A simple and fast heuristic algorithm for edge-coloring of graphs

Miquel Àngel Fiol, Joan Vilaltella

Universitat Politècnica de Catalunya

Graph-TA Workshop, February 19, 2013. UPC, Barcelona.

Overview

- We present a simple randomized heuristic algorithm to 3-edge-color cubic graphs, when possible.
- It can be easily generalized to Δ-edge-color graphs with maximum degree Δ, when possible (Δ is assumed to be fixed).
- The heuristic algorithm must fail sometimes (edge-coloring is assumed to be hard), but the empirical performance of the algorithm on benchmark graphs and large random cubic graphs is good.
- The computational complexity is $O(n^2)$ where n is the number of nodes in the graph, but the measured average computing time behaves linearly on random cubic graphs.

Background

- The chromatic index of a Δ-regular graph is $\chi' = \Delta$ or $\chi' = \Delta + 1$ (Vizing, 1964).
- Deciding the exact value of χ' is an *NP*-complete problem even for $\Delta = 3$ (Holyer, 1981).
- Edge-coloring a cubic graph would solve the decision problem. Hence, finding an edge-coloring of a general cubic graph is assumed to be a hard problem.
- Therefore, partial, approximate or probabilistic solutions are of interest.

Definitions

- *Random edge-coloring*: an arbitrary edge-coloring where the color assigned to each edge is randomly chosen.
- *Conflicting node*: a node of the graph with two or more equally colored incident edges.
- *Conflict level of a node*: the number of color repetitions in its incident edges (0 if they are properly colored).
- *Kempe chain*: given a graph, a subgraph consisting in a path or cycle whose edges are alternately colored with two colors.
- *Kempe interchange* or *Kempe switch*: the switching of the two colors of a Kempe chain along it. The color switch in a Kempe cycle of an edge-colored graph does not create new conflicting nodes.

The CND algorithm: basic idea

- We call the heuristic algorithm *Conflicting Node Displacement (CND)*.
- The starting point is a (pseudo-)random edge-coloring of the graph.
- The basic idea is to displace every conflict along the nodes of a Kempe chain until either the conflict is solved or the chain cycles back (at most all nodes will be visited).
- The algorithm is guaranteed to stop in $O(n^2)$ time, where n is the number of nodes in the graph, but it may fail to find an existing 3-edge-coloring.
- In our tests with benchmark graphs (arbitrary Δ) and random cubic graphs of increasing size ($\Delta = 3$), the algorithm always found a Δ-edge-coloring if any.

The CND algorithm: description

- While the set of conflicting nodes is not empty:
- Choose a conflicting node v with conflict level 1
- Perform a Kempe interchange starting at v and ending after the conflict is solved or the Kempe chain ends or cycles back (after n color switches at most: in the worst case the chain is a Hamiltonian cycle)
- If the number of conflicting nodes has decreased: $c = 0$
- Else: add 1 to c
- If $c > R$: break While loop
- Else: continue While loop
- Restart from a random initial coloring at most L times (iterations), or until the graph is edge-colored
- Output the current edge-coloring and the number of conflicting nodes

Generalization to graphs with maximum degree Δ

- While the set of conflicting nodes is not empty:
- Choose a conflicting node v with high conflict level
- Perform a Kempe interchange starting at v and ending after the conflict is solved or the Kempe chain ends or cycles back (after n color switches at most: in the worst case the chain is a Hamiltonian cycle)
- If the number of conflicting nodes has decreased: $c = 0$
- Else: add 1 to c
- If $c > R$: break While loop
- Else: continue While loop
- Restart from a random initial coloring at most L times (iterations), or until the graph is edge-colored
- Output the current edge-coloring and the number of conflicting nodes

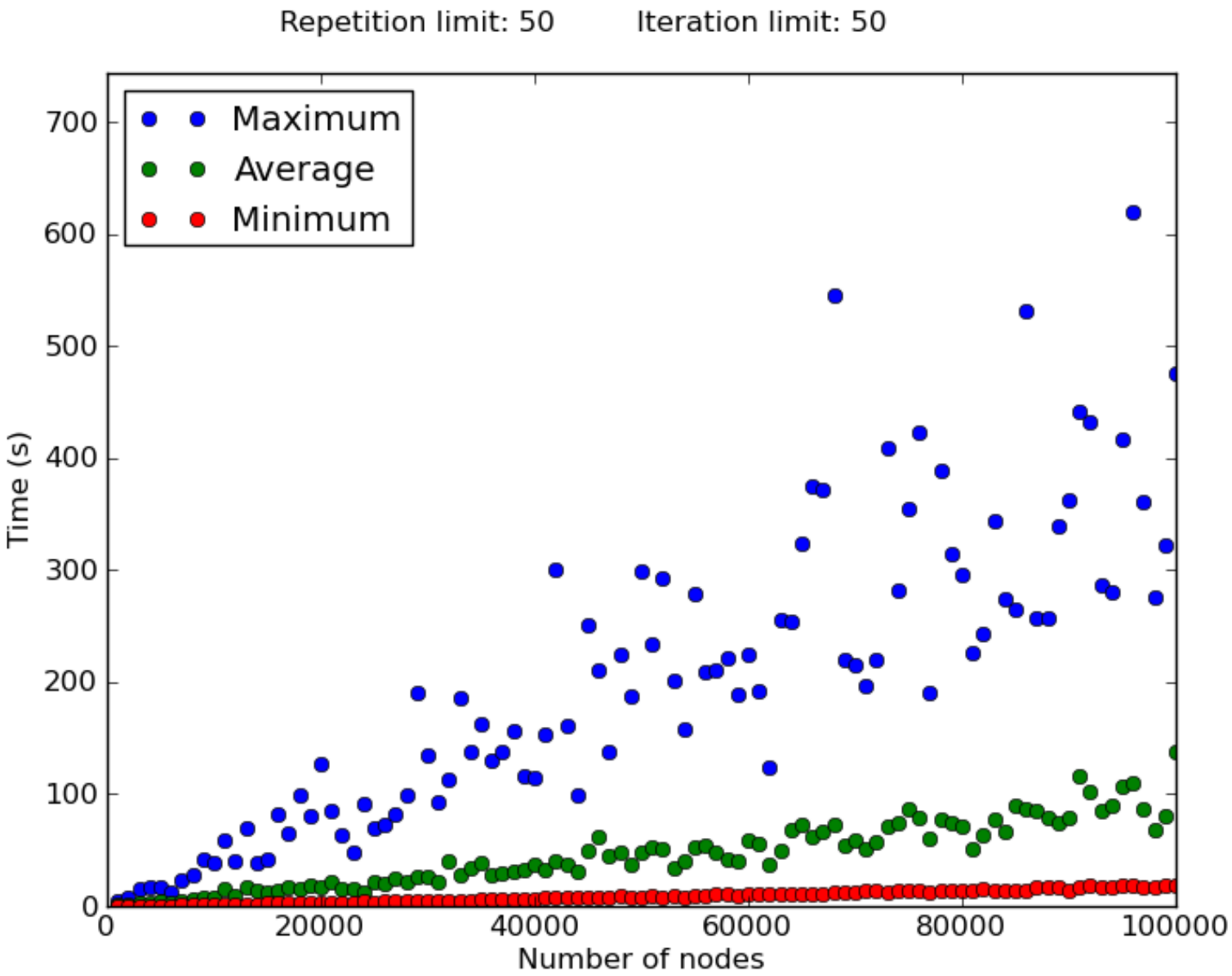
The CND algorithm: complexity analysis

- The algorithm restarts from a random initial coloring if the number of conflicting nodes has not decreased after R new choices, and the algorithm ends after L restarts.
- There are n nodes in the graph.
- After each choice, at most n color switches are done.
- Therefore, the maximum number of basic computational steps is RLn^2.
- The computational cost of each basic step depends on implementation details, but with an adequate implementation it can be bounded by a constant.
- Therefore, the computational complexity of the algorithm is $O(n^2)$. It is a worst-case estimation, with pessimistic assumptions.

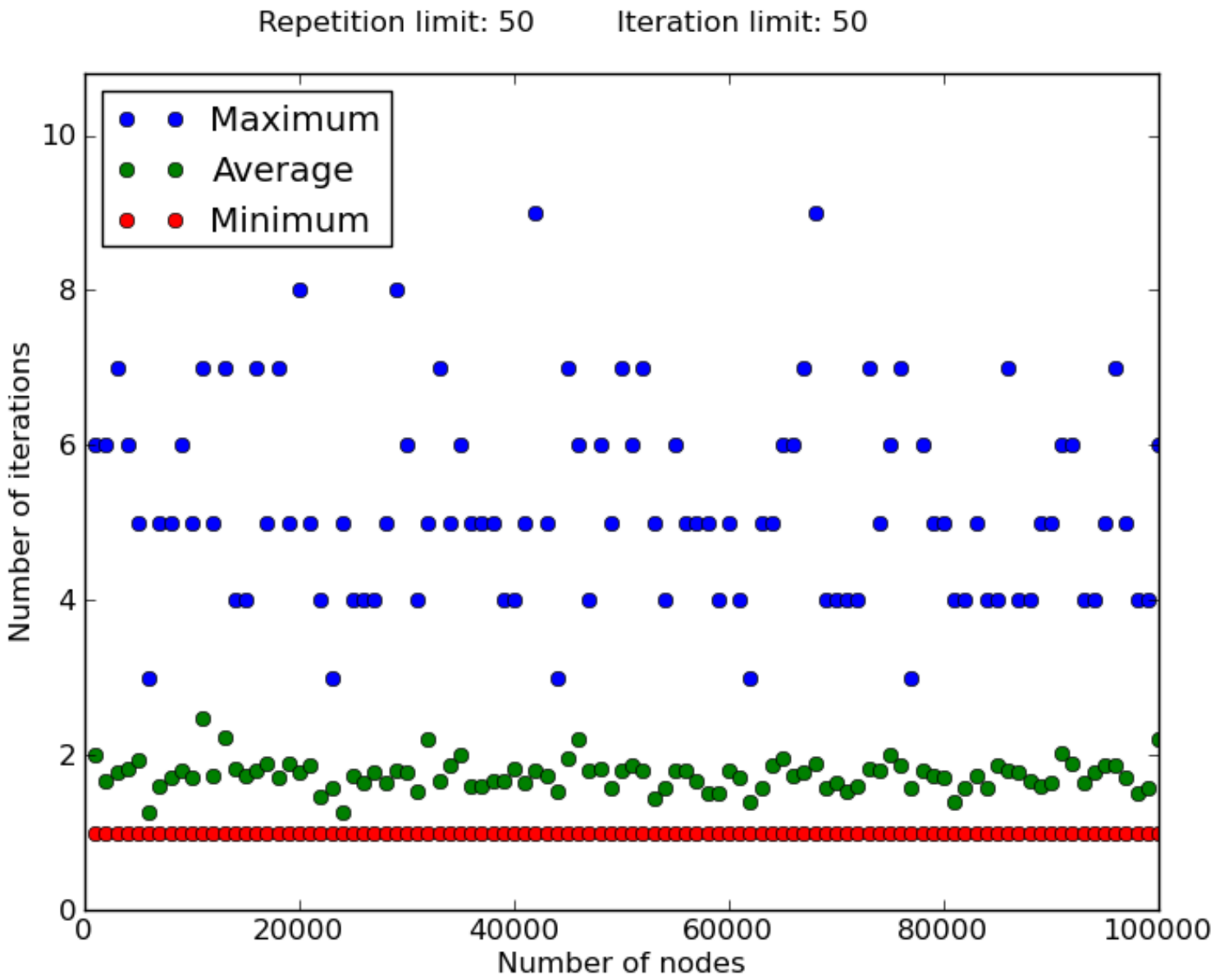
Empirical efficiency

- We plotted the minimum, average and maximum running time over sets of random cubic graphs of increasing order (30 graphs in each set).
- We also plotted the minimum, average and maximum values of the number of iterations and the average time per iteration.
- The behaviour of the average running time is linear, in agreement with the $O(n^2)$ estimation.
- The behaviour of the average time per iteration and the average number of iterations is, respectively, linear and constant.
- We obtained good performances with parameter values $R = 50$ and $L = 50$, even for large graphs (up to a million nodes).

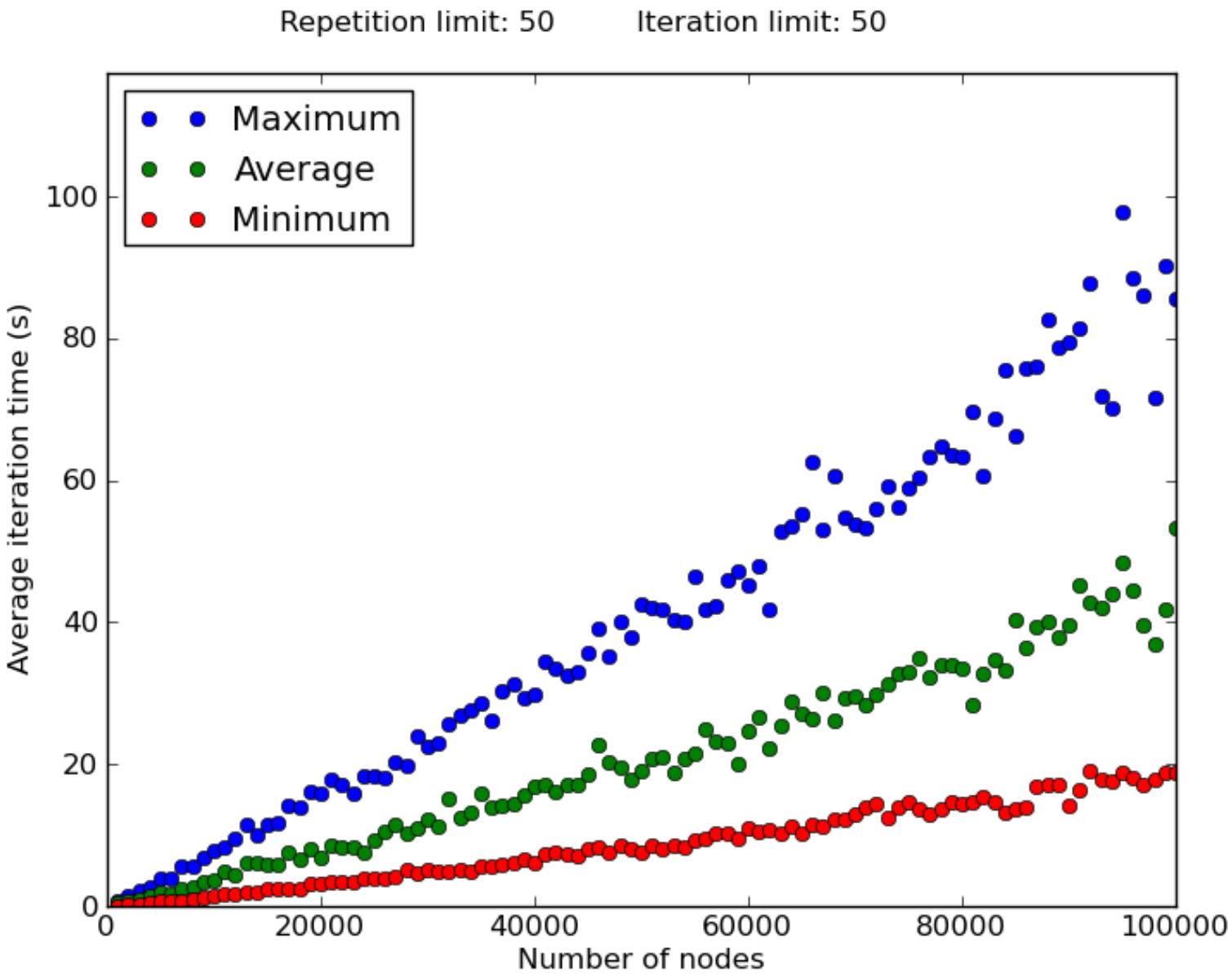
Data plots: edge-coloring time



Data plots: number of iterations



Data plots: time per iteration



Other fast heuristic algorithms

Fast Heuristics for the Edge Coloring of Large Graphs

Mario Hilgemeier, Nicole Drechsler, Rolf Drechsler
Euromicro Symposium on Digital System Design (DSD'2003)
pp. 230-237, Belek (Turkey), 2003
(a survey of simple heuristic algorithms)

Randomized Δ-Edge-Coloring via Quaternion of Complex Colors

Tony T. Lee, Yujie Wan, Hao Guan
arXiv:1104.1852v1 [cs.DS], 2011
(a faster, much more complicated, heuristic algorithm)

Efficient 3-vertex-coloring (recent)

Solving Hard Computational Problems Efficiently: Asymptotic Parametric Complexity 3-Coloring Algorithm.

Martín H., J.A. (2013)

PLoS ONE 8(1): e53437. doi:10.1371/journal.pone.0053437

Current and future lines of work

- Generalization to hypergraphs.
- Generalization to other edge-coloring rules.
- Test the performance of the CND algorithm on problems which are reducible to edge-coloring.
- Explore other conflict-based heuristic algorithms for similar problems, and test their performance.

Thanks

Thanks for your attention.

joan.vilaltella@ma4.upc.edu