A Subquadratic Bound for Online Bisection

Marcin Bienkowski ✉ 📵
University of Wrocław, Poland

Stefan Schmid ✉ 📵
TU Berlin, Germany
Weizenbaum Institute, Berlin, Germany

── **Abstract** ──────────────────────

The *online bisection problem* is a natural dynamic variant of the classic optimization problem, where one has to dynamically maintain a partition of n elements into two clusters of cardinality $n/2$. During runtime, an online algorithm is given a sequence of requests, each being a pair of elements: an inter-cluster request costs one unit while an intra-cluster one is free. The algorithm may change the partition, paying a unit cost for each element that changes its cluster.

This natural problem admits a simple deterministic $O(n^2)$-competitive algorithm [Avin et al., DISC 2016]. While several significant improvements over this result have been obtained since the original work, all of them either limit the generality of the input or assume some form of resource augmentation (e.g., larger clusters). Moreover, the algorithm of Avin et al. achieves the best known competitive ratio even if randomization is allowed.

In this paper, we present the first randomized online algorithm that breaks this natural quadratic barrier and achieves a competitive ratio of $\tilde{O}(n^{23/12})$ without resource augmentation and for an arbitrary sequence of requests.

1 Introduction

The clustering of elements into subsets that are related by some similarity measure is a fundamental algorithmic problem. The problem arises in multiple contexts: a well-known abstraction is *the bisection problem* [12] that asks for partitioning of n graph nodes (elements) into two clusters of size $n/2$, so that the number of graph edges in the cut is minimized. This problem is NP-hard and its approximation ratio has been improved in a long line of papers [20, 1, 8, 7, 13]; the currently best approximation ratio of $O(\log n)$ was given by Räcke [17].

Recently this problem has been studied in a *dynamic variant* [3, 18], where instead of a fixed graph, we are given a sequence of element pairs. Serving a pair of elements that are in different clusters costs one unit, while a request between two elements in the same cluster is free. After serving a request, an algorithm may modify the partition, paying a unit cost for each element that changes its cluster.

A natural motivation for this problem originates from data centers where communicating virtual machines (elements) have to be partitioned between servers (clusters) and the overall communication cost has to be minimized: by collocating virtual machines on the same server, their communication becomes free, while the communication between virtual machines in different clusters involves using network bandwidth. Modern virtualization technology

supports the seamless migration of virtual machines between servers, but migrations still come at the cost of data transmission. The goal is to minimize data transmission (across the network), comprising inter-cluster communication and migration.

This practical application motivates yet another aspect of the problem: the communication pattern (and, in particular, the sequence of communication pairs) is typically not known ahead of time. Accordingly, we study the dynamic variant in the *online setting*, where the sequence of communication pairs is not known a priori to an online algorithm ALG: ALG has to react immediately without the knowledge of future communication requests. To evaluate its performance, we use a standard notion of *competitive ratio* [6] that compares the cost of ALG to the cost of the optimal (offline) solution OPT: the ALG-to-OPT cost ratio is subject to minimization.

Previous Results.   Avin et al. introduced the online bisection problem and presented a simple deterministic online algorithm that achieves the competitive ratio of $O(n^2)$ [3]. Their algorithm belongs to a class of component-preserving algorithms (formally defined in Section 2). Roughly speaking, it splits the request sequence into epochs. Within a single epoch, it glues requested element pairs together creating components and assigns all elements of any component to the same cluster. If such an assignment is no longer feasible, i.e., components cannot be preserved (kept on the same cluster), an epoch terminates. It is easy to argue (cf. Section 2) that OPT pays at least 1 in an epoch, and any component-preserving algorithm pays at most n^2, thus being n^2-competitive.

Perhaps surprisingly, no better algorithm (even a randomized one) is known for the online bisection problem. On the negative side, a lower bound of $\Omega(n)$ [2] for deterministic algorithms follows by the reduction from online paging [21].

Our Contribution.   We present the first algorithm for the online bisection problem that beats the quadratic competitive ratio. All previous results with better ratios required some relaxation: either used resource augmentation or restricted the generality of the input sequence. Our IMPROVED COMPONENT BASED algorithm (ICB) is randomized, follows the component-preserving framework outlined above, and achieves the competitive ratio of $O(n^{23/12} \cdot \sqrt{\log n})$.

Our Algorithmic Ideas.   Assume that an algorithm follows the component-preserving framework and we want to improve its cost within a single epoch. We may look at the problem more abstractly: there is a set of "allowed" partitions (the ones that map elements to clusters in a component-preserving way), and this set is constantly shrinking. Consider an algorithm that, whenever it needs to change its partition, changes it to one chosen uniformly at random from the set of still allowed partitions. Using standard arguments, we may argue that the algorithm changes its partition at most $O(\log y)$ times within an epoch, where y is the number of "allowed" partitions at the beginning. As the cost of serving the request and the cost of changing the partition is at most $1 + n$, the overall cost of such routine is $O(n \cdot \log y)$.

At the beginning of an epoch $y = 2 \cdot \binom{n}{n/2}$, and thus $O(n \cdot \log y) = O(n^2)$. That is, the randomized routine itself would fail to beat the quadratic upper bound of [3] if it is applied to the entire epoch. However, we may execute it in the second stage of an epoch, once the number of "allowed" partitions drops appropriately.

In the first stage of an epoch, our proposed algorithm Icb carefully tracks the component sizes. In a single step, it needs to merge two components into a single one and to map all elements of the resulting component to the same cluster. To this end, it usually has to move one of the merged components to the other cluster. A crucial insight is that, most of the time, the moved component size can be expressed as a linear combination of a moderate number of existing component sizes: in such case, only limited number of existing components have to change their clusters.

Converting this intuition into an actual algorithm is not easy. To this end, we provide a way of maintaining the greatest common divisor (GCD) of a large subset of components, so that this GCD changes only a few times within an epoch. We use number-theoretic properties to argue that whenever Icb merges two components into a single one, then usually one of them is divisible by the current value of GCD, and thus the resulting repartitioning incurs the movement of only a moderate number of other components.

The low-cost argument depends, however, on the property that not only there are many components of sizes divisible by GCD, but also both clusters contain sufficiently many of them. Icb ensures this property by regularly running a "rebalancing" routine. At some point, maintaining this property is no longer possible. We prove that such failure guarantees that the total number of "allowed" partitions is appropriately low: Icb switches then to the second stage of an epoch, where it executes the randomized policy outlined above.

Related Work. The lack of progress toward improving the $O(n^2)$ upper bound motivated the investigation of simplified variants.

A natural relaxation involves resource augmentation, where each cluster of an online algorithm can accommodate $(1 + \varepsilon) \cdot (n/2)$ elements. The performance of an online algorithm is compared to Opt whose both clusters still have capacity $n/2$. Surprisingly, the competitive ratio remains $\Omega(n)$ even for large ε (but as long as $\varepsilon < 1$) [2]. On the positive side, Rajaraman and Wasim showed an $O(n \log n)$-competitive deterministic algorithm for a fixed $\varepsilon > 0$ [19].

Another relaxation was introduced by Henzinger et al. [11] who initiated the study of the so-called *learning variant*. In this variant, there exists a fixed partition $\bar{p}$ (unknown to an algorithm), and all requests are consistent with $\bar{p}$ (i.e., given between same-cluster pairs). Clearly, the optimal solution simply changes its partition to $\bar{p}$ at the very beginning. The deterministic variant is asymptotically resolved: the optimal competitive ratio is $\Theta(n)$ [15, 16]. For the model where the learning variant is combined with resource augmentation, Henzinger et al. gave a $\Theta(\log n)$-competitive deterministic solution (for any fixed $\varepsilon > 0$) [10].

The online bisection problem has also been studied in a generalized form, where there are $\ell > 2$ clusters, each of size n/ℓ. This extension is usually referred to as *online balanced graph partitioning*. Some of the results presented above can be generalized to this variant [2, 3, 10, 11, 15, 16, 19]. This generalization was investigated also in models with a large augmentation of $\varepsilon > 1$ [2, 9, 10, 18] and in settings with small (or even constant-size) clusters [2, 4, 16].

2 Preliminaries

We have a set V of n elements and two clusters 0 and 1. A valid partition of these elements is a mapping $p : V \to \{0, 1\}$ such that $|p^{-1}(0)| = |p^{-1}(1)| = n/2$, i.e., each cluster contains exactly $n/2$ elements. For two partitions p and p', we use $\mathrm{dist}(p, p') = |\{v \in V : p(v) \neq p'(v)\}|$ to denote the number of elements that change their clusters when switching from partition p to p'.

Problem Definition.   An input for the online bisection problem consists of an initial partition p_0 and a sequence of element pairs $((u_t, v_t))_{t \geq 1}$. In step $t \geq 1$, an online algorithm ALG is given a pair of elements (u_t, v_t): it pays a *service cost* of 1 if $p_{t-1}(u_t) \neq p_{t-1}(v_t)$ and 0 otherwise. Afterward, ALG has to compute a new partition p_t (possibly $p_t = p_{t-1}$) and pay $\mathrm{dist}(p_{t-1}, p_t)$ for changing partition p_{t-1} to partition p_t.

For an input $\mathcal{I}$ and an online algorithm ALG, we use $\mathrm{ALG}(\mathcal{I})$ to denote its total cost on $\mathcal{I}$, whereas $\mathrm{OPT}(\mathcal{I})$ denotes the optimal cost of an offline solution. ALG is γ-*competitive* if there exists β, such that $\mathrm{ALG}(\mathcal{I}) \leq \gamma \cdot \mathrm{OPT}(\mathcal{I}) + \beta$ for any input $\mathcal{I}$. While β has to be independent of $\mathcal{I}$, it may be a function of n. For a randomized algorithm ALG, we replace $\mathrm{ALG}(\mathcal{I})$ with its expectation $\mathbf{E}[\mathrm{ALG}(\mathcal{I})]$, taken over all random choices of ALG.

Component-Preserving Framework.   A natural way of tackling the problem is to split requests into epochs. In a single epoch, an online algorithm ALG treats requests as edges connecting requested element pairs. Edges in a single epoch induce connected components of elements. A *component-preserving* algorithm always keeps elements of each component in the same cluster. If it is no longer possible, the current epoch ends, all edges are removed (each element is now in its own singleton component), and a new epoch begins with the next step. We note that the currently best $O(n^2)$-competitive deterministic algorithm of [3] is component-preserving.

Now, we recast the online bisection problem assuming that we analyze a component-preserving algorithm ALG. First, observe that ALG completely ignores all intra-component requests (they also incur no cost on ALG). Consequently, we may assume that the input for ALG (within a single epoch) is a sequence of component sets $\mathcal{C}_t$, where:

- $\mathcal{C}_0$ is the initial set of n singleton components;
- $\mathcal{C}_t$ (presented in step $t \geq 1$) is created from $\mathcal{C}_{t-1}$ by merging two of its components, denoted c_x^t and c_y^t. They are merged into a component, denoted c_z^t, i.e.

$$\mathcal{C}_t = \mathcal{C}_{t-1} \cup \{c_z^t\} \setminus \{c_x^t, c_y^t\}.$$

For a given set of components $\mathcal{C}$, let $\mathcal{P}(\mathcal{C})$ denote the set of all $\mathcal{C}$-preserving partitions of n elements into two clusters, i.e., ones that place all elements of a single component of $\mathcal{C}$ in the same cluster. In response to $\mathcal{C}_t$, ALG chooses a $\mathcal{C}_t$-preserving partition p_t. If $\mathcal{P}(\mathcal{C}_t)$ is empty though, then ALG does not change its partition, and an epoch terminates. The following observations let us focus on ALG's behavior in a single epoch only.

▶ **Lemma 1.** *The epoch of any component-preserving algorithm contains at most $n-1$ steps, and the single-step cost is at most $n+1$.*

Proof. In each step, the number of components decreases. Thus, after $n-1$ steps, all elements would be in the same component, and hence $\mathcal{P}(\mathcal{C}_{n-1}) = \emptyset$. In a single step, an algorithm pays at most 1 for serving the request and changes the cluster of at most n elements.   ◀

▶ **Lemma 2.** *If a component-preserving algorithm ALG pays at most R in any epoch, then ALG is R-competitive.*

Proof. Fix any finished epoch $\mathcal{E}$ in an input (any epoch except possibly the last one). The final step of $\mathcal{E}$ serves as a certificate that any algorithm keeping a static partition throughout $\mathcal{E}$ has a non-zero cost. On the other hand, changing partition costs at least 1, and thus $\mathrm{OPT}(\mathcal{E}) \geq 1$.

The lemma follows by summing costs over all epochs except the last one. Observe that the cost of the last epoch is at most $n^2 - 1$ (by Lemma 1), and thus can be placed in the additive term β in the definition of the competitive ratio (cf. Section 2).   ◀

Note that by Lemma 1 and Lemma 2 the competitive ratio of any component-preserving algorithm (including that of [3]) is at most $(n-1) \cdot (n+1) < n^2$.

Notation. For an integer ℓ, we define $[\ell] = \{1, 2, \ldots, \ell\}$. For any finite set $A \subset \mathbb{N}_{>0}$, we use $\gcd(A)$ to denote the greatest common divisor of all integers from A; we assume that $\gcd(\emptyset) = \infty$.

For any component c, we denote its size (number of elements) by $\mathrm{SIZE}(c)$. Fix any component set $\mathcal{C}$ and an integer i. Let $\mathrm{CNT}_i(\mathcal{C}) \triangleq |\{c \in \mathcal{C} : \mathrm{SIZE}(c) = i\}|$ denote the number of components in $\mathcal{C}$ of size i.

Furthermore, fix a partition $p \in \mathcal{P}(\mathcal{C})$ and a cluster $y \in \{0, 1\}$. As p is $\mathcal{C}$-preserving, it is constant on all elements of a given component $c \in \mathcal{C}$, and thus we may extend p to components from $\mathcal{C}$. We define

$$\mathrm{CNT}_i(\mathcal{C}, p, y) \triangleq |\{c \in \mathcal{C} : p(c) = y \wedge \mathrm{SIZE}(c) = i\}|$$

as the number of components in $\mathcal{C}$ of size i that are inside cluster y in partition p.

We extend both notions to sets of sizes, i.e., for any set A, we set $\mathrm{CNT}_A(\mathcal{C}) \triangleq \sum_{i \in A} \mathrm{CNT}_i(\mathcal{C})$ and $\mathrm{CNT}_A(\mathcal{C}, p, y) \triangleq \sum_{i \in A} \mathrm{CNT}_i(\mathcal{C}, p, y)$.

3 A Subquadratic Algorithm

Our IMPROVED COMPONENT BASED algorithm (ICB) is component-preserving. It splits an epoch into two stages. The first stage is deterministic: with a slight "rebalancing" exception that we explain later, the components are remapped to minimize the cost of changing the partition in a single step. At a carefully chosen step that we define later, ICB switches to the second stage. In any step t of the second stage, if the current partition p_{t-1} is not $\mathcal{C}_t$-preserving, ICB chooses p_t uniformly at random from $\mathcal{P}(\mathcal{C}_t)$.

We now focus on describing the first stage of an epoch. Our algorithm ICB uses a few integer parameters defined below.

- Parameter $q \in [n]$. A component size is *large* if it is greater than q and is *small* otherwise.
- Parameter $w \in [n]$. If $\mathrm{CNT}_i(\mathcal{C}) \geq w$, we call size i *popular* (in $\mathcal{C}$).
- Parameter $d \in [n]$.

ICB works with any values of q, w, d, as long as they satisfy $6 \cdot q^4 + 3 \leq w$, $q \cdot (2 \cdot w + 1) \leq d$, and $2 \cdot d \leq n$. The parameter values yielding the competitive ratio of $O(n^{23/12} \cdot \sqrt{\log n})$ are chosen in Theorem 11.

3.1 Helper Notions

First, for any $k \in \mathbb{N}_{>0} \cup \{\infty\}$ we define

$$\langle k \rangle \triangleq \{\ell \cdot k : \ell \in \mathbb{N}\} \cap [q].$$

In particular, $\langle \infty \rangle = \emptyset$. That is, $\langle k \rangle$ contains all small component sizes that are divisible by k. Observe that

$$k = \gcd(\langle k \rangle) \qquad \text{for any } k \in [q] \cup \{\infty\}. \tag{1}$$

Second, we introduce the notion of a *balanced partition*. Fix a set of components $\mathcal{C}$, a value k, and an integer ℓ. For a given partition $p \in \mathcal{P}(\mathcal{C})$, we say that p is (k, ℓ)-balanced if

$$\mathrm{CNT}_{\langle k \rangle}(\mathcal{C}, p, y) \geq \ell \qquad \text{for } y \in \{0, 1\}.$$

■ **Algorithm 1** The first stage of an epoch of ICB.
Input: initial partition p_0, sequence of component sets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t, \ldots$
Output: sequence of partitions $p_1, p_2, \ldots, p_t, \ldots$, where p_t is $\mathcal{C}_t$-preserving.
Initialization: $g_0 \leftarrow 1$.

Processing $\mathcal{C}_t$ (step $t \geq 1$)

1: **if** $\mathcal{P}(\mathcal{C}_t) = \emptyset$ **then**   ▷ *no $\mathcal{C}_t$-preserving partition*
2:      $p_t \leftarrow p_{t-1}$
3:      **terminate** the current epoch
4: $B_t = \{i \in [q] \;:\; \mathrm{CNT}_i(\mathcal{C}_t) \geq w\}$ ▷ *B_t contains small popular sizes*
5: $g_t \leftarrow \gcd(\langle g_{t-1} \rangle \cap B_t)$
6: **if** $\mathcal{P}(\mathcal{C}_t, g_t, 2d) = \emptyset$ **then**   ▷ *no $\mathcal{C}_t$-preserving $(g_t, 2d)$-balanced partition*
7:      $p_t \leftarrow p_{t-1}$
8:      **terminate** the first stage of the current epoch
9: $p_t^* \leftarrow \arg\min_p\{\mathrm{dist}(p_{t-1}, p) : p \in \mathcal{P}(\mathcal{C}_t)\}$ ▷ *pick closest $\mathcal{C}_t$-preserving candidate*
10: **if** $p_t^* \in \mathcal{P}(\mathcal{C}_t, g_t, d)$ **then**   ▷ *rebalance if necessary*
11:      $p_t \leftarrow p_t^*$
12: **else**
13:      $p_t \leftarrow$ any partition from $\mathcal{P}(\mathcal{C}_t, g_t, 2d)$

That is, a (k, ℓ)-balanced partition p of $\mathcal{C}$ keeps at least ℓ small components of sizes divisible by k in each cluster. We use $\mathcal{P}(\mathcal{C}, k, \ell) \subseteq \mathcal{P}(\mathcal{C})$ to denote the set of all (k, ℓ)-balanced $\mathcal{C}$-preserving partitions.

3.2 Definition of the First Stage

The pseudo-code of ICB for the first stage is given in Algorithm 1; we describe it also below.

Computing GCD Estimator. Initially, in step t, in Lines 1–3, ICB verifies whether a $\mathcal{C}_t$-preserving partition exists, and terminates the epoch without changing the current partition otherwise.

Next, ICB sets B_t to be the set of small popular component sizes of $\mathcal{C}_t$ and computes the value of *GCD estimator* $g_t \in [q] \cup \{\infty\}$. The computation balances two objectives: on one hand, we want g_t to be the greatest common divisor of B_t, on the other hand, we do not want g_t to change too often. Therefore, g_t is defined by the following iterative process (cf. Lines 4–5): We initialize $g_0 = 1$ (i.e., $\langle g_0 \rangle = [q]$). In step t, we set $g_t = \gcd(\langle g_{t-1} \rangle \cap B_t)$. Note that this process ensures that $g_t \in [q] \cup \{\infty\}$ for any t.

Triggering the Second Stage. Lines 6–8 ensure that there exists a $(g_t, 2d)$-balanced partition of $\mathcal{C}_t$. If this is not the case, ICB terminates the first stage without changing its partition and switches to the second stage of an epoch in the next step.

Choosing a New Partition. Finally, in Lines 9–13, ICB chooses its new partition p_t. First, it computes a *candidate partition* p_t^* as a $\mathcal{C}_t$-preserving partition closest to p_{t-1}. If p_t^* is (g_t, d)-balanced, it simply outputs $p_t = p_t^*$. Otherwise, it discards p_t^*, and picks any $(g_t, 2d)$-balanced partition as p_t. We call such action *rebalancing*; we later show that it occurs rarely, i.e., in most cases $p_t = p_t^*$.

4   Analysis Roadmap

In this section, we describe the framework of our analysis in a top-down approach, listing the necessary lemmas that will be proven in the next sections, and showing that their combination yields the desired competitiveness bound.

Lemma 2 allows us to focus only on the cost of ICB in a single epoch $\mathcal{E}$. We denote its two stages by $\mathcal{E}_1$ and $\mathcal{E}_2$; the second stage may be empty if ICB terminates the first stage already in Lines 1–3. We identify $\mathcal{E}_1$ and $\mathcal{E}_2$ with the sets of the corresponding steps. In particular, we use T as the number of steps in the first stage, i.e., $\mathcal{E}_1 = [T]$.

The Second Stage.   We start from a simpler case, the cost analysis in $\mathcal{E}_2$. The lemmas stated below are proven in Section 6. We assume that $\mathcal{E}_2$ is non-empty as otherwise the associated cost is trivially zero. That is, ICB switches to the second stage because the condition in Line 6 becomes true, i.e., $\mathcal{P}(\mathcal{C}_T, g_T, 2d) = \emptyset$.

By observing that in the second stage, ICB is essentially a randomized algorithm solving the metrical tasks system (MTS) problem [5] on a uniform metric of $|\mathcal{P}(\mathcal{C}_T)|$ points, we obtain the following bound.

▶ **Lemma 3.** $\mathbf{E}[\text{ICB}(\mathcal{E}_2)] = O(n \cdot \log |\mathcal{P}(\mathcal{C}_T)|)$.

The usefulness of the lemma above depends on how well we can bound $|\mathcal{P}(\mathcal{C}_T)|$, the number of $\mathcal{C}_T$-preserving partitions. The second stage is executed only when $\mathcal{P}(\mathcal{C}_T, g_T, 2d) = \emptyset$, i.e., at step T, all $\mathcal{C}_T$-preserving partitions have less than $2d$ components of sizes from $\langle g_T \rangle$ in one of the clusters. This, together with combinatorial counting arguments, implies the following bound.

▶ **Lemma 4.** $|\mathcal{P}(\mathcal{C}_T)| = \exp(O(d \cdot \log n + z))$, where $z = \text{CNT}_{[n] \setminus \langle g_T \rangle}(\mathcal{C}_T)$.

The term $\text{CNT}_{[n] \setminus \langle g_T \rangle}(\mathcal{C}_T)$ denotes the number of components of sizes outside set $\langle g_T \rangle$ and we will bound it later using the behavior of ICB in $\mathcal{E}_1$.

The First Stage: Rebalancing Costs.   Now we switch our attention to the core of our approach, the first stage of an epoch. Recall that in a single step t, ICB pays at most 1 for serving the request and $\text{dist}(p_{t-1}, p_t)$ for changing the partition. We may upper-bound the latter term by $\text{dist}(p_{t-1}, p_t^*) + \text{dist}(p_t^*, p_t)$; we call the corresponding summands *switching cost* and *rebalancing cost*. It turns out that the latter part can be upper-bounded using the former. We define

$$\text{ICB}^{\text{ss}}(t) \triangleq 1 + \text{dist}(p_{t-1}, p_t^*),$$
$$\text{ICB}^{\text{rb}}(t) \triangleq \text{dist}(p_t^*, p_t).$$

Clearly, $\text{ICB}(t) \leq \text{ICB}^{\text{ss}}(t) + \text{ICB}^{\text{rb}}(t)$.

▶ **Lemma 5.** It holds that $\text{ICB}^{\text{rb}}(\mathcal{E}_1) \leq O(n \cdot \log q) + O(n/d) \cdot \text{ICB}^{\text{ss}}(\mathcal{E}_1)$.

The rough idea behind the lemma above (proved formally in Subsection 5.2) is that the rebalancing cost is at most n and between two consecutive rebalancing actions, ICB pays already $\Omega(d)$ of switching cost. This statement is not always true, but it fails at most for $O(\log q)$ consecutive rebalancing actions.

The First Stage: Serving and Switching Costs. By the argument above, it now suffices to estimate $\text{ICB}^{\text{ss}}(\mathcal{E}_1)$. In Subsection 5.1 we study the evolution of g_t as a function of time step t. Recall that $g_0 = 1$ and $g_t \in [q] \cup \{\infty\}$ for any t. We say that a step t is *g-updating* if $g_t \neq g_{t-1}$.

▶ **Lemma 6.** *The number of g-updating steps within $\mathcal{E}_1$ is at most $1 + \log q$. Furthermore, $\langle g_t \rangle \subseteq \langle g_{t-1} \rangle$ for each step t of $\mathcal{E}_1$.*

Recall that c_x^t and c_y^t are the components merged in step t. To bound the switching cost, we distinguish between regular and irregular steps. In regular ones, at least one merged component is small and its size is divisible by g_{t-1}.

▶ **Definition 7.** *A step t is* regular *if $\text{SIZE}(c_x^t) \in \langle g_{t-1} \rangle$ or $\text{SIZE}(c_y^t) \in \langle g_{t-1} \rangle$ and* irregular *otherwise.*

In Subsection 5.3, we argue that the switching and serving cost in regular steps is $o(n)$. To this end, we observe that Lines 10–13 executed in step $t - 1$ ensure (by running rebalancing if necessary) that p_{t-1} is (g_{t-1}, d)-balanced partition from $\mathcal{P}(C_{t-1})$, i.e., both clusters of partition p_{t-1} contain at least d small components of sizes divisible by g_{t-1}. This property becomes useful at the beginning of step t: using number-theoretic arguments, we may bound the number of components that need to be moved between clusters, so that eventually c_x^t and c_y^t end up in the same cluster.

▶ **Lemma 8.** *For any regular and not g-updating step t of $\mathcal{E}_1$, $\text{ICB}^{\text{ss}}(t) = O(q^4)$.*

Finally, in Subsection 5.4, we argue that there are $o(n)$ irregular steps and we also bound the number of components whose sizes are either large or not divisible by g_t.

▶ **Lemma 9.** *There are at most $O(q \cdot w + n/q)$ irregular steps in $\mathcal{E}_1$. Moreover, at any time t of $\mathcal{E}_1$, $\text{CNT}_{[n] \setminus \langle g_t \rangle}(\mathcal{C}_t) = O(q \cdot w + n/q)$.*

Estimating the Total Cost. We may now combine the bounds presented above to prove the desired competitive ratio.

▶ **Lemma 10.** *For any epoch $\mathcal{E}$, $\mathbf{E}[\text{ICB}(\mathcal{E})] = O((n^2/d) \cdot (q^4 + q \cdot w + n/q) + n \cdot d \cdot \log n)$.*

Proof. We split epoch $\mathcal{E}$ into two stages, $\mathcal{E}_1$ and $\mathcal{E}_2$, and let $T = |\mathcal{E}_1|$.

We first upper-bound the cost within $\mathcal{E}_1$. Let $R \subseteq [T]$ be the set of regular steps of $\mathcal{E}_1$ that are not g-updating. Then each step from $[T] \setminus R$ is either irregular or g-updating. By Lemma 6 and Lemma 9,

$$|[T] \setminus R| \leq (1 + \log q) + O(q \cdot w + n/q) = O(q \cdot w + n/q). \tag{2}$$

This allows us to upper-bound the serving and switching cost of ICB in $\mathcal{E}_1$ as

$$
\begin{aligned}
\text{ICB}^{\text{ss}}(\mathcal{E}_1) &= \sum_{t \in R} \text{ICB}^{\text{ss}}(t) + \sum_{t \in [T] \setminus R} \text{ICB}^{\text{ss}}(t) \\
&\leq \sum_{t \in R} O(q^4) + \sum_{t \in [T] \setminus R} (n+1) && \text{(by Lemma 8 and Lemma 1)} \\
&= |R| \cdot O(q^4) + O(q \cdot w + n/q) \cdot (n+1) && \text{(by (2))} \\
&= O(n \cdot (q^4 + q \cdot w + n/q)). && \text{(as } R \leq T \leq n-1\text{)}
\end{aligned}
$$

The total cost in $\mathcal{E}_1$ (including rebalancing) is then

$$
\begin{aligned}
\mathrm{ICB}(\mathcal{E}_1) &= \mathrm{ICB}^{\mathrm{ss}}(\mathcal{E}_1) + \mathrm{ICB}^{\mathrm{rb}}(\mathcal{E}_1) \\
&= \mathrm{ICB}^{\mathrm{ss}}(\mathcal{E}_1) + O(n \cdot \log q) + O(n/d) \cdot \mathrm{ICB}^{\mathrm{ss}}(\mathcal{E}_1) \qquad \text{(by Lemma 5)} \\
&= O(n^2/d) \cdot (q^4 + q \cdot w + n/q).
\end{aligned}
$$

The total expected cost in $\mathcal{E}_2$ (assuming $\mathcal{E}_2$ is present) is

$$
\begin{aligned}
\mathbf{E}[\mathrm{ICB}(\mathcal{E}_2)] &= O(n \cdot \log |\mathcal{P}(\mathcal{C}_T)|) &\text{(by Lemma 3)} \\
&= n \cdot O(d \cdot \log n + \mathrm{CNT}_{[n] \setminus \langle g_T \rangle}(\mathcal{C}_T)) &\text{(by Lemma 4)} \\
&= O(n \cdot (d \cdot \log n + q \cdot w + n/q)). &\text{(by Lemma 9)}
\end{aligned}
$$

Summing up, the total expected cost in the whole epoch is

$$
\begin{aligned}
\mathbf{E}[\mathrm{ICB}(\mathcal{E})] &= \mathrm{ICB}(\mathcal{E}_1) + \mathbf{E}[\mathrm{ICB}(\mathcal{E}_2)] \\
&= O((n^2/d) \cdot (q^4 + q \cdot w + n/q) + n \cdot d \cdot \log n). \qquad \text{(as } d \le n) \qquad \blacktriangleleft
\end{aligned}
$$

▶ **Theorem 11.** ICB *is* $O(n^{23/12} \cdot \sqrt{\log n})$*-competitive for the online bisection problem.*

Proof. We set $q = \lceil n^{1/6} \rceil$, $w = 6 \cdot q^4 + 3$, and $d = \lceil n^{11/12}/\sqrt{\log n} \rceil$. Note that these values satisfy $q \cdot (2 \cdot w + 1) \le d$ and $2 \cdot d \le n$ for sufficiently large n. Applying Lemma 10, we obtain,

$$
\mathbf{E}[\mathrm{ICB}(\mathcal{E})] = O\left((n^2/d) \cdot n^{5/6} + n \cdot d \cdot \log n\right) = O\left(n^{23/12} \cdot \sqrt{\log n}\right).
$$

The theorem follows immediately by Lemma 2. $\blacktriangleleft$

5 Analysis: the First Stage of ICB

5.1 Structural Properties

For succinctness of arguments, we extend the notion of divisibility. Recall that $a \mid b$ means that b is divisible by a and is well defined for any two positive integers a and b. We extend it also to the cases where a and b are possibly infinite: $a \mid \infty$ for any $a \in \mathbb{N}_{>0} \cup \{\infty\}$ and $\infty \nmid b$ for any $b \in \mathbb{N}_{>0}$.

▷ **Claim 12.** For any sets of integers A and B, it holds that $\gcd(A) \mid \gcd(A \cap B)$.

Proof. The claim follows trivially if $A \cap B = \emptyset$ as in such case $\gcd(A \cap B) = \infty$. Thus, we may assume that $A \cap B \ne \emptyset$ (and hence also $A \ne \emptyset$). Fix any $i \in A \cap B$: as $i \in A$, we have $\gcd(A) \mid i$. Hence, $\gcd(A)$ is a divisor of all numbers from $A \cap B$, and therefore $\gcd(A) \mid \gcd(A \cap B)$. ◁

We now show that not only is g_t monotonically non-increasing, but when it grows in a g-updating step, the new value is a multiplicity of the old one.

▶ **Lemma 6.** *The number of g-updating steps within $\mathcal{E}_1$ is at most $1 + \log q$. Furthermore,* $\langle g_t \rangle \subseteq \langle g_{t-1} \rangle$ *for each step t of $\mathcal{E}_1$.*

Proof. Fix any step t of $\mathcal{E}_1$. By Claim 12, $\gcd(\langle g_{t-1} \rangle) \mid \gcd(\langle g_{t-1} \rangle \cap B_t)$. Note, however, that $\gcd(\langle g_{t-1} \rangle) = g_{t-1}$ by (1), and $\gcd(\langle g_{t-1} \rangle \cap B_t) = g_t$ by the definition of g_t (cf. Line 4 of the algorithm). Thus, $g_{t-1} \mid g_t$ for any step t.

If $g_t < \infty$, then $g_{t-1} < \infty$, and consequently $\langle g_t \rangle \subseteq \langle g_{t-1} \rangle$ follows by the definition of $\langle \cdot \rangle$. Otherwise $g_t = \infty$, in which case $\langle g_t \rangle = \emptyset \subseteq \langle g_{t-1} \rangle$. Thus, the second part of the lemma follows.

For the first part, let ℓ be the number of all g-updating steps within $\mathcal{E}_1$; we denote them by $\tau(1), \tau(2), \ldots, \tau(\ell)$. Let $\tau(0) = 0$. For any $i \in \{0, \ldots \ell - 1\}$, it holds that $g_{\tau(i)} \mid g_{\tau(i+1)}$ and $g_{\tau(i)} \neq g_{\tau(i+1)}$, which implies $g_{\tau(i+1)} \geq 2 \cdot g_{t(i)}$. While it is possible that $g_{\tau(\ell)} = \infty$, we have $g_{\tau(\ell-1)} < \infty$, and thus $g_{\tau(\ell-1)} \leq q$. Hence, $g_{\tau(\ell-1)} \geq 2^{\ell-1} \cdot g_{\tau(0)} = 2^{\ell-1}$, and therefore $2^{\ell-1} \leq q$, which concludes the first part of the lemma. ◀

5.2 Rebalancing Cost

We first argue that between two consecutive rebalancing events ICB accrues sufficiently large serving and switching costs.

▷ **Claim 13.** Let a and b be two consecutive steps where rebalancing is executed. If $g_a = g_b$, then $\sum_{t=a+1}^{b} \mathrm{ICB^{ss}}(t) \geq d/3$.

Proof. For any step $t \in \{a+1, \ldots, b-1\}$, there is no rebalancing in t, and thus $p_t = p_t^*$.

$$
\begin{aligned}
\sum_{t=a+1}^{b} \mathrm{ICB^{ss}}(t) &= \sum_{t=a+1}^{b} (1 + \mathrm{dist}(p_{t-1}, p_t^*)) \\
&= (b-a) + \left(\sum_{t=a+1}^{b-1} \mathrm{dist}(p_{t-1}, p_t^*) \right) + \mathrm{dist}(p_{b-1}, p_b^*) \\
&= (b-a) + \sum_{t=a+1}^{b-1} \mathrm{dist}(p_{t-1}, p_t) + \mathrm{dist}(p_{b-1}, p_b^*) \\
&\geq (b-a) + \mathrm{dist}(p_a, p_b^*),
\end{aligned}
$$

where the final relation follows by the triangle inequality. If $b - a \geq d/3$, the lemma follows immediately, and thus we assume otherwise and we will show that $\mathrm{dist}(p_a, p_b^*) \geq d/3$. Let $g = g_a = g_b$.

As rebalancing was triggered in step b, we have $p_b^* \notin \mathcal{P}(\mathcal{C}_b, g, d)$. That is, partition p_b^* has less than d components from $\mathcal{C}_b$ of sizes from $\langle g \rangle$ in one of the clusters (say, in cluster 0). Observe that $\mathcal{C}_a$ can be obtained from $\mathcal{C}_b$ by going back in time and reversing component merges, i.e., performing $b - a$ splits of components. Each such split may create two extra components of size from $\langle g \rangle$. Hence, partition p_b^* keeps less than $d + 2 \cdot (b-a) < d + (2/3) \cdot d$ components of $\mathcal{C}_a$ of sizes from $\langle g \rangle$ in cluster 0.

Due to rebalancing in step a, we have $p_a \in \mathcal{P}(\mathcal{C}_a, g, 2d)$, i.e., p_a keeps $2d$ components of $\mathcal{C}_a$ in cluster 0. Therefore, $\mathrm{dist}(p_a, p_b^*) \geq d/3$, which concludes the proof. ◁

▶ **Lemma 5.** It holds that $\mathrm{ICB^{rb}}(\mathcal{E}_1) \leq O(n \cdot \log q) + O(n/d) \cdot \mathrm{ICB^{ss}}(\mathcal{E}_1)$.

Proof. Let ℓ be the number of rebalancing events within $\mathcal{E}_1$. Thus, there are $\ell - 1$ disjoint chunks between two consecutive steps with rebalancing events. By Lemma 6, at most $1 + \log q$ of these chunks contain g-updating steps. We may apply Claim 13 to the remaining chunks, which yields $\mathrm{ICB^{ss}}(\mathcal{E}_1) = \sum_{t \in [T]} \mathrm{ICB^{ss}}(t) \geq (\ell - 2 - \log q) \cdot (d/3)$. On the other hand, the cost of a single rebalancing event is at most n, and thus

$$
\begin{aligned}
\mathrm{ICB^{rb}}(\mathcal{E}_1) &\leq \ell \cdot n = (2 + \log q) \cdot n + (\ell - 2 - \log q) \cdot (d/3) \cdot (3n/d) \\
&\leq O(n \cdot \log q) + (3n/d) \cdot \mathrm{ICB^{ss}}(\mathcal{E}_1). \qquad \blacktriangleleft
\end{aligned}
$$

5.3 Bounding Switching Costs in Regular Steps

In this section, we show that the switching cost in regular steps is small. We start with a number-theoretic bound; its proof is deferred to the appendix.

$\triangleright$ **Claim 14.** Let $A = \{a_1, a_2, \ldots, a_k\} \subset \mathbb{N}_{>0}$ and $B = \{b_1, b_2, \ldots, b_\ell\} \subset \mathbb{N}_{>0}$ be two non-empty and disjoint sets of positive integers. Let $g = \gcd(A \uplus B)$ and $H = \max(A \uplus B)$. Then, there exist non-negative integers $r_1, r_2, \ldots r_k, s_1, s_2, \ldots s_\ell$, such that

$$\sum_{i=1}^{k} r_i \cdot a_i = g + \sum_{i=1}^{\ell} s_i \cdot b_i.$$

Moreover, $\sum_{i=1}^{k} r_i \cdot a_i \leq 3 \cdot (k + \ell) \cdot H^2$.

We now proceed to prove several helper claims showing that, in a non-g-updating step t, both clusters contain sufficiently many components whose sizes are divisible by g_{t-1}.

$\triangleright$ **Claim 15.** For any step t of $\mathcal{E}_1$, it holds that $g_{t-1} < \infty$.

Proof. As $g_0 = 1$, the lemma holds trivially for $t = 1$. Hence, we assume that $t > 1$. Suppose that $g_{t-1} = \infty$. Then, $\langle g_{t-1} \rangle = \emptyset$, and therefore, there is no $\mathcal{C}_{t-1}$-preserving $(g_{t-1}, 2d)$-balanced partition. Thus, when ICB executes Lines 6–8 in step $t - 1$, it would terminate $\mathcal{E}_1$ already in step $t - 1$. $\triangleleft$

$\triangleright$ **Claim 16.** For any non-g-updating step t, there exists a non-empty set $A \subseteq [q]$, such that
- $\gcd(A) = g_t$,
- $\text{CNT}_i(\mathcal{C}_{t-1}) \geq w - 1$ for any $i \in A$,
- $\text{CNT}_A(\mathcal{C}_{t-1}, p_{t-1}, y) \geq q \cdot w$ for any cluster $y \in \{0, 1\}$.

Proof. We will show that set $A \triangleq \langle g_{t-1} \rangle \cap B_t \subseteq [q]$ satisfies the properties of the lemma.

For the first property, observe that by Line 5 of the algorithm, $g_t = \gcd(\langle g_{t-1} \rangle \cap B_t)$, and thus $\gcd(A) = g_t$. As step t is not g-updating, $g_t = g_{t-1}$. By Claim 15, $g_{t-1} < \infty$, and thus $g_t < \infty$ as well, which implies that A is non-empty.

As $A \subseteq B_t$, the definition of B_t implies that $\text{CNT}_i(\mathcal{C}_t) \geq w$ for any $i \in A$. There is only one component, c_z^t, that is present in $\mathcal{C}_t$, but not present in $\mathcal{C}_{t-1}$. Thus, $\text{CNT}_i(\mathcal{C}_{t-1}) \geq w - 1$ for any $i \in A$. This proves the second property of the lemma.

Finally, to show the third property, we fix any $y \in \{0, 1\}$. Lines 10–13 executed in step $t - 1$ ensure that $p_{t-1} \in \mathcal{P}(\mathcal{C}_{t-1}, g_{t-1}, d)$, i.e., $\text{CNT}_{\langle g_{t-1} \rangle}(\mathcal{C}_{t-1}, p_{t-1}, y) \geq d$.

Fix any size $i \in \langle g_{t-1} \rangle \setminus A$. By the definition of A, we have $i \notin B_t$, and thus $\text{CNT}_i(\mathcal{C}_t) \leq w - 1$. As there are only two components, c_x^t and c_y^t, that are present in $\mathcal{C}_{t-1}$ but not present in $\mathcal{C}_t$, we have $\text{CNT}_i(\mathcal{C}_{t-1}) \leq w + 1$. Hence,

$$\begin{aligned} \text{CNT}_A(\mathcal{C}_{t-1}, p_{t-1}, y) &= \text{CNT}_{\langle g_{t-1} \rangle}(\mathcal{C}_{t-1}, p_{t-1}, y) - \text{CNT}_{\langle g_{t-1} \rangle \setminus A}(\mathcal{C}_{t-1}, p_{t-1}, y) \\ &\geq d - |\langle g_{t-1} \rangle \setminus A| \cdot (w + 1) \\ &\geq d - q \cdot (w + 1) \\ &\geq q \cdot w. \end{aligned}$$

where the last inequality follows as we assumed that $d \geq q \cdot (2 \cdot w + 1)$ in the definition of the algorithm. $\triangleleft$

$\triangleright$ **Claim 17.** For any non-g-updating step t, at least one of the following properties holds:
- $\text{CNT}_{g_t}(\mathcal{C}_{t-1}, p_{t-1}, y) \geq w$ for each cluster $y \in \{0, 1\}$.
- There exists two disjoint, non-empty sets $A_0, A_1 \subseteq [q]$, such that $\gcd(A_0 \uplus A_1) = g_t$ and for each $y \in \{0, 1\}$ and $i \in A_y$, it holds that $\text{CNT}_i(\mathcal{C}_{t-1}, p_{t-1}, y) \geq (w - 1)/2$.

Proof. Let A be the set guaranteed by Claim 16. As $|A| \leq q$, the third property of Claim 16 implies that

$$\text{CNT}_A(\mathcal{C}_{t-1}, p_{t-1}, y) \geq |A| \cdot w \qquad\qquad \text{for any } y \in \{0, 1\}. \tag{3}$$

If $|A| = 1$, then $\gcd(A) = g_t$ implies that $A = \{g_t\}$. In such a case, the first condition of the lemma holds.

Hence, below we assume $|A| \geq 2$ and we will partition A into A_0 and A_1, satisfying the second property of the lemma. For any cluster $y \in \{0, 1\}$, let

$$A'_y \triangleq \{ i \in A \ : \ \text{CNT}_i(\mathcal{C}_{t-1}, p_{t-1}, y) \geq (w - 1)/2 \}.$$

By the second property of Claim 16, $\text{CNT}_i(\mathcal{C}_{t-1}) \geq w - 1$ for any $i \in A$, and thus $A'_0 \cup A'_1 = A$. By (3) both A'_0 and A'_1 are non-empty. Thus, they satisfy all conditions of the second lemma property except being possibly non-disjoint. To fix it, we consider three cases.

- If $A'_0 \setminus A'_1 \neq \emptyset$, then we set $A_0 = A'_0 \setminus A'_1$ and $A_1 = A'_1$.
- If $A'_1 \setminus A'_0 \neq \emptyset$, then we set $A_1 = A'_1 \setminus A'_0$ and $A_0 = A'_0$.
- If $A'_0 \setminus A_1 = A'_1 \setminus A'_0 = \emptyset$, then $A'_0 = A'_1 = A$. As $|A| \geq 2$, we simply take any element $j \in A$, and set $A_0 = \{j\}$ and $A_1 = A \setminus \{j\}$.   ◁

▶ **Lemma 8.** *For any regular and not g-updating step t of $\mathcal{E}_1$, $\text{ICB}^{ss}(t) = O(q^4)$.*

Proof. Recall that $\text{ICB}^{ss}(t) = 1 + \text{dist}(p_{t-1}, p_t^*)$, and p_t^* is the partition from $\mathcal{P}(\mathcal{C}_t)$ closest to p_{t-1}. Thus, our goal is to construct a partition $p \in P(\mathcal{C}_t)$ (on the basis of p_{t-1}), such that $\text{dist}(p_{t-1}, p) = O(q^4)$.

Recall that c_x^t and c_y^t are the components merged in step t. We may assume that partition p_{t-1} maps c_x^t and c_y^t to two different clusters, as otherwise $p_{t-1} \in P(\mathcal{C}_t)$, and the lemma follows by simply taking $p = p_{t-1}$.

By the lemma assumption, $g_t = g_{t-1}$. As step t is regular, the size of either c_x^t or c_y^t (or both) is from $\langle g_{t-1} \rangle = \langle g_t \rangle$. Without loss of generality, we assume that $\text{SIZE}(c_x^t) \in \langle g_t \rangle$ and let $x = \text{SIZE}(c_x^t)$. As $x \in \langle g_t \rangle$, we have $g_t < \infty$ and $g_t \mid x$. Without loss of generality, we may assume that $p_{t-1}(c_x^t) = 1$, i.e., c_x^t is in cluster 1 at the beginning of step t.

We will create p from p_{t-1} by moving components between clusters so that c_x^t changes its cluster, c_y^t does not change its cluster, and in total, at most $O(q^4)$ elements change their clusters. This will ensure that $p \in \mathcal{P}(\mathcal{C}_t)$ and $\text{dist}(p_{t-1}, p) = O(q^4)$.

Assume first that p_{t-1} maps at least $x/g_t + 1$ components of size g_t to cluster 0 (i.e., $\text{CNT}_{g_t}(\mathcal{C}_{t-1}, p_{t-1}, 0) \geq x/g_t + 1$). At least x/g_t of these components are different than c_y^t, and thus, we may simply swap c_x^t with them, at a total cost of $2x \leq 2q$.

Hence, in the following, we assume that $\text{CNT}_{g_t}(\mathcal{C}_{t-1}, p_{t-1}, 0) < x/g_t + 1$. This implies $\text{CNT}_{g_t}(\mathcal{C}_{t-1}, p_{t-1}, 0) < q + 1 \leq w$. As the first property of Claim 17 is false, the second one must hold. That is, there exist two disjoint, non-empty sets $A_0, A_1 \subseteq [q]$, such that $\gcd(A_0 \uplus A_1) = g_t$. Furthermore, for any $y \in \{0, 1\}$ and $i \in A_y$, it holds that $\text{CNT}_i(\mathcal{C}_{t-1}, p_{t-1}, y) \geq (w - 1)/2$. By Claim 14 applied to sets A_0 and A_1, there exist non-negative integers r_i, such that

$$\textstyle\sum_{i \in A_0} r_i \cdot i = g_t + \sum_{i \in A_1} r_i \cdot i.$$

and $\sum_{i \in A_0} r_i \cdot i \leq 3 \cdot (|A_0| + |A_1|) \cdot q^2 \leq 3 \cdot q^3$. This also implies that $r_i \leq 3 \cdot q^3$ for any $i \in A_0 \uplus A_1$. Multiplying both sides by x/g_t, we obtain

$$\sum_{i \in A_0} \frac{x \cdot r_i}{g_t} \cdot i = x + \sum_{i \in A_1} \frac{x \cdot r_i}{g_t} \cdot i. \tag{4}$$

We create p from p_{t-1} by executing the following actions:

- For any $i \in A_0$, move $x \cdot r_i/g_t$ components of size i (other than c_y^t) from cluster 0 to cluster 1.
- For any $i \in A_1$, move $x \cdot r_i/g_t$ components of size i (other than c_x^t) from cluster 1 to cluster 0.

- Move component c_x^t from cluster 1 to cluster 0.

We observe that these actions are feasible: For any $i \in A_0$, we have $x \cdot r_i/g_t \leq q \cdot r_i \leq 3 \cdot q^4 \leq (w-1)/2 - 1$, so cluster 0 contains an appropriate number of components of size i (different from c_y^t). Analogously, for any $i \in A_1$, cluster 1 contains an appropriate number of components of size i (different from c_x^t). Next, the resulting partition p is $\mathcal{C}_t$-preserving as both c_x^t and c_y^t end up in cluster 0, and (4) ensures that the total number of elements in each cluster remains unchanged.

Finally, the number of elements that change their cluster is

$$\text{dist}(p_{t-1}, p) = 2 \sum_{i \in A_0} \frac{x \cdot r_i}{g_t} \cdot i \leq 2 \cdot q \cdot \sum_{i \in A_0} r_i \cdot i \leq 6 \cdot q^4. \qquad \blacktriangleleft$$

5.4 Bounding the Number of Irregular Merges

To bound the number of irregular steps, we trace the evolution of components. Recall that $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \{c_z^t\} \setminus \{c_x^t, c_y^t\}$, i.e., components c_x^t and c_y^t are merged in step t into component c_z^t. We say that components c_x^t and c_y^t are *destroyed* in step t and component c_z^t is *created* in step t. We extend these notions also to the (singleton) components of $\mathcal{C}_0$, where we say that they are created in step 0, and to components of $\mathcal{C}_T$, where we say that they are destroyed in step $T+1$.

We now fix a small component c created at time a and destroyed at time b. Note that $0 \leq a < b \leq T+1$. By Lemma 6, $\langle g_{b-1} \rangle \subseteq \langle g_a \rangle$. We say that the component c is

- *typical* if $\text{SIZE}(c) \in \langle g_{b-1} \rangle$,
- *mixed* if $\text{SIZE}(c) \in \langle g_a \rangle \setminus \langle g_{b-1} \rangle$,
- *atypical* if $\text{SIZE}(c) \notin \langle g_a \rangle$.

That is, each component is either large, typical, atypical, or mixed. In particular, in a regular merge, at least one of the merged components is typical.

$\triangleright$ **Claim 18.** Assume step t is not g-updating and both c_x^t and c_y^t are typical. Then, c_y^t is not atypical.

Proof. As components are typical, $\text{SIZE}(c_x^t) \in \langle g_{t-1} \rangle$ and $\text{SIZE}(c_y^t) \in \langle g_{t-1} \rangle$, and therefore $g_{t-1} \mid \text{SIZE}(c_x^t)$ and $g_{t-1} \mid \text{SIZE}(c_y^t)$. As $\text{SIZE}(c_z^t) = \text{SIZE}(c_x^t) + \text{SIZE}(c_y^t)$, we have $g_{t-1} \mid \text{SIZE}(c_z^t)$. Finally, as step t is not g-updating, $g_t = g_{t-1}$, and hence $g_t \mid \text{SIZE}(c_z^t)$. If c_z^t is large then the lemma follows immediately. If c_z^t is small, then we have $\text{SIZE}(c_z^t) \in \langle g_t \rangle$, and thus c_z^t cannot be atypical. $\triangleleft$

Merge Forest. It is convenient to consider the following *merge forest* $\mathcal{F}$, whose nodes correspond to all components created within $\mathcal{E}_1$. We connect these nodes by edges in a natural manner: the leaves of $\mathcal{F}$ correspond to initial singleton components of $\mathcal{C}_0$, and each non-leaf node of $\mathcal{F}$ corresponds to a component created by merging its children components. We say that the node of $\mathcal{F}$ is large/typical/atypical/mixed if the corresponding component is of such type.

Types of Irregular Merges.   To upper-bound the number of irregular merges, we subdivide them into three types.

- *All-large irregular merges*: both merged components are large (and the resulting component is clearly large as well).
- *Mixed-resulting irregular merges*: the created component is mixed.
- *Ordinary irregular merges*: all other irregular merges.

We bound the number of these merges separately in the following three lemmas.

▷ **Claim 19.** $\mathcal{F}$ contains at most n/q large nodes whose both children are also large.

Proof. Let L be the set of large nodes of $\mathcal{F}$. Clearly, L is "upward-closed", i.e., if L contains a node, then it also contains its parent. Let $\mathcal{F}_L$ be the sub-forest of $\mathcal{F}$ induced by nodes from L. We partition L into three sets: L_0, L_1 and L_2, where a component from L_i has exactly i children in $\mathcal{F}_L$. We need to show that $|L_2| \leq n/q$.

As L_2 contains the branching internal nodes of $\mathcal{F}_L$ and L_0 contains the leaves of $\mathcal{F}_L$, we have $|L_2| < |L_0|$. All components from L_0 are large, i.e., each of them consists of at least $q + 1$ nodes. Fix any two components from L_0. As they are leaves of $\mathcal{F}_L$, they are not in the ancestor-descendant relation in $\mathcal{F}_L$ (and not in $\mathcal{F}$), and hence the sets of their elements are disjoint. Thus, all components of L_0 are disjoint, which implies $|L_0| \cdot (q + 1) \leq n$. Summing up, $|L_2| < |L_0| \leq n/(q + 1)$.   ◁

▷ **Claim 20.** $\mathcal{F}$ contains at most $q \cdot w$ mixed nodes.

Proof. Fix any mixed node corresponding to component c that is created at step a and destroyed at step $b > a$. By Lemma 6, $\langle g_a \rangle \supseteq \langle g_{a+1} \rangle \supseteq \ldots \supseteq \langle g_{b-1} \rangle$. As $\textsc{size}(c) \in \langle g_a \rangle$ and $\textsc{size}(c) \notin \langle g_{b-1} \rangle$, there exists a step $t \in [a + 1, b - 1]$, such that $\textsc{size}(c) \in \langle g_{t-1} \rangle \setminus \langle g_t \rangle$. We say that component c is t-*mixed*.

We now fix a step t and show that the number of t-mixed components is at most $w \cdot |\langle g_{t-1} \rangle \setminus \langle g_t \rangle|$. Fix $j \in \langle g_{t-1} \rangle \setminus \langle g_t \rangle$. We show that at step t, the number of components of size j is at most w. Suppose for a contradiction that $\textsc{cnt}_j(\mathcal{C}_t) \geq w$. Then, $j \in B_t$. As $j \in \langle g_{t-1} \rangle$, we have $j \in \langle g_{t-1} \rangle \cap B_t$. On the other hand, $g_t = \gcd(\langle g_{t-1} \rangle \cap B_t)$, and thus $g_t \mid j$. However, as $j \in [q]$, we would then have $j \in \langle g_t \rangle$, a contradiction.

As any mixed node is t-mixed for a step $t \in [T]$, the total number of all mixed nodes is at most $\sum_{t \in [T]} w \cdot |\langle g_{t-1} \rangle \setminus \langle g_t \rangle| \leq w \cdot |\langle g_0 \rangle \setminus \langle g_T \rangle| \leq w \cdot |\langle g_0 \rangle| = w \cdot q$.   ◁

It remains to bound the number of ordinary irregular merges. To this end, we define the following amounts (for any step $t \geq 0$).

- a_t is the number of components in $\mathcal{C}_t$ that are atypical or mixed.
- I_t is the number of ordinary irregular merges in steps $1, 2, \ldots, t$.

▷ **Claim 21.**   For any step $t \geq 0$, it holds that $a_t = O(q \cdot w)$ and $I_t = O(q \cdot w)$.

Proof. Let R_t be the number of (regular or irregular) merges in steps $1, 2, \ldots, t$ in which c_z^t (the created component) is mixed. Let U_t be the number of g-updating steps among steps $1, 2, \ldots, t$. We inductively show that

$$a_t + I_t \leq 2 \cdot (R_t + U_t). \tag{5}$$

The lemma will follow as $R_t = O(q \cdot w)$ by Claim 20 and $U_t \leq 1 + \log q$ by Lemma 6.

The base case ($t = 0$) holds as both sides are then trivially equal to 0. We assume that (5) holds for step $t - 1$ and we show it for step t. Let $\Delta a = a_t - a_{t-1}$; we define ΔI, ΔR, and ΔU analogously. It suffices to show that

$$\Delta a + \Delta I \leq 2 \cdot (\Delta R + \Delta U). \tag{6}$$

Observe that $\Delta a \in \{-2, -1, 0, 1\}$ and $\Delta I \in \{0, 1\}$. Thus, if either $\Delta R \geq 1$ or $\Delta U \geq 1$, then (6) holds trivially. This happens when step t is g-updating or c_z^t is mixed.

Thus, in the remaining part of the proof, we assume that step t is not g-updating and the c_z^t is not mixed. In such case, $\Delta R = 0$ and $\Delta U = 0$, and thus it remains to show that

$$\Delta a + \Delta I \leq 0. \tag{7}$$

We consider a few cases depending on the merge type at step t. As c_z^t is not mixed, the merge cannot be mixed-resulting irregular.

- Merge is regular, i.e., at least one of the merged components, say c_x^t, is typical. Then, $\Delta I = 0$, and we will show that $\Delta a \leq 0$.
  - If c_y^t is also typical, then Claim 18 implies that c_z^t cannot be atypical. As we assumed c_z^t is not mixed, it must be either large or typical. Hence, $\Delta a = 0$.
  - If c_y^t is large, then c_z^t is large as well, and then $\Delta a = 0$.
  - If c_y^t is atypical or mixed, then $\Delta a \leq 0$.
- Merge is all-large irregular. Then, $\Delta I = 0$ and $\Delta a = 0$.
- Merge is ordinary irregular, i.e., $\Delta I = 1$. We will show that $\Delta a \leq -1$. If both c_x^t and c_y^t are atypical or mixed, then we immediately obtain $\Delta a \leq -1$. Otherwise, we note that the merge is irregular, and hence neither c_x^t nor c_y^t is typical. Thus, one of them is large and the second one is atypical or mixed. Then, c_z^t is large as well, and thus $\Delta a \leq -1$ as well.

In either case, (7) follows, which concludes the inductive proof. ◁

▶ **Lemma 9.** *There are at most $O(q \cdot w + n/q)$ irregular steps in $\mathcal{E}_1$. Moreover, at any time t of $\mathcal{E}_1$, $\mathrm{CNT}_{[n] \setminus \langle g_t \rangle}(\mathcal{C}_t) = O(q \cdot w + n/q)$.*

Proof. There are at most n/q all-large irregular steps by Claim 19, at most $q \cdot w$ mixed-resulting irregular steps by Claim 20, and $I_T = O(q \cdot w)$ ordinary irregular steps by Claim 21. This shows the first part of the lemma.

For the second part, fix any step t. Observe that components whose sizes are from $[n] \setminus \langle g_t \rangle$ are not typical, i.e., they must be either large, atypical, or mixed. Trivially, there are at most $n/(q+1)$ large components, and the number of atypical and mixed components is $a_t = O(q \cdot w)$ by Claim 21. ◀

6 Analysis: the Second Stage of ICB

▶ **Lemma 3.** $\mathbf{E}[\mathrm{ICB}(\mathcal{E}_2)] = O(n \cdot \log |\mathcal{P}(\mathcal{C}_T)|)$.

Proof. At the beginning of $\mathcal{E}_2$, there are $|\mathcal{P}(\mathcal{C}_T)|$ $\mathcal{C}_T$-preserving partitions. In a step t of $\mathcal{E}_2$, ICB chooses a new partition only when its current partition is not $\mathcal{C}_t$-preserving. In such case, it chooses a new partition p_t uniformly at random from $\mathcal{P}(\mathcal{C}_t)$.

Thus, we may treat the problem as the metrical task system (MTS) on $|\mathcal{P}(\mathcal{C}_T)|$ states, where the adversary makes the states (partitions) forbidden in some specified order. ICB then basically executes (a single phase of) the known randomized algorithm for MTS on a uniform metric [5]. By the result of [5], the expected number of times when ICB is forced to choose a new partition is $O(\log |\mathcal{P}(\mathcal{C}_T)|)$. Whenever that happens, ICB pays at most $n + 1$ (cf. Lemma 1), and thus $\mathbf{E}[\mathrm{ICB}(\mathcal{E}_2)] = (n + 1) \cdot O(\log |\mathcal{P}(\mathcal{C}_T)|)$. ◀

▶ **Lemma 4.** $|\mathcal{P}(\mathcal{C}_T)| = \exp(O(d \cdot \log n + z))$, *where* $z = \mathrm{CNT}_{[n] \setminus \langle g_T \rangle}(\mathcal{C}_T)$.

Proof. Recall that $|\mathcal{P}(\mathcal{C}_T)|$ is the number of ways we can feasibly assign components of $\mathcal{C}_T$ to two clusters at the end of the first stage. We partition components of $\mathcal{C}_T$ into set A containing components of sizes from $\langle g_T \rangle$ and set B containing components of sizes outside $\langle g_T \rangle$, i.e., $|B| = z$. We separately upper-bound the number of ways of assigning components from A and components from B to two clusters.

We assume that the second stage is present as otherwise $|\mathcal{P}(\mathcal{C}_T)| = 0$. Thus, the condition in Line 6 of the algorithm guarantees that $\mathcal{P}(\mathcal{C}_T, g_T, 2d) = \emptyset$, i.e., each feasible mapping has less than $2 \cdot d$ of components from A at least on one side. Thus, the overall number of ways of partitioning sets of A among two clusters is at most

$$\sum_{i=0}^{2d-1} 2 \cdot \binom{|A|}{i} \leq \sum_{i=0}^{2d-1} 2 \cdot \binom{n}{i} \leq 4d \cdot \binom{n}{2d} \leq 4d \cdot \frac{e^{2d} \cdot n^{2d}}{(2d)^{2d}} = \exp(O(d \cdot \log n)).$$

As the components of B can be assigned to two clusters in at most $2^{|B|} = 2^z$ ways, we have $|\mathcal{P}(\mathcal{C}_t)| \leq 2^z \cdot \exp(O(d \cdot \log n))$. ◀

7   Final Remarks

In this paper, we provided the first algorithm for the online bisection problem with the competitive ratio of $o(n^2)$. Extending the result to a more general setting of online balanced graph partitioning (i.e., multiple-cluster case) is an intriguing open problem. We note that our algorithm ICB has non-polynomial running time; we conjecture that without resource augmentation, achieving a subquadratic competitive ratio in polynomial time is not possible.

References

1   Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58(1):193–210, 1999. `doi:10.1006/jcss.1998.1605`.

2   Chen Avin, Marcin Bienkowski, Andreas Loukas, Maciej Pacut, and Stefan Schmid. Dynamic balanced graph partitioning. *SIAM Journal on Discrete Mathematics*, 34(3):1791–1812, 2020. `doi:10.1137/17M1158513`.

3   Chen Avin, Andreas Loukas, Maciej Pacut, and Stefan Schmid. Online balanced repartitioning. In *Proc. 30th Int. Symp. on Distributed Computing (DISC)*, pages 243–256, 2016. `doi:10.1007/978-3-662-53426-7_18`.

4   Marcin Bienkowski, Martin Böhm, Martin Koutecký, Thomas Rothvoß, Jirí Sgall, and Pavel Veselý. Improved analysis of online balanced clustering. In *Proc. 19th Workshop on Approximation and Online Algorithms (WAOA)*, pages 224–233, 2021. `doi:10.1007/978-3-030-92702-8_14`.

5   Alan Borodin, Nati Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992. `doi:10.1145/146585.146588`.

6   Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

7   Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31(4):1090–1118, 2002. `doi:10.1137/S0097539701387660`.

8   Uriel Feige, Robert Krauthgamer, and Kobbi Nissim. Approximating the minimum bisection size. In *Proc. 32nd ACM Symp. on Theory of Computing (STOC)*, pages 530–536, 2000. `doi:10.1145/335305.335370`.

9   Tobias Forner, Harald Räcke, and Stefan Schmid. Online balanced repartitioning of dynamic communication patterns in polynomial time. In *2nd Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 40–54, 2021. `doi:10.1137/1.9781611976489.4`.

10 Monika Henzinger, Stefan Neumann, Harald Räcke, and Stefan Schmid. Tight bounds for online graph partitioning. In *Proc. 32nd ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2799–2818, 2021. `doi:10.1137/1.9781611976465.166`.

11 Monika Henzinger, Stefan Neumann, and Stefan Schmid. Efficient distributed workload (re-)embedding. In *Proc. SIGMETRICS/Performance Joint Int. Conf. on Measurement and Modeling of Computer Systems*, pages 43–44, 2019. `doi:10.1145/3309697.3331503`.

12 Robert Krauthgamer. Minimum bisection. In *Encyclopedia of Algorithms*, pages 1294–1297. Springer, 2016. `doi:10.1007/978-1-4939-2864-4_231`.

13 Robert Krauthgamer and Uriel Feige. A polylogarithmic approximation of the minimum bisection. *SIAM Review*, 48(1):99–130, 2006. `doi:10.1137/050640904`.

14 Bohdan S. Majewski and George Havas. The complexity of greatest common divisor computations. In *1st Symposium on Algorithmic Number Theory (ANTS-I)*, pages 184–193, 1994. `doi:10.1007/3-540-58691-1_56`.

15 Maciej Pacut, Mahmoud Parham, and Stefan Schmid. Brief announcement: Deterministic lower bound for dynamic balanced graph partitioning. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 461–463, 2020. `doi:10.1145/3382734.3405696`.

16 Maciej Pacut, Mahmoud Parham, and Stefan Schmid. Optimal online balanced graph partitioning. In *Proc. 40th IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1–9, 2021. `doi:10.1109/INFOCOM42981.2021.9488824`.

17 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 255–264, 2008. `doi:10.1145/1374376.1374415`.

18 Harald Räcke, Stefan Schmid, and Ruslan Zabrodin. Approximate dynamic balanced graph partitioning. In *Proc. 34th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 401–409, 2022. `doi:10.1145/3490148.3538563`.

19 Rajmohan Rajaraman and Omer Wasim. Improved bounds for online balanced graph repartitioning. In *Proc. 30th European Symp. on Algorithms (ESA)*, pages 83:1–83:15, 2022. `doi:10.4230/LIPIcs.ESA.2022.83`.

20 Huzur Saran and Vijay V. Vazirani. Finding k cuts within twice the optimal. *SIAM Journal on Computing*, 24(1):101–108, 1995. `doi:10.1137/S0097539792251730`.

21 Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. `doi:10.1145/2786.2793`.

A   Proof of Claim 14

▷ **Claim 14.** Let $A = \{a_1, a_2, \ldots, a_k\} \subset \mathbb{N}_{>0}$ and $B = \{b_1, b_2, \ldots, b_\ell\} \subset \mathbb{N}_{>0}$ be two non-empty and disjoint sets of positive integers. Let $g = \gcd(A \uplus B)$ and $H = \max(A \uplus B)$. Then, there exist non-negative integers $r_1, r_2, \ldots r_k, s_1, s_2, \ldots s_\ell$, such that

$$\sum_{i=1}^{k} r_i \cdot a_i = g + \sum_{i=1}^{\ell} s_i \cdot b_i.$$

Moreover, $\sum_{i=1}^{k} r_i \cdot a_i \leq 3 \cdot (k + \ell) \cdot H^2$.

Proof. By the bound given by Majewski and Havas [14], there exist coefficients $\tilde{r}_1, \tilde{r}_2, \ldots \tilde{r}_k$ and $\tilde{s}_1, \tilde{s}_2, \ldots \tilde{s}_\ell$, such that $|\tilde{r}_i| \leq \max\{H/2g, 1\} \leq H$ and $|\tilde{s}_i| \leq \max\{H/2g, 1\} \leq H$ for any i, and

$$\sum_{i=1}^{k} \tilde{r}_i \cdot a_i = g + \sum_{i=1}^{\ell} \tilde{s}_i \cdot b_i. \tag{8}$$

However, these coefficients are not necessarily non-negative. To fix it, let

$$r_1 = \tilde{r}_1 + \lceil H/b_1 \rceil \cdot b_1 + \sum_{i=1}^{\ell} \lceil H/a_1 \rceil \cdot b_i,$$

$$r_i = \tilde{r}_i + \lceil H/b_1 \rceil \cdot b_1 \qquad\qquad \text{for } i \in \{2, \ldots, k\},$$

$$s_1 = \tilde{s}_1 + \lceil H/a_1 \rceil \cdot a_1 + \sum_{i=1}^{k} \lceil H/b_1 \rceil \cdot a_i,$$

$$s_i = \tilde{s}_i + \lceil H/a_1 \rceil \cdot a_1 \qquad\qquad \text{for } i \in \{2, \ldots, \ell\}.$$

We argue that the values above satisfy the lemma conditions. First, note that $r_i \geq \tilde{r}_i + H$ and $s_i \geq \tilde{s}_i + H$, and thus r_i and s_i are non-negative for every i.

To show that r_i's and s_i's satisfy the identity required by the lemma, we analyze the following term

$$
\begin{aligned}
\sum_{i=1}^{k}(r_i - \tilde{r}_i) \cdot a_i &= (r_1 - \tilde{r}_1) \cdot a_i + \sum_{i=2}^{k}(r_i - \tilde{r}_i) \cdot a_i \\
&= \left(\lceil H/b_1 \rceil \cdot b_1 + \sum_{i=1}^{\ell} \lceil H/a_1 \rceil \cdot b_i \right) \cdot a_1 + \sum_{i=2}^{k} \lceil H/b_1 \rceil \cdot b_1 \cdot a_i \\
&= \lceil H/a_1 \rceil \cdot a_1 \cdot \sum_{i=1}^{\ell} b_i + \lceil H/b_1 \rceil \cdot b_1 \cdot \sum_{i=1}^{k} a_i.
\end{aligned}
\tag{9}
$$

In the same way, but swapping the roles of a's and r's with b's and s's, we obtain

$$\sum_{i=1}^{\ell}(s_i - \tilde{s}_i) \cdot b_i = \lceil H/b_1 \rceil \cdot b_1 \cdot \sum_{i=1}^{k} a_i + \lceil H/a_1 \rceil \cdot a_1 \cdot \sum_{i=1}^{\ell} b_i. \tag{10}$$

Therefore, (9) and (10) together imply $\sum_{i=1}^{k}(r_i - \tilde{r}_i) \cdot a_i = \sum_{i=1}^{\ell}(s_i - \tilde{s}_i) \cdot b_i$. Combining this relation with (8), immediately yields

$$\sum_{i=1}^{k} r_i \cdot a_i = g + \sum_{i=1}^{\ell} s_i \cdot b_i.$$

It remains to upper-bound $\sum_{i=1}^{k} r_i \cdot a_i$. Note that for any $z \leq H$, it holds that $\lceil H/z \rceil \cdot z < (H/z + 1) \cdot z \leq H + z \leq 2H$. Hence, using $\tilde{r}_i \leq H$ (for every i) and (9), we obtain

$$
\begin{aligned}
\sum_{i=1}^{k} r_i \cdot a_i &= \sum_{i=1}^{k} \tilde{r}_i \cdot a_i + \sum_{i=1}^{k}(r_i - \tilde{r}_i) \cdot a_i \\
&\leq H \cdot \sum_{i=1}^{k} a_i + \lceil H/a_1 \rceil \cdot a_1 \cdot \sum_{i=1}^{\ell} b_i + \lceil H/b_1 \rceil \cdot b_1 \cdot \sum_{i=1}^{k} a_i \\
&\leq H \cdot \left(3 \cdot \sum_{i=1}^{k} a_i + 2 \cdot \sum_{i=1}^{\ell} b_i \right) \\
&\leq (3 \cdot k + 2 \cdot \ell) \cdot H^2. \qquad\qquad\qquad\qquad\qquad\qquad\qquad \lhd
\end{aligned}
$$