Distributed Selfish Load Balancing with Weights and Speeds

Clemens P.J. Adolphs[*]
University of British Columbia
6224 Agricultural Road
Vancouver, B.C., V6T 1Z1 Canada
cadolphs@phas.ubc.ca

Petra Berenbrink[†]
Simon Fraser University
8888 University Drive
Burnaby, B.C., V5A 1S6 Canada
petra@cs.sfu.ca

ABSTRACT

In this paper we consider neighborhood load balancing in the context of selfish clients. We assume that a network of n processors is given, with m tasks assigned to the processors. The processors may have different speeds and the tasks may have different weights. Every task is controlled by a selfish user. The objective of the user is to allocate his/her task to a processor with minimum load, where the load of a processor is defined as the weight of its tasks divided by its speed.

We investigate a concurrent probabilistic protocol which works in sequential rounds. In each round every task is allowed to query the load of one randomly chosen neighboring processor. If that load is smaller than the load of the task's current processor, the task will migrate to that processor with a suitably chosen probability. Using techniques from spectral graph theory we obtain upper bounds on the expected convergence time towards approximate and exact Nash equilibria that are significantly better than previous results for this protocol. We show results for uniform tasks on non-uniform processors and the general case where the tasks have different weights and the machines have speeds. To the best of our knowledge, these are the first results for this general setting.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Nonnumerical Algorithms and Problems—*Sequencing and Scheduling*

Keywords

Load balancing, reallocation, equilibrium, convergence

[*]The research was carried out during a visit to SFU.

[†]The author was funded by an NSERC Discovery Grant "Analysis of Randomized Algorithms"

General Terms

Algorithms, Theory

1. INTRODUCTION

In this paper we consider a variant of diffusion load balancing that is motivated by game theoretic concepts. In standard diffusion load balancing a graph is given whose vertices model *processors* and whose links can be used for communication and to exchange load items (called *tasks* in the following). In the beginning the tasks are arbitrarily distributed among the processors. The load balancing process works in sequential rounds. In every round every vertex is allowed to balance its load with all its neighbors by exchanging tasks. The goal is to balance the total system load globally, meaning to minimize the load difference between the nodes with minimum and maximum load. In contrast to standard diffusion load balancing, in *selfish load balancing* it is assumed that every task belongs to a selfish user. Instead of the nodes balancing the load with their neighbors, the users are now allowed to move their tasks over to a neighboring vertex. Of course, since the users are selfish they will never move items over to nodes having larger load. The goal is for such a system to converge as soon as possible to a *Nash Equilibrium*, where no user is able to decrease the load of its task by migrating it to a neighboring processor.

We revisit the selfish load balancing protocol introduced in [6]. In each round every user is allowed to check the load of one randomly chosen neighboring processor. In more detail, let us assume that the task of a user is assigned to vertex v. Then the user randomly chooses one of the neighbors of v. If the load of the chosen neighbor is smaller, the user will migrate its task with a suitably chosen probability to that processor. Note that if the probability is too large (for example, all users move their tasks over to the random neighbor as long as its load is smaller) the system would never be able to reach a balanced state. In this paper we choose the migration probability as a function of the load difference of the two involved processors. This means that no global information is necessary.

Similarly to [6] we consider several generalizations of the model. The tasks can have different *weights* which model, for example, the runtime of the tasks. If all tasks have the same weight we call them *uniform*. The processors have different *speeds*. Again, uniform speeds means that the speeds of all processors are the same. We show results for uniform tasks on processors with speeds. We also consider the general case with weighted tasks and machines with speeds. To our best knowledge these are the first results for this general setting.

We calculate upper bounds on the expected convergence time towards approximate and exact Nash equilibria that are significantly better than the previous results in [6]. For weighted tasks we consider a protocol that is different from the one given in [6]. In our new protocol, a player will move its tasks to the neighboring node only if the player with the task with maximum weight would do the same. A justification for this is that in many realistic settings, migration will only occur if the gain from doing so exceeds some threshold. Our analysis uses techniques from spectral graph theory similar to those used in [10].

Load Balancing is an important aspect of massively parallel computations as it must be ensured that resources are used to their full efficiency. Our load balancing strategies have the advantage that the users do not need any global load information of the system, they only have to know the load of a randomly chosen neighbor. Global information is often unavailable and global coordination usually very expensive and impractical. Another advantage is that the load items are moved to neighboring nodes only, which has the effect that load items that were initially on the same node tend to stay closely together. This is important if these tasks have to exchange information.

2. MODEL AND NEW RESULTS

The parallel system is represented by an undirected graph $G = (V, E)$ with vertices representing processors and edges representing direct communication links between them. The number of processors is n. The *degree* of a vertex $v \in V$ is $\deg(v)$. The maximum degree of the network is denoted by Δ, and for two nodes v and w the maximum of $\deg(v)$ and $\deg(w)$ is d_{vw}.

We define $s_i \in \mathbb{R}$ as the speed of processor i. We assume that the smallest speed is exactly one. $s_{\max}$ is the maximum speed and $s_{\min}$ is the minimum speed of any processor. If all speeds are equal the speeds are called *uniform*. If all s_i are integer multiples of a factor ϵ, i.e. for every speed s_i there exists an integer $n_i \in \mathbb{N}$ so that $s_i = n_i \cdot \epsilon$, we call them *ϵ-speeds*. We call ϵ the *granularity* of the speed distribution. Note that ϵ does not have to be an integer itself. Let $\mathcal{S} = \sum_{i \in V} s_i$ be the total capacity of the processors. We define the *speed vector* $\mathbf{s} = (s_1, \cdots s_n)^\top$ and the *speed matrix* $S \in \mathbb{N}^{n \times n}$ with $S_{ii} = s_i$ and $S_{ij} = 0$ for $i \neq j$.

The number of tasks in the system is m. Task ℓ has weight ω_ℓ with $\omega_\ell \in (0, 1]$. In the case of uniform tasks the size of all tasks is equal and we assume the weight of all tasks is one. Define $W = \sum_{i=1}^{m} \omega_i(x)$ as the total load of the system and $\bar{\omega} = W/m$ as the average load.

$\mathbf{X}^t$ with $\mathbf{X}^t = X_1^t, \ldots X_n^t$ models the system state. X_i^t is the set of tasks that are assigned to processor i at the end of round t. We define $\mathbf{W}^t = W_1^t, \ldots W_n^t$ as the weight vector with W_i^t as the weight of the tasks that are assigned to processor i at the end of round t. Define $\mathbf{L}^t = S^{-1}\mathbf{W}^t$ as the *load vector* with $L_i^t = W_i^t / s_i$ as the load of the tasks assigned to processor i. Note that $\mathbf{X}^t$, $\mathbf{W}^t$ and $\mathbf{L}^t$ are vectors of random variables. We will use x^t, w_i^t and ℓ^t to denote fixed values of these variables. For a fixed state x^t, $\mathbf{e} = \mathbf{w}^t - \bar{\mathbf{w}}$ is the deviation of the actual task vector from the average load vector. It is clear that $\sum_{i \in V} e_i(x) = 0$.

A state x^t of the system is called a *Nash equilibrium* (NE) if no single task can improve its perceived load by migrating to a neighboring node while all other tasks remain where they are, i.e., $\ell_i^t - \ell_j^t \leq 1/s_j$ for all $(i, j) \in E$. A state x^t of the system is called an *ε-approximate Nash equilibrium* (ε-approximate-NE) if no single task can improve its perceived load by a factor of $(1 - \varepsilon)$, i.e. $(1 - \varepsilon) \cdot \ell_i - \ell_j \leq 1/s_j$.

The Laplacian $L = L(G)$ is a matrix widely used in graph theory. It is the $n \times n$ matrix whose diagonal elements are $L_{ii} = \deg(i)$, and the off-diagonal elements are $L_{ij} = -1$ if $(i, j) \in E(G)$ and 0 otherwise. λ_2 denotes the second smallest eigenvalue of L. The generalized Laplacian LS^{-1} [10], where S is the diagonal matrix containing the speeds s_i, is used to analyze the behavior of migration in heterogeneous networks.

2.1 Uniform Tasks

In this section we review our results for uniform tasks on machines with unequal speeds. One round of the protocol is defined as follows. Every task selects a neighboring node uniformly at random. If migrating to that node would lower the load experienced by the task, the task migrates to that node with probability proportional to the load difference and the speeds of the processors. For a detailed description of the protocol see Algorithm 1 in Section 5.

The first result concerns convergence to an approximate Nash equilibrium. We use a potential function $\Psi_0(x)$ to measure how close to such an approximate Nash equilibrium the system is. This function will be defined in Sec. 5.

Theorem 2.1 *Let $\psi_c = 16n \cdot \Delta \cdot s_{\max}/\lambda_2$. Algorithm 1 reaches a state x with $\Psi_0(x) \leq 4 \cdot \psi_c$ in expected time*

$$\mathcal{O}\left(\ln\left(\frac{m}{n}\right) \cdot \frac{\Delta}{\lambda_2} \cdot s_{\max}^2\right).$$

If $m \geq 8 \cdot \delta \cdot s_{\max} \cdot \mathcal{S} \cdot n^2$ for some $\delta > 1$, this state is an ε-approximate-Nash equilibrium with $\varepsilon = 2/(1 + \delta)$.

From the state reached in Theorem 2.1, we then go on to prove the following bound for convergence to a Nash equilibrium.

Theorem 2.2 *Let $\psi_c = 16n \cdot \Delta \cdot s_{\max}/\lambda_2$ and assume* $\mathbf{s}$ *consists of ϵ-speeds. Let T be the first time step in which the system is in a Nash equilibrium. Then*

$$\mathbf{E}[T] = \mathcal{O}\left(n \cdot \frac{\Delta^2}{\lambda_2} \cdot \frac{s_{\max}^4}{\epsilon^2}\right).$$

These theorems are proven in Section 5. Our bound of Theorem 2.2 is smaller by at least a factor of $\Omega(\Delta \cdot \mathrm{diam}(G))$ than the bound found in [6] (see Observation 5.19).

We summarize the results for the most important graph classes in Table 1. The table gives an overview of asymptotic bounds on the expected runtime to reach an approximate or a exact Nash equilibrium. We omit the speeds from this table because they are independent of the graph structure and, therefore, the same for each column. We compare the results of this paper to the bounds obtained from [6]. These contain a factor $\mathcal{S} = \sum_i s_i$, which we replace with n, using $\mathcal{S} = \sum_i s_i \geq n$. The table shows that for the graph classes at hand, our new bounds are superior to those in [6].

2.2 Weighted Tasks

In Section 6, we study a slightly modified protocol (see 2) that allows tasks only to migrate to a neighboring processor

Table 1: Comparison with existing results

Graph	ε-approximate NE		Nash Equilibrium	
	This Paper	[6]	This Paper	[6]
Complete Graph	$\ln\left(\frac{m}{n}\right)$	$n^2 \cdot \ln(m)$	n^2	n^6
Ring, Path	$n^2 \cdot \ln\left(\frac{m}{n}\right)$	$n^3 \cdot \ln(m)$	n^3	n^5
Mesh, Torus	$n \cdot \ln\left(\frac{m}{n}\right)$	$n^2 \cdot \ln(m)$	n^2	n^4
Hypercube	$\ln(n) \cdot \ln\left(\frac{m}{n}\right)$	$n \cdot \ln^3(n) \cdot \ln(m)$	$n \cdot \ln^2(n)$	$n^3 \cdot \ln^5(n)$

if that would decrease their experienced load by a threshold depending on the speed of the processors. This protocol allows the tasks only to reach an approximate Nash Equilibrium.

The potential function used in the following theorem is introduced in Section 6 and is used similarly to the one in the unweighted case to measure progress towards an approximate Nash equilibrium.

Theorem 2.3 *Let $\psi_c = 16 \cdot n \cdot \Delta/\lambda_2 \cdot s_{\max}/s_{\min}^2$. Algorithm 2 reaches a state x with $\Psi_0(x) \leq 4 \cdot \psi_c$ in time*

$$\mathcal{O}\left(\ln\left(\frac{m}{n}\right) \cdot \frac{\Delta}{\lambda_2} \cdot \frac{s_{\max}^2}{s_{\min}}\right).$$

If $W > 8 \cdot \delta \cdot s_{\max}/s_{\min} \cdot \mathcal{S} \cdot n^2$ for some $\delta > 1$, this state is an $2/(1+\delta)$-approximate Nash equilibrium.

For uniform speeds the theorem gives a bound of

$$\mathcal{O}\left(\ln(m/n) \cdot \Delta/\lambda_2\right)$$

for the convergence time.

3. RELATED WORK

The work closest to ours is in [4, 5, 6]. [4] considers the case of identical machines in a complete graph. The authors introduce a protocol similar to ours that reaches a Nash Equilibria (NE) in time $\mathcal{O}(\log\log m + \text{poly}(n))$. An extension of this model to weighted tasks is studied in [5]. Their protocol converges to a NE in time polynomially in n, m, and the largest weight. In [6] the authors consider general graphs with processors with speed and weighted tasks. They use a potential function similar to ours for the analysis. The two main results of [6] for machines with speeds are presented in Table 1. [2] applies our techniques to discrete diffusive load balancing where each node sends the rounded expected flow of the randomized protocol to its neighbors.

Our paper relates to a general stream of works for selfish load balancing on a complete graph. There is a variety of issues that have been considered, starting with seminal papers on algorithms and dynamics to reach NE [12, 14]. More directly related are concurrent protocols for selfish load balancing in different contexts that allow convergence results similar to ours. Whereas some papers consider protocols that use some form of global information [13] or coordinated migration [18], others consider infinitesimal or splittable tasks [17, 3] or work without rationality assumptions [16, 1]. The machine models in these cases range from identical, uniformly related (linear with speeds) to unrelated machines. The latter also contains the case when there are access restrictions of certain agents to certain machines. For an overview of work on selfish load balancing see, e.g., [25].

Our protocol is also related to a vast amount of literature on (non-selfish) load balancing over networks, where results usually concern the case of identical machines and unweighted tasks. In expectation, our protocols mimic continuous diffusion, which has been studied initially in [9, 7] and later, e.g., in [23]. This work established the connection between convergence, discrepancy, and eigenvalues of graph matrices. Closer to our paper are discrete diffusion processes – prominently studied in [24], where the authors introduce a general technique to bound the load deviations between an idealized and the actual processes. Recently, randomized extensions of the algorithm in [24] have been considered, e.g., [11, 19].

4. SPECTRAL GRAPH THEORY

In this section, we will briefly collect some important results of spectral graph theory that we use in our proofs. For an excellent introduction, we recommend the book by Fan Chung [8]. Many important results are collected in an overview article by Mohar [22]. We omit a discussion of the most basic properties and refer to the extensive literature.

Results in this section are, unless indicated otherwise, taken from these sources. Let us begin by defining the matrix we are interested in.

Definition 4.1 Let $G = (V, E)$ be an undirected graph with vertices $V = \{1, \ldots n\}$ and edges E.

The *Laplacian* $L(G)$ of G is defined as

$$L(G) \in \mathbb{N}^{n \times n} \qquad L(G)_{ij} = \begin{cases} \deg(i) & i = j \\ -1 & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The second-smallest eigenvalue λ_2 is closely related to the connectivity properties of G. A first, albeit weak, result is the preceding lemma. A stronger result with a corollary useful for simple estimates is given in the next lemma.

Lemma 4.2 ([21]) *Let λ_2 be the second-smallest eigenvalue of the unweighted Laplacian of a graph G. Let* $\text{diam}(G)$ *be the diameter of graph G. Then*

$$\text{diam}(G) \geq \frac{4}{n \cdot \lambda_2}.$$

Corollary 4.3 *Using* $\text{diam}(G) \leq n$, *we get* $\lambda_2 \geq \dfrac{4}{n^2}$

Lemma 4.4 *This is another useful result by Fiedler [15]. Let λ_2 be the second-smallest eigenvalue of $L(G)$. Then,*

$$\lambda_2 \leq \frac{n}{n-1} \cdot \min\{\deg(i), i \in V\}.$$

For Δ the maximum degree of graph G, it immediately follows

$$\lambda_2 \leq \frac{n}{n-1} \cdot \Delta.$$

A stronger relationship between λ_2 and the network's connectivity properties is provided via the graph's Cheeger constant.

Definition 4.5 Let $G = (V, E)$ be a graph and $S \subset V$ a subset of the nodes. The *boundary* δS of S is defined as the set of edges having exactly one endpoint in S, i.e.,

$$\delta S = \{(i, j) \in E \mid i \in S, j \in V \setminus S\}.$$

Definition 4.6 Let $G = (V, E)$ be a graph. The *isoperimetric number* $i(G)$ of G is defined as

$$i(G) = \min_{\substack{S \subset V \\ |S| \leq |V|/2}} \frac{|\delta S|}{|S|}.$$

It is also called *Cheeger constant* of the graph.

The isoperimetric number of a graph is a measure of how well any subset of the graph is connected to the rest of the graph. Graphs with a high Cheeger constant are also called *expanders*. The following was proven by Mohar.

Lemma 4.7 ([20]) *Let λ_2 be the second-smallest eigenvalue of $L(G)$, and let $i(G)$ be the isoperimetric number of G. Then,*

$$\frac{i^2(G)}{2\Delta} \leq \lambda_2 \leq 2i(G).$$

4.1 Generalized Laplacian Analysis

Recall the speed-matrix S from the introduction. Instead of analyzing the Laplacian L, we are interested in the *generalized Laplacian*, defined as LS^{-1}. This definition is also used by Elsässer in [10] in the analysis of continuous diffusive load balancing in heterogeneous networks. In this reference, the authors prove a variety of results for the generalized Laplacian, which we restate here in a slightly different language.

It turns out that in the discussion of the properties of this generalized Laplacian, many results carry over from the analysis of the normal Laplacian. The similarity is made manifest by the introduction of a *generalized dot-product.*

Definition 4.8 For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we define the *generalized dot-product* with respect to S as

$$\langle \mathbf{x}, \mathbf{y} \rangle_S := \mathbf{x}^T S^{-1} \mathbf{y} = \sum_{i \in V} \frac{x_i \cdot y_i}{s_i}$$

Lemma 4.9 *The vector space $\mathbb{R}^n$ with $\langle \cdot, \cdot \rangle_S$ forms an* inner product space.

Remark 4.10 The fact that $\langle \cdot, \cdot \rangle_S$ is an *inner product* allows us to directly apply many results of linear algebra to it. For example, two vectors $\mathbf{x}$ and $\mathbf{y}$ are called orthogonal to each other, $\mathbf{x} \perp \mathbf{y}$, if $\mathbf{x} \cdot \mathbf{y} = 0$. Analogously, we call $\mathbf{x}$ and $\mathbf{y}$ orthogonal *with respect to S* if $\langle \mathbf{x}, \mathbf{y} \rangle_S = 0$.

Let us now collect some of the properties of LS^{-1}. These properties have also been used in [10]. We restate them here using the notation of the generalized dot product.

Lemma 4.11 *(Compare Lemma 1 in [10]) Let L be the Laplacian of a graph, and let S be the speed-matrix, $S = \operatorname{diag}(s_1, \cdots, s_n)$. Then the following holds true for the generalized Laplacian LS^{-1}.*

(1) The speed-vector $\mathbf{s} = (s_1, \ldots, s_n)^\top$ is (right-)eigenvector to LS^{-1} with eigenvalue 0.

(2) LS^{-1} is not symmetric any more. It is, however, still positive semi-definite.

(3) Since LS^{-1} is not symmetric, we have to distinguish left- and right-eigenvectors. Similar to the spectral theorem of linear algebra, we can find a basis of right-eigenvectors of LS^{-1} that are orthogonal with respect to S.

For arbitrary vectors, we know that $\langle \mathbf{x}, LS^{-1}\mathbf{x} \rangle_S \geq 0$ since $S^{-1}LS^{-1}$ is positive semi-definite. The next lemma bounds the generalized dot product of certain vectors with the Laplacian with the second smallest right-eigenvector of it. A similar version can also be found in [10, Section 3].

Lemma 4.12 *Let λ_2 be the second-smallest right eigenvalue of the generalized Laplacian, LS^{-1}. Let $\mathbf{e}$ be a vector that is orthogonal to the speed vector with respect to S, i.e. $\langle \mathbf{e}, \mathbf{s} \rangle_S = 0$. Then*

$$\langle \mathbf{e}, LS^{-1}\mathbf{e} \rangle_S \geq \lambda_2 \langle \mathbf{e}, \mathbf{e} \rangle_S.$$

The next technical lemma is needed to relate the spectra of L and LS^{-1}. We require this relation because most of the useful results and bounds for λ_2 apply to the normal Laplacian only.

Lemma 4.13 *Let μ_i denote the eigenvalues of LS^{-1} in ascending order and let λ_i denote the eigenvalues of L in ascending order. Finally, let s_i denote the speeds in descending order. Then*

$$\mu_{i+j-1} \geq \frac{\lambda_i}{s_j} \qquad 0 \leq i, j \leq n, \quad 0 \leq i + j - 1 \leq n \qquad (1)$$

$$\mu_{i+j-n} \leq \frac{\lambda_i}{s_j} \qquad 0 \leq i, j \leq n, \quad 0 \leq i + j - n \leq n. \qquad (2)$$

Corollary 4.14 *Let μ_2 denote the second smallest right eigenvalue of LS^{-1} and let λ_2 denote the second smallest eigenvalue of L. Let $s_{\max} = s_1$ be the largest speed and $s_{\min} = s_n$ the smallest speed. Then*

$$\frac{\lambda_2}{s_{\max}} \leq \mu_2 \leq \frac{\lambda_2}{s_{\min}}.$$

PROOF. Let $i = 2, j = 1$ in (1) and $i = 2, j = n$ in (2).

5. UNIFORM TASKS WITH SPEEDS

The pseudo-code of a single step of our protocol is given in Algorithm 1. Recall that d_{ij} is given by $\max\{\deg(i), \deg(j)\}$ and α is defined as $4s_{\max}$.

For the analysis we use the same (standard) *potential function* as the one used in [6]. For $r = 0, 1$, define

$$\Phi_r(x) := \sum_{i \in V} w_i(x) \cdot (w_i(x) + r)/s_i$$

The potential Φ_0 is minimized for the average task vector, $\bar{\mathbf{w}}$. We define the *normalized potential* as

$$\Psi_0(x) = \Phi_0(x) - m^2/\mathcal{S} = \sum_{i \in V} (w_i - \bar{w})^2 \; /s_i = \sum_{i \in V} e_i^2 \; /s_i.$$

We define $\Delta\Phi_r(X^t) := \Phi_r(X^{t-1}) - \Phi_r(X^t)$ as the potential drop in step t. $\Delta\Psi_0(X^t)$ is defined analogously. From the definition of Ψ_0, it becomes clear that $\Delta\Psi_0(X^t) = \Delta\Phi_0(X^t)$.

Algorithm 1: Distributed Selfish Load Balancing

```
begin
    foreach task $\ell$ in parallel do
        Let $i = i(l)$ be the current machine of task $l$
        Choose a neighboring machine $j$ uniformly at random
        if $\ell_i - \ell_j > 1/s_j$ then
            Move task $\ell$ from node $i$ to node $j$ with probability
                $p_{ij} := \frac{\deg(i)}{d_{i,j}} \cdot \frac{\ell_i - \ell_j}{\alpha \cdot (1/s_i + 1/s_j) \cdot w_i}$
        end
    end
end
```

The *maximum load difference* is defined as

$$L_\Delta(x) = \max_{i \in V} |w_i(x)/s_i - m/\mathcal{S}| = \max_{i \in V} |e_i/s_i|.$$

For a given state x, we define $f_{ij}(x)$ as the *expected flow* over edge (i,j), with

$$f_{ij}(x) = (\ell_i(x) - \ell_j(x))/(\alpha \cdot d_{ij} \cdot (1/s_i + 1/s_j))$$

if $\ell_i(x) - \ell_j(x) > 1/s_j$, and 0 otherwise. As an auxiliary quantity, we define

$$\Lambda^r_{ij}(x) := (2\alpha - 2) \cdot d_{ij} \cdot (1/s_i + 1/s_j) \cdot f_{ij}(x) + r/s_i - r/s_j.$$

We define the set of *non-Nash edges* as $\tilde{E}(x) := \{(i,j) \in E : f_{ij}(x) > 0\}$. This is the set of edges for which tasks have an incentive to migrate. Edges with $f_{ij}(x) = 0$ are called *Nash edges* or *balanced edges.*

Our improved bound builds upon results in [6]. In that paper, the randomized process is analyzed by lower-bounding the expected potential drop. Up to their Lemma 3.3 (restated below as Lemma 5.1) our analysis follows the steps of [6]. They then go on to link the potential drop to L_Δ. The authors show that as long as the potential Ψ_0 is large enough an edge must exist over which the load difference is at least $L_\Delta/\operatorname{diam}(G)$. Then they show an expected multiplicative potential drop, which is used to prove convergence to an approximate Nash equilibrium. Subsequently, a constant drop in Φ_1 is used to finally converge to a Nash equilibrium.

Lemma 5.1 (Lemma 3.3 in [6]) *For any step t and any state x,*

$$\mathbf{E}[\Delta\Phi_r(X^t) | X^{t-1} = x] \geq \sum_{(i,j) \in \tilde{E}(x)} f_{ij}(x) \cdot \left(\Lambda^r_{ij}(x) - \frac{1}{s_i} - \frac{1}{s_j} \right).$$

Here we prove a stronger bound by linking the potential drop to the Laplacian matrix. This results in a larger factor for the multiplicative drop, which then allows us to prove faster convergence to an approximate Nash equilibrium. In addition, the value of the potential Φ_1 in that approximate equilibrium is smaller than the value reached in [6]. This results in faster convergence to a Nash equilibrium using the constant drop in Φ_1. In summary, the main novelty of our analysis lies in significantly improving the bound on the expected multiplicative potential drop in Ψ_0 and the bound on Φ_1 after an approximate Nash equilibrium is reached. Other intermediate steps leading to and from these results are very similar to the steps in [6]. However, our results are also applicable to non-integer speeds. We note here that all proofs not appearing in the main text were omitted due to space limitations.

5.1 Approximate Nash Equilibrium

Lemma 5.2 *Under the condition that the system is in state x, the expected drop in the potentials Φ_0 and Ψ_0 is bounded by*

$$\mathbf{E}[\Delta\Psi_0(X^{t+1}) | X^t = x] \geq \sum_{(i,j) \in E} \left[\frac{\left(1 - \frac{2}{\alpha}\right) \cdot (\ell_i(x) - \ell_j(x))^2}{\alpha \cdot d_{i,j} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)} \right] - \frac{n}{\alpha}.$$

Proof. For brevity, we omit the argument x from all quantities. Note that we can look at the drop of either Φ_0 or Ψ_0. Substituting the particular forms of f_{ij} and Λ^0_{ij} into the bound provided by Lemma 5.1, we arrive at

$$\mathbf{E}[\Delta\Psi_0(X^{t+1}) | X^t = x] \geq \sum_{(i,j) \in \tilde{E}} \left[\frac{\left(2 - \frac{2}{\alpha}\right) \cdot (\ell_i - \ell_j)^2}{\alpha \cdot d_{i,j} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)} - \frac{(\ell_i - \ell_j)}{\alpha \cdot d_{i,j}} \right]. \quad (*)$$

We define subsets $\tilde{E}$, $\tilde{E}_1$ and $\tilde{E}_2$ of E,

$$\tilde{E} = \left\{ (i,j) \in E \mid \ell_i - \ell_j \geq \frac{1}{s_j} \right\}$$
$$\tilde{E}_1 = \left\{ (i,j) \in \tilde{E} \mid \ell_i - \ell_j \geq \frac{1}{s_i} + \frac{1}{s_j} \right\}$$
$$\tilde{E}_2 = \tilde{E} \setminus \tilde{E}_1.$$

Note that $\tilde{E} = \tilde{E}_1 \cup \tilde{E}_2$ and $\tilde{E}_1 \cap \tilde{E}_2 = \emptyset$. Thus, we can split the sum in $(*)$ into a sum over $\tilde{E}_1$ and a sum over $\tilde{E}_2$. We will now bound these sums individually.

Let $(i,j) \in \tilde{E}_1$ be an edge in $\tilde{E}_1$ so that $\ell_i \geq \ell_j$. Then the definition of $\tilde{E}_1$ and the non-negativity of $\ell_i - \ell_j$ allows us to deduce

$$\ell_i - \ell_j \geq \frac{1}{s_i} + \frac{1}{s_j} \Leftrightarrow \frac{1}{\frac{1}{s_i} + \frac{1}{s_j}} \cdot (\ell_i - \ell_j)^2 \geq \ell_i - \ell_j.$$

This allows us to bound

$$\sum_{(i,j) \in \tilde{E}_1} \left[\frac{\left(2 - \frac{2}{\alpha}\right) \cdot (\ell_i - \ell_j)^2}{\alpha \cdot d_{i,j} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)} - \frac{(\ell_i - \ell_j)}{\alpha \cdot d_{i,j}} \right] \geq \sum_{(i,j) \in \tilde{E}_1} \frac{\left(1 - \frac{2}{\alpha}\right) \cdot (\ell_i - \ell_j)^2}{\alpha \cdot d_{i,j} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)}. \quad (*)$$

Next, we turn to $\tilde{E}_2$ and bound

$$\sum_{(i,j)\in\tilde{E}_2}\left[\frac{\left(2-\frac{2}{\alpha}\right)\cdot(\ell_i-\ell_j)^2}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}-\frac{(\ell_i-\ell_j)}{\alpha\cdot d_{i,j}}\right].$$

The sum is over two terms, a positive and a negative one. For the first, positive term, we simply bound

$$\sum_{(i,j)\in\tilde{E}_2}\left[\frac{\left(2-\frac{2}{\alpha}\right)\cdot(\ell_i-\ell_j)^2}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}\right] \geq \sum_{(i,j)\in\tilde{E}_2}\left[\frac{\left(1-\frac{2}{\alpha}\right)\cdot(\ell_i-\ell_j)^2}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}\right]. \qquad (**)$$

For the edges in $\tilde{E}_2$, we have $\ell_i-\ell_j<1/s_i+1/s_j$. This allows us to bound the second, negative term via

$$\sum_{(i,j)\in\tilde{E}_2}\frac{\ell_i-\ell_j}{\alpha\cdot d_{i,j}}\leq\frac{1}{\alpha}\cdot\sum_{(i,j)\in\tilde{E}_2}\frac{1}{d_{i,j}}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right) \qquad (***)$$

Combining $(*)$, $(**)$ and $(***)$ yields

$$\mathbf{E}[\Delta\Psi_0(X^{t+1})|X^t=x]\geq \sum_{(i,j)\in\tilde{E}}\left[\frac{\left(2-\frac{2}{\alpha}\right)\cdot(\ell_i-\ell_j)^2}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}-\frac{(\ell_i-\ell_j)}{\alpha\cdot d_{i,j}}\right]\geq \sum_{(i,j)\in\tilde{E}}\frac{\left(1-\frac{2}{\alpha}\right)\cdot(\ell_i-\ell_j)^2}{\alpha\cdot d_{ij}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}-\sum_{(i,j)\in\tilde{E}_2}\frac{1}{\alpha\cdot d_{ij}}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right). \qquad (\dagger)$$

In the next step, we rewrite the sum over $\tilde{E}$ in $(\dagger)$ to a sum over all edges E, using $\tilde{E}=E\setminus(E\setminus\tilde{E})$. It generally holds for any terms $X_{(i,j)}$ that

$$\sum_{(i,j)\in\tilde{E}}X_{(i,j)}=\sum_{(i,j)\in E}X_{(i,j)}-\sum_{(i,j)\in E\setminus\tilde{E}}X_{(i,j)}.$$

We will apply this to $(\dagger)$. In the following, we therefore prove an upper bound on the sum over $E\setminus\tilde{E}$. Without loss of generality, let the nodes i and j of an edge be ordered such that $\ell_i\geq\ell_j$. For edges not in $\tilde{E}$, we have, by definition, $0\leq\ell_i-\ell_j\leq\frac{1}{s_j}$, so this part can be bound by

$$\begin{aligned}\sum_{(i,j)\in E\setminus\tilde{E}}\frac{\left(1-\frac{2}{\alpha}\right)}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}\cdot(\ell_i-\ell_j)^2 &\leq \sum_{(i,j)\in E\setminus\tilde{E}}\frac{1}{\alpha\cdot d_{ij}}\cdot\frac{s_i}{s_i+s_j}\cdot\frac{1}{s_j}\\ &\leq \sum_{(i,j)\in E\setminus\tilde{E}}\frac{1}{\alpha\cdot d_{ij}}\cdot\left(1-\frac{s_j}{s_i+s_j}\right)\cdot\frac{1}{s_j}\\ &= \sum_{(i,j)\in E\setminus\tilde{E}}\frac{1}{\alpha\cdot d_{i,j}}\cdot\left(\frac{1}{s_j}-\frac{1}{s_i+s_j}\right)\\ &\leq \sum_{(i,j)\in E\setminus\tilde{E}}\frac{1}{\alpha\cdot d_{i,j}}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right).\end{aligned}$$

This bound has the same form as the bound in $(***)$, only that it goes over $E\setminus\tilde{E}$ instead of $\tilde{E}_2$. These two sets are disjunct, since $\tilde{E}_2\subset\tilde{E}$. Therefore, we can combine the two sums into a single sum over $\tilde{E}_2\cup(E\setminus\tilde{E})=E\setminus\tilde{E}_1$. We then obtain from the following bound from $(\dagger)$.

$$\mathbf{E}[\Delta\Psi_0(X^{t+1})|X^t=x]\geq \sum_{(i,j)\in E}\left[\frac{\left(1-\frac{2}{\alpha}\right)\cdot(\ell_i-\ell_j)^2}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}\right]-\sum_{(i,j)\in E\setminus\tilde{E}_1}\frac{1}{\alpha\cdot d_{ij}}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right). \qquad (\dagger\dagger)$$

The first term of this bound already has the desired form. We will now bound the second term. Since it is negative, we have to upper bound the sum itself. First, note that $E\setminus\tilde{E}_1$ is a subset of E. As the term inside the sum is non-negative, we can write

$$\sum_{(i,j)\in E\setminus\tilde{E}_1}\frac{1}{\alpha\cdot d_{ij}}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)\leq\frac{1}{\alpha}\cdot\sum_{(i,j)\in E}\left[\frac{1}{d_{ij}\cdot s_i}+\frac{1}{d_{ij}\cdot s_j}\right].$$

Recall that d_{ij} is defined as $\max\{\deg(i),\deg(j)\}$, so we can bound

$$\begin{aligned}\frac{1}{\alpha}\cdot\sum_{(i,j)\in E}\left[\frac{1}{d_{ij}\cdot s_i}+\frac{1}{d_{ij}\cdot s_j}\right] &\leq \frac{1}{\alpha}\cdot\sum_{(i,j)\in E}\left[\frac{1}{\deg(i)\cdot s_i}+\frac{1}{\deg(j)\cdot s_j}\right]\\ &= \frac{1}{\alpha}\cdot\sum_{i\in V}\sum_{j\in\mathrm{Adj}(i)}\frac{1}{\deg(i)\cdot s_i}\\ &= \frac{1}{\alpha}\cdot\sum_{i\in V}\frac{1}{s_i}\leq\frac{n}{\alpha}.\end{aligned}$$

Inserting this bound into $(\dagger\dagger)$ yields the result.

We now use spectral graph theory to prove the following bound.

Lemma 5.3 *Let L be the Laplacian of the network. Let λ_2 be its second smallest eigenvalue. Then*

$$\mathbf{E}[\Delta\Psi_0(X^{t+1})|X^t=x]\geq\frac{\lambda_2}{16\Delta}\cdot\frac{1}{s_{\max}^2}\cdot\Psi_0(x)-\frac{n}{4\cdot s_{\max}}.$$

Proof. We start from the bound obtained in Lemma 5.2. In the course of this proof, we will use Lemma 4 4.12 for the task deviation vector $\mathbf{e}$. In order to use this lemma, we have to show that $\mathbf{e}$ satisfies the lemma's condition, i.e., $\langle\mathbf{e},\mathbf{s}\rangle_S=0$. This follows via

$$\langle\mathbf{e},\mathbf{s}\rangle_S=\sum_{i\in V}\frac{e_i\cdot s_i}{s_i}=\sum_{i\in V}e_i=0.$$

We can now begin with the main proof.

$$\begin{aligned}
&\mathbf{E}[\Delta\Psi_0(X^{t+1})|X^t = x] \\
&\quad \geq \sum_{(i,j)\in E}\left[\frac{\left(1-\frac{2}{\alpha}\right)\cdot(\ell_i(x)-\ell_j(x))^2}{\alpha\cdot d_{i,j}\cdot\left(\frac{1}{s_i}+\frac{1}{s_j}\right)}\right] - \frac{n}{\alpha} \\
&\quad \geq \frac{\frac{1}{2}}{4\cdot s_{\max}\cdot\Delta\cdot 2}\cdot\sum_{(i,j)\in E}\left(\ell_i - \frac{m}{\mathcal{S}} + \frac{m}{\mathcal{S}} - \ell_j\right)^2 \\
&\qquad\qquad - \frac{n}{4\cdot s_{\max}} \\
&\quad \geq \frac{1}{16\Delta}\cdot\frac{1}{s_{\max}}\cdot\sum_{(i,j)\in E}\left(\frac{e_i}{s_i}-\frac{e_j}{s_j}\right)^2 - \frac{n}{4\cdot s_{\max}} \\
(4.11) &= \frac{1}{16\Delta}\cdot\frac{1}{s_{\max}}\cdot\left(S^{-1}\mathbf{e}\right)^\top L\left(S^{-1}\mathbf{e}\right) - \frac{n}{4\cdot s_{\max}} \\
&= \frac{1}{16\Delta}\cdot\frac{1}{s_{\max}}\cdot\langle\mathbf{e}, LS^{-1}\mathbf{e}\rangle_S - \frac{n}{4\cdot s_{\max}} \\
(4.12,\ 4.14) &\geq \frac{1}{16\Delta}\cdot\frac{1}{s_{\max}}\cdot\frac{\lambda_2}{s_{\max}}\langle\mathbf{e},\mathbf{e}\rangle_S - \frac{n}{4\cdot s_{\max}} \\
&= \frac{1}{16\Delta}\cdot\frac{1}{s_{\max}}\cdot\frac{\lambda_2}{s_{\max}}\cdot\Psi_0 - \frac{n}{4\cdot s_{\max}}
\end{aligned}$$

It can be shown that as long as the expected value of the potential is sufficiently large, we can rewrite the potential drop as a multiplicative drop. Let λ_2 be the second smallest eigenvalue of the Laplacian $L(G)$ of the network. We define the *critical value* ψ_c as $\psi_c = 8\cdot n\cdot\Delta\cdot s_{\max}/\lambda_2$.

Lemma 5.4 *Let γ be defined such that $1/\gamma = \lambda_2/(32\Delta\cdot s_{\max}^2)$. Let t be a time step for which the expected value of the potential satisfies $\mathbf{E}[\Psi_0(X^t)] \geq \psi_c$. Then, the expected potential in time step $t+1$ is bounded by $\mathbf{E}[\Psi_0(X^{t+1})] \leq (1-1/\gamma)\cdot\mathbf{E}[\Psi_0(X^t)]$.*

This immediately allows us to prove the following.

Lemma 5.5 *For a given time step T, there either is a $t < T$ so that $\mathbf{E}[\Psi_0(X^t)] \leq \psi_c$, or $\mathbf{E}[\Psi_0(X^T)] \leq (1-1/\gamma)^T\cdot\mathbf{E}[\Psi_0(X^0)]$.*

As long as $\mathbf{E}[\Psi_0(X^t)] > \psi_c$ holds, the expected potential drops by a constant factor. This and several of the following results have the same form as several intermediate results in the proof of [6, Lemma 3.6], but our factor γ and the condition on $\mathbf{E}[\Psi_0(X^t)]$ are different. We can now derive a bound on the time it takes until $\mathbf{E}[\Psi_0(X^t)]$ is small.

Lemma 5.6 *Let $T = 2\gamma\cdot\ln(m/n)$. Then it holds that (1) there is a $t \leq T$ such that $\mathbf{E}[\Psi_0(X^T)] \leq \psi_c$ and (2) there is a $t \leq T$ such that the probability that $\Psi_0(X^t) \leq 4\cdot\psi_c$ is at least $\mathbf{Pr}[\Psi_0(X^t) \leq 4\cdot\psi_c] \geq 3/4$.*

Next, we show that states with $\Psi_0(x) \leq 4\cdot\psi_c$ are indeed ε-approximate Nash equilibria if the number of tasks exceeds a certain threshold. This requires one further observation.

Observation 5.7 For any state x, we have

$$L_\Delta(x)^2 \leq \Psi_0(x) \leq \mathcal{S}\cdot L_\Delta(x)^2.$$

Lemma 5.8 *Let $m \geq 8\cdot\delta\cdot n^2\cdot\mathcal{S}\cdot s_{\max}$ for some $\delta > 1$. Then a state x with $\Psi_0(x) \leq 4\cdot\psi_c$ is a $2/(1+\delta)$-approximate Nash equilibrium.*

Remark 5.9 If m is small, it still holds that we reach a state x with $\Psi_0(x) \leq 4\cdot\psi_c$, which is all we need to prove convergence to an exact Nash equilibrium in the next section. It is just that this intermediate state is then not an ε-approximate-Nash equilibrium.

We can now prove the main theorem.

Proof (Theorem 2.1). Lemma 5.8 ensures that after T steps the probability for *not* having reached a state x with $\Psi_0(x) \leq 4\cdot\psi_c$ is at most $1/4$. Hence, the expected number of times we have to repeat T steps is less than $1 + 1/4 + 1/4^2 + \cdots = 1/(1-1/4) < 2$. The expected time needed to reach such a state is therefore at most $2\cdot T$ with T from Lemma 5.6.

If we let the algorithm iterate until a state x with $\Psi_0(x) \leq 4\cdot\psi_c$ is obtained, Theorem 2.1 bounds the *expected* number of time steps we have to perform. However, by repeating a sufficient number of blocks with T steps, we can obtain arbitrarily high probability.

Corollary 5.10 *After $c\cdot\log_4 n$ many blocks of size T, a state with $\Psi_0(x) \leq 4\cdot\psi_c$ is reached with probability at least $1-1/n^c$.*

Proof. The probability for *not* reaching a state x with $\Psi_0(x) \leq 4\cdot\psi_c$ after t steps is at most $1/4^t$. We are interested in the complementary event, so its probability is at least $1-1/4^t$. For $t = c\cdot\log_4 n$ the statement follows immediately.

5.2 Nash Equilibrium

We now prove the upper bound for the expected time necessary to reach an exact Nash Equilibrium (Theorem 2.2, p.). Note that we assume that the speeds are all integer multiples of ϵ. If the speeds are arbitrary non-integers, convergence can become arbitrarily slow. The convergence factor α, which was $4s_{\max}$ in the original protocol, must be changed to $4s_{\max}/\epsilon$. For non-integer speeds, we have $\epsilon < 1$, so this effectively increases α.

To show convergence towards an exact Nash Equilibrium we cannot rely solely on the potential $\Psi_0(x)$, because when the system is close to a Nash equilibrium it is possible that the potential function increases even when a task makes a move that improves its perceived load. Therefore, we use a different potential function $\Phi_1(x)$, which we define as the *shifted potential function*

$$\Psi_1(x) = \Phi_1(x) - m^2/\mathcal{S} - m\cdot n/\mathcal{S} - n^2/4\mathcal{S} + 1/4\cdot\sum_i 1/s_i.$$

Let $\bar{s}_a$ and $\bar{s}_h$ denote the arithmetic mean and the harmonic mean of the speeds, i.e., $s_a = \sum_{i\in V} s_i/n$ and $s_h = n/\sum_{i\in V} 1/s_i$. Then, we can write

$$\Psi_1(x) = \Phi_1(x) - m^2/\mathcal{S} - m\cdot n/\mathcal{S} + n/4\cdot(1/\bar{s}_h - 1/\bar{s}_a).$$

Observation 5.11 The shifted potential $\Psi_1(x)$ has the following properties.

1. $\Psi_1(x) = \sum_{i\in V}[(e_i + 1/2)^2/s_i] - n/4\bar{s}_a$.
2. $\Psi_1(x) \geq 0$.
3. $\Psi_1(x) = \Psi_0(x) + \sum_{i\in V} e_i/s_i + n/4\cdot(1/\bar{s}_h - 1/\bar{s}_a)$.
4. $\Delta\Psi_1(X^t) = \Delta\Phi_1(X^t)$.

Before we can lower-bound the expected drop in $\Psi_1(x)$, we need a technical lemma regarding a lower bound to the load difference. It is similar to [6, Lemma 3.7], which concerned integer speeds, so the result here is more general.

Lemma 5.12 *Every edge (i,j) with $\ell_i - \ell_j > 1/s_j$ also satisfies $\ell_i - \ell_j \geq 1/s_j + \epsilon/s_i \cdot s_j$.*

Potential Ψ_1 differs from potential Ψ' defined in [6] by a constant only. Therefore, potential differences are the same for both potentials and we can apply results for Ψ' to Ψ_1.

Lemma 5.13 *If the system is in a state x that is not a Nash equilibrium, then*

$$\mathbf{E}[\Delta\Psi_1(X^{k+1})|X^t = x] \geq \frac{\epsilon^2}{8\Delta \cdot s_{\max}^3}.$$

Since the results of the previous section apply to Ψ_0 whereas now we work with Ψ_1, we add this technical lemma relating the two.

Lemma 5.14 *For any state x it holds $\Psi_1(x) \leq \Psi_0(x) + \sqrt{\Psi_0(x) \cdot n/\bar{s}_h} + n/4 \cdot (1/\bar{s}_h - 1/\bar{s}_a)$.*

To obtain a bound on the expected time the system needs to reach the NE, we use a standard argument from martingale theory. Let us abbreviate $V := \epsilon^2/(8\Delta \cdot s_{\max}^3)$. We introduce a new random variable Z_t which we define as $Z_t = \Psi_1(X^t) + t \cdot V$.

Lemma 5.15 *Let T be the first time step for which the system is in a Nash equilibrium. Then, for all times $t \leq T$ we have (1) $\mathbf{E}[Z_t|Z_{t-1} = z] \leq z$ (2) $\mathbf{E}[Z_t] \leq \mathbf{E}[Z_{t-1}]$.*

Corollary 5.16 *Let T be the first time step for which the system is in a Nash equilibrium. Let $t \wedge T$ be $\min\{t, T\}$. Then the random variable $Z_{t \wedge T}$ is a super-martingale.*

Corollary 5.17 *Let T be the first time step for which the system is in a Nash equilibrium. Then $\mathbf{E}[Z_T] \leq Z_0 = \Psi_1(X^0)$.*

Proof (Theorem 2.2). First, we assume that at time $t = 0$ the system is in a state with $\mathbf{E}[\Psi_0(X^T)] \leq 4 \cdot \psi_c$. Using the non-negativity of $\Psi_1(x)$ (Observation 5.11) allows us to state

$$\begin{aligned} V \cdot \mathbf{E}[T] &\leq \mathbf{E}[\Psi_1(X^T)] + V \cdot \mathbf{E}[T] = \mathbf{E}[Z_T] \\ \text{(Cor. 5.17)} &\leq \mathbf{E}[Z_0] = \Psi_1(X^0) \\ \text{(Lem. 5.14)} &\leq \Psi_0(X^0) + \sqrt{\Psi_0(X^0) \cdot \frac{n}{\bar{s}_h}} + \frac{n}{4} \cdot \left(\frac{1}{\bar{s}_h} - \frac{1}{\bar{s}_a}\right) \\ &\leq 4 \cdot \psi_c + \sqrt{4 \cdot \psi_c \cdot \frac{n}{\bar{s}_h}} + \frac{n}{4} \end{aligned}$$

Inserting the definition of ψ_c and dividing by V yields

$$\begin{aligned} \mathbf{E}[T] &\leq 8\Delta \cdot \frac{s_{\max}^3}{\epsilon^2} \cdot \\ &\left[\frac{64 \cdot n \cdot \Delta \cdot s_{\max}}{\lambda_2} + \sqrt{32 \cdot n^2 \cdot s_{\max}} \cdot \frac{2\Delta}{\lambda_2} + \frac{n}{4}\right] \\ &\leq 607 \cdot \Delta^2 \cdot \frac{s_{\max}^4}{\epsilon^2} \cdot \frac{n}{\lambda_2}. \end{aligned}$$

where we have used that $2\Delta/\lambda_2 \geq 1$ (Lemma 4.4) to pull that expression outside of the square root in the first line.

This bound was derived under the assumption that at $t = 0$ we had a state with $\mathbf{E}[\Psi_0(X^t)] \leq 4 \cdot \psi_c$. If this is not the case, let τ denote the number of time steps to reach such a state, and let T' denote the additional number of time steps to reach a NE from there. Combining the result from above with Theorem 2.1 allows us to write

$$\mathbf{E}[T] = \mathbf{E}[\tau + T'] = \mathcal{O}\left(\frac{n}{\lambda_2} \cdot \Delta^2 \cdot \frac{s_{\max}^4}{\epsilon^2}\right).$$

Corollary 5.18 *Similarly to Corollary 5.10, after $c \cdot \log_4 n$ blocks of T steps we have reached a Nash Equilibrium with probability at least $1 - 1/n^c$.*

Observation 5.19 Our bound in Theorem 2.2 is asymptotically lower than the corresponding bound in [6] by at least a factor of $\Omega(\Delta \cdot \mathrm{diam}(G))$.

Proof. Lemma 4.2 yields $n \cdot \mathrm{diam}(G) \geq 4/\lambda_2$. Additionally, we have $\mathcal{S} \geq s_{\max}$, since $s_{\max}$ occurs (at least once) in the sum of all speeds. Hence, the asymptotic bound from [6] is larger than

$$\mathcal{O}\left(n \cdot \frac{\Delta^2}{\lambda_2} \cdot s_{\max}^4 \cdot [\Delta \cdot \mathrm{diam}(G)]\right).$$

The first part of this expression is the bound of Theorem 2.2, so the expression in the square brackets is the additional factor of the bound from [6].

6. WEIGHTED TASKS

The set of tasks assigned to node i is called $x(i)$. The weight of node i becomes $w_i(x) = \sum_{\ell \in x(i)} \omega_\ell$ whereas the corresponding load is defined as $\ell_i(x) = w_i(x)/s_i$ in analogy to the unweighted case.

We present a protocol for weighted tasks that differs from the one described in [6]. A single step of that protocol is presented in Algorithm 2.

Algorithm 2: Distributed Selfish Load Balancing for weighted tasks

```
begin
    foreach task $\ell$ in parallel do
        Let $i = i(l)$ be the current machine of task $\ell$
        Choose a neighboring machine $j$ uniformly at random
        if $\ell_i - \ell_j > \frac{1}{s_j}$ then
            Move task $\ell$ from node $i$ to node $j$ with probability
                $p_{ij} := \frac{\deg(i)}{d_{i,j}} \cdot \frac{w_i - w_j}{2\alpha \cdot w_i}$
        end
    end
end
```

The notable difference to the scheme in [6] is that in our case, the decision of a task ℓ to migrate or not does not depend on that task's weight. In the original protocol, a load difference of more than w_ℓ/s_j would suffice for task ℓ

to have an incentive to migrate. In the modified protocol, a task will only move if the load difference is at least $1/s_j$. The advantage of this approach is that for an edge (i,j), either all or none of the tasks on node i have an incentive to migrate. This greatly simplifies the analysis. A justification for this step is that real systems usually have a cost associated with migration, so that a task might not want to migrate if the resulting gain is only marginal. We will show that the system rapidly converges to a state where $\ell_i - \ell_j \leq 1/s_j$ for all edges (i,j). Such a system is not necessarily a Nash equilibrium as $\ell_i - \ell_j$ might still be larger than the size of a given task ω_ℓ. We will show, however, that such a state is an ε-approximate NE.

In analogy to the unweighted case, we define the expected flow f_{ij} as the expected *weight* of the tasks migrating from i to j in state x. It is given by

$$f_{ij}(x) = \frac{\ell_i(x) - \ell_j(x)}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right) \cdot w_i(x)} \cdot \sum_{\ell \in x(i)} \omega_\ell = \frac{\ell_i(x) - \ell_j(x)}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)}.$$

The potentials Φ_0 and Φ_1 are defined analogously to the unweighted case. Here, we concentrate on Φ_0 alone. The average weight per node is W/n and the task deviation e_i is defined as $w_i - W/n$. We define $\Psi_0(x)$ in analogy to the unweighted case as the normalized version of Φ_0, with $\Psi_0 = \Phi_0 - W^2/\sum_i s_i = \sum_{i \in V} e_i^2/s_i$. The auxiliary quantity $\Lambda_{ij}(x)$ is defined analogously to the unweighted case as $\Lambda_{ij}(x) = (2\alpha - 2) \cdot d_{ij} \cdot (1/s_i + 1/s_j) \cdot f_{ij}(x)$.

6.1 Approximate Nash Equilibrium

In close analogy to [6, Lemma 3.1], we first bound the drop of the potential when the flow is exactly the expected flow.

Lemma 6.1 *Let $\tilde{\Delta}\Phi_0(X^{t+1}|X^t = x)$ denote the amount by which Φ_0 would drop if the system was in state x and the flow of tasks was exactly the expected flow f_{ij}. It holds $\tilde{\Delta}\Phi_0(X^{t+1}|X^t = x) \geq \sum_{(i,j)\in\tilde{E}} f_{ij} \cdot \Lambda_{ij}$.*

This lemma and its proof are formally equivalent to Lemma 3.1 in [6] and thus we omit the proof here.

Next, we bound the variance of the process.

Lemma 6.2 *The variances of the weights on the nodes are bounded via*

$$\sum_i \frac{\mathbf{Var}[w_i(X^t)|X^{t-1} = x]}{s_i} \leq \sum_{ij} f_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right).$$

Proof. As in the original reference, we introduce random variables A_i and C_i for the tasks *abandoning* and *coming to* node i, but now they count the *weight* of these tasks, not only their number. For the C_i, we again split it into Z_{ji} where Z_{ji} counts the weight migrating from j to i. Then

$$\mathbf{Var}[C_i] = \sum_{j:(j,i)\in\tilde{E}} \mathbf{Var}[Z_{ji}].$$

$$\mathbf{Var}[Z_{ji}] = \sum_{\ell \in x_j} \mathbf{Var}[\omega_\ell^{ji}].$$

Here ω_ℓ^{ji} is the random variable that is ω_ℓ if task ℓ moves from j to i and 0 otherwise. This variable follows a *Bernoulli distribution.* If p is the probability for the event to occur and if x is the value of the event, then the variance is

$$\mathbf{Var}[\mathsf{Ber}(x,p)] = x^2 \cdot p \cdot (1-p) \leq x^2 p.$$

This allows us to write

$$\begin{aligned}
\mathbf{Var}[Z_{ji}] &= \sum_{l \in x_j} \mathbf{Var}[\omega_\ell^{ji}] \\
&\leq \sum_{l \in x_j} \omega_\ell^2 \cdot \frac{\ell_j - \ell_i}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right) \cdot w_j} \\
&\leq \sum_{l \in x_j} \omega_\ell \cdot \frac{\ell_j - \ell_i}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right) \cdot w_j} \\
&= \frac{\ell_j - \ell_i}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)} = f_{ij},
\end{aligned}$$

where we use that $\omega_\ell^2 \leq \omega_\ell$ since all tasks have weight at most 1. Hence

$$\mathbf{Var}[C_i] = \sum_{j:(j,i)\in\tilde{E}} f_{ij}.$$

Similarly, we define the random variable $A_{\ell i}$ that is ω_l if task ℓ abandons node i and 0 otherwise. It is also Bernoulli-distributed.

$$\begin{aligned}
\mathbf{Var}[A_i] &= \sum_{\ell \in x_i} \mathbf{Var}[A_{\ell i}] \\
&= \sum_{\ell \in x_i} \mathbf{Var}\left[\mathsf{Ber}\left(\omega_\ell; \sum_{j:(i,j)\in\tilde{E}} \frac{\ell_i - \ell_j}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right) \cdot w_i}\right)\right] \\
&\leq \sum_{\ell \in x_i} \omega_\ell^2 \cdot \sum_{j:(i,j)\in\tilde{E}} \frac{\ell_i - \ell_j}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right) \cdot w_i} \\
&\leq \sum_{j:(i,j)\in\tilde{E}} \sum_{\ell \in x_i} \omega_\ell \cdot \frac{\ell_i - \ell_j}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right) \cdot w_i} \\
&= \sum_{j:(i,j)\in\tilde{E}} \frac{\ell_i - \ell_j}{\alpha \cdot d_{ij} \cdot \left(\frac{1}{s_i} + \frac{1}{s_j}\right)} = \sum_{j:(i,j)\in\tilde{E}} f_{ij}.
\end{aligned}$$

When we add the variance of C_i and A_i and sum over all nodes, we get, in formal analogy to the unweighted case,

$$\sum_i \frac{\mathbf{Var}[w_i(X^t)|X^{t-1} = x]}{s_i} = \sum_{ij} f_{ij} \left(\frac{1}{s_i} + \frac{1}{s_j}\right).$$

This allows us to formulate a bound on the expected potential drop in analogy to [6, Lemma 3.3] by combining Lemma 6.1 and Lemma 6.2.

Lemma 6.3 *The expected drop in potential Φ_0 if the system is in state x is at least*

$$E[\Delta\Phi_0(X^t)|X^{t-1} = x] \geq \sum_{(i,j)\in\tilde{E}} f_{ij}(x) \cdot (\Lambda_{ij}(x) - 2).$$

The proof is formally analogous to [6, Lemma 3.3].

Proof of Theorem 2.3. The rest of the proof is the same as the proof for the unweighted case in Section 5. One may verify that, indeed, Lemma 5.2 and all subsequent results do not rely on the specific form of ℓ_i or the underlying nature of the tasks. Using the same eigenvalue techniques as in the unweighted case, this allows us to obtain a bound involving the second smallest eigenvalue of the graph's Laplacian matrix. Following the steps of the unweighted case then allows us to prove the main result of this section.

Acknowledgements

The authors thank Thomas Sauerwald for helpful discussions.

7. REFERENCES

[1] H. Ackermann, S. Fischer, M. Hoefer, and M. Schöngens. Distributed algorithms for qos load balancing. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, SPAA'09, pages 197–203, New York, NY, USA, 2009. ACM.

[2] C. P. J. Adolphs and P. Berenbrink. Improved Bounds for Discrete Diffusive Load Balancing. To appear at IPDPS 2012, 2012.

[3] B. Awerbuch, Y. Azar, and R. Khandekar. Fast load balancing via bounded best response. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 314–322, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[4] P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. W. Goldberg, Z. Hu, and R. Martin. Distributed Selfish Load Balancing. *SIAM Journal on Computing*, 37(4):1163, 2007.

[5] P. Berenbrink, T. Friedetzky, I. Hajirasouliha, and Z. Hu. Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks. In L. Arge, M. Hoffmann, and E. Welzl, editors, *Proceedings of the 15th Annual European Symposium on Algorithms (ESA 2007)*, volume 4698/2007 of *Lecture Notes in Computer Science*, pages 41–52. Springer, Springer, October 2007.

[6] P. Berenbrink, M. Hoefer, and T. Sauerwald. Distributed selfish load balancing on networks. In *Proceedings of 22nd Symposium on Discrete Algorithms (SODA'11)*, pages 1487–1497, 2011.

[7] J. E. Boillat. Load balancing and Poisson equation in a graph. *Concurrency: Practice and Experience*, 2(4):289–313, Dec. 1990.

[8] F. R. K. Chung. *Spectral graph theory.* AMS Bookstore, 1997.

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

[10] R. Elsässer, B. Monien, and R. Preis. Diffusion Schemes for Load Balancing on Heterogeneous Networks. *Theory of Computing Systems*, 35(3):305–320, May 2002.

[11] R. Elsässer, B. Monien, and S. Schamberger. Distributing unit size workload packages in heterogeneous networks. *Journal of Graph Algorithms and Applications*, 10(1):51–68, 2006.

[12] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibrium in load balancing. *ACM Transactions on Algorithms*, 3(3):32–es, Aug. 2007.

[13] E. Even-Dar and Y. Mansour. Fast convergence of selfish rerouting. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA'05)*, pages 772–781, 2005.

[14] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In J. Baeten, J. Lenstra, J. Parrow, and G. Woeginger, editors, *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 190–190. Springer Berlin / Heidelberg, 2003.

[15] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[16] S. Fischer, P. Mahonen, M. Schongens, and B. Vocking. Load balancing for dynamic spectrum assignment with local information for secondary users. In *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pages 1 –9, oct. 2008.

[17] S. Fischer and B. Vöcking. Adaptive routing with stale information. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing (PODC'05)*, pages 276–283, New York, NY, USA, 2005. ACM.

[18] D. Fotakis, A. Kaporis, and P. Spirakis. Atomic congestion games: Fast, myopic and concurrent. *Theory of Computing Systems*, 47:38–59, 2010. 10.1007/s00224-009-9198-2.

[19] T. Friedrich and T. Sauerwald. Near-perfect load balancing by randomized rounding. In *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*, page 121, New York, New York, USA, May 2009. ACM Press.

[20] B. Mohar. Isoperimetric numbers of graphs. *Journal of Combinatorial Theory, Series B*, 47(3):274–291, Dec. 1989.

[21] B. Mohar. Eigenvalues, diameter, and mean distance in graphs. *Graphs and Combinatorics*, 7(1):53–64, Mar. 1991.

[22] Mohar, B. The Laplacian Spectrum of Graphs. In Y. Alavi, editor, *Graph theory, combinatorics, and applications*, volume 2, pages 871–898. Wiley, 1991.

[23] S. Muthukrishnan, B. Ghosh, and M. Schultz. First- and Second-Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing. *Theory of Computing Systems*, 31(4):331–354, July 1998.

[24] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of Markov chains and the analysis of iterative load-balancing schemes. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 694–703. IEEE Comput. Soc, 1998.

[25] B. Vöcking. Selfish Load Balancing. In N. Nisan, E. Tardos, T. Roughgarden, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 20. Cambridge University Press, 2007.