DM841

Discrete Optimization

Lecture 13

Examples

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Preprocessing rules

Polynomial time reduction of the graph G to G' such that given a feasible k-coloring for G' it is striaghtforward to derive a feasible k-coloring for G.

Searching for a k-coloring (k fixed)

- Remove under-constrained nodes: $v \in V, d(v) < k$
- Remove subsumed nodes: $v \in V$, if $\exists u \in V, uv \notin E, A(v) \subseteq A(u)$
- Merge nodes that must have the same color: eg, if any nodes are fully connected to a clique of size $k-1$, then these nodes can be merged into a single node with all the constraints of its constituents, because they must have the same color.

Outline

Construction Heuristics

- sequential heuristics
  1. choose a variable (vertex)
     a) static order: random (ROS),
        largest degree first, smallest degree last
     b) dynamic order: saturation degree (DSATUR) [Brélaz, 1979]
  2. choose a value (color): greedy heuristic

```
Procedure ROS
RandomPermutation π(Vertices);
forall the i in 1, ..., n do
    v := π(i);
    select min{c : c not in saturated[v]};
    col[v] := c;
    add c in saturated[w] for all w adjacent v;
```

$\mathcal{O}(nk + m) \rightsquigarrow \mathcal{O}(n^2)$

```
Procedure DSATUR
select vertex v uncolored with max degree;
while uncolored vertices do
    select min{c : c not in saturated[v]};
    col[v] := c;
    add c in saturated[w] for all w adjacent v;
    select uncolored v with max size of
      saturated[v];
```

$\mathcal{O}(n(n + k) + m) \rightsquigarrow \mathcal{O}(n^2)$

- partitioning heuristics
  - recursive largest first (RLF) [Leighton, 1979]
    iteratively extract stable sets

RLF [Leighton, 1979]

Procedure Recursive Largest First(G)
In $G = (V, E)$: input graph;
Out k : upper bound on $\chi(G)$;
Out c : a coloring $c : V \mapsto K$ of G;

$k \leftarrow 0$
while $|V| > 0$ **do**
  $k \leftarrow k + 1$ /* Use an additional color */
  FindStableSet(V, E, k) /* $G = (V, E)$ is reduced */
return k

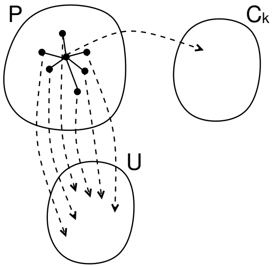
RLF

Key idea: extract stable sets trying to maximize edges removed.

Procedure FindStableSet(G, k)
In $G = (V, E)$: input graph
In k : color for current stable set
Var P : set of potential vertices for stable set
Var U : set of vertices that cannot go in current stable set

$P \leftarrow V$; $U \leftarrow \emptyset$;
forall the $v \in P$ **do** $d_U(v) \leftarrow 0$; /* degree induced by U */
while P not empty **do**
  select v in P with max d_U;
  move v from P to C_k; $V \leftarrow V \setminus \{v\}$
  forall the $w \in \delta_P(v)$ **do** /* neighbors of v in P */
    move w from P to U; $E \leftarrow E \setminus \{v, w\}$
    forall the $u \in \delta_P(w)$ **do**
      $d_U(u) \leftarrow d_U(u) + 1$

$\mathcal{O}(m + n\Delta^2) \rightsquigarrow \mathcal{O}(n^3)$



Outline

Local Search for Graph coloring

Different choices for the candidate solutions:

decision vs optimization	assignment of colors to V	level of feasibility	Performance
k-fixed	complete	proper	
k-fixed	partial	proper	$+++$
k-fixed	complete	improper	$+++$
k-fixed	partial	improper	$-$
k-variable	complete	proper	$++$
k-variable	partial	proper	$-$
k-variable	complete	improper	$++$
k-variable	partial	improper	$-$

imply different neighborhood structures and evaluation functions.

Local Search for GCP

Scheme: k-fixed / complete / improper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Evaluation function: conflicting edges or conflicting vertices
- Neighborhood: one-exchange

Naive approach: $O(n^2 k)$
Neighborhood examination

for all $v \in V$ **do**
    for all $k \in 1..k$ **do**
        compute $\Delta(v, k)$

Better approach:

- V^c set of vertices involved in a conflict
- $\mathbf{\Delta}(v,k)$ stores number of vertices adjacent to v in each color class k

Procedure Initialise_$\mathbf{\Delta}$(G,a)
$\mathbf{\Delta} = 0$
for each v in V **do**
  for each u in $A_V(v)$ **do**
    $\mathbf{\Delta}(u, \mathrm{a}(v)) = \mathbf{\Delta}(u, \mathrm{a}(v)) + 1$

Procedure Examine(G,N(a))
for each v in V^c **do**
  for each $k \in \Gamma$ **do**
    compute $\Delta(v,k) = \mathbf{\Delta}(v,k) - \mathbf{\Delta}(v, \mathrm{a}(v))$

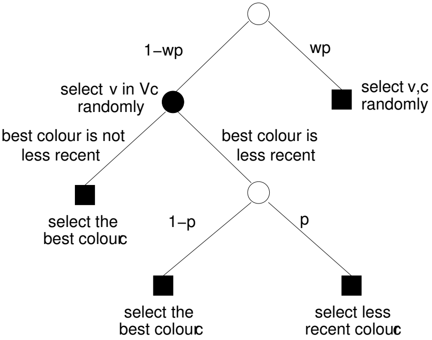
Procedure Update_$\mathbf{\Delta}$(G,a,v,k)
for each u in $A_V(v)$ **do**
  $\mathbf{\Delta}(u, \mathrm{a}(v)) = \mathbf{\Delta}(u, \mathrm{a}(v)) - 1$
  $\mathbf{\Delta}(u,k) = \mathbf{\Delta}(u,k) + 1$

Comet examples

Tabu Search

`./coloring.co`

Randomized Iterative Improvement



Guided Local Search

- evaluation function: $f'(s) = f(s) + \lambda \cdot \sum_{i=1}^{|E|} w_i \cdot I_i(\mathcal{C})$
  w_i is the penalty cost associated to edge i;
  $I_i(s)$ is an indicator function that takes the value 1 if edge i causes a colour conflict in s and 0 otherwise;
  parameter λ
- penalty weights are initialised to 0
- updated each time Iterative Improvement reaches a local optimum in f'; increment the penalties of all edges with maximal utility.

$$u_i = I_i(s) \cdot \frac{1}{1 + w_i}.$$

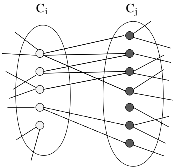
- once a local optimum is reached, the search continues for sw non-worsening exchanges (side walk moves) before the evaluation function f' is updated. Update of w_i and f' is done in the worst case in $O(k|V|^2)$.

Local Search for GCP

Scheme: k-variable / complete / proper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Neighborhood: one-exchange restricted to feasible moves
  Kempe chains



- Evaluation function: $f(s) = -\sum_{i=1}^{k} |C_i|^2$
  favours few large color classes wrt. many small color classes

Local Search for GCP
Iterated Greedy

Scheme: k-variable / complete / proper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Neighborhood: permutation of color classes + greedy algorithm
- Evaluation function: number of colors

Theorem

Let φ be a k-coloring of a graph G and π a permutation such that if $\varphi(v_{\pi(i)}) = \varphi(v_{\pi(m)}) = c$ then $\varphi(v_{\pi(j)}) = c, \forall i \leq j \leq m$.
Applying the greedy algorithm to π will produce a coloring using k or fewer colors.

Local Search for GCP

Scheme: k-variable / complete / improper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Neighborhood: one-exchange
- Evaluation function: $f(s) = -\sum_{i=1}^{k} |C_i|^2 + \sum_{i=1}^{k} 2|C_i||E_i|$

Ev. function chosen in such a way that an improvement in feasibility
(in the worst case by coloring a vertex to a new color class)
offsets any improvement in solution quality
(in the best case by moving a vertex to the first color class).

Local Search for GCP

[Blöchliger and N. Zufferey, 2008]

Scheme: k-fixed / partial / proper

Local Search

- Solution representation: collection of $k+1$ sets + assignment vector
- Evaluation function: size of impasse class (weighted by degree)
- Neighborhood: i-swap

References

Brélaz D. (1979). **New methods to color the vertices of a graph**. *Communications of the ACM*, 22(4), pp. 251–256.

Leighton F.T. (1979). **A graph coloring algorithm for large scheduling problems**. *Journal of Research of the National Bureau of Standards*, 84(6), pp. 489–506.