A New Analytical Method for Parallel, Diffusion-type Load Balancing

Petra Berenbrink[1], Tom Friedetzky[2], and Zengjian Hu[3]

[1,3]Simon Fraser University
School of Computing Science
8888 University Drive
Burnaby, BC, V5A 1S6, Canada
{petra,zhu}@cs.sfu.ca

[2]Durham University
Department of Computer Science
Science Labs, South Road
Durham, DH1 3LE, England, U.K.
tom.friedetzky@dur.ac.uk

Abstract

We propose a new proof technique which can be used to analyze many parallel load balancing algorithms. The technique is designed to handle concurrent load balancing actions, which are often the main obstacle in the analysis. We demonstrate the usefulness of the approach by analyzing various natural diffusion-type protocols. Our results are similar to, or better than, previously existing ones, while our proofs are much easier.

The key idea is to first sequentialize the original, concurrent load transfers, analyze this new, sequential system, and then to bound the gap between both.

1 Introduction

In this paper we study the following *neighbourhood load balancing problem.* We have n identical nodes that are connected by a network with maximum degree δ. Nodes are allowed to communicate with each other only if they are connected by an edge. Initially, each node stores some number of tokens (tasks, jobs, ...). The total number of tokens is time-invariant, i.e., neither do new tokens appear, nor do existing ones disappear. The objective is to distribute the tokens as evenly as possible among the nodes whilst minimizing the number of load balancing steps.

The *load* of a node at time t is the number of tokens the node stores at that time. At each time step, nodes compare their current load with the load of a subset of their neighbours, possibly with all of them. If the own load exceeds the load of such a neighbour by a certain amount, then they send a certain number of tokens to that neighbour. Clearly, it will take a "long" time until the system is balanced if the number of tokens sent is "small" compared to the load difference. On the other hand, if this amount is too big, then load might bounce back and forth. To prevent that, the amount of load that a node is allowed to forward to a neighbour is typically upper bounded by a function of the difference d and the maximum degree, δ, e.g., $d/(\delta + 1)$.

A common classification of neighbourhood load balancing schemes is the distinction between *diffusion* and *dimension exchange* methods. In the case of diffusion methods every node balances its load *concurrently* with *all* neighbours. In the case of the dimension exchange method, each node communicates with only one neighbour per time step. For the latter case it remains to specify the method of choosing balancing partners. Here, some well-known approaches are either to randomly generate a matching of the underlying network in every time step ([12]), or to fix the balancing partners in a round robin-like predetermined order ([3]). Another classification is that between *discrete* and *continuous* load balancing methods. In the former, tokens can not be split, and balancing partners are allowed to exchange only integer amounts of load. In the latter, load can be split into arbitrarily small pieces, and balancing partners are allowed to exchange fractional amounts of load.

So far, the analysis methods for *diffusion* and *dimension exchange* methods were quite different. For dimension exchange methods, potential functions [3, 12, 13, 15] are a widely used approach. In that approach, a suitable potential function is chosen that assigns a potential to every possible state of the system. Simplifying matters somewhat, it is then shown that the potential decreases in every time step. However, it

1-4244-0054-6/06/$20.00 ©2006 IEEE

turns out to be more challenging to use this method for diffusion type algorithms (see [13, 15]). This is mainly due to the concurrent load balancing actions which can change the load situation in one step drastically. Diffusion methods are normally analyzed with an algebraic approach (see [3, 18, 15]) that only works for the continuous case. See Section 2 for an overview of this approach.

In this paper we propose a very simple potential function technique to analyze discrete diffusion load balancing schemes. First, we sequentialize the load balancing actions of the diffusion approach in a suitable way, and then we show that the potential decreases after each of these sequential load balancing actions. We use the potential drop of the sequentialized load balancing scheme to lower bound the potential drop of the original system with concurrent load balancing actions. In more detail, we show that under certain conditions the potential drop of both the sequentialized and the concurrent system differ by only a constant factor. Our proof method simply neglects the concurrency in the original load balancing approach and maps the problem to the corresponding sequential load balancing method. To analyze the related sequential load balancing algorithm, we can using existing ideas, e.g., from [12].

We use our technique to analyze the standard diffusion algorithm in the continuous and the discrete case. Then we apply our technique to get results for the dynamic model of [10], where the network can change over the time. Again, we get results for the discrete and the continuous case. Finally, we step aside from the traditional neighbourhood load balancing approach and allow nodes to randomly choose their load balancing partners from among set set of all other nodes. Note that in this setting, a node can easily be forced to balance its load with many other nodes, such that many of concurrent load balancing actions will take place. Note also that this setting can be regarded as neighbourhood load balancing where the network topology is randomly chosen and changes from step to step. We call such a network *random* in the following. To our best knowledge, the results for discrete version of the dynamic network, and the random network are new.

The remainder of the paper is organized as follows. Section 2 introduces related work. We sketch our technique, and compare our results to previous ones in Section 3. Section 4 deals with continuous and discrete load balancing in the fixed network model. Section 5 deals with continuous and discrete load balancing in the dynamic network model, and Section 6 does the same for the random network model. Finally, Section 7 concludes the paper.

2 Related Work

In this section we review some related results. We partition the work into the two categories *continuous* and *discrete* load balancing strategies.

2.1 Continuous Load Balancing

Continuous load balancing is the "ideal" case in which tokens can be split arbitrarily. Hence it is possible to balance the load perfectly.

Diffusion. Cybenko [3] and, independently, Boillat [2], were the first to study the diffusion method. In the diffusion model of Cybenko, the work distribution at time step t is quantified by an n vector, $L^t = (\ell_1^t, \ldots, \ell_n^t)$, where ℓ_i^t is the load of node i at time $t \geq 0$. In each round t, node i and node j compare their load. If $\ell_j^t > \ell_i^t$, node j sends $\alpha_{ij}\left(\ell_j^t - \ell_i^t\right)$ tokens to node i. α is called the *diffusion factor* and is set to $1/(\delta+1)$, where δ is the maximum degree of the network. We can write $L^{t+1} = M \cdot L^t$, where $M = (m_{ij})$ is a matrix defined as

$$m_{ij} = \begin{cases} \alpha_{ij}, & \text{if } i \neq j \\ 1 - \sum_k \alpha_{ik} & \text{if } i = j. \end{cases}$$

M is commonly referred to as *diffusion matrix.* Cybenko [3] (see also [18, 15]) shows a tight connection between the convergence rate of his diffusion algorithm and the second largest eigenvalue of M. Let $\overline{\ell} = \sum_{i=1}^n \ell_i^t/n$ be the average load and let $b = (\overline{\ell}, \ldots, \overline{\ell})$ be the *balanced distribution.* For each $t \geq 0$, define the *error* $e^{(t)}$ to be $\ell^{(t)} - b$. Let $-1 \leq \mu_1 \leq \mu_2 \leq \ldots \leq \mu_n = 1$ be the set of eigenvalues of M and denote $\gamma = \max_{\mu_i \neq 1} |\mu_i|$ to be the second largest eigenvalue of M. Let $||e^{(t)}||_2$ be the ℓ_2 norm of the error vector $e^{(t)}$. It can be shown that $||e^{(t+1)}||_2 = ||M \cdot e^{(t)}||_2 \leq \gamma \cdot ||e^{(t)}||_2$, which implies $||e^{(t)}||_2 \leq \gamma^t \cdot ||e^{(0)}||_2$,

Subramanian and Scherson [18] observe similar relations between convergence time and the properties of the underlying network. They obtain the following bound on the convergence time T: $\Omega(\log(\sigma)/\Gamma) \leq T \leq O(n\sigma/\Gamma)$ and $\Omega(\log(\sigma)/\Lambda) \leq T \leq O(\sigma/\Lambda^2)$. where n is the size of the network, σ is the standard deviation of the initial load distribution. Γ and Λ are the network's electrical and fluid conductance, respectively.

Ghosh et al. [15] refer to the above diffusion model as the *first order scheme* and further generalize it to the so called *second order scheme*, where $L^t = \beta \cdot ML^{t-1} + (1-\beta) \cdot L^{t-2}$, with β a constant. L^t does not only relate to L^{t-1} but also L^{t-2}, hence the name second order. They show that the second order scheme converges much faster than the first order scheme for suitably chosen values of β. Diekmann et al. [7] extend the idea of [15] and propose a general framework to analyze the convergence behaviour of a wide range of diffusion type methods. They introduce the so called *Optimal Polynomial Scheme (OPS)*, which can determine an optimal balancing flow within m steps, where m is the number of distinct eigenvalues of the graph.

In [11] the authors analyze the diffusion algorithm for dynamically changing networks. The results are stated in Theorem 7. The proof method is similar to the one in [7].

2.1.1 Dimension Exchange

In [12], Ghosh and Muthukrishnan study the dimension exchange method for an arbitrary network G. To avoid concurrent load balancing actions they randomly generate a matching M_t in every step t. The nodes of the matching are then allowed to balance their load by exchanging half the load difference between every pair. They use a standard potential function argument. They first show that the probability for an edge to be included in the matching M_t is at least $1/8\delta$. Next, they estimate the expected potential drop by summing over all edges. The show that in each round the expected drop of ϕ is at least $\lambda_2/16\delta$. Here, λ_2 is defined as the second smallest Eigenvalue of the *Laplacian matrix* of G. The Laplacian matrix of G is defined as $L = D - A$, with A denoting the adjacency matrix of G and $D = (d_{ij})$ with $d_{ij} = 0$ if $i \neq j$ and d_{ii} the degree of node i.

2.2 Discrete Load Balancing

Discrete load balancing, in which only integer tokens are allowed to be transferred, is a more realistic model than continuous load balancing. In this case the network can not be completely balanced. To see that consider the line as a network where the load of node i is simply i. The load is certainly not totally balanced but no neighbouring pair of nodes would balance their load. Unfortunately, discrete load balancing can not be analyzed using the algebraic technique of [3].

Quite often, the continuous model is used to bound the convergence time of discrete load balancing. Since the approximation error is mainly caused by rounding, it is not significant when the system is far from the balanced state (see [12, 15]). For the discrete version of their random matching based algorithm, by carefully calculating how much error can be introduced by rounding, Ghosh and Muthukrishnan [12] prove that, as long as $\phi \geq 2\delta n/\lambda_2$, the rounding can at most slow down the convergence time by a factor of two.

Besides, using the same rounding technique as above, Muthukrishnan et al. [15] show that in the case of the discrete version of their first order scheme, the initial potential φ_0 can be reduced to $O(\delta^2 n^2/\epsilon^2)$ in $O\left(\log \varphi_0/(1-(1+\epsilon)\gamma^2)\right)$ steps.

Rabani, Sinclair and Wanka [16] propose a more general technique to study the discrete load balancing. Their idea is to approximate the discrete system by idealized Markov chains. Let M be the diffusion matrix of a diffusion algorithm, and let $\gamma, \mu = 1 - |\gamma|$ be the second largest eigenvalue and the eigenvalue gap of M, respectively. Furthermore, let $K = \max_{i,j} |\ell_i - \ell_j|$ be the discrepancy of the initial load vector ℓ. They show for the idealized Markov chain that $r = \frac{2}{\mu} \ln\left(\frac{Kn^2}{x}\right)$ rounds are sufficient to reduce the discrepancy to x. Next, to quantify the deviation of the actual load and the distribution generated by the Markov chain, they propose to use a natural quantity, the *local divergence* Ψ, which is the sum of load differences of the two systems across all edges of the network, aggregating over time. They obtain the following bound for Ψ: $\Psi(M) = O(\delta \log N/\mu)$. Finally, applying the knowledge of the second largest eigenvalue and of r from above, they get fairly tight convergence results for various network topologies, e.g., line graph, de Bruijn network, degree-d expander etc.

Using a Markov chain based approach, Elsässer and Monien [8] propose a new discrete diffusion scheme which is fully randomized and distributed. Let K be the initial discrepancy (defined as above) and δ be the maximum degree of the underlying graph. They show that, after $O\left(\frac{\delta}{\lambda_2}(\log n \log\log n + \log K)\right)$ steps, the algorithm can reduce the error bound $||e^{(k)}||_2$ to $O(\sqrt{n})$.

3 Our Results

The main contribution of this paper is a new proof technique which can be used to analyze many diffusion-type load balancing algorithms, where the concurrent load balancing actions are the main challenge to the analysis. We demonstrate that our approach can be used to analyze diffusion discrete and continuous load balancing in a variety of underlying network models.

The key idea is to first sequentialize the concurrent actions in a diffusion algorithm, and then check to which extent the concurrency can degrade the algorithm performance. We can show that under certain conditions, the potential drop of both the sequentialized system and the concurrent system differ by a constant factor only. Hence, one can simply "neglect" the concurrency, and the remaining analysis can be easily done using existing techniques like in [12]. To illustrate how the idea works, we first analyze Algorithm 1, a classic diffusion algorithm similar to the ones studied in [3, 18, 15]. Next, we consider Algorithm 2, which allows every user to randomly find its balancing partner. We again analyze it using the same proof idea; this shows that our technique is quite general.

Specifically, Section 4 analyzes a diffusion algorithm (Algorithm 1) with concurrent load balancing actions. For the proof, we use a standard potential function Φ (similar to the ones defined in [3, 12, 15, 18]). We can show that at each step, the potential drop of Algorithm 1 is at least some constant (0.5) times that of the corresponding sequentialized algorithm. In other words, the concurrency can degrade our algorithm performance by at most a factor of two. Finally, we adopt the proof idea in [12] to analyze the sequentialized algorithm so as to obtain the main convergence result (Theorem 4) for Algorithm 1.

Note that most existing results for diffusion-type algorithms consider the corresponding diffusion matrix of the network (see [2, 3, 18, 15]), while our result is expressed in terms of network parameters (e.g., the second-smallest eigenvalue of the Laplacian matrix, the maximum degree). Moreover, our approach is much simpler. Furthermore, due to the concurrent load balancing actions, our algorithm converges a constant times faster than the dimension exchange algorithm in [12].

Next, we analyze the discrete version of Algorithm 1 and obtain similar results to the ones in [12, 15]. We prove that as long as the potential is larger than a certain threshold (i.e., the system is "far" from the well-balanced state), the discrete case has similar convergence behaviour to the continuous case. For the same discrete diffusion algorithm, our result (Theorem 6) is stronger than the one in [15], as it only requires the potential to be larger than a term linear in n instead of quadratic. Furthermore, compared to the discrete dimension exchange algorithm in [12], our algorithm is still a constant times faster.

In Section 5 we use our proof method to get similar results similar to the ones in [11] for a dynamic network model where the active edges can change from round to round. In contrast to [11], we get also results for the discrete load balancing model.

In Section 6, we analyze Algorithm 2, which allows nodes to randomly to choose balancing partners. Note that Algorithm 2 also contains concurrent load balancing actions since a node may have been chosen by many other nodes. Using the same proof idea to handle the concurrency, one can show that Algorithm 2 also converges quickly, as in each round the system potential drops by at least a constant factor in expectation. This implies that Algorithm 2 has a strict logarithmic convergence time which does not rely on any network parameters. Note that our results for this model are stronger that the ones that we would get by simply applying our results for the dynamic model. To our best knowledge this is the first time that the diffusion scheme is analyzed in a model where nodes are allowed to randomly to choose balancing partners.

4 Diffusion Algorithm on Fixed Networks

In this section we present our results in the standard diffusion model for arbitrary networks. The next section deals with the *continuous case*, where tokens can be arbitrarily split. In section 4.2 we show how to use our technique to obtain results for the discrete case.

4.1 Continuous Case

First we need some more notation. $G = (V, E)$ is the underlying network. Let $\{e_1, e_2, \dots e_{|E|}\}$ be the set of edges of G. For each node $i \in V$, let d_i be the degree of i, and let $\delta = \max_{i \in V} d_i$. $\alpha = \min\limits_{S \subset V} \frac{|E(S,\overline{S})|}{\min(|S|,|\overline{S}|)}$ is the *edge expansion* of G, with $\overline{S} = V/S$, and $E(S, \overline{S})$ the set of edges with one endpoint in S and the other endpoint in $\overline{S}$. Furthermore, let $N(i) = \{j \in V | (i, j) \in E\}$ denote the set of all neighbours of node i. Let ℓ_i^t be the load of node i at the end of round t. Whenever clear from the context we will simply write ℓ_i in the following. Then the vector $L = \{\ell_1, \dots, \ell_n\}$ represents the entire load distribution. Now we are ready to define the load balancing algorithm we are considering in this section.

Algorithm 1 *diff-balancing(G)*
```
1: for every node $i \in V$ in parallel do
2:     for any $j \in N(i)$ do
3:         if $\ell_i > \ell_j$ then
4:             send $\frac{\ell_i - \ell_j}{4\max(d_i, d_j)}$ load from node $i$ to $j$
5:         end if
6:     end for
7: end for
```

Similar to the result in [12], Theorem 4 (presented below) is a function of the edge expansion value and the maximum degree of G. Let $0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of the Laplacian matrix of G (for the definition of Laplacian matrix, see Section 2.1.1). Let $L^t = \{\ell_1^t, \dots, \ell_n^t\}, t \geq 0$ be the load vector after t balancing steps and $\overline{\ell} = \sum_{i=1}^n \ell_i/n$ the average load.

In the following we will assume that all load vectors are *normalized*, i.e., $\ell_1^t \leq \ell_2^t \leq \ldots \leq \ell_n^t$. To analyze the algorithm, we will use the following potential function $\Phi(L^t) = \sum_{i=1}^n (\ell_i^t - \overline{\ell})^2$. Hence, $\Phi(L^{t-1}) - \Phi(L^t)$ is the potential drop in round t.

We assign weight $w_{ij} = \left|\ell_i^{t-1} - \ell_j^{t-1}\right| / 4\max(d_i, d_j)$ to each edge $e = (i,j)$ in every round. The weight w_{ij} is the load that will be transferred over $e = (i,j)$ in round t. Let $E^t = \{e_1^t, e_2^t, \ldots e_{|E|}^t\}$ be the set of edges sorted in increasing order of their weights. For the sake of the analysis, we now assume the edges are activated one by one starting with the edge e_1^t with the smallest weight. Then we can define $L^{(t,k)} = \left(\ell_1^{(t,k)}, \ldots, \ell_n^{(t,k)}\right)$ to be the load vector right after the activation of the first k edges $e_1^t, \ldots e_k^t$ in round t (applied to the load distribution L^{t-1}). $\Delta\Phi_\ell^t$ is the potential drop due to the activation of edge e_ℓ in round t. The next lemma lower bounds the potential drop due to a single edge activation.

Lemma 1 *Fix a round t. For all edges $e_\ell = (i,j) \in E$ we have $\Delta\Phi_\ell^t \geq w_{ij}\left|\ell_i^{t-1} - \ell_j^{t-1}\right|$.*

Proof: Assume $\ell_i^t \geq \ell_j^t$. Since all edges are activated in increasing order of their weights, the amount of load that node i can send to any other neighbour in round t before the activation of e_k, is at most $w_{ij} = \left|\ell_i^{t-1} - \ell_j^{t-1}\right| / 4\max(d_i, d_j)$. Node i has at most $d_i - 1$ additional neighbours, hence it can send at most $(d_i - 1) \cdot \frac{\left|\ell_i^{t-1} - \ell_j^{t-1}\right|}{4\max(d_i,d_j)}$ load to other neighbours before the activation of edge (i,j). Consequently,

$$\begin{aligned}
\ell_i^{(t,k-1)} &\geq \ell_i^{t-1} - (d_i - 1)\cdot w_{ij} \\
&= \ell_i^{t-1} - d_i \cdot \left(\frac{\left|\ell_i^{t-1} - \ell_j^{t-1}\right|}{4\max(d_i,d_j)}\right) + w_{ij} \\
&\geq \ell_i^{t-1} - \frac{1}{4}\left|\ell_i^{t-1} - \ell_j^{t-1}\right| + w_{ij}. \qquad (1)
\end{aligned}$$

Similarly, node j receives at most $(d_j - 1) \cdot \frac{\left|\ell_i^{t-1} - \ell_j^{t-1}\right|}{4\max(d_i,d_j)}$ before the activation of edge (i,j). Hence,

$$\begin{aligned}
\ell_j^{(t,k-1)} &\leq \ell_j^{t-1} + (d_j - 1)\cdot w_{ij} \\
&= \ell_j^{t-1} + d_j \cdot \left(\frac{\left|\ell_i^{t-1} - \ell_j^{t-1}\right|}{4\max(d_i,d_j)}\right) - w_{ij} \\
&\leq \ell_j^{t-1} + \frac{1}{4}\left|\ell_i^{t-1} - \ell_j^{t-1}\right| - w_{ij}. \qquad (2)
\end{aligned}$$

Consequently,

$$\begin{aligned}
\Delta\Phi_\ell^t &= \left(\ell_i^{(t,k-1)} - \overline{\ell}\right)^2 + \left(\ell_j^{(t,k-1)} - \overline{\ell}\right)^2 - \\
&\quad \left(\ell_i^{(t,k)} - \overline{\ell}\right)^2 - \left(\ell_j^{(t,k)} - \overline{\ell}\right)^2 \\
&\overset{(a)}{=} \left(\ell_i^{(t,k-1)}\right)^2 + \left(\ell_j^{(t,k-1)}\right)^2 - \\
&\quad \left(\ell_i^{(t,k)}\right)^2 - \left(\ell_j^{(t,k)}\right)^2 \\
&= \left(\ell_i^{(t,k-1)}\right)^2 + \left(\ell_j^{(t,k-1)}\right)^2 - \\
&\quad \left(\ell_i^{(t,k-1)} - w_{ij}\right)^2 - \left(\ell_j^{(t,k-1)} + w_{ij}\right)^2 \\
&= 2w_{ij}\left(\ell_i^{(t,k-1)} - \ell_j^{(t,k-1)} - w_{ij}\right) \\
&\overset{(b)}{\geq} 2w_{ij}\left(\frac{\left|\ell_i^{t-1} - \ell_j^{t-1}\right|}{2} + w_{ij}\right) \\
&\geq w_{ij}\left|\ell_i^{t-1} - \ell_j^{t-1}\right|.
\end{aligned}$$

Here (a) holds since $\ell_i^{(t,k-1)} + \ell_j^{(t,k-1)} = \ell_i^{(t,k)} + \ell_j^{(t,k)}$. (b) is due to Inequalities 1 and 2. □

Now it is straightforward to lower bound the potential decrease in a whole round.

Lemma 2

$$\Phi(L^{t-1}) - \Phi(L^t) \geq \frac{1}{4\delta}\sum_{(i,j)\in E}\left(\ell_i^{t-1} - \ell_j^{t-1}\right)^2.$$

Proof:

$$\begin{aligned}
&\Phi(L^{t-1}) - \Phi(L^t) \\
&\quad = \sum_{e_\ell=(i,j)\in E} \Delta\Phi_\ell^t \\
&\quad \overset{(a)}{\geq} \sum_{(i,j)\in E} w_{ij}\left|\ell_i^{t-1} - \ell_j^{t-1}\right| \\
&\quad = \sum_{(i,j)\in E}\left(\frac{\left|\ell_i^{t-1} - \ell_j^{t-1}\right|}{4\max\{d_i,d_j\}}\cdot\left|\ell_i^{t-1} - \ell_j^{t-1}\right|\right) \\
&\quad \geq \frac{1}{4\delta}\sum_{(i,j)\in E}\left(\ell_i^{t-1} - \ell_j^{t-1}\right)^2.
\end{aligned}$$

Here (a) is due to Lemma 1. □

We shall use the following lemma.

Lemma 3 *(From [12].)*

$$\lambda_2 = \min_x\left(\frac{x^T\mathcal{L}x}{x^Tx} \quad | \quad x \perp v_1, x \neq 0\right).$$

where $v_1 = (1,1,\ldots,1)^T$ and $x \perp v_1$ means that x is orthogonal to v_1.

Proof: Application of the the Courant-Fischer Minimax Theorem, see [12] for the full proof. □

It is now easy to derive the following theorem.

Theorem 4 *For any $\epsilon > 0$, after $T = \frac{4\delta \ln(1/\epsilon)}{\lambda_2}$ steps, we have $\Phi(L^T) \le \epsilon \cdot \Phi(L)$.*

Proof: Fix a round t. First we lower bound $\frac{\Phi(L^{t-1})-\Phi(L^t)}{\Phi(L^t)}$. The idea is similar to [12]. Define x to be a vector of length n with $x_i = \ell_i^{t-1} - \overline{\ell}$. Note that $\sum_{i=1}^n x_i = 0$, and that x is orthogonal to $v_1 = (1,1,\dots,1)^T$. Hence,

$$\begin{aligned}
&\frac{\Phi\left(L^{t-1}\right)-\Phi\left(L^{t}\right)}{\Phi\left(L^{t-1}\right)} \\
&\overset{(a)}{\ge} \frac{\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2}{4\delta\cdot\sum_{i=1}^n x_i^2} \\
&= \frac{\sum_{(i,j)\in E}\left(x_i-x_j\right)^2}{4\delta\cdot\sum_{i=1}^n x_i^2} \\
&= \frac{1}{4\delta}\left(\frac{x^T\mathcal{L}x}{x^Tx}\mid\sum_{i=1}^n x_i=0, x\neq 0\right) \\
&\ge \frac{1}{4\delta}\min_x\left(\frac{x^T\mathcal{L}x}{x^Tx}\mid x\perp v_1, x\neq 0\right) \\
&\overset{(b)}{=} \frac{\lambda_2}{4\delta}. \qquad (3)
\end{aligned}$$

Here, (a) holds by Lemma 2. (b) is due to Lemma 3. Hence, the potential drops by a constant factor in every round and we obtain

$$\begin{aligned}
\Phi(L^T) &\le \left(1-\frac{\lambda_2}{4\delta}\right)^T\Phi(L^0) \\
&= \left(\left(1-\frac{\lambda_2}{4\delta}\right)^{\frac{4\delta}{\lambda_2}}\right)^{\ln(1/\epsilon)}\cdot\Phi(L) \\
&< \left(\frac{1}{e}\right)^{\ln(1/\epsilon)}\Phi(L) = \epsilon\cdot\Phi(L),
\end{aligned}$$

where the second inequality is due to $\forall 0 < x < 1$, $(1-x)^{1/x} < 1/e$. □

4.2 Discrete Case

In this section we analyze the discrete version of Algorithm 1 under the assumption that only integral amounts of tokens can be transferred. This means that for each edge (i,j), we transfer $\left\lfloor\frac{|\ell_i-\ell_j|}{4\max(d_i,d_j)}\right\rfloor$ tokens. Theorem 6 upper bounds the balancing time for the discrete process. Note that it is no longer possible to balance the load completely. (See the example in the introduction.) Compared to the continuous version of the protocol, it takes longer for the discrete protocol converge against a "nearly balanced state", but the difference is only a multiplicative constant.

Lemma 5 *Fix a round t. If $\Phi\left(L^{t-1}\right) \ge 64\delta^3 n/\lambda_2$, $\frac{\Phi(L^{t-1})-\Phi(L^t)}{\Phi(L^{t-1})} \ge \lambda_2/8\delta$.*

Proof:

$$\begin{aligned}
&\frac{\Phi\left(L^{t-1}\right)-\Phi\left(L^{t}\right)}{\Phi\left(L^{t-1}\right)} \\
&\overset{(a)}{\ge} \sum_{(i,j)\in E}\left\lfloor\frac{\left|\ell_i^{t-1}-\ell_j^{t-1}\right|}{4\max(d_i,d_j)}\right\rfloor\cdot\left|\ell_i^{t-1}-\ell_j^{t-1}\right| \\
&\ge \sum_{(i,j)\in E}\left(\frac{\left|\ell_i^{t-1}-\ell_j^{t-1}\right|}{4\max(d_i,d_j)}-1\right)\cdot\left|\ell_I^{t-1}-\ell_j^{t-1}\right| \\
&\ge \frac{1}{4\delta}\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2-\sum_{(i,j)\in E}\left|\ell_i^{t-1}-\ell_j^{t-1}\right| \\
&\overset{(b)}{\ge} \frac{1}{4\delta}\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2- \\
&\qquad\sqrt{|E|\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2} \\
&\overset{(c)}{\ge} \frac{1}{8\delta}\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2.
\end{aligned}$$

Here (a) is due to Lemma 2. (b) follows from $\sum_{i=1}^m a_i \le \sqrt{m\sum_i a_i^2}$. For (c), by Lemma 3, we get

$$\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2 \ge \lambda_2\Phi\left(L^{t-1}\right) \ge 64n\delta^3.$$

Moreover, since $|E| \le n\delta$, we have

$$\begin{aligned}
&\sqrt{|E|\cdot\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2} \\
&\qquad\le \frac{1}{8\delta}\sum_{(i,j)\in E}\left(\ell_i^{t-1}-\ell_j^{t-1}\right)^2.
\end{aligned}$$

□

Remark. Lemma 5 is slightly stronger than Theorem 4 of [15], in that we only require the potential to be linear in n, while Theorem 4 of [15] requires the potential to be at least quadratic in n.

Theorem 6 *After $T = \frac{8\delta \ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)}{\lambda_2}$ steps, $\Phi\left(L^T\right) < 64\delta^3 n/\lambda_2$.*

Proof: By Lemma 5, the potential drops by a constant factor in every round as long as $\Phi(L^T) \geq 64\delta^3 n/\lambda_2$. Hence, After $T = \frac{8\delta \ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)}{\lambda_2}$ steps,

$$\begin{aligned}
\Phi(L^T) &\leq \left(1 - \frac{\lambda_2}{8\delta}\right)^T \Phi(L^0) \\
&= \left(\left(1 - \frac{\lambda_2}{8\delta}\right)^{\frac{8\delta}{\lambda_2}}\right)^{\ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)} \cdot \Phi(L) \\
&\leq \left(\frac{1}{e}\right)^{\ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)} \cdot \Phi(L) \\
&= 64\delta^3 n/\lambda_2,
\end{aligned}$$

Here, the second inequality is due to $\forall 0 < x < 1$, $(1-x)^{1/x} < 1/e$. □

5 Diffusion on Dynamic Networks

In [10], Elsässer et al. considered the diffusion process on *dynamic* networks, in which the set of nodes in fixed, but the set of of communication edges may vary from round to round. They assume that every node knows the edges that are active in a certain time step. The network can now be described by a sequence of "standard" graphs $(G_k)_{k\geq 0}$, where G_k is the underlying network at time step k. In this section we show how to use our analysis approach to get results for their network model. Similar to Section 4, we differentiate the *continuous* and the *discrete* cases. The proofs can be found in the appendix. Note that [10] only considers to continuous case.

5.1 Continuous Case

For the continuous case, Elsässer et al. proved the following theorem. We can show exactly the same result for Algorithm 1 with our proof method. In fact, Theorem 7 can be easily derived by Theorem 4.

Theorem 7 *(From [10]). Denote $\lambda_2^{(k)}$ and $\delta^{(k)}$ to be the second smallest eigenvalue and the maximum degree of G_k respectively. Let $A_K = \frac{\sum_{k=1}^{K}\left(\lambda_2^{(k)}/\delta^{(k)}\right)}{K}$ be the average value of $\lambda_2^{(k)}/\delta^{(k)}$ occurring during the first K iterations. Algorithm 1 needs at most K steps to reduce the system potential from $\Phi(L)$ to $\epsilon\Phi(L)$, where $K = O(\ln(1/\epsilon)/A_K)$.*

5.2 Discrete Case

For the discrete case, we combine Theorem 7 and Lemma 5 and obtain the following theorem for the discrete version of Algorithm 1.

Theorem 8 *Let $\lambda_2^{(k)}, \delta^{(k)}, A_K$ be defined as above. The discrete counterpart of Algorithm 1 needs at most K steps to reduce the system potential to*

$$\begin{aligned}
\Phi^* &= 64n \cdot \max_{k=1}^{K} \left\{ \left(\delta^{(k)}\right)^3 / \lambda_2^{(k)} \right\} \\
K &= O\left(\frac{\ln\left(\Phi(L)/\Phi^*\right)}{A_K}\right).
\end{aligned}$$

Similar to Lemma 5, one can show that whenever the potential is larger than some threshold Φ^* defined above, the potential drops at least by a factor of $\frac{\lambda_2^{(k)}}{8\delta^{(k)}}$ in iteration k. The rest part of the proof is similar to that of Theorem 7, thus the detail is omitted.

6 Randomly picking balancing partners

In this section, we consider an alternative load balancing method (Algorithm 2) which allows nodes to randomly choose their balancing partners. The algorithm proceeds in the following fashion: in each round, first every node randomly picks a balancing partner; later, load is transferred concurrently between the corresponding balancing partners. Note that unlike Algorithm 1, Algorithm 2 does not specify the underlying network topology. Using our analyzing technique to handle the concurrency, we can show that in each round, the system potential drops by at least a constant factor. This implies that Algorithm 2 has a strict logarithmic convergence time. Again, we first show results for the continuous case, and then for the discrete case.

6.1 Continuous Case

We denote by E the set of links whose endpoints are balancing partners, i.e., if node i chooses node j as balancing partner, we create a link (i,j) and add it to E. Moreover, let ℓ_i, $d(i)$ be the load and the number of balancing partners of node i. Our algorithm is as follows:

Below we analyze Algorithm 2. First note that by the classic result of balls into bins games (see, for example, [1]), there is at least one vertex having $\Theta\left(\frac{\log n}{\log\log n}\right)$ balancing partners, with high probability. Consequently, one can not simply use the result in Section 4, which is in terms of the maximum degree of the

Algorithm 2 Randomly picking balancing partners

```
1:  $E = \emptyset$
2:  **for** every node $i \in V$ do in parallel **do**
3:      pick $j \in V$ uniformly at random
4:      $E \leftarrow E \cup (i, j)$
5:  **end for**
6:  **for** every node $i \in V$ do in parallel **do**
7:      **for** every $j$ such that $(i, j) \in E$ **do**
8:          **if** $\ell_i > \ell_j$ **then**
9:              send $\frac{\ell_i - \ell_j}{4\max(d_i, d_j)}$ tokens from node $i$ to $j$
10:         **end if**
11:     **end for**
12: **end for**
```

underlying network. Instead, we prove the following result, which indicates that for a given link, it is unlikely for both sides of the link to have more than a constant number of balancing partners.

Lemma 9 *For a fixed link $(i, j) \in E$,*

$$\Pr[\max(d_i, d_j) \le 5 \,|\, (i, j) \in E] > 0.5.$$

Proof: By symmetry, we can assume that link (i, j) is built by node i. In this case, among the remaining $n-1$ nodes, there must be $d_i - 1$ nodes which choose i as their balancing partner. Since the probability for every node to choose i is $1/n$, we have $d_i \sim 1 + B(n-1, 1/n)$, where $B(n, p)$ is the binomial distribution. Next we consider node j. Note that node j has already connected to two links: (i, j) and another one that node j builds. Hence $d_j \sim 2 + B(n-2, 1/n)$ by similar reason as above. Next, we calculate

$$\begin{aligned}
&\Pr[d_i > 5 \,|\, (i, j) \in E] \\
&= \Pr[B(n-1, 1/n) \ge 5] \\
&\le \binom{n-1}{5} \cdot \left(\frac{1}{n}\right)^5 < \left(\frac{ne}{5}\right)^5 \left(\frac{1}{n}\right)^5 \\
&= \left(\frac{e}{5}\right)^5 < 0.05.
\end{aligned}$$

Similarly, we can prove that $\Pr[d_j > 5 \,|\, (i, j) \in E] < \left(\frac{e}{4}\right)^4 < 0.25$. Using $\Pr[A \text{ or } B] \le \Pr[A] + \Pr[B]$, we can show that

$$\begin{aligned}
&\Pr[\max(d_i, d_j) \le 5 \,|\, (i, j) \in E] \\
&= 1 - \Pr[d_i > 5 \text{ or } d_j > 5 \,|\, (i, j) \in E] \\
&> 1 - (\Pr[d_i > 5 \,|\, (i, j) \in E] + \\
&\quad \Pr[d_j > 5 \,|\, (i, j) \in E]) \\
&> 1 - (0.05 + 0.25) > 0.5.
\end{aligned}$$

□

Before we prove Lemma 11, we show the following result which shows that the potential drop at some step t is a constant times the current system potential.

Lemma 10 $\sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i^t - \ell_j^t)^2 = 2n \cdot \Phi(L^t)$.

Proof: Let $y_i = |\ell_i^t - \overline{\ell}|$, and denote A (or B) to be the set of indices i for which $\ell_i^t \le \overline{\ell}$ (or $\ell_i^t > \overline{\ell}$ resp.). First observe that

$$\sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 = \sum_{i \in B} \sum_{j \in A} (y_i + y_j)^2, \qquad (4)$$

and

$$\begin{aligned}
&\sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 \\
&= \sum_{i \in A} \sum_{j \in B} (y_i^2 + y_j^2 + 2 y_i y_j) \\
&= |B| \cdot \sum_{i \in A} y_i^2 + |A| \cdot \sum_{j \in B} y_j^2 + \\
&\quad 2 \cdot \sum_{i \in A} y_i \cdot \sum_{j \in B} y_j. \qquad (5)
\end{aligned}$$

Similar to (5), we get

$$\sum_{i \in A} \sum_{j \in A} (y_i - y_j)^2 = 2 \cdot |A| \cdot \sum_{i \in A} y_i^2 - 2 \cdot \left(\sum_{i \in A} y_i\right)^2 \qquad (6)$$

and

$$\sum_{i \in B} \sum_{j \in B} (y_i - y_j)^2 = 2|B| \sum_{j \in B} y_j^2 - 2 \left(\sum_{j \in B} y_j\right)^2. \qquad (7)$$

By Equations 4, 5, 6 and 7,

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i^t - \ell_j^t)^2 \\
&= \sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 + \sum_{i \in B} \sum_{j \in A} (y_i + y_j)^2 + \\
&\quad \sum_{i \in A} \sum_{j \in A} (y_i - y_j)^2 + \sum_{i \in B} \sum_{j \in B} (y_i - y_j)^2 \\
&= 2(|A| + |B|) \cdot \sum_{i \in A} y_i^2 + 2(|A| + |B|) \cdot \sum_{j \in B} y_j^2 + \\
&\quad 4 \cdot \left(\sum_{i \in A} y_i\right) \cdot \left(\sum_{i \in B} y_j\right) - \\
&\quad 2\left(\sum_{i \in A} y_i\right)^2 - 2\left(\sum_{j \in B} y_j\right)^2
\end{aligned}$$

$$\overset{(a)}{=} \quad 2\left(|A|+|B|\right)\cdot\left(\sum_{i\in A} y_i^2 + \sum_{j\in B} y_j^2\right)$$
$$\overset{(b)}{=} \quad 2n\cdot\Phi(L^t).$$

Here (a) holds since $\sum\limits_{i\in A} y_i = \sum\limits_{j\in B} y_j$, (b) is because $|A|+|B|=n$ and $\Phi(L^t)=\sum\limits_{i\in A} y_i^2 + \sum\limits_{j\in B} y_j^2$. □

Now we are ready to prove the following lemma.

Lemma 11 $E[\Phi(L^{t+1}\,|\,L^t=L)] \le \frac{19}{20}\Phi(L)$.

Proof:

$$\begin{aligned}
&E[\Phi(L^{t+1})|L^t=L]\\
&= \Phi(L) - \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\Pr[e_\ell=(i,j)\in E]\cdot\Delta\Phi_\ell(L)\right)\\
&\overset{(b)}{\le} \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}|\ell_i-\ell_j|\\
&= \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{(\ell_i-\ell_j)^2}{4\max(d_i,d_j)}\\
&\le \Phi(L) -\\
&\quad \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\Big[\Pr[\max(d_i,d_j)\le 5|(i,j)\in E]\times\\
&\quad\quad \frac{(\ell_i-\ell_j)^2}{4\cdot 5}\Big]\\
&\overset{(c)}{=} \Phi(L) - \frac{1}{40n}\sum_{i=1}^{n}\sum_{j=1}^{n}(\ell_i-\ell_j)^2\\
&\overset{(d)}{=} \Phi(L) - \frac{\Phi(L)}{20} = \frac{19}{20}\Phi(L).
\end{aligned}$$

Here (b) is from Lemma 1. (c) is due to Lemma 9. (d) holds by Lemma 10. □

Finally, we prove the following convergence theorem.

Theorem 12 *For $c>0$, after $T\ge 120c\ln\Phi(L)$ rounds,* $\Pr[\Phi(L^T)\le e^{-c}]\ge 1-\Phi(L)^{-c/4}$.

Proof: For any $t>0$, by linearity of expectation, we can repeatedly use Lemma 11, and obtain

$$E\left[\Phi\left(L^{t+30}\right)\right] \le \left(\frac{19}{20}\right)^{30}\Phi(L^t) \approx 0.21\Phi(L^t).$$

By Markov's inequality, $\Pr[\Phi(L^{t+30}) < \Phi(L^t)/2] \ge 1/2$. Denote a stage to be 30 rounds. For any $c>0$, let $k=4\log\Phi(L)$. For stage $0\le i\le k$, define the random variable

$$X_i = \begin{cases} 1 & \text{if } \Phi(L^{30(i+1)}) \le \Phi(L^{30i})/2.\\ 0 & \text{otherwise.}\end{cases}$$

If $X_i=1$, we say stage i is successful. We use Chernoff's inequality to bound the number of successful stages. Let $X=\sum_{i=0}^{k} X_i$. Clearly $E[X]\ge k/2$. By Chernoff,

$$\begin{aligned}
&\Pr[X\le c\ln\Phi(L)]\\
&\quad\le\ e^{-E[X]0.5^2/2} \le e^{-k/16} \le \Phi(L)^{-c/4}.
\end{aligned}$$

Hence, after $T=30c\cdot k=120c\cdot\ln\Phi(L)$ rounds, the number of successful stages is bigger than or equal to $c\ln\Phi(L)$ with probability at least $1-\Phi(L)^{-c/4}$. Consequently, for $T\ge 120c\cdot\ln\Phi(L)$, $\Pr[\Phi(L^T)\le e^{-c}]$ with probability at least $1-\Phi(L)^{-c/4}$. □

Remark. The random network model of this section can be viewed as a special case of the dynamic network model in Section 5. For random networks we are able to show that the potential drops by a constant factor in each round. Theorem 7 does not give a constant factor drop for our random networks.

6.2 Discrete Case

For the discrete case we use Algorithm 2 with one change. In every step, whenever $\ell_i > \ell_j$, we transfer $\left\lfloor\frac{\ell_i-\ell_j}{4\max(d_i,d_j)}\right\rfloor$ tokens from ℓ_i to ℓ_j. We show the following result indicating that whenever the potential $\Phi(L)$ is bigger than a threshold of $3200n$, the potential drops at least by a constant factor of $\frac{1}{40}$ in every iteration. The proofs are omitted due to space constraints.

Lemma 13 *If $\Phi(L)\ge 3200n$, $E[\Phi(L^{t+1}\,|\,L^t=L)] \le \frac{39}{40}\Phi(L)$.*

Finally, similar to Theorem 12, we directly obtain the following theorem:

Theorem 14 $\forall c>0$, *after* $T\ge 240c\ln\left(\frac{\Phi(L)}{3200n}\right)$ *rounds,* $\Pr[\Phi(L^T)\le 3200n]\ge 1-\left(\frac{\Phi(L)}{3200n}\right)^{-c/4}$.

7 Conclusion

In this paper we propose a new proof technique which can be used to analyze many parallel diffusive load balancing algorithms. The technique first sequentializes a diffusion algorithm with concurrent load balancing actions, and then shows that the concurrency only degrade the system performance by a constant factor. We demonstrate the strength of the technique by analyzing diffusion continuous and discrete load balancing algorithms for several network models.

References

[1] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, Eli Upfal: *Balanced Allocations. SIAM J. Comput.* 29(1): 180-200 1999.

[2] Jacques E. Boillat: *Load Balancing and Poisson Equation in a Graph.* In Journal of *Concurrency - Practice and Experience* 2(4): 289-314 1990.

[3] George Cybenko: *Dynamic Load Balancing for Distributed Memory Multiprocessors.* In Journal of *Parallel Distributed Computing* 7(2): 279-301 1989.

[4] Fan Chung:*Spectral Graph Theory. American Mathematical Society.* 1997.

[5] Ana Cortés, Ana Ripoll, F. Cedo, Miquel A. Senar, Emilio Luque: *An asynchronous and iterative load balancing algorithm for discrete load model.* In Journal of *Parallel and Distributed Computing.* 62(12): 1729-1746 2002.

[6] Ralf Diekmann. *Load Balancing Strategies for Data Parallel Applications.* Ph.D Thesis, University of Paderborn, Germany, 1998.

[7] Ralf Diekmann, Andreas Frommer, *Burkhard Monien: Efficient schemes for nearest neighbour load balancing.* In Journal of *Parallel Computing* 25(7): 789-812 1999.

[8] Robert Elsässer, Burkhard Monien: *Load balancing of unit size tokens and expansion properties of graphs.* In Proceedings of the *15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)* 2003: 266 - 273

[9] Robert Elsässer, Burkhard Monien, Robert Preis: *Diffusion Schemes for Load Balancing on Heterogeneous Networks.* In Journal of *Theory Comput. Syst.* 35(3): 305-320, 2002.

[10] Robert Elsässer, Burkhard Monien, Stefan Schamberger: *Load Balancing on Dynamic Networks. the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04)*: 193-200.

[11] Robert Elsässer, Burkhard Monien, Stefan Schamberger: *Load Balancing of Indivisible Unit Size Tokens in Dynamic and Heterogeneous Networks.* ESA 2004: 640-651.

[12] Bhaskar Ghosh, S. Muthukrishnan: *Dynamic Load Balancing in Parallel and Distributed Networks by Random Matchings.* In Proceedings of the *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)* 1994: 220-225.

[13] Bhaskar Ghosh, Frank Thomson Leighton, Bruce M. Maggs, S. Muthukrishnan, C. Greg Plaxton, Rajmohan Rajaraman, Andrea W. Richa, Robert Endre Tarjan, David Zuckerman: *Tight analyzes of Two Local Load Balancing Algorithms.* In *SIAM J. Comput.* 29(1): 29-64 (1999)

[14] Friedhelm Meyer auf der Heide, Brigitte Oesterdiekhoff, Rolf Wanka: Strongly Adaptive Token Distribution. In Journal of *Algorithmica* 15(5): 413-427, 1996.

[15] S. Muthukrishnan, Bhaskar Ghosh, Martin H. Schultz: *First- and Second-Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing.* In *Theory Comput. Syst.* 31(4): 331-354, 1998.

[16] Yuval Rabani, Alistair Sinclair, Rolf Wanka: *Local Divergence of Markov Chains and the Analysis of Iterative Load Balancing Schemes.* In Proceeding of *39th Annual Symposium on Foundations of Computer Science(FOCS)* 1998: 694-705.

[17] David Peleg, Eli Upfal: *The Token Distribution Problem. SIAM J. Comput.* 18(2): 229-243 1989.

[18] Raghu Subramanian, Isaac D. Scherson: *An Analysis of Diffusive Load-Balancing.* In Proceedings of the *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)* 1994: 220-225.