Christian Buchter

Integer Programming Models in Graph Coloring

Bachelor thesis in mathematics

Department of Mathematical Sciences

University of Copenhagen

Advisors

Søren Eilers & Michal Adamaszek

May 21, 2017

Abstract

The aim is to present and compare a few formulations of the graph colouring problem in the language of mixed-integer optimization. The student will learn and describe the principles of linear and integer optimization, formulate and implement a few known integer models of the graph colouring problem, and compare their performance on various benchmark graphs.

Contents

1 Linear Programming **1**

1.1 The structure of a linear program 1
1.3 feasible and optimal solutions 2
1.4 Convexity 2
1.10 The geometric/graphical intuition 4
1.11 Matrix representation of an LP and Slack variables 5
1.12 Duality 6
1.15 Efficiency of solving an LP 7

2 Integer Programming and Graphs **9**

2.1 Some Graph notation 9
2.9 Mixed Integer Programming - MIP 10
2.11 Relaxation 11

3 Graph colouring **14**

3.1 Fundamental colouring terms and results 14
3.8 Integer programming formulations 15

4 Colouring results **20**

A Test appendix **22**

Bibliography **23**

1. Linear Programming

In this chapter we introduce the concept of Linear programming. Most proofs will be omitted but proofs and more in depth explanations can be found in [Van15]

1.1 The structure of a linear program

In a linear program we want to maximise or minimize a given linear function $z : \mathbb{R}^n \to \mathbb{R}$ subject to a number of linear inequalities or equalities $g_i : \mathbb{R}^n \to \mathbb{R}$ with $g_i(x_1, ..., x_n) \geq b_i$,$g_i(x_1, ..., x_n) \leq b_i$ or $g_i(x_1, ..., x_n) = b_i$. The function, z, that we want to optimise is called the **Objective** and the set of inequalities are called the **Constraints**. A general linear problem is written:

$$\begin{aligned} \text{min/max} \quad & z(x_1, ..., x_n) \\ \text{s.t.} \quad & g_1(x_1, ..., x_n)\ \textbf{ordRel}\ b_1, \\ & \vdots \\ & g_m(x_1, ..., x_n)\ \textbf{ordRel}\ b_m \\ & \text{where each } \textbf{ordRel} \text{ can be } \ \leq, \geq \ \text{ or } \ = \end{aligned} \tag{1.1}$$

Example 1.2
A maths student wants to save money on his diet while still remaining healthy. To stay healthy his diet must contain at least $b_1 = 6$ units of protein, $b_2 = 15$ units of carbs, $b_3 = 5$ units of fat and $b_4 = 7$ units of vitamins.
He consider buying 3 food products with different nutritional values and prices:

1. x_1 is a take away meal costing 5 and containing 3 units of protein, 3 units of carbs, 2 units of fat and 1 unit of vitamins.

2. x_2 is a vegetable costing 1 and containing 1 unit of protein, 2 units of carbs, 0 units of fat and 4 units of vitamins.

3. x_3 is a type of bread costing 2 and containing $\frac{1}{2}$ unit of protein, 4 units of carbs, 1 unit of fat and 0 units of vitamins.

He then define an optimization problem minimizing the cost of food subject to getting the right nutrition

$$\begin{aligned} \min \quad & 5x_1 + x_2 + 2x_3 \\ \text{s.t.} \quad & 3x_1 + x_2 + \tfrac{1}{2}x_3 \geq 6, \\ & 3x_1 + 2x_2 + 4x_3 \geq 15, \\ & 2x_1 + x_3 \geq 5, \\ & x_1 + 4x_2 \geq 7, \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \tag{1.2}$$

1.3 feasible and optimal solutions

feasibility

Given a Linear problem on form 1.1 any point of $\mathbb{R}^n$ such that all m constraints are true is called a feasible solution. The set of all these points is called the feasible set. In the case Example 1.2 the feasible set is the set of all combinations of amounts of the different foods such that the nutritional requirements are met. If a problem has no feasible solutions, that problem is said to be infeasible. A feasibility problem is a special case in linear programming where our object function is constant and thus if any feasible solution exists, that solution is optimal.

optimal solutions

A feasible solution $\mathbf{x}_0 \in \mathbb{R}^n$ is said to be optimal if $z(\mathbf{x}_0) \geq z(\mathbf{x})$ (when maximizing) or $z(\mathbf{x}_0) \leq z(\mathbf{x})$ (when minimizing) for all feasible solutions $\mathbf{x} \in \mathbb{R}^n$.
In some instances when feasible solutions exist, but no maximal (or minimal) solution exists the problem is said to be *unbounded*. In that case one or more variable in the objective can approach ∞ or $-\infty$ in a solution, all while that solution remains feasible and the objective value diverges.

1.4 Convexity

Definition 1.5. *A set $X \in \mathbb{R}^n$ is said to be convex if for any two points $a, b \in X$ the straight line segment connecting a and b is entirely within X.*

Theorem 1.6. *A feasible set of a linear program is convex*

Proof.
The feasible subset of $\mathbb{R}^n$ of an LP is exactly the intersection of all the halfspaces given by each constraint function.
Since such halfspaces are convex and the intersection of convex sets is also convex, the entire feasible set is convex. □

Lemma 1.7. *Any solution between two feasible solutions is feasible.*

Proof.
Since the feasible region of an LP is convex, any point between two feasible points is within the feasible region. □

The Fundamental Theorem of Linear Programming

Definition 1.8. *An* ***Extreme point*** *of a feasible space in $\mathbb{R}^n$ is a point on the boundary of the feasible set that is the intersection of n constraint functions.*

Theorem 1.9. *If a set $\mathcal{S}$ of optimal solutions to a given LP exists, then some $s \in \mathcal{S}$ such that s is in an extreme point in the feasible set.*

Proof.
Let A be the feasible region to a given Linear problem and let $\mathbf{x}_0 = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ be an optimal solution with value $z(\mathbf{x}_0 = \zeta$. Suppose $\mathbf{x}_0$ is not in an extreme point of A. Then $\mathbf{x}_0$ is either in the open set $A' = A^O$ or on some point of the boundary $\bar{A} - A^O$ that is not an extreme point. In the first case at least one of the coordinates $x_1, \cdots, x_n$ can be increased or decreased until the solution reaches a point $\mathbf{x}_1$ on $\bar{A} - A^O$ with $z(\mathbf{x}_1) <$ or $> \zeta$. Thus that cannot be the case.
In the second case there are two options;

1. The hyperplane defined by $z(\mathbf{x}) = \zeta$ intersects one or more of the constraint functions in $\mathbf{x}_0$ but is not parallel to any of them.

2. The hyperplane defined by $z(\mathbf{x}) = \zeta$ is equal to the entire hyperplane defined by a constraint function or all points of the intersection of several constraint functions are contained within the hyperplane.

In the first case we have a contradiction since we can increase or decrease the values $x_1, \cdots, x_n$ while still remaining in the feasible region and thus find a vector $\mathbf{x}_1$ with $z(\mathbf{x}_1) <$ or $> \zeta$. In the second case the entire subset of the hyperplane contained in the feasible region is optimal and thus the extreme points of that set are also optimal solutions. □

1.10 The geometric/graphical intuition

From a geometric perspective the feasible region of an LP can be seen as a convex polyhedron encapsulated by the hyperplanes defined by each constraint function And the objective is a hyperplane that can be "pushed" orthogonally to either maximize or minimize a point of the plane such that it is contained in the feasible polyhedron.
In case of Example 1.2, with three real variables, x_1, x_2, x_3 the polyhedron will be a polyhedron in $\mathbb{R}^3$ and the objective plane will be the plane defined by the linear equation $5x_1 + x_2 + 2x_3 = \zeta$ where ζ is the value we want to minimize.

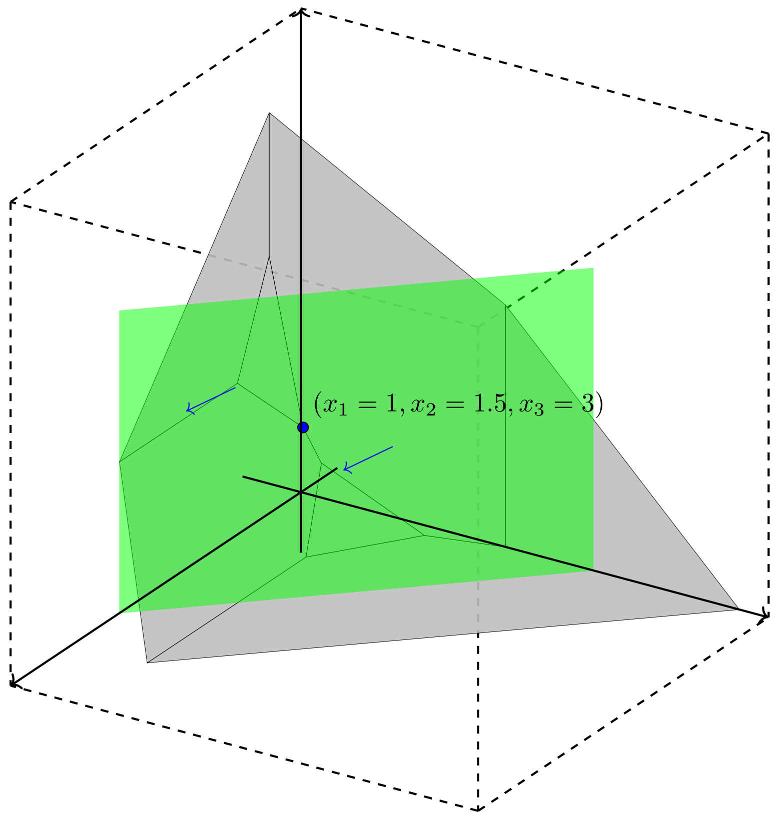

Figure 1.1: A graphical reppressentation of Example 1.2

1.11 Matrix representation of an LP and Slack variables

Since the objective function and the constraints of a linear program are all linear functions we can also write a linear program using matrix-vector products. The object can be written as the product of the vector of variables to be determined $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \mathbf{x}$, and the transposed vector of coefficients $[c_1, c_2, ..., c_n] = \mathbf{c}^T$ of each x_i in the objective.

Likewise the constraints can be written as the matrix-vector products of $n \times m$ constraint matrices,A where m is the number of a constraints with a certain order relation, and $\mathbf{x}$ with some order relation $(\leq, \geq \textit{ or } =)$ to a $1 \times m$ vector $\mathbf{b}$.

Rewriting any $\leq$ constraint to $\geq$ in case of minimization or any $\geq$ to $\leq$ in case of maximization (the equality constraints can be written with two inequalities) we can write the linear program on it's *canonical form*:

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \geq \mathbf{b}, \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1.3}$$

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1.4}$$

in case of Example 1.2 we write it on it's canonical form:

$$\begin{aligned} \min \quad & [5,1,2]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ \text{s.t.} \quad & \begin{bmatrix} 3 & 2 & 1 \\ 12 & 2 & 4 \\ 7 & 0 & 2 \\ 3 & 5 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \geq \begin{bmatrix} 5 \\ 10 \\ 5 \\ 10 \end{bmatrix}, \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1.5}$$

Much like how we made an LP on canonical form by manipulating the inequalities, we can also write any LP on it's *standard form*

$$\begin{aligned} \max/\min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1.6}$$

by introducing a $1 \times m$ vector, $\mathbf{s}$, with one slack variable for each constraint and thus creating a new LP on standard with the same solutions as the old one.

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} + \mathbf{s} = \mathbf{b}, \\ & \mathbf{x},\mathbf{s} \geq 0, \end{aligned} \tag{1.7}$$

These forms and the concept of slack variables will come in handy later (I hope)

1.12 Duality

Another important result in Linear programming is the concept of duality.

Definition 1.13. *Given a maximization linear program on canonical form 1.4 We define it's dual problem as the minimization problem:*

$$\begin{aligned} min \quad & \boldsymbol{b}^T \boldsymbol{y} \\ s.t. \quad & A^T \boldsymbol{y} \geq \boldsymbol{c}, \\ & \boldsymbol{y} \geq 0, \end{aligned} \tag{1.8}$$

Theorem 1.14. *An optimal solution to a linear program on it's primal form is also optimal in the dual form*

Proof. □ proof

Weak and strong duality

1.15 Efficiency of solving an LP

The simplex method

A popular algorithm for finding the optimal solution of LPs, taking advantage of the convexity of the feasible region and Theorem 1.9 is the **Simplex method**. [Van15], where Theorem 3.4 also gives a proof of 1.9 using the simplex method, explains this further. The basic idea of the algorithm is to move from extreme point to extreme point of the feasible set and never return to an already visited solution. Thus the algorithm will find an optimal solution after at most the number of extreme points iterations. If a problem has n variables and m constraints, an upper bound of the number of iterations is

$$\binom{n+m}{m}.$$

which is maximized when $n = m$. Thus we see that

$$\frac{1}{2n} 2^{2n} \leq \binom{n+m}{m} \leq 2^{2n}$$

Polynomial time algorithms

Even though the simplex method is one of the most popular algorithms, the worst case runtime is exponential compared to the problem size. Other algorithms have however been made, which has a polynomial worst case runtime. In 1979 Khachian(1979) gave the first algorithm, the *ellipsoid*

method, proved to run in polynomial time later [Kar84] gave a new, more efficient, algorithm also solving linear problems in polynomial time.
Even though these algorithms has better worst case time complexity compared to the simplex method, they are often slower in practise. They do however show that linear programming is in $\mathcal{P}$

In the next chapter we will introduce Integer programming which is $\mathcal{NP}$-hard.

2. Integer Programming and Graphs

In this chapter we introduce the concept of Integer programming. We will approach the subject with examples and results from graph theory, but apart from fundamental definitions we will omit most of the theory from this field of mathematics, but a more in depth explanation can be found in [Wol98].

2.1 Some Graph notation

Definition 2.2. *A* ***graph*** *G is an ordered set (V, E) where V is a set of vertices and E is a set of edges and each edge in E is a subset of V containing two vertices.*

Definition 2.3. *let $G = (V, E)$ be a graph. Two vertices $v, u \in V$ are said to be* ***adjacent*** *if the edge (v, u) (or (u, v)) is in E.*

Definition 2.4. *let $G = (V, E)$ be a graph. A subset S of V form an* ***independent set*** *if no two vertices in S are adjacent in G*

Definition 2.5. *let $G = (V, E)$ be a graph. A subset S of V form a* ***clique*** *if all pairs of two vertices in S are adjacent in G*

Definition 2.6. *let $G = (V, E)$ be a graph. A* ***maximal independent set*** *or a* ***maximal clique*** *is an independent set or a clique respectively where if any other vertex $v \in V$ is added to the set, it will lose the property of being an independent set or clique respectively*

Definition 2.7. *A* ***maximum independent set*** *or a* ***maximum clique*** *in a graph G is an independent set or a clique respectively where no other independent set or clique in G respectively have more vertices.*

Definition 2.8. *the* ***clique number*** *$\omega(G)$ of a graph G is the number of vertices in a maximal clique in G.*

2.9 Mixed Integer Programming - MIP

A Mixed integer problem (MIP) is a way of solving optimisation problems with integer values. The Structure of an integer problem is the same as a linear problem, but with the extra constraint that all or some of the variables are in $\mathbb{Z}$. This added constraint might seem insignificant, but it introduces crucial differences to problems with real variables. One significant difference is that the feasible space of an MIP consists of discrete sets and thus it is not convex.

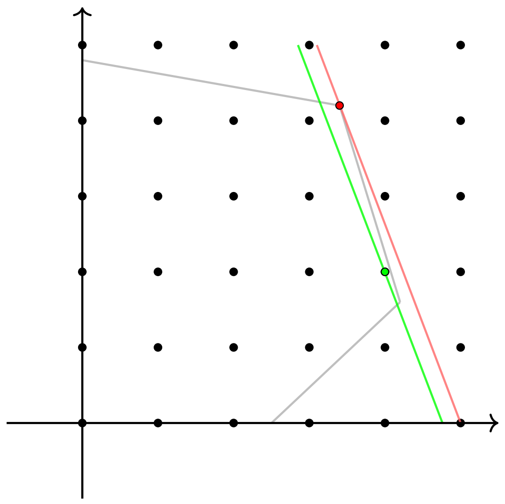

Figure 2.1: Figure showing the feasible region of an IP with the optimal solution in green and a red non-feasible solution that would have been optimal if the set was convex.

Integer and Binary Constraints

An important method in Integer programming is **Binary programming**. Here variables take values either 1 or 0 these binary variables are very useful since they can easily be used to say "Is variable x_i part of the solution? then $x_i = 1$ otherwise $x_i = 0$. Binary constraints will be used extensively once we start colouring Graphs, but another example of their usefulness is in determining maximal cliques and independent sets of graphs:

Example 2.10
Maximal clique and maximal independent set problem:
Let $G = (V, E)$ be a graph and let $H = (V, E')$ be it's complement graph.

Then

$$\begin{array}{ll} \max & \sum_{v\in V} x_v \\ \text{s.t.} & x_u + x_v \leq 1, \forall (u,v) \in E \\ & x_v \in \{0,1\},\ \forall v \in V \end{array} \tag{2.1}$$

calculates the maximal independent set and

$$\begin{array}{ll} \max & \sum_{v\in V} x_v \\ \text{s.t.} & x_u + x_v \leq 1, \forall (u,v) \in E' \\ & x_v \in \{0,1\},\ \forall v \in V \end{array} \tag{2.2}$$

Calculates the maximal clique

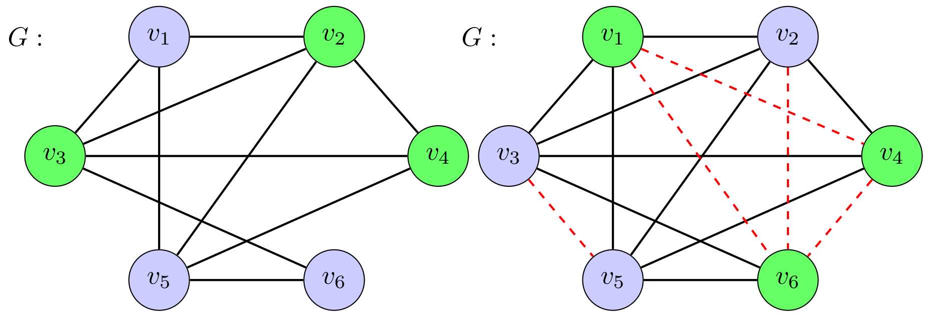

Figure 2.2: A maximal clique and a maximal independent set in a graph G

proof?

2.11 Relaxation

It is well known that 2.10 is an $\mathcal{NP}$-hard task. Thus unlike Linear programming, with real variables, integer programming can be used to calculate NP-hard tasks. in some cases it is however still useful to view the IP as a linear problem and assign real values to the variables instead of integer. This problem, where values in $\mathbb{Z}$ are assigned values in $\mathbb{R}$ instead and binary variables are assigned values in $[0,1]$ is called the **Linear relaxation** of the IP. And the solution to the linear relaxation of an IP can often tell us something about the solution to the IP itself or even give the integer solution.

Example 2.12
Some examples of cases where the relaxation is useful to different degrees are

the following formulations of the shortest path problem and the longest cycle problem in graphs:

Length of shortest path between two vertices

Let $G = (V, E)$ be a directed graph where V is the set of vertices and E is the set of edges. Let A be the adjacency matrix of G and introduce the set of binary values $x_{i,j}$ where

$$x_{i,j} = \begin{cases} 1 & \text{if } (i,j) \text{ is in the path} \\ 0 & \text{otherwise} \end{cases}$$

And the sets $V^+(i)$ and $V^-(i)$ where

$$\begin{aligned} V^+(i) &= \{v : (i,v) \in E\} \\ V^-(i) &= \{v : (v,i) \in E\} \end{aligned}$$

We want to minimise the $\sum x_{i,j}$ subject to conditions that ensures we get a path form $s \in V$ to $t \in V$ in G.

$$\begin{array}{lll} \min & \sum_{i,j \in V} x_{i,j} & \\ \text{s.t.} & \sum_{j \in V^+(s)} x_{s,j} = 1, & \\ & \sum_{i \in V^-(t)} x_{i,t} = 1, & \\ & \sum_{j \in V^+(v)} x_{v,j} = \sum_{i \in V^-(v)} x_{i,v}, \forall v \in V \backslash \{s,t\}, & \\ & x_{i,j} \leq A_{i,j}, & \forall i,j \in V, \\ & x_{i,j} \in \{0,1\}, & \forall i,j \in V. \end{array} \tag{2.3}$$

In this case the convex hull of the feasible set has all its extreme points in integer points. Thus the relaxed problem with $x_{i,j} \in [0,1]$ has has optimal solutions in integral values and we don't have to solve the original IP at all.

Length of shortest directed cycle in a graph

Let $G = (V, E)$ be a directed graph where V is the set of vertices and E is the set of edges. Let A be the adjacency matrix of G and introduce the set of binary values $x_{i,j}$ where

$$x_{i,j} = \begin{cases} 1 & \text{if } (i,j) \text{ is in the cycle} \\ 0 & \text{otherwise} \end{cases}$$

We want to minimise the $\sum x_{i,j}$ subject to conditions that ensures we get a cycle in G.

$$\begin{array}{lll} \min & \sum_{i,j\in V} x_{i,j} & \\ \text{s.t.} & \sum_{j\in V} x_{i_0,j} = \sum_{i\in V} x_{i,j_0}, & \forall j_0, i_0 \in V, \\ & x_{i,j} \le A_{i,j} \le 1, & \forall i,j \in V, \\ & \sum_{i,j\in V} x_{v,j} \ge 1 & \\ & x_{i,j} \in \{0,1\}, & \forall i,j \in V. \end{array} \tag{2.4}$$

In this case a solution to the relaxed problem with $x_{i,j} \in [0,1]$ will be a union of directed cycles and the objective value will always minimize to 1. Thus the relaxed problem does not tell us much about the actual optimal integral solution, but it does however tell us if a feasible solution exists.
In the case of Example 2.10 the relaxed problem doesn't tell us much at all. In some cases the relaxed problem will have the same solution as the IP but in most cases it will find a much higher value. Thus all we can get from the relaxation is an upper bound of the actual problem.

figures to examples?

3. Graph colouring

In this chapter we introduce main concept of graph vertex-colouring and some integer formulations for colouring graphs.

3.1 Fundamental colouring terms and results

Definition 3.2. *A* ***vertex-colouring*** *of a graph $G = (V, E)$ is an assignment of colors $C = 1, ..., k$ such that each $v \in V$ is assigned one color and no two adjacent vertices are assigned the same color.*

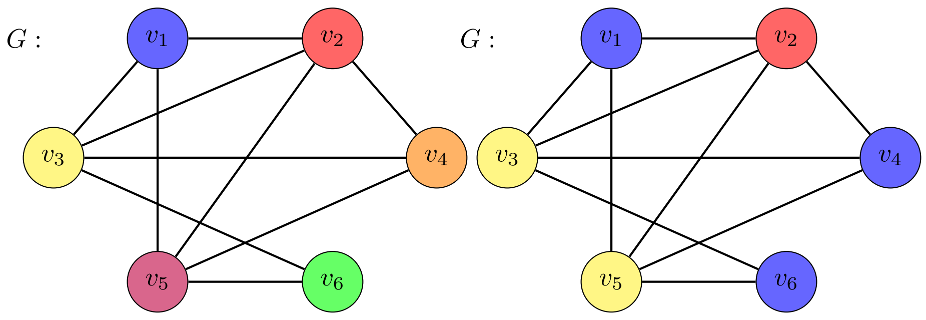

Figure 3.1: A graph coloured with the trivial and an optimal vertex colouring.

in this paper a colouring will always refer to a vertex-colouring.

Definition 3.3. *A* ***k-colouring*** *of a graph G is a vertex-colouring using k colours.*

Definition 3.4. *An* ***optimal colouring*** *of a graph, G, is a colouring of G using the minimum amount of colours possible.*

Definition 3.5. *The* ***chromatic number*** *$\chi(G)$ of a graph, G is the number number of colours in an optimal colouring of G.*

Theorem 3.6. *Vertices of one color form an independent set*

Proof.
Let $G = (V, E)$ be a graph and suppose that a set of vertices S are assigned the same color but does not form an independent set. Then there exists some edge in E connecting two vertices $s_1, s_2 \in S$. But that implies that s_1 and s_2 are assigned different colours which gives us contradiction □

Theorem 3.7. *Given a graph G*
$\chi(G) \geq \omega(G)$

Proof.
Let $G = (V, E)$ be a graph and suppose that a set S of $k = \omega(G)$ vertices form a maximal clique in G. Then each $s_i \in S$ has edges in E to all other vertices of S and they must each have distinct colours by 3.2. And since G_S is a sub graph of G, we get $\omega(G) = \chi(G_S) \leq \chi(G)$. □

3.8 Integer programming formulations

Here we formulate some integer problems to calculate an optimal colouring. The same formulations can be used with minor simplifications to calculate if feasibility problems of weather graphs are k-colourable.

The standard formulation

In the standard formulation we want to introduce $k|V|$ binary variables, $x_{v,c}$ for a suitable $k \geq \mathcal{X}(G)$. where

$$x_{v,c} = \begin{cases} 1 & \text{if vertex } v \text{ is assigned colour } c \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

and k binary variables y_c indicating if color c is used in the colouring.

Proposition 3.9. *The following formulation assigns an optimal colouring to the graph and gives the chromatic number:*

$$\begin{array}{lll} \mathit{min} & \sum_{c \in \{1...k\}} y_c & \\ \mathit{s.t.} & \sum_{c \in \{1...k\}} x_{v,c} = 1, \forall v \in V & \\ & x_{v,c} + x_{u,c} \leq 1, & \forall (u,v) \in E, \forall c \in \{1...k\} \\ & x_{v,c} - y_c \leq 0, & \forall v \in V, \forall c \in \{1...k\} \\ & x_{v,c} \in \{0,1\}, & \forall v \in V, \forall c \in \{1...k\} \\ & y_c \geq 0, & \forall c \in \{1...k\} \end{array} \tag{3.2}$$

Proof.
Given a graph $G = (V, E)$:

1. An optimal colouring is feasible and optimal in 3.2:
   Given an optimal colouring of G and assigning each $x_{v,c}$ the values specified in 3.1, we see that:

   a) the first condition is satisfied since each vertex only has one colour.

   b) the second condition is satisfied since no two adjacent vertices has the same colour.

   c) the third condition assign values to each y_c such that $y_{c_i} \geq 1$ if colour c_i is used to colour some vertex $\in V$ and otherwise greater than zero.

   thus the solution is feasible, and since the object is to minimize the sum of all y_c's each y_c will be minimized to $y_c = \begin{cases} 1 & \text{if colour } c \text{ is used} \\ 0 & \text{otherwise} \end{cases}$
   and since the colouring was optimal and used the fewest colours possible, their sum will be optimal and equal $\chi(G)$

2. An optimal solution to 3.2 returns $\chi(G)$ and assign the vertices colours such that the colouring is optimal:

□

A scheduling formulation

Another Integer formulation of the graph colouring problem for a graph $G = (V, E)$ is the scheduling formulation. Here we introduce V integer variables $\{X_1, \cdots X_v\}$ with the desired result: $X_v = c$ if vertex v is assigned colour c. And another E binary variables $x_{u,v} \forall (u, v) \in E$ with the desired result:

$$x_{u,v} = \begin{cases} 1 & \text{if vertecies } u, v \text{ are assigned colours } c_u, c_v \text{ respectively with } c_u < c_v \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Furthermore we introduce the variable c that has to be greater or equal to the amount of colours used.

Proposition 3.10. *The following formulation assigns an optimal colouring to the graph and gives the chromatic number:*

$$
\begin{array}{lll}
min & c & \\
s.t. & X_u - X_v + kx_{u,v} \leq k-1, & \forall (u,v) \in E \\
 & X_v - X_u - kx_{u,v} \leq -1, & \forall (u,v) \in E \ \textit{for} \ k \geq c \\
 & X_v - c \leq 0, & \forall v \in V \\
 & x_{u,v} \in \{0,1\}, & \forall (u,v) \in E
\end{array}
\tag{3.4}
$$

Proof.

1. An optimal colouring is feasible and optimal in 3.4:

   a) Since $X_u \neq X_v$ for all adjacent $u, v \in V$ we have the option of $X_u < X_v$ and $X_u > X_v$. If $X_u < X_v$ then $x_{u,v}$ must equal 1 in order for the first two constraint to be feasible. If $X_u > X_v$ then $x_{u,v}$ must be 0. But since $x_{u,v}$ is not inherited from the colouring, we can assign these the desired values and the constraints will be satisfied.

   b) Under the assumption that the colours assigned to G are natural numbers $\leq \chi(G)$, the highest value of X_v is exactly the chromatic number of G. The last constraint then ensures that c is at least $\chi(G)$ and when minimized it will become the chromatic number and optimal.

2. An optimal solution to 3.4 returns $\chi(G)$ and assign the vertices colours such that the colouring is optimal:
   Let ζ be an optimal solution to the problem.

   a) The third constraint ensures that no value of X_v is greater than ζ.

   b) The first and second constraints ensures that no two adjacent vertices have the same colour. Thus it is a proper colouring and ζ must be at least $\chi(G)$ for a solution to be feasible. And since ζ is optimal, we can conclude that $\zeta = \chi(G)$

□

The binary encoding formulation

The last formulation of the graph colouring problem for a graph $G = (V, E)$, that this thesis will focus on is the binary encoding formulation. Here we introduce $V \lceil log_2(k) \rceil$ integer variables $x_{v,b}$ with the desired result:

$$x_{v,b} = \begin{cases} 1 & \text{if vertex } v \text{ has the } b\text{'th bit in it's assigned colour set to 1} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

And another $2E \lceil log_2(k) \rceil$ binary variables $z_{u,v,b}$ and $z_{u,v,b}$ to help create the equality $z_{u,v,b} = |x_{u,b} - x_{v,b}|, \forall (u, v) \in E$. Furthermore we introduce the variable c that has to be greater or equal to the greatest value of any colour used.

Proposition 3.11. *The following formulation assigns an optimal colouring to the graph and gives the chromatic number:*

$$\begin{array}{lll} min & c & \\ s.t. & c - \sum_{b=0}^{B} 2^b \cdot x_{v,b} \leq -1, & \forall v \in V \\ & z_{v,u,b} - 2t_{v,u,b} + x_{v,b} - x_{u,b} = 0, & \forall (u,v) \in E \forall b \in [0, \cdots, B] \\ & \sum_{b=0}^{B} z_{v,u,b} \geq 1, & \forall (u,v) \in E \qquad \textit{for } k \geq c, B = \lceil log_2(k) \rceil \\ & x_{v,b} \in \{0,1\}, & \forall v \in V \forall b \in [0, \cdots, B] \\ & z_{u,v,b} \in \{0,1\}, & \forall (u,v) \in E \forall b \in [0, \cdots, B] \\ & x_{u,v,b} \in \{0,1\}, & \forall (u,v) \in E \forall b \in [0, \cdots, B] \end{array} \tag{3.6}$$

Proof.

1. An optimal colouring is feasible and optimal in 3.6:

   a) Given an optimal solution with values of $x_{v,b}$ defined as our desired results 3.5, we see that at, since adjacent vertices have different colours, at least one bit in the colours are different for such two vertices.
   For such a bit, $x_{v,b} - x_{u,b}$ from the second constraint function is either 1 or -1, and $z_{v,u,b}$ and $t_{v,u,b}$ must be either $z_{v,u,b} = 1$ and $t_{v,u,b} = 0$ or $z_{v,u,b} = 1$ and $t_{v,u,b} = 1$ respectively for the constraint

to be satisfied. But since these values are not inherited from the colouring, they can take the appropriate values.

b) Sine $z_{v,u,b} = 1$ for some bit of two adjacent vertices, the third constraint is also satisfied.

c) since the first constraint states that c is at least one higher than the value of the highest value colour used (since a vertex can be assigned colour 0 by this formulation), and if we assume the colouring has assigned colours values as low as possible, c will minimize to the chromatic number of the graph.

2. An optimal solution to 3.6 returns $\chi(G)$ and assign the vertices colours such that the colouring is optimal:

   a) since all $x_{v,b}$ has a value, all vertices have colours trivially.

   b) As established before, a $z_{v,u,b}$ value of 1 indicates that the b'th bit of vertex u and v are different. Likewise we see that since $x_{v,b} - x_{u,b} \in \{-1, 0, 1\}$ and $2t_u, v, b \in \{0, 2\}$ a $z_{v,u,b}$ value of 0 will only be feasible if the b'th bit of u and v are the same.
   By that observation we see that the third constraint ensures that at least one bit of the colours of two adjacent vertices are different, making the colours different. Thus the formulation will assign a proper colouring to a graph

   c) the first constraint states that c is at least the value of the highest value colour assigned to any vertex. When minimizing c the formulation will therefore assign the lowest amount of colours possible with the lowest possible values where the solution is still feasible. Thus it will assign an optimal colouring to a graph and return the chromatic number.

□

4. Colouring results

In this chapter I show the results from colouring a selection of graphs with the IP formulations formulated in Chapter 3. The IP's are solved using the Cplex solver and the k-values are found

Table 4.1: Results

Graph		standard	scheduling	binary
myciel3 Vertices=11 Greedy ub =4	Status:	MIP_optimal	MIP_optimal	MIP_optimal
	Best objective:	4.0	4.0	4.0
	Lower bound:	4.0	4.0	4.0
	Compute time:	0.062 seconds	0.047 seconds	0.156 seconds
myciel4 Vertices=23 Greedy ub =5	Status:	MIP_optimal	MIP_optimal	MIP_time_limit_fe
	Best objective:	5.0	5.0	5.0
	Lower bound:	5.0	5.0	4.0
	Compute time:	0.297 seconds	0.547 seconds	50.031 seconds
myciel5 Vertices=47 Greedy ub =6	Status:	MIP_optimal	MIP_time_limit_feasible	MIP_time_limit_fe
	Best objective:	6.0	6.0	6.0
	Lower bound:	6.0	4.0	3.0
	Compute time:	6.329 seconds	50.0 seconds	50.031 seconds
myciel6 Vertices=95 Greedy ub =7	Status:	MIP_time_limit_feasible	MIP_time_limit_feasible	MIP_time_limit_in
	Best objective:	7.0	7.0	-1
	Lower bound:	5.0	3.0	inf
	Compute time:	50.016 seconds	50.031 seconds	50.078 seconds
myciel7 Vertices=191 Greedy ub =8	Status:	MIP_time_limit_feasible	MIP_time_limit_feasible	MIP_time_limit_in
	Best objective:	8.0	8.0	-1
	Lower bound:	2.5	3.0	inf
	Compute time:	50.047 seconds	50.031 seconds	50.172 seconds

A. Test appendix

Bibliography

[Kar84] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Combinatorica* 4.4 (1984), pp. 373–395. ISSN: 1439-6912. DOI: 10.1007/BF02579150. URL: http://dx.doi.org/10.1007/BF02579150.

[Van15] Robert J Vanderbei. *Linear programming*. Springer, 2015.

[Wol98] Laurence A Wolsey. *Integer programming*. Vol. 42. Wiley New York, 1998.