Tight Analyses of Two Local Load Balancing Algorithms

Bhaskar Ghosh [1] *F. T. Leighton* [2] *Bruce M. Maggs* [3,4]
S. Muthukrishnan [5] *C. Greg Plaxton* [6,7] *R. Rajaraman* [6,7]
Andréa W. Richa [3] *Robert E. Tarjan* [8] *David Zuckerman* [6,9]

Abstract

This paper presents an analysis of the following load balancing algorithm. At each step, each node in a network examines the number of tokens at each of its neighbors and sends a token to each neighbor with at least $2d+1$ fewer tokens, where d is the maximum degree of any node in the network. We show that within $O(\Delta/\alpha)$ steps, the algorithm reduces the maximum difference in tokens between any two nodes to at most $O((d^2 \log n)/\alpha)$, where Δ is the maximum difference between the number tokens at any node initially and the average number of tokens, n is the number of nodes in the network, and α is the edge expansion of the network. The time bound is tight in the sense that for any graph with edge expansion α, and for any value Δ, there exists an initial distribution of tokens with imbalance Δ for which the time to reduce the imbalance to even $\Delta/2$ is at least $\Omega(\Delta/\alpha)$. The bound on the final imbalance is tight in the sense that there exists a class of networks that can be locally balanced everywhere (i.e., the maximum difference in tokens between any two neighbors is at most $2d$), while the global imbalance remains $\Omega((d^2 \log n)/\alpha)$. Furthermore, we show that upon reaching a state with a global imbalance of $O((d^2 \log n)/\alpha)$, the time for this algorithm to locally balance the network can be as large as $\Omega(n^{1/2})$. We extend our analysis to a variant of this algorithm for dynamic and asynchronous networks. We also present tight bounds for a randomized algorithm in which each node sends at most one token in each step.

[1]Department of Computer Science, Yale University, New Haven, CT 06520. Supported by ONR Grant 4-91-J-1576 and a Yale/IBM joint study. Email: ghosh@cs.yale.edu.

[2]Department of Mathematics and Laboratory for Computer Science, MIT, Cambridge, MA 02139. Supported by ARPA Contracts N00014-91-J-1698 and N00014-92-J-1799. Email: ftl@math.mit.edu.

[3]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. Email: {aricha, bmm}@cs.cmu.edu.

1 Introduction

A natural way to balance the workload in a distributed system is to have each work station periodically poll the other stations to which it is connected, and send some of its work to stations with less work pending. This paper analyzes the effectiveness of this local load balancing strategy in the simplified scenario in which each work station has a collection of independent unit-size jobs (called *tokens*) that can be executed on any other work station. We model a distributed system as a graph, where nodes correspond to work stations, and edges correspond to connections between stations, and we assume that in one unit of time, at most one token can be transmitted across an edge of the graph in each direction. Our analysis addresses only the static load balancing aspect of this problem; we assume that each processor has an initial collection of tokens, and that no tokens are created or destroyed while the tokens are being balanced.

We analyze the algorithms in this paper in terms of the initial *imbalance* of tokens, i.e., the maximum difference between the number of tokens at any node and the average number of tokens, which we denote Δ, the number of nodes in the graph, which we denote n, the maximum degree of the graph, d, and the node and edge expansion of the graph. We define the *node expansion* μ, of a graph G to be the largest value such that every set S of $n/2$ or fewer nodes in G has at least $\mu|S|$ neighbors outside of S. We define the *edge expansion* α, of a graph

[4]Supported in part by an NSF National Young Investigator Award, No. CCR-9457766, and by ARPA Contract F33615-93-1-1330.

[5]DIMACS, Rutgers University, Piscataway, NJ 08855. Supported by DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, a National Science Foundation Science and Technology Center, under NSF Contract STC-8809648. Email: muthu@dimacs.rutgers.edu.

[6]Department of Computer Science, University of Texas at Austin, Austin, TX 78712.
Email: {diz, plaxton, rraj}@cs.utexas.edu.

[7]Supported by the Texas Advanced Research Program under Grant No. ARP-93-003658-461.

[8]Department of Computer Science, Princeton University, and NEC Research Institute, 35 Olden Street, Princeton, NJ 08544. Research at Princeton University supported in part by the National Science Foundation, Grant No. CCR-8920505, and the Office of Naval Reserach, Contract No. N0014-91-J-1463. Email: ret@cs.princeton.edu.

[9]Supported in part by an NSF National Young Investigator Award, No. CCR-9457799.

G to be the largest value such that for every set S of $n/2$ or fewer nodes in G, there are at least $\alpha|S|$ edges in G with one endpoint in S and the other not in S.

The performance of an algorithm is characterized by the time that it takes to balance the tokens, and by the final balance that it achieves. We say that an algorithm *globally balances* (or just *balances*) to within t tokens if the maximum difference in the number of tokens between any two nodes in the graph is at most t. We say that an algorithm *locally balances* to within t tokens if the maximum difference in the number of tokens between any neighboring nodes in the graph is at most t.

We analyze two different types of algorithms in this paper, single-port and multi-port. In the *single-port* model, a node may transmit or receive at most one token in one unit of time. In the *multi-port* model, a node may simultaneously transmit or receive a token across all of its edges (there may be as many as d) in a single unit of time. Not surprisingly, the load balancing algorithms run faster in the multi-port model. In practice, however, single-port nodes may be prefered to multi-port nodes because they are easier and less costly to build.

1.1 Our results

This paper analyzes the simplest and most natural local algorithms in both the single-port and multi-port models.

In the single-port algorithm, a matching is randomly chosen at each step. First, each (undirected) edge in the network is independently selected to be a *candidate* with probability $1/4d$. Then each candidate edge (u, v) for which there is another candidate edge (u, x) or (y, v) is removed from the set of candidates. The remaining candidates form a matching M in the graph. For each edge (u, v) in M, if u and v have the same number of tokens, then nothing is sent across (u, v). Otherwise, a token is sent from the node with more tokens to the node with fewer. This algorithm was first analyzed in [15].

We analyze the performance of the single-port algorithm in terms of both the edge expansion and the node expansion of the graph. In terms of edge expansion, we show that the single-port algorithm balances to within $O(d \log n/\alpha)$ tokens in $O(d\Delta/\alpha)$ steps, with high probability. In terms of node expansion, the final imbalance is $O(\log n/\mu)$, and the time is $O(d\Delta/\mu)$, with high probability. (To compare these bounds, note that $\mu \le \alpha \le d\mu$.) The time bounds are tight in the sense that for many values of n, d, α, and Δ, there is an n-node degree-d graph with edge expansion α or node expansion μ and an initial placement of tokens with imbalance Δ where the time (for any algorithm) to balance to within even $\Delta/2$ tokens is at least $\Omega(d\Delta/\alpha)$. Similarly, in terms of node expansion, there exist classes of graphs where the time to balance to within even $\Delta/2$ tokens is at least $\Omega(d\Delta/\mu)$.

The multi-port algorithm is simpler and deterministic. At each step, a token is sent from node u to node v across edge (u, v) if at the beginning of the step node u contained at least $2d+1$ more tokens than node v. This algorithm was first analyzed in [2].

As in the single-port case, we analyze the multi-port algorithm in terms of both edge expansion and node expansion. In terms of edge expansion, the algorithm balances to within $O(d^2 \log n/\alpha)$ tokens in $O(\Delta/\alpha)$ steps. This bound is tight in the sense that for any network with edge expansion α, and any value Δ, there exists an initial distribution of tokens with imbalance Δ such that the time to reduce the imbalance to even $\Delta/2$ is $\Omega(\Delta/\alpha)$. In terms of node expansion, the algorithm balances to within $O(d \log n/\mu)$ tokens in $O(\Delta/\mu)$ time. This bound is tight in the sense that for many values of d, n, and μ, and any value Δ, there exists an n-node degree-d graph with node expansion μ and an initial distribution of tokens with imbalance Δ for which the time to balance to within $\Delta/2$ tokens is $\Omega(\Delta/\mu)$.

Both the single-port and multi-port algorithms will eventually locally balance the network, the single-port algorithm to within one token, and the multi-port algorithm to within $2d$ tokens. However, even after reducing the global imbalance to a small value, the time for either of these algorithms to reach a locally balanced state can be quite large. In particular, we show that after reaching a state that is globally balanced to within $O(d \log n/\mu)$ tokens, the multi-port algorithm may take another $\Omega(n^{1/2})$ steps to reach a state that is locally balanced to within $2d$ tokens. For networks with large node expansion and small degree, e.g., $\mu = \Omega(1)$ and $d = O(1)$, and small initial imbalance, e.g., $\Delta = O(d \log^2 n/\mu)$, the time to locally balance the network, $\Omega(n^{1/2})$, may be much larger than the time, $O(\Delta/\mu) = O(d \log^2 n/\mu^2) = O(\log^2 n)$ to reach a state that is globally balanced to within $O(d \log n/\mu)$ tokens. We prove similar bounds in terms of edge expansion and also for the single-port algorithm.

Thus far we have described a network model in which the nodes are synchronized by a global clock (i.e., a *synchronous* network), and in which the edges are assumed not to fail. With minor modifications, however, the load balancing algorithms can be made to work in both asynchronous and dynamic networks. In a *dynamic* network, the set of edges in the network may vary at each time step. In any time step, a *live* edge is one that can transmit one message in each direction. We assume that at each time step, each node in a synchronous dynamic network knows which of its edges are live. In an *asynchronous* network, the topology is fixed, but an adversary determines the speed at which each edge operates at every instant of time. For every undirected edge between two nodes, we allow at most two messages to be in transit at any instant in time. These messages may travel in opposite directions across the edge, or both may travel in one direction, while no message travels in the opposite direction. An edge is said to be *live* for a unit interval of time if every message that was in transit across the edge (in either direction) at the beginning of the interval is guaranteed to reach the end of the edge by the end of the interval. We analyze the performance of the multi-port load balancing algorithm under the assumption that at each time step, the set of live edges has some edge expansion α, or node expansion μ.

We also study the off-line load balancing problem, in which every node has knowledge of the global state of the network. This problem on static synchronous networks has been studied in [28]. We use their results to obtain tight bounds on off-line load balancing in terms of edge expansion and node expansion. In particular, we prove that any network can be balanced off-line to within two tokens in at most $\lceil (1+\mu)\Delta/\mu \rceil$ steps in the single-port model. Moreover, there exists a network and an initial token distribution for which any single-port off-line algorithm takes more than $\lceil (1+\mu)\Delta/\mu \rceil$ steps to balance the network to within one token. Similarly, in the multi-port model, any network can be balanced off-line to within d tokens in at most $\lceil \Delta/\alpha \rceil$ steps. It is easy to observe that for any network G there exists an initial token distribution such that any algorithm will take at least $\lceil \Delta/\alpha \rceil$ steps to balance G to within one token.

1.2 Previous and related work

Load balancing has been studied extensively since it comes up in a wide variety of settings including adaptive mesh partitioning [17, 37], fine grain functional programming [16], job scheduling in operating systems [13, 24], and distributed game tree searching [22, 25]. A number of models have been proposed for load balancing, differing chiefly in the amount of global information used by the algorithm [2, 11, 12, 15, 26, 30]. On these models, algorithms have been proposed for specific applications; also, proposed heuristics and algorithms have been analyzed using simulations and queuing–theoretic techniques [27, 34, 36]. In what follows, we focus on models that allow only local algorithms and on prior work that takes an analytical approach to the load balancing problem.

Local algorithms restricted to particular networks have been studied on counting networks [4, 23], hypercubes [20, 33], and meshes [17, 28]. Another class of networks on which load balancing has been studied is the class of expanders. Peleg and Upfal [31] pioneered this study by identifying certain small-degree expanders as being suitable for load balancing. Their work has been extended in [9, 18, 32]. These algorithms either use strong expanders to approximately balance the network, or the AKS sorting network [3] to perfectly balance the network. Thus, they do not work on networks of arbitrary topology. Also, these algorithms work by setting up fixed paths through the network on which load is moved and therefore fail when the network changes. In contrast, our local algorithm works on any arbitrary dynamic network that remains connected.

On arbitrary topologies, load balancing has been studied under two models. In the first model, any amount of load can be moved across a link in any time step [8, 12, 15, 19, 35]. The second model is the one that we adopt here, namely one in which at most one unit load can be moved across a link in each time step. Load balancing algorithms on the second model were first proposed and analyzed in [2] for the multi-port variant and in [15] for the single-port variant. The upper bounds established by them are suboptimal by a factor of $\Omega(\sqrt{n})$ or $\Omega(\log(n\Delta))$ in general. Our result here is an improved, in fact, an optimal bound for the same problem.

As remarked earlier, our multi-port results (and those in [2]) hold even for dynamic or asynchronous networks. In general, work on dynamic and asynchronous networks has been limited. In work related to load balancing for instance, an end-to-end communication problem, namely one in which messages are routed from a single source to a single destination, has been studied in [1, 7] on dynamic networks. Our scenario is substantially more involved since we are required to move load between several sources and destinations simultaneously. Another result on dynamic networks is the recent analysis of a local algorithm for the approximate multicommodity flow problem [5, 6]. While their result has several applications including the end-to-end communication problem mentioned above, it does not seem to extend to load balancing. Our result on load balancing is related to their work in the technique; however, our algorithm and analysis are simpler and we obtain optimal bounds for our problem.

The convergence of local load balancing algorithms is related to that of random walks on Markov Chains. Indeed the convergence bounds in both cases depend on the expansion properties of the underlying graph and they are established using potential function arguments. There are however two important differences. First, the analysis of the rapid convergence of random walks [21, 29] relies on averaging arbitrary probabilities across any edge. This corresponds to sending an arbitrary (possibly nonintegral) load along an edge which is forbidden in our model. In this sense, the analysis in [12] (and all references in the unbounded capacity model) are similar to the random walk analysis. Second, our argument uses an exponential potential function. The analyses in

[12, 21, 29], in contrast, use quadratic potential functions. Our potential function and our amortized analysis were necessary since a number of previous attempts using quadratic potential functions yielded suboptimal results [2, 15] for local load balancing.

1.3 Outline

The remainder of this paper is organized as follows. Section 2 contains some definitions. Section 3.1 analyzes the performance of the single-port algorithm. Section 3.2 analyzes the performance of the multi-port algorithm. In Section 4, we show that the time to reach a locally balanced state can be quite large, even if the network starts in a state that is well balanced globally. Section 5 describes extensions to dynamic and asynchronous networks. Finally, Section 6 presents tight bounds on off-line load balancing.

Due to space limitations, many of the proofs have been omitted in this version of the paper. Complete proofs may be found in [14].

2 Preliminaries

For any network $G = (V, E)$ with n nodes, m edges, and edge expansion α, we denote the number of tokens at $v \in V$ by $w(v)$. We denote the average number of tokens by ρ. For simplicity, throughout this paper we assume that ρ is an integer. We assign a unique *rank* from $[1, w(v)]$ to every token at v. The *height* of a token is its rank minus ρ. The height of a node is the maximum among the heights of all its tokens.

Consider a partition of V given by $\{S_i\}$, where the index i is an integer and S_i may be empty for any i. Let $S_{>j}$ be $\cup_{i>j} S_i$. Similarly we define $S_{\geq j}$, $S_{<j}$, and $S_{\leq j}$. We define index i to be *good* if $|S_i| \leq \alpha |S_{>i}|/2d$. An index that is not good is called a *bad* index. For any bad index i, we have $|S_{>i}| < |S_{\geq i}|/(1+\alpha/(2d))$. Since $|S_{>0}|/(1+\alpha/(2d))^{\log_{(1+\alpha/(2d))} n} \leq 1$, there can be at most $\lceil \log_{(1+\alpha/(2d))} n \rceil$ bad indices. It follows that at least half of the indices in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ are good.

3 Analysis for static synchronous networks

3.1 The single-port model

In this section, we analyze the single-port load balancing algorithm that is described in Section 1.1.

Theorem 3.1 *For an arbitrary network G with n nodes, degree d, edge expansion α, and initial imbalance Δ, the single-port algorithm balances within $O((d \log n)/\alpha)$ tokens in $O((d\Delta)/\alpha)$ steps, with high probability.*

Before every step we partition the set of nodes according to how many tokens they contain. For every integer i, we denote the set of nodes having $\rho + i$ tokens as S_i. Consider the first T steps of the algorithm, with T to be specified later. It holds that either $|S_{>0}| \leq n/2$ at the start of at least half the steps, or $|S_{\leq 0}| \leq n/2$ at the start of at least half the steps. Without loss of generality, assume the former is true. Since at least half of the indices in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ are good in any time step t, there exists an index j in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ that is good in at least half of those time steps in which $|S_{>0}| \leq n/2$. Hence j is good in at least $T/4$ steps.

With every token at height h we associate a potential of $\phi(h)$, where $\phi : N \to R$ is defined as follows:

$$\phi(x) = \begin{cases} 0 & \text{if } x \leq j, \\ (1+\nu)^x & \text{otherwise,} \end{cases} \tag{1}$$

where $\nu = \alpha/(cd)$, where $c > 1$ is a real constant to be specified later. The potential of the network is the sum of the potentials of all tokens in the network. While transmitting a token, every node sends its token with maximum height. Similarly, any token arriving into a node with height h is assigned height $h + 1$. It follows from the definition of the potential function, and the fact that the height of a token never increases, that the potential of the network never increases. In the following, we show that during any step when j is good, the expected decrease in potential of the network is at least an $\varepsilon\nu^2$ fraction of the potential before the step, where $\varepsilon > 0$ is a real constant to be specified later.

Before proving Theorem 3.1, we present an informal outline of the proof. For simplicity, let us assume that G is a constant-degree expander, i.e., $d = O(1)$ and $\mu = \Omega(1)$. Consider the scenario in which all of the indices greater than j are bad. In this situation, for every i greater than j, the size of the set $S_{\geq i}$ grows exponentially with decreasing i, and hence the number of tokens with height i grows exponentially with decreasing i. If the growth of $\phi(x)$ with increasing x is slower than the growth of $S_{\geq i}$ with decreasing i, then the total potential due to tokens at height i "dominates" the total potential due to tokens at height greater than i. In such a case the potential of $S_{>j}$ is essentially a constant times the potential of tokens at height $j + 1$. In addition, if the potential of tokens of height at most j is zero, then in every step when j is good, there is a constant fraction potential drop, because a constant fraction of nodes in $S_{>j}$ send tokens to $S_{<j}$ in such a step. The exponential function we have defined in Equation (1) satisfies the properties described above.

In general, the indices greater than j could form any sequence of good and bad indices that respects the upper bound on the number of bad indices. We consider the indices greater than j in reverse order and show by an amortized analysis that for each index i we can "view" all indices greater than or equal to i as bad. If i is bad, then this view is trivially preserved; otherwise, there is a significant potential drop across the cut $(S_{\leq i}, S_{>i})$, and this drop can be used to rearrange the potential of $S_{>i}$ in order to maintain the view that all indices greater than i are bad. We then invoke the argument for the case in which all indices greater than j are bad and complete the proof.

Consider step t of the algorithm. Let Φ_t denote the potential of the network after step $t > 0$. Let M_i be the set of tokens that are sent from a node in $S_{>i}$ to a node in $S_{<i}$. Let $m_i = |M_i|$. We say that a token p has an *i-drop* of $\phi(i+1) - \phi(i)$, if p moves from a node in $S_{>i}$ to a node in $S_{<i}$. Thus, the potential drop due to a token moving on an edge from node $u \in S_i$ to node $v \in S_{i'}$, $i > i' + 1$, can be expressed as the sum of k-drops for $i' < k < i$. In Lemma 3.1, we use this notion of i-drops to relate the total potential drop in step t, Ψ, to the $m_i's$.

Lemma 3.1

$$\Psi = \left(\sum_{i>j} m_i \nu (1+\nu)^i \right) + m_j (1+\nu)^{j+1}.$$

Proof: Let M be the set of tokens that are moved from a node in $S_{>j}$. (Note that tokens that start from and end at nodes in $S_{>j}$ also belong to M.) For any token p, let $a(p)$ (resp., $b(p)$) be the height of p before (resp., after) step t.

$$\begin{aligned}
\Psi &= \sum_{p \in M} (\phi(a(p)) - \phi(b(p))) \\
&= \sum_{p \in M,\, p \notin M_j} \sum_{b(p) \leq i < a(p)} (\phi(i+1) - \phi(i)) \\
&\quad + \sum_{p \in M_j} \sum_{j \leq i < a(p)} (\phi(i+1) - \phi(i)) \\
&= \left(\sum_{i>j} \sum_{p \in M_i} \nu (1+\nu)^i \right) + \left(\sum_{p \in M_j} (1+\nu)^{j+1} \right) \\
&= \left(\sum_{i>j} m_i \nu (1+\nu)^i \right) + m_j (1+\nu)^{j+1}.
\end{aligned}$$

(In the third step we rearrange the terms in the summations and also use the equation $\phi(i+1) - \phi(i) = \nu(1+\nu)^i$ for all $i > j$.) ■

We now describe the amortized analysis, alluded to earlier in this section, that we use to prove Theorem 3.1. We associate a charge of $\varepsilon \nu^2 \phi(h)$ with each token at height h. We show that we can pay for all the charges using the expected potential drop $E[\Psi]$, and thus place a lower bound on $E[\Psi]$. We consider the indices in $[j+1, \ell]$ in reverse order, where ℓ is the maximum token height. Corresponding to every i in $[j, \ell]$, we maintain a "debt" term, given by Γ_i below, which is the difference between the charges due to tokens at height greater than i and the sum of i'-drops for $i' > i$. Hence Γ_i is calculated by subtracting the sum of i-drop's from Γ_{i+1} and adding the charges due to tokens at height i. We will show that $E[\Gamma_i]$ is such that the indices in $[i+1, \ell]$ can be viewed as a sequence of bad indices. In other words, we upper bound $E[\Gamma_i]$ by $O(\nu |S_{\geq i}|(1+\nu)^i)$. It follows from this upper bound and the informal argument outlined earlier in this section that $E[\Gamma_{j+1}]$, i.e., the expected total debt, can be paid for by the expected drop across index j.

Formally, for any $i > j$, we define

$$\begin{aligned}
\Psi_i &= \sum_{k \geq i} m_k \nu (1+\nu)^k, \text{ and} \\
\Gamma_i &= (\varepsilon \nu^2) \left(\sum_{p: a(p) \geq i} (1+\nu)^{a(p)} \right) - \Psi_i.
\end{aligned}$$

We also define

$$\Gamma = (\varepsilon \nu^2) \left(\sum_{p: a(p) > j} (1+\nu)^{a(p)} \right) - \Psi.$$

Note that $\Phi_{t-1} = \sum_{p: a(p) > j} (1+\nu)^{a(p)}$ is the total potential of $S_{>j}$ prior to step t.

In order to prove the upper bound on $E[\Gamma_i]$, we place a lower bound on $E[m_i]$ that is obtained from the following lemma of [15].

Lemma 3.2 ([15]) *For any edge $e \in E$, the probability that e is selected in the matching is at least $1/(8d)$.* ■

Lemma 3.3 *There exists a real constant $\varepsilon > 0$ such that for all $i > j$, we have $E[\Gamma_i] \leq (\varepsilon \nu) |S_{\geq i}| (1+\nu)^i$.*

Proof: The proof is by reverse induction on i. For $i \geq \ell$, the claim holds trivially. Therefore, for the base case we consider $i = \ell$. Since $m_\ell = 0$, we have $\Psi_\ell = 0$. Thus, $\Gamma_\ell = (\varepsilon \nu^2) |S_\ell| (1+\nu)^\ell \leq (\varepsilon \nu) |S_{\geq \ell}| (1+\nu)^\ell$, since $\nu = \alpha/(cd) \leq 1/c \leq 1$ by our choice of c.

For the induction step we consider two cases, depending on whether i is good or bad. If i is good, then $|S_i| \leq \alpha |S_{>i}|/2d$. Since there are at most $\alpha |S_{>i}|/2$ edges from $S_{>i}$ to S_i, by the expansion property of the graph we find that there are at least $\alpha |S_{>i}|/2$ edges from $S_{>i}$ to $S_{<i}$. By Lemma 3.2 we have $E[m_i] \geq \alpha |S_{>i}|/(16d)$. Therefore, we have

$$\begin{aligned}
E[\Gamma_i] &= E[\Gamma_{i+1}] + (\varepsilon \nu^2) |S_{\geq i}| (1+\nu)^i - E[m_i] \nu (1+\nu)^i \\
&\leq E[\Gamma_{i+1}] + (\varepsilon \nu^2) |S_{\geq i}| (1+\nu)^i \\
&\quad - c\nu^2 |S_{>i}| (1+\nu)^i / 16 \\
&\leq E[\Gamma_{i+1}] - (\nu^2) |S_{\geq i}| (1+\nu)^i (f(c, \alpha, d) - \varepsilon) \\
&\leq (\varepsilon \nu) |S_{>i}| (1+\nu)^{i+1} \\
&\quad - (\nu^2) |S_{\geq i}| (1+\nu)^i (f(c, \alpha, d) - \varepsilon) \\
&\leq (\varepsilon \nu) |S_{\geq i}| (1+\nu)^i ((1+\nu) - \nu (f(c, \alpha, d) - \varepsilon)/\varepsilon),
\end{aligned}$$

where $f(c, \alpha, d) = c/(16(1 + \alpha/(2d)))$. (The third inequality follows from the fact that $|S_{>i}| \geq |S_{\geq i}|/(1 + \alpha/(2d))$. The fourth inequality follows from the induction hypothesis.)

The second case is when i is bad. Thus $|S_i| \geq \alpha|S_{>i}|/(2d)$. We have

$$\begin{aligned} E[\Gamma_i] &\leq E[\Gamma_{i+1}] + (\varepsilon\nu^2)|S_{\geq i}|(1+\nu)^i \\ &\leq (\varepsilon\nu)|S_{>i}|(1+\nu)^{i+1} + (\varepsilon\nu^2)|S_{\geq i}|(1+\nu)^i \\ &\leq (\varepsilon\nu)|S_{\geq i}|(1+\nu)^i((1+\nu)/(1+c\nu/2)+\nu). \end{aligned}$$

We now complete the induction step by determining values for c and ε such that the inequalities

$$((1+\nu) - \nu(f(c,\alpha,d) - \varepsilon)/\varepsilon) \leq 1 \quad (2)$$

$$(1+\nu)/(1+c\nu/2) + \nu \leq 1 \quad (3)$$

hold, and thus establish the induction step. Since $\alpha \leq d$, we can set c to be any constant greater than or equal to $(\alpha/d)+4$ (e.g., $c = 5$). For this choice of c, $\nu \leq (c-4)/c$, and hence $\nu \leq (c\nu - c\nu^2)/4$. Therefore,

$$\begin{aligned} (1+\nu)/(1+c\nu/2)+\nu &= (1+2\nu+c\nu^2/2)/(1+c\nu/2) \\ &\leq (1+c\nu/2)/(1+c\nu/2) = 1. \end{aligned}$$

Thus, Equation (3) is satisfied. Since $\alpha \leq d$, we find that $f(c, \alpha, d) \geq c/24$. We now set $\varepsilon = c/48$ to establish Equation (2). (For example, $c = 5$ and $\varepsilon = 5/48$.) ■

Applying Lemma 3.3 with $i = j + 1$, it follows that $E[\Gamma_{j+1}] \leq (\varepsilon\nu)|S_{\geq j+1}|(1+\nu)^{j+1}$. If j is good then $E[m_j] \geq \alpha|S_{>j}|/16d$, and by the definitions of Γ, Γ_{j+1}, and Ψ, we have

$$\begin{aligned} E[\Gamma] &= E[\Gamma_{j+1}] - E[m_j](1+\nu)^{j+1} \\ &\leq E[\Gamma_{j+1}] - \alpha|S_{>j}|(1+\nu)^{j+1}/(16d) \\ &\leq (\varepsilon\nu)|S_{>j}|(1+\nu)^{j+1} - \alpha|S_{>j}|(1+\nu)^{j+1}/(16d) \\ &\leq \nu|S_{>j}|(1+\nu)^{j+1}(\varepsilon - c/16) \\ &\leq 0, \end{aligned}$$

since $c/16 \geq \varepsilon$.

By the definitions of Ψ and Γ, we have $\Phi_t \leq \Phi_{t-1} - \Psi$ and $\Gamma = \varepsilon\nu^2\Phi_{t-1} - \Psi$. If j is good during step t, we have $E[\Gamma] \leq 0$, and therefore, $E[\Phi_t] \leq \Phi_{t-1}(1-\varepsilon\nu^2)$, where the expectation is taken over the random matching selected in step t. Thus, we have $E[\Phi_{t+T}] \leq \Phi_t(1-\varepsilon\nu^2)^{T/4}$, where the expectation is over all the random matchings in the T steps. By setting $T = \lceil(4\ln 4)/(\varepsilon\nu^2)\rceil$, we obtain $E[\Phi_{t+T}] \leq \Phi_t/4$. By Markov's inequality, the probability that $\Phi_{t+T} \geq \Phi_t/2$ is at most $1/2$. Therefore, using standard Chernoff bounds [10], we can show that in $T' = 8aT\lceil(\log\Phi_0 + \log n)\rceil$ steps, $\Phi_{T'} > 1$ with probability at most $O(1/(\Phi_0)^a + 1/n^a)$ for any constant $a > 0$. Since $\Phi_0 \leq n(1+\nu)^{\Delta+1}/\nu$, we have $\log\Phi_0 \leq (\Delta+1)(\nu) + \log n - \log\nu$. Therefore, for $T' = O(\Delta d/\alpha + d^2\log n/\alpha^2)$, we have $\Phi_t < 1$ with high probability which implies that after T' steps, $|S_{>2\log_{1+\alpha/(2d)} n}| = 0$ with high probability.

To establish balance in the number of tokens below the average, we use an averaging argument to show that after T' steps $|S_{<-2\log_{1+\alpha/(2d)} n}| \leq n/2$ with high probability, and then repeat the above arguments redefining the potential appropriately. This proves Theorem 3.1.

3.2 The multi-port model

In this section, we analyze the deterministic multi-port algorithm described in Section 1.1.

Theorem 3.2 *For an arbitrary network G with n nodes, degree d, edge expansion α, and initial imbalance Δ, the multi-port algorithm load balances to within $O((d^2\log n)/\alpha)$ tokens in $O(\Delta/\alpha)$ steps.* ■

The proof of Theorem 3.2 is similar to that of Theorem 3.1. For every integer i, let S_i be the set of nodes having between $\rho - d + 2id$ and $\rho + d - 1 + 2id$ tokens. Consider T steps of the multi-port algorithm. Without loss of generality, we assume that $|S_{>0}| \leq n/2$ holds before at least half of these steps. By Section 2, there exists an index j in $[1, 2\lceil\log_{1+\alpha/(2d)} n\rceil]$ that is good in at least $T/4$ steps.

We assign a potential of $(1+\nu)^h$ to every token at height $h \geq 2jd$, and a potential of zero to all other tokens, where $\nu = \alpha/cd^2$ for appropriately chosen c. Let Φ_t denote the potential of the network after step t of the algorithm. By proving claims analogous to Lemmas 3.1 and 3.3 we show that in every step t in which j is good, the potential drop is at least $\varepsilon\nu^2d^2\Phi_{t-1}$. Theorem 3.2 easily follows from this lower bound on the potential drop. The proofs of Theorems 3.1 and 3.2 can be modified to establish bounds in terms of μ instead of α.

4 Local load balancing can be expensive

In this section, we show that locally load balancing to within $2d$ tokens using the multi-port algorithm of [2] described in Section 1.1 can take $\Omega(n^{1/2})$ more time than globally load balancing to within $O(d\log n/\mu)$ tokens, for any n, d and μ. We extend this bound for the single-port algorithm presented in [15], i.e., the expected number of additional steps this algorithm may take to perform local (to within one token) rather than global balancing is $\Omega(dn^{1/2})$. Furthermore, we show that in the single-port case, we may be one step away from being locally balanced to within one token but have an expected running time of $\Omega(\mu n^{1/2})$ for reaching a locally balanced state. Finally, we prove an upper bound on the time to reach a locally balanced state from a globally balanced state.

We now define a graph G which will be the basis for G', the network used for showing the lower bounds on time for local load balancing. Let $G = (V, E)$ be a graph on n nodes, with node expansion μ and maximum degree d. Let $N(X) = \{y|(x, y) \in E, x \in X\}$, for any $X \subseteq V$. Let $v \in V$. Then $R_0 = \{v\}$ must have at least μ neighbors. Let $R_1 = N(R_0)$. Then $R_0 \cup R_1$ must have at least $\mu(1+\mu)$ neighbors not in $R_0 \cup R_1$. In general if we define recursively $R_i = N(R_{i-1}) - (R_{i-1} \cup R_{i-2})$, $i \geq 2$, we must have $|R_i| \geq \mu(1+\mu)^{i-1}$ if $|\cup_{j=0}^{i-1} R_j| \leq n/2$. Let k be the largest index such that $|\cup_{j=0}^{k-1} R_j| \leq n/2$. We know that $k = O(\log n/\log(1+\mu))$; here we assume wlog that $k = \log n/\log(1+\mu) = O(\log n/\mu)$, for $\mu \leq 1$. For the sake of simplicity, let $G - \cup_{i=0}^{k-1} R_j$ be isomorphic to $\cup_{i=0}^{k} R_j$. Let $L_k = R_k$ and L_i denote the corresponding copy of R_i, $i < k$, in $G - \cup_{i=0}^{k-1} R_j$. Let v' be the only node of L_0. Add a single edge e connecting v to v' and let $G' = G + e$. Note that the node expansion of G cannot be reduced by the addition of new edges and so the node expansion of G' is $\Omega(\mu)$.

We state all the results in this section in terms of the node expansion of the network, rather than in terms of its edge expansion. This is done for the sake of making our arguments more intuitive and clear. Nevertheless, one can construct a graph G as in the previous paragraph such that $\mu = \alpha/d$, where α is the edge expansion of G.

We present the two cases below according to the initial distribution of tokens over the nodes of G'. We use $w(v)$ to denote the number of tokens on node v. Again for simplicity, we assume k is even. We group the sets R_i's and L_i's into $\mathcal{L}$ and $\mathcal{R}$, groups of $k/2$ consecutive sets, and $\mathcal{M}$, a group of $k+1$ consecutive sets (note that we have $2k+1$ distinct sets). Let $\mathcal{L} = \{L_0, L_1, \ldots, L_{\frac{k}{2}-1}\}$, $\mathcal{R} = \{R_0, R_1, \ldots, R_{\frac{k}{2}-1}\}$ and $\mathcal{M} = \{L_{\frac{k}{2}}, L_{\frac{k}{2}+1}, \ldots, L_{k-1}, R_l(= L_k), R_{k-1}, \ldots, R_{\frac{k}{2}}\}$. Our choice for $\mathcal{L}, \mathcal{M}$ and $\mathcal{R}$ is such that the number of sets in $\mathcal{L}$ is roughly half the number of sets in $\mathcal{M}$. We also use the fact that the number of nodes in $\mathcal{L}$ is the same as the number of nodes in $\mathcal{R}$ for the following case.

The local algorithm may take $\Omega(n^{1/2})$ steps to locally balance G'. Suppose for every node z in $\mathcal{R}$, $w(z) = m + 1$ initially, where m is an integer such that $m \geq 2kd$. For all $z \in R_i$, $R_i \in \mathcal{M}$, let $w(z) = m - 2(i - \frac{k}{2})d$; for all $z \in L_i$, $L_i \in \mathcal{M}$, let $w(z) = m - 2(\frac{3k}{2} - i)d$. For all $z \in L_i$, $L_i \in \mathcal{L}$, let $w(z) = m - 2(i + \frac{k}{2} + 1)d$. Then w is globally balanced to within $O(\frac{d \log n}{\mu})$ tokens, but it is not locally balanced to within $2d$ tokens since $w(v) - w(v') > kd$ $(\geq 2d)$. See Figure 1.

There are at least $(1+\mu)^{\frac{k}{2}-1} = (1+\mu)^{\frac{\log n}{2\log(1+\mu)}-1} = \frac{\sqrt{n}}{(1+\mu)} \geq \frac{\sqrt{n}}{2}$ nodes in $\mathcal{R}$, as there are in $\mathcal{L}$. We claim that in order for G' to be locally balanced to within $2d$, we need to move from $\mathcal{R}$ to $\mathcal{L}$ a number of tokens at least equal to the number of nodes in $\mathcal{R}$ across edge e. Suppose we have $Q \geq \frac{\sqrt{n}}{2}$ nodes in $\mathcal{R}$. Since at most one token at a time can traverse e, it will take time $\Omega(Q) \geq \Omega(n^{1/2})$ to locally balance G' to within $2d$ tokens. It is easy to prove that no token moves from $\mathcal{M}$ to $\mathcal{R}$ in the first Q steps, what implies that no token moves from L_i to L_{i-1}, $L_i \in \mathcal{L}$, $i > 0$, durign these steps since every node in L_i had fewer tokens than any node in L_{i-1} initially. The proof of our claim proceeds as follows:

(i) We first show that $w(v') \leq m - (k+2)d + 1$ in the first Q steps. Suppose that $w(v') > m - (k+2)d + 1$ for the first time at some time step t. Then v' has at least 2 tokens more than it had initially. Since $w(z) \leq w(v')$, for all $z \in L_1$, at any time (we saw that no token ever moves from L_2 to L_1 and $w(z) \leq w(v')$ initially, for all $z \in L_1$) the (at least) 2 "extra" tokens on v' must have traversed e from v, one at a time, and v' must have not moved a token to any of its neighbors in L_1 at time $t-1$. Actually one token must have been sent through e at time $t' - 1 < t - 1$ and $w(v') = m - (k+2)d + 1$ at time t', and v' did not send a token to any node in L_1 at time $t' + 1$. This implies that v' had at most $2d$ tokens more than any node in L_1 at time t, which implies that any node in L_1 had at most $2d$ tokens more than any node in L_2 at time step $t' - 1$, etc. In general we must have that any node in L_i had at most $2d$ tokens more than any node in L_{i+1} at time step $t' - i$. Note that then at time t' the number of tokens on any node in L_i differs by at most $2d$ from the number of tokens on any node in L_{i+1}, for all $0 \leq i < k$. Hence at time t' each $z \in L_1$ has at least one more token than the initial value of $w(z)$. That is, we have at least Q "extra" tokens in $\mathcal{L}$. Since no token ever moves from $\mathcal{M}$ to $\mathcal{L}$, all those "extra" tokens must have come across edge e, taking at least Q total time. Thus $w(v') \leq m - (k+2)d + 1$ during the first Q steps.

(ii) We also show that we can only have: $w(v) \leq m - kd$ and $w(y) - w(x) \leq 2d$, $\forall x \in R_i, \forall y \in R_{i+1}, \forall R_i \in \mathcal{R}$ after Q steps. A counting argument on the number of tokens in $\mathcal{R}$ and the fact that no token is ever sent from $\mathcal{R}$ to $\mathcal{M}$ is sufficient to show this.

It follows from (i) and (ii) that either v sends a token to v' or $\mathcal{R}$ is not $2d$-locally balanced in any of the first Q steps, and so G' is not locally balanced before Q steps.

Note that the arguments above can be easily modified in order to hold for the single-port model (divide the number of tokens above height m by $2d$ on each node not in $\mathcal{R} \cup R_{\frac{k}{2}}$, consider differences of at most one token between "adjacent" R_i's or L_i's, and consider randomization in (i)), since the lower bound on the number of steps required to reach a locally balanced state is given only in terms of how many tokens need to traverse the edge e. Any edge is selected with probability $O(\frac{1}{d})$ at each iteration of the single-port algorithm. Thus the expected time for e being selected at some time step is

$\Omega(d)$. Hence we have that it will take $\Omega(dn^{1/2})$ expected time for G', if G' globally balanced to within $O(\frac{\log n}{\mu})$ tokens, to reach a locally balanced state to within 1 token in the single-port model.

The local algorithm may take one step or $\Omega(\mu n^{1/2})$ expected time to reach local balance. Here we consider the single-port model. Suppose we have the following distribution of tokens: for all $z \in R_{\frac{k}{2}-1}$, let $w(z) = m+1$ (where m is an integer such that $m \geq k$); for all $z \in R_i$, $i \leq \frac{k}{2} - 2$ (note that $R_i \in \mathcal{R}$), let $w(z) = m-(\frac{k}{2}-i-1)$; for all $z \in R_{\frac{k}{2}}$, let $w(z) = m-1$; for all $z \in R_i$, $i \geq \frac{k}{2}+1$ (note that $R_i \in \mathcal{M}$), let $w(z) = m-(i-\frac{k}{2})$; for all $z \in L_i$, $L_i \in \mathcal{M}$, let $w(z) = m - (\frac{3k}{2} - i)$. For all $z \in L_i$, $L_i \in \mathcal{L}$, let $w(z) = m-(\frac{k}{2}+i)$. Thus w is globally balanced to within $O(\frac{\log n}{\mu})$ tokens but it is not locally balanced to within 1 token, since $w(x) - w(y) > 1$, for any $x \in R_{\frac{k}{2}-1}$ and $y \in R_{\frac{k}{2}} \cup R_{\frac{k}{2}-2}$. See Figure 2.

The intuition for this case is that if all tokens move in the "right direction" initially, we reach a locally balanced state in a single time step. Otherwise, if a large number of tokens move in the "wrong direction" in the first step, it will take a long time to reach such a state. If every node in $R_{\frac{k}{2}-1}$ is matched with some node in $R_{\frac{k}{2}}$ (we can assume that every node in $R_{\frac{k}{2}-1}$ has a distinct neighbor in $R_{\frac{k}{2}}$), G' reaches local balance in a single time step. On the other hand, if we move tokens across a matching between the nodes in $R_{\frac{k}{2}-1}$ and $R_{\frac{k}{2}-2}$ then these tokens will clearly continue moving "down", and never move "up", The expected size of such a matching will be $\Omega(|R_{\frac{k}{2}-1}|/d)$ (each node in $R_{\frac{k}{2}-1}$ sees at least one node in $R_{\frac{k}{2}-2}$). Using a similar analysis (taking care of randomization) as in the previous case, we see that no token ever moves from $\mathcal{M}$ to $\mathcal{L}$ and that $w(v') \leq m - \frac{k}{2} + 1$ or $\mathcal{L}$ is not locally balanced, for each of the $\Omega(\sqrt{n})$ initial steps. Thus, once any of these tokens reaches v, it will eventually traverse e onto v'. (And they eventually will, since for $\mathcal{R}$ to be locally balanced either all such tokens are not in $\mathcal{R}$ anymore or v has one more token than it had initially.)

Hence eventually all the tokens that moved from $R_{\frac{k}{2}-1}$ to $R_{\frac{k}{2}-2}$ in the initial step will reach v'. Since $|R_{\frac{k}{2}-1}| \geq \frac{\mu n^{1/2}}{(1+\mu)^2}$ and $1 \leq (1+\mu)^2 \leq 4$, and since the expected time for edge e to be in the selected matching is $\Omega(d)$, the expected time for G' to reach local balance to within 1 token is $\Omega(\mu n^{1/2})$.

Convergence to a locally balanced state. We now prove that if the graph G is globally balanced to within Δ tokens, in $O(n\Delta^2/d)$ subsequent steps the multi-port algorithm locally balances to within $2d$ tokens. Define the potential Φ of the network as $\sum_{v \in V}(w(v) - \rho)^2$. If the network is globally balanced to within Δ tokens, then $\Phi = O(n\Delta^2)$. At any step, if there exists an edge (u, v) such that $|w(u) - w(v)| \geq 2d + 1$, then a token is transmitted along (u, v) resulting in a potential drop of at least d. Thus, within $O(n\Delta^2/d)$ steps the network is globally balanced. Similarly, we show that if the graph G is globally balanced to within Δ tokens, then the single-port algorithm locally balances to within 1 token in $O(nd\Delta^2)$ subsequent steps with high probability.

5 Extension to dynamic and asynchronous networks

In this section, we extend our results for the multi-port model of Section 3.2 to dynamic and asynchronous networks. We first prove that a variant of the local multi-port algorithm is optimal on dynamic synchronous networks in the same sense as for static synchronous networks. We then use a result of [2] that relates the dynamic synchronous and asynchronous models, to extend our results to asynchronous networks.

In the dynamic synchronous model, the edges of the network may fail or succeed dynamically. An edge $e \in E$ is *live* during step t if e can transmit a message in each direction during step t. We assume that at each step each node knows which of its adjacent edges are live. The local load balancing algorithm for static synchronous networks can be modified to work on dynamic synchronous networks. The algorithm presented here is essentially the same as in [2]. Every node u maintains an estimate $e^u(v)$ of the height of v for every neighbor v of u. (The value of $e^u(v)$ at the start of the algorithm is arbitrary.) In every step of the algorithm, which we call **DS**, each node u does the following:

(1) For each live neighbor v of u, if $w(u) - e^u(v) > 12d$, u sends a message consisting of $w(u)$ and a token, otherwise u sends a message consisting only of $w(u)$.

(1) For each message received from a live neighbor v, $e^u(v)$ is updated according to the message and if the message contains a token, $w(u)$ is increased by one.

For every integer i, we denote the set of nodes having between $\rho - 12d + 24id$ and $\rho + 12d - 1 + 24id$ tokens as S_i. Consider T steps of **DS**. We assume without loss of generality that $|S_{>0}| \leq n/2$ at the start of at least $T/2$ steps. There exists an index j in $[1, \lceil 2\log_{1+\alpha/(2d)} n \rceil]$ that is good in at least half the steps at the start of which $|S_{>0}| \leq n/2$. If index j is good at the start of step t, we call t a *good* step. For any token p, let $h_t(p)$ denote the height of p after step t, $t > 0$. For convenience, we denote the height of p at the start of **DS** by $h_0(p)$. Similarly, for $t \geq 0$, we define $h_t(u)$ for every node u and $e^u_t(v)$ for every edge (u, v). Let $\nu = \alpha/(cd^2)$, where $c > 0$ is a constant to be specified later. We define $\phi(x)$ to be $(1+\nu)^x$ if $x \geq 24jd - 11d$, and 0 otherwise. With every token at height h, we associate a potential of $\phi(h)$. Let Φ_t denote the total potential of the network after step

t.

We analyze **DS** by means of an amortized analysis over the steps of the algorithm. Let E_t be the set $\{(u,v) : (u,v)$ is live, $u \in S_{>j}$ and $h_{t-1}(u) - h_{t-1}(v) \geq 24d\}$. For every step t, we assign an amortized potential drop of

$$\hat{\Psi}_t = \frac{1}{50} \sum_{\substack{(u,v)\in E_t \\ h(u)>h(v)}} (\phi(h_{t-1}(u) - d) - \phi(h_{t-1}(v) + d)).$$

Lemma 5.1 *If the live edges of G have an edge expansion of α during every step of* **DS**, *then for every good step t we have $\hat{\Psi}_t \geq \Omega(\nu^2 d^2 \Phi_{t-1})$.* ■

The following crucial lemma relates the amortized potential drops to the actual potential drops.

Lemma 5.2 *For any initial load distribution and any step $t' > 0$, we have*

$$\sum_{t\leq t'} \hat{\Psi}_t \leq \left(\sum_{t\leq t'} \Psi_t\right) - 2\Phi_0 - 2m\phi(24jd).\ ■$$

In order to prove Lemma 5.2, we need to address two issues that may arise in the dynamic setting: (i) potential gains, i.e., when a token gains height, and (ii) when no actual potential drop occurs across an edge that joins nodes differing by at least $24d$ tokens. We show that for any of the above events to occur, "many" tokens should have lost height in previous steps. We use a part of this prior potential drop to account for (i) and (ii).

The main result follows from Lemmas 5.1 and 5.2. We first show that within $O(1/(\varepsilon\nu^2 d^2))$ steps, there is a step when the actual potential of the network either decreases by a factor of 2 or falls below a threshold value.

Lemma 5.3 *Let t be any integer such that at least $7/(\varepsilon\nu^2 d^2)$ of the first t steps are good. There exists $t' \leq t$ such that $\Phi_{t'} \leq \max\{\Phi_0/2, 2m\phi(24jd)\}$.* ■

Theorem 5.1 *For an arbitrary network G with n nodes, degree d and initial imbalance Δ, if the live edges at every step t of G have edge expansion α, then the dynamic synchronous multi-port algorithm load balances to within $O(d^2(\log n)/\alpha)$ tokens in $O(\Delta/\alpha)$ steps.* ■

A simple variant of **DS**, as suggested in [2], can be defined for asynchronous networks. As shown in [2], the analysis for the dynamic synchronous case can be used for asynchronous networks to yield the same time bounds. Hence, the multi-port local load balancing algorithm balances to within $O(d^2 \log n/\alpha)$ tokens in time $O(\Delta/\alpha)$ on aynchronous networks.

6 Tight bounds on off-line load balancing

In this section, we analyze the load balancing problem in the off-line setting for both single-port and multi-port models. We derive tight bounds on the minimum number of steps required to balance on arbitrary networks. We assume that the network is synchronous.

We first consider the network $G = (V, E)$ under the single-port model. For any subset X of V, let $\overline{X}$ denote $V \setminus X$, $m(X)$ denote the number of edges in a maximum matching between X and $\overline{X}$.

Lemma 6.1 *For any network $G = (V, E)$ with node expansion μ and any subset X of V, we have $m(X) \geq \mu \min\{|X|, |\overline{X}|\}/(1+\mu)$. Moreover, there exists a set X of size at most $(1+\mu)|V|/2$ such that $m(X) \leq \mu|X|/(1+\mu)$.* ■

Theorem 1 of [28] obtains tight bounds on the offline complexity of load balancing in terms of the function m. By invoking Lemma 6.1, we obtain

Lemma 6.2 ([28]) *Any network G with node expansion μ and initial imbalance Δ can be globally balanced to within 2 tokens in at most $\lceil \Delta(1+\mu)/\mu \rceil$ steps. Moreover, there exists a network G and an initial load distribution with imbalance Δ such that any algorithm takes at least $\lceil \Delta(1+\mu)/\mu \rceil$ steps to balance G to within one token.* ■

Lemma 6.2 implies that our local single-port algorithm is not optimal for all networks. However, there exists a class of graphs, e.g., constant-degree expanders, for which the local algorithm is optimal. A network for which the local algorithm is not optimal is the hypercube. The local algorithm balances in $\Omega(\Delta \log n)$ time, while there exists an $O(\Delta\sqrt{\log n})$ time load balancing algorithm for the hypercube [33].

The proof of Lemma 6.2 can be modified to establish the following result for the multi-port model.

Lemma 6.3 *Any network G with edge expansion α and initial imbalance Δ can be globally balanced to within d tokens in at most $\lceil \Delta/\alpha \rceil$ steps. Moreover, for every network G, there exists an initial load distribution with imbalance Δ such that any algorithm takes at least $\lceil \Delta/\alpha \rceil$ steps to completely balance G.* ■

Lemma 6.3 implies that the local multi-port algorithm is asymptotically optimal for *all* networks.

References

[1] Y. Afek, E. Gafni, and A. Rosen. The slide mechanism with applications in dynamic networks. In *Proceedings of the 11th ACM Symposium on Principles of Distributed Computing*, pages 35–46, August 1992.

[2] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 632–641, May 1993.

[3] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.

[4] J. Aspnes, M. Herlihy, and N. Shavit. Counting networks and multiprocessor co-ordination. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 348–358, May 1991.

[5] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multi-commodity flow. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 459–468, October 1993.

[6] B. Awerbuch and T. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 487–496, May 1994.

[7] B. Awerbuch, Y. Mansour, and N. Shavit. End-to-end communication with polynomial overhead. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 358–363, October 1989.

[8] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Chapter 7, Prentice–Hall, Englewood Cliffs, NJ, 1989.

[9] A. Broder, A. M. Frieze, E. Shamir, and E. Upfal. Near–perfect token distribution. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, pages 308–317, 1992.

[10] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[11] Y. C. Chow and W. Kohler. Models for dynamic load balancing in a heterogeneous multiple processor system. *IEEE Transactions on Computers*, C–28(5):57–68, November 1980.

[12] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 2(7):279–301, 1989.

[13] D. Eager, E. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, SE–12(5):662–675, 1986.

[14] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. Technical Report CMU-CS-95-131, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, March 1995.

[15] B. Ghosh and S. Muthukrishnan. Dynamic load balancing on parallel and distributed networks by random matchings. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 226–235, June 1994.

[16] B. Goldberg and P. Hudak. Implementing functional programs on a hypercube multiprocessor. In *Proceedings of the 4th Conference on Hypercubes, Concurrent Computers and Applications*, volume 1, pages 489–503, 1989.

[17] A. Heirich and S. Taylor. A parabolic theory of load balance. Research Report Caltech-CS-TR-93-25, Caltech Scalable Concurrent Computation Lab, Pasadena, CA, March 1993.

[18] K. T. Herley. A note on the token distribution problem. *Information Processing Letters*, 28:329–334, 1991.

[19] S. H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *Journal of Parallel and Distributed Computing*, 10:160–166, 1990.

[20] J. JáJá and K. W. Ryu. Load balancing and routing on the hypercube and related networks. *Journal of Parallel and Distributed Computing*, 14:431–435, 1992.

[21] M. R. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: the approximation of the permanent resolved. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 235–244, May 1988.

[22] R. Karp and Y. Zhang. A randomized parallel branch-and-bound procedure. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 290–300, May 1988.

[23] M. Klugerman and C. G. Plaxton. Small depth counting networks. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 417–428, May 1992.

[24] F. C. H. Lin and R. M. Keller. The gradient model load balancing method. *IEEE Transactions on Software Engineering*, 13(1):32–38, 1987.

[25] R. Lüling and B. Monien. Load balancing for distributed branch and bound algorithms. In *Proceedings of the 6th International Parallel Processing Symposium*, pages 543–549, March 1992.

[26] R. Lüling and B. Monien. A dynamic distributed load balancing algorithm with provable good performance. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 164–172, June 1993.

[27] R. Lüling, B. Monien, and F. Ramme. Load balancing in large networks: A comparative study. In *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing*, pages 686–689, December 1991.

[28] F. Meyer auf der Heide, B. Osterdiekhoff, and R. Wanka. Strongly adaptive token distribution. In *Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, pages 398–409, 1993.

[29] M. Mihail. Conductance and convergence of Markov chains - a combinatorial treatment of expanders. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 526–531, October 1989.

[30] L. M. Ni, C. Xu, and T. B. Gendreau. Distributed drafting algorithm for load balancing. *IEEE Transactions on Software Engineering*, SE–11(10):1153–1161, October 1985.

[31] D. Peleg and E. Upfal. The generalized packet routing problem. *Theoretical Computer Science*, 53:281–293, 1987.

[32] D. Peleg and E. Upfal. The token distribution problem. *SIAM Journal on Computing*, 18:229–243, 1989.

[33] C. G. Plaxton. Load balancing, selection and sorting on the hypercube. In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures*, pages 64–73, June 1989.

[34] J. Stankovic. Simulations of three adaptive, decentralized controlled, job scheduling algorithms. *Computer Networks*, 8:199–217, 1984.

[35] R. Subramanian and I. D. Scherson. An analysis of diffusive load balancing. In *Proceedings of the 1994 ACM Symposium on Parallel Algorithms and Architectures*, pages 220–225, June 1994.

[36] A. N. Tantawi and D. Towsley. Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32:445–465, April 1985.

[37] R. D. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience*, 3(5):457–481, 1991.

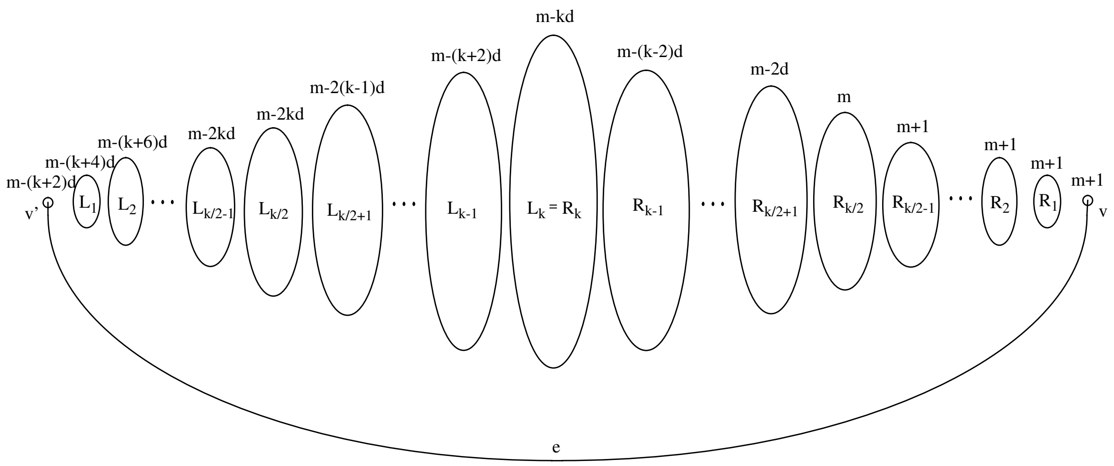

Figure 1: The initial tokens distribution on G' for the first case.

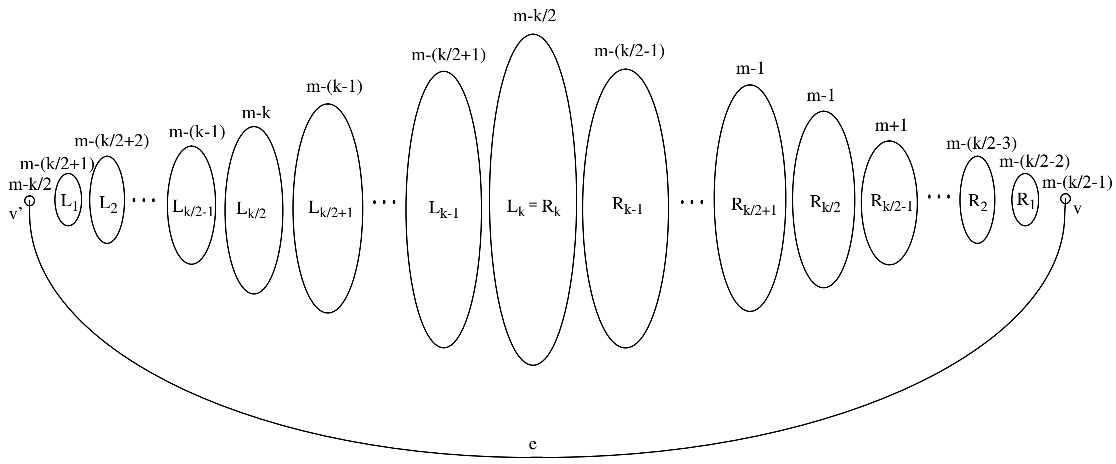

Figure 2: The initial tokens distribution on G' for the second case.