TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Load Balancing for Problems with Good Bisectors, and Applications in Finite Element Simulations:
Worst-case Analysis and Practical Results

Stefan Bischof, Ralf Ebner, Thomas Erlebach

TUM-I9811
SFB-Bericht Nr. 342/05/98 A
Mai 98

TUM–INFO–05-I9811-150/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Load Balancing for Problems with Good Bisectors, and Applications in Finite Element Simulations:
Worst-case Analysis and Practical Results

Stefan Bischof Ralf Ebner
Thomas Erlebach

Institut für Informatik
Technische Universität München
D-80290 München, Germany

{bischof|ebner|erlebach}@in.tum.de

http://www{mayr|zenger}.in.tum.de/

May 26, 1998

Abstract

This paper studies load balancing issues for classes of problems with certain bisection properties. A class of problems has α-bisectors if every problem in the class can be subdivided into two subproblems whose weight is not smaller than an α-fraction of the original problem. It is shown that the maximum weight of a subproblem produced by Algorithm HF, which partitions a given problem into N subproblems by always subdividing the problem with maximum weight, is at most a factor of $\lfloor 1/\alpha \rfloor \cdot (1-\alpha)^{\lfloor 1/\alpha \rfloor - 2}$ greater than the theoretical optimum (uniform partition). This bound is proved to be asymptotically tight. Two strategies to use Algorithm HF for load balancing distributed hierarchical finite element simulations and experimental results are presented. For this purpose, a certain class of weighted binary trees representing the load of such applications is shown to have 1/4-bisectors. This establishes a performance guarantee of 9/4 for load balancing in this case.

1 Introduction

Load balancing is one of the major research issues in the context of parallel computing. Irregular problems are often difficult to tackle in parallel because they tend to overload some processors while leaving other processors nearly idle. For these applications it is very important to find methods for obtaining a balanced distribution of load. Usually, the load is created by processes that are part of a (parallel) application program. During the run of the application, these processes perform certain calculations independently but have to communicate intermediate results or other data using message passing. One is usually interested in achieving a balanced load distribution in order to minimize the execution time of the application or to maximize system throughput.

Load balancing problems have been studied for a huge variety of models, and many different solutions regarding strategies and implementation mechanisms have been proposed. A good overview of recent work can be obtained from [SHK95], for example. [SS97] reviews ongoing research on dynamic load balancing, emphasizing the presentation of models and strategies within the framework of general classification schemes.

In this paper we study load balancing for a very general class of problems. The only assumption we make is that all problems in the class have a certain bisection property. Such classes of problems arise, for example, in the context of distributed hierarchical finite element simulations. We show how our general results can be applied to numerical applications in several ways. The remainder of the paper is structured as follows. In Section 2 we present and analyze a very general and simple algorithm that computes a good load distribution for classes of problems with α-bisectors. Section 3 briefly explains distributed finite element simulations with recursive substructuring. Two strategies for applying the algorithm from Section 2 to these applications are discussed. Section 4 shows that certain weighted trees, which model the load of applications in numerical simulations like the one discussed in Section 3, have 1/4-bisectors. This implies a performance guarantee[1] of 9/4 for load balancing these applications. Section 5 summarizes our results.

2 Using Bisectors for Load Balancing

In many applications a computational problem cannot be divided into many small problems as required for an efficient parallel solution directly. Instead,

[1] An algorithm has performance guarantee ρ if the maximum load produced by the algorithm is at most a factor of ρ larger than the maximum load of an optimum solution.

```
Input: problem p, positive integer 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;
    output P;
end.
```

Figure 1: Algorithm HF (Heaviest Problem First)

a strategy similar to *divide and conquer* is used repeatedly to divide problems into smaller subproblems. We refer to the division of a problem into two smaller subproblems as *bisection*. Assuming a weight function w that measures the resource demand, a problem p cannot always be bisected into two subproblems p_1 and p_2 of equal weight $\mathsf{w}(p)/2$. For many classes of problems, however, there is a bisection method that guarantees that the weights of the two obtained subproblems do not differ too much. The following definition captures this concept more precisely.

Definition 1 *Let* $0 < \alpha \le \frac{1}{2}$. *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* $\mathsf{w}(p_1), \mathsf{w}(p_2) \in [\alpha \mathsf{w}(p); (1-\alpha)\mathsf{w}(p)]$.

Note that this definition requires for the sake of simplicity that all problems in $\mathcal{P}$ can be bisected, whereas in practice this is not the case for problems whose weight is below a certain threshold. We assume, however, that the problem to be divided among the processors is big enough to allow further bisections until the number of subproblems is equal to the number of processors. This is a reasonable assumption for most relevant parallel applications. A definition very similar to ours (α-splitting) is used by Kumar, Grama, and Rao [KV87, KGV94] [KGGK94, pp. 315–318] under the assumption that the weight of a problem is unknown to the load balancing algorithm.

2.1 Tight Analysis of Algorithm HF

Figure 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 application of α-bisectors to the heaviest remaining subproblem. 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 gives a worst-case bound on the ratio between the maximum weight among the N subproblems produced by Algorithm HF and this ideal weight $\mathsf{w}(p)/N$.

Theorem 2 *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 p_1, ..., p_N such that*

$$\max_{1 \le i \le N} \mathsf{w}(p_i) \le \frac{\mathsf{w}(p)}{N} \cdot \left\lfloor \frac{1}{\alpha} \right\rfloor \cdot (1-\alpha)^{\lfloor \frac{1}{\alpha} \rfloor - 2} .$$

Proof: It is obvious that Algorithm HF uses $N-1$ bisections to partition p into N subproblems. In the following we show that the stated inequality regarding the maximum weight among these subproblems holds.

We introduce the bisection tree T to represent the run of the algorithm on input p and N. The root of T is the problem p. If the algorithm bisects a problem q into q_1 and q_2, nodes q_1 and q_2 are added to T as children of node q. In the end, T has N leaves, which correspond to the N subproblems computed by the algorithm, and all problems that were bisected by the algorithm appear as internal nodes with exactly two children. Figure 2 gives an example of a bisection tree for a problem of weight 44 from a class of problems with $\frac{1}{7}$-bisectors. We follow the convention of drawing the node with greater weight among two children of the same parent as a left child of that parent.

The following properties hold for bisection trees arising from classes of problems with α-bisectors. Let the leaves of the tree be p_1, ..., p_N and let $m := \max_{1 \le i \le N} \mathsf{w}(p_i)$.

(a) $\mathsf{w}(q) \ge m$ for all internal nodes q

(b) $\mathsf{w}(q) \ge \frac{1}{1-\alpha}\mathsf{w}(q')$ if q' is a child of q

(a) holds because the algorithm always bisects a subproblem of maximum weight; since one of the p_i has weight m, there must have been at least one subproblem of weight $\ge m$ during the whole run of the algorithm, and thus the algorithm never bisected a problem of weight $< m$. (b) follows directly from $\mathsf{w}(q') \le (1-\alpha)\mathsf{w}(q)$, which holds because the algorithm uses α-bisectors.

Now remove from the bisection tree all internal nodes which are not parent of a leaf. This partitions the bisection tree into a number of disjoint branches,

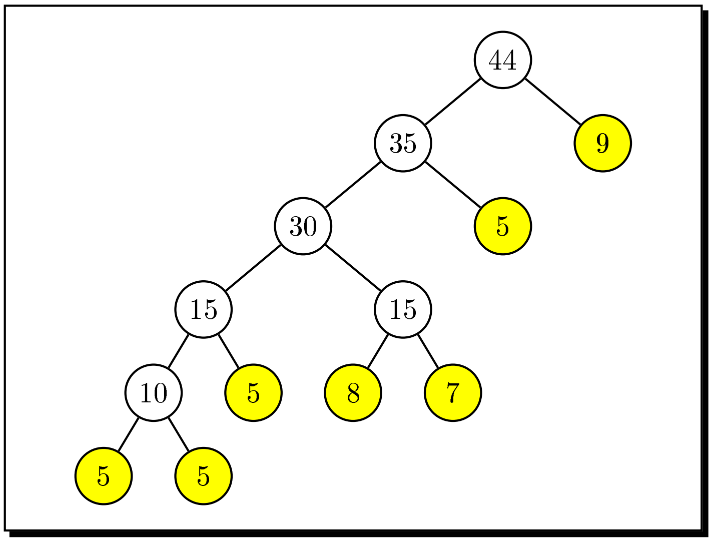

Figure 2: Example of a bisection tree

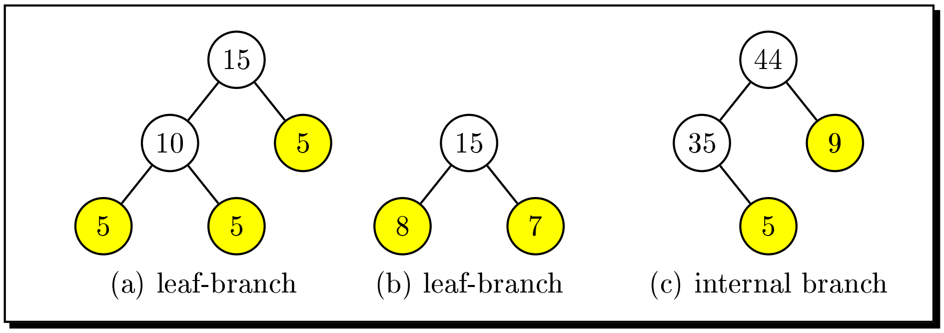

Figure 3: Branches obtained from the example tree

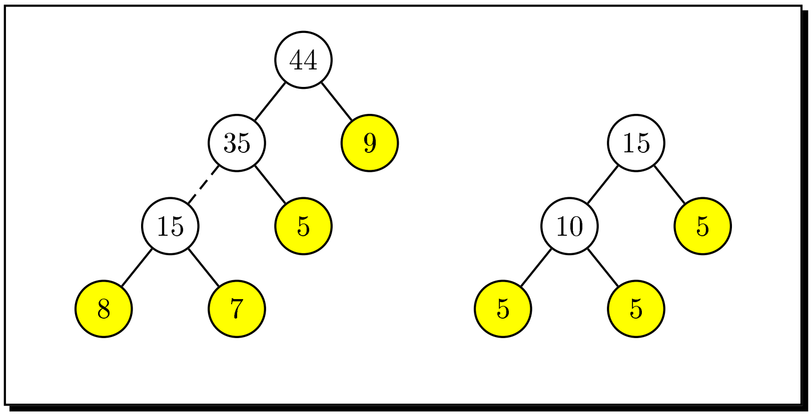

Figure 4: Composed leaf-branches obtained from the example tree

whose shape can be that of a *leaf-branch* (one of the internal nodes of the branch has two leaf children) or that of an *internal branch* (all the internal nodes of the branch have exactly one leaf child). The branches obtained from the example tree of Figure 2 are shown in Figure 3. Our goal is to derive a lower bound for the average weight of the leaves in each branch.

Consider a leaf-branch with k internal nodes, $k \geq 1$. Denote its internal nodes by $v_1, v_2, \ldots, v_k$ such that v_{i+1} is the parent of v_i for $1 \leq i \leq k-1$. Furthermore, let c_i denote the leaf child of v_i for $2 \leq i \leq k$, and let c_0 and c_1 denote the leaf children of v_1. As (a) implies $\mathsf{w}(v_1) \geq m$, we have by (b) $\mathsf{w}(v_i) \geq \left(\frac{1}{1-\alpha}\right)^{i-1} m$ for $1 \leq i \leq k$ and $\mathsf{w}(c_i) \geq \alpha \left(\frac{1}{1-\alpha}\right)^{i-1} m$ for $1 \leq i \leq k$. The average weight of the leaves $c_0, \ldots, c_k$ can now be bounded from below as follows:

$$\begin{aligned}
\frac{1}{k+1} \sum_{i=0}^{k} \mathsf{w}(c_i) &\geq \frac{1}{k+1} \left(m + \sum_{i=2}^{k} \alpha \left(\frac{1}{1-\alpha} \right)^{i-1} m \right) \\
&= \frac{1}{k+1} \left(m(1-\alpha) + \alpha m \sum_{i=0}^{k-1} \frac{1}{(1-\alpha)^i} \right) \\
&= \frac{1}{k+1} \left(m(1-\alpha) + m(\alpha - 1) \left(1 - \frac{1}{(1-\alpha)^k} \right) \right) \\
&= \frac{m}{(k+1)(1-\alpha)^{k-1}} .
\end{aligned}$$

If there are internal branches, we do not deal with them separately but instead attach them to leaf-branches. For example, one can consider the leaf-branches one by one and attach to each leaf-branch all internal branches that intersect the path from the leaf-branch to the root of the bisection tree and that have not been attached to a different leaf-branch beforehand. Here, attaching an internal branch to a leaf-branch means making the root of the leaf-branch a child of the bottom-most internal node of the internal branch, resulting in a new leaf-branch. We call the leaf-branches obtained by attaching zero or more internal branches to an original leaf-branch *composed leaf-branches*. Observe that conditions (a) and (b) are satisfied for these composed leaf-branches as well. Hence, the lower bound above also pertains to the average weight of the leaves in such a composed leaf-branch. The bisection tree from Figure 2 contained two leaf-branches and one internal branch as illustrated in Figure 3. Attaching the internal branch to one of the leaf-branches gives the composed leaf-branches shown in Figure 4.

As every leaf of the bisection tree appears in exactly one composed leaf-branch, we conclude that $\min_{k \in \mathbb{N}} \frac{m}{(k+1)(1-\alpha)^{k-1}}$ is a lower bound on the average weight of all leaves in the bisection tree. Therefore, we obtain

$$\mathsf{w}(p) = \sum_{i=1}^{N} \mathsf{w}(p_i) \geq Nm \min_{k \in \mathbb{N}} \frac{1}{(k+1)(1-\alpha)^{k-1}} \,. \tag{1}$$

Besides, we observe that

$$\min_{k \in \mathbb{N}} \frac{1}{(k+1)(1-\alpha)^{k-1}} = \frac{1}{\max_{k \in \mathbb{N}} \left((k+1)(1-\alpha)^{k-1}\right)} \,, \tag{2}$$

and we claim that $(k+1)(1-\alpha)^{k-1}$ as a function of $k \in \mathbb{N}$ is maximized for $\hat{k} = \lfloor 1/\alpha \rfloor - 1$. To see this, let $f(k) = (k+1)(1-\alpha)^{k-1}$ and consider the ratio $f(k)/f(k-1) = (1-\alpha)(k+1)/k$. We obtain:

$$\frac{f(k)}{f(k-1)} = \begin{cases} > 1 & \text{for } k < \frac{1}{\alpha} - 1 \\ = 1 & \text{for } k = \frac{1}{\alpha} - 1 \\ < 1 & \text{for } k > \frac{1}{\alpha} - 1 \end{cases} \,.$$

For a fixed value of α, $f(k)$ is monotone increasing from $k = 1$ to $k = \lfloor 1/\alpha \rfloor - 1$ and monotone decreasing for larger values of k. If $1/\alpha$ is not an integer greater than 2, $f(k)$ is maximum only for $k = \lfloor 1/\alpha \rfloor - 1$. If $1/\alpha$ is an integer greater than 2, $f(k)$ is maximum for $k = \lfloor 1/\alpha \rfloor - 1$ and for $k = \lfloor 1/\alpha \rfloor - 2$. In any case we have $\max_{k \in \mathbb{N}} \left((k+1)(1-\alpha)^{k-1}\right) = f(\lfloor 1/\alpha \rfloor - 1) = \lfloor 1/\alpha \rfloor \cdot (1-\alpha)^{\lfloor 1/\alpha \rfloor - 2}$, and the theorem follows with (1) and (2). □

For some values of α, Table 1 gives worst-case bounds on the ratio between $\max_{1 \leq i \leq N} \mathsf{w}(p_i)$ and $\frac{\mathsf{w}(p)}{N}$ as well as a value of k for which $(k+1)(1-\alpha)^{k-1}$ is maximized. These bounds show that the worst-case deviation from the ideal load distribution, in which $\mathsf{w}(p_i) = \frac{\mathsf{w}(p)}{N}$ for all $1 \leq i \leq N$, is bounded by a small constant for a wide range of α. Note that in many cases an ideal load distribution cannot be achieved by any algorithm.

Corollary 3 *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 p_1, ..., p_N such that*

$$\max_{1 \leq i \leq N} \mathsf{w}(p_i) \leq \frac{\mathsf{w}(p)}{N} \cdot \frac{1}{\mathrm{e}(1-\alpha)^2 \ln \frac{1}{1-\alpha}} \,.$$

Proof: In the proof of Theorem 2 it was shown that

$$\max_{1 \leq i \leq N} \mathsf{w}(p_i) \leq \frac{\mathsf{w}(p)}{N} \cdot \max_{k \in \mathbb{N}} ((k+1)(1-\alpha)^{k-1}) \,. \tag{3}$$

Table 1: Worst-case ratio of Algorithm HF for different values of α

α	k	ratio	α	k	ratio	α	k	ratio
0.02	49	18.96	0.21	3	2.50	0.31	2	2.07
0.04	24	9.78	0.22	3	2.43	0.32	2	2.04
0.06	15	6.73	0.23	3	2.37	0.325	2	2.025
0.08	11	5.21	0.24	3	2.31	0.33	2	2.01
0.10	8	4.30	0.25	2	2.25	0.331	2	2.007
0.12	7	3.72	0.26	2	2.22	0.332	2	2.004
0.14	6	3.29	0.27	2	2.19	0.333	2	2.001
0.16	5	2.99	0.28	2	2.16	0.334	1	2.00
0.18	4	2.76	0.29	2	2.13	0.40	1	2.00
0.20	3	2.56	0.30	2	2.10	0.50	1	2.00

Observe that the term maximized on the right hand side of this inequality is a differentiable function of k. Therefore, we define $f : \mathbb{R}^+ \to \mathbb{R}^+$ by $f(k) = (k+1)(1-\alpha)^{k-1}$. The derivative of f is:

$$f'(k) = (1-\alpha)^{k-1} \cdot ((k+1)\ln(1-\alpha) + 1) .$$

The derivative is zero for

$$(k+1)\ln(1-\alpha) = -1 ,$$

which is the case only for $\hat{k} = \frac{-1}{\ln(1-\alpha)} - 1$.

Substituting yields $f(\hat{k}) = \left(\mathrm{e}(1-\alpha)^2 \ln \frac{1}{1-\alpha}\right)^{-1}$, and this is the global maximum of f. Hence,

$$\max_{k\in\mathbb{N}}((k+1)(1-\alpha)^{k-1}) \leq \frac{1}{\mathrm{e}(1-\alpha)^2 \ln \frac{1}{1-\alpha}} ,$$

and the corollary follows directly from inequality (3). $\square$

In Figure 5 the worst-case bound for the ratio between $\max_{1\leq i\leq N} \mathsf{w}(p_i)$ and $\mathsf{w}(p)/N$ from Theorem 2 as well as the continuous approximation of this bound $\left(\mathrm{e}(1-\alpha)^2 \ln \frac{1}{1-\alpha}\right)^{-1}$ from Corollary 3 are plotted for $0.08 \leq \alpha \leq 0.5$. It turns out that the continuous approximation of the bound matches the discrete bound (cf. Table 1) almost exactly for $\alpha \leq 0.3$. To complete this part of our analysis, we observe that the exact upper bound on the ratio between $\max_{1\leq i\leq N} \mathsf{w}(p_i)$ and $\mathsf{w}(p)/N$ is 2 for $\alpha \geq 1/3$.

Now we give a lower bound for the worst-case ratio between the maximum weight subproblem generated by Algorithm HF and the ideal value given by a uniform partition. This will show that the upper bound from Theorem 2 is tight.

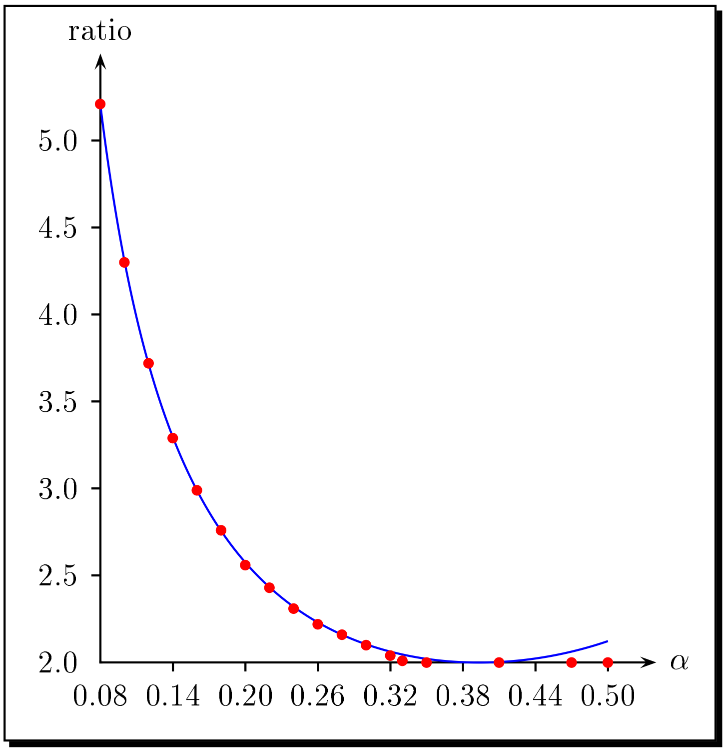

Figure 5: Plot of discrete (dotted) and continuous worst-case bounds

Theorem 4 *For each* $0 < \alpha \leq \frac{1}{2}$ *there exists a class of problems* $\mathcal{Q}^\alpha$ *that has* α*-bisectors and contains a family of problems* $(q^l)_{l\in\mathbb{N}}$ *such that*

$$\lim_{l\to\infty} \frac{\max_{1\leq i\leq N_l} \mathsf{w}(q_i^l)}{\frac{\mathsf{w}(q^l)}{N_l}} = \left\lfloor \frac{1}{\alpha} \right\rfloor \cdot (1-\alpha)^{\left\lfloor \frac{1}{\alpha} \right\rfloor - 2},$$

where $N_l = \lfloor 1/\alpha \rfloor \cdot 2^l - 1$ *and* $q_1^l, q_2^l, \ldots, q_{N_l}^l$ *are the subproblems generated by Algorithm HF on input* q^l *and* N_l*.*

Proof: Let $\mathcal{Q}^\alpha$ be a class of problems with weight function $\mathsf{w} : \mathcal{P} \to \mathbb{R}^+$ and the following properties:

(a) each $q \in \mathcal{Q}^\alpha$ with $\mathsf{w}(q) > 1$ can only be partitioned into 2 subproblems of weight $\mathsf{w}(q)/2$ each

(b) each $q \in \mathcal{Q}^\alpha$ with $\mathsf{w}(q) \leq 1$ can only be partitioned into 2 subproblems of weight $(1-\alpha)\mathsf{w}(q)$ and $\alpha\mathsf{w}(q)$

(c) for every $l \in \mathbb{N}$ there is a problem $q^l \in \mathcal{Q}^\alpha$ of weight $\mathsf{w}(q^l) = 2^l$

Clearly, $\mathcal{Q}^\alpha$ has α-bisectors according to Definition 1.

Let $k = \lfloor 1/\alpha \rfloor - 2$. For a given $l \in \mathbb{N}$, choose a problem $q^l \in \mathcal{Q}^\alpha$ of weight 2^l and let $N_l = (k+2)2^l - 1$. On input q^l and N_l, Algorithm HF proceeds as follows. After the first $2^l - 1$ bisections, there are 2^l subproblems of weight 1 each. We assign level 0 to these problems and call them active. As the weight of each active problem is ≤ 1, the next 2^l bisections performed by the algorithm subdivide all active problems on level 0 and generate subproblems of weight $1 - \alpha$ and α, which are assigned level 1. Now the 2^l problems on level 1 with weight $1 - \alpha$ become active. This process is repeated such that in phase i, $i \geq 0$, the algorithm subdivides all 2^l problems of weight $(1-\alpha)^i$ on level i. At the end of phase i there are $(i+2)2^l$ subproblems altogether. The subdivision process is finished when exactly one active subproblem on level k of weight $(1-\alpha)^k$ remains.

To ensure that the algorithm indeed subdivides the active problems on level i in phase i for all $0 \leq i \leq k$ and not the heaviest inactive problem, which has weight α, it is required that $(1-\alpha)^k \geq \alpha$. This is obvious for $k = 0, 1$. For $k \geq 2$, recall that the series $(1 - 1/(k+1))^k$ converges strictly decreasing from above to e^{-1} and note that $k = \lfloor 1/\alpha \rfloor - 2$ implies $\alpha \leq 1/(k+2) \leq 1/4$. Hence,

$$(1-\alpha)^k \geq \left(1 - \frac{1}{k+2}\right)^k \geq \left(1 - \frac{1}{k+1}\right)^k \geq \frac{1}{\mathrm{e}} \geq \frac{1}{4} \geq \alpha\,.$$

In the end, Algorithm HF has generated $(k+2)2^l - 1$ subproblems and a maximum weight of $(1-\alpha)^k$. Thus,

$$\max_{1 \leq i \leq N_l} \mathsf{w}(q_i^l) = \frac{\mathsf{w}(q^l)}{N_l} \cdot (1-\alpha)^k (k + 2 - 2^{-l})\,,$$

and the assertion of the theorem follows by substituting $k = \lfloor 1/\alpha \rfloor - 2$ and taking into account $\lim_{l \to \infty} 2^{-l} = 0$. $\square$

2.2 A Better Bound for Small N

Note that the bound of Theorem 2 is independent of N, the number of desired subproblems. Although we have shown that this bound is tight asymptotically, it is possible to obtain a better bound if N is sufficiently small. Again, we will show that this improved bound is tight. To establish this result, we need the following

Lemma 5 *Let $\alpha \leq 1/5$, $2 \leq k \leq 1/\alpha$. Then, with the assumptions of Theorem 2, for any leaf-branch of a bisection tree with k leaves $p_1, p_2, \ldots, p_k$ and root p:*

$$\max_{1 \leq i \leq k} \mathsf{w}(p_i) \leq \mathsf{w}(p)(1-\alpha)^{k-1}.$$

Proof: If all bisections are exact α-bisections we conclude that the maximum weight subproblem generated by Algorithm HF has weight $\mathsf{w}(p)(1-\alpha)^{k-1}$ since $\alpha < (1-\alpha)^{k-1}$ for $k \leq 1/\alpha$ and $\alpha \leq 1/5$. Clearly, the upper bound remains valid if the maximum weight subproblem is the leftmost leaf of the leaf-branch.

Therefore, we consider the case that the maximum weight subproblem does not result from the last bisection step. Let $m := \max_{1 \leq i \leq k} \mathsf{w}(p_i)$. The combined weight of the maximum weight leaf and the 2 leaves generated in the last bisection step is at least $2m$. The total weight of the remaining leaves can be bounded from below by

$$m \left(\left(\frac{1}{1-\alpha} \right)^{k-3} - 1 \right)$$

using the same argument as in the proof of Theorem 2. Thus,

$$m \leq \frac{\mathsf{w}(p)}{\left(\frac{1}{1-\alpha}\right)^{k-3} + 1},$$

and it remains to show that the right hand side of this inequality is no more than $\mathsf{w}(p)(1-\alpha)^{k-1}$. But since $\alpha \leq 1/5$ and $k \leq 1/\alpha$ we have

$$\begin{aligned} 1 &\leq 0.64 + e^{-1} \\ &\leq (1-\alpha)^2 + (1-\alpha)^{k-1} \\ &\leq \left(\left(\frac{1}{1-\alpha} \right)^{k-3} + 1 \right) (1-\alpha)^{k-1}. \end{aligned}$$ □

Theorem 6 *Let $\mathcal{P}$ be a class of problems with weight function $\mathsf{w} : \mathcal{P} \to \mathbb{R}^+$ that has α-bisectors and assume $\alpha \leq 1/5$. Given a problem $p \in \mathcal{P}$ and a positive integer $N \leq 1/\alpha$, Algorithm HF uses $N-1$ bisections to partition p into N subproblems $p_1, \ldots, p_N$ such that*

$$\max_{1 \leq i \leq N} \mathsf{w}(p_i) \leq \mathsf{w}(p)(1-\alpha)^{N-1}.$$

Proof: We will show that the worst-case bisection tree is a single leaf-branch if the assumptions of the theorem hold. The claim then follows immediately from the previous lemma.

Let us assume that the bisection tree generated by the run of Algorithm HF is not a single leaf-branch. Consequently, the bisection tree has internal nodes which are not parent of a leaf. We call these nodes *cut-nodes*. Let $m := \max_{1 \leq i \leq N} \mathsf{w}(p_i)$. Observe that a cut-node has weight at least $2m$.

If there are 2 or more cut-nodes we distinguish two cases. First, assume that there are 2 cut-nodes such that one is neither an ancestor nor a descendant of the other. Then their combined weight is at least $4m$ and thus $m \leq (1/4)\mathsf{w}(p)$. But we have $1/4 < e^{-1} < (1-\alpha)^{N-1}$ by the assumptions of the theorem. If there are 2 cut-nodes c_1 and c_2 such that c_1 is an ancestor of c_2, we know that c_1 is the parent of an internal node not on the path between c_1 and c_2 and thus the weight of c_1 is at least $3m$. We conclude that $m \leq (1/3)\mathsf{w}(p) < e^{-1}\mathsf{w}(p)$ in this case.

Now consider the case that there is exactly one cut-node c. If the maximum weight leaf is not in the subtree rooted at c we conclude $\mathsf{w}(p) \geq 3m$ and finish the proof for this case as above. Otherwise, let the children of c be x and y and assume without loss of generality that the maximum weight leaf is contained in the leaf-branch rooted at x. Denote the number of bisection steps in the leaf-branch rooted at x (y) by d_x (d_y), and let N' denote the number of leaves in the subtree rooted at c. Observe that $N' \geq 4$, $1 \leq d_x, d_y \leq N'-3$ and $d_x + d_y = N'-2$. As we have $\alpha \leq 1/5$ and the maximum weight leaf is contained in the leaf-branch rooted at x, we conclude $\mathsf{w}(x) \geq m(\frac{1}{1-\alpha})^{d_x}$ using Lemma 5. Since any internal node in the leaf-branch rooted at y has weight at least m, we have $\mathsf{w}(y) \geq m(\frac{1}{1-\alpha})^{d_y-1}$ as in the proof of Theorem 2. Combining these two bounds and substituting $d_y = N'-2-d_x$ yields:

$$\mathsf{w}(x) + \mathsf{w}(y) \geq m\left(\left(\frac{1}{1-\alpha}\right)^{d_x} + \left(\frac{1}{1-\alpha}\right)^{N'-3-d_x}\right) \geq m\left(\frac{1}{1-\alpha}\right)^{N'-1},$$

where the last inequality is equivalent to $(1-\alpha)^{N'-1-d_x} + (1-\alpha)^{d_x+2} \geq 1$. This can be shown to hold for $\alpha \leq 1/5$ by a straightforward calculation using analytic techniques. Hence, using $\mathsf{w}(c) = \mathsf{w}(x) + \mathsf{w}(y)$ we have

$$\mathsf{w}(c) \geq m\left(\frac{1}{1-\alpha}\right)^{N'-1}. \tag{4}$$

Assume that there are d_z nodes, $d_z \geq 0$, above c on the path from c to the root p of the bisection tree. Observe that

$$\mathsf{w}(p) \geq \mathsf{w}(c)\left(\frac{1}{1-\alpha}\right)^{d_z}. \tag{5}$$

As $N = N' + d_z$, Equations (4) and (5) imply $m \leq \mathsf{w}(p)(1-\alpha)^{N-1}$. □

It is easy to verify $N(1-\alpha)^{N-1} \leq \lfloor\frac{1}{\alpha}\rfloor \cdot (1-\alpha)^{\lfloor\frac{1}{\alpha}\rfloor-2}$ for $N \leq \lfloor\frac{1}{\alpha}\rfloor$ observing that the left-hand side of this inequality is monotone increasing

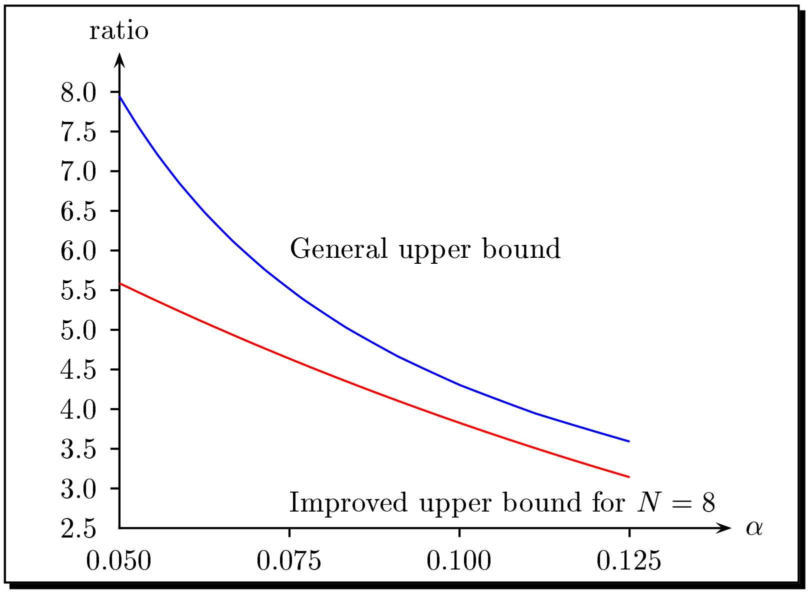

Figure 6: Comparison of general and improved upper bound

from $N = 1$ to $N = \left\lfloor \frac{1}{\alpha} \right\rfloor$ and the inequality holds trivially for the latter value of N. Figure 6 compares the general with the improved upper bound on the ratio between $\max_{1 \leq i \leq N} \mathsf{w}(p_i)$ and $\frac{\mathsf{w}(p)}{N}$ for $N = 8$.

Let $\hat{\alpha}$ be the real root of the equation $(1 - \alpha)^2 + (1 - \alpha)^3 - 1 = 0$. It can be shown that $\hat{\alpha} \approx 0.245122$ is the largest possible value for α in Lemma 5 and Theorem 6. For $\alpha > \hat{\alpha}$ there are indeed leaf-branches and bisection trees with a maximum weight leaf that is heavier than the upper bound provided by Lemma 5 and Theorem 6. If we choose $\alpha = 1/4$ and $N = 4$, for example, there is a leaf-branch whose maximum leaf weight is $(3/7)\mathsf{w}(p) > (3/4)^3\mathsf{w}(p)$. Furthermore, it is possible to construct bisection trees with $\max_{1 \leq i \leq N} \mathsf{w}(p_i) = \mathsf{w}(p)(1 - \alpha)/(2 - \alpha)$ for $N = 4$, $\alpha \leq (3 - \sqrt{5})/2$. Figure 7 illustrates these exceptional cases for $N = 4$.

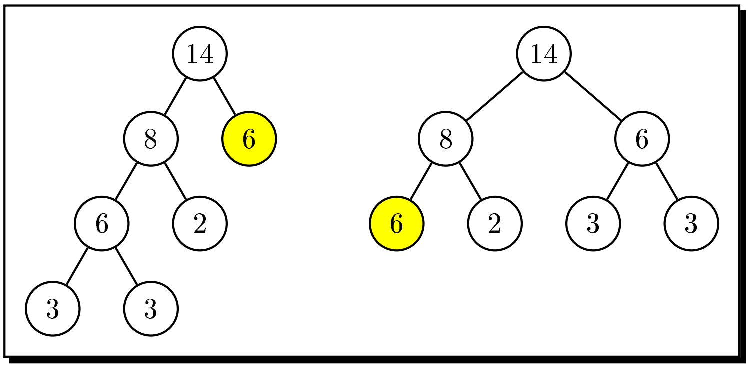

Figure 7: Worst-case leaf-branch and bisection tree for $N = 4$, $\alpha = \frac{1}{4}$

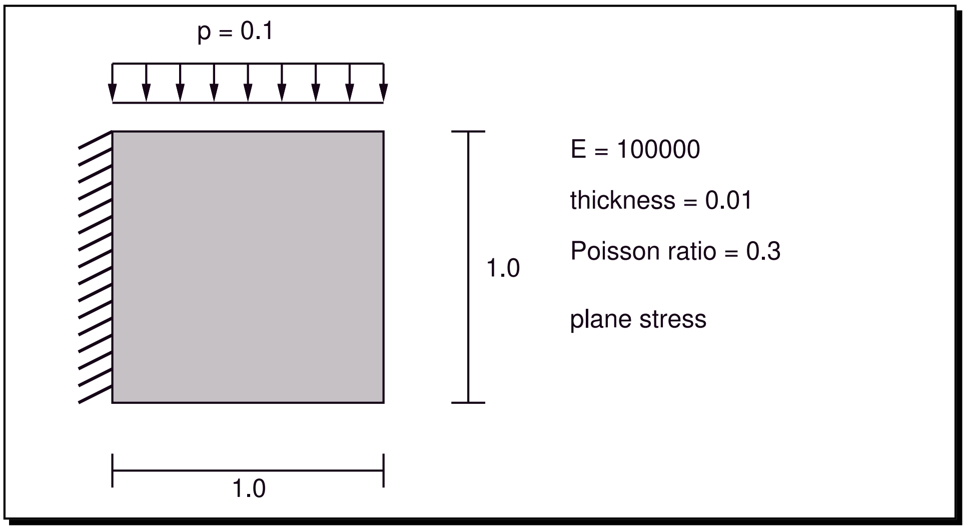

Figure 8: Static system of a short cantilever

3 Application of Algorithm HF to Distributed Finite Element Simulations

In this section, we present the application of Algorithm HF for load balancing in the field of numerical simulations with the finite element (FE) method [Bra97, Bur87]. The FE method is used in statics analysis, for example, to calculate the response of objects under certain loading and boundary conditions.

In [Hüt96, HS94], an adaptive FE method based on the principle of recursive substructuring has been developed. It is an iterative procedure where in several runs of computation the result is improved automatically until a predefined accuracy is reached. The costs for achieving this accuracy are much lower than with a non-adaptive procedure.

3.1 Recursive Substructuring

Starting an analysis with the FE method, an object is described by defining its shape and its structural properties. Then, the boundary and loading conditions have to be imposed on the object. A system of partial differential equations describes the relation between external loads and internal forces.

As an example from structural engineering, we consider a short cantilever under plane stress conditions, a problem from the domain of plane elasticity. The quadratic panel is uniformly loaded on its upper side. The left side of the cantilever is fixed as shown in Figure 8.

The physical properties for the material of the cantilever are given by the Youngs modulus E and the Poisson ratio ν. The differential equations (6) and (7) describe the response of the object under the external loads:

$$\frac{E}{1-\nu^2} \cdot \frac{\partial^2 u}{\partial x^2} + \frac{E}{2(1-\nu)} \cdot \frac{\partial^2 v}{\partial x \partial y} + \frac{E}{2(1+\nu)} \cdot \frac{\partial^2 u}{\partial y^2} = -f \tag{6}$$

$$\frac{E}{1-\nu^2} \cdot \frac{\partial^2 v}{\partial y^2} + \frac{E}{2(1-\nu)} \cdot \frac{\partial^2 u}{\partial x \partial y} + \frac{E}{2(1+\nu)} \cdot \frac{\partial^2 v}{\partial x^2} = -g, \tag{7}$$

where u and v are the unknown displacements and f and g the external forces in x- and y-direction, respectively.

We substructure the physical domain of the cantilever recursively (Figure 9, left). With the method of [Hüt96, HS94], a tree data structure is built reflecting the hierarchy of the substructured domain (see Figure 9, right). In each node, points on the separator line represent unknown values of displacement, and points on the border carry variable boundary conditions imposed by the parent node. Each tree node contains a system of linear equations whose *stiffness matrix* S determines the unknown displacement values dependent on the external forces:

$$S \begin{pmatrix} \hat{u} \\ \hat{v} \end{pmatrix} = \begin{pmatrix} \hat{f} \\ \hat{g} \end{pmatrix}.$$

In the leaves, the system of equations is constructed by a standard FE discretization. Roughly speaking, the equations are obtained by an approximation of the functions u, v, f, and g by linear combinations ($\hat{u}$, $\hat{v}$, $\hat{f}$, and $\hat{g}$, respectively) of partially bilinear basis functions with limited support within the discretizing mesh, and some additional algebraic and analytical transformations. The system of equations of an internal tree node is assembled out of the equations of its children, as described in [Hüt96].

Now, the task is to solve all those systems of linear equations. We use an iterative solver which traverses the tree several times, promoting displacements in top-down direction and reaction forces in bottom-up direction. In each node, the amount of work to be done stays the same during the iterations. But since the adaptive structure of the tree is not known *a priori*, it is essential to have a good load balancing strategy before the parallel execution of the solving phase.

3.2 Application of Algorithm HF

We assign a load value $\ell(p)$ to each tree node p, given by

$$\ell(p) = C_b\, n_b(p) + C_s\, n_s(p)$$

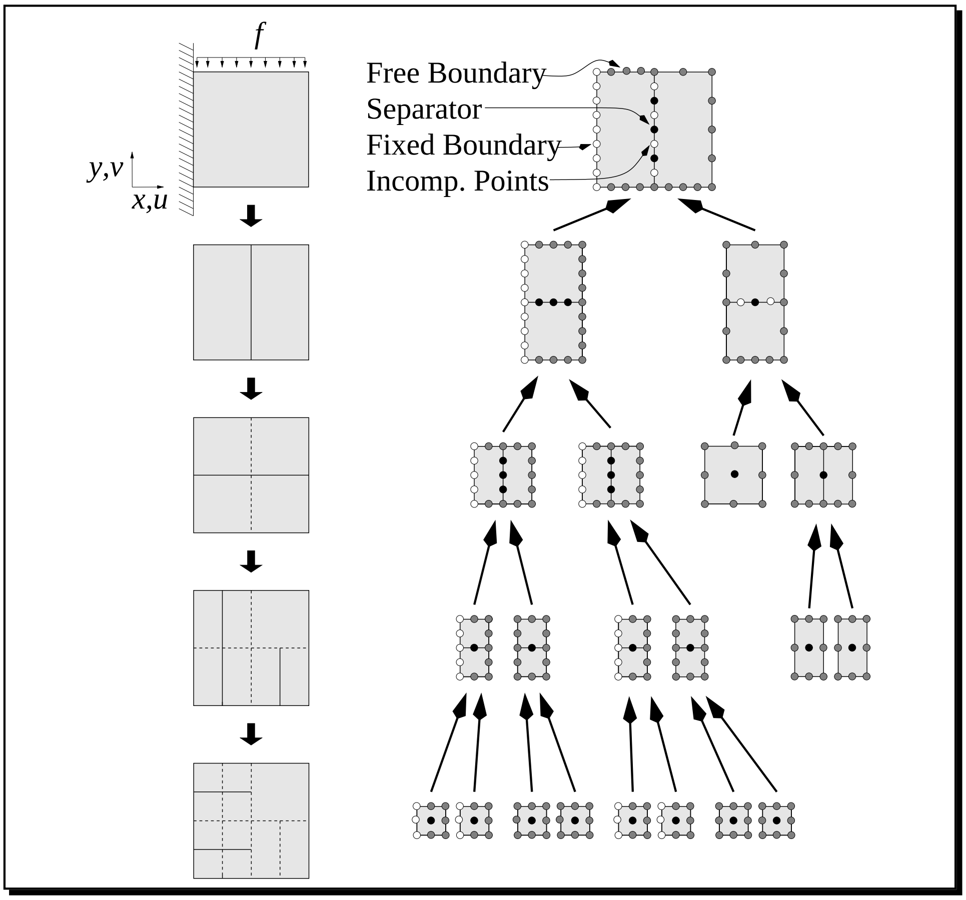

Figure 9: A coarse discretizing mesh and the resulting binary tree data structure for the short cantilever

with $n_b(p)$ points on the border without boundary conditions (grey points in Figure 9), and $n_s(p)$ points on the separator line of node p (black points belonging to the borders of both children in Figure 9). The load value $\ell(p)$ models the computing time of node p, where the constants C_s and C_b are independent of node p and $C_s \approx 6\,C_b$. Points with fixed boundary values as well as incompatible points on the separator (white points) do not contribute to the load value $\ell(p)$.

We can interpret the FE tree as an approximate (potential) bisection tree by accumulating the load of all nodes in the subtree rooted in node p to get the weight value $\mathsf{w}(p)$:

$$\mathsf{w}(p) = \begin{cases} \ell(p) & \text{if } p \text{ is leaf} \\ \ell(p) + \mathsf{w}(c_1) + \mathsf{w}(c_2) & \text{if } p \text{ is internal node or root,} \end{cases}$$

where c_1 and c_2 are the children of p.

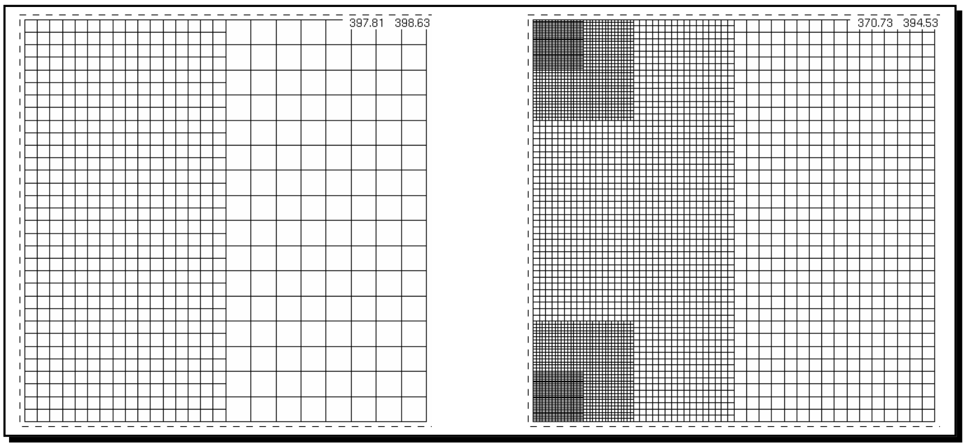

Figure 10: The discretizing meshes for the domain of the short cantilever

The weight values are collected during the tree construction phase by simply counting and accumulating the number of points on the separator of each tree node.

If we want to apply Algorithm HF to this tree of weight values, we must specify which bisection steps the algorithm can perform. Our first approach is to define a bisection step as the removal of the root node p of a subtree. This yields two subtrees rooted at the children c_1 and c_2 of p, and p is ignored for the remainder of the load balancing phase.

Such bisection steps do not exactly match Definition 1, because the weight of node p exceeds the weight sum $\mathsf{w}(c_1) + \mathsf{w}(c_2)$ of the children by $\ell(p)$. However, $\ell(p)$ (work load of the one-dimensional separator) is negligible compared to $\mathsf{w}(p)$ (work load of the two-dimensional domain) in our application if the FE tree is large enough. Hence, the results of Theorem 2 and Corollary 3 are well approximated.

Algorithm HF chops N subtrees off the FE tree, each of which can be traversed in parallel by the iterative solver. These N subtrees contain the main part of the solving work and may be distributed over the available N processors. The upper $N-1$ tree nodes cannot exploit the whole number of processors, anyway. Therefore, such a distribution does not sacrifice parallel potential in the upper tree levels.

3.3 Runtime Examples

In [EP98], a parallel implementation of the recursive substructuring technique is described using the dataflow language FASAN as coordination and automatic parallelization platform. For this paper, however, we used a hand-coded parallel version based on PVM [BDG+94] in order to minimize com-

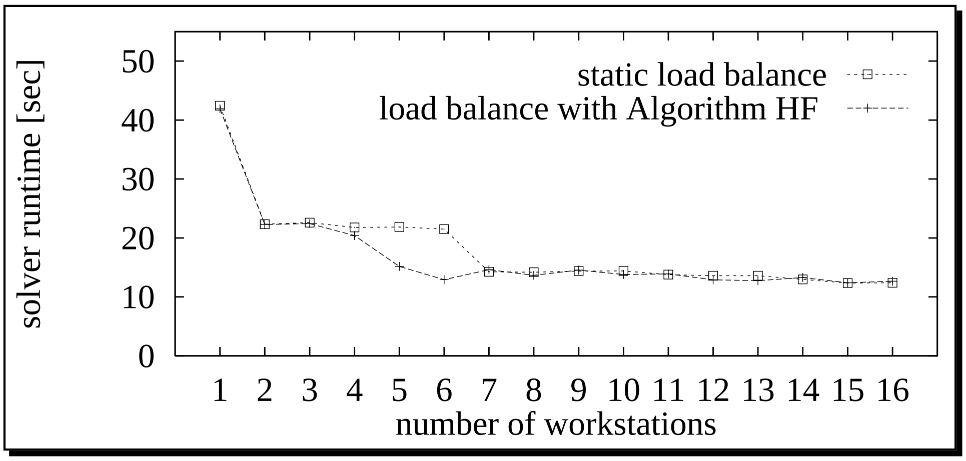

Figure 11: Runtime results for 1,279 tree nodes

munication overhead. The number of solver iterations (tree traversals) was fixed to 100. The experiments were run on a cluster of workstations of type HP 9000/720.

Figure 11 shows the runtime results of the numerical simulation of the short cantilever under uniform load described above. We have chosen quite a small tree of 1,279 element nodes and maximal depth 11 (see the left discretizing mesh in Figure 10, representing the leaves of the FE tree). Since adaptivity was limited to two additional tree levels only, the node weights resulted in $\alpha = 0.18571$.

For two workstations, the partitioning with Algorithm HF is identical to static partitioning (just chopping off the root node). Further speedup from 4 to 6 processors with Algorithm HF occurs earlier than with static partitioning (from 6 to 7 processors). In this comparatively small problem, it is mainly the critical path of the FE tree that determines the lower bound for the tree traversal time and inhibits further acceleration with more than five processors.

The effect of Algorithm HF is more important in larger simulations, where adaptivity for high numerical accuracy is distinct and where it is essential to split the biggest subtrees. The runtime results of a computation with a deeper FE tree of maximal depth 17 can be seen in Figure 12. It contains 11,263 nodes (Figure 10, right), and α has quite a bad value of 0.10615. Here, the runtime improvement with Algorithm HF is more significant, since the load value $\ell(p)$ of the root is quite small in relation to the weight sum $\mathsf{w}(p)$ of the whole tree. We observe that the distributed iterative solver is up to 70% faster with application of Algorithm HF compared to static partitioning if at least four processors are used. Even on 16 processors, static partitioning does not split the subtree which is responsible for the longest computing time.

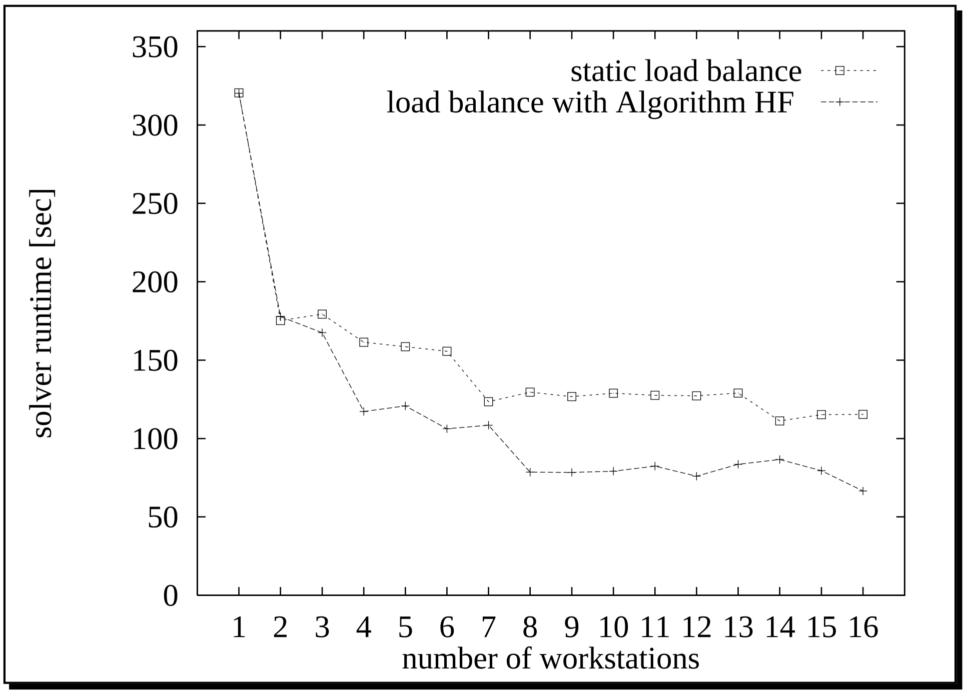

Figure 12: Runtime results for 11,263 tree nodes

Nevertheless, we clearly recognize, again, the influence of the critical tree path: Applying Algorithm HF, we reach nearly minimal runtime with eight processors already.

3.4 Further Improvements

For arbitrary adaptive FE simulations, we cannot give a limit for the bisection factor α. If α gets too small, we still have two possibilities to prevent too bad a partitioning:

- During the tree construction phase, we can choose between horizontal and vertical bisection of the subdomain of each node, whichever leads to the greater local value of α.
- Moreover, we might set N as a multiple of the number of available processors, so there is still a chance to compensate a small α value by assigning multiple partitions to one processor.

As the factor α is known immediately after the tree construction or refinement phase, the user can be warned before starting the iterative solver if the partitioning is not satisfactory.

To avoid the small-α-problem completely, another strategy using Algorithm HF allows the removal of a single edge of a tree as a bisection step. This strategy partitions the *entire* given FE tree into N subtrees of approximately equal size. Section 4 shows that FE trees satisfying the conditions

$$\begin{aligned} \ell(p) &\leq \ell(c_1) + \ell(c_2) \\ \ell(p) &\geq \ell(c_i) \qquad (i = 1, 2) \end{aligned}$$

have good bisectors. This application of Algorithm HF also takes into account that the main memory resources of the processors become the limiting factor if very high accuracy of the simulation is required. In this case, finding a partitioning of the entire tree (not only a set of equal-sized subtrees ignoring their ancestors in the tree) is necessary.

4 Weighted Trees with Good Bisectors

Let $\mathcal{T}$ be the set of all rooted binary trees with node weights $\ell(v)$ satisfying:

(1) $\ell(p) \leq \ell(c_1) + \ell(c_2)$ for nodes p with two children c_1 and c_2
(2) $\ell(p) \geq \ell(c)$ if c is a child of p

The weight of a tree $T = (V, E)$ in $\mathcal{T}$ is defined as $\mathsf{w}(T) = \sum_{v \in V} \ell(v)$.

This class $\mathcal{T}$ of binary trees models the load of applications in hierarchical finite element simulations, as discussed in Section 3. Recall that in these applications the domain of the computation is repeatedly subdivided into smaller subdomains. The structure of the domains and subdomains yields a binary tree in which every node has either two children or is a leaf. The resource demands (CPU and main memory) of the nodes in this FE tree are such that the resource demand at a node is at most as large as the sum of the resource demands of its two children. In order to parallelize the computation, it is necessary to distribute the FE tree among a number of processors in a balanced way.

Note that Conditions (1) and (2) ensure that the two subtrees obtained from a tree in $\mathcal{T}$ by removing a single edge are also members of $\mathcal{T}$.

The following theorem shows that trees from the class $\mathcal{T}$ can be $\frac{1}{4}$-bisected by removal of a single edge unless the weight of the tree is concentrated in the root.

Theorem 7 *Let $T = (V, E)$ be a tree in $\mathcal{T}$, and let r be its root. If $\ell(r) \leq \frac{3}{4}\mathsf{w}(T)$, then there is an edge $e \in E$ such that the removal of e partitions T into subtrees T_1 and T_2 with $\mathsf{w}(T_1), \mathsf{w}(T_2) \in [\frac{1}{4}\mathsf{w}(T); \frac{3}{4}\mathsf{w}(T)]$.*

Proof: We give a simple method to find the required edge. Pick an arbitrary node v of T as a start node. While $T \setminus \{v\}$ contains a subtree T' with

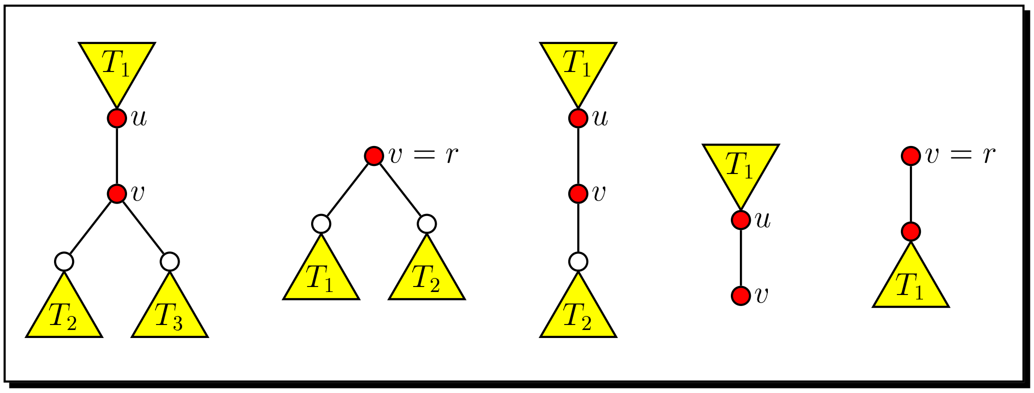

Figure 13: The 5 cases for v in proof of Theorem 7

$\mathsf{w}(T') > \frac{3}{4}\mathsf{w}(T)$, replace v by the node adjacent to v which is contained in T'. This process always terminates after less than $|V|$ iterations at a node v such that all subtrees T' in $T \setminus \{v\}$ satisfy $\mathsf{w}(T') \leq \frac{3}{4}\mathsf{w}(T)$. We claim that at least one of these subtrees also satisfies $\mathsf{w}(T') \geq \frac{1}{4}\mathsf{w}(T)$, and thus the edge connecting v and T' can be picked as the required separator edge. In order to prove the claim we distinguish several cases regarding the position of v in T (see Figure 13). For every case the assumption that all subtrees of $T \setminus \{v\}$ have weight $< \frac{1}{4}\mathsf{w}(T)$ will lead to a contradiction.

Case 1: v has degree 3. Let u be the parent of v. Let T_1 be the subtree of $T \setminus \{v\}$ that contains u, and let T_2 and T_3 be the other two subtrees. Assume that all three subtrees have weight $< \frac{1}{4}\mathsf{w}(T)$. Consequently, v must have weight $> \frac{1}{4}\mathsf{w}(T)$ because $\mathsf{w}(T) = \mathsf{w}(T_1) + \mathsf{w}(T_2) + \mathsf{w}(T_3) + \ell(v)$. But then u must also have weight $> \frac{1}{4}\mathsf{w}(T)$ because it is the parent of v, and this implies $\mathsf{w}(T_1) > \frac{1}{4}\mathsf{w}(T)$. A contradiction.

Case 2: v has degree 2 and is the root of T. Let T_1 and T_2 be the two subtrees of $T \setminus \{v\}$, and let u_1 and u_2 be the corresponding children of v. If both subtrees have weight $< \frac{1}{4}\mathsf{w}(T)$, it follows also that $\ell(u_1) < \frac{1}{4}\mathsf{w}(T)$ and $\ell(u_2) < \frac{1}{4}\mathsf{w}(T)$, which implies $\ell(v) \leq \ell(u_1) + \ell(u_2) < \frac{1}{2}\mathsf{w}(T)$. On the other hand, $\mathsf{w}(T) = \mathsf{w}(T_1) + \mathsf{w}(T_2) + \ell(v)$ implies $\ell(v) > \frac{1}{2}\mathsf{w}(T)$. A contradiction.

Case 3: v has degree 2 and is not the root of T. Let u be the parent of v, and let c be the child of v. Let T_1 be the subtree in $T \setminus \{v\}$ that contains u, and let T_2 be the other subtree. If both subtrees have weight $< \frac{1}{4}\mathsf{w}(T)$, $\mathsf{w}(T) = \mathsf{w}(T_1) + \mathsf{w}(T_2) + \ell(v)$ implies $\ell(v) > \frac{1}{2}\mathsf{w}(T)$. But then $\ell(u) \geq \ell(v) > \frac{1}{2}\mathsf{w}(T)$. A contradiction.

Case 4: v has degree 1 and is a leaf of T. Let T_1 be the tree $T \setminus \{v\}$, and let u be the parent of v. Assume that $\mathsf{w}(T_1) < \frac{1}{4}\mathsf{w}(T)$. Then $\mathsf{w}(T) = \mathsf{w}(T_1) + \ell(v)$ implies $\ell(v) > \frac{3}{4}\mathsf{w}(T)$. But then $\mathsf{w}(T_1) \geq \ell(u) \geq \ell(v) > \frac{3}{4}\mathsf{w}(T)$. A contradiction.

Case 5: v has degree 1 and is the root of T. Let T_1 be the tree $T \setminus \{v\}$. As $\ell(v) = \ell(r) \leq \frac{3}{4}\mathsf{w}(T)$, $\mathsf{w}(T) = \ell(v) + \mathsf{w}(T_1)$ implies $\mathsf{w}(T_1) \geq \frac{1}{4}\mathsf{w}(T)$. □

According to Theorem 2 a problem p from a class of problems that has $\frac{1}{4}$-bisectors can always be subdivided into N subproblems $p_1, \ldots, p_N$ such that $\max_{1 \leq i \leq N} \mathsf{w}(p_i) \leq \frac{\mathsf{w}(p)}{N} \cdot \frac{9}{4}$. The following corollary gives a condition on trees in $\mathcal{T}$ that ensures that they can be subdivided into N subproblems using $\frac{1}{4}$-bisectors.

Corollary 8 *Let $T = (V, E)$ be a tree in $\mathcal{T}$, and let r be its root. Let N be a positive integer. If $\mathsf{w}(T) \geq \frac{4}{3}(N-1)\ell(r)$, Algorithm HF partitions T into N subtrees by cutting exactly $N-1$ edges such that the maximum weight of the resulting subtrees is at most $\frac{9}{4} \cdot \frac{\mathsf{w}(T)}{N}$.*

Proof: After k bisection steps according to Theorem 7 there are $k+1$ subtrees. There is at least one subtree T' with weight at least $\frac{\mathsf{w}(T)}{k+1}$. Let r' be the root of T'. If $k+1 < N$, we have $\mathsf{w}(T') \geq \frac{\mathsf{w}(T)}{k+1} \geq \frac{\mathsf{w}(T)}{N-1} \geq \frac{4}{3}\ell(r) \geq \frac{4}{3}\ell(r')$, and another bisection step is possible.

The upper bound for the maximum weight of any subtree follows directly from Theorem 2 for $\alpha = 1/4$ (see also Table 1). □

Note that an optimal min-max k-partition of a weighted tree (i.e., a partition with minimum weight of the heaviest component after removing k edges) can be computed in linear time [BP95, Fre91]. These algorithms are preferable to our approach using Algorithm HF in the case of trees that are to be subdivided by removing a minimum number of edges. Since the heaviest subtree in the optimal solution does obviously not have a greater weight than the maximum generated by Algorithm HF, the bound from Corollary 8 still applies and provides a non-trivial worst-case performance guarantee for these optimal algorithms as well.

5 Conclusion and Future Work

The existence of α-bisectors for a class of problems was shown to allow good load balancing for a surprisingly large range of values of α. The maximum load achieved by Algorithm HF is at most a factor of $\lfloor 1/\alpha \rfloor \cdot (1-\alpha)^{\lfloor 1/\alpha \rfloor - 2}$ larger than the theoretical optimum (uniform distribution). This bound was proved to be tight. It gives a performance guarantee of factor 2 for $\alpha \geq 1/3$ and factor 3 for $\alpha \geq 1 - 1/\sqrt[4]{2} \approx 0.159$.

Load balancing for distributed hierarchical finite element simulations was discussed, and two strategies for applying Algorithm HF were presented. The first strategy tries to make the best use of the available parallelism, but requires that the nodes of the FE tree representing the load of the computation have good separators. The second strategy tries to partition the entire FE tree into subtrees with approximately equal load. For this purpose, it was proved that a certain class of weighted trees, which include FE trees, has 1/4-bisectors. Here, the trees are bisected by removing a single edge. Partitioning the trees by removing a minimum number of edges ensures that only a minimum number of communication channels of the application must be realized by network connections. Our results provide performance guarantees for balancing the load of applications with good bisectors in general and of distributed hierarchical finite element simulations in particular. For the latter application, we showed that the maximum resulting load is at most a factor of 9/4 larger than in a perfectly uniform distribution.

We implemented the load balancing methods proposed in this paper and integrated them into the existing finite element simulations software ARESO. We obtained considerable improvements already for small problems, as compared to the static (compile-time) processor allocation currently in use. Since ARESO is primarily a solver of hierarchical equation systems, it is not limited to statics simulations. Other physical problems described by elliptic partial differential equations are tractable as well. Currently, we add a component for CFD (computational fluid dynamics) simulations taken from [Fun97].

Among the applications of project B3 of SFB 342, we can find other hierarchical numerical distributed algorithms that could be accelerated with Algorithm HF: Domain decomposition in the process of chip layout with the placement tool GORDIAN [RR93, Reg97] may result in an unbalanced binary tree. The subsequent layout process could obviously be improved by load distribution with Algorithm HF.

Another application is the multi-dimensional adaptive numerical quadrature `aqho` [Bon93, Bon95]. It is based on an adaptively growing binary tree. Algorithm HF may be applied in much the same way as in the ARESO application, because each traversal visits all tree nodes and adds a new (and potentially incomplete) layer of leaves.

If a large number of processors is available it is highly desirable to accomplish the problem decomposition in parallel. It is possible to parallelize Algorithm HF while maintaining the worst-case bound on the maximum load. But this parallel algorithm seems to require a fairly high amount of communication to route subproblems to free processors. Therefore, we also investigate a different parallel load balancing strategy that avoids the above mentioned routing problem. The maximum load generated by this algorithm in the worst-case is higher than the corresponding bound for Algorithm HF only by a small factor.

Simulation results indicate that Algorithm HF performs very well on average even for very small values of α. Assume that the actual bisection parameter is drawn uniformly at random from the interval $[\alpha, \frac{1}{2}]$, $\alpha \ll \frac{1}{2}$, and that all bisections are independent and identically distributed. Then the observed ratio between the maximum load generated by Algorithm HF and the ideal load is much smaller than the worst-case bound. We think it necessary to confirm these experimental observations by a detailed average-case analysis of Algorithm HF under the above assumptions.

References

[BDG$^+$94] A. Beguelin, J. Dongarra, A. Geist, J. Weicheng, R. Manchek, and V. Sunderam. *PVM : Parallel Virtual Machine : A Users' Guide and Tutorial for Networked Parallel Computing.* The MIT Press, Cambrigde (MA) et. al., 1994.

[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. Vieweg Verlag, Braunschweig, 1993.

[Bon95] T. Bonk. *Ein rekursiver Algorithmus zur adaptiven numerischen Quadratur mehrdimensionaler Funktionen.* PhD thesis, Institut für Informatik, Technische Universität München, 1995.

[BP95] Ronald I. Becker and Yehoshua Perl. The shifting algorithm technique for the partitioning of trees. *Discrete Appl. Math.*, 62:15–34, 1995.

[Bra97] Dietrich Braess. *Finite Elemente.* Springer, Berlin, 1997. 2. überarbeitete Auflage.

[Bur87] D.S. Burnett. *Finite Element Analysis.* Addison-Wesley Publishing Company, 1987.

[EP98] Ralf Ebner and Alexander Pfaffinger. Higher Level Programming and Efficient Automatic Parallelization: A Functional Data Flow Approach with FASAN. In *Proceedings of the ParCo97 Parallel Computing Conference, 16–19 September 1997, Bonn Bad Godesberg.* Elsevier Science Publishers, Amsterdam, 1998. To appear.

[Fre91] Greg N. Frederickson. Optimal Algorithms for Tree Partitioning. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms SODA '91*, pages 168–177, New York, 1991. ACM Press.

[Fun97] Kilian Funk. Anwendung der algebraischen Mehrgittermethode auf konvektionsdominierte Strömungen. Diplomarbeit, Technische Universität München, 1997.

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

[Hüt96] Reiner Hüttl. *Ein iteratives Lösungsverfahren bei der Finite-Element-Methode unter Verwendung von rekursiver Substrukturierung und hierarchischen Basen.* PhD thesis, Institut für Informatik, Technische Universität München, 1996.

[KGGK94] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* The Benjamin/Cummings Pubishing Company, Inc., Redwood City, CA, 1994.

[KGV94] Vipin Kumar, Ananth Y. Grama, and Nageshwara Rao Vempaty. Scalable Load Balancing Techniques for Parallel Computers. *J. Parallel Distrib. Comput.*, 22(1):60–79, 1994.

[KV87] Vipin Kumar and Nageshwara Rao Vempaty. Parallel depth-first search, Part II: Analysis. *International Journal of Parallel Programming*, 16(6):501–519, 1987.

[Reg97] H. Regler. *Anwenden von Algebraischen Mehrgittermethoden auf das Plazierproblem im Chipentwurf und auf die numerische Simulation von Strömungen.* PhD thesis, Technische Universität München, 1997.

[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. Also available as technical report SFB 342/11/93 A, TU München.

[SHK95] Behrooz A. Shirazi, Ali R. Hurson, and Krishna M. Kavi, editors. *Scheduling and Load Balancing in Parallel and Distributed Systems.* IEEE Computer Society Press, Los Alamitos, CA, 1995.

[SS97] Thomas Schnekenburger and Georg Stellner, editors. *Dynamic Load Distribution for Parallel Applications.* TEUBNER-TEXTE zur Informatik. Teubner Verlag, Stuttgart, 1997.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

Liste aller erschienenen Berichte von 1990-1994
auf besondere Anforderung

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids
342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law
342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space
342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes
342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems
342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter
342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology
342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies
342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness
342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware / Software Co-Design
342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System
342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm
342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming
342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks
342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

Reihe A

342/16/95 A	Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik
342/17/95 A	Georg Stellner: Using CoCheck on a Network of Workstations
342/18/95 A	Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications
342/19/95 A	Thomas Schnekenburger: Integration of Load Distribution into ParMod-C
342/20/95 A	Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication
342/21/95 A	Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control
342/22/95 A	Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes
342/23/95 A	Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation
342/24/95 A	M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems
342/01/96 A	Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries
342/02/96 A	Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs
342/03/96 A	Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes
342/04/96 A	Thomas Huckle: Efficient Computation of Sparse Approximate Inverses
342/05/96 A	Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-line Monitoring Interface Specification
342/06/96 A	Ekkart Kindler: A Compositional Partial Order Semantics for Petri Net Components
342/07/96 A	Richard Mayr: Some Results on Basic Parallel Processes
342/08/96 A	Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht
342/09/96 A	P.P. Spies, C. Eckert, M. Lange, D. Marek, R. Radermacher, F. Weimer, H.-M. Windisch: Sprachkonzepte zur Konstruktion verteilter Systeme
342/10/96 A	Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFSLib – A File System for Parallel Programming Environments
342/11/96 A	Manfred Broy, Gheorghe Ştefănescu: The Algebra of Stream Processing Functions

Reihe A

342/12/96 A	Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete
342/13/96 A	Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks
342/14/96 A	Giannis Bozas, Michael Jaedicke, Andreas Listl, Bernhard Mitschang, Angelika Reiser, Stephan Zimmermann: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project
342/15/96 A	Richard Mayr: A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ -Calculus
342/16/96 A	Ursula Hinkel, Katharina Spies: Anleitung zur Spezifikation von mobilen, dynamischen Focus-Netzen
342/17/96 A	Richard Mayr: Model Checking PA-Processes
342/18/96 A	Michaela Huhn, Peter Niebert, Frank Wallner: Put your Model Checker on Diet: Verification on Local States
342/01/97 A	Tobias Müller, Stefan Lamberts, Ursula Maier, Georg Stellner: Evaluierung der Leistungsfähigkeit eines ATM-Netzes mit parallelen Programmierbibliotheken
342/02/97 A	Hans-Joachim Bungartz and Thomas Dornseifer: Sparse Grids: Recent Developments for Elliptic Partial Differential Equations
342/03/97 A	Bernhard Mitschang: Technologie für Parallele Datenbanken - Bericht zum Workshop
342/04/97 A	nicht erschienen
342/05/97 A	Hans-Joachim Bungartz, Ralf Ebner, Stefan Schulte: Hierarchische Basen zur effizienten Kopplung substrukturierter Probleme der Strukturmechanik
342/06/97 A	Hans-Joachim Bungartz, Anton Frank, Florian Meier, Tilman Neunhoeffer, Stefan Schulte: Fluid Structure Interaction: 3D Numerical Simulation and Visualization of a Micropump
342/07/97 A	Javier Esparza, Stephan Melzer: Model Checking LTL using Constraint Programming
342/08/97 A	Niels Reimer: Untersuchung von Strategien für verteiltes Last- und Ressourcenmanagement
342/09/97 A	Markus Pizka: Design and Implementation of the GNU INSEL-Compiler gic
342/10/97 A	Manfred Broy, Franz Regensburger, Bernhard Schätz, Katharina Spies: The Steamboiler Specification - A Case Study in Focus
342/11/97 A	Christine Röckl: How to Make Substitution Preserve Strong Bisimilarity
342/12/97 A	Christian B. Czech: Architektur und Konzept des Dycos-Kerns
342/13/97 A	Jan Philipps, Alexander Schmidt: Traffic Flow by Data Flow

Reihe A

342/14/97 A	Norbert Fröhlich, Rolf Schlagenhaft, Josef Fleischmann: Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level
342/15/97 A	Frank Weimer: DaViT: Ein System zur interaktiven Ausführung und zur Visualisierung von INSEL-Programmen
342/16/97 A	Niels Reimer, Jürgen Rudolph, Katharina Spies: Von FOCUS nach INSEL - Eine Aufzugssteuerung
342/17/97 A	Radu Grosu, Ketil Stølen, Manfred Broy: A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing
342/18/97 A	Christian Röder, Georg Stellner: Design of Load Management for Parallel Applications in Networks of Heterogenous Workstations
342/19/97 A	Frank Wallner: Model Checking LTL Using Net Unfoldings
342/20/97 A	Andreas Wolf, Andreas Kmoch: Einsatz eines automatischen Theorembeweisers in einer taktikgesteuerten Beweisumgebung zur Lösung eines Beispiels aus der Hardware-Verifikation –Fallstudie–
342/21/97 A	Andreas Wolf, Marc Fuchs: Cooperative Parallel Automated Theorem Proving
342/22/97 A	T. Ludwig, R. Wismüller, V. Sunderam, A. Bode: OMIS - On-line Monitoring Interface Specification (Version 2.0)
342/23/97 A	Stephan Merkel: Verification of Fault Tolerant Algorithms Using PEP
342/24/97 A	Manfred Broy, Max Breitling, Bernhard Schätz, Katharina Spies: Summary of Case Studies in Focus - Part II
342/25/97 A	Michael Jaedicke, Bernhard Mitschang: A Framework for Parallel Processing of Aggregat and Scalar Functions in Object-Relational DBMS
342/26/97 A	Marc Fuchs: Similarity-Based Lemma Generation with Lemma-Delaying Tableau Enumeration
342/27/97 A	Max Breitling: Formalizing and Verifying TimeWarp with FOCUS
342/28/97 A	Peter Jakobi, Andreas Wolf: DBFW: A Simple DataBase FrameWork for the Evaluation and Maintenance of Automated Theorem Prover Data (incl. Documentation)
342/29/97 A	Radu Grosu, Ketil Stølen: Compositional Specification of Mobile Systems
342/01/98 A	A. Bode, A. Ganz, C. Gold, S. Petri, N. Reimer, B. Schiemann, T. Schnekenburger (Herausgeber): “Anwendungsbezogene Lastverteilung”, ALV’98
342/02/98 A	Ursula Hinkel: Home Shopping - Die Spezifikation einer Kommunikationsanwendung in FOCUS
342/03/98 A	Katharina Spies: Eine Methode zur formalen Modellierung von Betriebssystemkonzepten

Reihe A

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B	Wolfgang Reisig: Petri Nets and Algebraic Specifications
342/2/90 B	Jörg Desel: On Abstraction of Nets
342/3/90 B	Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems
342/4/90 B	Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme
342/1/91 B	Barbara Paech1: Concurrency as a Modality
342/2/91 B	Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier-Toolbox — Anwenderbeschreibung
342/3/91 B	Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen
342/4/91 B	Werner Pohlmann: A Limitation of Distributed Simulation Methods
342/5/91 B	Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis
342/6/91 B	Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme
342/7/91 B	W. Reisig: Concurrent Temporal Logic
342/1/92 B	Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support
342/2/92 B	Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen
342/1/93 B	Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung
342/2/93 B	Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick
342/1/94 B	Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS