Brief Announcement: Tight Bounds for Online Balanced Repartitioning

ANONYMOUS AUTHOR(S)

Distributed applications, including batch processing, streaming, scale-out databases, or machine learning, generate a significant amount of network traffic. By collocating frequently communicating nodes (e.g., virtual machines) on the same clusters (e.g., server or rack), the network load can be reduced and application performance improved. However, as the communication pattern is a priori unknown and may change over time, it needs to be learned efficiently, in an online manner. This paper revisits the online balanced repartitioning problem (introduced by Avin et al. at DISC 2016) which asks for an algorithm that strikes an optimal tradeoff between the benefits of collocation (i.e., lower network load) and its costs (i.e., migrations). Our first contribution is a significantly improved lower bound of $\Omega(k \cdot \ell)$ on the competitive ratio, where ℓ is the number of clusters and k is the cluster size, even for a scenario in which the communication pattern is static and can be perfectly partitioned; we also provide a tight upper bound of $O(k \cdot \ell)$ for this scenario. In addition, we present a tight upper bound of $\Theta(\ell)$ for $k = 3$, for the general model in which the communication pattern can change arbitrarily over time.

CCS Concepts: • **Networks** → **Network algorithms**; • **Computer systems organization** → *Cloud computing*; *Distributed architectures.*

Additional Key Words and Phrases: online algorithms, competitive analysis, graph partitioning, clustering

ACM Reference Format:
Anonymous Author(s). 2020. Brief Announcement: Tight Bounds for Online Balanced Repartitioning. In *Proceedings of PODC '20.* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

Motivation and model. The *balanced repartitioning* problem (BRP) is a fundamental learning problem that finds applications in the context of distributed systems optimization [2]. We are given a set V of n nodes (e.g., virtual machines or processes), initially arbitrarily partitioned into ℓ clusters (e.g., servers or entire racks), each of size k. The nodes interact using a sequence of pairwise communication requests $\sigma = (u_1, v_1), (u_2, v_2), (u_3, v_3), \ldots$, where a pair (u_t, v_t) indicates that nodes u_t and v_t exchange a certain amount of data. Nodes in $C \subset V$ are *collocated* if they reside in the same cluster.

An algorithm serves a communication request between two nodes either *locally* at cost 0 if they are collocated, or *remotely* at cost 1 if they are located in different clusters. We refer to these two types of requests as *internal* and *external* requests, respectively. Before serving a request, an online algorithm may perform a *repartition*, i.e., it may move ("migrate") some nodes into clusters different from their current clusters, while respecting the capacity of every cluster. Afterwards, the algorithm serves the request. The cost of migrating a node from one cluster to another is $\alpha \in \mathbb{Z}^+$. For any algorithm ALG, its cost, denoted by $\mathrm{ALG}(\sigma)$, is the total cost of communications and the cost of migrations performed by ALG while serving the sequence σ.

Manuscript submitted to ACM

We consider two flavors of the problem in this paper. In the first one, which was also studied by Henzinger et al. [3] (SIGMETRICS 2019) and which we here mainly consider for the sake of a stronger lower bound, we assume that σ simply reveals the edges of a static graph that can be perfectly partitioned (i.e., in principle, no external requests are required). In the second one, we consider a general model where σ can be arbitrary; this was the original problem introduced by Avin et al. [2] at DISC 2016. For simplicity, we will refer to the former model as the *learning* model (as one has to learn the static communication graph) and to the latter as the *online* model.

Contributions. We provide an improved lower bound of $\Omega(k \cdot \ell)$ on the competitive ratio of any online deterministic online algorithm even in the learning model; the best known lower bound so far was $\Omega(k)$, even for a more general model [2]. We also present an asymptotically optimal, $O(k \cdot \ell)$-competitive algorithm for the restricted model. For the online model, we present an asymptotically optimal, $\Theta(\ell)$ algorithm for $k = 3$; the best known upper bound so far was $O(\ell^2)$ [2]. Our algorithms can be distributed similarly to the approach in [3].

Related work. The existing results in [2] and [3] primarily focus on a model where the online algorithm can use augmentation, i.e., has slightly larger clusters than the offline algorithm. In contrast, in our paper, we focus on a scenario where the nodes need to be perfectly balanced among the clusters. The problem has also been studied in a weaker model where the adversary can only sample requests from a fixed distribution [1]. For clusters of size 2 (i.e., $k = 2$), it is known that a constant competitive algorithm exists [2].

Preliminaries. In this paper, we use the technique of maintaining *(connected) components* of nodes, similar to previous approaches [2]. A *component* is a subset of frequently communicating nodes. We use this concept in both algorithms in this paper, but also in the lower bound, to incur high cost for any online algorithm that splits them. An algorithm is *component respecting* if it always keeps nodes that belong to the same component collocated. That is, if the algorithm needs to move a node, it moves the whole component containing the node.

2 THE LEARNING PROBLEM

Lower bound. The adversary constructs components of nodes based on the partitioning choices of a given online algorithm. Adversary issues requests between pairs of nodes that belong to the same component. If an online algorithm splits a component at any moment, the adversary issues requests to the split pair until the algorithm collocates them. We utilize the fact that the online algorithm does not know the components of the adversary, and it may split some components while evicting nodes. On the other hand, OPT initially collocates all nodes of each component and never splits them again.

Theorem 2.1. *The competitive ratio of any deterministic online algorithm for BRP is in $\Omega(k \cdot \ell)$ for any $k \geq 3$.*

Proof. We construct an instance of the problem with ℓ clusters $\{S_1, S_2, \ldots, S_\ell\}$, $|S_i| = k$. Let $I(C)$ denote the cluster where nodes of component C are located initially. Fix any online algorithm ALG. We join some nodes into components, and produce the input sequence by observing actions of ALG with respect to nodes of these components. If ALG at any point splits a component (i.e., spreads its nodes over two or more clusters), then we continue to issue requests between non-collocated nodes of the component until ALG collocates the nodes of the component. If ALG never collocates them, then it is not competitive.

Initially each node belongs to its own component of size 1. We say that a node is single if it belongs to a singleton component. We begin by forming a component of single $k - 1$ nodes on the cluster S_1.

Let x_0 be the only single node left on S_1 and $y_0 \in S_2$ be any single node on S_2. Next, we merge components of x_0 and y_0. Since these reside on different clusters of ALG, it produces an external request. ALG must collocate them in

a cluster other than S_1. To this end, ALG moves to a new configuration that replaces x_0 and y_0 with two other single nodes x_1 and y_1 respectively.

Next, we merge the component of x_1 and the largest component C s.t. $I(C) = I(x_1)$. ALG must collocate x_1 and C in a cluster other than S_1 and consequently replace x_1 with some other (single) node x_2. We repeat merging the component of single node x_i on S_1 and the largest component C' s.t. $I(C') = I(x_i)$, until there are only two single nodes left that originate from the same cluster. Formally, we denote these two single nodes by x^*, y^*, and we have $I(x^*) = I(y^*)$, and for any other pair of single nodes x' and y', we have $I(x') \neq I(y')$. At this point there are at most $\ell + 1$ single nodes left, otherwise, there would be more pairs of single nodes that were initially in the same cluster.

Given this sequence of requests, the optimal strategy is to migrate $\{x_0, y_0\}$ to the cluster $I(x^*)$ by swapping $\{x_0, y_0\}$ with $\{x^*, y^*\}$. Hence, OPT pays for 2 node swaps and ALG incurs at least one swap for each node in the sequence $X := x_0, x_1, \ldots$. We exclude at most $(k-1) + (\ell+1)$ nodes out of $k \cdot \ell$ nodes, therefore $|X| \geq k \cdot \ell - k - \ell$. Finally, $\mathsf{ALG}/\mathsf{OPT} \geq (k \cdot \ell - k - \ell)/2 \in \Omega(k \cdot \ell)$. □

Upper bound. Now we give an algorithm for a restricted variant of BRP, named PP-BRP, where an optimal offline algorithm (OPT) moves to a perfect partition at the beginning and stays there perpetually. The task of an online algorithm for PP-BRP is to recover (or learn) the perfect partition while not paying too much relative to OPT. The first request between any two non-collocated nodes reveals an edge of the communication graph. Our online algorithm collocates them immediately and never separates them since by assumption OPT does the same. Revealed edges, up to any point in the sequence, induce some connected subgraphs which grow as components.

We assume OPT begins with the initial configuration $P_I = I_1, \ldots, I_\ell$ and moves to the final partitioning $P_F = F_1, \ldots, F_\ell$. The *distance* of a configuration $P = C_1, \ldots, C_\ell$ from the initial configuration is the number of nodes in P that do not reside in their initial cluster. That is, $\mathrm{dist}(P, P_I) := \sum_{j=1}^{\ell} |C_j \setminus I_j|$. In other words, at least $\mathrm{dist}(P, P_I)/2$ node swaps are required in order to reach the configuration P from P_I, and thus $\mathsf{OPT} \geq \Delta := \mathrm{dist}(P_F, P_I)$. With each repartitioning, PPL moves to a configuration that minimizes the distance to the initial configuration P_I. As a result, PPL never ends up in a configuration that is more than Δ (single-node) migrations away from P_I. This invariant ensures that PPL does not pay too much while recovering P_F. We note that the *repartition* at Line 8 replaces the current configuration P with a perfect partition closest to P_I. Hence it never moves to a configuration beyond distance Δ. The scheme of the algorithm can be found in the Appendix A.

Property 1. *Let P be any configuration chosen by PPL at Line 8. Then, $dist(P, P_I) \leq \Delta$.*

Lemma 2.2. *The cost of repartitioning at Line 8 is at most $2 \cdot OPT$.*

Proof. Consider the repartitioning that transforms P_{t-1} to P_t upon the request σ_t. Let $M \subset V$ denote the set of nodes that migrate during this process. Let M^- and M^+ denote the subset of nodes that (respectively) enter or leave their original cluster during the repartitioning. Then, $M = M^+ \cup M^-$. Since $|M^-|$ nodes are not in their original cluster before the repartitioning (i.e., in P_{t-1}), the distance before the repartitioning is $\mathrm{dist}(P_{t-1}, P_I) \geq |M^-|$. Analogously, the distance afterwards is $\mathrm{dist}(P_t, P_I) \geq |M^+|$. Thus, $|M| \leq \mathrm{dist}(P_{t-1}, P_I) + \mathrm{dist}(P_t, P_I)$. By Property 1, $\mathrm{dist}(P_{t-1}, P_I), \mathrm{dist}(P_t, P_I) \leq \Delta \leq \mathsf{OPT}$ and thereby we have $|M| \leq 2 \cdot \mathsf{OPT}$. □

Theorem 2.3. *PPL reaches the final configuration P_F and it is $(2 \cdot k \cdot \ell)$-competitive.*

Proof. On each inter-cluster request, the algorithm enumerates all ℓ-way partitions of components that are in the same (closest) distance of P_I. That is, once it reaches a configuration P at distance $\Delta = \mathrm{dist}(P, P_I)$, it does not move to

a configuration P', $\text{dist}(P', P_I) > \Delta$, before it enumerates all configurations at distance Δ. Therefore, PPL eventually reaches $\Delta = \text{OPT}$ and the configuration P_F. There are at most $(k-1) \cdot \ell < k \cdot \ell$ calls to *repartition* (i.e., the number of internal edges in P_F). By Lemma 2.2, each repartition costs at most $2 \cdot \text{OPT}$. The total cost is therefore at most $2 \cdot \text{OPT} \cdot k \cdot \ell$, which implies the competitive ratio. □

3 THE ONLINE PROBLEM

Let us now discuss the general online model where the request sequence can be arbitrary. We show that the classic *rent-or-buy* approach [4] allows obtaining an optimal algorithm for $k = 3$. Upon receiving α requests between a pair of nodes, we collocate them in one cluster until the end of a phase (that we define precisely later).

Now we describe the algorithm ALG_3. It partitions nodes into components, and initially, each node belongs to its own singleton component. For each pair of nodes, ALG_3 maintains a counter, initiated to 0. Upon receiving a request to a pair that is not collocated in one cluster, it increases their counter by 1. If the counter for a pair (u, v) reaches α, ALG_3 merges the components of u and v, and moves to the closest component respecting partitioning. If no such partitioning exists, ALG_3 resets all components to singletons, resets all counters to 0, and ends the phase.

In our analysis, we distinguish among three types of clusters: C_1, C_2, C_3. In a cluster of type C_i, the size of the largest component contained in this cluster is i. Before bounding the competitive ratio of ALG_3, we introduce the lemma that estimates the cost of a single repartition of ALG_3. We defer its proof to Appendix B.

Lemma 3.1. *During a single repartition, ALG_3 exchanges at most 2 pairs of nodes.*

Theorem 3.2. *ALG_3 is $O(\ell)$-competitive.*

Proof. Fix a completed phase, and consider the state of ALG_3's counters at the end of it. We consider the incomplete phase later in this proof.

As ALG_3 is component respecting, it never increases any counter above α. We say that the pair (u, v) is *saturated* if the counter has value α, and *unsaturated* otherwise. Let σ be the input sequence that arrived during the phase. Let σ_{cost} be the requests that at the moment of arrival were external requests for ALG_3 (these are the only requests that incurred a cost for ALG_3). In our analysis, we partition σ_{cost} into subsequences σ_I and σ_E. The sequence σ_I (inter-component requests) are the requests from σ_{cost} issued to pairs that belong to the same component of ALG_3 at the end of the phase. The sequence σ_E (extra-component requests) denotes the requests from σ_{cost} that do not appear in σ_I.

Let $\text{ALG}_3(M)$ be the cost of migrations performed by ALG_3 in this phase. ALG_3 performs at most 2ℓ component merge operations, as exceeding this number means that a component of size 4 exists, and the phase should have ended already. Combining this with Lemma 3.1 gives us $\text{ALG}_3(M) \leq 8\alpha \cdot \ell$ (recall that an exchange costs 2α).

Now we bound $\text{ALG}_3(\sigma_I)$. A cluster of type C_3 contributes at most $3\alpha - 1$ to $\text{ALG}_3(\sigma_I)$, as 2 of pairs of nodes from the component are saturated and contribute α each, and the third, unsaturated pair contributes at most $\alpha - 1$. Other cluster types contribute less: C_1 contributes 0 and C_2 contributes α. Summing this over all ℓ clusters gives us $\text{ALG}_3(\sigma_I) \leq (3\alpha - 1) \cdot \ell \leq 3\alpha \cdot \ell$.

Moreover, ALG_3 paid for all requests from σ_E, and thus $\text{ALG}_3(\sigma_E) = |\sigma_E|$. In total, the cost of ALG_3 is at most $\text{ALG}_3(\sigma_I) + \text{ALG}_3(\sigma_E) + \text{ALG}_3(M) \leq 11\alpha \cdot \ell + |\sigma_E|$ during this phase.

Now we lower-bound the cost of OPT. By $\text{OPT}(\sigma_I)$ and $\text{OPT}(\sigma_E)$ we denote the cost of OPT on these input sequences (defined with respect to components of ALG_3 in this phase). By $\text{OPT}(M)$ we denote the cost of migrations performed by OPT in this phase.

We split the cost of OPT into parts coming from serving σ_I and σ_E. While serving these requests, OPT may perform migrations, and we account for them in both parts: we separately bound OPT by $\mathrm{OPT}(\sigma_I) + \mathrm{OPT}(M)$ and $\mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M)$. Combining those bounds gives us $\mathrm{OPT} \geq \max\{\mathrm{OPT}(\sigma_I) + \mathrm{OPT}(M), \mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M)\} \geq (\mathrm{OPT}(\sigma_I) + \mathrm{OPT}(M))/2 + (\mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M))/2$.

We have $\mathrm{OPT}(M) + \mathrm{OPT}(\sigma_I) \geq \alpha$, as the phase ended when the components of ALG_3 could not be partitioned without splitting them. Hence, for every possible configuration of OPT, there exists a non-collocated pair of nodes with at least α requests between them, and OPT either served them remotely or performed a migration.

Before we bound the competitive ratio, we relate the costs of ALG_3 and OPT with respect to requests σ_E. In a fixed configuration of OPT, it may mitigate paying for requests between at most 3ℓ pairs of nodes by collocating them in its clusters. Recall that σ_E consists of requests to unsaturated pairs, and it accounts only for requests that increased the counter (i.e., external requests), thus OPT may mitigate at most $3\ell \cdot (\alpha - 1)$ requests from σ_E. Faced with $W := |\sigma_E| - 3\ell \cdot (\alpha - 1)$ requests σ_E that it could not mitigate, OPT served some of them remotely and possibly performed some migrations to decrease its cost.

Now we estimate the cost of $\mathrm{OPT}(\sigma_E)$ while accounting savings from migrations. By performing a swap of nodes (u, v), OPT collocates u with two nodes u', u'', and v with two nodes v', v''. This may allow serving requests between (u, u'), (u, u''), (v, v') and (v, v'') for free afterwards. As σ_E consists of requests to unsaturated pairs, and it accounts only for external requests, there are at most $\alpha - 1$ requests between each of these pairs. By performing a single swap that costs 2α, OPT may avoid paying the remote serving costs for at most $4(\alpha-1)$ requests from σ_E. Thus, for serving σ_E, OPT pays at least $\mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M) \geq W \cdot \frac{2\alpha}{4(\alpha-1)} \geq |\sigma_E|/2 - 2\alpha \cdot \ell$. From that, we obtain $|\sigma_E| \geq 2(\mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M)) + 4\alpha \cdot \ell$. Let $E := \mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M)$. Finally, we are ready to bound the competitive ratio

$$\frac{\mathrm{ALG}_3(\sigma)}{\mathrm{OPT}(\sigma)} \leq \frac{11\alpha \cdot \ell + |\sigma_E|}{\alpha/2 + E/2} \leq \frac{27\alpha \cdot \ell + 2 \cdot E}{\alpha + E} \leq 27\ell = O(\ell).$$

It remains to consider the last, unfinished phase. Consider the case, where the unfinished phase is also the first one. Then, we cannot charge OPT due to the inability to partition the components. Instead, we use the fact that ALG_3 and OPT started with the same initial configuration. We charge OPT α for the first external α requests or a migration, and we follow the analysis regarding the unsaturated requests. If the input finished before the first α external requests, then ALG_3 is optimal. Now, consider the case, where there are at least two phases, then we split the cost α charged in the penultimate phase into last two phases, and follow the analysis regarding the unsaturated requests. This way, the competitive ratio increases at most twofold. □

REFERENCES

[1] Chen Avin, Louis Cohen, Mahmoud Parham, and Stefan Schmid. 2019. Competitive clustering of stochastic communication patterns on a ring. *Computing* 101, 9 (2019), 1369–1390. https://doi.org/10.1007/s00607-018-0666-x

[2] Chen Avin, Andreas Loukas, Maciej Pacut, and Stefan Schmid. 2016. Online Balanced Repartitioning. *DISC* (2016), 243–256.

[3] Monika Henzinger, Stefan Neumann, and Stefan Schmid. 2019. Efficient Distributed Workload (Re-)Embedding. In *ACM SIGMETRICS / IFIP Performance 2019*.

[4] Anna Karlin, Mark Manasse, and Lyle McGeoch Karlin. 1994. Competitive Randomized Algorithms for Nonuniform Problems. *Algorithmica* 11, 6 (1994), 542––571.

A PPL ALGORITHM

Algorithm 1 Perfect Partition Learner (PPL)

For each node v create a singleton component C_v and add it to $\mathcal{C}$
$P_0 := P_I$
for each request $\sigma_t = \{u, v\}, 1 \leq t \leq N$ **do**
  Let $C_1 \ni u$ and $C_2 \ni v$ be the container components
  if $C_1 \neq C_2$ **then**
    Unite the two components into a single component C' and $\mathcal{C} = (\mathcal{C} \setminus \{C_1, C_2\}) \cup \{C'\}$
    if $cluster(C_1, P_{t-1}) \neq cluster(C_2, P_{t-1})$ {i.e. if not in the same cluster} **then**
      $P_t = repartition(P_{t-1}, P_I, \mathcal{C})$ {move to P closest to P_I}
    end if
  end if
end for

B OMITTED PROOFS

Proof of Lemma 3.1. Observe that when the repartition is triggered by ALG_3, the resulting partitioning is component respecting. Otherwise, if it does not exist, ALG_3 simply ends the phase and performs no repartition.

Consider a request between u and v that triggered the repartition and let U and V be their respective clusters. Note that $U \neq V$, as otherwise, the request would not trigger a repartitioning. We consider cases based on the types of clusters U and V. Note that this is impossible that either U or V is of type C_3, as otherwise, we merge components of size 3, and no component respecting partitioning exists, a contradiction. If either U or V is of type C_1, then this cluster can fit the merged component, and the reconfiguration is local within U and V. For two clusters, any configuration can be reached within two swaps, due to the fact that clusters are indistinguishable.

Finally, we consider the case where both U and V are of type C_2. Note that (u, v) cannot both belong to a component of size 2, as this would mean that ALG_3 has the component of size 4, a contradiction with the case assumption that the component respecting partitioning exists. Otherwise, if either u or v belongs to a component of size 2, then it suffices to exchange components of size 1 between U and V. Finally, if u and v belong to components of size 1, then we must place them in a cluster different from U and V. Note that in such case, a C_1-type cluster W exists, as otherwise no component respecting partitioning exists. In this case ALG_3 performs one swap, i.e., it exchanges the nodes u and v with any two nodes of W. □