The Isomorphism of Simple File Allocation

C. V. RAMAMOORTHY, FELLOW, IEEE, AND BENJAMIN W. WAH, MEMBER, IEEE

Abstract—**In this paper, we show that the simple file allocation problem in computer science is isomorphic to the single commodity warehouse location problem in operations research. In simple file allocation, costs due to query and update accesses and storage are considered. Design requirements such as reliability, availability, and delay are not taken into account. Due to this isomorphism, many techniques which have been developed for the warehouse location problem can be applied to solve the simple file allocation problem. Furthermore, there are techniques and conditions developed for one problem which match closely with techniques and conditions developed for the other problem. Based on a combined set of conditions developed in computer science and operations research, a heuristic for file allocation is presented.**

Index Terms—**Heuristic, isomorphism, simple file allocation, single commodity warehouse location.**

I. INTRODUCTION

THE recent advances in large scale integrated logic and communication technology, coupled with the explosion in size and complexity of the application areas, have led to the design of distributed architectures. Basically, a *Distributed Computer System* (*DCS*) is considered as an interconnection of digital systems called processing elements, each having certain processing capabilities, communicating with each other through a network, and working on a set of jobs which may be related or unrelated [41]. Examples of this are large scale systems like ARPANET, local computer networks such as Ethernet, and office information systems.

One of the important problems on a DCS is to distribute files so that they can be accessed efficiently. The optimal file allocation problem was originally studied by Chu [7] who investigated the optimal distribution of files on a given number of computers that process common information files so that the allocation yields minimum overall operating costs. Chu studied the problem with respect to multiple files on a multiprocessor system. Subsequently, a lot of work have been done in partitioning the multiple file allocation problem to multiple single file allocation subproblems, e.g., [5], [34], [40], [42]. This single file allocation problem has been coined by Eswaran as the *File Allocation Problem* (*FAP*) [14]. Eswaran's definition incorporates costs due to query and update accesses and storage, but does not consider design requirements such as reliability, availability and delay.

Other generalizations of the FAP have also been studied. The placement of files and programs which use these files has been studied by Levin and Morgan [34], [40] and others [16]. Special characteristics of the processors and network such as node capability and network configuration have been incorporated into the FAP [36], [18]. The FAP was also studied jointly with network design to minimize the total costs of file storage and communication channels [37], [6], [27]. Design requirements such as delay, reliability and availability constraints have been included in the formulation of the basic FAP [7], [37], [36], [18], [27]. Distributing files to facilitate parallel searches and accesses have been investigated by Ghosh [21], [49], [17]. The dynamic nature of accesses has also been included in the studies. In this case, files are allowed to migrate over time in order to adapt to changing access requirements [34]. A slightly different problem is related to the distribution of processes in order to minimize the communication and execution costs [50] and has been studied together with the FAP [39].

Due to the multiplicity of studies on file allocation and the generality of different design requirements on optimization, we rename Eswaran's file allocation problem as the *Simple File Allocation Problem* (*SFAP*) to differentiate it from the more general FAP. The term "simple" is appropriate because only the most basic attributes of query, update and storage costs are considered. Other operational features such as the delay of return traffic and the concurrency control algorithm used are neglected. A dynamic version of the SFAP is the *Simple File Migration Problem* (*SFMP*).

The theme of this paper is to show that the SFAP and SFMP, which have been studied extensively in computer science, are isomorphic to two equally well-known problems in operations research, called the *Single Commodity Warehouse Location Problem* (*SCWLP*) and *Single Commodity Dynamic Warehouse Location Problem* (*SCDWLP*). Due to this isomorphism, many well-defined techniques which have been developed for the SCWLP, can be applied to solve the SFAP. Examples of these are the add-drop algorithm, the branch and bound technique, the steepest ascent algorithm, the dynamic programming method and other nonoptimal heuristics. Further, it is found that some techniques and conditions developed for one problem match very closely with techniques and conditions developed for the other problem. More of these will be discussed in Sections IV and V. By utilizing some conditions developed in computer science and

Manuscript received October 13, 1980; revised July 15, 1982. This work was supported in part by the ARO under Contract DAAG29-78-G-0189, and by the National Science Foundation under Grant ECS-81-05968.

C. V. Ramamoorthy is with the Computer Science Division, Department of Electrical Engineering and Computer Sciences, and the Electronics Research Laboratory, University of California, Berkeley, CA 94720.

B. W. Wah is with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

operations research, a heuristic for file allocation is presented in Section VI. We begin by first examining some of the previous work in file allocation and warehouse location.

II. Previous Work on File Allocation

Most of the previous work on file allocation is based on static distribution, that is, the allocation does not change with time. The SFAP has been shown to be *NP*-complete [14], i.e., a class of problems for which there is no known optimal algorithm with a computation time which increases polynomially with the size of the problem [28]. The computation times for all known optimal algorithms for this class of problems increase exponentially with the problem size, i.e., if n represents the size of the problem, then the computation times goes up as k^n where $k > 1$. These algorithms are, therefore, very expensive to run in real time. Grapa and Belford remarked that a particular solution to this problem solved a thirty node problem in one hour on an IBM 360/91 computer [25]. The difficulty in optmization was also exemplified in [45].

Basically, the algorithms for static allocation can be divided into two types: mathematical programming and exhaustive searches, and heuristics.

A. Mathematical Programming and Exhaustive Searches

This technique has been used by Chu [7], Casey [5], Levin and Morgan [34], [40] and Chen [6]. Using Casey's notations, the formulation of the SFAP is as follows:

I = index set of nodes with a copy of the file;
n = number of nodes in the DCS;
ψ_j = update load originating at node j;
λ_j = query load originating at node j;
$d_{j,k}$ = cost of communication of one query unit from j to k;
$d'_{j,k}$ = cost of communication of one update unit from j to k;
σ_k = storage cost of file at k.

An optimal allocation for a given file is then defined as an index set I which minimizes the cost function.

$$C(I) = \sum_{j=1}^{n} \left[\sum_{k \in I} \psi_j d'_{j,k} + \lambda_j \min_{k \in I} d_{j,k} \right] + \sum_{k \in i} \sigma_k .$$

By defining a control variable Y_j such that

$$Y_j = \begin{cases} 0 & j \notin I \\ 1 & j \in I. \end{cases}$$

The cost function can be written as

$$C(I) = \sum_{j=1}^{n} \left[\sum_{k=1}^{n} \psi_j d'_{j,k} Y_k + \lambda_j \min_{k \in i} d_{j,k} \right] + \sum_{k=1}^{n} \sigma_k Y_k .$$

The optimization problem for file placements is

minimize

$$C(I) = \sum_{j=1}^{n} \lambda_j \min_{k \in I} d_{j,k} + \sum_{k=1}^{n} F_k Y_k \tag{1}$$

subject to

$$Y_k = 0 \text{ or } 1 \qquad i = 1, \cdots, n$$

and

$$F_k = \sigma_k + \sum_{j=1}^{n} \psi_j d'_{j,k} .$$

The quantity F_k has been introduced as Z_k in [25]. Optimization problem (1) can be solved by using integer programming techniques [20]. Casey [5] and Levin and Morgan [34], [40] have used the hypercube technique to enumerate over a reduced set of possible solutions in order to find the optimum. However, the approach of using integer programming or exhaustive enumeration is only feasible when the problem size is small. Due to this difficulty, Grapa and Belford have developed some simple conditions to determine whether a copy of a file should be placed at a node [25]. This aids in reducing the complexity of the problem significantly.

B. Heuristics

Heuristics are "reasonable" or polynomial time search strategies which do not guarantee optimality. They are usually interactive algorithm. A feasible solution is first generated. A decision algorithm then decides whether to improve the solution or not and how to improve it. An example of a decision algorithm is the add-drop algorithm which perturbs on an existing solution to see if a better solution can be obtained [37]. Other heuristics include the greedy algorithm [27], [16], steepest ascent and subgradient method [16] and clustering algorithm [35]. Since heuristics do not always generate optimal solutions and it is difficult to solve analytically for the average- and worst-case behavior, evaluations are generally done by simulations on example cases. Therefore, heuristics may perform unpredictably on unanticipated cases.

Although the *NP*-completeness of SFAP results in the exponential nature of the optimal algorithms, some special cases of this problem can be solved in polynomial time. Ghosh proposed polynomial time algorithms to distribute a database for parallal searching [21], [49]. Kikuno *et al.* have found that the placement of multiple files on a tree network with no storage capacity constraint and no update cost is polynomially solvable [31]. The number of copies of a file can be optimized very easily to achieve the maximum read throughput when it is assumed that there are infinitely many reads and updates arrive stochastically [8].

Previous work on file migration are scarce. A typical method involves the application of a static algorithm whenever need arises. Leven has applied dynamic programming to migrate copies of a file over a multiperiod horizon [34]. The problem of deciding when to migrate files has been shown to be *NP*-complete [52]. File migration has been studied with respect to the memory hierarchy [46] and is currently being extended to include distributed systems.

III. Previous Work on Single Commodity Warehouse Location

The problem on warehouse location has been studied extensively by operations researchers for a long time. As early as 1951, Dantzig used the simplex method to solve the transportation problem [10]. In 1958, Baumol described a problem called the warehouse location problem [4]. Subsequently, several variations of the problem have been investigated.

A. Simple Plant Location Problem

Given a set of plants which can supply customers with a single type of goods and with no constraints on the amount shipped from any source, the problem is to determine the locations of plants that are most profitable to the company. The optimization is done by equating the marginal cost of warehouse operation with the transportation cost savings and incremental profits resulting from more rapid delivery. This problem has been solved using a "steepest ascent one point move heuristic" [38] and enumerative optimal algorithms [11], [48], [2]. Snyder studied a special case of the plant location problem in which the paths connecting two plant locations lie on a rectangular grid [47].

B. Single Commodity Warehouse Location Problem (SCWLP)

Given a set of factories, a set of customers and a set of possible warehouse locations, the problem is to locate the warehouses so that the fixed and operational costs of the system are minimum. A special form of the problem is to neglect the transportation costs from the factories to the warehouses which then becomes the simple plane location problem. Keuhn and Hamburger developed the add-drop heuristic for the problem [32] which was extended by Feldman and Ray to include nonlinear fixed costs [15]. Khumawala further extended Efroymson and Ray's work [11] and applied a branch and bound algorithm to solve the problem [30]. A linear programming dual formulation was proposed by Erlenkotter [13] which produced optimal solutions when augmented with a branch and bound technique. Cornuejols *et al.* proposed a Lagrangian approach for the bank account location problem and is found to perform very well [9].

C. Single Commodity Dynamic Warehouse Location Problem (SCDWLP)

This is a dynamic version of the simple plant location problem or warehouse location problem in which the locations of plants or warehouses are allowed to change over a planning horizon of multiple periods in order to adapt to changing demands of the customers. This problem was first proposed by Francis [19]. Wesolowsky and Erlenkotter studied the single facility migration problem [53], [12]. Sweenly and Tatham applied dynamic programming to solve the multifacility migration problem [51]. Rao and Rutenberg studied a dynamic multilocation problem in which time was continuous and demand could change at different rates [43].

D. Capacitated Warehouse Location Problem

Consider a set of warehouses with a finite and fixed capacity, the problem is to determine their locations so that the needs of customers can be satisfied and the cost of the system is minimum. Sa, Akinc and Khumawala solved the problem using branch and bound techniques [44], [30]. Giglio solved a special case of the SCDWLP in which demands are assumed to be growing at a decreasing rate [22].

E. Quadratic Assignment Problem

Given a set of plants which require certain fixed quantities of a single type of commodity to be shipped among them and a set of possible plant locations, the problem is to assign the plants to locations so that the total cost of the system is minimum. This problem was proposed by Koopman *et al.* [32]. Armour and Buffa presented a heuristic which considered pairwise exchanges of work centers and locations [3]. Gilmore and Lawler developed optimal algorithms which were computationally feasible for small problems [23], [33]. Lawler's solution required a large number of linear assignment problems to be solved. Hiller and Connors modified Gilmore and Lawler's algorithms and obtained a more efficient but suboptimal algorithm [26]. Graves and Whinston solved the problem using a probabilistic branch and bound algorithm [24].

Some of the problems defined above are more general than the others. In fact, $A \subseteq B \subseteq C$ and $A \subseteq B \subseteq D$. We are concerned in this paper with problems A, B and C. The formulations of problems A and B are identical. Using the notations of Efroymson and Khumawala [11], [30], the problem of SCWLP with m potential uncapacitated warehouses and n customers can be formulated as a mixed integer program.

minimize

$$Z = \sum_{i,j} D_j t_{i,j} X_{i,j} + \sum_i F_i Y_i$$

subject to

$$\sum_{i \in N_j} X_{i,j} = 1 \quad j = 1, \cdots, n$$

$$0 \le \sum_{j \in P_i} X_{i,j} \le n_i Y_i \quad i = 1, \cdots, m$$

$$Y_i = 0 \text{ or } 1 \quad i = 1, \cdots, m$$

where

$t_{i,j}$ = the per unit cost which includes the FOB cost at warehouse i, the warehouse handling cost and the transportation cost from warehouse i to customer j;

D_j = the demand of customer j;

$X_{i,j}$ = the portion of D_j supplied from warehouse i;

F_i = the fixed cost associated with warehouse i;

N_j = set of warehouses which can supply customer j;

P_i = set of those customers that can be supplied by warehouse i;

n_i = number of elements in P_i;

$$Y_i = \begin{cases} 1 & \text{if warehouse exists at site } i \\ 0 & \text{otherwise} \end{cases}$$

We assume that $m = n$ and that every warehouse can supply every customer. Let

I = index set of sites with a warehouse.

It has been shown in [2] that

$$X_{i,j} = \begin{cases} 1 & \text{if } t_{i,j} = \min_{k \in I} t_{k,j}, i \in I \\ 0 & \text{otherwise} \end{cases}$$

TABLE I
MAPPING BETWEEN THE SFAP AND SCWLP

SFAP		SCWLP	
Locations of computers	n	Possible warehouse sites	n
Locations of file	I	Locations of warehouse	I
Access for a file		Commodity flow	
Amount of access at j	λ_j	Customer demand at j	D_j
Per unit cost of communicating one query unit from j to k	$d_{j,k}$	Per unit cost of shipping commodity from plant to warehouse j and from warehouse j to customer k	$t_{j,k}$
File storage cost + multiple update cost for file at node k	F_k	Fixed cost of opening a warehouse at site k	F_k
File migration		Warehouse relocation	
Cost of migrating a copy of the file from j to k		Cost of relocating a warehouse from site j to site k	

That is, the commodity will be shipped to a customer from a warehouse with the minimum transportation costs. The optimization problem can be rewritten as

minimize

$$Z = \sum_{j=1}^{n} D_j \min_{k \in I} t_{k,j} + \sum_{i=1}^{n} F_i Y_i \quad (2)$$

subject to

$$Y_i = 0, 1 \quad i = 1, \cdots, n$$

IV. THE ISOMORPHISM BETWEEN SIMPLE FILE ALLOCATION AND SINGLE COMMODITY WAREHOUSE LOCATION

After defining the SFAP and SCWLP, we are ready to prove the following theorem.

Theorem 1: The SFAP (SFMP) and SCWLP (SCDWLP) are isomorphic.

Proof: The theorem can be proved by associating the variables of SFAP with the variables of SCWLP and similarly, the variables of SFMP with the variables of SCDWLP. This association is shown in Table I. An alternative way to prove the theorem is to note that (1) and (2) are actually identical with only a change of variables. The mapping of the variables are also shown in Table I. □

Using the isomorphism result, we have shown the equivalence of these two problems. Therefore, all the results available to operations researchers are available to computer scientists and vice versa. It is important to notice that the isomorphism holds because we have modeled the problem in the simplest form. Requirements on delay, reliability and availability in the general FAP have no corresponding counterpart in the general SCWLP. The operations involved in the general problems are also different. For example, the queries in a database have return traffic and the updates are controlled by a concurrency control algorithm. These have no analogies in a warehouse system. Nonetheless, these requirements can be added as constraints to the basic formulation (1). Optimization techniques developed for the SCWLP must be modified before they can be applied to solve the more general FAP.

V. IMPLICATIONS OF THE ISOMORPHISM

Since the SCWLP has been studied extensively for a long time, many techniques developed for the SCWLP can be used to solve the SFAP. These include the add-drop technique used in [29], [3], [15] which is a heuristic of complexity $O(n^4)$; the branch and bound algorithms used in [11], [44], [30], [1] which exhaustively enumerate over a reduced set of possible solutions in order to obtain the optimal allocations; the linear programming dual formulation proposed in [13] which produces optimal solutions when augmented with a branch and bound algorithm; the probabilistic branch and bound algorithm proposed in [24] which uses probabilistic estimation to generate a lower bound; the direct search or implicit enumeration algorithm used in [48], [2]; the steepest ascent algorithm used in [38] which is a suboptimal steepest ascent one point move algorithm; the dynamic programming method used in [51] in which some conditions are developed to reduce the number of solution vectors searched; the heuristics developed in [26] and [9]; and the polynomial algorithms for some special cases, e.g., a plant location problem on a grid-like network is solved in [47], a one facility plant migration problem is solved in [53].

Similarly, techniques developed in the SFAP and SFMP can be used to solve the SCWLP. These include the hypercube technique developed in [5], [34], [40] which is identical to Alcouffe and Muratet's optimal algorithm [2] and is an implicit enumeration with conditions to discontinue the search; the clustering technique used in [35]; the dynamic programming method used in [34] to solve for the optimal migration sequence which is similar to the method used in [51]; and the max-flow-min-cut network flow technique used in [50] which can be used to solve for the SCQAP.[1]

Besides the fact that techniques developed for both problems

[1] It can be shown that the general process allocation problem of Stone [50] is isomorphic to the single commodity quadratic assignment problem (SCQAP) [52]. Stone's problem is to study the allocation of processes to computers interconnected by data links. The processes communicate with each other and there is a fixed cost of allocating a process to a computer. The objective is to minimize the fixed cost of assignments and the cost of communications. The isomorphism is shown by associating processes with plants, accesses with commodity flows and locations of computers with possible plant sites.

are interchangeable, there are instances where techniques developed for one problem match very closely with techniques developed for the other problem. This is shown in the following corollary.

Corollary 1: One of the three conditions derived by Grapa and Belford [25] for a file to be placed or not to be placed at a node is weaker than the corresponding condition derived by Efroymson and Ray [11] for a warehouse to be opened or closed. Another condition derived by Grapa and Belford can be combined with the corresponding condition of Efroymson and Ray to form a stronger condition.

Proof: Before the conditions can be stated, some additional symbols must be defined. Let

$$K_0 = \{j: Y_j = 0\};$$
$$K_1 = \{j: Y_j = 1\};$$
$$K_2 = \{j: Y_j = \text{unassigned}\}.$$

In the SFAP, K_1, K_0 represent the set of nodes with and without a copy of the file, and K_2 represents the remaining nodes in the system. In the SCWLP, K_1, K_0 represent the set of sites for which a warehouse is opened and closed and K_2 represents the set of the remaining sites.

Two of the three of Grapa and Belford's conditions for a file to be placed or not to be placed at a node are[2] [25]

for $i \in K_2$,

$$Y_i = 1 \text{ if } \lambda_i \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{i,k} - d_{i,i})_+ > F_i \qquad (3)$$

$$Y_i = 0 \text{ if } \sum_{j=1}^{n} \lambda_j \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i}) < F_i \qquad (4)$$

where

$$(f)_+ = \begin{cases} f & \text{if } f \geq 0 \\ 0 & \text{if } f < 0. \end{cases}$$

Two of the three Efroymson and Ray's conditions for a warehouse to be opened or closed are[3] [11]

$$Y_i = 1 \text{ if } \sum_{j=1}^{n} \lambda_j \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{j,k} - d_{j,i})_+ > F_i \qquad (5)$$

$$Y_i = 0 \text{ if } \sum_{j=1}^{n} \lambda_j \min_{k \in K_1} (d_{j,k} - d_{j,i})_+ < F_i \qquad (6)$$

In order to show that condition (3) is weaker than condition (5), it is necessary to show

$$\sum_{j=1}^{n} \lambda_j \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{j,k} - d_{j,i})_+ \geq \lambda_i \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{i,k} - d_{i,i})_+$$

[2] Equation (3) has been augmented by the term $d_{i,i}$ on the RHS because the original condition of Grapa and Belford is not correct when $d_{i,i} > 0$.

[3] The variables in the following two conditions have been transformed into the corresponding variables in the SFAP with the use of Table I. In the original Efroymson and Ray's conditions, $j \in P_i$ which is the set of customers that can be supplied from plant (or warehouse) i. We have made the assumption that all customers can be supplied from any plant and, therefore, $j \in \{1, \cdots, n\}$. Note that $t_{j,k}$ in the SCWLP corresponds to $d_{k,j}$ in the SFAP.

$$\begin{aligned} \text{LHS} &= \lambda_i \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{i,k} - d_{i,i})_+ \\ &\quad + \sum_{\substack{j=1 \\ j \neq i}}^{n} \lambda_j \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{j,k} - d_{j,i})_+ \\ &\geq \lambda_i \min_{\substack{k \in K_1 \cup K_2 \\ k \neq i}} (d_{i,k} - d_{i,i})_+ \\ &= \text{RHS} \end{aligned}$$

To show whether condition (4) is weaker than condition (6), we assume that it is true.

That is,

$$\sum_{j=1}^{n} \lambda_j \min_{k \in K_1} (d_{j,k} - d_{j,i})_+ \leq \sum_{j=1}^{n} \lambda_j \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i}).$$

Comparing term by term, we would like to show that

$$\min_{k \in K_1} (d_{j,k} - d_{j,i})_+ \leq \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i}).$$

There are three possibilities (note that $i \in K_2$):

i) If $d_{j,i} \leq \min_{k \in K_1} d_{j,k}$ then $\min_{k \in K_1} (d_{j,k} - d_{j,i})_+$

$$= \min_{k \in K_1} (d_{j,k} - d_{j,i}) \leq \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i})$$

ii) If $\min_{k \in K_1} d_{j,k} < d_{j,i} \leq \max_{k \in K_1} d_{j,k}$

then $\min_{k \in K_1} (d_{j,k} - d_{j,i})_+ = 0$ and $\max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i}) \geq 0$

Therefore, LHS $\leq$ RHS

iii) If $\max_{k \in K_1} d_{j,k} < d_{j,i}$, then $\max_{k \in K_1} (d_{j,k} - d_{j,i}) < 0$

$$\text{LHS} = \min_{k \in K_1} (d_{j,k} - d_{j,i})_+ = 0$$

$$\text{RHS} = \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i}) = \max \left[\max_{k \in K_1} (d_{j,k} - d_{j,i}), \max_{k \in K_2} (d_{j,k} - d_{j,i}) \right]$$

If $\max_{k \in K_1} (d_{j,k} - d_{j,i}) \geq \max_{k \in K_2} (d_{j,k} - d_{j,i})$,

then LHS > RHS

If $\max_{k \in K_1} (d_{j,k} - d_{j,i}) < \max_{k \in K_2} (d_{j,k} - d_{j,i})$,

then LHS ? RHS

Therefore

$$\sum_{j=1}^{n} \lambda_j \min_{k \in K_1} (d_{j,k} - d_{j,i})_+ \leq \sum_{j=1}^{n} \lambda_j \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i})$$

$$\text{if } d_{j,i} \leq \max_{k \in K_1} d_{j,k}.$$

Condition (4) may not be weaker than condition (6) when

TABLE II
SUMMARY OF CONDITIONS FOR PLACEMENT AND NONPLACEMENT OF A FILE AT NODE $i \in K_2$

Condition	Rule	Use
a [EFR66]	$Y_i = 1$ if $\sum_{j=1}^{n} \lambda_j \min_{k \in K_1 \cup K_2, k \neq i} (d_{j,k} - d_{j,i})_+ > F_i$	Pre-assignment in each iteration
b Cor. 1	$Y_i = 0$ if $\sum_{j=1}^{n} \lambda_j \min_{k \in K_1} (d_{j,k} - d_{j,i})_+ < F_i$ or $\sum_{j=1}^{n} \lambda_j \max_{k \in K_1 \cup K_2} (d_{j,k} - d_{j,i}) < F_i$	Pre-assignment in each iteration
c [EFR66]	If $\min_{k \in K_1} (d_{j,k} - d_{j,i}) < 0$ $j \in \{0, 1, \ldots, n\}$ then n_i is reduced by 1.	Evaluating linear programming lower bound
d [GRA77b]	$Y_i = 0$ if $F_i - F_k > \sum_{j=1}^{n} \lambda_j (d_{j,k} - d_{j,i})_+$	Initial pre-assignment

$d_{j,i} > \max_{k \in K_1} d_{j,k}$. Conditions (4) and (6) should, therefore, be combined together to form a stronger condition. □

An independent study, without recognizing the isomorphism, has shown that condition (3) is weaker than condition (5), but did not show any relationship between conditions (4) and (6) [54].

From the above corollary, the conditions for $Y_i = 0$ and $Y_i = 1$ are summarized as conditions (a) and (b) in Table II. Condition (c) is taken directly from [11] while condition (d) is taken from [25]. These conditions are very useful in preassigning copies to nodes and reducing the computation time significantly.

VI. A HEURISTIC FOR THE SFAP

In this section, we propose a heuristic to solve the SFAP using the combined conditions in Table II. The heuristic is a greedy algorithm which extends the assignment in the best possible way without backtracking on the previous assignment. The solution obtained by such an algorithm depends on the criterion used in the extension. Since the previous assignment is never backtracked, a wrong decision may have been made earlier and the solution obtained may not be optimal. Several alternative selection criteria are, therefore, investigated.

In general, the n nodes of the DCS can be partitioned into three sets, K_0, K_1 and K_2. The heuristic starts with all the nodes unassigned, that is, $K_0 = K_1 = \emptyset$, $K_2 = \{1, 2, \cdots, n\}$. It first applies the conditions (a), (b) and (d) of Table II to see if any node can be assigned without any enumeration. Condition (d) is only applied initially because this condition is independent of K_0, K_1 and K_2. After all these nodes have been assigned, it has to decide how to extend the assignment further and whether to assign a copy of the file there. It does this by selecting a node in K_2 and extending the current assignment by the selected node. There are two possibilities: either to assign or not to assign a copy of the file there. Therefore, there are altogether $2 * |K_2|$ possible extended assignments which results in $2 * |K_2|$ candidate problems. The state of a candidate problem is made up of the states of allocation of the multiple copies. For each of the candidate problems, a representative value is calculated. The function of the representative value is to estimate the minimum cost of a candidate problem without actually enumerating over all the possible allocations for the unassigned nodes. Based of these $2 * |K_2|$ representative values, the selection criterion selects a node and decides whether or not to assign a copy of the file there. After this assignment has been made, the heuristic applies conditions (a) and (b) of Table II again and repeats the steps described above until all the nodes have been assigned. The general steps of the algorithm are shown in Fig. 1. We discuss some of these steps briefly here.

M-4: The candidate list, which is a list of states made up of the sets K_0, K_1, K_2 and their corresponding representative values, is assigned the empty set.

M-5-6: These two steps essentially achieve the following: a node is selected from K_2, and is assigned a copy or not assigned a copy of the file. A representative value is calculated for these two candidate problems, C_i and $\overline{C}_i$. The computed representative values and the corresponding assignments are attached to the candidate list. These steps are then repeated for each node in K_2.

M-7: This step selects from the candidate list, the candidate problem and the corresponding assignment of nodes using the selection criterion, and uses it for the next iteration. Steps M-5 to M-7, therefore, have selected a node and have decided whether a copy should be placed there. This node is removed from the K_2 list.

There are two basic parts of the algorithm, the selection criterion and the computation of the representative value.

S1: The selection criterion:

S1a: Select from the candidate list, the candidate problem with the minimum representative value.

S1b: Select from the candidate list the two candidate problems for which node i is extended, that have the maximum difference between the representative values. From these two candidate problems, select the candidate problem with the minimum representative value.

R1: The computation of the representative value:

R1a: A lower bound is computed by solving the linear program (1) without the integrality constraints. (This has been derived earlier by Efroymson and Ray [11]; see Appendix A.) Condition (c) of Table II is very useful in improving the lower bound.

R1b: The expected value of a candidate problem is computed by assuming that each of the remaining unassigned nodes has equal probability of having or not having a copy of the file (see Appendix B).

Using the two selection criteria and the two types of representative values, there are four different versions of the algorithm.

1) MINLB—minimum lower bound (S1a, R1a);

2) MINE—minimum expected value (S1a, R1b);

3) MAXDLB—minimum lower bound for a node $i \in K_2$ with the maximum difference in lower bounds between the two candidate problems for which node i is extended (S1b, R1a).

4) MAXDE—minimum expected value for a node $i \in K_2$ with the maximum difference in expected values between the

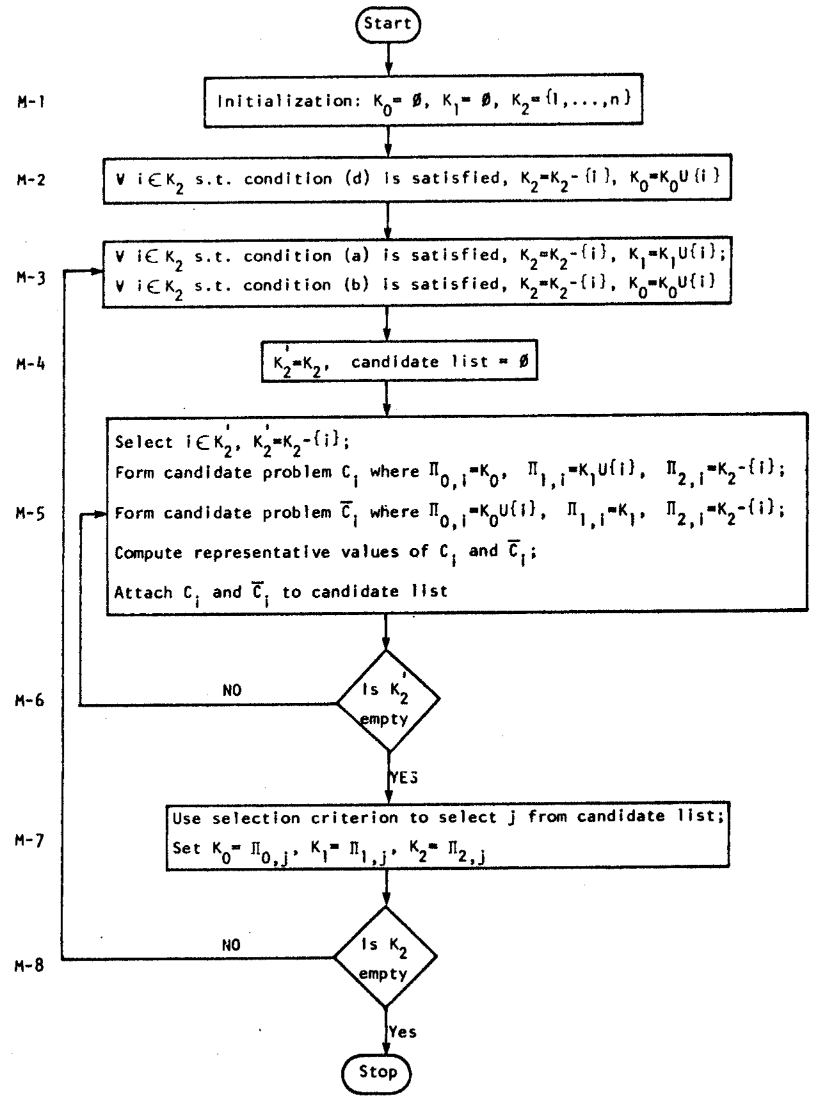
Start
M-1
Initialization: $K_0 = \emptyset$, $K_1 = \emptyset$, $K_2 = \{1,\dots,n\}$
M-2
$\forall\, i \in K_2$ s.t. condition (d) is satisfied, $K_2 = K_2 - \{i\}$, $K_0 = K_0 \cup \{i\}$
M-3
$\forall\, i \in K_2$ s.t. condition (a) is satisfied, $K_2 = K_2 - \{i\}$, $K_1 = K_1 \cup \{i\}$;
$\forall\, i \in K_2$ s.t. condition (b) is satisfied, $K_2 = K_2 - \{i\}$, $K_0 = K_0 \cup \{i\}$
M-4
$K_2' = K_2$, candidate list $= \emptyset$
M-5
Select $i \in K_2'$, $K_2' = K_2' - \{i\}$;
Form candidate problem C_i where $\Pi_{0,i} = K_0$, $\Pi_{1,i} = K_1 \cup \{i\}$, $\Pi_{2,i} = K_2 - \{i\}$;
Form candidate problem $\overline{C}_i$ where $\Pi_{0,i} = K_0 \cup \{i\}$, $\Pi_{1,i} = K_1$, $\Pi_{2,i} = K_2 - \{i\}$;
Compute representative values of C_i and $\overline{C}_i$;
Attach C_i and $\overline{C}_i$ to candidate list
M-6
NO
Is K_2' empty
YES
M-7
Use selection criterion to select j from candidate list;
Set $K_0 = \Pi_{0,j}$, $K_1 = \Pi_{1,j}$, $K_2 = \Pi_{2,j}$
M-8
NO
Is K_2 empty
Yes
Stop

Fig. 1. File assignment heuristic.

two candidate problems for which node i is extended (S1b, R1b).

To further illustrate the steps of the algorithm, it is applied on Casey's 5 node example [5].

Suppose the following matrix represents the query and update costs $d_{j,k}$ and $d'_{j,k}$ for a five-node system.

$$d = d' = \begin{bmatrix} 0 & 6 & 12 & 9 & 6 \\ 6 & 0 & 6 & 12 & 9 \\ 12 & 6 & 0 & 6 & 12 \\ 9 & 12 & 6 & 0 & 6 \\ 6 & 9 & 12 & 6 & 0 \end{bmatrix}$$

Let

$$\lambda = [\lambda_j] = [24\ \ 24\ \ 24\ \ 24\ \ 24]$$

$$\psi = [\psi_j] = [2\ \ 3\ \ 4\ \ 6\ \ 8]$$

$$\sigma = [\sigma_k] = [0\ \ 0\ \ 0\ \ 0\ \ 0]$$

and

$$F = [F_k] = [168\ \ 180\ \ 174\ \ 126\ \ 123].$$

By enumerating the $2^5 - 1$ possible allocations, it is found that a copy of the file should be allocated to nodes 1, 4 and 5 giving a cost of 705. The steps for the four possible versions of the algorithm are shown in Fig. 2(a)-(d), respectively. It is seen that two of these versions give the optimal solution.

VII. Evaluation of the Heuristic

The algorithm is evaluated by applying it on published examples in the SFAP and SCWLP.[4] The optimal solutions for these examples have been established in the literature. The deviation of the heuristic solutions from the optimal solutions can be used as an indication of the "goodness" of the heuristic. The heuristic is also compared against the add-drop algorithm of Keuhn and Hamburger [29].[5] The evaluation results are

[4] The first six sets of problems are taken from [5]. Problems 7-18 are taken from [29] and problems 19-22 are taken from problem 7 of [44, p. 1013].

[5] In an add-drop algorithm, a feasible distribution of files is first found. The total cost of the system can be improved by successive addition or deletion of file copies. When a feasible solution with a lower cost is found, it is adopted as a new starting solution and the process continues. Eventually, a local optimum is reached in which addition or deletion does not reduce the cost. The whole procedure can be repeated with a different starting feasible solution and several local optima can be obtained. The final solution is obtained by taking the minimum over all the local optima. Instead of directly using Keuhn and Hamburger's add-drop algorithm, which selects only 5 warehouse sites to be evaluated in each cycle, the add-drop algorithm used here allows for all the unassigned warehouse sites to be taken into consideration.

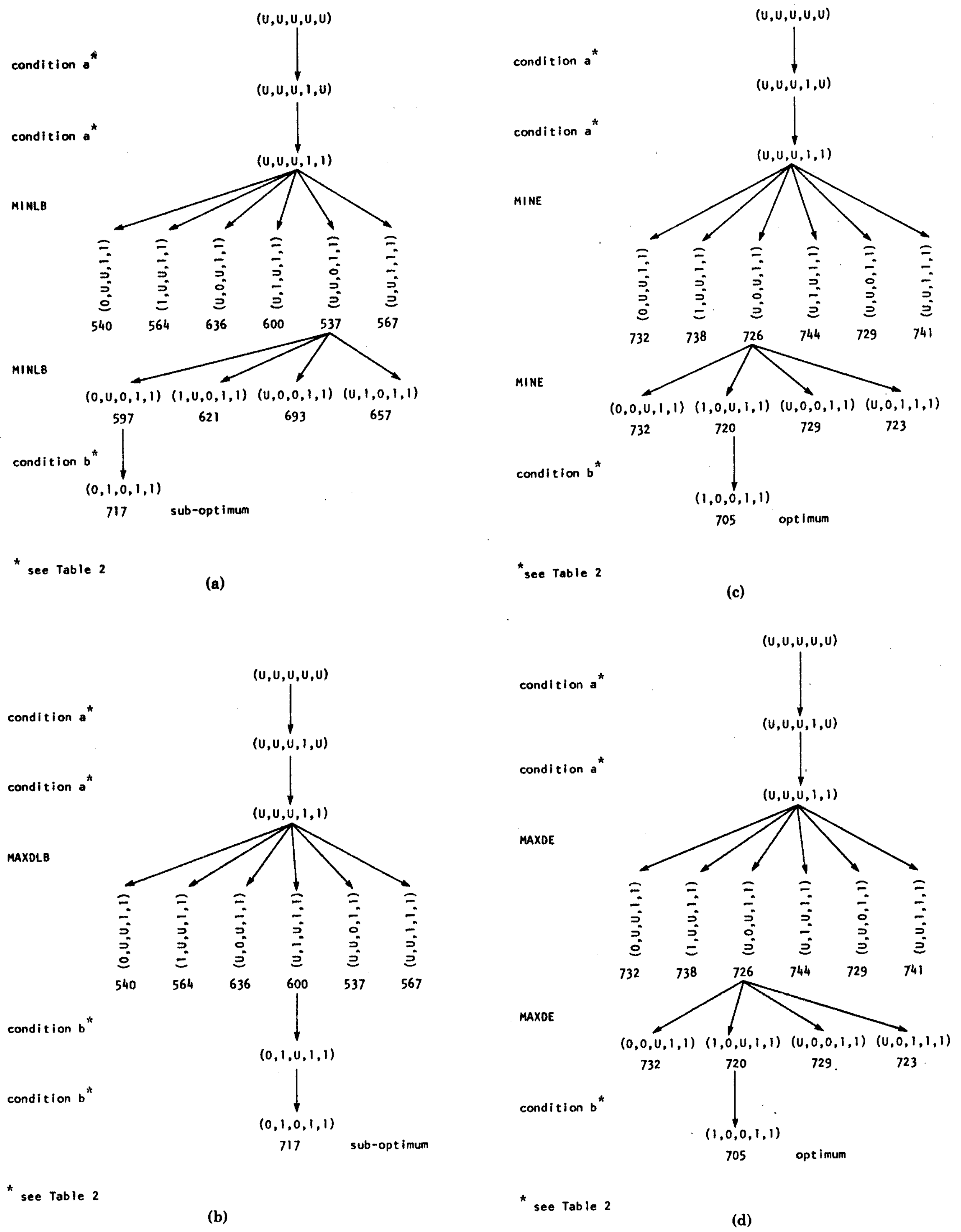

Fig. 2. (a) Evaluation of Casey's 5 node example using MINLB. (b) Evaluation of Casey's 5 node example using MINE. (c) Evaluation of Casey's 5 node example using MAXDLB. (d) Evaluation of Casey's 5 node example using MAXDE (*U* indicates that the node is unassigned).

TABLE III
PERCENT DEVIATIONS OF FILE ALLOCATION HEURISTIC FROM OPTIMAL SOLUTIONS

Prob.	Optimum Sol.	Add-Drop	MINLB	MINE	MAXDLB	MAXDE	Comments	
1	117596	0	0	0	8.43	0	$\alpha=0.1$	Casey's 19 node file allocation problem [5]
2	188738	0.03	0.31	0.31	0.31	0.31	$\alpha=0.2$	
3	242581	0	0	0.66	0	0.66	$\alpha=0.3$	
4	291790	0	1.39	0	1.39	0	$\alpha=0.4$	
5	431720	0	0	0	0	0	$\alpha=1.0$	
6	705	0.85	1.70	0	1.70	0	Casey's 5 node ex. [5]	
7	796648	0.11	0	0.78	0	0.78	Factory at Indianapolis	Keuhn and Hamburger's 24 warehouses, 50 customers warehouse location problem [29]
8	854704	0.15	0.09	0.89	0	0.89		
9	893782	0.14	0	0.71	0	0.71		
10	928942	0	0.61	0.94	1.49	0.99		
11	1092916	0.08	0	0.13	0.10	0.13	Factory at Jacksonville	
12	1145923	0.13	0	0.22	0	0.22		
13	1188241	0.13	0	1.37	0	1.37		
14	1244991	0.22	0.22	2.49	0	1.67		
15	614548	0.14	0	0.90	0	0.90	Factory at Baltimore and Ind'polis	
16	659983	0	0.12	0.80	0	0.80		
17	690746	0.03	0	0.74	0	0.74		
18	724886	0	0	0.42	0	0.49		
19	806145	0	0	0.88	0	0.38	Factory at Ind'polis, but not warehouse	Problem 7 of Sa [44]
20	870792	0.15	0	0.67	0	0.67		
21	919994	0.11	0	1.46	0	0.44		
22	970446	0	0.42	1.73	1.36	0.67		
	mean	0.10	0.22	0.73	0.67	0.58		
	std.dev.	0.18	0.46	0.62	1.83	0.44		

TABLE IV
EXECUTION TIME IN SECONDS OF HEURISTIC FOR THE CORRESPONDING PROBLEMS ON THE CDC 6400

Prob.	Add-Drop	MINLB	MINE	MAXDLB	MAXDE
1	0.57	11.45	22.79	11.42	103.71
2	0.43	11.59	23.57	11.66	105.57
3	0.43	11.77	23.60	11.76	105.50
4	0.43	11.80	23.48	11.79	105.26
5	0.29	11.80	23.80	11.84	105.10
6	0.04	0.08	0.09	0.06	0.24
7	11.46	8.08	11.85	8.29	26.41
8	9.36	13.55	13.52	11.23	35.73
9	5.34	20.89	13.91	20.99	37.61
10	3.61	8.48	8.29	8.50	21.42
11	12.08	6.40	9.13	6.39	17.74
12	8.62	12.66	12.64	12.71	30.62
13	7.82	22.16	21.26	22.83	62.03
14	7.02	40.18	33.47	40.33	112.93
15	9.75	5.48	12.44	5.50	25.35
16	7.33	5.49	4.37	4.64	7.90
17	5.58	6.82	7.01	6.84	17.25
18	3.79	2.66	3.75	2.68	7.26
19	12.24	4.94	9.16	4.95	16.04
20	9.02	13.63	13.81	11.29	34.39
21	6.76	23.27	22.05	20.97	67.74
22	5.94	22.79	28.81	22.02	73.40
mean	5.81	12.54	15.58	12.21	50.87
std.dev.	4.11	8.92	8.83	8.84	39.27

shown in Table III. The four proposed variations of the heuristic are all polynomial time algorithms and each has a complexity of $O(n^4)$ (the same as the add-drop algorithm). The execution times on the CDC 6400 are shown in Table IV.

It is seen from Tables III and IV that the algorithm MINLB gives the best results and has small execution times as compared with other algorithms. In fact, algorithm MINLB obtains the optimal solutions more often than the add-drop algorithm in general, but the worst-case behavior seems to be worse and the execution times are longer because the algorithm is more complex. On the other hand, algorithm MAXDLB produces more optimal solutions than algorithm MINLB, but its worst-case behavior seems to be worse. Algorithms MINE and MAXDE are much worse than algorithms MINLB and MAXDLB. Improvements can be obtained if we use the estimated lower bound (by estimating the mean and standard deviation and making an assumption of normal distribution), but the complexity of the algorithm is $O(n^5)$ and it takes too long to produce a solution for any of these problems (>600 s). However, we can still improve the heuristic solution by combining the results of the add-drop algorithm, the MINLB algorithm and the MAXDLB algorithm. In this case, over 80 percent of the problems will have optimal assignments and the time complexity of the combined algorithm is still $O(n^4)$.

VIII. CONCLUSION

In this paper, we have presented the proof of isomorphism between the simple file allocation (migration) problem and the single commodity (dynamic) warehouse location problem. It is found that many techniques developed for both problems are interchangeable or complementary. We have discovered that some of Grapa and Belford's conditions for locating a copy of the file at a node [25] can be augmented by the conditions of Efroymson and Ray for opening or closing a warehouse [11] to result in stronger conditions.

By combining the conditions developed in [11] and [25], we have developed a heuristic to solve the simple file allocation problem. The heuristic is a greedy algorithm and different criteria on selection are compared. It is found that a combination of these criteria is very promising and gives solutions very close to the optimal allocation based on sample problems published in both the simple file allocation problem and warehouse location problem. The technique presented in this paper is extendable to general file allocation problems with additional constraints and other *NP*-complete problems.

APPENDIX A
THE LINEAR PROGRAMMING LOWER BOUND OF A CANDIDATE PROBLEM [11]

Efroymson and Ray's formulation of the linear programming lower bound is based on the optimization problem of (1), with an exception that $\sum_{k=1}^{n} X_{j,k} d_{j,k}$ is not evaluated to be $\min_{k \in I} d_{j,k}$ where $X_{j,k}$ is the fraction of λ_j that is directed towards node k. By defining the following notations,

N_j = set of indexes of those nodes that can be accessed by user j;

P_k = set of indexes of those users that can access node k;

n_k = number of elements in P_k.

The optimization problem of Efroymson and Ray is *minimize*

$$C(I) = \sum_{j,k} \lambda_j d_{j,k} X_{j,k} + \sum_k F_k Y_k \qquad \text{(A-1)}$$

such that

$$1 = \sum_{k \in N_j} X_{j,k} \qquad (j = 1, \cdots, n)$$

$$0 \le \sum_{j \in P_k} X_{j,k} \le n_k Y_k \qquad (k = 1, \cdots, n)$$

$$Y_k = 0, 1$$

The linear programming solution to the above optimization problem, neglecting the integrality constraint of Y_k, is

$$X_{j,k} = \begin{cases} \text{if } \lambda_j d_{j,k} + \dfrac{f_k}{n_k} = \min\limits_{l \in K_1 \cup K_2} \left[\lambda_j d_{j,l} + \dfrac{f_l}{n_l}\right] \\ 1 \\ 0 \\ \text{otherwise} \end{cases} \tag{A-2}$$

$$Y_k = \left(\frac{1}{n_k}\right) \sum_{j \in P_k} X_{j,k} \tag{A-3}$$

where

$$f_k = \begin{cases} F_k & k \in K_2 \\ 0 & k \in K_1 \end{cases}$$

The proof of this can be found in [11]. Condition (c) of Table II is very useful in reducing the values of n_k.

Appendix B
The Expected Value of a Candidate Problem

The objective function (1) can be rewritten on condition on K_0 and K_1.

$$\begin{aligned} C(I) = & \sum_{k \in K_1} F_k \\ & + \sum_{j \in K_0} \lambda_j \min_{k \in I} d_{j,k} \\ & + \sum_{j \in K_2} \lambda_j \min_{k \in I} d_{j,k} + \sum_{k \in K_2} F_k Y_k \end{aligned}$$

$$C(I) = \sum_{k \in K_1} F_k + \sum_{j \in K_0 \cup K_2} \lambda_j \min_{k \in I} d_{j,k} + \sum_{k \in K_2} \tag{B-1}$$

where F_k is defined in (1).
Let

$$Z_1 = \sum_{j \in K_0 \cup K_2} \lambda_j \min_{k \in I} d_{j,k} \tag{B-2}$$

$$Z_2 = \sum_{k \in K_2} F_k Y_k \tag{B-3}$$

So

$$C(I) = \sum_{k \in K_1} F_k + Z_1 + Z_2 \tag{B-4}$$

Assuming that each of the combinations of Y_k for $k \in K_2$ can be assigned uniformly, we would like to find the expected value of $C(I)$. We first define some notations: For each row j of the cost matrix, we define a one-to-one mapping μ_j such that

$\mu_j: i \rightarrow k;\ i, k \in \{1, \cdots, n\}$ such that

if $\mu_j(i_1) \le \mu_j(i_2)$, then $d_{j,\mu_j(i_1)} \le d_{j,\mu_j(i_2)}$, $i_1, i_2 \in \{1, 2, \cdots n\}$.

The mapping μ_j maps the costs in row j onto a new set such that the costs of access from node j in the mapped matrix are monotonic.

$$d_{j,t} = \min_{k \in K_1} d_{j,k}$$

$t \in K_1$ is the node with the minimum cost of access from node $_j$.

$|\overline{K}_2| = |K_0 \cup K_1|$ (cardinality of $\overline{K}_2$)

$K = K_0 \cup K_1 \cup K_2$

$$C(K) = \begin{cases} 2^{n-|\overline{K}^2|-1} & \text{if } |K_1| = 0 \\ 2^{n-|\overline{K}_2|} & \text{if } |K_1| > 0 \end{cases}$$

$K^j_{2,k} = \{i: i \in K_2 \text{ and } \mu_j(i) \ge \mu_j(k)\}$

Now

$$E(C(I)) = \sum_{k \in K_1} F_k + E(Z_1) + E(Z_2)$$

$$\begin{aligned} E(Z_1) &= E\left(\sum_{j \in K_0 \cup K_2} \lambda_j \min_{k \in I} d_{j,k}\right) \\ &= \sum_{j \in K_0 \cup K_2} \lambda_j E\left(\min_{k \in I} d_{j,k}\right) \end{aligned}$$

$$E\left(\min_{k \in I} d_{j,k}\right) = \frac{\displaystyle\sum_{\substack{k \in K_2 \\ \mu_j(k) < \mu_j(t)}} d_{j,k} 2^{(|K^j_{2,k}|-1)} + d_{j,t} 2^{|K^j_{2,t}|}}{C(K)}$$

$$\begin{aligned} E(Z_2) &= E\left(\sum_{k \in K_2} F_k Y_k\right) \\ &= \sum_{k \in K_2} F_k E(Y_k) \end{aligned}$$

$$E(Y_k) = \frac{2^{(n-|\overline{K}_2|-1)}}{C(K)}$$

$$\begin{aligned} E(C(I)) = & \sum_{k \in K_1} F_k + \frac{1}{C(K)} \sum_{j \in K_0 \cup K_2} \\ & \cdot \lambda_j \left[\sum_{\substack{k \in K_2 \\ \mu_j(k) < \mu_j(t)}} d_{j,k} 2^{(|K^j_{2,k}|-1)} + d_{j,t} 2^{|K^j_{2,t}|}\right] \\ & + \left(\sum_{k \in K_2} F_k\right) \frac{2^{(n-|\overline{K}_2|-1)}}{C(K)} \end{aligned} \tag{B-5}$$

Acknowledgment

The authors would like to thank an anonymous referee who corrected the original proof of Corollary 1.

References

[1] U. Akinc and B. Khumawala, "An efficient branch and bound algorithm for the capacitated warehouse location problem," *Manage. Sci.*, vol. 23, pp. 585-594, Feb. 1977.

[2] A. Alcouffe and G. Muratet, "Optimum location of plants," *Manage. Sci.*, vol. 23, pp. 267-274, Nov. 1976.

[3] G. C. Armour and E. S. Buffa, "A heuristic algorithm and simulation approach to relative location of facilities," *Manage. Sci.*, vol. 9, pp. 294-309, Jan. 1963.
[4] W. J. Baumol and P. Wolfe, "A warehouse location problem," *Oper. Res.*, vol. 6, pp. 252-263, Mar.-Apr. 1958.
[5] R. G. Casey, "Allocation of copies of a file in an information network," *AFIPS, SJCC*, 1972, pp. 617-625.
[6] P. P. S. Chen and J. Akoka, "Optimal design of distributed information systems," *IEEE Trans. Comput.*, vol. C-29, pp. 1068-1080, Dec. 1980.
[7] W. W. Chu, "Multiple file allocation in a multiple computer system," *IEEE Trans. Comput.*, vol. C-18, pp. 885-889, Oct. 1969.
[8] E. G. Coffman, Jr. *et al.*, "Optimization of the number of copies in a distributed data base," *IEEE Trans. Software Eng.*, vol. SE-7, pp. 78-84, Jan. 1981.
[9] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms," *Manage. Sci.*, vol. 23, pp. 789-810, Apr. 1977.
[10] G. B. Dantzig, "Application of the simplex method to a transportation problem," in *Activity Analysis of Production and Allocation*, T. C. Koopmans, Ed. New York: Wiley, 1951, Cowles Commission Monograph, no. 13, ch. 23.
[11] M. A. Efroymson and T. C. Ray, "A branch and bound algorithm for plant location," *Oper. Res.*, pp. 361-368, May-June 1966.
[12] D. Erlenkotter, "Dynamic facility location and simple network models," *Manage. Sci. Notes*, vol. 26, p. 1131, May 1974.
[13] D. Erlenkotter, "A dual-based procedure for uncapacitated facility location," *Oper. Res.*, vol. 26, pp. 992-1009, Nov.-Dec. 1978.
[14] K. P. Eswaran, "Placement of records in a file and file allocation in a computer network," *Information Processing 74, IFIPS.* New York: North Holland, 1974.
[15] E. Feldman, F. A. Lehner, and T. L. Ray, "Warehouse location under continuous economies of scale," *Manage. Sci.*, vol. 12, pp. 670-684, May 1966.
[16] M. L. Fisher, and D. S. Hochbaum, "Database locations in computer networks," *J. Ass. Comput. Mach.*, vol. 27, pp. 718-735, Oct. 1980.
[17] M. J. Fisher *et al.*, "Optimal placement of identical resources in a distributed network," in *Proc. 2nd Int. Conf. Dist. Comput. Syst.*, Paris, France, 1981, pp. 324-336.
[18] D. V. Foster *et al.*, "File assignment in a star network," in *Proc. 1977 SIGMETRICS/CMG VIII Conf. Comput. Perform.*, Washington, DC, 1977, pp. 247-254.
[19] R. L. Francis, "A note on the optimum location of new machines in existing plant layouts," *J. Indust. Eng.*, Jan.-Feb. 1963.
[20] A. M. Geoffrion and R. E. Marsten, "Integer programming: a framework and state-of-the-art survey," *Manage. Sci.*, vol. 18, pp. 465-491, May 1972.
[21] S. P. Ghosh, "Distributing a data base with logical associations on a computer network for parallel searching," *IEEE Trans. Software Eng.*, vol. SE-2, pp. 106-113, June 1976.
[22] R. J. Giglio, "A note on the deterministic capacity problem," *Manage. Sci. Notes*, vol. 19, pp. 1096-1099, Aug. 1973.
[23] P. C. Gilmore, "Optimal and sub-optimal algorithms for the quadratic assignment problem," *J. Soc. Indust. Appl. Math.*, vol. 10, pp. 305-313, June 1962.
[24] G. W. Graves and A. B. Whinston, "An algorithm for the quadratic assignment problem," *Manage. Sci.*, vol. 16, pp. 453-471, Mar. 1970.
[25] E. Grapa and G. G. Belford, "Some theorems to aid in solving the file allocation problem," *Commun. Ass. Comput. Mach.*, vol. 20, pp. 878-882, Nov. 1977.
[26] F. S. Hiller and M. M. Connors, "Quadratic assignment problem algorithms and the location of indivisible facilities," *Manage. Sci.*, vol. 13, pp. 42-57, Sept. 1966.
[27] 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.*, vol. C-31, pp. 419-434, May 1982.
[28] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher Eds. New York: Plenum, 1972, pp. 85-104.
[29] A. A. Keuhn and M. J. Hamburger, "A heuristic program for locating warehouses," *Manage. Sci.*, vol. 9, pp. 643-666, July 1963.
[30] B. M. Khumawala, "An efficient branch and bound algorithm for the warehouse location problem," *Manage. Sci.*, vol. 18, pp. B-718-B731, Aug. 1972.
[31] T. Kijuno *et al.*, "On a file initial-placement problem in distributed database systems," Hiroshima Univ., Japan, CSG Tech. Res. 81-08, Apr. 1981.
[32] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, pp. 53-76, Jan. 1957.
[33] E. L. Lawler, "The quadratic assignment problem," *Manage. Sci.*, vol. 9, pp. 586-599, July 1963.
[34] K. D. Levin, "Organizing distributed data bases in computer networks," Ph.D. dissertation, Univ. Pennsylvania, Philadelphia, 1974.
[35] M. E. S. Loomis, "Data base design: object distribution and resource constrained task scheduling," Ph.D. dissertation, Dep. Comput. Sci., Univ. California, Los Angeles, 1975.
[36] M. E. S. Loomis and G. J. Popek, "A model for data base distribution, in *Computer Networks: Trends and Applications.* New York: IEEE, 1976, pp. 162-169.
[37] S. Mahmoud and J. S. Riordon, "Optimal allocation of resources in distributed information networks," *ACM Trans. Data Base Syst.*, vol. 1, pp. 66-78, Mar. 1976.
[38] A. S. Manne, "Plant location under economies of scale decentralization and computation," *Manage. Sci.*, vol. 11, pp. 213-235, Nov. 1964.
[39] R. Marcogliese and R. Novarese, "Module and data allocation methods in distributed systems," in *Proc. 2nd Int. Conf. Distrib. Comput. Syst.*, Paris, France, 1981, pp. 50-59.
[40] H. L. Morgan and K. D. Levin, "Optimal program and data locations in computer networks," *Commun. Ass. Comput. Mach.*, vol. 20, pp. 315-322, May 1977.
[41] C. V. Ramamoorthy and T. Krishnarao, "The design issues in distributed computed systems," *Infotech State Art Rep. Distrib. Syst.*, pp. 375-400, 1976.
[42] C. V. Ramamoorthy and B. W. Wah, "File placements of relations in a distributed relational data base," in *Proc. 1st Int. Conf. Distrib. Comput. Syst.*, Huntsville, AL, Oct. 1979.
[43] R. C. Rao and D. P. Rutenberg, "Multi-location plant sizing and timing," *Manage. Sci.*, vol. 23, pp. 1187-1198, July 1977.
[44] G. Sa, "Branch and bound and approximate solutions to the capacitated plant location problem," *Oper. Res.*, vol. 17, pp. 1005-1016, Nov.-Dec. 1969.
[45] L. V. Sickle and K. M. Chandy, "Computational complexity of network design algorithms," in *Information Processing 77, IFIPS.* New York: North Holland, 1977.
[46] A. J. Smith, "Optimization of I/O systems by cache disks and file migration," in *Performance Evaluation.* New York: North Holland, 1981, vol. 1, pp. 249-262.
[47] R. D. Snyder, "A note on the location of depots," *Manage. Sci.*, vol. 18, p. 97, Sept. 1971.
[48] K. Spielberg, "An algorithm for the simple plant location problem with some side conditions," *Oper. Res.*, vol. 17, pp. 85-115, Jan.-Feb. 1969.
[49] B. Srinivasan and R. Sankar, "Algorithms to distribute a database for parallel searching," *IEEE Trans. Software Eng.*, vol. SE-7, p. 112, Jan. 1981.
[50] H. S. Stone, "Multiprocessor scheduling with the aid of network flows," *IEEE Trans. Software Eng.*, vol. SE-3, pp. 85-93, Jan. 1977.
[51] D. J. Sweenly, and R. L. Tatham, "An improved long run model for multiple warehouse location," *Manage. Sci.*, vol. 22, pp. 748-758, Mar. 1976.
[52] B. W. Wah, "Data management in distributed systems and distributed data bases," Ph.D. dissertation, Univ. California, Berkeley, 1979.
[53] G. O. Wesolowsky, "Dynamic facility location," *Manage. Sci.*, vol. 19, pp. 1241-1248, July 1973.
[54] J. G. Kollias and M. Hatzopoulos, "Criteria to aid in solving the problem of allocating copies of a file in a computer network," *Computer J.*, vol. 24, no. 1, pp. 29-30, 1981.

C. V. Ramamoorthy (M'57-SM'76-F'78) received the undergraduate degrees in physics and technology from the University of Madras, Madras, India, the M.S. degree and the professional degree of Mechanical Engineer, both from the University of California, Berkeley, and the M.A. and Ph.D. degrees in applied mathematics and computer theory from Harvard University, Cambridge, MA.

He was associated with Honeywell's Electronic Data Processing Division from 1956 to 1971, last as Senior Staff Scientist. He was a Professor in the Department of Electrical Engineering and Computer Sciences, University of Texas, Austin. Currently, he is a Professor in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

Dr. Ramamoorthy was Chairman of Education Committee of the IEEE Computer Society and Chairman of the Committee to develop E.C.P.D. Accreditation Guidelines for Computer Science and Engineering Degree Programs. He also was the Chairman of the AFIPS Education Committee, a member of the Science and Technology Advisory Group of the U.S. Air Force, and a member of the Technology Advisory Panel of Ballistic Missile Defense (U.S. Army). Currently, he is Vice President of the IEEE Computer Society for Area Activities.

Benjamin W. Wah (S'74-M'77-S'78-M79) received the B.S. and M.S. degrees in electrical engineering and computer science from Columbia University, New York, in 1974 and 1975, the M.S. degree in computer science, and the Ph.D. degree in engineering both from the University of California, Berkeley, in 1976 and 1979, respectively.

Currently, he is an Assistant Professor in the School of Electrical Engineering, Purdue University, West Lafayette, IN. His current research interests include parallel computer architecture, distributed processing, and theory of computing.

File Allocation in a Distributed Computer Communication Network

LAURENCE J. LANING AND MICHAEL S. LEONARD

***Abstract*—An algorithm is presented to determine locations for the storage of copies of files in store-and-forward computer communications networks. The algorithm determines storage locations which minimize the sum of network file storage costs and message transmission costs. Networks that use adaptive routing techniques are the primary focus. Feasible file locations must satisfy network performance requirements for file availability and delay by message class. An effective method of evaluating delay constraints for networks using adaptive routing techniques is introduced. The algorithm uses the solution to a *p*-median problem to identify initial candidate file placements. Interaction between a set of file movement rules and a network simulator is employed to modify initial placements to find near-optimal locations which satisfy the network performance constraints.**

***Index Terms*—Adaptive routing, algorithm, computer communication network, distributed data management, file allocation, heuristic, mathematical programming, network delay, *P*-median, simulation.**

INTRODUCTION

THE traditional mode of computer use in industry, government and educational institutions has been to operate large central computer systems that could provide a great deal of information processing capability at a central site. With the advent and proliferation of time-sharing computer systems in the late 1960's, it was recognized that one could provide computing resources to many geographically dispersed users in an efficient manner. This has been followed by the design of computer networks that allow the resources of each computer site or node in such a network to share or distribute those resources (programs, data, computing capability, etc.). Examples of these networks are the ARPA (Advanced Research Projects Agency) network [1], the first of such networks (implemented in 1968); the Aloha system at the University of Hawaii [2]; the CYCLADES network in France [3]; and the DATAPAC network in Canada [4]. The concept of the distribution of resources is currently of great interest in many areas of information processing.

The nature of computer networks is such that network design involves making selections from among very large but finite sets of interrelated design alternatives. Those design alternatives include the determination of node locations, the topology of the network, the assignment of channel capacities for each link from a discrete list of available capacities, the design of the flow control and routing procedures to be employed, the selection of nodes in the network for file storage and other related factors. In reality, however, it is unlikely that one has the freedom or the ability to develop a network design in a setting where all possible design alternatives are candidates for consideration. Typically a proposed computer network will have at least some hardware and communication links already in place. The network designer must therefore select a design that is compatible with previously selected or fixed values of some design parameters. With the situation of limited design

Manuscript received April 2, 1980; revised November 3, 1981.

L. J. Laning is with the Management Systems Division, Procter & Gamble, Cincinnati, OH 45201.

M. S. Leonard is with the Department of Industrial Engineering, University of Missouri-Columbia, Columbia, MO 65201.