GIT-ICS-80/13

OPTIMAL PLACEMENT OF IDENTICAL RESOURCES IN A DISTRIBUTED NETWORK[†]

Michael J. Fischer*

Nancy D. Griffeth**

Leo J. Guibas***

Nancy A. Lynch**

October, 1980

*Computer Science Department
University of Washington
Seattle, Washington 98195

**Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

***
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

[†]This research was supported in part by the National Science Foundation under grants MCS77-02474, MCS77-15628, MCS78-01698, MCS80-03337, U.S. Army Research Office Contract DAAG29-79-C-0155 and Office of Naval Research grants N00014-79-C-0873 and N00014-80-C-02221.

OPTIMAL PLACEMENT OF IDENTICAL RESOURCES IN A DISTRIBUTED NETWORK

Michael J. Fischer
University of Washington

Nancy D. Griffeth
Georgia Institute of Technology

Leo J. Guibas
Xerox Palo Alto Research Center

Nancy A. Lynch
Georgia Institute of Technology

Contact Author: Nancy A. Lynch
Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332
U.S.A.
Phone: 404-894-2590
Telex: 542507GTRIOCAATL

This research was supported in part by the National Science Foundation under grants MCS77-02474, MCS77-15628, MCS78-01698, MCS80-03337, U.S. Army Research Office Contract DAAG29-79-C-0155 and Office of Naval Research grants N00014-79-C-0873 and N00014-80-C-0221.

Abstract

This paper addresses the problem of allocating resources to nodes of a distributed network so that the expected cost of servicing requests for the resources is minimized. The cost of servicing a request is defined to be the network distance from the request to the resource satisfying the request.

The network is assumed to be configured as a tree T, with requests arriving at the leaves. There are t requests for t resources. A whole resource placement allocates a whole number of resources to each node. A fair whole resource placement allocates to each subtree a number of resources equal to the expected number of requests in that subtree rounded up or rounded down.

It is shown that for any distribution of requests at the leaves, there is a whole resource placement which minimizes the expected cost of servicing the requests. An algorithm is described which computes such a placement using $O(t^2 \cdot |\text{edges}(T)|)$ arithmetic operations.

It is shown that if T is a balanced binary tree and the requests are uniformly distributed over its leaves, then there is a fair whole resource placement which minimizes the expected cost. An algorithm is described which computes such a placement in $O(t \log_2 n)$ operations, where $n = |\text{leaves}(T)|$.

Finally, it is shown that the expected cost for all t requests, using any fair whole resource placement, is $O(\sqrt{t}\,\sqrt{n})$ if n is $O(t)$, and $O(t \log \frac{n}{t})$ otherwise.

I. Introduction

We consider the problem of locating 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 for those resources. Various different costs would be expected to be important in different situations.

If, for example, the resources are processors, requests come in within a short time interval, and the processors are generally used for very long periods of time, then it is the execution time costs which are most important. A best placement of resources in this case might be one which minimizes the expected total of all the network distances (measured in some appropriate way) between requesting users and their granted resources, in an optimal matching of requests and resources. For in this situation, it is probably reasonable to expend considerable effort to match the entire set of requests to the set of resources in some way approaching an optimal matching. The total of the network distances provides a measure of the expected communication traffic introduced into the network by the computations.

On the other hand, if the resources are tickets in a system for selling tickets to sporting events (or airline seats), then the most relevant cost is the expected waiting time for a buyer before he receives his ticket, or equivalently, the expected total waiting time for all buyers. In this situation, it is probably not reasonable to expend effort attaining an optimal matching. The cost of a placement should include consideration of matchings of requests to resources achievable with partial and out-of-date information about other requests and not just optimal matchings. In this case, the expected total distance in an <u>optimal</u> matching is significant as a lower bound for the expected total waiting time in the achievable matchings.

The present paper and a companion paper now in progress [FGL] contain results which characterize optimal resource placements, and (where succinct characterizations are unavailable) provide fast algorithms for finding optimal placements. They also provide upper bounds for the costs of optimal placements and of certain nearly-optimal and easily-described placements. Both papers assume the network is configured as a tree. (This is not a restriction for an arbitrary distributed network, since a spanning tree can always be constructed for any (connected) network graph.) The present paper focuses on the "optimal matching" cost measure, while [FGL] analyzes the expected total waiting time for matchings achievable by particular distributed algorithms. The results of [FGL] rest heavily on those of this paper.

In this paper, we allow for the possibility that fractions of resources, and not only whole resources, might be located at some nodes (vertices) of the network tree. This is reasonable if, for instance, the resources are large blocks of available data storage space. It would be perfectly permissible for a user to obtain parts of his needed storage from several different nodes.

In Section 2, we present our notation, definitions and those results which apply to arbitrary trees and arbitrary probability distributions for arrivals of requests. §2.1 defines "matchings" formally in terms of network flows. §2.2 defines the expected total network distance in terms of expected total flow. This expected total flow is decomposed into the sum of the expected flows on all the edges of the network tree, and a binomial sum expression is given for the expected flow on each edge. It is seen that both the function describing the expected flow on each edge and the function describing the minimum possible expected total flow for any subtree are of a particularly simple form - they are unbounded, convex, piecewise linear functions on the nonnegative reals, with all vertices at integers. This immediately implies (in §2.3) that there are always optimal resource placements consisting of whole numbers of resources located at each vertex; it is never necessary to place fractions of resources at any node, in order to achieve an optimal placement. An algorithm using $O(t^2 \cdot |\text{edges}(T)|)$ arithmetic operations is presented, which always finds an optimal whole resource placement in this very general case. This is the fastest algorithm we have which is completely general. The rest of Section 2 and Section 3 yield characterizations and faster algorithms for special cases.

§2.4 considers what is required for a placement to optimize the expected flow over any particular edge in the tree - i.e. to be locally optimal. First, it is seen that the expected flow on any edge is always minimized by any placement which puts a number of resources exactly equal to the median of a certain binomial distribution in the subtree headed by that edge. It is not completely obvious how to determine the median of a binomial distribution. An interesting result of Uhlmann [U] is used to show that the median of a binomial distribution is always exactly equal to either the distribution's mean rounded up or rounded down. Thus, each edge individually has its flow optimized at a value which is either the mean rounded up or down. In cases where the means for all edges happen to be integers, it is possible to find a single placement which optimizes the expected flow on each edge of the tree; such a placement is, of course, globally optimal, and is trivial to determine. Unfortunately, however, it is not possible, in general, to obtain a single placement which optimizes the expected flow on all edges.

We call a placement which "almost optimizes" the flow on each edge - that is, one in which the number of resources in each subtree is either the mean rounded up or down - a fair whole placement. We see that it is always possible to obtain a fair whole placement, and while it is not always the case that there is a fair whole placement which is optimal, it is true that all fair whole placements are close to optimal.

In §2.5 we show that if there is a probability less than $\frac{1}{2}$ that any request will arrive in a particular subtree, then it is always bad to place even a very small fraction of a resource anywhere in that subtree.

In Section 3 we consider a special case of the general problem which permits better characterizations and much faster algorithms for determining optimal placements. Namely, we assume here that the network is configured as a complete binary tree, with equal probabilities of requests at all leaves of the tree. First, in §3.1 we give a precise level in the tree below which it is always bad to place any resources or parts of resources. This allows us to restrict consideration to fairly small trees, relative to the number of resources. It also implies that if the number t of resources is a power of two, then the best distribution is to place exactly one resource at each vertex across the level of the tree having exactly t vertices.

In §3.2 we show that the symmetry of the special case of Section 3 allows faster algorithms for determining optimal placements, by permitting pruning of the search space. In particular, an algorithm using $O(t^2 \log n)$ arithmetic operations is presented, where n is the number of leaves in the tree, which finds an optimal (possibly fractional) placement which is completely symmetric across the levels of the tree. Also, another algorithm using only $O(t^2)$ arithmetic operations is presented, which finds an optimal whole resource placement.

§3.3 contains the fastest algorithm we have. The results of §2.4 and §3.2 are combined to produce an algorithm using only $O(t \log n)$ arithmetic operations, which determines a fair whole placement which is optimal.

So far, we have not said anything about the actual cost of optimal placements. In Section 4, we analyze this cost for the case of a balanced tree, but an arbitrary probability distribution for request arrivals. One could always "centralize" all resources at the root of the tree, with an expected total cost of $t \log n$. One would hope that distributing the resources according to an optimal placement or nearly optimal (such as fair whole) placement would afford a considerable improvement. We show that this is indeed the case.

We give a direct analysis for arbitrary fair whole resource placements; this provides an upper bound for optimal placements as well. A recurrence equation is derived and solved, leading to the very small bound of $O(\sqrt{t}\,\sqrt{n})$ for the expected total cost if $n < 2t$ and $O(t \log(\frac{n}{t}))$ if $n \geq 2t$. This bound is <u>always</u> better than in the centralized case, and if the number t of resources is very much larger than the number n of leaves of the tree, the improvement is even more notable. In particular, note that in the very important case where t is roughly proportional to n (i.e. the number of resources in the network is proportional to the number of nodes), that the expected average cost per request is bounded by a constant, independent of the size of the network. This situation is very different from the centralized case, where the average cost grows proportionately with the log of the number of nodes in the network.

In [FGL], the expected waiting time is analyzed for particular new distributed resource matching algorithms. This study involves many more factors than the abstract problem of this paper, and so analysis is considerably more difficult. Simulation results are presented where analytic results are unavailable. In the general case, requests can enter so close together in time that consideration of optimal matching becomes possible, or so far apart in time that each one can get resolved before the system must consider the next one, or anywhere in between. In the former case, it is seen that the bounds of Section 4 of this paper provide a good approximation to the total expected waiting time. The latter case (where requests enter very far apart) can also be handled analytically; it is seen that the same recurrence derived in Section 4 of this paper can be reinterpreted to model this case also. Therefore, the expected total waiting time in this case is also $0(\sqrt{t}\,\sqrt{n})$ if $n < 2t$ and $0(t \log(\frac{n}{t}))$ if $n \geq 2t$.

For intermediate cases,
considerable interference can occur among requests; for instance, the system can expend considerable effort trying to secure a particular resource to satisfy a particular request, only to have that resource "snatched away" at the last moment by another request. These intermediate cases, however, appear to form a continuum between the two extreme cases with intermediate expected total waiting times. Hence, the same bounds seem to hold for these cases as well. These results are still in preliminary form and will be reported in final form at a later date.

2. Notation, Definitions and General Results

2.1 Matchings

In this subsection, we give formal definitions of matchings in terms of network flows.

Let T denote a tree and assume that the root of T has exactly one emanating edge, rootedge(T). Let vert(T), leaves(T) $\subseteq$ vert(T) and edges(T) denote the sets of vertices, leaves and edges of T respectively. If e $\in$ edges(T), then low(e) and high(e) denote the lower and higher endpoints of e, respectively. (We assume that T is oriented so that the root is the highest node.)

Let N denote the natural numbers, including 0, R^+ the nonnegative reals.

In the following, the reader should think of r as representing locations of requests, and s as representing a placement of resources.

If r, s $\in (R^+)^{\text{vert}(T)}$, (that is, the set of functions from vert(T) to R^+), then a matching m of r to s is a pair of functions, $(\text{upflow}_m, \text{downflow}_m)$, from edges(T) to R^+ such that for each v $\in$ vert(T), the following is true:

$$r(v) = s(v) + \sum_{e:\text{low}(e)=v} (\text{downflow}_m(e) - \text{upflow}_m(e)) + \sum_{e:\text{high}(e)=v} (\text{upflow}_m(e) - \text{downflow}_m(e)).$$

That is, the matching is formally identified with a description of the flow of resources. Note that information about specific "pairings" of requests to resources is not retained; this information is not necessary for determining the cost measures of this paper.

If $r \in (R^+)^{\mathrm{vert}(T)}$, then <u>total(r)</u> $= \sum_{v \in \mathrm{vert}(T)} r(v)$.

It follows from the definitions that if any matching is defined from r to s, then total(r) = total(s). Let <u>matchings(r,s)</u> denote the set of matchings of r to s. If $e \in \mathrm{edges}(T)$ and $m \in \mathrm{matchings}(r,s)$, then <u>netflow(e,m)</u> $= \mathrm{upflow}_m(e) - \mathrm{downflow}_m(e)$, and <u>flow(e,m)</u> $= \mathrm{upflow}_m(e) + \mathrm{downflow}_m(e)$.

If $m \in \mathrm{matchings}(r,s)$, then <u>cost(m)</u> $= \sum_{e \in \mathrm{edges}(T)} \mathrm{flow}(e,m)$.

If $r, s \in (R^+)^{\mathrm{vert}(T)}$, then <u>cost(r,s)</u> $= \min_{m \in \mathrm{matchings}(r,s)} \mathrm{cost}(m)$.

A matching m of r to s is <u>optimal</u> if cost(m) = cost(r,s).

If $r \in (R^+)^{\mathrm{vert}(T)}$, $e \in \mathrm{edges}(T)$, then <u>number(e,r)</u> $= \sum_{v \text{ a vertex below } e} r(v)$.

The following three easy theorems lead to an expression of cost(r,s) in terms of flows on all the edges.

<u>Theorem 2.1.1</u>. If m is any matching of r to s, $e \in$ edges(T), then netflow(e,m) = number(e,s) - number(e,r).

<u>Theorem 2.1.2</u>. If m is any matching of r to s, then m is optimal iff for all $e \in$ edges(T), |netflow(e,m)| = flow(e,m).

<u>Proof</u>. If m is any matching not satisfying the equality, then some $e \in$ edges(T) has flow(e,m) > |netflow(e,m)| and so both $\text{upflow}_m(e)$ and $\text{downflow}_m(e)$ are nonzero. Let m' be another matching of r to s, identical to m except that $\text{upflow}_{m'}(e) = \text{upflow}_m(e) - \epsilon$ and $\text{downflow}_{m'}(e) = \text{downflow}_m(e) - \epsilon$, where $\epsilon = \min(\text{upflow}_m(e), \text{downflow}_m(e))$. Clearly, cost(m') < cost(m).

Conversely, for any matching m, we have $\text{cost}(m) = \sum_{e\in \text{edges}(T)} \text{flow}(e,m)$

$$= \sum_{e\in \text{edges}(T)} (\text{upflow}_m(e) + \text{downflow}_m(e)) \geq \sum_{e\in \text{edges}(T)} |\text{upflow}_m(e) - \text{downflow}_m(e)|$$

$$= \sum_{e\in \text{edges}(T)} |\text{netflow}(e,m)|.$$

□

Thus, if m is an optimal matching of r to s, $e \in$ edges(T), then flow(e,m) = |number(e,s) - number(e,r)|. If $u \in R^+$, let <u>flow(e,r,u)</u> denote |u - number(e,r)|. so that if m is an optimal matching of r to s, $e \in$ edges(T), it is the case that flow(e,m) = flow(e,r,number(e,s)).

Theorem 2.1.3. If $r,s \in (R^+)^{\mathrm{vert}(T)}$ with total(r) = total(s), then

$$\mathrm{cost}(r,s) = \sum_{e \in \mathrm{edges}(T)} \mathrm{flow}(e,r,\mathrm{number}(e,s)).$$

Proof. Cost(r,s) = cost(m) for m an optimal matching of r to s,

$$= \sum_{e \in \mathrm{edges}(T)} \mathrm{flow}(e,m) = \sum_{e \in \mathrm{edges}(T)} \mathrm{flow}(e,r,\mathrm{number}(e,s)).$$

□

2.2 Expected Costs

In this section, we introduce probability distributions for the sets of requests and define costs of placements in terms of their expected costs for all sets of requests. For simplicity, we consider requests entering at the leaves of T only.

If ϕ is a probability function on leaves(T) and $t \in N$, then $r \in (R^+)^{\mathrm{vert}(T)}$ can be chosen according to a probability distribution determined as follows: for each i in turn, $1 \le i \le t$, ϕ is used to select a vertex in leaves(T). Then r(v) is the total number of times v is selected, for each $v \in \mathrm{vert}(T)$.

Fix ϕ, t as above. If $s \in (R^+)^{\mathrm{vert}(T)}$ with total(s) = t, then $\underline{\mathrm{expcost}_{\phi,t}}(s)$ denotes the expected value of cost(r,s), where r is chosen as described above. If $e \in \mathrm{edges}(T)$ and $u \in R^+$, then $\underline{\mathrm{expflow}_{\phi,t,e}}(u)$ denotes the expected value of flow(e,r,u) where r is chosen as above.

The following two expansions are easy to see.

<u>Theorem 2.2.1</u>. $\mathrm{Expcost}_{\phi,t}(s) = \sum_{e \in \mathrm{edges}(T)} \mathrm{expflow}_{\phi,t,e}(\mathrm{number}(e,s))$.

<u>Theorem 2.2.2</u>. $\mathrm{Expflow}_{\phi,t,e}(u) = \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |u-i|$, where $p = \sum_{\ell \text{ below } e} \phi(\ell)$.

The next theorem shows that the expflow function has a simple form.

<u>Theorem 2.2.3</u>. For any fixed ϕ, t, e, it is the case that $\mathrm{expflow}_{\phi,t,e}$ is an unbounded, convex, piecewise linear function from R^+ to R^+, with all vertices occurring at integer values.

Proof. By Theorem 2.2.2, $\text{expflow}_{\phi,t,e}$ is the sum of the functions g_i, for $0 \le i \le t$, where $g_i(u) = k_i|u-i|$, $k_i = \binom{t}{i}p^i(1-p)^{t-i}$, and $p = \sum_{\ell \text{ below } e} \phi(\ell)$. Each g_i is an unbounded, convex, piecewise linear function from R^+ to R^+ with a single integer vertex at $u = i$. Since addition preserves all four required properties, the result follows.

$\square$

If $e \in \text{edges}(T)$, let $\underline{E_e}$ denote the set consisting of e, together with all edges below e in T. If ϕ, t, e are as above, $u \in R^+$, then

$$\underline{\text{minexpcost}_{\phi,t,e}(u)} = \min_{s:\text{number}(e,s)=u} \left(\sum_{d \in E_e} \text{expflow}_{\phi,t,d}(\text{number}(d,s))\right).$$

That is, the minexpcost function describes the best possible expected total flow for any subtree, if exactly u resources are to be placed in that subtree. It is easily seen that the expected flow over the top edge in the subtree is uniquely determined by u. The best possible total cost is then obtained by a best possible split of the u resources among the immediate descendant subtrees, and an optimal arrangement of those resources within each descendant subtree. Thus, it is straightforward to observe the following relationship.

Theorem 2.2.4. If ϕ, t, e are as above, if $e_1,\dots,e_\ell$ are the immediate descendants of edge e, and if $u \in R^+$, then

$$\text{minexpcost}_{\phi,t,e}(u) = \text{expflow}_{\phi,t,e}(u) + \min_{\Sigma u_i \le u} \left(\sum_{1 \le i \le \ell} \text{minexpcost}_{\phi,t,e_i}(u_i)\right).$$

In order to obtain more information about the values of the minexpcost function, we first show that it is of a simple form.

Theorem 2.2.5. For any fixed ϕ, t, e, it is the case that $\mathrm{minexpcost}_{\phi,t,e}$ is an unbounded, convex, piecewise linear function from R^+ to R^+, with all vertices occurring at integer values.

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

If e is a lowest edge, then there is only one edge, e, in E_e. Thus, $\mathrm{minexpcost}_{\phi,t,e} = \mathrm{expflow}_{\phi,t,e}$, which has the needed properties by Theorem 2.2.3.

Now assume the result holds for edges below e, and let $e_1,\ldots,e_\ell$ denote the immediate descendant edges of e. Consider the expression for $\mathrm{minexpcost}_{\phi,t,e}(u)$ given in Theorem 2.2.4. The first term has the needed properties, by Theorem 2.2.3. It remains to show that the second term is convex, piecewise linear and has all its vertices at integers.

Write $\underline{f_i}$ for $\mathrm{minexpcost}_{\phi,t,e_i}$, $\underline{g(u)}$ for $\min\limits_{\Sigma u_i \le u} (\sum\limits_{1\le i\le \ell} f_i(u_i))$.

By inductive hypothesis, each f_i is unbounded, convex and piecewise linear with all vertices at integers. Therefore, for each i, $1 \le i \le \ell$, there is some $r_i \in N$ such that r_i is the smallest element of R^+ with $f_i(r_i) \le f_i(s)$ for all $s \ge r_i$. We consider two cases.

Case 1. $u \ge \sum\limits_{1\le i\le \ell} r_i$.

Then $g(u) = \sum\limits_{1\le i\le \ell} f_i(r_i)$, so that g is constant for u sufficiently large.

Case 2. $u < \sum_{1 \le i \le \ell} r_i$.

Then it is possible to select $u_1, \ldots, u_\ell$ minimizing $\sum_{1 \le i \le \ell} f_i(u_i)$ subject to the restriction $\Sigma u_i \le u$, in such a way that $u_i \le r_i$ for all i and $\Sigma u_i = u$. We give a procedure for selecting for any given u, a fixed decomposition $v_1, \ldots, v_\ell$ having these properties.

For each i, $1 \le i \le \ell$, and each j, $1 \le j \le r_i$, define <u>decrease(i,j)</u> $= f_i(j-1) - f_i(j)$. By choice of r_i and convexity of f_i, decrease is always positive. Now for any decomposition $u_1, \ldots, u_\ell$ with $u_i \le r_i$ for all i, it is the case that

(*) $$\sum_{1 \le i \le \ell} f_i(u_i) =$$

$$\sum_{1 \le i \le \ell} \left[f_i(0) - \left[\sum_{1 \le j \le \lfloor u_i \rfloor} \text{decrease}(i,j) + (u_i - \lfloor u_i \rfloor)\,\text{decrease}(i, \lfloor u_i \rfloor + 1) \right] \right].$$

This equation suggests that $g(u)$ can be minimized by choosing $v_1, \ldots, v_\ell$ so that the largest $\lceil u \rceil$ values of decrease(i,j) are used.

Let <u>U</u> denote the set of triples

$\{(\text{decrease}(i,j), i,j)) : 1 \le i \le \ell, 1 \le j \le r_i\}$.

Fix a total ordering $\lessdot$ of U satisfying the following properties.

(1) If decrease(i,j) < decrease(i',j'), then

$(\text{decrease}(i,j), i,j) \lessdot (\text{decrease}(i',j'), i',j')$,

(2) $(\text{decrease}(i,j+1), i,j+1) \lessdot (\text{decrease}(i,j), i,j)$.

That these two conditions are compatible with each other follows from the convexity of the f_i.

Now, for any given $u < \sum_{1 \le i \le \ell} r_i$, select $v_1, \ldots, v_\ell$ as follows.
Let $U^* \subseteq U$ contain the $\lceil u \rceil$ largest elements of U (according to $\lessdot$).
If $(\text{decrease}(i',j'), i',j')$ is the smallest element of U^*, then for $i \ne i'$ let $v_i = \max\{j\colon (\text{decrease}(i,j), i,j) \in U^*\}$, and let $v_{i'} = j' - 1 + u - \lceil u \rceil$. Then $\Sigma v_i = u$ and $v_i \le r_i$ for all i. By (*) above, it is clear that $v_1, \ldots, v_\ell$ minimizes $\sum_{1 \le i \le \ell} f_i(u_i)$.
The form of (*) and the definition of $\lessdot$ immediately imply that $\min_{\Sigma u_i \le u} (\sum_{1 \le i \le \ell} f_i(u_i))$ is piecewise linear with vertices at integers, and (because the decreases are used in decreasing order) that it is convex.

□

Examination of the proof of Theorem 2.2.5 allows us to sharpen Theorem 2.2.4 by stating that the minimum cost can always be achieved by placing whole resources on all vertices.

Theorem 2.2.6. If ϕ, t, e are as above, if $e_1, \ldots, e_\ell$ are the immediate descendant edges of e, and if $u \in N$, then

$$\text{minexpcost}_{\phi,t,e}(u) = \text{expflow}_{\phi,t,e}(u) + \min_{\substack{\Sigma u_i \le u \\ u_i \in N}} \Big(\sum_{1 \le i \le \ell} \text{minexpcost}_{\phi,t,e_i}(u_i) \Big).$$

2.3 Optimal Placements

We are interested in determining the "best" functions $s \in (R^+)^{vert(T)}$ in the following sense. We say $s \in (R^+)^{vert(T)}$ is *optimal* for ϕ, t provided $total(s) = t$ and $expcost_{\phi,t}(s) = minexpcost_{\phi,t,rootedge(T)}(t)$. The first characterization result follows immediately from Theorem 2.2.6 and shows that there are optimal s which take on integral values only.

Theorem 2.3.1. For any probability function ϕ on leaves(T), and any $t \in N$, there exists $s \in (N)^{vert(T)}$ which is optimal for ϕ, t.

Proof. Theorem 2.2.6 essentially provides an algorithm for producing such s. For any edge e of T, with immediate descendants $e_1,\ldots,e_\ell$, and any $u \in N$, one determines values of s for all nodes below e by considering all possible decompositions $u_1,\ldots,u_\ell$ with $\Sigma u_i \leq u$ and $u_i \in N$ for all i. For each such decomposition, one recursively determines values of s for all nodes below each e_i, and corresponding costs. The decomposition with the smallest total cost is chosen.

$\square$

In order to analyze the cost of determining an optimal placement as above, we do not perform a straightforward recursive analysis of the algorithm described in the proof of Theorem 2.3.1. Rather, we take advantage of repeated work in various recursive calls. During the algorithm, one must calculate $expflow_{\phi,t,e}(u)$ for all $e \in edges(T)$ and all u, $0 \leq u \leq t$. The number of arithmetic operations involved in one calculation of $expflow_{\phi,t,e}(u)$ is $O(t)$ (if performed judiciously), independent of e and u. It is these costs which dominate the total count of arithmetic operations, so that an $0(t^2 \cdot |\text{edges}(T)|)$ analysis results. We summarize this discussion in the following theorem.

Theorem 2.3.2. There is an algorithm using $0(t^2 \cdot |\text{edges}(T)|)$ arithmetic operations which, for any tree T, for any probability function ϕ on leaves(T) and for any $t \in N$, determines an $s \in (N)^{\text{vert}(T)}$ which is optimal for ϕ, t.

2.4 Optimizing Flow on Individual Edges

In this section, we show that the flow on each individual edge is optimized for a number equal to the median of an appropriate binomial distribution. However, we do not have results in this general case which use this local optimization to produce a globally optimal placement.

Let $n \in N - \{0\}$, $0 \le p \le 1$, $c \in R^+$. Let x be a random variable whose value is the number of successes in n independent trials, each of whose probability of success is p. Define median(n,p) as the smallest c such that $\Pr[x \le c] \ge \frac{1}{2}$.

Theorem 2.4.1. Let ϕ be a probability function on leaves(T), $t \in N - \{0\}$, $e \in$ edges(T). Then $\text{expflow}_{\phi,t,e}$ is minimized at u = median(t,p), where $p = \sum_{\ell \text{ below } e} \phi(e)$.

Proof. Write f for $\text{expflow}_{\phi,t,e}$. By Theorem 2.2.3, f is minimized at an integer u which is the smallest $r \in N$ with $f(r+1) - f(r)$ nonnegative.

Now, $f(r+1) - f(r) = \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |r+1-i| - \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |r-i|$, by Theorem 2.2.2,

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

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

$= 2\Pr[x \le r] - 1.$

Thus, u is the smallest $r \in N$ with $2\Pr[x \le r] - 1$ nonnegative, or $\Pr[x \le r] \ge \frac{1}{2}$; that is, $u = \text{median}(t,p)$.

$\square$

It is not obvious exactly how to determine the median of a binomial distribution. Although we do not have an exact characterization, we can use a result of Uhlmann to show that the median is not more than one from the mean. For $n \in N - \{0\}$, $0 \le p \le 1$, $c \in R^+$, x as above, we let $\underline{L_{n,c}(p)}$ denote $\Pr[x \le c]$.

We require the following theorem.

Theorem 2.4.2 (Uhlmann [U])

If $n, c \in N$, $n \ge 2$, then

$$L_{n,c}\left(\frac{c}{n-1}\right) \ge \frac{1}{2} \ge L_{n,c}\left(\frac{c+1}{n+1}\right) \qquad \text{if } 0 \le c \le \frac{1}{2}(n-1),$$

and

$$L_{n,c}\left(\frac{c}{n-1}\right) < \frac{1}{2} < L_{n,c}\left(\frac{c+1}{n+1}\right) \qquad \text{if } \frac{1}{2}(n-1) < c \le n-1.$$

We can now use Uhlmann's result to show that the median of a binomial distribution is always exactly equal to either the mean rounded up or rounded down.

Theorem 2.4.3

For any $n \in N$, $n \geq 1$, $0 \leq p \leq 1$, it is the case that $\lfloor np \rfloor \leq \text{median}(n,p) \leq \lceil np \rceil$.

Proof. If $n = 1$ or $p = 0$ or 1, the result is obvious, so assume $n \geq 2$ and $0 < p < 1$.

First, we assume $np \in N$ and show $\text{median}(n,p) = np$. Since $0 < p < 1$, we have $1 \leq np \leq n - 1$. It suffices to show:
(a) $L_{n,np-1}(p) < \frac{1}{2}$ and (b) $L_{n,np}(p) > \frac{1}{2}$.

(a) If $0 \leq np - 1 \leq \frac{1}{2}(n-1)$, then $L_{n,np-1}(p) = L_{n,np-1}(\frac{np}{n})$, $< L_{n,np-1}(\frac{np}{n+1})$ since decreasing the distribution probability serves to increase strictly the probability of at most np-1 successes, $\leq \frac{1}{2}$ by the first inequality of Theorem 2.4.2. On the other hand, if $\frac{1}{2}(n-1) < np - 1 \leq n-1$, then $L_{n,np-1}(p) = L_{n,np-1}(\frac{np-p}{n-1}) < L_{n,np-1}(\frac{np-1}{n-1}) < \frac{1}{2}$ by the second inequality.

(b) If $0 \leq np \leq \frac{1}{2}(n-1)$, then $L_{n,np}(p) = L_{n,np}(\frac{np}{n}) > L_{n,np}(\frac{np}{n-1}) \geq \frac{1}{2}$ by the first inequality, while if $\frac{1}{2}(n-1) < np \leq n - 1$, then

$L_{n,np}(p) = L_{n,np}(\frac{np+p}{n+1}) > L_{n,np}(\frac{np+1}{n+1}) > \frac{1}{2}$.

Therefore, if np is an integer, we have shown that $\text{median}(n,p) = np$. Now consider $np = k + \epsilon$, $k \in N$, $0 < \epsilon < 1$. If $q = p - \frac{\epsilon}{n}$, then

$nq = k = \lfloor np \rfloor$. Since nq is an integer, we have $\text{median}(n,q) = nq$. But since $p > q$, we have $\text{median}(n,p) \geq \text{median}(n,q) = \lfloor np \rfloor$. Similarly, if $q = p + \frac{1-\epsilon}{n}$, then $nq = k + 1 = \lceil np \rceil$. Again, median(n,q) = nq, and since $p < q$, we have median(n,p) $\le$ median(n,q) $= \lceil np \rceil$.

$\square$

Thus, each edge individually has its flow optimized at a value which is either the mean rounded up or the mean rounded down. However, it is not possible to achieve this local optimum consistently throughout the tree. In fact, in this very general setting, an optimal placement might <u>require</u> some subtree to contain a value <u>other</u> than the mean rounded up or down.

For any T, ϕ, t define $s \in (N)^{\text{vert}(T)}$ to be a <u>fair whole placement</u> for T, ϕ, t if total(s) = t and for each $e \in$ edges(T), $\lfloor tp_e \rfloor \le$ number(e,s) $\le \lceil tp_e \rceil$, where $\underline{p_e} = \sum_{\ell \text{ below } e} \phi(e)$. It is not difficult to see that for any T, ϕ, t, at least one fair whole placement exists; in fact, one can be constructed with all nonzero values at the leaves: we start at the root of T with t resources to distribute. Below any edge e in the tree, we will have either $\lfloor tp_e \rfloor$ or $\lceil tp_e \rceil$ to distribute. Assume e has descendants e_1 and e_2. If we have $\lfloor tp_e \rfloor$, then note that $\lfloor tp_{e_1} \rfloor + \lfloor tp_{e_2} \rfloor$ is either equal to $\lfloor tp_e \rfloor$ or to $\lfloor tp_e \rfloor - 1$. In the former case, distribute resources so that e_1 gets $\lfloor tp_{e_1} \rfloor$ and e_2 gets $\lfloor tp_{e_2} \rfloor$. In the latter case, it must be that tp_{e_2} is non-integral. Then $\lceil tp_{e_2} \rceil = \lfloor tp_{e_2} \rfloor + 1$, and we distribute the resources so that e_1 gets $\lfloor tp_{e_1} \rfloor$ and e_2 gets $\lceil tp_{e_2} \rceil$. The sum is then $\lfloor tp_e \rfloor$ in either case. The case where e has $\lceil tp_e \rceil$ is dual.

Example 2.4.1 Let T be a 32-leaf balanced binary tree (except for rootedge(T)), $\phi(\ell) = \frac{15}{256}$ for each of the leftmost 16 leaves ℓ, and $\frac{1}{256}$ for each of the rightmost 16 leaves. Let $t = 16$. The placement s which has $s(\ell) = 1$ for each of the leftmost 16 leaves ℓ, and 0 elsewhere, has number(e,s) = 16 for e the left descendant of rootedge(T). However, the mean for e is $t \cdot \frac{15}{16} = 15$, so that the left subtree contains a value other than its mean rounded up or down. One can calculate $\text{expcost}_{\phi,t}(s)$ and determine by exhaustive searching that it is optimal, and strictly smaller than $\text{expcost}_{\phi,t}(s')$ for any fair whole placement s'.

We note in general, however, that the optimal cost cannot be too much less than the cost of any fair whole placement.

Theorem 2.4.4. Let s be a fair whole placement for T, ϕ, t. Then $\text{expcost}_{\phi,t}(s) \le \text{minexpcost}_{\phi,t,\text{rootedge}(T)}(t) + |\text{edges}(T)|$.

Proof.

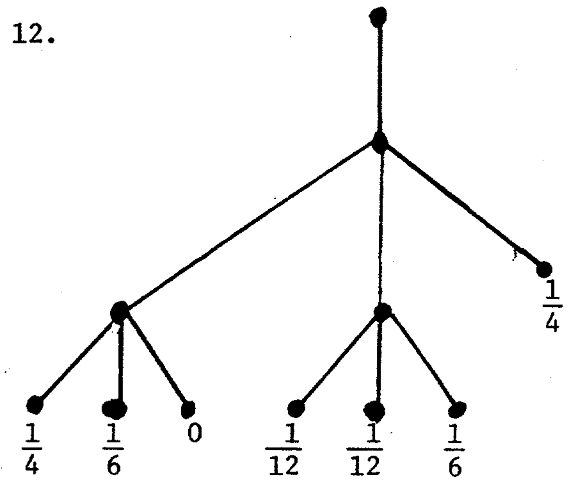
By the results of this section, $\text{expflow}_{\phi,t,e}$ is minimized at some u, where $\lfloor tp_e \rfloor \le u \le \lceil tp_e \rceil$. Thus, $|\text{number}(e,s) - u| \le 1$. But then $|\text{expflow}_{\phi,t,e}(\text{number}(e,s)) - \text{expflow}_{\phi,t,e}(u)| \le 1$, by calculations similar to those in the proof of Theorem 2.4.1. Since no placement can do better than the optimal on each edge, s incurs at most an extra cost of 1 per edge.

$\square$

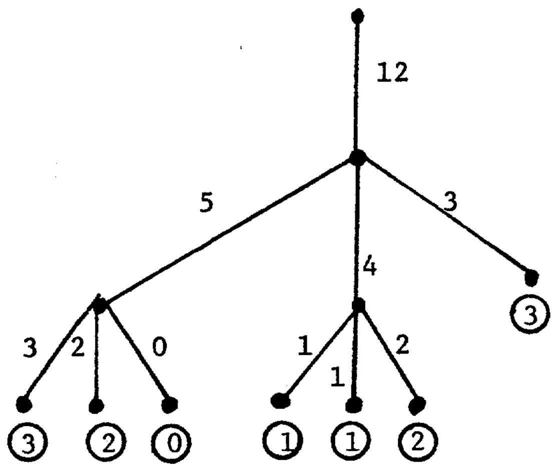
For some choices of T, t and ϕ, of course, it is possible to consistently achieve the median on each edge - for example, if the mean is an integer for every edge. But in general, we have no global optimality results. In §3 we obtain optimal results for a special case.

Example 2.4.2. If T is a balanced binary tree, n(T), the number of leaves of T, $=2^k$, $t = a \cdot 2^k$ for integer a, ϕ the uniform distribution, then $s(\ell) = a$ for each leaf ℓ is optimal, because it achieves the integer mean on each edge.

Example 2.4.3. Let T be the tree depicted below, ϕ as indicated on the leaves, t = 12.



Then the means on each edge are as indicated below, so all tickets can be placed at the levels as indicated by the circled numbers.



2.5 Nodes with Zero Placements

We conclude this section with a somewhat surprising characterization theorem. It says that if there is a probability less than 1/2 of any request arriving in a subtree, then it is bad to place even a small fraction of a resource anywhere in that subtree.

Theorem 2.5.1. Let $e \in \text{edges}(T)$ satisfy $(1-p)^t > \frac{1}{2}$, where $p = \sum_{\ell \text{ below } e} \phi(\ell)$. Let s be optimal for ϕ, t. Then $s(v) = 0$ for all $v \in \text{vert}(T)$ below e in T.

Proof. Assume not, and fix e, s exhibiting the contrary. Choose e_1 below e, a lowest edge for which $s(\text{low}(e_1)) > 0$. Then consider s' with $s'(\text{low}(e_1)) = 0$, $s'(\text{high}(e_1)) = s(\text{high}(e_1)) + s(\text{low}(e_1))$ and $s'(v) = s(v)$ for $v \notin \{\text{high}(e_1), \text{low}(e_1)\}$. We show that $\text{expcost}_{\phi,t}(s') < \text{expcost}_{\phi,t}(s)$, which is a contradiction to the optimality of s.

Now, $\text{expcost}_{\phi,t}(s) = \sum_{e \in \text{edges}(T)} \text{expflow}_{\phi,t,e}(\text{number}(e,s))$ and $\text{expcost}_{\phi,t}(s') = \sum_{e \in \text{edges}(T)} \text{expflow}_{\phi,t,e}(\text{number}(e,s'))$ by Theorem 2.2.1. Since all terms but one in the two sums are identical, we have that $\text{expcost}_{\phi,t}(s) - \text{expcost}_{\phi,t}(s') = \text{expflow}_{\phi,t,e_1}(\text{number}(e_1,s)) - \text{expflow}_{\phi,t,e_1}(\text{number}(e_1,s')) = \text{expflow}_{\phi,t,e_1}(s(\text{low}(e_1)) - \text{expflow}_{\phi,t,e_1}(0)$.

But $\text{expflow}_{\phi,t,e}(s(\text{low}(e_1))) = \sum_{i=0}^{t} \binom{t}{i} r^i (1-r)^{t-i} |s(\text{low}(e_1)) - i|$, where $r = \sum_{\ell \text{ below } e_1} \phi(\ell)$, and

$\text{expflow}_{\phi,t,e_1}(0) = \sum_{i=0}^{t} \binom{t}{i} r^i (1-r)^{t-i}(i)$, by Theorem 2.2.2. Thus, the

$$\text{difference} = \sum_{i=0}^{t} \binom{t}{i} r^i (1-r)^{t-i}(|s(\text{low}(e_1)) - i| - i) = (1-r)^t(s(\text{low}(e_1))) +$$

$$\sum_{i=1}^{t} \binom{t}{i} r^i (1-r)^{t-i}(|s(\text{low}(e_1)) - i| - i) \geq (1-r)^t(s(\text{low}(e_1))) +$$

$$\sum_{i=1}^{t} \binom{t}{i} r^i (1-r)^{t-i}(-s(\text{low}(e_1))) = s(\text{low}(e_1))[(1-r)^t - \sum_{i=1}^{t} \binom{t}{i} r^i (1-r)^{t-i}] =$$

$$s(\text{low}(e_1))[2(1-r)^t - 1] > 0 \text{ since } (1-r)^t \geq (1-p)^t > \frac{1}{2} .$$

This is the needed contradiction.

$\square$

3. Optimal Placements for Balanced Trees

In this section, we give several characterization results and algorithms for optimal placements in an important special case of the problem considered in Section 2. Namely, we assume the following for this section:

(a) T is a balanced binary tree with n(T) leaves (except that, as before, the root has a single emanating edge, rootedge(T)).

(b) $\phi(\ell) = \frac{1}{n(T)}$ for all $\ell \in$ leaves(T).

As before, $t \in N$. We classify the vertices of T into levels, the root at level 0.

For the special case being considered in this section, certain of the relevant definitions can be generalized to reflect the symmetry. For example, explicit mention of ϕ can be omitted since ϕ is determined by T. Also, if e_1 and e_2 are edges with high(e_1) and high(e_2) at the same level k, then $\text{expflow}_{t,e_1} = \text{expflow}_{t,e_2}$. Therefore, we write $\text{expflow}_{t,k}$ in place of either. Similarly, we write $\text{minexpcost}_{t,k}$ for $\text{minexpcost}_{\phi,t,e}$ where k is the level of high(e).

3.1. A Bound on Levels with Nonzero Placements

Theorem 2.5.1 can be used to bound the lowest level at which nonzero placement can occur in an optimal placement.

Theorem 3.1.1. Assume $t > 1$. Let $v \in$ vertices(T) be at level k, where k is greater than $\lceil \log t \rceil + 1$. Let s be optimal for ϕ, t. Then $s(v) = 0$.

Proof. By Theorem 2.5.1, it suffices to show that $(1 - \frac{1}{2^{\lceil \log t \rceil + 1}})^t > \frac{1}{2}$, for $t \in N$. It suffices to consider t a power of 2, and in this case, the inequality is just $(1 - \frac{1}{2t})^t > \frac{1}{2}$. But an inductive proof of this fact is easy: $\underline{t = 2}$: $(\frac{3}{4})^2 > \frac{1}{2}$

Inductive step. If the inequality is true for t, show it for 2t.

$$(1 - \frac{1}{2(2t)})^{2t} = ((1 - \frac{1}{4t})^2)^t = (1 - \frac{1}{2t} + \frac{1}{16t^2})^t > (1 - \frac{1}{2t})^t > \frac{1}{2}.$$

$\square$

If T is any balanced binary tree, $t \geq 1$ any integer, let $\underline{T_t}$ denote the tree consisting of exactly the levels of T from 0 to $\lceil \log t \rceil + 1$ inclusive. Let $\underline{\phi_t}$ be defined on the leaves of T_t by $\phi_t(\ell) = \frac{1}{2^{\lceil \log t \rceil}}$ for all ℓ. If $s \in (R^+)^{\text{vert}(T)}$ is such that $s(v) = 0$ for all v at levels below $\lceil \log t \rceil + 1$, then $s_t \in (R^+)^{\text{vert}(T_t)}$ can be defined from s by simply ignoring the missing vertices. Then it is easy to see that $\text{expcost}_{\phi,t}(s) = \text{expcost}_{\phi_t,t}(s_t) + t(\log(n(T)) - (\lceil \log t \rceil))$. We then have the following.

Theorem 3.1.2. If $t > 1$, then $\text{minexpcost}_{\phi,t,\text{rootedge}(T)}(t) = \text{minexpcost}_{\phi_t,t,\text{rootedge}(T_t)}(t) + t(\log(n(T)) - (\lceil \log t \rceil))$.

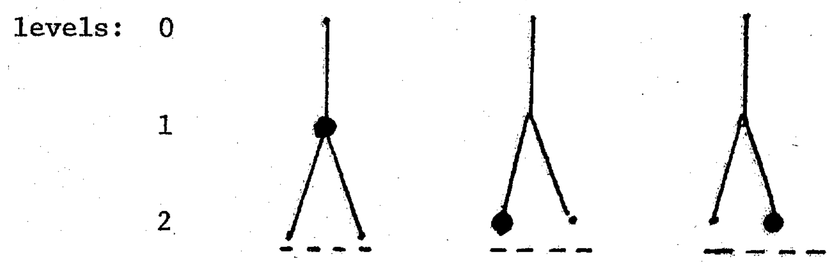
Proof. $\geq$ follows from Theorem 3.1.1 and the remarks.

$\leq$ follows because any placement for T_t can be augmented to a placement for T by placing zeros on the additional nodes, thereby incurring the stated cost.

$\square$

Thus, in the remainder of section 3 and in section 4, we assume that the number of levels in T is at most $\lceil \log t \rceil + 1$ (that is, that $n(T) < 2t$). The reader can then use Theorem 3.1.2 to infer corresponding results about cases where $n(T) \geq 2t$.

Example 3.1.1. The case where $t = 1$ is somewhat peculiar. The reader can verify that for all T with lowest level number at least 2, the following pictures all represent optimal placements.



That is, in the first case, $s(v) = \begin{cases} 1 & \text{for } v \text{ the son of the root,} \\ 0 & \text{otherwise,} \end{cases}$

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

This is the only value of t for which an optimal placement can have nonzero values below level $\lceil \log t \rceil + 1$.

Example 3.1.2. If T is balanced, $t = 2^k$, $n(T) > t$, and ϕ is uniform, then s which places 1 across the level with exactly t vertices 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 $\log t + 1$. But clearly putting 1 across all leaves is optimal for this subtree, as seen in Example 2.4.2.

3.2 Algorithms for Finding Optimal Placements

In this section, we prove two sharper versions of Theorem 2.2.4 for the special case of this section, and use them as bases for two algorithms for finding optimal placements.

Theorem 3.2.1. If $t \in N$, $u \in R^+$ and $k \in N$, $k \leq \log(n(T)) - 1$, then

$$\text{minexpcost}_{t,k}(u) = \text{expflow}_{t,k}(u) + 2\min_{u' \in \{0,1,\ldots,\lfloor\frac{u}{2}\rfloor,\frac{u}{2}\}} \text{minexpcost}_{t,k+1}(u').$$

Proof. $\leq$ is clear. We show $\geq$. Write f for $\text{minexpcost}_{t,k+1}$. Consider any u_1, $u_2 \in R^+$ with $u_1 + u_2 \leq u$ and $f(u_1) + f(u_2)$ minimal. We will produce $u' \in \{0,1,\ldots,\lfloor\frac{u}{2}\rfloor,\frac{u}{2}\}$ with $2f(u') \leq f(u_1) + f(u_2)$.

By Theorem 2.2.5, there exists $r \in N$ such that r is the smallest element of R^+ with $f(r) \leq f(s)$ for all $s \geq r$. We consider two cases.

Case 1. $u \geq 2r$.

Then if $u' = r$, it is clear that $2f(u') \leq f(u_1) + f(u_2)$.

Case 2. $u < 2r$.

Then $u_1 + u_2 = u$. Letting $u' = \frac{u_1+u_2}{2}$, we have that u' is in the required set; moreover, $2f(u') = 2f(\frac{u_1+u_2}{2}) \leq f(u_1) + f(u_2)$ by convexity of f. □

Theorem 3.2.2. For any $t \in N$, there exists s such that $s(v) = s(v')$ for all pairs of vertices v, v' at the same level, which is optimal for t.

Proof. Theorem 3.2.1 essentially provides a recursive algorithm.

□

We proceed as before to analyze the cost of determining an optimal placement with uniform values at each level. During the algorithm, one must calculate $\text{expflow}_{t,0}(t)$, $\text{expflow}_{t,1}(\frac{t}{2})$, ..., $\text{expflow}_{t,\log(n(T))}(\frac{t}{n(T)})$. In addition, one must calculate $\text{expflow}_{t,1}(i)$ for all $i \in N$, $i \leq \frac{t}{2}$, $\text{expflow}_{t,2}(\frac{i}{2})$ for all $i \in N$, $i \leq \frac{t}{2}$, $\text{expflow}_{t,3}(\frac{i}{4})$ for all $i \in N$, $i \leq \frac{t}{2}$, ..., $\text{expflow}_{t,\log(n(T))}(\frac{2i}{n(T)})$ for all $i \in N$, $i \leq \frac{t}{2}$.

This is a total of $O(t \log n(T))$ expflow computations. Since each such computation involves $O(t)$ arithmetic operations, we have the following theorem.

Theorem 3.2.3. There is an algorithm using $O(t^2 \log n(T))$ arithmetic operations which for any tree T (that is, satisfying the assumptions of this section), and for any $t \in N$, determines an s such that $s(v) = s(v')$ for all pairs of vertices v, v' at the same level, which is optimal for t.

Note that the bound of Theorem 3.2.3 represents an improvement over the bound in Theorem 2.3.2 applied to the special case of this section; the placements produced by the two algorithms have somewhat different properties, however. The remainder of this subsection deals with integral placements, the situation considered in Theorem 2.3.2.

Theorem 3.2.4. If $t, u, k \in N$, $k \leq \log(n(T)) - 1$, then

$$\text{minexpcost}_{t,k}(u) = \text{expflow}_{t,k}(u) + \min_{\substack{u_1,u_2 \in N \\ u_1 \le \lfloor \frac{u}{2} \rfloor, u_2 \le \lceil \frac{u}{2} \rceil, |u_2-u_1| \le 1}} (\text{minexpcost}_{t,k+1}(u_1) + \text{minexpcost}_{t,k+1}(u_2)).$$

Proof. Again, we show $\ge$.

Write $\underline{f}$ for $\text{minexpcost}_{t,k+1}$. By Theorem 3.2.1, there is a value

$u' \in \{0,1,\ldots,\lfloor \frac{u}{2} \rfloor, \frac{u}{2}\}$ such that $\text{minexpcost}_{t,k}(u) = \text{expflow}_{t,k}(u) + 2f(u')$.

If $u' \in \{0,\ldots,\lfloor \frac{u}{2} \rfloor\}$, then we simply take $u_1 = u_2 = u'$. If $u' = \frac{u}{2} \notin N$, then we take $u_1 = \lfloor \frac{u}{2} \rfloor$ and $u_2 = \lceil \frac{u}{2} \rceil$; in this case, piecewise linearity guarantees the required properties.

$\square$

Theorem 3.2.5. For any $t \in N$, there exists $s \in (N)^{\text{vert}(T)}$ which is optimal for t. Moreover, s has the additional properties:

(a) if e_1 and e_2 are two edges with $\text{high}(e_1) = \text{high}(e_2)$, then

$|\text{number}(e_1,s) - \text{number}(e_2,s)| \le 1$,

(b) if e is any edge with high(e) at level k, then $\text{number}(e,s) \le \lceil \frac{t}{2^k} \rceil$.

Proof. Theorem 3.2.4 essentially provides a recursive algorithm yielding a placement in $(N)^{\text{vert}(T)}$ and obviously satisfying (a). To see that (b) is also satisfied, assume the contrary, and let e be a highest edge in the tree with $\text{number}(e,s) > \lceil \frac{t}{2^k} \rceil$, where high(e) is at level k.

Since number(e,s) is integral, we have number(e,s) $\geq \left\lceil \frac{t}{2^k} \right\rceil + 1 \geq \frac{t}{2^k} + 1$.

e is not rootedge(T), so let e' be the edge $\neq$ e with high(e) = high(e'), and let e" be the edge immediately above e in T. Then number(e",s) $\leq \left\lceil \frac{t}{2^{k-1}} \right\rceil < \frac{t}{2^{k-1}} + 1$. Therefore, number(e',s) $\leq$

number(e",s) - number(e,s) $< (\frac{t}{2^{k-1}} + 1) - (\frac{t}{2^k} + 1) = \frac{t}{2^k}$.

But then |number(e,s) - number(e',s)| > 1, contradicting property (a).

□

Once again, we analyze the cost of determining an optimal placement with the properties of Theorem 3.2.5. One must calculate $\text{expflow}_{t,0}(t)$, and also $\text{expflow}_{t,1}(i)$ for all $i \in N$, $i \leq \left\lceil \frac{t}{2} \right\rceil$, $\text{expflow}_{t,2}(i)$ for all $i \in N$, $i \leq \left\lceil \frac{\left\lceil \frac{t}{2} \right\rceil}{2} \right\rceil = \left\lceil \frac{t}{4} \right\rceil$, ..., $\text{expflow}_{t,\log(n(T))}(i)$ for all $i \in N$, $i \leq \left\lceil \frac{t}{n(T)} \right\rceil$. This is a total of O(t) expflow computations, (since we are assuming that log(n(T)) is O(t)).

<u>Theorem 3.2.6</u>. There is an algorithm using $O(t^2)$ arithmetic operations, which for any tree T and for any $t \in N$, determines an $s \in (N)^{\text{vert}(T)}$ which is optimal for t, and which satisfies conditions (a) and (b) of Theorem 3.2.5.

3.3 A Fast Algorithm for Determining Optimal Placements

The results of §2.4 can be used to prune the algorithm's search space still further, leading to a much faster algorithm.

Theorem 3.3.1. Let $t, u, k \in N$, $k \le \log(n(T)) - 1$, $\left\lfloor \frac{t}{2^k} \right\rfloor \le u \le \left\lceil \frac{t}{2^k} \right\rceil$. Then

$$\text{minexpcost}_{t,k}(u) = \text{expflow}_{t,k}(u) + \min_{\substack{u_1,u_2 \in \{\lfloor \frac{t}{2^{k+1}} \rfloor, \lceil \frac{t}{2^{k+1}} \rceil\} \\ u_1+u_2 \le u}} (\text{minexpcost}_{t,k+1}(u_1) + \text{minexpcost}_{t,k+1}(u_2)).$$

Proof. Again, we show $\ge$. Write $\underline{f}$ for $\text{minexpcost}_{t,k+1}$.

By Theorem 3.2.4, there is a pair $u_1, u_2 \in N$ such that

$u_1 \le \left\lfloor \frac{u}{2} \right\rfloor$, $u_2 \le \left\lceil \frac{u}{2} \right\rceil$, $|u_2 - u_1| \le 1$ and $\text{minexpcost}_{t,k}(u) = \text{expflow}_{t,k}(u) +$

$f(u_1) + f(u_2)$. Assume $u_1 \le u_2$.

We must check that $u_1, u_2 \in \{\left\lfloor \frac{t}{2^{k+1}} \right\rfloor, \left\lceil \frac{t}{2^{k+1}} \right\rceil\}$. If $u_2 > \left\lceil \frac{t}{2^{k+1}} \right\rceil$, then

$u_2 \ge \left\lceil \frac{t}{2^{k+1}} \right\rceil + 1$, and also $u_1 \ge \left\lceil \frac{t}{2^{k+1}} \right\rceil$. Then $u \ge u_1 + u_2 \ge 2\left\lceil \frac{t}{2^{k+1}} \right\rceil + 1 > \left\lceil \frac{t}{2^k} \right\rceil$,

a contradiction. Thus, u_2 (and therefore u_1) $\le \left\lceil \frac{t}{2^{k+1}} \right\rceil$. On the other

hand, if $u_1 < \left\lfloor \frac{t}{2^{k+1}} \right\rfloor$, then $u_1 \le \left\lfloor \frac{t}{2^{k+1}} \right\rfloor - 1$, and also $u_2 \le \left\lfloor \frac{t}{2^{k+1}} \right\rfloor$.

Then $u - (u_1 + u_2) \ge \left\lfloor \frac{t}{2^k} \right\rfloor - (\left\lfloor \frac{t}{2^{k+1}} \right\rfloor - 1 + \left\lfloor \frac{t}{2^{k+1}} \right\rfloor) \ge 1$. Define a new decomposition

w_1, w_2 by $w_2 = u_2$, $w_1 = u_1 + 1$. Then $w_1 + w_2 \le u$. Now,

$\text{expflow}_{t,k+1}(w_1) < \text{expflow}_{t,k+1}(u_1)$ because $u_1 \leq \left\lfloor \frac{t}{2^{k+1}} \right\rfloor - 1$, by Theorems 2.4.1 and 2.4.3. Thus, $f(w_1) < f(u_1)$, by Theorem 2.2.4. (The first term in the sum of that theorem is less for argument w_1 than for argument u_1, and the minimum in the second term is taken over a larger possible range in the case of w_1.) But then the minimality of u_1, u_2 is contradicted. □

Theorem 3.3.2. For any $t \in N$, there exists a fair whole placement s which is optimal for t.

Proof. Theorem 3.3.1 yields a recursive algorithm.

□

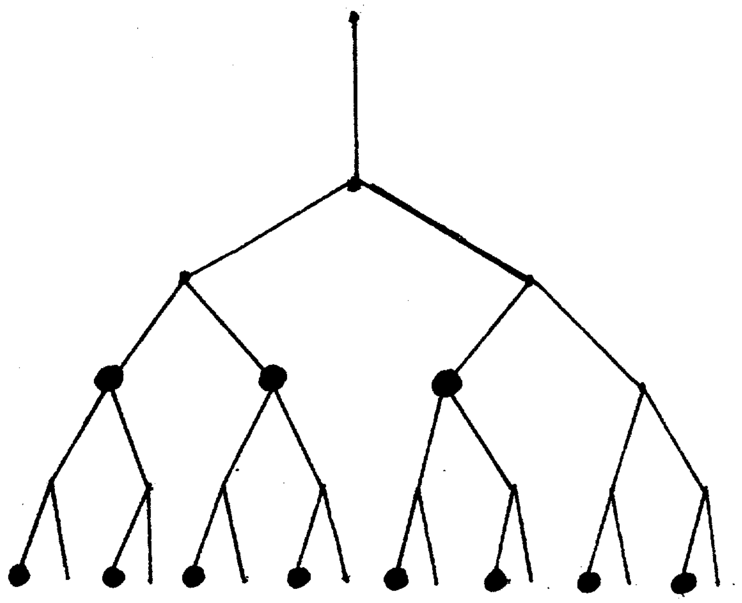
Note that a result similar to Theorem 3.3.2 does not hold in the general case-recall Example 2.4.1.

The algorithm resulting from Theorem 3.3.2 is extremely fast. Namely, one must calculate $\text{expflow}_{t,k}(i)$ for $i = \left\lfloor \frac{t}{2^k} \right\rfloor, \left\lceil \frac{t}{2^k} \right\rceil$, for each k, $1 \leq k \leq \log(n(T))$, for a total of only $O(\log(n(T)))$ expflow computations.

Theorem 3.3.3. There is an algorithm using $O(t \log(n(T)))$ arithmetic operations which for any tree T and for any $t \in N$, determines a fair whole placement s which is optimal for t.

3.4 Example

Some of the optimal placements discovered by our algorithms are rather unexpected. For example, the following represents an optimal placement of 11 resources in a balanced binary tree with uniform probability distribution:



4. Bounds on the Costs of Optimal Placements

In this section, we assume that T is a balanced binary tree (except for rootedge(T)) but ϕ is arbitrary. We show that optimal placements are much better than centralized placements; in fact, their expected cost is linear in the number of leaves of the tree. We continue to assume that $n(T) < 2t$, relying on the reader to infer results about the case where $n(T) \geq 2t$.

4.1 Cost of Centralized Placement

If $s(v) = 0$ everywhere except at low(rootedge(T)), $s(\text{low}(\text{rootedge}(T))) = t$, then $\text{expcost}_{\phi,t}(s) = t \log(n(T))$.

4.2 Cost of Fair Whole Placements

Since we do not have a direct characterization of optimal placements, we instead bound the expected cost of an arbitrary fair whole placement. This upper bound, of course, provides an upper bound for optimal placements. Moreover, by Theorem 2.4.4, the costs cannot differ by more than $2 \cdot n(T)$.

Define G(n,t) to be

$$\max_{\substack{\phi \\ T:n(T)=n \\ s \text{ a fair whole placement for } T,\phi,n}} \text{expcost}_{\phi,t}(s).$$

We establish a recurrence for G, namely:

$$(*) \quad \begin{cases} G(n,t) \leq \max_{t_1+t_2\leq t}\left(G(\frac{n}{2},t_1) + G(\frac{n}{2},t_2)\right) + 12\sqrt{t}\log n, \text{ for } n \geq 2, \\ G(1,t) = 0. \end{cases}$$

We first require some technical lemmas.

Lemma 4.2.1. For $t \in N$, $t \geq 1$, $s \leq t - 1$, $0 \leq p \leq 1$, it is the case that

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

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

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

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

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

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

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

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

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

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

$$= t\binom{t-1}{s}p^{s+1}(1-p)^{t-s}.$$

□

Lemma 4.2.2. If $t \in N$, $t \geq 1$, $0 \leq p < 1$, it is the case that

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

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

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

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

$$= 2t\binom{t-1}{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor+1} - \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp-i) \text{ by Lemma 4.2.1.}$$

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

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

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

Lemma 4.2.3. For $t \in N$, $t \geq 1$, $tp \geq 1$, it is the case that

$$\binom{t-1}{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor} \leq \left(1 + \frac{1}{4t}\right) \cdot \frac{e}{\sqrt{2\pi \lfloor tp \rfloor}} .$$

Proof. A version of Stirling's formula says that for all $n \in N$, it is the case that

$$\left(\frac{n}{e}\right)^n \sqrt{2\pi n} \leq n! \leq \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \left(1 + \frac{1}{4n}\right).$$

Then $\binom{t-1}{\lfloor tp\rfloor} = \frac{t-\lfloor tp\rfloor}{t} \cdot \binom{t}{\lfloor tp\rfloor}$

$$\le \left(\frac{t-\lfloor tp\rfloor}{\sqrt{t}}\right) \cdot \left(1 + \frac{1}{4t}\right) \cdot \left(\frac{t^t}{\sqrt{2\pi\lfloor tp\rfloor(t-\lfloor tp\rfloor)}\cdot\lfloor tp\rfloor^{\lfloor tp\rfloor}(t-\lfloor tp\rfloor)^{t-\lfloor tp\rfloor}}\right) .$$

Therefore,

$$\binom{t-1}{\lfloor tp\rfloor}(1-p)^{t-\lfloor tp\rfloor}p^{\lfloor tp\rfloor}$$

$$\le \left(\frac{t-\lfloor tp\rfloor}{\sqrt{t}}\right)\left(1 + \frac{1}{4t}\right)\left(\frac{t-tp}{t-\lfloor tp\rfloor}\right)^{t-\lfloor tp\rfloor}\left(\frac{tp}{\lfloor tp\rfloor}\right)^{\lfloor tp\rfloor}\left(\frac{1}{\sqrt{2\pi\lfloor tp\rfloor(t-\lfloor tp\rfloor)}}\right)$$

$$= \sqrt{\frac{t-\lfloor tp\rfloor}{2\pi t\lfloor tp\rfloor}} \cdot \left(1 + \frac{1}{4t}\right) \cdot \left(\frac{t-tp}{t-\lfloor tp\rfloor}\right)^{t-\lfloor tp\rfloor}\left(\frac{tp}{\lfloor tp\rfloor}\right)^{\lfloor tp\rfloor} .$$

But

$$\left(\frac{tp}{\lfloor tp\rfloor}\right)^{\lfloor tp\rfloor} = \left(1 + \frac{tp-\lfloor tp\rfloor}{\lfloor tp\rfloor}\right)^{\lfloor tp\rfloor} \le \left(1 + \frac{1}{\lfloor tp\rfloor}\right)^{\lfloor tp\rfloor} \le e,$$

$\frac{t-\lfloor tp\rfloor}{t} \le 1$ and $\left(\frac{t-tp}{t-\lceil tp\rceil}\right)^{t-\lceil tp\rceil} \le 1$. The lemma follows.

□

It is obvious that $G(1,t) = 0$. Now fix $n \geq 2$, t, and consider any T with $n(T) = n$, any ϕ, and s any fair share whole placement for T, ϕ, n. Let e_1 and e_2 be the immediate descendant edges of rootedge(T), $p = p_{e_1}$ and $q = 1 - p = p_{e_2}$. Now the expected value of $|tp - \text{number}(e_1,r)|$, $\text{Exp}_r|tp - \text{number}(e_1,r)|$, is equal to $\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |tp-i|$,

$= 2t\binom{t-1}{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor + 1}$ by Lemma 4.2.2, provided $p < 1$. If $p = 1$, then $\text{Exp}_r|tp - \text{number}(e_1,r)| = \text{Exp}_r|t - t| = 0$.

But $2t\binom{t-1}{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor + 1} \leq 2tp(1 + \frac{1}{4t}) \cdot \left(\frac{e}{\sqrt{2\pi \lfloor tp \rfloor}}\right)$ if $tp \geq 1$, which is at most $2 \cdot (\frac{5}{4}) \cdot \frac{e}{\sqrt{\pi}} \cdot \frac{tp}{\sqrt{tp}} < 4\sqrt{t}$. On the other hand, if $tp < 1$, then $2t\binom{t-1}{\lfloor tp \rfloor}(1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor + 1} = 2tp(1-p)^t < 2$. In any case, the expectation is less than $4\sqrt{t}$.

Let $\underline{T_i}$ denote the subtree of T headed by e_i, and $\underline{t_i}$ denote number(e_i,s), $i = 1,2$. Let $\underline{s_i}$ denote s restricted to the nodes of T_i, modified slightly by setting $s_i(\text{high}(\text{rootedge}(T_i))) = 0$. Let ϕ_1 be defined on leaves(T_1) by $\phi_1(\ell) = \frac{1}{p}\phi(\ell)$, and ϕ_2 on leaves(T_2) by $\phi_2(\ell) = \frac{1}{q}\phi(\ell)$.

Now consider any particular sequence $L = \ell_1, \ell_2, \ldots, \ell_t$ of elements of leaves(T). This sequence determines $r \in N^{\text{leaves}(T)}$ with total$(r) = t$, as before. We will bound cost(r,s). Let $L_i = \ell_{i,1}, \ell_{1,2}, \ldots, \ell_{i,a(i)}$, $i = 1, 2$, be the subsequence of L with elements in leaves(T_i), where $a(i) =$ number(e_i,r). Let $\overline{L}_i = \begin{cases} \ell_{i,1},\dots,\ell_{i,t_i} & \text{if } a(i) \geq t_i, \\ \ell_{i,1},\dots,\ell_{i,a(i)},d_{i,1},\dots,d_{i,t_i-a(i)} & \text{if } a(i) < t_i, \end{cases}$

$i = 1,2$, where the $d_{i,j}$ are chosen independently, at random, according to ϕ_i. That is, we truncate the sequence to length t_i if it is too long, and pad it out randomly if it is too short.

$\overline{L}_i$ determines $r_i \in N^{\text{leaves}(T_i)}$, $i = 1,2$. We claim that $\text{cost}(r,s) \leq \text{cost}(r_1,s_1) + \text{cost}(r_2,s_2) + (b_1+b_2) \cdot (2 \log n)$, where $b_i = a(i) - t_i$ if $a(i) > t_i$, 0 otherwise, $i = 1,2$. (This is because the requests represented by $\ell_{i,1},\dots,\ell_{i,t_i}$ can be matched to resources in a way which caused flow only within subtree T_i, $i = 1,2$. Each excess request can cause additional flow of at most 2 log n, that is, 1 on each of 2 log n edges, even if matching occurs at the greatest possible distance in the tree.)

We now take expectations over all L, assuming L is constructed from t independent trials using ϕ. We obtain:

$$\text{expcost}_{\phi,t}(s) \leq \text{expcost}_{\phi_1,t_1}(s_1) + \text{expcost}_{\phi_2,t_2}(s_2) + (\text{Exp}_r(b_1+b_2)) \cdot 2 \log n.$$

$$\leq \max_{t_1+t_2\leq t}(G(\tfrac{n}{2},t_1) + G(\tfrac{n}{2},t_2)) + (\text{Exp}_r(b_1+b_2)) \cdot 2 \log n.$$

It remains to bound the expectations.

Since $\lfloor tp \rfloor \le t_1 \le \lceil tp \rceil$, we have $|t_1 - tp| \le 1$, and, similarly, $|t_2 - tq| \le 1$. Letting $c_1 = a(1) - tp$ if $a(1) > tp$, 0 otherwise, and $c_2 = a(2) - tq$ if $a(2) > tq$, 0 otherwise, we see that $c_1 + c_2 = |tp - \text{number}(e_1,r)|$. But also, since $|t_1 - tp| \le 1$ and $|t_2 - tq| \le 1$, we see that $|b_i - c_i| \le 1$, $i = 1,2$. Then $b_1 + b_2 \le c_1 + c_2 + 2 \le |tp - \text{number}(e_1,r)| + 2$. Thus, $\text{Exp}_r(b_1 + b_2) \le \text{Exp}_r|tp - \text{number}(e_1,r)| + 2 < 4\sqrt{t} + 2 \le 6\sqrt{t}$. Thus, we have the recurrence (*).

We solve (*). Expanding and grouping like terms, and setting $k = \log n$, $x = \frac{1}{\sqrt{2}}$, $t = t_\lambda$ (λ the empty string), we obtain:

$$G(n,t) \le 12\Big[\sqrt{t_\lambda}\, k + (\sqrt{t_1} + \sqrt{t_2})(k-1) + (\sqrt{t_{11}} + \sqrt{t_{12}} + \sqrt{t_{21}} + \sqrt{t_{22}})(k-2)$$

$$+ \Big(\sum_{i_1,i_2,i_3 \in \{1,2\}} \sqrt{t_{i_1 i_2 i_3}}\Big)(k-3) + \dots$$

$$+ \Big(\sum_{i_1,\dots,i_{k-1} \in \{1,2\}} \sqrt{t_{i_1,\dots,i_{k-1}}}\Big)(1)\Big] + \sum_{i_1,\dots,i_k \in \{1,2\}} G(1, t_{i_1,\dots,i_k}),$$

for some choice of the t's, with each $t_{i_1,\dots,i_j,1} + t_{i_1,\dots,i_j,2} \le t_{i_1,\dots,i_j}$. But the last term of the sum is 0. Also, each summation

$\sum_{i_1,\dots,i_j \in \{1,2\}} \sqrt{t_{i_1,\dots,i_j}} \le 2^j \sqrt{\frac{t}{2^j}}$, because the square root function is concave and $\sum_{i_1,\dots,i_j \in \{1,2\}} t_{i_1,\dots,i_j} \le t$.

Thus, $G(n,t) \le 12\left[\sqrt{t}\ k + 2\sqrt{\frac{t}{2}}(k-1) + 2^2\sqrt{\frac{t}{2^2}}(k-2) + 2^3\sqrt{\frac{t}{2^3}}(k-3) + \ldots + 2^{k-1}\sqrt{\frac{t}{2^{k-1}}}(1)\right]$

$$= 12\sqrt{t}\left[k + \sqrt{2}\ (k-1) + (\sqrt{2})^2(k-2) + (\sqrt{2})^3(k-3) + \ldots + (\sqrt{2})^{k-1}(1)\right]$$

$$= \frac{12\sqrt{t}\ \sqrt{n}}{\sqrt{2}}\left[k\ x^{k-1} + (k-1)\ x^{k-2} + (k-2)\ x^{k-3} + \ldots + (1)\ x^0\right]$$

The expression in brackets is just $\frac{d(x^k + x^{k-1} + \ldots + x)}{dx} = \frac{d\left(x \cdot (\frac{1-x^k}{1-x})\right)}{dx}$

$$= x\left(\frac{(1-x)(-kx^{k-1}) - (1-x^k)(-1)}{(1-x)^2}\right) + \left(\frac{1-x^k}{1-x}\right)$$

$$= \frac{1 + kx^{k+1} - (k+1)x^k}{(1-x)^2}\ .$$

Thus, $G(n,t) \le \frac{12\sqrt{t}\ \sqrt{n}}{\sqrt{2}}\left(\frac{1 + \frac{\log n}{\sqrt{2n}} - \frac{\log n + 1}{\sqrt{n}}}{\left(1-\frac{1}{\sqrt{2}}\right)^2}\right)$

$$= \frac{12}{\sqrt{2}\left(1-\frac{1}{\sqrt{2}}\right)^2}\ \sqrt{t}\ \sqrt{n}\left(1 - \frac{1}{\sqrt{n}} - \frac{(\sqrt{2}-1)\log n}{\sqrt{2n}}\right)$$

$$\le \frac{12}{\sqrt{2}\left(1-\frac{1}{\sqrt{2}}\right)^2}\ \sqrt{t}\ \sqrt{n} = \frac{12\sqrt{2}}{3-2\sqrt{2}}\sqrt{t}\ \sqrt{n} = (36\sqrt{2} + 48)\sqrt{t}\ \sqrt{n}\ .$$

Thus, $G(n,t)$ is $O(\sqrt{t}\ \sqrt{n})$.

Therefore, we obtain a function which is $O(\sqrt{t}\ \sqrt{n(T)})$ as the expected cost for any fair share whole placement. We compare this with the centralized case. Since $n(T)$ is $O(t)$, this cost is always $O(t)$, so is better than the centralized case. If $n(T) \tilde{-} t$, we have an improvement of $O(t)$ vs. $O(t \log t)$. The improvement is even greater if $t >> n(T)$, approaching $O(\sqrt{t})$ vs. $O(t)$ as t increases for fixed $n(T)$.

Acknowledgements: The authors thank Carl Spruill, Charles Blair and Mike Paterson for contributing their ideas and suggestions for some of the results in this paper.

References

[FGL] Fischer, M., Griffeth, N. and Lynch, N. Work in progress.

[U] Uhlmann, V.W.,"Vergleich der hypergeometrischen mit der Binomial-Verteilung," Metrika, 10, 145-148 (1966).