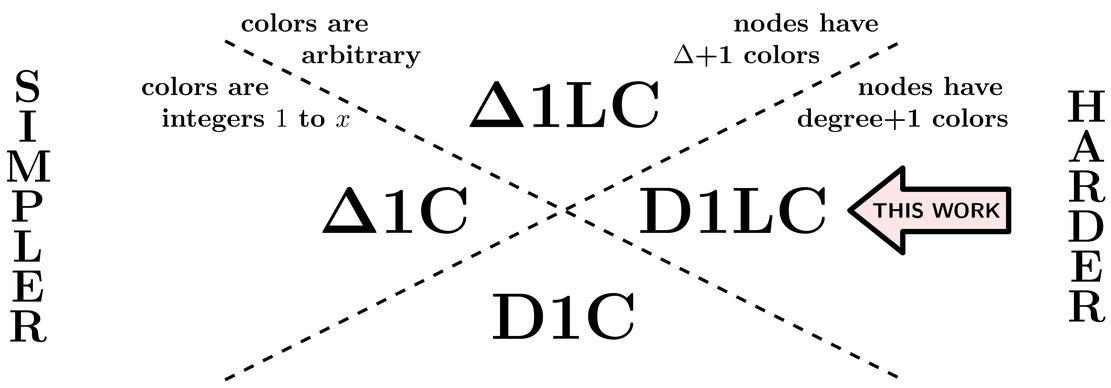
Near-Optimal Distributed Degree+1 Coloring

Magnús M. Halldórsson[1], Fabian Kuhn[2], Alexandre Nolin[1], Tigran Tonoyan[3]
[1]Reykjavik University [2]University of Freiburg [3]Krisp Technologies Inc.
mmh@ru.is; kuhn@cs.uni-freiburg.de; alexandren@ru.is; ttonoyan@gmail.com

1. Coloring problems



We aim to solve the problem *with high probability* (w.h.p.), which here means with probability $1 - 1/\operatorname{poly}(n)$, where $n = |V|$ is the number of nodes in the graph. Each node v has a *palette* (list of colors) of size at least $\deg(v) + 1$.

2. State of the art and our results

Model	Δ1C		D1LC before	D1LC now	
LOCAL	$\log^3 \log n$	[CLP20]	$\log n$	$\log^3 \log n$	this work
CONGEST	$\log^5 \log n$	[HKMT21]	$\log n$	$\log^5 \log n$	[HNT22]
Semi-streaming	YES	[ACK19]	?	YES	this work
Query	$\widetilde{O}(n^{1.5})$	[ACK19]	?	$\widetilde{O}(n^{1.5})$	this work
$\widetilde{O}(n)$ space MPC	$O(1)$	[ACK19]	?	$O(1)$	this work
CONGESTED CLIQUE	$O(1)$	[CFG+19]	?	?	

(randomized complexities)

We also obtain a palette sparsification result, which is the basis of our results for Semi-streaming, Query, and MPC, following a framework of []. More precisely, we show that if each node samples $O(\log^2 n)$ colors from its palette, w.h.p., the nodes can still solve D1LC even if each of them restricts itself to the colors it sampled.

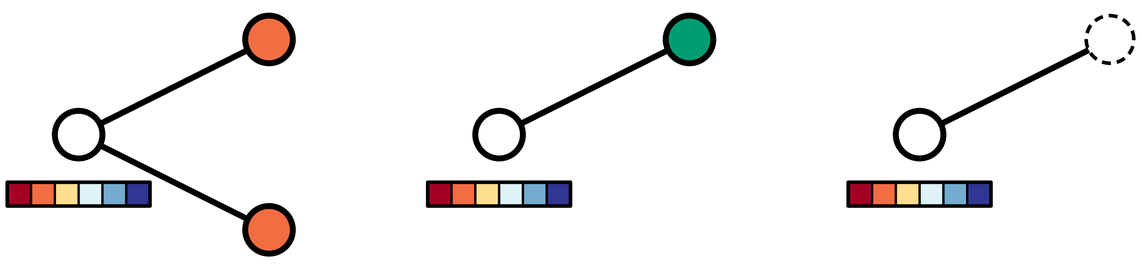
3. Slack Generation

The key to Δ1C vs D1LC

Slack of a node v. Difference between how many colors v has vs how many neighbors it has, $|\Psi_v| - d_v$.

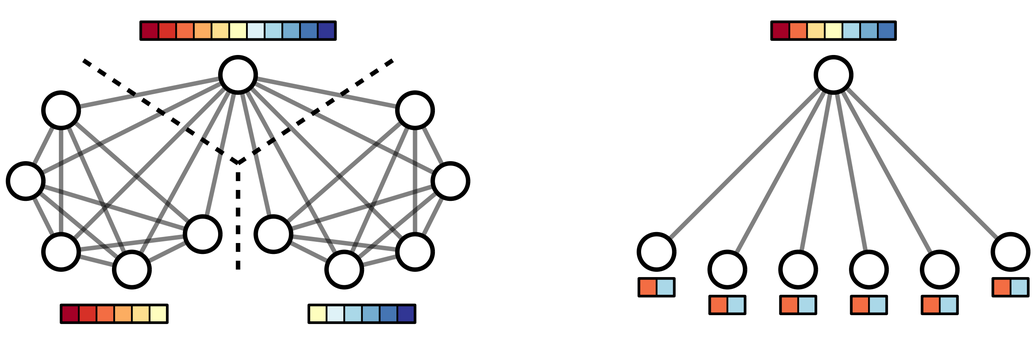
Idea: the higher it is, the less "competition" v has when it tries a random color.

There's essentially 3 types of slack:



First type is when two nodes in the neighborhood of v pick the same color. Second type is when a neighbor of v takes a color out of v's palette. Third type is when neighbors turn off temporarily to reduce competition for a while.

The left figure in the following example (from [CLP20]) shows that the third kind is absolutely necessary in D1LC, contrary to Δ1C. This is due to the fact that non-edges between nodes that do not share colors cannot provide slack. The right figure shows how the contribution to slack of a single color can be very large in D1LC, also in contrast with Δ1C.



We overcome the difficulty by giving a procedure that generates permanent slack for a subset of the nodes, and such that nodes that do not receive permanent slack can get significant temporary slack from the nodes that did by momentarily turning them off.

4. Some important concepts

Almost-clique decomposition (ACD): We partition the nodes into dense (many edges in neighborhood), sparse (few edges in neighborhood), and uneven (neighbors have much higher degree) nodes, following a version of [AA20]. Dense nodes are furthermore partitioned into diameter-2 sets called "almost-cliques".

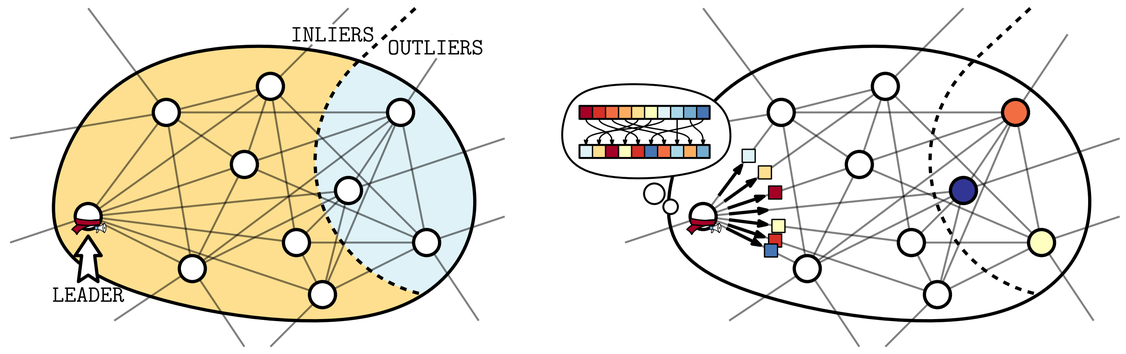
LOCAL model: The main model we study. Nodes on a graph are computationally unbounded agents, sending each other messages of arbitrary size over the edges in synchronous rounds. The complexity is the number of rounds.

Shattering: A frequent technique in distributed computing, in which a randomized algorithm is used to solve large parts of the graph, only leaving small unsolved parts to be dealt with using a deterministic algorithm.

Trying a color: The basic step in many coloring algorithms. A node "trying a color" consists of the node announcing the color to its neighbors and keeping it if none of its neighbors are already colored with this color or currently broadcasting it. Can be made more complex by introducing priorities between the nodes.

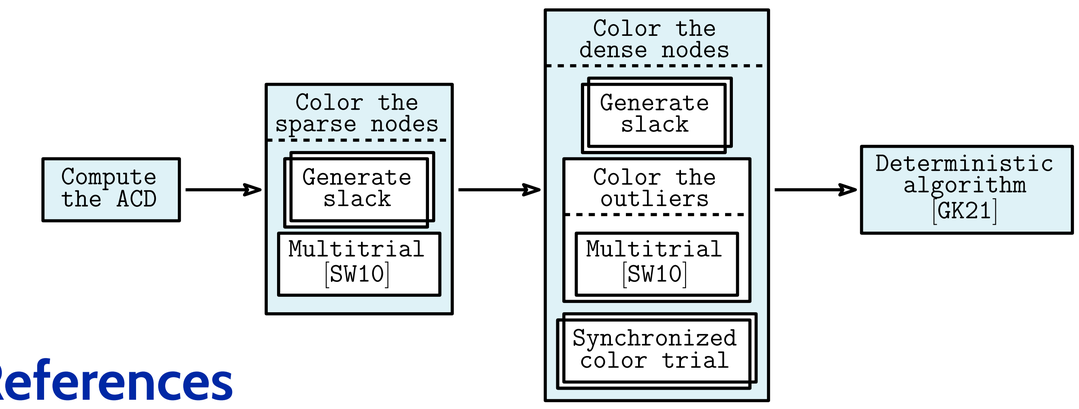
Multitrial: Given enough slack, nodes can try multiple colors in a single round to color themselves faster than if they were trying one color at a time. The secret behind many randomized $O(\log^* n)$ algorithms for coloring.

5. Dense primitive



A key idea in our algorithm is to have each almost-clique choose an appropriate *leader*, color nodes too distinct or disconnected from it (*outliers*) using temporary slack given by the other nodes (the *inliers*), and color the inliers efficiently by having the leader decide for them which colors they should try. Having the leader make this decision on their behalf ensures that color conflicts do not occur within the nodes receiving these colors. The leader simply shuffles its palette to give the inliers colors to try.

6. Sketch of algorithm



References

[AA20] Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta + 1)$ vertex coloring. In *APPROX/RANDOM*, 2020.

[ACK19] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In *SODA*, 2019.

[BEPS16] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM*, 2016.

[CFG+19] Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of (Δ+1) coloring in congested clique, massively parallel computation, and centralized local computation. In *PODC*, 2019.

[CLP20] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta + 1)$-coloring via ultrafast graph shattering. *SIAM Journal of Computing*, 2020.

[GK21] Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *FOCS*, 2021.

[HKMT21] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In *STOC*, 2021.

[HNT22] Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Overcoming congestion in distributed coloring. *CoRR*, abs/2205.14478, 2022. to appear at PODC'22.

[Ree98] Bruce A. Reed. ω, Δ, and χ. *J. Graph Theory*, 1998.

[SW10] Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In *PODC*. ACM, 2010.