Load Balancing for Problems with Good Bisectors, and Applications in Finite Element Simulations

Stefan Bischof, Ralf Ebner, and 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/

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 (i.e. workload) 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 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. The maximum resulting load is at most a factor of $9/4$ larger than in a perfectly uniform distribution 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.

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

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.

Input: problem p, positive integer N
begin
  $P \leftarrow \{p\}$;
  while $|P| < N$ **do**
    begin
    $q \leftarrow$ a problem in P with maximum weight;
    bisect q into q_1 and q_2;
    $P \leftarrow (P \cup \{q_1, q_2\}) \setminus \{q\}$;
    end;
  output P;
end.

Fig. 1. Algorithm HF (Heaviest Problem First)

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 $\mathsf{w}(p)/N$. All proofs are omitted in this paper due to space constraints but can be found in [BEE98].

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 $p_1, \ldots, p_N$ such that*

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

It is not difficult to see that the bound given in Theorem 2 is asymptotically tight. This means that for sufficiently large N there exists a sequence of bisections such that the maximum load generated by Algorithm HF is arbitrarily close to this bound. For this purpose, the original problem is first split into a number of subproblems $p_1, \ldots, p_n$ of equal weight. Each of $p_1, \ldots, p_{n-1}$ is then split into $\lfloor 1/\alpha \rfloor$ subproblems using a sequence of $\lfloor 1/\alpha \rfloor - 1$ exact α-bisections, whereas p_n is split similarly but without doing the final bisection step.

The construction outlined above suggests that the bound from Theorem 2 might not be tight for small values of N, however. Indeed, we can prove that for all $\alpha \le 1/5$ and for all $N \le 1/\alpha$, Algorithm HF partitions a problem from a class of problems with α-bisectors into N subproblems such that the weight of the heaviest subproblem is at most $\mathsf{w}(p)(1-\alpha)^{N-1}$. Hence, the ratio between the maximum weight and the ideal average weight is bounded by $N(1-\alpha)^{N-1}$, which is smaller than $\lfloor 1/\alpha \rfloor (1-\alpha)^{\lfloor 1/\alpha \rfloor - 2}$ for the range where this bound applies.

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 distributed numerical simulations with the finite element (FE) method [Bra97]. The FE method is used in statics analysis, for example, to calculate the response of objects under external load. In this paper, we consider a short cantilever under plane stress conditions, a problem from plane elasticity. The quadratic cantilever is uniformly loaded on its upper side. Its left side is fixed as shown in Figure 2, top left. The definition of the object's shape, its structural properties, and the boundary conditions lead to a system of elliptic partial differential equations.

We substructure the physical domain of the cantilever recursively. With the method of [Hüt96], a tree data structure is built reflecting the hierarchy of the substructured domain (see Figure 2). This data structure contains a system of linear equations $Su = f$ for each tree node. Its *stiffness matrix* S determines the unknown displacement values in the vector u at the black points shown in Figure 2 (right), dependent on the external and reaction forces f. In the leaves, the system of equations is constructed by standard

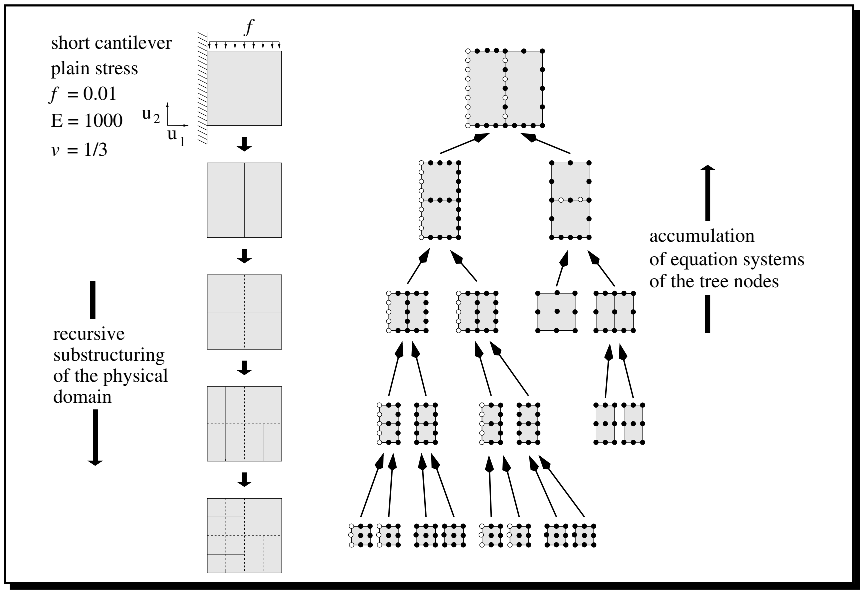

Fig. 2. The example application

FE discretization and the so-called virtual displacement method. In an internal node, the system of linear equations is assembled out of the equation systems of its children.

For the solution of all those equation systems, we use an iterative solver which traverses the tree several times, promoting displacements in top-down direction and reaction forces in bottom-up direction. Since the adaptive structure of the tree is not known *a priori*, a good load balancing strategy is essential before the parallel execution of the solving iterations.

We assign a load value $\ell(p) = C_b n_b(p) + C_s n_s(p)$ to each tree node p, with $n_b(p)$ black points on the border without boundary conditions, and $n_s(p)$ black points (not ghost points) belonging to the separator line of node p. 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 $C_s \approx 6C_b$. These values can be easily collected during the tree construction phase.

In order to distribute the work load among the processors, we need to know the computing time of whole subtrees of the FE tree. So for each node p, we accumulate the load in the subtree with root p to get the weight value $\mathsf{w}(p)$:

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

where c_1 and c_2 are the children of p.

For a bisection step according to Algorithm HF, we remove the root node p from the tree of weight values. This yields two subtrees rooted at the children c_1 and c_2 of p, and the weight of the root node p is ignored for the remainder of the load balancing phase. The results of Theorem 2 are well approximated because $\ell(p)$ (work load for the one-dimensional separator) is negligible compared to $\mathsf{w}(p)$ (work load for the two-dimensional domain) in our application with commensurately large FE trees.

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, whereas the upper $N-1$ tree nodes cannot exploit the whole number of processors, anyway.

Another strategy divides the *entire* given FE tree into N partitions of approximately equal size by cutting edges (and not root nodes of subtrees). For this purpose, we set $\mathsf{w}(p) = \ell(p)$ in each tree node. So this application of Algorithm HF takes into account that the main memory resources of the processors become the limiting factor if very high accuracy of the simulation is required.

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.

Note that (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 3. *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* $\mathsf{w}(T_1), \mathsf{w}(T_2) \in [\frac{1}{4}\mathsf{w}(T); \frac{3}{4}\mathsf{w}(T)]$.

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 4. *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 $\frac{9}{4} \cdot \frac{\mathsf{w}(T)}{N}$.*

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 [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 4 still applies.

5 Conclusion

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).

Load balancing for distributed hierarchical FE 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 worst-case bounds on the maximum load achieved for applications with good bisectors in general and for 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.

At present we are implementing the load balancing methods proposed in this paper and integrating them into the existing finite element simulations software. We expect considerable improvements as compared to the static (compile-time) processor allocation currently in use. See [BEE98] for first results.

References

[BEE98] Stefan Bischof, Ralf Ebner, and Thomas Erlebach. Load Balancing for Problems with Good Bisectors, and Applications in Finite Element Simulations: Worst-case Analysis and Practical Results. SFB-Bericht 342/05/98 A, SFB 342, Institut für Informatik, Technische Universität München, 1998. http://wwwmayr.in.tum.de/berichte/1998/TUM-I9811.ps.gz.

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

[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.

[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.

[SHK95] Behrooz A. Shirazi, Ali R. Hurson, and Krishna M. Kavi. *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.