Exact Graph Coloring via Hybrid Approaches

S.Gualandi, F.Malucelli

Dipartimento di Elettronica e Informazione,
Politecnico di Milano, Milano, Italy

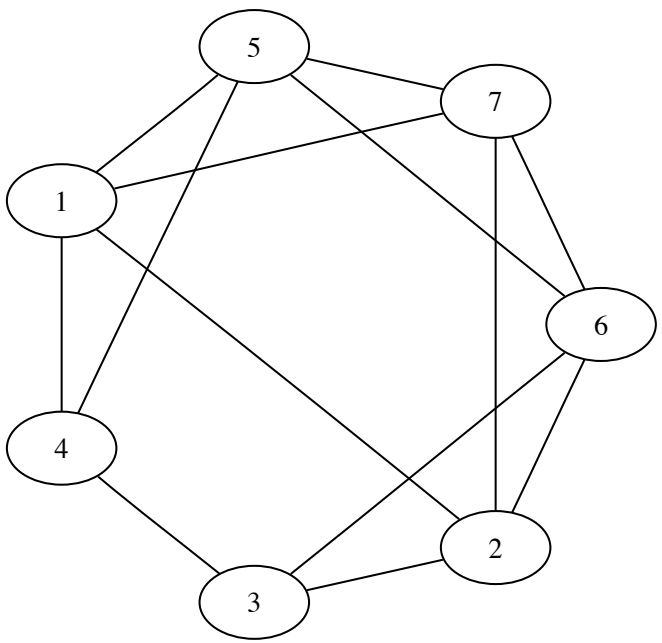
May 13, 2008

Outline

- Graph Coloring
- Formulations
  - Classical Formulation
  - Column Generation approach
  - Semidefinite Programming
  - Constraint Programming
- Exact Hybrid Approaches
  - Constraint Programming-based Column Generation
  - Exploiting SDP relaxations into Constraint Programming
- Computational results
- Conclusions

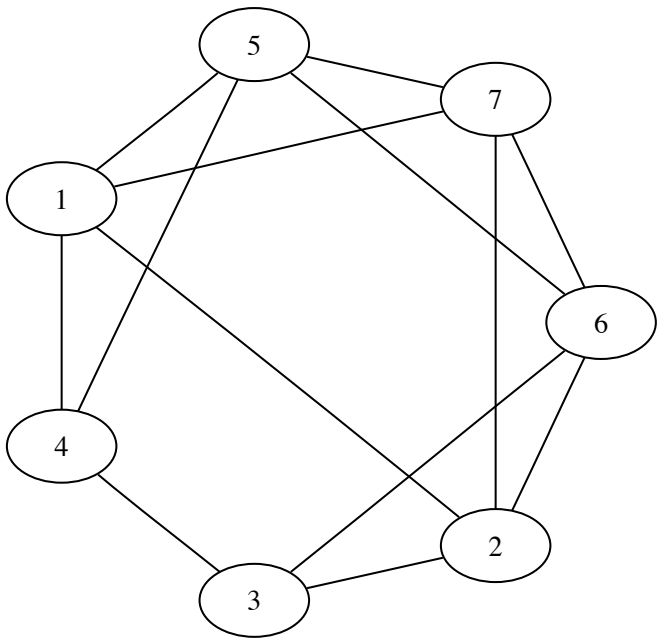
Graph Coloring

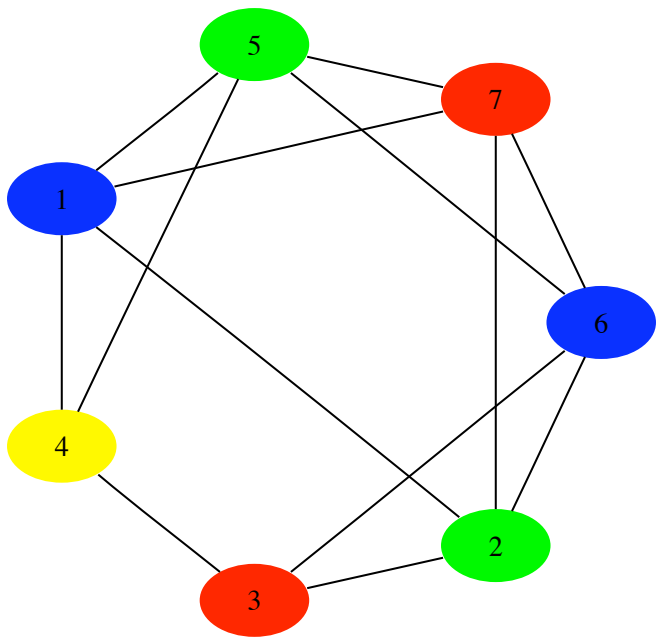
Given a graph $G = (V, E)$ and an integer k, a k-coloring of G is a mapping $c : V \rightarrow \{1, \ldots, k\}$, s.t. $c(i) \neq c(j), \forall \{i, j\} \in E$; Min-GCP consists in finding the minimum k such that a k-coloring exists. Min-GCP is NP-hard.



Graph Coloring

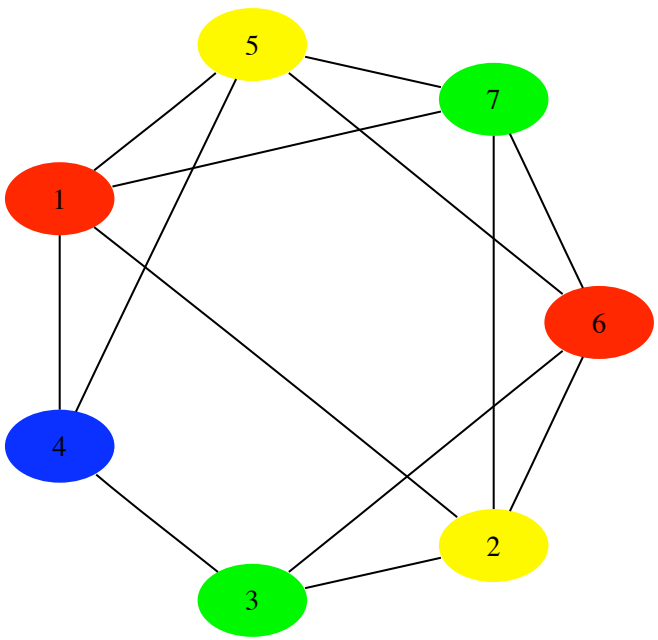
Given a graph $G = (V, E)$ and an integer k, a k-coloring of G is a mapping $c : V \rightarrow \{1, \ldots, k\}$, s.t. $c(i) \neq c(j), \forall \{i, j\} \in E$; Min-GCP consists in finding the minimum k such that a k-coloring exists. Min-GCP is NP-hard.

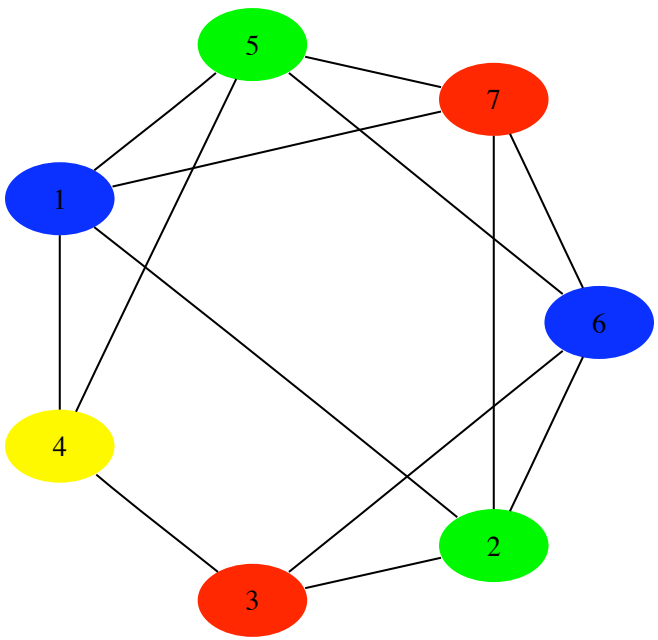




Graph Coloring

Given a graph $G = (V, E)$ and an integer k, a k-coloring of G is a mapping $c : V \rightarrow \{1, \ldots, k\}$, s.t. $c(i) \neq c(j), \forall \{i, j\} \in E$; Min-GCP consists in finding the minimum k such that a k-coloring exists. Min-GCP is NP-hard.





Classical Integer Programming Formulation

Integer Programming formulation

$$
\begin{aligned}
\min \quad & \textstyle\sum_{k \in K} y_k & & \\
\text{s.t.} \quad & \textstyle\sum_{k \in K} x_{ik} = 1, & & \forall i \in V, \\
& \textstyle\sum_{i \in C} x_{ik} \leq y_k, & & \forall C \in \mathcal{C}^*, \forall k \in K, \\
& x_{ik} \in \{0,1\}, & & \forall i \in V, \forall k \in K \\
& y_k \in \{0,1\}, & & \forall k \in K.
\end{aligned}
$$

Drawbacks

- It suffers from symmetric issues (permutations over indices k)
- The linear relaxations provides loose lower bounds

Example: $x_{11} = x_{61} = 1$,

$x_{23} = x_{24} = x_{33} = x_{34} = x_{43} = x_{44} = x_{53} = x_{54} = x_{73} = x_{74} = 0.5$

Column Generation Approach

Master Problem

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}} x_S \\ \text{s.t.} \quad & \sum_{S \in \mathcal{S}: i \in S} x_S \geq 1, \quad \forall i \in V, \\ & x_S \in \{0,1\}, \qquad \forall S \in \mathcal{S}. \end{aligned}$$

Pricing problem

$$\begin{aligned} \max \quad & \sum_{i \in V} \bar{\pi}_i y_i \\ \text{s.t.} \quad & y_i + y_j \leq 1, \quad \forall \{i,j\} \in E, \\ & y_i \in \{0,1\}, \quad \forall i \in V. \end{aligned}$$

Advantages

- It handles *color classes*, reducing symmetric issues
- It is the fastest exact method [MT96]

Drawbacks

- Since 1996, little if any improvements on this approach
- Difficult to generalize to other coloring problems
- The *linear relaxation* still provides *poor information*

Semidefinite Programming relaxations

A better relaxation can be derived using SDP

Semidefinite Programming relaxations

A better relaxation can be derived using SDP

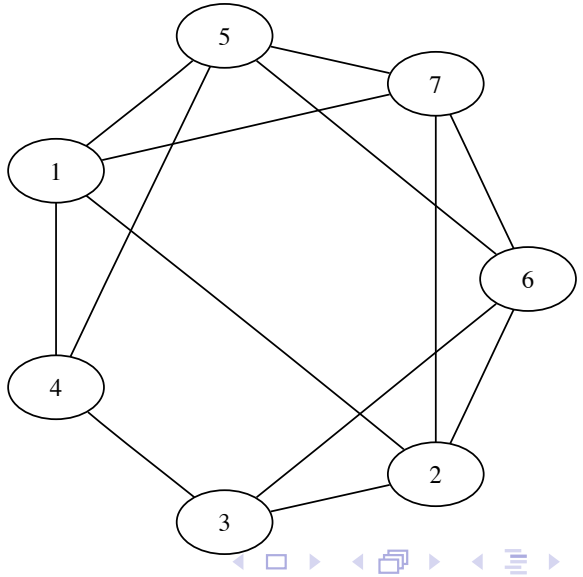
The idea is to assign unit vectors $v_i \in \mathbb{R}^n$ to every vertex $i \in V$, such that for any two adjacent vertices $v_i^T v_j \leq -\frac{1}{k-1}$. Define matrix V such that column i is given by v_i and let $X = V^T V$. The graph coloring problem can be posed as:

Semidefinite Programming relaxations

A better relaxation can be derived using SDP

The idea is to assign unit vectors $v_i \in \mathbb{R}^n$ to every vertex $i \in V$, such that for any two adjacent vertices $v_i^T v_j \leq -\frac{1}{k-1}$. Define matrix V such that column i is given by v_i and let $X = V^T V$. The graph coloring problem can be posed as:

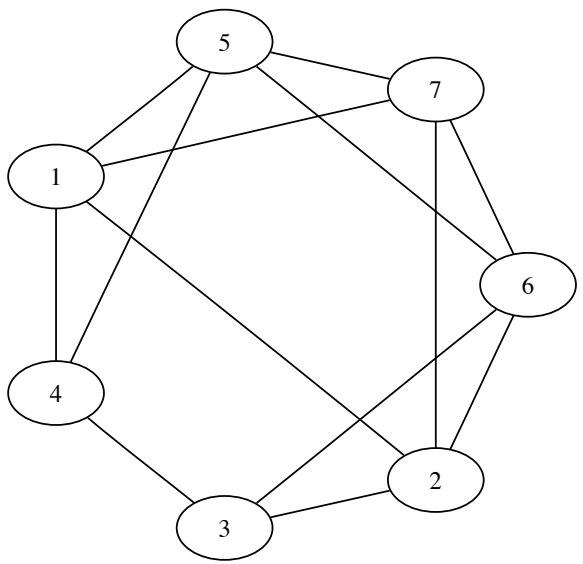
$$\begin{aligned} \min \quad & k \\ & \text{diag}(X) = e, \\ & X_{ij} \leq -\tfrac{1}{k-1}, \forall \{i,j\} \in E, \\ & X \succeq 0. \end{aligned}$$



Semidefinite Programming relaxations

	1	2	3	4	5	6	7
1	1.000						
2	-0.531	1.000					
3	0.064	-0.447	1.000				
4	-0.447	0.064	-0.501	1.000			
5	-0.411	0.696	0.064	-0.447	1.000		
6	0.696	-0.411	-0.447	0.064	-0.531	1.000	
7	-0.415	-0.415	0.420	0.420	-0.415	-0.415	1.000

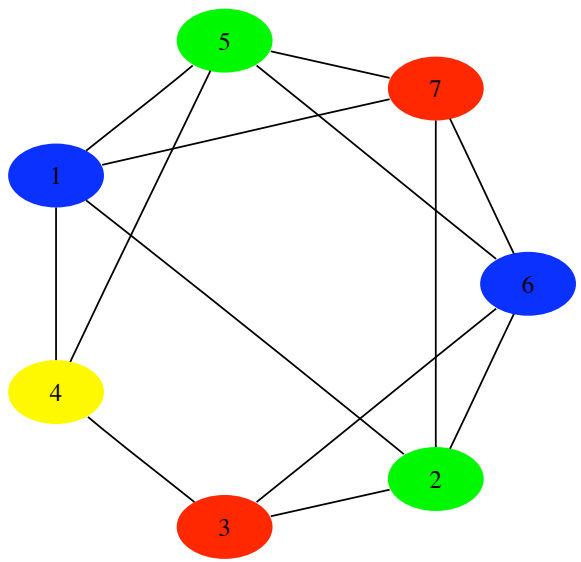
$$
\begin{aligned}
\min \quad & k \\
& \operatorname{diag}(X) = e, \\
& X_{ij} \leq -\tfrac{1}{k-1}, \forall \{i,j\} \in E, \\
& X \succeq 0.
\end{aligned}
$$



Semidefinite Programming relaxations

	1	2	3	4	5	6	7
1							
2							
3	0.064						
4		0.064					
5		**0.696**	0.064				
6	**0.696**			0.064			
7			**0.420**	0.420			

$$\begin{aligned} \min \quad & k \\ & \operatorname{diag}(X) = e, \\ & X_{ij} \leq -\tfrac{1}{k-1}, \forall \{i,j\} \in E, \\ & X \succeq 0. \end{aligned}$$



Semidefinite Programming relaxations

	1	2	3	4	5	6	7
1							
2							
3	0.064						
4		0.064					
5		**0.696**	0.064				
6	**0.696**			0.064			
7			**0.420**	0.420			

REMARK: *The value of X^*_{ij} in the optimal solution can be interpreted as the likehood that vertices i and j take the same color*

Constraint Programming approach

k-coloring problem

$$\begin{aligned} \textit{variables:} \quad & \texttt{domain}(y_i) = [1 \dots k], && \forall i \in V, \\ \textit{constraints:} \quad & \texttt{alldifferent}(\{y_i \mid i \in C\}), && \forall C \in \mathcal{C}^*. \end{aligned}$$

Advantages

- Simple and *intuitive* model
- Easy to generalize to other coloring problems (like bandwidth and multicoloring)

Drawbacks

- It solves iteratively many k-coloring problems
- Its efficiency depends strongly on the enumeration strategy

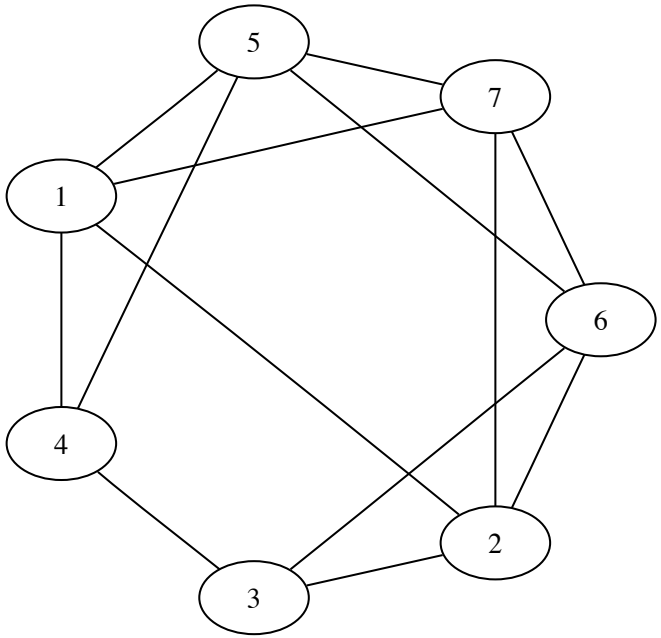
Constraint Programming approach

Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$



y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}

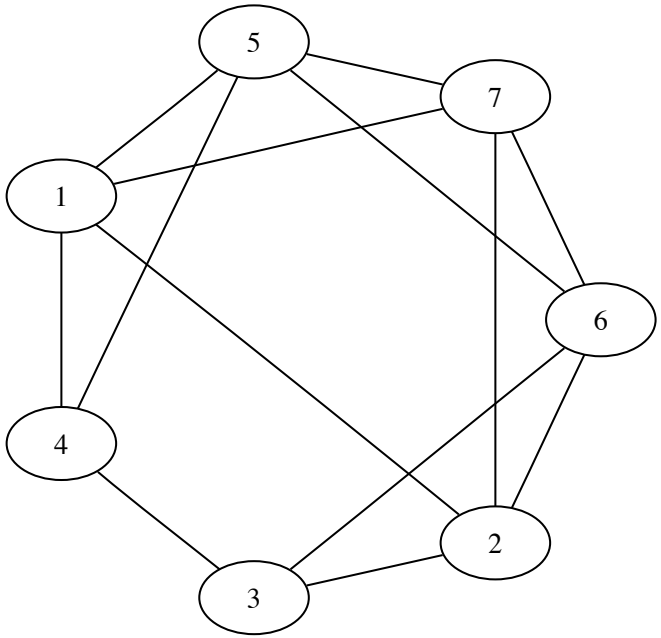
Constraint Programming approach

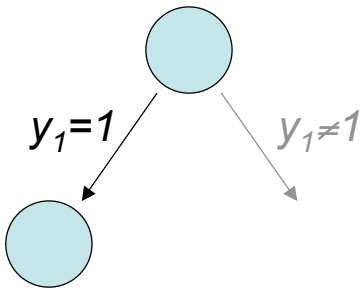
Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$





y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}

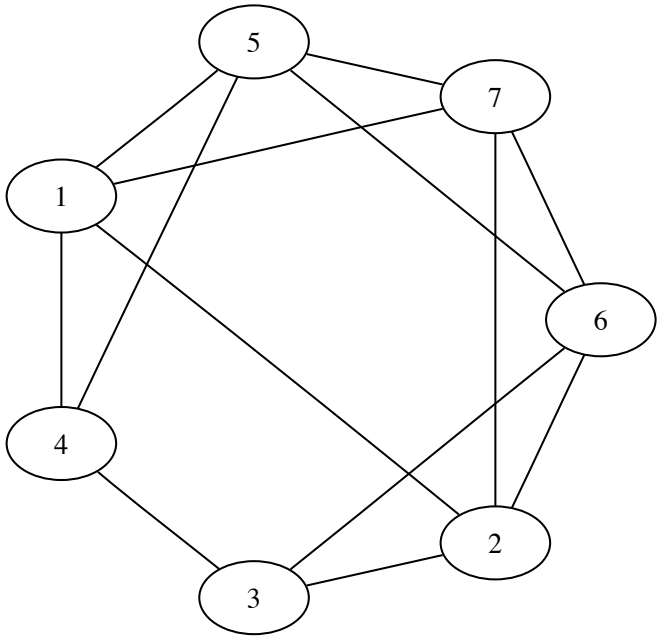
Constraint Programming approach

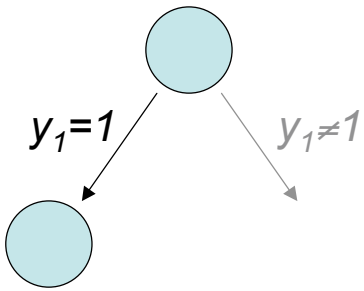
Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$





y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{**2,3**}	{1,2,3}	{**2,3**}	{**2,3**}	{1,2,3}	{**2,3**}
1	{2,3}	{1,2,3}	{2,3}	{2,3}	{1,2,3}	{2,3}
1	{2,3}	{**2,3**}	{2,3}	{2,3}	{**1**}	{2,3}

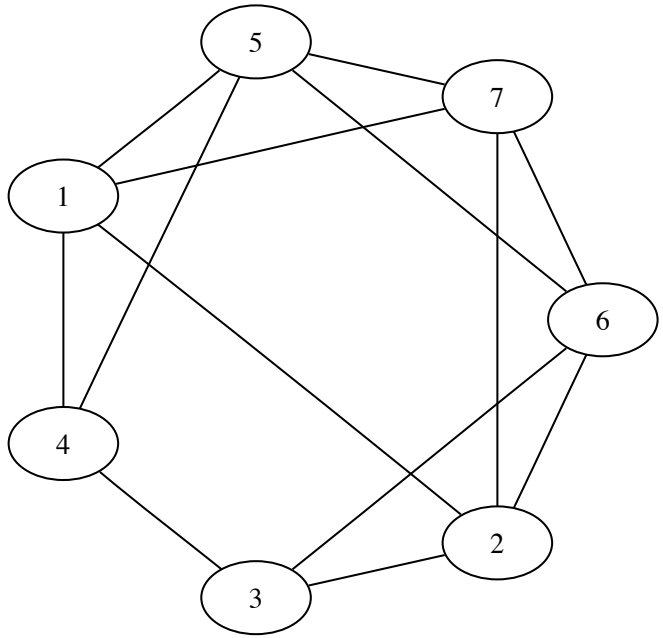
Constraint Programming approach

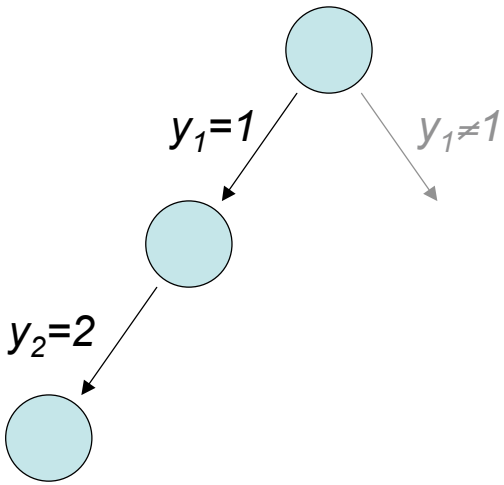
Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$





y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	**{2,3}**	{1,2,3}	**{2,3}**	**{2,3}**	{1,2,3}	**{2,3}**
1	{2,3}	{1,2,3}	{2,3}	{2,3}	{1,2,3}	{2,3}
1	{2,3}	**{2,3}**	{2,3}	{2,3}	**{1}**	{2,3}
1	**2**	{2,3}	{2,3}	{2,3}	{1}	{2,3}

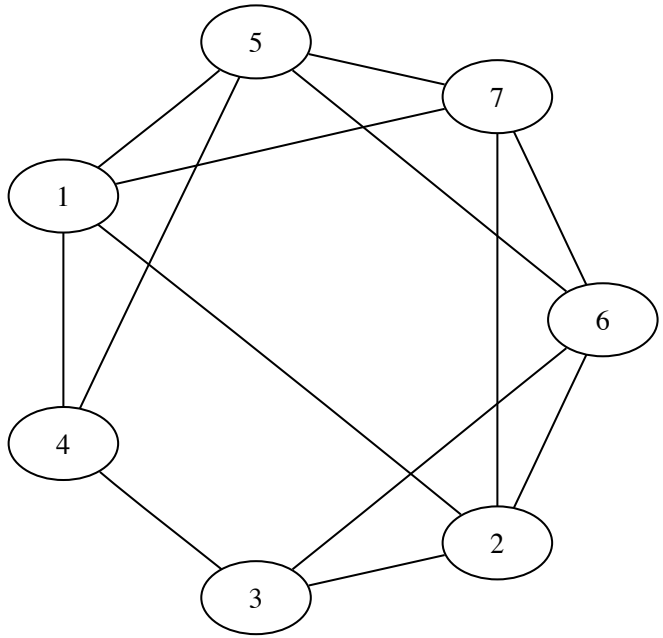
Constraint Programming approach

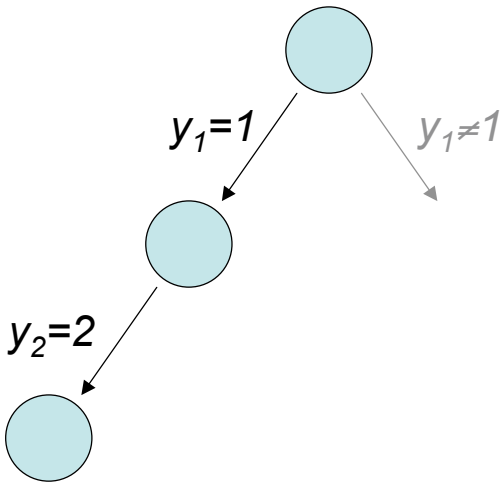
Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$





y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	**{2,3}**	{1,2,3}	**{2,3}**	**{2,3}**	{1,2,3}	**{2,3}**
1	{2,3}	{1,2,3}	{2,3}	{2,3}	{1,2,3}	{2,3}
1	{2,3}	**{2,3}**	{2,3}	{2,3}	**{1}**	{2,3}
1	**2**	{2,3}	{2,3}	{2,3}	{1}	{2,3}
1	2	**3**	{2,3}	{2,3}	{1}	**3**
1	2	3	**2**	**{}**	{1}	3

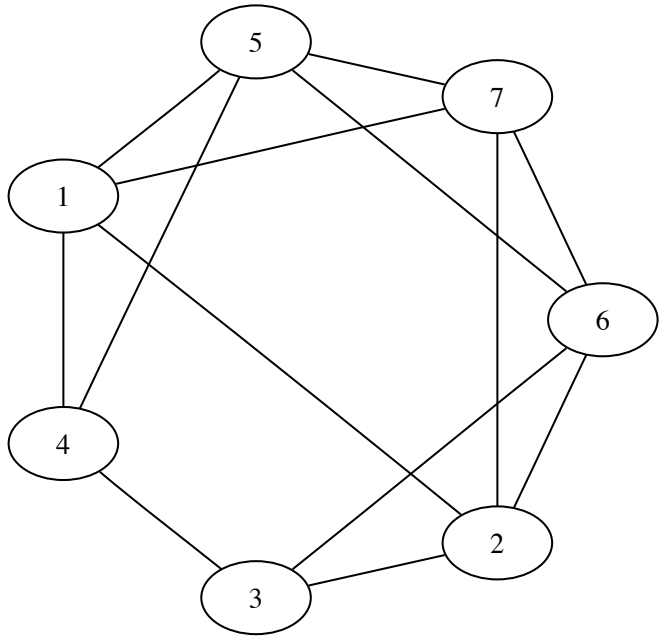
Constraint Programming approach

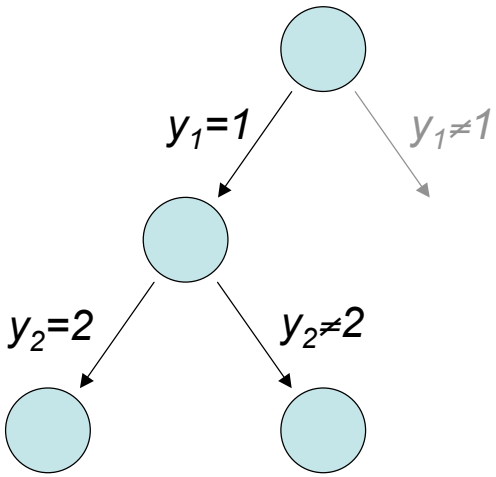
Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$





y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{**2,3**}	{1,2,3}	{**2,3**}	{**2,3**}	{1,2,3}	{**2,3**}
1	{2,3}	{1,2,3}	{2,3}	{2,3}	{1,2,3}	{2,3}
1	{2,3}	{**2,3**}	{2,3}	{2,3}	{**1**}	{2,3}
1	**3**	{2,3}	{2,3}	{2,3}	{1}	{2,3}

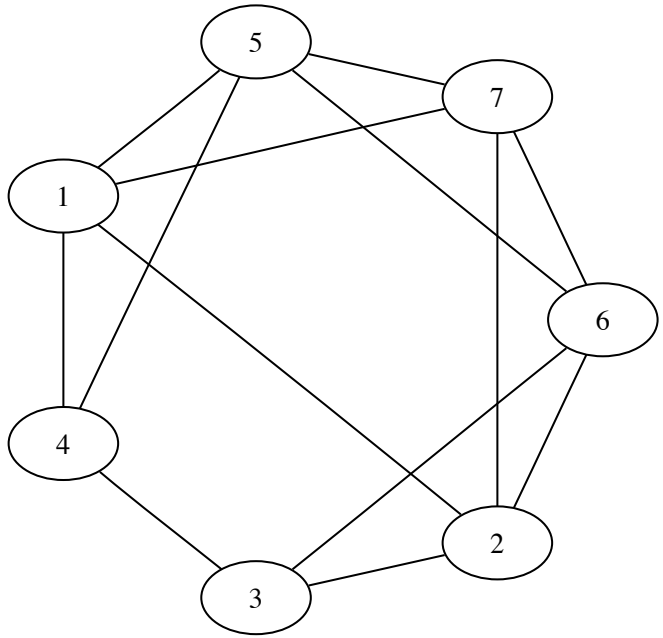
Constraint Programming approach

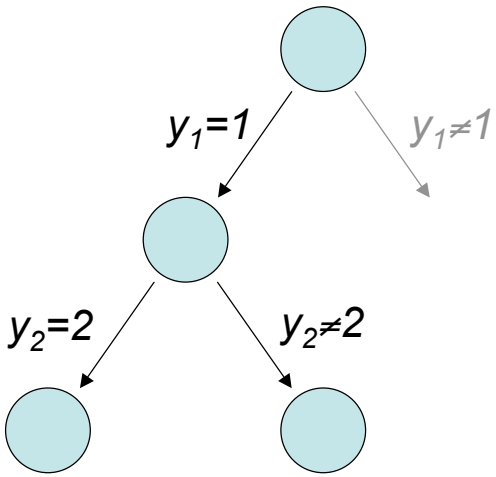
Let us suppose we are looking for a 3-coloring

CP model

variables: $y_i = \{1, 2, 3\}, \quad \forall i \in V$

constraints:
$\texttt{alldifferent}([y_1, y_2, y_7])$
$\texttt{alldifferent}([y_2, y_3, y_6])$
$\texttt{alldifferent}([y_1, y_4, y_5])$
$\texttt{alldifferent}([y_5, y_6, y_7])$
$\texttt{alldifferent}([y_3, y_4])$



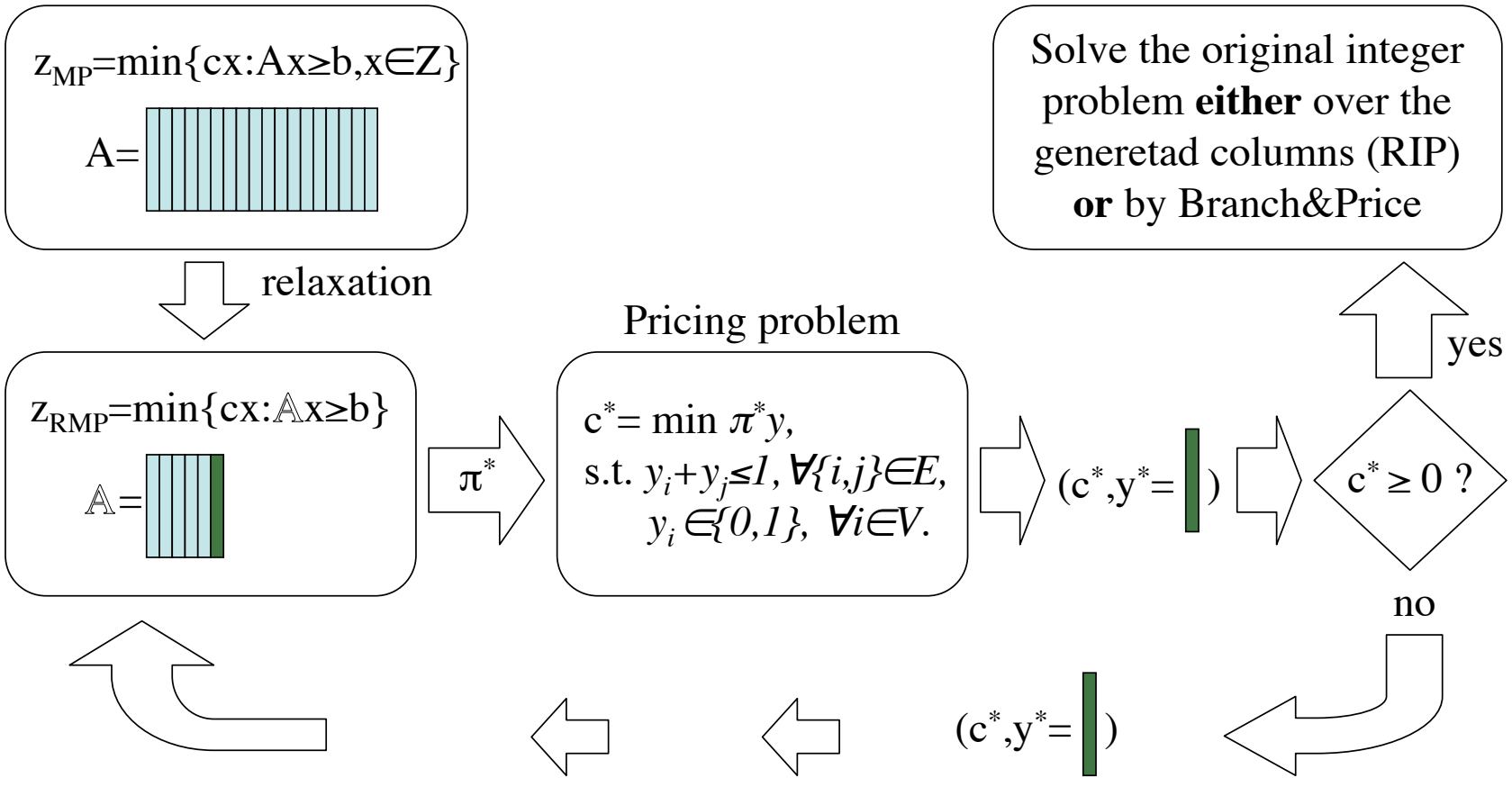


y_1	y_2	y_3	y_4	y_5	y_6	y_7
{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
1	{**2,3**}	{1,2,3}	{**2,3**}	{**2,3**}	{1,2,3}	{**2,3**}
1	{2,3}	{1,2,3}	{2,3}	{2,3}	{1,2,3}	{2,3}
1	{2,3}	{**2,3**}	{2,3}	{2,3}	{**1**}	{2,3}
1	**3**	{2,3}	{2,3}	{2,3}	{1}	{2,3}
1	3	**2**	{2,3}	{2,3}	{1}	**2**
1	3	2	**3**	{}	{1}	2

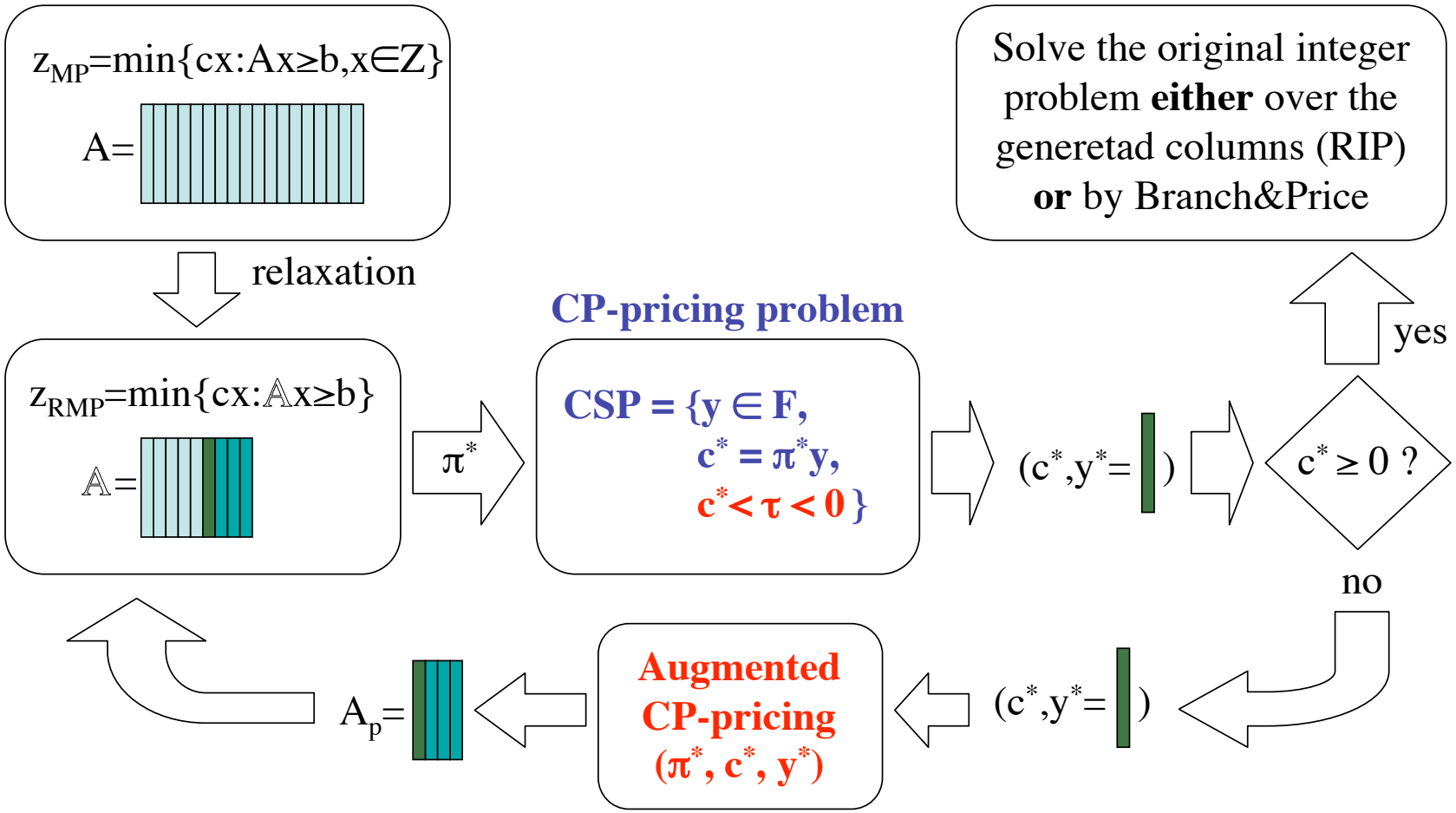
Outline

- Graph Coloring
- Formulations
  - Classical Formulation
  - Column Generation approach
  - Semidefinite Programming
  - Constraint Programming
- *Exact Hybrid Approaches*
  - Constraint Programming-based Column Generation
  - Exploiting SDP relaxations into Constraint Programming
- Computational results
- Conclusions

Constraint Programming-based Column Generation



Constraint Programming-based Column Generation



Enhancing the regular CP-based Column Generation

1. Adaptive thresholds in the pricing

Constraint Programming is used to find a maximal independet set of weight greater than or equal to a threshold τ. The value of this thresholds changes at each iteration of the Column Generation algorithm.

2. Augmented pricing problem

Once a negative reduced cost solution is found, we solve the *augmented pricing problem* that encodes *knowledge* from the integer master problem and generates *structured* columns. The augmented pricing problem is:

$$
\begin{aligned}
\textit{variables:} \quad & z_i \subseteq V, \forall i \in \{1, \ldots, k\} \\
\textit{constraints:} \quad & z_1 \equiv a_p \\
& \texttt{independentSet}(z_i, G) \in F, \quad \forall i \in \{1, \ldots, k\} \\
& \texttt{partition}([z_1, \ldots, z_k], V]) \\
& \texttt{cardinality}(z_i) \geq \texttt{cardinality}(z_{i+1}), \forall i = 2, \ldots, |V| - 1
\end{aligned}
$$

Enhancing the regular CP-based Column Generation

1. Adaptive thresholds in the pricing

Constraint Programming is used to find a maximal independet set of weight greater than or equal to a threshold τ. The value of this thresholds changes at each iteration of the Column Generation algorithm.

2. Augmented pricing problem

Once a negative reduced cost solution is found, we solve the *augmented pricing problem* that encodes *knowledge* from the integer master problem and generates *structured* columns. The augmented pricing problem is:

$$
\begin{aligned}
\textit{variables:} \quad & z_i \subseteq V, \forall i \in \{1, \ldots, k\} \\
\textit{constraints:} \quad & z_1 \equiv a_p \\
& \texttt{independentSet}(z_i, G) \in F, \quad \forall i \in \{1, \ldots, k\} \\
& \texttt{partition}([z_1, \ldots, z_k], V]) \\
& \texttt{cardinality}(z_i) \geq \texttt{cardinality}(z_{i+1}), \forall i = 2, \ldots, |V| - 1
\end{aligned}
$$

S.Gualandi. *Enhacing CP-based Column Generation for Integer Programs*. PhD Thesis. Forthcoming...

Outline

- Graph Coloring
- Formulations
  - Classical Formulation
  - Column Generation approach
  - Semidefinite Programming
  - Constraint Programming
- Exact Hybrid Approaches
  - Constraint Programming-based Column Generation
  - *Exploiting SDP relaxations into Constraint Programming*
- Computational results
- Conclusions

Exploiting SDP relaxations into CP

The efficiency of the Constraint Programming approach depends on the *variable and value selection heuristics*.
The best results for a regular CP approach are obtained with:

1. **Variable selection**: look for the vertex with the smallest ratio domain size against vertex degree. Then *break ties* using the vertex with the biggest number of suspensions
2. **Value selection**: pick the smallest color left in its domain

Exploiting SDP relaxations into CP

The efficiency of the Constraint Programming approach depends on the *variable and value selection heuristics*.
The best results for a regular CP approach are obtained with:

1. **Variable selection**: look for the vertex with the smallest ratio domain size against vertex degree. Then *break ties* using the vertex with the biggest number of suspensions
2. **Value selection**: pick the smallest color left in its domain

The optimal solution X^*_{ij} of a SDP relaxation gives the *likehood* that the two vertices i and j takes the same color

Exploiting SDP relaxations into CP

The hybridization consists in using the SDP relaxation X^* to derive effective variable and value selection heuristics:

Optimistic Variable selection

look for the vertex i with the smallest ratio domain size against vertex degree. Then, break ties by looking for the vertex with the *largest* X^*_{ij} value

Pessimistic Variable selection

look for the vertex i with the smallest ratio domain size against vertex degree. Then, break ties by looking for the vertex with the *smallest* X^*_{ij} value

Value selection

pick the smallest color left in the *intersection*: $\mathtt{dom}(y_i) \cap \mathtt{dom}(y_j)$

CP-based Column Generation vs. Column Generation

RIP-Heuristic: solve the restricted integer problem (RIP) over the generated columns
Strong Aug.: CP-based column generation with strong augmented pricing

					RIP-Heuristic			Strong Aug.		
Problem	$\|N\|$	$\|E\|$	$\lceil\chi_f\rceil$	χ	LB-UB	t_{RMP}	t_{RIP}	LB-UB	t_{UB}	time
myciel4	23	71	4	5	4-*5*	0.1	0.1	4-*5*	0.1	0.1
myciel5	47	236	4	6	4-*6*	0.1	0.3	4-*6*	0.1	0.1
queen8_8	64	1456	9	9	9-10	0.6	1.2	*9*	24.9	25.4
queen9_9	91	2112	9	10	9-11	3.9	16	9-*10*	29	32.6
queen10_10	100	1470	10	10	10-12	14	463	10-12	0.53	3600
will199GPIA	701	6772	7	7	7-8	1314	1336	*7*	3.8	448
le450_5d	450	9757	5	-	5-10	3600	-	5-6	20	3600
le450_15c	450	16680	12	-	15-24	3600	-	15-22	1191	3600
le450_15d	450	16750	15	-	15-25	3600	-	15-23	4.3	3600
DSJC125.1	125	736	5	5	5-6	132	136	5-6	0.5	1202
DSJC125.5	125	3891	**16**	?	16-19	8.5	1274	16-20	0.69	3600
DSJC125.9	125	6961	**43**	**44**	43-*44*	0.8	1.6	43-49	0.22	3600
DSJC250.1	250	3218	**6**	?	6-10	3600	-	6-9	1.42	3600
DSJC250.5	250	15668	**26**	?	26-43	253	3600	26-36	3.66	3600
DSJC250.9	250	27897	**71**	**73**	71-*73*	19	178	71-87	2.67	3600

SDP-CP vs. CP

					DSATUR		CP		CP-SDP	
Problem	$\|N\|$	$\|E\|$	ω	χ	back-tr.	time	back-tr.	time	back-tr.	time
myciel4	23	71	2	5	848	0,0	52	0,0	35	0,1
myciel5	47	236	2	6	378.311	0,6	4.816	0,1	808	0,5
myciel6	95	755	2	7	timeout		21.605.762	692	1.722.975	65,1
myciel7	191	2360	2	8	timeout		timeout		timeout	

Table: Mycielisky instances.

SDP-CP vs. CP

					DSATUR		CP		CP-SDP	
Problem	$\|N\|$	$\|E\|$	ω	χ	back-tr.	time	back-tr.	time	back-tr.	time
myciel4	23	71	2	5	848	0,0	52	0,0	35	0,1
myciel5	47	236	2	6	378.311	0,6	4.816	0,1	808	0,5
myciel6	95	755	2	7	timeout		21.605.762	692	1.722.975	65,1
myciel7	191	2360	2	8	timeout		timeout		timeout	

Table: Mycielisky instances.

Using Branch&Price, even instances `myciel5` was not solved within the timeout of 3600 sec.

Conclusions

1. Using the CP-based Column Generation within a Branch&Price, we close two open DIMACS instances (DSJC125.9, DSJC250.9) and improve the lower bounds of four *hard* DIMACS instances (DSJC125.5, DSJC250.5, DSJC500.9, DSJC1000.9)

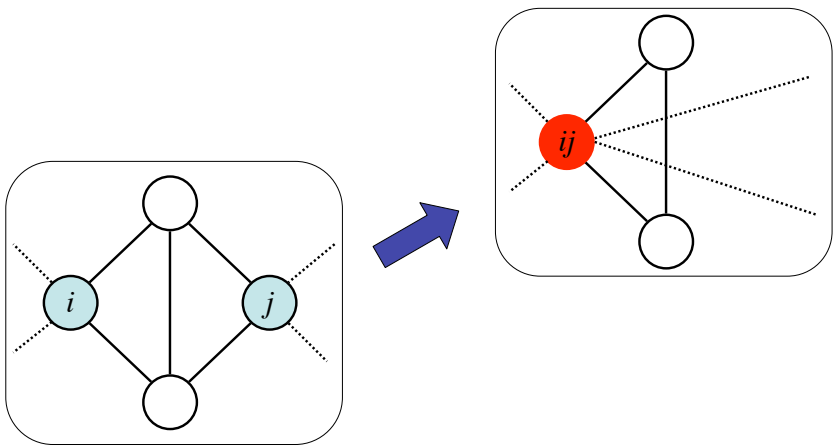
2. The Hybrid CP-SDP approach is very effective in proving optimality for small-medium instances having a gap between $\chi_f(G)$ and $\chi(G)$

Thanks for your attention

Branch&Price

- The column generation algorithm gives the fractional chromatic number $\chi_f(G)$, that is a lower bound on $\chi(G)$
- In order to get $\chi(G)$, we have implemented a branch-and-price, using the following branching rule:

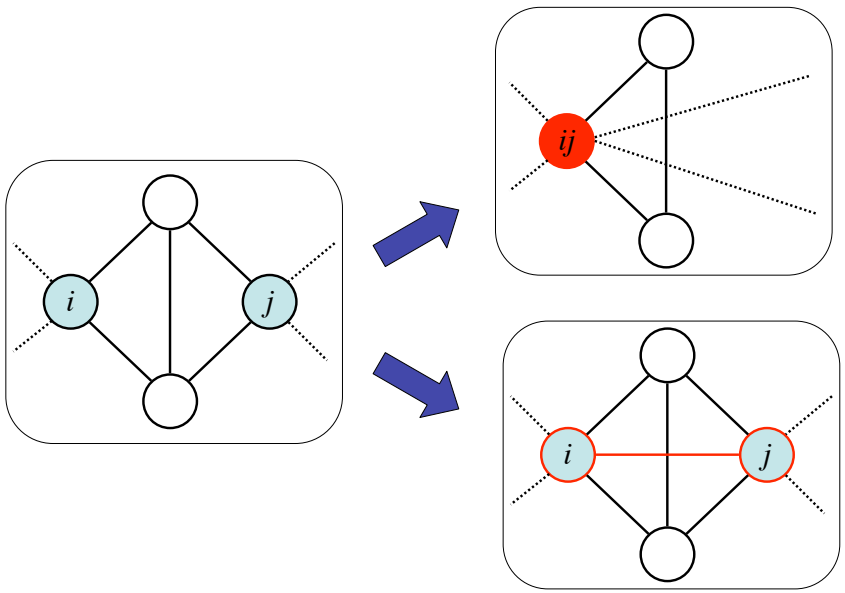
1. select a pair of vertices i and j being not adjacent and having the most negative reduced costs
2. *either* merge node i and j into a new node ij



Branch&Price

- The column generation algorithm gives the fractional chromatic number $\chi_f(G)$, that is a lower bound on $\chi(G)$
- In order to get $\chi(G)$, we have implemented a branch-and-price, using the following branching rule:

1. select a pair of vertices i and j being not adjacent and having the most negative reduced costs
2. *either* merge node i and j into a new node ij
3. *or* add a new edge $\{i, j\}$



A. Mehrotra and M. A. Trick.

A column generation approach for graph coloring.

INFORMS Journal on Computing, 8:344–354, 1996.