Matheuristic Variants of DSATUR for the Vertex Coloring Problem [1]

Federico Bruzzone,[1] PhD Candidate

Milan, Italy – 4 December 2025

[1] ADAPT Lab – Università degli Studi di Milano,
Website: federicobruzzone.github.io,
Github: github.com/FedericoBruzzone,
Email: federico.bruzzone@unimi.it,
Slides: federicobruzzone.github.io/activities/presentations/matheuristic-variants-of-DSATUR-for-the-vertex-coloring-problem.pdf

Notation

- $G = (V, E)$ is an undirected graph with vertex set $V = \{v_1, ..., v_n\}$ and edge set E, and denote $I = [\![1; n]\!] = [1; n] \cap \mathbb{Z}$.
- An edge $e = (v_i, v_j) \in E$ with $i < j$ links the underlying vertices (*for* ***VCP*** *there is no sense to consider loop/multiple edges*).
- For $i \in I$, δ_i denotes the set of neighbors of vertex v_i, and $d_i = |\delta_i| = \{j \in I \mid \text{ngb}(v_i, v_j) = 1\}$, where $\text{ngb}(v_i, v_j) = 1 \; iif \; (v_i, v_j) \in E$.[2]

[2]ngb stands for "neighbor".

A k-coloring of G

- It is an assignment of colors to vertices such that no two adjacent vertices share the same color.
- **VCP** consists in finding a k-coloring of G using the minimum number of colors k (the *chromatic number* $\chi(G)$).
- A valid k-coloring (c) fulfills: $\forall i < j, (v_i, v_j) \in E \Longrightarrow c_i \neq c_j$ where $c_* = [\![1; k]\!]$ is the color of v_*.

A partial k-coloring of G

- If a vertex may not be colored, we set $c_i = -1 \ s.t. \ c_i \in [\![1; k]\!] \cup \{-1\}$
- A partial k-coloring (c) is *feasible* if $\forall i < j, (v_i, v_j) \in E \implies c_i \neq c_j \lor c_i = c_j = -1$.
- Given v_i the saturation table S_i is the set of colors assigned to its colored neighbors: $S_i = \bigcup_{j \in \delta_i}^{n} \{c_j\} \setminus \{-1\}$, and $s_i = |S_i|$ is the saturation degree.
- A total order $\succcurlyeq$ over V is defined as: $v_i \succcurlyeq v_j \iff s_i > s_j \lor (s_i = s_j \land d_i \geq d_j)$

Compact ILP Formulations[3], feasible *iif* $\chi(G) \leq k$

$z_{i,c} \in \{0,1\}$ indicates if vertex v_i is assigned color c.

$y_c \in \{0,1\}$ indicates if color c is used in the coloring.

$$\min \sum_{c=1}^{k} y_c$$

$$s.t. \sum_{c=1}^{k} z_{i,c} = 1 \qquad \forall i \in I$$

$$z_{i,c} + z_{j,c} \leq y_c \qquad \forall (v_i, v_j) \in E, \ \forall c \in [\![1;k]\!]$$

The objective minimizes the number of used colors.

The 1st set ensures that each vertex is assigned exactly one color.

The 2nd set ensures that adjacent vertices do not share the same color.

[3] Efficient formulations: extended column generation by Furini and Malaguti [2], and reduced formulation to MWSSP by Cornaz et al. [3].

Observations

- Having an upper bound of the chromatic number as the initial value k (or simply $k = |V|$) guarantees the optimality of the solution.
- The size of k strongly affects the performance of ILP solvers.
- Symmetries in the model (e.g., colors are permutable) enlarge the search space for Branch-and-Bound algorithms (the same solution can be represented in multiple ways)

Representative ILP Model[4], asymmetric and easily to LP-relax

$x_{i,i'} \in \{0,1\}$, $\forall i, i' \in V s.t. i \leq i'$, indicates if verticies v_i and $v_{i'}$ share the same color and i is the minimum index of its color class.

$$\min_z \sum_{i=1}^{n} x_{i,i}$$

$$s.t. \sum_{i' \leq i} x_{i',i} \geq 1 \qquad \forall i \in I$$

$$x_{j,i} + x_{j,i'} \leq x_{j,j} \qquad \forall (v_i, v_{i'}) \in E, \forall j \leq i \leq i'$$

The objective counts the number of representative vertices (i.e., used colors).

The 1st set ensures either $x_{i,i} = 1$ (i is representative) or i's representative is a previous vertex $i' < i$ (all vertices must be colored).

The 2nd set expresses the color incompatibility between adjacent vertices and $x_{j,*} = 1 \implies x_{j,j} = 1$

[4] A vertex is representative of its color class if it has the minimum index among the vertices sharing the same color.

Standard DSATUR Algorithm

Algorithm 1: Standard DSATUR algorithm

Input: $G = (V, E)$ a non-empty and non-oriented graph
Initialization:
  define partial coloring c with $c_i := -1$ for all $i \in I$
  define saturation table S with $S_i := \emptyset$ for all $i \in I$
  initialize set $U := V$, and color $k := 0$
  while $U \neq \emptyset$
    find $u \in U$, a maximum of $\succcurlyeq$ in U.
    if $|S_u| = k$ **then** $k{:=}k + 1$ // a new color is added
    compute $c_i := \min S_u$ // assign color to u
    remove u from U
    for all $i \in \delta_u \cap U$, $S_i = S_i \cup \{c_i\}$ // update saturation
  end while
return color k and (c) a k-coloring of G

- DSATUR is an adaptive greedy heuristic proposed by Brélaz [4], which colors vertices iteratively.
- Selection of the uncolored verex to color is given with order $\succcurlyeq$, maximizing first the saturation degree and secondly the degree.
- Coloring a new vertex updates saturation, the iteration order of vertices is thus adaptive.

DASTUR Matheuristic Variants[5]

[5] N. Dupin, "Matheuristic Variants of DSATUR for the Vertex Coloring Problem," in *Metaheuristics* 2024 [1]

Initialization

Defining an initial partial coloring and computing the saturation table for the uncolored vertices, **before** starting the main DSATUR iterations.

Variants:

1. `maxDeg`: color the vertex with the maximum degree — equivalent to standard DSATUR by definition of $\succcurlyeq$, it would suffer from many ties;
2. `col-`n: cosider n vertices having the maximum degree and color them solving a representative ILP model for the *induced* subgraph — more depth pre-processing, it tries to prevent erroneous decisions in the initial steps of DSATUR;
3. `clq`: find a maximum clique[6] and color it with different colors — an exact pre-processing (not heuristic), it leads to a better initial saturation table S for the uncolored vertices;
4. `clq-col-`n: combine `clq` and `col-`n — best of both worlds.

[6] It is NP-hard, an heuristic can be used.

Local Optimization with Larger Neighborhoods

Let (c) be a partial k-coloring, where k is the number of colors used until now.

- $C = \{i \in I \mid c_i > 0\}$ is the set of colored vertices in (c).
- $U \subset \{i \in I \mid c_i = -1\}$ is a subset of uncolored vertices in (c).

We want to define an ILP formulation to assign a color to each vertex $u \in U$ while preserving the colors of vertices in C.

An **hybrid** formulation of assignment-based and representative-based formulations is used.

Matheuristic DSATUR Formulation

$$
\begin{aligned}
\min_{z} \sum_{u \in U} x_{u,u} & \\
s.t. \quad z_{i,l} + z_{i',l} \leq 1 & \quad \forall (v_i, v_{i'}) \in E_U, \\
& \quad \forall l \in [\![1; k]\!] \\
x_{u,i} + x_{u,i'} \leq x_{u,u} & \quad \forall (v_i, v_{i'}) \in E_U, \\
& \quad \forall u \in U, u \leq i \\
\sum_{i' \in U : i' \leq i} x_{i',i} + \sum_{l \in K_u} z_{i,l} \geq 1 & \quad \forall u \in U
\end{aligned}
$$

- Binary variables $x_{u,u'}$ are defined only for $u \leq u' \in U$, when considering $E_U = \{(v_u, v_{u'})\}_{u<u' \in U} \subset E$.
- Binary variables $z_{u,l}$, to assign previous colors, are defined for $u \in U$ and $l \in [\![1; k]\!]$ *s.t.* no neighbor u has color l in (c) – i.e., for all $u \in U$ and $l \in K_u$, where $K_u = \{l \in [\![1; k]\!] \mid \forall i \in C, c_i = l \Longrightarrow \mathrm{ngb}(i, j) = 0\}$

Matheuristic DSATUR Formulation

$$\min_z \sum_{u\in U} x_{u,u}$$

$$s.t. \quad z_{i,l} + z_{i',l} \leq 1 \qquad \forall (v_i, v_{i'}) \in E_U, \ \forall l \in [\![1; k]\!]$$

$$x_{u,i} + x_{u,i'} \leq x_{u,u} \qquad \forall (v_i, v_{i'}) \in E_U, \ \forall u \in U, u \leq i$$

$$\sum_{i'\in U : i' \leq i} x_{i',i} + \sum_{l \in K_u} z_{i,l} \geq 1 \qquad \forall u \in U$$

- The 1st set ensures that adjacent vertices in U do not share the same existing color l.
- The 2nd set ensures that two adjacent vertices in U cannot share the same representative color.
- The 3rd set ensures, $\forall i \in U$, that either it receives a previous color l in K_u or it receives a new color represented by another vertex i' in U with $i' \leq i$.

Matheuristic DSATUR Algorithm

Algorithm 2: Matheuristic DSATUR variants

Input: $G = (V, E)$ a non-empty and non-oriented graph
Parameters:
- an initialization strategy $\mathcal{S}$ (from Sect. 3.1) ;
- $o \in \mathbb{N}$, $o > 1$;
- $r \in \mathbb{N}$.

Initialization:
initialize colored set C, and color k with strategy $\mathcal{S}$.
initialize $W := V \setminus C$.
update partial coloring c and saturation table S with strategy $\mathcal{S}$.
while $W \neq \emptyset$
  sort W with order $\succcurlyeq$.
  define U_1 as the o first elements after sorting.
  define U_2 as the elements of rank $o+1$ and $\min(|W|, o+r)$ after sorting.
  solve ILP (15) with C and $U = U_1 \cup U_2$.
  $k := k + OPT$ where OPT is the optimal value of the last ILP.
  if $o + r \leqslant |W|$ **then** $U_1 = U$ **end if**
  set $W := W \setminus U_1$
  assign colors c_u of the ILP for $u \in U_1$
end while
return color k and (c) a k-coloring of G

- $\mathcal{S}$ for *initialization* induces k, C, S, c, and W.
- Simultaneously colors o vertices solving the matheuristic DSATUR ILP formulation (the standard DSATUR have $o = 1$ and $r = 0$).
- Having $r > 0$ ensures more depth in the local search and the possibility to reoptimize in later iterations (set $W := W \setminus U_1$).
- U_2 helps the ILP in having context when coloring critical vertices U_1.[7]
- $o + r \geq |W|$ holds in the last iteration, and $U_1 = U = W$ ensures both termination ($W \setminus U_1 = \emptyset$) and efficiency (no useless re-optimization—i.e., recoloring r vertices).

[7] $o + r$ should be fine-tuned according to the ILP solver capabilities and instance features.

Evaluation

- CPLEX 20.1 with its default parameter, except `CPX_PARAM_EPAGAP = 0.9` to stop computation to optimality knowing the objective function is integer, a time limit, and also no display in screen.
- A subset of 53 DIMACS instances removing easy instances for DSATUR (i.e., those solved optimally).
- `maxDeg`: The baseline algorithm starting with the maximum degree node.
- `col-`n: Results were generally disappointing compared to the baseline.
- `clq` : Significantly outperforms standard DSATUR by identifying the graph's "hardest" core first.
- `clq-col-80`: Providing a significant improvement over the original approach.
- Best clq: The top result achieved by selecting either the clq or clq-col-80 variant for each instance.
- Best clq+DSATUR: Highlighting the synergy between old and new methods.
- Best-DSATUR: Excluding the original algorithm.
- Best+DSATUR: Confirming that standard DSATUR is still superior for specific instances.

Comparison of DSATUR matheuristics

	#colors	gap	#BKS	#worse	#better	Q1	Q2	Q3
`maxDeg`	3240	32.03 %	1	0	0	0	0	0
`col-60`	3251	32.48 %	1	19	16	−1	0	1
`col-80`	3250	32.44 %	2	20	16	−1	0	1
`clq-col-80`	3214	30.97 %	2	18	17	−1	0	1
`clq`	3209	30.77 %	4	13	19	−1	0	0
Best `clq`	3181	29.63 %	6	7	26	−1	0	0
Best `clq`+DSATUR	3174	29.34 %	6	0	26	−1	0	0
Best-DSATUR	3163	28.89 %	6	3	34	−2	−1	0
Best+DSATUR	3160	28.77 %	6	0	34	−2	−1	0
BKS	2454	0.00 %	53	0	52	−14	−5	−3

- Using a maximum clique to initialize saturation drastically reduces the number of colors needed from the very first steps, avoiding early errors inherent in the greedy version.
- While `clq-col-n` provides the best results in terms of solution quality (lower k), it requires higher initial computation time due to the exact resolution of subgraphs.

Comparison with Larger Local Optimization

Init satur	o	r	#colors	gap	#BKS	#worse	#better	Q1	Q2	Q3
`maxDeg`	1	0	3240	32.03 %	1	0	0	0	0	0
`col-80`	1	0	3250	32.44 %	2	20	16	-1	0	1
`col-80`	20	60	3181	29.63 %	6	12	30	-3	-1	0
`col-80`	40	40	3218	31.13 %	5	20	26	-2	0	1
`col-80`	80	0	3322	35.37 %	2	35	13	0	1	2
`clq`	1	0	3209	30.77 %	4	13	19	-1	0	0
`clq`	40	40	3155	28.57 %	10	9	32	-3	-1	0
Best Clq			3134	27.71 %	10	4	37	-3	-1	0
Best-DSATUR			3125	27.34 %	10	3	40	-3	-2	-1
Best+DSATUR			3122	27.22 %	10	0	40	-3	-2	-1
BKS			2454	0.00 %	53	0	52	-14	-5	-3

- Depth and Re-optimization: Using $r > 0$ allows coloring the most critical vertices (U_1) while maintaining vision over their neighbors (U_2), reducing the "threshold effects" typical of standard DSATUR ($o = 1, r = 0$).
- As $o + r$ increases, the algorithm approaches an exact solver, but computational time grows; the matheuristic finds an optimal balance for medium-sized instances.

Thank You!

Bibliography

[1] N. Dupin, "Matheuristic Variants of DSATUR for the Vertex Coloring Problem," in *Metaheuristics*, M. Sevaux, A.-L. Olteanu, E. G. Pardo, A. Sifaleras, and S. Makboul, Eds., Cham: Springer Nature Switzerland, 2024, pp. 96–111.

[2] F. Furini and E. Malaguti, "Exact weighted vertex coloring via branch-and-price," *Discrete Optimization*, vol. 9, no. 2, pp. 130–136, 2012.

[3] D. Cornaz, F. Furini, and E. Malaguti, "Solving vertex coloring problems as maximum weight stable set problems," *Discrete Applied Mathematics*, vol. 217, pp. 151–162, 2017.

[4] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, Apr. 1979.