Threshold Load Balancing With Weighted Tasks

Petra Berenbrink[a], Tom Friedetzky[b], Frederik Mallmann-Trenn[a,c], Sepehr Meshkinfamfard[b], Chris Wastell[b]

[a] *School of Computing Science, Simon Fraser University, Burnaby, Canada*
[b] *School of Engineering and Computing Sciences, Durham University, Durham, UK*
[c] *Département Informatique, École normale supérieure, PSL Research University, Paris, France*

Abstract

We study threshold-based load balancing protocols for weighted tasks. We are given an arbitrary graph G with n nodes (resources, bins) and $m \geq n$ tasks (balls). Initially the tasks are distributed arbitrarily over the n nodes. The resources have a threshold and we are interested in the balancing time, i.e., the time it takes until the load of all resources is below or at the threshold. We distinguish between resource-based and user-based protocols. In the case of resource-based protocols resources with a load larger than the threshold are allowed to send tasks to neighbouring resources. In the case of user-based protocols the tasks make the migration decisions and we restrict ourselves to the complete graph in the model. Any task allocated to a resource with a load above the threshold decides whether to migrate to a neighboring resource independently of the other tasks.

For resource-controlled protocols we present results for arbitrary graphs. For the user-controlled migration we consider complete graphs and derive bounds for both above-average and tight thresholds.

Keywords: Threshold Load balancing, Random walks, Mixing time of random walks, Weighted Tasks
2010 MSC: 68M10, 68M12, 68M14, 68W10, 68W20

1. Introduction

We study threshold-based balls-into-bins schemes that can be used to balance load in distributed systems. The balls usually model tasks or data and the bins model the resources used to process the tasks or to store the data. The performance of a distributed system often depends on the maximum load of any of the machines. The higher the maximum load, the longer the execution

Email addresses: petra@sfu.ca (Petra Berenbrink), tom.friedetzky@dur.ac.uk (Tom Friedetzky), fmallman@sfu.ca (Frederik Mallmann-Trenn), sepehr.meshkinfamfard@dur.ac.uk (Sepehr Meshkinfamfard), christopher.wastell@dur.ac.uk (Chris Wastell)

time of the entire system. Hence, good load balancing schemes are crucial for efficient computations on distributed systems.

Most balls-into-bins games studied theoretically so far assume that the balls are of equal size. The size of a ball usually measures the computational requirements of the task it models, or the size of the data. However, this assumption is unrealistic in many cases. In this paper we study load balancing schemes with weighted balls which are, for example, able to model tasks with different service times.

We are given an arbitrary graph G with n nodes representing the resources (machines) and $m \geq n$ weighted tasks (balls) with a total weight of W. Initially the tasks are distributed arbitrarily over the n nodes. Similar to Ackermann et al. [1] and Hoefer and Sauerwald [2], we assume every resource has a threshold which is the maximum load the resource can accept. The threshold is the same for all resources (models with non-uniform thresholds are certainly conceivable, but are not focus of this paper). We distinguish between a tight threshold of $W/n + w_{max}$, and above-average thresholds of $(1+\varepsilon) \cdot W/n + w_{max}$, where w_{max} is the maximum weight of any task and ε is an arbitrary positive constant. Note that the thresholds must be chosen appropriately, i.e., they must be at least the average load. We assume that the average load can either be derived quickly by, for example, using a diffusion load balancing process[1], or the threshold are provided externally. In this paper we are interested in the balancing time, i.e., the time it takes until the load of all resources is below or at the threshold.

We distinguish between resource-based protocols where the resources with a load above the threshold (called overloaded resources) are allowed to send tasks to a neighboring resource, and user-based protocols where each task on an overloaded resource decides autonomously whether or not to migrate to a neighboring resource.

For resource-based protocols we consider arbitrary graphs. Our results are expressed in terms of the mixing time (for above-average thresholds) and hitting time (for tight thresholds) of random walks on the underlying graph. Subsequently we will show that these bounds are tight. For user-based protocols we consider complete graphs only. In all cases our bounds (for the weighted case) match the bounds of Ackermann et al. [1] and Hoefer and Sauerwald [2]. All protocols we consider are decentralized and do not require a global view of the system.

2. Contributions

We follow the work of Ackermann et al. [1] and Hoefer and Sauerwald [2]. We study threshold-based balls-into-bins protocols, but in contrast to these papers we assume that the balls (tasks) have arbitrary weights.

[1] Each resource keeps a value representing the current estimated average load and this value is initialized with the initial load of the resource. The resources then simulate continuous diffusion load balancing (always using their current estimate) for mixing time number of steps, at which point their estimates will be concentrated around the average load.

Resource-Controlled Protocols. For arbitrary graphs G and assuming above-average thresholds, in Theorem 3 we prove a balancing time of $O(\tau(G) \cdot \log m)$ w.h.p., where $\tau(G)$ is the mixing time of a random walk on G. Note that this bound does not depend on the weights of the tasks. In Hoefer and Sauerwald [2] the authors show a bound of $O(\mathcal{H}(G) \cdot \log n + \tau(G) \cdot \log m)$ on the expected balancing time, where $\mathcal{H}(G)$ is the hitting time of a random walk on G (see Section 4). Note that our bounds for weighted tasks match their bound for uniform tasks and are even stronger in the sense that we do not have the hitting time in our bound. The reader may refer to Table 2 to find a comparison of hitting times and mixing times of several common graphs. From Theorem 3.7 of Hoefer and Sauerwald [2] it follows that our bound is tight.

Graph	Mixing Time	Hitting Time
Complete Graph	$O(1)$	$O(n)$
Reg. Expander	$O(\log n)$	$O(n)$
Erdős-Rényi Graph	$O(\log n)$	$O(n)$
Hypercube	$O(\log n \log\log n)$	$O(n)$
Grid	$O(n)$	$O(n \log n)$

Table 1: Summary of mixing and hitting times for common graphs. All results come from Aldous and Fill [3]. We assume that that the regular expander has degree at least 3 and that that Erdős-Rényi Graph $G(n,p)$ fulfils $p > (1+\varepsilon)\log n/n$.

For tight thresholds we show (Theorem 7) a bound of $O(\mathcal{H}(G) \cdot \log m)$ on the expected balancing time. This bound matches the bound presented in Hoefer and Sauerwald [2]. Again, the bound is independent of the weight of the tasks. In Observation 8 we show that this bound is tight.

User-Controlled Protocols. We consider complete graphs for above-average and tight thresholds. For the former we derive a bound of $O(w_{max}/w_{min} \cdot \log m)$ on the expected balancing time (Theorem 11) where w_{max} and w_{min} denote the maximum and minimum weight of any task respectively. Moreover, for tight thresholds we derive (Lemma 12) a bound of $O\left(\frac{w_{max}}{w_{min}} \cdot \frac{\log m}{n^2}\right)$.

Both bounds match the bounds of Ackermann et al. [1] for uniform weights. However for weighted balls our bounds include the additional factor of w_{max}/w_{min}.

In Section 7, we briefly discuss experimental results highlighting cases where the theoretically derived bounds for user-controlled migration are tight and where not.

3. Related Work

There are many results for balls-into-bins games for uniform balls. Here we concentrate on protocols for weighted balls and on local protocols that use a threshold to allocate the balls.

3.1. Threshold-based protocols

In Adler et al. [4] the authors consider parallel threshold-based protocols. They investigate the trade-off between the number of rounds of communication between the resources and the final load. Specifically, for a given number of rounds of communication r, they prove a *lower* bound on the maximum load of $\Omega(\sqrt[r]{\log n \log\log n})$ for n unit-sized balls and n resources.

The papers of Ackermann et al. [1] and Hoefer and Sauerwald [2] are the most closely related to our work. However in both cases the authors consider only uniform balls. In Ackermann et al. [1] the authors show results for threshold-based balancing protocols for user-controlled migration on complete graphs. For above-average thresholds they show a bound on the balancing time of $O(\log m)$, and $O(n^2 \cdot \log m)$ for tight thresholds. They assume that the tasks know the the load and the threshold of their current resource. The results of Ackermann et al. [1] were generalized to arbitrary graphs by Hoefer and Sauerwald [2]. For the resource-controlled protocols the authors provide a bound of $O\left(\mathcal{H}(G) \cdot \log m\right)$ on the balancing time, where $\mathcal{H}(G)$ is the hitting time of a random walk on G. They also show an improved bound of $O\left(\mathcal{H}(G) \cdot \log n + \tau(G) \cdot \log m\right)$ for above-average thresholds, where $\tau(G)$ is the mixing time of a random walk on G. For the user-controlled protocol they provide a bound of $O\left(n^5 \cdot \mathcal{H}(G) \cdot \log m\right)$ to reach a balanced state.

In Berenbrink et al. [5] the authors consider a sequential balls-into-bins process that randomly allocates m uniform balls into n bins using thresholds. They analyze two allocation schemes that achieve a close to optimal maximum load of $\lceil m/n \rceil + 1$ and require only $O(m)$ random choices.

3.2. Weighted balls

Berenbrink et al. [6] were among the first to consider the problem of allocating weighted balls in parallel. Their key result is a generalization of the upper bound presented by Stemann [7] to weighted balls. They present a protocol that achieves a maximum load of $\gamma \cdot (m/n \cdot w_{avg} + w_{max})$ using $O\left(\log\log n/(\log\gamma \cdot ((m/n) \cdot \Delta + 1)\right)$ communication rounds, where w_{avg} and w_{max} denote the average and maximum weight respectively and $\Delta = w_{avg}/w_{max}$.

In Berenbrink et al. [8] the authors investigate generic multiple-choice balls-into-bins protocols with weighted balls, showing some surprisingly counter-intuitive properties. Talwar and Wieder [9] investigate a sequential balls-into-bins process where m weighted balls are allocated into n bins. The process allocates every ball into the least loaded of two randomly chosen bins. The case where each of the m balls has unit weight has been studied by Berenbrink et al. [10]. The authors show that the difference between the maximum load and the average load does not increase with m. Talwar and Wieder [9] show that as long as the weight distribution has finite second moment and satisfies a mild technical condition, the gap between the load of the heaviest bin and the average load is independent of the number balls thrown. Peres et al. [11] consider the so-called $(1+\beta)$-process where each ball goes to a random bin with probability β and to the least-loaded of two randomly chosen bins with a probability of

$(1-\beta)$. The authors show that for the $(1+\beta)$-process the gap between the minimum and average load is at most $\Theta(\log n/\beta)$, independent of m. They also show that the gap remains $\Theta(\log n/\beta)$ in the weighted case for a large class of weight distributions.

Berenbrink et al. [12] consider balls-into-bins games in a user-controlled setting. In the beginning the balls are arbitrarily distributed over the bins. The protocol works in parallel rounds. In every round each user (ball) randomly selects another bin and moves to the chosen bin if the load is smaller than the load of its current bin. In particular, they provide an upper bound on the balancing time of $\log\log m + \text{poly}(n)$, where $\text{poly}(n) = O(n^c)$ for some constant c. Berenbrink et al. [13] generalize these results to weighted balls. They show that their protocol yields an expected balancing time of $O(nmw_{max}^3\epsilon^{-2})$ for $\epsilon \in (0,1)$. Finally, Adolphs and Berenbrink [14] generalize the latter result to weighted balls and non-uniform resources.

4. Model

Let $[m] = \{1, 2, \ldots, m\}$ be the set of tasks and let $[n] = \{1, 2, \ldots, n\}$ be the set of resources[2], $m \geq n$. The resources are connected by an arbitrary undirected graph $G = (V, E)$. Let d_i be the degree of node i and d the maximum degree. Tasks on a resource r can move to a neighboring resource r' if $(r, r') \in E$.

Each task $i \in [m]$ has an associated weight $w_i \in \mathbb{R}$. Let w_{max} be the maximum weight and let $W = \sum_{i=1}^{m} w_i$ be the total weight of all tasks. We assume that $w_{min} \geq 1$. If this is not the case, then one can easily scale all parameters, such that $w_{min} = 1$.

For $t \geq 0$ let

$$x(t) = (x_1(t), x_2(t), \ldots, x_n(t))$$

denote the load vector at the beginning of step t (before the task removal) where $x_r(t)$ is the load of resource r. Then

$$X(t) = (X_1(t), X_2(t), \ldots, X_n(t))$$

denotes the state of the system at the beginning of step t, where $X_r(t)$ is the random variable that denotes the load of resource r. We use $b_r(t)$ to denote the (actual) number of tasks (balls) on machine r at time t. Our protocols will use a threshold

$$\text{Tr} = (1+\varepsilon) \cdot W/n + w_{max}$$

with $\varepsilon \geq 0$.

4.1. Random walks

For an undirected, connected graph G let $\mathbf{P}_{i,j}$ be the probability that the random walk moves from node i to node j. We consider standard random walks

[2]Throughout the paper we assume that n is bounded below by a large enough constant

for non-regular graphs with transition matrix $\mathbf{P}$, where $\mathbf{P}_{i,j} = 1/d$ for $i \neq j$ and $(i,j) \in E$ and $\mathbf{P}_{i,i} = (d - d_i)/d$, where d is the maximum degree of the graph. Let $\mathbf{P}^t$ be the t-th power of $\mathbf{P}$. Then $\mathbf{P}^t_{i,j}$ is the probability that a random walk starting from node i is located at node j after exactly t steps. The stationary distribution of the random walk on G is called $\pi(G)$ and it is the uniform distribution for this random walk.

In [15] the authors give a bound on the *mixing time* $\tau(G)$ of the random walk, which is defined as the expected time it takes for the random walk (defined above) on G to approach its the stationary distribution. A statement of the result can be found in Lemma 2. By Lemma 2, we can assume that $\tau(G) = 4 \log n/\mu$, where μ is the spectral gap of P. With $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ the n eigenvalues of $\mathbf{P}$ we have

$$\mu := 1 - \max_{2 \leq i \leq n} \{|\lambda_i|\}.$$

The hitting time $\mathcal{H}_{u,v}(G)$ of a random walk is defined as the expected time for a random walk to reach node $v \in V$ when starting from node $u \in V$. The maximum hitting time is

$$\mathcal{H}(G) := \max_{u,v \in V} \mathcal{H}_{u,v}(G).$$

We assume thus that all nodes know the maximum degree d or have a bound on it. The results in this paper hold for all random walks where the stationary distribution equals the uniform distribution. Note that the concrete random walk used will affect the spectral gap.

5. Resource-Controlled Migration

In this section, we consider a protocol in which each overloaded resource determines whether or not tasks should migrate (Algorithm 5.1). Tasks which are currently assigned to resource r can only move to neighboring resources of r (see the algorithm below). The protocol is distributed and every node requires only knowledge of its own load and the global threshold. The algorithm in Figure 5.1 shows one step of our protocol.

We assume that every resource stores all its tasks in a stack data structure. If several balls arrive at the same resource in one time step the new balls are added in an arbitrary order. The *height* $h^i_r(t)$ of task i on resource r at time t is the sum of the weights of all tasks in the data structure that are positioned below i. We say task i is *cutting the threshold* Tr if $h^i_r(t) < \mathrm{Tr}$ and $h^i_r(t) + w_i > \mathrm{Tr}$.

Each task on resource $r \in [n]$ can either be completely *above*, *completely below*, or *cutting* the threshold. Let $I^a_r(t)$ and $I^b_r(t)$ be the sets of tasks on resource r that are completely above and completely below the threshold at the beginning of step t respectively. $I^c_r(t)$ denotes the task on resource r which cutting the threshold at the beginning of step t.

We say that a task is *accepted* by a resource if the height of the task plus its weight is less than or equal to the threshold. Note that once a task is accepted by

Algorithm 5.1 Resource Controlled

for all resources r in parallel **do**
  if $x_r(t) >$ Tr **then**
    Remove each task $i \in I_r^a(t) \cup I_r^c(t)$ and reallocate the task to a neighboring resource that is chosen according to transition matrix $\mathbf{P}$.
  Assign new heights to all migrated balls

a resource, it will never leave that resource again. We call these tasks *inactive*, and the tasks that are not accepted by a resource are called *active*.

5.1. Above-Average Thresholds

In this section, we assume thresholds are larger by some constant factor than the average load W/n. i.e., $\text{Tr} = (1+\varepsilon) \cdot W/n + w_{max}$. The next lemma will be used in the proof of this section's main result. It estimates the probability that a randomly chosen resource has a load below or equal to the threshold. The result is quite easy to see and it allows us in Section 6 to simplify the potential analysis of [1].

Lemma 1. *Assume that r is a resource chosen uniformly at random in step t, then*

$$P[X_r(t) \leq Tr - w_{max}] \geq \varepsilon/(1+\varepsilon).$$

Proof. From a simple pigeonhole argument it follows that at any point in time (meaning also at the end of a step) there is a fraction of $\varepsilon/(1+\varepsilon)$ resources that can accept an additional task of any weight (not larger than w_{max}). Assume this is not the case. (Recall that $Tr = (1+\epsilon) \cdot W/n + w_{max}$.) Then there are at least

$$(1 - \varepsilon/(1+\varepsilon)) \cdot n + 1$$

many resources with a weight of at least $(1+\varepsilon) \cdot W/n$. Then

$$((1 - \varepsilon/(1+\varepsilon)) \cdot n + 1)(1+\varepsilon) \cdot W/n > W,$$

which is a contradiction. □

The following result holds for arbitrary graphs. It is stated in terms of the performance a random walk on G. Let $\tau(G)$ denote the mixing time of the underlying Markov chain. For complete graphs the result shows a balancing time of $O(\log m)$. We use the next lemma, shown in [2], to prove the main result of this section, Theorem 3.

Lemma 2 ([2, Lemma 2.1.])**.** *Let G be an arbitrary graph. Let $\mathbf{P}$ be the transition matrix of a random walk on G with stationary distribution π. Let $t \geq 4 \log n/\mu$, where μ is the spectral gap of $\mathbf{P}$. Then $\mathbf{P}^t_{i,j} = \pi_i \pm n^{-3}$.*

Theorem 3. *Let $Tr = (1+\varepsilon) \cdot W/n + w_{max}$. Let G is an arbitrary undirected graph and let $\tau(G)$ be the mixing time of the random walk on G with uniform*

stationary distribution. Let c be an arbitrary positive constant. Then, with a probability at least $1 - n^{-c}$ all tasks are allocated after

$$2(c+1) \cdot \tau(G) \cdot \log m / \log\left(\frac{2(1+\varepsilon)}{2+\varepsilon}\right)$$

time steps.

Proof. Following our algorithm we can assume that every active task performs a random walk with transition matrix $\mathbf{P}$, until it reaches a resource that has a load that is small enough to accept the task. If a task is accepted by a resource, the task becomes *inactive*. All other tasks are *active*. We now divide time into *phases* of length $2\tau(G)$ each, where $\tau(G)$ is the mixing time of the random walk on G.

We fix a phase j and consider the tasks in increasing order $1, 2, \ldots, m$. Consider the i-th task. If i is still active at the end of the $(j-1)$-st phase. Let $r_{i,j}$ be the resource that it visits in the last step of phase j. From Lemma 2 it follows that for every $1 \leq k \leq n$,

$$P[r_{i,j} = k] = n^{-1} \pm n^{-3}.$$

From Lemma 1 it follows that the probability that task i is successful in the last step in phase j is at least

$$\frac{\varepsilon n}{1+\varepsilon} \cdot \left(\frac{1}{n} - \frac{1}{n^3}\right) \geq \frac{\varepsilon}{2(1+\varepsilon)}.$$

The crucial property is that this does not depend on whether the tasks $1, 2, \ldots, i-1$ were successful in phase j: By Lemma 1 a resource—chosen uniformly at random—accepts i with probability at least $\frac{\varepsilon n}{1+\varepsilon}$ even if all other tasks are already inactive.

Now assume that after ℓ phases, with $\ell \geq (c+1) \cdot \log m / \log(2(1+\varepsilon)/(2+\varepsilon))$,there exists a task i that is still active. A necessary condition for the task to still be active is that it was not accepted by any resource in any of the last steps of the ℓ phases. The probability that the task was not accepted by a resource in any of the last steps of each of the ℓ rounds is at most

$$\left(1 - \frac{\varepsilon}{2(1+\varepsilon)}\right)^{\ell} \leq \left(\frac{1}{m}\right)^{c+1}.$$

The result follows by the union bound. □

5.2. Tight Threshold

The next theorem shows results for a tighter threshold

$$\mathrm{Tr} = W/n + 2w_{max}.$$

Our result bounds the balancing time in terms of the *hitting time* $\mathcal{H}(G)$.

We call an assignment of the weighted tasks to the resources *proper* if no resource has a weight of more than $W/n + w_{max}$. Note that it is trivial to calculate a proper assignment in a centralized manner. The simple *first fit* rule will work.

To analyze the protocol we will use a potential function that counts the weights of the tasks that are partially or completely above the threshold. At any $t \geq 0$ the potential Φ is defined as

$$\Phi(X(t)) = \sum_{i \in I^a(t) \cup I^c(t)} w_i, \tag{1}$$

where $I^a(t)$ ($I^c(t)$) is the set of all the tasks that are above (partially above) the threshold. For any $t > 0$ the potential change between subsequent states $X(t)$ and $X(t+1)$ is defined as

$$\Delta\Phi(t+1) = \Phi(X(t)) - \Phi(X(t+1)). \tag{2}$$

The next observation shows that the potential function is non-increasing.

Observation 4. *For any $t > 0$ we have*

$$\Delta\Phi(t+1) \geq 0.$$

Proof. Based on the definition, at any time $t > 0$, any task i can either be in $I^c(t) \cup I^a(t)$ or $I^b(t)$. For ease of presentation we assume that the protocol considers the tasks sequentially in an arbitrary order. In the following, we call a step where one of the tasks is considered a *sub-step*. Assume that task i is one of the tasks that is moved to a neighboring resource in sub-step t'. Note that task i can be moved to another resource if and only if $i \in I^c(t) \cup I^a(t)$ at the beginning of step t. At the beginning of step $t+1$, either $i \in I^c(t+1) \cup I^a(t+1)$ or $i \in I^b(t+1)$. In the former case the potential remains unchanged, in the latter case $\Delta\Phi(t+1) = w_i$. Also, note that due to our stack ordering the potential does not change due to a task that did not move to another resource in sub-step t'. □

The next lemma estimates the one-step potential decrease.

Lemma 5. *Assume $Tr = W/n + 2w_{max}$.*

$$\mathbb{E}\left[\Delta\Phi(t + 2\mathcal{H}(G)) \mid X(t) = x(t)\right] \geq \frac{\Phi(X(t))}{4}.$$

Proof. We consider a *phase* of length $2\mathcal{H}(G)$, where $\mathcal{H}(G)$ is the maximum hitting time. At the beginning of each phase each active task is assigned (using a proper assignment) to a resource, called the *target* resource, such that the maximum load assigned to any resource is at most $W/n + w_{max}$. Then for every active task we place a token on its current resource and let the tokens perform a random walk of length $\mathcal{H}(G)$. If a token lands during the phase

on the target resource of the corresponding task then the task is marked *blue*, otherwise the task is marked *red*. Let B and R be the set of blue and red tasks respectively. Also let $W(B)$ and $W(R)$ be the total weight of the blue and red tasks respectively. Note that

$$\Phi(X(t)) = W(B) + W(R)$$

since $\Phi(X(t))$ is the weight of the active tasks at the beginning of time step t.

Let $\mathcal{A}_t$ be an assignment at time t of tasks to targets (resources) such that the maximum load of any resource is at most $W/n + w_{max}$. We define $\Delta\Phi'(t + 2\mathcal{H}(G))$ to be the potential drop as a result of the blue tasks' allocation in $\mathcal{A}_t$ in $2\mathcal{H}(G)$ time, i.e., the potential drop due to tasks which 'hit' their target.

In symbols, $\Delta\Phi'(t + 2\mathcal{H}(G)) = W_t(B) - W_{t+2\mathcal{H}(G)}(B)$, where $W_{t'}(B)$ is the total weight of the blue tasks at time t'.

We know that the expected time for a random walk on graph G to hit the target resource is the maximum hitting time $\mathcal{H}(G)$. By Markov's inequality, $P[X > a] \leq \frac{\mathbb{E}[X]}{a}$, where X is a positive random variable and $a > 0$. We obtain that the probability for a random walk to hit its target resource after $2\mathcal{H}(G)$ time steps is at least $1/2$. Therefore we have

$$\mathbb{E}\left[\Delta\Phi'(t + 2\mathcal{H}(G))\right] \geq \Phi(X(t))/2.$$

Now we inspect the original process where every active task goes on a random walk until it is accepted by a resource that has sufficient space to accept it. Assume the following holds.

$$\mathbb{E}\left[\Phi(X(t)) - \Phi(X(t + 2\mathcal{H}(G)))\right] \geq \mathbb{E}\left[\frac{\Delta\Phi'(t + 2\mathcal{H}(G))}{2}\right] \tag{3}$$

Then we have

$$\begin{aligned}\mathbb{E}\left[\Delta\Phi(t + 2\mathcal{H}(G))\right] = \mathbb{E}\left[\Phi(X(t)) - \Phi(X(t + 2\mathcal{H}(G)))\right] &\geq \mathbb{E}\left[\frac{\Delta\Phi'(t + 2\mathcal{H}(G))}{2}\right] \\ &\geq \frac{\Phi(X(t))}{4},\end{aligned}$$

which finishes the proof. Thus, in the remainder we show (3). Without loss of generality, we can assume that each task, as long as it is not accepted by any resource, pursues its corresponding token. If it is accepted by one of the resources it is marked as inactive and its random walk terminates. We divide all the resources into two different sets. The resources that are not able to accept all the incoming tasks throughout a phase, $2\mathcal{H}(G)$ time steps, are labelled as *full*. Note that the load of a full resource has to be larger than $W/n + w_{max}$ by the end of a phase. A resource that is not categorized as full is called *good*. A good resource is able to accept all the incoming tasks within a phase without crossing the $W/n + w_{max}$ line.

Moreover, we categorize all blue and red tasks into three different sets. The

set of blue tasks that are accepted by a good resource during a phase is denoted by B_g. Similarly, R_g are the red tasks that are accepted by a good resource. It follows that R_g and B_g are tasks which have not taken the position of any other task at its target resource. The tasks B_f (R_f) are blue (red) tasks that are accepted by a full resource in a phase. Lastly, the blue and red tasks that remain active and are not accepted by any resource at the end of a phase are denoted by B_n and R_n respectively. Therefore we have

$$\Delta\Phi'(X(t+2\mathcal{H}(G))) = W(B) = W(B_g) + W(B_f) + W(B_n)$$

and

$$\Phi(X(t)) - \Phi(X(t+2\mathcal{H}(G))) = W(B_g) + W(B_f) + W(R_g) + W(R_f).$$

Firstly, we show that $W(B_f) + W(R_f) \geq W(B_n)$ and then we prove our claim, *i.e.*

$$\Delta\Phi(t+\mathcal{H}(G)) \geq \frac{1}{2}\Delta\Phi'(t+\mathcal{H}(G)).$$

In order to show that $W(B_f) + W(R_f) \geq W(B_n)$, let ℓ_b be the total weight of the tasks that are accepted by resource b at the beginning of the phase. Then it follows that

$$W(B_n) \leq \sum_{b \in \textsc{Full}} \frac{W}{n} + w_{max} - \ell_b. \tag{4}$$

The inequality in (4) is valid because $\sum_{b\in\textsc{Full}} W/n + w_{max} - \ell_b$ is an upper bound for the total weight of the blue tasks that are able to be assigned to a full resource during a phase (since it counts all full resources). By definition, the tasks in the set B_n reach their target resource but are not accepted. It follows that the target resource of these tasks are full resources since they were unable to accept all tasks during the phase. For a resource to not accept all tasks it must have load greater than $W/n + w_{max}$. It follows that each task in B_n is on a resource that has accepted tasks with weight at least $W/n + w_{max} - \ell_b$ otherwise the task would have been accepted.

Also we have

$$W(R_f) + W(B_f) \geq \sum_{b \in \textsc{Full}} \frac{W}{n} + w_{max} - \ell_b. \tag{5}$$

The tasks in the sets R_f and B_f are accepted during the phase and are on a full resource. Based on the definition of a full resource, which is a resource with a load larger than $W/n + w_{max}$ (since the resource did not accept all tasks during the phase), (5) holds. Note that $\mathrm{Tr} = W/n + 2w_{max}$.

Now we show that

$$\Delta\Phi(t+\mathcal{H}(G)) \geq \frac{1}{2}\Delta\Phi'(t+\mathcal{H}(G)).$$

Clearly $W(B_g) \geq 0$ and $W(R_f) \geq 0$, and therefore $W(B_g) + W(R_f) \geq 0$. Adding $W(R_f)$ on both sides gives

$$W(B_g) + 2W(R_f) \geq W(R_f).$$

Adding a $W(B_f)$ on both sides gives

$$W(B_g) + 2W(R_f) + W(B_f) \geq W(R_f) + W(B_f)$$

Since $W(B_f) + W(R_f) \geq W(B_n)$ we have

$$W(B_g) + 2W(R_f) + W(B_f) \geq W(B_n).$$

Dividing both sides by 2 gives

$$W(B_g)/2 + W(R_f) + W(B_f)/2 \geq W(B_n)/2.$$

Adding $W(B_g)/2 + W(B_f)/2$ on both sides gives

$$\begin{aligned} W(B_g) + W(R_f) + W(B_f) &\geq W(B_g)/2 + W(B_f)/2 + W(B_n)/2 \\ &= \frac{1}{2}(W(B_g) + W(B_f) + W(B_n)). \end{aligned}$$

The LHS is just $\Delta\Phi(t + \mathcal{H}(G)) - W(R_g)$, and the RHS is $\frac{1}{2}\Delta\Phi'(t + \mathcal{H}(G))$, so we obtain

$$\Delta\Phi(t + \mathcal{H}(G)) - W(R_g) \geq \frac{1}{2}\Delta\Phi'(t + \mathcal{H}(G)),$$

and as $W(R_g) \geq 0$ the claim of $\Delta\Phi(t + \mathcal{H}(G)) \geq \frac{1}{2}\Delta\Phi'(t + \mathcal{H}(G))$ follows. This concludes the proof of this lemma. □

Now we use the above lemma and the following *Drift Theorem* to prove the main result of this section.

Theorem 6 (Theorem 2. [16])**.** *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum s_{min}. Let $\{V(t)\}_{t \in \mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random variable that denotes the first point in time $t \in \mathbb{N}$ for which $V(t) = 0$. Suppose that there exists a constant $\delta > 0$ such that*

$$\mathbb{E}\left[V(t) - V(t+1) \mid V(t) = s\right] \geq \delta s \tag{6}$$

holds for all $s \in S$ with $\mathbb{P}\left[V(t) = s\right] > 0$. Then for all $s_0 \in S$ with $\mathbb{P}\left[V(0) = s_0\right] > 0$,

$$\mathbb{E}\left[T \mid V(0) = s_0\right] \leq \frac{1 + \ln(s_0/s_{min})}{\delta}. \tag{7}$$

Intuitively, Theorem 6 gives an upper bound on the expected run-time of a sequence of random variables, defined on a finite set, with a bounded decline in expectation.

Theorem 7. *Assume $Tr = W/n + 2w_{max}$. Let $\mathcal{H}(G)$ be the hitting time of the random walk on G with uniform stationary distribution. Let T be the first time it takes at which all tasks are allocated. Then*

$$\mathbb{E}[T] = O(\mathcal{H}(G) \cdot \ln(W)).$$

Proof. Since the maximum potential is bounded by W from above, this follows from the potential drop of Lemma 6 together with the Drift Theorem provided in Theorem 6 with parameters $\delta = 1/4$, $s_0 \leq W$, $s_{min} = w_{min} = 1$ and by considering intervals of length $2\mathcal{H}(G)$. □

The next observation shows that the bound of Theorem 7 is tight for uniform tasks. For weighted tasks the bound is not tight because the total weight W is super-polynomial in m.

Observation 8. *There is a class of graphs such that the resource-based protocol converges to a balanced state in an expected number of steps of $\Omega(\mathcal{H}(G) \cdot \log m)$ for tight thresholds.*

Proof. The proof follows the same line of argument as the proof of Theorem 3.7 in [2]. Instead of two cliques glued together with k edges we use the following graph G. Let G consist of a clique K of $n-1$ nodes and one single node u. This single node is connected to exactly k nodes of the clique for some $k < n$. The hitting time of this graph is $\Theta(n^2/k)$. Initially, we distribute the tasks on nodes of K in such a way that all nodes in K have a load of W/n. The remaining tasks are distributed on an arbitrary node of K. By using the same arguments as in [2], the required time becomes

$$\Omega(\mathcal{H}(G) \cdot \log(m/n)) = \Omega(\mathcal{H}(G) \cdot \log(m)).$$

□

6. User-Controlled Migration

In this section, we consider the user-controlled protocol for complete graphs. Before defining the protocol, which generalizes the protocol of [1] to the case of weighted tasks, we first present some necessary definitions.

We assume that the resources store the tasks in a stack data structure. Recall, the *height* $h_r^i(t)$ of task i in resource r is the sum of the weights of all tasks that are positioned below i. We say task i is *cutting the threshold* if $h_r^i(t) < \mathrm{Tr}$ and $h_r^i(t) + w_i > \mathrm{Tr}$. Then the potential of an overloaded resource r is called $\phi_r(t)$ and counts the weight of the task which is cutting the threshold (if there is any) plus the weights of the tasks which are above the threshold. The potential of a non-overloaded resource is zero. The potential $\Phi(t)$ at step t is defined as

$$\Phi(t) = \sum_{i=1}^{n} \phi_i(t).$$

Let $x_r(t)$ be the load (total weight of the tasks in r) of resource $r \in [n]$ at time step t. Also let $b_r(t)$ be the number of the tasks in r at time step t. The user-controlled protocol works as follows. Tasks leave an overloaded resource with a probability of $\alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}$ and move to a randomly chosen resource. Therefore, we assume that tasks know α, ϕ_r, w_{max} (or an estimate), and b_r.

Algorithm 6.1 User-Controlled

for all All users do in parallel **do**
  Let $r(i)$ denote the resource i is currently allocated to.
  if $x_{r(i)} > \mathrm{Tr}$ **then**
    With probability $\alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}$ migrate to resource chosen uniformly at random.

6.1. Above-Average Thresholds

To analyze the potential change we consider tasks leaving one after the other starting with all tasks leaving Resource 1, then Resource 2, and so on. For every resource we consider the tasks in the order of increasing heights. If it is clear from the context and the time step t is fixed, we will just write ϕ_r, b_r, and x_r.

The potential $\Phi(t+1)$

- decreases by w_i for every task i which was above the threshold ($h_r(t)+w_i > \mathrm{Tr}$) and migrates to a resource r' such that $h_{r'}(t+1) + w_i \leq \mathrm{Tr}$
- does not change for every task i which was above the threshold ($h_r(t) + w_i > \mathrm{Tr}$) and migrates to a resource r' such that $h_{r'}(t+1) + w_i > \mathrm{Tr}$
- increases by w_i for every task i which was below the threshold ($h_r(t)+w_i \leq \mathrm{Tr}$) and migrates to a resource r' such that $h_{r'}(t+1) + w_i > \mathrm{Tr}$.

For ease of presentation, the potential change caused by task i leaving r is accounted for at resource r as opposed to at the resource r'. First observe the following.

Observation 9. *Let t be an arbitrary time step and $\phi_r(t) > 0$. Then the number of tasks required to leave r such that $x_r(t+1) < Tr$ is at least*

$$\psi_r = \lceil \phi_r / w_{max} \rceil.$$

We now calculate the one-step potential change. First we assume $w_{min} = w_{max} = 1$, then the potential change can be estimated in the following way.

$$\mathbb{E}\left[\Delta\phi_r | i \text{ tasks leave}\right] \geq \begin{cases} \frac{\varepsilon}{1+\varepsilon} \cdot i & \text{if } i \leq \psi_r \text{ tasks leave} \\ \psi_r - i & \text{if } i > \psi_r \text{ tasks leave.} \end{cases}$$

In the first case fewer than ψ_r tasks leave, meaning that the potential decreases. Note that it does not matter if tasks above the threshold or below the

threshold leave since $w_{max} = w_{min}$, but also since $i \leq \psi_r$ which is a key insight for the case where $w_{max} \neq w_{min}$. The second case is pessimistic and assumes (i) that the ψ_r tasks leaving from above the threshold move to an overloaded resource, and (ii) that an additional $(i-\psi_r)$ tasks leave from below the threshold and move to a resource with a load $\geq$ Tr such that they are above the threshold.

From Lemma 1 it follows that there is a fraction of $\varepsilon/(1+\varepsilon) \cdot n$ resources which can accept additional tasks and which are still not overloaded at the end of a round. Hence, the probability that a leaving task decreases the potential Φ is at least $\varepsilon/(1+\varepsilon)$.

In the weighted setting, the potential decreases if $i \leq \psi_r$ tasks leave and land on an underloaded resource. Since every task leaves with the same probability we can assume that the expected size of a leaving task is x_r/b_r. If $i > \psi_r$ tasks leave resource r, then we forget about the tasks that move from above the threshold again and assume a potential increase by at most $(i-\psi_r) \cdot x_r/b_r$. Hence, in the weighted setting we have the following bounds on the potential change.

$$\mathbb{E}\left[\Delta\phi_r | i \text{ tasks leave}\right] \geq \begin{cases} \frac{\varepsilon}{1+\varepsilon} \cdot i \cdot \frac{x_r}{b_r} & \text{if } i \leq \psi_r \text{ tasks leave} \\ (\psi_r - i)\frac{x_r}{b_r} & \text{if } i > \psi_r \text{ tasks leave.} \end{cases}$$

We note that for the first case, due to the definition of ψ_r, the potential decreases in expectation by $\frac{x_r}{b_r}$ for every leaving task even if the height of this task is below the threshold.

In the next lemma we use this observation to calculate the one-step potential drop.

Lemma 10. *Let* $\alpha = \frac{\varepsilon}{120(1+\varepsilon)}$. *Assume* $Tr = (1+\varepsilon) \cdot W/n + w_{max}$ *and* $\Phi(X(t)) > 0$. *We have*

$$\mathbb{E}\left[\Delta\Phi(t+1) \mid X(t) = x(t)\right] \geq \frac{1}{2} \cdot \frac{\varepsilon}{1+\varepsilon} \cdot \Phi(X(t)).$$

Proof. Let $p_r(i)$ denote the probability that *exactly* i tasks leave resource r. We emphasize that all tasks on a given resource have the same probability to leave. For $1 \leq j \leq b_r$ we define a Bernoulli random variable $Y_r(j)$ which is 1 if task j on resource r leaves and 0 otherwise.

$$\begin{aligned} \mathbb{E}\left[\Delta\phi_r(t+1) \mid X(t) = x(t)\right] &= \sum_{i=0}^{b_r} \mathbb{E}\left[\Delta\phi_r | i \text{ tasks leave}\right] \cdot p_r(i) \\ &\geq \sum_{i=1}^{\psi_r} \frac{\varepsilon}{1+\varepsilon} \cdot i \cdot \frac{x_r}{b_r} \cdot p_r(i) - \sum_{i=\psi_r+1}^{b_r} i \cdot \frac{x_r}{b_r} \cdot p_r(i) \\ &\geq \frac{x_r}{b_r} \sum_{i=1}^{b_r} \frac{\varepsilon}{1+\varepsilon} \cdot i \cdot p_r(i) - 2\frac{x_r}{b_r} \sum_{i=\psi_r+1}^{b_r} i \cdot p_r(i) \end{aligned}$$

$$= \frac{x_r}{b_r}\frac{\varepsilon}{1+\varepsilon} \cdot \mathbb{E}\left[\sum_{i=j}^{b_r} Y_r(j)\right] - 2 \cdot \frac{x_r}{b_r} \sum_{i=\psi_r+1}^{b_r} i \cdot p_r(i),$$

where for the last step we used the fact that all tasks move with the same probability. Since

$$\mathbb{E}\left[Y_r(j)\right] = \alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r},$$

we get $\mathbb{E}\left[\sum_{i=j}^{b_r} Y_r(j)\right] = \alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil$. We now bound $\sum_{i=\psi_r+1}^{b_r} i \cdot p_r(i)$. We first observe that the term is maximized for small values of ϕ_r. Since $\psi_r \geq 1$ we derive

$$\begin{aligned}
\sum_{i=\psi_r+1}^{b_r} i \cdot p_r(i) &= \sum_{i=\psi_r+1}^{b_r} i\binom{b_r}{i}\left(\alpha\left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}\right)^i \cdot \left(1 - \alpha\left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}\right)^{b_r - i} \\
&\leq \sum_{i=\psi_r+1}^{b_r} i\binom{b_r}{i}\left(\alpha\left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}\right)^i \\
&\leq \sum_{i=\psi_r+1}^{b_r} i\left(\alpha\frac{e \cdot b_r}{i} \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}\right)^i \\
&\leq \sum_{i=\psi_r+1}^{b_r} i\left(\alpha\frac{e \cdot b_r}{\left\lceil \frac{\phi_r}{w_{max}} \right\rceil}\left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}\right)^i \\
&\leq \sum_{i=2}^{\infty} i(e\alpha)^i \leq \sum_{i=2}^{\infty}(e^2\alpha)^i \leq \frac{(e^2\alpha)^2}{1-e^2\alpha} \leq 2(e^2\alpha)^2 \leq 30\alpha^2,
\end{aligned}$$

where the second-last inequality follows from the geometric series using that $e^2\alpha < 1$. For $\alpha = \frac{1}{120(1+\varepsilon)}$ we have

$$\begin{aligned}
\mathbb{E}\left[\Delta\phi_r(t+1) \mid X(t) = x(t)\right] &\geq \frac{1}{(1+\varepsilon)}\left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \alpha \cdot \frac{x_r}{b_r} - 60\alpha^2 \cdot \frac{x_r}{b_r} \\
&\geq \alpha \cdot \frac{1}{2(1+\varepsilon)} \cdot \frac{x_r}{b_r} \cdot \frac{(\phi_r)}{w_{max}} \\
&\geq \alpha \cdot \frac{1}{2(1+\varepsilon)} \cdot \frac{w_{min}}{w_{max}} \cdot \phi_r.
\end{aligned}$$

Summing over all resources r with $\phi_r > 0$ yields

$$\mathbb{E}\left[\Delta\Phi(t+1) \mid X(t) = x(t)\right] \geq \alpha \cdot \frac{\varepsilon}{2(1+\varepsilon)} \cdot \frac{w_{min}}{w_{max}} \cdot \Phi(X(t)).$$

□

We can now use the lemma above to show the following result for complete

graphs.

Theorem 11. *Consider the complete graph. Assume* $Tr = (1+\varepsilon)\cdot W/n + w_{max}$ *and* $\alpha = \frac{\varepsilon}{120\cdot(1+\varepsilon)}$. *Let* T *be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}[T] = 2 \cdot \frac{1+\varepsilon}{\alpha \cdot \varepsilon} \cdot \frac{w_{max}}{w_{min}} \cdot \log m.$$

Proof. The result is following from Lemma 10 together with Theorem 6 (Drift Theorem). □

6.2. Tight Threshold

In this section we show results for tight thresholds on complete graphs.

Theorem 12. *Consider the complete graph. Assume* $Tr = W/n + w_{max}$ *and* $\alpha \leq \frac{1}{120n}$. *Let* T *be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}[T] = \frac{2 \cdot n}{\alpha} \cdot \frac{w_{max}}{w_{min}} \cdot \log m.$$

Proof. It is easy to see that at any point in time there is at least one resource which can accept an additional task of any weight at most w_{max}. Therefore, by replacing $\varepsilon/(1+\varepsilon)$ with $1/n$ and setting $\alpha \ll \frac{1}{n}$ in the proof of Theorem 11 the result follows. □

7. Simulations

In this section, we show some simulation results for the user-controlled protocol and complete graphs. Note that our bounds for the resource-controlled protocols are tight. In our simulations we assume $w_{min} = 1$, $\varepsilon = 0.2$, and $\alpha = 1$. We also assume that all tasks are initially held by the same resource. We chose $w_{min} = 1$ since one can always rescale all weights. Furthermore, the choice $\alpha = 1$ (recall that α is factor decreasing the probability that an unsatisfied user moves to another resource) shows that the factor $\alpha = \frac{\varepsilon}{120(1+\varepsilon)}$ we require in the analysis is quite conservative. Each data point is obtained by averaging over 1000 trials. We're interested in the balancing time as a function of the input parameters, such as total ask weight W and w_{max}.

In Figure 1 we have two different task sizes, 1 or 50. The x-axis shows the total task weight W. Let k denote the number of tasks with weight $w_{max} = 50$ and $m(W,k) = W - k \cdot w_{\max}$ is the number of tasks with weight $w_{min} = 1$.

The simulation shows that the balancing time is proportional to the logarithm of $m(W,k) + k$. Hence, the results seems to be more or less independent of the number of big tasks.

Given the outcome of the first simulation, we now consider in Figure 2 the case where there is only one task with weight w_{max}. On the x-axis we have the number of tasks and on the y-axis the balancing time, normalized by $\log m$. We show results for different sizes of maximum ball weights. This simulation

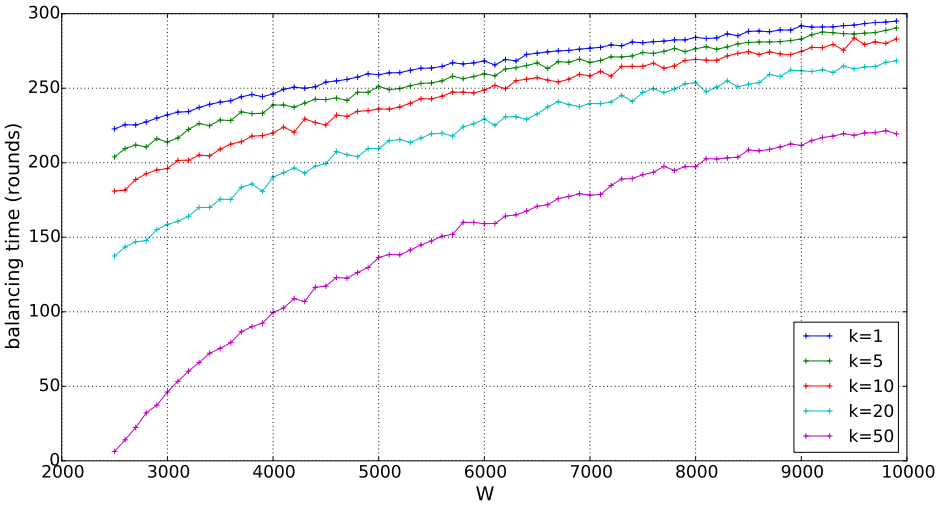

Figure 1: Balancing time in terms of k, where k denotes the number of tasks with weight $w_{max} = 50$ and $n = 1000$.

suggests that the upper bound of Theorem 11 is tight up to a constant factor; the balancing time of the simulation is logarithmic in m and almost linear in w_{max}/w_{min}.

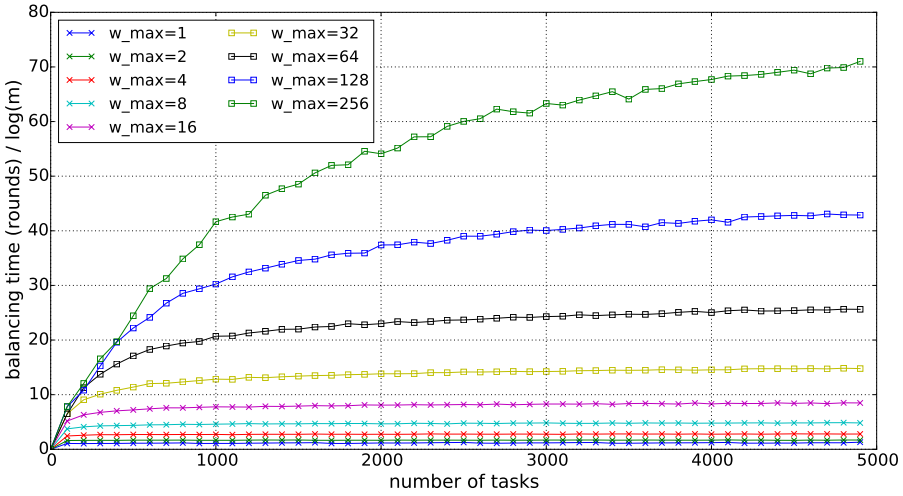

Figure 2: Normalized balancing time in terms of w_{max} with $n = 1000$.

Our simulations show that a small value of α is not necessary. We are leaving it as an open question whether the theoretical bound can also be shown for $\alpha = 1$.

8. Conclusion

We considered threshold-based load balancing protocols for weighted tasks. In the case of resource-based allocation we obtained tight bounds in the sense that there are families of graphs matching our bounds. In the case of user-based allocation we provided only upper-bounds for the complete graphs. It would be interesting to consider lower bounds in this setting. It might be interesting to study mixed protocols, which are both resource-based and user-based, as well as non-uniform thresholds.

9. Acknowledgements

We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

[1] H. Ackermann, S. Fischer, M. Hoefer, M. Schöngens, Distributed algorithms for QoS load balancing, Distributed Computing 23 (5-6) (2011) 321–330.

[2] M. Hoefer, T. Sauerwald, Brief announcement: Threshold load balancing in networks, in: 32nd Symposium on Principles of Distributed Computing PODC, ACM, Montreal, Canada, 2013, pp. 54–56, the full version is available at `https://arxiv.org/abs/1306.1402`.

[3] D. Aldous, J. A. Fill, Reversible Markov Chains and Random Walks on Graphs, unpublished. `http://www.stat.berkeley.edu/ aldous/RWG/book.html` (2002).

[4] M. Adler, S. Chakrabarti, M. Mitzenmacher, L. Rasmussen, Parallel randomized load balancing, Random Struct. Algorithms 13 (2) (1998) 159–188.

[5] P. Berenbrink, K. Khodamoradi, T. Sauerwald, A. Stauffer, Balls-into-bins with nearly optimal load distribution, in: Proceedings of the 25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA, 2013, pp. 326–335.

[6] P. Berenbrink, F. Meyer auf der Heide, K. Schröder, Allocating weighted jobs in parallel, in: Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures, SPAA, ACM, 1997, pp. 302–310.

[7] V. Stemann, Parallel balanced allocations, in: Proceedings. of the 8th ACM Symposium on Parallel Algorithms and Architectures, SPAA, ACM, New York, NY, USA, 1996, pp. 261–269.

[8] P. Berenbrink, T. Friedetzky, Z. Hu, R. Martin, On weighted balls- into-bins games, Theoretical Computer Science 409 (3) (2008) 511–520.

[9] K. Talwar, U. Wieder, Balanced allocations: the weighted case, in: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, 2007, pp. 256–265.

[10] P. Berenbrink, A. Czumaj, A. Steger, B. Vöcking, Balanced allocations: The heavily loaded case, SIAM J. Comput. 35 (6) (2006) 1350–1385.

[11] Y. Peres, K. Talwar, U. Wieder, The (1 + beta)-choice process and weighted balls-into-bins, in: Proceedings of the Twenty-First ACM-SIAM Symposium on Discrete Algorithms, SODA, 2010, pp. 1613–1619.

[12] P. Berenbrink, T. Friedetzky, L. Goldberg, P. Goldberg, Z. Hu, R. Martin, Distributed selfish load balancing, SIAM J. Comput. 37 (4) (2007) 1163–1181.

[13] P. Berenbrink, T. Friedetzky, I. Hajirasouliha, Z. Hu, Convergence to equilibria in distributed, selfish realelocation processes with weighted tasks, Algorithmica 62 (3-4) (2012) 767–786.

[14] C. Adolphs, P. Berenbrink, Distributed selfish load balancing with weights and speeds, in: ACM Symposium on Principles of Distributed Computing PODC, 2012, pp. 135–144.

[15] D. Levin, Y. Peres, E. Wilmer, Markov chains and mixing times, American Mathematical Society, 2006.

[16] B. Doerr, S. Pohl, Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet, in: Genetic and Evolutionary Computation Conference, GECCO, 2012, pp. 1317–1324.