The Vertex Coloring Problem

Enrico Malaguti – DEIS, University of Bologna

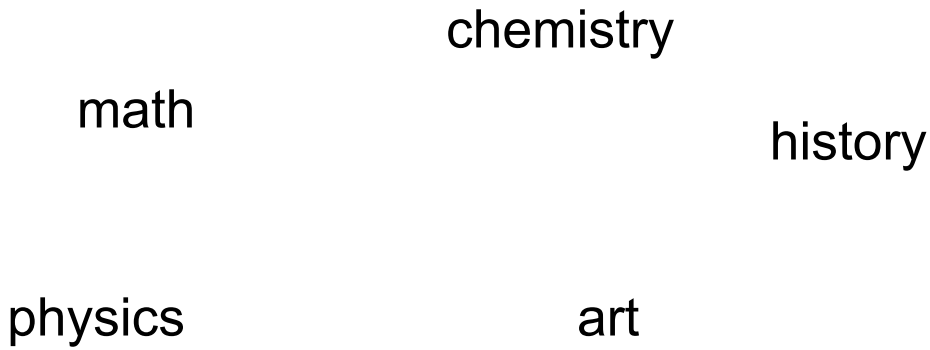
Map Coloring

- Color the map of England, in such a way that no two counties touching with a common stretch of boundary are given the same color, by using the smallest number of colors.

- the Four Color Conjecture was proposed by Francis Guthrie in 1852.

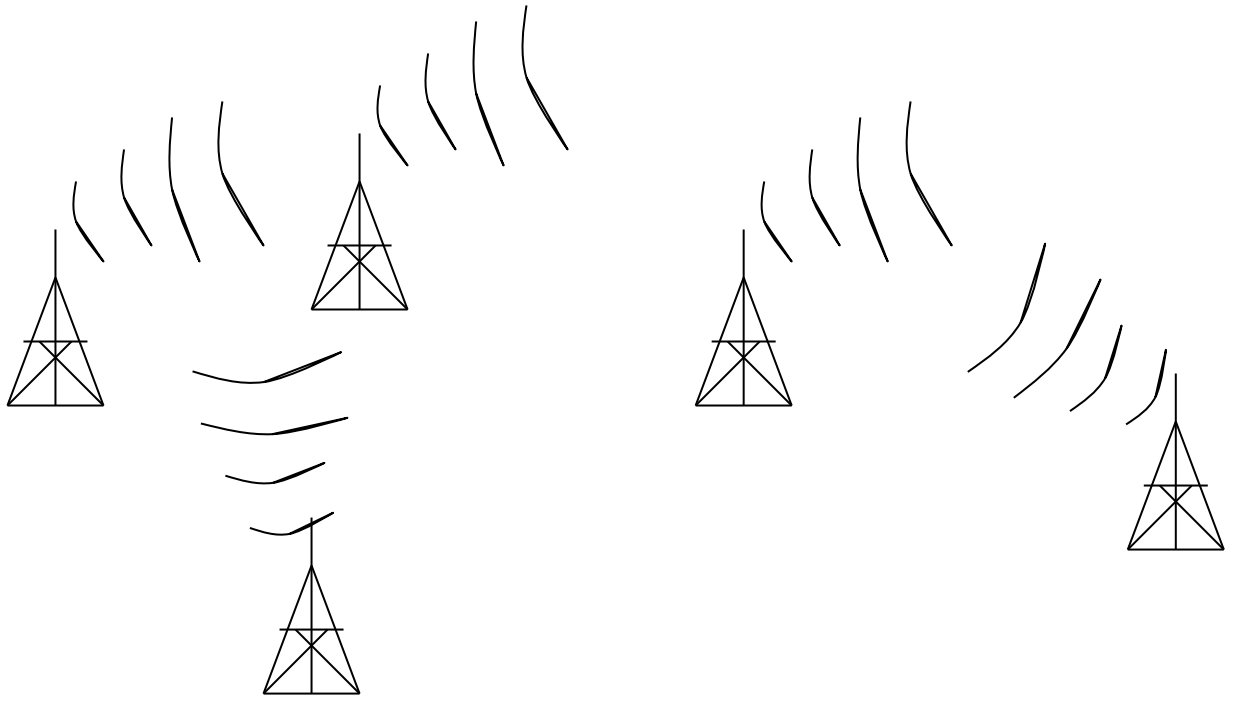
Scheduling

- Consider the following problem: in a university, assign exams to time slots in such a way:
  1) every student can do the exams of the courses he is taking;
  2) the total number of used time slots is minimized.



Frequency assignment

- Problem: assign a **frequency** to broadcast emitting stations in such a way that interfering stations use different frequencies.



Air Traffic Control

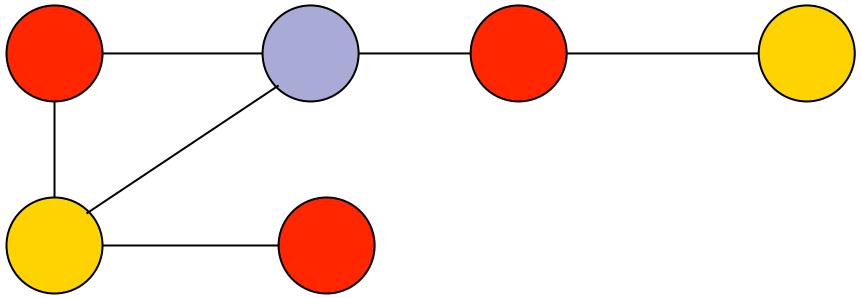
- Aircrafts are approaching an airport. The traffic control system assigns them an altitude, where they wait their landing time. If the arrival intervals of two planes overlap, they cannot use the same altitude. The available altitudes are limited, and have to be assigned efficiently.
- Equivalent: Train Platforming.

Outline

- Vertex Coloring Problem (VCP)
  - Definition
  - Applications
- Constructive Heuristics
- Descriptive Model and Algorithms
- Set-Covering Model and Algorithms
- Meta-heuristics: Tabu Search

Vertex Coloring Problem (VCP)

- Given an undirected graph ***G=(V,E)*** with ***n*** = |***V***| and ***m*** = |***E***|, assign a color to each vertex in such a way that colors on adjacent vertices are different and the number of colors used is minimized . The *chromatic number χ is* the minimum number of colors which can be used.



- A set of vertices receiving the same color is called a ***color class***.

- The **VCP is NP-Hard** (Garey and Johnson, 1979) and has several real-world applications such as: timetabling, register allocation, frequency assignment, scheduling, etc…

VCP applications: scheduling

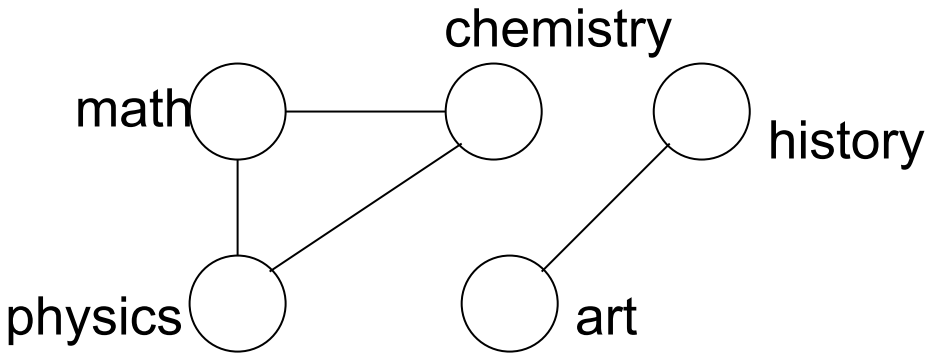
- Consider the following problem: in a university, assign exams to time slots in such a way:
  1) every student can do the exams of the courses he is taking;
  2) the total number of used time slots is minimized.

i.e. Color the graph $G=(V, E)$ where:

V = {exams}

E = {(i,j) s.t. it exists a student taking both courses i and j}

And one color corresponds to a time slot.



VCP applications: scheduling

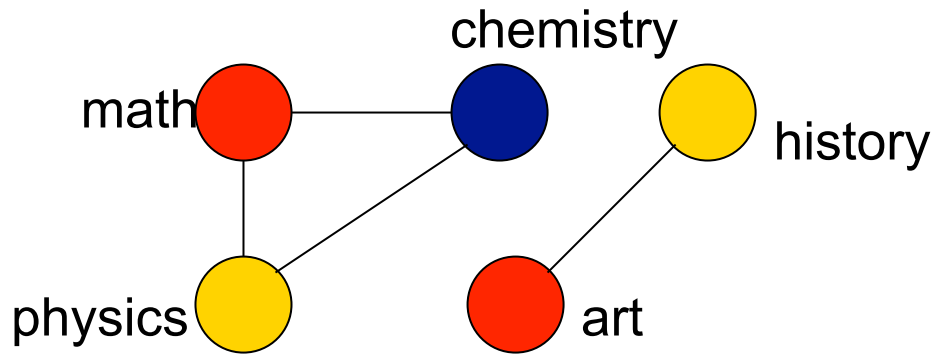
- Consider the following problem: in a university, assign exams to time slots in such a way:
  1) every student can do the exams of the courses he is taking;
  2) the total number of used time slots is minimized.

i.e. Color the graph $G=(V, E)$ where:

V = {exams}

E = {(i,j) s.t. exists a student taking both courses i and j}

And one color corresponds to a time slot.



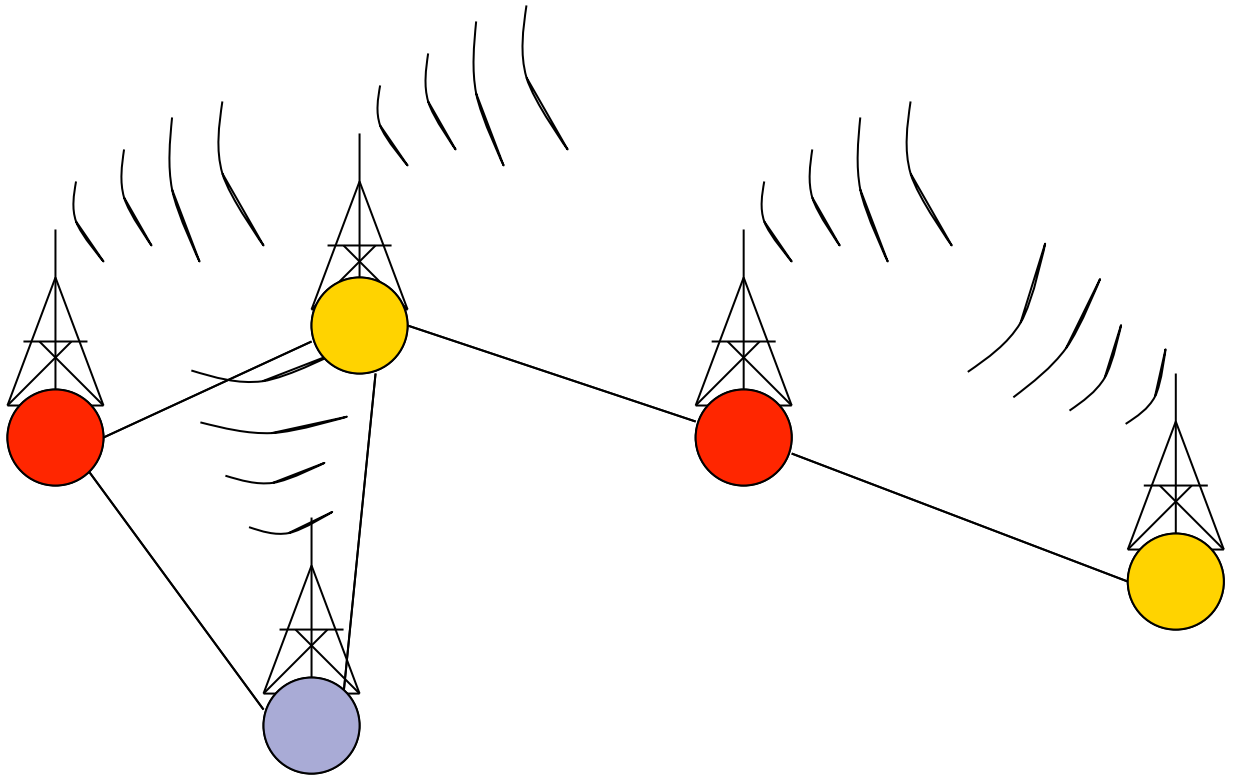
9-11 math, art

11-13 physics, history

15-17 chemistry

VCP applications: frequency assignment

- Problem: assign a **frequency** (color) to broadcast emitting stations in such a way that adjacent (and possibly interfering) stations use different frequencies (colors).



How difficult is the VCP in practice?

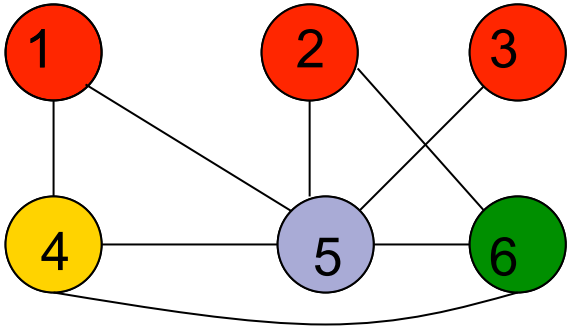
Some NP-Hard problems can be solved to optimality for instances of reasonable size:

TSP-> thousands of nodes (cutting planes)
BPP-> up to 1000 items (Branch and Bound, Branch and Price)

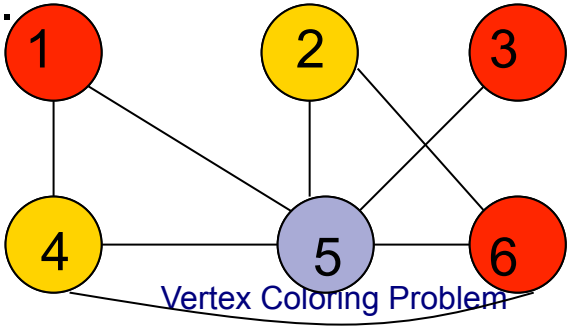
VCP is really difficult from the practical viewpoint: it cannot be consistently solved to optimality for graphs with more than 100 vertices.

Greedy Algorithms for the VCP

SEQ Algorithm: Vertex 1 is assigned to the first color class, and thereafter, vertices 2,…,*n* are assigned to the lowest indexed color class that contains no adjacent vertex.



DSATUR is similar to SEQ, but dynamically chooses the vertex to color next, picking the first vertex that is adjacent to the largest number of distinctly colored vertices (i.e. the vertex with maximum ***chromatic degree***).

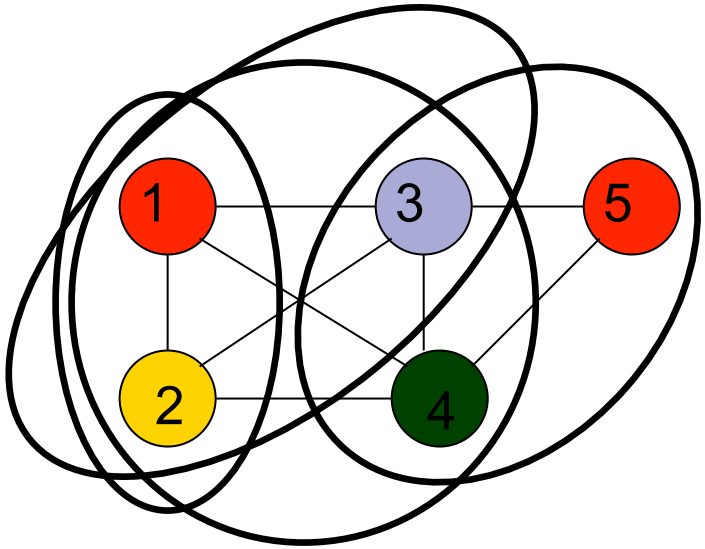


The clique bound

- A ***clique K*** is a complete subgraph of the original graph. The cardinality of a clique is a ***lower bound*** on the chromatic number χ. A clique is ***maximal*** if one can add no vertex still having a clique. The cardinality of the ***maximum*** clique is denoted by ω, thus $\chi \geq \omega$. Computing ω is NP-Hard.

clique k, $|k|=2$

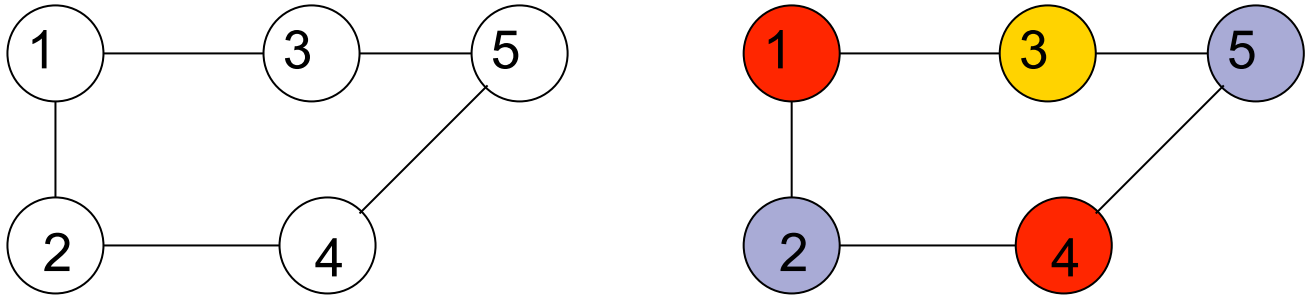
clique k^1, $|k^1|=3$



maximal clique k^2, $|k^2|=4=\omega$

maximal clique k^3, $|k^3|=3$

The clique bound



cardinality of any clique (and of the maximum clique) $|K| = |K_{max}| = 2$

chromatic number $\chi = 3$

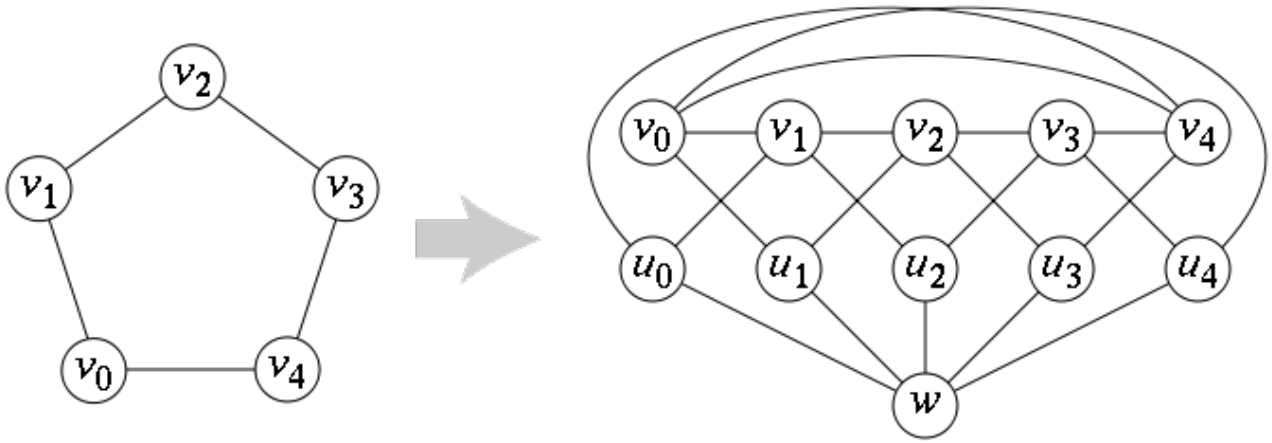
The worst case ratio χ/ω can be arbitrarily bad.

Mycielski graph $\mu(G)$

Let the n vertices of the given graph G be v_0, v_1, etc. The Mycielski graph of G contains G itself, together with n+1 additional vertices: a vertex u_i corresponding to each vertex v_i of G, and another vertex w. Each vertex u_i is connected by an edge to w. In addition, for each edge (v_i,v_j) of G, the Mycielski graph includes two edges, (u_i,v_j) and (v_i,u_j).

if G is triangle free → $\mu(G)$ is triangle free

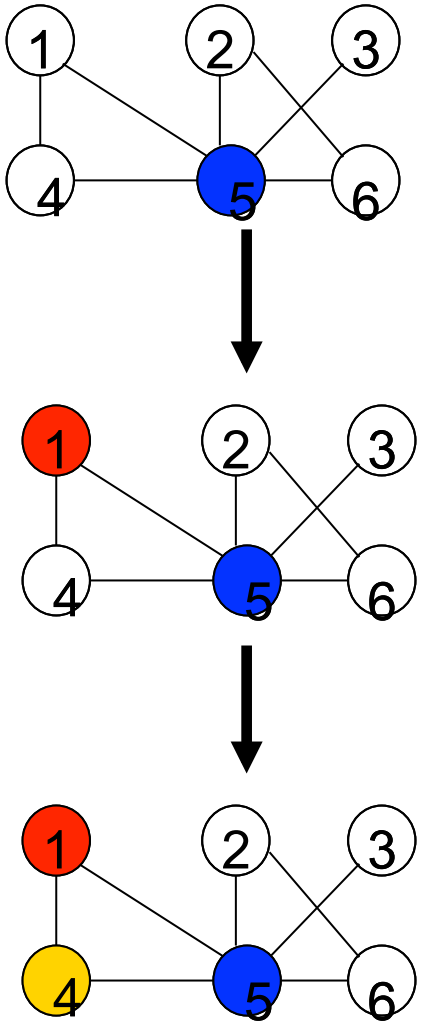
$$\chi(\mu(G)) = \chi(G)+1$$



source: wikipedia

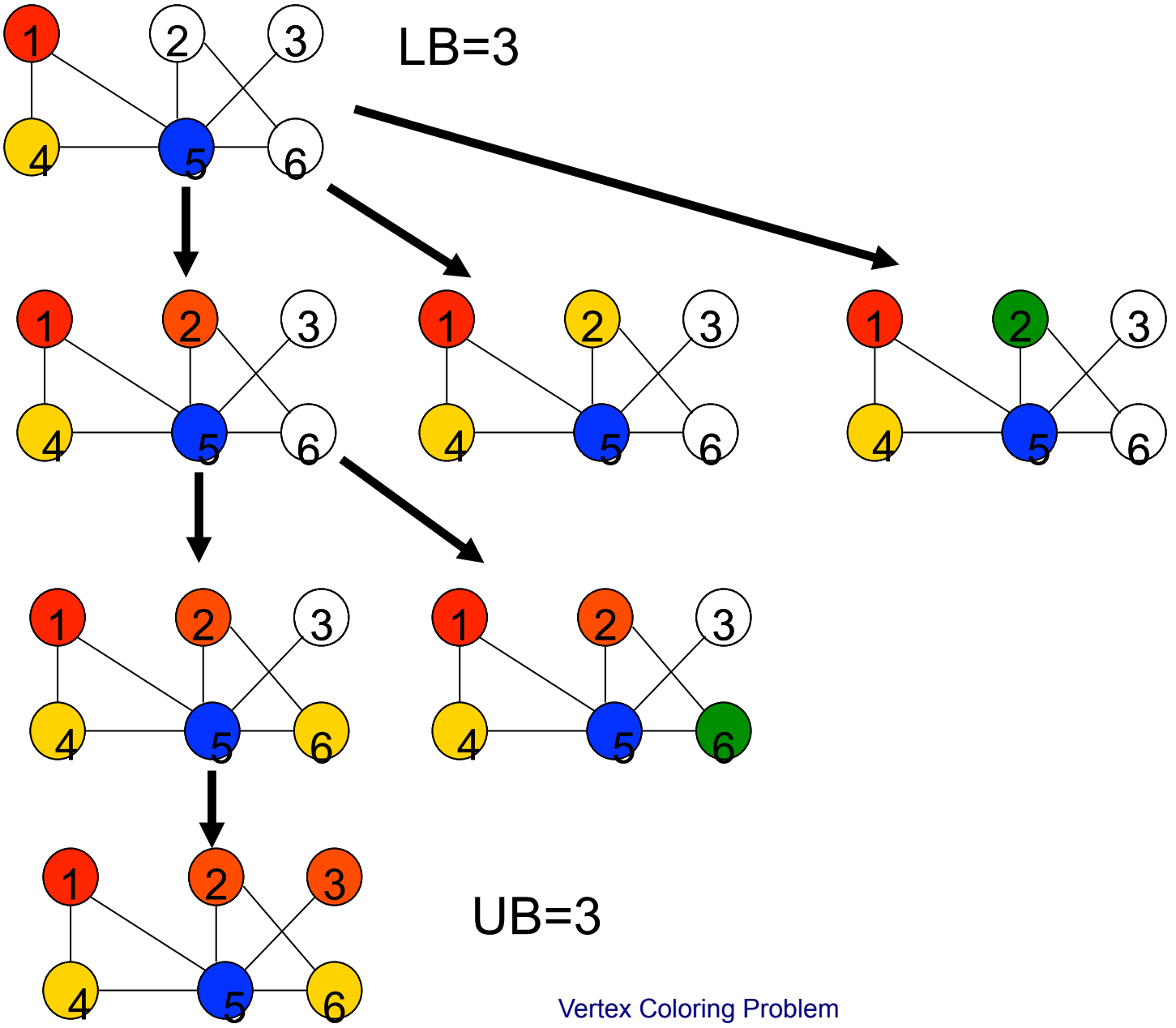
An early exact algorithm

- DSATUR (Branch and Bound - Brèlaz, 1979)
- Let *UB* be an upper bound on χ. Each sub-problem corresponds to a partial coloring of the graph. When this partial coloring uses *k* colors, and $k \geq UB$, the sub-problem can be fathomed;
- When all the vertices are colored and $k < UB$, the upper bound is updated (by setting $UB = k$);
- From each sub-problem using *k* colors, up to $k + 1$ new sub-problems are generated, by assigning, when feasible, one of the *k* colors to the next vertex to be colored, and by coloring it with color $k+1$ if $k+1<UB$;
- At each iteration, the vertex adjacent to the largest number of colors is chosen as next vertex to be colored.
- A lower bound (maximal clique) is obtained at the beginning.



UB=6

LB=3


LB=3
UB=3

ILP model (1) for VCP

- Binary variables:

$$x_{ih} = \begin{cases} 1 \text{ if vertex } i \text{ has color } h & i=1,\ldots,n \\ 0 \text{ otherwise} & h=1,\ldots,n \end{cases}$$

$$y_h = \begin{cases} 1 \text{ if color } h \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

$$\min \sum_{h=1}^{n} y_h \qquad (1)$$

$$\sum_{h=1}^{n} x_{ih} = 1 \qquad i = 1,..,n \qquad (2)$$

$$x_{ih} + x_{jh} \leq y_h \qquad (i,j) \in E, \quad h = 1,\ldots,n \qquad (3)$$

$$x_{i,h} \in \{0,1\} \qquad i = 1,\ldots,n \qquad h = 1,\ldots,n$$

$$y_h \in \{0,1\} \qquad h = 1,\ldots,n \qquad (4)$$

ILP Model (1) for VCP

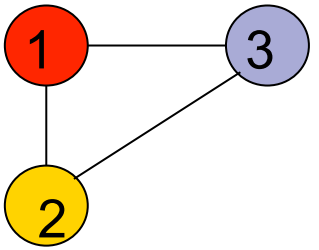
Model (1) is a weak model:

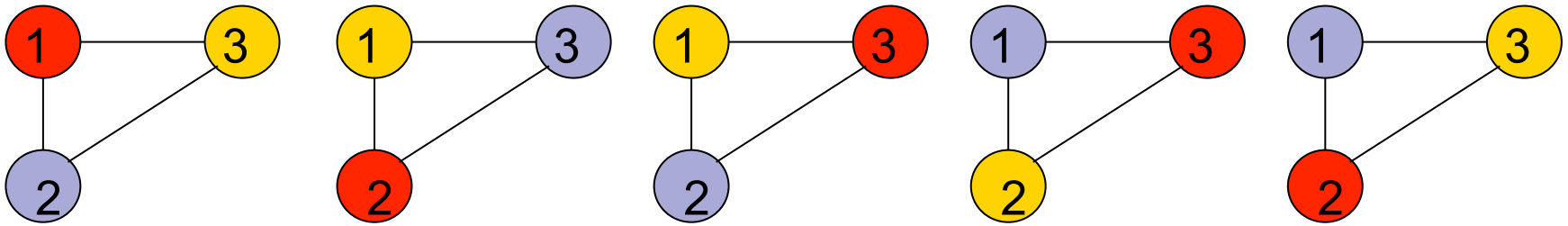
- Its continuos relaxation has the useless solution of value 2: $y_1 = 1,\ y_2 = 1;\quad y_h = 0 \quad h = 3,\ldots,n$

  $x_{i1} = x_{i2} = 1/2 \qquad i=1,\ldots,n$

  $x_{ih} = 0 \qquad i=1,\ldots,n \quad h=3,\ldots,n$

- Every solution of value k (<*n*) has $\binom{n}{k} k!$ equivalent representations, *k!* once the *n* colors have been chosen:





A stronger ILP model (1') for VCP

- Binary variables:

$$x_{ih} = \begin{cases} 1 \text{ if vertex } i \text{ has color } h & i=1,\ldots,n \\ 0 \text{ otherwise} & h=1,\ldots,n \end{cases}$$

$$y_h = \begin{cases} 1 \text{ if color } h \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

$$\min \sum_{h=1}^{n} y_h \qquad (1)$$

$$\sum_{h=1}^{n} x_{ih} = 1 \qquad i = 1,..,n \qquad (2)$$

$$\sum_{i \in K} x_{ih} \le y_h \qquad \forall \max clique K \subseteq V, \quad h = 1,\ldots,n \qquad (3)$$

$$x_{i,h} \in \{0,1\} \qquad i = 1,\ldots,n \qquad h = 1,\ldots,n$$

$$y_h \in \{0,1\} \qquad h = 1,\ldots,n \qquad (4)$$

A stronger ILP model (1') for VCP

$$\sum_{i \in K} x_{ih} \leq y_h \qquad \forall \max clique K \subseteq E, \quad h = 1,...,n \quad (3)$$

Constraints (3) are exponentially many -> cutting planes methods are needed;

The separation of (3) is NP-Hard (however, we can use heuristics);

Let K be the maximum clique of G, and $|K| = k$. The continuos relaxation of (1') has the following solution of value k:

$y_1 = 1, \ldots, y_k = 1; \quad y_h = 0 \quad h = k+1, \ldots, n$

$x_{i1}, \ldots, x_{ik} = 1/k \quad i=1,\ldots,n \quad x_{ih} = 0 \quad i=1,\ldots,n \quad h=k+1,\ldots,n$

Branch and Cut

- Separation methods: useful when the number of constraints of a ***Linear Program*** is exponential, e.g.

$$\sum_{i \in K} x_{ih} \leq y_h \qquad \forall \max cliqueK \subseteq V, \quad h = 1,...,n \quad (3)$$

1) Impose a subset of the constraints,
2) Solve the *LP*, get *x*,*
3) If (at least) one constraint is violated by *x*,* add the constraint(s) to the *LP* and go to 2,
4) *x** is optimal and feasible (possibly fractional).

- Branch and Bound: useful to obtain an *Integer Solution.*
- Branch and Cut: at each node of the Branch and Bound tree, apply separation.

A Branch and Cut Algorithm for the VCP
Méndez-Díaz and Zabala, 2006-08

They solve model (1) by trying to overcome its drawbacks: weak bound, symmetry

The bound provided by the LP relaxation is strengthened by adding: Clique inequalities, Independent Set inequalities, Hole inequalities, Neighborhood inequalities, Block Color inequalities, Multicolor Path inequalities, Multicolor Clique inequalities.

The symmetry of the formulation is reduced by means of symmetry breaking inequalities (in polynomial number):

$y_h \geq y_{h+1}$ $h=1,\ldots,n-1$ color $h+1$ can be used only if color h already used

$$\sum_{i \in V} x_{ih} \geq \sum_{i \in V} x_{ih+1} \quad h=1,\ldots,n-1$$ the cardinality of color class h cannot be smaller than the cardinality of color class $h+1$

A Branch and Cut Algorithm for the VCP
Méndez-Díaz and Zabala, 2006-08

The color classes are sorted by the minimum label of the vertices they include, and only colorings assigning color *h* to the *h*-th color class are considered. This model completely removes the symmetry due to color indistinguishability, and is obtained by imposing the additional constraints:

$$x_{ih} = 0 \qquad h \geq i+1$$

$$x_{ih} \leq \sum_{k=h-1}^{i-1} x_{kh-1} \qquad i \in V \setminus \{1\}, 2 \leq h \leq i$$

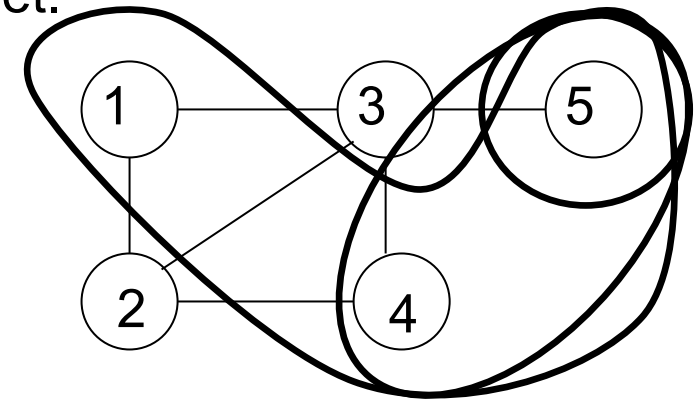
A Branch and Cut Algorithm for the VCP
Méndez-Díaz and Zabala, 2006-08

Branching rules:

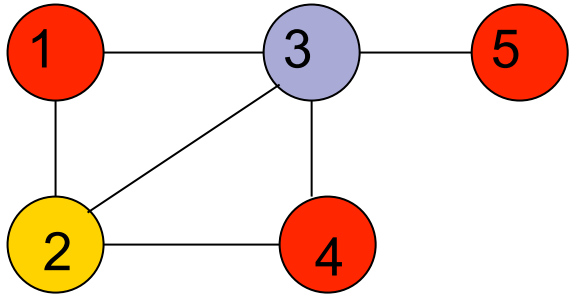
- Binary branching on a fractional variable x_{ih} → very unbalanced
- DSATUR like branching: choose a yet uncolored vertex and then consider all colors used so far. A new sub-problem is then formulated for every feasible assignment of one of these colors to the selected vertex. An additional sub-problem which assigns the first unused color to the vertex, is also created.

Stable Sets

- A ***Stable Set*** of ***G=(V,E)*** is a subset of vertices such that there is no edge in ***E*** connecting any pair. It is ***maximal*** if it is not strictly included in any other stable set.

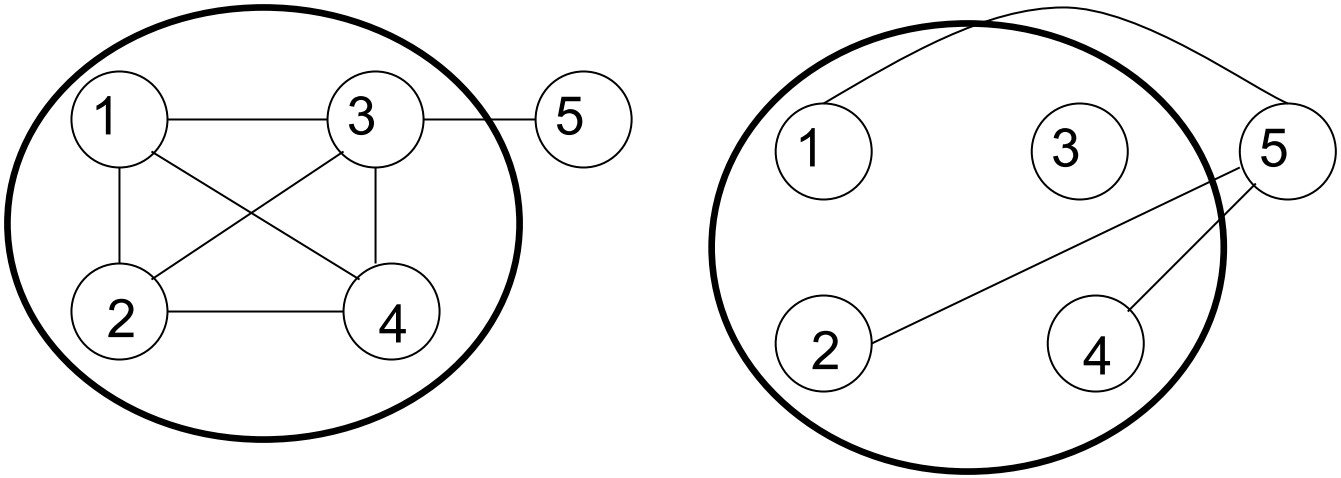


Feasible coloring -> partitioning of the graph in **stable sets.**



Cliques and Stable Sets

- Let $\overline{G} = (V, \overline{E})$ be the complementary graph of G, i.e. $\overline{E} = \{e : e \notin E\}$. If *K* is a clique in *G*, it is a stable set in $\overline{G}$



Set Partitioning Formulation for VCP

- **Feasible coloring ->** partition of the graph in **stable sets.** *S* = family of all stable sets of ***G***
- Binary variables: $x_s = \begin{cases} 1 \text{ if stable set } s \text{ is given a color} \\ 0 \text{ otherwise} \end{cases}$

s.t.

$$\min \sum_{s \in S} x_s \qquad (1)$$

$$\sum_{s: i \in s} x_s = 1 \qquad \forall i \in V \qquad (2)$$

$$x_s \in \{0,1\} \qquad \forall s \in S \qquad (3)$$

Constraints (2) can be replaced by: $\sum_{s: i \in S} x_s \geq 1 \qquad \forall i \in V \qquad (2')$

(less variables, dual stabilization)

Set Covering Formulation for the VCP (ILP Model (2))

$$\min \sum_{s \in S} x_s \qquad (1)$$

s.t.

$$\sum_{s: i \in S} x_s \geq 1 \qquad \forall i \in V \qquad (2')$$

$$x_s \in \{0,1\} \qquad \forall s \in S \qquad (3)$$

- ***S*** can be defined as the family of all ***maximal Stable Sets*** in *G*.
- The continuous relaxation of this formulation leads to ***tight lower bounds*** and avoids ***symmetry*** in the solution.
- The number of maximal stable sets (i.e. the number of "columns") is ***exponential*** w.r.t. the number of vertices ***n ->*** we need column generation***.***

Set Covering Formulation for the VCP: column generation

$$\min \sum_{s \in S} x_s \qquad P$$

$$\sum_{s: i \in S} x_S \geq 1 \qquad \forall i \in V$$

$$x_s \geq 0 \qquad \forall s \in S$$

exponentially many variables $s \in S$

$$\max \sum_{i \in V} \pi_i \qquad D$$

$$\sum_{i \in s} \pi_i \leq 1 \qquad \forall s \in S$$

$$\pi_i \geq 0 \qquad \forall i \in V$$

exponentially many constraints $s \in S$

We solve *P* with a subset *S'* of the variables. Given an optimal solution *(x*,π*)* of the *restricted master problem,* does it exist a set $s^* \in S$ s.t:

$$\sum_{i \in s^*} \pi_i^* > 1 \qquad ?$$

Set Covering Formulation for the VCP: column generation

To find a violated dual constraint (i.e., a variable to be added to the master), we need to solve the following *Weighted Stable Set Problem* (NP-Hard):

$$\max \sum_{i \in V} \pi_i^* z_i \qquad (4)$$

$$z_i + z_j \leq 1 \qquad (i,j) \in E \qquad (5)$$

$$z_i \in \{0,1\} \qquad i \in V \qquad (6)$$

Where π^*_i are the optimal values of the dual variables (i.e., are constants in (4)-(6)).

Branch and Price

- Column generation methods: useful when the number of variables of a ***Linear Program*** is exponential, e.g.

$$x_s \geq 0 \qquad \forall s \in S$$

- Branch and Bound: useful to obtain an *Integer Solution.*
- Branch and Price: at each node of the Branch and Bound tree, apply column generation techniques so as to generate only the variables needed.

  Potential Problem → branching may change the sub-problem!

Branch and Price

Variables: (stable) sets of vertices.

Binary branching: choose a fractional variable x_s and then:

1) either $x_s=1$ all vertices $i \in s$ receive the same color

2) or $x_s=0$ not all vertices $i \in s$ receive the same color

→unbalanced

→sub-problem is not a VCP anymore!

Branch and Price

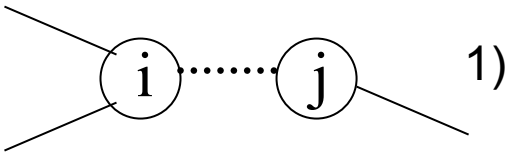
Branching rule (Zykov, 1949)

Basic idea: at each node of the search tree we select two vertices *i* and *j* such that $(i,j) \notin E$. Then,

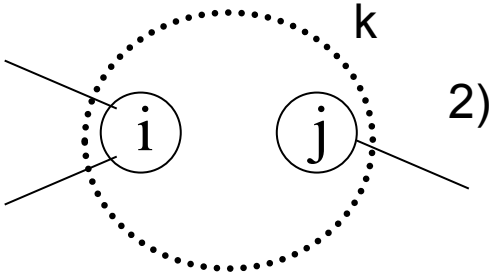
1) either add an edge between *i* and *j*, i.e. set $E = E \cup (i,j)$,

2) or collapse *i* and *j* in a single vertex *k* such that if $(i,h) \in E$ or $(j,h) \in E$, then $(k,h) \in E$.

In both cases 1) and 2) the resulting problem is still a VPC, with 1) one more edge, or 2) one less vertex.



1)



2)

Branch and Price Algorithms

Mehrotra and Trick, 1996

- Column generation: the *WSSP* is solved through a heuristic algorithm. When this fails in finding a stable set with negative reduced cost, an exact algorithm is used.
- Branching: the input graph is modified according to the Zykov scheme.

Gualandi and Malucelli, 2012

- Column generation: the *WSSP* is solved as a decision problem: find a stable set with negative reduced cost smaller than a given threshold, by using Constraint Programming techniques. When this fails, a ILP solver is used.
- Branching: the input graph is modified according to the Zykov scheme.

Branch and Price Algorithms

M., Monaci and Toth, 2011

Column generation:

1) detect possible negative reduced cost columns by considering the columns of a pool containing the independent sets found by a meta-heuristic algorithm for *VCP* (MMT, 2008);

2) if no column is found, the *WSSP* is solved through a meta-heuristic algorithm (tabu search);

3) when this fails in finding a stable set with negative reduced cost, a ILP solver is used.

- Branching:

  1) the input graph is modified according to the Zykov scheme;

  2) binary branching on the most fractional variable.

Branch and Price Algorithms

Held, Cook, Sewell (2012)

The master problem is solved in exact arithmetic.

- Branching:

  1) the input graph is modified according to the Zykov scheme;

Column generation:

- The slave problem is solved by an exact Branch-and-Bound method for the MWSSP.

They confirmed the correctness of previously published results.

DIMACS Benchmark Instances
Johnson and Trick, 2nd 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.

- To allow comparisons on results obtained with different machines, a *benchmark program* is available. CPU times can be scaled w.r.t. the performance obtained on this program.

Computational Results for the Exact Approaches

- *Algorithm DSATUR*: Brèlaz (Comm. ACM, 1979),
  Improved by Sewell (2nd DIMACS Implem. Challenge, 1993)
  Implemented by Mehrotra and Trick (INFORMS J. on C., 1996)

- *Branch and Cut Algorithm BC-Col (with the stronger lower bounding procedures)*:
  Méndez-Díaz and Zabala (Disc. Appl. Math. 2006, 2008)

- *Branch-and-Price Algorithm*: Gualandi and Malucelli (2012)

- *Branch-and-Price Algorithm*: M., Monaci, Toth. (2011)

			Mendéz-Díaz, Zabala (2006, 2008)						G-M (2009)			M-M-T (2009)		
					Sun ULTRA 1 140 MHz				Intel Dual 1.8 GHz			Pentium IV 2.4 GHz		
				BC-Col			DSATUR							
	n	opt	LB	UB	time	LB	UB	time	LB	UB	time	LB	UB	time
DSJC125.1	125	5	5	5	1	5	5	1	5	6	136	5	5	142
DSJC125.5	125	17	13	20	TL	9	19	TL	16	19	1274	17	17	18051
DSJC125.9	125	44	42	47	TL	29	45	TL	44	44	?	44	44	3897
DSJC250.1	250	?	5	9	TL	4	9	TL	6	9	TL	6	8	TL
DSJC250.5	250	?	14	36	TL	9	35	TL	26	36	TL	20	28	TL
DSJC250.9	250	?	48	88	TL	34	87	TL	71	73	178	71	72	TL
DSJC500.9	500	?	59	161	TL	43	160	TL	123	?	258	123	127	TL
le450_5d	450	5	5	10	TL	5	8	TL	5	6	TL	5	5	1
le450_15c	450	15	15	24	TL	13	23	TL	15	22	TL	15	15	TL
le450_15d	450	15	15	23	TL	13	23	TL	15	23	TL	15	15	TL
queen8_8	64	9	9	9	3	9	9	18	9	9	25	9	9	4
queen9_9	81	10	9	11	TL	9	10	TL	9	10	33	10	10	37
queen10.10	100	11	10	12	TL	10	12	TL	10	11	463	11	11	687
queen11.11	121	11	11	14	TL	11	13	TL				11	12	TL
myciel6	95	7	5	7	TL	2	7	TL	4	?	1	4	7	TL
myciel7	191	8	5	8	TL	3	8	TL	5	?	8	5	8	TL
will199GPIA	701	7				7	7	1	7	7	448	7	7	81
DSJR500.1	500	12	12	12	4	12	12	3				12	12	35
DSJR500.1c	500	85	80	88	TL	70	88	TL				85	85	289
DSJR500.5	500	122	119	130	TL	103	130	TL				122	122	342
wap01	2368	?	41	46	TL	39	48	TL				40	43	TL
wap05	905	50	50	51	TL	27	51	TL				50	50	293
wap06	947	40	40	44	TL	33	45	TL				40	40	175

Heuristic and Metaheuristic Approaches

- Greedy Algorithms:

  - Sequential Coloring (SEQ)
  - DSATUR: Brélaz (Comm. ACM 1979)
  - Recursive Largest Fit: (RLF) Leighton (J. Res. NBS 1979)
  - Backtracking: Bollobas, Thomason (Ann. Disc. Math. 1985)
  - Iterated Greedy: Culberson, Luo (2nd DIMACS Implem. Challenge 1993)

- Simulated Annealing Algorithm:

  Johnson, Aragon, Mc Geoch, Schevon (Oper. Res. 1991)
  Morgenstern (2nd DIMACS Impl. Challenge 1993)

- Tabu Search Algorithms:

  - TABUCOL: Hertz, de Werra (Computing 1987)

    Improvements:

    - Dorne, Hao (Metaheuristics:…, Kluwer 1998)
    - Blöchliger, Zufferey (Computers & O.R. 2008)

Heuristic and Metaheuristic Approaches

- Evolutionary Algorithms:
  - Fleurent, Ferland (2nd DIMACS Impl. Challenge 1993)
  - Davis (Handbook of Genetic Algs., Van Nostrand Reinhold 1998)
  - Galinier, Hao (J. Comb. Opt. 1999)

- MIPS-CLR Algorithm: Funabiki, Higashino (IEICE T.F. 2000)
- Variable Neighborhood Search Alg: Avanthay, Hertz, Zufferey (EJOR 2003)
- Adaptive Memory Alg: Galinier, Hertz, Zufferey (D.A.M. 2008)
- Two-Phase Alg: M., Monaci, Toth (INFORMS J. on C. 2008)
- Variable Search Space Alg: Hertz, Plumettaz, Zufferey (D.A.M. 2008)

Local Search Methods

- Definition of Solution: ***s***
- Definition of a Solution evaluating function: ***f(s)***
- Definition of a neighborhood: ***N(s)***

Local Search (MIN problem):

1) generate an initial solution s^*

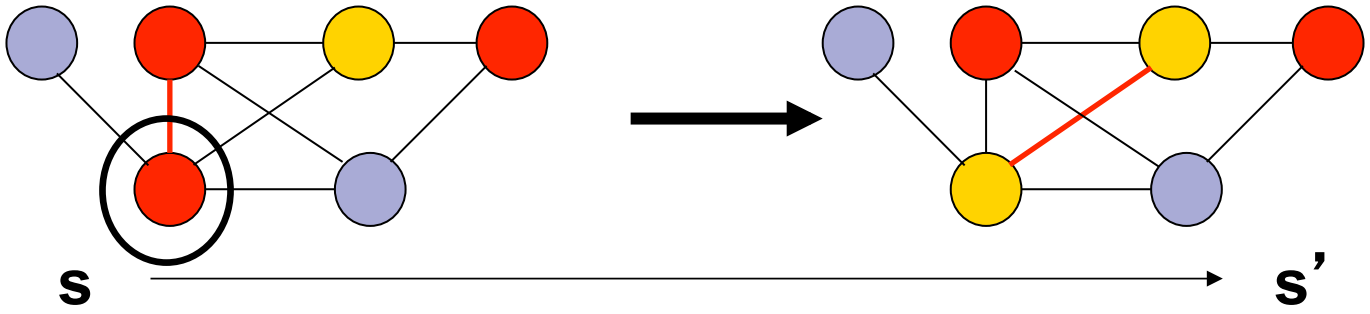
2) $s' = argmin_{s \in N(s^*)} f(s)$

3) if $f(s') < f(s^*)$ then $s^* = s'$, goto 2. Else stop.

Tabu Search: local search which always moves on the best solution $s' \in N(s)$, even if $f(s') \geq f(s^*)$. A Tabu List is used to avoid cycling.

Tabucol: a Tabu Search approach for the VCP (Hertz and de Werra ‘87)

- A target ***k*** is required for the number of colors to be used. **A solution *s* is a partition of *V* in *k* color classes (with possible conflicts)**. Resolving all the conflicts gives a feasible ***k-coloring***.
- The **solution evaluating function *f(s)*** is represented by the number of conflicts.
- A **move** consists in choosing one **conflicting vertex** and moving it to a new color class which minimizes the number of conflicts.
- To avoid cycling, a vertex can not enter the color class it left during the last ***T*** iterations (Tabu rule).

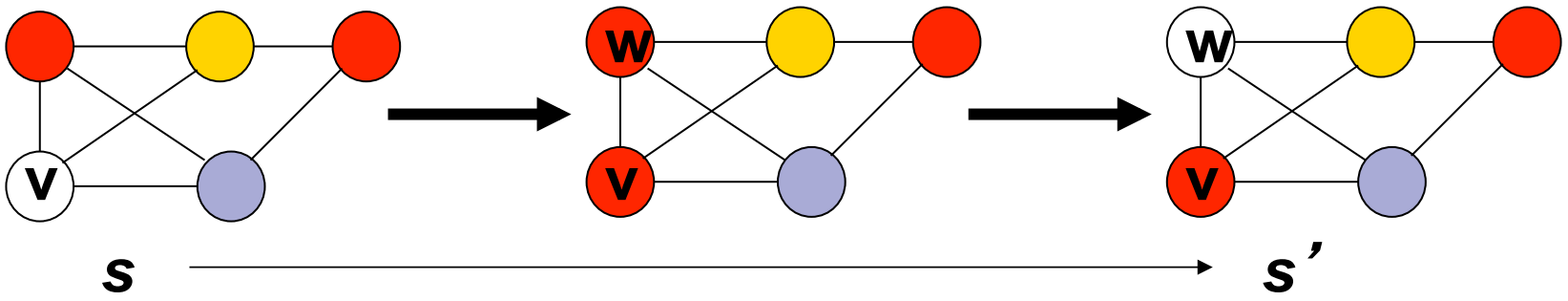


Tabu Search (M., Monaci and Toth, 2008)

- ***Impasse Class Neighborhood*** (Morgenstern 1996): a target ***k*** is required for the number of colors to be used. A solution ***s*** is a partition of *V* in *k+1* color classes in which all classes except possibly the last one are **stable sets**. Making this last class empty gives a feasible ***k-coloring***.

To move from a solution ***s*** to a neighbor solution ***s'***:

- randomly choose an uncolored vertex ***v*** (in class *k+1*)
- assign ***v*** to a color class
- move all vertices ***w*** in this class, which are adjacent to ***v***, to the class ***k+1***



Tabu Search approach MMT

- Objective function: $\min f(s) = \sum_{v \in K+1} \delta(v)$
- Tabu Search:

  - move from ***s*** to the best solution ***s'*** in the neighborhood (even if *f(s)<f(s')*)

  - Tabu move: to avoid cycling, a vertex can not take the same color it took in the last ***T*** (tabu tenure) iterations

- Complexity of the Tabu Search:

  time complexity of one iteration of the Tabu Search is ***O(n)*** in the worst case