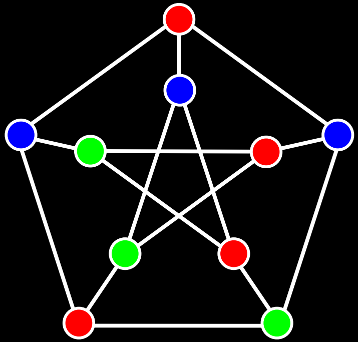
Coloring



Instance

Undirected unweigthed graph $G = \langle V, E \rangle$

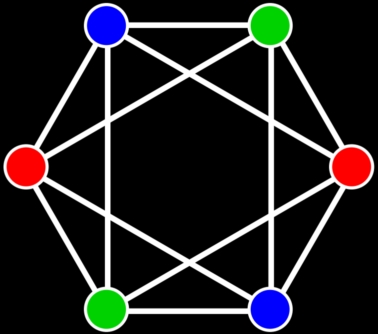
Feasible solution

Vertex labeling $\lambda : V \mapsto C$ such that $(v, w) \in E \Rightarrow \lambda(v) \neq \lambda(w)$

Objective function

$\min |C|$

Clique and Coloring



Integer Linear Programming

$$
\begin{aligned}
\min \sum u_a &\quad \text{subject to} && (1)\\
c_{v,a} + c_{w,a} \le u_a &\quad \forall (v,w) \in E, \forall a \in C && (2)\\
\sum_{a \in C} c_{v,a} \ge 1 &\quad \forall v \in V && (3)\\
u_a \in \{0,1\} &\quad \forall \text{ color } a \in C\\
u_a = 1 &\quad \text{ if color } a \text{ is used}\\
c_{v,a} \in \{0,1\} &\quad \forall (v,w) \in E\\
c_{v,a} = 1 &\quad \text{ if } v \text{ has color } a
\end{aligned}
$$

Simplification

Adjacent to all vertices

Let v be a vertex such that $N(v) = V \setminus \{v\}$. Color $G - v$ and assign a new color to v.

Included neighborhood

Let v, w be two nonadjacent vertices such that $N(v) \subseteq N(w)$. Color $G - v$ and assign to v the same color as w.

Separation

Articulation point

Let v be an articulation point, whose removal results in the connected components $C_1, \ldots, C_k$. Color each $G|(C_i \cup \{v\})$, fixing the color of v at the beginning.

Separating clique

Let Q be a clique, whose removal results in the connected components $C_1, \ldots, C_k$. Color each $G|(C_i \cup Q)$, fixing the colors of Q at the beginning.

Greedy

Data: graph $G = \langle V, E \rangle$
foreach *vertex* $v \in V$ **do**
  $color[v] \leftarrow$ the smallest color not used by a neighbor of v

Welsh–Powell

Data: graph $G = \langle V, E \rangle$
foreach *vertex* $v \in V$ *in decreasing order of degree* **do**
  $color[v] \leftarrow$ the smallest color not used by a neighbor of v

Iterated Greedy

1. Reorder color classes:
   - Largest class first
   - Reverse the classes of the current solution
   - Random rearrangement of the color classes
2. Apply Greedy

Main properties

1. Always a feasible solution
2. There exists a vertex ordering on which greedy is optimal

DSatur

Data: graph $G = \langle V, E \rangle$
while *there exists an uncolored vertex* **do**
  foreach *vertex* v **do**
    $saturation[v] \leftarrow$ the number of colors of neighbors of v
  $v \leftarrow$ an uncolored vertex with maximum saturation, ties are broken by taking the vertex with most uncolored neighbors;
  $color[v] \leftarrow$ the smallest color not used by a neighbor of v

Recursive Largest FIrst

Data: graph $G = \langle V, E \rangle$
while *there exists an uncolored vertex* **do**
    $v \leftarrow$ a uncolored vertex with maximum degree;
    $I \leftarrow \{v\}$;
    while *I is not a maximal independent set* **do**
        Add to I a vertex v that (1) is not adjacent to any vertex of I, (2) is adjacent to the maximum number of neighbors of I, and (3) has minimum degree
    Give a new color to all vertices in I;
    Remove from G all vertices in I

Simulated Annealing

Moves

Change color of a vertex

Objective function

$\min |(v, w) \in E : \lambda(v) = \lambda(w)|$

Main properties

1. Maintains a complete, unfeasible solution with k colors
2. Increase or decrease k according to the results

Simulated Annealing 2

Moves

Choose the color c of an uncolored vertex v, then uncolor all neighbors of v that are color c

Objective function

$\min$ the number of uncolored vertices

Main properties

1. Maintains an incomplete, partially feasible solution with k colors
2. Increase or decrease k according to the results
3. An incomplete solution can be completed

TabuCol

Moves

Pick a vertex v that has a neighbor with the same color, then change color of v from b to c

Tabu list

After the change of color, it is forbidden to move the color of v back to b for some iterations.

Main properties

1. Maintains an incomplete, partially feasible solution with k colors
2. Increase or decrease k according to the results
3. An incomplete solution can be completed.

Ant Colony

Data: graph $G = \langle V, E \rangle$

$t_{u,v} \leftarrow 1 \; \forall u \neq v \in V$, $k \leftarrow n$, $B \leftarrow$ single vertices /* $t_{u,v}$: trail matrix */

while *not stopping condition* **do**

    $\delta_{u,v} \leftarrow 0 \; \forall u \neq v \in V$ /* $\delta_{u,v}$ is the update matrix */

    foreach *ant a* **do**

        $S \leftarrow \text{BuildSolution}(k)$/* only k colors allowed */

        if *S is a partial solution* **then**

            $f_{v,i} \leftarrow \sum_u f_{u,v}/|S_i|$;

            complete the solution S with probability $Pr[v, i] = f^{\alpha}_{v,i} / \sum_u f^{\alpha}_{u,i}$ /* α: parameter */

            $\delta_{u,v} \leftarrow \delta_{u,v} + F(S) \; \forall u \neq v, c(u) = c(v) \in V$;

            if *S is feasible and better than B* **then**

                $B \leftarrow S$, $k \leftarrow |B| - 1$

        $t_{u,v} \leftarrow r t_{u,v} + \delta_{u,v} \; \forall u \neq v \in V$ /* r is the evaporating factor */

License