A new analytical method for parallel, diffusion-type load balancing

Petra Berenbrink\textsuperscript{a}, Tom Friedetzky\textsuperscript{b,*,} Zengjian Hu\textsuperscript{c}

\textsuperscript{a} School of Computing Science, Simon Fraser University, Burnaby, B.C., V5A 1S6, Canada
\textsuperscript{b} Department of Computer Science, Durham University, Durham, DH1 3LE, UK
\textsuperscript{c} D-Wave Systems, Burnaby, B.C., V5C 6G9, Canada

\textbf{A B S T R A C T}

We propose a new proof technique which can be used to analyse 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 analysing 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 sequentialise the original, concurrent load transfers, analyse this new, sequential system, and then to bound the gap between both.

\section*{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 \(\delta\). 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, \(\delta\), e.g., \(d/(\delta + 1)\). Note that, since every node is only allowed to communicate with its direct neighbours, our approach is distributed and uses local knowledge only.

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 \([10]\), or to fix the balancing partners in a round robin-like predetermined order \([4]\). 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 \([4,10–12]\) 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 turns out to be more challenging to use this method for diffusion type algorithms (see \([11,12]\)). This is mainly due to the concurrent load balancing actions which can change the load situation in one step drastically. Diffusion methods are normally analysed with an algebraic approach (see \([4,14,12]\)) 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 analyse discrete diffusion load balancing schemes. First, we sequentialise 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 sequentialised 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 sequentialised 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 analyse the related sequential load balancing algorithm, we can use existing ideas, e.g., from [10].

We use our technique to analyse the standard diffusion algorithm in the continuous and the discrete cases. Then we apply our technique to get results for the dynamic model of [7], where the network can change over time. Again, we get results for the discrete and the continuous cases. Finally, we step aside from the traditional neighbourhood load balancing approach and allow nodes to randomly choose their load balancing partners from among the 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 versions 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.

2.1.1. Diffusion

Cybenko [4] and, independently, Boillat [3], 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, \( L^t = (\ell^t_1, \ldots, \ell^t_n) \), where \( \ell^t_i \) is the load of node \( i \) at time \( t \geq 0 \). In each round \( t \), node \( i \) and node \( j \) compare their loads. If \( \ell^t_j > \ell^t_i \), node \( j \) sends \( \alpha (\ell^t_j - \ell^t_i) \) tokens to node \( i \). \( \alpha \) is called the diffusion factor and is set to \( 1/(\delta + 1) \), where \( \delta \) 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

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

\( M \) is commonly referred to as diffusion matrix. Cybenko [4] (see also [14, 12]) shows a tight connection between the convergence rate of his diffusion algorithm and the second largest eigenvalue of \( M \). Let \( \ell = \sum \ell_i/n \) be the average load and let \( b = (\ell, \ldots, \ell) \) be the balanced distribution. For each \( t \geq 0 \), define the error \( e^{(t)} \) to be \( e^{(t)} = b - \ell \). Let \( -1 \leq \mu_1 \leq \mu_2 \leq \cdots \leq \mu_n = 1 \) be the set of eigenvalues of \( M \) and denote \( \gamma = \max_{\mu \neq 1} |\mu| \) to be the second largest eigenvalue of \( M \). Let \( \|e^{(t)}\|_2 \) be the \( \ell_2 \) norm of the error vector \( e^{(t)} \). It can be shown that \( \|e^{(t)}\|_2 = \|M \cdot e^{(t)}\|_2 \leq \gamma \cdot \|e^{(0)}\|_2 \), which implies

\[
\|e^{(1)}\|_2 \leq \gamma^t \cdot \|e^{(0)}\|_2.
\]

Subramanian and Scherson [14] observe similar relations between convergence time and the properties of the underlying network. From Eq. (1), they obtain the following bound on the convergence time \( T \):

\[
\Omega \left( \frac{\log \sigma}{\gamma} \right) \leq T \leq O \left( \frac{n \gamma}{\sigma} \right)
\]

and

\[
\Omega \left( \frac{\log \sigma}{\lambda_2} \right) \leq T \leq O \left( \frac{\sigma}{\lambda_2} \right),
\]

where \( n \) is the size of the network, \( \sigma \) is the standard deviation of the initial load distribution, \( \Gamma \) and \( \Lambda \) are the network’s electrical and fluid conductance, respectively.

Ghosh et al. [12] refer to the above diffusion model as the first order scheme and further generalize it to the so called second order scheme, where

\[
L^t = \beta \cdot M L^{t-1} + (1 - \beta) \cdot L^{t-2},
\]

with \( \beta \) a constant. \( L^t \) does not only relate to \( L^{t-1} \) but also to \( L^{t-2} \), 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 \( \beta \). Diekmann et al. [5] extend the idea of [12] and propose a general framework to analyse 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 [8] the authors analyse the diffusion algorithm for dynamically changing networks. The results are stated in Theorem 7. The proof method is similar to the one in [5].

2.1.1. Dimension exchange

In [10], 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. They show that in each round the expected drop of \( \phi \) is at least \( 1/2 \delta \). Here, \( \lambda_2 \) 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 \( D = (d_i) \) with \( d_i = 0 \) if \( i \neq j \) and \( d_i \) the degree of node \( i \).

Note that several of the results presented above use techniques similar to techniques used for the analysis of the time it takes egodic Markov chains to reach their equilibrium state.
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 cannot be analysed using the algebraic technique of [4].

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 [10,12]). For the discrete version of their random matching based algorithm, by carefully calculating how much error can be introduced by rounding, Ghosh and Muthukrishnan [10] prove that, as long as \( \phi \geq 2\delta/\lambda_2 \), 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. [12] show that in the case of the discrete version of their first order scheme, the initial potential \( \phi_0 \) can be reduced to \( O(\delta^2 n^2 / e^2) \) in \( O((\log \phi_0) / (1 - (1 + e)^2)) \) steps.

Rabani, Sinclair and Wanka [13] 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 \( \ell \). They show for the idealized Markov chain that

\[
\frac{2}{\mu} \ln \left( \frac{Kn^2}{x} \right) \tag{2}
\]

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 \( \Psi \), which is the sum of load differences of the two systems across all edges of the network, aggregated over time. They obtain the following bound for \( \Psi \):

\[ \Psi(M) = O(\delta \log N / \mu) \].

Finally, applying the knowledge of the second largest eigenvalue and combining this with Eq. (2), 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 [6] propose a new discrete diffusion scheme which is fully randomized and distributed. Let \( K \) be the initial discrepancy (defined as above) and \( \delta \) be the maximum degree of the underlying graph. They show that, after \( O \left( \frac{k}{n^2} (\log n \log n + \log K) \right) \) 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 analyse 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 analyse diffusion discrete and continuous load balancing in a variety of underlying network models.

The key idea is to first sequentialise 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 sequentialised 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 [10]. Recently, a similar technique was used in [9,2]. To illustrate how the idea works, we first analyse Algorithm 1, a classic diffusion algorithm similar to the ones studied in [4,14,12]. Next, we consider Algorithm 2, which allows every user to randomly find its balancing partner. We again analyse it using the same proof idea; this shows that our technique is quite general.

Specifically, Section 4 analyses a diffusion algorithm (Algorithm 1) with concurrent load balancing actions. For the proof, we use a standard potential function \( \Phi \) (similar to the ones defined in [4,10,12,14]). 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 sequentialised algorithm. In other words, the concurrency can degrade the performance of our algorithm by at most a factor of two. Finally, we adopt the proof idea in [10] to analyse the sequentialised 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 [3,4,12,14]), 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 [10].

Next, we analyse the discrete version of Algorithm 1 and obtain similar results to the ones in [10,12]. 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 [12], 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 [10], our algorithm is still a constant times faster.

In Section 5 we use our proof method to get results similar to the ones in [8] for a dynamic network model where the active edges can change from round to round. In contrast to [8], we get also results for the discrete load balancing model.

In Section 6, we analyse Algorithm 2, which allows nodes to randomly 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 analysed in a model where nodes are allowed to randomly 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. Let $G = (V,E)$ be the underlying network. Let $\{e_1, e_2, \ldots, e_n\}$ be the set of edges of $G$. For each node $i \in V$, let $d_i$ be the degree of $i$, and let $\delta = \max_{i \in V} d_i$. $\alpha = \min_{(S, \overline{S})} |E(S, \overline{S})|$ 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 $\ell_i$ be the load of node $i$ at the end of round $t$. Whenever it is clear from the context we will simply write $\ell_i$ in the following. Then the vector $L = \{\ell_1, \ldots, \ell_n\}$ represents the entire load distribution. Now we are ready to define the load balancing algorithm that 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 [10]. Theorem 4 (presented below) is a function of the edge expansion value and the maximum degree of $G$. Let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of the Laplacian matrix of $G$ (for the definition of Laplacian matrix, see Section 2.1.1). Let $L^t = \{\ell_1^t, \ldots, \ell_n^t\}$, $t \geq 0$ be the load vector after $t$ balancing steps and $\overline{L} = \sum_{i=1}^n \ell_i / n$ the average load. In the following we will assume that all load vectors are normalized, i.e., $\ell_1^t \leq \ell_2^t \leq \cdots \leq \ell_n^t$. To analyse the algorithm, we will use the following potential function

$$\Phi(L') = \sum_{i=1}^n (\ell_i^t - \overline{L})^2.$$ 

The potential function sums up, for every node $i$, the difference between its load and the average load $\overline{L}$. $\Phi(L^{t-1}) - \Phi(L^t)$ is the potential drop in round $t$.

We assign a weight $w_{ij} = \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)}$ 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 $e_t^t$ with the smallest weight. Then we can define $L^{t,k} = (\ell_1^{t,k}, \ldots, \ell_n^{t,k})$ 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 $L^{t-1}$). $\Delta \Phi_t^k$ is the potential drop due to the activation of edge $e_t^t$ 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 $e_1 = (i,j) \in E$ we have

$$\Delta \Phi_t^k \geq w_{ij} |\ell_i^{t-1} - \ell_j^{t-1}|.$$ 

**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 $e_t$ is at most

$$w_{ij} = \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)}.$$ 

Node $i$ has at most $d_i - 1$ additional neighbours, hence it can send at most $(d_i - 1) \cdot \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)}$ load to other neighbours before the activation of edge $(i,j)$. Consequently,

$$\ell_i^{t, (k-1)} \geq \ell_i^{t-1} - (d_i - 1) \cdot w_{ij} = \ell_i^{t-1} - d_i \cdot \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)} + w_{ij}\right) \geq \ell_i^{t-1} - \frac{1}{4} |\ell_i^{t-1} - \ell_j^{t-1}| + w_{ij}. \quad (3)$$

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

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

Consequently,

$$\Delta \Phi_t^k = \left(\ell_i^{t,(k-1)} - \overline{L}\right)^2 + \left(\ell_j^{t,(k-1)} - \overline{L}\right)^2 - \left(\ell_i^{t,k} - \overline{L}\right)^2 - \left(\ell_j^{t,k} - \overline{L}\right)^2 \\
(a) = (\ell_i^{t,(k-1)})^2 + (\ell_j^{t,(k-1)})^2 - (\ell_i^{t,k})^2 - (\ell_j^{t,k})^2 \\
= (\ell_i^{t,(k-1)})^2 + (\ell_j^{t,(k-1)})^2 - (\ell_i^{t,k})^2 - (\ell_j^{t,k})^2 \\
- (\ell_i^{t,(k-1)} - w_{ij})^2 - (\ell_j^{t,(k-1)} + w_{ij})^2 \\
= 2w_{ij}(\ell_i^{t,(k-1)} - \ell_j^{t,(k-1)} - w_{ij}) \\
(b) \geq 2w_{ij} \left|\ell_i^{t-1} - \ell_j^{t-1}\right| \\
\geq w_{ij} |\ell_i^{t-1} - \ell_j^{t-1}|.$$ 

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 (3) and (4). \[ \square \]

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

**Lemma 2.** $\Phi(L^{t-1}) - \Phi(L^t) \geq \frac{1}{4d} \sum_{(i,j) \in E} |\ell_i^{t-1} - \ell_j^{t-1}|^2$.

**Proof.**

$$\Phi(L^{t-1}) - \Phi(L^t) \geq \sum_{e=(i,j) \in E} \Delta \Phi_t^k \\
\geq \sum_{(i,j) \in E} w_{ij} |\ell_i^{t-1} - \ell_j^{t-1}| \quad \text{(By Lemma 1)} \\
= \sum_{(i,j) \in E} \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)} \cdot |\ell_i^{t-1} - \ell_j^{t-1}|\right) \\
\geq \frac{1}{4d} \sum_{(i,j) \in E} \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)} \cdot |\ell_i^{t-1} - \ell_j^{t-1}|\right). \quad \square$$

We shall use the following lemma. See [10] for the full proof, which is an application of the Courant–Fischer Minimax Theorem.

**Lemma 3 (From [10]).**

$$\lambda_2 = \min_x \left(\frac{x^T \mathcal{L} x}{x^T x} \mid x \perp v_1, x \neq 0\right).$$
where \( v_1 = (1, 1, \ldots, 1)^T \) and \( x \perp v_1 \) means that \( x \) is orthogonal to \( v_1 \).

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 \( \Phi(L^T) \leq \epsilon \cdot \Phi(L^0) \).

**Proof.** Fix a round \( t \). We first lower bound \( \frac{\Phi(L^{t-1}) - \Phi(L^t)}{\Phi(L^{t-1})} \). The idea is similar to [10]. Define \( x \) to be a vector of length \( n \) with \( x_i = \ell_i^{t-1} - \tilde{t} \). Note that \( \sum_{i=1}^n x_i = 0 \), and that \( x \) is orthogonal to \( v_1 = (1, 1, \ldots, 1)^T \). Hence,

\[
\Phi(L^{t-1}) - \Phi(L^t) \geq \frac{\sum_{i,j \in E} (\ell_i^{t-1} - \ell_j^{t-1})^2}{\Phi(L^{t-1})} \quad \text{(By Lemma 2)}
\]

\[
= \frac{\sum_{i,j \in E} (x_i - x_j)^2}{4\delta \cdot \sum_{i=1}^n x_i^2}
\]

\[
= \frac{\sum_{i,j \in E} (x_i - x_j)^2}{4\delta \cdot \sum_{i=1}^n x_i^2}
\]

\[
= \frac{1}{\delta} \min_x \left( \frac{x^T L x}{x^T x} \right) \quad \text{subject to } x \perp v_1, x \neq 0
\]

\[
= \frac{\lambda_2}{4\delta}. \quad \text{(By Lemma 3.)} \quad (5)
\]

Hence, the potential drops by a constant factor in every round and we obtain

\[
\Phi(L^T) \leq \left( 1 - \frac{\lambda_2}{4\delta} \right)^T \Phi(L^0) = \left( 1 - \frac{\lambda_2}{4\delta} \right)^{\frac{\ln(1/\epsilon)}{\ln(1/\epsilon)}} \cdot \Phi(L^0) \leq \frac{1}{\epsilon} \Phi(L^0)
\]

where the second inequality is due to \( 0 < x < 1, (1 - x)^{1/x} < 1/\epsilon \). \( \square \)

**4.2. Discrete case**

In this section we analyse 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 \( \left\lfloor \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)} \right\rfloor \) 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 to converge against a “nearly balanced state”, but the difference is only a multiplicative constant.

**Lemma 5.** Fix a round \( t \). If \( \Phi(L^{t-1}) \geq 64\delta^3 n/\lambda_2 \) then \( \frac{\Phi(L^{t-1}) - \Phi(L^t)}{\Phi(L^{t-1})} \geq \frac{\lambda_2}{8\delta} \).

**Proof.**

\[
\frac{\Phi(L^{t-1}) - \Phi(L^t)}{\Phi(L^{t-1})} \geq \sum_{(i,j) \in E} \left( \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)} \right) \cdot |\ell_i^{t-1} - \ell_j^{t-1}| \]

\[
\geq \sum_{(i,j) \in E} \left( \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max(d_i, d_j)} - 1 \right) \cdot |\ell_i^{t-1} - \ell_j^{t-1}| \]

\[
\geq \frac{\lambda_2}{8\delta} \cdot \sum_{(i,j) \in E} (\ell_i^{t-1} - \ell_j^{t-1})^2 \]

\[
\geq \frac{\lambda_2}{8\delta} \cdot \Phi(L^{t-1})
\]

\[
\geq \frac{\lambda_2}{8\delta} \cdot \frac{64\delta^3 n}{\lambda_2} \]

\[
= \frac{8\delta}{\lambda_2} \cdot \Phi(L^{t-1}) \geq 64\delta^3 n/\lambda_2.
\]

According to (5) we have

\[
\Phi(L^T) \leq \left( 1 - \frac{\lambda_2}{8\delta} \right)^T \Phi(L^0) = \left( 1 - \frac{\lambda_2}{8\delta} \right)^{\frac{\ln(1/\epsilon)}{\ln(1/\epsilon)}} \cdot \Phi(L^0) \leq \frac{1}{\epsilon} \Phi(L^0)
\]

again, the second inequality is due to \( 0 < x < 1, (1 - x)^{1/x} < 1/\epsilon \). \( \square \)

**5. Diffusion on dynamic networks**

In [7], Elsässer et al. considered the diffusion process on dynamic networks, in which the set of nodes is fixed, but the set of communication edges may vary from round to round. They assume that in any given step, every node knows the set of active edges. The network can now be described by a sequence of “standard” graphs \( (G_n)_{n \geq 0} \), where \( G_n \) 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. Note that [7] 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 [7]). Denote \( \lambda_2^{(k)} \) and \( \delta^{(k)} \) to be the second smallest eigenvalue and the maximum degree of \( G_k \) respectively. Let
\[ A_k = \frac{\sum_{j=1}^k (\delta^{(j)}/\delta^{(k)})}{K} \] be the average value of \( \frac{\lambda^{(k)}}{\delta^{(k)}} \) occurring during the first \( K \) iterations. Algorithm 1 needs at most \( K \) steps to reduce the system potential to \( \Phi(L^0) \), where \( K = O(\ln(1/e)/A_k) \).

**Proof.** Let \( L^K \) be the load vector after \( K \) rounds of applying the discrete counterpart of Algorithm 1 on \( G_k \). Recall that \( A_k = \frac{\sum_{j=1}^k (\delta^{(j)}/\delta^{(k)})}{K} \). Then, for \( K \geq 4 \ln(1/e)/A_k \), it is true that

\[
\begin{align*}
\Phi(L^0) - \Phi(L^K) & \leq \prod_{k=1}^K \left( 1 - \frac{\lambda_k^2}{4\delta_k^2} \right) \\
& \leq \prod_{k=1}^K \left( e^{-\lambda_k^2/4\delta_k^2} \right) \\
& = e^{-K\lambda^2/4\delta^2} \leq e^{-\ln(1/e)} = e.
\end{align*}
\]

Here (a) is due to Eq. (5) of Theorem 4, (b) holds because \( 0 < x < 1, 1 - x < e^{-x} \). \( \square \)

### 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 \( \lambda_k^{(k)}, \delta_k^{(k)}, A_k \) be defined as above. The discrete counterpart of Algorithm 1 needs at most \( K \) steps to reduce the system potential to

\[ \Phi^* = 64n \cdot \max_{k=1}^K \left( \frac{\delta_k^{(k)}}{\lambda_k^{(k)}} \right)^3. \]

where \( K = O\left( \frac{\ln \Phi(L^0)/\Phi^*}{A_k} \right) \).

Similar to Lemma 5, one can show that whenever the potential is larger than some threshold \( \Phi^* \) defined above, the potential drops at least by a factor of \( \frac{1}{e^{1/2}} \) in iteration \( k \). The remaining part of the proof is similar to that of Theorem 7, thus the detail is omitted.

## 6. Randomly choosing balancing partners

In this section, we consider an alternative load balancing method (Algorithm 2) that 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 analysing technique to handle the concurrency, we can show that in round 1, the system potential drops by at least a constant factor. This implies that Algorithm 2 has a strict logarithmic convergence time.

**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 \( c_i, d_j \) be the load and the number of balancing partners of node \( i \). Our algorithm is as follows:

**Algorithm 2 Randomly picking balancing partners**

1. \( E = \emptyset \)
2. for every node \( i \in V \) do in parallel do
3. pick \( j \in V \setminus \{i\} \) uniformly at random
4. \( i \leftarrow E \cup (i, j) \)
5. end for
6. for every node \( i \in V \) do in parallel do
7. for every \( j \) such that \((i, j) \in E \)
8. if \( c_i > c_j \)
9. send \( x_{\max(d_i, d_j)} \) tokens from node \( i \) to \( j \)
10. end if
11. end for
12. end for

Below we analyse 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 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 \((i, j) \in E \), \( \Pr[\max(d_i, d_j) \leq 5 | (i, j) \in E] \geq 0.5 \).

**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 \( \Pr[d_i > 5 | (i, j) \in E] \):

\[
\Pr[d_i > 5 | (i, j) \in E] = \Pr[B(n - 1, 1/n) > 4] = \Pr[B(n - 1, 1/n) \geq 5] \\
\leq \binom{n - 1}{5} \left( \frac{1}{n} \right)^5 < \left( \frac{ne}{5} \right)^5 \left( \frac{1}{n} \right)^5 \\
= e^5 < 0.05.
\]

Similarly, we can prove that \( \Pr[d_j > 5 | (i, j) \in E] < \left( \frac{1}{n} \right)^4 < 0.25 \). Using \( \Pr[A \lor B] \leq \Pr[A] + \Pr[B] \), the following holds:

\[
\Pr[\max(d_i, d_j) \leq 5 | (i, j) \in E] \\
= 1 - \Pr[d_i > 5 \lor d_j > 5 | (i, j) \in E] \\
\geq 1 - \left( \Pr[d_i > 5 | (i, j) \in E] + \Pr[d_j > 5 | (i, j) \in E] \right) \\
> 1 - (0.05 + 0.25) > 0.5. \quad \square
\]

Before we prove Lemma 11, we show the following result.

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

**Proof.** Let \( \gamma_i = |\ell_i^j - \ell_i^j| \), and denote \( A \lor B \) to be the set of indices \( i \) for which \( \ell_i^j - \ell_i^j \leq T \) (or \( \ell_i^j > T \) resp.). First observe that
\[ \sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 = \sum_{i \in A} \sum_{j \in B} (y_i^2 + y_j^2 + 2y_iy_j) = |B| \sum_{j \in B} y_j^2 + |A| \sum_{i \in A} y_i^2 + 2 \sum_{i \in A} y_i \sum_{j \in B} y_j. \] (6)

Similar to (6), we get
\[ \sum_{i \in A} \sum_{j \in B} (y_i - y_j)^2 = \sum_{i \in A} \sum_{j \in B} (y_i^2 + y_j^2 - 2y_iy_j) = 2 \cdot |A| \sum_{j \in B} y_j^2 - 2 \left( \sum_{i \in A} y_i \right)^2. \] (7)
\[ \sum_{i \in A} \sum_{j \in B} (y_i - y_j)^2 = 2 \cdot |B| \sum_{j \in B} y_j^2 - 2 \left( \sum_{j \in B} y_j \right)^2. \] (8)

By Eqs. (6)–(8), we have:
\[ \sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i^j - \ell_j^i)^2 = \sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 + \sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 \]
\[ + \sum_{i \in A} \sum_{j \in B} (y_i - y_j)^2 + \sum_{i \in A} \sum_{j \in B} (y_i - y_j)^2 \]
\[ = 2 |A| |B| \sum_{i \in A} y_i^2 + 2 |A| |B| \sum_{j \in B} y_j^2 \]
\[ + 4 \left( \sum_{i \in A} y_i \right) \left( \sum_{j \in B} y_j \right) - 2 \left( \sum_{i \in A} y_i \right)^2 - 2 \left( \sum_{j \in B} y_j \right)^2 \]
\[ \overset{(a)}{=} 2 |A| |B| \left( \sum_{i \in A} y_i^2 + \sum_{j \in B} y_j^2 \right) \]
\[ \overset{(b)}{=} 2n \cdot \Phi(L^t). \]

Here (a) holds since \( \sum_{i \in A} y_i = \sum_{j \in B} y_j \), (b) is because \( |A| |B| = n \) and \( \Phi(L^t) = \sum_{i \in A} y_i^2 + \sum_{j \in B} y_j^2 \). Finally, by symmetry we get
\[ \sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i^j - \ell_j^i)^2 = n \cdot \Phi(L^t). \]

Now we are ready to prove the following lemma.

**Lemma 11.** \( \mathbb{E}[\Phi(L^{t+1}) | L^t = L] \leq \frac{19}{20} \Phi(L). \)

**Proof.** Recall that \( L \) is the set of links whose endpoints are balancing partners. For any \( i, j \in [n], i \neq j \), let \( W_{i,j} \) be a random variable indicating the amount of load items sent between node \( i \) and \( j \) and \( w_{i,j} = |\ell_i - \ell_j|/(4 \max(d_i, d_j)) \). Note that \( \mathbb{E}[W_{i,j}] = w_{i,j} \cdot \Pr[(i, j) \in E] \). By definition
\[ W_{i,j} = \begin{cases} 0 & \text{if } (i, j) \notin E \\ \frac{|\ell_i - \ell_j|}{4 \max(d_i, d_j)} & \text{otherwise.} \end{cases} \]

Let \( \Omega_k \) be the set of all possible choices for \( E \). Thus,
\[ \mathbb{E}[W_{i,j}] = \sum_{E \in \Omega_k} \left( \frac{|\ell_i - \ell_j|}{4 \max(d_i(E), d_j(E))} \right) \cdot \mathbb{P}[E] \]
\[ \geq \sum_{E \in \Omega_k, \max(d_i(E), d_j(E)) \leq 5} \left( \frac{|\ell_i - \ell_j|}{4 \cdot 5} \right) \cdot \Pr[(i, j) \in E] \]
\[ \geq \left( \frac{|\ell_i - \ell_j|}{4 \cdot 5} \right) \cdot \Pr[\max(d_i, d_j) \leq 5 \text{ and } (i, j) \in E] \]
\[ = \left( \frac{|\ell_i - \ell_j|}{20} \right) \cdot \Pr[\max(d_i, d_j) \leq 5 \text{ and } (i, j) \in E] \cdot \Pr[(i, j) \in E] \]
\[ \geq \frac{|\ell_i - \ell_j|}{20n}. \]

The last step is due to Lemma 9 and \( \Pr[(i, j) \in E] = 2/(n - 1) < 2/n \).

Now we are ready to bound the expected potential drop at step \( t \).
\[ \mathbb{E}[\Phi(L^{t+1}) | L^t = L] \leq \Phi(L) - \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l \in \mathbb{L}} w_{i,j} \cdot |l_i - l_j| \]
\[ = \Phi(L) - \sum_{i=1}^{n} \sum_{j=1}^{n} (w_{i,j} \cdot |\ell_i - \ell_j| \cdot \Pr[(i, j) \in E]) \]
\[ = \Phi(L) - \sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i - \ell_j) \cdot \mathbb{E}[W_{i,j}] \]
\[ < \Phi(L) - \frac{1}{20n} \sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i - \ell_j)^2 \]
\[ = \Phi(L) - \frac{\Phi(L)}{20} = \frac{19}{20} \Phi(L). \]

Here the first inequality is due to Lemma 1 and the last equality holds by Lemma 10. \( \square \)

Finally, we prove the following convergence theorem.

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

**Proof.** For any \( t > 0 \), by linearity of expectation, we can repeatedly use Lemma 11, and obtain
\[ \mathbb{E}[\Phi(L^{t+1})] \leq \frac{19}{20} \Phi(L^t). \]

By Markov’s inequality, \( \Pr[\Phi(L^{t+1}) < \Phi(L^t)/2] \geq 1/2 \). Denote a stage to be 30 rounds. Let \( k = 4 \log \Phi(L) \). For stage \( 0 \leq i \leq k \), define the random variable
\[ X_i = \begin{cases} 1 & \text{if } \Phi(L^{30(i+1)}) \leq \Phi(L^{30i})/2 \\ 0 & \text{otherwise.} \end{cases} \]

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 \( \mathbb{E}[X] \geq k/2 \). By Chernoff,
\[ \Pr[X \leq c \ln \Phi(L)] \leq e^{-c \ln \Phi(L)/2} \leq e^{-k/16} \leq \Phi(L)^{-c/4}. \]

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 \( T \geq 120c \cdot \ln \Phi(L) \), \( \Pr[\Phi(L^T) \leq e^{-c}] \geq 1 - \Phi(L)^{-c/4} \). \( \square \)

**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 alone 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 \( \ell_i \geq \ell_j \), we transfer \( \frac{|\ell_i - \ell_j|}{4 \max(d_i, d_j)} \) tokens from \( \ell_i \) to \( \ell_j \). We show the following result indicating that whenever the potential \( \Phi(L) \) is bigger than a threshold of 1600n, the potential drops at least by a constant factor of \( \frac{1}{20} \) in every iteration.

**Lemma 13.** If \( \Phi(L) \geq 1600n \), \( \mathbb{E}[\Phi(L^{t+1}) | L^t = L] \leq \frac{19}{20} \Phi(L) \).
**Proof.** Following the proof of Lemma 11, we can show that $\mathbb{E}[W_6] \leq |\ell_i - \ell_j|/20/n$. We then bound the expected potential change as follows:

$$\mathbb{E}[\Phi(L^{t+1})|L^t = L] \leq \Phi(L) - |\ell_i - \ell_j| \cdot \frac{n}{20} \cdot \frac{n}{20} \cdot \mathbb{E}[W_6]$$

$$= \Phi(L) - \frac{|\ell_i - \ell_j|}{n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \frac{|\ell_i - \ell_j|}{20} \right)$$

$$\leq (a) \Phi(L) - \frac{1}{20n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i - \ell_j)^2 + \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} |\ell_i - \ell_j|$$

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

**Theorem 14.** \( \forall \epsilon > 0, \text{ after } T \geq 240c \ln \left( \frac{\Phi(L)}{1600n} \right) \) rounds, \( \Pr[\Phi(L^T) \leq 1600n] \geq 1 - \left( \frac{\Phi(L)}{1600n} \right)^{-c/4}. \)

7. Conclusion

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

Acknowledgment

This paper was written while the third author was at the School of Computing Science, Simon Fraser University, Burnaby, B.C., V5A 1S6, Canada.

References