Different Graph Coloring Methods theoretical and experimental approach

Bogdán Zaválnij

University of Pécs
Institute of Mathematics and Informatics

Ljubljana, 2014

Table of Contents

Table of Contents

Discrete Optimization

- Many problems that need huge computational power (SAT, ILP, 0-1 LP, clique, etc.)
- Some of them even NP-hard (and NP-complete) problems
- Usage of supercomputers – may be several days or weeks running time on thousands of cores $\rightarrow$ if even feasible
- Usually a Branch-and-Bound algorithm

Research in graph optimization algorithms. Some of them polynomial, and needed for huge graph instances (shortest paths), some them NP-hard (maximum clique).

The Maximum Clique Problem

Definitions:

- Let G be a finite, simple graph: $G = (V, E)$, and C be a subgraph, which has the nodes as subset of V, and C is induced by these nodes.
- We call a graph a clique, if all of its nodes are connected to each other: $\forall v_i, \forall v_j \in V, i \neq j : (v_i, v_j) \in E$
- We call the clique size the number of the nodes in the clique.
- we call the clique size of the graph the size of its biggest clique (maximum clique), and denote it by $\omega(G)$.

The Problem:

- For many application we search for the size of a maximum clique.
- it is a well known NP-hard problem
- Obviously we can examine all the subgraphs: a problem of size $2^{|V|}$ – it would be unrealistic to do so.

Common Algorithms

Usually we can use some back-tracking algorithms:
Bron & Kerbosch (1973), Carraghan & Pardalos (1990).

The Carraghan & Pardalos algorithm is a classical Branch-and-Bound technique. We take the nodes of the graph each by one, and reduce the graph to their neighborhood.
If the reduced graph is *"not satisfactory"* we go back, if it is *"satisfactory"* we do the same (go forward).

- branching: we try several different nodes, if they should be in a maximum clique
- bound: we try to prune the branches of the search tree (number of nodes, coloring)

Auxiliary algorithm: coloring

Satisfactory in the original work was measured by the size of the subgraph ($|V'|$ – is it big enough to make a bigger clique with those we already included as the already found one?). Lately the pruning condition was suggested to be changed: we can use coloring of the subgraph (edge free coloring) instead of the size of the subgraph.

$$\omega(G) \leq \chi(G)$$

The combinatorial meaning of witch is that we cannot take more than one node from a color class to be a part of a clique.

Notes:

1. coloring is also used in branch factor modification
2. obviously $|V'|$ can be replaced with other, non combinatorial measurement – LP or SDP techniques are used, for example the Lovász' theta function

Coloring

- Finding the chromatic number is an NP-hard problem, so we use greedy colorings instead
- We can use it in the very beginning, comparing the number of colors to a greedily found solution
  - sometimes we can stop, as we could prove, that the solution reached the optimum
  - sometimes the solution is nearly optimal, and we do not need a better solution (for real industrial problems)
  - it can be used as a branching rule
- We can use the coloring for pruning the search tree,
  - if the remaining nodes' colors not enough to construct a big enough clique
  - $\Longrightarrow$ optimality test

Improving the optimality test with better coloring

- With better coloring we get a better optimality result and pruning
- But the optimality gap can be big
  - the coloring cannot surpass the chromatic number
  - the chromatic number can be arbitrary far from the clique number (Erdős's theorem, Mycielski graphs)
- Can we use another optimality test instead of the usual coloring?

Table of Contents

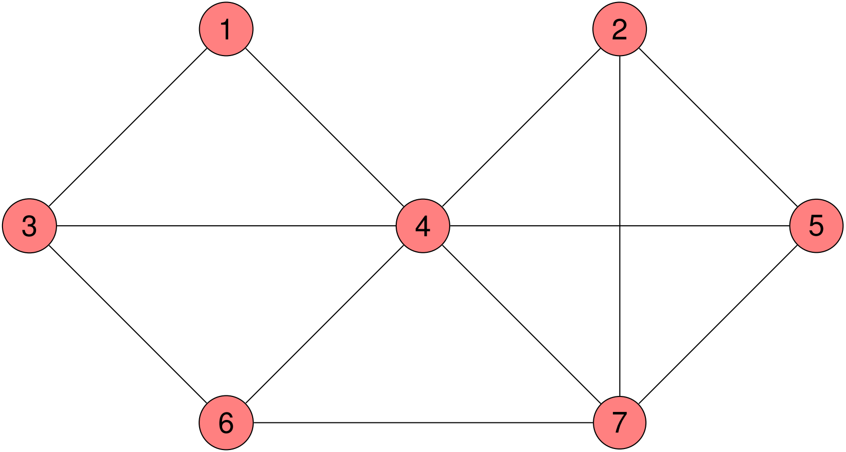
Edge Colorings

Note, that we are not interested in the coloring itself, we use it only for pruning purposes. So can we modify coloring, to get better pruning? We can ease some condition of the coloring definition, or we can change the subject of the coloring – what we will color.

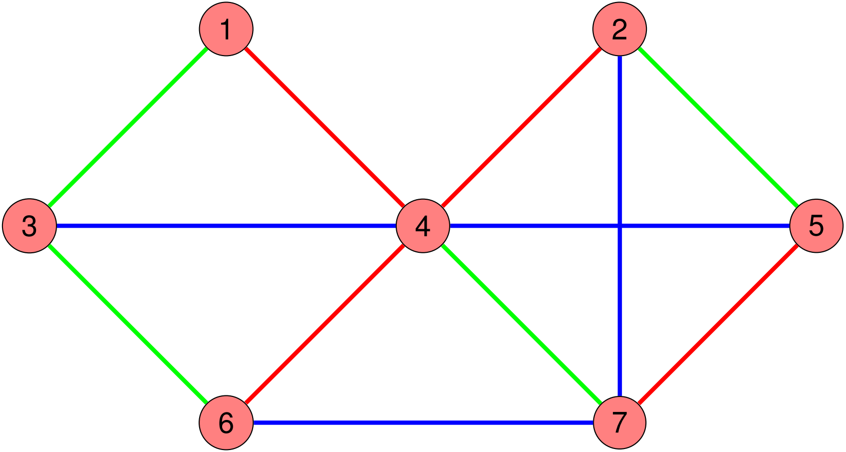
Edge Coloring Schemes (color edges instead of nodes):

1. intersecting edges should get different colors (Vizing)
2. edges constructing a triangle should get different colors
3. edges constructing a square (K_4-s) should get different colors
4. edges constructing a triangle or a square should get different colors

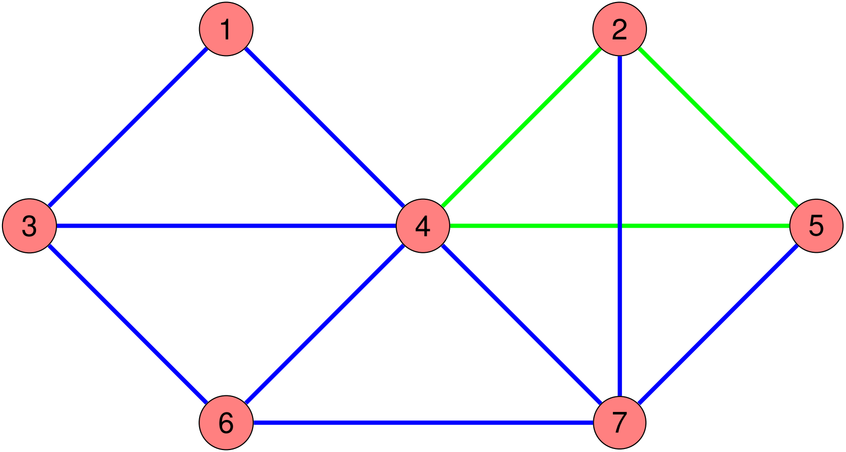
Edge Colorings



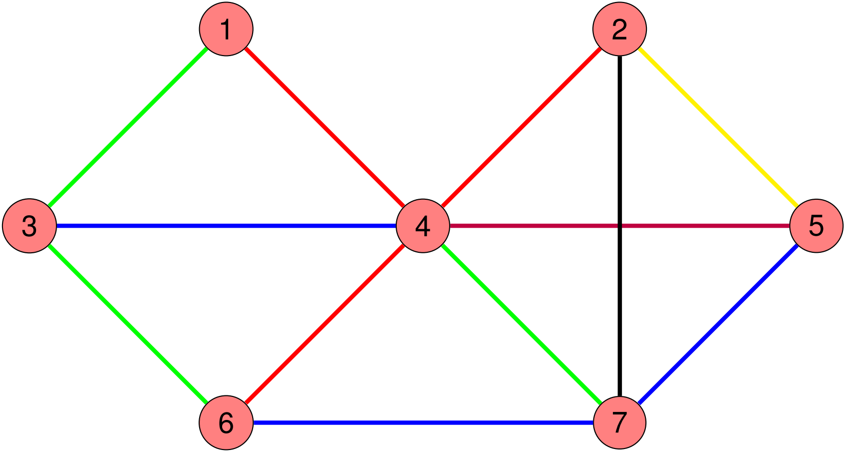
Edge Colorings – 2 (no triangles)



Edge Colorings – 3 (no squares (K_4))



Edge Colorings – 4 (no tr&sq)



Edge Colorings

Calculating the edge colors needed for a complete graph we can have an upper bound for a clique size of an arbitrary graph from a coloring – monotonic property.

The values of $\chi_2(K_n)$ for $3 \leq n \leq 8$.

n	3	4	5	6	7	8
$\chi_2(K_n)$	3	3	5	5	7	7

$\chi_3(K_n) = n - 2$

$\chi_4(K_n) = n(n-1)/2$, so if the edges of G have an 4th-type coloring with k colors and $\omega(G) = t$, then $t(t-1)/2 \leq k$.
(Our measurements showed that the 4th method can give better estimates.)

Edge Colorings

Calculating the edge colors needed for a complete graph we can have an upper bound for a clique size of an arbitrary graph from a coloring – monotonic property.

The values of $\chi_2(K_n)$ for $3 \leq n \leq 8$.

n	3	4	5	6	7	8
$\chi_2(K_n)$	3	3	5	5	7	7

$\chi_3(K_n) = n - 2$

$\chi_4(K_n) = n(n-1)/2$, so if the edges of G have an 4th-type coloring with k colors and $\omega(G) = t$, then $t(t-1)/2 \leq k$.
(Our measurements showed that the 4th method can give better estimates.)

s-free Coloring

Natural modification of the edge free coloring is the s-clique free coloring. We call a node partitioning s-free coloring, if the partitions do not have cliques of size s.

As partitions can have at most cliques of size $(s-1)$, so we can choose at most $(s-1)$ nodes from one partition in the final clique, so:

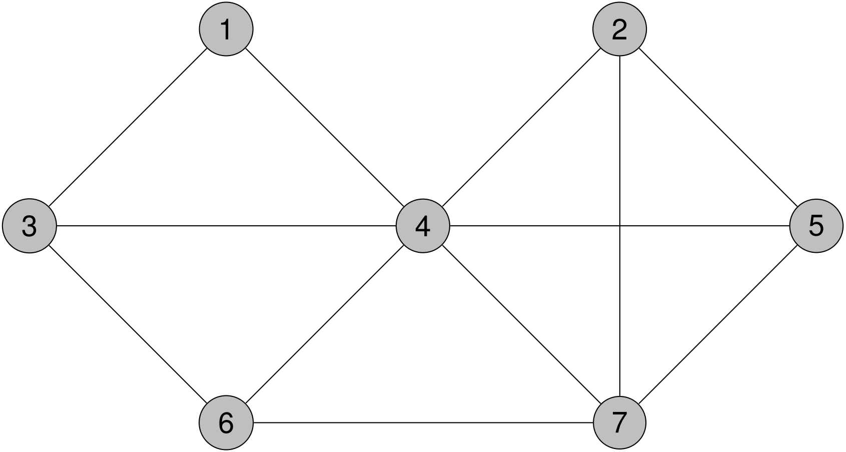
$$\omega(G) \leq \chi_{s-free}(G)(s-1)$$

Or for different s_i values: $\sum(s_i - 1)$

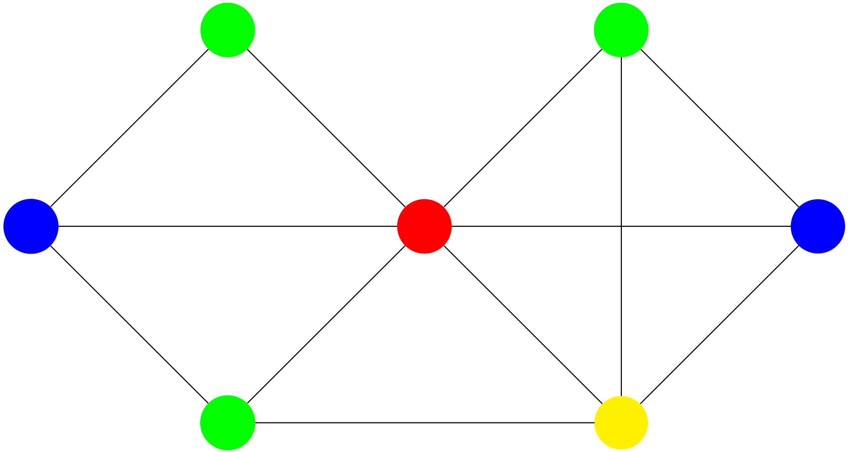
Note, that $s = 2$ is the original edge free coloring, so it is a natural evolvation of the original coloring scheme.

The s-clique free optimality results can go below the chromatic number. Example: the nodes of an odd circle. $\omega(G) = 2, \chi(G) = 3$, while 3-free coloring gives us a sharp bound as $\chi_{3-free}(G) = 1$

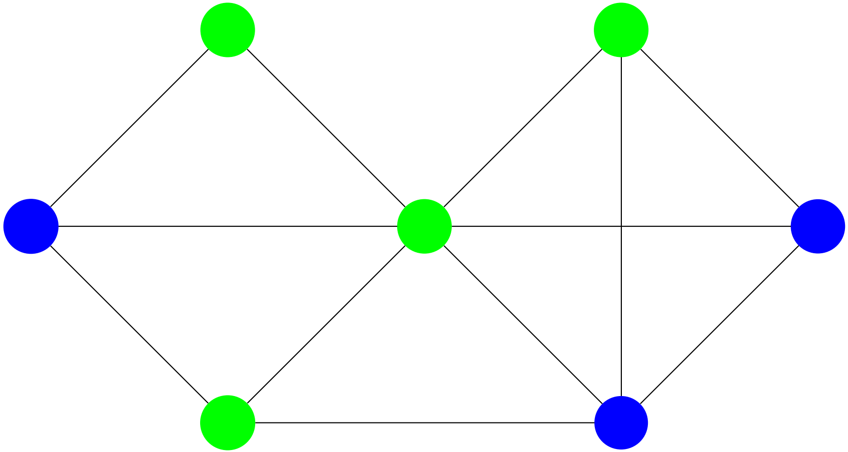
s-free Colorings



s-free Colorings – 2-free (legal coloring)



s-free Colorings – 3-free



s-free Colorings – 4-free

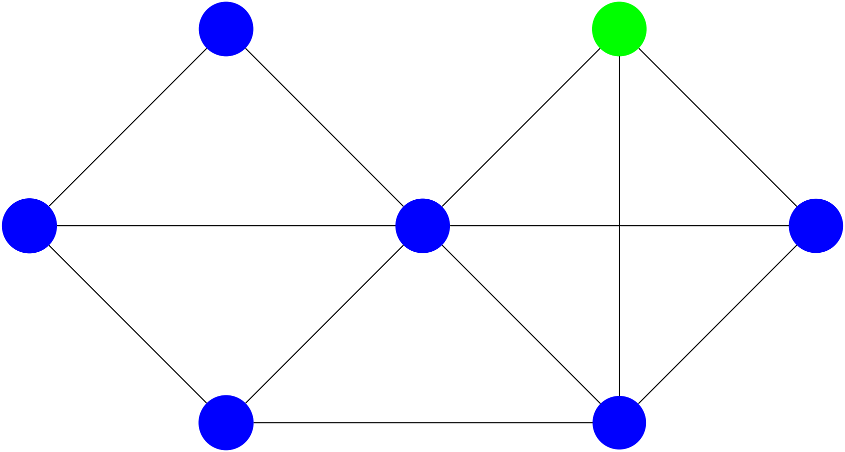


Table of Contents

Daniel Brélaz's DSatur (Degree SATuration) coloring

- Greedy algorithm
- Idea: we choose the least suitable node to color
  - we should calculate the fitting of the uncolored nodes into the color classes

1. Optimal node
   - minimal freedom (less suitable color classes)
   - maximum saturation (debated)
2. Color the node
   - Put into the first free color class, or
   - Open a new class (if no free)
3. Update information of remaining nodes
   - If the just colored node have a neighbour, than
   - It cannot be placed in that color class
   - The freedom of it will decreese by one

Daniel Brélaz's DSatur (Degree SATuration) coloring

- Greedy algorithm
- Idea: we choose the least suitable node to color
  - we should calculate the fitting of the uncolored nodes into the color classes

1. Optimal node
   - minimal freedom (less suitable color classes)
   - maximum saturation (debated)
2. Color the node
   - Put into the first free color class, or
   - Open a new class (if no free)
3. Update information of remaining nodes
   - If the just colored node have a neighbour, than
   - It cannot be placed in that color class
   - The freedom of it will decreese by one

Daniel Brélaz's DSatur (Degree SATuration) coloring

- Greedy algorithm
- Idea: we choose the least suitable node to color
  - we should calculate the fitting of the uncolored nodes into the color classes

1. Optimal node
   - minimal freedom (less suitable color classes)
   - maximum saturation (debated)
2. Color the node
   - Put into the first free color class, or
   - Open a new class (if no free)
3. Update information of remaining nodes
   - If the just colored node have a neighbour, than
   - It cannot be placed in that color class
   - The freedom of it will decreese by one

Daniel Brélaz's DSatur (Degree SATuration) coloring

- Greedy algorithm
- Idea: we choose the least suitable node to color
  - we should calculate the fitting of the uncolored nodes into the color classes

1. Optimal node
   - minimal freedom (less suitable color classes)
   - maximum saturation (debated)
2. Color the node
   - Put into the first free color class, or
   - Open a new class (if no free)
3. Update information of remaining nodes
   - If the just colored node have a neighbour, than
   - It cannot be placed in that color class
   - The freedom of it will decreese by one

Daniel Brélaz's DSatur (Degree SATuration) coloring

- Greedy algorithm
- Idea: we choose the least suitable node to color
  - we should calculate the fitting of the uncolored nodes into the color classes

1. Optimal node $\rightarrow$ MINLOC Reduction
   - minimal freedom (less suitable color classes)
   - maximum saturation (debated)
2. Color the node
   - Put into the first free color class, or
   - Open a new class (if no free)
3. Update information of remaining nodes
   - If the just colored node have a neighbour, than
   - It cannot be placed in that color class
   - The freedom of it will decreese by one

Modification of the DSatur coloring

- Edge free coloring
  - derivative graph
  - a node represents an edge
  - we connect two nodes if the rule for the coloring of the edges denies these edges the same color
  - (no actual graph storage is needed!)
- *s*-clique free
  - decision if a node can be put into a color class: we use the Carraghan–Pardalos maximum clique program itself
  - (a much smaller problem)
  - we need to distinguish between color classes not only by fits or not fits
  - we must calculate and store the level of the fitting of a node into a color class (what is the clique that node construct in that color class)

Parallelization

- Hard problem even for greedy algorithm
  - edge coloring: the derivative graph is too big
  - *s*-free coloring: too much calculation if *s* is big
- Bulk Synchronous Parallel solution
- $\rightarrow$ idea behind the MapReduce, Hadoop
  - we distribute the nodes between computers
  - we choose between local nodes (Map)
  - than we globally choose between the chosen nodes and find the appropriate color class (Reduce, MINLOC)
  - we calculate the fitting level locally (Map)

Tested on edge coloring [4], where the derivative graph can have even 5M nodes.

Results

Results gave better optimum than node coloring. (Some better than Lovász' Θ!)

Best results with the last version of edge coloring. (Sequential coloring result also better, than sequential coloring of the nodes.)

$\lvert V\rvert$	number of nodes
$\lvert E\rvert$	number of edges
ω	the actual clique number
χ_N	the actual node chromatic number
χ_E	the actual edge chromatic number
$\overline{\chi}_N$	estimate of χ_N by the algorithm
$\overline{\omega}$	the estimate for ω using $\overline{\chi}_N$
$\overline{\chi}_E$	estimate of the χ_E by the algorithm
$\hat{\omega}$	the estimate for ω using $\overline{\chi}_E$

Table: The symbols used for labeling the columns

Results – edge coloring

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_E$	$\hat{\omega}$
3	27	189	5	5	10	5
4	64	1296	10	10	31	8
5	125	5500	16	16	82	13
6	216	17550	24	24	192	20
7	343	46305	35	35	400	28
8	512	106624	45	45	747	39
9	729	221616	60	60	1289	51
10	1000	425250	75	75	2073	64
11	1331	765325	91	91	3135	79
12	1728	1306800	109	109	4582	96
13	2197	2135484	128	128	6509	114
14	2744	3362086	156	156	8948	134
15	3375	5126625	178	178	11981	155

Table: Monotonic matrix, Brelaz's coloring

Results – edge coloring

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_E$	$\hat{\omega}$
3	8	9	2	2	1	2
4	16	57	4	4	6	4
5	32	305	6	6	15	6
6	64	1473	12	12	50	10
7	128	6657	22	22	189	19
8	256	28801	42	42	762	39
9	512	121089	81	81	2908	76
10	1024	499713	157	157	10568	145
11	2048	2037761	306	306	37481	274

Table: Deletion error correcting code, Brelaz's coloring

Results – edge coloring

n	$\|V\|$	$\|E\|$	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_E$	$\hat{\omega}$
6	15	45	4	4	3	3
7	35	385	9	9	24	7
8	70	1855	14	14	109	15
9	126	6615	28	28	332	26
10	210	19425	44	44	861	42
11	330	49665	64	64	1880	61
12	495	114345	92	92	3612	85
13	715	242385	126	126	6289	112
14	1001	480480	169	169	10411	144
15	1365	900900	216	216	16633	181
16	1820	1611610	277	277	24877	223
17	2380	2769130	344	344	37123	272

Table: Johnson code Brelaz's coloring, $n =$ length of word, number of 1's is equal to 4, the Hamming distance is equal to 3

Results – edge coloring

n	k	$\lvert V\rvert$	$\lvert E\rvert$	ω	χ_N	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_E$	$\hat{\omega}$
15	1	15	105	15	15	15	15	105	15
15	2	105	4095	7	13	13	13	55	11
15	3	445	50050	5	11	11	11	46	10
15	4	1365	225225	3	9	9	9	16	5
15	5	3003	378378	3	7	8	8	3	3
16	1	16	120	16	16	16	16	120	16
16	2	120	5460	8	14	14	14	62	11
16	3	560	80080	5	12	12	12	57	11
16	4	1820	450450	4	10	10	10	32	8

Table: Kneser graph, Brelaz's coloring

Results – edge coloring

k	$\lvert V\rvert$	$\lvert E\rvert$	ω	χ_N	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_E$	$\hat{\omega}$
7	95	755	2	7	7	7	1	2
8	191	2360	2	8	8	8	1	2
9	383	7271	2	9	9	9	1	2
10	767	22196	2	10	10	10	1	2
11	1535	67355	2	11	11	11	1	2
12	3071	203600	2	12	12	12	1	2
13	6143	613871	2	13	13	13	1	2
14	12287	1847756	2	14	14	14	1	2

Table: Mycielski graphs, Brelaz's coloring

Results – edge coloring

n	k	$\|V\|$	$\|E\|$	ω	χ_N	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_E$	$\hat{\omega}$
3	7	285	29340	6	21	21	21	81	13
6	7	570	139905	12	42	42	42	434	29
9	7	855	331695	18	63	63	63	1053	46
12	7	1140	604710	24	84	84	84	1931	62
3	8	573	116523	6	24	24	24	107	15
6	8	1146	561375	12	48	48	48	558	33
9	8	1719	1334556	18	72	72	72	1362	52
12	8	2292	2436066	24	96	96	96	2540	71

Table: Product of an n-complete and a k-Mycielski graph, Brelaz's coloring

Results – 3-free coloring

n	k	$\|V\|$	$\|E\|$	ω	χ_N	$\overline{\chi}_N$	$\overline{\omega}$	$\overline{\chi}_{3-free}$	$\hat{\omega}$
3	7	285	29340	6	21	21	21	5	10
6	7	570	139905	12	42	42	42	13	26
9	7	855	331695	18	63	63	63	19	38
3	8	573	116523	6	24	24	24	5	10
6	8	1146	561375	12	48	48	48	12	24
9	8	1719	1334556	18	72	72	72	18	36

Table: Product of an *n*-complete and a *k*-Mycielski graph, Brelaz's coloring

Table of Contents

An Example of New Use: Transitive Tournaments

A tournament is a directed graph, where there is an edge (arc) between any two nodes. A transitive tournament is where there is a sequence of the nodes, such that $\forall v_i, \forall v_j \in V, i < j : (v_i, v_j) \in E$ (note the analogy for the clique!)

We can search maximum transitive tournament in an arbitrary directed graph, or in tournament. Later was a problem inspected by Moon, Erdős and Mooser. In tournaments specially we cannot use coloring, as there is not a defined one. But we can use the previous modified colorings.

An Example of New Use: Transitive Tournaments

For example we can introduce s-free coloring, where we partition the nodes such that, in each partition there cannot be a transitive tournament of size s. The s-free coloring can be performed with moderate time limits, and with it's use possibly problems of unsolvable size can be solved. (As coloring introduced to CP did the same effect.)

Also we can color the arcs, with similar rules as for edges (whether two arcs complete a 3-transitive-tournament or a 4-transitive-tournament)

Possibility of similar methods in other discrete optimization problems?

SZABO, S. and ZAVALNIJ, B. „Greedy algorithms for triangle free coloring.” *AKCE International Journal of Graphs and Combinatorics.* 9:(2) pp. 169–186. (2012)

SZABO, S. and ZAVALNIJ, B. „Coloring the edges of a directed graph.” *Indian Journal of Pure and Applied Mathematics.* April 2014, Volume 45, Issue 2, pp 239–260.

SZABO, S. and ZAVALNIJ, B. „Coloring the nodes of a directed graph.” *Acta Univ. Sapientiae, Informatica.* 6, 1 (2014) 117—131.

SZABO, S. and ZAVALNIJ, B. „Estimating clique size via coloring edges in graphs.” *AKCE International Journal of Graphs and Combinatorics.* (submitted.)

Thank you for your attention!

Questions?

- SZABO, S. and ZAVALNIJ, B. „Greedy algorithms for triangle free coloring." *AKCE International Journal of Graphs and Combinatorics.* 9:(2) pp. 169–186. (2012)
- SZABO, S. and ZAVALNIJ, B. „Coloring the edges of a directed graph." *Indian Journal of Pure and Applied Mathematics.* April 2014, Volume 45, Issue 2, pp 239–260.
- SZABO, S. and ZAVALNIJ, B. „Coloring the nodes of a directed graph." *Acta Univ. Sapientiae, Informatica.* 6, 1 (2014) 117—131.
- SZABO, S. and ZAVALNIJ, B. „Estimating clique size via coloring edges in graphs." *AKCE International Journal of Graphs and Combinatorics.* (submitted.)

Thank you for your attention!

Questions?

- SZABO, S. and ZAVALNIJ, B. „Greedy algorithms for triangle free coloring.” *AKCE International Journal of Graphs and Combinatorics.* 9:(2) pp. 169–186. (2012)
- SZABO, S. and ZAVALNIJ, B. „Coloring the edges of a directed graph.” *Indian Journal of Pure and Applied Mathematics.* April 2014, Volume 45, Issue 2, pp 239–260.
- SZABO, S. and ZAVALNIJ, B. „Coloring the nodes of a directed graph.” *Acta Univ. Sapientiae, Informatica.* 6, 1 (2014) 117—131.
- SZABO, S. and ZAVALNIJ, B. „Estimating clique size via coloring edges in graphs.” *AKCE International Journal of Graphs and Combinatorics.* (submitted.)

Thank you for your attention!

Questions?