Integer Programming Models in Graph Coloring

Bachelor defence by Christian Buchter

June 16th 2017

Presentation plan

1. Motivation and background
2. Linear programming
3. Graph colouring
4. Integer programming
5. IP colouring formulations
6. Lego graphs
7. Exciting results
8. Short summary

Presentation plan

1. Motivation and background
2. Linear programming
3. Graph colouring
4. Integer programming
5. IP colouring formulations
6. Lego graphs
7. Exciting results
8. Short summary

25 min.

Presentation plan

1. Motivation and background
2. Linear programming
3. Graph colouring
4. Integer programming
5. IP colouring formulations
6. Lego graphs
7. Exciting results
8. Short summary
9. Questions

25 min.

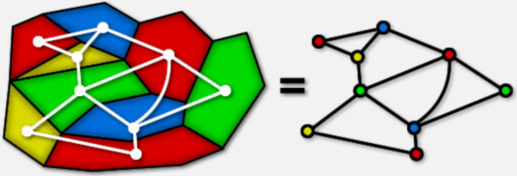
1. Motivation and background

1. Motivation and background

Solving a hard combinatorial problem in an efficient way.

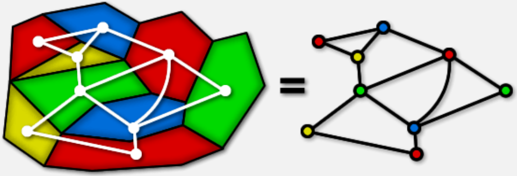
1. Motivation and background

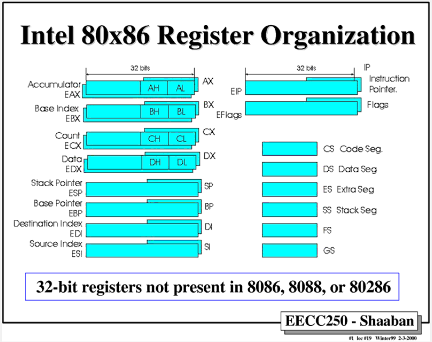
Solving a hard combinatorial problem in an efficient way.



1. Motivation and background

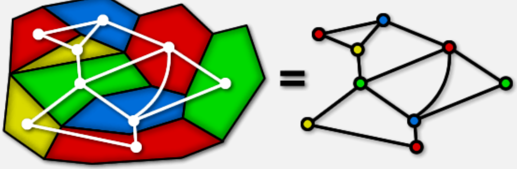
Solving a hard combinatorial problem in an efficient way.

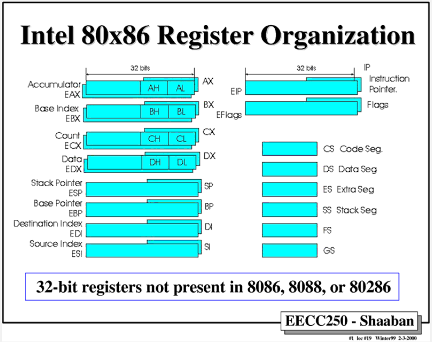




1. Motivation and background

Solving a hard combinatorial problem in an efficient way.





	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
kl. 08:00-10:00	B_1	A_1	C_1	A_1	B_1
kl. 10:00-12:00		A_2			B_2
Middagspause					
kl. 13:00-15:00	C_1	B_2	C_2	A_2	D
kl. 15:00-17:00	C_2				

1. Motivation and background

2. Linear programming

2. Linear programming

Optimizing an objective subject to constraints

2. Linear programming

Optimizing an objective subject to constraints

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

2. Linear programming

Optimizing an objective subject to constraints

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

$$x_1, ..., x_n \in \mathbb{R}$$

2. Linear programming

Optimizing an objective subject to constraints

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

$x_1, ..., x_n \in \mathbb{R}$

These problems are easy and can be solved in polynomial time.

2. Linear programming

Optimizing an objective subject to constraints

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

$$x_1, \ldots, x_n \in \mathbb{R}$$

These problems are easy and can be solved in polynomial time.

Example

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

2. Linear programming

The feasible set and optimal solutions

2. Linear programming

The feasible set and optimal solutions

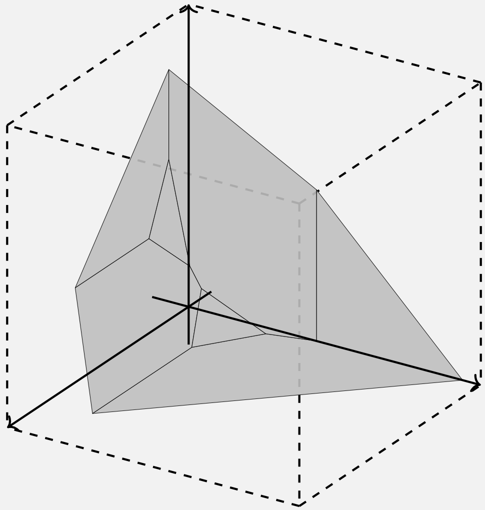


Figure: A graphical repressentation of an LP.

2. Linear programming

The feasible set and optimal solutions

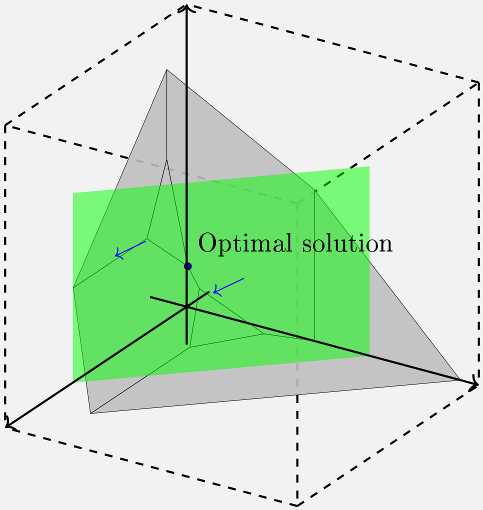


Figure: A graphical repressentation of an LP.

3. Graph colouring

3. Graph colouring

Definition

A ***graph*** *G is a pair (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.*

3. Graph colouring

Definition

A ***graph*** *G is a pair (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

A ***simple graph*** *G is a graph with no edges such that $(v, v) \in E$ for any vertex $v \in V$ and where $(u, v) \in E \Leftrightarrow (v, u) \in E$.*

3. Graph colouring

Definition

*A **graph** G is a pair (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

*A **simple graph** G is a graph with no edges such that $(v, v) \in E$ for any vertex $v \in V$ and where $(u, v) \in E \Leftrightarrow (v, u) \in E$.*

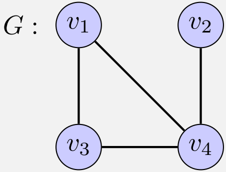


Figure: A visualisation of the simple graph $G = (V, E)$ with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{(v_1, v_3), (v_1, v_4), (v_2, v_4), (v_3, v_4)\}$.

3. Graph colouring

Definition

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

3. Graph colouring

Definition

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

$G:$

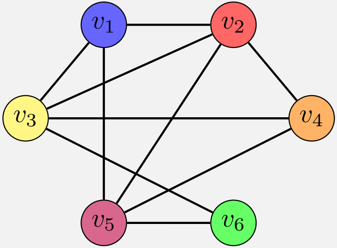


Figure: A graph coloured with the trivial and an optimal vertex colouring.

3. Graph colouring

Definition

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

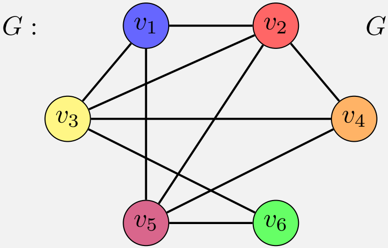


Figure: A graph coloured with the trivial and an optimal vertex colouring.

3. Graph colouring

Definition

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

3. Graph colouring

Definition

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

Theorem

Given a graph G*, with clique-number* $\omega(G)$

$$|V| \geq \chi(G) \geq \omega(G).$$

3. Graph colouring

Definition

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

Theorem

Given a graph G, with clique-number $\omega(G)$

$$|V| \geq \chi(G) \geq \omega(G).$$

Can we do this using linear programming?

4. Integer programming

4. Integer programming

A way of formulating optimization problems with integer values.

4. Integer programming

A way of formulating optimization problems with integer values.

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

4. Integer programming

A way of formulating optimization problems with integer values.

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

$$\exists x_i \in \{x_1, ..., x_n\} : x_i \in \mathbb{Z}$$

4. Integer programming

A way of formulating optimization problems with integer values.

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

$\exists x_i \in \{x_1, ..., x_n\} : x_i \in \mathbb{Z}$

These problems are hard to solve, but can solved in $\mathcal{NP}$-hard problems.

4. Integer programming

The feasible set and optimal solutions

4. Integer programming

The feasible set and optimal solutions

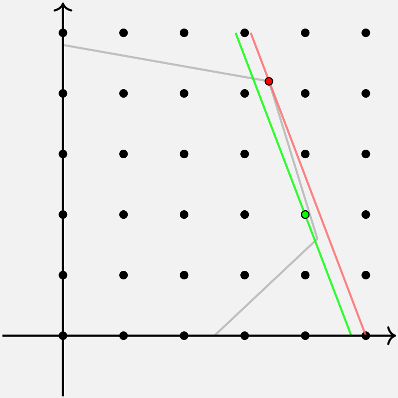


Figure: Figure showing the feasible region of an IP with the optimal solution in green and the optimal solution to the linear relaxation in red.

4. Integer programming

Using MIP to solve problems in graph theory

4. Integer programming

Using MIP to solve problems in graph theory

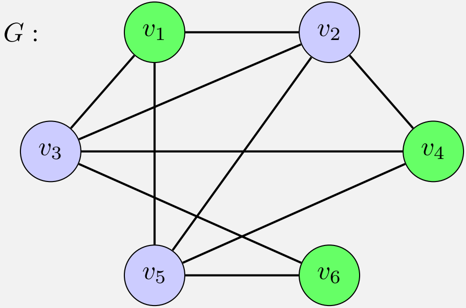


Figure: Shown in green is a maximum independent set in a graph G.

4. Integer programming

Using MIP to solve problems in graph theory

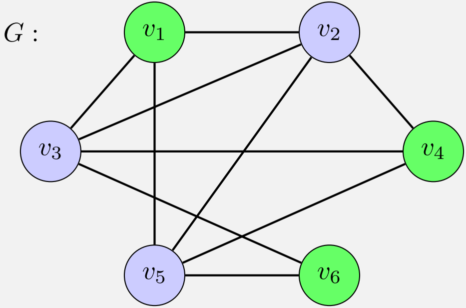


Figure: Shown in green is a maximum independent set in a graph G.

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

4. Integer programming

Some strategies used in integer programming

4. Integer programming

Some strategies used in integer programming

1. Big-M strategy, creating a constraint $x_1 \neq x_2$

4. Integer programming

Some strategies used in integer programming

1. Big-M strategy, creating a constraint $x_1 \neq x_2$

   For $x_1, x_2 \in \mathbb{Z}$, $t \in \{0, 1\}$ and suitable large M, two constraints

$$x_1 - x_2 + Mt \leq M - 1$$

and

$$x_2 - x_1 - Mt \leq -1$$

   ensures that no feasible solution has $x_1 = x_2$.

4. Integer programming

Some strategies used in integer programming

1. Big-M strategy, creating a constraint $x_1 \neq x_2$

   For $x_1, x_2 \in \mathbb{Z}$, $t \in \{0, 1\}$ and suitable large M, two constraints

   $$x_1 - x_2 + Mt \leq M - 1$$

   and

   $$x_2 - x_1 - Mt \leq -1$$

   ensures that no feasible solution has $x_1 = x_2$.

2. A strategy for determining the difference $|x_1 - x_2|$ of binary variables

4. Integer programming

Some strategies used in integer programming

1. Big-M strategy, creating a constraint $x_1 \neq x_2$

   For $x_1, x_2 \in \mathbb{Z}$, $t \in \{0,1\}$ and suitable large M, two constraints

   $$x_1 - x_2 + Mt \leq M - 1$$

   and

   $$x_2 - x_1 - Mt \leq -1$$

   ensures that no feasible solution has $x_1 = x_2$.

2. A strategy for determining the difference $|x_1 - x_2|$ of binary variables

   For $x_1, x_2, z, t \in \{0,1\}$, the constraint

   $$x_1 - x_2 - 2t + z = 0$$

   Ensures that in a feasible solution, $z = |x_1 - x_2|$

5. IP colouring formulations

5. IP colouring formulations

The standard formulation

5. IP colouring formulations

The standard formulation

For any $k \geq \chi(G)$

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

$k|V| + k$ binary variables and $|V| + k|E| + k|V|$ constraints.

5. IP colouring formulations

The scheduling formulation

5. IP colouring formulations

The scheduling formulation

For any $k \geq \chi(G)$

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

$|V| + |E|$ integer and binary variables and $2|E| + |V|$ constraints.

5. IP colouring formulations

The binary formulation

5. IP colouring formulations

The binary formulation

For any $k \geq \chi(G)$, suppose $B = \lceil log_2(k) \rceil$:

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

$|V| \lceil log_2(k) \rceil + 2|E| \lceil log_2(k) \rceil$ binary variables and $|V| + |E| + |E| \lceil log_2(k) \rceil$ constraints.

5. IP colouring formulations

Industrial MIP solvers

5. IP colouring formulations

Industrial MIP solvers

CPLEX

5. IP colouring formulations

Industrial MIP solvers

CPLEX

```
    my_rhs.append(-1.0) #X_v - c =< -1 forall v
    my_sense += "L"
    my_rownames.append("less than highest color "+str(vertex))
for vertex in range(len(graph)):
    for edge in range(vertex):
        if graph[vertex][edge] == 1:
            my_obj.append(0.0)
            my_ub.append(1.0) #x_{u,v} is binary
            my_colnames.append("x"+str(vertex)+","+str(edge))
            my_ctype += "I" #x_{u,v} is binary

            my_rhs.append(my_upperbound-1.0) #X_u - X_v + Kx_{u,v} =< K-1   , K >= c
            my_sense += "L"
            my_rownames.append("1 nonAdj"+str(vertex)+","+str(edge))
            my_rhs.append(-1.0) #X_v - X_u - Kx_{u,v} =< -1
            my_sense += "L"
            my_rownames.append("2 nonAdj"+str(vertex)+","+str(edge))
```

Figure: A section of my code implementing the scheduling formulation using the CPLEX Python-API.

6. Lego graphs

6. Lego graphs

Colouring $a \times b$-brick Lego buildings

6. Lego graphs

Colouring $a \times b$-brick Lego buildings

Constructing graphs to find an upper bound on their chromatic number

6. Lego graphs

Colouring $a \times b$-brick Lego buildings

Constructing graphs to find an upper bound on their chromatic number

d		b		e		c		a	
	a		d		b		e		c
e		c		a		d		b	
	b		e		c		a		d
a		d		b		e		c	
	c		a		d		b		e
b		e		c		a		d	
	d		b		e		c		a
c		a		d		b		e	
	e		c		a		d		b

Figure: The method for finding upper bounds on colours 1×2-brick buildings as developed by AHBS.

6. Lego graphs

Colouring $a \times b$-brick Lego buildings

Constructing graphs to find an upper bound on their chromatic number

d	c'	b	a'	e	d'	c	b'	a	e'
b'	a	e'	d	c'	b	a'	e	d'	c
e	d'	c	b'	a	e'	d	c'	b	a'
c'	b	a'	e	d'	c	b'	a	e'	d
a	e'	d	c'	b	a'	e	d'	c	b'
d'	c	b'	a	e'	d	c'	b	a'	e
b	a'	e	d'	c	b'	a	e'	d	c'
e'	d	c'	b	a'	e	d'	c	b'	a
c	b'	a	e'	d	c'	b	a'	e	d'
a'	e	d'	c	b'	a	e'	d	c'	b

Figure: The method for finding upper bounds on colours 1×2-brick buildings as developed by AHBS.

6. Lego graphs

6. Lego graphs

$G[a, b; c, d]$ Lego graphs:

6. Lego graphs

$G[a, b; c, d]$ Lego graphs:

A vertex at every possible position of a brick.

6. Lego graphs

$G[a, b; c, d]$ Lego graphs:

A vertex at every possible position of a brick.

$2cd$ vertices for $a = b$ and $4cd$ vertices for $a \neq b$.

6. Lego graphs

$G[a, b; c, d]$ Lego graphs:

A vertex at every possible position of a brick.

$2cd$ vertices for $a = b$ and $4cd$ vertices for $a \neq b$.

An edge if two bricks or their periodic translates touch.

6. Lego graphs

$G[a, b; c, d]$ Lego graphs:

A vertex at every possible position of a brick.

$2cd$ vertices for $a = b$ and $4cd$ vertices for $a \neq b$.

An edge if two bricks or their periodic translates touch.

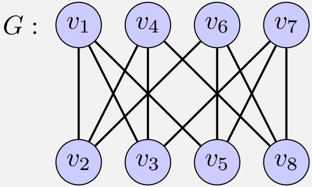


Figure: The graph $G[1, 1; 2, 2]$.

7. Exciting results

7. Exciting results

A comparison of the different formulations

7. Exciting results

A comparison of the different formulations

Graph			Standard			Scheduling			Binary		
Name	$\|V\|$	k	lb	ub	time	lb	ub	time	lb	ub	time
miles1000	128	44	42	42	2s	4	43	30.0m	?	?	30.0m
miles1500	128	73	73	73	21s	4	73	30.0m	?	?	30.0m
miles250	128	9	8	8	0s	8	8	55s	?	?	30.0m
miles500	128	21	20	20	0s	5	20	30.0m	?	?	30.0m
miles750	128	32	31	31	1s	4	31	30.0m	?	?	30.0m
myciel3	11	4	4	4	0s	4	4	0s	4	4	0s
myciel4	23	5	5	5	0s	5	5	0s	5	5	22s
myciel5	47	6	6	6	22s	5	6	30.0m	4	6	30.0m
myciel6	95	7	5	7	30.0m	4	7	30.0m	3	7	30.0m
myciel7	191	8	4	8	30.0m	4	8	30.0m	3	8	30.0m
Q_10_4_3	120	21	8	17	30.0m	4	18	30.0m	?	?	30.0m
Q_10_4_5	252	30	7	27	30.0m	3	30	30.0m	?	?	30.0m
Q_7_4	64	9	8	8	0s	5	8	30.0m	?	?	30.0m
Q_8_2	128	15	8	8	1s	5	8	30.0m	?	?	30.0m
Q_8_4	128	9	8	8	7s	5	8	30.0m	?	?	30.0m
Q_9_2	256	19	9	16	30.0m	3	18	30.0m	?	?	30.4m

Table: A section of the colouring results

7. Exciting results

A comparison of the different formulations

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:
  - The standard formulation: 62% of the test-set

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:
  - The standard formulation: 62% of the test-set
  - The scheduling formulation: 41% of the test-set

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:
  - The standard formulation: 62% of the test-set
  - The scheduling formulation: 41% of the test-set
  - The binary formulation: 19% of the test-set

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:
  - The standard formulation: 62% of the test-set
  - The scheduling formulation: 41% of the test-set
  - The binary formulation: 19% of the test-set
- The scheduling formulations found optimal upper bounds, but struggled to prove the lower bounds.

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:
  - The standard formulation: 62% of the test-set
  - The scheduling formulation: 41% of the test-set
  - The binary formulation: 19% of the test-set
- The scheduling formulations found optimal upper bounds, but struggled to prove the lower bounds.
- The binary formulation struggled to find any integer solutions at all.

7. Exciting results

A comparison of the different formulations

- Proven optimal solutions within 30 minutes:
  - The standard formulation: 62% of the test-set
  - The scheduling formulation: 41% of the test-set
  - The binary formulation: 19% of the test-set
- The scheduling formulations found optimal upper bounds, but struggled to prove the lower bounds.
- The binary formulation struggled to find any integer solutions at all.
- The standard formulation was fastest in most cases, but the scheduling formulation was better in some of the Lego graphs.

7. Exciting results

Lego graph results

7. Exciting results

Lego graph results

Graph			Standard			Scheduling			Binary		
Name	$\|V\|$	k	lb	ub	time	lb	ub	time	lb	ub	time
G_1_2_10_10	400	11	5	5	16.2m	5	5	57s	?	?	30.3m
G_2_2_6_6	72	8	5	5	2s	5	5	1s	5	5	36s
G_3_3_10_10	200	12	6	7	30.0m	4	7	30.0m	?	?	30.0m

Table: The best upper bounds found for 1×2,2×2 and 3×3 brick buildings.

7. Exciting results

Lego graph results

Graph			Standard			Scheduling			Binary		
Name	$\|V\|$	k	lb	ub	time	lb	ub	time	lb	ub	time
G_1_2_10_10	400	11	5	5	16.2m	5	5	57s	?	?	30.3m
G_2_2_6_6	72	8	5	5	2s	5	5	1s	5	5	36s
G_3_3_10_10	200	12	6	7	30.0m	4	7	30.0m	?	?	30.0m

Table: The best upper bounds found for 1×2,2×2 and 3×3 brick buildings.

Graph			Standard			Scheduling		
Name	$\|V\|$	k	lb	ub	time	lb	ub	time
G_1_2_10_12	480	4	5	?	1.9m	5	?	1.4m
G_1_2_12_12	576	4	5	?	5.7m	5	?	2.6m
G_3_3_10_10	200	6	7	?	20.9m	?	?	30.0m
G_3_3_12_12	288	6	7	?	1.5h			

Table: Results from trying to find 4-colourable 1×2, and 6-colourable 3×3 Lego graphs.

8. Short summary

8. Short summary

- Graph colouring is very useful for many different purposes.

8. Short summary

- Graph colouring is very useful for many different purposes.
- Mathematical optimization can be used for colouring graphs, but linear programming is not enough.

8. Short summary

- Graph colouring is very useful for many different purposes.
- Mathematical optimization can be used for colouring graphs, but linear programming is not enough.
- Different IP formulations perform better or worse depending on the graph, but no clear picture of when one outperforms another is visible.

8. Short summary

- Graph colouring is very useful for many different purposes.
- Mathematical optimization can be used for colouring graphs, but linear programming is not enough.
- Different IP formulations perform better or worse depending on the graph, but no clear picture of when one outperforms another is visible.
- No ground breaking new results have been found for colouring Lego buildings, but IP seems to be an efficient approach to finding such results.

8. Short summary

- Graph colouring is very useful for many different purposes.
- Mathematical optimization can be used for colouring graphs, but linear programming is not enough.
- Different IP formulations perform better or worse depending on the graph, but no clear picture of when one outperforms another is visible.
- No ground breaking new results have been found for colouring Lego buildings, but IP seems to be an efficient approach to finding such results.

Thank you for listening.

Questions