Brief Announcement: An Improved Lower Bound for Dynamic Balanced Graph Partitioning

Distributed applications, including batch processing, streaming, and scale-out databases generate a significant amount of network traffic and a considerable fraction of their runtime is due to network activity. In this paper, we study algorithms for collocating frequently communicating nodes in a distributed networked systems in an online fashion.

In the online graph repartitioning problem, we are given a sequence of pairwise communication requests between n nodes, revealed online. The objective is to service these requests by partitioning the nodes into ℓ clusters, each of size k, such that frequently communicating nodes are located in the same cluster. The partitioning can be updated by migrating nodes between clusters for a fixed cost. The goal is to jointly minimize the amount of inter-cluster communication and migration cost.

So far, the best known lower bound for the competitive ratio of deterministic algorithms was $\Omega(k)$, by reduction from online paging. We improve this result significantly by providing the lower bound of $\Omega(k \cdot \ell)$. We follow up with an optimal algorithm for $k = 3$. Finally, we provide a $O(k \cdot \ell)$-competitive algorithm for the variant where the optimal strategy is to move to a fixed configuration that incurs zero communication cost through the whole request sequence.

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

ACM Reference Format:

. 2020. Brief Announcement: An Improved Lower Bound for Dynamic Balanced Graph Partitioning. In *Proceedings of PODC '20*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Main technical contribution of this paper is the lower bound $\Omega(k \cdot \ell)$ for the competitive ratio of any deterministic algorithm for online balanced repartition problem (see Section 4). The best known lower bound so far was $\Omega(k)$ [**?**] which involves only two clusters.

So far, the competitive ratio of the best known algorithm was the component-based algorithm with the ratio $O(k^2 \cdot \ell^2)$ [**?**]. By applying classic rent-or-buy technique, we obtained an optimal algorithm for $k = 3$, that we present in Section 5. In Section 6, we study a restricted model of balanced repartition, called perfect partition. For that, we present an $O(k \cdot \ell)$-competitive algorithm, which is optimal. We believe that this result is an important building block on the way to an optimal algorithm for the general model.

Mahmoud: The mention of distributed alg, very much out of context!

A distributed adaptation of a similar algorithm has been analyzed in [**?**].

2 RELATED WORK

The problem was studied under resource augmentation [**?**], and under stochastic assumptions for the input[**?**]. For clusters of size 2 (i.e., $k = 2$), a constant competitive algorithm exists [**?**].

Manuscript submitted to ACM

The model is related to online caching [**? ? ? ?**]. In our setting, requests can be served remotely, which is reminiscent of caching models *with bypassing* [**? ? ?**].

The static offline version of the problem, called the *ℓ-balanced graph partitioning problem* is NP-complete, and cannot even be approximated within any finite factor unless P = NP [**?**]. The static variant where $\ell = 2$ corresponds to the minimum bisection problem, which is already NP-hard [**?**], and the currently best approximation ratio is $O(\log n)$ [**? ? ? ? ? ?**].

3 PRELIMINARIES

The online *balanced repartitioning* problem (BRP) is defined as follows. We are given a set of n nodes, initially arbitrarily partitioned into ℓ clusters, each of size k. The input consists of 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 fixed 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 *repartition*, i.e., it may (optionally) move some nodes into clusters different than their current clusters while respecting the capacity of every cluster. Only after a repartitioning is completed, 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 ALG(σ), is the total cost of communications and the cost of migrations performed by ALG while serving the sequence σ. The objective is to minimize the total incurred cost.

In this paper, we use the technique of maintaining *components* of nodes, similar to previous approaches [**?**]. A *component* is a subset of frequently communicating nodes. Initially, every node is trivially a *singleton* component. 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.

4 LOWER BOUND

In this section, we provide a lower bound $\Omega(k \cdot \ell)$ for the competitive ratio of any deterministic algorithm. It is sufficient to have clusters of capacity $k \geq 3$ for this lower bound. For $k = 2$ a constant competitive algorithm exists.

The adversary constructs components of nodes, and issues only requests among nodes in the same component. If an online algorithm splits the components at any moment, the adversary issues requests to split pairs of nodes. 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 never splits the components.

Theorem 4.1. *The competitive ratio of any deterministic 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 a component C are located initially. Fix any online algorithm ALG. We construct the input sequence for ALG as follows. First, we issue $k - 2$ (internal) requests so that ALG can form a component of $k - 1$ nodes on the cluster S_1. Recall that internal requests does not incur any cost for ALG. 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. Hence, if ALG never collocates them then it is not competitive.

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 issue a request between x_0 and y_0. Since this is an external request, ALG joins them into one component and collocates them in some cluster other

than S_1. For this, 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 issue a request between x_1 and the largest component C s.t. $I(C) = I(x_1)$. ALG must collocate x_1 and C in some cluster other than S_1 and consequently replaces x_1 with some other (single) node x_2. We repeat issuing requests between the 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 4 node migrations and ALG incurs at least one migration 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 $|X| \geq k \cdot \ell - k - \ell \in \Omega(k \cdot \ell)$. □

5 AN OPTIMAL ALGORITHM FOR CLUSTERS OF SIZE 3

In this section we show that the classic *rent-or-buy* approach[**? ?**] allows to obtain 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.

Lemma 5.1. *A single repartition of ALG_3 consists of at most 2 migrations.*

Proof. 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 upon 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 migrations, due to the fact that clusters are indistiguishable.

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 migration, i.e., it exchanges the nodes u and v with any two nodes of W. □

Theorem 5.2. *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 5.1 gives us $\text{ALG}_3(M) \leq 4\alpha \cdot \ell$.

Now we bound $\text{ALG}_3(\sigma_I)$. 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 $\text{ALG}_3(\sigma_I) + \text{ALG}_3(\sigma_E) + \text{ALG}_3(M) \leq 7\alpha \cdot \ell + |\sigma_E|$ 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 $\text{OPT}(\sigma_I) + \text{OPT}(M)$ and $\text{OPT}(\sigma_E) + \text{OPT}(M)$. Combining those bounds gives us $\text{OPT} \geq \max\{\text{OPT}(\sigma_I) + \text{OPT}(M), \text{OPT}(\sigma_E) + \text{OPT}(M)\} \geq (\text{OPT}(\sigma_I) + \text{OPT}(M))/2 + (\text{OPT}(\sigma_E) + \text{OPT}(M))/2$.

We have $\text{OPT}(M) + \text{OPT}(\sigma_I) \geq \alpha$, 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 $\text{OPT}(\sigma_E)$ 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 to serve 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 migration

that costs α, 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 $W \cdot \frac{\alpha}{4(\alpha-1)}$. This gives us

$$\mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M) \geq (|\sigma_E| - 3(\alpha-1)\cdot\ell) \cdot \frac{\alpha}{4(\alpha-1)} \geq |\sigma_E|/4 - 4\alpha\cdot\ell.$$

From that, we obtain $|\sigma_E| \geq 4(\mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M)) + 16\alpha\cdot\ell$. Let $E := \mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M)$. Finally, we are ready to bound the competitive ratio

$$\frac{\mathrm{ALG}_3(\sigma)}{\mathrm{OPT}(\sigma)} \leq \frac{7\alpha\cdot\ell + |\sigma_E|}{(\alpha + \mathrm{OPT}(\sigma_E) + \mathrm{OPT}(M))/2} \leq \frac{46\alpha\cdot\ell + 8\cdot E}{\alpha + E} \leq 46\ell = O(\ell).$$

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 unability 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 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. □

6 AN OPTIMAL ALGORITHM FOR PERFECT PARTITION

In this section, 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.

6.1 Perfect Partition Learner (PPL) Algorithm

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 $\mathrm{OPT} \geq \Delta := \mathrm{dist}(P_F, P_I)$. 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.

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

Lemma 6.1. *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

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

the repartitioning (i.e., in P_{t-1}), the distance before the repartitioning is $\text{dist}(P_{t-1}, P_I) \geq |M^-|$. Analogously, the distance afterwards is $\text{dist}(P_t, P_I) \geq |M^+|$. Thus, $|M| \leq \text{dist}(P_{t-1}, P_I) + \text{dist}(P_t, P_I)$. By Property 1, $\text{dist}(P_{t-1}, P_I), \text{dist}(P_t, P_I) \leq \Delta \leq \text{OPT}$ and thereby we have $|M| \leq 2 \cdot \text{OPT}$. □

Theorem 6.2. *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 $\Delta = \text{dist}(P, P_I)$, 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 6.1, 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. □

In Section 4 we constructed a $\Omega(k \cdot \ell)$ for BRP. Note that the lower bound holds also in the perfect partition model, as the constructed input sequence allows OPT to move to a perfect partition. The corollary is that PPL is optimal.

REFERENCES

[] Dimitris Achlioptas, Marek Chrobak, and John Noga. 2000. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science* 234, 1–2 (2000), 203–218.

[] Konstantin Andreev and Harald Räcke. 2006. Balanced Graph Partitioning. *Theory of Computing Systems* 39, 6 (2006), 929–939.

[] Sanjeev Arora, David R. Karger, and Marek Karpinski. 1999. Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems. *J. Comput. System Sci.* 58, 1 (1999), 193–210.

[] 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

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

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

[] Leah Epstein, Csanád Imreh, Asaf Levin, and Judit Nagy-György. 2011. On Variants of File Caching. In *Proc. 38th Int. Colloq. on Automata, Languages and Programming (ICALP).* 195–206.

[] Leah Epstein, Csanád Imreh, Asaf Levin, and Judit Nagy-György. 2015. Online File Caching with Rejection Penalties. *Algorithmica* 71, 2 (2015), 279–306.

[] Uriel Feige and Robert Krauthgamer. 2002. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.* 31, 4 (2002), 1090–1118.

[] Uriel Feige, Robert Krauthgamer, and Kobbi Nissim. 2000. Approximating the minimum bisection size (extended abstract). In *Proc. 32nd ACM Symposium on Theory of Computing (STOC).* 530–536.

[] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. 1991. Competitive paging algorithms. *Journal of Algorithms* 12, 4 (1991), 685–699.

[] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. 1976. Some Simplified NP-Complete Graph Problems. 1, 3 (1976), 237–267.

[] Monika Henzinger, Stefan Neumann, and Stefan Schmid. 2019. Efficient Distributed Workload (Re-)Embedding. In *ACM SIGMETRICS / IFIP Performance 2019*.
[] Sandy Irani. 2002. Page Replacement with Multi-Size Pages and Applications to Web Caching. *Algorithmica* 33, 3 (2002), 384–409.
[] Anna Karlin, Mark Manasse, and Lyle McGeoch Karlin. 1994. Competitive Randomized Algorithms for Nonuniform Problems. *Algorithmica* 11, 6 (1994), 542––571.
[] Robert Krauthgamer and Uriel Feige. 2006. A Polylogarithmic Approximation of the Minimum Bisection. *SIAM Rev.* 48, 1 (2006), 99–130.
[] Lyle McGeoch and Daniel Sleator. 1991. A Strongly Competitive Randomized Paging Algorithm. *Algorithmica* 6, 6 (1991), 816–825.
[] Harald Räcke. 2008. Optimal hierarchica decompositions for congestion minimization in networks. In *Proc. 40th ACM Symposium on Theory of Computing (STOC)*. 255–264.
[] Huzur Saran and Vijay Vazirani. 1995. Finding k Cuts within Twice the Optimal. *SIAM J. Comput.* 24, 1 (1995), 101–108.
[] Daniel Sleator and Robert Tarjan. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM* 28, 2 (1985), 202–208.

A OMMITED PROOFS

B TODOS

todo: Author anonymous conference!

todo: ACM format for review? Is there any specific format like this? With line numbers etc.

todo: CCS concepts are mandatory

todo: ACM Reference format starts with a "." Possibly we did not provide all info