



Competitive Vertex Recoloring

(Online Disengagement)

Yossi Azar¹ · Chay Machluf² · Boaz Patt-Shamir² · Noam Touitou¹

Received: 25 August 2022 / Accepted: 29 November 2022 / Published online: 3 January 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Motivated by placement of jobs in physical machines, we introduce and analyze the problem of online recoloring, or online disengagement. In this problem, we are given a set of n weighted vertices and a k -coloring of the vertices (vertices represent jobs, and colors represent physical machines). Edges, representing conflicts between jobs, are inserted in an online fashion. After every edge insertion, the algorithm must output a proper k -coloring of the vertices. The cost of recoloring a vertex is the vertex's weight. Our aim is to minimize the competitive ratio of the algorithm, i.e., the ratio between the cost paid by the online algorithm and the cost paid by an optimal, offline algorithm. We consider a couple of polynomially-solvable coloring variants. Specifically, for 2-coloring bipartite graphs we present an $O(\log n)$ -competitive deterministic algorithm and an $\Omega(\log n)$ lower bound on the competitive ratio of randomized algorithms. For $(\Delta + 1)$ -coloring, where Δ is the maximal node degree, we present tight bounds of $\Theta(\Delta)$ and $\Theta(\log \Delta)$ on the competitive ratios of deterministic and randomized algorithms, respectively (where Δ denotes the maximum degree). We also consider the fully dynamic case which allows edge deletions as well as insertions. All our algorithms are applicable to the case where vertices are arbitrarily weighted, and all our lower bounds hold even in the uniform weights (unweighted) case.

Keywords Coloring with recourse · Anti-affinity constraints · Online migration

✉ Boaz Patt-Shamir
boaz@tau.ac.il

Yossi Azar
azar@tauex.tau.ac.il

Chay Machluf
chay.machluf@gmail.com

Noam Touitou
noam.touitou@cs.tau.ac.il

¹ School of Computer Science, Tel Aviv University, Tel Aviv, Israel

² School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel

1 Introduction

The following situation is not uncommon in server farms, such as a data center of a cloud provider: Jobs (or virtual machines) are created and located at various physical machines, and then it turns out that some of the jobs cannot co-exist on the same machine. Such restrictions may be due to various reasons, e.g., conflicting resource requirements, security considerations etc. (cloud providers allow users to express these constraints using so-called “anti-affinity rules:” see, e.g., [1]). When such a conflict arises between co-located jobs, at least one of these jobs must migrate to another machine, at a cost. Motivated by such scenarios, in this we introduce and study the *Disengagement Problem*, in which disengagement (anti-affinity) requests arrive online, and the goal is to minimize the overall cost of job migrations.

More specifically, we abstract the problem as follows (more details in Sect. 2). We view the problem as an online graph coloring problem with recourse. Initially, we are given a vertex-weighted graph $G_0 = (V, \emptyset)$ of n isolated vertices, and a coloring $c_0 : V \rightarrow [k]$, where $k \in \mathbb{N}$ is a given parameter.¹ (Vertices correspond to jobs and colors correspond to machines.) The input is a sequence of edges e_1, e_2, \dots , that arrive one at a step. After each step i , we are required to output a new *proper* vertex coloring $c_i : V \rightarrow [k]$ such that under c_i , no edge in e_1, \dots, e_i is monochromatic. The cost of a given sequence of colorings c_0, c_1, \dots is the total weight of recolorings, i.e., the sum, over all vertices, of vertex weight times the number of times that vertex was recolored. Due to this formalization, in this paper we refer to the problem interchangeably as Vertex Recoloring or Disengagement.

We note that in many cases, disengagement restrictions are temporary. We refer to this setting as “fully-dynamic disengagement,” as opposed to “partially-dynamic disengagement”, in which edges are only added, and never removed. In this work, we mainly focus on the partially-dynamic variant.

Our Results We study online disengagement from the competitive analysis viewpoint [16]. That is, an algorithm for online disengagement is said to be c -competitive for some $c \geq 1$ if the following holds: given any instance, the recoloring cost of the algorithm is at most c times that of the optimal offline algorithm. (Note that an optimal offline solution never recolors a vertex more than once.) Since vertex coloring is a hard problem [18], we focus on two cases which are polynomially solvable, namely 2-colorable graphs and $(\Delta + 1)$ -coloring, where Δ denotes the maximum vertex degree.

1. For 2-coloring, we give a deterministic, $O(\log n)$ -competitive algorithm. We also show a matching lower bound of $\Omega(\log n)$ on the competitiveness of *randomized* algorithms.
2. For $(\Delta + 1)$ -coloring, we present:
 - A deterministic, $O(\Delta)$ -competitive algorithm, and a matching $\Omega(\Delta)$ lower bound.
 - A randomized, $O(\log \Delta)$ -competitive algorithm, and a matching $\Omega(\log \Delta)$ lower bound.

¹ We use the notation $[k] \stackrel{\text{def}}{=} \{1, 2, \dots, k\}$.

In Sect. 5.1, we give general lower bounds for the disengagement problem on a graph of n nodes with k colors: we show an $\Omega(k \log \frac{n}{k})$ lower bound for deterministic algorithms, and an $\Omega(\log k \log \frac{n}{k})$ lower bound for randomized algorithms.

We also briefly mention capacitated disengagement in Sect. 5.2, wherein there is a restriction upon the total weight that can be assigned to each color. If the total permissible weight (across all colors) equals the total vertex weight, we show a deterministic competitive ratio lower bound of $\Omega(n)$, and a randomized lower bound of $\Omega(\log(n))$.

In all the above cases, our algorithms work with weighted vertices, and our lower bounds apply even to unweighted instances.

We discuss the variant of *fully-dynamic* online disengagement, in which conflicts are temporary, as opposed to the *partially-dynamic* disengagement in which conflicts never disappear (see formalization in Sect. 2). It turns out that fully-dynamic disengagement is substantially harder than partially-dynamic disengagement. More specifically, even though in this case the offline cost is not bounded by n , all lower bounds of the partially-dynamic case apply as well. We also show that the special case of fully-dynamic disengagement with two colors on an odd cycle (which of course is not 2-colorable in the partially-dynamic case) is at least as hard as a certain metrical task system on an odd cycle, and hence admits a lower bound of $\Omega(n)$ -competitiveness for deterministic algorithms (whereas the partially-dynamic problem on even cycles is solvable by an $O(\log n)$ -competitive deterministic algorithm).

Our Techniques For the bipartite (2-coloring) problem, our algorithm maintains a partition of the vertices into connected components. Since there are two colors, each connected component has exactly two legal colorings. When two components merge due to the arrival of a monochromatic edge, the algorithm must choose between the two legal colorings for that new component, where each coloring implies recoloring one of the joined components. There are two natural heuristics for choosing which component to recolor. The greedy approach is to recolor the lighter component; another approach, with an eye to the offline solution (which, knowing all future edges, recolors every vertex at most once), is to recolor so as to minimize the change from the initial coloring of the graph (as measured in weighted Hamming distance). However, it is not hard to see that each of these algorithms has $\Omega(n)$ competitive ratio (cf. Sect. 3.1). Fortunately, it turns out that by a careful combination of these two noncompetitive approaches (using amortized analysis), we can obtain an $O(\log n)$ -competitive deterministic algorithm.

For $(\Delta + 1)$ -coloring, the key observation is that vertices that are recolored by the optimal algorithm must be incident on all edges which are monochromatic by their initial coloring. Therefore, our algorithm maintains a vertex cover of the edges which are monochromatic by their initial coloring. In our algorithm, upon the arrival of an edge, only vertices in that vertex cover are allowed to change their color. This way, we limit the set of vertices that change their color. Coupling this with an upper bound on the total number of recolorings which a vertex undergoes (using Δ), we can prove a bound on the competitive ratio. Specifically, we maintain a 2-approximate weighted vertex cover using the primal-dual (local ratio) 2-competitive algorithm; when a monochromatic edge arrives, an appropriate vertex from the vertex cover is recolored by a color not taken by any of its neighbors. In the deterministic case, which achieves $O(\Delta)$ -competitiveness, the new color is an arbitrary available color, and in

the randomized case, the new color is a random (uniformly chosen) available color. In the latter case, some subtle probabilistic analysis shows $O(\log \Delta)$ -competitiveness.

Related Work We do not know of previous work on *competitive recoloring* of vertices: some considered recoloring, some considered competitive coloring, but this is the first work to consider both simultaneously. We review some relevant results below.

Competitive Coloring Let us start with competitive coloring. Work on online vertex coloring, starting with the paper of Lovász et al. [15], assume that vertices arrive one by one, each vertex with all its incident edges. When a vertex arrives it must be assigned a color irrevocably, maintaining a legal vertex coloring at all times. The goal is to minimize the number of colors used by the algorithm. It is known that the competitive ratio in this case is $\Theta(n/\log n)$ [10, 11].

Dynamic Algorithms for Recoloring Recoloring has been considered in the dynamic data structures setting, in which one seeks to maintain a proper coloring while minimizing the number of recolorings per vertex/edge update (arrival or departure); see, e.g., [5, 8, 14, 17]. This model differs from ours in that we do not measure costs w.r.t. the number of steps, but w.r.t. an optimal recoloring algorithm OPT for the input sequence. In particular, when $\text{OPT} \ll T$, where T is the length of the input sequence, we remain competitive against OPT (rather than T); however, in dynamic recoloring (or recoloring with recourse), a constant number of recolorings per edge arrival results in $\Omega(T)$ cost.

For general graphs, coloring a graph using a competitively-small palette (w.r.t. its chromatic number) is NP-hard; thus, restricting the set of graphs or their chromatic number is necessary for polynomial running time. This is the case for the work of Kashyop et al. [14] (which focuses on bipartite graphs, max-degree bounded graphs, and graphs with bounded arboricity), and Bosek et al. [8] (which focuses on interval graphs). Indeed, our work also focuses on coloring problems which can be tackled in polynomial time, namely 2-coloring and $(\Delta + 1)$ -coloring. However, one can also consider coloring in general graphs, using an (exponential time) oracle for graph coloring. This is the case for the works of Barba et al. [5] and Solomon and Wein [17]. The best current result, due to [17], is a deterministic algorithm that maintains a coloring which is $O(\frac{\log^3 n}{d})$ -competitive with respect to the number of colors (against the graph's chromatic number), while using $O(d)$ amortized recolorings per update, where d is a free parameter (allowing randomization improves this result slightly). Henzinger et al. [13] give better results for bounded arboricity graphs.

Dynamic Graph Partitioning A problem closely related to online disengagement is “dynamic balanced graph partitioning,” introduced by Avin et al. [2, 3]. In this problem vertices are located in finite-capacity clusters (servers), and communication requests arrive online. Communication between vertices incurs cost unless the vertices are located in the same cluster. Vertices can migrate to other clusters, at a (larger) cost, and the algorithm is required to find a placement of the vertices in clusters at each step. The goal is to minimize cost, and the overall measure is the competitive ratio, possibly with resource augmentation (i.e., assuming that the capacity available to the algorithm is larger than the capacity available to the adversary). Intuitively, this problem is the flip side of disengagement: In partition, requests are to collocate vertices (subject to

capacity constraints), whereas in disengagement, requests are to separate them. In [3], the fully-dynamic variant of the partition problem is considered, where collocation requests are temporary: a pair of vertices may be collocated by a request and later separated. It is shown in [3] that the competitive ratio of deterministic algorithms for the dynamic case is $O(k \log k)$ and $\Omega(k)$, where k is the capacity of a cluster (both bounds allow for resource augmentation). Recently, the partially-dynamic variant of the partition problem was considered, in which it is guaranteed that there exists a feasible placement of the vertices to clusters so that all communication is local (i.e., occurs within a cluster). In [12], tight bounds on the competitive ratios of deterministic and randomized algorithms for partially-dynamic partition are given: $\Theta(\ell \log W)$ and $\Theta(\log \ell + \log W)$, respectively, where ℓ is the number of servers and W is the server capacity.

Paper Organization The remainder of this paper is organized as follows. In Sect. 2 we formalize the model and introduce some notation. In Sect. 3 we study the bipartite case. In Sect. 4 we study $(\Delta + 1)$ -disengagement. In Sect. 5 we prove a general lower bound for recoloring using k colors, and also discuss the capacitated case. In Sect. 6 we consider fully-dynamic disengagement. We conclude in Sect. 7 with a few open problems.

2 Model and Notation

Preliminaries Given a natural number k , we use $[k]$ to denote $\{1, 2, \dots, k\}$. In this paper we are concerned with colorings of vertices in graphs. Given an undirected simple graph $G = (V, E)$, a *proper k -coloring* of G is an assignment $c : V \rightarrow [k]$ such that $c(u) \neq c(v)$ for all $(u, v) \in E$.

We shall use the following definition extensively.

Definition 2.1 Let $G = (V, E)$ be a graph, and let $w : V \rightarrow \mathbb{R}^+$ be a vertex weight function. Let c, c' be two colorings of a graph G . The *weighted Hamming distance* between c and c' , denoted $\delta_H(c, c')$, is $\delta_H(c, c') \stackrel{\text{def}}{=} \sum_{v:c(v) \neq c'(v)} w(v)$.

Problem Statement In this paper, we study the following problem.

ONLINE DISENGAGEMENT (VERTEX RECOLORING)

Initial Input:

- A set V of n vertices with weights $w : V \rightarrow \mathbb{R}^+$
- The number of colors $k \in \mathbb{N}$
- A k -coloring c_0 of V

Online Input: In each step $i = 1, \dots, m$, an edge e_i between two vertices of V .

Output: After each step i , a proper k -coloring of the vertices c_i w.r.t edges $\{e_1, \dots, e_i\}$.

Goal: Minimize the total weight of vertex recolorings, i.e., $\sum_{i=1}^m \delta_H(c_{i-1}, c_i)$.

We use $E_i = \{e_1, e_2, \dots, e_i\}$ to denote the set of edges that have arrived up to and including the i th step, and we use $G_i = (V, E_i)$ to denote the known graph after step i .

Note that in Online Disengagement, as is the case for any partially-dynamic graph problem, the input sequence length can be assumed to be finite without loss of generality, as there are only $\binom{n}{2}$ possible distinct edges connecting vertices of V .

Special Cases In this paper we consider some special cases of online disengagement. The variants we consider are the following.

- In *Unweighted* Online Disengagement, all vertices have weight 1.
- In *Bipartite* Online Disengagement, we are guaranteed that the input graph is 2-colorable.
- In $(\Delta + 1)$ Online Disengagement, we are guaranteed that the maximum degree of the input graph does not exceed Δ , and the available number of colors is $\Delta + 1$.

Capacitated Disengagement

In Sect. 5.2 we consider capacitated disengagement, in which each color $i \in [k]$ has a capacity $W_i > 0$, and the total weight of vertices colored i must not exceed W_i .

Fully-dynamic Disengagement

Online disengagement, as stated above, is called *partially dynamic*, in the sense that edges only arrive and never leave. However, in some cases, the presence of edges restricting the solution may be temporary. In the *fully-dynamic* online disengagement variant, edges may leave too. The input and the output to the problem, as well as the cost measure, are the same, but the feasibility requirement is different: The coloring output by the algorithm after each step need be a proper coloring for $G_j = (V, E'_j)$, where $E'_j \subseteq E_j$. We discuss several different definitions of E'_j in Sect. 6.1.

3 Bipartite Disengagement

In this section we consider disengagement for bipartite graphs and $k = 2$. In other words, we are promised that the final graph G (and hence every intermediate graph) is 2-colorable, and the requirement is to find, at each step, a legal 2-coloring, while minimizing the overall cost.

Note that this case is stricter than $(\Delta + 1)$ -disengagement, since the maximum degree Δ is typically large (the case $\Delta = 1$ is trivial). Indeed, we show in this section that the competitive ratio in this case is $\Theta(\log n)$. We start with an $O(\log n)$ -competitive deterministic algorithm for the weighted case, and then prove an $\Omega(\log n)$ lower bound on the competitive ratio of any (randomized) algorithm that holds even in the unweighted case.

3.1 Simple Approaches

Any algorithm for 2-coloring maintains, throughout its execution, the partition of the nodes into connected components. Since there are only 2 colors, each connected component has two possible proper colorings. Whenever two components are joined by a new edge, if their colorings agree (i.e. the two ends of the arriving edge are colored differently), the algorithm may output the previous coloring unaltered. Otherwise, when the edge is monochromatic, the algorithm must decide which of the two colorings is adopted by the newly created component: each coloring implies recoloring all nodes

in one of the components being joined. There are two natural approaches one can consider. One approach we call “greedy” is to recolor the component of the smaller weight; another approach we call “conservative” is to choose the colorings which is closer to the initial coloring. Both approaches are not competitive, as we show below. For convenience, we refer to the two colors as red and blue.

The Greedy Algorithm First, consider the algorithm that always flips the color of the smaller component (weight-wise). Assume without loss of generality that we have at least two nodes initially colored red, and consider an input sequence of a connected component in the form of a double star:

1. Create an edge between two arbitrary red nodes. The node that stays red becomes the *red hub*, and the other one becomes the *blue hub*.
2. Do until there are no isolated nodes:
 - (a) Let v be some isolated node.
 - (b) Create a monochromatic edge between v and its matching hub.

Every isolated node is a component of size 1. When it is connected via an edge to the component of the hubs (which is at least of size 2) the algorithm flips its color. In fact, the online algorithm flips the color of every node in the graph except for the red hub (paying a cost of $n - 1$), while the optimal algorithm only chooses the other coloring for the two hubs, paying a cost of 1. We conclude that the competitive ratio of the greedy algorithm is $\Omega(n)$.

The Conservative Algorithm Consider now the algorithm which always prefers the coloring that is closer (i.e, smaller Hamming distance) to the initial coloring. Assume w.l.o.g. that initially, at least half of the nodes are red. Now, we subject it to the following input sequence, which creates a connected component in the form of a chain:

1. Create an edge between two red nodes
2. Do until there are no isolated red nodes:
 - (a) Let v be some red, isolated node.
 - (b) If one of the chain end-nodes u is red, create an edge (u, v) .
 - (c) Otherwise, create an edge between v and any of the chain end-nodes

The chain grows by one node each iteration; in half of the iterations it has an equal number of blue and red nodes. When that happens, the following holds:

1. One of the chain end-nodes is red (since the chain is of even length)
2. Flipping the colors of all of the nodes in the chain does not increase the distance from the initial coloring

Therefore, in every even iteration, a new edge connects two red nodes, to which the algorithm responds by flipping the color of every node in the chain. Overall, the algorithm pays a cost of $\Omega(n^2)$, while an optimal algorithm colors at most half of the nodes exactly once, or a cost of $O(n)$. It follows that the competitive ratio of the conservative algorithm is $\Omega(n)$ as well.

Algorithm 1 Recoloring for bipartite graphs, invoked upon arrival of edge $e_i = (u, v)$

```

1: if  $c_{i-1}(u) \neq c_{i-1}(v)$  then
2:    $c_i := c_{i-1}$ ; return // no recoloring
3: end if // otherwise, u and v belong in different components
4: Let  $S_1, S_2$  be the components of  $u$  and  $v$  in  $G_{i-1}$ , and let  $S$  be their common component in  $G_i$ 
5: Let  $w(S_i)$  be the sum of the weights of the nodes in the set  $S_i$ 
6: Define colorings  $\gamma_1$  and  $\gamma_2$  by flipping the colors of components  $S_1$  and  $S_2$ , respectively, i.e., for all  $z \in V$ :

```

$$\gamma_1(z) = \begin{cases} c_{i-1}(z), & \text{if } z \notin S_1 \\ 3 - c_{i-1}(z), & \text{if } z \in S_1 \end{cases}$$

$$\gamma_2(z) = \begin{cases} c_{i-1}(z), & \text{if } z \notin S_2 \\ 3 - c_{i-1}(z), & \text{if } z \in S_2 \end{cases}$$

```

7: if  $\delta(\gamma_1 \upharpoonright_S) \geq 2\delta(\gamma_2 \upharpoonright_S)$  then
8:    $c_i \leftarrow \gamma_2$ 
9: else if  $\delta(\gamma_2 \upharpoonright_S) \geq 2\delta(\gamma_1 \upharpoonright_S)$  then
10:   $c_i \leftarrow \gamma_1$ 
11: else if  $w(S_1) \geq w(S_2)$  then
12:   $c_i \leftarrow \gamma_2$ 
13: else
14:   $c_i \leftarrow \gamma_1$ 
15: end if

```

3.2 A Competitive Algorithm

While each approach discussed in Sect. 3.1 resulted in $\Omega(n)$ competitive ratio, the following combination of them is $O(\log n)$ -competitive: If one of the two possible coloring is such that at least two-thirds of the weighted nodes will have their original color c_0 (and therefore that coloring has less than half the Hamming distance to the original coloring than the alternative), that alternative is chosen by the algorithm. Otherwise, no coloring has a significantly smaller Hamming distance, and in this case the algorithm prefers the “cheaper” coloring, i.e., the algorithm recolors the component with the smaller weight.

Pseudocode of the algorithm to be executed upon the arrival of a new edge e_i is given in Algorithm 1. For a subset of nodes V' , we use $c \upharpoonright_{V'}$ to denote the restriction of the coloring c to the set of nodes V' . For convenience, we write $\delta(c)$ as a shorthand for $\delta_H(c, c_0)$. We also use $\delta(c \upharpoonright_{V'})$ to refer to $\delta_H(c \upharpoonright_{V'}, c_0 \upharpoonright_{V'})$, i.e. the weighted Hamming distance between the coloring c and the initial coloring c_0 restricted to the set of nodes V' .

We consider the partition of V into connected components. Let \mathcal{P}_i be the set of connected components after the arrival of e_i , and define $\mathcal{P}_0 = \{\{v\} \mid v \in V\}$. Note that in the bipartite case, the color of one node determines the coloring of all nodes in its connected component. Therefore, if e_i is monochromatic upon arrival, it necessarily connects two connected components $S_1, S_2 \in \mathcal{P}_{i-1}$ into a new component S (cf. line 4).

We now analyze the algorithm. Specifically, we prove the following theorem.

Theorem 3.1 *Algorithm 1 is $O(\log n)$ -competitive.*

Fix an input sequence $\{e_i\}_i$ and an optimal coloring c^* , i.e., $\delta(c^*)$ is minimal among all legal colorings of the final graph G . Denote $r := \delta(c^*)$, and let ALG denote the cost of Algorithm 1 on the given input sequence. We prove the theorem by showing that $\text{ALG} \leq O(r \log r)$.

We start by showing that if a component is not colored by the optimum solution and the algorithm in the same way, then the optimum solution recolors a significant part of the component. We introduce some additional terminology.

- Given a connected component S and a coloring c , we say that S is *well colored* by c if $c|_S = c^*|_S$ is consistent with c^* . Otherwise, we say that S is *badly colored*.
- Let $R \subseteq V$ be the subset of nodes v such that $c^*(v) \neq c_0(v)$. By definition, $w(R) = r$.

Proposition 3.2 *For all i and all $S \in \mathcal{P}_i$, if S is badly colored then $w(S) < 3w(S \cap R)$.*

Proof We prove the claim by induction on i , the number of edges inserted. For $i = 0$ we have that \mathcal{P}_0 is a collection of singletons. Let $S = \{v\} \in \mathcal{P}_0$. If S is badly colored then by definitions, $v \in R$, and hence $S \cap R = S$, proving the base case.

Assume now that the claim holds for all components in \mathcal{P}_{i-1} . It suffices to show that if a new component is created by the arrival of edge $e_i = (v_1, v_2)$, the claim holds for that component. So assume that S is created by merging $S_1, S_2 \in \mathcal{P}_{i-1}$, where $v_1 \in S_1$ and $v_2 \in S_2$.

If $c_{i-1}(v_1) \neq c_{i-1}(v_2)$, then $c_i = c_{i-1}$, and thus the component S is badly colored iff the components S_1, S_2 are both badly-colored (note that it cannot be that only one is badly colored, since that would imply the infeasibility of c^*). Using the induction hypothesis, if S is badly colored then

$$w(S) = w(S_1) + w(S_2) < 3(w(S_1 \cap R) + w(S_2 \cap R)) = 3w(S \cap R) ,$$

completing the proof for the case that $c_{i-1}(v_1) \neq c_{i-1}(v_2)$.

Assume henceforth that $c_{i-1}(v_1) = c_{i-1}(v_2)$. When the component S is formed, the algorithm considers the two colorings γ_1 and γ_2 for S , where γ_1 and γ_2 are formed from c_{i-1} by recoloring S_1 and S_2 , respectively. Thus, the colorings of S under γ_1 and γ_2 are negations of one another. One of these colorings, wlog γ_1 , is consistent with c^* , and thus $\delta(\gamma_1|_S) = w(S \cap R)$. Since γ_2 is the negation of γ_1 , it holds that $\delta(\gamma_2|_S) = w(S) - w(S \cap R)$.

If the component S is badly colored, it must be that γ_2 is chosen. For this to happen, it must hold that $\delta(\gamma_2|_S) < 2\delta(\gamma_1|_S)$. Simplifying, we get $w(S) < 3w(S \cap R)$, completing the proof. □

Corollary 3.3 *If a component has weight larger than $3r$, then the component is well-colored.*

Proof By Proposition 3.2 and the fact that for any component S , $w(S \cap R) \leq r$. □

Lemma 3.4 *The weight of nodes recolored by Algorithm 1 throughout the execution is at most $3r$.*

Proof Clearly, the total weight of nodes in R which change color during the algorithm is at most $w(R) = r$. It remains to bound the weight of nodes that change color in $V \setminus R$; denote the set of those nodes by U .

Let $\mathcal{B} \subseteq 2^V$ be the collection of subsets of V which were badly-colored components in the algorithm at some point. We claim that \mathcal{B} is a laminar collection – that is, for every two components $S_1, S_2 \in \mathcal{B}$ it holds that either $S_1 \subseteq S_2, S_2 \subseteq S_1$ or $S_1 \cap S_2 = \emptyset$. Indeed, \mathcal{B} is laminar since it is contained in the collection of all connected components which are formed in the algorithm, which is itself laminar.

Since \mathcal{B} is a laminar collection, there exists a subcollection $\mathcal{B}' \subseteq \mathcal{B}$ of disjoint components such that $\bigcup_{S \in \mathcal{B}} S = \bigcup_{S \in \mathcal{B}'} S$.

Consider any node $v \in U$. The coloring c^* is consistent with c_0 on this node v ; thus, for this node to change color during the algorithm, it must be part of some badly-colored component at some point during the algorithm. This implies that $U \subseteq \bigcup_{S \in \mathcal{B}} S = \bigcup_{S \in \mathcal{B}'} S$.

$$\begin{aligned} w(U) &= \sum_{S \in \mathcal{B}'} w(U \cap S) \leq \sum_{S \in \mathcal{B}'} w((V \setminus R) \cap S) \\ &= \sum_{S \in \mathcal{B}'} w(S \setminus R) \\ &\leq 2 \sum_{S \in \mathcal{B}'} w(S \cap R) && \text{by Proposition 3.2} \\ &\leq 2w(R) = 2r. \end{aligned}$$

□

Next, we define a potential function ϕ such that $\phi(i) = 6\delta(c_i)$. Note that $\phi(0) = 0$, and that ϕ is nonnegative. Following the conventional notation, denote $\Delta\phi_i = \phi(i) - \phi(i - 1)$ for every $i > 0$.

For every $i \in \mathbb{N}$, let ALG_i be the cost incurred by the algorithm in step i , i.e., $\delta_H(c_i, c_{i-1})$. Define

$$I^+ \stackrel{\text{def}}{=} \{i \mid \text{ALG}_i + \Delta\phi_i > 0\}.$$

Then we have

$$\begin{aligned} \text{ALG} &= \sum_i \text{ALG}_i \leq \sum_i (\text{ALG}_i + \Delta\phi_i) \\ &\leq \sum_{i \in I^+} (\text{ALG}_i + \Delta\phi_i) \leq \sum_{i \in I^+} 7\text{ALG}_i, \end{aligned} \tag{1}$$

where the last inequality uses the fact that changing the coloring of nodes of total weight s increases the coloring’s weighted Hamming distance from c_0 by at most s , and thus increases ϕ by at most $6s$.

Proposition 3.5 *If $v \in V$ is recolored in step $i \in I^+$, then the weight of the connected component of v grows by a factor of at least $\frac{5}{4}$ in step i . That is, denoting by S, S' the old and new components of v respectively, it holds that $w(S') \geq \frac{5}{4}w(S)$.*

Proof Node v changes color when some components $S_1, S_2 \in \mathcal{P}_{i-1}$ are merged to form $S \in \mathcal{P}_i$, and one of these components, wlog S_2 , changes color, where $v \in S_2$. We now must show that $w(S) \geq \frac{5}{4}w(S_2)$. Denote by γ_1 and γ_2 the colorings formed from c_{i-1} by recoloring S_1 and S_2 respectively; since S_2 is recolored, the algorithm chose γ_2 .

If γ_2 is chosen in line 12 of Algorithm 1, then trivially $w(S) \geq 2w(S_2) \geq \frac{5}{4}w(S_2)$ and the proof is complete.

Otherwise, the algorithm chose γ_2 in line 8 because $\delta(\gamma_1|_S) \geq 2\delta(\gamma_2|_S)$. Let us denote $s_1 = w(S_1), s_2 = w(S_2), x_1 = \delta(c_{i-1}|_{S_1})$ and $x_2 = \delta(c_{i-1}|_{S_2})$. It holds that $\delta(\gamma_1|_S) = s_1 - x_1 + x_2$ and $\delta(\gamma_2|_S) = s_2 - x_2 + x_1$. Thus,

$$s_1 + x_2 \geq s_1 - x_1 + x_2 \geq 2(s_2 - x_2 + x_1) \geq 2(s_2 - x_2),$$

and therefore $s_1 \geq 2s_2 - 3x_2$. Adding s_2 to both sides, we have that

$$s_1 + s_2 \geq 3(s_2 - x_2). \tag{2}$$

Now, recall that $i \in I^+$, and thus

$$0 \leq \text{ALG}_i + \Delta\phi_i = s_2 + 6((s_2 - x_2) - x_2) = 7s_2 - 12x_2. \tag{3}$$

Eq. (3) implies that $x_2 \leq \frac{7}{12}s_2$. Plugging into (2), we get $w(S) = s_1 + s_2 \geq 3s_2 - 3x_2 \geq 3s_2 - \frac{7}{4}s_2 = \frac{5}{4}s_2 = \frac{5}{4}w(S_2)$ which completes the proof of the proposition. \square

Proof of Theorem 3.1 By Eq. (1), it is enough to bound $7 \sum_{i \in I^+} \text{ALG}_i$. Consider any step i in which some component $S \in \mathcal{P}_{i-1}$ is recolored, upon being connected to a second component in \mathcal{P}_{i-1} . Consider the subset $W \stackrel{\text{def}}{=} \left\{ u \in S \mid w(u) < \frac{w(S)}{2n} \right\}$. It holds that $w(W) \leq n \cdot \frac{w(S)}{2n} = \frac{w(S)}{2}$, and thus $w(S) \leq 2w(S \setminus W)$. Therefore,

$$\text{ALG}_i = w(S) \leq 2w\left(\left\{ u \in S \mid w(u) \geq \frac{w(S)}{2n} \right\}\right). \tag{4}$$

Plugging Eq. (4) into Eq. (1), and denoting by $S_i \in \mathcal{P}_{i-1}$ the component that was recolored in step i , it thus remains only to bound the term

$$14 \sum_{i \in I^+} w\left(\left\{ u \in S_i \mid w(u) \geq \frac{w(S_i)}{2n} \right\}\right). \tag{5}$$

Using Lemma 3.4, and denoting by U the set of nodes recolored by the algorithm, we know that $w(U) \leq 3r$. Consider a node $u \in U$: each time the node u is recolored in a step from I^+ , the weight of the component containing u grows by a factor of

at least $\frac{5}{4}$ (by Proposition 3.5). Thus, each node can be recolored in steps from I^+ at most $O(\log n)$ times before the weight of u 's component exceeds $2n \cdot w(u)$. This implies that the term in Eq. (5) can be bounded by $O(\log n) \cdot w(U) = O(\log n) \cdot r$, and thus $\text{ALG} \leq O(\log n) \cdot r$, completing the proof. \square

3.3 A Lower Bound

We present a general lower bound in the unweighted case. We note that a similar argument is used in [14] by Kashyop et al., where it is shown in the context of data structures that any deterministic algorithm that maintains a 2-coloring must perform, in the worst case, t edge insertion require $\Omega(t \log t)$ recolorings.

Theorem 3.6 *The competitive ratio of any randomized algorithm for recoloring bipartite graphs is $\Omega(\log(n))$.*

Let $A(I)$ be the cost of algorithm A on an instance (input sequence) I . According to Yao's principle, it suffices to show an instance distribution D such that the expected cost of every deterministic algorithm A is $E[A(I)] = \Omega(\log n)E[\text{OPT}(I)]$, where the expectation is for instances chosen by D . In the proof below we construct such a distribution.

Proof Assume w.l.o.g. that n is a power of 2 (otherwise, we work only with $2^{\lceil \log n \rceil}$ nodes, leaving the others isolated). We construct an input sequence in $\log n$ phases, while maintaining the following invariant:

After each phase h : (i) there are $n/2^h$ connected components, and (ii) each connected component is a chain of 2^h nodes.

Clearly, the invariant holds before the execution begins, when there are n isolated nodes. Note that we allow for any initial coloring.

In each phase h , for $h = 1, \dots, \log n$, we connect segments in pairs by introducing $n/2^h$ edges: each edge connects *random endpoints* of two segments (this is the only randomization we use). This obviously maintains the invariant after phase h is completed.

Consider phase h for $h > 1$. The merging segments are of even length 2^{h-1} each, and hence each segment has exactly one endpoint of each color. It follows that by construction, each new edge is monochromatic with probability $1/2$. Therefore the expected cost incurred in phase $h > 1$ for any deterministic algorithm is

$$\frac{n}{2^h} \cdot \frac{1}{2} \cdot 2^{h-1} = \frac{n}{4},$$

because in phase h there are $n/2^h$ merges, and each merge has expected cost $\frac{1}{2} \cdot 2^{h-1}$. In summary, the cost of any deterministic algorithm on a random instance defined as above, over all phases, is at least $(\log n - 1) \cdot n/4 = \Theta(n \log n)$ (the cost of phase 1 depends on the initial coloring). On the other hand, the optimal cost for any n -node graph is never more than n : every node needs to be colored at most once, according to the final graph. The result follows. \square

4 ($\Delta+1$) Disengagement

In this section we consider $(\Delta + 1)$ -coloring, where Δ is an upper bound on the number of neighbors a node may have. We show that in this case, the competitive ratio of deterministic disengagement is $\Theta(\Delta)$ and that the randomized competitive ratio is $\Theta(\log \Delta)$.

4.1 Algorithms

We now present our algorithms for the $\Delta + 1$ disengagement problem. Algorithm 2 is used in both the deterministic and randomized versions: The only difference is the implementation of the “recolor” subroutine it invokes (Algorithms 3 and 4). To see how the algorithm works, let us define an auxiliary graph $G'_i = (V, E'_i)$ as follows:

$$(u, v) \in E'_i \iff (u, v) \in E_i \text{ and } c_0(u) = c_0(v) ,$$

i.e., the auxiliary graph G'_i includes only edges that connect nodes with the same initial color. The underlying idea of Algorithm 2 is to maintain a small-weight vertex cover of G' , denoted C , and apply recoloring only to nodes in C . (It turns out that there is no need to remember the initial coloring.) For a deterministic algorithm, we recolor using the procedure given in Algorithm 3: change the node’s color to the first available color, where a color is said to be available if it is not taken by any neighbor. For a randomized algorithm, we choose uniformly at random among the available colors (Algorithm 4).

To maintain a light vertex cover, we use the classical 2-approximation algorithm of Bar-Yehuda and Even [4] (we can use any other algorithm which processes edges one at a time). Whenever an uncovered edge is considered, the residual weights of its endpoints are reduced by the same amount so that one of them reaches zero. The node with residual weight 0 (possibly both) is then added to C .

Theorem 4.1 *Algorithms 2 with Algorithm 3 is a deterministic algorithm for $\Delta + 1$ disengagement with competitive ratio $O(\Delta)$. Algorithm 2 with Algorithm 4 is a randomized algorithm for $\Delta + 1$ disengagement with expected competitive ratio $O(\log \Delta)$.*

We first analyze the general framework of Algorithm 2.

Lemma 4.2 *After every step, C is a vertex cover of G' . Moreover, $w(C)$ is at most twice the weight of any vertex cover of G' .*

Proof We first argue that C is a vertex cover of G' . Let $e_i = (u, v) \in E'$. We consider two cases. If e_i is monochromatic upon arrival, then Algorithm 2 makes sure that if none of its endpoints are in C , then at least one of them enters C (lines 8–18), so e_i is covered by C in this case. If $c_i(u) \neq c_0(u)$, then node u was necessarily recolored at some point in the past. However, since Algorithm 2 recolors only nodes in C , we must have $u \in C$, so e_i is covered by C in this case too. Finally, the approximation bound of $w(C)$ follows from [4]. \square

Algorithm 2 $(\Delta + 1)$ -disengagement: Upon arrival of edge (u, v) in step i

```

1: if  $c_{i-1}(u) \neq c_{i-1}(v)$  then return //  $c_i = c_{i-1}$ , no recoloring
2: else if  $|\{u, v\} \cap C| = 1$  then
3:   let  $x \in \{u, v\}$  be the node in  $C$ 
4:   recolor( $x$ )
5: else if  $|\{u, v\} \cap C| = 2$  then
6:   let  $x \in \{u, v\}$  be the node with the higher degree // break ties arbitrarily
7:   recolor( $x$ )
8: else if  $\{u, v\} \cap C = \emptyset$  then
9:    $w_i(u) \leftarrow w_{i-1}(u) - \min(w_{i-1}(u), w_{i-1}(v))$  //  $w_0(v) = w(v)$  for all  $v \in V$ 
10:   $w_i(v) \leftarrow w_{i-1}(v) - \min(w_{i-1}(u), w_{i-1}(v))$ 
11:  if  $w_i(u) = 0$  then
12:     $C \leftarrow C \cup \{u\}; x \leftarrow u$ 
13:  end if
14:  if  $w_i(v) = 0$  then
15:     $C \leftarrow C \cup \{v\}; x \leftarrow v$ 
16:  end if
17:  recolor( $x$ )
18: end if

```

Lemma 4.3 Every algorithm pays a cost of at least $w(C)/2$.

Proof Consider any solution to the given input. The edges in E' are monochromatic by the initial coloring, so every disengagement algorithm has to recolor at least one endpoint of every edge in E' at least once. Therefore, the set of nodes recolored by any solution is a vertex cover of G' . The result follows now from Lemma 4.2. \square

We now consider the way recoloring is done. First, the deterministic version.

Lemma 4.4 Algorithm 2 with Algorithm 3 pays at most $\Delta \cdot w(C)$.

Proof By the code, Algorithm 2 recolors only nodes in C . Observe that a node may be recolored only when a new incident edge is introduced. Since the maximum degree of a node is Δ , the result follows. \square

If we use randomized recoloring, we have the following. Let H_Δ denote the Δ^{th} harmonic number, i.e., $H_\Delta = 1 + \frac{1}{2} + \dots + \frac{1}{\Delta}$.

Lemma 4.5 The expected cost of Algorithm 2 with Algorithm 4 is at most $H_\Delta \cdot w(C)$

Algorithm 3 recolor(u): recolor node u with the first available color

```

1: Let  $S = \{c(v) \mid (u, v) \in E\}$ 
2:  $c \leftarrow \min(\{1, \dots, \Delta + 1\} \setminus S)$ 
3:  $c(u) \leftarrow c$ 

```

Algorithm 4 recolor(u): randomly recolor node u with an available color

```

1: Let  $S = \{c(v) \mid (u, v) \in E\}$ 
2: Let  $c \in_R (\{1, \dots, \Delta + 1\} \setminus S)$  // choose randomly
3:  $c(u) \leftarrow c$ 

```

Proposition 4.6 Consider any two distinct nonadjacent nodes $u, v \in C$. At any iteration, it holds that

$$\Pr(c(u) = c(v)) \leq \max \left\{ \frac{1}{\Delta + 1 - \deg(u)}, \frac{1}{\Delta + 1 - \deg(v)} \right\}.$$

Proof Let $w \in \{u, v\}$ be the node that changed its color last (since both $u, v \in C$, they both changed colors so w is always defined). It holds that:

$$\begin{aligned} \Pr(c(u) = c(v)) &= \Pr(w = v) \cdot \Pr(c(u) = c(v) \mid w = v) \\ &\quad + \Pr(w = u) \cdot \Pr(c(u) = c(v) \mid w = u). \end{aligned}$$

Let A_v denote the event that $w = v$, and assume that A_v holds. Let t' denote the last iteration in which v changed its color, and let $\deg'(v)$ be its degree at time t' . Then v chose its color uniformly at random from a set of available colors S whose size is at least

$$(\Delta + 1 - \deg'(v)) \geq (\Delta + 1 - \deg(v)) \geq \min\{\Delta + 1 - \deg(u), \Delta + 1 - \deg(v)\}.$$

Thus:

$$\begin{aligned} \Pr(c_u = c_v \mid A_v) &= \sum_{S' \subseteq [\Delta+1]} \sum_{s \in S'} \Pr(S = S' \mid A_v) \cdot \Pr(c_u = c_v = s \mid S = S', A_v) \\ &\leq \sum_{S' \subseteq [\Delta+1]} \sum_{s \in S'} \Pr(S = S' \mid A_v) \cdot \Pr(c_u = s \mid S = S', A_v) \cdot \underbrace{\Pr(c_v = s \mid c_u = s, S = S', A_v)}_{=1/|S'|} \\ &= \sum_{S' \subseteq [\Delta+1]} \Pr(S = S' \mid A_v) \cdot \frac{1}{|S'|} \cdot \underbrace{\left(\sum_{s \in S'} \Pr(c_u = s \mid S = S', A_v) \right)}_{\leq 1} \\ &\leq \sum_{S' \subseteq [\Delta+1]} \Pr(S = S' \mid A_v) \cdot \max \left\{ \frac{1}{\Delta + 1 - \deg(u)}, \frac{1}{\Delta + 1 - \deg(v)} \right\} \\ &= \max \left\{ \frac{1}{\Delta + 1 - \deg(u)}, \frac{1}{\Delta + 1 - \deg(v)} \right\}. \end{aligned}$$

□

Proposition 4.7 Consider any two nonadjacent nodes $v \in C$ and $u \notin C$. At any iteration, it holds that

$$\Pr(c(u) = c(v)) \leq \frac{1}{\Delta + 1 - \deg(v)}.$$

Proof The proof is similar to that of Proposition 4.6. Observe the last iteration in which v had changed its color, and denote its degree at that iteration as $\deg'(v)$.

At that time, v chose a random color from a set S of available colors, such that $|S| \geq \Delta + 1 - \deg'(v) \geq \Delta + 1 - \deg(v)$. Additionally, since $u \notin C$, it has its initial coloring $c(u) = c_0(u)$. Therefore:

$$\begin{aligned} \Pr(c_u = c_v) &\leq \sum_{S' \subseteq [\Delta+1]} \Pr(S = S') \cdot \Pr(c_v = c_u \mid S = S') \\ &\leq \sum_{S' \subseteq [\Delta+1]} \Pr(S = S') \cdot \frac{1}{|S'|} \\ &\leq \frac{1}{\Delta + 1 - \deg(v)} \sum_{S' \subseteq [\Delta+1]} \Pr(S = S') = \frac{1}{\Delta + 1 - \deg(v)}, \end{aligned}$$

where the second inequality is due to the fact that $\Pr(c_v = c_u \mid S = S')$ equals 0 if $c(u) \notin S'$ and equals $\frac{1}{|S'|}$ otherwise. □

Proof of Lemma 4.5 Fix the input sequence. Let X be a random variable representing the overall cost of running the algorithm, and let X^v be a random variable representing the cost incurred by recoloring a given node $v \in V$. We bound $E[X^v]$.

Since Algorithm 2 recolors only the nodes in C , consider $v \in C$. Let $e_1^v, \dots, e_{\deg(v)}^v$ be the subsequence of input edges incident on v . Let X_j^v denote the expected cost of recoloring v due to edge e_j^v . We bound $E[X_j^v]$ as follows. Suppose that v is recolored for the first time when e_{j_0} is input (if none exists, we are trivially done). Then for $j = j_0$ we have $E[X_{j_0}^v] = w(v)$. For $j > j_0$, consider the arrival of $e_j^v = (v, u)$.

- If $u \notin C$, then v changes its color w.p $\leq \frac{1}{\Delta+1-j}$ (according to Proposition 4.7)
- If $u \in C$ and $j < \deg(u)$, then v does not change its color
- If $u \in C$ and $j \geq \deg(u)$, then v changes its color w.p $\leq \frac{1}{\Delta+1-j}$ (according to Proposition 4.6)

Therefore, the expected cost incurred by v due to e_j^v is at most $\frac{w(v)}{\Delta+1-j}$.

It follows that the expected cost incurred by the recoloring of node $v \in C$ is at most

$$\begin{aligned} E[X^v] &= E \left[\sum_{j=1}^{\deg(v)} X_j^v \right] \leq w(v) + \sum_{j=j_0+1}^{\deg(v)} E[X_j^v] \leq w(v) + \sum_{j=2}^{\deg(v)} \frac{w(v)}{\Delta + 1 - j} \\ &\leq w(v) \cdot H_{\deg(v)}. \end{aligned}$$

We can therefore summarize that the expected cost due to all nodes is at most

$$\begin{aligned} E[X] &= E \left[\sum_{v \in V} X^v \right] = E \left[\sum_{v \in C} X^v \right] = \sum_{v \in C} E[X^v] \leq \sum_{v \in C} w(v) \cdot H_{\deg(v)} \\ &\leq w(C) \cdot H_{\Delta}. \end{aligned}$$

□

Proof of Theorem 4.1 Follows from the lower bound on the optimum cost of Lemma 4.3, and from the upper bounds on the cost of the deterministic algorithm (Lemma 4.4), and on the expected cost of the randomized algorithm (Lemma 4.5). \square

We note that our deterministic algorithm does not require knowledge of Δ , but the randomized one does.

4.2 Lower Bounds for $(\Delta + 1)$ -Recoloring

We now state lower bounds on the competitive ratios of deterministic and randomized algorithms for $(\Delta + 1)$ -recoloring. We prove the lower bounds in unweighted instances, i.e., the weight of each node is one. We start with a deterministic lower bound.

Theorem 4.8 *The competitive ratio of any deterministic algorithm solving $(\Delta + 1)$ -recoloring is $\Omega(\Delta)$, for any initial coloring.*

We need the following lemma.

Lemma 4.9 *The optimal algorithm’s cost after the i -th step is at most the number of non-isolated nodes in $G_i = (V, E_i)$.*

Proof Since edges are constantly added to the graph G_i , any coloring that is proper for G_i is also proper for G_j , where $j \leq i$. The optimal offline algorithm can color G_i greedily using $\Delta + 1$ colors, and use this coloring until the i -th step. Since every node is recolored at most one time, and the optimal algorithm might only recolor the non-isolated nodes, its cost is at most the number of such nodes. \square

Proof of Theorem 4.8 Let a deterministic algorithm be given. We construct an input sequence in phases as follows. First we choose a set of some $\Delta + 2$ nodes, denoted V' . Note that since $|V'| > \Delta + 1$, according to the pigeonhole principle, there must be at least two nodes $u, v \in V'$ with the same color. We then proceed in phases, where each phase is described as follows.

1. Do until $\exists v \in V'$ with $\deg(v) = \Delta$:
 - (a) Select two nodes $u_1, u_2 \in V'$ s.t. $c(u_1) = c(u_2)$
 - (b) Add edge (u_1, u_2)
2. Let $V_\Delta = \{v \in V' \mid \deg(v) = \Delta\}$. (Note that $1 \leq |V_\Delta| \leq 2$.)
3. Let $V_0 \subseteq V$ be a set of $|V_\Delta|$ isolated nodes
4. $V' \leftarrow V_0 \cup V' \setminus V_\Delta$

Note that the number of phases can be as large as we wish, since it is bounded only by the number of nodes, regardless of k and Δ . Consider the cost of the online algorithm. Every new edge is monochromatic, so it pays a cost of 1 for each new edge. After s phases, at least s nodes have left V' , each with a degree of Δ . It follows that the cost of the online algorithm for s phases is at least $s \cdot \Delta/2$ (since every edge could be counted twice).

On the other hand, by Lemma 4.9, the optimal cost is no more than the number of non-isolated nodes. This number is bounded by the number of nodes that ever were in

V' , and hence, in s phases, it is at most $2s + \Delta + 2$: at most $2s$ nodes were removed from V' , and at most $\Delta + 2$ nodes are in V' after s phases. It follows that for $s \gg \Delta$ phases, the competitive ratio of any deterministic algorithm is $\Omega(\Delta)$. \square

Intuitively, in the $\Omega(\Delta)$ lower-bound proof, both Online and Offline always have to pay (a cost of at least 1) when the new edge creates a new connected component (of size 2), since it involves two nodes that still have their initial coloring. However, the Offline algorithm does not have to pay when the edge is incident on a node that Offline had already changed. We now present a lower bound for randomized algorithms.

Theorem 4.10 *The competitive ratio of any randomized algorithm for $(\Delta + 1)$ -coloring is $\Omega(\log(\Delta))$, assuming that in the initial coloring there are at least two nodes of each color.*

Proof Let $A(I)$ be the cost of algorithm A on an instance (input sequence) I . According to Yao's principle, if there is an instance distribution, for which the expected cost of every deterministic algorithm A is $E[A(I)] \geq L$, then for every randomized algorithm there exists an instance I such that the algorithm's expected cost on I is at least L . We construct a distribution over input sequences as follows. First we pick a random permutation $\sigma : [\Delta + 1] \rightarrow [\Delta + 1]$ of the colors. We then introduce the edges in phases as follows. In the first phase we pick two nodes $v_0, v_1 \in V$ of color $\sigma(1)$ and connect them with an edge. In each subsequent phase $i, i \geq 2$, we pick a node v_i of color $\sigma(i)$ and connect it to all nodes v_0, \dots, v_{i-1} . Clearly, after phase i , the input edges constitute a clique over v_0, v_1, \dots, v_i . After phase Δ we stop (we can repeat the construction with a fresh set of $\Delta + 1$ isolated nodes).

For the cost analysis, let C_i denote the set of colors used by nodes v_0, \dots, v_i after phase i . Clearly $|C_i| = i + 1$. In particular, for any deterministic algorithm, there exists at least one color $c_i^* \in C_i \setminus \{\sigma(1), \dots, \sigma(i)\}$. Consider now the node v_{i+1} , added in phase $i + 1$. Its color is $\sigma(i + 1)$, which is uniformly distributed over $[\Delta + 1] \setminus \{\sigma(1), \dots, \sigma(i)\}$, and hence $\Pr[\sigma(i + 1) = c_i^*] = \frac{1}{\Delta + 1 - i}$. It follows that the expected number of monochromatic edges (which is the expected cost of the deterministic algorithm) in phase $i + 1$ is at least $\frac{1}{\Delta + 1 - i}$. Also by construction, the number of monochromatic edges in the first phase is 1. Therefore, the total expected cost across all phases is

$$E[\text{cost}] \geq 1 + \sum_{i=1}^{\Delta-1} \frac{1}{\Delta + 1 - i} = H_{\Delta} = O(\log \Delta),$$

where H_n denotes the n th harmonic sum. On the other hand, an optimal algorithm would pay a cost of 1 by recoloring only a single node in the first iteration with the single color that is never used, namely color $\sigma(\Delta + 1)$. \square

5 General Lower Bounds for k Colors

By combining the techniques of Sects. 3.3 and 4.2, we can prove stronger lower bounds on the competitiveness of recoloring algorithms. We note that while such coloring is

NP-hard in general, the lower bound presented in Sect. 5.1 holds for a particular family of easily-colorable graphs.

In Sect. 5.2 we briefly discuss the *capacitated* case, in which each machine (i.e., color) has a limited capacity, and provide a lower bound for the tight case, wherein the total (color) capacity equals the total (node) weight.

5.1 General Lower Bound

Theorem 5.1 *The competitive ratio of recoloring n nodes with k colors is $\Omega(k \log(n/k))$ for deterministic algorithms and $\Omega(\log k \cdot \log(n/k))$ for randomized algorithms.*

Proof We use a single framework, with unweighted graphs, for both the deterministic and the randomized case. We assume that $n \geq 3k$. No generality is lost: If $n < 3k$, the claimed bounds follow from Theorems 4.8 and 4.10, by using only k nodes and leaving the others isolated (the case $k > n$ is equivalent to $k = n$).

Assume further that $n/k = 2^M$ for some integer M (otherwise, define $n' = k \cdot 2^{\lfloor \log(n/k) \rfloor}$ and use n' instead of n).

In the construction, we use k -cliques, i.e., fully-connected graphs of k nodes. We say that two k -cliques K_1, K_2 are *aligned* in edge set E if for every node $u_1 \in K_1$, there is exactly one node $u_2 \in K_2$ such that $(u_1, u_2) \notin E$. Such nodes are called *partners*. We use the term *K -chain of length d* to refer to a sequence of d aligned k -cliques. Given a node u in a k -chain, $\text{partner}(u)$ denotes the set of nodes that are partners of u , including u . We note the following immediate fact for any k -chain Z .

- Any proper coloring c of Z satisfies, for all nodes $u, v \in Z$: if $v \in \text{partner}(u)$ then $c(u) = c(v)$.

We now describe the construction in phases. First there is a setup phase (phase 0), in which $\frac{n}{k} \binom{k}{2} = n(k-1)/2$ edges are introduced to create n/k disjoint k -cliques. Next we run $M = \log(n/k)$ phases, such that the following invariant holds. □

Invariant 5.2 *After each phase $m = 0, \dots, M$, there are $(n/k) \cdot 2^{-m}$ k -chains of length 2^m .*

To maintain the invariant, in each phase $m \geq 1$ we group the k -chains in pairs and “stitch” each pair of k -chains to obtain k -chains of double length: we do that by selecting an extreme k -clique from each k -chain and introducing $k(k-1)$ edges, that connect each node in these k -cliques to $k-1$ nodes of the other k -clique.

More specifically, let u_1^1, \dots, u_k^1 denote the nodes of a k -clique which is the end clique in one of the k -chains, and let u_1^2, \dots, u_k^2 denote the nodes of a clique at the endpoint of the other k -chain. The goal of stitching is to make these cliques aligned while making any online algorithm pay as much as possible. The way stitching is implemented, described next, is the only difference between the deterministic and the randomized cases.

Deterministic Stitching. We proceed node by node. We connect u_1^1 to a node $u_{j_1}^2$ such that $c(u_1^1) = c(u_{j_1}^2)$. As a result, at least one of these nodes (and all its partners) changes its color. Since the algorithm is deterministic, the new coloring c' can be

computed. We then connect u_1^1 to a node $u_{j_2}^2$ such that $c'(u_1^1) = c'(u_{j_2}^2)$. We continue this way, connecting u_1^1 to $k - 1$ nodes. We then proceed to u_2^1 , again connecting it to a node of the same color in each step. However, for u_2^1 we do that only $k - 2$ times, since the color of u_1^1 and its partner will never appear in the sequence. We complete the edges of u_2^1 by connecting it to the partner of u_1^1 . In general, for node u_j^1 , we introduce $k - j$ edges that connect it to nodes of the same color (at the time the edges arrive), and then edges that connect u_j^1 to the partners of u_1^1, \dots, u_{j-1}^1 .

Randomized Stitching. This case is simpler to describe: For each node u_j^1 in the first k -clique we pick a random permutation $\sigma_j : [k] \rightarrow [k]$, and we introduce the following $k - 1$ edges:

$$\left(u_j^1, u_{\sigma_j(1)}^2\right), \left(u_j^1, u_{\sigma_j(2)}^2\right), \dots, \left(u_j^1, u_{\sigma_j(k-1)}^2\right).$$

This concludes the description of randomized stitching.

In both cases of stitching, the final graph is a k -chain of length n/k . Note that this graph is easily colorable using k colors: pick any k -clique K , assign the k colors to the nodes of K arbitrarily in a 1-1 fashion, and then color each remaining node by the color of its partner in K . Clearly, this can be done off-line at cost at most n .

In Lemma 5.3 below we show that for deterministic algorithms, the cost of every phase $m = 1, \dots, M$ is at least $n(k - 1)/4$, for a total cost of at least $Mn(k - 1)/4 = \Omega(nk \log(n/k))$. In Lemma 5.4 below we show that for randomized algorithms, the expected cost of every phase $m = 1, \dots, M$ is at least $\frac{1}{4}nH_{k/2}$, for total expected cost of at least $\frac{1}{4}MnH_{k/2} = \Omega(n \log k \log(n/k))$. Since the optimal cost is at most n , the result follows.

Lemma 5.3 *Given any deterministic algorithm, the cost of every phase $m = 1, \dots, M$ is at least $n(k - 1)/4$.*

Proof By construction, when introducing the edges for node u_j^1 , the first $k - j$ edges are monochromatic, and cause recoloring. Since all partners are recolored, and since by Invariant 5.2 in phase m there are 2^{m-1} aligned cliques in each k -chain, we have that the cost incurred by the edges of node u_j^1 is $(k - j)2^{m-1}$, hence the total cost due to a single stitching in phase m is $\sum_{j=1}^k (k - j)2^{m-1} = k(k - 1)2^{m-2}$. Since by Invariant 5.2 there are $(n/k)2^{-m}$ pairs to stitch in phase m , the lemma follows. \square

Lemma 5.4 *Given any randomized algorithm, the expected cost of every phase $m = 1, \dots, M$ is at least $\frac{1}{4}nH_{k/2}$.*

Proof We use Yao’s principle. The distribution of the input instances is defined in the specification of the phases. It therefore suffices to bound from below the expected cost of phase m for any deterministic algorithm. Consider the edges added for node u_j^1 , and let us focus on edge $(u_j^1, u_{\sigma_j(l)}^2)$, for some $l \in [1, k - j]$. The color of one of its endpoints is deterministic, given $\sigma_j(1), \dots, \sigma_j(l - 1)$, and the other is uniformly random from $[k] \setminus \{\sigma_j(1), \dots, \sigma_j(l - 1)\}$. Therefore, the probability that $(u_j^1, u_{\sigma_j(l)}^2)$ is monochromatic is $\frac{1}{k-l+1}$. Since recoloring a node implies recoloring all its 2^{m-1}

partners, we have that the total expected cost due to the edges incident on u_j^1 in phase m is at least $\sum_{l=1}^{k-j} \frac{1}{k-l+1} 2^{m-1} = (H_k - H_j)2^{m-1}$. Summing over all nodes, we get that the total expected cost of stitching a pair of chains in phase m is at least

$$\sum_{j=1}^k 2^{m-1} (H_k - H_j) \geq 2^{m-1} \sum_{j=1}^{\lceil k/2 \rceil} (H_k - H_j) \geq 2^{m-1} \cdot \frac{k}{2} \cdot H_{k/2},$$

where H_x here means $H_{\lfloor x \rfloor}$. Since there are $(n/k)2^{-m}$ pairs of chains to connect in phase m , the result follows. □

5.2 Lower Bounds for the Capacitated Case

In the capacitated disengagement variant of the problem, we model the case where machines have limited capacity. This constraint is formalized as follows. We assume that for each color $i \in [k]$ we are given a *color capacity* $W_i \in \mathcal{N}$, and we require that in any coloring output by the recoloring algorithm, the total weight of nodes colored i is at most W_i . We assume that the sum of all color capacities is at least the sum of all node weights. The case where the total capacity equals the total weight is called *tight*.² Note that if the instance is tight, then a node cannot be recolored alone: recoloring must be done by swapping colors with other nodes.

We prove the following results.

Theorem 5.5 *The competitive ratio of deterministic algorithms for tight capacitated recoloring is $\Omega(n)$, even if nodes are unweighted.*

Theorem 5.6 *The competitive ratio of randomized algorithms for tight capacitated recoloring is $\Omega(\log n)$, even if nodes are unweighted.*

We use the same framework for both the deterministic and the randomized cases. There are $n = kW$ nodes of weight 1 each, and the capacity of each of the k colors is W .

The adversary constructs the following input sequence, which selects a set of $W - 1$ nodes (denoted A') with the same initial color, and creates edges between them and all of the other nodes, except one:

1. $A' \leftarrow \{u \mid c_0(u) = 1\}$
2. Let a_0 be some node in A'
3. $A' \leftarrow A' \setminus \{a_0\}$
4. $V' \leftarrow V \setminus A'$
5. Do $(n - W)$ times
 - 51 select a node $v \in V'$ (see below)
 - 52 create the edges $A' \times \{v\}$
 - 53 $V' \leftarrow V' \setminus \{v\}$

² In the case of weighted nodes, an algorithm for tight capacitated disengagement needs to solve the NP-hard problem of PARTITION [9] even when $k = 2$. Our lower bounds, however, apply to the unweighted nodes case, for which PARTITION is trivial.

The difference between the deterministic and the randomized cases is the way the selection of step 51 is implemented. We now proceed to present of the remainder of the proofs.

Proof of Theorem 5.5 For deterministic algorithms, the node $v \in V'$ selected in step (51) is such that its color is the color of some node in A' . This is always possible since there are exactly W nodes in each color, and the set A' is of size $(W - 1)$ (the first node that fulfills it is a_0). Hence, any online deterministic algorithm pays a cost of at least 2 in each iteration of the loop of step (5), for a total cost of at least $2(n - W)$ for all iterations.

After step (5) is repeated $(n - W)$ times, only one node $v' \notin A'$ may have the same color as *any* node in A' . Therefore, any valid coloring must have all nodes in A' as well as v' in the same color. The optimal algorithm swaps the colors of node a_0 and node v' , paying a cost of 2 for the entire sequence. The competitive ratio of any online deterministic algorithm for the capacitated case is $\Omega(n)$. \square

Proof of Theorem 5.6 Again we use Yao's principle. We specify an input distribution and prove a lower bound for the expected cost of any deterministic algorithm. The distribution is defined by implementing step (51) using a random selection: v is selected uniformly at random from V' .

As before, there must be at least one node in V' (but not in A') that is colored the same as some node in A' , and therefore the expected cost of any deterministic algorithm in step (5) is at least $2 \cdot \frac{1}{|V'|}$. The size of V' starts at $(n - W)$, and decreases by 1 each loop until it reaches 1. Therefore we can bound from below the total expected cost for any deterministic algorithm:

$$E_{cost} \geq 2 \cdot \left(1 + \sum_{|V'|=2}^{(n-W)} \frac{1}{|V'|} \right) = 2 \cdot H_{n-W}$$

Note that $H_{n-W} = H_{n-n/k} \geq H_{n/2}$ for $k > 1$. As the optimal algorithm pays a cost of 2, we get that the competitive ratio of any randomized algorithm in the capacitated case is $\Omega(\log(n))$. \square

6 Fully Dynamic Disengagement

6.1 Models of Fully-Dynamic Disengagement

In previous sections, we considered the partially-dynamic variant of the disengagement problem, in which every new edge is an additional, permanent constraint. In this section, we consider the fully-dynamic variant of the online disengagement problem (abbreviated FD below), where at every iteration i , the edge constraints may also be deleted (in FD, input sequences may be of unbounded length). It turns out that FD is more difficult than partially-dynamic disengagement. In this section we first discuss various plausible models of FD and show their equivalence, and then show that even a very simple dynamic instance forces an $\Omega(n)$ lower bound on the competitiveness

of deterministic algorithms. The latter result is obtained by reduction from Metrical Task Systems.

There are several (yet equivalent) ways to define the fully-dynamic Disengagement problem:

The single edge model (SE): At iteration i upon the arrival of edge $e_i = (u, v)$, the algorithm must make sure that $c_i(u) \neq c_i(v)$, i.e. only the coloring of the last edge must be maintained.

Expiration time model (ET): Every edge arrives with a prescribed expiration time - after which it is deleted.

Edge insertions and deletions model (InD): In this model, at every iteration either an existing edge in the graph is deleted, or a non-existing one appears. The algorithm is required at every iteration to maintain a proper coloring of the graph.

We first argue that the models are essentially equivalent.

Theorem 6.1 *A lower bound on the competitive ratio of (randomized) algorithms under one of the three models of dynamic disengagement, holds under the other two models as well.*

Theorem 6.1 is proven using the following three lemmas.

Lemma 6.2 *If ρ_1 is a lower bound on the deterministic (or randomized) competitive ratio for the “SE” model, then ρ_1 is a lower bound on the deterministic (respectively, randomized) competitive ratio for the “ET” model.*

Proof It can be easily shown that any input sequence of the “SE” model is also an input sequence of the “ET” model, with the expiration time of every edge set to 1. Since the lower bound for the special case also holds for the general case, the claim holds. \square

Lemma 6.3 *If ρ_2 is a lower bound on the deterministic (or randomized) competitive ratio for the “ET” model, then ρ_2 is a lower bound on the deterministic (respectively, randomized) competitive ratio for the “InD” model.*

Proof It can also be demonstrated that any input sequence of the “ET” model can be represented using the “InD” model. Each time an edge is presented (or expires) in the former model, it is inserted (or deleted) in the latter model. \square

Lemma 6.4 *If ρ_3 is a lower bound on the deterministic (or randomized) competitive ratio for the “InD” model, then ρ_3 is a lower bound of the deterministic (respectively, randomized) competitive ratio for the “SE” model.*

To prove Lemma 6.4, we first prove the following lemma:

Lemma 6.5 *Let A be a (possibly randomized) disengagement algorithm in the “SE” model, with a bounded competitive ratio. Then for any input sequence I in the “InD” model, there exists an input sequence I_D in the “SE” model such that with probability 1, A , when run on I_D , outputs a proper coloring of G_i , where $G_i = (V, E_i)$ is the graph in the “InD” model, after iteration i .*

Proof We construct I_D by repeating the edges of E_i cyclically. Specifically, for each i , I_D contains a sequence of *phases*, where in each phase, all edges of E_i are introduced (in any order). The number of phases depends on A : we claim that for any $\varepsilon > 0$ there exists M_ε such that the probability that A outputs a proper coloring of G_i after M_ε phases is least $1 - \varepsilon$.

To prove the claim, suppose, for contradiction, that for some $\varepsilon > 0$, the probability of A outputting a proper coloring of G_i after any number of phases is never more $1 - \varepsilon$. Then, with probability of at least ε , A pays at least 1 in each phase (when a monochromatic edge is inserted). The total expected cost of A is therefore unbounded.

On the other hand, the optimal cost is at most $|V|$, contradicting the assumption that A has bounded competitive ratio. \square

Proof of Lemma 6.4 Assume, for contradiction, that there exists an algorithm A_d for the “SE” model with a competitive ratio strictly smaller than ρ_3 . We construct an algorithm A_s for the “InD” model as follows. Given input $I = \{G_1, G_2, \dots\}$ (in the “InD” model), A_s runs A_d on input I_D (in the “SE” model) as defined in Lemma 6.5.

According to Lemma 6.5, since A_d has a bounded competitive ratio, then A_d outputs a solution to E_i w.p. 1. Hence, A_s is well defined.

To analyze the cost, consider iteration i of A_s . Let c_i^m denote the output of A_d after E_i was input to it for m -th time, and let c_i denote the output of A_s in the i -th step.

Let OPT_s denote an optimal solution to I and let OPT_d denote an optimal solution to I_d . Then we have

$$\begin{aligned} \text{cost}(A_s(I)) &= \sum_{i=1} \delta_H(c_i, c_{i-1}) \\ &\leq \sum_{i=1} \sum_{m \geq 1}^M \delta_H(c_i^m, c_i^{m-1}) && \text{by construction} \\ &= \text{cost}(A_d(I_d)) \\ &< \rho_3 \cdot \text{cost}(\text{OPT}_d(I_d)) && \text{by assumption on competitiveness of } A_d \\ &\leq \rho_3 \cdot \text{cost}(\text{OPT}_s(I)) && \text{because } \text{OPT}_s(I) \text{ is also a solution to } I_d, \end{aligned}$$

contradicting the assumption that ρ_3 is a lower bound on the competitiveness of deterministic algorithms for the “InD” model. \square

We note that FD is at least as hard as the partially-dynamic model.

Corollary 6.6 *If ρ is a lower bound on the deterministic (or randomized) competitive ratio for partially-dynamic disengagement, then ρ is a lower bound of the deterministic (respectively, randomized) competitive ratio for fully-dynamic disengagement.*

Proof The partially dynamic (PD) disengagement model can be directly represented by the FD “ET” model, with the expiration time of every edge set to ∞ . Hence, the lower bound for the PD case also holds for the “ET” model, and according to Theorem 6.1, such bound holds for any of the other FD models. \square

6.2 Full-Dynamic Disengagement and Metrical Task Systems

We now give evidence to the hardness of FD disengagement using a reduction from metrical task systems (MTS). Let us first define the terms.

In the MTS problem [7], a metric space M of n points is given. A server is always located at some point in the metric space. Requests arrive in an online fashion, where the i 'th request is a cost function $f_i : M \rightarrow \mathbb{R}^+$. The algorithm responds to a request f_i by moving the server from its current location $u \in M$ to a new location $v \in M$, paying the distance between the locations in the metric space (or not moving the server, and paying nothing). Then, the algorithm must pay the cost of the current location of the server, which is $f_i(v)$. We show that FD is as hard as a particular type of an MTS.

Theorem 6.7 *If there exists an $O(f(n))$ -competitive algorithm for FD disengagement for every graph of n nodes with 2 colors, then there exists an $O(f(n))$ -competitive algorithm for metrical task system on an odd cycle on length n .*

In [7], a lower bound of $\Omega(n)$ -competitive was presented for every deterministic algorithm for MTS on every set of n points. Hence we have the following.

Corollary 6.8 *Any deterministic algorithm for FD disengagement with n nodes is $\Omega(n)$ -competitive.*

A lower bound of $\Omega(\log n / \log \log n)$ -competitiveness for every randomized algorithm for MTS on every set of n points was given in [6]. This implies the following corollary.

Corollary 6.9 *Any randomized algorithm for FD disengagement with n nodes is $\Omega\left(\frac{\log n}{\log \log n}\right)$ -competitive.*

Proof of Theorem 6.7 We assume without loss of generality that the functions in the MTS input are in fact Kronecker delta functions: they are supported on only one point, and the cost of that point is 1. We also assume w.l.o.g. that the FD disengagement algorithm is lazy, i.e., it only recolors endpoints of the current request.

We now describe the reduction. Suppose we are given an instance \mathcal{M} of MTS with an odd-cycle metric space C_1 of n points. We construct an instance \mathcal{D} of FD with n points arranged in a cycle, such that the nodes of C_1 correspond to the n edges of C_2 in the natural way. The intended interpretation of nodes in C_1 is colorings in \mathcal{D} , where a node x of C_1 means in C_2 that only the edge corresponding to x is monochromatic (and all other edges in C_2 are bichromatic). The initial coloring in \mathcal{D} is such that the only monochromatic edge in \mathcal{D} is the edge corresponding to the initial position of the server in \mathcal{M} . Thereafter, whenever a point $x \in C_1$ is requested in \mathcal{M} , the corresponding edge in C_2 is requested in \mathcal{D} . To see that the intended interpretation is maintained, consider any algorithm ALG_D for FD. Easy induction shows that after every step, there is exactly one monochromatic edge in the odd cycle C_2 : The base case holds by the initial coloring; for the induction step, note that if a non-monochromatic edge is requested, no recoloring takes place. Otherwise, the currently monochromatic edge e is requested, and ALG_D must recolor exactly one endpoint of e . If the clockwise

(say) endpoint of e is recolored, then e is no longer monochromatic, and the next edge clockwise from e becomes monochromatic (and similarly for counter-clockwise), and the induction is proved.

Given the reduction, algorithm ALG_M for the MTS problem is obtained from a given algorithm ALG_D for FD by taking the MTS instance \mathcal{M} , reducing it online to the FD instance \mathcal{D} as described above, running ALG_D on \mathcal{D} , and interpreting the responses of ALG_D back in \mathcal{M} . That is, whenever ALG_D moves the monochromatic edge clockwise, ALG_M moves the server clockwise, and the same for counter-clockwise. Clearly, we have that ALG_D has the same cost on \mathcal{D} as ALG_M has on \mathcal{M} .

Finally, denoting by OPT_M and OPT_D the optimal solutions to \mathcal{M} and \mathcal{D} respectively, note that $\text{OPT}_D \leq 2\text{OPT}_M$: OPT_D can move the monochromatic edge whenever OPT_M moves the server, and can move the monochromatic edge clockwise and then immediately counter-clockwise to simulate paying a penalty (at a cost of 2). This completes the proof of the theorem. \square

7 Conclusion

In this paper we have introduced the problem of online disengagement and determined its competitive ratio in the case that conflicts are permanent and either that the final graph is bipartite or that the number of colors is larger than the maximum degree. Many problems remain open.

A major problem we leave open is the competitive ratio in the fully-dynamic case, where conflicts are temporary.

Other natural variants that we leave for future research are the following.

- Explore further the *capacitated disengagement* case (Sect. 5.2). Using the coloring formalism, in this case we assume that each color l comes with prescribed maximum capacity W_l such that in any coloring output by the algorithm, the total number (or, more generally, weight) of nodes assigned to color l does not exceed W_l .
- The *list recoloring* variant represents the case where jobs have both affinities and anti-affinities, i.e., each job is given both a subset of machines on which it may run, and disengagement requests express job separation constraints. Formally, this is modeled by requiring the colorings output by the algorithm to be list colorings. The lists may be fixed or change online.
- In *multiple disengagement*, disengagement requests are arbitrary sets of jobs. If the requirement is that at least one of the jobs in a conflict set is not collocated with all others, we have a hypergraph recoloring problem.
- Both Kashyop et al. [14] and Solomon and Wein [17] provide dynamic-coloring algorithms in the case of bounded (or constant) arboricity. Their methods might prove useful for the competitive recoloring case as well.

Note that all versions listed above make sense in either the static or the dynamic variants of the disengagement problem.

Declarations

Conflict of interest We are not aware of any conflicts of interest regarding the research presented in this paper.

References

1. Allybokus, Z., Perrot, N., Leguay, J., Maggi, L., Gourdin, E.: Virtual function placement for service chaining with partial orders and anti-affinity rules. *Networks* **71**(2), 97–106 (2018)
2. Avin, C., Bienkowski, M., Loukas, A., Pacut, M., Schmid, S.: Dynamic balanced graph partitioning. *SIAM J. Discrete Math.* **34**(3), 1791–1812 (2020)
3. Avin, C., Loukas, A., Pacut, M., Schmid, S.: Online balanced repartitioning. In: 30th International Symposium on Distributed Computing (DISC) (2016)
4. Bar-Yehuda, R., Even, S.: A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms* **2**(2), 198–203 (1981)
5. Barba, L., Cardinal, J., Korman, M., Langerman, S., van Renssen, A., Roeloffzen, M., Verdonschot, S.: Dynamic graph coloring. *Algorithmica* **81**(4), 1319–1341 (2019)
6. Bartal, Y., Bollobas, B., Mendel, M.: A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pp. 396–405 (2001)
7. Borodin, A., Linial, N., Saks, M.E.: An optimal on-line algorithm for metrical task system. *J. ACM* **39**(4), 745–763 (1992)
8. Bosek, B., Disser, Y., Feldmann, A.E., Pawlewicz, J., Zych-Pawlewicz, A.: Recoloring interval graphs with limited recourse budget. In: Proceedings of 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020), volume 162 of Leibniz International Proceedings in Informatics (LIPIcs), 17:1–17:23, Dagstuhl, Germany (2020)
9. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco (1979)
10. Halldórsson, M.M.: Parallel and on-line graph coloring. *J. Algorithms* **23**(2), 265–280 (1997)
11. Halldórsson, M.M., Szegedy, M.: Lower bounds for on-line graph coloring. *Theoret. Comput. Sci.* **130**(1), 163–174 (1994)
12. Henzinger, M., Neumann, S., Räcke, H., Schmid, S.: Tight bounds for online graph partitioning. In: Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2021), pp. 2799–2818 (2021)
13. Henzinger, M., Neumann, S., Wiese, A.: Explicit and implicit dynamic coloring of graphs with bounded arboricity (2020). CoRR [arXiv:2002.10142](https://arxiv.org/abs/2002.10142)
14. Kashyop, M.J., Narayanaswamy, N.S., Nasre, M., Potluri, S.M.: Dynamic coloring for bipartite and general graphs (2019). CoRR, [arXiv:1909.07854](https://arxiv.org/abs/1909.07854)
15. Lovász, L., Saks, M., Trotter, W.T.: An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Math.* **75**(1–3), 319–325 (1989)
16. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. *Commun. ACM* **28**(2), 202–208 (1985)
17. Solomon, S., Wein, N.: Improved dynamic graph coloring. *ACM Trans. Algorithms* **16**(3), 1–24 (2020)
18. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.* **3**(6), 103–128 (2007)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.