Improved Analysis of Online Balanced Clustering[*]

Marcin Bienkowski[1], Martin Böhm[1], Martin Koutecký[2], Thomas Rothvoß[3],
Jiří Sgall[2], and Pavel Veselý[2]

[1] Institute of Computer Science, University of Wrocław, Poland
[2] Computer Science Institute of Charles University, Faculty of Mathematics and Physics, Prague, Czechia
[3] University of Washington, Seattle, United States

Abstract. In the online balanced graph repartitioning problem, one has to maintain a clustering of n nodes into ℓ clusters, each having $k = n/\ell$ nodes. During runtime, an online algorithm is given a stream of communication requests between pairs of nodes: an inter-cluster communication costs one unit, while the intra-cluster communication is free. An algorithm can change the clustering, paying unit cost for each moved node. This natural problem admits a simple $O(\ell^2 \cdot k^2)$-competitive algorithm Comp, whose performance is far apart from the best known lower bound of $\Omega(\ell \cdot k)$. One of open questions is whether the dependency on ℓ can be made linear; this question is of practical importance as in the typical datacenter application where virtual machines are clustered on physical servers, ℓ is of several orders of magnitude larger than k. We answer this question affirmatively, proving that a simple modification of Comp is $(\ell \cdot 2^{O(k)})$-competitive.

On the technical level, we achieve our bound by translating the problem to a system of linear integer equations and using Graver bases to show the existence of a "small" solution.

Keywords: Clustering · Graph partitioning · Balanced partitioning · Graver basis · Online algorithms · Competitive analysis

1 Introduction

We study the *online balanced graph repartitioning problem*, introduced by Avin et al. [3]. In this problem, an algorithm has to maintain a time-varying partition of n nodes into ℓ clusters, each having $k = n/\ell$ nodes. An algorithm is given an online stream of communication requests, each involving a pair of nodes. A communication between a pair of nodes from the same cluster is free, while inter-cluster communication incurs unit cost. In response, an algorithm may

change the mapping of nodes to clusters, also paying a unit cost for changing a cluster of a single node. After remapping, each cluster has to contain k nodes again.

We focus on an online scenario, where an (online) algorithm has to make irrevocable remapping decisions after each communication request without the knowledge of the future. The problem can be seen as a dynamic (and online) counterpart of a so-called ℓ-balanced graph partitioning problem [1], where the goal is to partition the graph into ℓ equal-size parts to minimize the total weight of edges in the cut. In particular, $\ell = 2$ corresponds to the well-studied graph bisection problem [9,13].

A main practical motivation originates from server virtualization in datacenters. There, nodes correspond to n virtual machines run on ℓ physical ones. Each physical machine (a cluster) has the capacity for accommodating k virtual machines. Communication requests are induced by distributed applications running in the datacenter. While communication within a physical machine is practically free, the inter-cluster communication (between different physical machines) generates considerable load and affects the overall running time (see, e.g., [5]). Due to the current capabilities of modern architectures, migrating a virtual machine to another physical machine (remapping of a node) is possible, but it incurs a certain load.[4]

To evaluate the performance of an online algorithm ALG, we use a standard notion of competitive ratio [4] which is the supremum over all possible inputs of ALG-to-OPT cost, where OPT denotes the optimum offline solution for the problem.

1.1   Component-Based Approach

Our contribution is related to the following natural algorithm (henceforth called COMP) proposed by Avin et al. [3]. We define it in detail in section 3; here, we give its informal description. COMP operates in phases, in each phase keeping track of components of nodes that communicated in this phase (i.e., connected components of a graph whose edges are communication requests). COMP always keeps all nodes of a given component in a single cluster; we call this property *component invariant.* When components are modified, to maintain the invariant, some nodes may have to change their clusters; the associated cost can be trivially bounded by n. As this changes the mapping of nodes to clusters, we

---

[4] In the reality, the network load generated by a single request is much smaller than the load of migrating a whole virtual machine. This has been captured by some papers, which assigned cost $\alpha \geq 1$ to the latter event (α is then a parameter of the problem). However, this additional difficulty can be resolved by standard rent-or-buy approaches (reacting only to every α-th request between a given pair of nodes). Therefore, and also to keep the description simple, in this paper, we assume that $\alpha = 1$.

call it *remapping event*. If such remapping does not exist[5], the current phase terminates, and it is possible to show that OPT paid at least 1 in this phase. The number of remapping events is equal to the number of times connected components are modified, which can be upper-bounded by $n - 1$. Thus, the overall cost of COMP in a single phase is $O(n^2) = O(\ell^2 \cdot k^2)$, while that of OPT is at least 1. This shows that COMP is $O(\ell^2 \cdot k^2)$-competitive.

1.2   Related Work

Perhaps surprisingly, no better algorithm than COMP (even randomized one) is known for the general case. Some improvements were, however, given for specific values of k and ℓ.

The dependency of the competitive ratio on k is at least linear for deterministic algorithms: a lower bound of $\Omega(k)$ follows by the reduction from online paging [15] and holds already for $\ell = 2$ clusters [2]. For $k = 2$, $O(1)$-competitive (deterministic) algorithms are known: a 7-competitive algorithm was given by Avin et al. [3,2] and was later improved to a 6-competitive one by Pacut et al. [12]. However, already for $k \geq 3$, the competitive ratio of any deterministic algorithm cannot be better than $\Omega(k \cdot \ell)$ [11,12]. For the special case of $k = 3$, Pacut et al. [12] showed that a variant of COMP is $O(\ell)$-competitive. The lack of progress towards improving the $O(\ell^2 \cdot k^2)$ upper bound for the general case motivated the research of simplified variants.

Henzinger et al. [8] initiated the study of a so-called *learning variant*, where there exists a fixed partitioning (unknown to an algorithm) of nodes into clusters, and the communication requests are *consistent* with this mapping, i.e., all requests are given between same-cluster node pairs. Hence, the implicit goal of an algorithm is to learn such static mapping. The deterministic lower bound of $\Omega(k \cdot \ell)$ also holds for this variant, and, furthermore, there exists a deterministic algorithm that asymptotically matches this bound [11,12].

Another strand of research focused on a resource-augmented scenario, where each cluster of an online algorithm is able to accommodate $(1 + \epsilon) \cdot k$ nodes (but the online algorithm is still compared to OPT, whose clusters have to keep k nodes each). For $\epsilon > 1$, the first deterministic algorithm was given by Avin et al. [2]; the achieved ratio was $O(k \cdot \log k)$ (for an arbitrary ℓ). Suprisingly, the ratio remains $\Omega(k)$ even for large ϵ (as long as the algorithm cannot keep all nodes in a single cluster) [2].

When these two simplifications are combined (i.e., the learning variant is studied in a resource-augmented scenario), asymptotically optimal results are due to Henzinger et al. [8,7]. They show that for any fixed $\epsilon > 0$, the deterministic ratio is $\Theta(\ell \cdot \log k)$ [8,7] and the randomized ratio is $\Theta(\log \ell + \log k)$ [7]. Furthermore, for $\epsilon > 1$, their deterministic algorithm is $O(\log k)$-competitive [7].

---

[5] Deciding whether such remapping exists is NP-hard. As typical for online algorithms, however, our focus is on studying the disadvantage of not knowing the future rather than on computational complexity.

1.3   Our Contribution

We focus on the general variant of the balanced graph repartitioning problem. We study a variant of COMP (see subsection 1.1) in which *each remapping event is handled in a way minimizing the number of affected clusters.*

   We show that the number of nodes remapped this way is $2^{O(k)}$ (in comparison to the trivial bound of $n = \ell \cdot k$). The resulting bound on the competitive ratio is then $(\ell \cdot k) \cdot 2^{O(k)} = \ell \cdot 2^{O(k)}$, i.e., we replaced the quadratic dependency on ℓ in the competitive ratio by the linear one. We note that the resulting algorithm retains the $O(\ell^2 \cdot k^2)$-competitiveness guarantee of the original COMP algorithm as well.

   Given the lower bound of $\Omega(\ell \cdot k)$ [11,12], the resulting strategy is optimal for a constant k. We also note that for the datacenter application described earlier, k is of several orders of magnitude smaller than ℓ.

   We achieve our bound by translating the remapping event to a system of linear integer equations so that the size of the solution (sum of values of variables) is directly related to the number of affected clusters. Then, we use algebraic tools such as Graver bases to argue that these equations admit a "small" solution.[6]

2   Preliminaries

An offline part of the input is a set of $n = \ell \cdot k$ nodes and an initial valid mapping of these nodes into ℓ clusters. We call a mapping *valid* if each cluster contains exactly k nodes.

   An online part of the input is a set of requests, each being a pair of nodes (u, v). The request incurs cost 1 if u and v are mapped to different clusters. After each request, an online algorithm may modify the current node mapping to a new valid one, paying 1 for each node that changes its cluster.

   For an input I and an algorithm ALG, $\text{ALG}(I)$ denotes its cost on input I, whereas $\text{OPT}(I)$ denotes the optimal cost of an offline solution. ALG is γ-competitive if there exists a constant β such that for any input I, it holds that $\text{ALG}(I) \leq \gamma \cdot \text{OPT}(I) + \beta$. While β has to be independent of the online part of the sequence, it may depend on the offline part, i.e., be a function of parameters ℓ and k.

3   Better Analysis for COMP

Algorithm COMP [3] splits input into phases. In each phase, COMP maintains an auxiliary partition $\mathcal{R}$ of the set of nodes into *components*; initially, each node is in its own singleton component. COMP maintains the following *component invariant*: for each component $S \in \mathcal{R}$, all its nodes are inside the same cluster.

   Assume a request (u, v) arrives. Two cases are possible.

---

[6] One could also bound the size of a solution along the lines of Schrijver [14, Corollary 17.1b], which boils down to a determinant bound, same as our proof. In general, Graver basis elements may be much smaller, but in out specific case, the resulting bound would be asymptotically the same.

– If u and v are within the same component of $\mathcal{R}$, then by the component invariant, they are in the same cluster. COMP serves this request without paying anything, without changing $\mathcal{R}$, and without remapping nodes.
– If u and v are in different components S_a and S_b, COMP merges these components into $S_{ab} = S_a \uplus S_b$ by removing S_a and S_b from $\mathcal{R}$, and adding S_{ab} to $\mathcal{R}$. (By $A \uplus B$ we denote the *disjoint union* of sets A and B.)
  If components S_a and S_b were in different clusters, the resulting component S_{ab} now spans two clusters, which violates the component invariant. To restore it, COMP verifies whether there exists a valid mapping of nodes into clusters preserving the component invariant (also for the new component S_{ab}). In such case, a *remapping event* occurs: among all such mappings, COMP chooses one minimizing the total number of affected clusters. (The original variant of COMP [3] simply chose any such mapping.) If, however, no such valid mapping exists, COMP resets $\mathcal{R}$ to the initial partition in which each node is in its own component and starts a new phase.

Lemma 1. *Assume each remapping event affects at most $f(\ell, k)$ clusters (for some function f). Then* COMP *is $O(\ell \cdot k^2 \cdot f(\ell, k))$-competitive.*

Proof. Fix any input I and split it into phases according to COMP. Each phase (except possibly the last one) terminates because there is no valid node mapping that would preserve the component invariant. That is, for any fixed valid mapping of nodes to clusters, the phase contains an inter-cluster request. Thus, during any phase, either OPT pays at least 1 for node remapping, or its mapping is fixed within phase, and then it pays at least 1 for serving requests.

On the other hand, within each phase, COMP modifies family $\mathcal{R}$ of components at most $n - 1 = \ell \cdot k - 1$ times since the number of components decreases by one with each modification. Each time it happens, it pays 1 for the request, and then, if the remapping event is triggered, it pays additionally at most $k \cdot f(\ell, k)$ as it remaps at most k nodes from each affected cluster. Hence, the overall cost in a single phase is $(\ell \cdot k - 1) \cdot (1 + k \cdot f(\ell, k)) = O(\ell \cdot k^2 \cdot f(\ell, k))$.

The cost of COMP in the last phase is universally bounded by $O(\ell \cdot k^2 \cdot f(\ell, k))$ and in the remaining phases, the COMP-to-OPT cost ratio is at most $O(\ell \cdot k^2 \cdot f(\ell, k))$, which concludes the proof.   □

The trivial upper bound on $f(\ell, k)$ is ℓ, which together with Lemma 1 yields the already known bound of $O(\ell^2 \cdot k^2)$ [3]. In the remaining part, we show that if COMP tries to minimize the number of affected clusters for each remapping event, then $f(\ell, k)$ can be upper-bounded by $2^{O(k)}$. Thus, our analysis beats the simple approach when the number of clusters ℓ is much larger than the cluster capacity k. This ratio is also optimal for constant k as the lower bound on the competitive ratio is $\Omega(\ell \cdot k)$ [11,12].

3.1   Analyzing a Remapping Event

Recall that we want to analyze a remapping event, where we have a given valid mapping of nodes to clusters and a family of components $\mathcal{R}$ satisfying the component invariant. COMP merges two components S_a and S_b into one, and the

resulting component $S_{ab} = S_a \uplus S_b$ spans two clusters. As we assume that it is possible to remap nodes to satisfy the component invariant, the size of S_{ab} is at most k.

We first express our setup in the form of a system of linear integer equations. The definition below assumes that component invariant holds, i.e., each component is entirely contained in some cluster.

Definition 1 (Cluster configuration). *A* configuration *of a cluster C is a vector $\mathbf{c} = (c_1, \ldots, c_k)$ where $c_i \geq 0$ is the number of components of size i in C.*

We denote the set of all possible cluster configurations by $\mathcal{C}$, i.e., $\mathcal{C}$ contains all possible k-dimensional vectors $\mathbf{c} = (c_1, \ldots, c_k)$ such that $\sum_{i=1}^{k} i \cdot c_i = k$. For succinctness, we use $\mathrm{nd}(\mathbf{c}) = \sum_{i=1}^{k} i \cdot c_i$. The number of configurations is equal to the partition function $\pi(k)$, which denotes the number of possibilities how k can be expressed as a sum of a multiset of non-negative integers. From the known bounds on π, we get $|\mathcal{C}| = \pi(k) \leq 2^{O(\sqrt{k})}$ [6].

We take two clusters that contained components S_a and S_b, respectively, and we virtually replace them by a *pseudo-cluster* $\hat{C}$ that contains all their components (including S_{ab} and excluding S_a and S_b). Let $\hat{\mathbf{c}}$ be the configuration of this pseudo-cluster; note that $\mathrm{nd}(\hat{\mathbf{c}}) = 2k$, and hence $\hat{\mathbf{c}} \notin \mathcal{C}$. We define the extended set of configurations $\mathcal{C}^{\mathrm{ext}} = \mathcal{C} \uplus \{\hat{\mathbf{c}}\}$.

Let $\mathbf{x}$ be a $|\mathcal{C}^{\mathrm{ext}}|$-dimensional vector, indexed by possible configurations from $\mathcal{C}^{\mathrm{ext}}$, such that, for any configuration $\mathbf{c} \in \mathcal{C}^{\mathrm{ext}}$, $x_{\mathbf{c}}$ is the number of clusters with configuration $\mathbf{c}$ before remapping event. Note that $\mathbf{x} \geq \mathbf{0}$, $x_{\hat{\mathbf{c}}} = 1$ and $\|\mathbf{x}\|_1 = \sum_{\mathbf{c} \in \mathcal{C}^{\mathrm{ext}}} x_{\mathbf{c}} = \ell - 1$. Let

$$\mathbf{u} = \sum_{\mathbf{c} \in \mathcal{C}^{\mathrm{ext}}} x_{\mathbf{c}} \cdot \mathbf{c} \, . \tag{1}$$

That is, $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, where u_i is the total number of components of size i (in all clusters). Clearly, $\mathrm{nd}(\mathbf{u}) = (\ell - 2) \cdot k + 2 \cdot k = \ell \cdot k$. We rewrite (1) as

$$\mathbf{u} = A\mathbf{x} \, , \tag{2}$$

where A is the matrix with k rows and $|\mathcal{C}^{\mathrm{ext}}|$ columns. Its columns are equal to vectors of configurations from $\mathcal{C}^{\mathrm{ext}}$.

As $\mathbf{x}$ describes the current state of the clusters, in the following, we focus on finding an appropriate vector $\mathbf{y}$ describing a *target* state of the clusters, i.e., their state after the remapping event takes place.

Definition 2. *An integer vector $\mathbf{y}$ is a* valid target vector *if it is $|\mathcal{C}^{\mathrm{ext}}|$-dimensional and satisfies $\mathbf{y} \geq 0$, $A\mathbf{y} = \mathbf{u}$ and $y_{\hat{\mathbf{c}}} = 0$.*

Lemma 2. *For any valid target vector $\mathbf{y}$, it holds that $\|\mathbf{y}\|_1 = \ell$.*

Proof. Let $\ell' = \|\mathbf{y}\|_1$. Then $\mathbf{y} = \sum_{i=1}^{\ell'} \mathbf{y}^i$, where $\mathbf{y}^i$ is equal to 1 for some configuration $\mathbf{c} \neq \hat{\mathbf{c}}$ and 0 everywhere else. As $\mathbf{u} = A\mathbf{y} = \sum_{i=1}^{\ell'} A\mathbf{y}^i$, we obtain

$\ell \cdot k = \mathrm{nd}(\mathbf{u}) = \sum_{i=1}^{\ell'} \mathrm{nd}(A\mathbf{y}^i)$. For any i, vector $A\mathbf{y}^i$ is a single column of A corresponding to a configuration $\mathbf{c} \neq \hat{\mathbf{c}}$, and thus $\mathrm{nd}(A\mathbf{y}^i) = k$. This implies that $\ell' = \ell$, which concludes the proof. $\qquad\square$

Lemma 3. *There exist a valid target vector* $\mathbf{y}$.

Proof. After the remapping event takes place, it is possible to map nodes to different clusters so that the component invariant holds (after merging S_a and S_b into S_{ab}), i.e., each component is entirely contained in some cluster (not in the pseudo-cluster). Thus, each cluster has a well-defined configuration in $\mathcal{C}$, and $y_{\mathbf{c}}$ is simply the number of clusters with configuration $\mathbf{c}$ after remapping. $\qquad\square$

Lemma 4. *Fix a valid target vector* $\mathbf{y}$. *Then, there exists a node remapping that affects* $(1/2) \cdot \|\mathbf{x} - \mathbf{y}\|_1 + 1/2$ *clusters.*

Proof. We define vector $\tilde{\mathbf{x}}$, such that $\tilde{x}_{\mathbf{c}} = x_{\mathbf{c}}$ for $\mathbf{c} \neq \hat{\mathbf{c}}$, and $\tilde{x}_{\hat{\mathbf{c}}} = 2$. Hence $\|\tilde{\mathbf{x}}\|_1 = \ell$. By Lemma 2, $\|\mathbf{y}\|_1 = \ell$ as well. For $\mathbf{c} \in \mathcal{C}$, the value of $\tilde{x}_{\mathbf{c}}$ denotes how many clusters have configuration $\mathbf{c}$, with $\tilde{x}_{\hat{\mathbf{c}}} = 2$ simply denoting that there are two clusters whose configuration is not equal to any configuration from $\mathcal{C}$.

Now, for any configuration $\mathbf{c} \in \mathcal{C}^{\mathrm{ext}}$, we fix $\min\{\tilde{x}_{\mathbf{c}}, y_{\mathbf{c}}\}$ clusters with configuration $\mathbf{c}$. Our remapping does not touch these clusters and there exists a straightforward node remapping which involves only the remaining clusters. Their number is equal to

$$(1/2) \cdot \sum_{\mathbf{c} \in \mathcal{C}^{\mathrm{ext}}} |\tilde{x}_{\mathbf{c}} - y_{\mathbf{c}}| = 1 + (1/2) \cdot \sum_{\mathbf{c} \in \mathcal{C}} |\tilde{x}_{\mathbf{c}} - y_{\mathbf{c}}| = 1 + (1/2) \cdot \sum_{\mathbf{c} \in \mathcal{C}} |x_{\mathbf{c}} - y_{\mathbf{c}}|$$
$$= (1/2) + (1/2) \cdot \sum_{\mathbf{c} \in \mathcal{C}^{\mathrm{ext}}} |x_{\mathbf{c}} - y_{\mathbf{c}}|$$
$$= (1/2) + (1/2) \cdot \|\mathbf{x} - \mathbf{y}\|_1,$$

and thus the lemma follows. $\qquad\square$

We note that a valid target vector $\mathbf{y}$ guaranteed by Lemma 3 may be completely different from vector $\mathbf{x}$ describing the current clustering, and thus it is possible that $\|\mathbf{x} - \mathbf{y}\|_1 = \Omega(\ell)$. We however show that on the basis of $\mathbf{y}$, we may find a valid target vector $\mathbf{y}'$, such that $\|\mathbf{x} - \mathbf{y}'\|_1$ is small, i.e., at most $2^{O(k)}$.

3.2 Using Graver Basis

Lemma 3 guarantees the existence of vector $\mathbf{z} = \mathbf{x} - \mathbf{y}$, encoding the reorganization of the clusters. We already know that $A\mathbf{z} = \mathbf{0}$ must hold, but there are other useful properties as well; for instance, if $z_{\mathbf{c}} > 0$ for a configuration $\mathbf{c}$, then $z_{\mathbf{c}} \leq x_{\mathbf{c}}$ (i.e. the corresponding reorganization does not try to remove more clusters of configuration $\mathbf{c}$ than $x_{\mathbf{c}}$). Our goal is to find another vector $\mathbf{w}$ that also encodes the reorganization and $\|\mathbf{w}\|_1$ is small.

The necessary condition for $\mathbf{w}$ is that it satisfies $A\mathbf{w} = \mathbf{0}$, and thus we study properties of matrix A, defined by (2), in particular its Graver basis. For an introduction to Graver bases, we refer the interested reader to a book by Onn [10].

Definition 3 (Sign-compatibility). *Given two vectors $\mathbf{a}$ and $\mathbf{b}$ of the same length, we say that they are* sign-compatible *if for each coordinate i the sign of a_i is the same as the sign of b_i (i.e., $a_i \cdot b_i \geq 0$). We say that $\mathbf{a} \sqsubseteq \mathbf{b}$ if $\mathbf{a}$ and $\mathbf{b}$ are sign-compatible and $|a_i| \leq |b_i|$ for every coordinate i. Note that $\sqsubseteq$ imposes a partial order.*

Definition 4 (Graver basis). *Given an integer matrix A, its* Graver basis *$\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal elements of the* lattice $\mathcal{L}^*(A) = \{\mathbf{h} \,|\, A\mathbf{h} = \mathbf{0}, \mathbf{h} \in \mathbb{Z}^n, \mathbf{h} \neq \mathbf{0}\}$.

Lemma 5 (Lemma 3.2 of [10]). *Any vector $\mathbf{h} \in \mathcal{L}^*(A)$ is a sign-compatible sum $\mathbf{h} = \sum_i \mathbf{g}^i$ of Graver basis elements $\mathbf{g}^i \in \mathcal{G}(A)$, with some elements possibly appearing with repetitions.*

Having the tools above, we may now prove the existence of a remapping involving small number of clusters.

Lemma 6. *If there exists a valid target vector $\mathbf{y}$, then there exists a valid target vector $\mathbf{y}'$, such that $\mathbf{x} - \mathbf{y}' \in \mathcal{G}(A)$.*

Proof. By (2) and the lemma assumption, $A\mathbf{x} = A\mathbf{y}$. Let $\mathbf{z} = \mathbf{x} - \mathbf{y}$. Then, $z_{\hat{\mathbf{c}}} = x_{\hat{\mathbf{c}}} - y_{\hat{\mathbf{c}}} = 1$ and $A\mathbf{z} = \mathbf{0}$, and thus $\mathbf{z} \in \mathcal{L}^*(A)$.

Using Lemma 5, we may express $\mathbf{z}$ as $\mathbf{z} = \sum_i \mathbf{g}^i$, where $\mathbf{g}^i \in \mathcal{G}(A) \subseteq \mathcal{L}^*(A)$ for all i and all $\mathbf{g}^i$ are sign-compatible with $\mathbf{z}$. As $z_{\hat{\mathbf{c}}} = 1$, the sign-compatibility means that there exists $\mathbf{g}^j$ appearing in the sum with $g_{\hat{\mathbf{c}}}^j = 1$. We set $\mathbf{w} = \mathbf{g}^j$.

Let $\mathbf{y}' = \mathbf{x} - \mathbf{w}$. Clearly $\mathbf{x} - \mathbf{y}' = \mathbf{w} \in \mathcal{G}(A)$. It remains to show that $\mathbf{y}'$ is a valid target vector. We have $A\mathbf{y}' = A\mathbf{x} - A\mathbf{w} = \mathbf{u} - \mathbf{0} = \mathbf{u}$ and $y_{\hat{\mathbf{c}}}' = x_{\hat{\mathbf{c}}} - w_{\hat{\mathbf{c}}} = 1 - 1 = 0$. To show that $\mathbf{y}' \geq \mathbf{0}$, we consider two cases. If $z_{\mathbf{c}} \geq 0$, then by sign-compatibility $0 \leq w_{\mathbf{c}} \leq z_{\mathbf{c}}$, and thus $y_{\mathbf{c}}' = x_{\mathbf{c}} - w_{\mathbf{c}} \geq x_{\mathbf{c}} - z_{\mathbf{c}} = y_{\mathbf{c}} \geq 0$. On the other hand, if $z_{\mathbf{c}} < 0$, then again by sign-compatibility, $w_{\mathbf{c}} \leq 0$, and thus $y_{\mathbf{c}}' = x_{\mathbf{c}} - w_{\mathbf{c}} \geq x_{\mathbf{c}} \geq 0$. $\square$

To complete our argument, it remains to bound $\|\mathbf{g}\|_1$, where $\mathbf{g}$ is an arbitrary element of $\mathcal{G}(A)$. We start with a known bound for ℓ^∞-norm of any element of the Graver basis.

Lemma 7 (Lemma 3.20 of [10]). *Let q be the number of columns of integer matrix M. Let $\Delta(M)$ denote the maximum absolute value of the determinant of a square sub-matrix of M. Then $\|\mathbf{g}\|_\infty \leq q \cdot \Delta(M)$ for any $\mathbf{g} \in \mathcal{G}(M)$.*

Lemma 8. *For any $\mathbf{g} \in \mathcal{G}(A)$, it holds that $\|\mathbf{g}\|_1 \leq 2^{O(k)}$.*

Proof. We start by showing that $\Delta(A) \leq e^k$. Let $\tilde{A}$ be the matrix A with each row multiplied by its index. By the definition of a configuration, the column sums of $\tilde{A}$ are equal to k, with the exception of the column corresponding to the configuration $\hat{\mathbf{c}}$ whose sum is equal to $2k$. As all entries of $\tilde{A}$ are non-negative, the same holds for ℓ_1-norms of columns of $\tilde{A}$.

Fix any square sub-matrix B of A and let j be the number of its columns (rows). Let $\tilde{B}$ be the corresponding sub-matrix of $\tilde{A}$. Since $\tilde{B}$ is obtained from B

by multiplying its rows by j distinct positive integers, $|\det(\tilde{B})| \geq j! \cdot |\det(B)|$. (This relation holds with equality if and only if B contains (a part of) the first j rows of A.)

It therefore remains to upper-bound $|\det(\tilde{B})|$. Hadamard's bound on determinant states that the absolute value of a determinant is at most the product of lengths (ℓ_2-norms) of its column vectors, which are in turn bounded by the ℓ_1-norms of columns of $\tilde{B}$. These are not greater than ℓ_1-norms of the corresponding columns of $\tilde{A}$ and thus $|\det(\tilde{B})| \leq 2 \cdot k^j$.

Combining the above bounds and using $j \leq k$, we obtain

$$|\det(B)| \leq \frac{|\det(\tilde{B})|}{j!} \leq \frac{2 \cdot k^j}{j!} \leq \frac{2 \cdot k^k}{k!} \leq 2 \cdot \mathrm{e}^{k-1} \leq \mathrm{e}^k \ .$$

As B was chosen as an arbitrary square sub-matrix of A, $\Delta(A) \leq \mathrm{e}^k$. Our matrix A has $|\mathcal{C}^{\mathrm{ext}}|$ columns, and hence Lemma 7 implies that $\|\mathbf{w}\|_\infty \leq |\mathcal{C}^{\mathrm{ext}}| \cdot 2^{O(k)}$. As $\mathbf{w}$ is $|\mathcal{C}^{\mathrm{ext}}|$-dimensional, $\|\mathbf{w}\|_1 \leq |\mathcal{C}^{\mathrm{ext}}| \cdot \|\mathbf{w}\|_\infty \leq |\mathcal{C}^{\mathrm{ext}}|^2 \cdot \mathrm{e}^k$. Finally using $|\mathcal{C}^{\mathrm{ext}}| \leq 2^{O(\sqrt{k})}$, we obtain $\|\mathbf{w}\|_1 \leq 2^{O(k)}$. $\qquad\square$

Corollary 1. *The remapping event of* COMP *affects at most* $2^{O(k)}$ *clusters.*

Proof. Combining Lemma 3 with Lemma 6 yields the existence of a valid target vector $\mathbf{y}'$ satisfying $\|\mathbf{x} - \mathbf{y}'\|_1 \in \mathcal{G}(A)$. By Lemma 8, $\|\mathbf{x} - \mathbf{y}'\|_1 \leq 2^{O(k)}$. Thus, plugging $\mathbf{y}'$ to Lemma 4 yields the corollary. $\qquad\square$

3.3   Competitive Ratio

Combining Lemma 1 with Corollary 1 immediately yields the desired bound on the competitive ratio of COMP.

Theorem 1. *The variant of* COMP *in which each remapping event is handled in a way minimizing the number of affected clusters is* $(\ell \cdot 2^{O(k)})$-*competitive.*

References

1. Andreev, K., Räcke, H.: Balanced graph partitioning. Theory of Computing Systems **39**(6), 929–939 (2006). https://doi.org/10.1007/s00224-006-1350-7
2. Avin, C., Bienkowski, M., Loukas, A., Pacut, M., Schmid, S.: Dynamic balanced graph partitioning. SIAM Journal on Discrete Mathematics **34**(3), 1791–1812 (2020). https://doi.org/10.1137/17M1158513
3. Avin, C., Loukas, A., Pacut, M., Schmid, S.: Online balanced repartitioning. In: Proc. 30th Int. Symp. on Distributed Computing (DISC). pp. 243–256 (2016). https://doi.org/10.1007/978-3-662-53426-7_18
4. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
5. Chowdhury, M., Zaharia, M., Ma, J., Jordan, M.I., Stoica, I.: Managing data transfers in computer clusters with Orchestra. In: ACM SIGCOMM. pp. 98–109 (2011). https://doi.org/10.1145/2018436.2018448

6. Erdős, P.: On an elementary proof of some asymptotic formulas in the theory of partitions. Annals of Mathematics **43**(3), 437–450 (1942). https://doi.org/10.2307/1968802
7. Henzinger, M., Neumann, S., Räcke, H., Schmid, S.: Tight bounds for online graph partitioning. In: Proc. 32nd ACM-SIAM Symp. on Discrete Algorithms (SODA). pp. 2799–2818 (2021). https://doi.org/10.1137/1.9781611976465.166
8. Henzinger, M., Neumann, S., Schmid, S.: Efficient distributed workload (re-)embedding. In: 2019 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems. pp. 43–44 (2019). https://doi.org/10.1145/3309697.3331503
9. Krauthgamer, R., Feige, U.: A polylogarithmic approximation of the minimum bisection. SIAM Review **48**(1), 99–130 (2006). https://doi.org/10.1137/050640904
10. Onn, S.: Nonlinear discrete optimization. Zurich Lectures in Advanced Mathematics, European Mathematical Society (2010)
11. Pacut, M., Parham, M., Schmid, S.: Brief announcement: Deterministic lower bound for dynamic balanced graph partitioning. In: Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC). pp. 461–463 (2020). https://doi.org/10.1145/3382734.3405696
12. Pacut, M., Parham, M., Schmid, S.: Optimal online balanced graph partitioning. In: Proc. 40th IEEE Int. Conf. on Computer Communications (INFOCOM). pp. 1–9 (2021). https://doi.org/10.1109/INFOCOM42981.2021.9488824
13. Räcke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: Proc. 40th ACM Symp. on Theory of Computing (STOC). pp. 255–264 (2008). https://doi.org/10.1145/1374376.1374415
14. Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons (1998)
15. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**(2), 202–208 (1985). https://doi.org/10.1145/2786.2793