Safe Lower Bounds for Graph Coloring

Stephan Held

joint work with

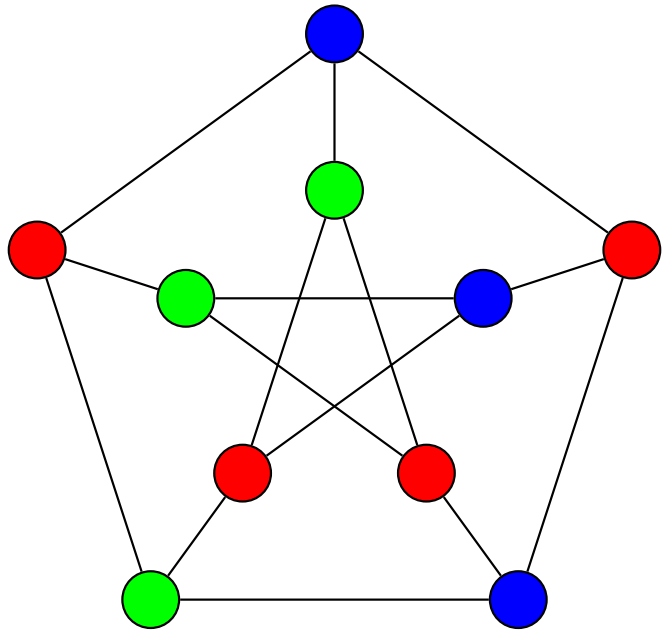
Edward C. Sewell and William Cook

Aussois
Janurary, 2011

Outline

- The Coloring IP
- A new algorithm for finding maximum-weight stable sets
- Safe computations with inaccurate floating-point arithmetic
- Computational results

The Graph Coloring Problem



Graph Coloring

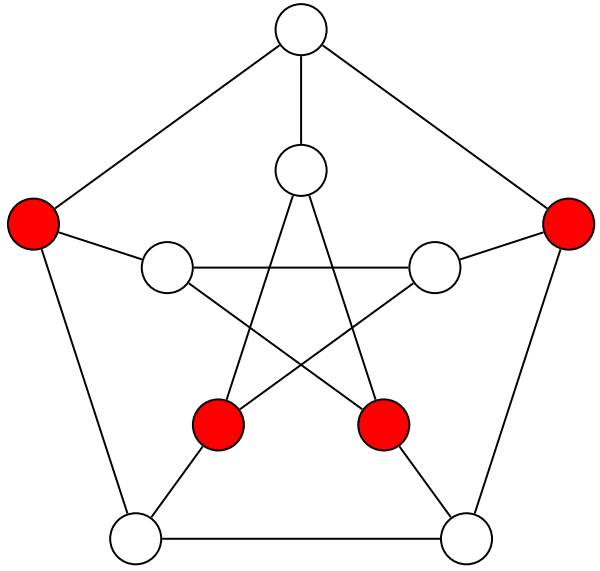
A coloring $c : V(G) \to \mathbb{N}$ assigns colors/numbers to the vertices such that adjacent vertices are colored differently.
The minimum number $\chi(G)$ of colors needed to color the vertices is called the chromatic number.

Colorings & Stable Sets

Given a graph $G = (V, G)$ and a coloring c, each color class

$$[v]_c = \{w \in V \,;\; c(v) = c(w)\}$$

is a stable set,
i.e. $\{v_1, v_2\} \notin E$ for all $v_1, v_2 \in [v]_c$.



Alternative Definition

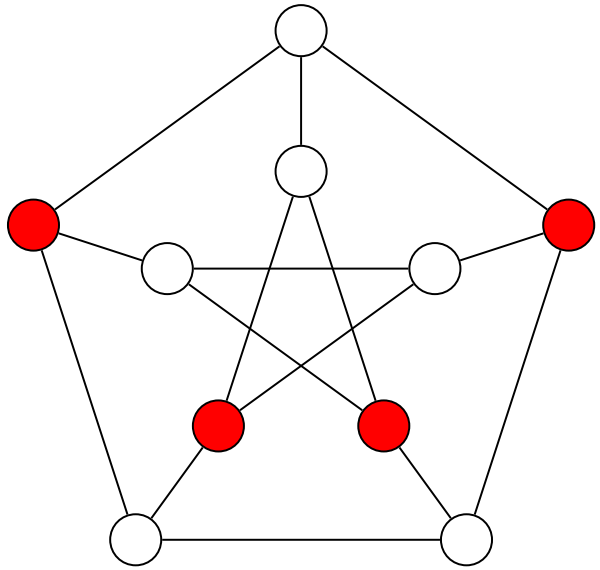
A coloring is a partition of V into stable sets.

Colorings & Stable Sets

Given a graph $G = (V, G)$ and a coloring c, each color class

$$[v]_c = \{w \in V \,;\; c(v) = c(w)\}$$

is a stable set,
i.e. $\{v_1, v_2\} \notin E$ for all $v_1, v_2 \in [v]_c$.



Alternative Definition

A coloring is a partition of V into stable sets.

Coloring IP for Computing $\chi(G)$

Let G be a graph and $\mathcal{S}$ be the set of stable sets in G

$$\begin{aligned} \chi(G) = \quad & \min \quad \sum_{S \in \mathcal{S}} x_S & & \\ & s.t. \quad \sum_{S \in \mathcal{S}: v \in S} x_S = 1 & \forall v \in V & \qquad \text{(CIP)} \\ & \qquad x_S \in \{0,1\} & \forall S \in \mathcal{S} & \end{aligned}$$

The LP-relaxation

$$\begin{aligned} \chi_f(G) := \quad & \min \quad \sum_{S \in \mathcal{S}} x_S & & \\ & s.t. \quad \sum_{S \in \mathcal{S}: v \in S} x_S \geq 1 & \forall v \in V & \qquad \text{(CLP)} \\ & \qquad x_S \in [0,1] & \forall S \in \mathcal{S} & \end{aligned}$$

defines a lower bound $\lceil \chi_f(G) \rceil$ for (CIP).

$\chi_f(G)$ is called the fractional chromatic number.

Coloring IP for Computing $\chi(G)$

Let G be a graph and $\mathcal{S}$ be the set of maximal stable sets in G

$$\begin{aligned} \chi(G) = \quad & \min \quad \sum_{S \in \mathcal{S}} x_S & & \\ & s.t. \quad \sum_{S \in \mathcal{S}: v \in S} x_S \geq 1 & \forall v \in V & \quad \text{(CIP)} \\ & \quad\; x_S \in \{0,1\} & \forall S \in \mathcal{S} & \end{aligned}$$

The LP-relaxation

$$\begin{aligned} \chi_f(G) := \quad & \min \quad \sum_{S \in \mathcal{S}} x_S & & \\ & s.t. \quad \sum_{S \in \mathcal{S}: v \in S} x_S \geq 1 & \forall v \in V & \quad \text{(CLP)} \\ & \quad\; x_S \in [0,1] & \forall S \in \mathcal{S} & \end{aligned}$$

defines a lower bound $\lceil \chi_f(G) \rceil$ for (CIP).

$\chi_f(G)$ is called the fractional chromatic number.

Coloring IP for Computing $\chi(G)$

Let G be a graph and $\mathcal{S}$ be the set of maximal stable sets in G

$$\begin{aligned} \chi(G) = \quad & \min \quad \sum_{S \in \mathcal{S}} x_S & & \\ & s.t. \quad \sum_{S \in \mathcal{S}: v \in S} x_S \geq 1 & \forall v \in V & \qquad \text{(CIP)} \\ & \quad x_S \in \{0,1\} & \forall S \in \mathcal{S} & \end{aligned}$$

The LP-relaxation

$$\begin{aligned} \chi_f(G) := \quad & \min \quad \sum_{S \in \mathcal{S}} x_S & & \\ & s.t. \quad \sum_{S \in \mathcal{S}: v \in S} x_S \geq 1 & \forall v \in V & \qquad \text{(CLP)} \\ & \quad x_S \in [0,1] & \forall S \in \mathcal{S} & \end{aligned}$$

defines a lower bound $\lceil \chi_f(G) \rceil$ for (CIP).

$\chi_f(G)$ is called the fractional chromatic number.

Column Generation

Given a proper subset $\mathcal{S}' \subset \mathcal{S}$ of stable sets and a dual optimum solution $(\pi_v)_{v \in V}$ to the restricted LP

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}'} x_S & & \\ s.t. \quad & \sum_{S \in \mathcal{S}' : v \in S} x_S \geq 1 & \forall v \in V & \qquad \text{(CLP-r)} \\ & x_S \in [0,1] & \forall S \in \mathcal{S}', & \end{aligned}$$

a set $S \in \mathcal{S} \setminus \mathcal{S}'$ can improve (CLP-r) only if

$$\sum_{v \in S} \pi_v > 1.$$

Column Generation

Given a proper subset $\mathcal{S}' \subset \mathcal{S}$ of stable sets and a dual optimum solution $(\pi_v)_{v \in V}$ to the restricted LP

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}'} x_S & & \\ s.t. \quad & \sum_{S \in \mathcal{S}' : v \in S} x_S \geq 1 & \forall v \in V & \qquad \text{(CLP-r)} \\ & x_S \in [0,1] & \forall S \in \mathcal{S}', & \end{aligned}$$

a set $S \in \mathcal{S} \setminus \mathcal{S}'$ can improve (CLP-r) only if

$$\sum_{v \in S} \pi_v > 1.$$

Maximum-Weight-Stable-Set-Problem (MWSS)

Given a dual optimum solution $(\pi_v)_{v \in V}$ we have to solve a maximum-weight stable set problem:

$$\begin{aligned} \alpha_\pi(G) := \quad & \max \sum_{v \in V} \pi_v y_v & & \\ & s.t. \quad y_v + y_w \leq 1 & & \forall \{v, w\} \in E \\ & \qquad y_v \in \{0, 1\} & & \forall v \in V \end{aligned} \tag{MWSS}$$

If $\sum_{v \in V} \pi_v y_v \leq 1$: The current solution x is optimum for (CLP) and $\chi_f(G) = \sum_{v \in V} \pi_v$.

If $\sum_{v \in V} \pi_v y_v > 1$: $\{v \in V \,;\, y_v = 1\}$ is an improving stable set

Solving (CLP)

Previous Work

- Mehrotra & Trick (1996) demonstrated the efficacy of branch & price on many (small) DIMACS benchmarks:
  - $\chi_f(G)$ close to $\chi(G)$ on most instances
  - branch & price is "fast"
- Recently,
  - Gualandi and Malucelli (2010)
  - Malaguti, Monaci and Toth (2010)

  applied branch & price again.

Solving (CLP)

Previous Work

- Mehrotra & Trick (1996) demonstrated the efficacy of branch & price on many (small) DIMACS benchmarks:
  - $\chi_f(G)$ close to $\chi(G)$ on most instances
  - branch & price is "fast"
- Recently,
  - Gualandi and Malucelli (2010)
  - Malaguti, Monaci and Toth (2010)

  applied branch & price again.

Open Questions

- How to solve the MWSS-problems efficiently?
  - bottleneck for Gualandi and Malucelli (2010) and Malaguti et al. (2010)

- How to avoid the inaccuracy of floating-point arithmetic?
  - Gurobi 3.0.0 returned

    $$\chi_f(G) = 16.0000000000001315$$

    on `queen16_16`, when column generation terminated. But $17 = \lceil \chi_f(G) \rceil$ is not a valid lower bound!
  - Also: MWSS subproblems must be solved exactly!

Solving the Maximum-Weight-Stable-Set-Problem

Most improving columns can be found by

Heuristics

1. greedy start solution
2. improved by local search

. . . but eventually we have to solve the problem exactly!

Solving the Maximum-Weight-Stable-Set-Problem

Exact Algorithms

IP: Branch & Cut

- efficient for sparse graphs
- DIMACS benchmarks are dense:
  3 000% integrality gap after adding clique and odd hole cuts.
- floating-point uncertainty

Combinatorial Enumeration

- Östergård (2002), a.k.a. CLIQUER: Fastest for dense graphs (edge density ~ 0.9) with sparse complement.
- Sewell (1993): Fast algorithm for unweighted graphs and edge densities $\sim 0.1 - 0.8$
  (variant of Balas and Yu's branching scheme)

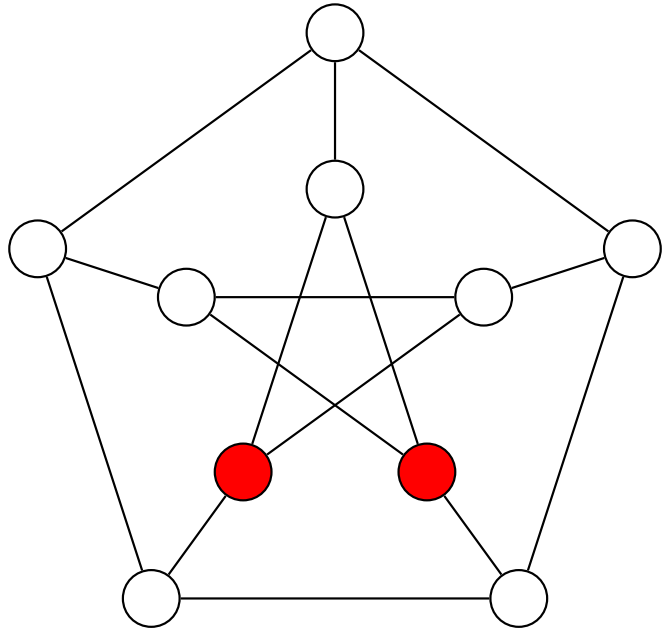
here: edge density of a graph $G = (V, E)$: $\frac{|E|}{|V|(|V|-1)/2}$

A New Combinatorial Branch & Bound Algorithm

Variant of Balas & Yu (1986), Balas & Xue (1992), and Sewell (1998)

Notation

- current stable set S
- blocked vertices X
  $(S \cup \{x\} \not\subset S^{opt} \quad \forall x \in X)$
- free vertices F
  (used to grow S)



High-level idea

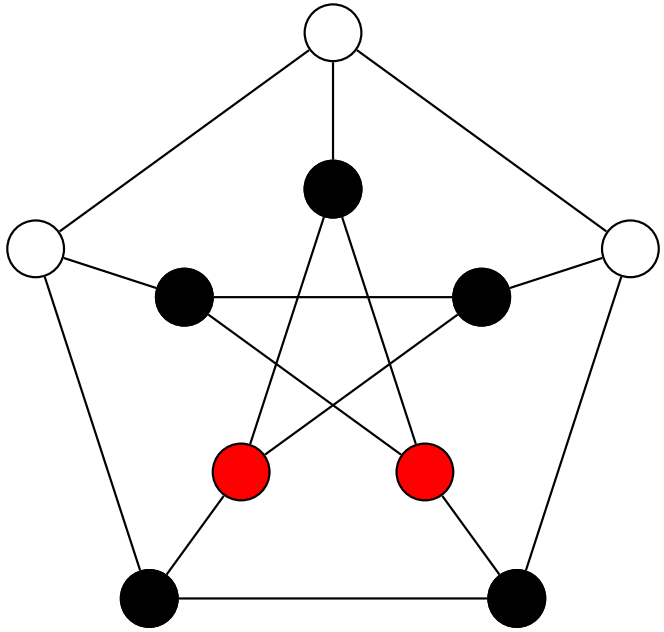
Starting with $S = \emptyset, F = V$, and $X = \emptyset$ the algorithms enumerates all solutions recursively.

A New Combinatorial Branch & Bound Algorithm

Variant of Balas & Yu (1986), Balas & Xue (1992), and Sewell (1998)

Notation

- current stable set S
- blocked vertices X
  ($S \cup \{x\} \not\subset S^{opt} \quad \forall x \in X$)
- free vertices F
  (used to grow S)



High-level idea

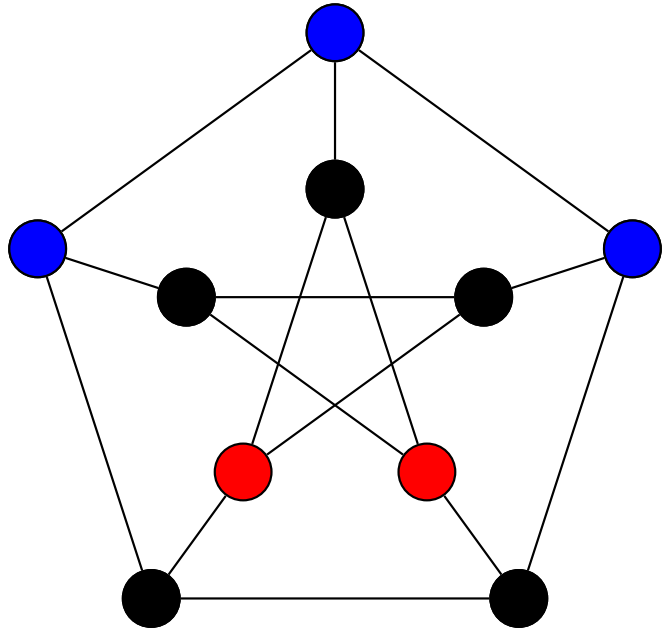
Starting with $S = \emptyset, F = V$, and $X = \emptyset$ the algorithms enumerates all solutions recursively.

A New Combinatorial Branch & Bound Algorithm

Variant of Balas & Yu (1986), Balas & Xue (1992), and Sewell (1998)

Notation

- current stable set S
- blocked vertices X
  ($S \cup \{x\} \not\subset S^{opt} \quad \forall x \in X$)
- free vertices F
  (used to grow S)



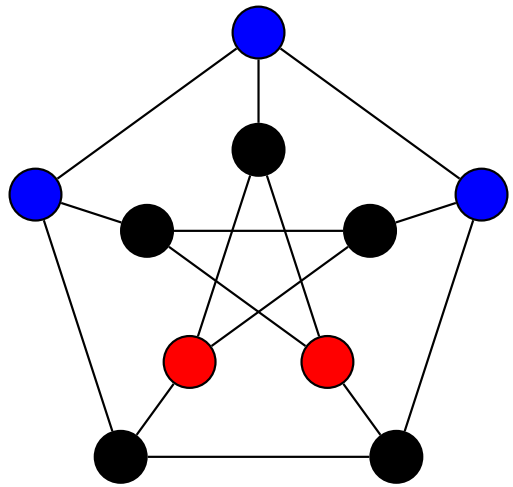
High-level idea

Starting with $S = \emptyset, F = V$, and $X = \emptyset$ the algorithms enumerates all solutions recursively.

A New Combinatorial Branch & Bound Algorithm

Basic recursion (Balas & Yu 1986)

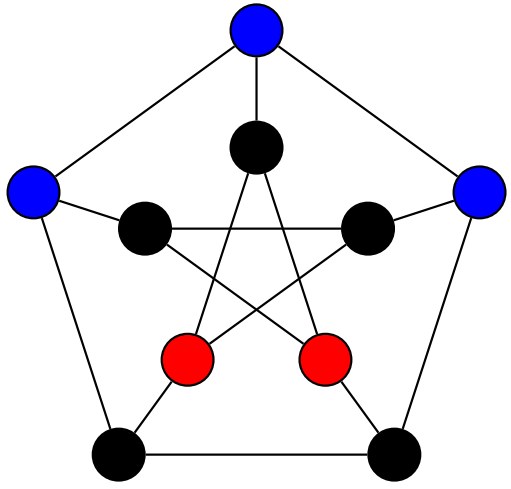
function MWSS_RECURSION(S,F,X)
    Choose branching vertices:
    $F'' = \{f_1, f_2, \ldots, f_p\} \subseteq F$
    for $i = p$ down to 1 **do**
        $F_i = F \setminus (N(f_i) \cup \{f_i, f_{i+1}, \ldots, f_p\})$;
        MWSS_RECURSION$(S \cup \{f_i\}, F_i, X)$;
        $X = X \cup \{f_i\}$;
    end for
end function



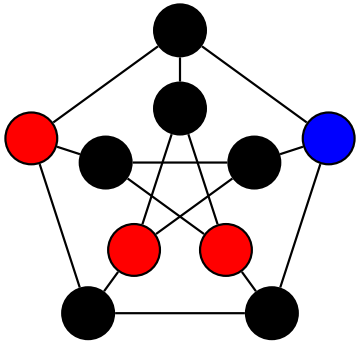
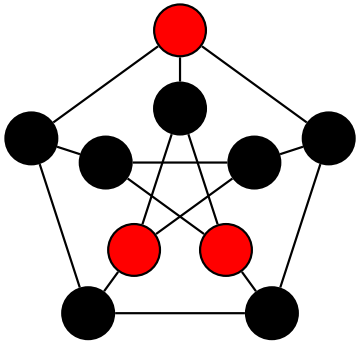
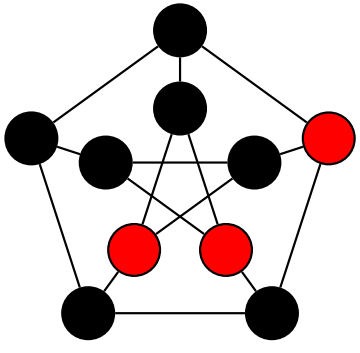
A New Combinatorial Branch & Bound Algorithm

Basic recursion (Balas & Yu 1986)

function MWSS_RECURSION(S,F,X)
  Choose branching vertices:
  $F'' = \{f_1, f_2, \ldots, f_p\} \subseteq F$
  for $i = p$ down to 1 **do**
    $F_i = F \setminus (N(f_i) \cup \{f_i, f_{i+1}, \ldots, f_p\})$;
    MWSS_RECURSION$(S \cup \{f_i\}, F_i, X)$;
    $X = X \cup \{f_i\}$;
  end for
end function

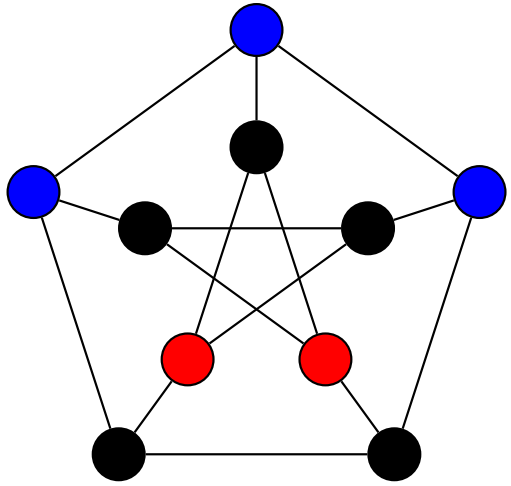


In our example

A New Combinatorial Branch & Bound Algorithm

function MWSS_RECURSION(S,F,X)
  Choose branching vertices:
  $F'' = \{f_1, f_2, \ldots, f_p\} \subseteq F$
  for $i = p$ down to 1 **do**
    $F_i = F \setminus (N(f_i) \cup \{f_i, f_{i+1}, \ldots, f_p\})$;
    MWSS_RECURSION$(S \cup \{f_i\}, F_i, X)$;
    $X = X \cup \{f_i\}$;
  end for
end function



Pruning I — Local Pruning

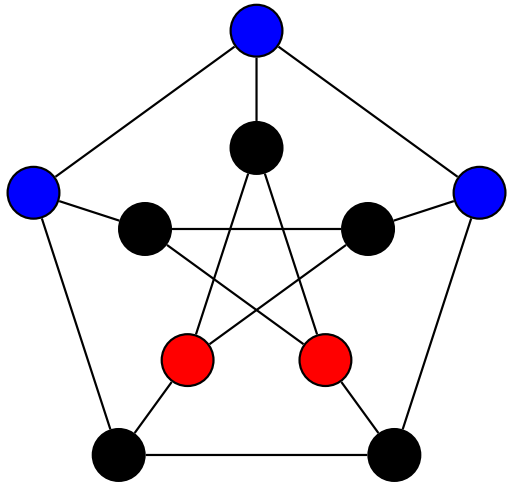
The current branch can be pruned if

$$\exists\, x \in X \text{ with } \pi_x \geq \pi\left((S \cup F) \cap N(x)\right)$$

Another branch using $S \setminus N(x) \cup \{x\}$ will yield a better solution.

A New Combinatorial Branch & Bound Algorithm

function MWSS_RECURSION(S,F,X)
  Choose branching vertices:
  $F'' = \{f_1, f_2, \ldots, f_p\} \subseteq F$
  for $i = p$ down to 1 **do**
    $F_i = F \setminus (N(f_i) \cup \{f_i, f_{i+1}, \ldots, f_p\})$;
    MWSS_RECURSION$(S \cup \{f_i\}, F_i, X)$;
    $X = X \cup \{f_i\}$;
  end for
end function



Pruning II

Let LB be a lower bound for $\alpha_\pi(G)$. The current branch can be pruned if

$$\pi(S) + \alpha_\pi(G[F]) \leq LB.$$

(requires a good upper bound for $\alpha_\pi(G[F])$)

$G[F] = (F, \{\{v, w\} \in E : v, w \in F\})$: induced subgraph

Pruning II — An Upper Bound for $\alpha_\pi(G)$

Balas & Xue (1991) proposed weighted clique covers.

Weighted Clique Covers

A weighted clique cover for (G, π) is a set of cliques $K_1, \ldots, K_r$ together with a positive weight Π_i for each K_i such that for each $v \in V$:

$$\pi_v \leq \sum_{i : v \in K_i} \Pi_i.$$

The weight of the weighted clique cover is $\sum_{i=1}^{r} \Pi_i$. The minimum weight of a clique cover is called $\Phi_\pi(G)$ and we have

$$\alpha_\pi(G) \leq \Phi_\pi(G) \leq \sum_{i=1}^{r} \Pi_i.$$

We find clique covers greedily similar to Babel (1994).

Branching — Choosing the branching vertices F''

Suppose that $F' \subseteq F$ and it can be shown that

$$\alpha_\pi\left(G\left[F'\right]\right) \leq LB - \pi\left(S\right).$$

Let $F'' = F \backslash F' = \{f_1, f_2, \ldots, f_p\}$ and let

$$F_i = F \backslash \left(N\left(f_i\right) \cup \{f_i, f_{i+1}, \ldots, f_p\}\right).$$

If $\alpha_\pi\left(G\left[F\right]\right) > LB - \pi\left(S\right)$, then

$$\alpha_\pi\left(G\left[F\right]\right) = \max_{i=1,\ldots,p} \pi_{f_i} + \alpha_\pi\left(G\left[F_i\right]\right).$$

Branching I — Extended Weighted Clique Cover Heuristic

"Good" sets F' and F'' can be found, while growing the clique cover.

Branching — Choosing the branching vertices F''

Suppose that $F' \subseteq F$ and it can be shown that

$$\alpha_\pi\left(G\left[F'\right]\right) \leq LB - \pi\left(S\right).$$

Let $F'' = F \backslash F' = \{f_1, f_2, \ldots, f_p\}$ and let

$$F_i = F \backslash \left(N\left(f_i\right) \cup \{f_i, f_{i+1}, \ldots, f_p\}\right).$$

If $\alpha_\pi\left(G\left[F\right]\right) > LB - \pi\left(S\right)$, then

$$\alpha_\pi\left(G\left[F\right]\right) = \max_{i=1,\ldots,p} \pi_{f_i} + \alpha_\pi\left(G\left[F_i\right]\right).$$

Branching I — Extended Weighted Clique Cover Heuristic

"Good" sets F' and F'' can be found, while growing the clique cover.

Branching — Choosing the branching vertices F''

Suppose that $F' \subseteq F$ and it can be shown that

$$\alpha_\pi\left(G\left[F'\right]\right) \leq LB - \pi\left(S\right).$$

Let $F'' = F \backslash F' = \{f_1, f_2, \ldots, f_p\}$ and let

$$F_i = F \backslash \left(N\left(f_i\right) \cup \{f_i, f_{i+1}, \ldots, f_p\}\right).$$

If $\alpha_\pi\left(G\left[F\right]\right) > LB - \pi\left(S\right)$, then

$$\alpha_\pi\left(G\left[F\right]\right) = \max_{i=1,\ldots,p} \pi_{f_i} + \alpha_\pi\left(G\left[F_i\right]\right).$$

Branching I — Extended Weighted Clique Cover Heuristic

"Good" sets F' and F'' can be found, while growing the clique cover.

Branching — Choosing the branching vertices F''

Suppose that $F' \subseteq F$ and it can be shown that

$$\alpha_\pi\left(G\left[F'\right]\right) \leq LB - \pi\left(S\right).$$

Let $F'' = F \backslash F' = \{f_1, f_2, \ldots, f_p\}$ and let

$$F_i = F \backslash \left(N\left(f_i\right) \cup \{f_i, f_{i+1}, \ldots, f_p\}\right).$$

If $\alpha_\pi\left(G\left[F\right]\right) > LB - \pi\left(S\right)$, then

$$\alpha_\pi\left(G\left[F\right]\right) = \max_{i=1,\ldots,p} \pi_{f_i} + \alpha_\pi\left(G\left[F_i\right]\right).$$

Branching I — Extended Weighted Clique Cover Heuristic

"Good" sets F' and F'' can be found, while growing the clique cover.

Branching — Choosing F'' locally

Lemma

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$ and $\alpha_\pi(G[F]) + \pi(S) > LB$, any maximum-weight stable set of $G[F]$ must contain a vertex in $N(x)$.

Branching II — Local Branching

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$, we set $F'' = N(x) \cap F$.

Lemma

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, there is a maximum-weight stable set of $G[F]$ that contains x.

Branching III — Singular Branching

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, we set $F'' = \{f\} \cap F$.

Branching — Choosing F'' locally

Lemma

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$ and $\alpha_\pi(G[F]) + \pi(S) > LB$, any maximum-weight stable set of $G[F]$ must contain a vertex in $N(x)$.

Branching II — Local Branching

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$, we set $F'' = N(x) \cap F$.

Lemma

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, there is a maximum-weight stable set of $G[F]$ that contains x.

Branching III — Singular Branching

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, we set $F'' = \{f\} \cap F$.

Branching — Choosing F'' locally

Lemma

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$ and $\alpha_\pi(G[F]) + \pi(S) > LB$, any maximum-weight stable set of $G[F]$ must contain a vertex in $N(x)$.

Branching II — Local Branching

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$, we set $F'' = N(x) \cap F$.

Lemma

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, there is a maximum-weight stable set of $G[F]$ that contains x.

Branching III — Singular Branching

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, we set $F'' = \{f\} \cap F$.

Branching — Choosing F'' locally

Lemma

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$ and $\alpha_\pi(G[F]) + \pi(S) > LB$, any maximum-weight stable set of $G[F]$ must contain a vertex in $N(x)$.

Branching II — Local Branching

If there is an $x \in X$ with $\pi_x \geq \pi(S \cap N(x))$, we set $F'' = N(x) \cap F$.

Lemma

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, there is a maximum-weight stable set of $G[F]$ that contains x.

Branching III — Singular Branching

If there is an $x \in F$ with $\pi_x \geq \pi(F \cap N(x))$, we set $F'' = \{f\} \cap F$.

Branching — Choosing F''

Overall Branching

- We pick the rule generating the minimum number $|F''|$ of branches.
- Ties are broken in favor of the 1st rule and then the 3rd rule.

Impact of Local Branching & Pruning

- Local branching and pruning greatly reduce the size of the B&B-tree.
- unsolvable instances (within 1 week) can now be solved within 2 h.

Further Tuning for Coloring

- we can start with $LB = 1$ and stop ...
- ... once a feasible solution of weight > 1 is found.

Safe Computations

With Inaccurate Floating Point Operations

Difficulties

- Floating point executions are inaccurate:
  - How to interpret
$$\chi_f(G) = 16.000000000001315?$$
- Exact LP solvers, e.g. QSopt_exact, would be too slow in a branch & price framework.

Safe Computations with Unreliable Floating Point Operations

- Any integral vector $\pi^{int} \in \mathbb{N}^{|V|}$ with

$$\sum_{v \in S} \pi_v^{int} \leq K \qquad \forall S \in \mathcal{S}$$

  determines a valid lower bound $K^{-1} \cdot \sum_{v \in V} \pi_v^{int}$ for $\chi_f(G)$.
- We solve the MWSS problem with scaled integers

$$\begin{aligned} \pi_v^{int} &:= \lfloor \pi_v K \rfloor \\ \text{with} \qquad K &:= \left\lfloor \frac{\text{INT\_MAX}}{|V|} \right\rfloor. \end{aligned}$$

- Now, column generation terminates with a lower bound

$$\underline{\chi_f}(G) \leq \chi_f(G).$$

Safe Computations with Unreliable Floating Point Operations

Lemma

The difference between $\underline{\chi_f}(G)$ and the floating point representation of $\chi_f(G)$ returned by the LP-solver is at most

$$\frac{|V|^2}{\text{INT\_MAX}}.$$

An upper bound for $\chi_f(G)$

Using an exact LP-solver on the final restricted LP (CLP-r) gives a good upper bound $\overline{\chi_f}(G)$ for $\chi_f(G)$, and an interval

$$\chi_f(G) \in [\underline{\chi_f}(G), \overline{\chi_f}(G)].$$

We found $\lceil \underline{\chi_f}(G) \rceil = \lceil \overline{\chi_f}(G) \rceil$ for all DIMACS instances we could solve.

Computational Results on Open DIMACS Instances

Fractional chromatic numbers

Instance	$\|V\|$	$\|E\|$	$\lceil\chi_f(G)\rceil$	$\omega(G)$	old LB	old UB	Time (sec.)
DSJC250.5	250	15668	26	12	26	28	18
DSJC250.9	250	27897	71	42	71	72	8
DSJC500.1	500	12458	*	5	6	12	*
DSJC500.5	500	62624	43	13	16	48	439
DSJC500.9	500	224874	123	54	123	126	100
DSJC1000.1	1000	49629	*	6	6	20	*
DSJC1000.5	1000	249826	73	14	17	83	142014
DSJC1000.9	1000	449449	215	63	215	223	5033
r1000.1c	1000	485090	96	89	96	98	2634
C2000.5	2000	999836	*	16	16	148	*
C4000.5	4000	4000268	*	17	17	271	*
latin_square_10	900	307350	90	90	90	98	76
abb313GPIA	1557	65390	8	8	8	10	3391
flat1000_50_0	1000	245000	50	14	14	50	3331
flat1000_60_0	1000	245830	60	14	14	60	29996
flat1000_76_0	1000	246708	72	14	14	82	190608

* : not finished within 10 hours.

Computational Results on Open DIMACS Instances

Instances where $\lceil \chi_f(G) \rceil$ was unknown

Instance	$\lvert V \rvert$	$\lvert E \rvert$	$\lceil \chi_f(G) \rceil$	$\omega(G)$	old LB	old UB	Time (sec.)
wap01a	2368	110871	41	41	41	45	20643
wap02a	2464	111742	40	40	40	44	236408
wap03a	4730	286722	*	40	40	50	*
wap04a	5231	294902	*	40	40	46	*
wap06a	947	43571	40	40	40	43	382
wap07a	1809	103368	40	40	40	45	25911
wap08a	1870	104176	40	40	40	45	18015
1-Insertions_5	202	1227	3	2	3	6	33
1-Insertions_6	607	6337	3	2	3	7	1167
2-Insertions_4	149	541	3	2	3	5	13
2-Insertions_5	597	3936	3	2	3	6	851
3-Insertions_4	281	1046	3	2	3	5	59
3-Insertions_5	1406	9695	3	2	3	6	6959
4-Insertions_4	475	1795	3	2	3	5	262
4-FullIns_5	4146	77305	7	6	7	9	33
5-FullIns_4	1085	11395	8	7	8	9	3

* : not finished within 10 hours.

Computational Results on Open DIMACS Instances

Branch & Price

Instance	LB	UB	B&B nodes	Time (seconds)
DSJC250.9	72	72	3225	11094
DSJC1000.9	216	223	29	12489

Lower bounds from dense induced subgraphs $G[X]$, $X \subset V$

Instance	$\lvert X\rvert$	$\lvert E(G[X])\rvert$	$\lceil \chi_f(G[X])\rceil$	old LB	UB	Time
DSJC500.1	300	5436	9	6	12	16 days
DSJC1000.1	350	8077	10	6	20	$<$ 36 days
C2000.5	1400	502370	99	16	148	$<$ 24 days
C4000.5	1500	589939	107	17	272	$<$ 26 days
wap03a	2500	164008	40	40	48	$<$ 3 days
wap04a	2500	159935	40	40	46	$<$ 1 days

Computational Results for the MWSS Problem

Instance	$\rho(G)$ in %	CPLEX 12.2 STD	CPLEX 12.2 LB	CLIQUER 1.2 STD	CLIQUER 1.2 LB	New Algoritm STD	New Algoritm LB
1-Insertions_6	3.4	2969	2065	***	***	61	65
2-Insertions_5	2.2	7	1	***	***	115	1
3-Insertions_5	0.9	4	1	***	***	10576	7
4-Insertions_4	1.5	3	1	***	***	14	1
C2000.5.1029	50	***	***	***	30586	***	11373
DSJC1000.1.3915	9.9	***	***	***	***	***	***
DSJC1000.5	50	***	***	1076	1057	3634	3547
DSJC1000.9	89.9	***	***	1	1	2	2
DSJC250.1	10.3	***	16288	***	***	5941	2281
DSJC250.5	50.3	2737	2557	1	1	1	1
DSJC250.9	89.6	319	317	1	1	1	1
DSJC500.1.117	9.9	***	***	***	***	***	***
DSJC500.5	50.1	***	***	9	9	32	32
DSJC500.9	90.1	24318	22105	1	1	1	1
flat1000_50_0	49	***	***	50	52	765	752
flat1000_60_0	49.2	***	***	262	273	1943	1916
flat1000_76_0	49.3	***	***	1332	1444	4349	4320
unsolved		10	9	8	7	3	2

Thank You!