Well-Balanced Allocation on General Graphs

Nikhil Bansal* Ohad Feldheim†

Abstract

We study the graphical generalization of the 2-choice balls-into-bins process, where rather than choosing any two random bins, the bins correspond to vertices of an underlying graph, and only the bins connected by an edge can be chosen.

For any $k(n)$ edge-connected, $d(n)$-regular graph on n vertices and any number of balls, we give an allocation strategy which guarantees that the maximum gap between the bin loads is $O((d/k)\log^4 n \log\log n)$, with high probability. We further show that the dependence on k is tight and give an $\Omega((d/k)+\log n)$ lower bound on the gap achievable by any allocation strategy, for any graph G. In particular, our result gives polylogarithmic bounds for natural graphs such as cycles and tori, where the classical greedy allocation appears to result in a polynomial gap. Previously such a bound was known only for graphs with good expansion.

The construction is based on defining certain orthogonal flows on cut-based Räcke decomposition of graphs. The allocation algorithm itself, however, is simple to implement and takes only $O(\log(n))$ time per allocation, and can be viewed as a global version of the greedy strategy that compares average load on sets of vertices, rather than on individual vertices.

*CWI Amsterdam and TU Eindhoven, N.Bansal@cwi.nl. Supported by the NWO VICI grant 639.023.812.

†Hebrew University of Jerusalem Israel, ohad.feldheim@mail.huji.ac.il. Supported by ISF grant 1327/19.

1 Introduction

Randomized balls-into-bins models serve as useful abstractions for various problems arising in hashing, load balancing and resource allocation in parallel and distributed systems and have been extensively studied in the areas of probability, economics and algorithms (see e.g. [RS98, DR96, AK14]). The balls typically represent tasks or items, that need to be allocated to resources that are modeled by the bins, and the goal is to minimize either the maximum load or the gap between the maximum and minimum load as much as possible.

The models differ by what kind of control is available to the algorithm over the allocation process. In the classical *single-choice* model, the algorithm has no control and each ball is placed in a bin chosen uniformly at random. For m balls and n bins, it is well known that for $m = n$, the heaviest bin has load $(1 + o(1)) \ln n / \ln \ln n$ with high probability (w.h.p.). For $m \geq n \log n$, the bins have load in the range $m/n \pm \Theta(\sqrt{(m \log n)/n})$ i.e., deviating from the average load of m/n by $\Theta(\sqrt{(m \log n)/n})$.

Perhaps the simplest and most well-studied controlled variant of this model is the *2-choice* model. In this model, at each step the algorithm is given two uniformly chosen bins into one of which it must allocate the ball. This modification, which may appear minor, leads to substantial improvements. In a seminal result, Azar, Broder, Karlin and Upfal [ABKU94] showed that if a ball is placed in the least loaded of $d \geq 2$ uniformly sampled bins, then for $m = O(n)$, the maximum load reduces to $\ln \ln n / \ln d + \Theta(m/n)$. They also establish that, asymptotically, this *greedy* allocation strategy is optimal for this model. These results were extended by Berenbrink, Czumaj, Steger and Vöcking [BCSV06] to arbitrary m, who showed that the maximum load is $m/n + \ln \ln n / \ln d + O(1)$ with probability $1 - 1/\text{poly}(n)$. It is worth noting that even for $d = 2$ choices, the excess load over the average does not increase with m, unlike for the case of $d = 1$.

Graphical process. In many natural settings, there are restrictions on which pairs of bins can be queried or where the ball can be placed. An elegant generalization of the 2-choice process, called the *graphical process*, was introduced by Kenthapadi and Panigrahy [KP06]. Here there is an underlying graph $G = (V, E)$ on $n = |V|$ vertices, and at each step, a uniformly random edge $e = (u, v)$ is chosen and the ball must be placed on one of the two endpoints of e. Notice that the 2-choice process corresponds to $G = K_n$. This motivates the following natural question:

Given a graph G what is the best load balance obtainable by a graphical two-choice allocation strategy?

Extending the results for the classical 2-choice process ($G = K_n$), [KP06] showed that if G is n^ϵ-regular, then for $m = n$ balls, the greedy strategy, has maximum load is $\ln \ln n + O(\log 1/\epsilon)$. An extension to hypergraphs was considered in [God08].

The setting of arbitrary m, also the focus of our work, was considered by Peres, Talwar, Weider [PTW15]. They investigated the greedy strategy, and showed a gap of $\Theta(\log n)$ for regular expander graphs between the maximum and minimum bin load. This gap is the best possible for any strategy, up to constants. More generally they showed that the gap is $O((\log n)/\beta)$ for any d-regular graph with edge-expansion[1] β.

Remark 1.1. *Let us define* upper gap *as the difference between the maximum load and the average load, and* gap *as the maximum difference between the bins loads. Both of these objectives have been studied extensively, and they can sometimes differ significantly, e.g., for $G = K_n$ where the gap is $\Theta(\log n)$ and the upper gap is $\Theta(\log \log n)$. However, this difference often disappears in the setting of general graphs, or more general choice models that we consider here. For example, for a constant degree expander G, the gap is $\Theta(\log n)$ while the upper gap is $\Omega(\log n / \log \log n)$. We discuss this further in Sections 1.2 and 5.*

Graphs with low expansion and limitations of the greedy strategy. While the upper-bound of $O((\log n)/\beta)$ by Peres et al., implies a poly-logarithmic gap for well-expanding graphs, it gives only polynomially large $O(n \log n)$ and $O(n^{1/2} \log n)$ bounds for graphs such as cycles or two-dimensional grids, with low expansion. The dependence on β in the result of [PTW15] is not tight and they leave open the problem of getting the right bound, even for these simple classes of graphs.

[1] For any subset $S \subset V$ with $|S| \leq n/2$, $E(S, \overline{S}) \geq \beta d |S|$.

While for certain non-expander graphs, such as high-dimensional balanced grids, it is expected that the greedy algorithm will result in polylogarithmic gaps, and only the tools to prove it are missing, there are many graphs on which the greedy strategy appears to be inherently quite sub-optimal. In particular let us consider the case of a cycle, which has been well studied, the conjectured gap and upper gap estimates for greedy are $\Omega(\sqrt{n})$. Some evidence for this conjecture was given by Alistarh, Nadiradze and Sabour in [ANS]. They consider a model in which after allocating the ball to one of the vertices of the requested edge, the new load of the two vertices becomes the average of their load. Even in this model, which intuitively should have a lower gap than the standard greedy algorithm, they show an $\Omega(\sqrt{n})$ lower bound for the typical gap. We also provide supporting simulation results for this prediction in figure 1.

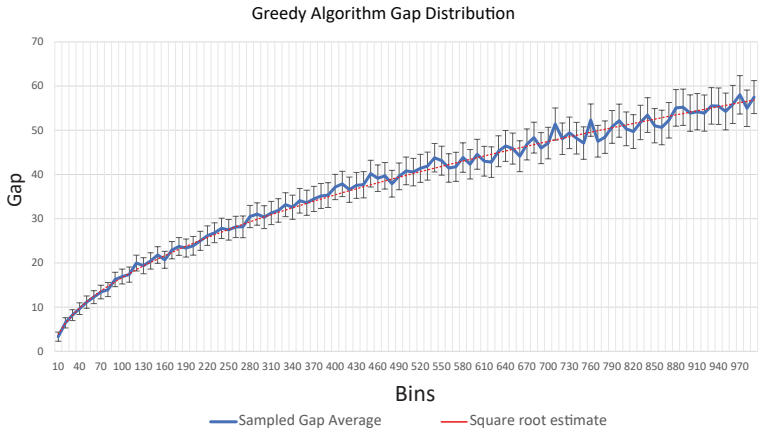

Figure 1: The gap for the greedy algorithm on cycles, averaged over 84 runs of 10^9 balls for cycles of sizes from 10 to 1000. The dotted guide is the function $f(x) = 1.85\sqrt{x} - 1$, error margins for 95% confidence are provided. The graph clearly shows the polynomial growth of the gap.

More generally, it has been conjectured that the load fluctuations under greedy are similar to the fluctuations in classical statistical mechanics models. In particular Peres (in private communication) suggested that, up to a $\log n$ factor the gap should be the same as that of a Gaussian free field on G. Both these conjectures and simulation results suggest that on very poorly expanding graphs, such as cycles and unbalanced tori the imbalance in load should be polynomial in n.

This raises the natural question whether there exist other allocation strategies, ideally simple to implement, which achieve substantially better gaps on such graphs.

1.1 Our Results

We give an allocation strategy that achieves the best possible gap, up to polylogarithmic factors, for the graphical process on any graph G. More formally, we show the following.

Theorem 1.1. *Let $G = (V, E)$ be any k edge-connected, d-regular graph on n vertices. There is an allocation strategy for the graphical process on G, that guarantees for any time $t \in \mathbb{N}$,*

$$\mathrm{gap}_G(t) = O((d/k)\log^4 n \log\log n)$$

with probability at least $1 - 1/poly(n)$. Here $\mathrm{gap}_G(t)$ is the maximum difference in vertex loads at time t.

In addition, we show that this is the best possible, up to polylogarithmic factors, for every graph G.

Theorem 1.2. *For every k edge-connected, d-regular graph G and for any allocation strategy for G, for any time t, with constant probability, $gap_G(t) = \Omega(d/k + \log n)$.*

Theorem 1.1 implies an allocation strategy with gap polylog(n) for cycles and grids, and more generally for any graph G with $k = \Omega(d/\text{polylog}(n))$.

In fact, our strategy is shown to have gap $O(\alpha_G(d/k)\log^2 n)$ where α_G is the congestion ratio for oblivious routing on G based on a Räcke decomposition tree [Räc02] (see Section 1.3). The bound in Theorem 1.1 follows from a result of Harrelson, Hildrum and Rao [HHR03] who showed that $\alpha_G = O(\log^2 n \log\log n)$ for any G.

The Algorithm. The allocation strategy is simple and incurs only $O(\log n)$ worst case running time per allocation. It can be viewed as a more *global* version of the greedy strategy, where instead of comparing the loads on two vertices, one compares the average load on two random sets chosen from a fixed small collection.

More specifically, the algorithm maintains a table of size $O(n)$ with entries consisting of the average load on certain subsets S of V. When a random edge $e = (x, y)$ is requested, it picks two sets S, S' according to a *fixed* distribution P_e specifying probabilities $p_e(S, S')$ over pairs of these subsets. It then assigns the ball to x or y depending on which among S and S' has larger average load. The sets S form a balanced hierarchical decomposition of V, so when a ball is assigned to some vertex v, only the $O(\log n)$ entries for sets S containing v are updated.

Ideas and Techniques. The main tools in obtaining Theorem 1.1 are *binary tree decompositions* and *orthogonal flows*. This method, inspired by discrepancy theory tools, defines pairs of sets on the graph in a hierarchical fashion and constructs the distributions P_e using a mixture of probabilistic allocation strategies. Each of these strategies affects only the relative allocation probabilities of the sets in a pair, without affecting these for other pairs. For a more thorough overview of our method see Section 2.

The lower bound in Theorem 1.2 is based on a simple observation is that if we fix any minimum cut $(S, \overline{S})$ of size k, there are not enough edges crossing the cut to balance out the random load fluctuations that arise due to the requests for edges with both endpoints in S.

Finally, we make a few remarks concerning the model and our results.

Remark 1.2. *The requirement that G is regular is standard in the area, and is made so that the expected load under a random strategy is equal on all bins. The results extend to irregular graphs in a natural way where one measures the gap with loads normalized suitably by the degree.*

Remark 1.3. *Our results do not require the full power of two choices. In the $1+\beta$ graphical choice model, at each step an edge is given only with probability β, and with probability $1-\beta$ the algorithm is given no choice and the ball is allocated to a random bin. Our bound on the gap extends directly to this model with factor $O(1/\beta)$ loss. More background on this model is provided in Section 1.2 below.*

Remark 1.4. *While better bounds are known for congestion in oblivious routing [ACF+04, Räc08, AF09], our method uses the full power of a single Räcke decomposition tree as the demands we define for our flows depend on the tree itself.*

1.2 Related work and models

The literature on ball-into-bins processes is extensive and it is impossible to cover even a small portion of the developments and applications. The first appearance of a 2-choice type result was in [KLadH96] in the context of online hashing. Following the result of Azar et al., [ABKU94] several variations of the model have been studied. Variations include models in which balls are eliminated over time [Mit91, CFM+98] – either by age or at random and parallel allocation of the balls with limited communication [Ste96, ACMR98]. Many of the earlier results are surveyed in Mitzenmacher's Thesis [Mit91] and in his survey with Richa and Sitaraman [MRS01]. A more recent survey is due to Wieder [Wie17].

In his thesis Mitzenmacher suggested the model of $1+\beta$ choice for $\beta < 1$, where the algorithm is given two choices with probability β and only one choice with probability $1-\beta$. His motivation for introducing this model stems from a problem in queuing theory. Vöcking [Vöc03] show that for d-choice, dependent

can improve over the greedy algorithm of [ABKU94], resulting in maximum load of $\Theta(\log\log(n)/d)$ (cf. $\Theta(\log\log(n)/\log d)$).

The heavily loaded case of the two-choice problem was first analyzed in [BCSV06] (see a neat and short proof by Talwar and Wieder [TW14]). In [PTW15], Peres, Talwar and Wieder considered the $1+\beta$ choice model for complete graphs and showed that there both gap and upper deviation is $\Theta(\log n)$. The drift and potential methods introduced in this work allowed the authors to relate this result to the graphical case for expanders in [PTW15] and inspired methods used here as well.

1.3 Notation and Preliminaries

A graphical process is specified for some fixed underlying d-regular graph $G=(V,E)$. Henceforth we denote $n=|V|$ and $m=|E|$ (typically in balls-into-bins literature, m is used for the number of balls, but we will think of the allocation process as indefinite, using t for the index of the allocated ball). At each time step $t=1,2,\ldots,$ an edge $e=e_t=(u,v)\in E$ is chosen (*requested*) uniformly at random, and a ball must be assigned to one of its endpoints u or v. We often refer to the vertices of G as bins. A vertex u has load ℓ at time t, if ℓ balls have been assigned to it after t allocations. Let $L^t:V\to\mathbb{N}$ denote the load vector at time t. We assume $L^0(u)=0$ for all $u\in V$ so that the total load $\|L^t\|_1=t$ for each $t\in\mathbb{N}$. Hence, the average load at a vertex at time t is t/n.

An allocation strategy, upon the request $e=(u,v)$, decides whether to assign the ball to u or to v (possibly based on the entire history so far). The goal of the strategy is to minimize the gap, where we define the gap at time t as

$$\mathrm{gap}(t)=\mathrm{gap}_G(t)=\max_u L^t(u)-\min_u L^t(u).$$

As $\max_u|L^t(u)-t/n|\le\mathrm{gap}_G(t)\le 2\max_u|L^t(u)-t/n|$, sometimes we will work with $\max_u|L^t(u)-t/n|$ and we refer to this the maximum deviation from the average load. For a subset $S\subset V$, we denote the total load on vertices in S by $L^t(S):=\sum_{u\in S}L^t(u)$, and the average load by $\overline{L}^t(S):=L^t(S)/|S|$.

Hierarchical cut-based decomposition. A *hierarchical decomposition* of G is a recursive partition of the vertex set V until each resulting set is a singleton vertex. Such a decomposition is naturally viewed as a rooted tree $T=(V_T,E_T)$, where each node $i\in V_T$ corresponds to a subset $S_i\subset V$. The root r of T corresponds to V, the leaves to singleton sets $\{u\}$ for $u\in V$. For any node $i\in V_T$, its children $j_1,\ldots,j_k$ correspond to the sets $S_{j_1},\ldots,S_{j_k}$ obtained by partitioning S_i.

Removing an edge (i,j) of T, where j is a child of i, partitions the leaves of T (or nodes of G) into S_j and $V\setminus S_j$. In a *cut-based* decomposition, we associate an edge $(i,j)\in E_T$ with the cut $(S_j,V\setminus S_j)$ of G, and set its capacity $c_T(i,j)=C_G(S_j)$, the capacity of the cut $(S_j,V\setminus S_j)$ in G.

We will work with G and its decomposition T. To avoid confusion, we use u,v,x,y to index the vertices of G, and i,j,j' to index the vertices of T. Sometimes we index the leaves of T by u,v to highlight the correspondence with vertices of G. We will refer to the vertices of T as nodes.

Räcke Trees and Oblivious Routing. Let $G=(V,E,c)$ be an undirected graph with edge capacities $c_e\ge 0$. An oblivious routing for G specifies for each ordered pair of nodes $u,v\in V$, a flow template $f_{uv}:V\times V\to\mathbb{R}$ on the edges, that describes how to send one unit of flow from u to v. So each f_{uv} satisfies

$$f_{uv}(x,y)=-f_{uv}(y,x),\ \text{for all } x,y\in V,\ \text{and}\ \sum_y f_{uv}(x,y)=1_{(x=u)}-1_{(x=v)}\ \text{for all } x\in V.$$

Let $\vec{d}=(d(u,v))_{u,v}$ be any multi-commodity demand vector with commodities $(u,v)\in V\times V$ and demands $d(u,v)\ge 0$. An oblivious routing of $\vec{d}$ on G sends $d(u,v)$ units of each commodity (u,v) according to f_{uv}.

In a breakthrough work, Räcke [Räc02] showed that for any edge capacitated $G=(V,E,c)$, there exists a cut-based decomposition tree $R=(V_R,E_R,c_R)$, and flow templates f_{uv} for each $u,v\in V$ satisfying the following remarkable property. Consider any demand vector $\vec{d}$. On R, as the $d(u,v)$ units are routed along

the unique path from leaves u to v, let $g_{R,\vec{d}}(i,j) = \sum_{u\in S_j, v\notin S_j} d(u,v) + \sum_{u\notin S_j, v\in S_j}$, be the total flow across $(i,j) \in E_R$ and let $\text{cong}(R,\vec{d}) = \max_{(i,j)\in E_R} g_{R,\vec{d}}(i,j)/c_R(i,j)$ be the maximum edge-congestion on R. Similarly, consider the oblivious $g_{G,\vec{d}}$ for $\vec{d}$ on G and let $\text{cong}(G,\vec{d}) = \max_{e\in E} g_{G,\vec{d}}(e)/c_e$ be the maximum edge-congestion on G. Then for any $\vec{d}$,

$$\text{cong}(R,\vec{d}) \le \text{cong}(G,\vec{d}) \le \alpha_G \text{ cong}(R,\vec{d}).$$

We refer to α_G as congestion ratio for oblivious routing on G. The best bound on α_G is $O(\log^2 n \log\log n)$ [HHR03], and the corresponding R and flow templates f_{uv} can be found in polynomial time. Almost linear time constructions with slightly worse α_G are also known [RST14]. Below, we also use the fact that in both of these constructions, the tree R satisfies $|S_j| \le 3/4|S_i|$ for each edge (i,j) and that R has depth $O(\log n)$.

2 Overview

Before describing the details, we first provide a high level overview of the idea and the algorithm.

Upon the arrival of an edge request $e = (u,v)$, by choosing whether to allocate the ball to u or v, an allocation strategy can *bias* the expected load toward u or v. As the vertex loads fluctuate below and above the average load over time, these biases must depend on the current load vector, and the goal is to design an allocation strategy that results in a self-regulating process that keeps the load deviations small.

Balancing average load on sets. Let $G = (V,E)$ be the graph underlying the process. Consider some *binary* hierarchical decomposition of G, represented by a binary tree $T = (V_T, E_T)$, with nodes $i \in V_T$ corresponding to subsets $S_i \in V$ and leaves — to vertices in G. Let r be the root of T, and for a leaf u, consider the unique path $u = i_0, i_1, \ldots, i_h = r$ from u to r in T, so that $\{u\} = S_{i_0} \subset S_{i_1} \subset \ldots \subset S_{i_h} = V$. As $\overline{L}^t(r) = t/n$ and $\overline{L}^t(u) = L^t(u)$, the load deviation for u at time t is at most

$$|L^t(u) - t/n| = \left|\sum_{j=1}^{h} \left(\overline{L}^t(S_{i_{j-1}}) - \overline{L}^t(S_{i_j})\right)\right| \le \sum_{j=1}^{h} \left|\overline{L}^t(S_{i_{j-1}}) - \overline{L}^t(S_{i_j})\right|,$$

and hence to control the load deviation for each $u \in V$, up to an $h = O(\log n)$ factor, it suffices to control the gap $|\overline{L}^t(S_{i_{j-1}}) - \overline{L}^t(S_{i_j})|$ for each parent-child node pair.

Fix a non-leaf node $i \in V_T$, and let $\ell(i), r(i)$ denote its left and right children. To bound the average load gap between S_i and its children, a simple computation shows that it suffices to bound $|\overline{L}(S_{\ell(i)}) - \overline{L}(S_{r(i)})|$. So a natural idea is to assign each non-leaf node i the task of balancing $\overline{L}(S_{\ell(i)})$ and $\overline{L}(S_{\ell(i)})$. To do this, one can try to assign biases to edges to create a *relative bias* from the most loaded among $S_{\ell(i)}$ and $S_{r(i)}$ to the least loadeed. Notice that the relative biases should be strong enough for each sibling pair, to get good bounds on the balance. In particular, if $k = \Omega(d)$, then to achieve the desired polylogarithmic gap in Theorem 1.1, the total bias between every two siblings $S_{\ell(i)}$ and $S_{r(i)}$ must be $\Omega(d/\text{polylog}(n))$.

Orthogonal Multi-commodity Flows. A key difficulty in implementing this idea is that the bias on an edge e affects several sets S_i, (see Figure 2). In this example, we would like to use edges between S_1 and S_2 to create a bias from S_2, which has higher average load, to S_1. However, this create a bias towards S_{12} compared to S_{11}, even though S_{12} already has higher load. In general, it is not clear how to create edge biases to balance the relative bias *simultaneously* for all sibling pairs.

To get around this, we do two things. First the graph decomposition is constructed carefully, as we discuss later. Second, for each non-leaf node i in T, we create a flow F_i between vertices in $S_{\ell(i)}$ and $S_{r(i)}$ We can view the flow across an edge e as the bias that we would like to assign to e to balance $\overline{L}(S_{\ell(i)})$ and $\overline{L}(S_{r(i)})$. So the direction of the flow F_i depends on which side has higher average load.

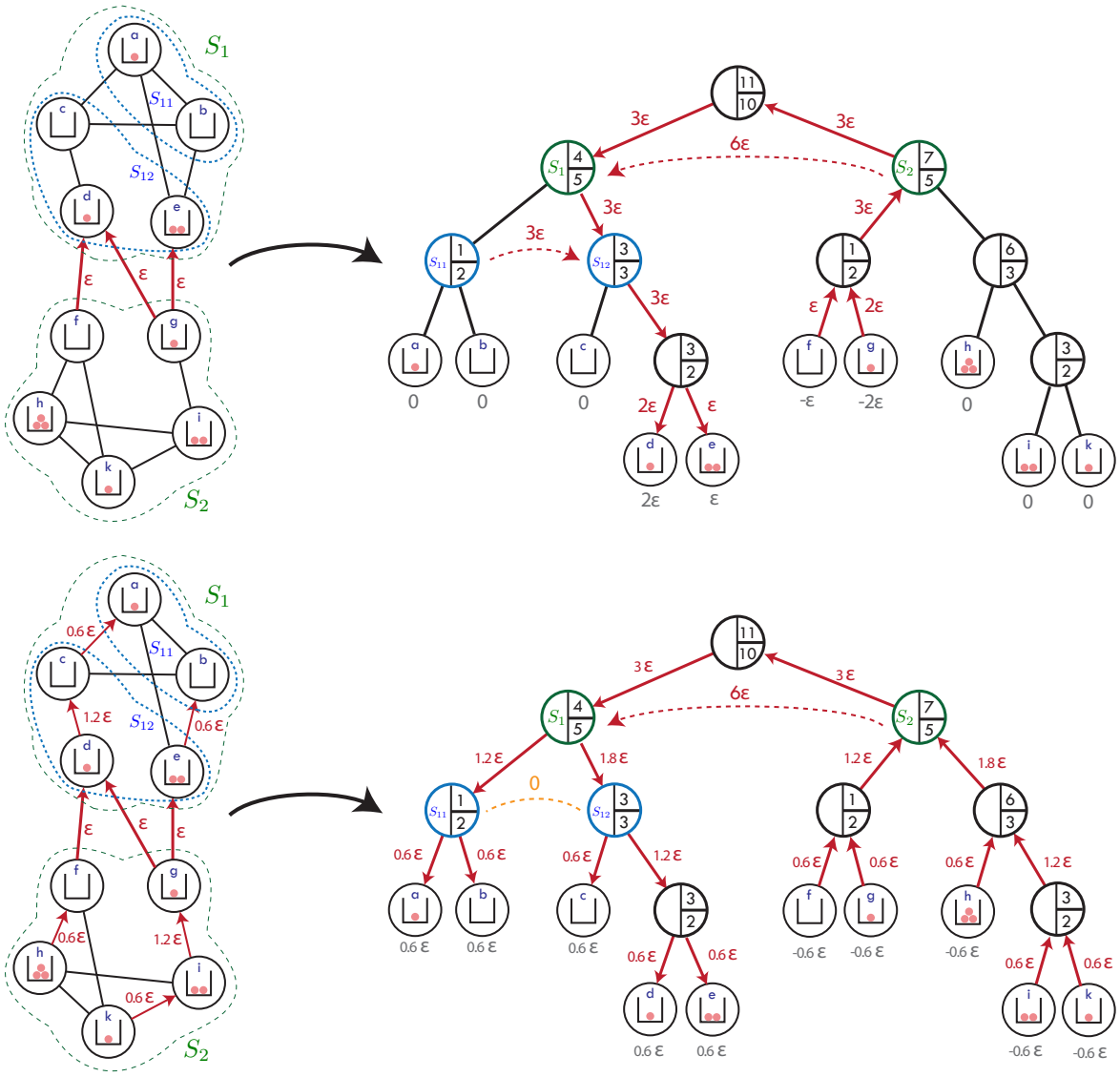

Figure 2: On the **left**, two edge biases on a graph G are illustrated by red arrows. Each of these is aimed at creating a relative bias from S_2 with average load $\overline{L}(S_2) = \frac{7}{5}$ to S_1 with $\overline{L}(S_2) = \frac{4}{5}$. On the **right** are their impact on the graph's decomposition tree. Each node of the tree represents a vertex subset S (equivalently, a cut) in the graph. Inside the node is the average load of the cut (balls divided by bins). The horizontal dashed edges indicate the relative bias between two sibling cuts. The edge from a S to its parent is oriented by the total bias on that S. **Above** we see the non-orthogonal flow incurred by the edges the cross the green cut. Observe how these create an undesired relative bias from the lightly loaded S_{11} to the heavily loaded S_{12}. **Below** we see an orthogonal flow which has no impact on the relative bias of any pair of sibling sets except S_1 and S_2.

However, as the bias on e affects each set S_j that e crosses, we choose the demands between vertices in $S_\ell(i)$ and $S_{r(i)}$ to define F_i such that the flow F_i does not affect the balancing task of any other non-leaf node $j \neq i$. More formally for any non-leaf node $j \neq i$, the quantity $\overline{L}(S_{\ell(j)}) - \overline{L}(S_{\ell(j)})$ incurs zero bias due to the flow F_i. We refer to this as the *orthogonal* property of demands and it allows to create biases that simultaneously balance $\overline{L}(S_{\ell(j)}), \overline{L}(S_{r(j)})$ for all pairs. This also is illustrated in Figure 2.

To do this, we define a multi-commodity flow for each tuple (i, u, v) with $u \in S_{\ell(i)}$ and $v \in S_{r(i)}$ and assign it a demand $d_i(u, v)$ proportional to $1/|S_{\ell(i)}||S_{r(i)}|$. If any non-leaf node $j \neq i$ in the subtree rooted at i, this ensures that the flow F_i is split in proportion to $|S_{\ell(j)}|$ and $|S_{r(j)}|$ along the two children of j, ensuring orthogonality as $\overline{L}(S_{\ell(j)}) - \overline{L}(S_{r(j)})$ incurs zero bias. Moreover if i and j do not lie along some root-leaf path, the tree structure ensures that the flows F_i and F_j do not affect each other.

In principle we could have generated these flows ad-hoc, however it is desirable that our allocation rule shall be polylogarithmically fast and simple to implement. The fact that we use a separate commodity for the flow between every pair of sibling sets allows us to simply change the sign of that flow when the load balance between the sets shifts.

Räcke Trees. The above describes a strategy on a tree. To realize this strategy using flows, we use Raecke's cut-based decomposition for G. Roughly speaking, this ensures that the decomposition does not have bottlenecks and sufficiently high biases can be created between the relevant sets. Moreover in this realization the different commodities are *oblivious* to each other, that is, the flow realization for each commodity depends only on the demand of that commodity. This enables the simple implementation of our algorithm.

In general, Räcke trees need not be binary. To get around this we work with an auxiliary binary tree T obtained from R, that we call a *balancing-flow tree*. Even though T is not a valid cut decomposition anymore, it maintains the properties needed for our purpose.

3 Algorithm

Let $G = (V, E)$ be the underlying graph for the graphical process and let $k = k(G)$ denote the edge-connectivity (the size of the minimum cut) of G.

The algorithm consists from two parts. A preprocessing stage, and an online allocation stage. In the preprocessing stage we analyze the graph structure and compute the decomposition tree. In the online allocation stage we use these to realize an allocation strategy that uses $O(\log n)$ operations per request to balance the load at all times.

3.1 Computing the strategy

Given G, we first construct a Räcke cut-based decomposition $R = (V_R, E_R, c_R)$ tree for G, together with the flow templates f_{uv} for each ordered pair $u, v \in V$, as defined in Section 1.3.

Let α_G be the congestion ratio for G with respect to R. We apply the algorithm of [] to construct R, so that $\alpha_G = O(\log^2 n \log\log n)$ and the running time is polynomial. Let $R_i \subset V$ denote the subset corresponding to $i \in V_R$, and recall that $|R_j| \leq (3/4)|R_i|$ for each $(i, j) \in E_R$, so that the depth of R is $O(\log n)$. Moreover, as G is k edge-connected, each edge $(i, j) \in E_R$ has capacity $c_R(i, j) \geq k$.

Next, we convert R into a *binary* hierarchical decomposition tree $T = (V_T, E_T)$ of G. The nodes of T will define the sets whose average load we will balance in the allocation strategy. To do this we will define certain flows on T, and we call T a balancing-flow tree. We do not define the edge capacities c_T, as we will not view T as a cut-based decomposition. The reader may wish to assume that R is already binary, and skip the paragraph below.

Creating the tree T

Apply the following step repeatedly to R until all non-leaf nodes have degree 2.

Choose any node i with $p > 2$ children $j_1, \ldots, j_p$. Let $w(h) = |S_{j_h}|/|S_i|$ for $h \in [p]$, so that $\sum_{h=1}^{p} w(h) = 1$.

(i) If $w(h) > 1/4$ for some h, make h the left child of i. Create a new right child b with $p - 1$ children $[p] \setminus \{h\}$. The node b corresponds to the set $S_i \setminus S_h$.

(ii) Otherwise, arbitrarily partition $[p]$ into two sets A and B so that $w(A), w(B) \in [1/4, 3/4]$. Create two children a and b of i, where the children of a (resp. b) are the nodes in A (resp. B). The nodes a and b corresponds to sets $\cup_{j\in A} S_j$ and $\cup_{j\in B} S_j$.

Clearly, T is binary. Also T and R have the same set of leaves as R is a contraction of T. Also note that for any node a, if $|S_a|/S_{p(a)}| > 3/4$ where $p(a)$ is the parent of a (due to step (i) above), then a must be a node in the original tree Räcke tree R.

The following observation will be quite useful later.

Claim 3.1. *Let a, b be two nodes in T such that a is an ancestor b, and let q be the distance from a to b. Then $|S_b| \leq (3/4)^{\lfloor q/2 \rfloor}|S_a|$. In particular, T has depth $O(\log n)$.*

Proof. Consider some path c, d, e of length 2 in T, where c is the ancestor of e. It suffices to show that either $|S_e| \leq (3/4)|S_d|$ or $|S_d| \leq (3/4)|S_c|$. To see this observe that if d was a node in R, this follows by the well-balancedness of a Räcke tree for the edge (d, e). Otherwise, (c, d) was produced according to rule (ii) and thus $|S_e| \leq |S_d| \leq (3/4)|s_c|$. □

Defining the demands on T

Let I_T denote the set of internal (non-leaf) vertices of T. For $i \in I_T$, let $\ell(i)$ and $r(i)$ denote its left and right children As described in Section 2, each node $i \in I_T$ will be associated with a flow f_i which will be used to balance $\overline{L}(S_{\ell(i)})$ and $\overline{L}(S_{r(i)})$, the average load on subsets corresponding to its left and right children.

To this end, we define a collection of demands D_i for each $i \in i_T$. The flow produced by these demands will determine the allocation strategy as we will describe later below.

Let $\beta = 1/(8\alpha_G)$. For each $i \in I_T$, the collection D_i consists of $|S_{\ell(i)}||S_{r(i)}|$ commodities (i, u, v) for each pair of vertices $u \in S_{\ell(i)}, v \in S_{r(i)}$. For each such pair, we create the demand from u to v of

$$d_i(u, v) = \frac{\beta k}{|S_{\ell(i)}||S_{r(i)}|}.$$

For a pair of nodes $j \in V_T$ and $i \in I_T$, let

$$d_i(j) = \sum_{u\in S_j, v\notin S_j} d_i(u, v) - \sum_{u\notin S_i, v\in S_j} d_i(u, v), \tag{1}$$

denote the total demand from D_i leaving the set S_j.

Let T_i denote the subtree of T rooted at i. Note that $d_i(j) = 0$ for any node $j \notin T_i$, as the demands in D_i are between the vertices in S_i. Also, $d_i(i) = 0$.

These demands satisfy a useful orthogonality property, as illustrated in Figure 2.

Lemma 3.2. *For any $i, j \in I_T$, with $i \neq j$ we have that*

$$\frac{d_i(\ell(j))}{|S_{\ell(j))}|} - \frac{d_i(r(j))}{|S_{r(j)}|} = 0.$$

For $j = i$ we have

$$\frac{d_i(\ell(j))}{|S_{\ell(j))}|} - \frac{d_i(r(j))}{|S_{r(j)}|} = \beta k \left(\frac{1}{|S_{\ell(i)}|} + \frac{1}{|S_{r(i)}|} \right).$$

Proof. We first assume that $i = j$. The property clearly holds for any node $j \notin T_i$ as $d_i(\ell(j)) = d_i(r(j)) = 0$.

Consider $j \in T_i$. As $j \neq i$, suppose without loss of generality that j lies in the left subtree at i. Then the total demand leaving $\ell(j)$ is exactly

$$d_i(\ell(j)) = \sum_{u\in S_{\ell(j)}} \sum_{v\in S_{r(i)}} d_i(u, v) = |S_{\ell(j)}||S_{r(i)}| \frac{\beta k}{|S_{\ell(i)}||S_{r(i)}|} = \beta k \frac{|S_{\ell(j)}|}{|S_{\ell(i)}|}. \tag{2}$$

Similarly, $d_i(r(j)) = \beta k|S_{r(j)}|/(|S_{\ell(i)}|)$ and the claim follows. If j lies in the right subtree at i (so the demands enter j), the only difference is that sign of $d_i(\ell(j))$ is swapped.

For $j = i$, we have $d_{\ell(j)} = \beta_k$ and $d_{r(j)} = -\beta$ and the claim follows. □

The following fact will be useful later in realizing the T flow on G.

Lemma 3.3. *The total demand, either entering or leaving, across any edge $(a, b) \in E_T$, with b child of a, is at most $8\beta k \leq k/\alpha_G$.*

Proof. The only demand leaving (resp. entering) b is due to demands in D_i for nodes $i \in V_T$ that are ancestors of b and for which b lies in the left (resp. right) subtree of i. Let $d(b, i)$ be the distance between b and i. Then by Claim (3.1) and (2), the total flow leaving b and entering b due to the demands D_i is at most $\beta k|S_b|/|S_{\ell(i)}| \leq \beta k(3/4)^{\lfloor (d(b,i)-1)/2 \rfloor}$. Summing up over all $i \geq 1$, this is at most $8\beta k$. □

Now consider the total load on the edges of R by the demands $d_i(u, v)$. As R is a contraction of T, the flow on an edge $(i, j) \in E_R$, where j is a child of i, is exactly the total flow entering and leaving the node j in T, which by the Lemma above is $8\beta k$. In particular, this implies the following on the tree R.

Lemma 3.4. *Consider the multi-commodity flow in R due to commodities $d_i(u, v)$ for all $i \in I_T$ and $u, v \in V$. Then the total flow on any edge is at $8\beta k$. As each edge in R has capacity at least k, the congestion of any edge is at most $8\beta k/k = 1/\alpha_G$.*

Computing the balancing flows for each node of I_T

We now map these demands and the associated flow back to G. Recall the flow templates f_{uv} for each pair of vertices $u, v \in V$ in G, given by the oblivious routing associated with the tree R.

For each edge $e = (x, y)$ of G and node $i \in I_T$, let

$$g_i(x, y) = \sum_{u,v \in V} f_{uv}(x, y) d_i(u, v),$$

be the total (signed) flow from x to y due to the demands in the collection D_i. Observe that $g_i(x, y) = -g_i(y, x)$ for all x, y as $f_{uv}(x, y) = -f_{uv}(y, x)$ for all $u, v, x, y \in V$.

Note that the $g_i(x, y)$ only depend on the graph G (and $R, T, \{f_{uv}\}_{uv}$ that are derived from G). In particular, they do not depend on the current load vector L^t at the vertices of G, and will be fixed henceforth.

Having computed the $g_i(x, y)$, we are now ready to define the allocation strategy.

3.2 Allocation Strategy

At each time step $t + 1$, for $t = 0, 1, 2, \ldots$, we allocate the arriving ball using the following strategy.

1. Let $e = (x, y)$ be the edge where the ball arrives at time $t + 1$. Choose $i \in I_T$ with probability
$$p_e(i) := |g_i(x, y)|.$$
2. Let $L = L^t$ denote the current load vector.
Compare $\overline{L}(\ell(i))$ and $\overline{L}(r(i))$ (the average load on $S_{\ell(i)}$ and $S_{r(i)}$).
   (a) If $\overline{L}(\ell(i)) > \overline{L}(r(i))$: allocate the ball to y if $g_i(x, y) > 0$, else allocate it to x.
   (b) If $\overline{L}(\ell(i)) \leq \overline{L}(r(i))$: allocate the ball to x if $g_i(x, y) > 0$, else allocate it to y.
3. With remaining probability $1 - \sum_i p_e(i)$, allocate the ball uniformly and randomly to either x or y.

Implementation. The allocation strategy can be easily implemented to run in $O(\log n)$ time per arriving request as follows.

First, for each $e = (x, y)$, we create a table G_e of size $O(n)$, with entries for each internal node $i \in I_T$, that contains $g_i(x, y)$. This table is fixed and does not change over time.

We maintain another table F of size $O(n)$ with entries $\overline{L}(j)$ for each node $j \in V_T$ that change over time. Whenever a ball is allocated to some vertex u, we increase $\overline{L}(j)$ by $1/|S_j|$ for each node j on the path from u to the root r in T. This requires only $O(\log n)$ update operations.

Upon the request for edge e, we lookup the entry for i with probability $p_e(i)$ in the table G_e, and the entries $\overline{L}(\ell(i))$ and $\overline{L}(r(i))$ in the table F and use the allocation strategy above to allocate the ball.

4 Analysis

Our goal is to prove the bound on the gap as given by Theorem 1.1. To this end, we first bound the gap between the average load on sets corresponding to any two siblings in T. In particular, we show the following.

Lemma 4.1. *For any non-leaf node $i \in V_T$, for every constant $c > 0$,*

$$\Pr[\overline{L}(S_{\ell(i)}) - \overline{L}(S_{r(i)}) > 8(d/k)\alpha_G \, c \log n] = O(n^{-c}).$$

Let us first see how this directly implies Theorem 1.1.

Proof of Theorem 1.1. Fix any non-leaf node $i \in I_T$ and consider some child $j \in \{\ell(i), r(i)\}$ of i. Let j' be the sibling of j. We first observe that the difference between average load of siblings is no less than the difference for a parent-child pair. This follows as

$$|\overline{L}(S_j) - \overline{L}(S_{j'})| = \left| \frac{L(S_j)}{|S_j|} - \frac{L(S_i) - L(S_j)}{|S_{j'}|} \right| = \frac{|S_i|}{|S_{j'}|} |\overline{L}(S_j) - \overline{L}(S_i)| \geq |\overline{L}(S_j) - \overline{L}(S_i)|, \tag{3}$$

where we use the fact that $L(S_i) = L(S_j) + L(S_{j'})$, $|S_i| = |S_j| + |S_{j'}|$ and $|S_i| \geq |S_{j'}|$.

Fix some vertex u of G, and consider the path $u = i_0, i_1, \ldots, i_h = r$ in T from the leaf u to the root r. As $L(u) = \overline{L}(u)$ for a leaf u and $\overline{L}(r)$ is the average load over V, the load deviation for u is

$$|\overline{L}(u) - \overline{L}(r)| = \left| \sum_{g=1}^{h} \left(\overline{L}(i_{g-1}) - \overline{L}(i_g) \right) \right| \leq \sum_{g=1}^{h} \left| \overline{L}(i_{g-1}) - \overline{L}(i_g) \right|.$$

Applying Lemma 4.1 with $c > 1$, taking a union bound over the $O(n)$ edges $(i, j) \in E_T$ and using (3), we get that w.h.p. $|\overline{L}(S_i) - \overline{L}(S_j)| \leq (d/k)\alpha_G \, c' \log n]$ for all edges $(i, j) \in E_T$. As the height of T is $h_T = O(\log n)$ by Claim 3.1, this gives that, w.h.p., for every vertex u,

$$|\overline{L}(u) - \overline{L}(r)| = O((d/k)\alpha_G \log^2 n). \qquad \square$$

To prove Lemma 4.1 we will show that whenever $\overline{L}_{\ell(i)} > \overline{L}_{r(i)}$, the strategy creates sufficient drift to decrease $\overline{L}_{\ell(i)} - \overline{L}_{r(i)}$, and vice versa.

4.1 Computing the Drifts and Biases

We now compute these quantities.

Consider the ball arriving at time $t + 1$ and let $L = L^t$ denote the current load vector. To describe the allocation strategy more compactly, it will be convenient to define

$$Q_L(i) = \begin{cases} +1 & \text{if } \overline{L}(\ell(i)) \leq \overline{L}(r(i)) \\ -1 & \text{otherwise.} \end{cases}$$

For an edge $e = (x, y)$, let $q_e(x)$ denote the probability that the allocation strategy assigns the ball to x. Note that $q_e(x)$ (and most of other quantities below) depend on the vector Q_L, but we drop this for ease of notation, as we only consider the ball at $t + 1$.

Lemma 4.2. $q_e(x) = \frac{1}{2} + \frac{1}{2}\sum_i g_i(x, y))Q_L(i)$.

Proof. Recall the allocation strategy in Section 3.2 can be written as follows. Upon the request $e = (x, y)$, it samples $i \in I_T$ with probability $p_e(i) = |g_i(x, y)|$ and allocates the ball to x with probability $(1 + \text{sgn}(g_i(x, y))Q_L(i)/2$. Here $\text{sgn}(a) = 1$ if $a > 0$ and -1 otherwise. So

$$\begin{aligned} q_e(x) &= \sum_{i \in I_T} |g_i(x, y)| \left(\frac{1 + \text{sgn}(g_i(x, y))Q_L(i)}{2} \right) + \left(1 - \sum_{i \in i_T} |g_i(x, y)| \right) \frac{1}{2} \\ &= \frac{1}{2} + \frac{1}{2} \left(\sum_{i \in I_T} g_i(x, y)Q_L(i) \right). \end{aligned}$$

□

Let us define the bias towards x due to the edge $e = (x, y)$ as

$$b_e(x) := 2q_e(x) - 1 = \sum_{i \in I_T} g_i(x, y))Q_L(i).$$

We next show that the probabilities $q_e(x, y)$ lie in $[0, 1]$.

Lemma 4.3. *For an edge $e = (x, y)$, we have that $b_e(x) \in [-1, 1]$.*

Proof. As $Q_L(i)$ is ± 1,

$$|b_e(x)| \leq \sum_{i \in I_T} |g_i(x, y)| \leq \sum_{i \in I_T} \sum_{u,v} |f_{uv}(x, y)||d_i(u, v)|.$$

This is at most the total multi-commodity flow on the edge (x, y) due to the demands in D_i for all $i \in I_T$. By Lemma 3.4, and the property of Räcke decomposition, this is at most $\alpha_G \, \text{cong}(R, \vec{d}) \leq 1$. □

The biases for sets. Let us now compute the probability $q(S_i)$ that the ball at time $t + 1$ is assigned to a vertex in the set S_i corresponding to some node $i \in V_T$.

Let $N(x)$ be the set of neighbors of vertex x in G. As $q_x(e)$ is the probability of allocating ball to x when edge $e = (x, y)$ is chosen, and each e arrives uniformly with probability $1/m$,

$$q(S_i) = \frac{1}{m} \sum_{x \in S_i} \sum_{y \in N(x)} \left(\frac{1}{2} + b_x(x, y) \right) = \frac{|S_i|}{n} + \frac{1}{m} \sum_{x \in S_i} \sum_{y \in N(x)} b_x(x, y), \tag{4}$$

where we use that $\sum_{x \in S_i} \sum_{y \in N(x)} 1 = d|S_i|$ as G is d-regular, and that $m = nd/2$.

Let

$$b(S_i) := \frac{1}{m} \sum_{x \in S_i} \sum_{y \in N(x)} b_x(x, y) = q(S_i) - \frac{|S_i|}{n} \tag{5}$$

denote the bias towards set S_i, over the stationary probability $|S_i|/n$.

The following key lemma that relates the demands D_j defined on the nodes $j \in I_T$ to the bias $b(S_i)$. Recall from (1), that $d_j(i)$ is the total flow out of or into the subtree T_i due to the demands in D_j.

Lemma 4.4. *For any set S_i corresponding to a node i of T*

$$b(S_i) = \frac{1}{m} \sum_{j \in i_T} Q_L(j) d_j(i).$$

Proof. By the definition of $b_x(x,y)$ and $g_i(x,y)$,

$$\begin{aligned}\sum_{x\in S_i}\sum_{y\in N(X)} b_x(x,y) &= \sum_{x\in S_i}\sum_{y\in N(x)}\sum_{j\in i_T} g_j(x,y))Q_L(j)\\ &= \sum_{x\in S_i}\sum_{y\in N(x)}\sum_{j\in i_T} Q_L(j)\sum_{u,v} f_{uv}(x,y)d_j(u,v)\\ &= \sum_{j} Q_L(j\in i_T)\sum_{u,v} d_j(u,v)\sum_{x\in S_i}\sum_{y\in N(x)} f_{uv}(x,y).\end{aligned}$$

Noting that $\sum_{y\in N(x)} f_{uv}(x,y) = 1_{x=u} - 1_{x=v}$ for any x,

$$\sum_{x\in S_i}\sum_{y\in N(x)} f_{uv}(x,y) = \sum_{x\in S_i}(1_{x=u}-1_{x=v}) = 1_{u\in S_i} - 1_{v\in S_i}.$$

Plugging back this gives that

$$\sum_{x\in S_i}\sum_{y\in N(x)} b_x(x,y) = \sum_{j\in i_T} Q_L(j)\sum_{u,v} d_j(u,v)(1_{u\in S_i}-1_{v\in S_i}) = \sum_{j\in i_T} Q_L(j)d_j(i). \qquad \square$$

The following quantity will be crucial in the proof of Lemma 4.1.

Lemma 4.5. *For a node i and its children $\ell(i)$ and $r(i)$,*

$$\frac{q(S_{\ell(i)})}{|S_{\ell(i)}|} - \frac{q(S_{r(i)})}{|S_r(i)|} = \frac{\beta k Q_L(i)}{m}\left(\frac{1}{|S_{\ell(i)}|}+\frac{1}{|S_{r(i)}|}\right).$$

Proof. By definition for bias in (5), we have $q(S_{\ell(i)}) = |S_{\ell(i)}|/n + b(S_{\ell(i)})$ and $q(S_{r(i)}) = |S_{r(i)}|/n + b(S_{r(i)})$. By Lemma 4.4.

$$\frac{q(S_{\ell(i)})}{|S_{\ell(i)}|} - \frac{q(S_{r(i)})}{|S_r(i)|} = \frac{b(S_{\ell(i)})}{|S_{\ell(i)}|} - \frac{b(S_{r(i)})}{|S_r(i)|} = \sum_{j\in I_T}\frac{1}{m}Q_L(j)\left(\frac{d_j(\ell(i))}{|S_{\ell(i)}|} - \frac{d_j(r(i))}{|S_{r(i)}|}\right).$$

By Lemma 3.2, for any $i\neq j$ the right side is zero, the only contribution is due to $j=i$ which is

$$\frac{\beta k Q_L(i)}{m}\left(\frac{1}{|S_{\ell(i)}|}+\frac{1}{|S_{r(i)}|}\right). \qquad \square$$

4.2 Proving the concentration

We now prove Lemma 4.1. We first develop a simple concentration lemma to be used in our analysis.

2-point concentration. The set up is the following. There are two bins, 1 and 2, associated with so called steady state probabilities π_1, π_2 which satisfy $\pi_1+\pi_2 = 1$. At each time step a ball arrives and let ℓ_1^t, ℓ_2^t denote the loads of the bins at the end of time t. In addition, fix $b \le \min(\pi_1,\pi_2)/2$, and assume that the allocation process of the ball arrives at $t+1$ is as follows.

A parameter $b_t > b$ also satisfying $b_t \le \max(\pi_1,\pi_2)/2$ is decided independently from the future of the process. Then, if $\ell_1^t/\pi_1 > \ell_2^t/\pi_2$ the ball is allocated to either bin 1 with probability $\pi_1 - b_t$ or to bin 2 with probability $\pi_2 + b_t$. Otherwise, if $\ell_1^t/\pi_1 \le \ell_2^t/\pi_2$, the ball is allocated to bin 1 with probability $\pi_1 + b_t$, or bin 2 with probability $\pi_2 - b_t$.

For any time t, let $d(t) = \ell_1^t/\pi_1 - \ell_2^t/\pi_2$ denote the normalized difference in the loads of the bin.

Lemma 4.6. *For any time t and any $x \ge 0$ it holds that*

$$\Pr[|d(t)| \le x/b] = O\big(\exp(-x/8)\big).$$

Proof. We define the potential function $\Phi(t) = \cosh(\alpha d(t))$, where $\alpha = b/8$. So $\Phi(0) = 1$ initially at $t = 0$.

Fix a time t and let $\Delta\Phi = \Phi(t+1) - \Phi(t)$ and denote $d = d(t)$ and $\Delta d = d(t+1) - d(t)$. Then, by Taylor expansion, and using $(d/dx)\cosh(x) = \sinh(x)$ and $(d/dx)\sinh x = \cosh x$,

$$\Delta\Phi = \sinh(\alpha d)\Big(\alpha\Delta d + (\alpha\Delta d)^3/3! + \dots\Big) + \cosh(\alpha d)\Big(\alpha(\Delta d)^2/2! + (\alpha\Delta d)^4/4! + \dots\Big).$$

Let $\gamma = 1/\pi_1 + 1/\pi_2$. As $b \le \max(\pi_1, \pi_2)/2$ we have that $\gamma b \le 1$. As either ℓ_1^t or ℓ_2^t rises by 1, we have $|\Delta d| \le \gamma$, and hence $|\alpha\Delta d| \le 1/2$. Bounding $|(\alpha\Delta d)^i| \le (\alpha\Delta d)^2 2^{-i+2}$ for $i \ge 2$, and using that $\cosh x - 1 \le |\sinh(x)| \le \cosh x$ for all x,

$$\Delta\Phi = \sinh(\alpha d)(\alpha\Delta d) + \cosh(\alpha d)(\alpha\Delta d)^2).$$

For $d > 0$,

$$\mathbb{E}(\Delta d) = (\pi_1 - b_t)\frac{1}{\pi_1} - (\pi_2 + b_t)\frac{1}{\pi_2} = -b_t\left(\frac{1}{\pi_1} + \frac{1}{\pi_2}\right) = -b_t\gamma \le -b\gamma$$

and otherwise, for $d \le 0$, we have $\mathbb{E}[\Delta d] = b_t\gamma \ge b\gamma$.

For $d > 0$, we have

$$\mathbb{E}[(\Delta d)^2] = (\pi_1 - b_t)\left(\frac{1}{\pi_1}\right)^2 + (\pi_2 + b)\left(\frac{1}{\pi_2}\right)^2 = \gamma + b_t(-1/\pi_1^2 + 1/\pi_2^2) \le \gamma + b_t\gamma^2 \le 2\gamma,$$

where the last step uses the fact that $b\gamma \le 1$. The same bound also holds for $d \le 0$.

This gives

$$\begin{aligned}
\mathbb{E}[\Delta\Phi(t) \mid \Phi(t)] &\le \alpha\sinh(\alpha d)\mathbb{E}[\Delta d] + 2\alpha^2\cosh(\alpha d)\mathbb{E}[(\Delta d)^2] \\
&\le -\alpha|\sinh(\alpha d)|b\gamma + 4\alpha^2\gamma\cosh(\alpha d) \\
&\le -\alpha b\gamma(\Phi(t) - 1) + \alpha b\gamma\Phi(t)/2 = -\alpha b\gamma(\Phi(t) - 2)/2.
\end{aligned}$$

Thus, taking expectation and using $\Delta(\Phi(t)) = \Phi(t+1) - \Phi(t)$ we obtain

$$\mathbb{E}[\phi(t+1)] \le \alpha b\gamma + (1 - \alpha b\gamma/2)\mathbb{E}[\phi(t)].$$

Taking induction over t and recalling that $\Phi(0) = 1$ we obtain

$$\mathbb{E}[\Phi(t)] \le 2\big(1 - (1 - \tfrac{\alpha b\gamma}{2})^t\big) + (1 - \tfrac{\alpha b\gamma}{2})^t \le 2.$$

The result now follows by applying Markov's inequality to $\Phi(t)$. □

Proof of Lemma 4.1.

We apply the set up above to bound $|\overline{L}(\ell(i)) - \overline{L}(r(i))|$. Let bin 1 correspond to $S_{\ell(i)}$ and bin 2 to $S_{r(i)}$. We only focus on the steps when the ball is allocated to some vertex in S_i. Let $\pi_1 = |S_{\ell(i)}|/|S|$ and $\pi_2 = |S_{r(i)}|/|S|$ so that $\pi_1 + \pi_2 = 1$. The probability of allocating the ball to $S_{\ell(i)}$ conditioned on it being allocated to S_i is $q(S_{\ell(i)})/q(S_i)$, and similarly $q(S_{r(i)})/q(S_i)$ for $S_{r(i)}$.

Writing $\pi_1 - b_t = q(S_{\ell(i)})/q(S_i)$ and $\pi_2 + b_t = q(S_{r(i)})/q(S_i)$, gives that

$$b_t\left(\frac{1}{\pi_1} + \frac{1}{\pi_2}\right) = \frac{q(S_{r(i)})}{\pi_2 q(S_i)} - \frac{q(S_{\ell(i)})}{\pi_1 q(S_i)} = \frac{|S_i|}{q(S_i)}\left(\frac{q(S_{r(i)})}{|S_{r(i)}|} - \frac{q(S_{\ell(i)})}{|S_{\ell(i)}|}\right),$$

which by Lemma 4.5 equals

$$-\frac{|S_i|}{q(S_i)}\frac{\beta k Q_L(i)}{m}\left(\frac{1}{|S_{\ell(i)}|} + \frac{1}{|S_{r(i)}|}\right) = -\frac{1}{q(S_i)}\frac{\beta k Q_L(i)}{m}\left(\frac{1}{\pi_1} + \frac{1}{\pi_2}\right),$$

which gives that

$$b_t = -\frac{\beta k Q_L(i)}{mq(S_i)}.$$

Now, $q(S_i) = |S_i|/n + b(S_i) \le 2|S_i|/n$, where be use that $b(S_i) \le 8\beta k/m = O\big(1/(\alpha_G n)\big) = O(1/n)$.

$$|b_t| \ge \frac{\beta k n Q_L(i)}{2m|S_i|} = \beta k/(d|S_i|) =: b. \tag{6}$$

Applying Lemma 4.6, we obtain

$$\Pr\left[\left|\frac{L(S_{\ell(i)})}{\pi_1} - \frac{L(S_{r(i)})}{\pi_2}\right| \ge x/b\right] = O\big(\exp(-x/8)\big).$$

As $\pi_1 = |S_{\ell(i)}|/|S_i|, \pi_2 = |S_{\ell(i)}|/|S_i|$ and using the definition of b in (6), gives that for any $x > 0$,

$$\Pr\left[|\overline{L}(S_{\ell(i)}) - \overline{L}(S_{r(i)})| \ge 8x(d/k)\alpha_G\right] = O\big(\exp(-x/8)\big),$$

which implies the desired claim.

5 Lower Bounds

We describe various simple but instructive lower bounds. First we prove Theorem 1.2. Then, we show an $\Omega(\log(n)/\log\log n)$ lower bound on the upper gap for bounded degree graphs so that on expanders, the upper gap is roughly of the same order as the gap (unlike the case of complete graphs).

We also remark that [PTW15] showed that even for complete graphs, under the $1+\beta$ choice model with $\beta = 1/2$, the upper gap is typically $\Omega(\log n)$ which is similar to the gap, unlike for the 2-choice model where the upper gap $O(\log\log n)$ and gap is $\Omega(\log n)$.

5.1 Proof of Theorem 1.2

We show for any d-regular k edge-connected graph G, under any allocation strategy, the gap is $\Omega(d/k + \log n)$ with at least constant probability.

The $\Omega(\log n)$ bound follows from the following folklore argument. Fix any time t and consider any interval I of $O(n\log n)$ steps just before t. For a fixed vertex v and time step, the probability that some edge incident to it is chosen is $d/m = 2/n$. So with $\Omega(1)$ probability, there is some vertex v for which no incident edge is chosen during I. So during I, the load of v cannot change under any strategy, while the average load increases by $\log n$.

Hence, to prove Theorem 1.2, it suffices to show the following.

Lemma 5.1. *Let G be any d-regular, k-connected graph. Then for any allocation strategy, and at any time t the gap is $\Omega(d/k)$ with constant probability.*

Proof. Let $C = (S, \overline{S})$ be some minimum cut in G, and let S be the larger side with $|S| \ge n/2$. Fix a time t and consider some time interval I of length T just before t. We specify T later. Let Y be the number of edges requested with both endpoints in S. As an edge is requested uniformly at random at each time, and as there are $(d|S| - k)/2$ such edges, we obtain

$$\mathbb{E}[Y] = (d|S| - k)/2 \cdot T/m = (|S| - k/d)T/n.$$

As $k \le d$ and $|S| \ge n/2$, this at least $T/3$, provided that $n \ge 6$.

By standard estimates, with at least constant positive probability, we have $Y \ge \mathbb{E}[Y] + \sqrt{T}$. Conditioned on this event, the average load on vertices in S during I exceeds the stationary load of T/n by at least

$$\frac{Y}{|S|} - \frac{T}{n} \ge \frac{\mathbb{E}[Y] + \sqrt{T}}{|S|} - \frac{T}{n} = \frac{(|S|d - k)T}{|S|dn} + \frac{\sqrt{T}}{|S|} - \frac{T}{n} = \frac{\sqrt{T}}{|S|} - \frac{kT}{nd|S|}.$$

Choosing $T = cn^2d^2/k^2$ for small enough c and as $|S| \ge n/2$, this is $\Omega(d/k)$. □

5.2 Upper gap for bounded degree graphs

Next we show that even for bounded degree expanders, under the greedy strategy the maximum load at time t is typically $t/n + \tilde{\Omega}(\log n)$. This is unlike for complete graphs where this deviation is $O(\log \log n)$.

Lemma 5.2. *For any d-regular G with $d = O(1)$, at any time t and for any allocation strategy, with constant probability the maximum vertex load is $t/n + \Omega(\log n / \log \log n)$.*

Proof. Consider an interval of requests of length $dn/2$. If at the beginning of the interval there was a vertex of height $\log n/4d \log \log n$ above the average, then we are done. Otherwise, by Markov's inequality, at least $\frac{1}{2}$ of the edges have initial height equal or higher than $\log n/4 \log \log n$ below average.

As the number of these edges is at least $m = nd/4$ and they are requested uniformly at random, this is like throwing m balls uniformly in $m/2$ bins and hence some edge e is requested $s = \Theta(\log m / \log \log m)$ with constant probability. So for any allocation strategy, the load on one of the endpoint of e increases by at least $s/2$ while the average load only increases by $d/2 = O(1)$ during the interval. □

Concluding Remarks

In this paper we have shown how to obtain in the graphical two-choice model an asymptotically polylogarithmic load for any d-regular d-connected graph. One may ask if this strategy offers any improvement in feasible sizes. To show that it indeed does, we conclude the paper with with a simulation result of our strategy for large cycles, compared against the asymptotic behavior of the greedy choice model (Figure 3).

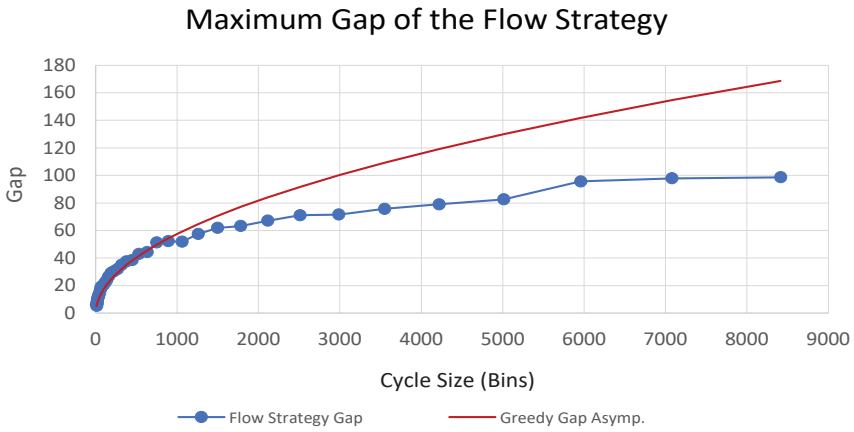

Figure 3: The gap for our flow algorithm on cycles for the best possible Räcke tree, averaged over 32 runs of $n^{2.5}$ balls for cycles of sizes $n = 10$ to 5000.

Acknowledgments

We heartily thank Guy Bensky and Shlomo Ron for their coding assistance in making our simulations.

References

[ABKU94] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. In *Symposium on theory of computing*, pages 593–602, 1994.

[ACF+04] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Räcke. Optimal oblivious routing in polynomial time. *J. Comput. Syst. Sci.*, 69(3):383–394, 2004.

[ACMR98] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. Parallel randomized load balancing. *Random Structures & Algorithms*, 13(2):159–188, 1998.

[AF09] Reid Andersen and Uriel Feige. Interchanging distance and capacity in probabilistic mappings. *CoRR*, abs/0907.3631, 2009.

[AK14] Roc Armenter and Miklós Koren. A balls-and-bins model of trade. *American Economic Review*, 104(7):2127–51, 2014.

[ANS] Dan Alistarh, Giorgi Nadiradze, and Amirmojtaba Sabour. Dynamic averaging load balancing on cycles. In *ICALP 2020*, volume 168 of *LIPIcs*, pages 7:1–7:16.

[BCSV06] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM Journal on Computing*, 35(6):1350–1385, 2006.

[CFM+98] Richard Cole, Alan Frieze, Bruce M. Maggs, Michael Mitzenmacher, Andréa W Richa, Ramesh Sitaraman, and Eli Upfal. On balls and bins with deletions. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 145–158. Springer, 1998.

[DR96] Devdatt P. Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *BRICS Report Series*, 3(25), 1996.

[God08] P. Brighten Godfrey. Balls and bins with structure: balanced allocations on hypergraphs. In *Symposium on Discrete algorithms*, pages 511–517, 2008.

[HHR03] Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *Symposium on Parallel algorithms and architectures*, pages 34–43, 2003.

[KLadH96] Richard M. Karp, Michael Luby, and F. Meyer auf der Heide. Efficient pram simulation on a distributed memory machine. *Algorithmica*, 16(4):517–542, 1996.

[KP06] Krishnaram Kenthapadi and Rina Panigrahy. Balanced allocation on graphs. In *SODA*, volume 6, pages 434–443, 2006.

[Mit91] Michael Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, Harvard University, 1991.

[MRS01] Michael Mitzenmacher, Andrea W. Richa, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. *Combinatorial Optimization*, 9:255–304, 2001.

[PTW15] Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the $(1 + \beta)$-choice process. *Random Struct. Algorithms*, 47(4):760–775, 2015.

[Räc02] Harald Räcke. Minimizing congestion in general networks. In *Foundations of Computer Science, FOCS*, pages 43–52, 2002.

[Räc08] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Symposium on Theory of Computing*, pages 255–264, 2008.

[RS98] Martin Raab and Angelika Steger. "balls into bins" — a simple and tight analysis. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer, 1998.

[RST14] Harald Räcke, Chintan Shah, and Hanjo Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Symposium on Discrete Algorithms, SODA*, pages 227–238, 2014.

[Ste96] Volker Stemann. Parallel balanced allocations. In *Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, pages 261–269, 1996.

[TW14] Kunal Talwar and Udi Wieder. Balanced allocations: A simple proof for the heavily loaded case. In *International Colloquium on Automata, Languages, and Programming*, pages 979–990. Springer, 2014.

[Vöc03] Berthold Vöcking. How asymmetry helps load balancing. *Journal of the ACM (JACM)*, 50(4):568–589, 2003.

[Wie17] Udi Wieder. Hashing, load balancing and multiple choice. *Foundations and Trends in Theoretical Computer Science*, 12:275–379, 2017.