An Integer Linear Programming Approach to Graph Coloring

Megan Duff and Rebecca Robinson

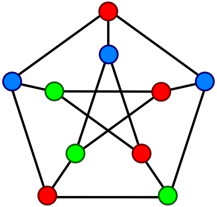
Graph Coloring

Definition (Vertex Coloring of a Graph)

A proper vertex coloring of a graph G is an assignment of colors to the vertices of the graph such that no two adjacent vertices are assigned the same color.

Definition (Chromatic Number)

The minimal number of colors needed to properly vertex color a graph, denoted $\chi(G)$

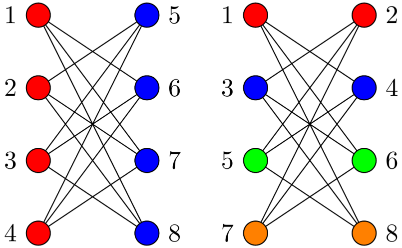


Bounds on Chromatic Number

- Trivial bound: $\chi(G) \leq n$, where $n = |V(G)|$.
- Bound using independence number $\alpha(G)$ (the size of the largest independent set): $\chi(G) \leq \frac{n}{\alpha(G)}$.
- Brooks' Theorem: $\chi(G) \leq \Delta(G)$ as long as G is connected and is not a complete graph or an odd cycle, in which case $\chi(G) = \Delta(G) + 1$.

Greedy Coloring

- Greedy coloring is a method to color a graph
- Label vertices $v_1, v_s, \ldots, v_n$. Color each vertex with lowest color available
- There is always an ordering to give the chromatic number, but there are n! different orderings
- There exist orderings that lead to "bad" colorings



Finding $\chi(G)$ is NP-hard.

Applications of Graph Coloring

- Assigning radio frequencies
  - Draw an edge between two vertices if the corresponding radio stations are too close together so that the frequencies would interfere
- Zoos and pet stores
  - Can't have some animals in the same enclosures, some fish in the same tank, etc
- Resource allocation
  - Create a vertex for each task and connect two vertices when the corresponding tasks require the same resource
  - A proper coloring ensures no two tasks that require the same resource are done at the same time
  - Chromatic number would give the optimal way to do the tasks simultaneously
- Sudoku puzzle solvers

Assignment Integer Linear Program

Consider the following integer linear program:

$$
\begin{array}{lll}
\text{(ASS-S)} & \min \sum_{1 \leq i \leq H} w_i & \\
\text{s.t.} & \sum_{i=1}^{H} x_{vi} = 1 & \forall v \in V \\
 & x_{ui} + x_{vi} \leq w_i & \forall (u,v) \in E,\ i = 1, \ldots, H \\
 & x_{vi}, w_i \in \{0,1\} & \forall v \in V,\ i = 1, \ldots, H
\end{array}
$$

- $x_{vi} = 1$ when vertex v is assigned color i and 0 otherwise.
- $w_i = 1$ if color i is used and 0 otherwise.

Assignment Integer Linear Program

Downfalls of this linear program:

- Exponentially many equivalent solutions
- Continuous relaxation does not work: $x_{v1} = x_{v2} = 0.5$, $x_{vj} = 0 \forall j \in \{3, ..., H\}$, $w_1 = w_2 = 1$, $w_i = 0$ for all other w_i

Additional constraints to "fix" some of these problems:

$$w_i \leq \sum_{v \in V} x_{vi} \qquad i = 1, \dots, H$$

$$w_i \leq w_{i-1} \qquad i = 2, \dots, H$$

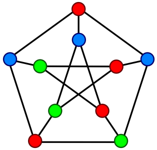
Assignment Integer Linear Program

AMPL's Assignment Code:

$$
\begin{aligned}
\text{(ASS-S)} \quad & \min \textstyle\sum_{1\le i\le H} w_i \\
\text{s.t.} \quad & \textstyle\sum_{i=1}^{H} x_{vi} = 1 && \forall v \in V \\
& x_{ui} + x_{vi} \le w_i && \forall (u,v) \in E,\ i = 1,\dots,H \\
& x_{vi}, w_i \in \{0,1\} && \forall v \in V,\ i = 1,\dots,H
\end{aligned}
$$

```
set V;
set E within (V cross V);
param N;

var w{1..N} binary;
var x{V,1..N} binary;

minimize Colors: sum{i in 1..N} w[i];

subject to Assigned {i in V}:
sum{j in 1..N}x[i,j]=1;

subject to Edges {(i,j) in E, k in 1..N}:
x[i,k] + x[j,k] <= w[k];
```

Assignment Integer Linear Program: Peterson Graph



```
data;

set V:= 1 2 3 4 5 6 7 8 9 10;

set E:= (1,3) (1,4) (1,6) (2,5) (2,4) (2,7)
(3,5) (3,8) (4,9) (5,10) (6,7) (7,8) (8,9) (9,10) (10,6);

param N:=10;
```

```
ampl: model Assignment.mod; data Peterson.dat; option solver CPLEX;
ampl: solve;
CPLEX 12.9.0.0: optimal integer solution; objective 3
153 MIP simplex iterations
0 branch-and-bound nodes
ampl: display x;
x [*,*]
:    1   2   3   4   5   6   7   8   9   10    :=
1    1   0   0   0   0   0   0   0   0    0
2    1   0   0   0   0   0   0   0   0    0
3    0   1   0   0   0   0   0   0   0    0
4    0   1   0   0   0   0   0   0   0    0
5    0   0   1   0   0   0   0   0   0    0
6    0   1   0   0   0   0   0   0   0    0
7    0   0   1   0   0   0   0   0   0    0
8    1   0   0   0   0   0   0   0   0    0
9    0   0   1   0   0   0   0   0   0    0
10   1   0   0   0   0   0   0   0   0    0
;
```

Run Time: 0.102745

Note: 20 different 3 colorings

Assignment Integer Linear Program: Path Graph

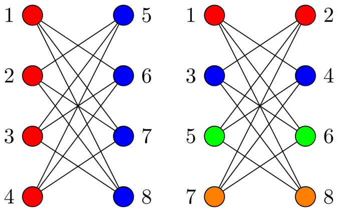
70 vertices path with 69 edges:

```
data;

set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70;

set E:= (1,2) (2,3) (3,4) (4,5) (5,6) (6,7) (7,8) (8,9) (9,10)
(10,11) (11,12) (12,13) (13,14) (14,15) (15,16) (16,17) (17,18) (18,19) (19,20)
(20,21) (21,22) (22,23) (23,24) (24,25) (25,26) (26,27) (27,28) (28,29) (29,30)
(30,31) (31,32) (32,33) (33,34) (34,35) (35,36) (36,37) (37,38) (38,39) (39,40)
(40,41) (41,42) (42,43) (43,44) (44,45) (45,46) (46,47) (47,48) (48,49) (49,50)
(50,51) (51,52) (52,53) (53,54) (54,55) (55,56) (56,57) (57,58) (58,59) (59,60)
(60,61) (61,62) (62,63) (63,64) (64,65) (65,66) (66,67) (67,68) (68,69) (69,70);

param N:= 70;
```

Vertices	Edges	Run Time
70	69	0.606761
140	139	2.16679
200	199	7.1204

Assignment Integer Linear Program: Crown Graph

Vertices	Edges	Run Time
60	30*29=870	19.8884



Assignment Integer Linear Program: Mycielski Graph of Order 6

A Mycielski graph M_k of order k is a triangle-free graph with chromatic number k having the smallest possible number of vertices.

- There exists triangle free graphs with large chromatic numbers.

```
set V:= 0 1 2 3 4 5 6 7 8 9
 10 11 12 13 14 15 16 17 18 19
 20 21 22 23 24 25 26 27 28 29
  30 31 32 33 34 35 36 37 38 39
 40 41 42 43;
set E:= (0,1) (0,4) (0,6) (0,9) (0,13) (0,16) (0,22) (0,23) (0,26) (0,27)
 (0,30) (0,34) (0,37) (1,2) (1,5) (1,7) (1,12) (1,14) (1,17) (1,18)
 (1,24) (1,28) (1,33) (1,35) (1,38) (1,39) (2,3) (2,6) (2,8) (2,13)
 (2,15) (2,19) (2,23) (2,25) (2,27) (2,29) (2,34) (2,36) (2,40) (3,4)
 (3,7) (3,9) (3,14) (3,16) (3,18) (3,24) (3,26) (3,28) (3,30) (3,35)
 (3,37) (3,39) (4,5) (4,8) (4,12) (4,15) (4,17) (4,19) (4,25) (4,29)
 (4,33) (4,36) (4,38) (4,40) (5,10) (5,13) (5,16) (5,20) (5,22) (5,23)
 (5,26) (5,31) (5,34) (5,37) (5,41) (6,10) (6,11) (6,12) (6,14) (6,20)
 (6,24) (6,31) (6,32) (6,33) (6,35) (6,41) (7,10) (7,13) (7,15) (7,20)
 (7,23) (7,25) (7,31) (7,34) (7,36) (7,41) (8,10) (8,14) (8,16) (8,20)
 (8,24) (8,26) (8,31) (8,35) (8,37) (8,41) (9,10) (9,11) (9,12) (9,15)
 (9,20) (9,25) (9,31) (9,32) (9,33) (9,36) (9,41) (10,17) (10,18) (10,19)
 (10,27) (10,28) (10,29) (10,30) (10,38) (10,39) (10,40) (11,13) (11,16) (11,17)
 (11,18) (11,19) (11,27) (11,30) (11,34) (11,37) (11,38) (11,39) (11,40) (12,21)
 (12,23) (12,26) (12,27) (12,30) (12,42) (13,21) (13,24) (13,28) (13,32) (13,42)
 (14,21) (14,23) (14,25) (14,27) (14,29) (14,42) (15,21) (15,24) (15,26) (15,28)
 (15,30) (15,42) (16,21) (16,25) (16,29) (16,32) (16,42) (17,21) (17,23) (17,26)
 (17,31) (17,32) (17,42) (18,21) (18,23) (18,25) (18,31) (18,32) (18,42) (19,21)
 (19,24) (19,26) (19,31) (19,32) (19,42) (20,21) (20,27) (20,28) (20,29) (20,30)
 (20,42) (21,33) (21,34) (21,35) (21,36) (21,37) (21,38) (21,39) (21,40) (21,41)
 (22,24) (22,25) (22,28) (22,29) (22,32) (22,33) (22,35) (22,36) (22,38) (22,39)
 (22,40) (22,42) (23,43) (24,43) (25,43) (26,43) (27,43) (28,43) (29,43) (30,43)
 (31,43) (32,43) (33,43) (34,43) (35,43) (36,43) (37,43) (38,43) (39,43) (40,43)
 (41,43),(42,43);

param N:=44;
```

44 vertices and 232 edges:
Run Time: 338.305 s

Set Covering Integer Linear Program

Definition (Independent Set)

A set of vertices are independent if no two vertices in the set are adjacent

Note: Minimal number of independent sets = Chromatic number
Consider the following Integer Linear Program which finds the minimal number of independent sets:

$$
\begin{aligned}
\text{(COV)} \quad & \min \textstyle\sum_{s \in S} x_s \\
\text{s.t.} \quad & \textstyle\sum_{s \in S : v \in s} x_s \geq 1 \quad \forall v \in V \\
& x_s \in \{0, 1\} \quad \forall s \in S
\end{aligned}
$$

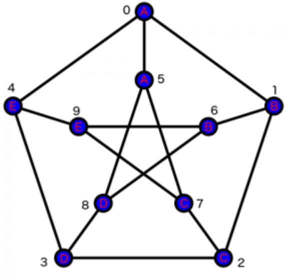
- S : family of independent sets, $x_s = 1$ if the independent set $s \in S$ is used to cover the vertices

Downfalls:

- Exponential number of variables

Set Covering Integer Linear Program

Some of independent sets:

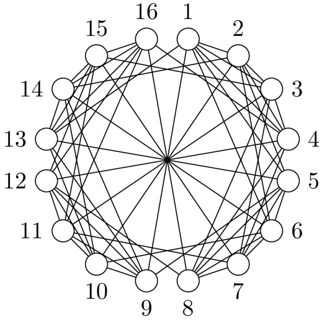


{0},{1},{2},{3},{4},{5},{6},{7},{8},{9}, {0,2}, {0,3}, {0,6}, {0,9}, {0,7}, {0,8}, {1,4}, {1,3}, {1,5}, {1,8},{1,9},{1,5},{1,7},{2,4}, {2,8}, {2,9}, {2,5}, {2,6}, {3,6},{3,7},{3,5},{3,9},{4,5},{4,6},{4,7},{4,8}, {5,6}, {5,9}, {6,7}, {7,8}, {0,2,9}, {0,2,8}, {0,2,6}, {0,3,7}, {0,3,9}, {0,3,6}, etc.
(46+ independent sets)
Downfalls: How do you know you have every independent set covered?

Sudoku as a Graph Coloring Problem

- We consider a 4x4 Sudoku puzzle.
- Label grid squares in Sudoku puzzle and construct corresponding graph by having two vertices be adjacent if:
  - they are in the same row
  - they are in the same column
  - they are in the same 2x2 subgrid
- A proper coloring of the graph translates to a solution to the Sudoku puzzle.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



Sudoku as a Graph Coloring Problem

In Sudoku, some of the grid squares are already filled in. We have to account for this in the program.

```
subject to ExistingColors {(i,j) in C}:

x[i,j] = 1;
```

```
set C := (2,3) (5,1) (8,2) (11,4) (14,1);
```

	3		
1			2
		4	
	1		

Sudoku as a Graph Coloring Problem

```
ampl: model 4x4model.mod; data 4x4data.dat; solve;
Solution determined by presolve;
objective Colors = 4.
ampl: display x;
x [*,*]
:     1   2   3   4    :=
1     0   1   0   0
2     0   0   1   0
3     1   0   0   0
4     0   0   0   1
5     1   0   0   0
6     0   0   0   1
7     0   0   1   0
8     0   1   0   0
9     0   0   1   0
10    0   1   0   0
11    0   0   0   1
12    1   0   0   0
13    0   0   0   1
14    1   0   0   0
15    0   1   0   0
16    0   0   1   0
;
```

2	3	1	4
1	4	3	2
3	2	4	1
4	1	2	3

This program can easily be extended to the 9x9 case simply by changing the edge set to be the edges of the 9x9 graph.

References

Droogendijk, Leen. “A Triangle-Free, 6-Chromatic Graph with 44 Vertices.” Mathematics Stack Exchange, 1 Feb. 1966, https://math.stackexchange.com/questions/1561029/a-triangle-free-6-chromatic-graph-with-44-vertices.

Dhaharavath, Bharathi. Sudoku Game Solution Using Graph Coloring. http://www.cs.kent.edu/ dragan/ST-Spring2016/SudokuGC.pdf.

“Independent Set (Graph Theory).” Wikipedia, Wikimedia Foundation, 25 Nov. 2019, https://en.wikipedia.org/wiki/Independent$_{s}et_{(}graph_{t}heory).$

Jabrayilov, Adalat, and Petra Mutzel. New Integer Linear Programming Models for the Vertex Coloring Problem . 11 July 2007, https://arxiv.org/pdf/1706.10191.pdf.

“Graph Coloring.” Wikipedia, Wikimedia Foundation, 12 Nov. 2019, https://en.wikipedia.org/wiki/Graph$_{c}oloring.$