Brief Announcement: Deterministic Lower Bound for Dynamic Balanced Graph Partitioning

Maciej Pacut
maciej.pacut@univie.ac.at
Faculty of Computer Science,
University of Vienna
Austria

Mahmoud Parham
mahmoud.parham@univie.ac.at
Faculty of Computer Science,
University of Vienna
Austria

Stefan Schmid
stefan_schmid@univie.ac.at
Faculty of Computer Science,
University of Vienna
Austria

ABSTRACT

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), we can reduce the network load and improve application performance. However, the communication pattern of different applications is often unknown a priori and may change over time, hence it needs to be learned in an online manner. This paper revisits the online balanced partitioning problem (introduced by Avin et al. at DISC 2016) that 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 deterministic 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 an asymptotically tight upper bound of $O(k \cdot \ell)$ for this scenario. For $k = 3$, we contribute an asymptotically tight upper bound of $\Theta(\ell)$ for the general model in which the communication pattern can change arbitrarily over time. In contrast to most prior work, our algorithms respect all capacity constraints and do not require resource augmentation.

CCS CONCEPTS

• **Theory of computation** → **Online algorithms**; • **Networks** → **Network algorithms**; • **Computer systems organization** → *Distributed architectures.*

KEYWORDS

online algorithms, competitive analysis, distributed computing, graph partitioning, clustering, self-adjusting networks

ACM Reference Format:
Maciej Pacut, Mahmoud Parham, and Stefan Schmid. 2020. Brief Announcement: Deterministic Lower Bound for Dynamic Balanced Graph Partitioning. In *Proceedings of ACM Symposium on Principles of Distributed Computing 2020 (PODC '20).* ACM, New York, NY, USA, 3 pages. https://doi.org/TBD

PODC '20, July 29-August 2, 2020, Toronto, Canada

ACM ISBN TBD…$TBD
https://doi.org/TBD

1 INTRODUCTION

The popularity of data-centric, distributed applications has led to an explosive growth of network traffic, especially in data centers [8, 9]. The performance of these distributed applications often critically depends on the underlying network [7], and efficient operation of these networks is important. At the same time, distributed systems are often highly virtualized today, and provide interesting new opportunities for resource optimization. In particular, it has become possible to operate data centers in a more demand-aware manner: by dynamically migrating nodes (e.g., virtual machines) which communicate frequently topologically closer to each other, network traffic can be reduced significantly. However, migrations entail overhead and should be used moderately.

This paper studies the algorithmic problem underlying such demand-aware optimizations, aiming to strike a balance between the benefits of migrations (e.g., reduced network load) and their costs. In particular, we are interested in an online variant of the problem: since communication patterns can change over time, an online algorithm needs to react dynamically to new traffic patterns, and migrate nodes accordingly. Ideally, this algorithm should perform close to an optimal offline algorithm, without requiring any information about future traffic demands.

Model. The *dynamic balanced graph partitioning* problem (BRP) is a fundamental learning problem that finds applications in the context of distributed systems optimization [4, 5]. 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. Afterward, 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 $\text{ALG}(\sigma)$, is the total cost of communications and the cost of migrations performed by ALG while serving the sequence σ.

Related work. The two works closest to ours are by Avin et al. (on the general partitioning model) [4, 5] and by Henzinger et al. (on the learning model) [6]. The static offline version of the partitioning problem, i.e., a problem variant where migration is not allowed, where all requests are known in advance, and where the goal is to find an assignment of n nodes to ℓ physical machines, each of capacity n/ℓ, is known as the *ℓ-balanced graph partitioning problem* [2]. Dynamic graph partitioning problems are generally fundamental in computer science, and arise in many different contexts [1, 10].
Contributions. This paper presents several new results on the dynamic graph partitioning problem without augmentation. For the learning model, we present a lower bound of $\Omega(k \cdot \ell)$ on the competitive ratio of any online deterministic online algorithm (that holds also in the general partitioning model). We complement this result with an asymptotically optimal, $O(k \cdot \ell)$-competitive algorithm The best known lower bounds so far were $\Omega(k)$ for the general partitioning model [4, 5], and $\Omega(k)$ for the learning model [6]. For the general partitioning model, we design an asymptotically optimal, $\Theta(\ell)$-competitive algorithm for $k = 3$, improving the best known upper bound so far $O(\ell^2)$ [4, 5].

2 THE LEARNING MODEL

In this section, we study a *learning* variant of dynamic balanced graph partitioning, where the communication pattern is *static*: whether a pair of nodes ever communicate or not, is determined a priori and is unknown to algorithms, and such pairs communicate forever. Any algorithm must eventually collocate pairs of communicating nodes, as otherwise it cannot be competitive. As in Henzinger et al. [6], we assume that the communication graph admits a *perfect partition*, i.e., a partition in which no inter-cluster request ever occurs. The algorithm's objective is to *learn* the (static) communication graph while serving all requests, and without executing too many migrations.

2.1 Lower Bound

We present a lower bound $\Omega(k \cdot \ell)$ for the competitive ratio of any deterministic online algorithm for the learning problem. Later, we elaborate on how to efficiently transform it to a lower bound for the general partitioning problem. The lower bound requires $k \geq 3$. In contrast, for $k = 2$ the learning problem is trivial: immediate collocation of communicating pairs is 1-competitive.

Throughout this paper, we often refer to groups of communicating nodes. We use this concept slightly differently in the lower bound than the upper bounds. In our algorithms, we group nodes with a communication history into *components*. In this section, we group nodes that may ever communicate, into *ground sets*.

Given a perfect partition, every subset of nodes that belong to the same cluster in this partition is a *ground set*. Any competitive algorithm under the learning model maintains a perfect partition of ground sets into clusters. On each inter-cluster request, a ground set is revealed. An algorithm recovers the (hidden) perfect partition gradually over inter-cluster requests, by merging pairs of ground sets involved in these requests.

The adversary constructs ground sets depending on the choices of a deterministic online algorithm. Once we construct a ground set, it lasts until the end of the input sequence. We say that a ground set is a *singleton* if it contains exactly one node, which is an *isolated* node.

We start by constructing a ground set of size $k-1$ on an arbitrarily chosen cluster. In any partition, there must exist an isolated node collocated with the ground set of size $k-1$. We issue requests between this node and some node that was initially collocated with it. By repeating such requests, almost every node is once collocated with the first ground set. In comparison, we show that there exists an optimal offline algorithm OPT that performs only two node exchanges ("swaps").

Theorem 2.1. *The competitive ratio of any deterministic online algorithm for the learning model of Dynamic Balanced Graph Partitioning is at least $\Omega(k \cdot \ell)$ for any $k \geq 3$ and $\ell \geq 2$.*

Proof. Fix any online algorithm ALG. For a ground set C of nodes that are initially collocated in one cluster, let $I(C)$ denote the cluster. We refer to $I(C)$ as the cluster of *origin*, when C is clear from the context. Initially, all nodes are isolated, i.e., each node is in a singleton ground set. First, we choose a cluster arbitrarily and create a ground set B of $k-1$ nodes in this cluster, and issue requests between its nodes. Each cluster hosts exactly k nodes, and in any feasible partition, a single isolated node must be collocated with B. At any time, we refer to the isolated node currently collocated with B as the *pivot* node. Let x_0 denote the first pivot node.

Then, we join the pivot node to a larger ground set to force its eviction. Precisely, we create a ground set $\{x_0, y_0\}$, where y_0 is an arbitrary isolated node. Since ALG does not have $\{x_0, y_0\}$ collocated, the adversary issues an external request to this pair so that ALG collocates it. ALG cannot collocate $\{x_0, y_0\}$ with B (as B's size is $k-1$), hence it collocates them in a different cluster. In order to preserve a feasible partition of nodes after collocating $\{x_0, y_0\}$, ALG must replace x_0 with another isolated node that becomes the new pivot.

We proceed in similar steps by joining the current pivot node to a ground set of the same origin residing in a different cluster. Consider the step i, when the isolated node x_i is collocated with B. We issue a request between x_i and some node in C_i, where C_i is the largest ground set s.t. $I(C_i) = I(x_i), C_i \neq \{x_0, y_0\}$. Then ALG must collocate the new ground set $\{x_i\} \cup C_i$ in one cluster. Any feasible partition replaces x_i with some isolated node x_{i+1}, as the new ground set $\{x_i\} \cup C_i$ may not be ever split. We terminate the process once the number of remaining isolated nodes is less than $\ell + 3$. At each step i, the number of isolated nodes decreases either by one or by two if C_i is a singleton. Therefore, once the process terminates, in any case at least $\ell + 1$ isolated nodes are left.

Next, we argue that a feasible partition exists when the process terminates. This implies that a feasible partition exists after any earlier step as well. Since there are at least $\ell + 1$ isolated nodes left, there must be two isolated nodes x^* and y^*, with the same cluster of origin, i.e., $I(\{x^*\}) = I(\{y^*\})$. Consider the partition P^* obtained from the initial partition after swapping x_0 and y_0 with x^* and y^* (respectively). In this partition, the ground set $\{x_0, y_0\}$ is collocated in the cluster $I(\{x^*, y^*\})$. Note that after the first request $\{x_0, y_0\}$, we issue requests only between nodes that have the same cluster of origin and all these nodes are collocated in P^*. Therefore all ground sets constructed so far are collocated in P^*, and it is a feasible partition.

Consider nodes x^* and y^* and the partition P^* obtained previously. OPT moves to P^* by performing only two node swaps. Precisely, OPT collocates $\{x_0, y_0\}$ by swapping them with x^* and y^*. No ground set is split in P^* and OPT pays only for the two swaps.

ALG performs at least one swap at each step i, and some ground set grows. Consider any ground set $C^* \neq B$ after the termination. This ground set has grown exactly $|C^*| - 1$ times until the termination. Let $\mathcal{S}$ be the set of all ground sets after the process terminates. Thus, $\mathcal{S}$ includes ground sets B, $\{x_0, y_0\}$, and (up to) $\ell + 2$ singleton ground sets. Among the remaining ground sets in $\mathcal{S}$, no two ground sets have the same origin. Otherwise, the smaller ground set is either a singleton, which contradicts the bound $\ell+2$ on the number of singletons, or we have joined nodes to it at some step, contradicting our choice of the largest C_i at step i. Hence, there are at most $\ell - 1$ such ground sets, one per possible cluster of origin, excluding the cluster containing B. Therefore, $|\mathcal{S}| \leq 1+1+(\ell+2)+(\ell-1) = 2\ell+3$. Note that among all non-singleton ground sets in $\mathcal{S}$, only B does not grow during the process. Thus, the total number of times that a ground set in $\mathcal{S}$ has grown is

$$\sum_{C^* \in \mathcal{S}} (|C^*| - 1) - (k-1) = \sum_{C^* \in \mathcal{S}} |C^*| - \sum_{C^* \in \mathcal{S}} 1 - (k-1)$$
$$\geq k\ell - (2\ell + 3) - (k-1) = (k-2)(\ell-1) - 4,$$

which bounds the number of swaps performed by ALG. The competitive ratio is then ALG/OPT $\geq ((k-2)(\ell-1)-4)/2$. □

2.2 Upper Bound

We present an asymptotically optimal algorithm for the learning problem. The algorithm collocates a pair as soon as they communicate and it never separates them. In order to preserve collocated pairs, we employ the concept of components, introduced by Avin et al. [4, 5].

We maintain subsets of frequently communicating nodes as *components*. Initially, each node constitutes a single-node component which we refer to as a *singleton* component, and the node in such component is an *isolated* node. We keep all nodes of a component always collocated in the same cluster, i.e., when we move a node, we move the whole component that contains it. A partition that has every component collocated is a *component respecting* partition. We maintain a balanced partition of our components as long as such partition exists, a reminiscent of partitioning given integers into sets of equal sum [3]. In contrast, our partition is time-varying: two components are merged into one component once they communicate, and we adjust the partition accordingly.

Perfect Partition Learner algorithm. Now we describe the algorithm PPL. On each inter-cluster request $\{u, v\}$, PPL creates new components by merging the two components that contain nodes u and v. In order to collocate nodes of the new component, PPL moves to a component respecting partition that minimizes the distance to the initial partition.

Theorem 2.2. *PPL is $O(k \cdot \ell)$-competitive.*

3 GENERAL PARTITIONING MODEL: OPTIMAL ALGORITHM FOR CLUSTERS OF SIZE 3

Now we discuss the general online model where the request sequence can be arbitrary. The algorithm analyzed in this section is a modified version of the algorithm DET proposed by Avin et al. [4, 5], which for $k = 3$ is $O(\ell^2)$-competitive.

Component-based algorithm. The algorithm ALG_3 partitions nodes into components, and initially, each node is isolated (belongs to its own component). For each pair of nodes $\{x, y\}$, ALG_3 maintains a counter $C_{\{x,y\}}$ and increments it on every external request between x and y. Once $C_{\{x,y\}} = \alpha$, 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 singleton components, resets all counters to 0, and ends the phase.

In our algorithm, we choose the closest partition after a component merge instead of an arbitrary one. This allows to bound the cost of repartition by a constant:

Lemma 3.1. *In a single repartition of nodes, ALG_3 exchanges at most two pairs of nodes.*

This modification alone is insufficient to obtain $O(\ell)$-competitive algorithm: pairs of nodes that did not reach the collocation threshold α incur the cost $O(\ell^2)$. We carefully analyse this cost and relate it to the cost of OPT.

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

ACKNOWLEDGMENTS

Research supported by ERC Consolidator project, Self-Adjusting Networks (AdjustNet), grant agreement No. 864228, Horizon 2020, 2020-2025.

REFERENCES

[1] Dan Alistarh, Jennifer Iglesias, and Milan Vojnovic. Streaming min-max hypergraph partitioning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1900–1908. Curran Associates, Inc., 2015.
[2] Konstantin Andreev and Harald Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.
[3] George Andrews and Kimmo Eriksson. *Integer Partitions.* Cambridge University Press.
[4] Chen Avin, Marcin Bienkowski, Andreas Loukas, Maciej Pacut, and Stefan Schmid. Dynamic Balanced Graph Partitioning. *SIAM Journal on Discrete Mathematics (SIDMA)*, 2020.
[5] Chen Avin, Andreas Loukas, Maciej Pacut, and Stefan Schmid. Online Balanced Repartitioning. *DISC*, pages 243–256, 2016.
[6] Monika Henzinger, Stefan Neumann, and Stefan Schmid. Efficient distributed workload (re-)embedding. In *ACM SIGMETRICS / IFIP Performance 2019*, 2019.
[7] Jeffrey C Mogul and Lucian Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012.
[8] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 123–137, 2015.
[9] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM Computer Communication review*, 45(4):183–197, 2015.
[10] Isabelle Stanton. Streaming balanced graph partitioning algorithms for random graphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 1287–1301, 2014.