Fourth International Conference on Advanced Science and Technology, March 19, 1988, Chicago, pp. 154-161.

FILE ALLOCATION PROBLEMS ON COMMUNICATION-DOMINANT TWO-LEVEL LOCAL BROADCAST NETWORKS

Yao-Nan Lien[*], *and Yih-Long Chang*[**]

ABSTRACT

The file allocation problem on the communication-dominant two-level broadcast networks is studied in this paper. The communication cost to read a file in such networks is reduced to a three-value variable such that the problem is simplified comparing to the same problem on the general point-to-point networks. The simple file allocation is solved in $O(|S||N|)$ time, where $|S|$ is the total number of sites and $|N|$ is the total number of local networks in the system. The general file allocation with storage limit constraints is NP-hard, and is proved to be isomorphic to the multiple-choice-multiple-knapsack problem. The problem is transformed into a pure zero-one integer programming problem and can be optimally solved by a search algorithm based on Balas's additive algorithm. Finally, a heuristic algorithm with guaranteed performance is proposed. In many cases, the heuristic algorithm can provide efficient solutions.

INDEX TERMS: distributed computing, distributed database, file allocation, local computer network, hierarchical local computer network.

1. INTRODUCTION

This paper studies the file allocation problems on communication dominant two-level local multiaccess networks. As the technology of mini-computers and local-area networks (LANs) advances, distributed computing systems (DCS) based on local-area computer networks are becoming a very attractive means to provide information processing to local user communities. Universities, hospitals, and buildings with branch offices are examples of institutions which can profit from DCS. In environments such as these, both users and resources are physically dispersed, requiring special strategies for system control and resource management in order to deliver information processing capability to users. Processor power, memory storage, extended secondary storage, and the communication bandwidth of the network are important considerations in a distributed computing system. In most cases, these resources are not evenly utilized and overall system efficiency is effected by resulting bottlenecks. Among many tunable design parameters, the allocation of files is considered to be a very important one to improve system performance. A good file allocation strategy can significantly increase data availability and reduce the remote data-access overhead.

Allocation of files has been extensively studied as the *file allocation problem* (FAP) in distributed databases.[18,2,3] The FAP entails the distribution of possibly replicated files to a set of sites on a computer network to minimize the overall overhead. To reduce remote data-access overhead and to increase data availability, multiple copies of files are allocated to sites that demonstrate strong access locality at the cost of increased overhead on consistency maintenance and data storage. FAP was originally investigated by Chu,[2] who studied it with respect to multiple files on a multiprocessor system. He considered the problem as an integer programming problem in which the objective is to minimize the overall operating overhead in conjunction with the constraints of response time and available storage capacity. Subsequently, much work has been done in this area.[3,10,18,19] A special case of FAP is the *simple file allocation problem* (SFAP),[16] in which multiple copies of a single file are to be allocated and the effects of queries, updates, and data storage are represented as costs.

Both FAP and SFAP on general point-to-point networks are known to be NP-hard. Recently, these problems in a local area environment were studied by Wah and Lien[20], and some interesting results were discovered for systems connected by a local multiaccess network. By considering the broadcast characteristics of multiaccess networks, more efficient algorithms have been found.

The FAP in a local-area environment is important since the locality of access of data is usually much lower than that in a wide-area environment.

Although efficient file allocation algorithms in local broadcast networks have been found, they are not applicable for larger and more complex local-area systems. The number of sites that a single local broadcast network can support is normally under 100. New communication architectures will have to be used to support larger systems. A very attractive approach is to interconnect a set of local networks with a backbone network through gateways. The Andrew system of Carnegie-Mellon University is an example of this approach.[13] Other examples were discussed at the recent ACM SIGOPS Workshop on Accommodating Heterogeneity.[15] Due to the relative difficulty of the improvement on the communication technology compared to the improvement on the computation technology, the operation of such systems may be dominated by the communication overhead. It is important to reduce the communication overhead in the system design. Therefore, this paper will concentrate on the minimization of communication overhead. A more general model considering both communication overhead and computation overhead can be found in the other paper.[11]

A model for this problem is suggested in Section 2. In Section 2, the simplest form of Simple File Allocation problem is solved by a divide-and-conquer method. In Section 4, the general FAP problem with storage constraints is transformed into a linear pure zero-one integer programming problem and is proved isomorphic to a special version of knapsack problem, the multiple-choice-multiple-knapsack problem. In Section 5, the problem is solved optimally by a search algorithm based on Balas's additive algorithm. The algorithm is suitable for evaluation of small to moderate sized problems since it is not a polynomial algorithm. A heuristic algorithm with a guaranteed performance is then proposed in Section 6.

2. MODELS

2.1. One-Level Local Broadcast Networks

A one-level system is formed by connecting a set of sites or nodes to a single communication medium. Messages transmitted by a site can be addressed to one or more destinations using broadcast capability. Transmission of messages is controlled by an access control protocol. The effective distance of a local-area network is no more than a few kilometers, and the data rate is no higher than 10 MBPS unless special technologies, such as fiber optics, are used. CSMA/CD networks (e.g. Ethernet) and some ring networks (e.g. Token Ring) are the most popular networks in this category. Their logical properties are as follows.

(a) The per-unit cost of inter-site communication is site independent due to the multiaccess/broadcast capability.

* Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210. (614) 292-5236; lien@cis.ohio-state.edu, cbosgd!osu-cis!tut!lien.

** Department of Management and Information Systems, University of Arizona, Tucson, AZ 85721. (602) 621 7497; chang@arizmis.BITNET.

International Conference on Advanced Science and Technology, Illinois, March 1988.

minimize

$$C(I) = \sum_{f,n,s} \left[\lambda_{n,s}^{f} d_{n,s}^{f}(I) + \phi_{n,s}^{f} d'^{f}_{n,s}(I) + \sigma_{n,s}^{f} I_{n,s}^{f} \right] \quad (1)$$

subject to

$$\sum_{n \in N} \sum_{s \in S_n} I_{n,s}^{f} \geq 1, \quad \forall f \in F$$

$$\sum_{f \in F} \ell^{f} I_{n,s}^{f} \leq C_{n,s}, \quad \forall n \in N, s \in S,$$

where

$$d_{n,s}^{f}(I) = \begin{cases} 0 & I_{n,s}^{f} = 1 \\ d_\beta & Y_n^f = 1 \text{ and } I_{n,s}^{f} = 0 \\ d_\alpha + 2 \cdot d_\beta & Y_n^f = 0 \end{cases}$$

$$d'^{f}_{n,s}(I) = \begin{cases} 0 & I_{n,s}^{f} = 1 \text{ and } \sum_{n' \in N} \sum_{s' \in S_{n'}} I_{n',s'}^{f} = 1 \\ d_\beta' & \sum_{s' \in S_n} I_{n,s'}^{f} \geq 1 \text{ and } \sum_{n' \in N} Y_{n'}^{f} = 1 \\ d_\alpha' + \left[\sum_{\substack{n' \in N \\ n' \neq n}} Y_{n'}^{f} + 1 \right] d_\beta' & \text{otherwise} \end{cases}$$

$$I_{n,s}^{f} = \begin{cases} 1 & f \text{ is allocated to site } (n,s) \\ 0 & \text{otherwise} \end{cases}$$

Note that in Eq. (1) the number of remote networks that have file f is $\left[\sum_{n' \in N} Y_{n'}^{f} - 1 \right]$ if $Y_n^f = 1$, and is $\left[\sum_{n' \in N} Y_{n'}^{f} \right]$ otherwise.

2.4. Remarks

The general FAP on the proposed network model is NP-hard. This can be easily proved by reducing it from the standard 0-1 knapsack problem. The proof will not be shown here since it is a general case for one-level homogeneous local broadcast networks.[20] In general, the problem cannot be solved efficiently and optimally by combinatorial algorithms or mathematical programming methods. Specialized optimization algorithms and faster heuristic solutions are needed.

One way to reduce the complexity of the problem is to identify the important parameters and ignore all others. Two extreme cases are found in systems dominated by the communication overhead and in systems dominated by the processing overhead. Only the first of these systems is examined in this paper.

Although local-area communication network technology is available for most systems, a high-speed local-area network is still very expensive. For the immediate future, LANs with a speed of around 10 MBPS will be most popular. Due to the networking overhead, the effective end-to-end speed may be as low as 1 MBPS. Since bandwidth is shared by many sites, the average bandwidth per site is much lower. On the other hand, processing capacity is easier to increase. With the advance of VLSI and multiprocessing technologies, processing speed has increased significantly in the past 15 years and will continue to increase in the near future. With this increase, the processing capacity of each site can be made much greater than the communication bandwidth. Thus, communication overhead will dominate the overall system overhead. This is especially true in a high user/data/process mobility environment. The FAP for such systems is referred to as the *communication-dominant FAP*.

3. SIMPLE FILE ALLOCATION

SFAP on general point-to-point networks has been shown to be NP-hard [4], and numerous exhaustive optimal algorithms and suboptimal heuristics have been studied.[18,3]

In this section, a polynomial solution to the SFAP for two-level local broadcast networks is proposed. Since one broadcast on each network is needed to query or update the file, the problem is found to be polynomially solvable with respect to the total number of sites and the total number of local networks.

In the following cost function for a given allocation I, the index f is dropped because the problem is defined with respect to a single file.

$$C(I) = \sum_{n \in N} \sum_{s \in S_n} \left[\lambda_{n,s} d_{n,s}(I) + \phi_{n,s} d'_{n,s}(I) + \sigma_{n,s} I_{n,s} \right]$$

Since the update cost depends on the number of copies and the number of local networks that have the file, the problem is solved by considering following cases independently.

1. *Single-copy allocation:*
   Only one copy is allocated to a site. In this case, no remote update is needed for the updates initiated from local sites.
2. *Single-network allocation:*
   Multiple copies of the file are allocated to a single local network. In this case, no internet update is needed for the updates initiated from the network that has the file.
3. *Multiple-network allocation:*
   More than one local network are allocated with the file. In this case, the update cost is dependent on the number of local networks that have the file.

SFAP in each case can be solved independent of other cases. The optimal solution can be obtained by comparing the results of all cases. The solutions for the individual cases and the global optimal solution are presented in the rest of this section.

Single-copy Allocation : Denote τ as the total communication cost when all queries and updates are internet in this case. It is

$$\tau = \sum_{n \in N} \sum_{s \in S} \left[\lambda_{n,s} \left[d_\alpha + 2 \cdot d_\beta \right] + \phi_{n,s} \left[d_\alpha' + 2 \cdot d_\beta' \right] \right].$$

Denote $\Delta_{n,s}$ as the cost that can be saved from τ if the file is allocated to site (n,s), which is

$$\Delta_{n,s} = \sum_{s' \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} (d_\alpha' + d_\beta') \right] + \left[\lambda_{n,s} d_\beta + \phi_{n,s} d_\beta' \right] - \sigma_{n,s}. \quad (2)$$

The first term is the cost of internet queries and updates that can be saved from those requests initiated from the sites in network n since these requests are neighbor queries/updates. The second term is the additional cost that can be saved from the requests initiated from site (n,s). The third term is the cost to store the file. Obviously, The file should be allocated to the site with the maximum of $\Delta_{n,s}$. The cost of single-copy allocation is then

$$C(I) = \tau - \max_{\substack{s \in S \\ n \in N}} \left[\Delta_{n,s} \right]. \quad (3)$$

Single-network Allocation :

When only network n has the file, the cost Δ_n that can be saved from τ, where τ is the same as defined in single-copy allocation, is

$$\Delta_n = \sum_{s \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} (d_\alpha' + d_\beta') \right] + \sum_{s \in S_n} \left[\lambda_{n,s} d_\beta - \sigma_{n,s} \right] I_{n,s}. \quad (4)$$

$$Z^f_{n,s} \le I^f_{n,s}, \quad \forall\ n \in N, s \in S, f \in F; \tag{16}$$

$$W^f_{n,s} \le \sum_{s' \in S_n} I^f_{n,s'} - Z^f_{n,s}, \quad \forall\ n \in N, s \in S, f \in F, \tag{17}$$

$Z^f_{n,s}$ and $W^f_{n,s}$ are 0-1 variables.

Eq. (14) forces $Z^f_{n,s}$ and $W^f_{n,s}$ become 0 when $\sum_{n' \in N} Y^f_{n'} > 1$, which is the part of third case of $d'^f_{n,s}(I)$. $Z^f_{n,s} + W^f_{n,s}$ will be equal to 1 when $\sum_{n' \in N} Y^f_{n'} = 1$, because of the minimization in Eq. (13). Further, Eqs. (15) and (17) will result in $Z^f_{n,s} = 1$ when $I^f_{n,s} = 1$ and $\sum_{n' \in N} \sum_{s' \in S} I^f_{n',s'} = 1$, since the coefficient of $Z^f_{n,s}$ is more negative than that of $W^f_{n,s}$ in Eq. (13). Therefore, Eq. (13) represents the original values of $d'^f_{n,s}(I)$ in problem P.

After combining Eqs. (10) and (17) into problem P, we could restate it as a pure zero-one problem, namely, Q, as follows.

Problem Q

$$\text{minimize } C(I) = \sum_{f,n,s} (a^f_{n,s} Y^f_n + b^f_{n,s} Z^f_{n,s} + c^f_{n,s} W^f_{n,s} + d^f_{n,s} I^f_{n,s} + e^f_{n,s}) \tag{18}$$

$$\text{subject to } \sum_{n \in N, s \in S_n} I^f_{n,s} \ge 1, \quad \forall f \in F; \tag{19}$$

$$\sum_{f \in F} l^f I^f_{n,s} \le C_{n,s}, \quad n \in N, s \in S_n; \tag{20}$$

$$M Y^f_n - \sum_{s \in S_n} I^f_{n,s} \ge 0, \quad \forall\ n \in N, f \in F; \tag{21}$$

$$\sum_{s \in S_n} I^f_{n,s} - Y^f_n \ge 0, \quad \forall\ n \in N, f \in F; \tag{22}$$

$$M(1 - Z^f_{n,s} - W^f_{n,s}) - \sum_{n' \in N} Y^f_{n'} + 1 \ge 0, \quad \forall\ n \in N, s \in S, f \in F; \tag{23}$$

$$M(1 - Z^f_{n,s}) - \sum_{s' \in S_n} I^f_{n,s'} + 1 \ge 0, \quad \forall\ n \in N, s \in S, f \in F; \tag{24}$$

$$I^f_{n,s} - Z^f_{n,s} \ge 0, \quad \forall\ n \in N, s \in S, f \in F; \tag{25}$$

$$Z^f_{n,s} + W^f_{n,s} \le \sum_{s' \in S_n} I^f_{n,s'}, \quad \forall\ n \in N, s \in S, f \in F; \tag{26}$$

and Y^f_n, $Z^f_{n,s}$, $W^f_{n,s}$, and $I^f_{n,s}$ are 0-1 valued, where

$$a^f_{n,s} = -(d_\alpha + d_\beta)\lambda^f_{n,s} + (|N|-1)d_\beta'\phi^f_{n,s},$$

$$b^f_{n,s} = -(d_\alpha' + d_\beta')\phi^f_{n,s},$$

$$c^f_{n,s} = -d_\alpha'\phi^f_{n,s},$$

$$d^f_{n,s} = \sigma^f_{n,s} - d_\beta\lambda^f_{n,s},$$

$$e^f_{n,s} = (d_\alpha + 2d_\beta)\lambda^f_{n,s} + (d_\alpha' + d_\beta')\phi^f_{n,s},$$

and M is a non-Archimedean large number.

The Isomorphism to the Multiple-choice-multiple-knapsack Problem

The general FAP on this network is really isomorphic to a special case of *0-1 multiple-choice-multiple-knapsack* problem (MCMKP). A special case of *0-1 multiple-knapsack problem* is as follows: Given a set of positive values $C=\{C_i \mid i=1,2,\ldots,m\}$, and n sets of values, $(p_{1i},\ldots,p_{ji},\ldots,p_{mi},w_i)$, for $i=1,2,\ldots,n$, find 0-1 integers $x_{11},x_{12},\ldots,x_{mn}$ so as to

$$\text{maximize } \sum_{i=1}^{m} \sum_{i=1}^{n} p_{ji} x_{ji},$$

$$\text{subject to } \sum_{i=1}^{n} w_i x_{ji} \le C_j, \quad \forall\ j,$$

$$x_{ji} \in \{0,1\}, \text{ and}$$

$$\sum_{j=1}^{m} x_{ji} \ge 1, \forall\ i.$$

We may think of that n ($n=|F|$) categories of items are to be allocated to m ($m=|S|$) knapsacks with capacities $c_1,c_2,\ldots,c_m$. All items in a category are identical. A profit of p_{ji} is produced if one item of category i is allocated to knapsack j. At most one item of each category can be allocated to a particular knapsack and at least one item of each category must be allocated to one of the knapsacks. Since there are m items in each category, each category can be considered having unlimited supply. The 0-1 multiple-choice-multiple-knapsack problem is the same as the 0-1 multiple-knapsack problem except that each category of items may have more than one variations and only one variation can be used in allocating a category of items to a knapsack. The informal proof of the isomorphism between the FAP on this network and the MCMKP is as follows.

1. Map the sites in FAP to the knapsacks in MCMKP and the available storage capacity of each site to the capacity of corresponding knapsack.
2. Map the files in FAP to the categories of items in MCMKP and the length of a file to the weight of the corresponding category of items. The weights of different variations of an item are identical, but their profits are different. The profit of allocating an item of category f to the knapsack (n,s) has four variations:

(a) $$p^f_{n,s}(1) = \sum_{s' \in S_n} \left[\lambda^f_{n,s'}(d_\alpha + d_\beta) + \phi^f_{n,s'}d_\alpha'\right] + \left[\lambda^f_{n,s}d_\beta + \phi^f_{n,s}d_\beta' - \sigma^f_{n,s}\right] - \sum_{\substack{n' \in N \\ n' \ne n}} \sum_{s' \in S_{n'}} \phi^f_{n',s'}d_\beta',$$

(b) $$p^f_{n,s}(2) = \sum_{s' \in S_n} \left[\lambda^f_{n,s'}(d_\alpha + d_\beta) + \phi^f_{n,s'}d_\alpha'\right] + \left[\lambda^f_{n,s}d_\beta - \sigma^f_{n,s}\right] - \sum_{\substack{n' \in N \\ n' \ne n}} \sum_{s' \in S_{n'}} \phi^f_{n',s'}d_\beta',$$

(c) $$p^f_{n,s}(3) = \sum_{s' \in S_n} \left[\lambda^f_{n,s'}(d_\alpha + d_\beta)\right] + \left[\lambda^f_{n,s}d_\beta - \sigma^f_{n,s}\right] - \sum_{\substack{n' \in N \\ n' \ne n}} \sum_{s' \in S_{n'}} \phi^f_{n',s'}d_\beta',$$

(d) $$p^f_{n,s}(4) = \left[\lambda^f_{n,s}d_\beta - \sigma^f_{n,s}\right].$$

One further constraint to the mapping is that the choices is dynamic before the allocation is actually made. This dynamic property increases the complexity of the problem. The mapping in step 2 is explained as follows. The following fix cost is associated for each problem instance:

$$\tau = \sum_{n \in N} \sum_{s \in S} \sum_{f \in F} \left[\lambda^f_{n,s}\left[d_\alpha + 2 \cdot d_\beta\right] + \phi^f_{n,s}(d_\alpha' + d_\beta')\right].$$

This cost is really the cost when all queries and updates are assumed involving internet communication. However, the communication costs of updates are not completely counted. For each update, only the communication cost on the home network of the originating site is counted. Whenever there is a copy of a file allocated to a site, a certain cost, which is mapped to the profit in MCMKP, is saved. To maximize the profit for a MCMKP instance will minimize the cost of the corresponding instance in FAP. The profit $p^f_{n,s}(1)$ is the saved cost of allocating file f to site (n,s) when it is the unique copy in the allocation. The third term in $p^f_{n,s}(1)$ is the communication cost on the network n for the updates generated by remote networks. This cost is not counted in τ. The profit $p^f_{n,s}(2)$ is the saved cost of allocating file f to site (n,s) when only network n has the file and site (n,s) is the first site in the network that has the file. The first term is the cost saved from the neighbor queries and

very likely to allocate a copy to each of these sites. The files in category (c) are very likely those frequently used system files such as C compiler and terminal/printer drivers. To replicate these files into each system may be a very good solution. Simple allocation strategies such as *best-first* and *all-beneficial* may be very useful in these pre-allocation.

4. The optimal algorithms may be able to provide good guidelines for the development of approximate solutions. For example, the optimal algorithm proposed in [20], may be modified in such a way that solving each knapsack problem using heuristic algorithms instead of optimal algorithms.

6. HEURISTIC SOLUTIONS

Fast heuristic algorithms are needed when the number of variables is far beyond what a system can handle in a reasonable amount of time and when the accessing locality changes rapidly. It is not difficult to obtain good heuristics for file allocation problems. However, they are generally difficult to evaluate. In this section, we first discuss several different approaches to cope with the problem and propose a heuristic solution with the guaranteed performance.

"Divide-and-conquer" is one of the frequently used method to solve a complicated problem approximately. The following approaches all belong to the category of "divide-and-conquer."

Approach 1.

Repeatly take a file to solve as a simple file allocation problem until all files are allocated.

Approach 2.

Allocate a copy of each file to the system to generate an initial solution, then improve the solution by adding more copies to the system. The allocation of extra copies in each site can be solved as a standard zero-one knapsack problem.

Approach 3.

Allocate files to each site as a standard zero-one knapsack problem, then to replace the missing file to the system to satisfy the feasibility constraints.

Approach 4.

The combination of above approaches.

It is very hard to compare the merit of these approaches. In the rest of this section, a heuristic solution based on the second approach with the performance evaluation is presented.

One difficulty of approach 2 and 3 is that the "profits" of files are unknown before the allocation is done. Approach 1 does not have this problem. However, it does not take the storage constraints into account, which is the major reason that an optimum solution cannot be obtained by divide-and-conquer approaches. Therefore, the approach may not be an elegant one. Fortunately, it is reasonable to assume that the number of single-copy and single-network files is small such that the second approach can be adopted by simply assuming all files are allocated to more than one network. If this is not the case, we can use the first approach to identify these two types of files and allocate them first to make the assumption valid. The heuristic algorithm in the following follows approach 2.

Heuristic Algorithm HC2BFAP

1. Assume all files are allocated to more than one network in the final allocation such that the profit of allocating file f to site (n,s) is $\left(\lambda_{n,s}^{f} d_{\beta} - \sigma_{n,s}^{f}\right)$.
2. An arbitrary site is selected to allocate each file such that total length of all files in a site is no greater than $\lceil \frac{|F|}{|S|} \rceil k$, where k is the maximum length of all files.
3. For each site, allocate more files using standard 0-1 knapsack algorithms for the remaining storage capacity.

□

For simplicity, we assume the storage capacity of every site is no less than $\lceil \frac{|F|}{|S|} \rceil k$. Otherwise, Step 2 must be modified. Under this assumption, Step 1 is always feasible since each site can hold at least $\lceil \frac{|F|}{|S|} \rceil$ files. In step 2, the arbitrary site to store the first copy of a file can be selected using the single-copy allocation of SFAP developed in Section 2. In step 3, either optimal or heuristic 0-1 knapsack algorithms can be used. Since there exist pseudo polynomial optimal algorithms, optimal solutions may not be difficult to get in practical cases. Here, we assume an optimal algorithm is used.

Although HC2BFAP is not the best one of its type, its performance can be estimated based on the theorems developed by Lien.[9] More elegant algorithms can be developed based on it such that the performance may be predicted easily.

The Performance of Algorithm HC2BFAP

Because of the FAP problem is isomorphic to a special case of the multiple-choice-multiple-knapsack problem, the performance of algorithm HC2BFAP can be evaluated using the properties of knapsack problems developed in Lien's paper.[9] The evaluation is explained as follows. The FAP is first transformed into the MCMKP described in Section 3. The problem becomes to maximize the total profit since

$$(\text{ total cost }) = \tau - (\text{ total profit }).$$

In the rest of this section, the minimization of overall cost is referred to as the maximization of total profit. The evaluation of HC2BFAP is to find the maximum deviation between the profit generated by HC2BFAP and that can be generated by an optimal algorithm. Since it is hard to evaluate HC2BFAP against the optimal algorithm directly, it is done by comparing to another value: the total profit of $|N|$ 0-1 knapsack problem (one for each site). In which, each site is solved as an independent 0-1 standard knapsack problem. This allocation is called *many-knapsack allocation* (MANY-KNAPSACK) in this article for convenience. The profit of allocating file f to site (n,s) is the maximum of $p_{n,s}^{f}(1)$, $p_{n,s}^{f}(2)$, $p_{n,s}^{f}(3)$ and $p_{n,s}^{f}(4)$. We denote P^{*}, $P(H)$, and $P^{*}(KNAP)$ as the maximum profit generated by an optimum solution of FAP, by algorithm HC2BFAP, and by a MANY-KNAPSACK allocation. We also denote $P_{n,s}(KNAP)$ as the profit generated by solving the allocation of site (n,s) as a standard 0-1 knapsack problem and $P_{n,s}(H)$ as the profit produced by site (n,s) in algorithm HC2BFAP. In other words, $P^{*}(KNAP) = \sum_{n \in N} \sum_{s \in S_n} P_{n,s}(KNAP)$ and $P(H) = \sum_{n \in N} \sum_{s \in S_n} P_{n,s}(H)$.

The following steps are then taken to complete the evaluation.

1. For each site (n,s), estimate the maximum difference between $P_{n,s}(H)$ and $P_{n,s}(KNAP)$. Call this maximum difference as the *maximum deviation* of site (n,s).
2. Estimate the maximum difference between $P^{*}(KNAP)$ and $P(H)$ by adding the maximum deviation of all sites together.
3. It is easy to prove that P^{*} is always less than or equal to $P^{*}(KNAP)$ since the solution of FAP is also a feasible solution of the MANY-KNAPSACK allocation.[9] Therefore, the value obtained in step 2 is an upper bound of the deviation between $P(H)$ and P^{*}.

The following notations are used in the theorems derived in this section.

$p_{n,s}^{f}(0)$ - $\max\{p_{n,s}^{f}(1), p_{n,s}^{f}(2), p_{n,s}^{f}(3), p_{n,s}^{f}(4)\}$.

$\overline{p_{n,s}^{f}}$ - $p_{n,s}^{f}(0) - \min\{p_{n,s}^{f}(1), p_{n,s}^{f}(2), p_{n,s}^{f}(3), p_{n,s}^{f}(4)\}$.

Lemma 1:

In a FAP problem instance, the maximum profit produced by an optimal solution is no greater than the maximum profit produced by solving each site as a standard 0-1 knapsack problem in which the profit of allocating a file f to site (n,s) is the maximum of $p_{n,s}^{f}(1)$, $p_{n,s}^{f}(2)$, $p_{n,s}^{f}(3)$, $p_{n,s}^{f}(4)$, i.e.

$$P^{*} \le P^{*}(KNAP) = \sum_{n \in N} \sum_{s \in S} P_{n,s}(KNAP).$$

(b) The communication cost of updating a file is independent of the number of times that a file is replicated in a broadcast network. (This does not include the associated computational overhead.)

Some networks, such as Ethernet, have the following additional property.

(c) Since the message order is preserved in all sites, synchronization and consistency control is simplified.

Although the file allocation problem does not depend on property (c), it is important to point out that this property is very useful in a distributed environment since it makes the synchronization, concurrency control , and dynamic query processing much easier than that in point-to-point networks.[14, 5, 7, 12, 17, 20, 6, 8, 10] The communication overhead is assumed to be proportional to the volume of data to be transmitted. Although this assumption may not be true for short messages in which the constant overhead in each data unit may be significant when compared to overhead for the actual data, it is a reasonable assumption when a large volume of data is transmitted.

2.2. Homogeneous Two-Level Local Broadcast Networks

The network with which we are concerned in this paper consists of three components: *a backbone network*, a set of *local networks*, and a set of homogeneous *local systems* (sites). The per-unit disk-access costs and the processing costs are identical for all sites. Both the backbone network and the set of local networks are one-level local broadcast networks as described in the last section, except that the backbone network may have a higher data rate. The networks are organized into two-levels: each local system is connected to a local network; and all local networks are connected to the backbone network through gateways. Properties (b) and (c) of one-level networks are not preserved in two-level broadcast networks, but many problems involving distributed control can still be simplified. This will be discussed later in the paper. An example of a two-level local broadcast network is shown in Figure 1.

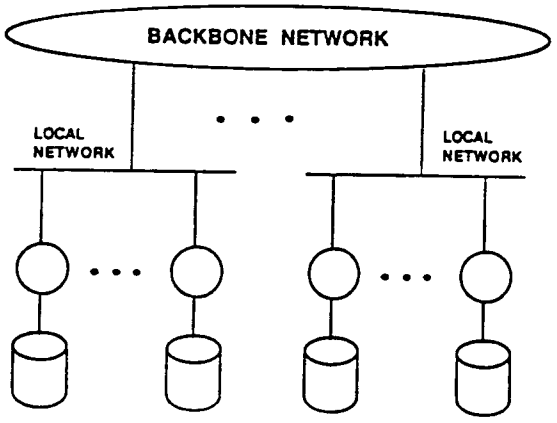

Figure 1 An example of Two-level Local multiaccess network.

Since transmissions on different networks are independent, the total bandwidth is higher than the bandwidth of a single local network. Buffering may be necessary in the gateways that connect local networks to the backbone network. The local network to which a particular site is connected is called the *home network* of the site. With respect to this site, other local networks are called *remote networks*. A query (resp., *update*) initiated from a site is called a *local query* (resp., *update*) if it is directed to local files, a *neighboring query* (resp., *update*) if it is directed to files located in another site in the home network, and an *internet query* (resp., *update*) if it is directed to files in a remote network. Neighboring and internet queries (resp., *update*) can be called *remote* queries (resp., *updates*).

Only one transmission is needed to transmit a message from one site to another site on the same local network. Processors in both sites and the home network are involved in this transmission. Three transmissions are needed to transmit a message from a given site to a remote network: one on the home network, one on the backbone network, and one on the target network. The message is first transmitted to the home network in which the sender resides. It is then forwarded to the backbone network, through one gateway, and on to the remote network through another gateway. Processors in both sites, the two local networks, the backbone network, and gateways for the two local networks are involved in this transmission.

2.3. A Model for File Allocation Problems

Assuming homogeneity, the disk overhead to access or modify a file is independent of the location of the file. Although disk overhead depends on the number of copies in an update, it is usually insignificant when compared to the overhead in identifying target data and concurrency control. The problem can therefore be simplified by not considering disk overhead. Since accessing local data does not involve networking, it can be said to involve no overhead. Therefore, only communication overhead is considered since the computation overhead is ignored as we mentioned in Section 1.

Notation definitions are as follows.

N - set of local networks

S - set of sites in the system

S_n - set of sites in the local network n

F - set of files in the database, $|F| = m$

(n,s)- site s of network n, $s \in S_n$

$\lambda_{n,s}^f$ - query load originating at site (n,s) for file f per unit time

$\phi_{n,s}^f$ - update load originating at site (n,s) for file f per unit time

d_α - per-unit cost on the backbone network for a remote query

d_β - per-unit cost on the local network for a remote query

d_β' - per-unit cost on the local network in a remote update

$\sigma_{n,s}^f$ - storage cost per unit time of file f at site (n,s)

l^f - length of file f

$C_{n,s}$ - storage capacity at site (n,s)

I - an allocation

$$I_{n,s}^f = \begin{cases} 1 & \text{if file } f \text{ is allocated to site } (n,s) \\ 0 & \text{otherwise} \end{cases}$$

$$Y_n^f = \begin{cases} 1 & \sum_{s \in S_n} I_{n,s}^f \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

The cost to transmit a request to the site containing the requested data is assumed nil; only transmission of the requested data is counted. Since the protocols for handling queries and updates may be different, their corresponding overheads may also be different. Local queries do not involve networking overhead. For each data unit in a neighboring query, a cost of d_β is incurred. The cost for an internet query is d_α.

The cost of an update is different from that of a query in two respects. First, the per-unit cost of updates may be higher than that of queries. Second, the number of network transmissions may depend on the number of data copies. Considering the update cost for each data unit, there is no cost if a unique copy is allocated at the site where the update is initiated. It is d_β' for each site having a copy of the file when only sites in the home network have the file. The cost for an internet update is more complicated: one local transmissions on network n, one transmission on the backbone network, and one transmission on each remote network having at least one copy of the updated file are needed.

Obviously, at least one copy of every file should be allocated in the system. Further, the total size of allocated files in a site should not exceed the site's capacity.

The problem is formulated as follows:

Similar to the first term in Eq. (2), the first term in Eq. (4) is the cost of internet queries and updates that can be saved. The second term is the additional cost that can be saved from all sites allocated with a copy. To allocate a copy of f to a site, a cost of $\left[\lambda_{n,s} d_\beta\right]$ can be saved, but a cost of $\sigma_{n,s}$ for storage is needed to be paid. The cost of neighbor updates cannot be saved since it is still needed to update the copies in the home network. The optimum allocation for network n in this case is to maximize the second term. Therefore, all sites having positive $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ should have a copy. If there is only one site or no site in network n having positive $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$, the network should be discarded since Eq. (4) is not correct for these two cases and they should be considered in the single-copy allocation case. Except these two cases, the cost that can be saved from τ in an optimum allocation for network n is

$$\Delta_n = \sum_{s \in S_n} \left[\lambda_{n,s}(d_\alpha + d_\beta) + \phi_{n,s}(d_\alpha' + d_\beta')\right] + \sum_{s \in S_n} D\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right] \quad (5)$$

where

$$D(v) = \begin{cases} v & v > 0 \\ 0 & v \le 0 \end{cases}.$$

The network with the maximum Δ_n is selected and all sites in the network with positive $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ is allocated a copy. Unless all local networks are discarded, the cost of single-network-multiple-copy allocation is then equal to

$$C(I) = \tau - \max_{n \in N}\left[\Delta_n\right].$$

Multiple-network Allocation : Call the local network that have the file as the *allocated local network*. The cost of update depends on the number of allocated local networks in this case. It becomes

$$d'_{n,s}(I) = \begin{cases} 0 & I_{n,s} = 1 \text{ and } \sum_{n' \in N} \sum_{s' \in S_{n'}, s' \neq s} I_{n',s'} = 0 \\ d_\alpha' + q \cdot d_\beta' & \sum_{s' \in S_n} I_{n,s'} > 0 \\ d_\alpha' + (q+1) \cdot d_\beta' & \sum_{s' \in S_n} I_{n,s'} = 0 \end{cases}$$

where q is the number of allocated local networks ($q = \sum_{n \in N} Y_n$). The Multiple-Network case can be solved by considering the fixed number of allocated local networks, then comparing all possible numbers. The case that exact q allocated local networks is called a *q-network-allocation* and can be solved network by network.

Denote τ_q as the total communication cost when all queries are internet queries and the copies in all q local networks are updated by internet communications in a q-network-allocation. It is

$$\tau_q = \sum_{n \in N} \sum_{s \in S} \left[\lambda_{n,s}\left[d_\alpha + 2 \cdot d_\beta\right] + \phi_{n,s}\left[d_\alpha' + (q+1) \cdot d_\beta'\right]\right].$$

When at least one copy is allocated to network n, the cost of remote queries and updates that can be saved is

$$\sum_{s \in S_n} \left[\lambda_{n,s}(d_\alpha + d_\beta) + \phi_{n,s} d_\beta'\right]. \quad (6)$$

The first term within the summation is the saved remote query cost, while the second term is saved neighbor update cost since only $(q-1)$ copies are stored in other local networks. For each copy allocated at site (n,s), the following cost can be saved too:

$$\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]. \quad (7)$$

Obviously, the optimum allocation for each network is to allocate a copy to those sites with positive $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$. If there is no such a site, either not to allocate the file to the network or to allocate a copy to the site with the maximum of $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ if

$$\sum_{s' \in S_n} \left[\lambda_{n,s} \cdot (d_\alpha + d_\beta) + \phi_{n,s} d_\beta'\right] + \max_{s \in S_n}\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right] > 0. \quad (8)$$

When less than q networks are allocated, the q-network allocation case should be discarded since it is dominated by either the (q-1)-network case or the single-network Allocation case. The optimal solution of multiple-network allocation can be obtained by comparing the costs of all q-network cases. Similar to the single-network allocation, the multiple-network allocation should be discarded if there is only one or no network being allocated with a copy. The cost of an optimum allocation is then

$$C(I) = \tau - \sum_{n \in N} Y_n \left[\sum_{s \in S_n} \left[\lambda_{n,s}(d_\alpha + d_\beta) + \phi_{n,s} d_\beta'\right]\right] - \sum_{s \in S} \left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right] I_{n,s}. \quad (9)$$

Global Optimum Solution: A global optimum solution can be easily obtained by comparing the results of the three cases. The time complexity of single-copy allocation and single-network allocation is $O(|S|)$, where $|S|$ is the total number of sites. The complexity of multiple-network allocation is $O(|S||N|)$, where $|N|$ is the number of local networks. Therefore, the complexity of the algorithm is $O(|S||N|)$.

4. GENERAL FAP WITH STORAGE CAPACITY CONSTRAINTS

In this section, the general FAP on the network is studied. With the storage constraints, the problem, named P, can be easily proved NP-hard since the FAP on single level local broadcast networks, which is a special case of the FAP in this network, is NP-hard.[20] However, problem P could be converted to a pure zero-one problem, and then the special algorithms could be used to solve this zero-one problem with small to medium size. We will describe the transformation to the pure zero-one problem in the following, and the special zero-one algorithm will be discussed in the next section. At the end of this section, the problem is informally proved to be isomorphic to a special case of multiple-choice-multiple-knapsack problem.

Eq. (1) is rephrased as follows. Since Y_n^f is a binary variable indicating the existence of at least one copy of file f on network n. Two constraints can be added:

$$\sum_{s \in S_n} I_{n,s}^f \le M Y_n^f, \quad \forall \ n \in N, f \in F \quad (10)$$

$$\sum_{s \in S_n} I_{n,s}^f \ge Y_n^f, \forall \ n \in N, f \in F \quad (11)$$

where M is a non-Archimedean large number to force Y_n^f become 1 when $\sum_{s \in S_n} I_{n,s}^f \ge 1$ in Eq. (10). The unit cost of query and update can be rewritten as

$$d_{n,s}^f(I) = d_\beta Y_n^f + (d_\alpha + 2 \cdot d_\beta)(1 - Y_n^f) - d_\beta I_{n,s}^f \quad (12)$$

and

$$d'^f_{n,s}(I) = d_\alpha' + \left(\sum_{n' \in N} Y_{n'}^f - Y_n^f + 1\right) d_\beta' - (d_\alpha' + d_\beta') Z_{n,s}^f - d_\alpha' W_{n,s}^f \quad (13)$$

where

$$\left(\sum_{n' \in N} Y_{n'}^f - 1\right) \le M(1 - Z_{n,s}^f - W_{n,s}^f), \quad \forall \ n \in N, s \in S, f \in F; \quad (14)$$

$$\left(\sum_{s \in S_n} I_{n,s}^f - 1\right) \le M(1 - Z_{n,s}^f), \quad \forall \ n \in N, s \in S, f \in F; \quad (15)$$

updates generated from the local network, while the second term is the additional cost that can be saved from local queries and updates. The profit $p^f_{n,s}(3)$ is the saved cost of allocating file f to site (n,s) when more than one network has the file and site (n,s) is the first site in network n that has the file. The profit $p^f_{n,s}(4)$ is the saved cost of allocating file f to site (n,s) when the conditions in $p^f_{n,s}(1)$, $p^f_{n,s}(2)$, and $p^f_{n,s}(3)$ are not applicable.

The isomorphism permits many techniques developed for the knapsack problems to solve FAP. There are many optimal algorithm and heuristic algorithms along with known properties applicable to solve FAP.[9]

5. OPTIMAL SOLUTIONS OF PROBLEM Q

Problem Q itself is a combinatorial problem. For large problems, the number of 0-1 variables or constraints may be huge. It is almost impossible to solve such problems with tens of thousands of 0-1 variables or constraints. However, in most practical systems, the scale of a problem can be reduced by excluding those files that can be allocated manually. Furthermore, the number of local networks and the number of sites on each local network tend to be small. Therefore, the scale of the problem can be reduced. Further discussion is presented at the end of this section.

These small to medium size problems then can be solved by special zero-one algorithms. Balas's additive algorithm [1] has demonstrated to be efficient in solving pure zero-one linear problems. In the following, we will describe a modified additive algorithm to solve problem Q.

By using the following operations, we can transform problem Q to the standard form, Q', described below:

(a) If the objective coefficient of a variable x_j is negative, replace it by $(1-x_j')$, that is to set $x_j = 1 - x_j'$.

(b) If a constraint in Eqs. (19) to (26) is the form of $\le$, then multiply both sides by -1.

(c) After (a) and (b) finished, reorder all the variables as the objective coefficients in the increasing order.

After the above three steps finished, problem Q is transformed into Q':

$$\text{minimize} \quad C = \sum_{j=1}^{n} c_j x_j \tag{27}$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \ge b_i, \; i=1,2,\ldots,m, \tag{28}$$

where x_j=0 or 1, j=1,2, . . . ,n, and $0 \le c_1 \le c_2 \le , \ldots , \le c_n$.

Let $A_j = \sum_{i=1}^{m} a_{ij}$, $B = \sum_{i=1}^{m} b_i$, then we have the following surrogate constraint:

$$\sum_{j=1}^{n} A_j x_j \ge B. \tag{29}$$

We also define

$$A_{i,k} = \sum_{j=k}^{n} \max \{a_{ij}, 0\}$$

and $X_k = (x_1, x_2, \ldots, x_k)$, where $k \le n$.

X_k is called a partial solution which means we have assigned $x_1, x_2, \ldots, x_k$ a value of 0 or 1. For a partial solution X_k, it could be one of the following conditions:

(a) feasible,

(b) definitely infeasible, and

(c) inconclusive.

Condition (a) can be tested by replacing the partial solution X_k into Eq. (28). If all constraints are satisfied, X_k is feasible, then further assignments of x_{k+1} to x_n will only worsen the objective function. Therefore, if X_k is feasible and its corresponding objective (C) is less than other previous solutions, we can store it as an incumbent solution.

Conditions (b) and (c) can be tested by the following two steps:

Step 1: If $\sum_{j=1}^{k} A_j x_j + \sum_{j=k+1}^{n} \max \{A_j, 0\} < B$, then X_k is definitely infeasible; otherwise go to next step.

Step 2: If $\sum_{j=1}^{k} a_{ij} x_j + A_{i,k+1} < b_i$, for any i=1,2, . . . ,m, then X_k is definitely infeasible.

If both steps are not satisfied, then the partial solution X_k is inconclusive. We will use the set W to collect all inconclusive partial solutions, that is $W = \{X_k \mid X_k \text{is inconclusive}\}$. Now, we will present a systematic branch-and-bound method which is a modified additive algorithm to solve the problem Q'.

The Algorithm

Step 1: Use the simple file allocation methods described in Section 2 to find solutions for each file. Make any adjustment to find a feasible solution for multiple files. Find C and set $C_u = C$ and store this solution in S. Set $X_0 = (\,)$, $W = \{X_0\}$;

Step 2: If $W = \varnothing$, then stop.

Step 3: Remove one X_k from W,

(i) Calculate C for X_k. If $C \ge C_u$, then there is no better solution from this X_k, and go to Step 2.

(ii) Check condition (a). If X_k is feasible, then $S = X_k, C_u = C$ and go to Step 2.

(iii) Check conditions (b) and (c). If X_k is definitely infeasible, then go to Step 2.

Step 4: Let $X_{k+1} = X_k \cup (x_{k+1} = 1)$, $X_{k+1}' = X_k \cup (x_{k+1} = 0)$ and $W = W + \{X_{k+1}, X_{k+1}'\}$, then go to Step 2.

The above algorithm is additive as a matter of fact that the only operations used are addition, subtraction, and comparison. By using the newest bound rule, the test of conditions (a) to (c) could be achieved by using results from the previous iteration. For example, to test the feasibility of X_k, we establish the following values:

$$\sum_{j=1}^{k} a_{ij} x_j \tag{30}$$

$$\sum_{j=1}^{k} A_j + \sum_{k+1}^{n} \max \{A_j, 0\} \tag{31}$$

$$\sum_{j=1}^{k} a_{ij} x_j + A_{i,k+1} \tag{32}$$

If we conclude that the two partial solutions X_{k+1} and X_{k+1}' are branched, and the tests of these two partial solutions can be simply done based on Eqs. (30) to (32), thus, in most cases, will reduce a lot of computation time.

Reduction of Problem Size

In most real systems, the number of files may be very large such that an optimum solution cannot be obtained in a reasonable amount of time. None the less, optimal algorithms are still useful for the following reasons.

1. It provides a bound to evaluate the approximate solutions.
2. In some moderate size stable systems, there is no need to solve the file allocation problem in real time since it is a design problem. It is only needed when the file system or database is to be reorganized. That will not happen very often provided that the file access locality is relatively stable. Therefore, a long execution time is likely to be tolerable as long as the ratio of the execution time to the mean time between reorganization is reasonable, say, greater than 1/30.
3. In many cases, the problem size can be reduced by allocating the following files separately: (a) rarely used files; (b) files with strong accessing locality in a few sites; and (c) frequently used files with weak accessing locality. For rarely used file, the best way is very likely not to replicate. For the files in category 2, the best way is

Proof: Since an optimum allocation of FAP is also a feasible solution of allocating each site as a standard 0-1 knapsack problem, this can be easily proved using Corollary 4.1 in Lien's paper. [9]

□

Denote $P'_{n,s}$ as the profit produced by allocating files to site (n,s) when the storage capacity of the site is only $(1+\lceil\frac{|F|}{|S|}\rceil)k$ and the profit of allocating a file f to a site (n,s) is $p^f_{n,s}(0)$, where k is the maximum length of all files.

Theorem 1:

The largest possible error that algorithm HC2BFAP can produce is no greater than

$$\text{(a)} \quad \sum_{n\in N}\sum_{s\in S_n}\left[P'_{n,s}+\sum_{f\in F}p^f_{n,s}\right], \text{ or}$$

$$\text{(b)} \quad P^*\left[\sum_{n\in N}\sum_{s\in S_n}\left[\varepsilon_{n,s}\frac{P_{n,s}(KNAP)}{P^*}\right]\right] \quad \text{if} \quad \sum_{f\in F}p^f_{n,s}\le P'_{n,s}, \ \forall f,$$

$$\text{where } \varepsilon_{n,s}=\frac{(q+1)k}{C_{n,s}},\ q=\lceil\frac{|F|}{|S|}\rceil, \text{ and } L^f\le k, \forall f.$$

Proof: This can be proved using Theorem 4 and Corollary 4.2 of Lien's paper. [9]

□

To estimate the error using Theorem 1 needs to solve some instances of the knapsack problem. Although knapsack problem is not difficult to solve, it is not convenient either. Part (b) may be easier to use than part (a) in some cases. To use part (b) of Theorem 1, the summation of $p^f_{n,s}$ is required to be small comparing to $Pnsp$. This may not be true in practical cases. However, after examining the definition of $p^f_{n,s}(1), p^f_{n,s}(2), p^f_{n,s}(3)$, and $p^f_{n,s}(4)$, we found that $p^f_{n,s}(4)$ occurs much more frequently than others. Thus, we can really release this condition and make $P_{n,s}(KNAP)$ to be the maximum profit of another knapsack problem instance that takes $p^f_{n,s}(0)$ as the profit of allocation f to site (n,s). The resulting error estimation will be more close to the real upper bound (but, not necessarily to be an upper bound.) Furthermore, if the difference between P^* and $P(KNAP)$ is not too big, $\left[\varepsilon_{n,s}\frac{P_{n,s}(KNAP)}{P^*}\right]$ can be replaced by $\left[\varepsilon_{n,s}\frac{P_{n,s}(KNAP)}{P(KNAP)}\right]$. In this case, it is even more easier to use.

6. CONCLUSION

The file allocation problems on the communication-dominant two-level broadcast networks are solved in this article. The communication cost to read a file in this networks is reduced to a three-value variable such that the problem is simplified comparing to the general point-to-point networks. In Section 2, the simple file allocation, in which only one file is to be allocated to the system, is solved in $O(|S||N|)$ time, where $|S|$ is the total number of sites and $|N|$ is the total number of local networks in the system. The general file allocation with storage limit constraints is NP-hard, and is proved to be isomorphic to a multiple-choice-multiple-knapsack problem in Section 3. The problem is transformed into a pure zero-one integer programming problem. In Section 4, a search algorithm based on Balas's additive algorithm is designed to solve the integer programming problem optimally. The algorithm takes the advantage of two-value property of zero-one integer programming to get the efficiency. It is suitable for medium size problems. Finally, a heuristic algorithm with guaranteed performance is proposed in Section 5. In most cases, the heuristic algorithm can provide efficient solutions.

References

1. Balas, Egon, "An Additive Algorithm for Sovling Linear Programs with Zero-one Variables," *Operatings Research*, vol. 13, pp. 517-549, July-August 1965.
2. Chu, W. W., "Multiple File Allocation in a Multiple Computer System," *IEEE Trans. on Comp.*, vol. C-18, no. 10, pp. 885-889, Oct. 1969.
3. Dowdy, L. W. and D. V. Foster, "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287-313, June 1982.
4. Eswaran, K. P., "Placement of Records in a File and File Allocation in a Computer Network," *Information Processing 74*,, pp. 304-307, IFIPS, 1974.
5. Gouda, M. G. and U. Dayal, "Optimal Semi-join Schedules for Query Processing in Local Distributed Database Systems," *Proc. ACM SIGMOD Conference*, pp. 164-175, May 1981.
6. Hevner, A. R., O. Q. Wu, and S. B. Yao, "Query Optimization on Local Area Networks," *ACM Trans. on Office Information Systems*, vol. 3, no. 1, pp. 35-62, Jan. 1985.
7. Kerschberg, Larry, Peter D. Ting, and S. Bing Yao, "Query Optimization in Star Computer Networks," *ACM Transactions on Database Systems*, vol. 7, no. 4, pp. 678-711, Dec. 1982.
8. Lien, Yao-Nan, *Distributed Databases on Local Multiaccess Computer Systems*, School of Electrical Engineering, Purdue University, Aug. 1986.
9. Lien, Yao-Nan, "Some Properties of 0-1 Knapsack Problems," *Conference on Combinatorics and Complexity*, p. 105, Chicago, IL., May 1987.
10. Lien, Yao-Nan and Benjamin Wah, "Design and Performance Study of DDBLMN: A Distributed Database on a Local Computer System," *Proceedings of HICSS-20, Volume 2*, pp. 407-418, Kona, Hawaii, Jan. 1987.
11. Lien, Y. N., Y. L. Chang, and B. W. Wah, "File Allocation Problems on Homogeneous Two-level Local Broadcast Networks," *To appear in the 4th Proc. of Int'l Conf. on Data Engineering*, Los Angeles, CA, Feb. 1988.
12. Masuyama, S., S. Muro, T. Mizutani, T. Ibaraki, and T. Hasegawa, "Shortest Semijoin Schedules for Local Area Distributed Database Systems," *Proc. 16th Annual Hawaiian Int'l Conf. on System Sciences*, pp. 284-293, IEEE, 1983.
13. Morris, James H., Mahadev Satyanarayanan, Michael H. Conner, John H. Howard, David S. H. Rosenthal, and F. Donelson Smith, "ANDREW : A Distributed Personal Computing Environment," *Commun. of the ACM*, vol. 29, no. 3, pp. 184-201, March 1986.
14. Nguyen, Gia Toan, "Distributed Query Management for a Local Network Database System," *Proceedings of the Second International Conference on Distributed Computing Systems Paris*, April 1981.
15. Notkin, David, Norman Hutchinson, Jan Sanislo, and Michael Schwartz, "Heterogeneous Computing Environments: Report on The ACM SIGOPS Workshop on Accommodating Heterogeneity," *Commun. ACM*, vol. 30, no. 2, pp. 132-140, Feb. 1987.
16. Ramamoorthy, C. V. and B. W. Wah, "The Isomorphism of Simple File Allocation," *IEEE Trans. on Computers*, vol. C-32, no. 3, pp. 221-232, March 1983.
17. Sacco, G. M., "Distributed Query Evaluation in Local Area Networks," *Proc. of Int'l Conf. on Data Engineering*, pp. 510-516, Los Angeles, CA, April 1984.
18. Wah, B. W., "File Placement on Distributed Computer Systems," *IEEE Computer*, vol. 17, no. 1, pp. 23-32, Jan. 1984.
19. Wah, B. W. and Y. N. Lien, "The File-Assignment And Query-Processing Problems in Local Multiaccess Networks," *Proc. of Int'l Conf. on Data Engineering*, pp. 228-235, Los Angeles, CA, April 1984.
20. Wah, B. W. and Y. N. Lien, "Design of Distributed Databases on Local Computer Systems with A Multiaccess Network," *IEEE Trans. on Software Eng.*, vol. SE-11, no. 7, pp. 606-619, July 1985.