1}(H_p)$. Noticing that any partial subgraph G' of a graph G gives rise to an induced subgraph $L(G')$ of $L(G)$, the following fact is straightforward.

Fact 4.3 *$L(G)$ is H-free if and only if G contains no graph in $L^{-1}(H)$ as a partial subgraph.*

We now define the closure $Cl(G)$ of G as the set of all graphs obtained from G by possibly adding edges between non adjacent vertices of G. In Figure 4.2, graphs in $Cl(P4)$ are displayed. We also extend this definition to the closure of several graphs: $Cl(\{G_1, \ldots, G_p\}) = Cl(G_1) \cup \ldots \cup Cl(G_p)$.

Noticing that a graph does not contain G as a partial subgraph if and only if it is $Cl(G)$-free, we have

Fact 4.4 *$L(G)$ is H-free if and only if G is $Cl(L^{-1}(H))$-free.*

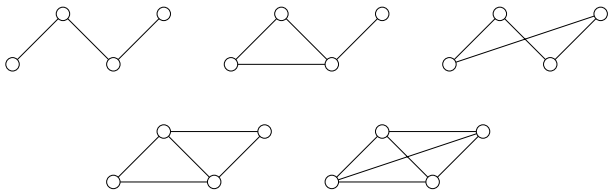

Figure 4.2: $Cl(P4)$

We can now prove the following lemma.

Lemma 4.5 *$\overline{MCP}$ is $\mathcal{NP}$-hard in $\{H_1, \ldots, H_p\}$-free graphs, where p is finite and no H_i has all its connected components of the form $A_{i,j,k}$.*

Proof: We first prove that the set of graphs $Cl(L^{-1}(\{H_1, \ldots, H_p\}))$ contains no graph with all its connected components of the form $S_{i,j,k}$. Since a partial subgraph of such a graph also has all its connected components of the form $S_{i,j,k}$, we need only consider graphs in $L^{-1}(\{H_1, \ldots, H_p\})$. Clearly, for any i, the connected components of H_i are the line graphs of the connected components of any graph in $L^{-1}(H_i)$. Further, by the definition of $L^{-1}(H_i)$, no graph in $L^{-1}(H_i)$ can have all its connected components of the form $S_{i,j,k}$, otherwise H_i would have all its connected components of the form $A_{i,j,k}$.
So if one can solve $\overline{\text{MCP}}$ in $\{H_1, \ldots, H_p\}$-free graphs with a polynomial time algorithm, one can use it to solve MSP in $\{Cl(L^{-1}(\{H_1, \ldots, H_p\})), K_3\}$-free graphs, after having transformed the instance graph G to $L(G)$. Now $L^{-1}(H_i)$ is a finite set (since it contains only graphs with $|V(H_i)|$ edges and no isolated vertex), hence so are $L^{-1}(\{H_1, \ldots, H_p\})$ and $Cl(L^{-1}(\{H_1, \ldots, H_p\}))$. MSP is therefore $\mathcal{NP}$-hard in $\{Cl(L^{-1}(\{H_1, \ldots, H_p\})), K_3\}$-free graphs, which permits to conclude. ■

As we now have a result of the same form as Theorem 4.1 for $\overline{\text{MCP}}$ we only need to switch to complementary graphs to express it in terms of MCP.

Theorem 4.6 *MCP is $\mathcal{NP}$-hard in $\{H_1, \ldots, H_p\}$-free graphs, where p is finite and each $\overline{H_i}$ has at least one connected components which is not of the form $A_{i,j,k}$.* ■

For the case $p = 2$, most $\mathcal{NP}$-hard cases obtained from Theorem 4.6 are already obtained in [KKTW01]. However, the followings are new ones.

- $\{K_3 \oplus K_1, H\}$-free graphs, where H is a forest of at least 5 vertices and of maximum degree at least three (corresponds to *Problem 2* in [KKTW01]);
- $\{C_4 \oplus K_1, H\}$-free graphs and $\{\overline{C_k}, H\}$-free graphs ($k \geq 6$), where H contains a partial subgraph of $2K_2$ as an induced subgraph (corresponds to *Problem 3* in [KKTW01]);

For larger values of p, any finite combination of graphs of the form described in Theorem 4.6 provides an $\mathcal{NP}$-hard case. If we list those which are minimal (with respect to induced subgraphs) until 5 vertices, we find that MCP is $\mathcal{NP}$-hard in $\{4K_1, 2K_1 \oplus K_2, 2K_2, K_1 \oplus K_3, C_5, C_4 \oplus K_1\}$-free graphs, which is a quite restrictive class. Indeed, the complement of each graph in the above list has one connected component which is not of the form $A_{i,j,k}$. For instance, $\overline{2K_1 \oplus K_2}$ is a K_4 without an edge. Since it contains two triangles, it is not an $A_{i,j,k}$. Another interesting $\mathcal{NP}$-hard case is the class of $\{C_5, P_5\} \bigcup (\bigcup_{i=4}^{k} \overline{C_i})$-free graphs: it is contained in the class of $(\bigcup_{i\geq 2} C_{2i+1}) \bigcup (\bigcup_{i=2}^{k} \overline{C_{2i+1}})$-free graphs, where MCP is hence $\mathcal{NP}$-hard. Notice that the class of perfect graphs, where MCP is polynomially solvable, is the class of $(\bigcup_{i\geq 2} C_{2i+1}) \bigcup (\bigcup_{i\geq 2} \overline{C_{2i+1}})$-free graphs (see [Ber70],[CRST], [GLS88]).

4.2.1 Concluding remarks

Our theorem is of the same form as Theorem 4.1 for MSP, but while this last one covers most $\mathcal{NP}$-hard cases (defined by hereditary properties) of MSP, it is not the case for our result with MCP. For instance we know by Theorem 4.2, that MCP is $\mathcal{NP}$-hard in K_3-free graphs. This case is not covered by Theorem 4.6, since each connected component of a $\overline{K_3}$ is an isolated vertex, i.e. an $A_{1,0,0}$. It is nevertheless a strong result in the sense that any class defined by a set $\{H_1, \ldots, H_p\}$ of forbidden induced subgraphs none of which has all its connected components of the form $\overline{A_{i,j,k}}$ can be further restricted by forbidding another graph H_{p+1} of that type.
As a final remark, if we now compare the state of the art in the investigation of polynomial and $\mathcal{NP}$-hard hereditary classes for MCP and for MSP, we can see that MCP is slighty more difficult than MSP in the sense that there are many classes $\mathcal{C}$ of graphs (for instance line graphs or O_4-free graphs) where MSP is polynomially solvable in $\mathcal{C}$, and where MCP is $\mathcal{NP}$-hard. In fact, to our knowledge, there is no class of graphs where the converse happens. It would be interesting to find out whether such classes exist.

Chapter 5

Column Generation for Graph Coloring

In this last chapter, we are interested in the graph (vertex-)coloring problem (MCP) in general graphs. Recall that this problem is $\mathcal{NP}$-hard, even when restricted for instance to line graphs, complements of line graphs, planar graphs (graphs which can be drawn on the plane without edge crossings) or graphs G with $\alpha(G) \leq 3$.

We study here a linear programming approach of MCP. We first review different integer programming formulations of MCP and how they are related, then we give some properties of the underlying polyhedra and finally propose an improvement to the original algorithm, which substantially increases the performance on some instances. We use the following definitions and notations. For an integer linear program (P), $Conv(P)$ is the convex hull of all (integer) solutions of (P). The dimension $Dim(P)$ of (P) is the number of linearly independent points in $Conv(P)$. The polyhedra $Conv(P)$ is defined by a unique minimal set of inequalities, each of which are called $facets$ of $Conv(P)$. A necessary and sufficient condition for an inequality to define a facet is to be satisfied with equality by $Dim(P)$ affinely independent solutions of (P). For further concepts not defined here about linear programming, we refer to [Sch86].

5.1 Integer programming formulations of MCP and relationships

We provide in this section four integer programming formulations of MCP, and show how they are related.

5.1.1 Standard formulation

As usual, let $G = (V, E)$ be a graph, $n = |V|$, $m = |E|$ and $\overline{\chi}$ an upper bound on the chromatic number $\chi(G)$ of G. Consider the following integer linear programming formulation (E) (E stands for 'edge constraints') of the graph coloring problem.

$$\min \quad \sum_{j=1}^{\overline{\chi}} y_j \tag{5.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{\overline{\chi}} x_{ij} = 1 \quad \forall\, i \in V \tag{5.2}$$

$$x_{ij} + x_{i'j} - y_j \leq 0 \quad \forall\, (i, i') \in E,\ j \in \{1, \ldots, \overline{\chi}\} \tag{5.3}$$

$$x_{ij}, y_j \in \{0, 1\} \quad \forall\, i \in V,\ j \in \{1, \ldots, \overline{\chi}\} \tag{5.4}$$

Table 5.1: (E)-formulation of the graph coloring problem

In this formulation, variable x_{ij} has value one if vertex i has color j, and 0 otherwise; variable y_j has value 1 if color j is used to color the graph and 0 otherwise. The objective expresses that a minimum number of colors has to be used while constraints (5.2) ensure that each vertex is colored and contraints (5.3) force two adjacent vertices to get different used colors.

The linear relaxation (E^l) of this formulation, obtained by replacing the boolean constraints (5.4) by non negativity constraints, permits to find a lower bound on the optimal value, which rounded up is also a lower bound on the chromatic number of G. It may then be used to design an exact algorithm, whose efficiency will essentially depend on this lower bound's sharpness. Consider the fractional solution of (E^l) given by

$$y_j = \begin{cases} 1 & j = 1, 2 \\ 0 & j = 3, \ldots, \overline{\chi} \end{cases} \quad \text{and} \quad x_{ij} = \begin{cases} \frac{1}{2} & j = 1, 2 \\ 0 & j = 3, \ldots, \overline{\chi} \end{cases}$$

It clearly satisfies constraints (5.2) and (5.3), and the corresponding objective function value is equal to 2. But since the chromatic number of a graph may be arbitrarily large (it is equal to n if G is a clique), the bound obtained with (E^l) is extremely bad.

The four authors of [CMaPZ02] have studied the polyhedral structure of the underlying polytope and showed that one can substantially improve this bound by adding cutting planes to the formulation. The branch-and-cut algorithm they designed is able to color, for instance, any random graph (see Section 5.4.3) with up to 60 vertices in less than one minute, which is much better than solving the boolean constrained formulation with a general mixed integer programming (MIP) solver (for instance the one implemented in CPLEX).

5.1.2 Dantzig-Wolfe decomposition

There is another type of linear programming formulations for MCP, which provide much better lower bounds, but at the cost of a much larger problem size. Those formulations have a natural interpretation, but they seem less known than formulation (E). This is probably due to the fact that they do not have a number of variables and constraints polynomial in the size of the instance graph. Further, despite the apparent dissimilarity of both types of formulations, there is a succession of transformations which brings the (E)-formulation to those with larger size. The main step in this problem translation is the Dantzig-Wolfe decomposition principle (see [Las70]), which permits the replacement of a linear programming formulation with specific structure by another equivalent one, with many more variables but less constraints. In [Joh89], it is used for cutting stock, crew scheduling and clustering problems. As mentioned in [MT96], this principle can be applied to (E), since it is of the appropriate form. In the remainder of this section, this is done explicitly, since to our knowledge it has not been exposed in full details earlier.

A matrix is said (see [Las70]) to have a *p-block angular* structure, if one can permute its row and column sets so that it becomes like in Table 5.2, with any element outside blocks A_i and B_i $i = 1, \ldots, p$ being zero.

$$\begin{pmatrix} A_1 & A_2 & \cdots & A_p \\ B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_p \end{pmatrix}$$

Table 5.2: A p-block angular matrix.

If we put the rows corresponding to constraints (5.2) on the top, and if we give the variables the order

$$y_1,\ x_{11},\ x_{21},\ \ldots,\ x_{n1},\ y_2,\ x_{12},\ \ldots,\ x_{n2},\ \ldots,\ \ldots,\ y_{\overline{\chi}},\ x_{1\overline{\chi}},\ \ldots,\ x_{n\overline{\chi}},$$

the matrix obtained of formulation (E) appears in the same form as Table 5.2. Table 5.3 shows this structure with the same notations as in Table 5.2 (where $\mathbf{1}$ denotes a vector of 1's of the appropriate size, $\mathbf{0}$ a vector of 0's), and after having grouped the variables in $\overline{\chi}$ parts of the form $\mathbf{z_j} = (y_j, x_{1j}, x_{2j}, \ldots, x_{nj})^T$ $(j = 1, \ldots, \overline{\chi})$. In Table 5.4, we can see the detailed structure of the whole matrix.

Problem (E) can be viewed as follows. Among all feasible solutions of (5.7) and (5.8), find those which satisfy (5.6) and minimize (5.5). Let us define $\mathcal{U}_j = \{\mathbf{u_j^k} : k = 0, \ldots, |\mathcal{U}_j| - 1 =: K_j\}$ (this numbering will be justified later) the set of solutions of (5.7) and (5.8) for each j in $\{1, \ldots, \overline{\chi}\}$,

$$\min \quad \sum_{j=1}^{\overline{\chi}} y_j \tag{5.5}$$

$$\text{s.t.} \quad \sum_{j=1}^{\overline{\chi}} A_j \mathbf{z_j} \quad = \quad \mathbf{1} \tag{5.6}$$

$$B_j \mathbf{z_j} \quad \leq \quad \mathbf{0} \quad \forall\, j \in \{1, \ldots, \overline{\chi}\} \tag{5.7}$$

$$\mathbf{z_j} \quad \in \quad \{0,1\}^{n+1} \quad \forall\, j \in \{1, \ldots, \overline{\chi}\} \tag{5.8}$$

Table 5.3: (E)-formulation of MCP.

y_1	x_{11}	x_{21}	$\cdots$	x_{n1}	y_2	x_{12}	x_{22}	$\cdots$	x_{n2}	$\cdots$	$y_{\overline{\chi}}$	$x_{1\overline{\chi}}$	$x_{2\overline{\chi}}$	$\cdots$	$x_{n\overline{\chi}}$
0	1	0	$\cdots$	0	0	1	0	$\cdots$	0		0	1	0	$\cdots$	0
0	0	1	$\ddots$	$\vdots$	0	0	1	$\ddots$	$\vdots$	$\ldots$	0	0	1	$\ddots$	$\vdots$
$\vdots$	$\vdots$	$\ddots$	$\ddots$	0	$\vdots$	$\vdots$	$\ddots$	$\ddots$	0		$\vdots$	$\vdots$	$\ddots$	$\ddots$	0
0	0	$\cdots$	0	1	0	0	$\cdots$	0	1		0	0	$\cdots$	0	1
-1	$\cdots$	1	$\cdots$	1											
-1															
$\vdots$		(*constraints* (5.3))					0			$\ldots$			0		
-1	$\cdots$	1	1	$\cdots$											
					-1	$\cdots$	1	$\cdots$	1						
					-1										
	0				$\vdots$		(*constraints* (5.3))			$\ldots$		0			
					-1	$\cdots$	1	1	$\cdots$						
		$\vdots$					$\vdots$			$\ddots$			$\vdots$		
											-1	$\cdots$	1	$\cdots$	1
											-1				
	0					0				$\ldots$	$\vdots$		(*constraints* (5.3))		
											-1	$\cdots$	1	1	$\cdots$

Table 5.4: Matrix of (E) in standard form.

and replace $\mathbf{z_j}$ by $\sum_{k=0}^{K_j} \lambda_j^k \mathbf{u_j^k}$ and add the constraints $\sum_{k=0}^{K_j} \lambda_j^k = 1$ to ensure that exactly one integer solution from each set $\mathcal{U}_j$ is used[1]. This leads to the formulation of Table 5.5, with set of variables $\{\lambda_j^k : k = 0, \ldots, K_j \text{ and } j = 1, \ldots, \overline{\chi}\}$ ($(\mathbf{x})_i$ denotes the i$^{\text{th}}$ component of vector $\mathbf{x}$).

$$\min \quad \sum_{j=1}^{\overline{\chi}} \sum_{k=0}^{K_j} \lambda_j^k (\mathbf{u_j^k})_1 \tag{5.9}$$

$$\text{s.t.} \quad \sum_{j=1}^{\overline{\chi}} \sum_{k=0}^{K_j} (A_j \mathbf{u_j^k}) \lambda_j^k = \mathbf{1} \tag{5.10}$$

$$\sum_{k=0}^{K_j} \lambda_j^k = 1 \quad \forall\, j \in \{1, \ldots, \overline{\chi}\} \tag{5.11}$$

$$\lambda_j^k \in \{0,1\} \quad \forall\, j \in \{1, \ldots, \overline{\chi}\},\ k \in \{0, \ldots, K\} \tag{5.12}$$

Table 5.5: Intermediate formulation.

We can simplify this formulation with the following observations. First, since for any j the sets of constraints (5.7) in Table 5.3 correspond to the edges of the same graph, the B_j matrices are all equal and consequently so are the sets $\mathcal{U}_j$. We may then write $\mathbf{u^k}$ instead of $\mathbf{u_1^k} = \mathbf{u_2^k} = \ldots = \mathbf{u_{\overline{\chi}}^k}$, and K instead of K_j.
The $B_1 = \ldots = B_{\overline{\chi}}$ matrices are all of the form $[\mathbf{-1}, M]$, M being the edge-vertex incidence matrix of G, so all $\mathbf{u^k}$ except one are of the form $\begin{bmatrix} 1 \\ \mathbf{s^k} \end{bmatrix}$, $\mathbf{s^k}$ being the incidence vector of a stable set of G. The only solution, called $\mathbf{u^0}$, which is not of this form is the zero vector. Hence λ_j^0 appears neither in the objective (5.9), nor in constraints (5.10). So we may remove constraints (5.11) from the formulation by replacing λ_j^0 with $1 - \sum_{k=1}^{K} \lambda_j^k$. Since λ_j^0 only occurs in constraints (5.11) and (5.12), we only need to add the constraints

$$1 - \sum_{k=1}^{K} \lambda_j^k \geq 0 \quad \Leftrightarrow \quad \sum_{k=1}^{K} \lambda_j^k \leq 1$$

for $j \in \{1, \ldots, \overline{\chi}\}$ to ensure that $\lambda_j^k \in \{0,1\}$ for $j \in \{1, \ldots, \overline{\chi}\}$. Matrices A_j are all of the form $[\mathbf{0}, I]$, I being the identity matrix, so we can replace $A_j \mathbf{u^k}$ by $\mathbf{s^k}$. The new formulation is displayed in Table 5.6.

Let now λ' be an optimal solution of the problem obtained from Table 5.6, by removing constraints (5.15). Since the resulting problem is a relaxation of the previous one, the optimal objective value is at most $\overline{\chi}$. In other words, in an optimal solution, there are at most $\overline{\chi}$ components of λ' which have value 1, while the others have value 0. If λ' violates some constraints (5.15), the corresponding "coloring" will have two stable sets with the same color. Since there

[1] Those constraints correspond to the convexity constraints in the continuous case.

$$\min \quad \sum_{k=1}^{K} \sum_{j=1}^{\overline{\chi}} \lambda_j^k \tag{5.13}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \mathbf{s^k} \sum_{j=1}^{\overline{\chi}} \lambda_j^k \quad = \mathbf{1} \tag{5.14}$$

$$\sum_{k=1}^{K} \lambda_j^k \quad \leq 1 \qquad \forall\, j \in \{1, \ldots, \overline{\chi}\} \tag{5.15}$$

$$\lambda_j^k \in \{0,1\} \qquad \forall\, j \in \{1, \ldots, \overline{\chi}\}, k \in \{1, \ldots, K\} \tag{5.16}$$

Table 5.6: Intermediate formulation.

are sufficiently available colors, we can always replace them by other colors, which amounts to changing the components with value one, so that constraints (5.15) become satisfied, and the objective remains unchanged. We conclude that those constraints are redundant and can be removed from the formulation.
Replace $\sum_{j=1}^{\overline{\chi}} \lambda_j^k$ by a single variable λ^k. Similarly as above, we can restrict this variable to $\{0,1\}$, since in a coloring each stable set appears at most once. We finally obtain the (Part)-formulation ("Part" stands for partitioning, constraints (5.18) being equalities) depicted in Table 5.7. The lower bound obtained by solving the linear relaxation of this last formulation is much

$$\min \quad \sum_{k=1}^{K} \lambda^k \tag{5.17}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \lambda^k \mathbf{s^k} \quad = \mathbf{1} \tag{5.18}$$

$$\lambda^k \in \{0,1\} \qquad \forall\, k \in \{1, \ldots, K\} \tag{5.19}$$

Table 5.7: (Part)-formulation.

sharper than the one provided by the (E)-formulation. The main drawback is that the (Part)-formulation has a number of variables equal to the number of stable sets in G, which in general grows exponentially with the size of G. However, as will be seen in the sequel, the set of stable sets has some structure which can be exploited in the column generation, in order to drastically speed up the resolution of the linear relaxation.

5.1.3 Large size formulations and comparisons

We start here with the last formulation of MCP found in previous section and express it in terms of graphs the following way.

$$\min \quad \sum_{S \in \mathcal{S}} x_S \tag{5.20}$$

$$s.t. \quad \sum_{S \ni v} x_S = 1 \quad \forall v \in V \qquad (MCP(Part)) \tag{5.21}$$

$$x_S \in \{0,1\} \quad \forall S \in \mathcal{S} \tag{5.22}$$

Here $\mathcal{S}$ is the set of all stable sets of G, and $x_S = 1$ if the stable set S corresponds to a color class. Notice that by summing up all equalities, we have the simple relation $\sum_{S \in \mathcal{S}} |S| x_S = |V|$, which also holds for the linear relaxation. Denote $\mathcal{S}_2 = \{S \in \mathcal{S} : |S| \geq 2\}$. One can make the convex hull of feasible solutions to this formulation full-dimensional by replacing variables x_S with $|S| = |\{v\}| = 1$, by $1 - \sum_{\{S \in \mathcal{S}_2 : v \in S\}} x_S$. This substitution ensures the satisfaction of constraints (5.21), but the fact that $1 - \sum_{S \in \mathcal{S}_2} x_S = x_{\{v\}} \geq 0 \ \forall \ v$ brings the inequalities $\sum_{S \in \mathcal{S}_2} x_S \leq 1$. After having transformed the objective function accordingly, we get the following formulation.

$$\max \quad \sum_{S \in \mathcal{S}_2} (|S| - 1) x_S \tag{5.23}$$

$$s.t. \quad \sum_{S \ni v, S \in \mathcal{S}_2} x_S \leq 1 \quad \forall v \in V \qquad (MCP(P)) \tag{5.24}$$

$$x_S \in \{0,1\} \qquad \forall S \in \mathcal{S}_2 \tag{5.25}$$

Another formulation can be obtained by observing that one does not need to partition V, but only to cover it with a minimum number of stable sets. Moreover, stable sets which are not inclusionwise maximal can be removed from the formulation, since in a covering they can always be replaced by maximal sets. Let $\mathcal{S}_{max}$ denote the set of maximal stable sets; we have the following set covering formulation.

$$\min \quad \sum_{S \in \mathcal{S}_{max}} x_S \tag{5.26}$$

$$s.t. \quad \sum_{S \ni v} x_S \geq 1 \quad \forall v \in V \qquad (MCP(C)) \tag{5.27}$$

$$x_S \in \{0,1\} \quad \forall S \in \mathcal{S}_{max} \tag{5.28}$$

5.1.4 Equivalence of the linear relaxations of the large size formulations

Consider the linear relaxations $MCP^l(Part), MCP^l(P)$ and $MCP^l(C)$ of the above formulations, denote by x^{*Part}, x^{*P} and x^{*C} their optimal solutions and z^{*Part}, z^{*P} and z^{*C} their optimal values, respectively. The following proposition shows that the bounds obtained by the three linear relaxations are equivalent.

Proposition 5.1 $z^{*C} = z^{*Part} = |V| - z^{*P}$.

Proof: The following equalities can be deduced from the above development leading from $MCP(Part)$ to $MCP(P)$.

$$z^{*Part} = \sum_{S \in \mathcal{S}} x_S^{*Part} = \sum_{S \in \mathcal{S}} x_S^{*Part} + (|V| - \sum_{S \in \mathcal{S}} |S| x_S^{*Part})$$

$$= |V| - \sum_{S \in \mathcal{S}} (|S| - 1) x_S^{*Part} = |V| - \sum_{S \in \mathcal{S}_2} (|S| - 1) x_S^{*P} = |V| - z^{*P}.$$

It remains to show that $z^{*C} = z^{*Part}$. From a feasible solution x^{Part} of $MCP^l(Part)$, one can always construct a feasible solution of same value x^C of $MCP^l(C)$ by applying the following steps.

1. set $x_S^C = 0 \; \forall \; S \in \mathcal{S}_{max}$;
2. for each $S \in \mathcal{S}$ such that $x_S^{Part} > 0$, choose a set $S' \in \mathcal{S}_{max}$ containing S, and set $x_{S'}^C = x_{S'}^C + x_S^{Part}$.

Since the coefficients in the objective function are all equal to one in both formulations, $z^{*C} \leq z^{*Part}$.

Now given a solution x^C of $MCP^l(C)$, apply the following steps to obtain a solution x^{Part} of $MCP^l(Part)$, with objective value equal to the value of x^C.

1. set $x_S^{Part} = x_S^C$ if $S \in \mathcal{S}_{max}$ and $x_S^{Part} = 0$ if $S \in (\mathcal{S} \backslash \mathcal{S}_{max})$;
2. for each $v \in V$ define $Cov(v) = \sum_{S \ni v} x_S^{Part}$;
3. for each $S \in \mathcal{S}_{max}$ such that $x_S^C > 0$, do:
   (a) partition S into classes $C_1, \ldots, C_q$, where v_1 and v_2 belong to the same class C_i if $Cov(v_1) = Cov(v_2) =: Cov(C_i)$, and order them in decreasing order of Cov;
   (b) set $S' = S$
   (c) for $i = 1$ to q, do:
      i. set $\delta = \min(Cov(C_i) - 1, x_{S'}^{Part})$;
      ii. set $x_{S'}^{Part} = x_{S'}^{Part} - \delta$;
      iii. set $S' = S' \backslash C_i$;
      iv. set $x_{S'}^{Part} = \delta$.

At the beginning, x^{Part} has the same value as x^C, but may be an unfeasible solution of $MCP^l(Part)$, since some inequalities of $MCP^l(C)$ may not be satisfied with equality. Each

time x^{Part} is modified (steps 3(c)ii and 3(c)iv) the values of Cov change, and the specific choices for δ ensure that they finally all become equal to one. Further, the objective value $\sum_{S \in \mathcal{S}} x_S^{Part}$ is the same at the beginning and at the end of each loop in 3c. Hence we obtain a feasible solution x^{Part} of $MCP^l(Part)$ with same value as x^C, which shows that $z^{*C} \geq z^{*Part}$. ■

The optimal solutions x^{*C}, x^{*Part} and x^{*P} correspond to colorings of G with fractional colors, such that each pair of adjacent vertices are colored with disjoint sets of colors, and the sum of the fractions of colors assigned to a given vertex is at least 1. The values $z^{*C} = z^{*Part} = |V| - z^{*P}$ give a lower bound of good quality on $\chi(G)$, better known as the *fractional chromatic number* $\chi_{frac}(G)$ of G. See [Sch97, LPU95] for some of its properties.

Proposition 5.1 can be somewhat generalized. Define $C(\mathcal{P}, Q)$, where $\mathcal{P}$ is a collection of subsets of a ground set Q, as the problem of covering Q with a minimum number of sets taken in $\mathcal{P}$. With this notation, we may write $MCP(C) = C(\mathcal{S}_{max}, V)$. An *independence system* $\mathcal{P}$ on a ground set Q is a set of subsets of Q such that $S_1 \subseteq S_2$ and $S_2 \in \mathcal{P}$ imply $S_1 \in \mathcal{P}$. Since a subset of a stable set is also a stable set, $\mathcal{S}$ is an independence system on the ground set V. The proof of Proposition 5.1 only uses the fact that $\mathcal{S}$ is the independence system whose inclusionwise maximal elements are $\mathcal{S}_{max}$, so we have the following.

Corollary 5.2 *Let $\mathcal{P}$ be an independence system on the ground set Q, $\mathcal{P}_{max}$ its set of maximal sets, z^{*P} and z^{*max} the optimal values of the linear relaxations of $C(\mathcal{P}, Q)$ and $C(\mathcal{P}_{max}, Q)$, respectively. Then $z^{*P} = z^{*max}$.*

5.2 Polyhedral results on the set covering formulation

In this section, we consider formulation $C(\mathcal{S}_{max}, V)$. The dimension of $Conv(C(\mathcal{S}_{max}, V))$ is $|\mathcal{S}_{max}|$ if and only if $\mathcal{S}_{max} \backslash \{S\}$ is a cover of V, for any $S \in \mathcal{S}_{max}$. This amounts to saying that each vertex v of V belongs to at least two maximal stable sets of G, which is achieved if and only if $V \backslash N(v)$ is not a stable set. In the opposite case, we can remove $S = V \backslash N(v)$ from G and solve $C(\mathcal{S}_{max} \backslash \{S\}, N(v))$. The optimal coloring of G would then be obtained by adding S with a new color to the optimal coloring obtained for $N(v)$. We will thus assume in this section that each vertex belongs to at least two maximal stable sets, and hence that $Conv(C(\mathcal{S}_{max}, V))$ is full-dimensional.

In [CS89, Sas89], G. Cornuejols and A. Sassano define the useful concept of *bipartite incidence graph* for the study of set covering polytopes. In the special case of MCP, the bipartite incidence graph $B(\mathcal{S}_{max}, V, E)$ is the bipartite graph with node sets $\mathcal{S}_{max}$, V and with edge set $E = \{(S, v) \in \mathcal{S}_{max} \times V : v \in S\}$. A subset S' of $\mathcal{S}_{max}$ such that each node of V has at least one neighbor in S' will be called a *cover* of V. It is easy to see that the covers of V are exactly the solutions of $C(\mathcal{S}_{max}, V)$. The cardinality of a minimum cover of V is called the *covering number* of V and is denoted by $\beta(V)$.

The following result was proved in [CS89] for the general set covering polytope. For the ease of exposition, we present it in terms of the bipartite graph $B(\mathcal{S}_{max}, V, E)$ associated to formulation $C(\mathcal{S}_{max}, V)$.

Proposition 5.3 *Let $\mathcal{S}^1 \subseteq \mathcal{S}_{max}$ and $V^1 = \{v \in V : N_B(v) \subseteq \mathcal{S}^1\}$. Assume that $\sum_{S \in \mathcal{S}^1} x_S \geq \beta(V^1)$ defines a facet of $Conv(C(\mathcal{S}^1, V^1))$. Then it defines a facet of $Conv(C(\mathcal{S}_{max}, V))$ if and only if for every $S \notin \mathcal{S}^1$,*

$$\beta(V^1 \cup V^2 \cup V^3) = \beta(V^1 \cup V^3)$$

where $V^2 = \{v \in V : N_B(v) \cap \mathcal{S}^1 \neq \emptyset$ and $S \in N_B(v)\}$
and $V^3 = \{v \in V : N_B(v) = \{S\}\}$.

In the next section this result is used to give a necessary and sufficient condition for an inequality of the form $\sum_{S \in \mathcal{S}'} x_S \geq 1$ ($\mathcal{S}' \subseteq \mathcal{S}_{max}$) to define a facet.

5.2.1 All facets with right hand side equal to 1

In this section, to avoid confusion, we denote by $N_G(v)$ the neighborhood of a vertex v if we refer to the graph G and $N_B(v)$ if we refer to the bipartite incidence graph B. For a pair $v \in V$ and $w \in V$ of vertices, we say that v *dominates* w if $N_G(w) \subset N_G(v)$ (it follows that v and w are not adjacent). Notice that in this case, $N_B(v) \subset N_B(w)$. We also denote by $\mathcal{S}_v$ the set of all maximal (inclusionwise) stable sets containing v.

Proposition 5.4 *Let v be a vertex of V. Then the inequality*

$$\sum_{S \in \mathcal{S}_v} x_S \geq 1$$

defines a facet of $Conv(C(\mathcal{S}, V))$ if and only if vertex v is not dominated.

Proof: Let S be a maximal stable set not containing vertex v and consider the bipartite incidence subgraph $B'(\mathcal{S}_v \cup \{S\}, V', E')$ where $V' = \{w \in V : N_B(w) \subseteq \mathcal{S}_v \cup \{S\}\}$ and $(S', u) \in E'$ iff $u \in S'$. Since v is not dominated, for every $w \in V'$, $N_{B'}(w) = N_{B'}(v) = \mathcal{S}_v$ or $S \in N_B(w)$ (i.e. $N_{B'}(w)$ cannot be strictly included in $N_{B'}(v)$). We can thus partition V' into three subsets: $V^1 = \{w \in V' : N_B(w) = N_B(v)\}$, $V^2 = \{w \in V' : N_B(w) \subseteq \mathcal{S}_v$ and $S \in N_{B'}(w)\}$ and $V^3 = \{w \in V' : N_B(w) = \{S\}\}$, see Figure 5.1. Note that $v \in V^1$.

Clearly, $\sum_{S \in \mathcal{S}_v} x_S \geq 1$ defines a facet of $Conv(C(\mathcal{S}_v, V^1))$. In order to apply Proposition 5.3, we have to show that $\beta(V^1 \cup V^2 \cup V^3) = \beta(V^1 \cup V^3)$. Consider $w \in V^2$. Since $N_{B'}(w) \cap \mathcal{S}_v \neq \emptyset$, v and w occur in a common stable set and hence are not adjacent in G. Moreover, w cannot be adjacent to another vertex u of V^2, since both u and w belong to the stable set S. Thus $V^2 \cup \{v\}$ is stable in G and there must be a stable set in $\mathcal{S}_v$ containing $V^2 \cup \{v\}$, which proves

that $\beta(\{v\} \cup V^2) = 1$. Further, since $N_{B'}(w) = N_{B'}(v)$ for all $w \in V^1$, we also have that $\beta(V^1 \cup V^2) = 1$. Now if $V^3 = \emptyset$, then $\beta(V^1 \cup V^2 \cup V^3) = \beta(V^1 \cup V^2) = 1$, and if $V^3 \neq \emptyset$, $\beta(V^1 \cup V^2 \cup V^3) = \beta(V^1 \cup V^3) = 2$. In both cases, the condition of Proposition 5.3 is fullfilled. ■

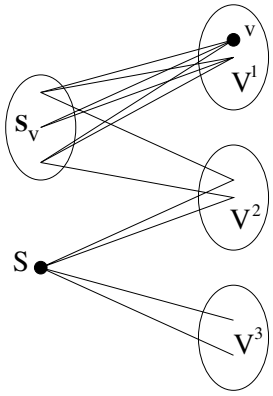

Figure 5.1: The bipartite incidence graph of the proof of Proposition 5.4.

A dominated vertex v in G is irrelevant for $\chi(G)$, since in any optimal coloring, it can have the same color as the dominating vertex. So one can remove all dominated vertices from G without decreasing $\chi(G)$. If those reductions are done, all inequalities (5.27) are facet defining. In Section 5.4.3 are shown some computational results obtained by applying this rule as a preprocessing in an exact algorithm.

Moreover, if no vertex of G is dominated, those facets, which will be called *vertex cover facets*, are the only facet-defining inequalities with right hand side equal to 1. Indeed, consider such an inequality $\sum_{S \in \mathcal{S}'} x_S \geq 1$ with $\mathcal{S}' \neq \mathcal{S}_v, v \in V$. To have a chance to define a facet, $\mathcal{S}'$ can not have any set $\mathcal{S}_v = \{S \in \mathcal{S} : S \ni v\}$ as a subset, since the inequality $\sum_{S \in \mathcal{S}_v} x_S \geq 1$ would be stronger. But then the solution

$$x_S = \begin{cases} 0 & if\ S \in \mathcal{S}' \\ 1 & otherwise \end{cases}$$

covers all vertices of V, but violates the inequality $\sum_{S \in \mathcal{S}'} x_S \geq 1$.

5.2.2 Facet defining inequalities with right hand side larger than 1

A graph G is called χ-*critical* if, for any $v \in V$, $\chi(G[V \backslash \{v\}]) = \chi(G) - 1$. We use the auxiliary graph $G^* = (\mathcal{S}_{max}, E^*)$, where $E^* = \{(S, S') : \beta(V \backslash (S \cap S')) = \beta(V) - 1\}$ [Sas89]. Notice that $\beta(V \backslash (S \cap S'))$ is either $\beta(V)$ or $\beta(V) - 1$, the last case occurring only when $S \cap S'$ intersects all χ-critical subgraphs of G. We first provide another set of inequalities with all coefficients equal to one.

In [Sas89], graph G^* is used to give a sufficient condition for a class of such inequalities to be

facet defining. We translate it here in our terms.

Lemma 5.5 *If $Conv(\mathcal{S}_{max}, V)$ is full dimensional and if G^* is connected, then inequality*

$$\sum_{S \in \mathcal{S}_{max}} x_S \geq \chi(G)$$

is facet defining.

Since the first hypothesis is fulfilled (see beginning of Section 5.2), we only need to show that G^* is connected to prove that a condition is sufficient for the inequality to be facet defining.

Proposition 5.6 *Let G be a χ-critical graph. Then inequality*

$$\sum_{S \in \mathcal{S}_{max}} x_S \geq \chi(G)$$

is facet defining if and only if $\overline{G}$ is connected.

Proof:

Necessity. Assume $\overline{G}$ is not connected. Denote by V' a subset of V such that $\overline{G}[V']$ is not connected to $\overline{G}[V \backslash V']$. Since no stable set in G can intersect both V' and $V \backslash V'$, we can partition $\mathcal{S}_{max}$ in two parts $\mathcal{S}'$ and $\mathcal{S}_{max} \backslash \mathcal{S}'$. Then both inequalities $\sum_{S \in \mathcal{S}'} x_S \geq \chi(G[V'])$ and $\sum_{S \in \mathcal{S}_{max} \backslash \mathcal{S}'} x_S \geq \chi(G[V \backslash V'])$ are valid, and their sum is precisely $\sum_{S \in \mathcal{S}_{max}} x_S \geq \chi(G)$, which is thus not facet defining.
Sufficiency. Saying that $\overline{G}$ is connected amounts to saying that there is a path from any vertex v to w in $\overline{G}$, which is equivalent to saying that there is a sequence of cliques $C_1, C_2, \ldots, C_k$ in $\overline{G}$, such that $C_i \cap C_{i+1} \neq \emptyset$ for all $i \in \{1, \ldots, k-1\}$, $v \in C_1$ and $w \in C_k$. This is also equivalent to the existence of a sequence of stable sets in G with the same property, which holds also if we restrict to maximal stable sets. Moreover, from the above remark and since G is critical, $\chi(G[V \backslash (S_1 \cap S_2)]) = \chi(G) - 1$, for any $S_1, S_2 \in \mathcal{S}_{max}$ such that $S_1 \cap S_2 \neq \emptyset$. So $E^* = \{(S_1, S_2) : S_1 \cap S_2 \neq \emptyset\}$ and G^* is connected. We can now apply Lemma 5.5, which permits to conclude. ■

Facets with right hand side larger than 1 may be obtained using Chvátal's well-known procedure (see [Chv73]):

1. Sum up a subset $Q' \subseteq Q$ of inequalities from the initial set (5.27);
2. Divide the resulting inequality by a positive number k;
3. Round up all coefficients and the right hand side to the nearest integer.

Chvátal called the *elementary closure* of a set of inequalities Q the set of inequalities which can be obtained from Q with one iteration of this procedure.

Applying this to the problem $(C(\mathcal{S}, V))$ gives for any $V' \subseteq V$ and $k > 0$ a valid inequality for $Conv(C(\mathcal{S}, V))$. But specific choices of V' and k may lead to inequalities which are not dominated by the initial ones or even facets of $Conv(C(\mathcal{S}, V))$. We will see that inequalities belonging to the elementary closure of the set (5.27) have a natural interpretation.

Example: Consider the graph G displayed in Figure 5.2 with chromatic number 4 and clique number 3. The set of all maximal stable sets of G is:

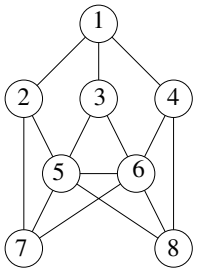

Figure 5.2: A graph G such that $\chi(G) = 4$ and $\omega(G) = 3$.

$$\mathcal{S}_{max} = \{\{1,5\},\{1,6\},\{1,7,8\},\{2,3,4\},\{2,3,8\},\{2,6\},\{3,4,7\},\{3,7,8\},\{4,5\}\}.$$

The inequality

$$x(1,7,8) + x(2,3,4) + x(2,3,8) + x(3,4,7) + 2x(3,7,8) \geq 2$$

defines a facet of $Conv(C(\mathcal{S}_{max}, V))$ which can be obtained using the rounding procedure with the set of inequalities

$$\begin{array}{rllll} & \text{x(2,3,4)} + \text{x(2,3,8)} & + \text{x(3,4,7)} & + \text{x(3,7,8)} & \geq \ 1 \\ \text{x(1,7,8)} & & + \text{x(3,4,7)} & + \text{x(3,7,8)} & \geq \ 1 \\ \text{x(1,7,8)} & + \text{x(2,3,8)} & & + \text{x(3,7,8)} & \geq \ 1 \end{array}$$

and $k = 2$. It can also be obtained by the following argument: the set $\{3,7,8\}$ must be covered by the set $\{3,7,8\}$ or at least two sets among $\{1,7,8\}$, $\{2,3,4\}$, $\{2,3,8\}$ and $\{3,4,7\}$. ■

Such an interpretation holds for any inequality obtained with one iteration of Chvátal's rounding procedure with set $V' \subseteq V$ and number k:

$$\sum_{S \in \mathcal{S}_{max}} \alpha_S x_S \geq \left\lceil \frac{|V'|}{k} \right\rceil \tag{5.29}$$

with $\alpha_S = \left\lceil \frac{|V' \cap S|}{k} \right\rceil$, $\forall\ S \in \mathcal{S}_{max}$. In the following such an inequality will be denoted by Chv(V',k). Next proposition gives a necessary condition for such an inequality to define a facet of $Conv(C(\mathcal{S}_{max}, V))$.

Proposition 5.7 *If Chv(V',k) defines a facet, then* $\left\lceil \frac{|V'|}{k} \right\rceil > \omega(G[V'])$.

Proof: Assume $\left\lceil \frac{|V'|}{k} \right\rceil \leq \omega(G[V'])$, let $C \subseteq V'$ be a maximum clique in $G[V']$, and let $v \in C$ and $w \in C$, $v \neq w$. Notice that v and w are adjacent since they both belong to the same clique C. Consider $\mathcal{S}_v = \{S \in \mathcal{S}_{max} | S \ni v\}$ and $\mathcal{S}_w = \{S \in \mathcal{S}_{max} | S \ni w\}$. A set that would be simultaneously in $\mathcal{S}_v$ and $\mathcal{S}_w$ would contain vertices v and w, which is impossible since it must be stable, hence $\mathcal{S}_v \cap \mathcal{S}_w = \emptyset$. This holds for any pair of vertices from C, so

$$\sum_{v \in C} \sum_{S \ni v} x_S = \sum_{S \cap C \neq \emptyset} x_S \geq |C| = \omega(G[V']) \tag{5.30}$$

is the sum of the cover inequalities of the vertices in C and hence is valid. But since all coefficients are 1, since $\alpha_S \geq 1\ \forall\ S$ such that $S \cap C \neq \emptyset$ and since $\omega(G[V']) \geq \left\lceil \frac{|V'|}{k} \right\rceil$, inequality (5.30) dominates inequality Chv(V',k). ■

In our example, this condition is fulfilled: $\left\lceil \frac{|\{3,7,8\}|}{2} \right\rceil = 2 > 1 = \omega(G[\{3,7,8\}])$.

5.3 Polyhedral results on the set packing formulation

We focus now on formulation $MCP(P)$ and study the corresponding polytope. Recall that we work with $\mathcal{S}_2$, the set of stable sets of size at least 2. With this formulation, all solutions consisting of only one stable set from $\mathcal{S}_2$ satisfy all constraints. There are as many such solutions as there are variables, and since they are affinely independent, $Conv(MCP(P))$ is full-dimensional.

The *conflict graph* of the set of stable sets $\mathcal{S}_2$ is $\mathcal{G} = (\mathcal{S}_2, \{(S, S') : S \cap S' \neq \emptyset\})$. A clique in $\mathcal{G}$ is then a set of stable sets of $\mathcal{S}_2$, having pairwise nonempty intersections. We will say that a clique is maximal if it is maximal under inclusion, i.e. for a maximal clique $\mathcal{C} \subseteq \mathcal{S}_2$, and for each S in $\mathcal{S}_2 \backslash \mathcal{C}$ there is a $S' \in \mathcal{C}$ such that $S \cap S' = \emptyset$. The following result is proved in [Pad73] for general set packing polyhedra, and we adapt it here to $MCP(P)$.

Proposition 5.8 *[Pad73] An inequality of the form*

$$\sum_{S \in \mathcal{C}} x_S \leq 1$$

is a facet of $Conv(MCP(P))$ if and only if $\mathcal{C}$ is a maximal clique in $\mathcal{G}$.

In the next section, we define and characterize in terms of graphs a class of such inequalities which correspond to maximal cliques in $\mathcal{G}$.

5.3.1 Majority set cliques

Given a set of vertices $X \subseteq V$, we call *majority set* clique the subset $\mathcal{C}_X = \{S \in \mathcal{S}_2 : |S \cap X| \geq \frac{|X|+1}{2}\}$. Obviously, $\mathcal{C}_X$ is a clique. If $|X| = 1$, the majority set clique $\mathcal{C}_X = \mathcal{C}_{\{v\}}$ is a *single vertex clique*. In what follows, we first characterize the maximal single vertex cliques, then the maximal cliques with $|X| = 2$ and finally those with $|X| \geq 3$. For $v \in V$, the *anti-neighborhood* of v is defined as $AN(v) = V - \{N(v) \cup v\}$. We assume that $|AN(v)| > 0$: a vertex v with $AN(v) = \emptyset$ belongs to no stable set of $\mathcal{S}_2$, and can be removed from G without changing the $MCP(P)$ formulation; $\chi(G)$ is then just decreased by one.

Proposition 5.9 *Assume $AN(v) = \{w\}$. Then $\mathcal{C}_{\{v\}}$ is a maximal clique (in fact the isolated vertex $\{v, w\}$ of $\mathcal{G}$) if and only if $N(w) = N(v) = V \backslash \{v, w\}$.*

Proof: Notice first that $\mathcal{C}_{\{v\}} = \{v, w\}$ and $N(w) \subseteq N(v)$, otherwise $|AN(v)| > 1$. If $N(w) = N(v)$, then any set $B \subseteq V$, with $B \neq \{v, w\}$ and $|B| \geq 2$ intersecting $\{v, w\}$ must contain at least one edge of G and hence $B \notin \mathcal{S}_2$, so $\mathcal{C}_{\{v\}}$ is maximal. If $N(w) \subset N(v)$, then $\{x, w\} \in \mathcal{S}_2$ for any $x \in AN(w) - \{v\}$ and $\mathcal{C}_{\{v\}}$ is not maximal. ■

Proposition 5.10 *Assume $|AN(v)| \geq 2$. Then $\mathcal{C}_{\{v\}}$ is a maximal clique in $\mathcal{G}$ if and only if $AN(v)$ is not a stable set in G.*

Proof: If $AN(v)$ is a stable set of G, then $AN(v) \in \mathcal{S}_2$ and $AN(v) \cap S \neq \emptyset \; \forall \; S \in \mathcal{C}_{\{v\}}$, i.e. $\mathcal{C}_{\{v\}}$ is not maximal. If there is an edge (x, y) in the subgraph induced by $AN(v)$, then any stable set $S \notin \mathcal{C}_{\{v\}}$ contains at most one vertex in the set $\{x, y\}$, and hence can not intersect both $\{v, x\} \in \mathcal{C}_{\{v\}}$ and $\{v, y\} \in \mathcal{C}_{\{v\}}$. Thus $\mathcal{C}_{\{v\}}$ is maximal. ■

In the case where $AN(v)$ is a stable set (possibly of size one), the unique maximal clique containing $\mathcal{C}_{\{v\}}$ is $\mathcal{C}_{\{v\}} \cup \{S \in \mathcal{S}_2 : AN(v) \subseteq S\}$.

As mentioned in last section, if a graph G contains two non-adjacent vertices x and y such that $N(x) \subseteq N(y)$ (y dominates x), x can be removed from G without changing $\chi(G)$. In particular, if all such vertices are removed, no anti-neighborhood of a vertex can be a stable set. Consequently, in the resulting graph, all inequalities (5.24) are facets.

Assume $X = \{v, w\}$, and that v and w are not adjacent (otherwise $\mathcal{C}_X$ is empty). Since $\frac{|X|+1}{2} = 1.5$, $\mathcal{C}_X = \{S \in \mathcal{S}_2 : \{v, w\} \subseteq S\}$, $\mathcal{C}_X \subseteq \mathcal{C}_{\{v\}}$ and $\mathcal{C}_X \subseteq \mathcal{C}_{\{w\}}$. As a consequence of Proposition 5.9, if $N(v) = N(w) = V \backslash \{v, w\}$, then $\mathcal{C}_X = \mathcal{C}_{\{v\}} = \mathcal{C}_{\{w\}}$ is a maximal single vertex clique. This is the only case where $\mathcal{C}_X$ is maximal. Indeed, if at least one vertex in X, say v, has another non-neighbor, say u, then $\{u, v\} \in \mathcal{C}_{\{v\}}$, and $\mathcal{C}_X \subset \mathcal{C}_{\{v\}}$.

Assume now that $|X| \geq 3$. Next proposition gives a necessary and sufficient condition for $\mathcal{C}_X$ to define a maximal clique in $\mathcal{G}$.

Proposition 5.11 *Let $X \in V$ be a set of at least 3 vertices. Then the majority set clique $\mathcal{C}_X$ is maximal if and only if*

1. *$|X|$ is odd and*
2. *X is stable.*

Proof: If X satisfies 1 and 2, then for any $S \in \mathcal{S}_2 - \mathcal{C}_X$, we have $|X \cap S| \leq \frac{|X|-1}{2}$. Hence $|X \backslash S| \geq \frac{|X|+1}{2} \geq 2$ which means that $X \backslash S \in \mathcal{C}_X$, while $S \cap (X \backslash S) = \emptyset$. Thus $\mathcal{C}_X$ is maximal.
If X does not contain a stable set of cardinality larger than or equal to $\frac{|X|+1}{2}$ then $\mathcal{C}_X$ is empty. Now, if $\mathcal{C}_X$ is nonempty, let $\bar{S} \in \mathcal{C}_X$ s.t. $|\bar{S}| = \frac{|X|+1}{2}$.
If X is not stable, let u and v be two vertices in X such that $(u, v) \in E$. Let $X' = X \backslash \{u, v\}$ and S be any stable set of $\mathcal{C}_X$. Since S is stable, it contains either u or v, or none of them. Hence,

$$|S \cap X'| \geq |S \cap X| - 1 \geq \frac{|X| + 1}{2} - 1 = \frac{|X| - 1}{2} = \frac{|X'| + 1}{2}$$

which means that $S \in \mathcal{C}_{X'}$. Consequently, $\mathcal{C}_X \subseteq \mathcal{C}_{X'}$. Further, removing any vertex $w \in \bar{S}$ from that set yields a stable set belonging to $\mathcal{C}_{X'}$, but not to $\mathcal{C}_X$, hence $\mathcal{C}_X \subset \mathcal{C}_{X'}$, and $\mathcal{C}_X$ is not maximal.
Assume now that X is a stable set with even cardinality. Let v be any vertex of X and S be a stable set of $\mathcal{C}_X$. Then

$$|S \cap (X \backslash \{v\})| \geq |S \cap X| - 1 \geq \frac{|X| + 1}{2} - 1 = \frac{|X| - 1}{2} = \frac{|X \backslash \{v\}|}{2}$$

which is not integer. So $|S \cap (X \backslash \{v\})| \geq \frac{|X \backslash \{v\}|+1}{2}$, which means that $S \in \mathcal{C}_{X \backslash \{v\}}$ and $\mathcal{C}_X \subseteq \mathcal{C}_{X \backslash \{v\}}$. Further, let S' be a stable set of $\mathcal{C}_X$ with cardinality $\frac{|X|}{2} + 1$ such that $v \notin S'$ and $w \in S'$. Then $S \backslash \{w\}$ belongs to $\mathcal{C}_{X \backslash \{v\}}$ but not to $\mathcal{C}_X$ which implies that $\mathcal{C}_X \subset \mathcal{C}_{X \backslash \{v\}}$ and $\mathcal{C}_X$ is not maximal. ■

It follows that if one wants to find all majority set cliques which are not in the initial formulation, one needs only to scan the stable sets S with $|S| \geq 3$ and odd.

5.3.2 Other cliques

There are lots of other maximal cliques in $\mathcal{G}$. Here are some illustrating examples.

Proposition 5.12 *Let X be a stable set of odd size at least 5, and $S \subset X$ such that $|S| = \frac{|X|+1}{2}$, $\mathcal{A} = \{T \in \mathcal{S}_2 : T \cap X = S\}$ and $\mathcal{B} = \{T \in \mathcal{S}_2 : T \cap X = X \backslash S\}$. Then $(\mathcal{C}_X \backslash \mathcal{A}) \cup \mathcal{B}$ is a maximal clique in $\mathcal{G}$.*

Proof: Since $\mathcal{C}_X$ is a clique, so is $\mathcal{C}_X \backslash \mathcal{A}$. Further, each stable set in $\mathcal{C}_X \backslash \mathcal{A}$ intersects $X \backslash S$. So $(\mathcal{C}_X \backslash \mathcal{A}) \cup \mathcal{B}$ is a clique. To prove that it is maximal, consider a stable set $S' \notin (\mathcal{C}_X \backslash \mathcal{A}) \cup \mathcal{B}$. If

$S' \in \mathcal{A}$, then obviously $S' \cap (X \backslash S) = \emptyset$, while $(X \backslash S) \in \mathcal{B}$ (notice that $|X \backslash S| \geq 2$). Assume now that $S' \notin (\mathcal{C}_X \cup \mathcal{B})$, and consider the stable set $(X \backslash S')$, which has empty intersection with S'. Since $|S' \cap X| \leq \frac{|X|-1}{2}$, $|X \backslash S'| \geq \frac{|X|+1}{2}$ which means that $(X \backslash S') \in \mathcal{C}_X$. Further, since $S' \notin \mathcal{B}$, $(X \backslash S') \notin \mathcal{A}$. So $(X \backslash S') \in \mathcal{C}_X \backslash \mathcal{A}$, which permits to conclude. ■

Starting from a clique as defined in Proposition 5.12, one can do the same replacement with another set S' of size $\frac{|X|+1}{2}$ instead of S , provided that $(X \backslash S') \cap (X \backslash S) \neq \emptyset$. For doing the same a third time with S'' such that $|S''| = \frac{|X|+1}{2}$, one should ensure that $(X \backslash S'') \cap (X \backslash S) \neq \emptyset$ and that $(X \backslash S'') \cap (X \backslash S') \neq \emptyset$, and so on. Noticing that there are many possible choices for S, then for S' and so on, can give an idea of the huge number of maximal cliques in $\mathcal{G}$. Furthermore, there are

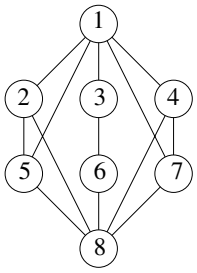

Figure 5.3: A graph with a clique facet which is not derived from a majority set clique.

other cliques which are neither majority set cliques, nor obtainable with the above construction. For the graph displayed in Figure 5.3, the set $\mathcal{C} = \{\{2,3,4\},\{2,6,7\},\{3,5,7\},\{4,5,6\}\}$ is a maximal clique of $\mathcal{G}$ which is in no previous case. Notice that this graph has no dominated vertex, so all single vertex cliques induce facets.

5.4 An improvement on a branch and price algorithm

In this section, we present an exact algorithm for solving MCP. Of course, this algorithm does not run in polynomial time, MCP being an $\mathcal{NP}$-hard problem. Our algorithm uses the branch and price methodology, and is based on a previous algorithm proposed by A. Mehrotra and M. Trick [MT96]. We first depict the working of the original approach, then we propose a preprocessing improving somewhat our implementation, and illustrate this improvement with some computational results.

5.4.1 Original algorithm

In order to allow comparison with Mehrotra and Trick's algorithm, we tried to follow the description in [MT96] as closely as possible. There are however some parts that were not implemented in our code; so our code is somewhat slower (if we take into account the progresses made in

computer speed during the past eight years). We describe in this subsection the most important points of our algorithm, and the choices we made to enhance the global performance.

A branch and price is a branch and bound algorithm, where the bound is obtained by solving a linear program, whose number of variables is so large that column generation is needed for solving it. The first step of a branch and bound for a minimization problem consists in computing an upper bound $\overline{\chi}$ on the chromatic number. This can simply be achieved by finding a feasible solution heuristically. In [MT96], the initial solution is found in a greedy way. The upper bound is then updated each time a new better solution is found. We use for the initial coloring a somewhat more sophisticated approach, namely a local search on the partial colorations set, with a look-ahead neighborhood and penalty evaporation in order to avoid cycling, by I. Blöchliger [Blö01]. The use of a more performant algorithm permits a quick finding of a good coloring; it is however not crucial in the global performance of the branch and bound. Indeed, when starting with a bad solution (e.g. found greedily) other solutions of better quality are usually found in the first nodes of the enumeration tree.

The linear program is in our case the linear relaxation of $MCP^l(C)$ (see Section 5.1.3), so the variables correspond to the maximal stable sets and the $n = |V|$ constraints express that each vertex must be covered by at least one such set. The pricing problem associated is easily shown to be a maximum weight stable set problem. More precisely, given dual values $\lambda_1, \ldots, \lambda_n$, a variable x_S may enter the basis if $\sum_{i \in S} \lambda_i > 1$. If no such variable exists, i.e. if no stable set has a weight larger than one, the optimal value of the linear program has been reached. The lower bound obtained this way has been introduced in Section 5.1.4 as the fractional chromatic number and is in a large majority of cases a sharp bound. For instance, it is exact in the set of graphs G such that $\chi(G) = \omega(G)$.

Branching methodology

The branching rule is as follows. Take two non adjacent vertices a and b and create two branches. In one branch, they get the same color, and in the other they get different colors. To create a subproblem where they have the same color (branch $\texttt{SAME}(a, b)$), construct a new graph from the original one by identifying vertices a and b; to simulate that they have different colors (branch $\texttt{DIFFER}(a, b)$), create another graph by adding an edge between a an b (see Figure 5.4).

The main advantage is that both subproblems are of the same type as the original one, i.e. MCP, and can hence be solved recursively. If the rule would have been to fix a variable to 1 or 0, which is standard in integer programming, this would not have been the case. In Figure 5.5 the whole algorithm is represented.

For the choice of the nodes a and b, A. Mehrotra and M. Trick use the fractional coloring found for the current graph. Their criterion is the following.

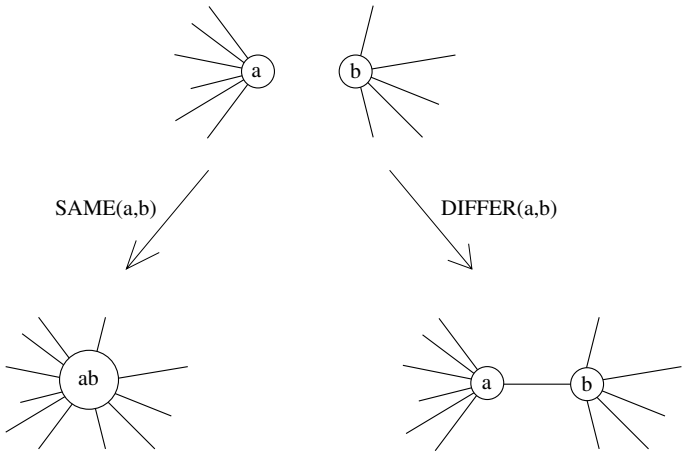

Figure 5.4: Branching rule.

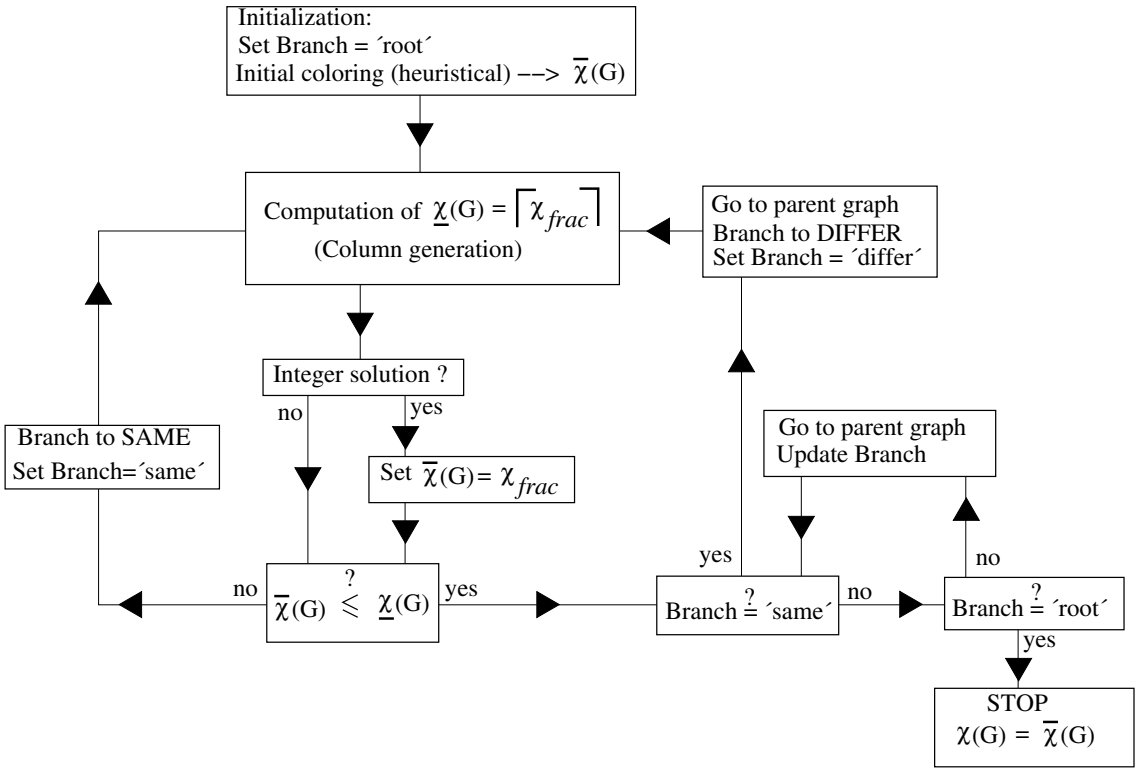

Figure 5.5: Branch and price algorithm for MCP(C).

1. Choose a stable set S such that x_S is as close as possible to $\frac{1}{2}$;
2. Choose $a \in S$;
3. Choose $S' \neq S$, such that $a \in S'$ and $x_{S'} > 0$;
4. Choose $b \in (S \backslash S') \cup (S' \backslash S)$.

The aim of this choice is to forbid, in each branch, that at least one of both x_S and $x_{S'}$ is used fractionally. In this way, an integer coloring is likely to be found faster. In our comparisons, we use this criterion for the choices of a and b.

We tried some other criteria, especially in order to produce a better balance between both subproblems. Indeed, in most colorings, vertex a has a color different from the color of vertex b, so forcing them to have the same color is more restrictive than forcing them to have different colors, and the subproblem `SAME`(a,b) is in most cases somewhat easier than `DIFFER`(a,b). To counter this difference, we tried to branch by choosing the vertices a and b so as to maximize $|N(a) \cap N(b)|$; the argument is that vertices that have a large common neighbourhood are likely to get the same color. Simulations showed that this did not improve the algorithm in most graphs tested, both in terms of the number of nodes visited as in running time. However, there are some structured graphs, where lot of time could be saved by combining this criterion with a preprocessing procedure, see Sections 5.4.2, 5.4.3 and 5.4.3.

Accelerating the pricing

As mentioned above, the pricing problem is a maximum weight stable set problem, the weights being given by the dual values. To prove that a basis is optimal, one has to solve this problem exactly at least once. But if one wants to prove that some variable may enter the basis, one needs only to find a solution of weight larger than one. This can be done heuristically, as in [MT96]: first they run a greedy algorithm for MSP; if it returns a solution, the corresponding variable enters the basis, and otherwise an exact algorithm is started in order to find a solution with certainty if it exists, or to prove that it does not exist. An improvement they also proposed was to generate several stable sets at a time, by running the greedy algorithm repeatedly.

We tried to apply this reasoning a step further, i.e. if the greedy algorithm does not provide a satisfactory solution, first try a more sophisticated heuristic, and run the exact algorithm only if this last heuristic also fails. The heuristic chosen was a *variable neighborhood search* (VNS) [HM01, HMU01], which is described in Figure 5.6.

Here $N_1, \ldots, N_{k_{max}}$ denote a sequence of neighborhood types. They are usually defined in such a way that s and any solution in $N_k(s)$ become more different as k increases. In our case the local search algorithm used was tabu search (see Section 2.6 for a description).

```
1: Construct an initial solution s_0;
2: repeat
3:     Set k = 1;
4:     repeat
5:         Generate a solution s' randomly in N_k(s);
6:         Improve s' by a local search; denote s'' the new solution;
7:         if s'' is better than s then
8:             Set s = s'' and k = 1;
9:         else
10:            Set k = k + 1;
11:        end if
12:    until k = k_max
13: until the stopping condition is met
```

Figure 5.6: Variable neighborhood search

In our experiments, adding the VNS step did not systematically decrease the total running time. Of course, the VNS could many times find entering variables where the greedy algorithm failed, but our implementation was too time consuming and the results showed that it was worth skipping the VNS and directly run the exact procedure. However, those numerical results do not allow us to reject the strategy of applying more sophisticated heuristics for the pricing problem. First, our implementation could be improved. Second, the instances we ran were not very large, and the exact algorithm rarely took more than one second; if this had been the case, it is likely that applying VNS would have lead to appreciable savings in time.

In Mehrotra and Trick's implementation, the greedy algorithm is called once, so at most one stable set is generated at each iteration. We tested our algorithm with 1, 5, 10, 50 and 200 calls to the greedy algorithm. The best results were obtained with 5 calls, so we used this value in our tests.

5.4.2 Preprocessing

As mentioned in the beginning of this section, our implementation is not as performant as the one of Mehrotra and Trick. The main drawback in our code seems to lie in the way we implemented the exact procedure for solving the weighted independent set problem; we invested indeed less effort than is suggested in [MT96]. This can also explain the fact that calling several times the greedy algorithm brought in our case some savings in execution time (which is not the case in [MT96]), since it permitted to call fewer times the exact algorithm.
The preprocessing we propose next depends only on the structure of the graphs encountered in the enumeration tree. Since our branching rule is exactly the same as the one used by Mehrotra

and Trick, the differences between the two implementations should not invalidate the positive results given at the end of the section.

We see here how by applying two simple vertex deletion rules at each node of the branch and bound tree can sometimes slightly speed up the algorithm. The first rule is built on the notion of domination defined in section 5.2. Recall that a vertex v dominates a vertex w if $N(w) \subset N(v)$, and a dominated vertex can always be removed from the graph without decreasing the chromatic number. The second rule requires the knowledge of a lower bound $\underline{\chi}$ on the chromatic number. A vertex v such that $d(v) < \underline{\chi} - 1$ can also be removed from the graph, since there is always a color in the set $\{1, \ldots, \underline{\chi} - 1\}$, available for vertex v, after having optimally colored $G[V \backslash \{v\}]$. Further, this last reduction is worth being applied, as we have a good lower bound on $\chi(G)$.

Hence the following simple procedure sometimes permits to reduce the graph G, without changing $\chi(G)$.

Input: Graph G, lower bound $\underline{\chi}$ on $\chi(G)$
Output: Possibly reduced graph G'

1: **repeat**
2: Remove each dominated vertex from G;
3: Remove each vertex v such that $d(v) < \underline{\chi} - 1$;
4: **until** no more vertex can been removed this way.

Since at each node of the branch and bound tree the problem is a graph coloring one, this preprocessing can be called each time a new subproblem has been created. The complexity of checking if a node is dominated being roughly in $O(|V|^2)$, each loop is in $O(|V|^3)$. This is small as compared to the time necessary to solve $MCP^l(G)$, which requires to solve at least one weighted MSP on G to optimality.

5.4.3 Computational results

The next tables show the performance of the algorithm, in terms of CPU time and number of nodes visited in the enumeration tree. To show its contribution to the global performance, results are displayed with and without preprocessing. An additional column shows the number of vertices which have been removed at each node, using the procedure. Sometimes, when the alternative criterion for the choice of vertices a and b (see Section 5.4.1) permits the preprocessing to produce substantially better results than the first one (what is the case for some structured graphs), those results are displayed in an extra table. The tests have been run on machines with a processor of 2 GHz, and the sizes of graphs have been chosen so that the algorithm finishes most of the times within an hour. Linear programs are solved using the CPLEX 9.0 callable library from a program written in C++. All times are given in seconds.

Random graphs

The graphs of type `Rand_n_p` are randomly generated on n vertices, such that for each pair of vertices, there is an edge connecting them with probability p.
If p is near to 0 or 1, the lower bound $\lceil \chi_{frac(G)} \rceil$ is very often equal to $\chi(G)$, and its quality decreases as p approaches $\frac{1}{2}$. This makes `Rand_n_p` with $p \simeq \frac{1}{2}$ the most difficult type of random graphs to color.
For this reason, our tests have only been run with values $p = 0.3, 0.5\ 0.7$, and $n = 70, 80\ 90$, which is about the limit size beyond which actual exact algorithms do not finish in reasonable time. Table 5.8 shows those results for random graphs with 70, 80 and 90 vertices. Values are averages on five instances, and between brackets is the number of problems solved within an hour, if it is lower than five.

			Without preprocessing		With preprocessing		
G	$\chi(G)$	$\chi_{frac}(G)$	Time	Nodes	Time	Nodes	Elim./Node
`Rand_70_0.3`	7.75	6.76	701(4)	1561	667(4)	1496	0.001
`Rand_70_0.5`	11.8	10.78	40.3	270.6	41.1	271.4	0.002
`Rand_70_0.7`	17.2	16.31	8.68	150.6	5.98	94.6	0.052
`Rand_80_0.3`	8	7.31	18.5(4)	6.5	18.8(4)	6.5	0
`Rand_80_0.5`	13	11.69	822(4)	2862	515(4)	2548	0.0003
`Rand_80_0.7`	19	18	10.8	90.2	8.83	93.4	0.032
`Rand_90_0.3`	9	7.89	241(2)	113	229(2)	113	0
`Rand_90_0.5`	14	12.73	389(2)	953	241(2)	953	0
`Rand_90_0.7`	20.6	19.36	88.6	705.4	74	627.4	0.024

Table 5.8: Results on random graphs

These results give evidence that it is worth running the preprocessing. Indeed, even if the number of deleted vertices is small in comparison with the number of nodes in the enumeration tree, it often permits to save a substantial amount of time.

Flat graphs

The following graphs were created using the generator available at J. Culberson's "Graph Coloring Page"[Cul96]. The graphs of the form `Flat_n_k_p` consist of n vertices partitioned into k stable sets of sizes $\lfloor \frac{n}{k} \rfloor$ and $\lceil \frac{n}{k} \rceil$, with the vertices belonging to different parts being linked with probability p, and in such a way that all vertices have almost (± 1) the same degree.
Flat graphs with small but strictly positive p are the most difficult to color: indeed, if p is near 1, there will probably be a clique of size k, if p is near 0, this will not be the case and there may exist a better partition.
The flat graphs we used for our experiments have sizes of 80, 90 and 100 vertices, and probabil-

ities 0.1 and 0.2. All values are averages on five graphs, and as for random graphs, the number of graphs colored are indicated in brackets, if it is lower than five.

			Without preprocessing		With preprocessing		
G	$\chi(G)$	$\chi_{frac}(G)$	Time	Nodes	Time	Nodes	Elim./Node
Flat_80_9_0.1	4	3.75	67.1	1	68.7	1	0.2
Flat_80_9_0.2	6	5.09	72.0	1	74.9	1	0
Flat_90_9_0.1	4	3.13	1410(3)	1	1269(3)	1	0.33
Flat_90_9_0.2	6	5.43	192(4)	1	189(4)	1	0
Flat_100_10_0.1	5	4.02	1651(1)	1	819(1)	1	1
Flat_100_10_0.2	-	-	-	-	-	-	-

Table 5.9: Results on flat graphs

From the results in Table 5.9, the only graphs G for which the algorithm takes less than one hour have $\chi(G) = \lceil \chi_{frac}(G) \rceil$. For those graphs, the computation of $\chi_{frac}(G)$ may take several minutes; however, it is interesting to point out that half the total time has been saved, thanks to the removal of a unique vertex in Flat_100_10_0.1. For graphs of the form Flat_100_10_0.2, the time limit was reached for all instances.

Geometric graphs

A geometric graph of type G_n_d is constructed by uniformly generating n points in a square of side 1, and linking two vertices if their points are at a distance of at most d. The reverse geometric graph RG_n_d is obtained the same way, but by linking the vertices which are separated by a distance of *at least* d. Notice that if the set of points and a distance d are given, the corresponding geometric and reverse geometric graphs are complements of each other.
We tested our algorithm first on geometric graphs with 100, 150, 200, 250, 300 vertices, $d = 0.1, 0.5, 0.9$, then on reverse geometric graphs with the same sizes and distances. The performances are reported in Table 5.10, where again values are averages on five distinct graphs. As appears from the results, these graphs are not difficult to color. For almost the whole set of geometric graphs tested, $\chi_{frac}(G) = \chi(G)$, and the enumeration tree has only one node. The only graphs for which several nodes are visited, are of the form Geom_n_0.5; this happens only when the initial heuristic fails to find an optimal coloring. Notice the large number of vertices removed by the preprocessing, which is a direct consequence of the structure of the graph: vertices near of a corner have a good chance of having a small degree, or of being dominated. For reverse geometric graphs the effect is even stronger: in most cases, more than half of the graph is removed.

			Without preprocessing		With preprocessing		
G	$\chi(G)$	$\chi_{frac}(G)$	Time	Nodes	Time	Nodes	Elim./Node
G_100_0.1	4.8	4.8	0.45	1	0.35	1	32.4
G_100_0.5	31.8	31.8	2.04	6.6	1.60	3.4	3.94
G_100_0.9	71.8	71.8	3.08	1	2.65	1	8.4
G_150_0.1	6.8	6.8	0.66	1	0.61	1	28.2
G_150_0.5	42	42	7.89	17	6.04	14.2	1.59154
G_150_0.9	103.8	103.8	9.51	1	7.3	1	7.8
G_200_0.1	7.4	7.4	0.99	1	1.06	1	19
G_200_0.5	54.4	54.4	39.4	48.2	31.6	34.2	0.53
G_200_0.9	137.8	137.8	16.6	1	16.3	1	8.2
G_250_0.1	8.8	8.7	2.11	1	2.21	1	20.2
G_250_0.5	66.2	66	106	101	72.7	78.6	0.44529
G_250_0.9	174.4	174.4	28.9	1	25.2	1	9.8
G_300_0.1	10.2	10.2	2.39	1	2.71	1	12
G_300_0.5	80.4	80.3	277	90.2	175	57.4	0.70731
G_300_0.9	207	207	72.7	1	50.5	1	18.4
RG_100_0.1	44.8	44.7	1.63	1	0.73	1	55.2
RG_100_0.5	6	5.53	1.33	1	0.12	1	85.8
RG_100_0.9	3.2	3.2	1.04	1	0.04	1	95
RG_150_0.1	55.4	55.4	3.31	1	1.11	1	93.6
RG_150_0.5	6.8	6.8	2.35	1	0.17	1	131
RG_150_0.9	3.6	3.6	1.68	1	0.07	1	146
RG_200_0.1	61.6	61.6	5.88	1	1.5	1	138
RG_200_0.5	7	6.48	3.33	1	0.29	1.8	98.9
RG_200_0.9	3.4	3.3	3.20	1	0.11	1	195
RG_250_0.1	67	66.8	10.7	1	2.55	1	168
RG_250_0.5	7	6.65	4.97	1	0.42	1	225
RG_250_0.9	4	3.7	2.78	1	0.2	1	244
RG_300_0.1	72.6	72.2	19.6	8.2	6.13	3.8	47
RG_300_0.5	7	6.75	7.32	1	0.64	1	266
RG_300_0.9	3.8	3.7	4.37	1	0.3	1	296

Table 5.10: Results on geometric graphs

Queen graphs

The queen graph `Queen_n_m` is obtained by associating a vertex to each square of an $n \times m$ chessboard, and linking two vertices a and b if a queen could move in one step from the square of vertex a to the square of vertex b. It is clear that $\chi(\texttt{Queen\_n\_m}) \geq \max(n, m)$, since the vertex set corresponding to a row or column is a clique of size n or m. It is known that if $n = m$ is not a multiple of 2 or 3, then $\chi(\texttt{Queen\_n\_m}) = n$. Recently, the converse has been shown not to be true. Indeed, on each `Queen_n_n` graph, with $12 \leq n \leq 24$, $n \mod 2 = 0$ or $n \mod 3 = 0$, a coloring with n colors has been found [Chv03]. However, the most difficult `Queen_n_n` graphs for our algorithm are still those with n multiple of 2 or 3.

Tests have been run on 7 queen graphs having from 36 to 100 vertices. Results on Table 5.11 show that the preprocessing brings no improvement. This is not surprising, as vertices have all

			Without preprocessing		With preprocessing		
G	$\chi(G)$	$\chi_{frac}(G)$	Time	Nodes	Time	Nodes	Elim./Node
Queen_6_6	7	7	0.45	1	0.64	1	0
Queen_7_7	7	7	1.12	1	1.63	1	0
Queen_8_8	9	8.44	1.82	1	2.78	1	0
Queen_8_9	9	9	-	-	-	-	-
Queen_9_9	10	9	109	57	108	57	0
Queen_9_10	10	10	-	-	-	-	-
Queen_10_10	11	10	-	-	-	-	-

Table 5.11: Results on queen graphs

about the same degree and symmetrical characteristics.
There is however a way of solving those problems more quickly. In Section 5.4.1, we describe the branching strategy consisting in choosing the pair of vertices a and b maximizing $|N(a) \cap N(b)|$. In Table 5.12, results show how it can improve the performance on queen graphs. Those improvements can be explained by the ability of our branching strategy to create high degree vertices, which are likely to dominate some other vertices.

Mycielski graphs

In [Myc55], J. Mycielski proposes the following graph transformation. Given a graph $G = (\{x_1, \ldots, x_n\}, E)$, construct a new graph $M(G)$ with vertex set $\{y_1, \ldots, y_n, z_1, \ldots, z_n, w\}$, and edge set so that $\{z_1, \ldots, z_n\}$ is a stable set, y_i is linked to y_j if and only if x_i is linked to x_j, y_i is linked to z_j if and only if x_i is linked to x_j and w is linked to all z_i. It is not a difficult task to prove that $\chi(M(G)) = \chi(G) + 1$, while $\omega(M(G)) = \omega(G)$. Hence this transformation

G	Time	Nodes	Elim./Node
Queen_6_6	0.49	1	0
Queen_7_7	1.13	1	0
Queen_8_8	2.04	1	0
Queen_8_9	757.88	2221	0.1
Queen_9_9	103.33	37	0
Queen_9_10	81.46	99	0.1
Queen_10_10	1895.89	297	0.08

Table 5.12: Results on queen graphs with preprocessing and another branching strategy.

permits one to obtain graphs with arbitrarily large gaps between chromatic and clique numbers. The graph Myciel_1 is the K_2, so Myciel_2=$M(K_2) = C_5$, Myciel_3 is a graph with chromatic number 4 and clique number 2, and more generally $\chi(\texttt{Myciel\_k}) = k + 1$.
It is shown in [LPU95] that

$$\chi_{frac}(\texttt{Myciel\_k}) = \chi_{frac}(\texttt{Myciel\_k-1}) + \frac{1}{\chi_{frac}(\texttt{Myciel\_k-1})}$$

so $\chi_{frac}(\texttt{Myciel\_2}) = 2.5$, $\chi_{frac}(\texttt{Myciel\_3}) = 2.9$, $\chi_{frac}(\texttt{Myciel\_4}) \simeq 3.24$, $\chi_{frac}(\texttt{Myciel\_5}) \simeq 3.55$, and so on. The gap $\chi(\texttt{Myciel\_k}) - \lceil \chi_{frac}(\texttt{Myciel\_k}) \rceil$ thus becomes arbitrarily large as k increases. This makes Mycielski graphs the most difficult graphs to color of our whole instance set. We see on Table 5.13 how inefficient our algorithm becomes when applied on Mycielski

				Without preprocessing		With preprocessing		
G	$\|V\|$	$\chi(G)$	$\chi_{frac}(G)$	Time	Nodes	Time	Nodes	Elim./Node
Myciel_4	23	5	3.24	1.35	659	1.13	517	1.34
Myciel_5	47	6	3.55	-	-	-	-	-

Table 5.13: Results on Mycielski graphs

graphs. This is clearly due to the large gap between $\chi(G)$ and $\chi_{frac}(G)$, which produces a large enumeration tree. Nevertheless, applying again the alternative branching of Section 5.4.1 with the preprocessing brings better results, as can be seen in Table 5.14. In average, more than one

G	Time	Nodes	Elim./Node
Myciel_4	0.43	167	2.23
Myciel_5	117	42903	1.5

Table 5.14: Results on Mycielski graphs with preprocessing and another branching strategy.

vertex is deleted from the graph, which is very large if we take into account the huge number of nodes visited. This is likely due to the tendency of our branching strategy of creating high

degree vertices, which have great probability of dominating the numerous low degree vertices of Mycielski graphs.

5.5 Concluding remarks

Throughout this section, we have seen several linear programming approaches of MCP, and some relationships. We have begun the study of the polyhedral structure of $MCP(C)$ and $MCP(P)$ by characterizing the simplest facets. Finally, we followed the column generation approach first described in [MT96], and were able to somewhat improve it by adding a vertex deletion rule at each node of the branch and bound tree.
As one may have noticed, the polyhedral results found in Section 5.2 were not explicitly applied in last Section. In fact, we tried to add majority set cliques inequalities to formulation $MCP(P)$ for some small graphs. Unfortunately, we found no example where the optimal value could be improved this way, so we decided not to implement a separation procedure for them. We also tried to add inequalities corresponding to holes on five vertices to our algorithm, but this also gave only disappointing results: it permitted only after significant computational efforts to slightly increase the optimal value of the linear program. This was rarely sufficient to improve the lower bound by 1. We further tried to add cutting planes to the branch and price algorithm with the $MCP(P)$ formulation, corresponding to odd holes in the conflict graph of stable sets (see Section 5.3 for the conflict graph and [GLS88] for the odd holes separation procedure). This also gave bad results. More generally, we observed lots of symmetry in $MCP(C)$ and $MCP(P)$ formulations, in the sense that sometimes many fractional colorings exist with the same (or almost the same) objective value. This can explain the failure of the separation methods we tested: a solution which may have been cut off by an inequality can most times be replaced by another solution of about the same value.
We did not try further cutting plane generation. One of the main drawbacks with this approach applied to a column generation framework is that the pricing problem looses his MSP structure after having added some cutting planes, and has some additional requirements on the vertices involved in a cut. This complicates the implementation and slows down the execution. Nevertheless, these observations should not prevent us from trying further cutting plane approaches, and this direction shall deserve further investigations.

Conclusion

During this research, several goals have been pursued, like proving the solvability or $\mathcal{NP}$-hardness of a problem, characterizing a class of facets, or improving some algorithm. Some of those goals have been reached, some have not. In addition, there are some results which have been obtained as byproducts of the main research directions.

In Chapter 1, the - a posteriori - ambitious goal was to prove solvability or $\mathcal{NP}$-hardness of the maximum stable set problem (MSP) in the class of P_5-free graphs. Although we did not succeed in our initial goal, we nevertheless came across some new interesting classes of graphs, while trying to find a characterization of P_5-free augmenting graphs. To the contrary, the polynomial subcase of MSP in *banner*-free graphs (Section 1.3) was found after succeeding in the research of an efficient algorithm to detect augmenting chains in $(S_{1,2,i}, banner)$-free graphs. The results found in this Chapter are a contribution to the structural knowledge of the bound between solvable and $\mathcal{NP}$-hard subcases of MSP, which is probably one of the $\mathcal{NP}$-hard problems for which this bound is the most precisely known. The most attractive open case remains the P_5-free graphs.

In Chapter 2, we have started the study of a new kind of graph optimization problem motivated by security purposes in mobile ad hoc networks. Our first steps consisted in evaluating the quality of a constructive heuristic and detecting some cases where it produces a suboptimal solution. We proposed alternative constructions inspired by the first method and applied tabu search which is a framework suitable for each variant of the problem. We also examined the case of complete graphs, and obtained some bounds on the optimal solution of a variant of the problem. The most important objective we could not attain was the characterization of the complexity of the problems in consideration. The principal reason was the lack of similar $\mathcal{NP}$-hard problems: unlike many other optimization problems, a solution is here a collection of partial subgraphs, so it is of size in $O(|V||E|)$, while the sizes of solutions for most $\mathcal{NP}$-hard problems are in $O(|V|)$ or $O(|E|)$. In our opinion, the problem is so general that $\mathcal{NP}$-hardness should hold and be proved for some simplified version.

Chapter 3 was devoted to the exposition of a concise result at the border between easy and difficult problems in discrete tomography. There is hope for the existence of more sophisticated algorithms for generalizing the present procedure. Recall that the case of four or more colors is

$\mathcal{NP}$-hard, while the case of three colors is still open.

The main Theorem of Chapter 4 is also a byproduct of the research on coloring problems. By combining several ideas, we could obtain new families of $\mathcal{NP}$-hard subcases, and somewhat enrich the knowledge of the boundary between hard and easy cases for MCP. Only little systematic research has been made in this area, and this boundary is not as well-known as for MSP. Nevertheless, as lots of $\mathcal{NP}$-hard cases have been found, we now have precise directions to search for polynomial cases.

In the last Chapter, the aim was to characterize some facets in the polyhedra of $MCP(C)$ and $MCP(P)$. After computer-aided polyhedra calculations for small graphs, we quickly realized that a complete characterization would be difficult, if not out of reach. However, we were able to characterize in simple graph theoretical terms the facets associated to inequalities with small coefficients. Though we could not apply directly those results in separation procedures, it gave us the idea of trying the preprocessing which could appreciably improve the branch and price algorithm. Adding cutting planes is a direction which has not been sufficiently explored, partly on account of the difficulty in implementing, and should be further investigated.

This research has shown how fruitful it can be to deal at the same time with several different topics. Putting together ideas from different fields has lead to substantial progresses and has provided a collection of nontrivial byproducts.

Another aspect which is worth being pointed out is the contribution of computational assistance. Beyond permitting comparison of two algorithms in an objective manner, computers can be used to perform enumerative tasks not achievable by hand. Output obtained this way can often give evidence of some fact, provide counterexamples, or suggest new research directions. A regular use of such approaches convinced us to go further in our investigations, or sometimes helped us to realize that some direction was indeed a deadlock.

Finally it is implicit that besides working with computers, working with people is the most important and efficient way to progress in research. During my thesis, I had the chance to collaborate with researchers in various fields of combinatorial optimization; they could reveal to me interesting unexplored areas, and provide me with the tools necessary to their study.

Bibliography

[AIST77] H. Ariyoshi, M. Ide, I. Shirakawa, and S. Tsukiyama. A new algorithm for generating all the maximal independent sets. *SIAM J. Computing*, 6:505–517, 1977.

[AL03] V.E. Alekseev and V.V. Lozin. Augmenting graphs for independent sets. To appear in Discr. Appl. Math., 2003.

[AL04] V.E. Alekseev and V.V. Lozin. Local transformations of graphs preserving independence number. *Discr. Appl. Math.*, 135:17–39, 2004.

[Ale83] V.E. Alekseev. On the local restrictions effect on the complexity of finding the graph independence number. *Combinatorial-algebraic methods in applied mathematics, Gorkiy Univ. Press*, pages 3–13 (in Russian), 1983.

[Ale91] V.E. Alekseev. On the number of maximal independent sets in graphs from hereditary classes. *Combinatorial-algebraic methods in applied mathematics*, pages 5–8 (in Russian), 1991.

[Ale03] V.E. Alekseev. On easy and hard hereditary classes of graphs with respect to the independent set problem. *Discr. Appl. Math.*, 132:17–26, 2003.

[Ale04] V.E. Alekseev. A polynomial algorithm for finding largest independent sets in fork-free graphs. *Discr. Appl. Math.*, 135:3–16, 2004.

[AM99] C. Arbib and R. Mosca. On $(P_5, diamond)$-free graphs. Research report, Department of Pure and Applied Mathematics, University of Aquila, 1999.

[AMO93] R. Ahuja, T.L. Magnanti, and J. B. Orlin. Network flows. *Practice-Hall, (London)*, 1993.

[BBC+01] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self-organization in mobile ad hoc networks: The approach of terminodes. *IEEE Comm. Magazine*, June:166–174, 2001.

[BCH01] L. Buttyan, S. Capkun, and J.-P. Hubaux. The quest for security in mobile ad hoc networks. In *Proc. ACM Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001.

[BCH03] L. Buttyan, S. Capkun, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on mobile computing*, 2:52–64, 2003.

[Ber57] C. Berge. Two theorems in graph theory. *Proc. Nat. Acad. Sci. USA*, 43:842–844, 1957.

[Ber70] C. Berge. *Graphs and Hypergraphs.* North-Holland publishing company, 1970.

[BH99] A. Brandstädt and P.L. Hammer. On the stability number of claw-free P_5-free and more general graphs. *Discrete Appl. Math.*, 95:163–167, 1999.

[BL01] A. Brandstädt and V.V. Lozin. A note on α-redundant vertices in graphs. *Discrete Appl. Math.*, 108:301–308, 2001.

[BLNP95] E. Barcucci, A. Del Lungo, M. Nivat, and R. Pinzani. Reconstructing digital pictures from x-rays. Technical report, Dipartimento de Sistemi e Informatica, Universita i Firenze, 1995.

[Blö01] I. Blöchliger. A new heuristic for the graph coloring problem. Master's thesis, École Polytechnique Fédérale de Lausanne, 2001.

[CCY96] G.J. Chang, J.-J. Chen, and J.-H. Yan. Quasi-threshold graphs. *Discrete Appl. Math.*, 69:247–255, 1996.

[CD01] M. Chrobak and C. Dürr. Reconstruction polyatomic structures from discrete x-rays: $\mathcal{NP}$-completeness proof for three atoms. *Theoretical Computer Science*, 259:81–98, 2001.

[Chv73] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[Chv03] V. Chvátal. V. Chvátal home page, coloring the queen graphs, 2003. `http://www.cs.rutgers.edu/~chvatal/queengraphs.html`.

[CMaPZ02] P. Coll, J. Marenco, and I. Méndez Díaz P. Zabala. Facets of the graph coloring polytope. *Annals of Operations Research*, 116(1-4):79–90, 2002.

[Cos95] D. Costa. *Méthodes de recherche constructives, séquentielles et évolutives pour des problèmes d'affectation sans contraintes.* PhD thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 1995.

[CPdW03] M.C. Costa, C. Picouleau, and D. de Werra. On some special cases of an image reconstruction problem. ORWP 03/02, École Polytechnique Fédéral de Lausanne, SB-IMA-ROSE, 2003.

[CPS85] D.G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14:926–934, 1985.

[CPSdW03] M.C. Costa, C. Picouleau, D. Schindl, and D. de Werra. A solvable case of image reconstruction in discrete tomography. ORWP 03/06, École Polytechnique Fédéral de Lausanne, SB-IMA-ROSE, 2003.

[CRST] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. Submitted for publication.

[CS89] G. Cornuejols and A. Sassano. On the 0,1 facets of the set covering polytope. *Mathematical Programming*, 43:45–55, 1989.

[Cul96] J. Culberson. Graph coloring page, 1996. `http://web.cs.ualberta.ca/~joe/Coloring`.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[EHdW84] C. Ebenegger, P.L. Hammer, and D. de Werra. Pseudo-boolean functions and stability of graphs. *Annals of Discrete Mathematics*, 19:83–98, 1984.

[GG97] R.J. Gardner and P. Grintzmann. Discrete tomography: determination of finite sets by x-rays. *Trans AMS*, 349(6):2271–2296, 1997.

[GHL03] M.U. Gerber, A. Hertz, and V.V. Lozin. Stable sets in two subclasses of banner-free graphs. *Discrete Appl. Math.*, 132:121–136, 2003.

[GHS03] M. Gerber, A. Hertz, and D. Schindl. P_5-free augmenting graphs and the maximum stable set problem. *Discrete Appl. Math.*, 132:109–119, 2003.

[GJ99] M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1999.

[GL03] M.U. Gerber and V.V. Lozin. On the stable set problem in special P_5-free graphs. *Discrete Appl. Math.*, 125:215–224, 2003.

[Glo86] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operation Research*, 13:533–549, 1986.

[GLS88] M. Grötschel, M. L. Lovàsz, and A. Schriver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

[Gol78] M.C. Golumbic. Trivially perfect graphs. *Discrete Math.*, 24:105–107, 1978.

[GR97] V. Giakoumakis and I. Rusu. Weighted parameters in $(P_5, \bar{P}_5)$-free graphs. *Discrete Appl. Math.*, 80:255–261, 1997.

[Han86] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on Numerical Methods in Combinatorial Optimization*, 1986.

[HK99] G. Herman and A. Kuba. Discrete tomography: Foundations, algorithms and applications. *ed. Birkhauser*, 1999.

[HLS03] A. Hertz, V. Lozin, and D. Schindl. Finding augmenting chains in extensions of *claw*-free graphs. *Inform. Proc. Letters*, 86:311–316, 2003.

[HM01] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European J. of Operational Research*, 130:449–467, 2001.

[HMU01] P. Hansen, N. Mladenović, and D. Urošević. Variable neighborhood search for the maximum clique problem. Technical Report 45, GERAD, 2001.

[Jac03] T. Jaccard. Sécurité dans les réseaux mobiles. Master's thesis, École Polytechnique Fédérale de Lausanne, 2003.

[Joh89] E. L. Johnson. Modeling and strong linear programs for mixed integer programming. In *Algorithms and Model Formulations in Mathematical Programming*, pages 1–43. Springer-Verlag Berlin, s. w. wallace edition, 1989.

[JSdW03] T. Jaccard, D. Schindl, and D. de Werra. Some simple optimization techniques for self-organized public key management in mobile ad hoc networks. ORWP 03/07, École Polytechnique Fédérale de Lausanne, SB-IMA-ROSE, 2003.

[KKTW01] D. Král, J. Kratochvíl, Z. Tuza, and G.J. Woeginger. Complexity of coloring graphs without forbidden induced subgraphs. *Lecture Notes in Computer Science*, 2204:254–262, 2001.

[Las70] L. S. Lasdon. *Optimization Theory for Large Systems.* Macmillan Series in Operations Research, 1970.

[Loz00] V.V. Lozin. Stability in P_5 and banner-free graphs. *European J. Oper. Research*, 125:292–297, 2000.

[LPU95] M. Larsen, J. Propp, and D. Ullman. The fractional chromatic number of Mycielski's graphs. *J. of Graph Theory*, 19:411–416, 1995.

[Mah90] N.V.R. Mahadev. Vertex deletion and stability number. ORWP 90/02, École Polytechnique Fédérale de Lausanne, DMA-ROSE, 1990.

[Min80] G.J. Minty. On maximal independent sets of vertices in claw-free graphs. *J. Combin. Theory Ser. B*, 28:284–304, 1980.

[Mos97] R. Mosca. Polynomial algorithms for the maximum independent set problem on particular classes of P_5-free graphs. *Inform. Proc. Letters*, 61:137–144, 1997.

[Mos99] R. Mosca. Stable sets in certain P_6-free graphs. *Discrete Appl. Math.*, 92:177–191, 1999.

[Mos03] R. Mosca. Some results on maximum stable sets in certain P_5-free graphs. *Discrete Appl. Math.*, 132:175–183, 2003.

[MS99] R.M. McConnell and J.P. Spinrad. Modular decomposition and transitive orientation. *Discrete Math.*, 201:189–241, 1999.

[MT96] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. *INFORMS, Journal on Computing*, 8(4):344–354, 1996.

[Myc55] J. Mycielski. Sur le coloriage des graphes. *Colloquim Mathematics*, 3:161–162 (in French), 1955.

[NT01] D. Nakamura and A. Tamura. A revision of minty's algorithm for finding a maximum weight stable set of a claw-free graph. *J. Oper. Res. Soc. Japan*, 44:194–204, 2001.

[Pad73] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[Pic01] C. Picouleau. Reconstruction of domino tiling from its two orthogonal projections. *Theoretical Computer Science*, 255:437–447, 2001.

[Pic02] C. Picouleau. Reconstruction of convex polyonimoes from orthogonal projections of their contours. Technical report, Laboratoire CEDRIC, CNAM, 2002.

[Pol74] S. Poljak. A note on stable sets and coloring of graphs. *Comment. Math. Univ. Carolinae*, 15:307–309, 1974.

[Rys63] H. Ryser. *Combinatorial Mathematics.* Mathematical Association of America and Quinn & Boden, Rahway, NJ, 1963.

[Sas89] A. Sassano. On the facial structure of the set covering polytope. *Mathematical Programming*, 44:181–202, 1989.

[Sbi80] N. Sbihi. Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Math.*, 29:53–76 (in French), 1980.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization, 1986.

[Sch97] Edward R. Scheinerman. *Fractional Graph Theory.* Wiley-Interscience, 1997.

[Sch04] D. Schindl. Some new $\mathcal{NP}$-hard hereditary classes for graph coloring. ORWP 04/ 01, École Polytechnique Fédérale de Lausanne, SB-IMA-ROSE, 2004.

[Yan82] M. Yannakakis. The complexity of the partial order dimension problem. *SIAM J. Algebraic and Discrete Methods*, 3:351–358, 1982.

Curriculum vitae

Nom:	David SCHINDL
Date de naissance:	17 juillet 1976
Nationalité:	Autrichienne
État civil:	Célibataire
Adresse:	Ch. des Aubépines 5, CH-1004 Lausanne

Formation

1991-1995	Maturité fédérale de type C (scientifique) à Genève.
1996-2001	Diplôme d'ingénieur mathématicien de l'École Polytechnique Fédérale de Lausanne (EPFL), prix ASRO du meilleur travail de diplôme en recherche opérationnelle sur le plan national.

Activités professionnelles

2001-2004	Assistanat en recherche opérationnelle à l'Institut de Mathématiques de l'EPFL (Prof. D. de Werra): aide aux exercices en graphes et algorithmique, encadrement d'étudiants pour projets de semestre, arbitrage d'articles pour journaux en mathématiques discrètes. Chercheur invité à trois reprises pour collaboration avec Prof. M. Labbé à l'Université Libre de Bruxelles. Présentations à des conférences (Edimbourg (2002); Montréal, Lausanne, Copenhague (2003); Zinal, Paris, Fribourg et Bruxelles (2004)).
2004	Cours d'optimisation combinatoire de 15 heures donné dans le cadre d'un DEA à Beyrouth.