Ultra-Fast Load Balancing on Scale-Free Networks

Karl Bringmann[1], Tobias Friedrich[2,3], Martin Hoefer[4], Ralf Rothenberger[2,3] and Thomas Sauerwald[5]

[1] Institute of Theoretical Computer Science, ETH Zurich, Switzerland
[2] Friedrich-Schiller-Universitt Jena, Germany
[3] Hasso Plattner Institute, Potsdam, Germany
[4] Max Planck Institute for Informatics, Saarbrcken, Germany
[5] University of Cambridge, United Kingdom

Abstract. The performance of large distributed systems crucially depends on efficiently balancing their load. This has motivated a large amount of theoretical research how an imbalanced load vector can be smoothed with local algorithms. For technical reasons, the vast majority of previous work focuses on regular (or almost regular) graphs including symmetric topologies such as grids and hypercubes, and ignores the fact that large networks are often highly heterogenous.
We model large scale-free networks by Chung-Lu random graphs and analyze a simple local algorithm for iterative load balancing. On n-node graphs our distributed algorithm balances the load within $\mathcal{O}((\log\log n)^2)$ steps. It does not need to know the exponent $\beta \in (2,3)$ of the power-law degree distribution or the weights w_i of the graph model. To the best of our knowledge, this is the first result which shows that load-balancing can be done in double-logarithmic time on realistic graph classes.

1 Introduction

Load balancing. Complex computational problems are typically solved on large parallel networks. An important prerequisite for their efficient usage is to balance the work load efficiently. Load balancing is also known to have applications to scheduling [17], routing [6], numerical computation such as solving partial differential equations [16, 19], and finite element computations [13]. In the standard abstract formulation of load balancing, processors are represented by nodes of a graph, while links are represented by edges. The objective is to balance the load by allowing nodes to exchange loads with their neighbors via the incident edges. Particularly popular are decentralized, round-based iterative algorithms where a processor knows only its current load and that of the neighboring processors. We focus on *diffusive* load balancing strategies, where each processor decides how many jobs should be sent and balances its load with its neighbors in each round. As the degrees of the topologies of many networks follow heavy tailed statistics, our main interest lies on *scale-free* networks.

Diffusion. On networks with n nodes, our balancing model works as follows: At the beginning, each node i has some work load $x_i^{(0)}$. The goal is to obtain (a good approximation of) the balanced work load $\overline{x} := \sum_{i=1}^{n} x_i^{(0)}/n$ on all nodes. On heterogenous graphs with largely varying node degrees it is natural to consider a multiplicative quality measure: We want to find an algorithm which achieves $\max_i x_i^{(t)} = \mathcal{O}(\overline{x})$ at the earliest time t possible. Load-balancing is typically considered *fast* if this can be achieved in time logarithmic in the number of nodes. We aim at double-logarithmic time, which we call *ultra-fast* (following the common use of the superlative "ultra" for double-logarithmic bounds [4, 10, 18]).

The diffusion model was first studied by Cybenko [6] and, independently, Boillat [1]. The standard implementation is the *first order scheme* (FOS), where the load vector is multiplied with a diffusion matrix $\mathbf{P}$ in each step. For regular graphs with degree d, a common choice is $\mathbf{P}_{ij} = 1/(d+1)$ if $\{i,j\} \in E$. Already Cybenko [6] in 1989 shows for regular graphs a tight connection between the convergence rate of the diffusion algorithm and the absolute value of the second largest eigenvalue $\lambda_{\max}$ of the diffusion matrix $\mathbf{P}$. While FOS can be defined for non-regular graphs, its convergence is significantly affected by the loops which are induced by the degree discrepancies. Regardless of how the damping factor is chosen, FOS requires $\Omega(\log n)$ rounds on a broad class of non-regular graphs. For a proof and discussion of this statement we refer to the full version of this paper.

Scale-free networks. Many real-world graphs have a power law degree distribution, meaning that the number of vertices with degree k is proportional to $k^{-\beta}$, where β is a constant intrinsic to the network. Such networks are synonymously called scale-free networks and have been widely studied. As a model for large scale-free networks we use the *Chung-Lu random graph model* with a power-law degree distribution with exponent $\beta \in (2,3)$. (See Section 2 for a formal definition.) This range of β's is typically studied as many scale-free networks (e.g. co-actors, protein interactions, internet, peer-to-peer [15]) have a power law exponent with $2 < \beta < 3$. It is known that the diameter of this graph model is $\Theta(\log n)$ while the average distance between two vertices is $\Theta(\log \log n)$ [3].

Results. Scale-free networks are omnipresent, but surprisingly few rigorous insights are known about their ability to efficiently balance load. Most results and developed techniques for theoretically studying load balancing only apply to regular (or almost-regular) graphs. In fact, we cannot hope for ultra-fast balancing on almost-regular graphs: Even for expander graphs of maximum degree d, there is a general lower bound of $\Omega(\log n/\log d)$ iterations for *any* distributed load balancing algorithms (for a proof of this statement we refer to the full version of this paper). Our main result (cf. Theorem 2.1) shows that within $\mathcal{O}((\log \log n)^2)$ steps, our simple local balancing algorithm (cf. Algorithm 1) can balance the load on a scale-free graph with high probability. The algorithm assumes that the initial load is only distributed on nodes with degree $\Omega(\mathrm{polylog}\, n)$ (cf. Theorem 2.2), which appears to be a natural assumption in typical load balancing applications. As the diameter of the graph is $\Theta(\log n)$, ultra-fast balancing is impossible if the initial load is allowed on arbitrary vertices. As standard FOS

requires $\Omega(\log n)$ rounds, our algorithm uses a different, novel approach to overcome these restrictions.

Algorithm. The protocol proceeds in waves, and each wave (roughly) proceeds as follows. First, the remaining load is balanced within a core of high-degree nodes. These nodes are known to compose a structure very similar to a dense Erdős-Rényi random graph and thereby allow very fast balancing. Afterwards, the load is disseminated into the network from high- to low-degree nodes. Each node absorbs some load and forwards the remaining to lower-degree neighbors. If there are no such neighbors, the excess load is routed back to nodes it was received from. In this way, the load moves like a wave over the graph in decreasing order of degree and then swaps back into the core. We will show that each wave needs $\mathcal{O}(\log \log n)$ rounds. The algorithm keeps initiating waves until all load is absorbed, and we will show that only $\mathcal{O}(\log \log n)$ waves are necessary.

Techniques. There are a number of technical challenges in our analysis, mostly coming from the random graph model, and we have to develop new techniques to cope with them. For example, in scale-free random graphs there exist large sparse areas with many nodes of small degree that result in a high diameter. A challenge is to avoid that waves get lost by pushing too much load deep into these periphery areas. This is done by a partition of nodes into layers with significantly different degrees and waves that proceed only to neighboring layers. To derive the layer structure, we classify nodes based on their realized degrees. However, this degree might be different from the expected degree corresponding to the weights w_i of the network model, which is unknown to the algorithm. This implies that nodes might not play their intended role in the graph and the analysis (cf. Definition 4.2). This can lead to poor spread and the emersion of a few, large single loads during every wave. Here we show that several types of "wrong-degree" events causing this problem are sufficiently rare, or, more precisely, they tend to happen frequently only in parts of the graph that turn out not to be critical for the result. At the core, our analysis adjusts and applies fundamental probabilistic tools to derive concentration bounds, such as a variant of the method of bounded variances (cf. Theorem 4.1).

2 Model, Algorithms, and Formal Result

Chung-Lu random graph model. We consider random graphs $G = (V, E)$ as defined by Chung and Lu [3]. Every vertex $i \in V = \{1, \dots, n\}$ has a weight w_i with $w_i := \frac{\beta-2}{\beta-1} d n^{1/(\beta-1)} i^{-1/(\beta-1)}$ for $i = 1, 2 \dots, n$. The probability for placing an edge $\{i, j\} \in E$ is then set to $\min\{w_i\, w_j / W, 1\}$ with $W := \sum_{i=1}^{n} w_i$. This creates a random graph where the expected degrees follow a power-law distribution with exponent $\beta \in (2, 3)$, the maximum expected node degree is $\frac{\beta-2}{\beta-1} d n^{1/(\beta-1)}$ and d influences the average expected node degree Chung and Lu [3]. The graph has a *core* of densely connected nodes which we define as

$$C := \left\{ i \in V \colon \deg_i \geqslant n^{1/2} - \sqrt{n^{1/2} \cdot (c+1) \ln n} \right\}.$$

Algorithm 1: Balance load in waves from core to all other nodes

```
repeat
    for phase t ← 1 to log log n do
        for 32/(3−β) rounds do                          // 1. diffusion on the core
            Nodes v with deg(v) ⩾ ω_0 perform diffusion with P = D^{-1}A
        for L rounds do                                 // 2. downward propagation
            Every node absorbs at most m/nt^2 load.
            All remaining load is forwarded in equal shares to neighbors on
            the next lower layer.
        for L rounds do                                 // 3. upward propagation
            All nodes send their load back to the the next higher layer over
            the edges they received it from. The distribution of load
            amongst these edges can be arbitrary.
until terminated;
```

Distributing the load in waves. Our main algorithm is presented in Algorithm 1. It assumes that an initial total load of m resides exclusively on the core C of the network. The first rounds are spend on simple diffusion on the core with diffusion matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the degree matrix. Afterwards, the algorithm pushes the load to all other nodes in waves from the large to the small degree nodes and the other way around. To define the direction of the waves, the algorithm partitions the nodes into layers, where on layer k we have all nodes v of degree $\deg_v \in (\omega_k, \omega_{k-1}]$, where $\omega_0 = n^{1/2} - \sqrt{n^{1/2} \cdot (c+1) \ln n}$ and $\omega_{k+1} = \omega_k^{1-\varepsilon}$ for a constant

$$0 < \varepsilon < \min\left\{ \tfrac{(3-\beta)}{(\beta-1)},\ \tfrac{\beta-2}{3},\ \tfrac{1}{2}\left(1 - \sqrt{\tfrac{3}{\beta+1}}\right)\right\}.$$

For every layer k we have $\omega_k > 2^{\frac{1}{\varepsilon(\beta-1)}}$. The last layer ℓ is the first, for which $\omega_\ell \leqslant 2^{\frac{1}{\varepsilon(\beta-1)}}$ holds. In this case, we define the interval simply to include all nodes with degree less than $\omega_{\ell-1}$. Note that in total we obtain at most $L := \frac{1}{\log(1/(1-\varepsilon))}\left(\log\log n + \log\frac{\varepsilon(\beta-1)}{2}\right)$ layers. To choose an appropriate ε, we have to know lower and upper bounds on β. These bounds are either known or can be chosen as constants arbitrarily close to 2 and 3. The algorithm therefore does not need to know the precise β. Our main result is then as follows.

Theorem 2.1. *Let $G = (V, E)$ be a Chung-Lu random graph as defined above. For any load vector $x^{(0)} \in \mathbb{R}^n_{\geqslant 0}$ with support only on the core C of the graph, there is a $\tau = \mathcal{O}((\log\log n)^2)$ such that for all steps $t \geqslant \tau$ of Algorithm 1, the resulting load vector $x^{(t)}$ fulfills $x_u^{(t)} = \mathcal{O}(\overline{x})$ for all $u \in V$ w. h. p.*[1]

[1] w. h. p. is short for "with high probability". We use w.h.p. to describe events that hold with probability $1 - n^{-c}$ for an arbitrary large constant c.

Reaching the core. Algorithm 1 and Theorem 2.1 above require that the initial total load resides exclusively on the core C of the network. As the diameter of the network is $\Theta(\log n)$ [3], we cannot hope to achieve a double-logarithmic balancing time if all the initial load starts at an arbitrary small and remote vertex. However, we can allow initial load on all nodes with at least some polylogarithmic degree by adding an *initial phase* in which all nodes send all their load to an arbitrary neighbor on the next-highest layer. This initial local routing phase succeeds if all nodes with at least this polylogarithmic degree have at least one neighbor on the next-highest layer. The following theorem, the proof of which can be found in the full version of this paper, formalizes this result, while the rest of the paper proves Theorem 2.1.

Theorem 2.2. *Let $G = (V, E)$ be a Chung-Lu random graph as defined above. For any load vector $x^{(0)} \in \mathbb{R}^n_{\geqslant 0}$ with support only on nodes with degree $\Omega((\log n)^{\max(3,2/(3-\beta))}$, the initial phase reaches after $L = \Theta(\log\log n)$ steps a load vector $x^{(L)}$ such that $x_u^{(L)}$ has support only on the core C w. h. p.*

3 Analysis of Load Balancing on the Core

We start our analysis of Algorithm 1 with its first step, the diffusion on the core. Recall the definition of the core C of the network and consider the core subgraph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ induced by C.

Lemma 3.1. *The core subgraph $\widetilde{G}$ of G fulfills*

$$|1 - \lambda_k(L)| \leqslant \Theta\left(\frac{\sqrt{(c+1)\ln(4n)}}{n^{(3-\beta)/4}} + \frac{(2c\ln n)^{1/4}}{n^{1/8}}\right)$$

for all eigenvalues $\lambda_k(L) > \lambda_{\min}(L)$ of the normalized Laplacian $L(\widetilde{G})$ w. h. p.

The proof of Lemma 3.1 is based on Theorem 2 from [12] and can be found in the full version of this paper. The following lemma states that after only a constant number of diffusion rounds in $\widetilde{G}$, the load of node $v \in C$ is more or less equal to $m \cdot w_v/W_0$.

Lemma 3.2. *After $\frac{32}{3-\beta}$ rounds of diffusion with $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ in the core subgraph $\widetilde{G}$, each node $v \in C$ has a load of at most $\mathcal{O}\left(w_y / \sum_{x\in C} w_x\right)$ w. h. p.*

The proof of Lemma 3.2 uses eq. 12.11 from [14] and can be found in the full version, too. An implication of the lemma is, that there is a constant $\varepsilon_0 > 0$ such that each node $v \in C$ has a load of at most $(1+\varepsilon_0)\frac{w_v}{W_0}m$ after the first phase of Algorithm 1.

4 Analysis of Top-Down Propagation

We continue our analysis of Algorithm 1. This section studies the downward/upward propagation.

Many of the proofs in this section are based on the following variant of the method of bounded variances [8], which might be useful in its own respect.

Theorem 4.1. *Let $X_1, \dots, X_n$ be independent random variables taking values in $\{0,1\}$, and set $\mu := \mathbf{E}\left[\sum_{i=1}^{n} X_i\right]$. Let $f := f(X_1, \dots, X_n)$ be a function satisfying*

$$|f| \leqslant M,$$

and consider an error event $\mathcal{B}$ such that for every $\mathbf{X}_n \in \overline{\mathcal{B}}$

$$|f(\mathbf{X}_n) - f(\mathbf{X}'_n)| \leqslant c$$

for every $\mathbf{X}'_n$ that differs in only one position X_i from $\mathbf{X}_n$, and for some $c > 0$. Then for any $0 \leqslant t \leqslant c\mu$ we have

$$\Pr\left[\left|f - \mathbf{E}\left[f\right]\right| > t + \tfrac{(2M)^2}{c}\Pr[\mathcal{B}]\right] \leqslant \tfrac{2M}{c}\Pr[\mathcal{B}] + 2\exp\left(-\tfrac{t^2}{16c^2\mu}\right).$$

The proof of this theorem closely follows the one of the method of bounded variances as can be found in [8]. For the sake of brevity, however, all proofs are omitted in this version of the paper and interested readers are referred to the full version.

First note that our algorithm deals with a random graph and therefore it might happen that some of the nodes' neighborhood look significantly different from what one would expect by looking at the expected values. We call these nodes *dead-ends* as they can not be utilized to effectively forward load. This definition will be made precise in Definition 4.2 below.

Only for the sake of analysis we assume that dead-ends do not push load to neighbors on the next lowest layer, but instead keep all of it. In reality the algorithm does *not* differentiate between nodes which are dead-ends and nodes which are no dead-ends. We also assume in this section that nodes do not consume any load during the top-down distribution.

The main goal of this section is twofold. We first show that no node which is not a dead-end, gets too much load. Then we show that the total load on all dead-ends from the core down to a layer with nodes of a certain constant degree is at most a constant fraction of the total load. The converse means that at least a constant fraction of load reaches the nodes of the last layer we are considering.

We define $V_k = \{v \mid w_v \in (\omega_k, \omega_{k-1}]\}$ as the set of nodes on layer k and $n_k = |V_k|$. Let $W_k = \sum_{v \in V_i} w_v$ be the total weight of nodes in layer k. Let $\gamma := \frac{1}{2}\left(d\frac{\beta-2}{\beta-1}\right)^{\beta-1}$. From the given weight sequence and the requirements $\omega_k > 2^{\frac{1}{\varepsilon(\beta-1)}}$ and $\omega_k < n^{1/(\beta-1)}$, we can easily derive the following bounds. For all $0 \leqslant k < \ell$ it holds that $\frac{\gamma}{2} \cdot n\omega_k^{1-\beta} \leqslant n_k \leqslant 4\gamma \cdot n\omega_k^{1-\beta}$.This implies $W_k \geqslant \frac{\gamma}{2} \cdot n\omega_k^{2-\beta}$.Let $\overline{d} = \frac{W}{n}$ the expected average degree.

For a node $v \in V_k$ we consider two partial degrees. Let D_v^h be the number of edges to nodes in the higher layer $k-1$, and D_v^ℓ is the number of edges to nodes

in the lower layer $k+1$. Note that D_v^h and D_v^ℓ are random variables, composed of sums of independent Bernoulli trials:

$$D_v^h = \sum_{u \in V_{k-1}} \mathsf{Ber}\left(\frac{w_v \cdot w_u}{W}\right) \quad \text{and} \quad D_v^\ell = \sum_{u \in V_{k+1}} \mathsf{Ber}\left(\frac{w_v \cdot w_u}{W}\right) \ .$$

In our proofs we will apply several well-known Chernoff bounds which use the fact that partial degrees are sums of independent Bernoulli trials.

We now define four properties which will be used throughout the analysis.

Definition 4.2. *A node $v \in V$ is a* dead-end *if one of the following holds:*

- ⟨**D1**⟩ **In-/Out-degree:** *A node $v \in V_k$ has this property if either $|D_v^h - \mathbf{E}\left[D_v^h\right]| \geqslant \mathbf{E}\left[D_v^h\right]^{2/3}$ or $|D_v^\ell - \mathbf{E}\left[D_v^\ell\right]| \geqslant \mathbf{E}\left[D_v^\ell\right]^{2/3}$.*
- ⟨**D2**⟩ **Wrong layer:** *A node $v \in V_k$ has this property if it has a degree that deviates by at least $w_v^{2/3}$ from its expected degree.*
- ⟨**D3**⟩ **Border:** *A node $v \in V_k$ has this property if it does not fulfill property ⟨D2⟩ and if it is of weight at least $\omega_{k-1} - \omega_{k-1}^{2/3}$ or at most $\omega_k + \omega_{k-1}^{2/3}$ and if it is assigned to the wrong layer.*
- ⟨**D4**⟩ **Induced Out-degree:** *A node $v \in V_k$ has this property if it fulfills none of the properties ⟨D1⟩ – ⟨D3⟩ and if it has at least $(\omega_k W_{k+1}/W)^{2/3}$ many lower-layer neighbors with properties ⟨D2⟩ or ⟨D3⟩.*

The next lemma shows that for a non-dead-end node $v \in V_k$ the received load x_v in phase k is almost proportional to the "layer-average load" $m \cdot w_v / W_k$. For dead-ends, the received load can be higher, but the probability to receive significantly higher load is small.

Lemma 4.3. *For $v_k \in V_k$ and the received load x_v in phase k the following holds. If v is not a dead-end,*

$$x_v \leqslant (1+\varepsilon_k) \cdot m \cdot \frac{w_v}{W_k} \ ,$$

where for every layer k the error term ε_k is given by

$$(1+\varepsilon_k) = (1+\varepsilon_{k-1}) \cdot (1+\mathcal{O}(\omega_k^{-1+\beta/3})) \cdot (1+\mathcal{O}(\omega_k^{-(3-\beta)/6})) \ ,$$

so $\varepsilon_k \leqslant \varepsilon_{k+1}$ and $\varepsilon_k = \mathcal{O}(1)$.

Now we want to show that on each layer with sufficiently large constant weight at most a small fraction of the total load remains on dead-ends. To do so, we show that for each property ⟨D1⟩ – ⟨D4⟩ the nodes with these properties only contribute a small enough fraction to the total dead-end load of each layer. We begin by bounding the contribution of ⟨D1⟩-nodes to the total dead-end load.

Lemma 4.4. *If $\varepsilon \leqslant (3-\beta)/(\beta-1)$ and $\omega_k > \left(\frac{2\bar{d}}{\gamma}\left(\frac{1}{2e-1}\right)^3\right)^{2/(3-\beta)}$, the probability that a node $v \in V_k$ is a ⟨D1⟩-node is at most $2\exp(-c \cdot \omega_k^{(3-\beta)/6})$, for $c = \frac{1}{4}\left(\frac{\gamma}{2\bar{d}}\right)^{1/3}$.*

An implication of the former lemma is that there are no ⟨D1⟩-nodes on layers with weight at least polylog(n). Now that we have an understanding of which layers actually contain ⟨D1⟩-nodes, we can start to derive high probability upper bounds on the total load that is left on these nodes throughout the top-down phase.

Lemma 4.5. *If $v \in V_k$ is a ⟨D1⟩-node, then*

$$\mathbf{Pr}\left[x_v \geqslant \alpha \cdot \frac{m \cdot w_v}{W_k} \right] < \exp\left(-\Omega(\omega_k^{(3-\beta)/2} \cdot \min\{\alpha - 1, (\alpha-1)^2\}) \right) \ .$$

Now we use the tail bound from Lemma 4.5 and overestimate the load distribution of ⟨D1⟩-nodes with an exponential distribution. In particular, for each node $v \in V_k$ we introduce the variable X_v that measures the "⟨D1⟩-load" of this node, i.e. the load that each ⟨D1⟩-node keeps. We can now show that for each node $v \in V_k$ the following random variable stochastically dominates the ⟨D1⟩-load X_v.

Definition 4.6. *For a node $v \in V_k$ let*

$$\widehat{X}_v = \begin{cases} 0 & \text{with prob. } 1 - \widehat{p}_v \\ \ell_v \left(1 + Exp(\lambda_v) + \mathbf{E}\left[\, D_v^h \,\right]^{-2/3}\right) & \text{with prob. } \widehat{p}_v \ , \end{cases}$$

where $\widehat{p}_v = 2\exp\left(-\frac{\mathbf{E}\left[\, D_v^h \,\right]^{1/3}}{4} \right)$ is an upper bound for the probability that v is a ⟨D1⟩-node, $\lambda_v = \frac{1}{4}\mathbf{E}\left[\, D_v^h \,\right]$ and $\ell_v = 2(1+\varepsilon_k)m\frac{w_v}{W_k}$.

Note that our ⟨D1⟩-load overestimates the contribution of v to the total load left on ⟨D1⟩-nodes during the top-down phase. In particular, if v is not a ⟨D1⟩-node, then no ⟨D1⟩-load is left on v and consequently the contribution is 0. Otherwise, we use the tail bound from Lemma 4.5 as follows. We overestimate the load by assuming that at least twice the layer-average load is present on v. For the additional load, we can apply the tail bound under the condition $\alpha \geqslant 2$, which implies that this excess load is upper bounded by an exponentially distributed random variable with a parameter $\lambda_v = \frac{1}{4}\mathbf{E}\left[\, D_v^h \,\right]$.

We first obtain a high probability bound on the total load left on ⟨D1⟩-nodes in each layer k during the top-down phase.

Lemma 4.7. *For every constant $c > 0$ and any k the total load left on ⟨D1⟩-nodes in layer k is at most*

$$4(1+\varepsilon_k)m\frac{\omega_{k-1}}{W_k}c\ln n + 40(1+\varepsilon_k)m\frac{\omega_{k-1}}{W_k}n_k \exp\left(-\frac{1}{4}\left(\omega_k \frac{W_{k-1}}{W} \right)^{1/3} \right)$$

with probability at least $1 - n^{-c}$.

Now we take a closer look at nodes with property ⟨D2⟩. We can employ a Chernoff Bound to show that nodes with polylogarithmically large weights

do not deviate by $w_v^{2/3}$ from their expected degree with high probability. This means that none of these nodes fulfills property $\langle$D2$\rangle$ with high probability. In the following analysis we can therefore concentrate on nodes with weight at most polylog(n). This observation is crucial for the proof of Lemma 4.8.

Lemma 4.8. *For any k all nodes $v \in V_k$ with property $\langle$D2$\rangle$ contribute at most*

$$\mathcal{O}\left(\left(1+\omega_k^{-2/3}\right)^3 \omega_k^{\frac{4-\beta+\varepsilon(\beta-1)}{1-\varepsilon}} \cdot \exp\left(-\omega_k^{1/3}/4\right) m\right) + \mathcal{O}\left(\frac{\text{polylog}(n)}{\sqrt{n}} m\right)$$

to the total dead-end load of all layers with probability at most $1-\frac{3}{n^C}$, for a constant $C > 1+(\beta-2)\left(1+\frac{1}{1-\varepsilon}\right)$.

After successfully bounding the contribution of nodes with properties $\langle$D1$\rangle$ and $\langle$D2$\rangle$ to dead-end load, we will now turn to the border nodes with property $\langle$D3$\rangle$. We already know that these nodes cannot deviate too much from their expected degrees, because they do not fulfill property $\langle$D2$\rangle$ by definition. Therefore they can only be on one of two layers. We still have to differ between nodes in the upper half of a border and those in the lower half. The following lemma bounds the contribution of nodes in the upper half of a border.

Lemma 4.9. *For any k all nodes $v \in V_k$ with property $\langle$D3$\rangle$ and $\omega_k \leqslant w_v \leqslant \omega_k + \omega_{k-1}^{2/3}$ contribute at most*

$$\Theta\left(\omega_k^{-\varepsilon(\beta-2)} + \frac{\omega_k^{\beta-2}\omega_{k+1}^{\beta-2} \cdot c\ln n}{n}\right) m$$

to the total dead-end load of layer $k+1$ w. h. p.

The following lemma about the contribution of nodes in the lower half of a border uses the smoothness of the weight distribution.

Lemma 4.10. *For any k all nodes $v \in V_{k+1}$ with property $\langle$D3$\rangle$ and $\omega_k - \omega_k^{2/3} \leqslant w_v \leqslant \omega_k$ contribute at most*

$$(1+\varepsilon_k)\left(\frac{6\left(d\frac{\beta-2}{\beta-1}\right)^{\beta-1}}{\frac{\gamma}{2}}\omega_k^{-1/3} + \frac{2\cdot\omega_k^{\beta-1}}{\frac{\gamma}{2}n} + \frac{d\cdot\omega_k^{\beta-2}\omega_{k+1}^{\beta-2}\cdot c\ln n}{\left(\frac{\gamma}{2}\right)^2 n}\right) m$$

to the total dead-end load of layer k w. h. p.

At last we have to show that the dead-end load of nodes with property $\langle$D4$\rangle$ is properly bounded. We already know, that each of these nodes obeys the upper bound from Lemma 4.3. Therefore it is sufficient to bound the number of these nodes. To bound the number of these nodes in V_k, we simply have to bound the total number of edges lost between nodes from V_k and nodes with properties $\langle$D2$\rangle$ or $\langle$D3$\rangle$ from V_{k+1}. This idea helps us to proof the following lemma.

Lemma 4.11. *Let* $\varepsilon < \min\left\{\frac{\beta-2}{3}, \frac{1}{2}\left(1-\sqrt{\frac{3}{\beta+1}}\right)\right\}$. *Then the following statements hold:*

(1) For all $k > 0$ *the total load of nodes* $v \in V_k$ *with property* $\langle \mathrm{D4} \rangle$ *is at most*

$$\mathcal{O}\left(\omega_k^{\frac{2-\beta}{3(1-\varepsilon)}} + \omega_k^{\frac{(2\beta^2-11\beta+14)(\beta-2)}{27(1-\varepsilon)}} + n^{\frac{3+1-\varepsilon+2(\beta-2)(1-\varepsilon)^2}{6}-1} + \exp\left(-\omega_k^{\frac{1-\varepsilon}{3}}/4\right)\omega_k^{\frac{1}{1-\varepsilon}+(\beta-2)} + \frac{\mathrm{polylog}(n)}{\sqrt{n}}\right) m \qquad \text{w. h. p.}$$

(2) For $k = 0$ *there are no* $\langle \mathrm{D4} \rangle$*-nodes w. h. p.*

Finally, we bound the total load left on dead-ends during the top-down phase.

Lemma 4.12. *For every constant* c*, there exists a constant* c' *such that if we run the top-down phase on layers with* $\omega_i \geqslant c'$*, then with probability at least* $1-1/n^{-c}$ *we obtain a total load of at most* $m/2$ *on all dead-ends on these layers.*

The last lemma implies that with high probability, for a suitably chosen *key layer* at most half of the load is left on dead-ends during the top-down phase on this and the above layers. In particular, our upper bound on the load of non-dead-end nodes in Lemma 4.3 implies that on this layer, every such node gets at most a load of $(1+\varepsilon_k) \cdot m \cdot w_v/W_k$. On the other hand, a load of $m/2$ passes through this layer w.h.p. In the worst case all non-dead-ends get the maximum load of $(1+\varepsilon_k) \cdot m \cdot w_v/W_k$. This results in at least $n\frac{\gamma}{4(1+\varepsilon_k)}{\omega_k}^{-\frac{\beta-1}{(1-\varepsilon)}}$ nodes which absorb m/n load each, causing a decrease of unassigned load by a constant fraction of at least $\frac{\gamma}{4(1+\varepsilon_k)}{\omega_k}^{-\frac{\beta-1}{(1-\varepsilon)}}$. Here, $\omega_k \geqslant c'$ where c' is as chosen in Lemma 4.12.

5 Analysis of Iterative Absorption

Algorithm 1 sends all unassigned load back to the top, balances it within the top layer, and restarts the top-down distribution step. Observe that all the arguments made for the analysis of the downward propagation can be applied for any value of m. The absorption of load during these iterations is adjusted according to the following scheme. We let each of the nodes absorb at most a load of $m/(n \cdot t^2)$ in round t. This scheme is executed for $t = \log\log n$ rounds and then repeated in chunks of $\log\log n$ rounds until all load is assigned. We will show that with high probability after a constant number of repetitions, all load is assigned. In addition, as $\sum_{t=1}^{\infty} 1/t^2 = \mathit{\Pi}^2/6$, each node receives a load of $(1+\mathcal{O}(1)) \cdot m/n$.

In particular, our aim is to show that using this scheme we need only $\mathcal{O}(\log\log n)$ top-down distribution steps to reduce the total unassigned load in the system to $m' = m/\log^c n$, for any constant c. This is shown in the lemma below. Given this result, we run the protocol long enough such that c becomes a sufficiently large constant. We want to show that, if this is the case, each node

on a layer with polylogarithmic degree gets a load of at most m/n, resulting in all remaining load being absorbed. As each non-dead-end on this layer gets a share of at most $w_v \frac{W_k}{W} m' = \frac{\text{polylog}(n)}{n} m' = m/n$ they fulfill the requirement. The same bound holds for $\langle$D4$\rangle$-nodes by definition. As $\langle$D1$\rangle$- and $\langle$D2$\rangle$-nodes do not appear on layers of at least polylogarithmic degree, we can ignore them as well. All we need to care about now are $\langle$D3$\rangle$-nodes. We can derive upper bounds on their load similar to the ones for non-dead-ends using results on the expected number of edges between these nodes and both their possible next-highest layers. This is a simple corollary from the proof of Theorem 2.2 which we defer to the full version of the paper. It now remains to show the following lemma, the proof of which can be found in the full version of this paper.

Lemma 5.1. *Using the repeated absorption scheme of Algorithm 1, for any constant c, only $\mathcal{O}(\log \log n)$ rounds suffice to reduce the unassigned load in the network to $m/\log^c n$.*

6 Discussion

To the best of our knowledge, we have presented the first double-logarithmic load balancing protocol for a realistic network model. Our algorithm reaches a balanced state in time *less than the diameter* of the graph, which is a common lower bound for other protocols (e.g. [9]). Note that our Theorem 2.1 can be interpreted outside of the intended domain: It reproves (without using the fact) that the giant component is of size $\Theta(n)$ (known from [3]) and that rumor spreading to most vertices can be done in $\mathcal{O}(\log \log n)$ (known from [10]).

Our algorithm works fully distributed, and nodes decide how many tokens should be sent or received based only on their current load (and those of its neighbors). We expect our wave algorithm to perform very robust against node and edge failures as it does *not* require global information on distances [9] or the computation of a balancing flow [7].

Our Theorem 2.2 allows initial load on nodes with degree $\Omega(\text{polylog}\, n)$. Future work includes a further relaxation of this assumption, for instance, by employing results about greedy local-search based algorithms to find high degree nodes [2, 5]. Another interesting direction is to translate our load balancing protocol into an algorithm which samples a random node using the analogy between load and probability distributions. Such sampling algorithms are crucial for crawling large-scale networks such as online social networks like Facebook, where direct sampling is not supported [11].

Bibliography

[1] J. E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Pract. Exper.*, 2:289–313, 1990.

[2] C. Borgs, M. Brautbar, J. T. Chayes, S. Khanna, and B. Lucier. The power of local information in social networks. In *8th Intl. Workshop Internet & Network Economics (WINE)*, pp. 406–419, 2012.

[3] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99: 15879–15882, 2002.

[4] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Phys. Rev. Lett.*, 90:058701, 2003.

[5] C. Cooper, T. Radzik, and Y. Siantos. A fast algorithm to find all high degree vertices in graphs with a power law degree sequence. In *9th Intl. Workshop Algorithms and Models for the Web Graph (WAW)*, pp. 165–178, 2012.

[6] G. Cybenko. Load balancing for distributed memory multiprocessors. *J. Parallel and Distributed Comput.*, 7:279–301, 1989.

[7] R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25:789–812, 1999.

[8] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[9] R. Elsässer and T. Sauerwald. Discrete load balancing is (almost) as easy as continuous load balancing. In *29th Symp. Principles of Distributed Computing (PODC)*, pp. 346–354, 2010.

[10] N. Fountoulakis, K. Panagiotou, and T. Sauerwald. Ultra-fast rumor spreading in social networks. In *23rd Symp. Discrete Algorithms (SODA)*, pp. 1642–1660, 2012.

[11] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *29th IEEE Conf. Computer Communications (INFOCOM)*, pp. 2498–2506, 2010.

[12] F. C. Graham and M. Radcliffe. On the spectra of general random graphs. *Electr. J. Comb.*, 18, 2011.

[13] K. H. Huebner, D. L. Dewhirst, D. E. Smith, and T. G. Byrom. *The Finite Element Methods for Engineers*. Wiley, 2001.

[14] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. AMS, 2008.

[15] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:pp. 167–256, 2003.

[16] R. Subramanian and I. D. Scherson. An analysis of diffusive load-balancing. In *6th Symp. Parallelism in Algorithms and Architectures (SPAA)*, pp. 220–225, 1994.

[17] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica. Load balancing in dynamic structured peer-to-peer systems. *Performance Evaluation*, 63:217–240, 2006.

[18] R. van der Hofstad. Random graphs and complex networks. Available at `www.win.tue.nl/~rhofstad/NotesRGCN.pdf`, 2011.

[19] D. Zhanga, C. Jianga, and S. Li. A fast adaptive load balancing method for parallel particle-based simulations. *Simulation Modelling Practice and Theory*, 17:1032–1042, 2009.