On the Efficiency of Nearest Neighbor Load Balancing for Random Loads

Peter Sanders
Universität Karlsruhe, D-76128 Karlsruhe
E-mail: sanders@ira.uka.de

Abstract

Nearest neighbor load balancing algorithms like load diffusion are popular due to their simplicity, flexibility and robustness. In this paper we show that they are also asymptotically very efficient when a random rather than a worst case initial load distribution is considered. For processor loads described by independent random variables we show that diffusion needs $\Theta\left((\log n)^{2/d}\right)$ time to achieve good load balance on a d-dimensional mesh or torus network with n^d processors. We also argue that some but not all of the nearest neighbor algorithms known to perform better than diffusion in the worst case also perform better for random loads. In addition, we use the maximum norm for defining the quality of load balancing which has more direct implications for the execution time of the underlying application than previous definitions. Previouly known results for worst case instances are adapted to the new quality criterion.

1 Introduction

Load balancing is one of the key aspects of parallel computing. One interesting problem class in this context which is amenable to rigorous analysis is the following: A processor's (PE's) load can be accurately determined and arbitrarily subdivided into multiple pieces which can be independently solved on different PEs. Any portion of load can be communicated to a neighboring PE in unit time. A related and often more realistic model assumes the load to consist of discrete equally sized load units and transmission costs which are proportional to the number of units transferred.

There is a number of simple load balancing algorithms for distributed memory computers based on this model which only require local communication. Every PE cycles through identical iterations during which it determines its own load and that of its neighbors. Based on this local information the PE decides how much of its load is transmitted to each neighbor. The PEs are usually assumed to proceed synchronously (but refer to [16] for an asynchronous algorithm). Note that this is no real restriction because algorithms which only use local communication can always replace synchronous execution by local synchronzation [13]. Algorithms with local control are particularly attractive for dynamic load balancing applications, i.e., in settings where a load balancing process should run concurrently with an application which can generate or consume load at any time. Load balancing algorithms with global control have difficulties in this context because collecting

global information can take so long that it is outdated before the collection is finished. In [14] some results for a simple class of dynamic loads are derived but most papers use the easier to define static load balancing problem where a load balancer is judged by looking at the time required to balance an initial load distribution. However, this does not necessarily mean, that the intended application requires only static load balancing. It can be expected that a local load balancing algorithm which performs well in the static setting is also useful as a dynamic load balancer.

Perhaps the conceptually simplest local algorithm is *diffusive load balancing*. A constant fraction of the load difference between all neighboring PEs is exchanged at every step. By exploiting the fact that this algorithm can be modeled by a simple partial difference equation (The *diffusion* equation or *Poisson* equation) analytical results about the convergence rate in the worst case can be obtained for meshes, tori, hypercubes and (less accurately) for arbitrary networks [2, 1, 17, 19]. It turns out that diffusion can be rather slow in the worst case. For example on an n-PE linear array $\Omega(n^2)$ iterations are required in order to achieve approximate load balancing. The discrete analog to diffusion is to exchange a single load unit whenever there is a load difference [8]. Unfortunately this scheme often gets stuck in arbitrarily unbalanced configurations.

Interestingly, it can be considerably more efficient to exchange load with one neighbor after the other rather than with all neighbors at once. The corresponding *dimension exchange* algorithms are analyzed in [2, 7, 19]. The discrete version of dimension exchange is shown in [15] to quickly converge up to a maximal (global) load difference of d load units on a d-dimensional mesh and a simple extension even ensures the best possible convergence. Similar algorithms for expander graphs are considered in [4].

All in all, there are significant results about the worst case performance of nearest neighbor load balancing algorithms. But little progress has been made in understanding the gap between the worst case and the actual load patterns of real applications. Therefore, random loads are often used as a possibly more realistic model (e.g. [19]). There is empirical evidence that random loads are considerably simpler to load balance than the worst case bounds predict. Whether worst case loads or random loads are a better model certainly depends on the application one has in mind. But there is one important argument which favors the random load model: For worst case instances there are simple and efficient load balancing algorithms which use global control (often based on prefix-sum computations) and local algorithms may have difficulties competing with these algorithms. (For example, why bother with a diffusion load balancing algorithm which requires time $\Omega(n^2)$ on a mesh of n^d PEs, if a prefix sum followed by a routing operation can do it more accurately in time $\Theta(n)$.) But for situations which are simpler to balance, even the time needed to collect global information may exceed the time needed by a good local algorithm to balance the load.

Finally, we are at the point where the objective of this paper can be explained. The goal is to find closed form expressions for the performance of local load balancing algorithm for random initial loads. This may help to understand previous simulation results and to guide future studies. Furthermore, this gives a rigorous justification, why local load balancing can be more efficient than algorithms with global control.

The remainder of this paper is structured as follows. In Section 2 we give a more detailed description of the machine and load model. In particular, we define

a criterion for the quality of load balancing which is more directly related to the execution time of the overall application than previous criteria. Section 3 introduces the diffusion algorithm and adapts previous results in order to derive worst case lower bounds on the balancing time. The core part of this paper is Section 4 which analyzes the performance of diffusion for random problem instances and derives upper bounds for worst case instances. Because it turns out that all the main points can already be made for the linear array, we first restrict the proofs to this notationally simpler case. Section 5 then explains which modifications have to be made in order to extend the results to higher dimensional mesh and torus networks. The scope is further expanded in Section 6 where we discuss how the results might be transferred to algorithms which are superior to diffusion for worst case instances. Section 7 summarizes and discusses the results and identifies some possible future directions of research.

2 The model

The considered machine is a d-dimensional mesh consisting of n^d processors (PEs) which work synchronously. We will sometimes note how the results can be transferred for torus networks. Generalizations for meshes with unequal side length are also possible but are not considered here since we do not expect any new insights from that. Previous results indicate that the performance of load balancing is usually governed by the maximal side length [17, 19]. The PEs are numbered by an index set $\mathcal{I} := \{0, \ldots, n-1\}^d$.

The load at time t is described by the $|\mathcal{I}|$-dimensional *load vector* $\mathbf{l}(t)$. The load of a PE $l_i(t)$ is measured by the sequential execution time required to consume it. Whenever the intended time is clear form the context, the t is omitted. Let $l_{\text{avg}} := \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} l_i$ denote the average load. As explained in the introduction we analyze a static load balancing scenario where the initially given load $\mathbf{l}(0)$ is to be balanced while the sum of all PE loads (and therefore also l_{avg}) remains constant. The fully balanced situation can therefore be defined by the vector $\mathbf{l}_{\text{avg}} := (l_{\text{avg}}, \ldots, l_{\text{avg}})^T$. Let $\mathbf{e} := \mathbf{l} - \mathbf{l}_{\text{avg}}$ denote the deviation from the balanced state or *error vector*. Previous works have judged the quality of the achieved load balancing in terms of the Euclidean norm $\|\mathbf{e}\|_2 = \sqrt{\sum_{i\in\mathcal{I}} e_i^2}$ of $\mathbf{e}$. It is the "default" in mathematics and it is easier to handle than other measures. But we do not know of any immediate connection between $\|\mathbf{e}\|_2$ and the performance of the underlying application. We therefore use a different model:

Definition 1. The load is considered to be balanced when $\|\mathbf{e}\|_\infty = \max_{i\in\mathcal{I}} |e_i| \leq \varepsilon$ for some positive constant ε.

In particular, in a balanced situation the remaining parallel execution time required for consuming the load is at most $(1+\varepsilon)l_{\text{avg}}$ and for the next $(1-\varepsilon)l_{\text{avg}}$ time units all PEs can work productively without additional communication. The *balancing time* T_{bal} can then be defined as the smallest t for which $\mathbf{l}(t)$ is balanced. Algorithms with discrete loads have the additional constraint that $l_i(t)$ must always be an integer and that only one load unit per time step can be transmitted.

Random loads are modeled by considering the $l_i(0)$ to be independent, identically distributed, bounded random variables with expectation $\mathbf{E}l_i(0) = \bar{l}$ and $l_i(0) \leq \hat{l}$ such that $\hat{l}/\bar{l}$ is a constant. For example, the uniform distribution used in [8, 19] has $\hat{l} = 2\bar{l}$.

We use the following widely used definition of asymptotic behavior with high probability for our results (e.g. [10]):

Definition 2. $X \in \tilde{\mathrm{O}}(f(n))$ iff

$$\forall \beta \in \mathbb{R}_+ : \exists c \in \mathbb{R}_+ : \exists n_0 : \forall n \geq n_0 : \quad \mathbf{P}[X \leq cf(n)] \geq 1 - n^{-\beta} \ .$$

For example, the execution time of a load balancing algorithm is considerered to be of the order $T(n)$ with high probability if $cT(n)$ iterations suffice for all but a polynomially small fraction of all random instances.

We have $\mathbf{E}l_{\text{avg}} = \bar{l}$. For sufficiently large n, l_{avg} will be so close to $\bar{l}$ that the two quantities can be used almost interchangeably. In particular, the following Lemma allows us to analyze the load balancing time in terms of the deviation from the expected load rather than in terms of the deviation from the actual average load: (We omit the proof which is quite simple and rather uninteresting.)

Lemma 3. *For all $\varepsilon > 0$ there is a $\delta > 0$ such that if the time required to achieve $\left\|\mathbf{l} - \bar{\mathbf{l}}\right\|_\infty < \delta\bar{l}$ (with $\bar{\mathbf{l}} := (\bar{l}, \ldots, \bar{l})^T$) is in $\tilde{\mathrm{O}}(T(n))$ then the same time suffices to achieve $\|\mathbf{e}\|_\infty \leq \varepsilon l_{\text{avg}}$ with high probability.*

3 Diffusive load balancing

Synchronous diffusive load balancing can be defined by the simple rule:

$$l_i(t+1) := l_i(t) + \alpha \sum_{j \in \Delta(i)} (l_j - l_i) \ . \tag{1}$$

Where $\Delta(i)$ is the set of neighbors of PE i. A lower bound for the efficiency of diffusion can be derived from previous results:

Theorem 4. *The worst case balancing time for diffusion on d-dimensional meshes with n^d PEs is in $\Omega(n^2)$.*

Proof. Let ε denote any positive constant. Using [17, 19] it can be seen that there is a choice of $\mathbf{l}(0)$ such that $\|\mathbf{e}(0)\|_\infty = al_{\text{avg}}$ $(a > \varepsilon)$ and $\|\mathbf{e}(t)\|_\infty = |\lambda|^t \|\mathbf{e}(0)\|_\infty$ with $|\lambda| \in 1 - \Theta(1/n^2)$. ($\lambda$ is the subdominant eigenvalue of a matrix which can be used to define the diffusion rule and $\mathbf{l}(0)$ is $\mathbf{l}_{\text{avg}}$ plus an appropriate scaling of the corresponding eigenvector.) So, there is a constant c such that for sufficiently large n, $\|\mathbf{e}(t)\|_\infty \geq (1 - c/n^2)^t a \approx e^{-ct/n^2} a > \varepsilon$ for $t < \frac{\ln \frac{a}{\varepsilon}}{c} n^2$. □

4 The linear array

The analysis of diffusion for the linear array proceeds as follows. In Section 4.1 we derive a simple closed form formula for $l_i(t)$ as a linear combination of the coefficients of $\mathbf{l}(0)$. Two key observations make this possible. The linearity of diffusion implies that the behavior for a unit load already contains all the information we need. Furthermore, a special treatment of the border PEs 0 and $n-1$ can be avoided by introducing periodic boundary conditions. Section 4.2 establishes the main result of this section, namely that with high probability $T_{\text{bal}} \in \mathrm{O}((\log n)^2)$ for random instances. The main tool making this possible are Hoeffding bounds. Finally in sections 4.3 and 4.4 the missing lower bounds for random instances respectively upper bounds for the worst case are derived.

4.1 A closed form solution

To get the main idea for finding a closed form solution we start with some simplifications. We assume that there is an infinite number of PEs – one for each integer. Also, we fix the diffusion parameter α to $1/2$. This value is identified as optimal for the linear array in [19]. (For the torus it is optimal for odd n.) The diffusion rule (1) can now be written as

$$l_i(t+1) = \frac{l_{i-1}(t) + l_{i+1}(t)}{2} \ .$$

Furthermore, assume[1] $l_i(0) = [i = 0]$, i.e. initially there is only a unit load on PE 0. Then we get the following load development:

$$\begin{array}{lll}
\mathbf{l}(0) & = & (\ldots, \quad 0, \ 1, \ 0, \quad \ldots)^T \\
\mathbf{l}(1) & = 1/2 & (\ldots, \quad 0, \ 1, \ 0, \ 1, \ 0, \quad \ldots)^T \\
\mathbf{l}(2) & = 1/4 & (\ldots, \quad 0, \ 1, \ 0, \ 2, \ 0, \ 1, \ 0, \quad \ldots)^T \\
\mathbf{l}(3) & = 1/8 & (\ldots, \quad 0, \ 1, \ 0, \ 3, \ 0, \ 3, \ 0, \ 1, \ 0, \quad \ldots)^T \\
\mathbf{l}(4) & = 1/16 & (\ldots, 0, \ 1, \ 0, \ 4, \ 0, \ 6, \ 0, \ 4, \ 0, \ 1, \ 0, \ldots)^T
\end{array}$$

$\mathbf{l}(t)$ is basically the t-th row of a Pascal triangle. More precicely,

$$l_i(t) = 2^{-t}\binom{t}{\frac{t+i}{2}}[\text{even}(t+i)] \ .$$

Another useful observation is that $l_i(t)$ equals the probability that a particle starting a random walk at PE 0 at time 0 is at PE i at time t. Yet another useful interpretation is that $l_i(t)$ also equals 2^{-t} times the number of paths a particle can take to come from PE 0 to PE i in t steps.

The above result can be generalized for arbitrary initial load patterns by scaling, shifting and adding several unit-load solutions.

Theorem 5. $l_i(t) = 2^{-t} \sum_j \binom{t}{\frac{t+(i-j)}{2}} l_j(0)\,[\text{even}(t+(i-j))]$.

Proof. Instead of justifying the entire sequence of the above heuristic steps the theorem can be proved directly by induction over t using the well known relation $\binom{t}{k} + \binom{t}{k-1} = \binom{t+1}{k}$. □

This simple formula can also be used for a finite array if we introduce the periodic boundary condition $l_i := l_{g(i \bmod 2n)}$ with $g(i) = \min(i, 2n-i-1)$. (The ring topology can be treated similarly by using the periodic boundary condition $l_i := l_{i \bmod n}$.) So, instead of considering a finite linear array, we now use an infinite line of PEs with a periodic repetition of the initial load of the finite array alternating with its mirror image. Figure 1 shows the structure of this periodic pattern.

4.2 Upper bound for random instances

This section is devoted to the proof of the following theorem:

[1] We adopt the notation from [5] to define $[P] = 1$ if the predicate P ist true and $[P] = 0$ else.

Figure 1: Periodic boundary conditions for the mesh.

Theorem 6. *For random problem instances diffusion with parameter $\alpha = 1/2$ on an n PE linear array has*

$$T_{\text{bal}} \in \tilde{\mathrm{O}}\left((\log n)^2\right) \quad .$$

Our principal tool will be the following *Hoeffding bound* theorem which we cite from [6, Theorem 2.6.7] in a form slightly rewritten for our purposes:

Theorem 7 (Hoeffding bounds). *If $X_1, \ldots, X_n$ are independent random variables with $a_i \le X_i \le b_i$, then for $\varepsilon \mathbf{E}\sum_i X_i > 0$*

$$\mathbf{P}\left[\sum_i X_i > (1+\varepsilon)\mathbf{E}\sum_i X_i\right] \le \exp\left[-\frac{\varepsilon^2\left(\mathbf{E}\sum_i X_i\right)^2}{\sum_i (b_i - a_i)^2}\right] \quad . \tag{2}$$

Let β denote an arbitrary positive constant and let $t = c(\ln n)^2$ for a value of c still to be determined. Since $\mathbf{P}\left[\left\|\mathbf{l} - \bar{\mathbf{l}}\right\|_\infty > \varepsilon\bar{l}\right] \le n\mathbf{P}\left[\left|l_i(t) - \bar{l}\right| > \varepsilon\bar{l}\right]$ for any fixed i, it suffices to show that $\mathbf{P}\left[\left|l_i(t) - \bar{l}\right| > \varepsilon\bar{l}\right] \le n^{-\beta-1}$. We consider the cases $l_i(t) > (1+\varepsilon)\bar{l}$ and $l_i(t) < (1-\varepsilon)\bar{l}$ separately:

Lemma 8. $\mathbf{P}\left[l_i(t) > (1+\varepsilon)\bar{l}\right] \le \frac{1}{2}n^{-\beta-1}$.

Proof. From Theorem 5 we know that $l_i(t) = \sum_j \binom{t}{\frac{t+(i-j)}{2}} 2^{-t}\left[\text{even}(t+(i-j))\right] l_j(0)$. By appropriately reordering this sum it can be written as $l_i(t) = \sum_k \binom{t}{k} 2^{-t} l_{j_k}(0)$ and for $t < n$ the nonzero terms are independent random variables in the range $\left[0, 2^{-t}\binom{t}{k}\hat{l}\right]$. (Note that this also holds if PE i is near the border since for $t < n$ there are never any PEs whose initial load contributes to two nonzero terms of the sum.) Furthermore, $\mathbf{E}l_i(t) = \bar{l}$. Therefore we can use the Hoeffding-bound (2) to see that

$$\mathbf{P}\left[l_i(t) > (1+\varepsilon)\bar{l}\right] \le \exp\left[-\frac{\varepsilon^2\bar{l}^2}{4^{-t}\hat{l}^2\sum_k\binom{t}{k}^2}\right] \quad .$$

We now use the relation $\sum_k \binom{t}{k}^2 = \binom{2t}{t}$ (For example [5, Equation 5.23]) and the Stirling approximation for the binomial coefficient $\binom{2t}{t} \le 4^t\sqrt{\frac{1}{\pi t}}$.

$$= \exp\left[-\frac{\varepsilon^2\bar{l}^2 4^t}{2^{-2t}\hat{l}^2\binom{2t}{t}}\right] \le \exp\left[-\frac{\varepsilon^2\bar{l}^2\sqrt{\pi t}}{\hat{l}^2}\right] = n^{-\frac{\varepsilon^2\bar{l}^2\sqrt{c\pi}}{\hat{l}^2}}$$

$$\le \frac{1}{2}n^{-\beta-1} \text{ for } c \ge \frac{(\beta+1+1/\log n)^2\hat{l}^4}{\pi\varepsilon^4\bar{l}^4} \quad .$$

□

Lemma 9. $\mathbf{P}\left[l_i(t) < (1-\varepsilon)\bar{l}\right] \le \frac{1}{2}n^{-\beta-1}$.

Proof. Analogous to the proof of Lemma 8. The Hoeffding bound can be used to bound deviations below the expected value by substituting $\varepsilon \leftarrow -\varepsilon$ and $X_i \leftarrow -X_i$. □

Putting lemmata 8 and 9 together we can conclude that $\mathbf{P}\left[\left|l_i(t) - \bar{l}\right| > \varepsilon\bar{l}\right] \le n^{-\beta-1}$. ■

4.3 Lower bound for random instances

Theorem 10. *There are distributions of $l_j(0)$ such that for most random instances of diffusive load balancing the execution time is in $\Omega((\log n)^2)$.*

Proof. Consider the distribution $\mathbf{P}\left[l_j(0) = \hat{l}\right] = \bar{l}/\hat{l}$, $\mathbf{P}[l_j(0) = 0] = 1 - \bar{l}/\hat{l}$. Fix any constant $\gamma < 1/\log(\hat{l}/\bar{l})$. (Mentally) subdivide the array into equal sized intervals of length $\gamma \log n$.[2] Using elementary calculations, it can be proved that with high probability there is at least one interval in which every PE receives load $\hat{l}$. Balancing the load of this highly loaded interval can be shown to require $\Omega((\log n)^2)$ steps (using similar techniques we are using in Theorem 14 to show worst case upper bounds). □

The proof can also be adapted to other load distributions like the uniform distribution. The only requirement is that there is a constant nonzero probability that $l_i(0) > (1+\delta)l_{\text{avg}}$ for some constant $\delta > \varepsilon$.

4.4 Upper bound for worst case instances

We start with some simple observations which constitute the building blocks of the proof. First, "moving" initially present load closer to a PE can only increase its load at time t. (Note that this operation can produce nonperiodic load patterns but we do not claim that the modified pattern corresponds to a legal initial situation.)

Lemma 11. *For any t, substituting $l_j(0) \leftarrow l_j(0) - a$ and $l_k(0) \leftarrow l_k(0) + b$ can only increase $l_i(t)$ if $b \geq a$, $k - j$ is even and $|k - i| \leq |i - j|$.*

Lemma 12. $\|\mathbf{e}(t)\|_\infty$ *is a decreasing function.*

Lemma 13. $l_i(t)$ *does only depend on* $l_j(0)$ *if* $\text{even}(t) = \text{even}(i - j)$.

Theorem 14. *The worst case balancing time for diffusion on the linear array with diffusion parameter $\alpha = 1/2$ is in $O(n^2)$.*

Proof. We again look at the maximum load first. Consider any PE i and any legal initial load pattern. Due to the monotonicity of $\|\mathbf{e}(t)\|_\infty$ we can restrict ourselves to even times t without loss of generality. Due to Lemma 13 we can therefore disregard all positions j with $\text{odd}(i - j)$.[3] By exploiting the symmetry properties of $\mathbf{l}(0)$ and repeatedly applying Lemma 11 it can be shown that $l_i(t)$ can only get larger if we transform $\mathbf{l}(0)$ to $l_j(0) := l_{\text{avg}}[i \neq j \wedge \text{even}(i - j)] + 2nl_{\text{avg}}[i = j]$. (Strictly speaking we only need to redefine $l_j(0)$ for $|i - j| \leq t$.) Figure 2 shows this transformation operations. Therefore

$$\begin{aligned} l_i(t) &\leq 2^{-t} l_{\text{avg}} \left(\sum_{j \neq i} \binom{t}{\frac{t+(i-j)}{2}} [\text{even}(t + (i - j))] + 2n \binom{t}{t/2} \right) \\ &\leq l_{\text{avg}} \left(1 + 2n 2^{-t} \binom{t}{t/2} \right) \end{aligned}$$

[2] Slightly tighter results are possible by selecting subsets of PEs which include only every other PE in an interval of length $2\gamma \log n$.

[3] Note that we do not lose any information by doing that. For our periodic input patterns all loads with $\text{odd}(i - j)$ will also appear as a "mirror image" at an even distance.

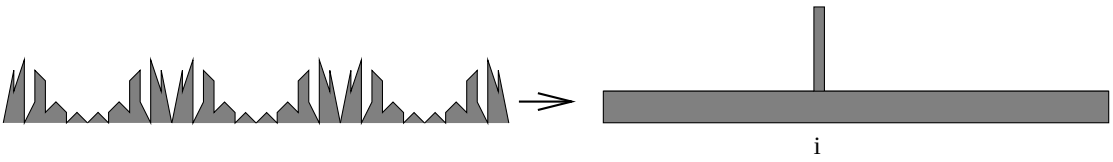

Figure 2: Transformation of initial load for worst case bound.

using the Stirling approximation for $\binom{2t}{t}$ again, we get

$$l_i(t) \leq l_{\text{avg}} \left(1 + 2n\sqrt{\frac{2}{t\pi}}\right) \leq l_{\text{avg}}(1+\varepsilon) \text{ for } t \geq \frac{8n^2}{\pi\,\varepsilon^2} \ .$$

To bound the behavior of the minimum load we can construct a similar set of lemmata and transform $\mathbf{l}(0)$ to $l_j(0) := l_{\text{avg}}\,[i \neq j \wedge \text{even}(i-j)] - 2n l_{\text{avg}}\,[i=j]$. Then the "hole" in $\mathbf{l}(0)$ will produce a load deficit smaller than $\varepsilon l_{\text{avg}}$ for some $t \in \mathrm{O}(n^2)$. (Note that the negative load at PE 0 causes no problems because the transformed configuration need not be a legal initial load configuration.) □

5 Higher dimensions

Some generalizations are immediate. The optimal diffusion parameter for a d-dimensional mesh is $\alpha = \frac{1}{2d}$ [19]. By introducing periodic boundary conditions for all dimensions we can write

$$l_i(t+1) = \frac{1}{2d}\sum_{k=1}^{d}(l_{i+u_k}(t) + l_{i-u_k}(t))$$

(where u_k denotes the k-th unit vector). Furthermore,

$$l_i(t) = \sum_{j\in\mathbb{Z}^d} \omega_{i-j}(t) l_j(0)$$

and the coefficients ω_{i-j} are independent of the load. In order to determine ω_k, it is again sufficient to consider the development of a unit load at PE 0. The probabilistic and combinatorial interpretation of this setting is still applicable. $\omega_{(k_1,\ldots,k_d)}$ is zero if $t + k_1 + \cdots + k_d$ is odd. For even t, we have $\omega_k \leq \omega_0$ for all k. Still, the coefficients ω_k are more difficult to determine and manipulate now. Only for $d = 2$ we know a relatively simple closed form:

Theorem 15. *For* $d = 2$, $\omega_{(x,y)}(t) = 4^{-t}\binom{t}{\frac{t+x+y}{2}}\binom{t}{\frac{t+x-y}{2}}\,[\text{even}(t+x+y)]$.

Proof. By induction over t. □

Fortunately we do not really need to know all the ω_k if we have a good estimate for ω_0.

Lemma 16. $\omega_0(t) \in \mathrm{O}\left(t^{-d/2}\right)$.

Proof. Assume without loss of generality that t is divisible by $2d$. In the probabilistic interpretation ω_0 is the "return to origin" probability of a d-dimensional random walk. Following [3, Section XIV.7] we get

$$\omega_0 = 2^{-t}\binom{t}{t/2} \sum_{k_1+\cdots+k_d=t/2} \left(d^{-t/2}\frac{(t/2)!}{\prod_i k_i!}\right)^2 \quad .$$

Using $\sum a_i^2 \leq (\max a_i)(\sum a_i)$ and the fact that the terms inside $(\cdot)^2$ constitute a multinomial distribution (and therefore sum to 1) we get

$$\leq 2^{-t}\binom{t}{t/2} d^{-t/2} \max_{k_1+\cdots+k_d=t/2} \frac{(t/2)!}{\prod_i k_i!} = 2^{-t}\binom{t}{t/2} d^{-t/2}\frac{(t/2)!}{(\frac{t}{2d})!^d}$$

and using the Stirling approximation

$$\leq \sqrt{\frac{2}{\pi t}}\frac{d^{-t/2}\sqrt{\pi t}(t/2)^{(t/2)}e^{-t/2}(1+\Theta(1/t))}{\left(\sqrt{\pi t/d}(\frac{t}{2d})^{(\frac{t}{2d})}e^{-\frac{t}{2d}}\right)^d} = \sqrt{2}\,(\pi t/d)^{-d/2}\,(1+\Theta(1/t)) \quad .$$

□

In particular, the proof of Theorem 14 can now be adapted for arbitrary d:

Theorem 17. *The worst case balancing time for diffusion on the d-dimensional mesh with diffusion parameter $\alpha = \frac{1}{2d}$ is in $O(n^2)$.*

For random instances we encounter a new problem. If i is near one of the corners of the mesh then the random variables $l_j(0)$ which contribute to $l_i(t)$ are not all independent even for small t. The reason is that due to the reflections implicit in the periodic boundary conditions there may be up to d indices which are determined by the same value (for $t < n$). But $l_i(t)$ is nevertheless the sum of $\Theta\left(t^d\right)$ independent random variables so that the Hoeffding bound can be applied. The expectation of the sum of these variables is still $\bar{l}$. Using a similar trick as in the proof of Lemma 16 the sum over the squares of the bounds of these random variables is not larger than $\hat{l}^2 d\omega_0$. We can now modify the proof of the one-dimensional case (Theorem 6) to conclude that $\mathbf{P}\left[l_i(t) > (1+\varepsilon)\bar{l}\right] \leq e^{-\Theta\left(t^{d/2}\right)}$ so that there is a $t \in O\left((\log n)^{2/d}\right)$ which ensures that with high probability no PE has load more than $(1+\varepsilon)\bar{l}$. Despite of this admittedly imprecise argument we dare to conclude:

Theorem 18. *For random problem instances diffusion with parameter $\alpha = \frac{1}{2d}$ on the d-dimensional mesh has*

$$T_{\text{bal}} \in \tilde{O}\left((\log n)^{2/d}\right) \quad .$$

The corresponding theorem for torus networks is even simpler to prove because there is no trouble with dependent variables. The lower bound for random instances can be proved as in the one-dimensional case by showing that with high probability there is a highly loaded subcube of side length $\Omega\left((\log n)^{1/d}\right)$:

Theorem 19. *There are distributions of $l_j(0)$ such that for most random instances of diffusive load balancing the execution time is in $\Omega\left((\log n)^{2/d}\right)$.*

6 Other algorithms

As explained in the introduction, diffusion is a relatively slow algorithm for worst case instances. Algorithms with global control and the dimension exchange algorithm are more efficient. For random instances global algorithms still need time $\Omega(n)$ and diffusion is now asymptotically superior. For dimension exchange, the situation is more complicated but also more interesting:

Theorem 20. *(Generalized) dimension exchange load balancing with the optimal parameters from [19] needs load balancing time $T_{\text{bal}} \in \Omega(n)$ both in the worst case and for random instances.*

Proof. In every step of the optimized algorithm from [19] a PE gives a fraction in $1 - \Theta(1/n)$ of its load to its current communication partner. Using a calculation analogous to the proof of Theorem 4 we can conclude that it takes $\Omega(n)$ steps until a piece of load of size $(1+a)l_{\text{avg}}$ $(a > \varepsilon)$ can be reduced below $(1+\varepsilon)l_{\text{avg}}$. □

We conjecture, that the Ω in the above proof can be replaced by a Θ. For a heuristic interpretation it is again useful to employ the analogy of the random movement of a particle. Diffusion corresponds to a completely symmetric random walk and this is the reason why it is so slow in the worst case – it takes time $\Theta(k^2)$ to get k steps away from the origin on the average. Dimension exchange with a very asymmetric parameter is like adding a lot of "stiffness" to the motion of the particle such that a random motion only emerges on large scales.[4] But for random instances this is disadvantageous because it is sufficient to balance $O(\log n)$ neighboring PEs in order to get global balance. This also gives a hint how dimension exchange could be improved for random instances:

Conjecture 21. *Dimension exchange with a parameter in $1 - \Theta(1/\log n)$ has $T_{\text{bal}} \in \tilde{O}\left((\log n)^{1/d}\right)$ for random instances.*

The idea is to adapt the stiffness of motion to the scale in which load balancing is possible. We then get an algorithm which is again more efficient than diffusion. More generally, we expect that many load balancing algorithms have an asymptotic performance for random instances which is identical to the worst case performance on $O(\log n)$ PEs (possibly present parameters need to be tuned for this machine size). The reason is that it can be proved that the network can be cut into subnetworks of size $O(\log n)$ without making load balancing impossible. Therefore it would be sufficient to run the algorithm locally on the subnetworks. It does not seem to be a very far fetched assumption that an algorithm which is identical to the one it was derived from, with the only difference, that its communication abilities are pruned, performs no better on the average than the original algorithm. For example, the discrete dimension exchange algorithm described in [15] presumably needs load balancing time $T_{\text{bal}} \in \tilde{O}\left(\|\mathbf{e}(0)\|_\infty (\log n)^{1/d}\right)$ for random instances. This is optimal in the sense that with high probability a random instance cannot be balanced asymptotically faster due to sheer bandwidth limitations. (Remember that in the discrete variant of our model only one load unit per iteration can be moved.)

[4] Essentially we use the same strategy when we stir a cup of tea in order to "balance" the sugar.

7 Conclusions

Simple local load balancing algorithms can be much more efficient for random instances than in the worst case. For example, diffusion requires time $T_{\text{bal}} \in \Theta(n^2)$ to achieve approximate load balance on a d dimensional mesh with n^d PEs. For random problem instances $\Theta\left((\log n)^{2/d}\right)$ steps are sufficient with high probability. These results can also be adapted to torus networks.[5] In contrast, algorithms which employ some kind of global control often have the same asymptotic performance for worst case and random instances and are therefore inferior for situations for which random instances are a more appropriate model. Nevertheless, diffusion is still slow compared to other algorithms like the best variants of dimension exchange. However, in the continuous case the parameters of dimension exchange need to be tuned depending on the machine size and the class of problems to be balanced. We see the role of diffusion mainly as a simple and robust algorithm which can be used as a "first try". Furthermore, its simplicity makes it a good choice for analyzing fundamental properties of load balancing.

There remains a lot of work to be done. One could consider different interconnection networks or asynchronous operation. There are also more sophisticated algorithms about which little analytical results are known. For example, the gradient model load balancing approach [11, 12] might be interesting because it is a mixture between global and local operation. Regarding the goal of this paper the largest gap is the discrepancy between the load model (static load balancing of random instances) and the intended use (dynamic load balancing for real applications). What we would really like to have are dynamic load models which are at the same time meaningful for a variety of applications and analytically tractable. However, we believe that this will at least be very difficult because the more accurate a model gets, the more narrow is the domain of applications for which it yields meaningful results. For example, consider two of the applications for which nearest neighbor balancing has been used successfully: For particle simulation [8] it is crucial that interacting particles are at nearby PEs. While this property is easy to maintain on one-dimensional networks if nearest neighbor load balancing is used, the situation is much more complicated for higher-dimensional networks. For best first branch-and-bound (e.g. [9, 18]) we are not so much interested in a balanced distributions of the number of search tree nodes but we want to have some nodes with close to optimal evaluation on every PE. We believe that there will always be room for simplified abstract models because they make it possible to judge the relative merits of different algorithms for a wide spectrum of settings.

References

[1] J. E. Boillat. Load balancing and Poisson equation in a graph. *Concurrency: Practice and Experience*, 2(4):289–311, 1990.

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

[3] W. Feller. *An Introduction to Probability Theory and its Applications*. Wiley, 3rd edition, 1968.

[5] This is particularly easy for odd n. For even n, convergence in the worst case is only possible for $\alpha < \frac{1}{2d}$. For example, our calculations can be modified for the case $\alpha = \frac{1}{4d}$.

[4] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, T. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. In *ACM Symposium on the Theory of Computing*, 1995.

[5] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison Wesley, 1992.

[6] M. Hofri. *Probabilistic Analysis of Algorithms.* Springer, 1987.

[7] G. Horton. A multi-level diffusion method for dynamic load balancing. *Parallel Computing*, 19:209–218, 1993.

[8] G. A. Kohring. Dynamic load balancing for parallelized particle simulations on MIMD computers. *Parallel Computing*, 21:683–693, 1995.

[9] N. Kuck, M. Middendorf, and H. Schmeck. Generic branch-and-bound on a network of transputers. In R. Grebe et al., editors, *Transputer Applications and Systems*, pages 521–535. IOS Press, 1993.

[10] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17:157–205, 1994.

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

[12] R. Lüling, B. Monien, and F. Ramme. Load balancing in large networks: A comparative case study. In *3th IEEE Symposium on Parallel and Distributed Processing.* IEEE, 1991.

[13] K. Nakamura. Asynchronous cellular automata and their computational ability. *Systems, Computers, Controls*, 5:58–66, 1974.

[14] X. Qian and Q. Yang. An analytical model for load balancing on symmetric multiprocessor systems. *Journal of Parallel and Distributed Computing*, 20:198–211, 1994.

[15] P. Sanders and T. Worsch. Konvergente lokale Lastverteilungsverfahren und ihre Modellierung durch Zellularautomaten. In *PARS Workshop*, volume 14 of *PARS Mitteilungen*, pages 285–291, 1995.

[16] J. Song. A partially asynchronous and iterative algorithm for distributed load balancing. *Parallel Computing*, 20:853–868, 1994.

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

[18] S. Tschöke, M. Räcke, R. Lüling, and B. Monien. Solving the traveling salesman problem with a parallel branch-and-bound algorithm on a 1024 processor network. Technical report, Universität Paderborn, 1994.

[19] C. Xu and F. C. Lau. The generalized dimension exchange method for load balancing in k-ary n-cubes and variants. *Journal of Parallel and Distributed Computing*, 24(1):72–85, 1995.