Comput. Opns Res. Vol. 14, No. 5, pp. 349–361, 1987
Printed in Great Britain. All rights reserved

0305-0548/87 $3.00 + 0.00
Copyright © 1987 Pergamon Journals Ltd

THE FILE ALLOCATION PROBLEM—A QUEUEING NETWORK OPTIMIZATION APPROACH

M. M. SRINIVASAN*
Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109-2117, U.S.A.
and
K. KANT†
Department of Computer Science, The Pennsylvania State University, University Park, PA 16802, U.S.A.

(*Received May* 1986; *revised February* 1987)

Scope and Purpose—A problem which arises in the design of a distributed information processing system is the question of where the data files, which are shared by a number of users in remote locations in the system, are to be located so that some desired objective is met. This problem has received considerable attention in recent literature and is commonly known as the File Allocation Problem. A comprehensive review of the problem, and the approaches taken in addressing the same, is given by Dowdy and Foster [*Computing Surveys* **14**(2), 287–313 (1982)]. The file allocation problem has been shown to be a complex one. Obtaining the optimal allocation would probably require an exhaustive search of all possible candidates. The objective of this paper is to obtain an allocation which minimizes the mean response time for requests made by the users of the system. Schemes are developed to obtain near optimal allocations in a reasonably short period of time.

Abstract—A set of customers use a connected network of computer installations, each accessing the network from a particular node. These customers share information contained in a set of data files. A typical customer's need is characterized by a request requiring a subset of these files being accessed in a Markovian sequence. The cycle time for this customer is the total time taken on the average to complete his request sequence. The objective is to locate a single copy of each of the files in such a way that a weighted sum of these response times is minimized. The problem is modelled as a Closed Queueing Network optimization problem. Models are developed for both single and multiple chain cases. An incremental analysis approach is used to solve the single chain case. For the multiple chain case, it is shown how this model approximates to a set partitioning problem under certain conditions. Efficient heuristics are developed to solve this partitioning problem. With certain simplifying assumptions, the associated communication problem is then included in the model.

1. INTRODUCTION

The File Assignment Problem (FAP) has generally been recognized as crucial to the design of a good distributed information system. It has hence received a considerable amount of attention in the literature since the time it was first investigated by Chu [1]. The FAP entails allocating a set of F distinct files among a set of M computer installations (nodes). The allocation is to be made so as to optimize some objective function.

An example of the FAP is in the allocation of the public access files (such as compilers, library routines, etc.) among the nodes of a campus wide network of mini and micro computers. At issue here are decisions such as how many copies of each file is needed, and where they should be placed so as to obtain the desired objective. At one extreme, one could have a copy of each file placed at each node in the network. Apart from the feasibility of such a scheme owing to limited storage capacities at some of the nodes, this would involve considerable effort in maintaining file integrity when the number of updates to these files is large. At the other extreme, one could have a single copy of each file, placed somewhere in the network. One option here would be to maintain a centralized data base, accessed

*M. M. Srinivasan received the Master of Technology degree from the Indian Institute of Technology, Madras, and the Ph.D. degree in Industrial Engineering and Management Sciences from Northwestern University. He is currently an Assistant Professor in the Industrial and Operations Engineering Department at the University of Michigan, Ann Arbor, Michigan. His research interests are in performance evaluation and distribution data processing systems.

†Krishna Kant received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, the Master of Engineering degree in electrical communication engineering from the Indian Institute of Science, Bangalore, and the Ph.D. degree in computer science from the University of Texas, Dallas. He is currently an Assistant Professor of Computer Science at Pennsylvania State University, University Park. His research interests are in the areas of computational geometry, fault-tolerant computing, performance evaluation, and operating systems.

by every node in the system. This would, however, be poor both in terms of reliability and performance. Hence, it is clear that several conflicting issues need to be resolved in the FAP.

The "file" in the FAP need not be a file in the conventional sense. For example, consider the problem of assigning tasks to processors in multiprocessor systems. With the introduction of large parallel machines, and advances in programming to take advantage of such machines, the problem of task allocation (and, possibly, reallocation or migration) is becoming more important. The tasks involved cooperate in order to solve the problem. Thus one could view the subproblems to be solved as transactions that are served by the processes residing on various processors. In this sense, the tasks are like "files" accessed by the control flowing through the system. As another example, consider an expert system implemented on a parallel machine. Apart from the problems of parallel execution of logic programs, which is a topic of current research, the problem of partitioning and allocating the rule base and factual data base are also worth considering. Thus in this allocation problem, the database fragments become "files" and the inference tasks operating on them become the requests.

Work on the file allocation has proceeded in two directions. One direction is towards formulating the optimization problem as one which minimizes some cost function typically consisting of file storage costs, and/or communication costs [1–9]. The solution here is usually based on solving a constrained 0–1 integer programming problem with either a linear or a non-linear objective function. The other direction is based on formulating the problem as one which optimizes the performance (e.g. the overall response time/throughput) of the system. Here, the problem is modelled as one of optimizing a queueing network of single servers [1, 10–15]. The queueing networks considered here are assumed to have the product form (PF) property [16].

Both approaches have their advantages and limitations. A major limitation in the first approach is that delays due to queueing are either ignored or not realistically modelled. In the second approach, costs of communication and storage are typically not considered.

In this paper we take the second approach and model the FAP as a queueing network optimization problem, with the objective of minimizing overall mean response times to customer requests.

1.1. The FAP modeled as a queueing network optimization problem

To see how the FAP gives rise to a queueing network optimization problem, consider a network of service facilities (nodes) used by a number of customers. For example, the nodes could be the computers in an Apollo ring, and the customers could be the processes running on each such computer. (This is an example of the FAP in the conventional sense.) These customers share information contained in a set of distinct files. Suppose that a single copy of each file is to be allocated among the nodes in the network. In the context of the Apollo ring network again, there could be a number of file servers which could store the files. (These file servers operate in a demand paged environment and retrieve one or more pages of information for transmission across the network to the processes making the requests.) Since it is likely that several customers could, at the same time, require information from one or more files stored at a node, some of the requests have to queue up for service.

We may consider a customer request as accessing one or more of these files in some sequence. The result of a request typically generates some information which is to be transmitted back to the customer originating the request. There can also be some information being transmitted between files in the request sequence. If these files are located on different nodes one must also consider transmission times and possible queueing delays at the communication servers. Let the mean time taken to complete a typical request sequence be called the "cycle time". The cycle time depends on the allocation and thus we want to find an allocation that gives the minimum cycle time. For tractability, it is usually assumed that the queueing network models of the system satisfy the PF property.

1.2. Previous work on performance models of the FAP

One of the earliest works on performance models was the paper by Chen [11], where the problem was posed as an open network optimization problem with a single customer chain (i.e. a single request type). The files, however, were allowed to be split among the nodes. Closed single chain network models which again allowed non-integral assignment of files were considered by Trivedi *et al.* [14, 15], and Geist and Trivedi [13]. In many cases, however, integral assignments of files are much more realistic. A model by Foster *et al.* [12] considered a single chain closed queueing network optimization problem where only integral assignments were allowed. However, their solution method uses a complex two-stage iterative approach, alternately solving a non-linear program in one stage and an integer program in the other. It appears unlikely that the solution scheme could effectively be applied to a reasonably sized problem.

The above methods generally restrict analysis to a central server network, where only one node makes all the demands. None of the above models considered communication times/delays. A model by Bryant and Agre [10] considers a more general closed model with multiple chains (i.e. multiple request types). It also incorporates communication delays. The solution method outlined is a heuristic which, starting with an initial arbitrary allocation, considers moving a file to all other nodes in order to obtain the best location for it. For each hypothesized move, the resulting queueing network is solved for the cycle times for each customer chain using the approximate algorithm of Schweitzer [17]. This scheme is iteratively repeated with each file till no further improvement is noticed. While this model is the most comprehensive performance model of the FAP among those reviewed above, the solution method is a simple enumeration technique which, further, need not find the optimal solution.

1.3. Research purpose

In this paper we develop heuristics that can substantially cut down the complexity of determining a near optimal integral allocation. As noted earlier, the objective here is to minimize the mean overall response times to customer requests. In general, there may be several types of requests for file accesses. We assume that the models are closed, i.e. for each type of request, a fixed number of statistically equivalent requests circulate endlessly in the network. The problem is then modelled, as a closed queueing network optimization problem with multiple chains, where each chain represents a request type and each station corresponds to a node in the system. While closed models are more difficult to deal with than open models, they are more realistic.

The rest of this paper is organized as follows: in Section 2 the model of the FAP is developed. Sections 3 and 4 then consider algorithms for the single and multiple chain versions of the FAP. Section 5 indicates how communication delays can be modelled, and Section 6 presents the conclusions.

2. THE MODEL OF THE FAP

Let there be M nodes in the problem and R chains. A total of F different files are to be allocated among these nodes. Each customer starts at a designated node, accesses a subset of F files according to a Markovian sequence, and returns to the originating node. Processing each file in the sequence demands a certain amount of work from the system. In particular, processing file f for customer type r takes $T_{r,f}$ operations. Each node m executes at a certain rate S_m, which is the number of operations executed per second. The amount of time demanded by customer r from the file f, which we call the mean file service time demand on file f by the rth chain, if this file is placed in node m, is then $T_{r,f}/S_m$.

For modelling convenience, a customer making requests from node m is assumed to access a dummy file Z_m as the last file in his sequence. There are M such files, one for each node and we shall term these files "sentinel files". Apart from performing a policing duty, these files are also convenient to model any local computation that is performed when the customer's request returns to the node from which it originated.

Let $x_{m,f}$ be a 0–1 variable which is set to 1 if file f is allocated to node m and is set to 0 otherwise.

Every allocation $X = \{x_{m,f}\}$ generates a mean service time demand, $L_{mr}(X)$, on the nodes for each type of customer. This is the sum of the various mean file service time demands by customer type r for the files placed at node m, viz.

$$L_{mr}(X) = \sum_{f=1}^{F} x_{m,f} \cdot T_{r,f}/S_m. \tag{1}$$

Let $L_r(X)$ represent the total mean service demand by the rth customer chain from the network. Then,

$$L_r(X) = \sum_{m=1}^{M} L_{mr}(X). \tag{2}$$

Let $W_r(N, X)$ denote the cycle time of chains of customers when network population vector is N and allocation vector is X.

The optimization problem can now be written as:

Problem 1

$$\min \sum_{r=1}^{R} \beta_r \cdot W_r(N, X)$$

$$\text{s.t.} \quad \sum_{m=1}^{M} x_{m,f} = 1; \qquad f = 1, 2, \ldots, F \text{ (single copy only)} \tag{3}$$

where

$$x_{m,f} \in \{0, 1\} \qquad \{\text{integral assignments only}\}.$$

Here, $\beta_1, \ldots, \beta_R$ are specified weights.

3. THE FAP FOR A SINGLE CUSTOMER CHAIN

In this section we consider the special case where there is only one request type, i.e. a single chain in the model. This special case is important owing to the fact that the cycle time in such a network is a monotonic function of the nodal service times and population. We will drop the chain subscript from all quantities in this case.

For the single chain case, the FAP is:

Problem 2

Minimize $W(N, X)$ subject to

$$\sum_{m=1}^{M} x_{m,f} = 1; \qquad f = 1, \ldots, F,$$

$$x_{m,f} \in \{0, 1\}; \qquad f = 1, \ldots, F;\ m = 1, \ldots, M.$$

Assuming negligible communication delays, the Mean Value Analysis (MVA) algorithm [18] for single server nodes yields:

$$W(N, X) = \sum_{m=1}^{M} L_m(X) + \sum_{m=1}^{M} L_m(X) \cdot Q_m(N-1, X), \tag{4}$$

where $L_m(X)$ is defined by equation (1), and $Q_m(N-1, X)$ is the mean queue length that would form at node m with $N-1$ customers in the system, and with allocation X.

The cycle time $W(N, X)$ is a complex, non-linear function of the allocation variables and can be computed with $O(MN)$ operations by known iterative algorithms [16]. It is a convex function of the allocation variables [14], and hence the non-linear integer program, Problem 1 could be solved using a branch and bound procedure for small problems. As the number of variables increase, straightfoward application of branch and bound algorithms could require a lot of computational effort, and this motivates consideration of some approximate solution techniques.

A heuristic interchange search technique is proposed, and this operates as follows: it starts with an allocation which attempts to balance the loads among the nodes as evenly as possible. This initial assignment is made by assigning each file, in turn, to a node such that the loads are maintained as uniformly distributed as possible among the nodes after each assignment. Once this initial allocation, X_0, is made, the cycle time for this allocation is evaluated. The heuristic then proceeds as follows: each possible combination of node-pairs are considered in turn. The files present in a node-pair are then examined to determine whether moving a file from its present node to the other (a SHIFT operation), or a pairwise exchange of files between nodes (a SWAP operation) would reduce the cycle time. With F files and M nodes, $F(M-1)$ possible SHIFTs and at most $F(F-1)$ possible SWAPs are examined. Assuming $F > M$, thus $O(F^2)$ such possibilities are examined. The cycle time which would result from each such operation is evaluated, and the lowest of such times, $W(X_1)$, corresponding to an allocation X_1, is compared with $W(X_0)$. If $W(X_1) > W(X_0)$, the search terminates with X_0 as the best allocation found; otherwise the allocation X_1 is chosen, and this completes one iteration of the search. The next iteration now tries to find an assignment X_2 which has a cycle time less than that of assignment X_1. This process continues till no further decrease in cycle time is possible. Obviously the iterations must terminate, but results obtained need not be optimal.

For each potential SHIFT or SWAP operation considered, the exact evaluation of cycle time takes $O(MN)$ operations as noted earlier. This means that each iteration of the search could take $O(MNF^2)$ operations. We now describe a heuristic for the FAP which obtains good estimates of the cycle time for each candidate allocation in an iteration with much less computation, thereby improving the efficiency of the search.

3.1. Incremental analysis

This approach evaluates the cycle time exactly once at the start of each iteration. Each candidate allocation in the search process is now, however, evaluated with $O(1)$ computations. This is based on incremental analysis which is described below:

Suppose that at the start of some iteration k, the allocation X_{k-1} is known. Then the loads generated at the nodes, L_m, $m = 1, \ldots, M$, and the resulting cycle time $W(X_{k-1})$, can be determined for this allocation. The heuristic then examines each file to determine the effect of a reassignment due to either a SHIFT operation or a SWAP operation. If this reassignment was done, it would change the loads at nodes i and j by some amounts ΔL_i and ΔL_j, leaving the loads at all other nodes unchanged. The resulting cycle time for this reassignment X'_k is then related to $W(X_{k-1})$ by the Taylor series:

$$W(X'_k) = W(X_{k-1}) + \left(\Delta L_i \frac{\partial}{\partial L_i} + \Delta L_j \frac{\partial}{\partial L_j}\right) W(X_{k-1}) + \frac{1}{2!}\left\{\Delta L_i \frac{\partial}{\partial L_i} + \Delta L_j \frac{\partial}{\partial L_j}\right\}^2 W(X_{k-1}) + \ldots \qquad (5)$$

and, for reasonably small ΔL_i, ΔL_j, we can drop the higher order terms.

For notational convenience, we shall write W to mean $W(X_{k-1})$. For the allocation X_{k-1}, let $Q_n(N)$ denote the mean queue length at node n, $n = 1, \ldots, M$, at population N, and let λ_N be the corresponding throughput of the network. Let

$$U_i(N) = \lambda_N L_i \qquad (6)$$

and

$$\Delta Q_i(N) = Q_i(N) - Q_i(N-1). \qquad (7)$$

Lemma 1

$$\frac{\partial}{\partial L_i} W = \frac{N}{L_i \lambda_N} \Delta Q_i(N). \qquad (8)$$

A proof of Lemma 1 is given in Ref. [19].

Proposition 1

$$\frac{\partial^2}{\partial L_i^2} W = t_1 + t_2, \qquad (9)$$

where

$$t_i \approx \frac{N}{U_i(N)} \left\{ \frac{[Q_i(N)]^2}{L_i U_i(N)} \left[\frac{1 - U_i(N)}{1 - U_i(N)(N-1)/N} \right] - \frac{[Q_i(N-1)]^2}{L_i U_i(N-1)} \left[\frac{1 - U_i(N)}{1 - U_i(N-1)(N-2)/(N-1)} \right] \right\} \qquad (10)$$

and

$$t_2 = \frac{N}{U_i(N)}\left[\frac{Q_i(N)}{L_i}(1 - U_i(N))\Delta Q_i(N)\right]. \tag{11}$$

The derivation of this approximate expression is given in Ref. [19]. Using arguments similar to that used for Proposition 1, it is possible to obtain an approximate expression for the partial derivative of W with respect to L_i and L_j and this expression is given by Proposition 2. The details of the derivation are omitted.

Proposition 2

$$\frac{\partial^2}{\partial L_i \delta L_j} W = t_3 + t_4, \tag{12}$$

where

$$t_3 \approx -\frac{N}{\lambda_N L_i L_j}\left[\frac{Q_i(N)\Delta Q_j(N)}{1 - U_i(N)(N-1)/N} - \frac{Q_i(N-1)\Delta Q_j(N-1)}{1 - U_i(N-1)(N-2)/(N-1)}\right] \tag{13}$$

and

$$t_4 = \frac{N}{\lambda_N L_i L_j}\Delta Q_i(N)\Delta Q_j(N). \tag{14}$$

3.2. Experimental results

Table 1 tabulates the results of using the incremental approach on some randomly generated problems. Results presented are for some typical problems, indicating the improvement in cycle times resulting from the search over that obtained by the initial load balancing allocation. For illustrating the effectiveness of the heuristic, the values of the cycle time obtained for the initial allocation were scaled to a 100.00 with the cycle time for the best allocation found by the heuristic being correspondingly scaled. For test cases where the number of nodes, files and customer types were reasonably small, the optimum allocation was evaluated by an exhaustive search. For these cases, the cycle times corresponding to the optimal allocation, appropriately scaled, is also indicated in Table 1. In general, the improvements ranged from 0 to 43%, with the most significant improvements occurring when the network populations were small. The tests were carried out using a VAX 11/750 machine running the VMS operating system. Table 1 indicates the time taken by the heuristic and the exhaustive search to obtain these allocations.

In general, in the absence of communication delays, and when the customer populations are large, the FAP for the single chain case generally reduces to a problem of balancing the loads among the

Table 1. Result of some test problems for single chain case

Problem number	Size of problem (nodes, files, customers)	Minimum found		Time taken (s)	
		Heuristic	Exhaustive	Heuristic	Exhaustive
1	2, 5, 12	100.00	100.00	0.02	0.23
2	3, 4, 8	95.42	95.42	0.04	0.50
3	2, 10, 6	95.74	95.14	0.08	4.46
4	3, 6, 10	97.36	97.16	0.05	5.27
5	3, 7, 5	93.20	92.79	0.07	9.41
6	3, 7, 5	97.28	96.40	0.07	10.57
7	4, 6, 8	100.00	100.00	0.03	31.59
8	4, 7, 5	97.01	95.34	0.09	86.41
9	3, 9, 8	95.03	94.79	0.05	125.89
10	3, 16, 5	87.28	—	0.24	—
11	4, 9, 15	98.09	—	0.21	—
12	3, 18, 15	96.47	—	0.56	—
13	12, 29, 15	86.11	—	2.16	—
14	15, 50, 10	78.92	—	8.16	—
15	18, 55, 5	56.79	—	8.02	—
16	18, 55, 30	92.97	—	8.12	—

nodes as evenly as possible. This follows from the fact that in the case of an isolated $M/M/1$ station, the response time is a convex function of load with monotonically increasing derivative.

4. FAP FOR MULTIPLE CHAIN NETWORKS

Let $w_{mr}(\bar{N}, X)$ denote the mean residence time of a customer belonging to chain r, at station m when the population vector is $\bar{N}$ and allocation vector is X. Then, under the PF assumption, the MVA algorithm gives:

$$w_{mr}(\bar{N}, X) = L_{mr}(X)\left[1 + \sum_{j=1}^{R} q_{mj}(\bar{N} - e_r, X)\right],$$

where $q_{mj}(\bar{N} - e_r, X)$ is the queue length of chain j customers at station m when one chain r customer is removed from the network. In order to use this equation, we assume that:

$$q_{mj}(\bar{N} - e_r, X) \approx \frac{L_{mj}(X)}{L_j(X)}(N_j - \delta_{jr}),$$

where $\delta_{jr} \triangleq 1$, if $j = r$, else $= 0$. The above equation makes the assumption that the mean queue lengths are proportional to the loads at the nodes. This then gives the approximation to the mean residence time as $\tilde{w}_{mr}(\bar{N}, X)$, where

$$\tilde{w}_{mr}(\bar{N}, X) = L_{mr}(X)\left[1 - \frac{L_{mr}(X)}{L_r(X)} + \sum_{j=1}^{R} \frac{L_{mj}(X)N_j}{L_j(X)}\right].$$

The approximation for the cycle time for chain r, $\tilde{W}_r(\bar{N}, X)$, is then given by

$$\tilde{W}_r(\bar{N}, X) = \sum_{m=1}^{M} \tilde{w}_{mr}(\bar{N}, X). \tag{15}$$

As it was not possible to provide error bounds for the above approximation, its performance was studied experimentally. About 2000 different network configurations were randomly generated for testing, and the error in estimating the cycle times for each chain was obtained. This was expressed as the ratio of the difference between the exact values and the approximate values, to the exact values. The errors were less than 5% whenever the allocations gave a network that was reasonably balanced with regard to the time demands at the nodes. The maximum error found was about 38% for some chain, on an unbalanced allocation. Although such errors of 38% on an experimental result are surely unacceptable, the errors in the approximation for network configurations which gave lower objective function values, namely the allocations which gave a more balanced network, were within 5% as mentioned above. Hence this approximation is expected to perform better for a problem such as the FAP. These errors are based on total network populations of up to 30. It is expected that larger network populations could increase the errors. For larger populations, however, an approximation based on a load balancing heuristic gives good results.

4.1. The simplified model for multiple chains

Now consider the case where all processors operate at the same speed S. Assuming negligible communication times and delays, the total load on chain r, $L_r(X)$ is given, using equations (1) and (2) as

$$L_r(X) = \sum_{m=1}^{M} \sum_{f=1}^{F} \frac{x_{m,f}R_{r,f}}{X} = \frac{1}{S}\sum_{f=1}^{F} T_{r,f}. \tag{16}$$

Hence, $L_r(X)$ is independent of the allocation, namely, $L_r(X) = L_r$.

From (15), the estimate, $\tilde{W}_r(\bar{N}, X)$, of the cycle time for the rth customer is

$$\tilde{W}_r(\bar{N}, X) = \sum_{m=1}^{M} L_{m,r}(X)\left[1 + \sum_{j=1}^{R} N_j L_{m,j}(X)/L_j - L_{m,r}(X)/L_r\right]. \tag{17}$$

Therefore, after some elementary algebra, we can write

$$\sum_{r=1}^{R} \beta_r \tilde{W}_r(\bar{N}, X) = \sum_{r=1}^{R} \beta_r L_r + \sum_{m=1}^{M}\sum_{f=1}^{F}\sum_{g=1}^{F} d_{f,g} x_{m,f} x_{m,g}, \tag{18}$$

where

$$d_{f,g} = \sum_{r=1}^{R} \frac{\beta_r T_{r,f}}{S^2}\left[\sum_{j=1}^{R} N_j T_{j,g}/L_j - T_{r,g}/L_r\right]. \tag{19}$$

The first term in the objective function is now a constant and is removed from the objective function. Problem 1 is now reformulated as

Problem 1′

$$\min \sum_{m=1}^{M}\sum_{f=1}^{F}\sum_{g=1}^{F} d_{f,g} x_{m,f} x_{m,g}$$

$$\text{s.t.} \sum_{m=1}^{M} x_{m,f} = 1; \qquad f = 1, 2, \ldots, F,$$

$$x_{m,f} \in \{0, 1\}; \qquad f = 1, \ldots, F;\ m = 1, \ldots, M,$$

with $d_{f,g}$ as defined by equation (19).

The set partitioning problem. Problem 1′ can now be interpreted as a set partitioning problem as follows: We have a weighted, undirected complete graph G. The vertices in this graph are the files labelled 1 through F. The edges connecting two vertices f and g represent the queueing delays that would be induced if files f and g were placed on the same node. Let $w(f, g) = (d_{f,g} + d_{g,f})$ represent the weight on the edge connecting vertex (file) f with vertex g in this graph. We can partition this graph into M vertex disjoint cliques. The problem is then to do the partitioning such that the sum of the weights on the edges in all the cliques is minimized.

This is still a hard problem. It falls into the class of NP-complete problems and is essentially in the same form as the k-min cluster problem discussed by Sahni and Gonzalez [20]. As a result, we look for a heuristic solution technique. Some heuristics have been proposed for the maximization version for this type of a partitioning problem [20, 21]. The maximization version is to partition the set of F vertices into M disjoint sets (nodes) such that the weights on the edges joining vertices in different sets (nodes) is maximized. (Obviously, this is equivalent to the problem of minimizing the sum of the weights on the arcs within the subsets.) Sahni and Gonzalez show that the maximization version of the problem, termed as the k-max cut problem, has an ε-approximation algorithm. They give a one-step heuristic algorithm for this version which obtains a bound on the closeness of their heuristic solution with respect to the optimal value. If EW^* represents the optimal value for this version of the problem, and if EW is the value found by their heuristic, then the error bound obtained by them is

$$\frac{|EW^* - EW|}{EW^*} \leqslant \frac{1}{M}.$$

Hence, for the maximization version, the value returned by the heuristic quickly approaches the optimal as the number of partitions required increases.

Since the heuristic proposed by Sahni and Gonzalez [20] does guarantee a good bound on the maximization version it is hence very likely that it performs well when the minimization version is considered. We use it to obtain an initial allocation and then try to improve it by using another heuristic based on the scheme detailed by Lin and Kernighan [21]. The heuristic proposed by them is a version of an interchange search method and is similar, in method of operation, to the search technique used in Section 3 for the single chain case. More details on this technique may be found in Ref. [19].

The algorithm, proposed here, which we call LOCSEARCH, involves a one time cost of $O(R^2F^2 + F^3M)$ for the initial allocation. The cost per iteration is then $O(F^2)$. In contrast, the interchange search algorithm proposed by Bryant and Agre [10], which we shall call INTSEARCH, has time complexity $O(FMR^2)$ per iteration.

4.2. *Experimental results*

The LOCSEARCH algorithm was tested for its effectiveness in finding good solutions. To determine how close the best allocation found by LOCSEARCH was relative to the optimal allocation, it was compared to the optimal allocation found by exhaustive search. A large number of randomly generated examples were tested. The tests were carried out on a VAX 11/750 machine running the VMS operating system. It may be noted that with M nodes and F files, the number of allocations to be considered in an exhaustive search is M^F, and hence exhaustive search becomes prohibitively expensive in terms of computational effort as the number of nodes and files increase. For example, a reasonably small problem with 4 nodes, 6 files and 4 customer chains, required almost half an hour of CPU time to obtain the optimal solution through an exhaustive search. Hence the performance of the heuristic in obtaining allocations close to the optimal could perforce only be tested out for problems of a reasonably small size in the number of nodes and number of files. For larger problems, the performance of the heuristic was tested only by comparing the improvement obtained by the technique over that found by a simple load balancing heuristic. Some of these results are tabulated in Table 2.

The LOCSEARCH heuristic is also compared with the INTSEARCH heuristic of Bryant and Agre [10], as shown in Table 2. The INTSEARCH heuristic used the same starting allocation here, as used by LOCSEARCH for purposes of comparison of the algorithms. The time taken by the LOCSEARCH and INTSEARCH algorithms in obtaining the final allocation is reported, as also the time taken for the exhaustive search, where applicable. The network populations were restricted to be between 1 and 5 customers per chain.

Table 2, the minimum values reported represent the cycle time values for the network based on the final allocations found by the two heuristics and by the exhaustive search technique. These values have been normalized relative to the value of the cycle time for the initial allocation found by the load balancing heuristic. The value of the allocation found by the load balancing heuristic was, for convenience, scaled to a 100.00. Thus, an entry of 85.7 in a column for "Min found", for example, would represent an improvement in cycle time of 14.3% over the allocation found by the load balancing heuristic.

Table 2. Same speed at all nodes; initial allocation: load balance

	LOCSEARCH		INTSEARCH		EXHAUSTIVE	
Size of problem M, F, R	Min found	Time taken (s)	Min found	Time taken (s)	Min found	Time taken (s)
3, 4, 3	85.7	0.21	85.7	0.62	85.7	9.3
3, 4, 5	73.6	0.32	73.6	1.06	73.6	29.0
3, 5, 4	97.5	0.34	100.0	0.73	97.2	71.0
3, 6, 4	78.7	0.44	78.7	1.96	78.7	171.0
4, 4, 6	100.3	0.47	100.0	1.64	99.6	339.1
2, 10, 5	92.2	0.84	100.0	1.08	82.9	487.7
4, 6, 4	87.2	0.51	89.0	1.62	87.2	1525.0
7, 10, 5	98.6	1.60	99.3	16.44	—	—
8, 16, 4	97.4	34.0	99.1	35.75	—	—
10, 25, 15	97.1	32.63	99.1	597.43	—	—

Table 3. Same speed at all nodes; initial allocation: Sahni and Gonzalez [20]

Size of problem M, F, R	Sahni and Gonzalez	LOCSEARCH (Min found)	EXHAUSTIVE (Min found)
3, 4, 3	85.73	85.73	85.73
3, 4, 5	73.64	73.64	73.64
3, 5, 4	98.77	97.19	97.19
3, 6, 4	93.31	78.68	78.68
4, 4, 6	101.57	100.31	99.60
2, 10, 5	103.30	92.19	82.19
4, 6, 4	95.51	87.20	87.20
10, 25, 15	100.75	96.78	—
7, 10, 5	107.14	98.58	—
8, 16, 4	104.66	97.39	—

Table 2 shows that the load balancing heuristic performs well in some cases too, although the partitioning algorithm almost always improves on the initial solution found thus.

Now, the effectiveness of the algorithm of Sahni and Gonzalez [20] in obtaining a good initial allocation was tested. This was done by repeating the LOCSEARCH algorithm over all the problems generated earlier, except that the starting allocation was now provided by the algorithm of Sahni and Gonzalez, instead of by the load balancing heuristic. Table 3 reports the results of some of these experiments. These results are for the same test problems as reported in Table 2. In Table 3, the cycle time values returned by the heuristic of Sahni and Gonzalez and by LOCSEARCH have both been normalized relative to the cycle time found by the load balancing heuristic. It can be seen from the table that the algorithm of Sahni and Gonzalez is much more effective in providing a good allocation compared to that found by the load balancing heuristic for the problems of smaller size. In fact, in two cases, it does provide the optimal allocation. However, the load balancing heuristic appears more effective in providing initial allocations for larger problems.

About 100 different problems were thus tested. Based on these test results, we make some observations:

(i) It appears that LOCSEARCH runs faster than INTSEARCH although, on one occasion, it returned with a value worse than the initial allocation value. This is due to the error in the approximation scheme adopted by us for the solution of the queueing network.

(ii) The best allocations found by both LOCSEARCH and INTSEARCH are usually quite close to the optimum, wherever it was possible to evaluate the optimum through exhaustive search.

(iii) In general, for these cases, LOCSEARCH appears to do much better than INTSEARCH both in terms of speed of execution and in finding better allocations. However, in quite a few cases, the allocations found represented only a marginal improvement over the initial load balancing heuristic. Also, from Table 2, it can be seen that LOCSEARCH performs significantly faster than the INTSEARCH heuristic as the problem size gets large.

We now consider the use of the above approach on networks where the speeds of the various nodes were allowed to be different. The major difficulty here is that the weights on the arcs between files f and g, viz $w(f, g)$, are no longer constant, but vary for each allocation, since each allocation causes the L_r values to change. The LOCSEARCH algorithm can be modified to account for this. This modified algorithm has a more complex objective function since equation (15) is now used in place of (16) for the objective function, and this results in the presence of a variable term in the denominator of the objective function. The search method is more complex as a result, although it proceeds along similar lines as for the case where all nodes were identical.

The initial allocation is based on balancing the loads at all nodes. We also do not have a partitioning algorithm in the former sense, since the $w(f, g)$ values change with the allocation. Also, internal weights are to be recalculated for each possible change in allocation, viz. a time complexity of $O(R^2F^2)$ for each change in allocation. This makes the algorithm less effective in terms of execution speed. However, comparison with the performance of INTSEARCH for these cases indicates that LOCSEARCH does better on these cases too. About 20 different test problems were randomly generated and the heuristics were tested here. Table 4 details some of the representative results.

The results in Table 4 indicate that (i) on the average LOCSEARCH takes less than half the time taken by INTSEARCH, although it is not as effective as before in finding better allocations than the

Table 4. Different speeds at nodes; initial allocation: load balancing

Size of problem M, F, R	LOCSEARCH Min found	LOCSEARCH Time taken (s)	INTSEARCH Min found	INTSEARCH Time taken (s)	EXHAUSTIVE Min found	EXHAUSTIVE Time taken (s)
4, 4, 4	65.8	0.94	62.0	1.73	62.0	98.7
4, 4, 6	59.6	1.56	63.1	3.33	59.6	261.9
4, 4, 6	69.9	1.57	74.1	3.39	69.9	269.4
4, 4, 6	100.0	0.57	100.0	2.11	100.0	336.9
2, 12, 5	97.8	2.99	96.6	1.82	94.0	1504.9
3, 8, 6	95.0	3.86	97.2	6.21	92.3	5059.7
4, 7, 6	88.6	3.52	89.7	6.90	85.4	23172.6
4, 8, 8	91.3	7.16	91.7	13.70	—	*
7, 12, 10	83.5	25.50	85.6	74.0	—	—
10, 26, 15	87.8	361.87	87.6	685.0	—	—
11, 29, 16	90.3	491.79	90.3	1442.7	—	—

*Job was aborted after it took more than 100,000 s of CPU time.

INTSEARCH heuristic; and (ii) both heuristics find values up to 40% lower than those found by the load balancing heuristic.

5. MODELING COMMUNICATION TIMES/DELAYS

So far, we have ignored the communication times and their attendant delays. In general, including these effects in the model makes it very difficult to analyze the model except, probably, through brute force enumeration techniques. In this section, one approach is outlined for modeling the communications problem is outlined. This is an approximation technique for which no error bounds are available, and assumes that the sum of the mean communication times/delays in a cycle, henceforth referred to as the communications component of the cycle time, is a relatively small component of the cycle time: the major component being the sum of the mean residence times at the node. An alternate approach may be found in Ref. [19].

5.1. A two stage approach to model the communication component

This approach first obtains a set of promising allocations ignoring the communication delay altogether. The next stage then evaluates each allocation by considering the time demands these allocations place on the communication channels, with their attendant queueing delays. We try to adjust the mean service times at the nodes in such a manner that the mean response times at the nodes obtained with these adjusted service time demands absorb these communication times and delays. This is an iterative procedure which we outline below. The underlying premise here is that the communications component of the cycle time is not significant enough to affect the allocations found promising in stage 1.

The primary reason for considering a two-stage approach to the FAP is twofold: (i) to avoid large increase in problem size, since modeling the channel servers as service stations in the model, may increase the number of stations from M to $M(M+1)/2$; and (ii) to avoid assumption of a fixed communication path between each pair of nodes. In the second stage, we use an open model consisting of only communication servers which carry the traffic between various files (whose locations were determined in step 1). More specifically, let $\lambda_{mnr}(\bar{N}, X)$ denote the throughput of the communication server sending chain r traffic from node m to node n and let $\Lambda_r(\bar{N}, X)$ be the throughput of chain r for allocation X. For this chain, let $v_{mr}(X)$ be the visit ratio to node m, and $f_{mnr}(X)$ the fraction of traffic from node m, bound for node n. Then

$$\lambda_{mnr}(\bar{N}, X) = v_{mr}(X) f_{mnr}(X) \Lambda_r(\bar{N}, X).$$

Let σ_i denote the set of files allocated to node i and let z_{fgr} be the average number of bits transmitted by a chain r request from file f to file g. Then the average time, τ_{mnr}, needed to transmit data from node m to node n for a chain r request is

$$\tau_{mnr} = \sum_{f\in\sigma_n} \sum_{g\in\sigma_n} z_{fgr}\mu_{mn}, \tag{20}$$

where μ_{mn} is the speed of the communication link connecting nodes m and n. Thus we have an R-chain open network with up to $M(M-1)/2$ stations. It can be solved easily to get the mean response times, $\delta_{nmr}(\bar{N}, X)$. Assuming that there is no fixed path between nodes for a message to travel on, we can solve for the optimal routing for these messages using one of several schemes outlined by various authors [22–24]. For example, Kleinrock [22] considers each channel server as an $M/M/1$ queue in his model, and considers the problem of obtaining a routing of the messages for minimizing the total delay experienced in the network for all the messages circulating in the network.

Let us suppose we have solved the routing problem given an allocation X, and obtained the mean response times $\delta_{nmr}(\bar{N}, X)$ arising therefrom for transmitting messages between every pair of nodes n, m for each customer class r. We now try to adjust the mean service time demands, $L'_{mr}(X)$ at the nodes in such a manner that the resulting nodal response times $w'_{mr}(\bar{N}, X)$ absorb the communication delays (i.e. the mean response times at the channels) $\delta_{mnr}(\bar{N}, X)$. For clarity of the ensuing discussion, we henceforth omit the subscript X. Hence, we first set

$$w'_{mr}(\bar{N}) = w_{mr}(\bar{N}) + \sum_{n=1}^{M} f_{mnr}\delta_{mnr}(\bar{N}). \tag{21}$$

The MVA algorithm gives

$$w'_{mr}(\bar{N}) = L'_{mr}[1 + Q'_m(\bar{N} - e_r)]$$

and this is approximated as (also see Schweitzer [17])

$$w'_{mr}(\bar{N}) = L'_{mr}\left[1 + \sum_{\substack{j=1 \\ j\neq 1}}^{R} q'_{mj}(\bar{N}) \frac{N_r - 1}{N_r} q'_{mr}(\bar{N})\right]. \tag{22}$$

Once the $w'_{mr}(\bar{N})$ values have been estimated using equation (21), the new cycle time with these mean response times is calculated, and the throughput is obtained using Little's result [25]. Now the new mean queue lengths at the nodes is obtained at population vector $(\bar{N})$. These values are now used in (22) to obtain the L'_{mr} values.

With these new L'_{mr} values, the closed network can now be re-solved for the candidate allocations to determine better values for the throughputs to be used in stage 2, and hence better estimates for w_{mr} for use in (21) and so on. If the communications component is small to begin with, such iteration may, however, not be necessary.

If there were several candidate allocations to begin with, the best one can be selected on the basis of the cycle time of the network after accounting for the communication times.

6. CONCLUSION

The FAP modeled as a queueing network optimization problem is a complex problem. Usually this problem is addressed as a special case of a queueing network optimization problem with a single class of customers, without considering communication delays. In many cases, the problem has been posed as one for which continuous valued solutions are adequate. The objective of this paper was to extend the means by which this complex problem may be addressed and where integer solutions were necessary.

This paper considered approximate heuristic solutions to the FAP which enables the problem to be modeled with many customer chains, and which found integer allocations. A single chain version of the model was first addressed, and this was analyzed using an incremental analytic approach. The multiple chain version of the problem was transformed, by an approximation, into a graph partitioning problem which was then analyzed with the aid of some existing algorithmic techniques which were adapted for this case. Finally, some means of approximately modeling communication times and delays have been proposed.

REFERENCES

1. W. W. Chu, Optimal file allocation in a computer network. *IEEE Trans. Comput.* **C18,** 885–889 (1969).
2. R. G. Casey, Allocation of copies in an information network. *Proc. AFIPS Spring Joint Computer Conf.*, Vol. 40, pp. 617–625. AFIPS Press, Montvale, N.J. (1972).
3. S. Ceri, G. Martella and G. Pelagatti, Optimal file allocation in a computer network: A solution method based on the knapsack problem. *Comput. Networks* **6,** 345–357 (1982).
4. M. L. Fisher and D. S. Hochbaum, Data base location in computer networks. *J. ACM* **27,** 718–735 (1980).
5. K. B. Irani and N. G. Khabbaz, A methodology for the design of communication networks and the distribution of data in distributed supercomputer systems. *IEEE Trans. Comput.* **C31,** 419–434 (1982).
6. L. J. Laning and M. S. Leonard, File allocation in a distributed computer communication network. *IEEE Trans. Comput.* **C32,** 232–244 (1983).
7. K. D. Levin and H. L. Morgan, A dynamic optimization model for distributed databases. *Opns Res.* **26,** 824–835 (1978).
8. S. Mahmoud and J. S. Riordan, Optimal allocation of resources in distributed information networks. *ACM Trans. Database Syst.* **1,** 66–78 (1976).
9. H. L. Morgan and K. D. Levin, Optimal program and data location in computer networks. *Commun. ACM* **20,** 315–322.
10. M. Bryant and J. R. Agre, A queueing network approach to the module allocation problem in distributed systems. *Performance Eval. Rev.* **10,** 191–204 (1981).
11. P. P.-S. Chen, Optimal file allocation in multilevel storage systems. *Proc. AFIPS National Computer Conf.*, Vol. 42, pp. 277–282. AFIPS Press, Arlington, Va. (1973).
12. D. V. Foster, L. W. Dowdy and J. E. Ames, File assignment in a computer network. *Comput. Networks* **5,** 341–349 (1981).
13. R. M. Geist and K. S. Trivedi, Optimal design of multilevel storage hierarchies. *IEEE Trans. Comput.* **C31,** 249–259 (1982).
14. K. S. Trivedi, R. A. Wagner and T. M. Sigmon, Optimal selection of CPU speed, device capacities, and file assignments. *J. ACM* **27,** 457–473 (1980).
15. K. S. Trivedi and T. M. Sigmon, Optimal design of linear storage hierarchies. *J. ACM* **28,** 270–288 (1981).
16. C. H. Sauer and K. M. Chandy, *Computer Systems Performance Modelling.* Prentice-Hall, Englewood Cliffs, N.J. (1981).
17. P. Schweitzer, Approximate analysis of multiclass closed networks of queues. *Proc. Int. Conf. Stochastic Control and Optimization*, Amsterdam (1979).
18. M. Reiser and S. S. Lavenberg, Mean value analysis of closed multichain queueing networks. *J. ACM* **27,** 313–322 (1980).
19. M. M. Srinivasan and K. Kant, The file allocation problem—a queueing network optimization approach. Technical Report TR85-19 (Revised), Department of Industrial and Operations Engineering, The University of Michigan (1986).
20. S. Sahni and T. Gonzalez, P-complete approximation problems. *J. ACM* **23,** 555–565 (1976).
21. S. Lin and B. W. Kernighan, An efficient heuristic procedure for partitioning graphs. *Bell Syst. tech. J.* Feb. 291–307 (1970).
22. L. Kleinrock, *Queueing Systems*, Vol. 2, *Computer Applications.* Wiley, New York (1976).
23. M. Reiser, A queueing network analysis of computer communications with window flow control. *IEEE Trans. Commun.* **27,** 1199–1209 (1979).
24. H. Kobayashi and M. Gerla, Optimal routing in closed queueing networks. *ACM Trans. Comput. Syst.* **1,** 294–310 (1983).
25. J. D. C. Little, A proof of the queueing formula $L = \lambda W$. *Opns Res.* **9,** 383–387 (1961).