Column Generation, Dantzig-Wolfe, Branch-Price-and-Cut

Q & A and Exercise Session

Marco Lübbecke · OR Group · RWTH Aachen University, Germany

@mluebbecke



Operations Research

RWTH AACHEN UNIVERSITY

This Exercise

1. a theoretic part
2. a more practical part

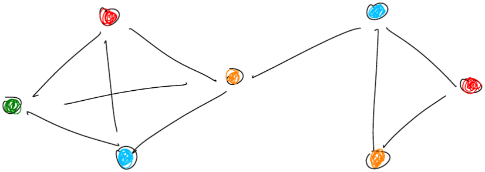
Reminder: Vertex Coloring Problem

Data

$G = (V, E)$ undirected graph

Goal

color all vertices such that adjacent vertices receive different colors, minimizing the number of used colors



Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

$$x_{ic} \in \{0,1\} \quad i \in V, c \in C$$ // color vertex i with c?

Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

s.t. $$\sum_{c \in C} x_{ic} = 1 \qquad i \in V \qquad \text{// color each vertex}$$

$$x_{ic} \in \{0,1\} \qquad i \in V,\ c \in C \qquad \text{// color vertex } i \text{ with } c\text{?}$$

Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{c \in C} x_{ic} = 1 && i \in V && \text{// color each vertex} \\
& x_{ic} + x_{jc} \le 1 && ij \in E,\ c \in C && \text{// avoid conflicts} \\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C && \text{// color vertex } i \text{ with } c\text{?}
\end{aligned}
$$

Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{c \in C} x_{ic} = 1 && i \in V && \text{// color each vertex} \\
& x_{ic} + x_{jc} \leq 1 && ij \in E,\ c \in C && \text{// avoid conflicts} \\
& x_{ic} \leq y_c && i \in V,\ c \in C && \text{// couple } x \text{ and } y \text{ variables} \\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C && \text{// color vertex } i \text{ with } c\text{?} \\
& y_c \in \{0,1\} && c \in C && \text{// do we use color } c\text{?}
\end{aligned}
$$

Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

$$\begin{aligned}
\chi(G) = \min \quad & \sum_{c\in C} y_c && \text{// minimize number of used colors}\\
\text{s.t.} \quad & \sum_{c\in C} x_{ic} = 1 && i \in V && \text{// color each vertex}\\
& x_{ic} + x_{jc} \leq 1 && ij \in E,\ c \in C && \text{// avoid conflicts}\\
& x_{ic} \leq y_c && i \in V,\ c \in C && \text{// couple } x \text{ and } y \text{ variables}\\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C && \text{// color vertex } i \text{ with } c\text{?}\\
& y_c \in \{0,1\} && c \in C && \text{// do we use color } c\text{?}
\end{aligned}$$

- $\chi(G)$ is called the *chromatic number of G*.

Vertex Coloring: A Compact Integer Program

- notation: C set of available colors

$$
\begin{aligned}
\chi(G) = \min \quad & \sum_{c \in C} y_c && && \text{// minimize number of used colors} \\
\text{s.t.} \quad & \sum_{c \in C} x_{ic} = 1 && i \in V && \text{// color each vertex} \\
& x_{ic} + x_{jc} \leq y_c && ij \in E,\ c \in C && \text{// avoid conflicts} \\
& x_{ic} \leq y_c && i \in V,\ c \in C && \text{// couple } x \text{ and } y \text{ variables} \\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C && \text{// color vertex } i \text{ with } c\text{?} \\
& y_c \in \{0,1\} && c \in C && \text{// do we use color } c\text{?}
\end{aligned}
$$

- $\chi(G)$ is called the *chromatic number of G*.
- alternative linking of variables x and y possible

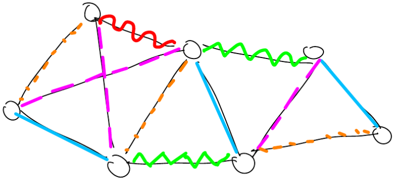
Similar: Edge Coloring Problem

Data

$G = (V, E)$ undirected graph

Goal

color all *edges* such that *incident edges* receive different colors, minimizing the number of used colors



It is your Turn!

- formulate the edge coloring problem as a compact integer program

Edge Coloring: A Compact Integer Program

- notation: $\delta(i) = \{\{i,j\} \mid \{i,j\} \in E\}$ edges incident with $i \in V$

Edge Coloring: A Compact Integer Program

- notation: $\delta(i) = \{\{i,j\} \mid \{i,j\} \in E\}$ edges incident with $i \in V$

$$x_{ec} \in \{0,1\} \quad e \in E, c \in C$$ // color edge e with c?

Edge Coloring: A Compact Integer Program

- notation: $\delta(i) = \{\{i,j\} \mid \{i,j\} \in E\}$ edges incident with $i \in V$

s.t. $\sum_{c \in C} x_{ec} = 1 \qquad e \in E$ // color each edge

$x_{ec} \in \{0,1\} \qquad e \in E,\ c \in C$ // color edge e with c?

Edge Coloring: A Compact Integer Program

- notation: $\delta(i) = \{\{i,j\} \mid \{i,j\} \in E\}$ edges incident with $i \in V$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{c \in C} x_{ec} = 1 && e \in E && \text{// color each edge} \\
& \sum_{e \in \delta(i)} x_{ec} \leq y_c && i \in V,\ c \in C && \text{// avoid conflicts} \\
& x_{ec} \in \{0,1\} && e \in E,\ c \in C && \text{// color edge } e \text{ with } c\text{?}
\end{aligned}
$$

Edge Coloring: A Compact Integer Program

- notation: $\delta(i) = \{\{i,j\} \mid \{i,j\} \in E\}$ edges incident with $i \in V$

$$
\begin{array}{llll}
\text{s.t.} & \displaystyle\sum_{c \in C} x_{ec} = 1 & e \in E & \text{// color each edge} \\
& \displaystyle\sum_{e \in \delta(i)} x_{ec} \leq y_c & i \in V,\ c \in C & \text{// avoid conflicts} \\
& x_{ec} \in \{0,1\} & e \in E,\ c \in C & \text{// color edge } e \text{ with } c\text{?} \\
& y_c \in \{0,1\} & c \in C & \text{// do we use color } c\text{?}
\end{array}
$$

Edge Coloring: A Compact Integer Program

- notation: $\delta(i) = \{\{i,j\} \mid \{i,j\} \in E\}$ edges incident with $i \in V$

$$
\begin{array}{rlll}
\chi'(G) = \min & \sum_{c \in C} y_c & & \text{// minimize number of used colors} \\
\text{s.t.} & \sum_{c \in C} x_{ec} = 1 & e \in E & \text{// color each edge} \\
& \sum_{e \in \delta(i)} x_{ec} \leq y_c & i \in V,\ c \in C & \text{// avoid conflicts} \\
& x_{ec} \in \{0,1\} & e \in E,\ c \in C & \text{// color edge } e \text{ with } c\text{?} \\
& y_c \in \{0,1\} & c \in C & \text{// do we use color } c\text{?}
\end{array}
$$

- $\chi'(G)$ is called the *chromatic index of G*.

Reminder: Dantzig-Wolfe Reformulation

$$
\begin{aligned}
\min \quad & \sum_{c \in C} y_c \\
\text{s.t.} \quad & \sum_{c \in C} x_{ic} = 1 && i \in V \\
& x_{ic} + x_{jc} \leq y_c && ij \in E,\ c \in C \\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C \\
& y_c \in \{0,1\} && c \in C
\end{aligned}
$$

Reminder: Dantzig-Wolfe Reformulation

$$
\begin{array}{llll}
\min & \sum_{c \in C} y_c & & \\
\text{s.t.} & \sum_{c \in C} x_{ic} = 1 & i \in V & \\
& x_{ic} + x_{jc} \leq y_c & ij \in E,\ c \in C & \text{// reformulate} \\
& x_{ic} \in \{0,1\} & i \in V,\ c \in C & \text{// reformulate} \\
& y_c \in \{0,1\} & c \in C & \text{// reformulate}
\end{array}
$$

Reminder: Dantzig-Wolfe Reformulation

$$\begin{aligned}
\min \quad & \sum_{c \in C} y_c \\
\text{s.t.} \quad & \sum_{c \in C} x_{ic} = 1 && i \in V \\
& x_{ic} + x_{jc} \leq y_c && ij \in E,\ c \in C && \text{// reformulate} \\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C && \text{// reformulate} \\
& y_c \in \{0,1\} && c \in C && \text{// reformulate}
\end{aligned}$$

- consider

$$X_c = \operatorname{conv}\left\{x_{ic} \in \{0,1\}, i \in V,\ y_c \in \{0,1\} \mid x_{ic} + x_{jc} \leq y_c, ij \in E\right\}, \qquad c \in C$$

the convex hull of incidence vectors of stable sets in G in color c

Reminder: Dantzig-Wolfe Reformulation

$$X_c = \text{conv}\left\{x_{ic} \in \{0,1\}, i \in V,\ y_c \in \{0,1\} \mid x_{ic} + x_{jc} \leq y_c, ij \in E\right\}, \qquad c \in C$$

- we express every $\begin{pmatrix} \mathbf{x}_c \\ y_c \end{pmatrix} \in X_c$ as convex combination of extreme points of X_c

  // by construction we know that these extreme points are incidence vectors of stable sets

$$\begin{pmatrix} x_{1c} \\ \vdots \\ x_{|V|c} \\ y_c \end{pmatrix} = \sum_{q \in Q^c} \begin{pmatrix} x_{1cq} \\ \vdots \\ x_{|V|cq} \\ y_{cq} \end{pmatrix} \cdot \lambda_q^c, \qquad \sum_{q \in Q^c} \lambda_q^c = 1,\ \ \lambda_q^c \geq 0, q \in Q^c, \quad c \in C$$

- that is, we can replace $x_{ic} = \sum_{q \in Q^c : i \in q} \lambda_q^c$ and $y_c = 1 \iff \mathbf{x}_q \neq \mathbf{0}$

Reminder: Dantzig-Wolfe Reformulation

- we can replace $x_{ic} = \sum_{q \in Q^c : i \in q} \lambda^c_q$ and $y_c = 1 \iff \mathbf{x}_q \neq \mathbf{0}$ in the “master” constraints

$$
\begin{aligned}
\min \quad & \sum_{c \in C} y_c \\
\text{s.t.} \quad & \sum_{c \in C} x_{ic} = 1 && i \in V \\
& x_{ec} \geq 0 && e \in E,\, c \in C \\
& y_c \geq 0 && c \in C
\end{aligned}
$$

Reminder: Dantzig-Wolfe Reformulation

- we can replace $x_{ic} = \sum_{q \in Q^c : i \in q} \lambda_q^c$ and $y_c = 1 \iff \mathbf{x}_q \neq \mathbf{0}$ in the "master" constraints

$$\min \sum_{c \in C} \sum_{q \in Q^c : \mathbf{x}_q \neq \mathbf{0}} \lambda_q^c \qquad \text{// minimize number of (non-empty) stable sets}$$

$$\text{s.t.} \sum_{c \in C} \sum_{q \in Q^c : i \in q} \lambda_q^c = 1 \quad i \in V \qquad \text{// every vertex must appear in exactly one stable set}$$

$$\sum_{q \in Q^c} \lambda_q^c = 1 \quad c \in C \qquad \text{// exactly one stable set per color, could be } \textit{empty}!$$

$$\lambda_q^c \geq 0 \quad q \in Q^c$$

It is your Turn!

- review and understand the above Dantzig-Wolfe reformulation
- think about what would change on the edge coloring model

Operations Research | RWTH AACHEN UNIVERSITY

Dantzig-Wolfe Reformulation for Edge Coloring

- we consider the convex hull of incidence vectors of matchings in G in color c

$$X'_c = \operatorname{conv}\left\{x_{ec} \in \{0,1\}, e \in E,\ y_c \in \{0,1\} \mid \sum_{e \in \delta(i)} x_{ec} \leq y_c,\ i \in V\right\}, \qquad c \in C$$

Dantzig-Wolfe Reformulation for Edge Coloring

- we consider the convex hull of incidence vectors of matchings in G in color c

$$X'_c = \operatorname{conv}\Big\{x_{ec} \in \{0,1\}, e \in E,\ y_c \in \{0,1\} \mid \sum_{e \in \delta(i)} x_{ec} \leq y_c,\ i \in V\Big\}, \qquad c \in C$$

- the formulas look very similar, except that we talk about edges and matchings
- replace $x_{ec} = \sum_{q \in Q^c : e \in q} \lambda_q^c$ and $y_c = 1 \iff \mathbf{x}_q \neq \mathbf{0}$ in the "master" constraints

The Restricted Master Problem for Edge Coloring

- the restricted master problem also looks similar
- Q_c index set of matchings in G in color c, $c \in C$

$$\min \sum_{c \in C} \sum_{q \in Q^c : \mathbf{x}_q \neq \mathbf{0}} \lambda_q^c \quad \text{// minimize number of (non-empty) matchings}$$

$$\text{s.t.} \sum_{c \in C} \sum_{q \in Q^c : e \in q} \lambda_q^c = 1 \quad e \in E \quad \text{// every edge must appear in exactly one matching}$$

$$\sum_{q \in Q^c} \lambda_q^c = 1 \quad c \in C \quad \text{// exactly one matching per color, could be } \textit{empty}!$$

$$\lambda_q^c \geq 0 \quad q \in Q^c$$

Reminder: Column Generation: Vertex Coloring

- solving the restricted master problem gives an optimal dual solution $\{\pi_i\}_{i\in V}, \{\sigma_c\}_{c\in C}$
- the pricing problem is to minimize the reduced cost over X_c

$$
\begin{aligned}
\min \quad & \sum_{c\in C} y_c - \sum_{i\in V} \pi_i x_{ic} - \sum_{c\in C} \sigma_c \\
\text{s.t.} \quad & x_{ic} + x_{jc} \leq y_c && ij \in E,\ c \in C \\
& x_{ic} \in \{0,1\} && i \in V,\ c \in C \\
& y_c \in \{0,1\} && c \in C
\end{aligned}
$$

- one realizes that this decomposes into $|C|$ independent problems

It is your Turn!

- think about how the pricing problem for the edge coloring problem looks like

Column Generation for Edge Coloring

- solving the restricted master problem gives an optimal dual solution $\{\pi_e\}_{e\in E}, \{\sigma_c\}_{c\in C}$
- the pricing problem is to minimize the reduced cost over X'_c

$$
\begin{aligned}
\min \quad & \sum_{c\in C} y_c - \sum_{e\in E} \pi_e x_{ec} - \sum_{c\in C} \sigma_c \\
\text{s.t.} \quad & \sum_{e\in\delta(i)} x_{ec} \le y_c && i \in V,\ c \in C \\
& x_{ec} \in \{0,1\} && e \in E,\ c \in C \\
& y_c \in \{0,1\} && c \in C
\end{aligned}
$$

- one realizes that this decomposes into $|C|$ independent problems

Do we really need to know all this?

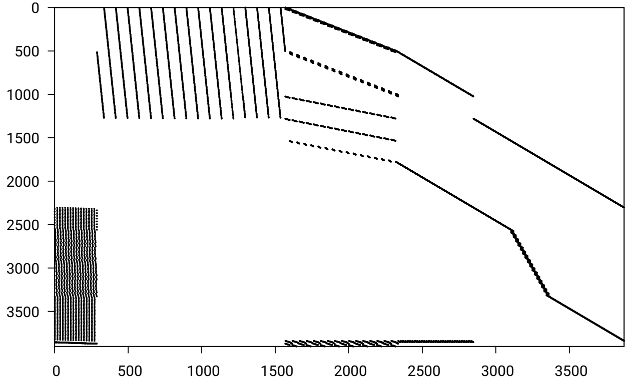
- well, yes and no

What is GCG?

- a branch-price-and-cut solver, based on SCIP
- reads MIP, performs DW reformulation, does BP&C
- pricing: MIP/specialized, heuristic/exact, parallel, ...
- branching: original/Ryan-Foster/generic
- cuts from original: combinatorial/from basis
- primal heuristics: original/master/mixed
- stabilization, early branching, ...
- no modeling language/user interaction required
- download: `gcg.or.rwth-aachen.de` or `scipopt.org`

What is necessary to make this work?

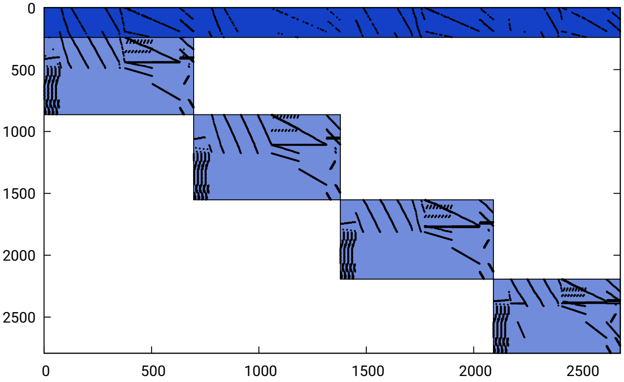
- coefficient matrices of integer programs



b2c1s1

What is necessary to make this work?

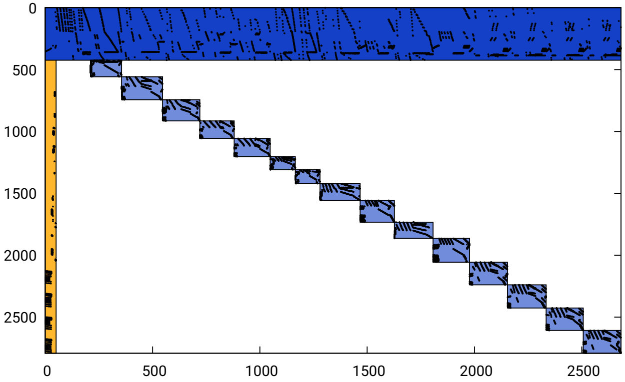
- coefficient matrices of integer programs



b2c1s1 (with 4 blocks)

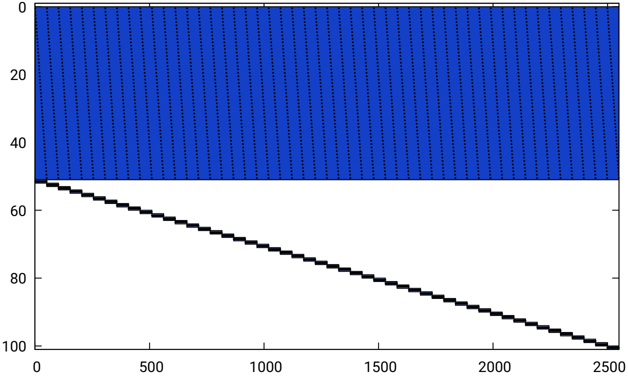
What is necessary to make this work?

- coefficient matrices of integer programs



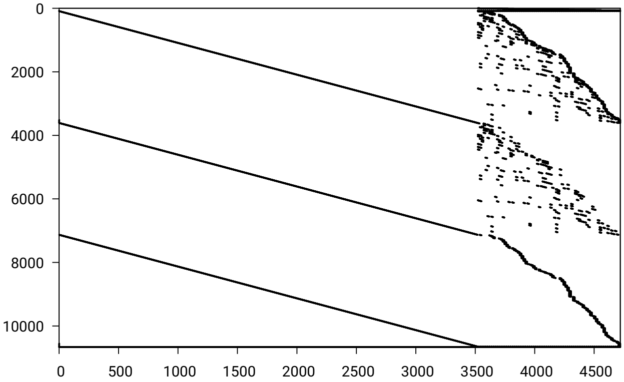
b2c1s1 (with 15 blocks)

For Textbook Models: Matrix Adjacency not ideal



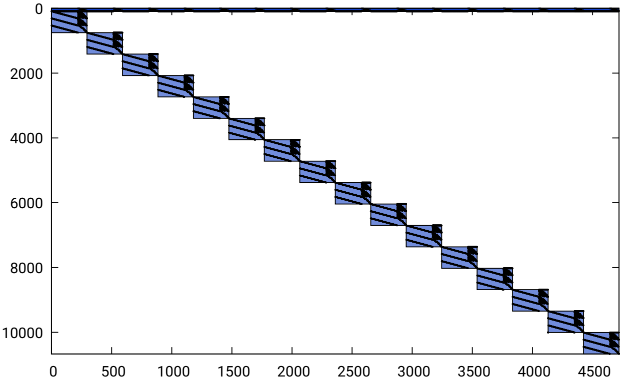
cpmp p2050-1 (with 50 blocks)

What did they have in Mind?



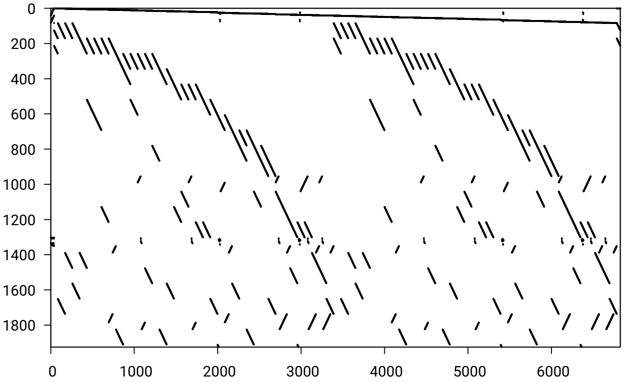
ns1778858

What did they have in Mind?



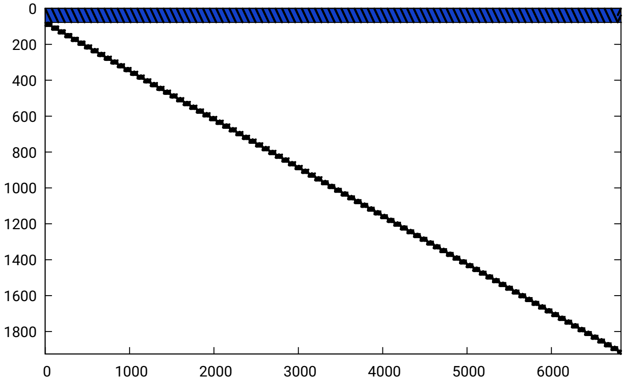
ns1778858 (with 16 blocks)

Number of Blocks?



ns4-pr9

Number of Blocks?



ns4-pr9 (with 88 blocks)

GCG “Detectors” detect Model Structure

- these pictures are automatically generated by GCG
- so-called “detectors” identify certain structures like “blocks”

It is your Turn!

let us invoke GCG

1. open a terminal

   // click new → terminal

2. call GCG from the terminal

   // just type `gcg`

It is your Turn!

- GCG can solve integer programs, just like SCIP
- you can read and optimize `LP` and `MPS` (and other) files

- try to `read bpp-2001-150.lp`
- then `optimize` the model
- you can also `display solution` afterwards
- if you have seen enough you can `quit` — just as in SCIP

! you can even invoke SCIP on the command line and compare the performance!

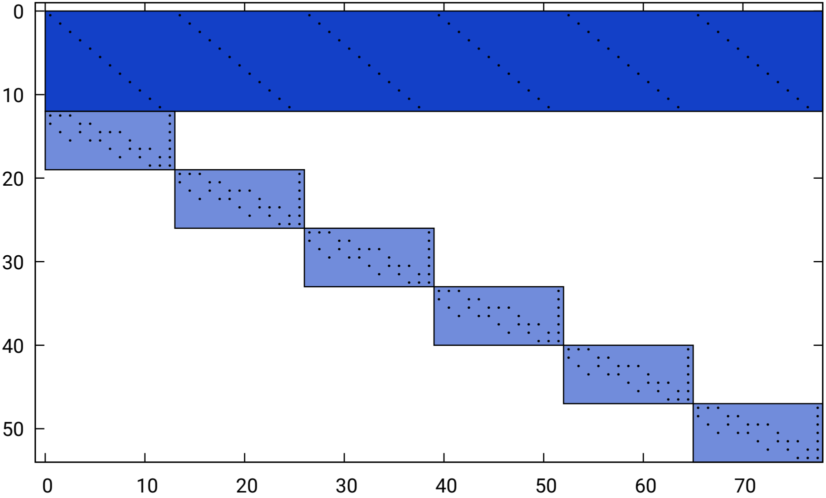
ZIMPL Model for Edge Coloring: edge_coloring.zpl

```
# definitions of sets V, E, C, and delta[i] are not displayed here

# use color c for edge <i,j> in E?
var x[E*C] binary;
# use color c?
var y[C] binary;

minimize cost: sum<c> in C: y[c];

# every edge must receive exactly one color
subto assign:
      forall <i,j> in E: sum<c> in C: x[i,j,c] == 1;

# edges incident to a vertex must cannot receive the same color
subto conflict:
      forall <i,c> in V*C: sum<j,k> in delta[i]: x[j,k,c] <= y[c];
```

It is your Turn!

- start GCG from the terminal, then
- `read edge_coloring.zpl`
- `optimize`

GCG can visualize the Matrix/Model Structure



It is your Turn!

1. start GCG from the terminal, then
2. `read edge_coloring.zpl`
3. (optionally: `presolve`)
4. `detect`

- this has invoked the detection manually, now you can check what GCG has found:

5. `explore`

- there are many “decompositions”...

The Explore Menu

```
Summary            presolved   original
                   ---------   --------
detected                0          20
user given (partial)    0           0
user given (full)       0           0
============================================================================
  id  nbloc nmacon nlivar nmavar nstlva spfwh history pre nopcon nopvar usr sel
 ---- ----- ------ ------ ------ ------ ----- ------- --- ------ ------ --- ---
   0     6     12      0      0      0 0.8241    cC  no      0      0  no  no
   1     6     18      0      0      0 0.2778    cC  no      0      0  no  no
   2     6     24      0      0      0 0.2315    cC  no      0      0  no  no
   3     6     30      0     18      0 0.1937    cC  no      0      0  no  no
   4     6     36      0     24      0 0.1474    cC  no      0      0  no  no
   5     6     36      0     24      0 0.1474    cC  no      0      0  no  no
   6    18      0     72      0      0 0.1293    vC  no      0      0  no  no
   7     8     36      0      0      0 0.1147    cC  no      0      0  no  no
   8     6     30      0      0      0 0.1104    cC  no      0      0  no  no
   9    12     42      0      6      0 0.1026    cC  no      0      0  no  no
============================================================================
```

- new GCG 3.1.0, the explore menu offers e.g., sorting facilities

The Explore Menu

- currently, the most interesting functionality is to browse the list
- enter `help` for available commands
- I often use `visualize`, this produces a PDF

It is your Turn!

- at the moment, the visualization relies on gnuplot and a pdf viewer
- you cannot start a pdf viewer today but you can download (and view) the pdf

Influencing the Detection

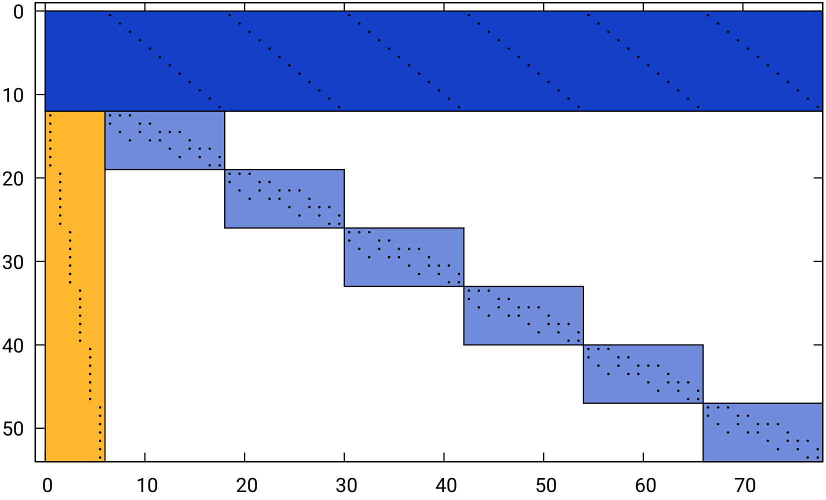
- there are many settings which influence the detection
- unfortunately, most of this is not yet documented
- the current best bet is to

→ read the SCIP 6.0 release report, chapter 5
`opus4.kobv.de/opus4-zib/files/6936/scipopt-60.pdf`

→ check the ("internal") GCG 3.1.0 documentation
`gcg.or.rwth-aachen.de/dev/`

→ get in touch with us dev team

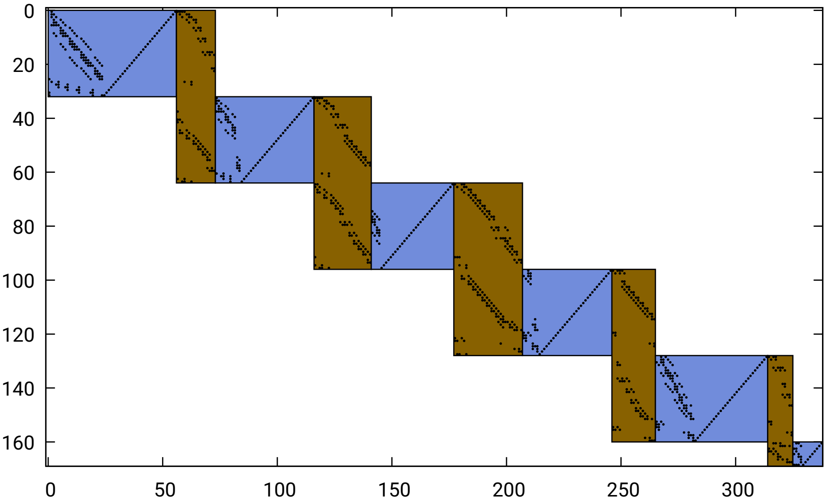
GCG can visualize the Matrix/Model Structure



It is your Turn!

- start GCG from the terminal, then
- `read enlight13.mps.gz`
- `set detection detectors stairheur enabled TRUE`
  // this enables a “staircase” detector
- `presolve`
- `detect`
- `explore`
- `visualize 0`

GCG can visualize a Staircase Structure



Operations Research
RWTH AACHEN UNIVERSITY

You can provide the Decomposition Information

- in addition to an `LP` or `MPS` file you can give a `DEC` file
- it essentially contains the information which constraint will take which role

$\rightarrow$ "block" or "master"

It is your Turn!

- download and inspect the file `edge_coloring.lp`
- download and inspect the file `edge_coloring-cC-27-6-dec.dec`
- start GCG from the terminal, then
- `read edge_coloring.lp`
- `read edge_coloring-cC-27-6-dec.dec`
  // you can create such DEC files yourself when GCG does not detect “your” structure
- `explore`
- `visualize 0`
- `optimize`

Stay in Touch

- if you have use cases, comments, questions, wishes, ...
- contact us `@mluebbecke`, `luebbecke@or.rwth-aachen.de`
- we also listen on the SCIP mailing list