INFORMATION AND COMPUTATION 96, 1–54 (1992)

Optimal Placement of Identical Resources in a Tree*

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY D. GRIFFETH

Bell Communications Research, Morristown, New Jersey

LEONIDAS GUIBAS

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

The problem of placing a number t of identical resources at nodes of a tree so as to minimize the total expected cost of servicing a set of t requests arriving randomly at nodes is considered. The cost of servicing a particular set of requests is the total distance in the tree between each request and its assigned resource. Distance is measured by the number of edges along the unique path from the request to the resource. Optimal placements can be found in time $O(mt)$, where m is the number of edges in the tree. Allowing resources to be split into fractional-sized pieces which can be placed separately neither reduces the cost of an optimal placement nor provides an obvious way to find optimal placements significantly faster. Simple, natural "fair" placements whose cost differs from optimality by at most the number of edges in the tree are described. For any fixed tree T, the cost of these placements grows as $O(\sqrt{t})$, where the constant implicit in the "O" notation depends on the size and shape of T. In the case of balanced trees with k leaves, that constant is at most $\sqrt{2k/\pi}$. The placement problem becomes somewhat simpler for a complete (rooted) d-ary tree with a symmetric probability density function for request arrivals, and in that case slightly stronger results are possible. For example, an optimal placement can be found in time $O(\min\{\ell, \log_d t\} + t)$, where ℓ is the height of the tree, and the placement is symmetric and fair.

* This research was supported in part by the National Science Foundation under Grants MCS77-02474, MCS77-15628, MCS78-01698, MCS80-03337, MCS-8116678, MCS-8200854, DCR-8302391, DCR-8405478, and CCR-8915206, by the U.S. Army Research Office under Contracts DAAG29-79-C-0155 and DAAG29-84-K-0058, by the Office of Naval Research under Contracts N00014-79-C-0873, N00014-80-C-0221, N00014-82-K-0154, N00014-85-K-0168, and N00014-89-J-1980, and by the Defense Advanced Research Projects Agency (DARPA) under Contracts N00014-83-K-0125 and N00014-89-J-1988.

0890-5401/92 $3.00

CONTENTS

1. INTRODUCTION

We consider the problem of placing some number t of identical resources at nodes of a distributed network in such a way as to minimize the expected cost of servicing a random set of t requests. Each request originates at a randomly chosen node according to an underlying probability distribution, so requests can be regarded as independent identically distributed random variables over nodes. We often identify a request with the node at which it originates.

Each request is serviced by assigning a distinct resource to it. The *cost* of servicing a request is the distance from the request to its assigned resource. Distance between two nodes is measured by the number of edges along the shortest path that joins them. In this paper, we consider only tree networks, so the shortest path is always unique.

This problem arises in several contexts. Suppose, for example, that the resources are processors and that each request is to establish a virtual connection with a processor which will be used for a very long period of time. An important cost might be the total amount of communication traffic introduced into the network over the duration of the connection, which grows in proportion to the distance between a request and its assigned processor. Because of the long holding times, it is probably reasonable to expend considerable effort to find a good assignment of resources to requests. Assuming each set of random requests is serviced optimally but that processors cannot be moved from their initial positions, we can ask where processors should be placed in the network so as to minimize the expected communication costs.

For a similar example, suppose that the resources are copies of a read-only file residing on separate disks and that requests are programs which make such heavy use of the file that it is necessary to assign each a distinct copy of the file to avoid unacceptable disk contention. One might then want to minimize the amount of network traffic needed for all the

programs to communicate with their files, or equivalently, to minimize the average communication delay incurred by the programs while waiting for responses from the files. Again, we ask where the disks should be placed in the network so as to minimize the expected cost.

Thus, we are considering the resource placement problem under the following assumptions:

- Requests arrive randomly and independently at nodes.
- Resources are permanently assigned to nodes and cannot be moved in response to a request.
- The underlying network is a tree.
- The cost of servicing a particular set of requests is the least total distance between requests and their assigned resources taken over all possible assignments of resources to requests.
- The cost of a particular placement of t resources is the expected cost of servicing a set of t random requests.

A placement is *optimal* if its cost is the least among all placements of the same number of resources. We give upper bounds on the costs of optimal placements, and we give certain easily described placements which are close to optimal. We also provide fast algorithms for finding optimal and near-optimal placements.

All our results assume that the network is configured as a tree. In the case of general connected networks, one can apply these results to a spanning tree of the network, but of course the cost function then considers only paths that lie in the tree. This may or may not be a useful cost function depending on the way routes are constructed.

Our methods also allow us to obtain upper bounds for each edge of the tree on the expected number of paths between requests and assigned resources on which that edge appears. This can be used to determine the capacities required on the links and nodes to handle communication traffic between requests and their assigned resources.

Our results are intended to apply to networks with point-to-point communication. For broadcast networks such as Ethernet, it is not clear whether the strategy of configuring the network as a tree can be used to advantage for resource-allocation problems. Also, our results apply only to situations in which only one resource is required by each request. If it is important to consider multiple resources—e.g., various different files—then it might be necessary to consider the distances of the different resources from each other as well as from the requests. In this case, an approach more complicated than ours seems to be required.

We generalize the above to allow for the possibility that fractions of resources, rather than whole resources, might be located at some nodes of

the network tree. Each request is then for a unit quantity of resource, and a given request can be satisfied by fractional resources located at several different nodes. This situation is reasonable if, for instance, the resources are large blocks of available data storage space, for it might be perfectly permissible for a user to obtain parts of his needed storage from different places.

The rest of the paper is organized as follows. In Section 2, we present some mathematical preliminaries, in particular, some properties of convex, piecewise linear functions. Our cost measures turn out to be functions of this type. In Section 3, we show that our problem can be regarded as a min-cost flow problem, sometimes known as the Hitchcock transportation problem. In Section 4, we derive properties of optimal placements and present an algorithm for finding an optimal placement which runs in time $O(mt)$, where m is the number of edges in the tree and t the number of resources. In Section 5, we bound the expected cost of optimal placements by analyzing the cost of a natural placement which puts a number of resources on each node equal to the expected number of requests at that node. Surprisingly, this placement is not always optimal, although its cost is never more than an amount m greater than the optimum. In Section 6, we analyze the special case of a complete tree with a symmetric request probability function.

A preliminary version of this paper appeared as (Fischer *et al.*, 1981). Further related results appear in (Lynch *et al.*, 1986).

2. Mathematical Preliminaries

In this section, we present some basic notation and prove a collection of results which are applied in many places later in the paper. The results give interesting properties for a certain class of convex, piecewise linear functions. The cost measures we are interested in are functions of this type. The reader may prefer to skip this section for now, returning to it later when its results are required.

2.1. *Notation*

We use the following notation.

$\mathbf{N}$ denotes the natural numbers, including 0.

$\mathfrak{R}$ denotes the real numbers.

$\mathfrak{R}^+$ denotes the nonnegative real numbers.

2.2. *Convex Functions and Convolution*

Let t is a positive integer. Let F_t be the set of functions $f: [0, t] \to \mathfrak{R}^+$ such that f is convex and piecewise linear with all singularities occurring at

integer values. In this subsection, we provide some definitions and basic results involving functions in F_t.

For $f \in F_t$, let $f'^-(x)$ be the "left-hand" derivative of f at $x \in (0, t]$ (i.e. the limit of $(f(x')-f(x))/(x'-x)$ as x' approaches x from below), and let $f'^+(x)$ be the "right-hand" derivative of f at $x \in [0, t)$ (i.e. the limit of $(f(x')-f(x))/(x'-x)$ as x' approaches x from above). For completeness, define $f'^-(0) = -\infty$ and $f'^+(t) = +\infty$. The following is obvious from the definition of F_t.

LEMMA 2.2.1. *Let $f \in F_t$ and $u \in [0, t]$. Then*

1. $f'^-(u) \leqslant f'^+(u)$, *and equality holds if* $u \notin \mathbf{N}$.
2. *If* $u < v$ *then* $f'^+(u) \leqslant f'^-(v)$.
3. *If* $n \in \mathbf{N}$ *and* $u \in (n-1, n]$ *then* $f'^-(u) = f'^-(n)$.
4. *If* $n \in \mathbf{N}$ *and* $u \in [n, n+1)$ *then* $f'^+(u) = f'^+(n)$.

DEFINITION 2.2.2. Let $f \in F_t$. $x \in [0, t]$ is the *left* (*right*) *minimum* of f if it minimizes f over the interval $[0, t]$ and is the smallest (largest) value having this property. We write $x = \text{leftmin}(f)$ ($x = \text{rightmin}(f)$).

LEMMA 2.2.3. *Let $f \in F_t$ and $u \in [0, t]$. If $u \in [\text{leftmin}(f), \text{rightmin}(f)]$, then*

$$f'^-(u) \leqslant 0 \leqslant f'^+(u). \tag{1}$$

Moreover, if $u = \text{leftmin}(f)$ then $f'^-(u) < 0$ and if $u = \text{rightmin}(f)$ then $0 < f'^+(u)$.

Proof. Obvious. ∎

Let $d \in \mathbf{N}$. We use $\mathbf{u}$ to denote the vector $(u_1, ..., u_d) \in (\Re^+)^d$, and we let $\sum \mathbf{u} = \sum_{i=1}^d u_i$.

We define the "convolution" of two functions in F_t. The convolution of f_1 and f_2 describes the "best" way to split up a given nonnegative amount u of "resource" into two parts, u_1 and u_2, so as to minimize the sum of $f_1(u_1)$ and $f_2(u_2)$. In our later applications, the functions f_1 and f_2, as well as their convolution, are cost functions.

DEFINITION 2.2.4. Let $f_1 \in F_{t_1}$, $f_2 \in F_{t_2}$. We define $f = f_1 * f_2$ ("the convolution") by

$$f(u) = \min_{\substack{u_1 \in [0, t_1] \\ u_2 \in [0, t_2] \\ u_1 + u_2 = u}} [f_1(u_1) + f_2(u_2)].$$

for $u \in [0, t_1 + t_2]$.

LEMMA 2.2.5. *Convolution is associative. In particular,*

$$(f_1 * f_2 * \cdots * f_d)(u) = \min_{\substack{\mathbf{u} \in [0,\, t]^d \\ \sum \mathbf{u} = u}} \sum_{i=1}^{d} f_i(u_i).$$

for $u \in [0, dt]$, $f_1, f_2, \ldots, f_d \in F_t$.

Proof. It suffices to show it for $d = 3$. Then the general result follows by induction. The $d = 3$ case is immediate from the definition and the associativity of addition. ∎

DEFINITION 2.2.6. Let $f_1, f_2, \ldots, f_d \in F_t$. We say that $\mathbf{u} \in [0, t]^d$ is a *minimizing vector for* $(f_1, f_2, \ldots, f_d)$ if

$$(f_1 * f_2 * \cdots * f_d)\left(\sum \mathbf{u}\right) = \sum_{i=1}^{d} f_i(u_i).$$

It follows from the above that a minimizing vector $\mathbf{u}$ for $(f_1, f_2, \ldots, f_d)$ minimizes $\sum f_i(v_i)$ over all $\mathbf{v}$ with $\sum \mathbf{v} = \sum \mathbf{u}$. The convolution $(f_1 * f_2 * \cdots * f_d)(u)$ is the minimum so achieved. We now characterize minimizing vectors.

LEMMA 2.2.7. *Let* $f_1, f_2, \ldots, f_d \in F_t$. *The vector* $\mathbf{u} = (u_1, u_2, \ldots, u_d)$ *is a minimizing vector for* $(f_1, f_2, \ldots, f_d)$ *iff there exists* γ *such that*

$$f_i'^{-}(u_i) \leqslant \gamma \leqslant f_i'^{+}(u_i)$$

for all i, $1 \leqslant i \leqslant d$.

Proof. ($\Rightarrow$) Let $\mathbf{u}$ be a minimizing vector for $(f_1, f_2, \ldots, f_d)$, and suppose to the contrary that no such γ exists. Suppose $f_i'^{-}(u_i) > f_j'^{+}(u_j)$ for some choice of i and j as illustrated in Fig. 1. Then the local perturbation

$$\begin{Bmatrix} u_i \leftarrow u_i - \varepsilon \\ u_j \leftarrow u_j + \varepsilon \end{Bmatrix}$$

for sufficiently small $\varepsilon > 0$ reduces $\sum f_i(u_i)$ while leaving $\sum \mathbf{u}$ unchanged, contradicting the assumption that $\mathbf{u}$ is a minimizing vector. Hence, $f_i'^{-}(u_i) \leqslant f_j'^{+}(u_j)$ for all i, j, so the interval

$$[\max_i \{f_i'^{-}(u_i)\}, \min_j \{f_j'^{+}(u_j)\}]$$

is non-empty, and any γ in that interval satisfies the conditions of the lemma.

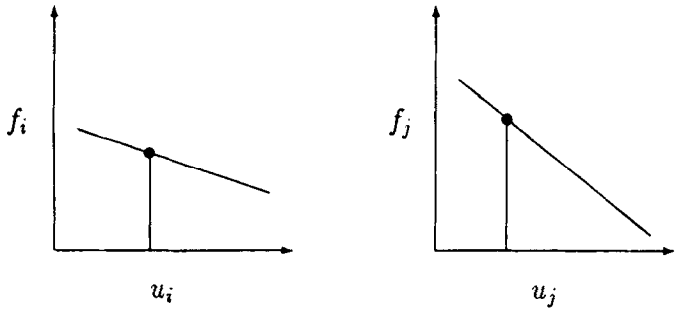

FIG. 1. A nonminimizing vector.

($\Leftarrow$) For the converse, suppose $f_i'^{-}(u_i) \leqslant \gamma \leqslant f_i'^{+}(u_i)$ for all i, $1 \leqslant i \leqslant d$. Let $\mathbf{v}$ be any vector in $[0, t]^d$ with $\sum \mathbf{v} = \sum \mathbf{u}$. It follows from the convexity of each f_i that

$$f_i(v_i) \geqslant f_i(u_i) + \gamma(v_i - u_i).$$

Summing, we get

$$\sum_{i=1}^{d} f_i(v_i) \geqslant \sum_{i=1}^{d} f_i(u_i) + \gamma \sum_{i=1}^{d} (v_i - u_i) = \sum_{i=1}^{d} f_i(u_i).$$

Thus, $\mathbf{u}$ is a minimizing vector for $(f_1, f_2, ..., f_d)$ as desired. ∎

Lemma 2.2.7 gives us an easy way of visualizing the convolution $f_1 * f_2 * \cdots * f_d$ using the "kinetic" approach introduced in (Guibas *et al.*, 1983). Regard each f_i as describing a polygonal path in the plane, and imagine d cars, one traveling along each such path. Car i starts at point $(0, f_i(0))$, $1 \leqslant i \leqslant d$. Only one car moves at a time, and cars move with a constant velocity along the x-axis. The goal is for the cars to keep the sum of their y positions as small as possible. Clearly, the best strategy is for the car on the path of least (or "most negative") slope to move first until it reaches a corner. Because each f_i is convex and piecewise linear, once a car starts moving, it can keep moving until it reaches a corner (where f_i changes slope).

The value of the convolution $(f_1 * f_2 * \cdots * f_d)(u)$ is just the sum of the y positions of the cars at time u, and the positions of the cars at time u is a minimizing vector for u. It follows from this description that the slope of the convolution at $u \notin \mathbf{N}$ is the same as the slope of the f_i that corresponds to the car that was moving at that time. Fig. 2 illustrates this notion of "merging" in computing the convolution of two functions.

The construction above can be described more formally as follows. Let $f_1, ..., f_d \in F_t$. Let $\gamma_{i,j} = f_i'^{+}(j)$ for $i \in \{1, ..., d\}$ and $j \in \{0, ..., t-1\}$. Because

each f_i is convex, each sequence $\hat{\gamma}_i = (\gamma_{i,0}, ..., \gamma_{i,t-1})$ is nondecreasing. Let $\zeta = (\zeta_0, \zeta_1, ..., \zeta_{dt-1})$ be the nondecreasing sequence of slopes that results from merging the sequences $\hat{\gamma}_1, ..., \hat{\gamma}_d$. There is a natural one-to-one correspondence between the elements in ζ and the elements in the sequences $\hat{\gamma}_1, ..., \hat{\gamma}_d$, since ζ is just a reordering of those elements. We specify this correspondence by functions α and β such that ζ_k corresponds to $\gamma_{\alpha(k),\beta(k)}$.

We define inductively a function $f(u)$ and an indexed set of vectors $\mathbf{v}^{(u)}$ for $u \in [0, dt]$ as follows: $f(0) = \sum f_i(0)$, and $\mathbf{v}^{(0)} = (0, 0, ..., 0)$. Let k be a nonnegative integer and assume that $f(u')$ and $\mathbf{v}^{(u')}$ are defined for all u' in the interval $[0, k]$, where $k \leqslant dt - 1$, and let $u \in (k, k+1]$. Define $f(u) = f(k) + \zeta_k \cdot (u - k)$. Define $\mathbf{v}^{(u)} = \mathbf{v}^{(k)} + \mathbf{I}_{\alpha(k)} \cdot (u - k)$, where $\mathbf{I}_j$ is the unit vector that is 0 in all components except for the j th component which is 1.

LEMMA 2.2.8. *Let $f(u)$ and $\mathbf{v}^{(u)}$ be as defined above.*

1. $\sum \mathbf{v}^{(u)} = u$, *and if* $k \in \mathbf{N}$, *then* $\mathbf{v}^{(k)} \in \mathbf{N}^d$.
2. *Each* $\mathbf{v}^{(u)}$ *is a minimizing vector for* $(f_1, f_2, ..., f_d)$.

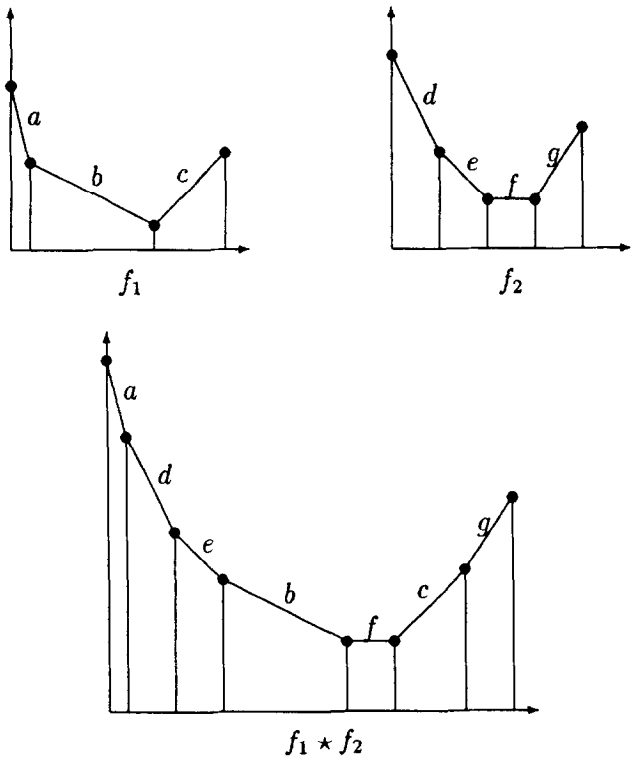

FIG. 2. The convolution of two functions.

3. $v_{\alpha(k)}^{(k)} = \beta(k)$ *for all integers* $k \in [0, dt]$.
4. $f(u) = \sum_i f_i(v_i^{(u)})$.
5. $f = f_1 * f_2 * \cdots * f_d$.

Proof. Part 1 is immediate from the definition of $\mathbf{v}^{(u)}$.

For part 2, the vector $\mathbf{v}^{(0)} = (0, 0, ..., 0)$ is a minimizing vector. It is easily shown for each $u \in (0, dt]$ that the vector $\mathbf{v}^{(u)}$ satisfies the conditions of Lemma 2.2.7; hence, $\mathbf{v}^{(u)}$ is a minimizing vector for $(f_1, f_2, ..., f_d)$.

For part 3, an easy induction on k shows that

$$v_i^{(k)} = |\{k' < k \mid \alpha(k') = i\}|$$

and

$$\beta(k) = |\{k' < k \mid \alpha(k') = \alpha(k)\}|.$$

The result follows immediately by taking $i = \alpha(k)$.

We show part 4 by induction on $k' = \lceil u \rceil$. If $k' = \lceil u \rceil = 0$ then $u = 0$ and part 4 obviously holds.

Now suppose $0 < k' = \lceil u \rceil \leqslant dt$. Let $k = k' - 1$, let $i = \alpha(k)$, and let $z = \beta(k)$. Since $\zeta_k = \gamma_{i,z}$, it follows that $f_i(z + (u-k)) = f_i(z) + \zeta_k \cdot (u-k)$. Hence,

$$f(u) = f(k) + \zeta_k \cdot (u-k) = f(k) + f_i(z + (u-k)) - f_i(z).$$

By the induction hypothesis, $f(k) = \sum_j f_j(v_j^{(k)})$. Thus,

$$f(u) = \sum_j f_j(v_j^{(k)}) + f_i(z + (u-k)) - f_i(z).$$

By part 3, $\beta(k) = v_{\alpha(k)}^{(k)}$, so $z = v_i^{(k)}$. The result follows from the fact that $v_j^{(u)} = v_j^{(k)}$ for $j \neq i$, and $v_i^{(u)} = v_i^{(k)} + (u-k) = z + (u-k)$.

Finally, part 5 follows immediately from parts 1, 2, and 4 and the definitions of convolution and minimizing vector. ∎

Lemma 2.2.8 says in particular that is is always possible to get minimizing vectors for integer values with integral components.

COROLLARY. *If* $f_1, f_2, ..., f_d \in F_t$, *then*

$$f = f_1 * f_2 * \cdots * f_d \in F_{dt}.$$

Proof. From Lemma 2.2.8, it is clear that f is convex and has slope discontinuities only at integer values. ∎

2.3. *Decreasing Closure*

Recall that the convolution $(f_1 * f_2 * \cdots * f_d)(u)$ gives the minimum value of $\sum f_i(u_i)$ over all ways of partitioning u into d pieces $u_1, \ldots, u_d$ with $\sum u_i = u$. In our applications, however, we do not require that all of u be "used," so we wish instead to minimize $\sum f_i(u_i)$ subject only to the constraint that $\sum u_i \leqslant u$. Thus, we consider in this section an asymmetric variant of the convolution.

DEFINITION 2.3.1. If $f \in F_t$, then define $f_{\leqslant}$ by

$$f_{\leqslant}(u) = \min_{v \in [0, u]} f(v),$$

for $u \in [0, t]$. We call $f_{\leqslant}$ the *decreasing closure* of f.

LEMMA 2.3.2. *Let $f \in F_t$. Then $f_{\leqslant}$ is a non-increasing function in F_t. Moreover,*

$$f_{\leqslant}(u) = \begin{cases} f(u) & \text{for } u \leqslant \text{leftmin}(f) \\ f(\text{leftmin}(f)) & \text{for } u > \text{leftmin}(f). \end{cases}$$

Proof. Obvious. ∎

DEFINITION 2.3.3. In case $f = f_1 * f_2 * \cdots * f_d$, we call $f_{\leqslant}$ the *decreasing convolution* of $(f_1, f_2, \ldots, f_d)$.

Note that $f_{\leqslant} \in F_{dt}$ by Lemma 2.3.2, assuming each $f_i \in F_t$.

LEMMA 2.3.4. *Let $f_1, f_2, \ldots, f_d \in F_t$. Then*

$$(f_1 * f_2 * \cdots * f_d)_{\leqslant}(u) = \min_{\substack{\mathbf{u} \in [0, t]^d \\ \sum \mathbf{u} \leqslant u}} \sum_{i=1}^{d} f_i(u_i)$$

for $u \in [0, dt]$.

Proof. Obvious. ∎

The following is a key optimality definition, which is used later to describe the "best" way of splitting up a set of resources when not all of them must be used.

DEFINITION 2.3.5. Let $f_1, f_2, \ldots, f_d \in F_t$, $u \in [0, dt]$. We say that $\mathbf{u} \in [0, t]^d$ is *optimal for u and $(f_1, f_2, \ldots, f_d)$* if $\sum \mathbf{u} \leqslant u$ and

$$(f_1 * f_2 * \cdots * f_d)_{\leqslant}(u) = \sum_{i=1}^{d} f_i(u_i).$$

The following lemma characterizes optimal vectors in terms of the relationship between the sum of their components and the minimum points of the convolution.

LEMMA 2.3.6. *Let* $f = f_1 * f_2 * \cdots * f_d$. *A vector* $\mathbf{u}$ *is optimal for* u *and* $(f_1, f_2, ..., f_d)$ *iff* $\mathbf{u}$ *is a minimizing vector for* $(f_1, f_2, ..., f_d)$ *and*

$$\min(u, \text{leftmin}(f)) \leqslant \sum \mathbf{u} \leqslant \min(u, \text{rightmin}(f)).$$

Proof. Let $f_1, f_2, ..., f_d \in F_t$ and let $f = f_1 * f_2 * \cdots * f_d$. It suffices to consider three cases.

Case 1. $u < \text{leftmin}(f)$. Then f and $f_{\leqslant}$ are strictly decreasing and equal on the interval $[0, u]$, so $\mathbf{u}$ is optimal for u iff $\sum \mathbf{u} = u$ and $\mathbf{u}$ is a minimizing vector for $(f_1, f_2, ..., f_d)$.

Case 2. $\text{leftmin}(f) \leqslant u \leqslant \text{rightmin}(f)$. Then f and $f_{\leqslant}$ are constant and equal in the interval $[\text{leftmin}(f), u]$, so $\mathbf{u}$ is optimal for u iff $\sum \mathbf{u} \in [\text{leftmin}(f), u]$ and $\mathbf{u}$ is a minimizing vector for $(f_1, f_2, ..., f_d)$.

Case 3. $u > \text{rightmin}(f)$. Then f is strictly increasing in the interval $[\text{rightmin}, u]$ and $f_{\leqslant}(u) = f(\text{leftmin}(f)) = f(\text{rightmin}(f))$, so $\mathbf{u}$ is optimal for u iff $\sum \mathbf{u} \in [\text{leftmin}(f), \text{rightmin}(f)]$ and $\mathbf{u}$ is a minimizing vector for $(f_1, f_2, ..., f_d)$. ∎

Now we give a definition to describe the case where all the functions f_i are simultaneously minimized.

DEFINITION 2.3.7. Let $f_1, f_2, ..., f_d \in F_t$. A vector $\mathbf{u}$ is *saturated* for $(f_1, f_2, ..., f_d)$ provided that $f_i'^{-}(u_i) \leqslant 0 \leqslant f_i'^{+}(u_i)$ for all i, $1 \leqslant i \leqslant d$.

It is obvious from this definition that a saturated vector simultaneously minimizes each of the f_i and thus is a minimizing vector for $(f_1, f_2, ..., f_d)$.

The following lemma gives some properties of saturated vectors.

LEMMA 2.3.8. *Let* $\mathbf{u}$ *and* $\mathbf{v}$ *be vectors and let* $u, v \in [0, dt]$.

1. *If* $\mathbf{u}$ *is optimal for* u *and* $(f_1, f_2, ..., f_d)$, *then either* $\sum \mathbf{u} = u$ *or else* $\mathbf{u}$ *is saturated for* $(f_1, f_2, ..., f_d)$.

2. *If* $\mathbf{u}$ *is saturated for* $(f_1, f_2, ..., f_d)$, $\sum \mathbf{u} \leqslant \sum \mathbf{v}$, *and* $\mathbf{v}$ *is optimal for* v *and* $(f_1, f_2, ..., f_d)$, *then* $\mathbf{v}$ *is saturated for* $(f_1, f_2, ..., f_d)$.

Proof. 1. Let $\mathbf{u}$ be optimal for u. Then $\sum \mathbf{u} \leqslant u$. If $\sum \mathbf{u} = u$, we are done, so assume $\sum \mathbf{u} < u$. We show $\mathbf{u}$ is saturated.

Assume to the contrary that $\mathbf{u}$ is not saturated. Then some f_j is not minimized by u_j. But then perturbing u_j by $\pm$ some small ε yields a new vector

$\mathbf{u}'$ such that $\sum f_i(u_i') < \sum f_i(u_i)$ and $\sum \mathbf{u}' < u$. This contradicts the assumed optimality of $\mathbf{u}$.

2. Since $\mathbf{u}$ is saturated and $\sum \mathbf{u} \leqslant \sum \mathbf{v}$, $\mathbf{u}$ is optimal for $\sum \mathbf{v}$. But $\mathbf{v}$ is also optimal for v, so $\sum f_i(u_i) = \sum f_i(v_i)$. Since $\mathbf{u}$ is saturated, it follows that $\mathbf{v}$ is also. ∎

The following lemma shows how minor perturbations (not crossing integer boundaries) of optimal vectors remain optimal.

LEMMA 2.3.9. *Let* $\mathbf{u}$ *be optimal for* u *and* $(f_1, f_2, ..., f_d)$. *Let* $\mathbf{v} \in [0, t]^d$ *and* $v \in [0, dt]$ *such that*

(a) $\lfloor u_i \rfloor \leqslant v_i \leqslant \lceil u_i \rceil$,

(b) $\lfloor u - \sum \mathbf{u} \rfloor \leqslant v - \sum \mathbf{v} \leqslant \lceil u - \sum \mathbf{u} \rceil$.

Then $\mathbf{v}$ *is optimal for* v *and* $(f_1, f_2, ..., f_d)$.

Proof. By Lemma 2.3.6, $\mathbf{u}$ is a minimizing vector for $(f_1, f_2, ..., f_d)$, so Lemma 2.2.7 yields γ such that $f_i'^{-}(u_i) \leqslant \gamma \leqslant f_i'^{+}(u_i)$ for all i, $1 \leqslant i \leqslant d$. But then condition (a) implies that $f_i'^{-}(v_i) \leqslant \gamma \leqslant f_i'^{+}(v_i)$ for all i since either $u_i \in \mathbf{N}$ and $u_i = v_i$ or $u_i \notin \mathbf{N}$ and

$$f_i'^{-}(v_i) \leqslant f_i'^{-}(u_i) = \gamma = f_i'^{+}(u_i) \leqslant f_i'^{+}(v_i).$$

Another use of Lemma 2.2.7 shows that $\mathbf{v}$ is a minimizing vector.

If $\mathbf{u}$ is saturated, then γ can be taken equal to 0, so $\mathbf{v}$ is also saturated. This suffices since condition (b) implies $\sum \mathbf{v} \leqslant v$. On the other hand, if $\mathbf{u}$ is not saturated, then Lemma 2.3.8 implies that $\sum \mathbf{u} = u$, and condition (b) implies that $\sum \mathbf{v} = v$. Since $\mathbf{u}$ is optimal for u, γ can be assumed to be $\leqslant 0$. Thus, $\mathbf{v}$ must be optimal for v and $(f_1, f_2, ..., f_d)$. ∎

A variation of Lemma 2.2.8 holds for the decreasing clusure of the convolution.

LEMMA 2.3.10. *Let* $f_1, f_2, ..., f_d \in F_t$ *and let* $f = f_1 * f_2 * \cdots * f_d$. *Then there is a sequence of vectors* $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, ..., \mathbf{u}^{(dt)}$ *such that*

1. *Each* $\mathbf{u}^{(i)}$ *has integral components and is optimal for* i *and* $(f_1, f_2, ..., f_d)$.

2. $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \delta'^{(i)}$, $1 \leqslant i \leqslant dt$, *where* $\delta'^{(i)}$ *is a unit vector of all* 0's *except for at most one* 1. *Furthermore, if* $\delta'^{(i)} = (0, ..., 0)$, *then* $\delta'^{(j)} = (0, ..., 0)$ *for all* $j \geqslant i$.

Proof. The proof follows from Lemmas 2.3.2 and 2.2.8. After the integer u where the global minimum of $f_1 * f_2 * \cdots * f_d$ is passed, $(f_1 * f_2 * \cdots * f_d)_{\leqslant}$ stays constant. ∎

The final lemma of this subsection deals with the symmetric case—where all the component functions are the same. It says that if there are not enough resources for the "average" number to reach the leftmin, then the vector cannot be saturated.

LEMMA 2.3.11. *Let $f_i = g \in F_i$ for all i, $1 \leqslant i \leqslant d$. Let* $\mathbf{u}$ *be optimal for u and $(f_1, f_2, ..., f_d)$, and assume $u/d <$* leftmin(g). *Then* $\mathbf{u}$ *is not saturated for $(f_1, f_2, ..., f_d)$, and $\sum \mathbf{u} = u$.*

Proof. Since $\sum \mathbf{u} \leqslant u$, there is some i with $u_i \leqslant u/d$. Since $u/d <$ leftmin(f), it follows that $\mathbf{u}$ is not saturated. Lemma 2.3.8 implies the other conclusion. ∎

3. PROBLEM FORMULATION

In this section, we formulate the resource placement problem, first in terms of matchings of resources to requests and then as a min-flow problem. The latter formulation is used in the remainder of the paper.

3.1. *Basic Definitions*

A *tree* $T = (V_T, E_T)$ is a finite undirected acyclic graph, where V_T is the vertex (node) set and E_T is the edge set. We abuse notation and write $\mu \in T$ to mean $\mu \in V_T$. We let $m_T = |E_T|$ be the number of edges in T, and we use L_T to denote the set of leaves (vertices of degree one). We often omit subscripts when the meaning is clear from context.

We arbitrarily orient each edge e of T by distinguishing its two endpoints—one is called "low(e)," the other is called "high(e)," and we think of e as being directed from low(e) to high(e). The removal of e divides T into two disconnected components: $A(e)$ is the component containing high(e) and $B(e)$ is the component containing low(e). (It is suggestive to read A as "above" and B as "below.") We write $\hat{A}(e)$ ($\hat{B}(e)$) to denote the subtree of T consisting of $A(e)$ ($B(e)$) together with the edge e.

A tree T is *rooted* if it has a distinguished node called the root and denoted by root. If $v, \omega \in V$ and $v \neq \omega$, we say ω is a *descendant* of v if v lies on the unique path from ω to root, and ω is an *immediate descendant* (also called a *child*) of v if v is adjacent to ω on that path. We assume that all edges are directed towards root, that is, for any edge e, high(e) is closer to root than to low(e). A node v of a rooted tree is at *level j* if there are j edges on the path from v to root. Thus, root is at level 0, and all edges are directed from vertices with larger level numbers to vertices with smaller numbers. A tree T is a *complete d-ary* tree if it is rooted, every

non-leaf vertex has exactly d immediate descendants, and every leaf is at the same level.

In dealing with rooted trees, one commonly uses the node μ to name the subtree consisting of μ and all of its descendants. In this work, we find it convenient to denote subtrees by edges rather than by nodes, using the notation $B(e)$ introduced above. This has the disadvantage that the entire tree T cannot be so named, for $T \neq B(e)$ for any e. Therefore, as a technical convenience, we assume that every rooted tree T has an extra edge "rootedge" such that low(rootedge) = root. Thus, $\hat{B}$(rootedge) $= T$, and B(rootedge) is the subtree below and including root. The other end of rootedge, high(rootedge), is a distinguished leaf of T denoted by "top." We treat top as a special case. It is henceforth *not* considered to be a leaf of T, nor is it a descendant of root, and we define its level to be -1. (In fact, for consistency, we define root to be a descendant of top.) Thus, all edges are directed towards top, and a node μ is at level j if there are $j+1$ edges on the (directed) path from μ to top.

A set of *requests* is described by a function $r: V \to \mathbf{N}$ such that $r(v)$ is the number of requests originating at vertex v. If T is rooted, then we always assume that no requests originate at top, thus, $r(\text{top}) = 0$. A *placement* of resources is described by a function $s: V \to \Re^+$ such that $s(v)$ is the number of resources placed at vertex v. A placement s is *whole* provided that $s(v) \in \mathbf{N}$ for all $v \in V$.[1]

For any tree T and function $g: V_T \to \Re^+$, let

$$\text{total}(T, g) = \sum_{v \in V_T} g(v).$$

For the remainder of the paper, we fix a particular, arbitrary positive integer t, which represents both the total number of requests and the total number of resources. In particular, for every set r of requests and every placement s that we consider, we assume that total$(T, r) =$ total$(T, s) = t$.

Finally, $\phi: V \to \Re^+$ is a *probability density function* on T if $\phi(v) \in [0, 1]$ for all $v \in V$, and total$(T, \phi) = 1$. If T is rooted, then we assume that $\phi(\text{top}) = 0$. The function $\phi(v)$ is the probability that a random request arrives at node v. We say that ϕ is *symmetric* if T is rooted and ϕ depends only on the level of a vertex in T. (In other words, ϕ is the same for all vertices at the same level.)

[1] Although we permit resources to be placed at top, it will turn out that in any optimal placement, $s(\text{top}) = 0$, for it will always reduce the cost of the placement to move any resources from top to root. Thus, our restriction to trees containing a special rootedge will lead to no loss of generality in our results.

3.2. *Matching Requests to Resources*

Our problem is to determine how to place resources in a tree T so as to minimize the expected cost of matching requests to resources, where requests arrive at vertices of the tree according to a probability density function ϕ on V. We first formulate this problem as a matching problem on a tree.

Let T be a tree.[2] If r is a set of requests and s a placement for T, then a *matching* for r, s on T is a function $M: V \times V \to \mathfrak{R}^+$ such that

$$r(\mu) = \sum_{v \in V} M(\mu, v) \tag{2}$$

and

$$s(v) = \sum_{\mu \in V} M(\mu, v). \tag{3}$$

The value $M(\mu, v)$ gives the number of requests at node μ satisfied by resources from node v. A function satisfying conditions (1) and (2) will exist whenever $\texttt{total}(T, r) = \texttt{total}(T, s)$, which we always assume. The *cost* of a matching M is defined by

$$\texttt{cost}(M) = \sum_{\mu, v \in V} M(\mu, v) \cdot d(\mu, v),$$

where $d(\mu, v)$ is the number of edges in the unique path from μ to v in T. A matching M is *optimal* for r, s if $\texttt{cost}(M)$ is minimal over all matchings for r, s.

EXAMPLE 3.2.1. Figure 3 illustrates the above definitions. Two different matchings are given for identical request arrivals and resource placements. A request arrival at a vertex is indicated by an r, a resource by an s. The two s's on the right-hand leaf indicate that two resources have been placed there.

3.3. *Min-Cost Flows*

Given a matching M for r, s, it is natural to think of each request "flowing" along the unique path in the tree to its matching resource. This leads us to formulate a related min-cost flow problem, also known as the Hitchcock transportation problem (cf. Papadimitriou and Steiglitz, 1982).

[2] Neither this nor the following subsection requires the assumption that T be rooted. We will restrict attention to rooted trees later, starting in Subsection 3.4.

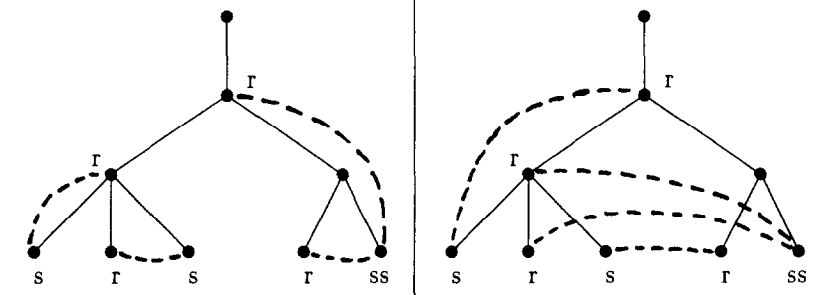

FIG. 3. Two matchings of the same request arrivals and resource placements for the same tree.

We show that the cost of an optimal matching for r, s is the same as the cost of the flow in the related flow problem for r, s, thus enabling us to restrict attention to flows in the remaining sections of this paper.

A *flow* for r, s on T is a function $f: V \times V \to \mathfrak{R}$. $f(\mu, \nu)$ specifies the amount of "flow" along the edge $\{\mu, \nu\}$ moving out of μ and into ν. (A negative value for $f(\mu, \nu)$ indicates "flow" moving into μ from ν.) The flow function satisfies three "conservation of flow" conditions:

$$r(\mu) - \sum_{\nu} f(\mu, \nu) - s(\mu) = 0, \tag{4}$$

$$f(\mu, \nu) = -f(\nu, \mu), \tag{5}$$

$$f(\mu, \nu) = 0 \quad \text{if} \quad \{\mu, \nu\} \notin E. \tag{6}$$

Equation (4) says that the net flow at a node is zero. Equation (5) says that the flow leaving node μ along edge $\{\mu, \nu\}$ is the same as the flow entering node ν along the same edge. Equation (6) says that there is no flow along nonedges.

Using our fixed orientation of edges, we can regard f as a function $E \to \mathfrak{R}$ by letting $f(e) = f(\texttt{low}(e), \texttt{high}(e))$. We say that flow along e is *forwards* if $f(e) \geq 0$ and *backwards* otherwise. The *cost* of a flow, $\texttt{cost}(f)$, is simply $\sum_e |f(e)|$.

Because T is a tree and the number of requests is equal to the number of resources, the flow along e is exactly the difference between the number of requests and the number of resources in $B(e)$ (or equivalently, the difference between the number of resources and the number of requests in $A(e)$). This is because conservation of flow ensures that flow is not lost,

and the only place for excess flow in $B(e)$ to go is across the edge e. Hence, there is a *unique* flow for r, s on T, which is given by[3]

$$\begin{aligned} \texttt{flow}_{r,s}(e) &= \texttt{total}(B(e), r) - \texttt{total}(B(e), s) \\ &= \texttt{total}(A(e), s) - \texttt{total}(A(e), r). \end{aligned} \tag{7}$$

It is easily verified that $\texttt{flow}_{r,s}$ as defined here satisfies Eqs. (4)–(6).

We now relate the costs of matchings to the cost of flows. Let M be a matching over r, s. The quantity $\sum_{\mu \in B(e)} \sum_{\nu \in A(e)} M(\mu, \nu)$ gives the total amount[4] of requests in $B(e)$ for which the path to the assigned resource includes e. Similarly, $\sum_{\mu \in A(e)} \sum_{\nu \in B(e)} M(\mu, \nu)$ gives the total amount of requests in $A(e)$ whose corresponding path includes e. The following is a direct consequence of Eqs. (2), (3), and (7).

LEMMA 3.3.1. *Let M be a matching for r, s. Then*

$$\texttt{flow}_{r,s}(e) = \sum_{\mu \in B(e)} \sum_{\nu \in A(e)} M(\mu, \nu) - \sum_{\mu \in A(e)} \sum_{\nu \in B(e)} M(\mu, \nu).$$

THEOREM 3.3.2. *Let M be a matching for r, s. Then*

$$\texttt{cost}(\texttt{flow}_{r,s}) \leqslant \texttt{cost}(M).$$

Proof. The proof is immediate from Lemma 3.3.1. ∎

We now show that the bound of Theorem 3.3.2 is tight.

THEOREM 3.3.3. *Given T, r, s, there exists a matching M for r, s such that*

$$\texttt{cost}(M) = \texttt{cost}(\texttt{flow}_{r,s}).$$

Moreover, if s is a whole placement, then M can be chosen so that $M(\mu, \nu) \in \mathbf{N}$ for all $\mu, \nu \in V$.

Proof. We proceed by induction on the number $\#(r, s)$ of edges e for which $|\texttt{flow}_{r,s}(e)| > 0$.

Basis: $\#(r, s) = 0$. Then $\texttt{flow}_{r,s}(e) = 0$ for every edge e. It follows from Eq. (4) that $r(\mu) = s(\mu)$ for all μ. Hence, the function M defined by $M(\mu, \mu) = r(\mu)$ and $M(\mu, \nu) = 0$ when $\mu \neq \nu$ is a matching, and $\texttt{cost}(M) = 0 = \texttt{cost}(\texttt{flow}_{r,s})$ as desired.

[3] Strictly speaking, we define $\texttt{flow}_{r,s}(\mu, \nu) = \texttt{flow}_{r,s}(e)$ if $\mu = \texttt{low}(e)$ and $\nu = \texttt{high}(e)$ for some edge e, $\texttt{flow}_{r,s}(\mu, \nu) = -\texttt{flow}_{r,s}(e)$ if $\mu = \texttt{high}(e)$ and $\nu = \texttt{low}(e)$ for some edge e, and $\texttt{flow}_{r,s}(\mu, \nu) = 0$ if $\{\mu, \nu\} \notin E$.

[4] Recall that we do not require a matching to be integral.

Inductive step: $\#(r, s) > 0$. We first show that there are nodes μ, ν such that $\texttt{flow}_{r,s}(\mu, \nu) > 0$, and for all μ', $\texttt{flow}_{r,s}(\mu', \nu) \geqslant 0$. In words, there is a node ν with inward flow along some edge and no outward flow. Suppose this were not the case. Then from every node ω with positive inward flow, there would be another node $\omega' \neq \omega$ for which $\texttt{flow}_{r,s}(\omega, \omega') > 0$. Because T is finite and a tree, a simple induction would show that there must be an edge $\{\omega, \omega'\}$ for which $\texttt{flow}_{r,s}(\omega, \omega') > 0$ and $\texttt{flow}_{r,s}(\omega', \omega) > 0$. But this is impossible by Eq. (5).

Let $c = \texttt{flow}_{r,s}(\mu, \nu)$. By Eq. (4) and the fact that no flow leaves v, $s(v) \geqslant c$. Define a new placement s' by

$$s'(\mu) = s(\mu) + c,$$
$$s'(\nu) = s(\nu) - c,$$
$$s'(\sigma) = s(\sigma) \qquad \text{for all} \quad \sigma \neq \mu, \nu.$$

Note that if s is a whole placement, then $c = \texttt{flow}_{r,s}(e) \in \mathbf{N}$ for all edges e, so also s' is a whole placement.

Now consider $\texttt{flow}_{r,s'}$. It is easily seen that

$$\texttt{flow}_{r,s'}(\mu, \nu) = 0,$$
$$\texttt{flow}_{r,s'}(\sigma, \tau) = \texttt{flow}_{r,s}(\sigma, \tau) \qquad \text{for all} \quad (\sigma, \tau) \neq (\mu, \nu).$$

It follows that $\texttt{cost}(\texttt{flow}_{r,s'}) = \texttt{cost}(\texttt{flow}_{r,s}) - c$ and $\#(r, s') = \#(r, s) - 1$.

By induction, there exists a matching M' for r, s' such that $\texttt{cost}(M') = \texttt{cost}(\texttt{flow}_{r,s'})$. Moreover, M' is integer-valued if s' is whole. We define a new matching M which is like M' except that M assigns a total of c units of requests to ν which M' assigned instead to μ. Let $\alpha(\sigma)$ denote the amount of requests from σ that are to be reassigned. Since $\sum_\sigma M'(\sigma, \mu) = s'(\mu) \geqslant c$, α can be chosen so that $\texttt{total}(T, \alpha) = c$, $\alpha(\sigma) \leqslant M'(\sigma, \mu)$, and α is integer-valued if M' and c are. Now define M:

$$M(\sigma, \mu) = M'(\sigma, \mu) - \alpha(\sigma),$$
$$M(\sigma, \nu) = M'(\sigma, \nu) + \alpha(\sigma),$$
$$M(\sigma, \tau) = M'(\sigma, \tau) \qquad \text{if} \quad \tau \neq \{\mu, \nu\}.$$

It is easily verified that $\sum_\tau M(\sigma, \tau) = r(\sigma)$ and $\sum_\sigma M(\sigma, \tau) = s(\tau)$ for all τ and that M is integer-valued if s is whole. Thus, M is a matching for r, s. Moreover, since only c resources were reassigned, $\texttt{cost}(M) \leqslant \texttt{cost}(M') + c = \texttt{cost}(\texttt{flow}_{r,s})$. It follows from Theorem 3.3.2 that $\texttt{cost}(M) = \texttt{cost}(\texttt{flow}_{r,s})$ as desired. ∎

It follows from these theorems that the cost of the optimal matching is the same as the cost of a particular flow, $\texttt{flow}_{r,s}$.

3.4. *Expected Flow*

From now on in the paper, we assume that T is rooted. Let ϕ be a probability density function on V and recall that $\phi(\texttt{top}) = 0$. A random $r: V \to \Re^+$ can be chosen as follows: for each i in turn, $1 \leqslant i \leqslant t$, ϕ is used to select a vertex (other than top). Then for each $v \in V$, $r(v)$ is the total number of times v is selected.

For edge e and placement s, let

$$\texttt{edgecost}_{e,\phi}(s) = \text{expected value of } |\texttt{flow}_{r,s}(e)|,$$

where r is chosen randomly according to ϕ as just described. This gives the amount of flow across the edge e "averaged" over request sets. Note that the edgecost measure is independent of edge orientation.

In the course of our development, we will be interested in the total edgecost on the subtrees $\hat{B}(e)$ of T consisting of the edges "below" and including e, so we define

$$\texttt{treecost}_{e,\phi}(s) = \sum_{d \in \hat{B}(e)} \texttt{edgecost}_{d,\phi}(s).$$

We are of course also interested in the total edgecost over the whole of T. Since $T = \hat{B}(\texttt{rootedge})$, we define our whole tree measure,

$$\texttt{cost}_{\phi}(s) = \texttt{treecost}_{\texttt{rootedge},\phi}(s).$$

Our central problem is to minimize $\texttt{cost}_{\phi}(s)$ over all placements s with $\texttt{total}(T, s) = t$. To do so, we will be looking at some constrained minimization problems, namely, restricting the placements over which we are minimizing to those which place a fixed number $u \in [0, t]$ of resources in the subtree $B(e)$. Thus, we extend the definition of edgecost to the domain $[0, t]$ by defining

$$\texttt{edgecost}_{e,\phi}(u) = \min\{\texttt{edgecost}_{e,\phi}(s) \mid \texttt{total}(B(e), s) = u \text{ and } \texttt{total}(T, s) = t\}.$$

It follows from Eq. (7) and the definition of edgecost that the function $\texttt{edgecost}_{e,\phi}(s)$ is a continuous function of s, where s can be regarded as a vector in a vector space with dimension $|V|$. Since the minimum is taken over a closed and bounded set, this minimum exists.

Intuitively, $\texttt{edgecost}_{e,\phi}(u)$ is the smallest average flow across e that can be achieved by any placement that puts u resources on the subtree $B(e)$ and $(t-u)$ resources on the subtree $A(e)$.

In a similar way, we extend the definitions of treecost and cost to the domain $[0, t]$, as follows:

$$\texttt{treecost}_{e,\phi}(u) = \min\{\texttt{treecost}_{e,\phi}(s) \mid \texttt{total}(B(e), s) = u \text{ and } \texttt{total}(T, s) = t\},$$

and

$$\texttt{cost}_{\phi}(u) = \texttt{treecost}_{\texttt{rootedge},\phi}(u).$$

Since $\texttt{treecost}_{e,\phi}$ is a continuous function of s, and the minimum is taken over a closed and bounded set, this minimum exists.

Removing the constraint on the number of resources to be placed in $B(e)$, we define global quantities:

$$\texttt{minedgecost}_{e,\phi} = \min_{u \in [0,t]} \texttt{edgecost}_{e,\phi}(u).$$

$$\texttt{mintreecost}_{e,\phi} = \min_{u \in [0,t]} \texttt{treecost}_{e,\phi}(u).$$

$$\texttt{mincost}_{\phi} = \texttt{mintreecost}_{\texttt{rootedge},\phi}.$$

Use of the minimum operator is justified as before. Note that $\texttt{mincost}_{\phi}$ is independent of the particular choice for rootedge.

We say a placement s is *optimal* for ϕ and t if $\texttt{total}(T, s) = t$ and

$$\texttt{cost}_{\phi}(s) = \texttt{mincost}_{\phi}.$$

It is easy to see that such a placement exists, for any ϕ and t.

An *optimal whole placement* for ϕ and t is a placement s that is both optimal and whole. An optimal whole placement obviously has minimal cost among all whole placements, but it is not *a priori* obvious that a minimal cost whole placement is always optimal. We show in Section 4 that optimal whole placements always exist.

4. Optimal Placements

In this section, we develop some useful and interesting properties of optimal placements. First, in Section 4.1, we ask whether the total cost can be minimized by minimizing the flow on each edge individually. A simply stated and elegant result is that the number of resources which minimizes the flow on an edge e, when placed anywhere in the subtree below e, is either $\lfloor tp \rfloor$ or $\lceil tp \rceil$, where p is the probability of a request being in the subtree. (The actual placement of the resources in this subtree clearly has no

effect on the flow on the edge.) However, Example 4.1.8 shows that it is not always possible to minimize the flow on all edges simultaneously.

Next, in Section 4.2, we show that whenever the probability of at least one request arriving in a subtree is less than $\frac{1}{2}$, it is never advantageous to place any resources at all in the subtree. This result is used in Section 6 to improve the efficiency of an algorithm for finding optimal placements by allowing the tree to be pruned.

In Section 4.3, we derive properties of the function $\texttt{treecost}_{e,\phi}(u)$, the least total expected flow in the subtree $\hat{B}(e)$ for any placement of $u \in [0, t]$ resources on the vertices of $B(e)$. This leads to a recursive expression for the treecost as a function of u and permits us to show that the treecost can always be achieved by a placement that puts an integral number of resources on each node. In other words, the ability to split resources and place fractional amounts on nodes is of no help, as long as the requests come in integral units. This same recursive decomposition also leads to an efficient algorithm for finding an optimal placement.

Finally, in Section 4.4, we show how to convert an optimal placement which may put fractional numbers of resources on nodes into another optimal placement which puts whole resources on each node and which is "close" to the original fractional placement. Also, the whole placement can be found quickly.

4.1. *Optimizing Flow on Individual Edges*

In this section, we show that the function $\texttt{edgecost}_e(u)$, which expresses the expected flow on an edge e in terms of the number u of resources placed in the subtree below e, is convex, piecewise linear, and has all singularities at integers. Moreover, $\texttt{edgecost}_e(u)$ is minimized by taking u equal to the median of a certain binomial density function.

We begin by giving an explicit characterization of the edgecost function.

THEOREM 4.1.1. *Let* $p = \texttt{total}(B(e), \phi)$, $u \in [0, t]$. *Then*

$$\texttt{edgecost}_e(u) = \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |u-i|.$$

Proof. In the ith term of the series, $|u-i|$ is the absolute flow across e that results when exactly i requests land in $B(e)$, and $\binom{t}{i} p^i (1-p)^{t-i}$ is the probability of that event occuring. ∎

THEOREM 4.1.2. *Let* ϕ *be a probability density function,* $t \in \mathbf{N} - \{0\}$, *and* $e \in E$. *Then* $\texttt{edgecost}_e \in F_t$.

Proof. By Theorem 4.1.1, $\texttt{edgecost}_e(u)$ is the sum of functions $g_i(u) = c_i|u-i|$, $i \in \mathbf{N}$, $0 \leqslant i \leqslant t$, where c_i does not depend on u. Each g_i,

restricted to the interval $[0, t]$, is a convex, piecewise linear function from $[0, t]$ to $\mathfrak{R}^+$ with a singularity at $u = i$. The result follows because addition preserves all required properties. ∎

LEMMA 4.1.3. *Let* $u, u' \in [0, t]$. *Then* $|\texttt{edgecost}_e(u) - \texttt{edgecost}_e(u')| \leqslant |u - u'|$.

Proof. It suffices to show the result for $u' = u + 1$, so assume that $u' = u + 1$. Let $p = \texttt{total}(B(e), \phi)$. Then by Theorem 4.1.1,

$$\begin{aligned}
&\texttt{edgecost}_e(u') - \texttt{edgecost}_e(u) \\
&\quad = \sum_{i=0}^{t} \binom{t}{i} p^i (1\text{-}p)^{t-i} |u + 1 - i| - \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |u - i| \\
&\quad = \sum_{i=0}^{u} \binom{t}{i} p^i (1-p)^{t-i} - \sum_{i=u+1}^{t} \binom{t}{i} p^i (1-p)^{t-i} \\
&\quad = 2 \sum_{i=0}^{u} \binom{t}{i} p^i (1-p)^{t-i} - 1.
\end{aligned}$$

Since

$$-1 \leqslant 2 \sum_{i=0}^{u} \binom{t}{i} p^i (1-p)^{t-i} - 1 \leqslant 2 \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} - 1 = 1,$$

we have $|\texttt{edgecost}_e(u) - \texttt{edgecost}_e(u')| \leqslant 1$, as needed. ∎

Let $t \in \mathbf{N} - \{0\}$, $0 \leqslant p \leqslant 1$. Let x be a random variable whose value is the number of successes in t independent trials, each of whose probability of success is p. That is, the value of x is given by the binomial density function with parameters t and p. Define $\texttt{median}(t, p)$ as the smallest $c \in \mathbf{N}$ such that $\Pr[x \leqslant c] \geqslant \frac{1}{2}$.

LEMMA 4.1.4. *Let* ϕ *be a probability density function on* V, *let* $t \in \mathbf{N} - \{0\}$, *and let* $e \in E$. *Then* $\texttt{edgecost}_e(u)$ *is minimized at* $u = \texttt{median}(t, p)$, *where* $p = \texttt{total}(B(e), \phi)$. *Moreover, if* $0 \leqslant v < \texttt{median}(t, p)$ *then* $\texttt{edgecost}_e(v) > \texttt{edgecost}_e(\texttt{median}(t, p))$.

Proof. Write f for $\texttt{edgecost}_e$, and let $n \in \mathbf{N}$, $0 \leqslant n \leqslant t - 1$. By the calculations in the proof of Lemma 4.1.3,

$$\begin{aligned}
f(n+1) - f(n) &= 2 \sum_{i=0}^{n} \binom{t}{i} p^i (1-p)^{t-i} - 1 \\
&= 2\Pr[x \leqslant n] - 1.
\end{aligned}$$

As a consequence of Theorem 4.1.2, f is minimized at an integer u which is the smallest integer $n \in [0, t-1]$ with $f(n+1)-f(n)$ nonnegative, if such an n exists, or at $u=t$ otherwise. In the former case, u is the smallest integer $n \in [0, t-1]$ with $2\Pr[x \leqslant n]-1$ nonnegative, or $\Pr[x \leqslant n] \geqslant \frac{1}{2}$; that is, $u = \texttt{median}(t, p)$. If no such n exists, then $\Pr[x \leqslant n] < \frac{1}{2}$ for all $n \leqslant t-1$. But since $\Pr[x \leqslant t] = 1$ we have $u = t = \texttt{median}(t, p)$. In either case, $u = \texttt{median}(t, p)$ is the smallest integer minimizing f. Since f is piecewise linear, it follows that $f(v) > f(u)$ if $v \in \Re^+$ and $0 \leqslant v < u$. ∎

Two results of Jogdeo and Samuels (1968, Theorem 3.2 and Corollary 3.1) give the following theorem. It is also implicit in some earlier work by Uhlmann (1963, 1966).

THEOREM 4.1.5. *Let* $t \in \mathbf{N}$, $t \geqslant 1$, *and* $0 \leqslant p \leqslant 1$. *Then*

$$\texttt{median}(t, p) \in \{\lfloor tp \rfloor, \lceil tp \rceil\}.$$

Since tp is the mean number of successes in t independent trials with success probability p, this theorem implies that for a binomial distribution, the mean and median differ by less than one.

Lemma 4.1.4 shows that placing $\texttt{median}(t, p)$ resources in $B(e)$ minimizes the expected flow on e, and Theorem 4.1.5 shows that the median lies close to the mean. We now show that *any* minimizing u lies close to the mean.

THEOREM 4.1.6. *Let* ϕ *be a probability density function on* V, *let* $t \in \mathbf{N} - \{0\}$, *let* $e \in E$, *and let* $p = \texttt{total}(B(e), \phi)$. *Let* u *be any value in* $[0, t]$ *which minimizes* $\texttt{edgecost}_e(u)$. *Then*

$$\lfloor tp \rfloor \leqslant u \leqslant \lceil tp \rceil.$$

Proof. By Lemma 4.1.4 and Theorem 4.1.5, $u \geqslant \texttt{median}(t, p) \geqslant \lfloor tp \rfloor$. Now let $q = \texttt{total}(A(e), \phi) = 1-p$ be the probability of a request in $A(e)$, and let $v = t-u$ be the number of resources placed on $A(e)$ when u resources are placed on $B(e)$. Let $\texttt{edgecost}'_e(v)$ be the expected absolute value of the flow on e when v resources are placed on $A(e)$. Then $\texttt{edgecost}'_e(v) = \texttt{edgecost}_e(u)$, so v is a value which minimizes $\texttt{edgecost}'_e$. By interchanging the roles of A and B, we can apply Lemma 4.1.4 and Theorem 4.1.5 again to conclude that $v \geqslant \texttt{median}(t, q) \geqslant \lfloor tq \rfloor$. Hence,

$$u = t - v \leqslant t - \lfloor tq \rfloor.$$

Since $t - \lfloor tq \rfloor = t - \lfloor t(1-p) \rfloor = \lceil tp \rceil$, we conclude that $u \leqslant \lceil tp \rceil$. ∎

A natural question is whether it is always possible to place resources in such a way as to simultaneously minimize the flow on all edges, and thereby obtain an optimal placement. The following examples show that this is sometimes, but not always, possible.

Example 4.1.7 shows that it is sometimes possible to place resources in a way that simultaneously minimizes the flow on all edges.

EXAMPLE 4.1.7. Let T be a complete binary tree having 7 vertices (excluding `top`) with maximum level 2. Let $\phi(v) = \frac{1}{7}$ for each vertex $v \in V$ except for `top`. Let $i \in \mathbf{N} - \{0\}$, $t = 7i$ and let s be the placement which puts i resources on each vertex. Then s simultaneously minimizes the edgecost over each edge, by Theorem 4.1.6, and so is an optimal placement.

Example 4.1.8 shows that it is not always possible to place resources in a way that simultaneously minimizes the flow on all edges.

EXAMPLE 4.1.8. This example gives a tree and request probabilities for which no placement minimizes the expected flow on all edges simultaneously. Let T be the 32-leaf complete binary tree shown in Fig. 4, $t = 16$, $\phi(v) = \frac{15}{256}$ for each of the leftmost 16 leaves, $\phi(v) = \frac{1}{256}$ for each of the rightmost 16 leaves, and $\phi(v) = 0$ for all other vertices.

By Theorem 4.1.6, the edgecost on edge e is minimized by placing 15 resources in the subtree below e. The same theorem implies that edgecost on an edge adjacent to a leaf is minimized by placing either 0 or 1 resource on the leaf; computation using the formula of Theorem [4.1.1] gives an edgecost greater than 0.9 for a placement of 0 resource and less than 0.7 for a placement of 1 resource for each leaf in the subtree below e. Thus, placing

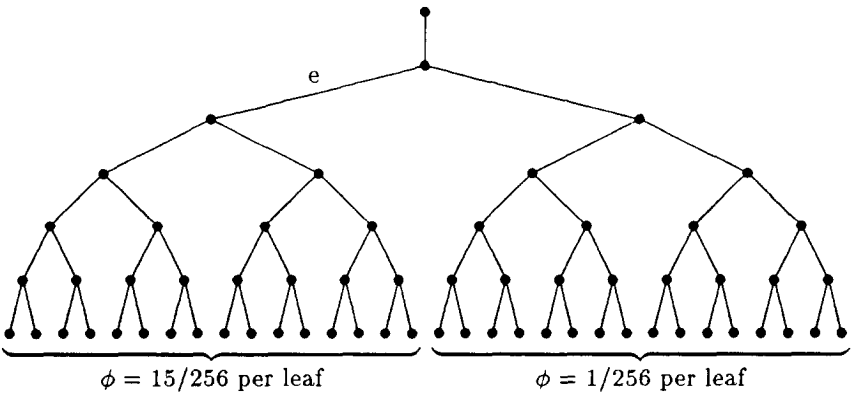

FIG. 4. A tree for which no placement minimizes the expected flow on all edges simultaneously.

1 resource on the leaf minimizes the cost on the adjacent edge. But then to minimize the cost on every leaf edge we would have to place all 16 resources in the subtree below e, thus increasing the cost on e.

4.2. *Subtrees with Zero Resources*

In this section, we show that if the probability is less than $\frac{1}{2}$ of at least one request arriving in a subtree, then no optimal placement puts even a small fraction of a resource anywhere in that subtree. It follows (cf. Section 6.1) that in a complete d-ary tree with a symmetric probability density function, no optimal placement puts any resource below level $\lceil \log_d t \rceil$.

THEOREM 4.2.1. *Fix probability density function ϕ and fix $t \in \mathbf{N} - \{0\}$. Let $e \in E$ satisfy $(1-p)^t > \frac{1}{2}$, where $p = \texttt{total}(B(e), \phi)$. Let s be an optimal placement for ϕ, t. Then $s(v) = 0$ for all $v \in B(e)$.*

Proof. Assume not, and fix e, s exhibiting the contrary. Choose e' below e to be a lowest edge (i.e., closest to the leaves) for which $x = s(\texttt{low}(e')) > 0$. Define a new placement s', where $s'(\texttt{low}(e')) = 0$, $s'(\texttt{high}(e')) = s(\texttt{high}(e')) + x$, and $s'(v) = s(v)$ otherwise. We show that $\texttt{cost}(s') < \texttt{cost}(s)$, contradicting the optimality of s.

From the definition of $\texttt{edgecost}_{e'}$, e' is the only edge for which s and s' have different expected absolute flows, so $\texttt{cost}(s) - \texttt{cost}(s') = \texttt{edgecost}_{e'}(x) - \texttt{edgecost}_{e'}(0)$. Let $p_{e'} = \texttt{total}(B(e'), \phi)$. Applying Theorem 4.1.1 and using the fact that $|x-i| - i \geqslant -x$, we obtain the difference

$$\begin{aligned}
&\sum_{i=0}^{t} \binom{t}{i} p_{e'}^i (1-p_{e'})^{t-i} (|x-i| - i) \\
&\quad \geqslant \binom{t}{0} p_{e'}^0 (1-p_{e'})^{t-0} \cdot x - \sum_{i=1}^{t} \binom{t}{i} p_{e'}^i (1-p_{e'})^{t-i} \cdot x \\
&\quad = x \cdot (2(1-p_{e'})^t - 1).
\end{aligned}$$

Since $(1-p_{e'})^t \geqslant (1-p)^t > \frac{1}{2}$, the difference is positive, giving the needed contradiction. ∎

4.3. *Optimal Placements in Subtrees*

We now characterize the function $\texttt{treecost}_e(u)$, which represents the least total expected flow in the subtree $\hat{B}(e)$ for any placement of $u \in [0, t]$ resources on the vertices of $B(e)$. Like $\texttt{edgecost}_e$, $\texttt{treecost}_e$ is convex, piecewise linear, and has all its singularities at integers. However, the points at which it achieves its minimum appear to be harder to characterize.

THEOREM 4.3.1. *Let ϕ be a probability density function, let $t \in \mathbf{N} - \{0\}$, and let $e \in E$. Let $e_1, \ldots, e_d$ be the immediate descendant edges of* $\mathtt{low}(e)$, *and let $u \in [0, t]$. Then*

$$\mathtt{treecost}_e(u) = \mathtt{edgecost}_e(u) + \min_{\mathbf{u}: \sum \mathbf{u} \leq u} \sum_{i=1}^{d} \mathtt{treecost}_{e_i}(u_i). \tag{8}$$

Proof. Recall that for any placement s, $\mathtt{treecost}_e(s)$ is the total expected flow in the subtree $\hat{B}(e)$, and $\mathtt{treecost}_e(u)$ is the least such cost, subject to the constraint that exactly u resources be placed in $B(e)$. It follows immediately from the definitions that for any placement s,

$$\mathtt{treecost}_e(s) = \mathtt{edgecost}_e(s) + \sum_{i=1}^{d} \mathtt{treecost}_{e_i}(s),$$

where $e_1, \ldots, e_d$ are the immediate descendants of edge e. Since $\mathtt{edgecost}_e(s)$ depends only on $\mathtt{total}(B(e), s)$, the theorem then follows by minimizing over all placements with $\mathtt{total}(B(e), s) = u$ and $\mathtt{total}(T, s) = t$. ∎

Theorem 4.1.2 shows that $\mathtt{edgecost}_e \in F_t$. We now show that $\mathtt{treecost}_e \in F_t$, and we relate the shape of $\mathtt{treecost}_e$ to that of the component functions $\mathtt{edgecost}_e$ and $\mathtt{treecost}_{e_i}$ appearing in Eq. 8. Lemma 2.3.4 provides the induction step for the proof that $\mathtt{treecost}_e$ has the same simple form as $\mathtt{edgecost}_e$.

THEOREM 4.3.2. *For any fixed probability density function ϕ, any $t \in \mathbf{N} - \{0\}$, and any $e \in E$,* $\mathtt{treecost}_e \in F_t$.

Proof. We use induction on edges in the tree, working from the leaves towards the root.

If e is a lowest edge, then it is the only edge in $\hat{B}(e)$. Thus, $\mathtt{treecost}_e = \mathtt{edgecost}_e \in F_t$ by Theorem 4.1.2.

Now assume the result holds for edges below e, and let $e_1, \ldots, e_d$ denote the immediate descendant edges of e. Consider the expression for $\mathtt{treecost}_e(u)$ given in Theorem 4.3.1. The first term is in F_t by Theorem 4.1.2, the second term is in F_t by Lemmas 2.3.2 and 2.3.4, and F_t is closed under addition. ∎

The following is immediate from Theorems 4.3.1 and 4.3.2.

COROLLARY. *Let ϕ, e, etc., be as in Theorem* 4.3.1. *Then*

$$\mathtt{treecost}_e = \mathtt{edgecost}_e + (\mathtt{treecost}_{e_1} * \mathtt{treecost}_{e_2} * \cdots * \mathtt{treecost}_{e_d})_{\leq}$$

restricted to the domain $[0, t]$.

Lemma 2.2.8 and Theorem 4.3.1 lead immediately to a dynamic programming algorithm for computing $\texttt{mincost}_\phi$ and for finding an optimal placement given a (rooted) tree T and probability density function ϕ. Moreover, the placement found consists of whole resources only. Namely, one computes and stores $\texttt{treecost}_e(u)$ for each edge e and each integer $u \in \{0, 1, ..., t\}$, working from the leaves of T towards the root. We will show that each such value can be computed in amortized time $O(1)$; hence the total time is $O(mt)$, where m is the number of edges in T.

The algorithm works in four phases. In the first phase, it computes $p_e = \texttt{total}(B(e), \phi)$ for all edges e, working up the tree from the leaves. This takes time $O(m)$.[5]

In the second phase, it computes $\texttt{edgecost}_e(u)$ for each edge e and each $u \in \{0, 1, ..., t\}$. Given e, it first computes

$$\sum_{i=0}^{n} \binom{t}{i} p_e^i (1-p_e)^{t-i}$$

for $n = 0, 1, ..., t$. This takes time $O(t)$. Next it computes $\texttt{edgecost}_e(0)$ in time $O(t)$ using Theorem 4.1.1. Finally, it computes $\texttt{edgecost}_e(n)$ for $n = 1, 2, ..., t$ using the formula

$$\texttt{edgecost}_e(n) = \texttt{edgecost}_e(n-1) + 2\sum_{i=0}^{n-1} \binom{t}{i} p_e^i (1-p_e)^{t-i} - 1$$

and the values previously stored (cf. the proof of Lemma 4.1.4). This takes time $O(1)$ for each value of n for a total of $O(t)$. Thus, the total time spent working on edge e is $O(t)$, or $O(mt)$ for the whole tree.

In the third phase, it computes $\texttt{treecost}_e(u)$ for each edge e and each $u \in \{0, 1, ..., t\}$. Again it works up from the leaves. Let $e_1, ..., e_d$ be the immediate descendant edges of e, and assume $\texttt{treecost}_{e_i}(u')$ has already been computed for each $1 \leqslant i \leqslant d$ and each $u' \in \{0, ..., t\}$. It now computes integer vectors $\mathbf{u}^{(n)}$ for each $n = 0, 1, ..., t$ such that $\sum \mathbf{u}^{(n)} \leqslant n$ and

$$\texttt{treecost}_e(n) = \texttt{edgecost}_e(n) + \sum_{i=1}^{d} \texttt{treecost}_{e_i}(u_i^{(n)}).$$

By Lemma 2.3.10, these vectors can be constructed so that $\mathbf{u}^{(0)} = (0, 0, ..., 0)$, and $\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \boldsymbol{\delta}^{(n+1)}$, where $\boldsymbol{\delta}^{(n+1)}$ either is $(0, 0, ..., 0)$ or a unit vector. The vector chosen for $\mathbf{u}^{(n+1)}$ among these possibilities is the one which minimizes the value of

$$\sum_{i=1}^{d} \texttt{treecost}_{e_i}(u_i^{(n)}).$$

[5] By "time" we mean the unit-cost time measure for a RAM (Aho *et al.*, 1974).

Thus, to compute $\mathbf{u}^{(n+1)}$ from $\mathbf{u}^{(n)}$, one finds the k which yields the greatest reduction in $\texttt{treecost}_{e_k}(u_k^{(n)})$, that is, the k which minimizes the difference

$$\texttt{treecost}_{e_k}(u_k^{(n)}+1)-\texttt{treecost}_{e_k}(u_k^{(n)}).$$

If the difference for that k is $\leqslant 0$, then take $\boldsymbol{\delta}^{(n+1)}=\mathbf{I}_k=(0, ..., 0, 1, 0, ..., 0)$, the unit vector with a 1 in the k^{th} component and 0's elsewhere. Otherwise, take $\boldsymbol{\delta}^{(n+1)}=(0, 0, ..., 0)$. In this way, $\mathbf{u}^{(n+1)}$ can be found from $\mathbf{u}(n)$ in $O(d)$ steps, so the time to compute $\texttt{treecost}_e(n)$ for each value of n is $O(dt)$.

To total up the time used in the third phase, let d_e be the number of immediate descendant edges of e. Summing over all edges, the total time is $O(\sum_e d_e t)$. However, $\sum_e d_e = m-1$ since each edge except for `rootedge` is counted exactly once. Hence, the total time for the third phase, as well as the time for phases 1–3 combined, is again $O(mt)$.

We have not yet actually produced a placement. A fourth phase can be used to find an optimal whole placement s. In this case, the algorithm works from `rootedge` down towards the leaves. Let u be a number of resources that minimizes $\texttt{treecost}_{\texttt{rootedge}}(u)$, and let $s(\texttt{top})=t-u$. Then the algorithm has u resources to place on $B(\texttt{rootedge})$, and inductively, it has $n\in\mathbf{N}$ resources to place on $B(e)$. If $\texttt{low}(e)$ is a leaf, then $s(\texttt{low}(e))=n$. Otherwise, let $\mathbf{u}^{(n)}$ be the vector computed above which minimizes $\sum_i \texttt{treecost}_{e_i}(u_i^{(n)})$. Let $s(\texttt{low}(e))=n-\sum \mathbf{u}^{(n)}$, and recursively place $u_i^{(n)}$ resources on each $B(e_i)$, where as usual $e_1, ..., e_d$ are the immediate descendant edges of e. This takes an additional time of only $O(\sum_e d_e)=O(m)$.

The above proves the following:

THEOREM 4.3.3. *Given a tree T, a probability density function ϕ, and a number $t\in\mathbf{N}-\{0\}$, there is an algorithm for computing* $\texttt{mincost}_\phi$ *and for finding an optimal placement s of t resources on T in time $O(mt)$, where m is the number of edges in T. The placement thereby produced is a whole placement.*

Example 4.3.4 shows the edgecost and treecost for each edge in a tree.

EXAMPLE 4.3.4. Let T be a complete binary tree having 15 vertices (excluding `top`) with maximum level 3. The probability density function ϕ has value $\frac{3}{16}$ for each of the leftmost four leaves, and value $\frac{1}{16}$ for each of the rightmost four leaves. We assume $t=4$. The tree is shown in Fig. 5.

Let e_1 and e_2 denote the two edges hanging off the root. By symmetry, for each $i\in\{1, 2\}$, all of the edges at the same level in tree $B(e_i)$ have the same cost.

Table 1 gives values of $\texttt{edgecost}_e(u)$, for all edges of the tree and for all values of u. Nonroot edges e are identified by the notation $e_{i_1, i_2, ...,}$

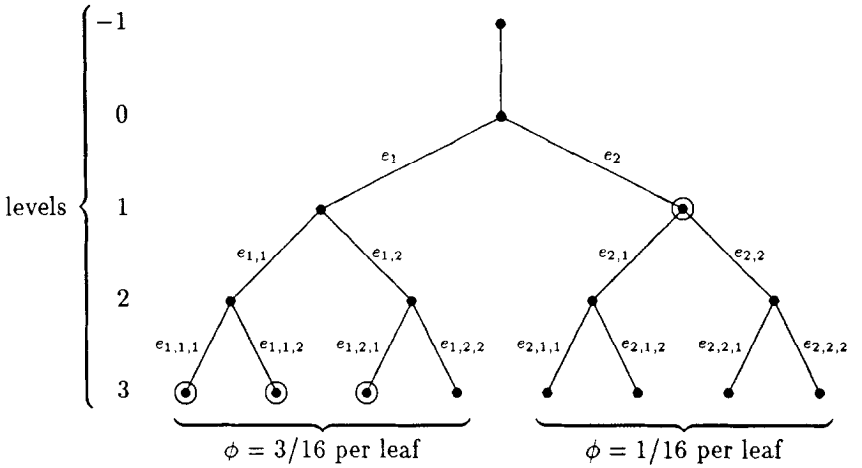

FIG. 5. The tree of Example 4.3.4.

where $i_1, i_2, \ldots$ is the path from the root to $\texttt{low}(e)$. A star $(*)$ in the path indicates that the same table line applies to both the left and the right branches.

Table 2 gives the treecosts. Note, for example, how the entry for edge e_1 and $u = 3$ is calculated. It is obtained by adding the corresponding edgecost, which is 0.632812, to the minimum sum of treecosts for the edge $e_{1,1}$ with u_1 resources and $e_{1,2}$ with u_2 resources, where the sum $u_1 + u_2$ is no more than 3. In this case, one can take $u_1 = 1$ and $u_2 = 2$, yielding additional costs of 2.176788 and 2.085999, respectively. The total treecost is then $0.632812 + 2.176788 + 2.085999 = 4.895599$.

The optimal placement is obtained as in the algorithm. In this case, we obtain a placement as shown by the circles in Fig. 5.

TABLE I

Table of Edgecosts.

	Values of u				
Edge e	0	1	2	3	4
rootedge	4.000000	3.000000	2.000000	1.000000	0.000000
e_1	3.000000	2.007812	1.109375	0.632812	1.000000
e_2	1.000000	0.632812	1.109375	2.007812	3.000000
$e_{1,*}$	1.500000	0.805176	0.842773	1.539551	2.500000
$e_{2,*}$	0.500000	0.672363	1.514648	2.500488	3.500000
$e_{1,*,*}$	0.750000	0.621613	1.297791	2.252472	3.250000
$e_{2,*,*}$	0.250000	0.794952	1.751892	2.750031	3.750000

TABLE II

Table of Treecosts.

Edge e	Values of u				
	0	1	2	3	4
rootedge	16.000000	13.184601	10.462952	8.895599	7.528412
e_1	9.000000	7.184601	5.462952	4.895599	5.171997
e_2	3.000000	2.632812	3.109375	4.007812	5.000000
$e_{1,*}$	3.000000	2.176788	2.085999	2.782776	3.743225
$e_{2,*}$	1.000000	1.172363	2.014648	3.000488	4.000000
$e_{1,*,*}$	0.750000	0.621613	1.297791	2.252472	3.250000
$e_{2,*,*}$	0.250000	0.794952	1.751892	2.750031	3.750000

4.4. *Whole Placements*

Theorem 4.3.3 shows that for any tree T and probability density function ϕ on request arrivals, there is an optimal whole placement. Thus, the ability to reduce granularity of resources by splitting into fractional sized pieces does not result in superior placements.

We now show the more general result that *any* optimal placement s of t resources can be converted to an optimal *whole* placement s', and this can be done in a way so that

- the total amount placed by s' on each subtree $B(e)$ is the floor or ceiling of the total amount placed by s, and
- the amount placed by s' on each node is the floor or ceiling of the amount placed by s.

(Note that neither of these properties implies the other.) Moreover, s' can be found quickly from s. This result will let us prove theorems for general placements and then convert them to whole placements while preserving the properties of interest.

We begin by showing that any placement s can be "rounded" to yield a whole placement s' satisfying the above two properties. Then we show that if s is optimal, then *any* placement s' which is sufficiently close to s is also optimal. The placement obtained from s by rounding is sufficiently close in the sense which we now make precise.

Let s, s' be placements on T, and let $e \in E$. We say s' is an *e-neighbor* of s, written $s \rhd_e s'$, if the following two conditions hold.

1. For every $e' \in \hat{B}(e)$,

$$\lfloor \mathtt{total}(B(e'), s) \rfloor \leqslant \mathtt{total}(B(e'), s') \leqslant \lceil \mathtt{total}(B(e'), s) \rceil.$$

2. For every v in $B(e)$, $\lfloor s(v) \rfloor \leqslant s'(v) \leqslant \lceil s(v) \rceil$.

We say s' is a *neighbor* of s, written $s \rhd s'$, if $s \rhd_{\texttt{rootedge}} s'$. Note that these relations are transitive, but need not be symmetric.

LEMMA 4.4.1. *Let s, s' be placements on T. If $s \rhd s'$, then*

$$\lfloor s(\texttt{top}) \rfloor \leqslant s'(\texttt{top}) \leqslant \lceil s(\texttt{top}) \rceil.$$

Proof. We have

$$s(\texttt{top}) = t - \texttt{total}(B(\texttt{rootedge}), s)$$

and

$$s'(\texttt{top}) = t - \texttt{total}(B(\texttt{rootedge}), s').$$

Since

$$\lfloor \texttt{total}(B(\texttt{rootedge}), s) \rfloor \leqslant \texttt{total}(B(\texttt{rootedge}), s') \leqslant \lceil \texttt{total}(B(\texttt{rootedge}), s) \rceil$$

and t is an integer, it follows that

$$\lfloor t - \texttt{total}(B(\texttt{rootedge}), s) \rfloor \leqslant t - \texttt{total}(B(\texttt{rootedge}), s') \leqslant \lceil t - \texttt{total}(B(\texttt{rootedge}), s) \rceil,$$

i.e., $\lfloor s(\texttt{top}) \rfloor \leqslant s'(\texttt{top}) \leqslant \lceil s(\texttt{top}) \rceil$. ∎

For any $u \in \mathfrak{R}^+$ and $b \in \{0, 1\}$, let $\texttt{round}_b(u) = \lfloor u \rfloor$ if $b = 0$, and $\texttt{round}_b(u) = \lceil u \rceil$ if $b = 1$.

LEMMA 4.4.2. *Let T be a tree and let s be a placement of $t \in \mathbf{N} - \{0\}$ resources on T. Let e be any edge of T, and let $u = \texttt{total}(B(e), s)$. Finally, let $b \in \{0, 1\}$. Then there exists a whole placement s' of t resources such that $s \rhd_e s'$ and $\texttt{total}(B(e), s') = \texttt{round}_b(u)$. Moreover, s' can be found from s in time $O(m_{\hat{B}(e)})$.*[6]

Proof. We use induction on the height of e, working from the leaves towards the root.

If e is a lowest edge, then it is the only edge in $\hat{B}(e)$. Let $v = \texttt{low}(e)$, and define $s'(v) = \texttt{round}_b(s(v))$.[7] The properties of the lemma trivially hold.

[6] Recall that $m_{\hat{B}(e)}$ is the number of edges in $\hat{B}(e)$.

[7] Technically speaking, we should define s' on all of T. However, the properties of interest of s' depend only on its values on the nodes of $B(e)$, so we leave the remainder of s' unspecified.

Now assume the result holds for edges below e. Let $e_1, \ldots, e_d$ denote the immediate descendant edges of e, and let $v = \texttt{low}(e)$. Let $u_0 = s(v)$ and let $u_i = \texttt{total}(B(e_i), s)$. With these definitions, $u = \sum_{i=0}^{d} u_i$.

We claim that there exist $b_0, \ldots, b_d \in \{0, 1\}$ such that

$$\sum_{i=0}^{d} \texttt{round}_{b_i}(u_i) = \texttt{round}_b(u). \tag{9}$$

This follows from the facts that

$$\sum_{i=0}^{d} \texttt{round}_0(u_i) \leqslant \texttt{round}_0(u) \leqslant \texttt{round}_1(u) \leqslant \sum_{i=0}^{d} \texttt{round}_1(u_i)$$

and $\texttt{round}_1(u_i) - \texttt{round}_0(u_i) = 1$ if $u_i \notin \mathbf{N}$, so complementing such a b_i changes the sum by ± 1. We may assume without loss of generality that $b_i = 0$ if $u_i \in \mathbf{N}$ since then $\texttt{round}_1(u_i) - \texttt{round}_0(u_i) = 0$.

Let $k = \texttt{round}_b(u)$, and let $k_i = \texttt{round}_{b_i}(u_i)$, $i = 0, 1, \ldots, d$. By induction, for $1 \leqslant i \leqslant d$, there exist placements s_i' satisfying the statement of the lemma for e_i and b_i, so in particular, $\texttt{total}(B(e_i), s_i') = k_i$. Let s' coincide with s_i' on $B(e_i)$, and let $s'(v) = k_0$. We have $s \rhd_{e_i} s'$ by construction and the induction hypothesis. Also

$$\texttt{total}(B(e), s') = k_0 + \sum_{i=1}^{d} k_i = \sum_{i=0}^{d} \texttt{round}_{b_i}(u_i) = \texttt{round}_b(u).$$

Thus, s' has the needed properties.

We now consider an algorithm to find s' from s. The algorithm proceeds in two phases. The first phase computes and stores $\texttt{total}(B(e'), s)$ for each edge e' in $\hat{B}(e)$. This takes time $O(m_{\hat{B}(e)})$ by working up the tree from the leaves.

The second phase works recursively starting at e and $\texttt{round}_b(u)$. The algorithm first finds $b_0, \ldots, b_d$ satisfying Eq. (9). It next sets

$$s'(v) = \texttt{round}_{b_0}(s(v)).$$

Finally, it applies itself recursively to each of the pairs e_i, b_i to complete the computation of s'. The total time is easily seen to be $O(m_{\hat{B}(e)})$ once we show how to compute the b's in time $O(d)$.

In fact, the b's can be computed quite simply. Assume for the moment that $b = 0$ and define $\psi_{-1} = u - \texttt{round}_b(u)$. Then $\psi_{-1} \in [0, 1)$. For $i = 0, \ldots, d$, consider $\psi' = \psi_{i-1} - u_i + \texttt{round}_{b'}(u_i)$ for each $b' \in \{0, 1\}$. Exactly one of the two values for b' puts ψ' in the range $[0, 1)$. Choose b_i to be that value, and let ψ_i be the corresponding value of ψ'. Then

$$\psi_d = u - \mathtt{round}_b(u) - \sum_{i=0}^{d} u_i + \sum_{i=0}^{d} \mathtt{round}_{b_i}(u_i)$$

$$= -\mathtt{round}_b(u) + \sum_{i=0}^{d} \mathtt{round}_{b_i}(u_i).$$

Hence, $\psi_d \in \mathsf{N}$, but since also $\psi_d \in [0, 1)$ we have $\psi_d = 0$. Thus, the b_i's so constructed satisfy Eq. (9). The case $b = 1$ is handled similarly. In either case, it is clear that the time required is only $O(d)$. ▮

LEMMA 4.4.3. *Let T be a tree and let s, s' be placements of $t \in \mathsf{N} - \{0\}$ resources on T. Assume that $s \rhd s'$. If s is an optimal placement of t resources on T, then s' is also an optimal placement.*

Proof. We first show by induction that for every edge e in T,

$$\mathtt{treecost}_e(s') = \mathtt{treecost}_e(\mathtt{total}(B(e), s')),$$

that is, s' is optimal on every subtree for the number of resources which it places there.

The base case is trivial since if e is a lowest edge, then there is only one way of placing $u' = \mathtt{total}(B(e), s')$ resources on $B(e)$.

Now assume the result holds for edges below e. Let $e_1, \ldots, e_d$ denote the immediate descendant edges of e, and let $v = \mathtt{low}(e)$. Let $u_i = \mathtt{total}(B(e_i), s)$ for $1 \leqslant i \leqslant d$, and let $u = \mathtt{total}(B(e), s)$. Similarly, let $u_i' = \mathtt{total}(B(e_i), s')$, and $u' = \mathtt{total}(B(e), s')$. Note that for any i, if $u_i \in \mathsf{N}$, then $u_i' = u_i$, and in any case, $\mathtt{round}_0(u_i) \leqslant u_i' \leqslant \mathtt{round}_1(u_i)$. Similarly, $\mathtt{round}_0(u - \sum \mathbf{u}) \leqslant u' - \sum \mathbf{u}' \leqslant \mathtt{round}_1(u - \sum \mathbf{u})$.

Let $f_i = \mathtt{treecost}_{e_i}$, $1 \leqslant i \leqslant d$, and let $f = (f_1 * f_2 * \cdots * f_d)$. $\mathbf{u} = (u_1, \ldots, u_d)$ is optimal for u and $(f_1, f_2, \ldots, f_d)$ since s is optimal. Thus, by Lemma 2.3.9, $\mathbf{u}'$ is optimal for u' and $(f_1, f_2, \ldots, f_d)$. This implies that

$$\sum f_i(u_i') = f_{\leqslant}(u').$$

By the induction hypothesis,

$$f_i(s') = f_i(u_i')$$

for $1 \leqslant i \leqslant d$. Hence,

$$\begin{aligned}
\mathtt{treecost}_e(s') &= \mathtt{edgecost}_e(s') + \sum f_i(s') \\
&= \mathtt{edgecost}_e(u') + \sum f_i(u_i') \\
&= \mathtt{edgecost}_e(u') + f_{\leqslant}(u') \\
&= \mathtt{treecost}_e(u') \quad \text{(by Theorem 4.3.1)} \\
&= \mathtt{treecost}_e(\mathtt{total}(B(e), s')).
\end{aligned}$$

The claim follows by induction.

Thus, in particular, we see that

$$\begin{aligned}\texttt{cost}(s') &= \texttt{treecost}_{\texttt{rootedge}}(s') \\ &= \texttt{treecost}_{\texttt{rootedge}}(\texttt{total}(B(\texttt{rootedge}), s')).\end{aligned}$$

Now let $u = u' = t$,

$$u_1 = \texttt{total}(B(\texttt{rootedge}), s),$$

and

$$u'_1 = \texttt{total}(B(\texttt{rootedge}), s').$$

Let $d = 1$, $f_1 = \texttt{treecost}_{\texttt{rootedge}}$. Lemma 4.4.1 allows us to apply Lemma 2.3.9 once again, which yields that $\mathbf{u}'$ is optimal for u and (f_1). This implies that

$$f_1(u'_1) = \min_{v \leq t} \{f_1(v)\}.$$

But the right side of the preceding equation is just `mincost`. Putting these equations together yields $\texttt{cost}(s') = \texttt{mincost}$, as needed. ▮

The following main theorem says that if s is optimal, then there exists an optimal whole neighbor s' that can be found quickly from s. Thus, the best of the whole placements is also optimal among *all* placements, and the cost of finding it is little more than the cost of finding any optimal placement. In other words, restricting attention to whole placements imposes no penalty in terms of the quality of the placement obtained and little in terms of the time to find it.

THEOREM 4.4.4. *Let T be a tree and let s be an optimal placement of $t \in \mathbf{N} - \{0\}$ resources on T. Then there exists an optimal whole placement s' of t resources such that $s \rhd s'$. Moreover, s' can be computed from s in time $O(|E_T|)$.*

Proof. This is an immediate consequence of Lemmas 4.4.2 and 4.4.3. ▮

In general, even if a placement s is not optimal, Lemma 4.4.2 gives us a whole neighbor s', but now we do not know that s' is optimal. However, just from the individual edge bounds, we can give an upper bound on the cost of s' in terms of the cost of s.

THEOREM 4.4.5. *Let T be a tree and let s, s' be placements of $t \in \mathbf{N} - \{0\}$ resources on T. If $s \rhd s'$, then $\texttt{cost}(s') \leqslant cost(s) + |E_T|$.*

Proof. Fix any edge e, and let $u = \texttt{total}(B(e), s)$ and $u' = \textit{total}(B(e), s')$. Then $\lfloor u \rfloor \leqslant u' \leqslant \lceil u \rceil$, since $s \rhd s'$. So $|u - u'| \leqslant 1$. Lemma 4.1.3 implies that $|\texttt{edgecost}_e(u) - \texttt{edgecost}_e(u')| \leqslant 1$. Summing over all the edges yields the result. ∎

5. Fair Placements

A very natural placement is to put a number of resources on each node equal to the expected number of requests at that node. We call this the "exact fair" placement. Rather surprisingly, the exact fair placement is not always optimal, but we show that its cost is never more than $|E_T|$ greater than the cost of the optimal placement. We also show that its cost is only $O(\sqrt{t})$, where the constant implicit in the "big-oh" notation depends on the tree T. This provides an upper bound on the cost of the optimal placement as well.

5.1. *Fair Placements*

For any T, ϕ, t, and for each edge $e \in E$, let $p_e = \texttt{total}(B(e), \phi)$. Call a placement $s: V \to \mathfrak{R}^+$ *fair* if $\lfloor tp_e \rfloor \leqslant \texttt{total}(B(e), s) \leqslant \lceil tp_e \rceil$ for every edge e. If in fact $\texttt{total}(B(e), s) = tp_e$ for every edge e, then we say s is an *exact fair* placement.

For any T, ϕ, and t, an exact fair placement exists and is unique, namely, let $s(v) = t\phi(v)$ for $v \in V$. Also, s is easily computed in time $O(|E_T|)$. By Lemma 4.4.2, it follows that there is a fair whole placement s', and s' can also be found in time $O(|E_T|)$. Note that in the special case that ϕ is nonzero only on leaves, then s and s' are nonzero only on leaves.

The following theorem shows that optimality results for fair placements carry over to fair whole placements.

THEOREM 5.1.1. *Let T be an arbitrary tree, let ϕ be an arbitrary probability density function, and let $t \in \mathbf{N} - \{0\}$. If there is an optimal fair placement of t resources on T, then there is an optimal fair whole placement.*

Proof. The proof follows from Theorem 4.4.4. ∎

Any placement that simultaneously minimizes the flow on each edge is optimal, and by Theorem 4.1.6 it is also fair. However, the following example shows that there does not always exist a placement which is both optimal and fair, so it is also not always possible to place resources in a way that simultaneously minimizes the flow on all edges.

EXAMPLE 5.1.2. Consider the complete binary tree with maximum level 6 in which the probability of a request at `root` is 0.04, the probability of

a request at a leaf is 0.015, and the probability of a request elsewhere is 0. Consider the case where $t = 64$. Then no fair placement is optimal.

Fair placements are constrained to place the following numbers of resources in subtrees of level j:

j	Number of resources
0	64
1	30 or 31
2	15 or 16
3	7 or 8
4	3 or 4
5	1 or 2
6	0 or 1

Calculating edgecosts using Theorem 4.1.1 shows that the choices which give the smaller costs on the edges leaving the respective subtrees are 64, 31, 15, 8, 4, 2, and 1. It is not difficult to see that there are only two reasonable candidates for an optimal fair placement:

(a) Place 31 resources in each level 1 subtree and 15 resources in each level 2 subtree. At levels 3 and below, place the larger possible choice in as many subtrees as possible, and the smaller possible choice in the others.

(b) Place 31 resources in each level 1 subtree, 15 resources in two of the level 2 subtrees, and 16 resources in the other two level 2 subtrees. Again, at levels 3 and below, place the larger possible choice in as many subtrees as possible, and the smaller possible choice in the others.

However, the placement in which one resource is located at every leaf has a smaller cost than either fair placement (a) or (b).

The following theorem gives a special case in which fair placements *are* optimal.

THEOREM 5.1.3. *Let T be an arbitrary tree and let $t \in \mathbf{N} - \{0\}$. Suppose ϕ is such that $tp_e \in \mathbf{N}$ for all edges e. Then the exact fair placement s of t resources on T is optimal.*

Proof. Since $\texttt{total}(B(e), s) = tp_e \in \mathbf{N}$, s is a whole placement, and by Theorem 4.1.6 it simultaneously minimizes $\texttt{edgecost}_e$ for all edges e. Hence, s is optimal. ∎

Example 4.1.7 gives a special case of this theorem—a tree for which the minimum cost for each edge is in **N**. Another special case of this theorem is given by the following example.

EXAMPLE 5.1.4. Let T be a complete d-ary tree with d^h leaves. Let $t = id^h$ for $i \in \mathbf{N} - \{0\}$, and let ϕ be the uniform distribution on the leaves. The exact fair placement, which puts i resources on each leaf and 0 elsewhere, is optimal.

Another case for which there exist optimal fair placements (and hence, optimal fair whole placements) will be given later in Theorem 6.3.1. We now show that fair placements are never too far from optimal.

THEOREM 5.1.5. *Let s be any fair placement for an arbitrary tree T, an arbitrary probability density function ϕ, and arbitrary $t \in \mathbf{N} - \{0\}$. Then* $\texttt{mincost} \leqslant \texttt{cost}(s) \leqslant \texttt{mincost} + m$, *where m is the number of edges in T.*

Proof. By Theorem 4.1.6, $\texttt{edgecost}_e$ is minimized at some u', where $\lfloor tp_e \rfloor \leqslant u' \leqslant \lceil tp_e \rceil$ and $p_e = \texttt{total}(B(e), \phi)$. Let s be a fair placement, and let $u = \texttt{total}(B(e), s)$. Since s is fair, $\lfloor tp \rfloor \leqslant u \leqslant \lceil tp \rceil$.

Thus, $|u - u'| \leqslant 1$. But then Lemma 4.1.3 implies that $|\texttt{edgecost}_e(u) - \texttt{edgecost}_e(u')| \leqslant 1$. Thus, s incurs at most an extra cost of 1 per edge over the optimal placement. ∎

5.2. *Bounds on Fair Placements*

We now turn to the analysis of the exact fair placement on an arbitrary tree T. For each edge e of T, we define a constant β_e inductively. If e is a lowest edge, then $\beta_e = 1$. Otherwise, let $e_1, \ldots, e_d$ be the immediate descendant edges of e. Then

$$\beta_e = 1 + \sqrt{\sum_{i=1}^{d} \beta_{e_i}^2}.$$

As before, let $p_e = \texttt{total}(B(e), \phi)$.

THEOREM 5.2.1. *Let T be an arbitrary tree, let $t \in \mathbf{N} - \{0\}$, let ϕ be a probability density function, and let s be the exact fair placement for t and ϕ on T. Then*

$$\texttt{cost}(s) \leqslant c\beta_{\texttt{rootedge}} \sqrt{tp_{\texttt{rootedge}}}$$

for $c = \sqrt{2/\pi}$.

We will need several lemmas to establish this result. Lemmas 5.2.2 and 5.2.3 provide a rather surprising closed form for the summation given in Theorem 4.1.1 for $\texttt{edgecost}_e(tp)$. One would expect to see summations of this form in other contexts as well, so that these lemmas may well have broader application.

LEMMA 5.2.2. *For* $t \in \mathbf{N} - \{0\}$, $s \leqslant t-1$, $0 \leqslant p \leqslant 1$, *it is the case that*

$$\sum_{i=0}^{s} \binom{t}{i} p^i (1-p)^{t-i} (tp - i) = tp \binom{t-1}{s} p^s (1-p)^{t-s}.$$

Proof. Applying the identity $i\binom{t}{i} = t\binom{t-1}{i-1}$ gives

$$\binom{t}{i} p^i (1-p)^{t-i} (tp - i) = tp \binom{t}{i} p^i (1-p)^{t-i} - t \binom{t-1}{i-1} p^i (1-p)^{t-i},$$

for $1 \leqslant i \leqslant t$. Since $\binom{t}{i} = \binom{t-1}{i} + \binom{t-1}{i-1}$, this expression is in turn equal to

$$tp \binom{t-1}{i} p^i (1-p)^{t-i} + tp \binom{t-1}{i-1} p^i (1-p)^{t-i} - t \binom{t-1}{i-1} p^i (1-p)^{t-i}$$

$$= tp \binom{t-1}{i} p^i (1-p)^{t-i} - tp \binom{t-1}{i-1} p^{i-1} (1-p)^{t-(i-1)}.$$

Summing from 1 to s, we get

$$tp \left[\sum_{i=1}^{s} \binom{t-1}{i} p^i (1-p)^{t-i} - \sum_{i=1}^{s} \binom{t-1}{i-1} p^{i-1} (1-p)^{t-(i-1)} \right].$$

Changing the limits of summation gives

$$tp \left[\sum_{i=1}^{s} \binom{t-1}{i} p^i (1-p)^{t-i} - \sum_{i=0}^{s-1} \binom{t-1}{i} p^i (1-p)^{t-i} \right]$$

$$= tp \left[\binom{t-1}{s} p^s (1-p)^{t-s} - \binom{t-1}{0} p^0 (1-p)^{t-0} \right].$$

Putting this all together gives

$$\sum_{i=0}^{s} \binom{t}{i} p^i (1-p)^{t-i} (tp - i)$$

$$= \binom{t}{0} p^0 (1-p)^{t-0} (tp - 0) + \sum_{i=1}^{s} \binom{t}{i} p^i (1-p)^{t-i} (tp - i)$$

$$= tp(1-p)^t + tp \left[\binom{t-1}{s} p^s (1-p)^{t-s} - \binom{t-1}{0} p^0 (1-p)^{t-0} \right]$$

$$= tp \binom{t-1}{s} p^s (1-p)^{t-s}. \quad \blacksquare$$

LEMMA 5.2.3. *If* $t \in \mathbf{N} - \{0\}$, $0 \leqslant p < 1$, *then*

$$\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{(t-i)} |tp - i| = 2tp \binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor} (1-p)^{t - \lfloor tp \rfloor}.$$

Proof. Applying Lemma 5.2.2 to the summation for i from 0 to $\lfloor tp \rfloor$, we get

$$\begin{aligned}
&\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |tp - i| \\
&\quad = \sum_{i=0}^{\lfloor tp \rfloor} \binom{t}{i} p^i (1-p)^{t-i} (tp - i) - \sum_{i=\lfloor tp \rfloor + 1}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp - i) \\
&\quad = 2 \sum_{i=0}^{\lfloor tp \rfloor} \binom{t}{i} p^i (1-p)^{t-i} (tp - i) - \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp - i) \\
&\quad = 2tp \binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor} (1-p)^{t - \lfloor tp \rfloor} - \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp - i).
\end{aligned}$$

But by rearranging terms and applying the identity $i\binom{t}{i} = t\binom{t-1}{i-1}$, we see that the summation in the second term of the last line above is 0:

$$\begin{aligned}
&\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp - i) \\
&\quad = \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} tp - \sum_{i=1}^{t} \binom{t}{i} p^i (1-p)^{t-i} i \\
&\quad = tp \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} - tp \sum_{i=1}^{t} \binom{t-1}{i-1} p^{i-1} (1-p)^{(t-1)-(i-1)} \\
&\quad = tp - tp = 0. \quad \blacksquare
\end{aligned}$$

The next lemma provides tight upper and lower bounds on $\texttt{edgecost}_e$, for p bounded away from 0 and 1.

LEMMA 5.2.4. *Let* $t \in \mathbf{N}$, $t \geqslant 3$ *and assume* $1 \leqslant tp < t - 1$. *Then*

$$\varepsilon_1(t) \sqrt{\frac{2tp}{\pi} \left(\frac{tp}{\lfloor tp \rfloor} - p \right)} < \texttt{edgecost}_e(tp) < \varepsilon_2(t) \sqrt{\frac{2tp}{\pi} \left(\frac{tp}{\lfloor tp \rfloor} - p \right)},$$

where $\varepsilon_1(t) = \frac{1}{2}(\frac{11}{12} - \frac{13}{12} \cdot (1/((12t+1)(t-1))))$ *and* $\varepsilon_2(t) = (1 - t/(2t+1)^2)$.

Proof. Using the form of Stirling's formula (Feller, 1950):

$$\sqrt{2\pi}\, n^{n+1/2}\, e^{-n} e^{(12n+1)^{-1}} < n! < \sqrt{2\pi}\, n^{n+1/2}\, e^{-n} e^{(12n)^{-1}},$$

we can bound $\binom{t}{j}$:

$$\frac{t^{t+1/2}}{\sqrt{2\pi}\, j^{j+1/2}(t-j)^{(t-j)+1/2}} f_t(j) < \binom{t}{j} < \frac{t^{t+1/2}}{\sqrt{2\pi}\, j^{j+1/2}(t-j)^{(t-j)+1/2}} g_t(j),$$

where

$$f_t(j) = \exp\left(\frac{1}{12t+1} - \frac{1}{12j} - \frac{1}{12(t-j)}\right)$$

and

$$g_t(j) = \exp\left(\frac{1}{12t} - \frac{1}{12j+1} - \frac{1}{12(t-j)+1}\right).$$

Define $\psi_t(j)$ so that

$$\binom{t}{j} = \frac{t^{t+1/2}}{\sqrt{2\pi}\, j^{j+1/2}(t-j)^{(t-j)+1/2}} \psi_t(j).$$

Then clearly $f_t(j) < \psi_t(j) < g_t(j)$.

From Theorem 4.1.1 and Lemma 5.2.3,

$$\begin{aligned}
&\texttt{edgecost}_e(tp) \\
&\quad = 2tp\binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} \\
&\quad = 2tp\left(\frac{t-\lfloor tp \rfloor}{t}\right)\binom{t}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} \\
&\quad = 2p(t-\lfloor tp \rfloor)\left[\frac{t^{t+1/2}}{\sqrt{2\pi}\, \lfloor tp \rfloor^{\lfloor tp \rfloor+1/2}(t-\lfloor tp \rfloor)^{(t-\lfloor tp \rfloor)+1/2}} \psi_t(\lfloor tp \rfloor)\right] \\
&\qquad \times p^{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} \\
&\quad = (\sqrt{2/\pi})\, p(t-\lfloor tp \rfloor)\left(\sqrt{\frac{t}{\lfloor tp \rfloor(t-\lfloor tp \rfloor)}}\right)\left(\frac{tp}{\lfloor tp \rfloor}\right)^{\lfloor tp \rfloor} \\
&\qquad \times \left(\frac{t-tp}{t-\lfloor tp \rfloor}\right)^{t-\lfloor tp \rfloor} \psi_t(\lfloor tp \rfloor) \\
&\quad = \sqrt{\frac{2tp}{\pi}\left(\frac{tp}{\lfloor tp \rfloor} - p\right)}\left(\frac{tp}{\lfloor tp \rfloor}\right)^{\lfloor tp \rfloor}\left(\frac{t-tp}{t-\lfloor tp \rfloor}\right)^{t-\lfloor tp \rfloor} \psi_t(\lfloor tp \rfloor).
\end{aligned}$$

Thus we need only show that

$$\varepsilon_1(t) < \left(\frac{tp}{\lfloor tp \rfloor}\right)^{\lfloor tp \rfloor} \left(\frac{t-tp}{t-\lfloor tp \rfloor}\right)^{t-\lfloor tp \rfloor} \psi_t(\lfloor tp \rfloor) < \varepsilon_2(t).$$

First, we examine $\psi_t(\lfloor tp \rfloor)$. Since $f_t(\lfloor tp \rfloor) < \psi_t(\lfloor tp \rfloor) < g_t(\lfloor tp \rfloor)$ we find a lower bound for f_t and an upper bound for g_t for j an integer in the range $1 \leqslant j \leqslant t-2$. $f_t(j)$ is minimized at the endpoints of this interval, so taking $j=1$, we have

$$f_t(j) \geqslant \exp\left(\frac{1}{12t+1} - \frac{1}{12} - \frac{1}{12(t-1)}\right)$$

$$= \exp\left(-\left(\frac{1}{12} + \frac{13}{12} \cdot \frac{1}{(12t+1)(t-1)}\right)\right).$$

Two terms of Taylor's formula give

$$e^{-x} = 1 - x + \left(\frac{x^2}{2}\right) e^{-\eta} \qquad \text{for} \quad \eta \in [0, x],$$

so

$$e^{-x} \geqslant 1 - x \qquad \text{when} \quad x > 0.$$

Thus,

$$f_t(j) \geqslant \frac{11}{12} - \frac{13}{12} \cdot \frac{1}{(12t+1)(t-1)} = 2\varepsilon_1(t).$$

Similarly, $g_t(j)$ is maximized at $j = t/2$, so that

$$g_t(j) \leqslant \exp\left(\frac{1}{12t} - \frac{2}{6t+1}\right) \leqslant \exp\left(-\left(\frac{1}{2(2t+1)}\right)\right).$$

Three terms of Taylor's formula give

$$e^{-x} = 1 - x + \frac{x^2}{2} - \left(\frac{x^3}{6}\right) e^{-\eta} \qquad \text{for} \quad \eta \in [0, x],$$

so

$$e^{-x} \leqslant 1 - x + \frac{x^2}{2} \qquad \text{when} \quad x > 0.$$

Thus,

$$g_t(j) \leqslant 1 - \frac{1}{2(2t+1)} + \frac{1}{8(2t+1)^2} \leqslant 1 - \frac{t}{(2t+1)^2} = \varepsilon_2(t).$$

Hence,

$$2\varepsilon_1(t) \leqslant \psi_t(t) \leqslant \varepsilon_2(t).$$

Finally, let

$$h(\delta) = \left(1 + \frac{\delta}{\lfloor tp \rfloor}\right)^{\lfloor tp \rfloor} \left(1 - \frac{\delta}{t - \lfloor tp \rfloor}\right)^{t - \lfloor tp \rfloor},$$

so

$$h(tp - \lfloor tp \rfloor) = \left(\frac{tp}{\lfloor tp \rfloor}\right)^{\lfloor tp \rfloor} \left(\frac{t - tp}{t - \lfloor tp \rfloor}\right)^{t - \lfloor tp \rfloor}.$$

It suffices to show that $h(tp - \lfloor tp \rfloor)$ lies in the interval $[\frac{1}{2}, 1]$, for then

$$\varepsilon_1(t) \leqslant h(tp - \lfloor tp \rfloor)\psi_t(t) \leqslant \varepsilon_2(t).$$

We show in fact that $h(\delta) \in [\frac{1}{2}, 1]$ for all $\delta \in [0, 1)$, $t \geqslant 3$, and $1 \leqslant \lfloor tp \rfloor \leqslant t - 2$. By examining the derivative and the extreme values, we see that h is a monotone decreasing function of δ, decreasing from a maximum of 1 when $\delta = 0$ to a minimum of

$$h(1) = \left(1 + \frac{1}{\lfloor tp \rfloor}\right)^{\lfloor tp \rfloor} \left(1 - \frac{1}{t - \lfloor tp \rfloor}\right)^{t - \lfloor tp \rfloor}$$

as δ approaches 1. The first factor is increasing in $\lfloor tp \rfloor$ and is minimized when $\lfloor tp \rfloor$ is as small as possible, i.e., when $\lfloor tp \rfloor = 1$. The second factor is increasing in $t - \lfloor tp \rfloor$ and is minimized when $t - \lfloor tp \rfloor$ is as small as possible, i.e., when $t - \lfloor tp \rfloor = 2$. Hence, $1 \geqslant h(\delta) \geqslant h(1) \geqslant \frac{1}{2}$ as desired. ∎

Lemma 5.2.5 provides an upper bound on $\texttt{edgecost}_e$ for use in Lemma 5.2.6.

LEMMA 5.2.5. *For all trees $T = (V, E)$ and all probability density functions ϕ on T, and for $t \in \mathbf{N} - \{0\}$ and $0 \leqslant p \leqslant 1$,*

$$\texttt{edgecost}_e(tp) < \left(\frac{2}{\sqrt{\pi}}\right)\sqrt{tp}.$$

Proof. Using Lemma 5.2.4, we see that for $\lfloor tp \rfloor$ between 1 and $t - 2$, inclusive,

$$\begin{aligned}\texttt{edgecost}_e(tp) &< \left(\sqrt{\frac{2}{\pi}}\right)\sqrt{tp(2-p)} \\ &= \left(\frac{2}{\sqrt{\pi}}\right)\sqrt{tp(1-p/2)} \\ &\leqslant \left(\frac{2}{\sqrt{\pi}}\right)\sqrt{tp}.\end{aligned}$$

For $\lfloor tp \rfloor = 0$, Lemma 5.2.3 gives

$$\begin{aligned}\texttt{edgecost}_e(tp) &= 2tp\binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} \\ &= 2tp(1-p)^t.\end{aligned}$$

Let ρ be the ratio of $\texttt{edgecost}_e(tp)$ to the desired bound, so

$$\rho = \frac{2tp(1-p)^t}{(2/\sqrt{\pi})\sqrt{tp}} = \sqrt{\pi tp}\,(1-p)^t.$$

We wish to show that $\rho \leqslant 1$. Fixing t, we find that ρ is maximized when $p = 1/(2t+1)$. Substituting in, we get

$$\begin{aligned}\rho &\leqslant \sqrt{\frac{\pi t}{2t+1}}\left(1-\frac{1}{2t+1}\right)^t \\ &\leqslant \sqrt{\frac{\pi t}{2t}}\left(1-\frac{1}{3t}\right)^t \\ &\leqslant \sqrt{\frac{\pi}{2}}\cdot e^{-1/3} \\ &= 0.898\ldots \leqslant 1.\end{aligned}$$

Similarly, for $\lfloor tp \rfloor = t-1$, Lemma 5.2.3 gives

$$\begin{aligned}\texttt{edgecost}_e(tp) &= 2tp\binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} \\ &= 2tp^t(1-p).\end{aligned}$$

This time, the ratio of $\texttt{edgecost}_e(tp)$ to our bound gives

$$\rho = \frac{2tp^t(1-p)}{(2/\sqrt{\pi})\sqrt{tp}} = \sqrt{\pi t}\,p^{t-1/2}(1-p).$$

Calculus shows ρ is maximized by taking $p = (t-\frac{1}{2})/(t+\frac{1}{2})$. Substituting in, we get

$$\begin{aligned}\rho &\leqslant \sqrt{\pi t}\left(\frac{t-\frac{1}{2}}{t+\frac{1}{2}}\right)^{t-1/2}\left(1-\frac{t-\frac{1}{2}}{t+\frac{1}{2}}\right) \\ &= \sqrt{\pi t}\left(1-\frac{1}{t+\frac{1}{2}}\right)^{t+1/2}\left(\frac{1}{t-\frac{1}{2}}\right).\end{aligned}$$

If $t = 1$, this simplifies to

$$\sqrt{\pi} \cdot (\tfrac{1}{3})^{3/2} \cdot 2 = \sqrt{\frac{4\pi}{27}} < 1.$$

If $t \geqslant 2$, we have

$$\rho \leqslant \sqrt{\pi t} \cdot e^{-1} \cdot \left(\frac{1}{t - \frac{1}{2}}\right) = \left(\frac{\sqrt{\pi}}{e}\right) \cdot \left(\frac{\sqrt{t}}{t - \frac{1}{2}}\right) < 1$$

since both of these factors are less than 1.

Finally, if $\lfloor tp \rfloor = t$, then $p = 1$, so $\texttt{edgecost}_e(tp) = 0$. ∎

This last lemma is proved inductively and implies Theorem 5.2.1.

LEMMA 5.2.6. *Let T be an arbitrary tree, let $t \in \mathbf{N} - \{0\}$, let ϕ be a probability density function, and let s be the exact fair placement for t and ϕ on T. Let $e \in E$. Then*

$$\texttt{treecost}_e(s) \leqslant c\beta_e \sqrt{tp_e},$$

for $c = 2/\sqrt{\pi}$.

Proof. We use induction on edges in the tree, working from the leaves towards the root.

If e is a lowest edge, then $s(\texttt{low}(e)) = tp_e$ and $\beta_e = 1$, so $\texttt{treecost}_e(s) = \texttt{edgecost}_e(s) \leqslant c\sqrt{tp_e}$ by Lemma 5.2.5.

Now, assume the result holds for edges below e. Let $e_1, \ldots, e_d$ denote the immediate descendant edges of e, and let $v = \texttt{low}(e)$. By the definitions,

$$\texttt{treecost}_e(s) = \texttt{edgecost}_e(s) + \sum_{i=1}^{d} \texttt{treecost}_{e_i}(s).$$

By the induction hypothesis,

$$\texttt{treecost}_{e_i}(s) \leqslant c\beta_{e_i} \sqrt{tp_{e_i}}.$$

By Lemma 5.2.5,

$$\texttt{edgecost}_e(s) \leqslant c\sqrt{tp_e}.$$

Hence, putting everything together, we have

$$\texttt{treecost}_e(s) \leqslant c\sqrt{tp_e} + \sum_{i=1}^{d} (c\beta_{e_i} \sqrt{tp_{e_i}}).$$

Applying Schwarz's inequality[8] to the second term, we get

$$\begin{aligned}\texttt{treecost}_e(s) &\leqslant c\sqrt{tp_e} + c\sqrt{t}\,(\sqrt{\textstyle\sum \beta_{e_i}^2})(\sqrt{\textstyle\sum p_{e_i}}) \\ &\leqslant c(1+\sqrt{\textstyle\sum \beta_{e_i}^2})\sqrt{tp_e} \qquad \text{since} \quad \textstyle\sum p_{e_i} \leqslant p_e \\ &= c\beta_e\sqrt{tp_e}. \quad \blacksquare\end{aligned}$$

COROLLARY. *Let T be an arbitrary tree, let $t \in \mathbf{N} - \{0\}$, let ϕ be a probability density function, and let s be any fair placement on T for t and ϕ. Then*

$$\texttt{cost}(s) \leqslant c\beta_{\texttt{rootedge}}\sqrt{tp_{\texttt{rootedge}}} + |E_T|$$

for $c = \sqrt{2/\pi}$.

Proof. The proof is immediate from Theorems 5.2.1 and 5.1.5. ∎

If T is a complete d-ary tree, then a simple inductive argument shows that $\beta_e = \sqrt{k_e}$, where k_e is the number of leaves in $B(e)$. Hence, we obtain the following theorem as a corollary to Theorem 5.2.1.

THEOREM 5.2.7. *Let T be a complete d-ary tree, ϕ an arbitrary probability density function, and $t \in \mathbf{N} - \{0\}$. Let s be the exact fair placement for t and ϕ on T. Then*

$$\texttt{cost}(s) \leqslant c\sqrt{ktp_{\texttt{rootedge}}}$$

for $c = \sqrt{2/\pi}$ and k the number of leaves of T.

We call s a *centralized placement* if $s(v) = 0$ everywhere except at root, and $s(\texttt{root}) = t$. For such an s, and assuming all requests appear at the leaves with equal probability, we have $\texttt{cost}(s) = t\log_d k$. The ratio of expected cost for a centralized placement to the exact fair placement is at least

$$\frac{t\log_d k}{c\sqrt{kt}}.$$

When t is small relative to k, then the centralized placement is superior. However, for $t = \Omega(k)$, the exact fair placement is better by a factor of $\Omega(\log_d k)$, and for $t >> k$, the ratio approaches $\sqrt{t}$.

Similar remarks apply to fair placements in general, using Theorems 5.1.5 and 5.2.7 to bound the cost of any fair placement by $c\sqrt{kt} + m$, where m is the number of edges of T, and observing that

$$m = \frac{kd-1}{d-1}$$

for a complete d-ary tree.

[8] $\sum(x_i \cdot y_i) \leqslant (\sqrt{\sum x_i^2}) \cdot (\sqrt{\sum y_i^2})$.

6. Placements on Symmetric Trees

Not surprisingly, the placement problem becomes somewhat simpler in the case of a complete d-ary tree with a symmetric probability density function ϕ. In such a tree, we can use the following notation without ambiguity:

$\phi(j) = \phi(v)$ for all vertices v at level j.

$p_j = \texttt{total}(B(e), \phi)$ for an edge e with $\texttt{low}(e)$ at level j.

$\texttt{edgecost}_j = \texttt{edgecost}_e$ for all edges e with $\texttt{low}(e)$ at level j.

$\texttt{treecost}_j = \texttt{treecost}_e$ for all edges e with $\texttt{low}(e)$ at level j.

6.1. *Levels with Zero Placement*

In the special case of a complete tree T with a symmetric probability density function ϕ, we can characterize levels below which it is suboptimal to place any resources.

THEOREM 6.1.1. *Let T be a complete d-ary tree with $d \geq 2$, let ϕ be a symmetric probability density function, and let s be an optimal placement for ϕ and $t \in \mathbf{N} - \{0\}$. Assume further that $t > 1$ or $d > 2$. Then $s(v) = 0$ for every vertex v at level $j > \lceil \log_d t \rceil$.*

Proof. Define p_i as above. The number of nodes at level j is d^j, so $p_j \leq 1/d^j$.

If $t = 1$, then $d > 2$ by assumption, so $p_j \leq \frac{1}{3}$ for $j \geq 1$. Theorem 4.2.1 gives $s(v) = 0$ for all v of level 1 or greater as desired.

Now assume $t > 1$. From Theorem 4.2.1, it suffices to show that

$$\left(1 - \frac{1}{d^{\lceil \log_d t \rceil + 1}}\right)^t > \frac{1}{2}.$$

Let h satisfy $d^{h-1} < t \leq d^h$, i.e., $h = \lceil \log_d t \rceil$. Then

$$\left(1 - \frac{1}{d^{\lceil \log_d t \rceil + 1}}\right)^t \geq \left(1 - \frac{1}{d^{h+1}}\right)^{d^h},$$

so that it suffices to show the inequality

$$\left(1 - \frac{1}{d^{h+1}}\right)^{d^h} = \left(1 - \frac{1}{dx}\right)^x > \frac{1}{2}$$

for $x = d^h$. This follows since $(1 - 1/(dx))^x$ is a monotone increasing function of x and d, both x and d are ≥ 2, and $(1 - 1/(2 \cdot 2))^2 = \frac{9}{16} > \frac{1}{2}$. ∎

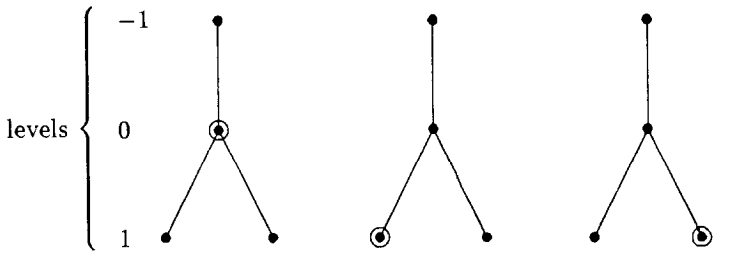

FIG. 6. The case $t = 1$ and $d = 2$.

EXAMPLE 6.1.2. The case where $t = 1$ and $d = 2$ is somewhat peculiar. The reader can verify that for all T with 2 levels, with $\phi(v) = 0.5$ for each node v at level 1, and 0 elsewhere, the diagrams in Fig. 6 all represent optimal placements. That is, in the first case,

$$s(v) = \begin{cases} 1 & \text{for } v = \texttt{root}, \\ 0 & \text{otherwise}, \end{cases}$$

while in the other two cases,

$$s(v) = \begin{cases} 1 & \text{for } v \text{ the left (resp. right) child of the root}, \\ 0 & \text{otherwise}. \end{cases}$$

These are the only values of d and t for which an optimal placement can have nonzero values below level $\lceil \log_d t \rceil$.

Theorem 6.1.1 lets us ignore certain levels when computing an optimal placement for a complete d-ary tree with a symmetric probability density function ϕ.

EXAMPLE 6.1.3. If $h \in \mathbf{N}$, $k \geqslant t = d^h$, and ϕ is symmetric and nonzero only on leaves, then the placement with one resource on each of the t vertices at level h is optimal: nothing is placed below this level, and an optimal placement for the whole tree results from an optimal placement within levels up to h. Since allocating exactly one resource to each of these vertices minimizes $\texttt{edgecost}_e$ on all edges simultaneously, this placement is optimal.

Let T be a tree with k leaves, and assume $k \geqslant t$. We will use the following notation: Let $T_t = (V_t, E_t)$ denote the tree consisting of the vertices of T at levels from -1 to $\lceil \log_d t \rceil$ inclusive and the edges of T between them. The

leaves L_t of T_t are the vertices at level $\lceil \log_d t \rceil$ of T. Let ϕ_t be defined on the vertices v of L_t with $v = \texttt{low}(e)$ by

$$\phi_t(v) = \texttt{total}(B(e), \phi) = p_{\lceil \log_d t \rceil}$$

and on $v \in V_t - L_t$ by $\phi_t(v) = \phi(v)$.

By Theorem 6.1.1, no optimal placement will have $s(v) \neq 0$ for v below level $\lceil \log_d t \rceil$. And if $s: V \to \Re^+$ is such that $s(v) = 0$ for all $v \in V$ at levels below $\lceil \log_d t \rceil$, then $s_t: V_t \to \Re^+$ can be defined from s simply by ignoring the missing vertices.

Let

$$\texttt{extracost}^{(T)} = \sum_{i=\lceil \log_d t \rceil + 1}^{\lceil \log_d k \rceil} t\phi(i)\, d^i (i - \lceil \log_d t \rceil).$$

This represents the expected cost of moving requests which appear at vertices of T below the lowest level of T_t to resources on level $\lceil \log_d t \rceil$.

In the special case where ϕ is nonzero only on leaves, $\phi(i) = 0$ for $i < \lceil \log_d k \rceil$ and $\phi(i) = 1/d^i$ for $i = \lceil \log_d k \rceil$, so

$$\texttt{extracost}^{(T)} = t \cdot (\lceil \log_d k \rceil - \lceil \log_d t \rceil).$$

In any case, it is easy to see that

$$\texttt{cost}^{(T)}(s) = \texttt{cost}^{(T_t)}(s_t) + \texttt{extracost}^{(T)}.$$

(Superscripts distinguish the trees under consideration.)

The following theorem allows us to construct an optimal placement for T with request probabilities ϕ given an optimal placement for T_t with request probabilities ϕ_t, where $t \leqslant k$.

THEOREM 6.1.4. *If $k \geqslant t > 1$, then*

$$\texttt{mincost}^{(T)} = \texttt{mincost}^{(T_t)} + \texttt{extracost}^{(T)}.$$

Proof. An optimal placement s_t for T_t can be augmented to a placement s for T by letting $s(v) = 0$ for vertices at levels greater than $\lceil \log_d t \rceil$, so that

$$\texttt{mincost}^{(T)} \leqslant \texttt{cost}^{(T)}(s) = \texttt{mincost}^{(T_t)} + \texttt{extracost}^{(T)}.$$

Conversely, any optimal placement s for T can be restricted to a placement s_t for T_t, as described above. The cost of placement s is

$$\texttt{mincost}^{(T)} = \texttt{cost}^{(T_t)}(s_t) + \texttt{extracost}^{(T)} \geqslant \texttt{mincost}^{(T_t)} + \texttt{extracost}^{(T)}. \quad \blacksquare$$

6.2. *Complete Trees*

In this subsection, we show that a complete d-ary tree in which the request arrivals are symmetric has an optimal placement s which is also symmetric in the sense that s places the same number of resources on every node of a given level. Because of the symmetry, s can be completely described by listing for each level the number of resources which it places on every node of that level. Moreover, only one subtree of each level need be examined in determining s, so the time needed, when the algorithm of Theorem 4.3.3 is modified in the obvious way, drops to $O(td \log_d k)$, where k is the number of leaves of the tree. We shall see in Theorem 6.2.2 that the time can be reduced still further to $O(\min\{\ell, \log_d t\} + t)$.

Theorem 4.4.4 can also be modified to yield from s an optimal whole placement s' such that $s \rhd s'$ at the cost of only $O(\min\{\ell, \log_d t\})$ additional operations. Because this bound is in general much smaller than the number of nodes in the tree, we must find an economical way of describing s'.

We begin by presenting a refinement of the characterization of the function `treecost` for symmetric probability density functions ϕ.

THEOREM 6.2.1. *Let T be a complete d-ary tree with maximum level ℓ. Let ϕ be a symmetric probability density function, let $t \in \mathbf{N} - \{0\}$, and let $h \in \mathbf{N}$, with $h < \ell$. Then for all $u \in [0, t]$,*

$$\begin{aligned}&\texttt{treecost}_h(u)\\ &\quad = \texttt{edgecost}_h(u)\\ &\qquad + d \min\{\texttt{treecost}_{h+1}(u') \mid u' \in \{0, 1, \ldots, \lfloor u/d \rfloor, u/d\}\}.\end{aligned} \tag{10}$$

Proof. $\leqslant$ is clear. We show $\geqslant$. Write f for $\texttt{treecost}_{h+1}$, and define $f_i = f$ for all i, $1 \leqslant i \leqslant d$.

Let $\mathbf{u}$ be optimal for u. By Theorem 4.3.1,

$$\texttt{treecost}_h(u) = \texttt{edgecost}_h(u) + \sum f(u_i).$$

We will prove the theorem by producing $u' \in \{0, 1, \ldots, \lfloor u/d \rfloor, u/d\}$ with $df(u') \leqslant \sum f(u_i)$.

Let n be the left minimum of f. (Recall the definition in Section 2.2.) We consider two cases.

Case 1. $u \geqslant dn$. Choose $u' = n$. $u' \in \{0, 1, \ldots, \lfloor u/d \rfloor\}$ and $df(u') \leqslant \sum f(u_i)$, since u' minimizes f over the interval $[0, t]$.

Case 2. $u < dn$. Then Lemma 2.3.11 implies that $\sum u_i = u$. Choose $u' = u/d$. Then $df(u') = df(\sum u_i/d) \leqslant \sum f(u_i)$ by convexity of f. ∎

COROLLARY. *For any complete tree T, symmetric probability density function ϕ, and $t \in \mathbf{N} - \{0\}$, there is an optimal placement s of resources such that $s(v) = s(v')$ for all pairs of vertices v, v' at the same level.*

Theorem 6.2.1 leads immediately to an algorithm for finding an optimal placement which is also symmetric in the case of a complete tree and a symmetric request function. Namely, proceed as in the algorithm of Theorem 4.3.3, but it is only necessary to compute $\texttt{edgecost}_j(u)$ and $\texttt{treecost}_j(u)$ at level j for $u \in \{0, 1, ..., \lceil t/d^j \rceil\}$. For each such u, $\texttt{treecost}_j(u)$ is computed using Eq. (10). In doing the minimization, values of $\texttt{treecost}_{j+1}(u')$ have been computed previously for $u' \in \{0, 1, ..., \lceil u/d \rceil\}$ since

$$\left\lceil \frac{u}{d} \right\rceil \leqslant \left\lceil \frac{\lceil t/d^j \rceil}{d} \right\rceil = \left\lceil \frac{t}{d^{j+1}} \right\rceil.$$

To compute $\texttt{treecost}_{j+1}(u')$ for $u' = u/d$, one may interpolate between $\texttt{treecost}_{j+1}(\lfloor u/d \rfloor)$ and $\texttt{treecost}_{j+1}(\lceil u/d \rceil)$ since $\texttt{treecost}_{j+1}$ is linear on the interval $[\lfloor u/d \rfloor, \lceil u/d \rceil]$. By observing that the sequence of minimizing u's corresponding to successive values of u is (weakly) monotonic, it is possible to compute all of the needed values at level j in $O(1 + t/d^j)$ steps. Summing over all levels gives a total time of $O(\ell + t)$.

In case $\ell > \log_d t$ (that is, $k = d^\ell > t$), from Theorem 6.1.4, it suffices to find an optimal placement for ϕ_t in T_t. But T_t has maximum level $\lceil \log_d t \rceil$, so the time required is only $O(\lceil \log_d t \rceil + t)$. (One minor detail is worth mentioning: it is necessary to compute the leaf probabilities for ϕ_t in T_t within time $O(\lceil \log_d t \rceil + t)$. This can be done without examining the ϕ probabilities for levels of T below $\lceil \log_d t \rceil$, by calculating the total probability for ϕ above level $\lceil \log_d t \rceil$, and then dividing the excess equally among the leaves of T_t.)

This proves the following:

THEOREM 6.2.2. *Given a complete d-ary tree T with maximum level ℓ, a symmetric probability density function ϕ, and a number $t \in \mathbf{N} - \{0\}$, there is an algorithm for finding a symmetric optimal placement s of t resources on T in time $O(\min\{\ell, \log_d t\} + t)$.*

In the remainder of this subsection, we give results about optimal whole placements in symmetric trees.

COROLLARY. *For any complete tree T, symmetric probability density function ϕ, and $t \in \mathbf{N} - \{0\}$, there is an optimal, whole placement s such that $|s(v) - s(v')| \leqslant 1$ for all pairs of vertices v, v' at the same level.*

Proof. The proof is by the corollary to Theorem 6.2.1 and Lemma 4.4.2. ∎

Let s be an optimal placement such that $s(v) = s(v')$ for all pairs of vertices v, v' at the same level. By Lemma 4.4.2, an optimal whole placement s' with $s \rhd s'$ can be derived from s in time $O(m)$. We can combine this bound with the bound from Theorem 6.2.2 to get a bound for producing an optimal whole placement for a complete tree with symmetric probability function. However, this bound does not adequately exploit the symmetry.

It is possible to obtain s' from s at the cost of only $O(\min\{\ell, \log_d t\})$ additional operations. Because this bound is in general much smaller than the number of nodes in the tree, we require more economical ways of describing s and s'. First, s is easily described by listing, for each level up to $\min\{\ell, \lceil \log_d t \rceil\}$, the number of resources which it places on every node of that level.

Second, we need an economical way of describing s'. Suppose s places u_j resources on a subtree whose root v is at level j. Then s' places either $\lfloor u_j \rfloor$ or $\lceil u_j \rceil$ resources on that subtree. Thus, s' can be described by the following pieces of information for each level j:

1. the numbers $\lfloor u_j \rfloor$ and $\lceil u_j \rceil$,

2. the number of subtrees rooted at level j on which s' places $\lfloor u_j \rfloor$ resources and the number on which s' places $\lceil u_j \rceil$ resources,

3. for each of the two kinds of subtrees rooted at level j, the number of resources which s' places at the root of the subtree, which is either $\lfloor s(v) \rfloor$ or $\lceil s(v) \rceil$.

This information is sufficient to reconstruct s' in time linear in the size of s (when s is represented economically) and so is a reasonable representation of s'.

Using these economical representations, the construction described for Lemma 4.4.2 produces s' from s in time $O(\min\{\ell, \log_d t\})$.

6.3. *Requests Only at Leaves*

In this subsection, we examine trees which are complete and have symmetric probability functions, and in addition have requests arriving only at leaves. In this case, we see that the corollary to Theorem 6.2.1 can be strengthened to yield a fair placement. Recall that Example 5.1.2 shows that not all complete d-ary trees with symmetric probability density functions have optimal placements that are fair, so that the assumption that requests occur only at leaves is necessary.

THEOREM 6.3.1. *Let T be a complete d-ary tree and let ϕ be a symmetric probability density function that is nonzero only on the leaves of T with level not equal to -1. Let $t \in \mathbf{N} - \{0\}$. Then there exists an optimal placement s of t resources on T which has $s(v) = s(v')$ for all v, v' at the same level, such that s is fair.*

Proof. Let s be the symmetric optimal placement of t resources given by the corollary to Theorem 6.2.1. Then $\texttt{total}(B(e), s)$ is the same for every edge of the same level j, so let u_j denote that common value.[9]

We now show that s is fair, that is, that $\lfloor tp_j \rfloor \leqslant u_j \leqslant \lceil tp_j \rceil$. Suppose not. Let g be the least value of j for which this condition is violated. $g > 0$ since $u_0 = t$ and $p_0 = 1$. Then

$$u_{g-1} = du_g + s(g-1),$$

where for any j, $s(j) = s(v)$ for v a node at level j. We consider two cases.

Case 1. $u_g > \lceil tp_g \rceil$. Then

$$u_{g-1} \geqslant du_g > d\lceil tp_g \rceil \geqslant \lceil dtp_g \rceil = \lceil tp_{g-1} \rceil,$$

so the condition is violated for $g-1$. This contradicts the choice of g.

Case 2. $u_g < \lfloor tp_g \rfloor$. Then

$$du_g < d\lfloor tp_g \rfloor \leqslant \lfloor dtp_g \rfloor = \lfloor tp_{g-1} \rfloor \leqslant u_{g-1}$$

by the choice of g. Hence, $s(g) > 0$. Since $u_g < \lfloor tp_g \rfloor \leqslant \texttt{median}(t, p_g)$, we have

$$\texttt{edgecost}_{e_i}(u_g) > \texttt{edgecost}_{e_i}(\texttt{median}(t, p_g))$$

by Lemma 4.1.4, where e_i is any edge at level g. But then s is not optimal, for moving $\min\{s(g), \texttt{median}(t, p_g) - u_g\} > 0$ resources from $\texttt{high}(e_i)$ to $\texttt{low}(e_i)$ would reduce the cost.

We conclude that s is in fact a fair placement, as desired. ∎

COROLLARY. *Let T be a complete d-ary tree and let ϕ be a symmetric probability density function that is nonzero only on the leaves of T. Let $t \in \mathbf{N} - \{0\}$. Then there exists a fair whole placement s of t resources on T which is optimal.*

Proof. The proof follows from Theorems 6.3.1 and 5.1.1. ∎

The final example shows that fair whole placements might be optimal for symmetric trees even when the placements are far from being symmetric.

EXAMPLE 6.3.2. Consider the placement shown in Fig. 7. It represents an optimal whole placement of 11 resources in a complete binary tree with symmetric probability density function in which requests arrive only at leaves. The optimality can be verified using the algorithm of Theorem 4.3.3.

[9] The level of an edge e is the level of $\texttt{low}(e)$.

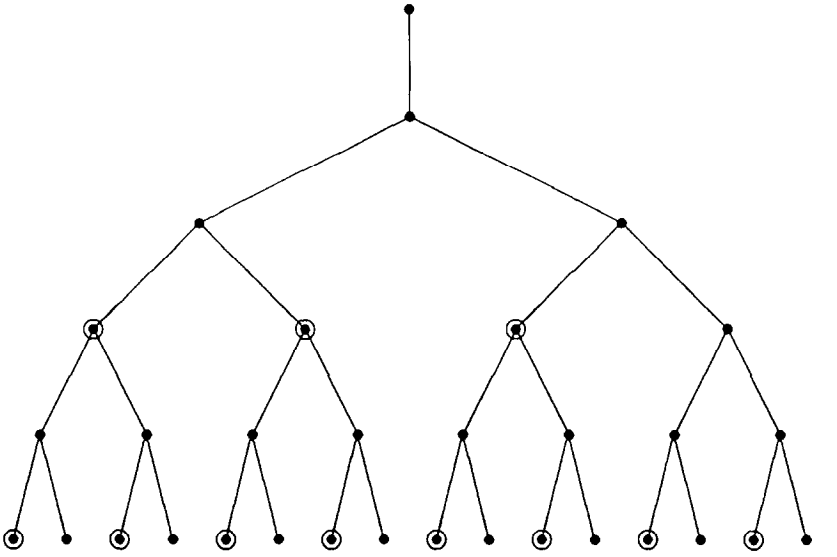

FIG. 7. An optimal placement of 11 resources with equally likely requests only at leaves.

ACKNOWLEDGMENTS

We thank Carl Spruill, Charles Blair, and Mike Paterson for contributing their ideas and suggestions for some of the results in this paper, and we are grateful to Gene Stark and Shmuel Zaks for careful readings of early drafts. We thank Ken Goldman for writing a program to compute costs according to the algorithm of Section 4.3. We also thank the anonymous referees for many helpful comments.

RECEIVED July 31, 1987; FINAL MANUSCRIPT RECEIVED July 30, 1990

REFERENCES

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, MA.

FELLER, W. (1950), "An Introduction to Probability Theory and Its Applications," Vol. 1, 3rd Ed., Wiley, New York.

FISCHER, M. J., GRIFFETH, N. D., GUIBAS, L. J., AND LYNCH, N. A. (1981), Optimal placement of identical resources in a distributed network, *in* "Proceedings, 2nd International Conference on Distributed Computing Systems," IEEE, France.

GUIBAS, L. J., RAMSHAW, L., AND STOFLI, J. (1983), A kinetic framework for computational geometry, *in* "Proceedings, 24th IEEE Symposium on Foundations of Computer Science," pp. 100–111.

JOGDEO, K., AND SAMUELS, S. M. (1968), Monotone convergence of binomial probabilities and a generalization of Ramanujan's, *Ann. Math. Statist.* **39**(4), 1191–1195.

LYNCH, N. A., GRIFFETH, N. D.,, FISCHER, M. J., AND GUIBAS, L. J. (1986), Probabilistic analysis of a network resource allocation algorithm, *Inform. and Control* **68**, 47–85.

PAPADIMITRIOU, C. H., AND STEIGLITZ, K. (1982), "Combinatorial Optimization: Algorithms and Complexity," Prentice–Hall, Englewood Cliffs, NJ.

UHLMANN, W. (1963), Ranggrossen als Schatzfunktionen, *Metrika* **7**, 23–40.

UHLMANN, W. (1966), Vergleich der Hypergeometrischen mit der Binomial-Verteilung, *Metrika* **10**, 145–158.