PROCEEDINGS OF 1980 COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE

OUTSTANDING PAPER AWARD, IEEE, 1981.

An Efficient Heuristic for File Placement on Distributed Databases

Benjamin W. Wah

School of Electrical Engineering,
Purdue University,
West Lafayette, IN 47907.

ABSTRACT

In this paper, we present an efficient heuristic for the placement of multiple copies of a file on a distributed database. The single file placement problem is studied because under most circumstances, the placement of multiple files can be decomposed into the placement of single file. The algorithm presented is a greedy algorithm and different selection criteria are used at each step to aid in the selection of the node to assign. 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 the literature.

Research supported by Purdue Research Foundation Summer XL-grant

1. 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 (PE's)*, each having certain processing capabilities, communicating with each other through an interconnection network and working on a set of jobs, which may be related or unrelated [RAM76, AND75]. This definition encompasses a wide range of configurations from an uni-processor system with different functional units to a multiplicity of general purpose computers (e.g. ARPANET).

Data on a DCS are managed through a *Database (DB)* which is a collection of stored operational data used by the application systems of some particular enterprise [DAT77]. A *Distributed Database (DDB)* can be regarded as the data stored at different locations of a DCS. It can be considered to exist only when data elements at multiple locations are interrelated and/or there is a need to access data stored at some locations from another location.

In processing data on a DCS, there are several major components of costs, namely, storage costs, processing costs, and communication costs. The relative size of each cost governs the control and processing strategies of a system. In general, communication is still relatively expensive as compared with processing and storage. The design of efficient control and coordination schemes that would minimize the amount of communication traffic is therefore a very critical problem. The placement of files on a DDB is a problem that involves the tradeoff between storage and communication. If a user accesses a file that is available locally, no communication costs will be incurred in the access. However, there are fixed overhead costs in storing the file and the additional communications due to updates from other users in the system. On the other hand, if the file is not available locally, the users have to pay a cost in terms of delay in accessing the file and also additional traffic in the network before he can make the access. The objective is therefore to place the file optimally on the system so that the total cost is minimum. In this paper, an algorithm for the placement of multiple copies of a single file on a DDB that minimizes communication traffic is presented.

Most of the previous work on file allocation is based on static distribution, that is, the allocation does not change with time. Chu has studied the optimal static file allocation problem which allocates files on a number of computers so that the allocation yields minimum overall operating costs [CHU69]. This problem is directed toward the optimal placement of multiple files on the DCS. Subsequently, a lot of work have been done to partition the problem of allocating multiple files to multiple sub-problems of allocating multiple copies of individual files, e.g. [CAS72, LEV74, MOR77, RAM79a, WAH79]. This partitioning is possible because some special characteristics and strategies of the system is assumed, such as the program-data file relationship, [LEV74, MOR77] and the query processing strategies [RAM79a, WAH79]. This single file allocation problem has been coined by Eswaran as the *File Allocation Problem (FAP)* [ESW74]. The formulation of the FAP generally transforms all the constraints on the system into a common unit of cost which may include file access costs, multiple update costs, file storage costs and file migration costs. Likewise, other constraints such as response time, queueing delays and storage capacity can be included in the formulation.

The static algorithms are usually 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 [GRA77]. The difficulty in optimization is also exemplified in [SIC77]. Moreover, the problem has been shown to be NP-complete [ESW74], 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 [KAR72]. The computation times for all known optimal algorithms for this class of problem increases exponentially with the problem size, i.e., if n represent the size of the problem, then the computation time goes up as k^n where k>1. In order to achieve a polynomial execution time, heuristics are generally used which sacrifice optimality for efficiency. Techniques such as clustering [LOO75, LOO76] and add-drop [MAH76] have been applied to solve some general file allocation problem with complicated constraints and assumptions.

In this paper, a heuristic to solve the basic FAP of Eswaran is presented. More complicated constraints are not considered because these constraints are difficult to justify in practice and under most of these circumstances, the minimization of communication costs due to accesses and updates is the most important consideration. The heuristic designed is based on a property of the FAP. It is shown in [WAH79, RAM79b] that the FAP is isomorphic to the single commodity warehouse location problem which has been studied extensively in operations research. Based on this property, a combined set of conditions to drive the heuristic is developed from results in operations research and computer science. Evaluations are carried out on sample problems in file placement and warehouse location.

2. FORMULATION OF THE FILE ALLOCATION PROBLEM

The formulation of this problem follows from Casey [CAS72] and Levin and Morgan [LEV74, LEV75, MOR77]. The symbols used in this formulation are shown in Table 1. An optimal allocation for a given file is defined as an index set I which minimizes the cost function.

$$C(I) = \sum_{j=1}^{n} \left[\sum_{k \in I} U_j M_{j,k} + Q_j \min_{k \in I} S_{j,k} \right] + \sum_{k \in I} F_k$$

By defining a control variable Y_j such that

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

$Y_j=1$ means that a copy of the file is assigned to node j. The cost function can be written as:

$$C(I) = \sum_{j=1}^{n} \left[\sum_{k=1}^{n} U_j M_{j,k} Y_k + Q_j \min_{k \in I} S_{j,k} \right] + \sum_{k=1}^{n} F_k Y_k$$

The optimization problem for file placements is:
min

$$C(I) = \sum_{j=1}^{n} Q_j \min_{k \in I} S_{j,k} + \sum_{k=1}^{n} C_k Y_k \qquad (1)$$

subject to

$$Y_k = 0 \text{ or } 1 \text{ (integer)} \quad k=1,\ldots,n$$

and

$$C_k = F_k + \sum_{j=1}^{n} U_j M_{j,k} \qquad (2)$$

The quantity C_k has been introduced as Z_k in [GRA77]. Optimization problem (1) can be solved by using integer programming techniques [GEO72]. Casey [CAS72] and Levin and Morgan [LEV74, MOR77] 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 suitable when the problem size is small. Due to this difficulty, Grapa and Belford have done some pioneering work in developing three simple conditions to check whether a copy of a file should be placed at a node [GRA77]. These reduce the complexity of the problem tremendously because many alternatives can be eliminated. Based on the proof of isomorphism [WAH79], some of Grapa and Belford's conditions are shown to be weaker than the conditions developed by Efroymson and Ray [EFR66] for the single commodity warehouse location problem. A refined set of conditions are compiled from [GRA77] and [EFR66] and used in the heuristic. This is shown in Table 2. Due to the fact that the nodes may not be completely assigned, these conditions are defined with respect to the set of assigned and unassigned nodes. Let

$K_0 = \{j, Y_j=0\}$ set of nodes with a copy not assigned

$K_1 = \{j, Y_j=1\}$ set of nodes with a copy assigned

$K_2 = \{j, Y_j=\text{unassigned}\}$ set of unassigned nodes

The conditions in Table 2 show the situation under which a node should be assigned or not assigned a copy of the file when there is a partial assignment on the system. For example, condition (a) shows that a copy of the file should be assigned to node i when the cost of accessing it from another node is greater than the fixed cost of keeping a copy at node i. The heuristic presented in the next section uses these conditions to decide whether any particular node in a partial assignment should be assigned a copy of the file.

3. A HEURISTIC FOR THE FAP

In this section, we propose a heuristic to solve the FAP. The heuristic is a greedy algorithm which extends the assignment in the best possible way without backtracking on the previous assignment. The solution obtained using such an algorithm depends on the criterion used in the extension. Because 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.

Essentially, the heuristic starts with all the nodes unassigned. It first applies the conditions of Table 2 to see if any node can be assigned without any enumeration. After all these nodes have been assigned, it comes to a point at which it has to decide what node to extend the assignment and whether or not to assign a copy of the file there. It does this by extending the current assignment by one node. For each of these extended assignments, 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 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 n different nodes on the DCS. In general, the n nodes of the DCS can be partitioned into three sets, K_0, K_1 and K_2.) For each of the candidate problems, a representative value is calculated. The function of the representative value is to estimate the minimum of the candidate problem without actually enumerating over all the allocations for the unassigned nodes. Based on these $2*|K_2|$ representative values, the selection criterion selects the node and decide whether or not to assign a copy of the file there. After this assignment has been made, the algorithm is ready to check for the conditions of Table 2 again and therefore it repeats the steps described above until all the nodes have been assigned. The general steps of the algorithm are shown in Figure 1. We discuss each of these steps briefly here.

M-1 This is to initialize the candidate problem - all nodes are unassigned at this point. The candidate list, which is a list of states made up of the sets K_0, K_1, K_2 and its corresponding representative value, is assigned the empty set.

M-2-5 These four steps essentially achieve the following: a node is selected from the un-assigned set, K_2, and is assigned a copy or not assigned a copy of

the file. A representative value is calculated for each of the candidate problems. The computed representative value and the corresponding assignments are attached to the candidate list. These steps are then repeated for each node in K_2.

M-6 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-2 to M-6 therefore have selected a node and have decided whether a copy should be placed at that node. This node is removed from the K_2 list.

M-7 The steps M-2 to M-6 are repeated until the K_2 list is empty.

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

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 of $Y_i=0$ and $Y_i=1$. 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 (Eq. 1) without the integrality constraints. (This has been derived earlier by Efroymson and Ray [EFR66], see Appendix A);

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 with the maximum difference in lower bounds between $Y_i=0$ and $Y_i=1$ ($i \in K_2$) (S1b, R1a);
4. MAXDE - minimum expected value for a node i with the maximum difference in expected values between $Y_i=0$ and $Y_i=1$ ($i \in K_2$) (S1b, R1b);

To further illustrate the steps of the algorithm, it is applied on Casey's 5 node example [CAS72]. Suppose the following matrix represents the query cost $S_{i,j}$ for a five-node system.

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

Let

$Q = [Q_i] = [\ 24\ 24\ 24\ 24\ 24]$

$U = [U_i] = [\ 2\ 3\ 4\ 6\ 8]$

$F = [F_i] = [\ 0\ 0\ 0\ 0\ 0]$

and

$C = [C_i] = [\ 168\ 180\ 174\ 126\ 123]$.

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

4. EVALUATION OF THE HEURISTIC

The algorithm is evaluated by applying it on the published examples in the FAP and the Single Commodity Warehouse Location Problem[1]. 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 [KEU63][2]. The evaluation results are shown in Table 3. The four proposed variations of the heuristic are all polynomial 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 4.

It is seen from Tables 3 and 4 that the algorithm MINLB gives the best results and has an execution time very small 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 than the add-drop algorithm 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 the standard deviation and making an assumption of normal distribution), but the complexity of the algorithm will become $O(n^5)$ and it takes too long to produce a solution for any of these problems (> 600 seconds). 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% of the problems have optimal assignments and the complexity of the combined algorithm is still $O(n^4)$.

5. CONCLUSION

In this paper, we have presented a heuristic to solve the file allocation problem. A heuristic is necessary because the file allocation problem is NP-complete and

[1] The first six sets of problems are taken from [CAS72]. Problems 7 to 18 are taken from [KEU63] and problems 19 to 22 are taken from problem 7 of [SA 69, p. 1013].

[2] 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.

the search for optimal algorithms that run in exponential time is impractical. 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 file allocation problem and the warehouse location problem. The technique presented in this paper is extendable to file placement problems with additional constraints and other NP-complete problems.

APPENDIX A THE LINEAR PROGRAMMING LOWER BOUND OF A CANDIDATE PROBLEM [EFR66]

Efroymson and Ray's formulation of the linear programming lower bound is based on the optimization problem of Eq. 1, with an exception that $\sum_{k=1}^{n} X_{j,k} S_{j,k}$ is not evaluated to be $\min_{k \in I} S_{j,k}$ where $X_{j,k}$ is the fraction of Q_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:
min

$$C(I) = \sum_{j,k} Q_j S_{j,k} X_{j,k} + \sum_{k} C_k Y_k \qquad \text{(A-1)}$$

such that

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

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

$$Y_k = 0, 1$$

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

$$X_{j,k} = \begin{cases} 1 & \text{if } S_{j,k} + \dfrac{g_k}{n_k} = \min\limits_{i \in K_1 \cup K_2} \left[S_{j,i} + \dfrac{g_i}{n_i} \right] \\ 0 & \text{otherwise} \end{cases} \qquad \text{(A-2)}$$

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

where

$$g_k = \begin{cases} C_k & k \in K_2 \\ 0 & k \in K_1 \end{cases}$$

The proof of this can be found in [EFR66].

APPENDIX B THE EXPECTED VALUE OF A CANDIDATE PROBLEM

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

$$C(I) = \sum_{i \in K_1} C_i + \sum_{i \in K_0} Q_i * \min_{j \in I} S_{i,j} + \sum_{i \in K_2} Q_i * \min_{j \in I} S_{i,j} + \sum_{i \in K_2} C_i Y_i$$

$$C(I) = \sum_{i \in K_1} C_i + \sum_{i \in K_0 \cup K_2} Q_i * \min_{j \in I} S_{i,j} + \sum_{i \in K_2} C_i Y_i \qquad \text{(B-1)}$$

where C_i is defined in Eq. 2.
Let

$$Z_1 = \sum_{i \in K_0 \cup K_2} Q_i * \min_{j \in I} S_{i,j} \qquad \text{(B-2)}$$

$$Z_2 = \sum_{i \in K_2} C_i Y_i \qquad \text{(B-3)}$$

So

$$C(I) = \sum_{i \in K_1} C_i + Z_1 + Z_2 \qquad \text{(B-4)}$$

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

$$\mu_i : j \to k;\ j,k \in \{1, \ldots, n\} \text{ such that } S_{i,\mu_i^{-1}(k)} \leq S_{i,\mu_i^{-1}(k+1)}$$

The mapping μ_i maps the original set of nodes onto a new set such that the costs of access from node i in the mapped matrix are in increasing order.

$$S_{i,t} = \min_{j \in K_1} S_{i,j}$$

$t \in K_1$ is the node which has the minimum cost of access from node i.

$|\overline{K_2}| = |K_0 \cup K_1|$ (*cardinaltiy of* $\overline{K_2}$)

$K = K_0 \cup K_1 \cup K_2$

$$C(K) = \begin{cases} 2^{n-|\overline{K_2}|} - 1 & \text{if } |K_1| = 0 \\ 2^{n-|\overline{K_2}|} & \text{if } |K_1| > 0 \end{cases}$$

$K_{2iq} = \{x : x \in K_2 \text{ and } \mu_i(x) \geq \mu_i(q)\}$

Now

$$E(Z) = \sum_{i \in K_1} C_i + E(Z_1) + E(Z_2)$$

$$E(Z_1) = E\Big(\sum_{i \in K_0 \cup K_2} Q_i * \min_{j \in I} S_{i,j} \Big) = \sum_{i \in K_0 \cup K_2} Q_i * E\Big(\min_{j \in I} S_{i,j} \Big)$$

$$E\Big(\min_{j \in I} S_{i,j}\Big) = \frac{\left[\sum\limits_{\substack{q \in K_2 \\ \mu_i(q) < \mu_i(t)}} S_{i,q} 2^{(|K_{2iq}|-1)} + S_{i,t} 2^{|K_{2it}|} \right]}{C(K)}$$

$$E(Z_2) = E\Big(\sum_{j \in K_2} C_j Y_j \Big) = \sum_{j \in K_2} C_j E(Y_j)$$

$$E(Y_i) = \frac{2^{(n-|\overline{K_2}|-1)}}{C(K)}$$

$$E(Z) = \sum_{i \in K_1} C_i + \frac{1}{C(K)} \sum_{i \in K_0 \cup K_2} Q_i * \left[\sum_{\substack{q \in K_2 \\ \mu_i(q) < \mu_i(t)}} S_{i,q} 2^{(|K_{2iq}|-1)} + S_{i,t} 2^{|K_{2it}|} \right] + \Big(\sum_{i \in K_2} C_i \Big) \frac{2^{(n-|\overline{K_2}|-1)}}{C(K)} \qquad \text{(B-5)}$$

REFERENCES

[AND75] Anderson, G. A. and Jensen, E. D., "Computer Interconnection Structures: Taxonomy, Characteristics and Examples", *Computing Surveys*, Vol. 7, No. 4, December, 1975.

[CAS72] Casey, R. G., "Allocation of Copies of a File in an Information Network", *AFIPS, SJCC*, 1972, pp. 617-625.

[CHU69] Chu, W. W., "Multiple File Allocation in a Multiple Computer System", *IEEE Trans. on Comp.*, Vol. C-18, No. 10, Oct. 1969, pp. 885-889.

[DAT77] Date, C. J., *An Introduction to Data Base Systems*, 2nd Edition, Addison-Wesley, 1977.

[EFR66] Efroymson, M. A., and Ray, T. C., "A Branch and Bound Algorithm for Plant Location", *Operations Research*, May-June 1966, pp. 361-368.

[ESW74] Eswaran, K. P., "Placement of Records in a File and File Allocation in a Computer Network", *Information Processing, 74*, IFIPS, North Holland Publishing Co., 1974.

[GEO72] Geoffrion, A. M. and Marsten, R. E., "Integer Programming: A Framework and State-of-the-Art Survey", *Management Science*, Vol. 18, No. 9, May, 1972, pp. 465-491.

[GRA77] Grapa, E., Belford, G. G., "Some Theorems to Aid in Solving the File Allocation Problem", *CACM*, Vol. 20, No. 11, Nov. 1977, pp. 878-882.

[KAR72] Karp, R. M., "Reducibility among Combinatorial Problems", *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher eds., Plenum Press, New York, 1972, pp. 85-104.

[KUE63] Kuehn, A. A., and Hamburger, M. J., "A Heuristic Program for Locating Warehouses", *Management Science*, Vol. 9, No. 4, July 1963, pp. 643-666.

[LEV74] Levin, K. D., *Organizing Distributed Data Bases in Computer Networks*, Ph.D. Dissertation, University of Pennsylvania, 1974.

[LEV75] Levin, K. D., Morgan, H. L., "Optimizing Distributed Data Bases-A Framework for Research", *Proc. NCC*, 1975, pp. 473-478.

[LOO75] Loomis, M. E. S., *Data Base Design: Object Distribution and Resource Constrained Task Scheduling*, Ph.D. Dissertation, Comp. Sci. Dept., UCLA, 1975.

[LOO76] Loomis, M. E. S., and Popek, G. J., "A Model for Data Base Distribution", *Comp. Networks: Trends and Applications, 1976*, IEEE, pp. 162-169.

[MAH76] Mahmoud, S., Riordon, J. S., "Optimal Allocation of Resources in Distributed Information Networks", *ACM Trans. on Data Base Systems*, Vol. 1, No. 1, March 1976, pp. 66-78.

[MOR77] Morgan, H. L., and Levin, K. D., "Optimal Program and Data Locations in Computer Networks", *CACM*, Vol. 20, No. 5, May, 1977, pp. 315-322.

[RAM76] Ramamoorthy, C. V., and Krishnarao, T., "The Design Issues in Distributed Computer Systems", *Inftotech State of the Art Report on Distributed Systems, 1976*, pp. 375-400.

[RAM79a] Ramamoorthy, C. V., and Wah, B. W., "File Placements of Relations in a Distributed Relational Data Base", *Proc. First International Conference on Distributed Computer Systems*, Huntsville, Alabama, Oct. 1979.

[RAM79b] Ramamoorthy, C. V., and Wah, B. W., "The Isomorphism between File Placement and Warehouse Location", submitted for publication.

[SA 69] Sa, G., "Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem", *Operations Research*, Vol. 17, No. 6, Nov-Dec 1969, pp. 1005-1016.

[SIC77] Sickle, L. V., and Chandy, K. M., "Computational Complexity of Network Design Algorithms", *Information Processing 77*, IFIPS, North Holland Publishing Co., 1977.

[WAH79] Wah, B. W. *A Systematic Approach to the Management of Data on Distributed Databases* Ph. D. Dissertation, University of California, Berkeley, 1979.

Notations defined in this thesis for file a	Casey's notations	Explanation
I	I	= index set of nodes with a copy of the file;
n	n	= number of nodes in the DCS;
U_j	ψ_j	= update load originating at node j per unit time;
Q_j	λ_j	= query load originating at node j per unit time;
$S_{j,k}$	$d_{j,k}$	= cost of communication of one query unit from j to k;
$M_{j,k}$	$d'_{j,k}$	= cost of communication of one update unit from j to k;
F_k	σ_k	= storage cost of file at k per unit time.

Table 1 Mapping between the Defined Notations in this paper and Casey's Notations [CAS72]

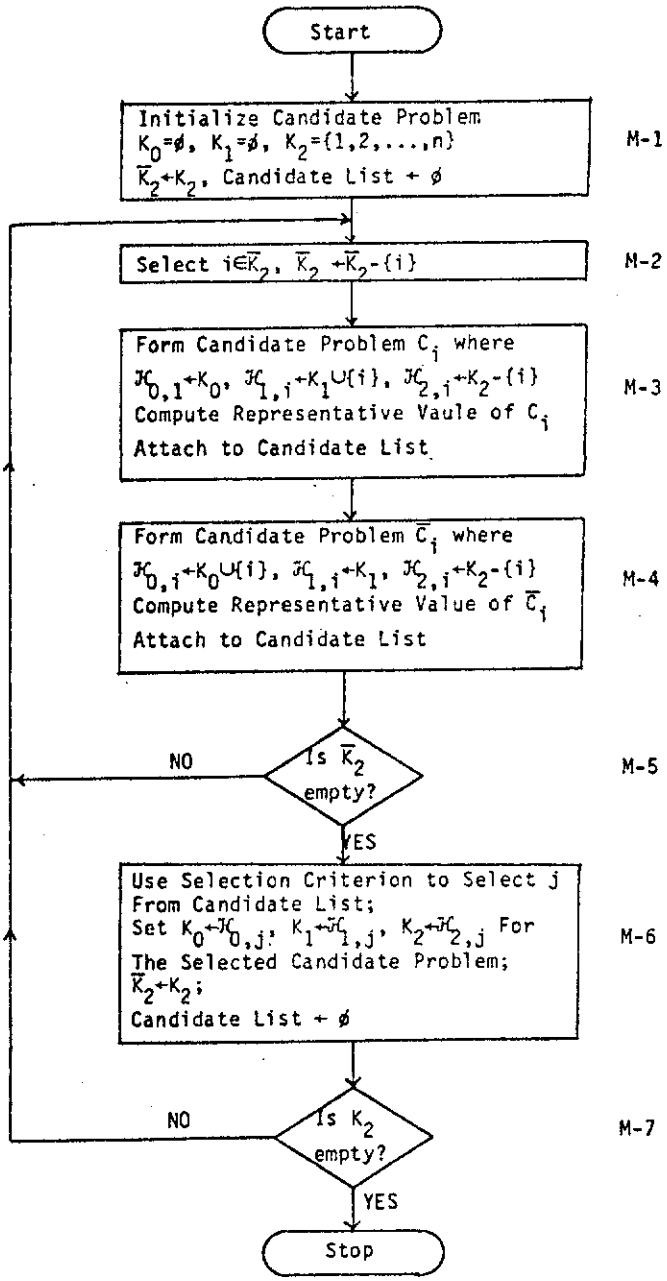

Figure 1 File Assignment Algorithm

Condition	Rule
a	$Y_i=1$ if $\sum_{j=1}^{n} Q_j \min_{k \in K_1 \cup K_2} (S_{j,k}-S_{j,i})_+ > C_i$
b	$Y_i=0$ if $\sum_{j=1}^{n} Q_j \min_{k \in K_1} (S_{j,k}-S_{j,i})_+ < C_i$
c	If $\min_{k \in K_1} (S_{j,k}-S_{j,i}) < 0 \quad j \in \{1,\ldots,n\}$, then n_i is reduced by 1
d	$Y_i=0$ if $C_i-C_k > \sum_{j=1}^{n} Q_j (S_{j,k}-S_{j,i})_+$

Table 2 Summary of Conditions for Placement and Non-placement of a file at node $i \in K_2$ (The first three conditions are from [EFR66]; the last condition is from [GRA77])

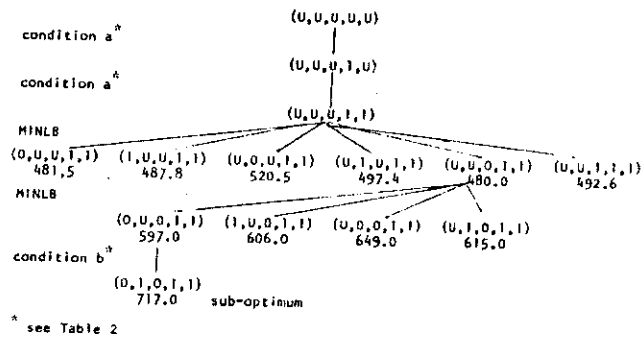

Figure 2a Evaluation of Casey's 5 node Example using MINLB (U indicates that the node is unassigned)

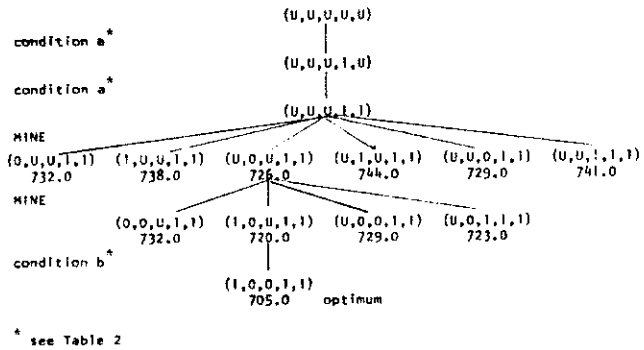

Figure 2b Evaluation of Casey's 5 node Example using MINE (U indicates that the node is unassigned)

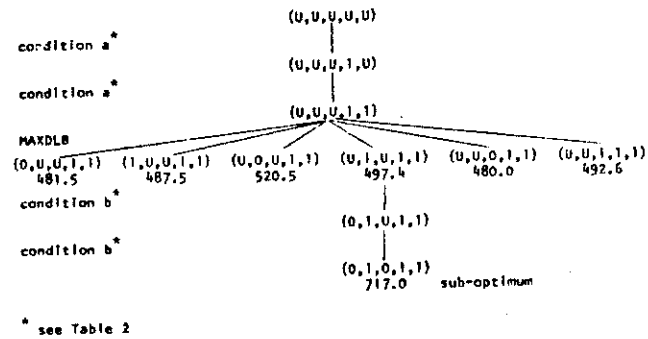

Figure 2c Evaluation of Casey's 5 node Example using MAXDLB (U indicates that the node is unassigned)

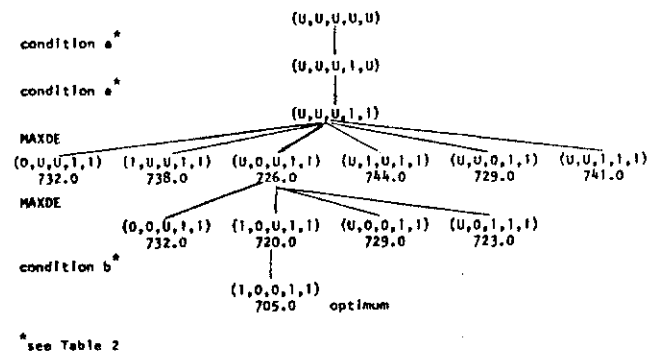

Figure 2d Evaluation of Casey's 5 node Example using MAXDE (U indicates that the node is unassigned)

Prob.	Optimum Sol.	Add-Drop	MINLB	MINE	MAXDLB	MAXDE	Comments	
1	117596	0	0	0	8.43	0	α=0.1	Casey's 19
2	188736	0.03	0.31	0.31	0.31	0.31	α=0.2	node file
3	242581	0	0	0.66	0	0.66	α=0.3	allocation
4	291790	0	1.39	0	1.39	0	α=0.4	problem
5	431720	0	0	0	0	0	α=1.0	[CAS72]
6	705	0.85	1.70	0	1.70	0	Casey's 5 no-	de ex. [CAS72]
7	796648	0.11	0	0.78	0	0.78	Factory	keuhn and
8	854704	0.15	0.09	0.89	0	0.89	at Ind-	Hamburger's
9	893782	0.14	0	0.71	0	0.71	ianapolis	24 ware-
10	928942	0	0.61	0.94	1.49	0.99		houses, 50
11	1092916	0.08	0	0.13	0.10	0.13	Factory	customers
12	1145923	0.13	0	0.22	0	0.22	at Jack-	warehouse
13	1186241	0.13	0	1.37	0	1.37	sonville	location
14	1244991	0.22	0.22	2.49	0	1.67		problem
15	614548	0.14	0	0.90	0	0.90	Factory	[KEU63]
16	859983	0	0.12	0.80	0	0.80	at Balt-	
17	690746	0.03	0	0.74	0	0.74	imore and	
18	724886	0	0	0.42	0	0.49	Ind'polis	
19	806145	0	0	0.88	0	0.38	Factory at	Problem 7
20	870792	0.15	0	0.67	0	0.67	Ind'polis,	of Sa
21	919994	0.11	0	1.46	0	0.44	but not	[SA 69]
22	970446	0	0.42	1.73	1.36	0.67	warehouse	
	mean	0.10	0.22	0.73	0.67	0.58		
	std.dev.	0.18	0.46	0.62	1.83	0.44		

Table 3 % Deviations of File Allocation Heuristic from Optimal Solutions

Prob.	Add-Drop	MINLB	MINE	MAXDLB	MAXDE	Comments	
1	0.57	11.45	22.79	11.42	103.71	α=0.1	Casey's 19
2	0.43	11.59	23.57	11.66	105.57	α=0.2	node file
3	0.43	11.77	23.60	11.76	105.50	α=0.3	allocation
4	0.43	11.80	23.48	11.79	105.26	α=0.4	problem
5	0.29	11.80	23.80	11.84	105.10	α=1.0	[CAS72]
6	0.04	0.08	0.09	0.06	0.24	Casey's 5 no-	de ex. [CAS72]
7	11.46	8.08	11.85	8.29	26.41	Factory	keuhn and
8	9.36	13.55	13.52	11.23	35.73	at Ind-	Hamburger's
9	5.34	20.89	13.91	20.99	37.61	ianapolis	24 ware-
10	3.61	8.48	8.29	8.50	21.42		houses, 50
11	12.08	6.40	9.13	6.39	17.74	Factory	customers
12	8.62	12.66	12.64	12.71	30.62	at Jack-	warehouse
13	7.82	22.16	21.26	22.83	62.03	sonville	location
14	7.02	40.18	33.47	40.33	112.93		problem
15	9.75	5.48	12.44	5.50	25.35	Factory	[KEU63]
16	7.33	5.49	4.37	4.64	7.90	at Balt-	
17	5.58	6.82	7.01	6.84	17.25	imore and	
18	3.79	2.66	3.75	2.68	7.26	Ind'polis	
19	12.24	4.94	9.16	4.95	16.04	Factory at	Problem 7
20	9.02	13.63	13.81	11.29	34.39	Ind'polis,	of Sa
21	6.76	23.27	22.05	20.97	67.74	but not	[SA 69]
22	5.94	22.79	28.61	22.02	73.40	warehouse	
mean	5.81	12.54	15.58	12.21	50.87		
std.dev.	4.11	8.92	8.83	8.64	39.27		

Table 4 Execution Time of Heuristic in seconds on the CDC6400