Exact Algorithms for the Vertex Coloring Problem

Branch and Bound:
Algorithm DSATUR, Brélaz (*Comm. ACM*, 1979);

Branch-and-Price:
Mehrotra, Trick (*INFORMS J. on. Computing*, 1996),

Branch and Cut:
Maximal Clique

- A **clique** $K$ of a graph $G$ is a complete subgraph of $G$.
- A clique is **maximal** if no vertex can be added still having a clique.

- The cardinality of any (maximal) clique of graph $G$ represents a *Lower Bound* for the problem.

- A fast *greedy algorithm* (Johnson, *J. Comp. Syst. Sci.* 1974) can be used to compute a maximal clique $K$ of $G(V,E)$:

  Given an ordering of the vertices, consider the candidate vertex set $W$. Set $W = V$, $K = \emptyset$ and iteratively:

  * Choose the vertex $v$ of $W$ of maximum degree and add it to the current clique $K$.
  * Remove from $W$ vertex $v$ and all the vertices not adjacent to the current clique $K$.

- Different orderings of the vertices generally produce different maximal cliques.
ILP models for VCP: Model VCP-ASSIGN

- Binary variables:
  \[ x_{ih} = \begin{cases} 
  1 & \text{if vertex } i \text{ has color } h \\
  0 & \text{otherwise} 
  \end{cases} \quad i = 1, \ldots, n \]
  \[ y_h = \begin{cases} 
  1 & \text{if color } h \text{ is used} \\
  0 & \text{otherwise} 
  \end{cases} \quad h = 1, \ldots, n \]

\[
\min \sum_{h=1}^{n} y_h 
\]

(1)

\[
\sum_{h=1}^{n} x_{ih} = 1 \quad i = 1, \ldots, n \quad (2)
\]

\[
x_{ih} + x_{jh} \leq y_h \quad \forall i, j : (i, j) \in E \quad h = 1, \ldots, n \quad (3)
\]

\[
x_{i,h} \in \{0,1\} \quad i = 1, \ldots, n \quad h = 1, \ldots, n \quad (4)
\]

\[
y_h \in \{0,1\} \quad h = 1, \ldots, n \quad (4)
\]
Independent Sets

- An *Independent Set* (or *Stable Set*) of $G = (V, E)$ is a subset of $V$ such that there is no edge in $E$ connecting a pair of vertices.
- It is **maximal** if no vertex can be added still having an independent set.

For VCP: all the vertices of an independent set can have the same color

*Feasible coloring* $\rightarrow$ *partitioning* of the graph into independent sets.
Independent Sets and Cliques

- Given a graph $G = (V, E)$

Its “complementary graph” $\bar{G} = (V, \bar{E})$, with $\bar{E} = \{(i, j): (i, j) \notin E\}$

- Independent set of $G$ $\rightarrow$ clique of $\bar{G}$ (and viceversa)

- Clique of $G$ $\rightarrow$ independent set of $\bar{G}$ (and viceversa)
Set Covering Formulation  SC -VCP

\[
\begin{align*}
\min & \sum_{s \in S} x_s \\
\text{s.t.} & \\
\sum_{s: i \in s} x_s & \geq 1 \quad \forall i \in V \\
x_s & \in \{0,1\} \quad \forall s \in S
\end{align*}
\]

- \( S \) can be defined as the family of all the \textit{maximal Independent Sets} (or \textit{Stable Sets}) of graph \( G \).
- The \textit{LP Relaxation} of this formulation leads to \textit{tight lower bounds}, and \textit{symmetry} in the solution is avoided, but the number of maximal independent sets (i.e. the number of “columns”) can be \textit{exponential} in the number of vertices \( n \).
Set Covering Formulation SC-VCP

Branch-and-Price Algorithms

**SC-VCP: Master Problem**

- **LP Relaxation of SC-VCP**: exponentially many variables (columns, independent sets).
- **Column Generation procedure**:
  
  Solve the **LP Relaxation of the SC-VCP formulation** by considering a subset of independent sets (columns): **Restricted Master Problem (RMP)**;

  Detect possible negative reduced cost columns by solving the corresponding **“Pricing Problem”**, add them to the **RMP** and iterate.
**Pricing Problem**

- \( c_i \) is the *optimal dual variable* associated with the \( i \)-th “covering constraint” in the SC-VCP formulation (weight of vertex \( i \)).

The Pricing Problem requires the solution of a *Maximum Weighted Independent Set Problem (MWISP)* (NP-Hard).

- \( y_i = 1 \) if vertex \( i \) is in the independent set, 0 otherwise

\[
\max \sum_{i=1}^{n} c_i \cdot y_i \\
y_i + y_j \leq 1 \quad \forall i, j : (i, j) \in E \\
y_i \in \{0, 1\} \quad i = 1, \ldots, n
\]
**Pricing Problem (MWISP) (2)**

- $c_i$ is the optimal dual variable associated with the $i$-th “covering constraint” in the SC-VCP formulation.
- $y_i = 1 \{ \text{if vertex } i \text{ is in the independent set} \}, 0 \text{ otherwise}$

$$\max \sum_{i=1}^{n} c_i y_i$$

$$y_i + y_j \leq 1 \quad \forall i, j : (i, j) \in E$$

$$y_i \in \{0,1\} \quad i = 1,\ldots,n$$

If the optimal solution value is greater than 1, then an independent set (column) with negative reduced cost has been found.
Column Generation procedure:
detection of possible negative reduced cost columns by alternatively solving:


Branch-and-Price Algorithm 2
(Malaguti, Monaci, T.; Discrete Optimization 2010)

Column Generation procedure:
detect possible negative reduced cost columns by solving MWISP, by using:

1) a Tabu Search heuristic algorithm (TS) which produces maximum weighted independent sets;

2) an ILP Solver (CPLEX 10.2), if the algorithm TS fails in finding negative reduced cost columns.
Branch-and-Cut Algorithm

- **Improvement of the VCP-ASSIGN Formulation by adding new valid inequalities.**

- **Initialization Phase:**
  - Preprocessing procedure to reduce the number of vertices to be considered.
  - Initial Upper Bound computed through the execution of algorithm DSATUR with a short time limit (5 seconds).
  - Initial Lower Bound computed by finding a maximal clique through a greedy algorithm.

- **New Branching Rules.**
Computational Results for the Exact Approaches


- **Branch and Cut Algorithm BC-COL**: (with the stronger lower bounding procedures):


- Comparable CPU times.
DIMACS Benchmark Instances

Johnson, Trick, 2\textsuperscript{nd} DIMACS Implementation Challenge 1993

- **DIMACS benchmark graph instances**
  compose a variety of graph classes used for evaluating the performance of VCP algorithms:

- random graphs: DSJC\_n.x;
- geometric random graphs: DSJR\_n.x; r\_n.x;
- quasi-random graphs: flat\_n.x;
- artificial graphs: le\_n.x; latin\_square\_10;
  Queen\_rn.rn; myciel\_k
- real-world application-related graphs.
## VCP: Exact Algorithms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BC-COL</td>
<td>DSATUR</td>
<td></td>
</tr>
<tr>
<td>common instances</td>
<td></td>
<td></td>
<td>39</td>
</tr>
<tr>
<td>(w.r.t. M-M-T)</td>
<td>66</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>proven optimal solutions</td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>only algorithm</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Lower Bound</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LB &gt; LB (MMT)</td>
<td>9</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>LB = LB (MMT)</td>
<td>43</td>
<td>23</td>
<td>27</td>
</tr>
<tr>
<td>LB &lt; LB (MMT)</td>
<td>14</td>
<td>42</td>
<td>10</td>
</tr>
<tr>
<td>global LB gap w.r.t. MMT</td>
<td>92</td>
<td>326</td>
<td>*</td>
</tr>
<tr>
<td>Upper Bound</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UB &lt; UB (MMT)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UB = UB (MMT)</td>
<td>24</td>
<td>18</td>
<td>27</td>
</tr>
<tr>
<td>UB &gt; UB (MMT)</td>
<td>42</td>
<td>48</td>
<td>12</td>
</tr>
<tr>
<td>global UB gap w.r.t. MMT</td>
<td>333</td>
<td>304</td>
<td>*</td>
</tr>
</tbody>
</table>