Periodic Load Balancing on the N-Cycle: Analytical and Experimental Evaluation

Christian Rieß and Rolf Wanka

Computer Science Department, University of Erlangen-Nuremberg, Germany
sichries@informatik.stud.uni-erlangen.de,rwanka@cs.fau.de

Abstract. We investigate the following very simple load-balancing algorithm on the N-cycle (N even) which we call *Odd-Even Transposition Balancing* (OETB). The edges of the cycle are partitioned into two matchings canonically. A matching defines a single step, the two matchings form a single round. Processors connected by an edge of the matching perfectly balance their loads, and, if there is an excess token, it is sent to the lower-numbered processor. The difference between the real process where the tokens are assumed integral and the idealized process where the tokens are assumed divisible can be expressed in terms of the local divergence [1]. We show that Odd-Even Transposition Balancing has a local divergence of $N/2 - 1$. Combining this with previous results, this shows that after $O(N^2 \log(KN))$ rounds, any input sequence with initial imbalance K is perfectly balanced. Experiments are presented that show that the number of rounds necessary to perfectly balance a load sequence with imbalance K that has been obtained by pre-balancing a random sequence with much larger imbalance is significally larger than the average number of rounds necessary for balancing random sequences with imbalance K.

1 Introduction

Background. In the standard abstract formulation of load balancing in a distributed network, processors are modeled as the vertices of a graph and links between them as edges. Each processor initially has a collection (called *load*) of unit-size, integral jobs (called *tokens*). The object is to balance the number of tokens at each processor by transmitting tokens along edges according to some local scheme. This problem has obvious applications to job scheduling and other coordination tasks in parallel and distributed systems. It also arises in the context of finite element computations, and in simulations of physical phenomena.

One load-balancing approach is the *dimension exchange* paradigm [2, 3], where the network is decomposed into a sequence $M_1, \ldots, M_d$ of perfect matchings. The edges of the matching are oriented. We write $[i{:}j]$ for a single edge

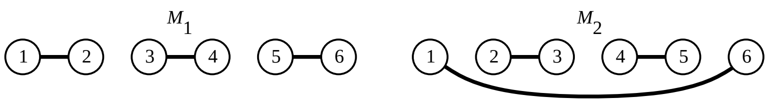

Fig. 1. The two matchings of OETB on the N-cycle for $N = 6$. Note that the excess token goes to the lower-numbered processor.

connecting processors i and j. Each balancing round consists of d steps, one for each matching. In step k, each pair $[i{:}j]$ of processors holding x_i and x_j tokens, resp., that are paired in matching M_k balance their load as closely as possible: their loads become $\lceil \frac{x_i+x_j}{2} \rceil$ and $\lfloor \frac{x_i+x_j}{2} \rfloor$, resp., (this means that the excess token, if there is any, is sent to processor i). This model is equivalent to the *(periodic) balancing circuit* paradigm [4]. Such a circuit is composed of a sequence of N wires connected by simple toggling devices called *balancers*. The matchings of the dimension exchange paradigm correspond to the balancers forming a round of d steps. The purpose of such a circuit is to balance the flow of tokens along the wires. Rounds are repeated until the total load is spread among the processors and wires, resp., as evenly as possible.

The Problem. We investigate the following very simple load-balancing algorithm on the N-cycle (N even) which we call *Odd-Even Transposition Balancing.* The edges of the cycle are partitioned into two matchings canonically (e. g., see Fig. 1) that together form a single round. We call this algorithm *Odd-Even Transposition Balancing* (OETB) due to its similarity to Odd-Even Transposition Sort [5, p. 240].

We are interested in upper bounds on the number of rounds necessary to perfectly balance the tokens in terms of the initial imbalance (or discrepancy) K (i. e., $\max_i |x_i - x_j|$) and the cycle's graph theoretical properties. If the tokens are allowed to be subdivided arbitrarily, this *idealized* balancing process can be described in terms of Markov chains (see [6]; for the related *diffusion* paradigm, see [2, 7, 8]). However, there is a deviation between this idealized process and the actual token process. In [1], the *local divergence* Ψ has been introduced that allows for upper bounding the difference between the two processes. In our setting, Ψ only depends on the matchings of the balancing method. In this paper, we compute the local divergence of OETB exactly.

A further question we address experimentally is the hardness of balancing. We run OETB on random load sequences with certain initial discrepancy K until discrepancy K' is ensured. Then we compare the number of steps necessary to perfectly balance these "pre-balanced" sequences with the number of steps necessary to perfectly balance random load sequences with initial discrepancy K'.

Related Work. The dimension exchange method was introduced by Cybenko [2] and by Hosseini *et al.* [3] in the context of load balancing on the hypercube which explains the name of the paradigm. Balancing circuits were introduced by Aspnes *et al.* [4]. They replace the comparators of some hypercubic sorting circuits on $N = 2^k$ wires by balancers and show that these depth-$O(k^2)$ circuits perfectly balance any input sequence. Interestingly, the number of steps does not depend on the inital discrepancy K of the input sequence. A complementary result was shown by Aharonson and Attiya [9]. They show that, if $N \neq 2^k$, there is no fixed balancing circuit that perfectly balances any input. Therefore, if $N \neq 2^k$, a fixed circuit has to be applied repeatedly to the input sequence, and the number of repetitions depends at least on K.

A similar approach to load balancing is *diffusion* [2, 7, 8]. Here, for each i, processor i with load x_i and d neighbors shifts about $x_i/(d+1)$ tokens to every neighbor. In [2, 7, 8], this method is analyzed assuming that the tokens can be subdivided arbitrarily.

In both paradigms, one can identify the actual token process and the idealized process where it is assumed that the load can be arbitrarily subdivided. We refer to $x_i^{(t)}$ as the number of tokens stored in processor i after t rounds, and to $\xi_i^{(t)}$ as the idealized, i. e., fractal load of processor i after t rounds.

Rabani *et al.* [1] introduced the local divergence Ψ of a load-balancing algorithm (regardless whether it is a diffusion or a dimension exchange method) which characterizes the deviation between the idealized and the actual token process. Ψ does not depend on K, only on the algorithm and the used network. It is shown that, for all t, $\max_i |\xi_i^{(t)} - x_i^{(t)}| \leq \Psi$, and that, in general, $\Psi = O(d \cdot (\log N)/(1-\lambda))$, where d is the degree of the network, N the number of processors, and λ the second eigenvalue of a matrix that describes the algorithm (see Sec. 2). As a consequence, it is shown that any load sequence with initial discrepancy K is transformed into a sequence with imbalance $O(\Psi)$ in $O(\log(K \cdot N)/(1-\lambda))$ rounds. A further reduction of the imbalance cannot be shown with this approach. For diffusive load balancing on the N-cycle, $\Psi = \frac{3}{4}N$ is proved.

For the dimension exchange model, a sorting-based upper bound of $O(K \cdot N)$ for perfect balancing on networks that have an almost Hamiltonian cycle, is also presented in [1].

Our Contribution. In Sect. 3, we prove $\Psi = N/2 - 1$. In order to compute the exact value we also compute the exact powers of the so called round matrix. As $\lambda = 1 - \Theta(1/N^2)$ for the round matrix, this means that after $O(N^2 \log(KN))$ rounds the discrepancy is at most $N/2 - 1$. After further $\frac{1}{2}N^2$ rounds the load is perfectly distributed among the processors.

In Sect. 4 experiments are presented that show that the number of rounds necessary to perfectly balance a load sequence with imbalance K that has been obtained by pre-balancing a random sequence with much larger imbalance is significantly larger than the average number of rounds necessary for balancing random sequences with imbalance K.

2 Details on the Model and on the Results

Odd-Even Transposition Balancing (OETB) works on the N-cycle, N even, with processors $1, \ldots, N$. The first matching is $M_1 = \{[1{:}2], [3{:}4], \ldots, [N-1{:}N]\}$, the second is $M_2 = \{[2{:}3], [4{:}5], \ldots, [N-2{:}N-1]\} \cup \{[1{:}N]\}$ (e. g., see Fig. 1). The *step matrices* P_1 and P_2 according to M_1 and M_2, resp., and the *round matrix* P are (in Fig. 1, they are presented for $N = 6$):

$$P^{(1)} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, P^{(2)} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}, P = P^{(1)} \cdot P^{(2)} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

We write $P^{(t,1)} = P^{t-1} \cdot P^{(1)}$, $P^{(t,2)} = P^t$, $P^{(t,1)-1} = P^{(t-1,2)}$, and $P^{(t,2)-1} = P^{(t,1)}$. $P^{(1,1)-1}$ is the identity matrix. E. g., $P^{(2,1)} = P^{(1)} \cdot P^{(2)} \cdot P^{(1)}$. Note that all matrices are block matrices. In the round matrix $P^{(t,k)}$, the denominator is always 2^{2t+k}.

The load sequence after round t, $t \geq 0$, is $x^{(t)} = (x_1^{(t)}, \ldots, x_N^{(t)})$. $x_i^{(t)}$ denotes the number of tokens processor i stores after round t. The *discrepancy* (or imbalance) of a load squence is $\mathcal{D}(x^{(t)}) = \max_{ij} |x_i^{(t)} - x_j^{(t)}|$. $K = \mathcal{D}(x^{(0)})$ is the initial discrepancy. The goal is to determine the number T of rounds required to reduce the discrepancy to some specific value ℓ: we refer to this as ℓ-smoothing. In general, the number of rounds required to ℓ-smooth an initial sequence will depend on both ℓ and the initial discrepancy.

In an idealized setting, single tokens are allowed to be split between the processors involved in a balancing step. In this setting, and with $\xi^{(0)} = x^{(0)}$, it is easy to see that $\xi^{(t)} = \xi^{(0)} \cdot P^t$. The number of rounds t to ℓ-smooth $\xi^{(0)}$ is bounded above by $t \leq 2/(1-\lambda) \cdot \ln(KN^2/\ell)$, where λ is the second eigenvalue of (a symmetrization of) P [1]. For OETB, it is easy to see that $\lambda = 1 - \Theta(1/N^2)$ because the underlying graph is the N-cycle.

In order to relate the deviation between the integral and the idealized process, the local divergence has been introduced [1]. Here, we present it already adapted to OETB on the N-cycle.

Definition 1 ([1]). *The* local divergence *(adapted to OETB and the N-cycle) is*

$$\Psi(P) = \max_l \sum_{t=1}^{\infty} \Big(\sum_{[i:j] \in M_1} \left| P_{li}^{(t,1)-1} - P_{lj}^{(t,1)-1} \right| + \sum_{[i:j] \in M_2} \left| P_{li}^{(t,2)-1} - P_{lj}^{(t,2)-1} \right| \Big)$$

Theorem 1 ([1]). *The maximum deviation between the idealized process and the integral process satisfies* $\max_i |\xi_i^{(t)} - x_i^{(t)}| \leq \Psi(P^{\mathrm{T}})$ *for all* t, *where* P *is the round matrix and* P^{T} *its transpose.*

Note that the bound depends on the local divergence computed on the transpose of P.

In this paper, we compute the local divergence for OETB exactly:

Theorem 2. *For OETB on the N-cylce,* $\Psi(P^{\mathrm{T}}) = N/2 - 1$.

Corollary 1. *OETB needs* $O(N^2 \log(KN))$ *rounds to* 1*-smooth any load sequence with initial discrepancy* K.

3 Computation of the Local Divergence of OETB

This section is devoted to the proof of Theorem 2.

3.1 Obtaining an Expression for $P_{ij}^{(t,k)}$

Simplifying *P*. Using the block structure of the matrix enables us to work with $n \times n$ matrices, $n = N/2$, by choosing every second row and column from the original matrix. E. g., with $N = 6$ and $n = 3$, we get the reduced $n \times n$ matrix $Q = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} \end{pmatrix}$.

No information is lost in the smaller matrix because its entries are exactly the same as the neighboring ones in the full matrix; additionally the calculation gets handier:

$$\Psi(Q^T) = \max_l \sum_{t=1}^{\infty} \sum_{k=1}^{2} \left(|Q_{1l}^{(t,k)-1} - Q_{nl}^{(t,k)-1}| + \sum_{i=1}^{n} |Q_{il}^{(t,k)-1} - Q_{i+1,l}^{(t,k)-1}| \right)$$

Error propagation on the infinite circle. Consider an infinite linear array with the nodes numbered from $-\infty$ to ∞ omitting 0, and let node 1 be the pivot. Fig. 2 shows the error contribution of the surrounding nodes:

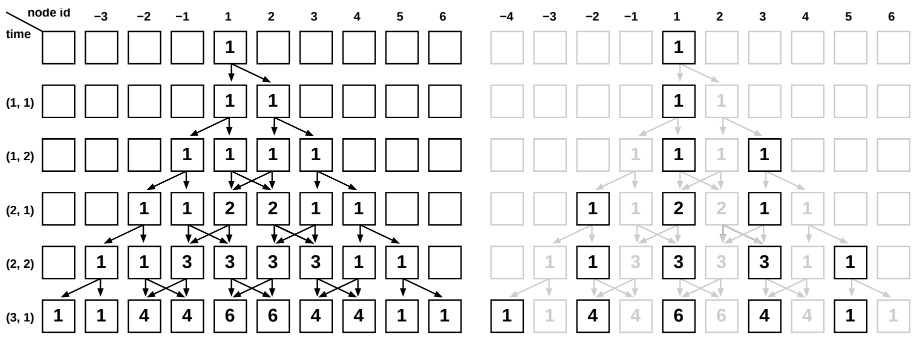

Fig. 2. Token exchange skeleton on an infinite circle – if every second node is left out the binomial coefficients are clearly visible. In order to obtain $P^{(t,k)}$ one has to multiply the shown numbers by $2^{-(2t+k)}$.

We are interested in the contribution of the neighboring nodes to the error in the pivot. Therefore we trace the error contribution of every single node back in time. On the infinite array the binomial coefficients show up, which becomes obvious if every second node is masked out (see Fig. 2 right).

Winding up the infinite array. We will see that winding up the infinite array leads to an explicit expression for the local divergence of OETB.

When winding the series around the cycle with n nodes then every n-th entry is mapped on the same node. The following sum from [5, p. 89] extracts and sums up every n-th entry from a sequence, starting with entry r:

$$\sum_{m \bmod n=r} a_m z^m = \frac{1}{n} \sum_{0\le k<n} \omega_n^{-kr} G(\omega_n^k z) \ , 0 \le r < m, \tag{1}$$

where $\omega_n = \cos(2\pi/n) + \mathrm{i} \cdot \sin(2\pi/n)$ denotes an n-th primitive root of unity, and $G(z) = \sum_{m=-\infty}^{\infty} a_m z^m$ a generating function for the a_m.

We choose a generating polynomial that fixes the maximum coefficient to z^0, symmetrically surrounded by the other coefficients of $z^{\pm i}$. A double time step from $(t-1,k)$ to (t,k) is obtained by multiplying the existing series of coefficients by

$$\frac{1}{2^2}\left(1 \cdot z^{-1} + 2 + 1 \cdot z^1\right)$$

This double step avoids the difficulties that arise on handling a single step from (t,k) to $(t,k)+1$, where the existing polynomial must be multiplied alternately by $\frac{1}{2}(z^{-1}+1)$ and $\frac{1}{2}(1+z^1)$.

Consequently we distinguish two cases when determining $Q_{ij}^{(t,k)}$, one for the time steps $(t,1)$ and one for the time steps $(t,2)$. Note the offset of $\frac{1}{2}$ and that for $d=2$ one requires an additional factor, which comes from the first single time step:

$$Q_{ij}^{(t,k)} = \begin{cases} \frac{1}{n}\sum\limits_{k=0}^{n} \omega_n^k \cdot \frac{1}{2}\left(\frac{1}{4}\omega_n^k + \frac{2}{4} + \frac{1}{4}\omega_n^{k(n-1)}\right)^t & \text{for } d=1 \\ \frac{1}{n}\sum\limits_{k=0}^{n} \omega_n^k \cdot \frac{1}{4}\left(1 + \omega_n^{k(n-1)}\right)\left(\frac{1}{4}\omega_n^k + \frac{2}{4} + \frac{1}{4}\omega_n^{k(n-1)}\right)^t & \text{for } d=2 \end{cases}$$

Four different cases

- Distinguishing odd and even time steps
  The representation of $Q_{ij}^{(t,k)}$ suggests a first split of the calculation in two cases, one for the steps $(t,1)$ and one for the steps $(t,2)$. Thus $\Psi(Q^T) = \Psi_1(Q^T) + \Psi_2(Q^T)$ where $\Psi_i(Q^T)$ denotes the sum over all steps (t,i).
- Splitting to $n \equiv 0 \bmod 2$ and $n \equiv 1 \bmod 2$
  Both cases end up in the same result, but the calculation is slightly different.

Lemma 1. $\Psi_1(Q^T) = \frac{n}{2} \ , \Psi_2(Q^T) = \frac{n}{2} - 1$ *if* $n \equiv 0 \bmod 2$
$\Psi_1(Q^T) = \frac{n}{2} - \frac{1}{2n} \ , \Psi_2(Q^T) = \frac{n}{2} + \frac{1}{2n} - 1$ *if* $n \equiv 1 \bmod 2$

The calculations for the four cases are very similar. Instead of a full proof of Lemma 1 we present the calculation for $\Psi_2(Q^T)$ in the case that $n \equiv 0 \bmod 2$, which contains all important intermediate steps and is still reasonably short. For a full presentation of all four cases see [10].

3.2 Rewriting the Sum as an Easier Expression

Several simplifications apply in the case of OETB to the original equation:

- block circularity
  Since the round matrix is block circular, it is possible to leave out the maximization over l and put $l = 1$, since all columns are equal up to circular shifts, thus

$$\Psi_2(Q^T) = \sum_{t=1}^{\infty} \left(|Q_{11}^{(t,1)-1} - Q_{n1}^{(t,1)-1}| + \sum_{i=1}^{n} |Q_{i1}^{(t,1)-1} - Q_{i+1,1}^{(t,1)-1}| \right)$$

- symmetry of one column
  Since the entries in one column of Q are symmetrical it is possible to sum over only half of the matrix and multiply it by 2:

$$\Psi_2(Q^T) = \sum_{t=1}^{\infty} 2 \sum_{i=1}^{\lceil n/2 \rceil} |Q_{i1}^{(t,1)-1} - Q_{i+1,1}^{(t,1)-1}|$$

- telescoping the sum
  The sum over the differences $Q_{i1}^{(t,1)-1} - Q_{i+1,1}^{(t,1)-1}$ collapses to a simple difference $Q_{11}^{(t,1)-1} - Q_{\lceil n/2 \rceil,1}^{(t,1)-1}$ due to the unimodality of the winded up binomial coefficients that form the entries of the matrix:

$$\Psi_2(Q^T) = \sum_{t=1}^{\infty} 2 \left(Q_{11}^{(t,1)-1} - Q_{j1}^{(t,1)-1} \right)$$

3.3 Calculating $\boldsymbol{\Psi(Q^T)}$

$$\begin{aligned}
\Psi_2(Q^T) &= \sum_{t=1}^{\infty} 2 \left(Q_{11}^{(t,1)-1} - Q_{\lceil n/2 \rceil 1}^{(t,1)-1} \right) \\
&= \frac{1}{n} \sum_{t=0}^{\infty} \sum_{k=0}^{n-1} \left(1 - \omega_n^{k\lceil n/2 \rceil}\right) \left(\frac{1}{2} + \frac{1}{2}\omega_n^{k(n-1)}\right) \left(\frac{2}{4} + \frac{1}{4}\omega_n^{k} + \frac{1}{4}\omega_n^{k(n-1)}\right)^t \\
&= \frac{1}{n} \sum_{k=0}^{n-1} \frac{\left(1-\omega_n^{k\lceil n/2 \rceil}\right)\left(\frac{1}{2}+\frac{1}{2}\omega_n^{k(n-1)}\right)}{1-\left(\frac{2}{4}+\frac{1}{4}\omega_n^{k}+\frac{1}{4}\omega_n^{k(n-1)}\right)} \\
&= \frac{1}{2n} \sum_{k=1}^{n-1} \frac{1+\cos\left(\frac{2\pi k}{n}\cdot(n-1)\right)-\cos\left(\frac{2\pi k}{n}\cdot\left\lceil\frac{n}{2}\right\rceil\right)-\cos\left(\frac{2\pi k}{n}\cdot\left(\left\lceil\frac{n}{2}\right\rceil-1\right)\right)}{\sin^2\left(\frac{\pi k}{n}\right)}
\end{aligned}$$

The final task is to handle the cases for $\lceil \frac{n}{2} \rceil$ properly. As mentioned earlier, we focus on the case $n \equiv 0 \bmod 2$, the other one is substantially the same, but lengthier.

$$\Psi_2(Q^T) = \frac{1}{2n} \sum_{k=1}^{n-1} \frac{1 + \cos\left(\frac{2\pi k}{n}\right) - \cos\left(\pi k\right) - \cos\left(\pi k + \frac{2\pi k}{n}\right)}{\sin^2\left(\frac{\pi k}{n}\right)}$$

The sum can be split into even and odd k now. For even k, we have $(1 - \cos(\pi k)) = 2$ and $\cos\left(\frac{2\pi k}{n}\right) - \cos\left(\pi k + \frac{2\pi k}{n}\right) = 2\cos\left(\frac{4\pi k}{n} + \frac{2\pi}{n}\right)$; for odd k, the summand is zero:

$$\Psi_2(Q^T) = \frac{1}{2n} \sum_{k=0}^{n/2-1} \frac{2 + 2\cos\left(\frac{4\pi k}{n} + \frac{2\pi}{n}\right)}{\sin^2\left(\frac{2\pi k}{n} \frac{\pi}{n}\right)}$$

According to Bromwich [11, p. 216], $n^2 \sin^{-2}(n\phi) = \sum_{k=0}^{n-1} \sin^{-2}\left(\frac{k\pi}{n} + \phi\right)$, which can finally be used for solving to

$$\begin{aligned} \Psi_2(Q^T) &= \tfrac{2}{n} \sum_{k=0}^{n/2-1} \left(\sin^{-2}\left(\tfrac{2\pi k}{n} + \tfrac{\pi}{n}\right) - 1\right) \\ &= \tfrac{2}{n}\left(\tfrac{n}{2}\right)^2 \sin^{-2}\left(\tfrac{\pi n}{2n}\right) - \tfrac{2}{n}\left(\tfrac{n}{2}\right) = \tfrac{n}{2} - 1 = \tfrac{N}{4} - 1 \end{aligned}$$

$\square$

4 Experimental Results

4.1 Worst Case Bounds for 1-Smoothing

In [1], a further, sorting-based upper bound of $O(N \cdot K)$ on the number of rounds to 1-smooth load sequences with initial discrepancy K is shown, in contrast to the local divergence-based upper bound of $O(N^2 \log(KN))$ rounds for Ψ-smoothing.

Now, we investigate the question whether a break-even point for the two bounds can be identified experimentally. Since a general method for generating bad case inputs is not yet known, we produce random load vectors and simulate the balancing process.

Unfortunately we were not able to identify a break-even point, probably because the randomly generated load distributions were too far from the (assumed) worst case bound, but the bad inputs showed to be useful when compared to the average case behavior of inputs with larger discrepancy, as shown in the following section.

4.2 Average Case Behaviour of 1-Smoothing

In this section, we show that balancing a randomly chosen input pre-balances the input in a way that makes it much more robust against further balancing steps.

Thus balancing for a specified period of time takes away the obvious differences in a randomly chosen input and seems to leave a what we call "inner discrepancy" that is on average hard to 1-smooth.

It is surprising to see that from a fixed discrepancy k_0 it is on average much harder to 1-smooth a randomly chosen input with an initial discrepancy of $k_1 \gg k_0$ that was pre-balanced to a discrepancy of k_0 than taking the hardest vector from a number of random experiments with an initial discrepancy of k_0.

Consequently, if the average running time of random load distributions with an *initial* discrepancy of k_0 is considered, it is in most cases smaller than the average *remaining* balancing time of random load distributions that were balanced down from an initial discrepancy of $k_1 \gg k_0$ to k_0.

Since in the average inputs with larger initial discrepancy need more rounds for 1-smoothing, this leads to the assumption that these additional rounds are "created" by earlier balancing steps: Balancing a randomly chosen input several rounds leaves a load distribution of discrepancy k_i that is very hard to 1-smooth because of the previous balancing steps. See Table 1 and Fig. 3 and 4 for some typical results.

Table 1. Experimental evaluation: Some typical results are shown for four different setups. One column contains a single experiment setup, and in the last two rows the obtained significant difference between sorting a pre-balanced input further and sorting randomly generated input. Cf. Figures 3 and 4 for tracing the discrepancy's decrease.

number of nodes (constant)	200	200	100	100
larger initial discrepancy	3200	1600	3200	1600
number experiments (larger discrepancy)	10000	10000	10000	10000
smaller initial discrepancy	400	400	100	400
number experiments (smaller discrepancy)	30000	30000	52030	10000
average number rounds larger dataset	6984	5349	1632	1957
worst example from smaller discrepancy	5418	5418	1176	1896

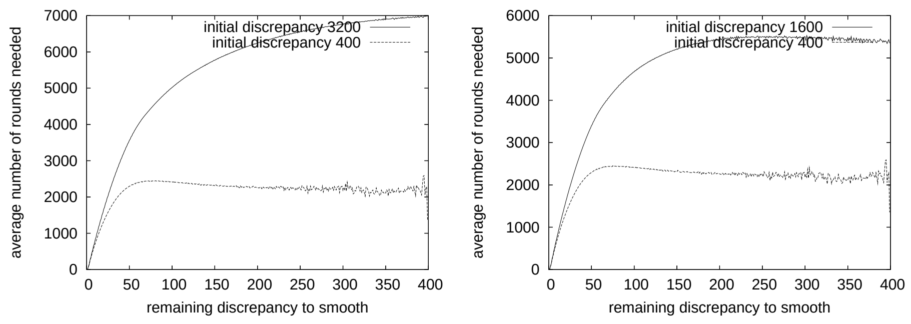

Fig. 3. Left: Example on 200 nodes. Starting out with a large initial discrepancy of 3200 results in a very difficult to 1-smooth pre-balanced string in the average when it reaches a discrepancy of 400 (solid line). Compare the average (much faster) progress when starting out with an initial discrepancy of 400 (dashed line). On the right the same setting with a large initial discrepancy of 1600

In Figures 3 and 4 is the average progress on a remaining discrepancy of 400 or 100, resp., shown for two cases, for distributions that are

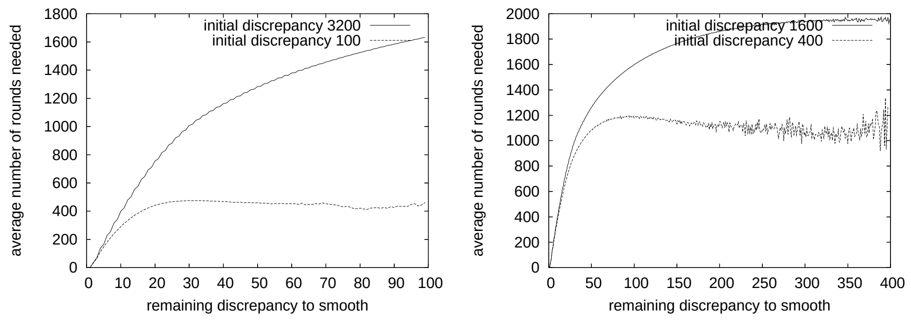

Fig. 4. This property seems to be invariant under the number of processors or the initial discrepancy: On the left an experiments with 100 nodes, large initial discrepancy of 3200 (solid line), smaller initial discrepancy of 100 (dashed line), on the right also 100 nodes, and the discrepancies 1600 and 400

1. pre-balanced from an initial discrepancy of 3200 down to 400 (solid line)
2. randomly chosen with an initial discrepancy of 400 (dashed line)

The two curves show the average number of remaining rounds until the load balancing distributions were 1-smoothed.

References

1. Rabani, Y., Sinclair, A., Wanka, R.: Local divergence of Markov chains and the analysis of iterative load-balancing schemes. In: Proc. 39th IEEE Foundations of Computer Science (FOCS). (1998) 694–703
2. Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors. Journal of Parallel and Distributed Computing **7** (1989) 279–301
3. Hosseini, S., Litow, B., Malkawi, M., McPherson, J., Vairavan, K.: Analysis of a graph coloring based distributed load balancing algorithm. Journal of Parallel and Distributed Computing **10** (1990) 160–166
4. Aspnes, J., Herlihy, M., Shavit, N.: Counting networks. Journal of the ACM **41** (1994) 1020–1048
5. Knuth, D.E.: The Art of Computer Programming, Volume 3: Sorting and Searching. 2nd edn. Addison-Wesley, Reading, Massachusetts (1998)
6. Busch, C., Mavronicolas, M.: A combinatorial treatment of balancing networks. Journal of the ACM **43** (1996) 794–983
7. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computations: Numerical Methods. Prentice-Hall (1989)
8. Boillat, J.E.: Load balancing and Poisson equation in a graph. Concurrency: Practice and Experience **2** (1990) 289–313
9. Aharonson, E., Attiya, H.: Counting networks with arbitrary fan-out. Distributed Computing **8** (1995) 163–169
10. Rieß, C.: Load-Balancing auf dem Kreis. Studienarbeit, Department of Computer Science, University of Erlangen-Nuremberg (2006)
11. Bromwich, T.: An Introduction to the Theory of Infinite Series. MacMillan (1926)