ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, Emo Welzl, Peter Widmayer

Solution 1: Reducing the Number of Colors in a Single Round

To be precise, the method reduces the numbers of colors from k to $k' = 2\lceil \log_2 k \rceil$. (We can set $C_0 = 2$, i. e. we do not really need the constant C_0 – except that we should take note of the fact that $k' < k$ holds only for $k \geq 7$. For some small values it can even happen that the number of colors increases.)

Let $\phi_{\text{old}} : V \to \{0, \ldots, k-1\}$ be given.

Well-definedness: Let $v \in V$ and let u be its parent. Since ϕ_{old} is a proper coloring, we have $\phi_{\text{old}}(v) \neq \phi_{\text{old}}(u)$ and hence there actually exists a smallest index i_v wherein these two numbers differ. If we let $\ell = \lfloor \log_2(k-1) \rfloor + 1 = \lceil \log_2 k \rceil$ denote the number of bits in the bit representation of the largest color $(k-1)$, then we have $i_v \in \{1, \ldots, \ell\}$. Hence we have defined a map

$$\begin{array}{rccc} \phi_{\text{new}} & : & V & \longrightarrow & [\ell] \times \{0,1\} \\ & & v & \longmapsto & (i_v, b_v) \end{array}$$

whose range is indeed of cardinality at most $2\ell = k'$.

Properness: Assume that v and w are neighbours such that $\phi_{\text{new}}(v) = \phi_{\text{new}}(w)$. Assume without loss of generality that v is the parent of w, and let u denote the parent of v. Also without loss of generality assume that $b_v = 0$. Then the i_vth bit of $\phi_{\text{old}}(u)$ is 1, and from $i_v = i_w$ it follows that b_w is the i_vth bit of w and $b_w = 1$. This contradicts the assumption $(b_v = b_w)$.

Solution 2: Coloring Trees That Are Not Rooted

Remark. What we will show here precisely is that if we have a determinstc $o(\log_\Delta n)$-round LOCAL algorithm for coloring trees with colors in $\{1, \ldots, K\}$, then there are trees on which the algorithm needs $K \in \Omega(\Delta / \log \Delta)$. We will *not* show here that the algorithm may also need to use all colors between 1 and K.

Notation for the graphs given in the question: $G(n, \Delta)$.

Let $\mathcal{A}$ be any algorithm for coloring trees, and let $r(n, \Delta)$ denote its round complexity. Assume $r(n, \Delta) = o(\log_\Delta n)$. In words this means that the round complexity is asymptotically dominated by the girth of $G(n, \Delta)$. In particular there exist numbers n_0, Δ_0 such that, for all $n \geq n_0$ and all $\Delta \geq \Delta_0$, $2 \times r(n, \Delta)$ is strictly smaller than the girth of $G(n, \Delta)$.

Now let $n \geq n_0$ and $\Delta \geq \Delta_0$. We argue that we can run $\mathcal{A}$ on $G(n, \Delta)$, even though $G(n, \Delta)$ is not a tree. Indeed, in $r(n, \Delta)$ rounds, every single vertex can only learn (everything about) its $r(n, \Delta)$-neighbourhood; so every single vertex v will do the same operations as in the tree G_v

obtained from G by deleting all edges whose distance from v is larger than r. The resulting coloring will be proper (because if v picked the same color as some neighbour u, then the same would happen in the tree obtained from G by deleting all edges whose distance from v *and* u is larger than r).

We have shown that, for n, Δ sufficiently large, $\mathcal{A}$ computes a coloring of $G(n, \Delta)$. Hence the number of colors used by $\mathcal{A}$ must be at least as large as the chromatic number of $G(n, \Delta)$, which answers the question.

Solution 3: Randomized Network Decomposition

For completeness's sake we give the solution for the general setting, i.e. for any $\varepsilon \in (0, 1)$. It suffices to show that, for some $\alpha > 0$, the event "$\max_u r_u \leq \frac{\alpha}{\varepsilon} \log n + 1$" happens with high probability.

For every fixed vertex u we have (calculating with the geometric distribution):

$$\begin{aligned}
\Pr\left[r_u > \frac{\alpha}{\varepsilon} \log n + 1\right] &= 1 - \Pr\left[r_u \leq \frac{\alpha}{\varepsilon} \log n + 1\right] \\
&= 1 - \sum_{y=1}^{\lfloor \alpha \log n/\varepsilon \rfloor + 1} \Pr[r_u = y] \\
&= 1 - \sum_{y=1}^{\lfloor \alpha \log n/\varepsilon \rfloor + 1} \varepsilon(1-\varepsilon)^{y-1} \\
&= 1 - \varepsilon \sum_{y=0}^{\lfloor \alpha \log n/\varepsilon \rfloor} (1-\varepsilon)^{y} \\
&= 1 - \varepsilon \frac{1 - (1-\varepsilon)^{\lfloor \alpha \log n/\varepsilon \rfloor + 1}}{1 - (1-\varepsilon)} \\
&= (1-\varepsilon)^{\lfloor \alpha \log n/\varepsilon \rfloor + 1} \\
&\leq (1-\varepsilon)^{\alpha \log n/\varepsilon} \leq e^{-\alpha \log n} = n^{-\alpha}.
\end{aligned}$$

Now we use a union bound:

$$\Pr\left[\max_u r_u > \frac{\alpha}{\varepsilon} \log n\right] = \Pr\left[\exists u : r_u > \frac{\alpha}{\varepsilon} \log n\right] \leq \sum_u \Pr\left[r_u > \frac{\alpha}{\varepsilon} \log n\right] \leq n^{1-\alpha},$$

and the statement follows if we let $\alpha = 3$.

Solution 4: More Algorithms for Color Reduction

(a) In order to make our life easier and get rid of rounding issues, we change the question a little: We assume $k \geq \Delta + 2$ (otherwise we cannot in general reduce the number of colors) and we let $k' = k - \lceil \frac{k}{2\Delta+4} \rceil$.

We put the colors of the given k-coloring into $\lfloor \frac{k}{\Delta+2} \rfloor$ buckets, each of size $\Delta+2$ except the last one, which may have size between $\Delta+2$ to $2\Delta+3$. Our assumption above ($k \geq \Delta+2$) guarantees that this is possible. Within each bucket we can in one round reduce the number of colors by 1, using the method from lemma 8.17. We can do this in all buckets in parallel, so within one round we have reduced the number of colors by $\lfloor \frac{k}{\Delta+2} \rfloor$. This number is certainly $\geq \lceil \frac{k}{2\Delta+4} \rceil$ for all $k \geq \Delta+2$.

(The choice of the factor 2 before the Δ was quite arbitrary; all that matters is that in this way we were able to change the $\lfloor\rfloor$ into a $\lceil\rceil$. You do not need to worry too much about such rounding technicalities; it is easy to get them wrong. But you should understand that we need the $\lceil\rceil$ in order to solve the next question.)

(b) First we run the $O(\log^* n)$ algorithm that gives us a $C\Delta^2$-coloring for some constant C. Then we repeat the algorithm from (a) as often as possible, that is, until $k \leq \Delta+1$.

In order to analyze the number of iterations of (a) that we need, observe that the number of colours drops in each iteration by at least a $(1-\frac{1}{2\Delta+4})^{-1}$-factor, hence the number of iterations is bounded by

$$\log_{(1-\frac{1}{2\Delta+4})^{-1}}(C\Delta^2) = \frac{\ln(C\Delta^2)}{\ln(2\Delta+4)-\ln(2\Delta+3)} \leq \frac{\ln(C\Delta^2)}{1/(2\Delta+4)} = O(\Delta \ln \Delta)$$

where we have used the logarithm inequality from Special Assignment 1, 2b.

It follows from the termination criterion of our algorithm that the final coloring uses no more that $\Delta+1$ colors.