DISCUSSION PAPER NO. 58

"AN INTEGER GENERALIZED TRANSPORTATION MODEL FOR OPTIMAL JOB ASSIGNMENT IN COMPUTER NETWORKS"*

by

V. Balachandran**

October 25, 1973

Revised October 1974

* Awarded I prize in the 3rd Annual Student Paper Competition of Operations Research Society of America, May 1973.

** Graduate School of Management, Northwestern University

ABSTRACT

This paper investigates assignment of tasks characterized by a parameter-matrix in a network of functionally similar computers. This is formulated by a periodic review model utilizing Boolean variables, based on optimization of an "utility" function over a constraint set. A computationally efficient integer-generalized transportation model is applicable due to existence of relative efficiencies of computers for jobs. Since a job is to be processed exclusively by one computer, it is shown that an optimal solution to the problem posed above, is a basic feasible solution to a generalized transportation problem that is slightly modified. Then a branch and bound solution procedure is used to prevent splitting of a job among computers. An algorithm with computational results is also provided.

1. INTRODUCTION

Many new and interesting network topologies are presently in the design or implementation phases of development. A number of models have been used to help with node location and link capacity decisions. (See, for example, the article about the ARPA network by Frank, Kahn, and Kleinrock) [15]. This paper examines the operational problem of dynamic job allocation in a network with functionally similar nodes. A computer network is an interconnected group of independent computer systems which communicate with each other and share resources such as programs, data, hardware and software. A survey of current networks, the number and types of computers existing in these networks, their topology, functions, management and other details are given in a survey of networks by Peterson and Veit [22]. A detailed bibliography of computer networks are given at the end of the above paper. Theoretical insights of computer networks are given in "Computer Networks" edited by Randall Rustin [23].

Some networks have as a primary goal the linking of functionally different and specialized computing facilities to avoid duplication of unique hardware configurations. Global job scheduling in such environments will not normally be a difficult problem because most programs would be designed for only one of these specialized systems and the network would be used to distribute the unique capabilities of each node over a large area. Of course each computer would have to schedule its own local job mix and allocate its resources while satisfying the network demands.

In a network of functionally similar nodes the allocation of jobs can become a major problem. The nodes may be distributed over a wide geographical

area or be localized. An example of this type of design would be a set of cooperating general purpose computers in which each machine could solve many of the problems submitted to the network although it would probably hold a competitive advantage over other computers on a specialized subset of the jobs. Load sharing between machines would be common in this design. A program initiated at a node that was temporarily overloaded would be sent to another node having similar functional capabilities but which was temporarily underloaded.

Although load sharing implies nodes that have many identical functional capabilities, it does not imply that the physical equipment need be identical. Obviously the easiest way of constructing a network designed for a high degree of load sharing is to use identical operating systems and identical hardware configurations at each node. However, equipment families such as the IBM 360 and 370 lines strive to provide functional compatibilities across a wide range of hardware models. Standardization of higher level languages is another trend that will improve the prospects of load sharing in the future. For example, the Carnegie-Mellon University Computation Center uses a mini-computer as the central communications switch linking a large number of terminals and remote input devices to an IBM 360/67 and a Univac 1108. A user may write and debug a FORTRAN program on the interactive time-sharing systems residing on the 360/67 and then execute long production runs on the 1108. Alternatively, a user may run the same FORTRAN program on either machine depending on the primary memory requirements of his job. In addition to load sharing, local or distributed networks of functionally homogeneous systems may increase total system reliability, provide better communication among users, and provide peak computing power for short periods to a single task if it may be split

into parallel subtasks. (See Bowden [9]).

2. THE MODEL

Consider a network of n computers, to be called hereafter as nodes, and a set of links that span these nodes. Jobs will be characterized by a vector of requirements, $\underline{R}_{jk}$, in which each of m components. is an estimated requirement of job j_k.the j-th job to be processed originating from node k. for a resource such as CPU time, primary memory, a certain compiler. tape units, or secondary memory. The amount of processing required by job j_k at node i will be of special interest and will be denoted by p_{ijk}. The user gives an expected processing time p_{kjk} assuming the job j_k is preferred to be processed at node k itself. He also gives a requirement vector indicating the storage space needed, other I/O requirements etc. The network 'supervisor' (a hardware) can convert all these requirement specifications in terms of those parameters for any other computer i if it is decided that the job j_k is scheduled for processing at node i for any job category (category decided by the requirement vector given by the user). In particular, the processing time p_{kjk} (given) of job j_k at node k can be converted to that at any other node i by multiplying this p_{kjk} by m_{ijk}, a factor which gives the speed of computer k relative to computer i, for job j_k($m_{kik} \equiv 1$, so that $p_{ijk} = m_{ijk}p_{kjk}$). The discussion of the case where this conversion factor m_{ijk} is independent of jobs j_k and depends only upon nodes k and i has been dealt by this author and J.W.McCredie, O.I.Mikhail [1]. However it was seen that this assumption is crucial in the optimization as the job j_k plays a vital-role in a network when multi-processing is quite common. This paper eseentially deals with the more general case where

$m_{ijk} \neq m_{i\ell k}$ where j_k and ℓ_k are two distinct jobs submitted at node k. However, if $m_{ijk} = m_{i\ell k}$ for all jobs j_k, ℓ_k submitted at node k the problem reduces to that given in [1]. The functional capabilities of node i will be represented as a vector $\underline{C}_i$ in which each of the m elements will be the capacity of the i-th node with respect to one of the m resources. Job j_k will be deemed compatible for node i if and only if $\underline{R}_{jk} \leq \underline{C}_i$. The following definitions will be used in all of the formulations of the allocation problems.

n: total number of computers in the network

$I = \{1,2,3,\ldots,i,\ldots,n\}$

the set of all terminal nodes, i(sources where the jobs are actually processed)

$K = \{1,2,3,\ldots,k,\ldots,n\}$

the set of all originating nodes, k (origins)

$J_k = \{1,2,\ldots,j_k,\ldots,g_k\}$

the set of all jobs j submitted at node k and still in the network

$J = \{1,2,\ldots,g_1,g_1+1,\ldots,g_1+g_2,\ldots,\sum_{k=1}^{n} g_k\}$, the set of all jobs

$N = |J|$, the cardinality of the set J, the total number of jobs in the entire network

i: a terminal node (a computer in the network), $i \in I$

k: an originating node (a computer in the network), $k \in K$

(Note: i can be equal to k.)

j_k: j-th job submitted at k-th node, $j_k \in J_k$

$\underline{R}_{jk}$: a vector of m components r_{jkw} where r_{jkw} is the estimated w-th requirement of job j_k

$\underline{C}_i$: a vector of m components c_{iw} where c_{iw} is the maximum capacity

of the i-th node with respect to resource w

m_{ijk}: a constant multiplication factor available at the network center which gives the relative speed of processing of job j_k at computer i to that at computer k.

p_{ijk}: processing time of job j_k at i-th node ($p_{ijk} = m_{ijk} \cdot p_{kjk}$)

v_{ijk}: utility (a measure defined and known based on cost, time priority, etc.) of job j_k processed at i-th node

t_{ijk}: time to transmit job j_k to node i

Note: It will be a very high time if it is impossible to transmit job j_k to node i due to either nonexistence of link (i,k) or due to the capacity of link (i,k). (We assume $t_{kjk} \equiv 0$.)

ℓ_{ik}: capacity of link (i,k)

d_{ijk}: expected waiting time of job j_k at node i (delay)

q_i: cost per unit time of processing at node i (includes I/O cost)

q'_{ik}: cost per unit time of transmission through link (i.k). It is assumed that $q'_{kk} = 0$) (This cost includes both onward and return transmission).

x_{ijk}: = 1 if job j_k is processed at node i .
= 0 otherwise.

T_i: total amount of processing time currently assigned to node i (periodic decision time). Generally $T_i = T$ for all $i \in I$ where T is the planning horizon - (periodicity)

TR_i: expected time for onward transmission to node i from all other nodes whenever node i is idle. If one assumes that the transmission time is much smaller than actual processing time, then the computer i never

waits for a job to be transmitted. In other words, by the time the i-th computer has processed a job, the next job assigned to be processed has already been transmitted. This implies that $TR_i = 0$. This assumption is a reality in every computer network in USA [20].

$TT_i = T_i - TR_i$ (note if we assume that $TR_i = 0$ as mentioned above then $TT_i = T_i$).

The following sections of the paper investigate different approaches to the problem of dynamically allocating jobs to the different nodes. The general philosophy is one of periodic review. Jobs are submitted to the network and an allocation decision is made periodically. (A discussion of this periodicity, T, is given in section 7). At each decision point all jobs currently in the network are assigned by the algorithm under study so that some objective function is improved and all constraints are met. All the computer networks that are existing (and will be installed in future) [22] can broadly be divided into mainly three categories as far as network management and users are concerned. These are given below:

(i) Governmental Organizations (e.g. ARPA, OCTOPUS, DLS, etc.)

(ii) Quasi Governmental and educational institutions (e.g. MERIT, TUCC, TSS, etc.)

(iii) Private and Commercial enterprises (e.g. CYBERNET, etc.)

For those users under category (i), the network is already established for them and hence cost is of less or no concern. Their primary objective is to get the maximum number of jobs done or get the maximum utility of the network, which we call "maximizing thruput" or "maximizing total value." These users are also interested in minimum turnaround time. For those in the second

category a total budget might be allotted and they may get an appropriation of the budget for their use. Thus these users in addition to the above two questions, give less weight to cost considerations. But the users and management in the third category which operates purely on a commercial basis give very high importance to costs of computer time and are prepared to wait a little if it is comparatively inexpensive. Thus we will have three objective functions given later as (10). (16), and (18) which can be combined to one utility function as given in (19). Assume that any user (or management) can come up with relative preference factors (weights) u_1. u_2, u_3, for these three objectives, such that $0 \leq u_1 \leq 1$ and $\sum_{i=1}^{3} u_i = 1$ so that an overall objective utility function as given by (19) is derived. Then this function can be optimized based on a constraint set given later, (7)-(9). In the rest of the paper these different formulations and solution procedures are given.

3. INTEGER PROGRAMMING FORMULATIONS

A primary consideration for the management of the network is to maximize the total number of jobs processed at various nodes within a given chosen period when a periodic review system is implemented. Two cases are considered. For the first, all jobs are treated equally, and for the second, a value for each job is given based on the type of customer, importance of job, etc. In these two models the constraints are mainly compatibility of the job at a node to be processed and secondly the fixed time period which is available for each node. This model will be formulated as a 0-1 integer program (IP). Instead

of specifically including the compatibility constraints as constraints in the IP formulation, the corresponding variable will be defined to be zero. In other words, if $\underline{R}_{jk} > \underline{C}_i$ for node $i \in I$, then $x_{ijk} = 0$. Due to this device we are left with only time constraints. Further we do not want a job to be processed at more than one node. Thus the IP formulation for maximum thruput from the management point of view when all jobs valued equally will be:

(1) Maximize $\sum_{i\in I} \sum_{k\in K} \sum_{j_k \in J_k} x_{ijk}$

subject to

(2) $\sum_{k\in K} \sum_{j_k \in J_k} p_{ijk} x_{ijk} \leq TT_i$ for each $i \in I$

(where $P_{ijk} = m_{ijk} P_{kjk}$)

(3) $\sum_{i\in I} x_{ijk} \leq 1$ for each $j_k \in J_k$ and $k \in K$

($\underline{R}_{jk} \leq \underline{C}_j$ is implicit)

(It may not be feasible for all jobs to be processed within the available time.)

(4) and $x_{ijk} = 0, 1$ for $i \in I$, $j_k \in J_k$

In this formulation we need only P_{kjk}, the processing time needed for each job j_k at the originating node k. However it is possible to find P_{ijk} needed in (2) with the use of the conversion factor m_{ijk} existing at the network. Thus the data required to compute the processing times of all nodes $i \in I$ can ge generated internally by the network "supervisor". There can be times during the planning horizon T. when node i may be dedicated to specific users or planned or maintenance. If such times. if any. are deducted we have available time T_i for $i \in I$. Sometimes, a node i is idle since the jobs earmarked to be processed at i from other nodes have not reached due to transmission. Such times can be accumulated within planning horizon T and this time is generally called the "time for onward transmission to node i from all other nodes whenever node i is to be idle." Based on theoretical developments given by Jackson [19], Conway, Maxwell and Miller [10], or Frank [14], estimates of such expected nonoverlapping transmission times are obtained based on past data of the queueing process. Generally TR_i is comparitively much smaller to T. Since the intent of the paper is not on the time-dependent queueing processes, these are not discussed here and interested readers can get relevant materials from the references given above. Thus the available time for the i-th computer is $TT_i = T_L - TR_i$ as given in equation (2).

A slight modification to the maximum thruput formulation is to maximize the total value of the jobs. Instead of assuming that all jobs are treated equally, values are assigned for each job at each node based upon (management's)customer preference. job priority, node preference and other management considerations. If a job j_k is not compatible for node i, then the variable x_{ijk} is set to zero as mentioned earlier. The objective function for this formulation becomes:

(5) Maximize $\sum_{i \in I} \sum_{k \in K} \sum_{j_k \in J_k} v_{ijk} x_{ijk}$

Computer codes for (0,1) programs are available to solve these two problems. (e.g. Balas [7], Geoffrion [17]). However the number of variables which we are considering is so huge that it is computationally infeasible by these IP codes. (e.g. In a network of 5 computers and 100 jobs in each we have 2500 variables). The procedure given in Section (5) is not only computationally feasible but also efficient.

4. NETWORK FLOW PROBLEM

It is easy to see that both the previous problems can be formulated as network-flow problems [1], if constraint (4) is removed from the formulation for each $j_k \in J_k$ and $k \in K$. Also, it is known that the flow problem is computationally more efficient as compared to a 0-1 integer program. If the flow problem yields a solution which satisfies constraint (4) for each job, then the solution is optimal for the 0-1 problem and is computationally efficient. However, the formulation has no guarantee that a job will not be processed (split) at more than one computer since x_{ijk} may be fractional. If it is not split, then the minimal cost flow problem is equivalent to max thruput. Thus if we are satisfied with a near optimal solution, we can manipulate the solution by some heuristic and eliminate the splits.

5. AN INTEGER GENERALIZED TRANSPORTATION PROBLEM

The main drawback of the network flow problem [13] was the splitting of a job j_k to be processed at more than one node. If this problem is formulated as an ordinary transportation problem (OTP) with rows corresponding to

computers and columns corresponding to jobs then due to totally unimodular nature of the OTP, the solution will be all integer. However, since each job has to be processed by only one source, we need exactly one basis element per column (job) whose value is the column total. This special kind of transportation problem has been discussed by Srinivasan and Thompson [26]. An implicit enumeration approach was used by DeMaio and Roveda [11] for solving the same problem. It was shown [26] that the former approach is computationally more efficient than the later, in view of the fact that the number of rows is much smaller than the number of columns in the OTP Tableau. However, neither of these approaches are feasible since the relative efficiency of the computers are job dependent. In other words m_{ijk} is not always equal to $m_{i\ell k}$ when $j \neq \ell$. Hence the ordinary transportation model is not applicable in this environment. Due to the presence of relative efficiencies m_{ijk}, we will have a generalized transportation model. (GTP)

Consider the set of computers (nodes) $I = \{1,\ldots,n\}$ having a fixed available processing time $TT_i = (T_i - TR_i)$ as given in Equation (2) and a set of jobs $J = \{1,2,\ldots,g_1,g_1+1,\ldots,g_1+g_2,\ldots,\sum_{k=1}^{n} g_k\}$ with $|J| = N$,

($|\ |$ represents the cardinality) with known demands of processing time $p_{ijk} = m_{ijk} \cdot p_{kjk}$. In the nomenclature of the Generalized Transportation Problem (GTP) [2] let us use only two subscripts, i for rows $i \in I$ (computer time constraints) and j for columns for $j \in J$ (jobs). Thus with this two subscripted notation $p_{ijk} = p_{ij}$ where job j_k is replaced by job j. It is to be noticed that we also know the originating node k

since the value of j ($j=1,\ldots,N$) can identify the node k. Thus the model II, viz, maximize thruput by maximizing total value given by the objective function (5) and constraints (2)-(4) will be

(6) Minimize $Z_v = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$

where $c_{ij} = -v_{ij}$ (In, model I, $v_{ij} = 1$ for all i,j)

Subject to the constraints

(7) $\sum_{j \in J} p_{ij} x_{ij} \leq TT_i$ for $i \in I$

where $TT_i = T_i - TR_i$

(8) $\sum_{i \in I} x_{ij} \leq 1$ for $j \in J$

(9) $x_{ij} = 0$ or 1 for $i \in I$, $j \in J$.

Following the GTP tableau terminology given in GTP literature [2,8,12] create a slack column $N + 1$, without a constraint codresponding to this column and a fictitious (slack) row (computer) $n + 1$ with abundant available time TT_{n+1}. This fictitious computer, thus can process any job that can not be processed by any existing source $i \in I$. To convert problem (6)-(8) to the standard GTP format [2,12] as presented by problem (10)-(13), the following are defined.

Sets: $I' = I \cup \{(n+1)\}$ and $J' = J \cup \{(N+1)\}$

$p_{ij} = 1$ for $i = n+1$ and $j \in J'$

$c_{ij} = M$(a large positive constant) for $i = n+1$ and $j = J$

$p_{ij} = 1$; $c_{ij} = -M$; for $i \in I'$ and $j = N+1$

$TT_{n+1} = M'$ (a large positive constant)

With these additional data defined above, we get an equivalent problem (10)-(14) given below in the standard GTP format, which ensures always a feasible solution. (For details see [2,12]).

(10) Minimize $Z_v = \sum_{i \in I'} \sum_{j \in J'} c_{ij} x_{ij}$

subject to the constraints

(11) $\sum_{j \in J'} p_{ij} x_{ij} = TT_i$ for $i \in I'$

(12) $\sum_{i \in I'} x_{ij} = 1$ for $j \in J$

(Note that we are not putting $j \in J'$ above)

(13) $x_{ij} \geq 0$ for $i \in I'$ and $j \in J'$ and

(14) $x_{ij} = 0$, or 1 for $i \in I'$, $j \in J$

(Note that $x_{i,N+1}$ can be fractional)

The problem (10)-(13) is a standard GTP, and hence can be solved by the "generalized stepping stone method" as shown in [8, 12]. We assume the reader is familiar with the usual terminology of GTP [2,8,12], that a <u>cell</u> is an index pair (i,j) with <u>row</u> (computer) $i \in I'$ and column (job) $j \in K'$, a <u>basis</u> B to the problem (10)-(13) is a collection of $(m + N + 1)$ cells defined as a <u>one-forest</u> (see definition (11)in [2]) consisting of mutually disconnected <u>one-trees</u> where each one-tree, a tree with one extra edge, contains exactly one <u>cycle</u> or one <u>loop</u> [2,8,12]. (We follow the terminology used in [2], though similar terminologies are used in [8,12]. A solution $X = \{x_{ij}\}$ is basic if $x_{ij} = 0$ for $(i,j) \notin B$. A basic solution is <u>feasible</u> if the $\{x_{ij}\}$ satisfy the constraints (11)-(13). It is well known [8,12] that the generalized

stepping stone algorithm yields a basic optimal solution to the problem (10)-(13).

DEFINITION 1. We define P to be the standard GTP (10)-(13) and IP to be the Integer Generalized Transportation Problem (IFTP) (10)-(14). (note that $x_{i,N+1}$ need not be integers). We define a basis to (10)-(13) to be unique-sourced, if corresponding to every column (job) $j \in J$, there is a unique source (row) i_j such that $(i,j) \in B$ if and only if $i = i_j$.

It is shown in [2, 8, 12] that B has n+N+1 cells. Since a unique-sourced basis has exactly one cell for each column $j \in J$, it follows that the slack column (N+1) must have the remaining n+1 cells; i.e. $(i,N+1) \in B$ for every $i \in I'$. Thus the unique-sourced basis consists of N+1 mutually disconnected one-trees where each one-tree contains a unique loop (a cell in the slack column) [2]. It is known that the solution of a GTP need not be all-integer since the total-unimodularity property is not valid due to the presence of coefficients p_{ij} in (11). However since each column total is exactly unity and we are requiring a unique-sourced basis, each x_{ij} for every $j \in J$ is exactly either 0 or 1. Thus, for each $j \in J$ there will be only one x_{ij} which gets a value of 1, say $i = r$, so that every other x_{ij}, $i \neq r$ will be identically equal to zero. Thus the solution $X = \{x_{ij}\}$ will be all-integer and hence we call it as integer generalized transportation problem (IGTP). It is to be seen from [2, 8, 1]], that the two-tree (see equation 10 in [2]), created, while pivoting, consists of two loops (generally) which show the computational efficiency since only cells of the two-tree change while the rest of the basic cells are not affected (See algorithm 4 of [2]). (We don't have cycles (definition (2) in [2]) at the optimum.)

Theorem 1, below, establishes the relationship between problems P (GTP) and IP (IGTP).

THEOREM 1. There is a one-to-one mapping between feasible solutions to IP and unique-sourced basic feasible solutions to P.

PROOF: Consider any feasible solution $\{x_{ij}\}$ to IP. By (12)-(14), and since the column total for each $j \in J$ is $\equiv 1$, it follows that corresponding to every job $j \in J$ there is a unique computer i_j such that $x_{ij} > 0$, if and only if $i = i_j$. Corresponding to this solution let B be defined to be the set of $(n + N + 1)$ cells $\{[(i_j,j) \text{ for } j \in J] \cup [(i, N+1) \text{ for } i \in I]\}$. Thus, the solution $\{x_{ij}\}$ is basic since $x_{ij} = 0$ for $(i,j) \notin B$. It is a feasible solution for P since $\{x_{ij}\}$ is feasible for IP. This correspondence is unique, since B is defined uniquely.

To prove the converse, let us assume that a unique-sourced basic feasible solution $\{x_{ij}\}$ to P exists. By (12) and (13) and due to unique-sourcedness, it follows that corresponding to every column $j \in J$ there is an unique row i_j such that $x_{ij} = 1$ if $i = i_j$ and zero elsewhere. Thus (14) is satisfied and from (11)-(13) it follows that $\{x_{ij}\}$ is feasible to IP as well. Further this correspondence is unique due to (11)-(14).

By Theorem 1 and from the fact that the problem P and IP share the same objective function (10) it is easy to see the following:

COROLLARY 1. There is a one-to-one correspondence between optimal solutions to IP and those optima among the unique-sourced basic feasible solutions to P.

We will now provide a solution procedure to IP in parallel to the subtour-elimination algorithms for the travelling-salesman problem [23]. This alogirthm is basically a branch and bound method which begins by partitioning the set of unique-sourced basic feasible solutions and then calculating the lower bounds on the costs of all solutions in a subset and improving the lower bound until we get the optimal solution. The

initial bound is found by solving the ordinary GTP given by (10)-(13) [2, 8, 12]. If the basic optimal solution is unique-sourced, then we have an optimal solution to IP (Corr. 1). Assume it is not. Let t be one of the columns $j \in J$ which has more than one cell $\in B$ and let (s,t) be one such cell. (Though any such cell (s,t) can be chosen, a heuristic for a good choice of (s,t) for computational efficiency is given later). This choice of (s,t) leads to two branches (sub-problems):

(a) the subset in which (s,t) is a cell in the optimum to IP given by a unique-sourced basic feasible solution of P.

(b) the subset in which (s,t) is not a cell in the optimum to IP.

The two new GTP corresponding to (a) and (b) are solved to determine the lower bounds for all unique-sourced basic optimal solutions in their respective subsets. If the optimal solution corresponding to any one subset is unique-sourced and the cost of this solution is less than or equal to the lower bounds on all other subsets then such a solution is optimal. If not, then one selects that subset having the smallest lower bound and branches again into two subproblems. Eventually this algorithm assures of finding an optimum unique-sourced basic optimal solution to IP due to Corollary 1.

Certain discussions of the above procedure are now relevant. First, it is clear that the procedure for branching on a non unique-sourced basis excludes that basis from the two subsets but does not exclude any unique-sourced basis. The algorithm converges infinite number of steps since the total number of bases is finite and excludes at least one basis at each iteration. Secondly, the branching procedure results in a partition of the unique-sourced basic feasible solutions in that subset and hence the algorithm

can be expected to be computationally efficient, since the size of the problem of each subset keeps decreasing as shown in the following sentences. Third and most importantly, for the subproblem with (s,t) required to be in the optimal solution, by unique-sourcedness (s,t) is the only cell in column t. Hence we can eliminate column t from further consideration and thus modify TT_s by $TT_s - p_{st}$ and solve a reduced GTP. This reduction in TT_s may further simplify the problem since the cells (s,t) for which the value $p_{s,j}$ is greater than the new value of TT_s cannot possibly be in the optimum solution (such cells can be eliminated by defining $c_{sj} = \infty$). Fourth, the optimal solutions to subproblems of branches can be obtained by applying the cost operator [3] (a part of operator theory [3,5] which obtains optimal solutions due to simultaneous variation of several parameters of a given problem when the optimal solution of the same problem is known), so that the new optimal solution of that problem from which the new branches are obtained. Furthermore, the backtracking steps of the branch and bound procedure may also be done in this way. Fifth, a non-unique-sourced basis can have at most n columns (the number of computers in the network) which contain more than one basic cell, since the total number of basic cells in n + N + 1 and there are N + 1 columns which need at least one basic cell per column [2]. Note that the "absorbing cell" (n + 1, N + 1) is always in the basis [2]. (For example, if the network contains 5 computers and we are analysing 1000 jobs in all, we will have

non-unique-sourcedness in at most 5 columns initially.) Consequently, the fraction of maximum number of columns which violate unique-sourcedness (fraction jobs that may be split) is $(n+1)/(N+1)$ Thus in any network environment where the number of computers are much less than the total number of jobs, the algorithm is found to be highly efficient computationally. (See Section 7)

Let us now discuss the choice of (s,t) upon which the branching may take place. Let J^* represent those columns of J that have two or more basic cells. Note that J^* is a very small subset of J. Given a column of $j \in J^*$, it is found computationally efficient (similar to LP pivoting rules) to branch on that basic cell (i,j) for which c_{ij} is the smallest. Then along the branch in which (i,j) is excluded from the optimal solution cost can be expected to increase approximately by $\Delta_j = (c_{aj} - c_{ij})x_{ij}$ where c_{aj} is the next smallest cost of a basic cell in column j and x_{ij} is the amount alloted via the smallest cost basic cell (i,j). Consequently for branching, we can choose the column $t \in J^*$ for which Δ_t is largest and branch on (s,t) where (s,t) has the lowest cost among all basic cells in column t.

Algorithm 1 given below summarizes the above result for the IGTP. (Definition 1)

A1: Algorithm for finding an optimal solution to the unique-sourced generalized transportation problem IGTP, (6)-(9).

(1) Set up the problem P as defined by (10)-(13). Let P_1 denote problem P, and $\Omega_1 = \emptyset$ denote the set of cells required to be included in the optimum solution. Let $\Psi_1 = \emptyset$ denote the set of cells excluded from the optimum solution. Let X_1 be the optimum solution to P_1 with basis B_1 and cost Z_1. (This can be obtained by any algorithm available in the literature [2, 8, 12]). Let $S = \{1\}$ denote the set of problems under consideration and let $m = 1$ denote the total number of problems generated so far.

(2) Choose the problem P_k for which Z_k is the smallest for $k \in S$. If B_k is unique-sourced go to (8). Else go to (3).

(3) Find the set of columns J^* where the basis B_k has more than one basic cell in each column of J^*. Find the two basic cells (i,j) and (a,j) for which the unit costs are the smallest and second smallest respectively. Let $\Delta_j = (c_{aj} - c_{ij})x_{ij}$ and choose the $t \in J^*$ for which Δ_j is largest. Select the basic cell (s, t) with least cost in column t for branching.

(4) Define P_{m+1} as the problem obtained from P_k by <u>including</u> (s, t) to be an additional basic cell i.e., $\Omega_{m+1} = \Omega_k \cup \{(s,t)\}$ and let $\Psi_{m+1} = \Psi_k$. The problem P_{m+1} can be obtained from P_k by dropping column t and defining TT_s to be $TT_s - p_{st}$. For columns j such that $p_{sj} >$ (new) TT_s; set $c_{sj} = M$ (a large positive quantity).

(5) Define P_{m+2} as the problem obtained from P_k by <u>excluding</u> (s, t) from the optimal basis. Set $\Psi_{m+2} = \Psi_k \cup \{(s, t)\}$ (i.e. $c_{s,t} = M$) and let $\Omega_{m+2} = \Omega_k$.

(6) Denote the basic optimal solutions to P_{m+1}, P_{m+2} to be X_{m+1} and X_{m+2} with bases B_{m+1} and B_{m+2}. Define Z_{m+1} = optimal cost to $P_{m+1} + \sum_{(i,j) \in \Omega_{m+1}} c_{ij}$ and Z_{m+2} = optimal cost to $P_{m+2} + \sum_{(i,j) \in \Omega_{m+2}} c_{ij}$.

(7) Drop k from the set S and include $(m+1)$ and $(m+2)$ to S. Let $(m+2)$ replace the old m and go to (2).

(8) The optimal solution to the IGTP given by (6)-(9) is given by X_k and Ω_k with the associated cost = optimum cost to $P_k + \sum_{(i,j)\in\Omega_k} c_{ij}$. Stop.

From a computational point of view, it is not necessary to store the problems P_k for every $k\in S$. It is enough if the sets Ω_k and Ψ_k are stored for $k\in S$. To construct problem P_k from the original problem P, we first set $c_{ij} = M$ for $(i,j)\in \Psi_k$. Next for every $(i,j)\in \Omega_k$, we drop column j and modify TT_i by $TT_i - p_{ij}$. Finally we eliminate those cells (i,j) with $j\in J$ for which $p_{ij} > TT_i$ by defining $c_{ij} = M$. Finally, on a pragmatic note, it is to be emphasized that at most n columns will have more than one cell in the basis. In other words at most n jobs will be split. Since we are concerned with a sequential assignment problem, we can just ignore these jobs that are split, do not proceed for any branch and bound procedure and just process only those jobs which are not split. In other words we can leave those jobs that are split for the next 'batch' of processing probably with a very high 'utility' ($v_{ijk} = M$) this time, so that they are certainly processed in the next "batch", (i.e. $x_{ijk}=1$). Another heuristic is to look at a very small, generalized transportation problem with columns ($\leq n$) corresponding to split jobs and the rows correspond to those which had basic cells in the NN + 1 column with the slack time as the new TT_i. Once again the column totals are unity, while the row total for ith row, (if present) will be

$$T_i - \sum_{(i,j)\in B} P_{ij}.$$

This problem may yield some more columns with just one cell in the basis which correspond to additional jobs that can also be processed. However these are only sub-optimal solutions but reduce total computational effort.

6. OTHER OBJECTIVE FUNCTIONS

In Section 2 when we defined the problem we considered three categories of networks users/management. So far, the objective function was maximizing thruput with values assigned for jobs as given by equation (6) defined by Z_v. However, for networks of the type CYBERNET [22], cost is of prime concern. Let us define the constraint set as given by equations(7), (8), and (9) by C and the constraint set as given by (11)-(14) by C'. Thus for the two other objective functions which are given below, the constraint sets C and C' are identical to that for the objective function given in (6) or (10) respectively. Let q_i represent the cost per unit time of processing at node $i \in I$ (this time includes CPU and IO), t_{ijk} the time to transmit job j_k to node i (as given earlier), and q'_{ik} represent the cost per unit time of transmission through link (i,k) then the cost of processing job j_k at node i is $(q_i p_{ijk} + q'_{ik} t_{ijk})$. Thus the objective function similar to (6) will be (in the double subscripted form)

$$(15)\quad \text{Min } z_c = \sum_{i \in I} \sum_{j \in J} (q_i p_{ij} + q'_{ij} t_{ij})\, x_{ij}$$

subject to the constraint set C. It is to be noted that in equation (15) above the same device of two subscripts is used to reduce the three subscripts, as given in Section 5. Following the discussion of Section 5 this model (15) over the constraint set C can be converted to the IGTP given by (10) over C'. Thus the new objective function equivalent to (15) will be:

$$(16)\quad \text{Min } Z_c = \sum_{i \in I'} \sum_{j \in J'} (q_i p_{ij} + q'_{ij} t_{ij})\, x_{ij}$$

over the constraint set C'. Let the coefficients of x_{ij} in Z_c be r_{ij}. This could be solved as an IGTP following the discussion of Section 5.

A third objective that is relevant to many network managers (especially to those of cateogry (i) and (ii)) will be to minimize the "turnaround" time. Cost of processing is generally of 'little concern' to managers in these categories. An equivalent to the above problem can be obtained by minimizing the overall expected delay. Based on past records and history of the queueing mechanism, utilizing information relative to the arrival process, queue discipline and service process [10, 19], expected waiting time for each job category at each node can be estimated. Following the nomenclature of Section 2, let d_{kij} be this expected delay. Then the new objective function to replace (6) of Section 5 will be:

$$(17)\quad \text{Minimize } Z_d = \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij}$$

subject to the same constraint set C. Again the IGTP objective similar to that of (10) for (17) will be

$$(18)\quad \text{Minimize } Z_d = \sum_{i \in I'} \sum_{j \in J'} d_{ij} x_{ij}$$

over the constraint set C'. This IGTP given by (18) and C' can also be solved.

It is now easy to see that either a bi-criterion or a multiple criterion function could be made utilizing the objective functions (10), (16), and (18) over the constraint set C'. The theoretical discussion of such a multicriterion function over a common constraint set is given by Jeffrion [16], Roy [24] for a continuous variable.

Extension of this concept to Boolean variables is given by H. Pasternak and U. Passy [21]. However the computational time for implementing such a multiple objective criteria is considerably high and economically unprofitable. Consequently, if a network manage can come up with utilities (weights) u_1, u_2, u_3, associated with these objective functions such that $0 \leq u_i \leq 1$ and $\sum_{i=1}^{3} u_i = 1$, then the following weighted objective function may be a good representation: viz

$$(19) \quad \text{Minimize} \quad z_w = \{u_1 z_v + u_2 z_c + u_3 z_d\}$$

over the constraint set C', where z_v, z_c, and z_d are given by (10), (16), and (18) respectively. This problem (19) over C' is an IGTP and can be solved by the method given in Section 5. Parametric analysis for future decisions, if necessary, can be made utilizing the theory given by Balachandran and Thompson [3, 4, 5]. Further, the sensitivity of the planning horizon T_i of the model can be analyzed by the "Operator Theory" utilizing "~~Run~~ Operators" as given in [3,5]. These are left for a future paper.

ALGORITHM FOR INTEGER GENERALIZED TRANSPORTATION PROBLEM

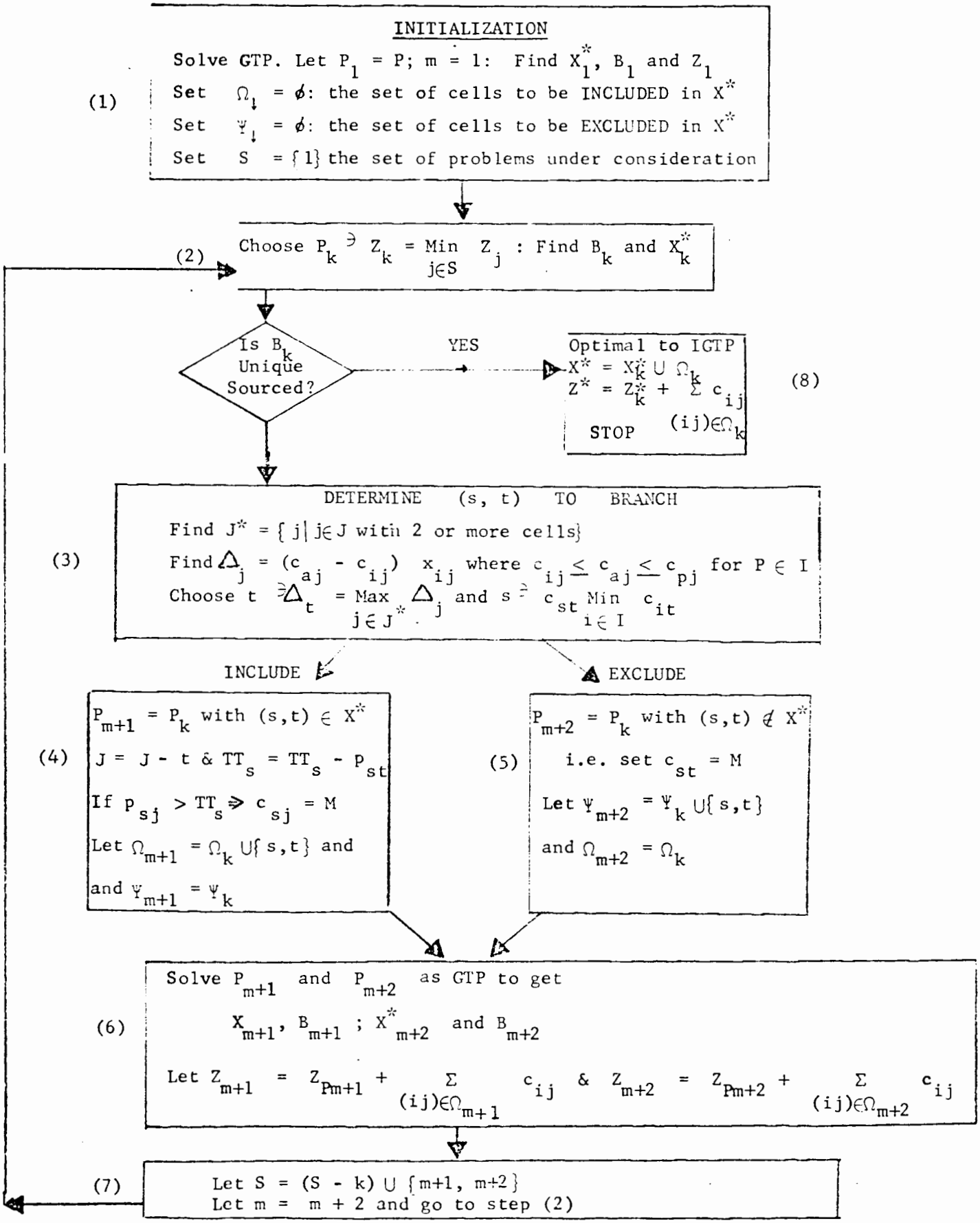

TABLE II

FOUR NODE COMPUTER NETWORK (REAL DATA)

NO.	NO. OF JOBS	HORIZON	NO. OF BRANCHES	CPU TIME
1	426	45 minutes	7	2.932 secs
2	532	45 minutes	8	3.024 secs
3	546	45 minutes	8	3.136 secs
4	548	45 minutes	8	3.128 secs
5	680	90 minutes	8	3.218 secs
6	738	90 minutes	9	3.516 secs
7	746	90 minutes	8	3.221 secs
8	789	90 minutes	9	3.518 secs

TABLE III

COMPUTATIONAL EXPERIENCE FOR 3.5...10 NODE COMPUTER NETWORKS WITH RANDOMLY GENERATED DATA OF p_{ijk} AND m_{ijk} WITH 60 MINUTES PLANNING HORIZON AND 500 JOBS FOR EACH CASE AND THIRTY RUNS.

	NO. OF BRANCHES		CPU TIME	
NO.OF COMPUTERS	MEAN	STD.DEV.	MEAN	STD.DEV
3	6.52	0.63	2.726	.016
5	9.73	0.76	3.893	.023
6	10.16	0.78	4.549	.018
7	12.26	0.92	5.168	.043
8	14.48	0.87	6.713	.028
9	17.31	0.86	7.927	.025
10	20.28	0.54	9.549	.017

ACKNOWLEDGEMENTS

The author sincerely thanks Professors V. Srinivasan and G. L. Thompson for their helpful comments and guidance in the preparation of this paper. The author also expresses his grateful appreciation to both the referees for their excellent and constructive criticisms which are responsible for a significant improvement in the presentation, and the notations used in this paper.

REFERENCES

[1] Balachandran, V., J.W. McCredie and O. Mikhail, "Optimal Job Assignment in Computer Networks," G.S.I.A. Working paper, presented in ORSA-TIMS-AIIE conference, Atlantic City, N.J., November 1972.

[2] Balachandran, V. and G.L. Thompson, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem-I: Basic Theory," Naval Research Logistics Quarterly. Vol. 22, No. 1, March 1975 (Forthcoming).

[3] _____, and _____, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem-II: Rim, Cost and Bound Operators," Naval Research Logistics Quarterly, Vol. 22 No. 1, March 1975 (Forthcoming).

[4] _____, and _____, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem-III: Weight Operators." Naval Research Logistics Quarterly, Vol. 22, No. 2. June 1975 (Forthcoming).

[5] _____, and _____, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem-IV: Global Operators," Naval Research Logistics Quarterly, Vol. 22, No. 2, June 1975 (Forthcoming).

[6] _____, and _____, "An efficient algorithm of the Generalized Transportation Problem with the three index method," Management Science Research Report, Carnegie-Mellon University, Pittsburgh, Pa.

[7] Balas, E., "An Additive Algorithm for Solving Linear Programs with (0-1) Variables," Operations Research, Vol. 13, 1965, pp. 517-546.

[8] Balas, E. and Ivanescu (L.P. Hammer), "On the Generalized Transportation Problem," Management Science, 11, (1964), pp. 188-202.

[9] Bowdon, E., "Priority Assignment in a Network of Computers," IEEE Transactions on Computers, Vol. C-18, No. 11, November 1969.

[10] Conway, R.W., W.L. Maxwell and L.W. Miller, Theory of Scheduling, Addison-Wesley, Reading, Massachusetts, 1967.

[11] DeMaio, A.O., and C.A. Roveda, "An all Zero-One Algorithm for a Certain Class of Transportation Problems," Operations Research, 19, 1971, pp. 1406-1418.

[12] Eiseman, K., "The Generalized Stepping-Stone Method for the Machine Loading Model," Management Science, 11, (1964), pp. 154-176.

[13] Ford, Fulkerson, Flows in Networks, Princeton University Press, 1964, Princeton, N.J.

[14] Frank, H., "Optimal Design of Computer Networks," in Computer Networks edited by R. Randall, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[15] Frank, H., R. Kahn, L. Kleinrock, "Computer Communication Network Design -Experience with Theory and Practice," AFIPS Conference Proceedings, Vol. 40, Spring 1972, pp. 255-270.

[16] Geoffrion, A.M., "Solving Bicriterion Mathematical Programs," Operations Research, Vol. 15, 1967, pp. 39-54

[17] Geoffrion, A.M., and A.B. Nelson, "User's Instruction for 0-1, Integer Linear Programming," The Rand Corporation, RM-5627-PR. May 1968.

[18] Glover, F., D. Karney, D. Klingman and A. Napier,"A Computation Study on Start Procedures, Basis Change Criteria, and Solution Alogithms for Transportation Problems" Management Science, Vol. 20, No. 5, Jan. 1974, pp. 793-813.

[19] Jackson, J., "Networks of Waiting Lines," Operations Research, 5, 1957.

[20] Kleinrock, L., "Survey of Analytical Methods in Queueing Networks," in Computer Networks, edited by R. Randall, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[21] Pasternak, H. and U. Passy, "Bicriterion Mathematical Programs with Boolean Variables," Mimeograph Series #106, Technion, Israel Institute of Technology, HAIFA, ISRAEL.

[22] Peterson, J.J. and S.A. Veit, "Survey of Computer Networks," MTP-357, The MITRE Corporation, September 1971.

[23] Randall, R. (editor), Computer Networks, Courant Computer Science Symposium 3, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[24] Roy, B., "Problems and Methods With Multiple Objective Functions", Mathematical Programming, Vol. 1. 1971. pp. 239-266.

[25] Shapiro, D., "Algorithms for the Solution of the Optimal Cost Travelling Salesman Problem," Sc. D. Thesis, Washington University, St. Louis, 1966.

[26] Srinivasan, V. and G.L. Thompson, "An Algorithm for Assigning Uses to Sources in a Special Class of Transportation Problems," Management Sciences Research Report #263, Carnegie-Mellon University, Pittsburgh, Pa.

7. COMPUTATIONAL RESULTS

The algorithm stated in this paper has been tested for computational experience. The code that is used to solve at the first step as a generalized transportation problem was developed and coded in Fortran IV by this author and Professor G. L. Thompson of Carnegie-Mellon University and is operable at the UNIVAC 1108 computer located at Carnegie-Mellon University and also at the CDC 6400 computer of the Northwestern University. The only real data that was used were from the computer network of two computers (Univac 1108. and IBM 360-65) of Carnegie-Mellon and the data of the computer network from Wright-Patterson Airforce Base, Ohio, consisting of four computers (CDC 6600, two IBM 7090 and one IBM 360 with computer graphics). However several randomly generated data for fictitious networks consisting of up to ten computers and a maximum of thousand jobs were run at Northwestern University's Vogelback Computer center for computational experience. The current code has a dimensional restriction of 1000 jobs and 10 computer, though this can be revised depending upon the computer's core. It is clear that the efficiency of the algorithm depends upon the efficiency of the primary code of the Generalized Transportation Problem. This author and Professor G. L. Thompson are revising this code currently by changing different starting rules and identifying the pivoting cell and using the four index method, similar to the "augmented predecision list structure and index method" of Glover Karney, Klingman and Napier [18]. The results of these findings will be reported soon [6].

The peak-period planning horizon for batch processing jobs for the Carnegie Network was 60 minutes and in the non-peak period it was 120 minutes. The jobs that were submitted ranged from 248 to 822 for the 60 minute period and 187 to 1012 for the two hour period. Two computers are in the network

and the CDC 6400 computational time, number of branches and jobs involved are given in Table 1. Note that we create an extra fictitious computer before solving. Similarly in the four computer network of Wright-Patterson Air Force Base the planning horizon were forty-five minutes for peak period and ninety minutes in the night time. Table 2 provides relevant results. In Table 3 results obtained for fictitious networks of 3, 5, ... 10 computers with randomly generated data with a planning horizon of sixty minutes are presented with an assumption of 500 jobs for each case. The mean and the standard deviation of computational time based on 30 runs for each case are also given. The number of Boolean variables required here varies from 744 to 5500 variables. Though Geoffrion's code [17] may be used, it is clear that certain problems of larger size that were considered cannot be solved at all. Further the smallest problem of 744 variables when solved as a zero one problem was aborted without reaching an optimal solution after sixty minutes of computer time were used up. (The planning horizon was sixty minutes for that network.) It is not clear whether this problem can be solved at all in any reasonable CPU time as a zero-one problem , whereas Tables I, II, III shows the computational efficiency of this specially structured zero-one problems, when solved utilizing the algorithm presented in this paper.