Load Balancing Using Bisectors – A Tight Average-Case Analysis

Stefan Bischof, Thomas Schickinger, and Angelika Steger

Institut für Informatik, Technische Universität München
D-80290 München, Germany
{bischof|schickin|steger}@in.tum.de
http://www14.in.tum.de/

Abstract. In parallel computation we often need an algorithm for dividing one computationally expensive job into a fixed number, say N, of subjobs, which can be processed in parallel (with reasonable overhead due to additional communication). In practice it is often easier to repeatedly bisect jobs, i.e., split one job into exactly two subjobs, than to generate N subjobs at once. In order to balance the load among the N machines, we want to minimize the size of the largest subjob (according to some measure, like cpu-time or memory usage).
In this paper we study a recently presented load balancing algorithm, called Heaviest First Algorithm (Algorithm HF), that is applicable to all classes of problems for which bisections can be computed efficiently. This algorithm implements a very natural strategy: During $N-1$ iterations we always bisect the largest subproblem generated so far.
The maximum load produced by this algorithm has previously been shown to differ from the optimum only by a constant factor even in the worst-case. In this paper we consider the average-case, assuming a natural and rather pessimistic random distribution for the quality of the bisections. Under this model the heaviest load generated by Algorithm HF is proved to be only twice as large as the optimum with high probability. Furthermore, our analysis suggests a simpler version of Algorithm HF which can easily be parallelized.

1 Introduction

Dynamic load balancing for irregular problems is a major research issue in the context of parallel and distributed computing. Often it is essential to achieve a balanced distribution of load in order to reduce the execution time of an application or to maximize system throughput.

We consider a general scenario, where an irregular problem is generated at run-time and must be split into subproblems that can be processed on different processors. If N processors are available, the problem is to be split into N subproblems and the goal is to balance the weight of the subproblems assigned to the individual processors. The weight of a problem (or subproblem) represents the load (CPU load, for example) caused by the problem on the processor to which it is assigned. It is assumed that the weight of a problem can be calculated (or approximated) easily once it is generated.

Instead of splitting the original problem into N subproblems in a single step, we use repeated *bisections* for generating the N subproblems, because in practice it is often easier to find efficient bisection methods than to find a method for splitting a problem into an arbitrary number of subproblems in one step. A bisection subdivides a problem into exactly two subproblems. Optimization problems in planar graphs (and their generalizations) are a classical field where a similar approach is commonly used (see for example [LT79, AST90]), and which may be adapted to fit into this model.

Load balancing using bisectors can and has been applied to a variety of practical problems. Our research is motivated by an application from distributed numerical simulations, namely a parallel solver for partial differential equations using a variant of the finite element method (FEM) with adaptive, recursive substructuring [HS94, BEE98]. Other possible application domains include chip layout [RR93] or multi-dimensional adaptive numerical quadrature [Bon93].

In this paper, we will not be concerned with details regarding any particular application. Instead, we study parallel load balancing from a more abstract point of view. We only assume that bisectors can be computed efficiently for the problems under consideration.

The remainder of the paper is organized as follows. In Sect. 2 we introduce our load balancing model and briefly review previous results regarding the worst-case behavior of Algorithm HF. In Sect. 3 we present our main result and the theoretical analysis. Finally, Sect. 4 contains some concluding remarks.

2 Load Balancing Model and Previous Work

As in [BEE98], we study the following simplified model for dynamic load balancing. The parallel system, on which we want to solve a problem p, consists of N processors or machines. The goal of the load balancing is to split p into N subproblems $p_1, \ldots, p_N$, which can be solved individually. In our analysis we neglect the overhead which results from the combination of the partial solutions to the solution of the initial problem. This simplification is justified if the bisections produce only loosely coupled subproblems. Assuming a weight function w that measures the resource demand (for example, CPU load or memory requirement, depending on the application at hand) of subproblems, the goal is to minimize the maximum weight among the resulting subproblems, i.e., to minimize $\max_{1\leq i\leq N} \mathsf{w}(p_i)$.

As mentioned before, we assume that the load balancing algorithm can only split a subproblem into *two* subproblems in a single step, using a bisection method. Repeated bisections must then be used to split a problem into N subproblems. If problem q is bisected into subproblems q_1, q_2 we assume that $\mathsf{w}(q) = \mathsf{w}(q_1) + \mathsf{w}(q_2)$, i.e., the sum of the weights of the subjobs equals the weight of the initial job.

In general, it can not be expected that bisections always split a problem p of weight $\mathsf{w}(p)$ into two subproblems with weight $\mathsf{w}(p)/2$ each. For many classes of problems, however, there are bisection methods that guarantee that the weights of the two obtained subproblems do not differ too much. The following definition captures this concept more precisely.

Definition 1. *Let* $0 < \alpha \leq \frac{1}{2}$. *A class* $\mathcal{P}$ *of problems with weight function* $\mathsf{w} : \mathcal{P} \to \mathbb{R}^+$ has α-bisectors *if every problem* $p \in \mathcal{P}$ *can be efficiently divided into two problems* $p_1 \in \mathcal{P}$ *and* $p_2 \in \mathcal{P}$ *with* $\mathsf{w}(p_1) + \mathsf{w}(p_2) = \mathsf{w}(p)$ *and* $\mathsf{w}(p_1), \mathsf{w}(p_2) \in [\alpha \mathsf{w}(p); (1-\alpha)\mathsf{w}(p)]$.

```
algorithm HF(p,N)
begin
      P := {p};
      while |P| < N do
            begin
            q := a problem in P with maximum weight;
            bisect q into q1 and q2;
            P := (P ∪ {q1, q2}) \ {q};
            end;
      return P;
end.
```

Fig. 1. Algorithm HF (Heaviest Problem First)

Fig. 1 shows Algorithm HF, which receives a problem p and a number N of processors as input and divides p into N subproblems by repeated bisection of the heaviest remaining subproblem. There is also an efficient parallel version of this algorithm as shown in [BEE99].

A perfectly balanced load distribution on N processors would be achieved if a problem p of weight $\mathsf{w}(p)$ was divided into N subproblems of weight exactly $\mathsf{w}(p)/N$ each.

The following theorem shows that the worst-case behavior of Algorithm HF differs from the optimum only by a factor depending on α but not on N.

Theorem 2. [BEE98] *Let* $\mathcal{P}$ *be a class of problems with weight function* $\mathsf{w} : \mathcal{P} \to \mathbb{R}^+$ *that has* α*-bisectors. Given a problem* $p \in \mathcal{P}$ *and a positive integer* N, *Algorithm HF uses* $N-1$ *bisections to partition* p *into* N *subproblems denoted by* $p_1, \ldots, p_N$ *such that*

$$\max_{1 \leq i \leq N} \mathsf{w}(p_i) \leq \frac{\mathsf{w}(p)}{N} \cdot r_\alpha , \quad \textit{where} \quad r_\alpha = \left\lfloor \frac{1}{\alpha} \right\rfloor \cdot (1-\alpha)^{\lfloor \frac{1}{\alpha} \rfloor - 2} .$$

Note that r_α is equal to 2 for $\alpha \geq 1/3$, below 3 for $\alpha \geq 1 - 1/\sqrt[4]{2} \approx 0.159$, and below 10 for $\alpha \geq 0.04$. Hence, Algorithm HF achieves provably good load balancing for classes of problems with α-bisectors for a surprisingly large range of α.

3 Average-Case Analysis of Algorithm HF

3.1 Main Result

The following stochastic model for an average-case scenario that may arise from practical applications seems reasonable: If a problem q is bisected into q_1 and q_2 we can find a *bisection parameter* $\alpha \in [0, \frac{1}{2}]$ such that $\mathsf{w}(q_1) = \alpha\mathsf{w}(q)$ and $\mathsf{w}(q_2) = (1-\alpha)\mathsf{w}(q)$. Assume that the actual bisection parameter α is drawn uniformly at random from the interval $[\hat{\alpha}, \hat{\beta}]$, $0 \leq \hat{\alpha} \leq \hat{\beta} \leq 1/2$, and that all $N-1$ bisection steps are independent and identically distributed. We will write $\alpha \sim U[\hat{\alpha}, \hat{\beta}]$ if α is uniformly distributed on $[\hat{\alpha}, \hat{\beta}]$. In this paper, we study the case $\hat{\alpha} = 0, \hat{\beta} = \frac{1}{2}$, which seems rather pessimistic for practical applications, where in many cases α-bisectors (for $\alpha > 0$) are available. Note that the assumption $\alpha \sim U[0, \frac{1}{2}]$ could be changed to $\alpha \sim U(0, \frac{1}{2}]$ throughout our analysis without any modifications in the proofs.

Using this model we obtain the following result:

Theorem 3. *If Algorithm HF starts with a problem of weight W and produces a set P of N subproblems after $N-1$ bisections then for $\varepsilon = 9\sqrt{\ln(N)/N}$*

$$\Pr\left[(2-\varepsilon)\frac{W}{N} \leq W_{\max}^N < (2+\varepsilon)\frac{W}{N}\right] = 1 - o(1/N) \ ,$$

where $W_{\max}^N := \max\{\mathsf{w}(p) \ : \ p \in P\}$ denotes the weight of the heaviest subproblem in P.

Thus, with high probability the maximum load differs from the best attainable value $\frac{W}{N}$ only by a factor of roughly two.

3.2 Outline of the Analysis

For our analysis we assume that Algorithm HF executes infinitely many iterations of the loop and consider the *infinite bisection tree* (IBT) generated by this process. This tree grows larger with each iteration of Algorithm HF. There is a one-to-one mapping from the subproblems produced by Algorithm HF to the nodes in the IBT. At a every point in time the subproblems in P correspond to the leaves in the part of the IBT which has been generated so far. The root is the initial problem p. If a node/subproblem q is split two new nodes/subproblems q_1 and q_2 are appended to q and, thus, the IBT is an infinite binary tree. One can also imagine that the IBT exists a priori and Algorithm HF visits all nodes one by one. When we say that Algorithm HF visits or expands a node in the IBT we adopt this view on the model. The IBT is the infinite version of the *bisection tree* (BT) employed in [BEE98].

Now let's define the probability space more formally: We set $\Omega := [0,1]^{\mathbb{N}}$ with the uniform distribution. As usual in the continuous case, the set of events is restricted to the Borel σ-field on $[0,1]^{\mathbb{N}}$, which we denote by $\mathbb{F}$.

An element $s := (\alpha_1, \alpha_2, \ldots) \in \Omega$ has the following interpretation: $\alpha_i \sim U[0,1]$ corresponds to the relative weight of the, say, left successor of the node which is split by the i-th bisection. We do not consider the actual bisection parameter (which is equal to α_i or $(1-\alpha_i)$), since studying $U[0,1]$ instead of $U[0,\frac{1}{2}]$ simplifies some calculations and we do not care which successor node is heavier. We call α_i the *generalized bisection parameter* of bisection i. Note that for every $s \in \Omega$ the corresponding IBT can be computed by simulating Algorithm HF.

We call a node v in the IBT *d-heavy* iff $\mathsf{w}(v) \geq d$, and accordingly we use the term *d-light*. If the value of d is obvious from the context we just say *heavy* and *light* for short.

For every node v in the complete binary tree T we define the random variables H_v^d (for $d > 0$)

$$H_v^d := \begin{cases} 1 & \text{if } \mathsf{w}(v) \geq d \\ 0 & \text{otherwise} \end{cases} \quad .$$

Since T is a superset of the nodes in any IBT, the random variable $H^d := \sum_{v \in T} H_v^d$ counts the number of d-heavy nodes in the IBT. If a node v is not part of the IBT, H_v^d is zero.

Additionally, we define for each $v \in T$ a random variable X_v. If v is part of the IBT X_v denotes the weight of v relative to its ancestor p in the tree, i.e., $X_v := \mathsf{w}(v)/\mathsf{w}(p)$. Thus, X_v corresponds to the generalized bisection parameter α and, as mentioned above, we will assume that $X_v \sim U[0,1]$. We won't be interested in the value of X_v when v is not part of the IBT, so we don't specify how X_v behaves in this case.

The random variables H^d directly correspond to the performance of Algorithm HF, since H^d equals the number of iterations after which the set P of processes generated by Algorithm HF contains only light nodes. This is due to the fact that Algorithm HF visits all d-heavy nodes before it expands any d-light node.

Our analysis proceeds as follows: First we show that the expected number of heavy nodes $\mathbb{E}[H^d]$ is comparatively small. Then we prove that H^d is sharply concentrated around $\mathbb{E}[H^d]$ and thus H^d is small with high probability. Finally, the desired results for Algorithm HF follow easily.

3.3 Expected Number of Heavy Nodes

If we look at a node v_l on level l in the IBT (level 0 contains only the root, which has weight W, level i contains all nodes at BFS-distance i) and denote its ancestors on the levels $0, \ldots, l-1$ by $v_0, \ldots, v_{l-1}$ we get

$$\mathsf{w}(v_l) = W \cdot \prod_{i=1}^{l} X_{v_i} \quad .$$

The following lemma enables us to analyze the distribution of products of $U[0,1]$-distributed random variables exactly.

Lemma 4. *Let $X_1, \ldots, X_n$ be independent random variables with the exponential distribution and $\mathbb{E}[X_1] = \ldots = \mathbb{E}[X_n] = 1/\lambda$. Then for $X := \sum_{i=1}^n X_i$*

$$\Pr[X \le t] = \begin{cases} 1 - \sum_{i=0}^{n-1} \frac{(\lambda t)^i}{i!} \cdot e^{-\lambda t} = e^{-\lambda t} \cdot \sum_{i=n}^{\infty} \frac{(\lambda t)^i}{i!} & \text{if } t \ge 0 \\ 0 & \text{otherwise} \end{cases} .$$

Proof. See [Fel71, p. 11].

It is easy to show (see [Fel71, pp. 25]) that a random variable $X_i \sim U[0,1]$ can be transformed to a random variable $Y_i := -\ln X_i$ with exponential distribution and $\mathbb{E}[Y_i] = 1$. If we want to analyze $X := \prod_{i=1}^n X_i$ we can equivalently analyze $Y := -\ln X := \sum_{i=1}^n Y_i$. Combining this fact with Lemma 4 yields the following lemma.

Lemma 5. *Let $X := \prod_{i=1}^n X_i$ be the product of independent random variables with $X_i \sim U[0,1]$. If we define Y as $Y := -\ln X$ we obtain for $t \in [0,1]$*

$$\Pr[X \ge t] = \Pr[Y \le -\ln t] = t \cdot \sum_{i=n}^{\infty} \frac{(-\ln t)^i}{i!} .$$

Proof. See [Fel71, p. 25].

In the following discussion we will often be concerned with the probability p_l^d that an arbitrary node on level l in the IBT is d-heavy.

Lemma 6. *Let v_l be an arbitrary node on level l of the IBT and denote the ancestors of v_l in the tree by $v_0, \ldots, v_{l-1}$. If the root of the IBT has weight W ($W \ge d$ for a fixed $d > 0$) then for the probability*

$$p_l^d := \Pr[\mathsf{w}(v_l) \ge d]$$

the following holds (with $l \ge \ln(W/d)$ and $l \ge 1$):

$$p_l^d = \frac{d}{W} \cdot \sum_{i=l}^{\infty} \frac{(\ln(W/d))^i}{i!} \le \frac{d}{W} \cdot \left(\frac{\ln(W/d) \cdot e}{l}\right)^l .$$

Proof. Since $\mathsf{w}(v_l)$ is the product of W and the generalized bisection parameters of the ancestors of v_l in the IBT we obtain using Lemma 5

$$p_l^d = \Pr\left[W \cdot \prod_{i=1}^{l} X_{v_i} \ge d\right] = \Pr\left[\sum_{i=1}^{l} Y_{v_i} \le -\ln\left(\frac{d}{W}\right)\right] = \frac{d}{W} \cdot \sum_{i=l}^{\infty} \frac{(\ln(W/d))^i}{i!} .$$

This proves the first equation. The second inequality can be shown as follows:

$$\sum_{i=l}^{\infty} \frac{(\ln(W/d))^i}{i!} = (\ln(W/d))^l \cdot \sum_{i=0}^{\infty} \frac{(\ln(W/d))^i}{(i+l)!} = \frac{(\ln(W/d))^l}{l^l} \cdot \sum_{i=0}^{\infty} \frac{(\ln(W/d))^i \, l^l}{(i+l)!}$$
$$\le (\ln(W/d))^l \cdot \frac{1}{l^l} \cdot \sum_{i=0}^{\infty} \frac{l^{i+l}}{(i+l)!} \le (\ln(W/d))^l \cdot \frac{e^l}{l^l} .$$

Now we are in a position to state our first main result, namely the expected value for the number of heavy nodes in the IBT.

Theorem 7. *For an IBT where the root has weight W with $W > d$, it holds that*

$$\mathbb{E}[H^d] = \frac{2}{d}W - 1 \ .$$

Proof. Let $T(l)$ denote the set of nodes on level l in the complete binary tree. For a single node $v \in T(l)$ we obtain

$$\mathbb{E}[H_v^d] = p_l^d \cdot \Pr[v \in \mathrm{IBT}] = p_l^d \ ,$$

because $\Pr[v \in \mathrm{IBT}] = 1$. We just sketch how this can be shown: If $v \notin \mathrm{IBT}$ we can find a node $v' \in \mathrm{IBT}$ which is never bisected. This implies that there are infinitely many nodes in the tree with weight at least $d' := \mathsf{w}(v')$. Furthermore, we observe that sequences $s \in [0,1]^{\mathbb{N}}$ where only finitely many components belong to $[\varepsilon, 1-\varepsilon]$ (for $0 < \varepsilon < 0.5$) occur with probability zero. If, on the other hand, infinitely many components belong to $[\varepsilon, 1-\varepsilon]$ there are only finitely many d'-heavy nodes in the tree.

The theorem follows by linearity of expectation using the value of p_l^d which we derived in Lemma 6.

$$\begin{aligned}
\mathbb{E}[H^d] &= \sum_{l=0}^{\infty} \sum_{v \in T(l)} \mathbb{E}[H_v^d] = \sum_{l=0}^{\infty} 2^l \cdot p_l^d \\
&= \frac{d}{W} \cdot \sum_{l=0}^{\infty} \sum_{i=l}^{\infty} 2^l \cdot \frac{(\ln(W/d))^i}{i!} = \frac{d}{W} \cdot \sum_{i=0}^{\infty} \sum_{l=0}^{i} 2^l \cdot \frac{(\ln(W/d))^i}{i!} \\
&= \frac{d}{W} \cdot \sum_{i=0}^{\infty} \frac{(\ln(W/d))^i}{i!} (2^{i+1} - 1) = \frac{d}{W} \cdot \left[2 \left(\frac{W}{d} \right)^2 - \frac{W}{d} \right] = \frac{2}{d} W - 1 \ .
\end{aligned}$$

3.4 Sharp Concentration

Theorem 7 gives us an idea how good Algorithm HF performs. If we set $d = 2W/N$ we get $\mathbb{E}[H^d] = N - 1$. This is exactly the number of bisections that are necessary to produce N subproblems. Thus, on the average, we have bisected every heavy subproblem after $N-1$ iterations of Algorithm HF, and all the N generated subproblems have weight smaller than $2W/N$, which exceeds the optimal value W/N only by a factor of two. In the following part we will show that Algorithm HF really behaves the way this intuitive argument suggests. This is due to the fact that H^d is sharply concentrated around its expectation.

The following lemma shows that with high probability all heavy nodes reside rather close to the root of the IBT.

Lemma 8. *With probability $1 - o(1/N)$ all nodes of the IBT on level l with $l \geq k \cdot \ln((WN)/c)$ and $k > 2e$ are (c/N)-light (for $W \geq c/N$ and $c > 0$).*

Proof. Setting $l := k \cdot \ln((WN)/c)$ for $k > 1$ such that $l \in \mathbb{N}$ and applying Lemma 6 for $d = c/N$, we obtain

$$\begin{aligned}&\Pr[\exists\ d\text{-heavy node on level } l] \\ &\quad \le 2^l \cdot p_l^d \le 2^{k\ln((WN)/c)} \cdot \frac{c}{WN} \cdot \left(\frac{e}{k}\right)^{k\ln((WN)/c)} = \left(\frac{c}{WN}\right)^{1+k\ln k - k - k\ln 2} .\end{aligned}$$

A simple analysis of the exponent shows that $\Pr[\exists\ d\text{-heavy node on level } l] = o(1/N)$ for $k > 2e$.

Remark 9. The error term $o(1/N)$ in Lemma 8 is chosen rather arbitrarily and could be changed to $o(1/\text{poly}(N))$ without essential changes in the proofs.

Lemma 8 immediately yields a rather weak upper bound on the total number of heavy nodes, which we will improve later.

Corollary 10. *With probability $1 - o(1/N)$ it holds that $H^{c/N} = \mathcal{O}(N \log N)$ (for $c > 0$).*

Proof. Lemma 8 shows that with high probability only the first, say, $6 \cdot \ln(WN/c)$ levels contain heavy nodes. On every level the number of (c/N)-heavy nodes is at most WN/c since the weights of all nodes on the same level must have sum W.

Since we want to show that with high probability $H^{2W/N} \approx N$, Corollary 10 is still far off from our desired result, but we already know that the possibly infinite runtime of Algorithm HF until all subproblems are $(2W/N)$-light is $\mathcal{O}(N \log N)$ with probability $1 - o(1/N)$. In order to improve this result, we define a martingale and apply the method of bounded differences to show a sharp concentration around the mean.

For the definition of the martingale we denote by $\mathbb{F}_i := \sigma(\alpha_1, \ldots, \alpha_i)$ the σ-algebra generated by the random variables α_i (for $i \ge 1$) (see [Fel71, p. 116f]). Furthermore we set $\mathbb{F}_0 := \{\emptyset, \Omega\}$. The sequence $\mathbb{F}_0 \subset \mathbb{F}_1 \subset \cdots \subset \mathbb{F}$ forms a *filter*. As mentioned in [Fel71, p. 212f]

$$Z_i^d := \mathbb{E}[H^d \mid \mathbb{F}_i] = \mathbb{E}[H^d \mid \alpha_1, \ldots, \alpha_i]$$

defines a martingale, which is also called a *Doob-martingale*. It holds that $Z_0^d = \mathbb{E}[H^d]$.

Z_i^d is a function from Ω to $\mathbb{R}$. The intuitive interpretation of Z_i^d is as follows: Given a sequence $s := (\alpha_1, \ldots, \alpha_i, \alpha_{i+1}, \ldots) \in \Omega$ of general bisection parameters, $Z_i^d(s)$ tells us how many heavy nodes we expect in the IBT if the generalized bisections parameters of the first i bisections correspond to $s_i := (\alpha_1, \ldots, \alpha_i)$.

Let T_i denote the part of the IBT visited by Algorithm HF up to the i-th iteration (for example T_0 contains only the root of the tree). When we know s_i we can simulate Algorithm HF and compute T_i. Therefore, evaluating Z_i^d corresponds to calculating the expected value of H^d, given the tree T_i which is generated by the first i bisections of Algorithm HF. In order to capture this intuition, we use the following notation:

$$Z_i^d = \mathbb{E}[H^d \mid \mathbb{F}_i] =: \mathbb{E}[H^d \mid T_i] .$$

In the following lemma we show that $|Z^d_{i+1} - Z^d_i|$ is small. This a prerequisite for the application of the method of bounded differences.

Lemma 11. *For all $i \geq 0$ it holds that $|Z^d_{i+1} - Z^d_i| \leq 2$.*

Proof. Throughout this proof we assume that $s \in \Omega$ is fixed, and show that $|Z^d_{i+1}(s) - Z^d_i(s)| \leq 2$.

Since we know the prefixes s_i and s_{i+1} of s, we can compute the trees T_i and T_{i+1}. Let v denote the node that is bisected at the $(i+1)$-st iteration.

If v is light ($\mathsf{w}(v) < d$) the claim follows easily: All nodes not yet visited by Algorithm HF must also be light, because v is a heaviest leaf in the expanded part of the tree. Thus, we have already seen all heavy nodes and obtain $Z^d_{i+1} = Z^d_i = H^d$.

Now we assume that $\mathsf{w}(v) \geq d$ (in the sequel we write $w := \mathsf{w}(v)$ for short). Let v_1 and v_2 be the two nodes generated by the bisection of v. In addition, let $\mathsf{w}(v_1) = \alpha w$ and $\mathsf{w}(v_2) = (1-\alpha)w$ (without loss of generality $0.5 \leq \alpha \leq 1$).

We set $T_i = I \cup L \cup \{v\}$ and $T_{i+1} = I \cup L \cup \{v\} \cup \{v_1\} \cup \{v_2\}$, where I denotes the set of interior nodes of T_i and L denotes the set of leaves of T_i except v. T is the complete binary tree (as before) and $T(u)$ is the subtree of T beginning at node u. Moreover, we introduce the abbreviation $\overline{H}^d(u \mid T_i) := \sum_{x \in T(u)} \mathbb{E}[H^d_x \mid T_i]$ for the expected number of heavy nodes in $T(u)$ under the condition that we already know the prefix T_i of the IBT.

We observe that all nodes in I must be d-heavy. This yields

$$Z^d_i = \mathbb{E}[H^d \mid T_i] = \sum_{u \in T} \mathbb{E}[H^d_u \mid T_i] = |I| + \sum_{u \in L} \overline{H}^d(u \mid T_i) + \overline{H}^d(v \mid T_i)$$

$$Z^d_{i+1} = |I| + \sum_{u \in L} \overline{H}^d(u \mid T_i) + 1 + \overline{H}^d(v_1 \mid T_{i+1}) + \overline{H}^d(v_2 \mid T_{i+1}) \enspace ,$$

since $\mathbb{E}[H^d_v \mid T_{i+1}] = 1$, because v is heavy.

We bound $|Z^d_i - Z^d_{i+1}|$ by considering three cases and using estimates for $\overline{H}^d(\tau \mid T_i)$ (for $\tau \in \{v, v_1, v_2\}$). Note that, if u is a d-heavy leaf in T_i, we can use Theorem 7 and obtain $\overline{H}^d(u \mid T_i) = \frac{2}{d}\mathsf{w}(u) - 1$.

Case 1: $\alpha w < d$ In this case v_1 and v_2 are light. Therefore, it holds that $w < 2d$ and $\overline{H}^d(v_1 \mid T_{i+1}) = \overline{H}^d(v_2 \mid T_{i+1}) = 0$. It follows that

$$Z^d_i - Z^d_{i+1} = \overline{H}^d(v \mid T_i) - (1 + \overline{H}^d(v_1 \mid T_{i+1}) + \overline{H}^d(v_2 \mid T_{i+1})) = \frac{2}{d}w - 1 - 1 < 2 \enspace .$$

Furthermore, we obtain due to $w \geq d$ that $Z^d_i - Z^d_{i+1} \geq 0$.

The cases **Case 2:** $(1-\alpha)w < d$ **and** $\alpha w \geq d$ and **Case 3:** $(1-\alpha)w \geq d$ are similar.

The next lemma is basic probability theory, but we prefer to state it separately in order to render the proof of the main theorem more readable.

Lemma 12. *Let A and B be two events over a probability space Ω. If* $\Pr[B] = 1 - o(1/N)$ *then*

$$\Pr[A] \le \Pr[A \cap B] + o(1/N) \ .$$

Proof. Bound the terms in $\Pr[A] = \Pr[A|B]\Pr[B] + \Pr[A|\bar{B}]\Pr[\bar{B}]$.

We will use the following theorem from the method of bounded differences (see [McD89]):

Theorem 13. *Let $Z_0, Z_1, \dots$ be a martingale sequence such that $|Z_k - Z_{k-1}| \le c_k$ for each k, where c_k may depend on k. Then, for all $t \ge 0$ and any $\lambda > 0$*

$$\Pr[|Z_t - Z_0| \ge \lambda] \le 2\exp\left(-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}\right) \ .$$

Using this inequality we prove the next theorem:

Theorem 14. *With probability $1 - 2e^{-k^2/9} - o(1/N)$ it holds for $d = c/N$ with $c = O(1)$ and $k > 0$ that*

$$|H^d - \mathbb{E}[H^d]| < k\sqrt{\frac{2W}{c}} \cdot \sqrt{N} \ .$$

Proof. First we note that if $H^d \le t$ then $Z_t^d = H^d$, because after t steps Algorithm HF has certainly bisected all H^d heavy nodes. Hence, only light nodes remain and we know the exact value of H^d, since we have already seen all heavy nodes. Consequently, $H^d = \mathbb{E}[H^d \mid T_t] = Z_t^d$ in this case.

We have shown in Lemma 11 that $|Z_i^d - Z_{i+1}^d| \le 2 =: c_i$. Now we apply Theorem 13 for $\lambda = N^\gamma$ with $\gamma = 0.5 + \varepsilon$ ($\varepsilon > 0$). For $t' = \mathcal{O}(N \log N)$ we obtain from Corollary 10 and Lemma 12:

$$\begin{aligned}
\Pr[|H^d - \mathbb{E}[H^d]| \ge N^\gamma] &\le \Pr[|H^d - \mathbb{E}[H^d]| \ge N^\gamma \ \wedge \ H^d \le t'] + o(1/N) \\
&= \Pr[|Z_{t'}^d - Z_0^d| \ge N^\gamma \ \wedge \ H^d \le t'] + o(1/N) \\
&\le \Pr[|Z_{t'}^d - Z_0^d| \ge N^\gamma] + o(1/N) \\
&\le 2 \cdot \exp\left(-\frac{N^{2\gamma}}{2\sum_{i=1}^{t'} c_i^2}\right) + o(1/N) = o(1/N) \ .
\end{aligned}$$

Now we know that with high probability $H^d < t'' := 2W/d - 1 + N^\gamma = (2W/c)N - 1 + N^\gamma$. If we apply Theorem 13 one more time using this estimate for H^d and performing similar calculations we get (for N large enough)

$$\begin{aligned}
\Pr[|H^d - \mathbb{E}[H^d]| \ge k\sqrt{(2W/c)N}] &\le \Pr[|Z_{t''}^d - Z_0^d| \ge k\sqrt{(2W/c)N}] + o(1/N) \\
&< 2e^{-k^2/9} + o(1/N) \ .
\end{aligned}$$

The results for the random variable $H^{c/N}$ immediately yield the desired consequences for the performance of Algorithm HF.

Theorem 3. *If Algorithm HF starts with a problem of weight W and produces a set P of N subproblems after $N-1$ bisections then for $\varepsilon = 9\sqrt{\ln(N)/N}$*

$$\Pr\left[(2-\varepsilon)\,\frac{W}{N} \le W_{\max}^{N} < (2+\varepsilon)\,\frac{W}{N}\right] = 1 - o(1/N) \ ,$$

where $W_{\max}^{N} := \max\{\mathsf{w}(p) \ : \ p \in P\}$ denotes the weight of the heaviest subproblem in P.

Proof. First we show that $p^{+} := \Pr\left[W_{\max}^{N} \ge (2+\varepsilon)\,\frac{W}{N}\right] = o(1/N)$.

For $c' := (2+\varepsilon)W$ and $\beta := \frac{c'}{N}$ it holds that $p^{+} = \Pr\left[W_{\max}^{N} \ge \beta\right] = \Pr[H^{\beta} > N-1]$, because Algorithm HF expands heavy subproblems before light subproblems. Due to Theorem 7 we have $\mathbb{E}[H^{\beta}] = \frac{2}{\beta}W - 1 = \frac{2}{2+\varepsilon}N - 1$ and, thus, we can rewrite p^{+} as $p^{+} = \Pr\left[H^{\beta} - \mathbb{E}[H^{\beta}] > \frac{\varepsilon}{2+\varepsilon}N\right]$.

Since (for N sufficiently large) $\frac{\varepsilon}{2+\varepsilon}N \ge \frac{\varepsilon}{2.5}N \ge 3.6 \cdot \sqrt{\ln(N)\cdot N}$ and $1 \ge \sqrt{2/(2+\varepsilon)} = \sqrt{2W/c'}$, we obtain using Theorem 14 for $k = 3.6\sqrt{\ln(N)}$ that

$$p^{+} \le \Pr\left[|H^{\beta} - \mathbb{E}[H^{\beta}]| \ge k\sqrt{2W/c'}\cdot\sqrt{N}\right] = o(1/N) \ .$$

The proof for $p^{-} := \Pr\left[W_{\max}^{N} < (2-\varepsilon)\frac{W}{N}\right] = o(1/N)$ is similar.

Simulation results complement Theorem 3 and show that also for small N Algorithm HF exhibits a good average-case behavior. Already for $N \approx 100$ the predicted value for $W_{\max}^{N}$ from Theorem 3 and the 'real' value seen in simulations differ only by a relative error of less than one percent. The standard deviation is tiny, as one could expect due to Theorem 14.

3.5 Simplified Parallel Version of Algorithm HF

The proofs did not depend on the exact order in which Algorithm HF processes the nodes in the tree. We only made use of the fact that heavy nodes are processed before light nodes. Hence, it would suffice to run Algorithm HF until all subproblems are d-light with $d = (2+\varepsilon)\frac{W}{N}$ and $\varepsilon = 9\sqrt{\ln(N)/N}$. In this case we know that the number of generated subproblems is at most N with high probability. This variant of Algorithm HF does not need priority queues and requires only constant time per iteration to find the node q which shall be bisected next. Furthermore, there is a natural parallel version of the modified Algorithm HF, because for each subproblem it can be decided independently if further bisections should be applied to it. The sharp concentration result shows that this modification makes no substantial difference for the quality of the load distribution.

4 Conclusion

In this paper we have analyzed the Heaviest First Algorithm for dynamic load balancing. From a practitioners point of view this algorithm is applicable to a wide range of problems, because it only depends on the fact that some bisection algorithm is available. Additionally, it is very easy to implement and can be efficiently parallelized.

Our analysis focused on the average-case performance of Algorithm HF. Under rather pessimistic assumptions concerning the distribution of the bisection parameters we showed that the size of the maximum subproblem produced by Algorithm HF differs from the best attainable value only by a factor of roughly two. We believe that this provides a reasonable explanation for the good performance of Algorithm HF in simulations and practical applications. In order to improve the understanding of Algorithm HF further it would be interesting to transfer our results to more general distributions.

References

[AST90] N. Alon, P. Seymour, and R. Thomas. A separator theorem for nonplanar graphs. *Journal of the AMS*, 3(4):801–808, 1990.

[BEE98] Stefan Bischof, Ralf Ebner, and Thomas Erlebach. Load Balancing for Problems with Good Bisectors, and Applications in Finite Element Simulations. In *Proceedings of the Fourth International EURO-PAR Conference on Parallel Processing EURO-PAR'98*, volume 1470 of *LNCS*, pages 383–389, Berlin, 1998. Springer-Verlag. Also available as technical report (http://www14.in.tum.de/berichte/1998/TUM-I9811.ps.gz).

[BEE99] Stefan Bischof, Ralf Ebner, and Thomas Erlebach. Parallel Load Balancing for Problems with Good Bisectors. In *Proceedings of the 2nd Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing IPPS/SPDP'99*, Los Alamitos, 1999. IEEE Computer Society Press. To appear.

[Bon93] T. Bonk. A New Algorithm for Multi-Dimensional Adaptive Numerical Quadrature. In W. Hackbusch, editor, *Adaptive Methods – Algorithms, Theory and Applications: Proceedings of the 9th GAMM Seminar, Kiel, January 22–24, 1993*, pages 54–68, 1993.

[Fel71] William Feller. *An Introduction to Probability Theory and its Applications. Volume II.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, Chichester, second edition, 1971.

[HS94] Reiner Hüttl and Michael Schneider. Parallel Adaptive Numerical Simulation. SFB-Bericht 342/01/94 A, Technische Universität München, 1994.

[LT79] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

[McD89] C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141 of *London Math. Soc. Lecture Note Series*, pages 148–188. Cambridge University Press, Cambridge, 1989.

[RR93] H. Regler and U. Rüde. Layout optimization with Algebraic Multigrid Methods (AMG). In *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods, Copper Mountain, April 4–9, 1993*, Conference Publication, pages 497–512. NASA, 1993.