1-1-1981

Heuristic approaches to quadratic assignment problems.

Suresh Chandra Jaisingh
University of Windsor

HEURISTIC APPROACHES TO QUADRATIC ASSIGNMENT PROBLEM

by

Suresh Chandra Jaisingh

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of
Industrial Engineering in partial fulfillment
of the requirements for the Degree
of Doctor of Philosophy at
The University of Windsor.

Windsor, Ontario, Canada.

1981

ABSTRACT

This study is directed towards the development of heuristic algorithms for the solution to the quadratic assignment problem. This is the problem of minimizing the 'cost' of assigning n 'facilities' to n given 'sites', when there is an interflow. The combinatorial nature of the problem indicates that for a problem of size n, there are n! possible assignment vectors. Since it is computationally infeasible to generate all the possible assignment vectors for $n \geq 12$, heuristic procedures are developed which generate only a subset of them.

The algorithm developed in this study is an iterative optimization procedure which begins by constructing the matrix of lower bounds on the 'costs' of locating facilities at different sites. Any feasible assignment vector may or may not have optimum values associated with its components in this matrix. Thus, the algorithm seeks to remove the deviations from the optimum values in the elements of the lower bound matrix. After all the elements in the matrix have been adjusted, a linear assignment problem is solved, which results in a feasible assignment vector as well as an improved objective function value for the quadratic assignment problem. The procedure is repeated until the desired accuracy in the value of the objective function is obtained.

not more than 10, even for large size problems. Maximum improvement in the value of the objective function is observed to be in the first few iterations. It then decreases in subsequent iterations. Further, the proposed algorithm is independent of starting solution.

Finally, sensitivity analysis is carried out to study the affect of varying the parameters in the distance or flow matrix on the layout. No obvious pattern is observed. However, it is concluded that , to reduce the computational efforts, the information contained in the final iteration of the original problem could be used to study the effects of the changed parameter on the layout. Reapplying the algorithm is computationally inefficient.

ACKNOWLEDGEMENTS

The author wishes to express sincerest gratitude to his Dissertation advisor, Dr. R.S. Lashkari, for providing direction and encouragement to my Ph.D. program and to this research. His accessibility and his willingness to discuss any of the difficulties encountered is specially appreciated.

Special thanks are extended to Dr. K.G. Murty of the University of Michigan for serving as the external examiner to the dissertation committee.

Also the author wishes to express thanks to other members of the Dissertation Advisory Committee for stimulating and evaluating ideas. These members are: Dr. A. Raouf, Professor of Industrial Engineering, Professor A.A. Danish, Associate Professor of Industrial Engineering, Dr. M. Shridhar, Professor of Electrical Engineering and Dr. R. Barron, Associate Professor of Mathematics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Facilities location problems have been the subject of analysis for many years. However, it was not until the emergence of interest in operations research that the subject received renewed attention in a number of disciplines. During the recent years, economists, operations researchers, urban planners, management scientists, home economists, and engineers from several disciplines, have discovered a common interest in their concern for the location and layout of facilities. Each group has attempted to bring to the subject different interpretations of the problem and different approaches to its solution. The industrial engineers find it useful in laying out activities, offices, or departments in a building, etc.

There is a multitude of problems within facilities location. Of these we shall restrict our attention to those problems which are of a general nature. "Facilities location" shall now be designated as problems involving the assignment of n distinct facilities to n_1 distinct locations ($n \leq n_1$), when there is a cost function to be minimized. The facilities location problems which are of special interest in this study, are classified in the following section.

1.1 Classification of Facilities Location Problems

(i) Quadratic Assignment Problem (QAP)

The general QAP can be stated as follows. Given n^4 coefficients c_{ijpq} $\forall_{ijpq}$ the problem is to find an nxn permutation matrix $X = [x_{ij}]$ so as to minimize

$$\sum_{ij} \sum_{pq} c_{ijpq} x_{ij} x_{pq} \qquad (1.1)$$

This was first formulated in location context by Koopmans and Beckmann [1957]. It can also be stated as the determination of nxn permutation matrix $X = [x_{ij}]$ so as to minimize

$$f(X) = \sum_{ij} a_{ij} x_{ij} + \sum_{ij} \sum_{pq} f_{jq} d_{ip} x_{ij} x_{pq} \qquad (1.2)$$

where $A = [a_{ij}]$, $F = [f_{jq}]$ and $D = [d_{ip}]$ are nxn matrices, representing the following parameters:

a_{ij} = fixed cost associated with the location of facility j at location i

f_{jq} = the number of units of commodity to be transported from facility j to facility q

d_{ip} = the cost of transporting one unit of commodity from location i to location p

Each assignment of facilities to locations is given by a permutation

matrix $X = [x_{ij}]$ where

x_{ij} = 1 if facility j is assigned to location i

= 0 otherwise

This formulation is of great interest and provides the basic framework for a wide class of problems.

It is interesting to note that if $f_{jq} = 0$, for all (j,q), Equation (1.2) represents a linear assignment problem, which is employed in various algorithms for QAP, and which is formulated in the following section.

(ii) Linear Assignment Problem (LAP)

Consider n facilities to be located, one at each location and assume that there are exactly n locations available. Let x_{ij} be the variable defined in 1.1(i). Thus

$$x_{ij} = 0 \text{ or } 1 \text{ for } i = 1,..,n \text{ and } j = 1,..,n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for } i = 1,..,n$$

The last condition states that exactly one facility is located at each location i. Likewise, each facility j must be located at exactly one location, which leads to the condition:

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for } j = 1,..,n$$

If c_{ij} is the cost of locating facility j at location i, the linear assignment problem is defined as:

$$\text{Minimize } f(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} \tag{1.3}$$

Subject to

$$\left.\begin{aligned} &\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,..,n \\ &\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,..,n \\ &x_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j \end{aligned}\right\} \tag{1.4}$$

1.2 Applications

(i) Plant Layout Problem [Koopmans & Beckmann, 1957]

This is the problem of locating n plants uniquely at n locations in such a way that the total interplant transportation cost is minimized. In the context of the formulation given in (1.2), the variables can be defined as:

a_{ij} = fixed cost associated with the location of plant j at location i

f_{jq} = the number of units of commodity to be transported from plant q to plant j

d_{ip} = the cost of transporting one unit of commodity from location p to location i

Each assignment of plants to locations is given by a permutation matrix $X = [x_{ij}]$, where

x_{ij} = 1 if location i is assigned to plant j

= 0 otherwise

This formulation is often employed in (i) placement of electronic modules on a computer backplane so as to minimize wire length, number of crossings, etc. [Hanan and Kurtzberg, 1972] ; (ii) locating machines, departments, or offices within a plant so as to minimize transportation efforts [Armour and Buffa, 1963] ; (iii) arranging indicators and controls in a control room so as to minimize eye fatigue [McCormick, 1970]; (iv) laying out offices in building, or operating rooms in hospitals with the monetary objective of reducing the cost of office accomodation [Whitehead and Elders, 1964] , and relocating civil service departments with the social objective of providing employment in developing areas [Beale and Tomline, 1972] ; (v) locating hospital departments so as to minimize the total distance travelled by the patients [Elshafei, 1977]; and (vi) assigning n people to serve on m committees, (committee/coworker performance problem) [Maybee, 1978] .

As a generalization to this formulation, Lawler [1963] discussed the multicommodity version, in which there is a flow f_{jq}^{t} for each commodity t and a cost per unit flow between location j and q of

d^t_{jq}. As another generalization Graves and Whinston [1970] point out the possibility of a cost component w_{ijpq} that depends on a pair of assignments. Combining these two, Pierce and Crowston [1971]gave a more general cost function as

$$\text{Minimize } f(x) = \sum_{ij} a_{ij}\, x_{ij} + \sum_{ij}\sum_{pq} w_{ijpq}\, x_{ij}\, x_{pq} + \sum_{ij}\sum_{pq}\sum_{t} f^t_{jq}\, d^t_{ip}\, x_{ij}\, x_{pq}$$

where

$$c_{ijpq} = \begin{cases} w_{ijpq} + \sum f^t_{jq}\, d_{ip} & \text{if } i \neq p \text{ or } j \neq q \\ c_{ij} + f_{jj}\, d_{ii} & \text{if } i = p \text{ and } j = q \end{cases}$$

(ii) Travelling Salesman Problem

The'Travelling Salesman Problem' is a special case of the Koopmans and Beckmann formulation in which d_{ip} represents the distance between the pairs of cities and f_{jq} represents the cyclic permutation matrix of the form:

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

This formulation is used in solving the 'Candidates Problem'. This is the problem of finding optimum tours for the candidates in such a way that the total transportation costs are minimized [Lawler, 1963].

(iii) Rotor Balancing Problem [Murty, 1976]

This is the problem of mounting n blades on a horizontal rotor in such a way that the static balance is achieved relative to two orthogonal axes. It is assumed that n mounting positions are equidistant around the circumference of a circular rim of radius r. Let m_i designate the distance from the centre of gravity to the mounting end of blade i, and W_i designate the blade's weight. If the mounting positions are numbered counter-clockwise from one of the axes, the orthogonal components of the moment arm $W_i(r + m_i)$ produced by mounting blade i in position j can be expressed as:

$$h_{ij} = W_i (r + m_i) \cos (2\pi j/n)$$

and,

$$V_{ij} = W_i (r + m_i) \sin (2\pi j/n).$$

If it is desired to assign blades to positions in such a way that the sum of the moment components are both as close to zero as possible, the equivalent QAP as formulated by Maybee [1978] can be expressed as the determination of the permutation matrix $X = [x_{ij}]$ so as to

$$\text{minimize } f(x) = \sum_{ij} \sum_{pq} (h_{ij} h_{pq} + V_{ij} V_{pq}) x_{ij} x_{pq}$$

which is of the form of general QAP if

$$c_{ijpq} = h_{ij} h_{pq} + V_{ij} V_{pq}.$$

CHAPTER II

SURVEY OF EXISTING METHODS OF SOLUTION

Over the years a number of solution procedures have been developed. In asmuch as a systematic evaluation of these procedures is not available, it is the purpose of this chapter to provide a general classification scheme, and to present a brief survey of the important exact and heuristic algorithms that exist to date.

2.1 Exact Solution Procedures

Exact solutions can be sought in several ways. One way is to enumerate all the assignments, and choose the one with the minimum cost. Since the number of assignments is n!, it is practical to do so for small size problems. However, difficulties arise for moderate and large values of n. The first mathematical approach in the direction of developing an exact solution is given by Wimmert [1958]. He presented a method based on ranking the cost matrix and choosing coordinates close to the diagonal. However, Conway and Maxwell [1961] gave a counter example to invalidate Wimmert's model. The exact solution procedures developed so far can be grouped as follows.

(i) Solutions Based on Integer Programming

An equivalent integer linear program to the general QAP was formulated by Lawler [1963]. If n^2 variables x_{ij} are linearized by defining n^4 variables as

$$y_{ijpq} = x_{ij}\, x_{pq}$$

The equivalent linear program can be stated as

$$\text{Minimize} \quad \sum_{ijpq} c_{ijpq}\, y_{ijpq}$$

$$\text{Subject to} \quad \sum_{j} x_{ij} = 1 \quad (i = 1,2,..,n)$$

$$\sum_{i} x_{ij} = 1 \quad (j = 1,2,..,n)$$

$$\sum_{ijpq} y_{ijpq} = n^2$$

$$x_{ij} + x_{pq} - 2y_{ijpq} \geq 0 \quad (i,j,p,q, = 1,2,..,n)$$

$$x_{ij} = 0 \text{ or } 1 \quad (i,j = 1,2,..,n)$$

$$y_{ijpq} = 0 \text{ or } 1 \quad (i,j,p,q = 1,2,..,n)$$

The proof of the equivalence of the two problems can be referred to in Lawler [1963]. No computational experience is known to be available for this approach.

Love and Wong [1976] gave the binary mixed integer programming formulation for solving the QAP with rectilinear distances as follows:

$$\text{Minimize} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} W_{ij} \, (R_{ij} + L_{ij} + A_{ij} + B_{ij})$$

$$\text{Subject to} \quad \left.\begin{aligned} R_{ij} - L_{ij} &= x_i - x_j \\ A_{ij} - B_{ij} &= y_i - y_j \end{aligned}\right\} \qquad \begin{aligned} &i = 1,..,n-1 \\ &j = i+1 \end{aligned}$$

$$x_i + y_i = \sum_{k=1}^{n} s_k \, \alpha_{ik} \qquad i = 1,..,n$$

$$x_i - y_i = \sum_{k=1}^{n} p_k \, \alpha_{ik} \qquad i = 1,..,n$$

$$\sum_{k=1}^{n} \alpha_{ik} = 1 \qquad i = 1,..,n$$

$$\sum_{i=1}^{n} \alpha_{ik} = 1 \qquad k = 1,..,n$$

$$\alpha_{ik} = 0 \text{ or } 1 \qquad i,k = 1,..,n$$

$$R_{ij}, L_{ij}, A_{ij}, B_{ij}, x_1,..,x_n, y_1,..,y_n \geq 0$$

where:

n = number of facilities and number of locations

w_{ij} = non-negative flow between facility i and facility j

h_{ij} = horizontal distance between facility i and facility j if facility i is to the right of facility j; otherwise $h_{ij} = 0$

L_{ij} = horizontal distance between facility i and facility j if facility i is to the left of facility j; otherwise $L_{ij} = 0$

A_{ij} = vertical distance between facility i and facility j if facility i is above facility j; otherwise $A_{ij} = 0$

B_{ij} = vertical distance between facility i and facility j if facility i is below facility j; otherwise $B_{ij} = 0$

(x_i, y_i) = location of facility i, i = 1,..,n

s_k = sum of coordinates of location k, k = 1,..,n

p_k = difference of coordinates of location k, value of the first coordinate minus value of the second coordinate.

The computational results based on this approach reveal that only small size problems involving up to 8 or 9 facilities can be solved. It is concluded that the prospects of integer programming approach are not very promising until efficient integer programming codes are developed.

Bazarra and Sherali [1980] formulated the QAP as a mixed integer linear program by introducing a number of new variables and constraints. The problem is defined as the minimization of the function

$$\sum_{i=1}^{m-1} \sum_{j=1}^{m} \sum_{k=i+1}^{m} \sum_{l=1}^{m} c_{ijkl}\, y_{ijkl}$$

Subject to

$$\sum_{k=i+1}^{m} \sum_{l=1}^{m} y_{ijkl} - (m-i)\, x_{ij} = 0 \qquad \begin{matrix} i=1,..,m-1 \\ j=1,..,m \end{matrix}$$

$$\sum_{i=1}^{k-1} \sum_{\substack{j=1 \\ j\neq l}}^{m} y_{ijkl} - (k-1)\, x_{kl} = 0 \qquad \begin{matrix} k=2,..,m \\ l=1,..,m \end{matrix}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i=1,..,m$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j=1,..,m$$

x_{ij} binary $\quad i,j=1,..,m$

$y_{ijkl} \geq 1 \quad i=1,..,m-1;\ k=i+1,..,m$
$\qquad\qquad\qquad\quad j,l=1,..,m;\ j\neq l$

The above problem has m^2 integer and $m^2(m-1)^2/2$ continuous variables, and $2m^2$ linear constraints, as opposed to (m^2+m^4) integer variables and m^4+2m+1 constraints in Lawler's formulation. This problem can further be decomposed into a linear integer master problem in m^2 zero-one variables, and a linear subproblem and iterated between these two problems until a suitable termination criterion is met in a finite number of steps.

Their computational experience reveals that the procedure required close to m! cuts in order to verify optimality, even when the starting solution was optimal. It was therefore suggested to operate the procedure as a heuristic by terminating it prematurely. Its applicability as a heuristic procedure was demonstrated with the help of test problems given by Nugent, et al. [1968].

(ii) Solutions Based on Branch and Bound

The idea of branch and bound dates back to the algorithm of Little, et al. [1963] employed for solving the travelling salesman problem.

The "branch" term represents that the procedure is continually concerned with choosing the next feasible branch of the tree to elaborate and evaluate, while the "bound" term indicates their emphasis on the effective use of bounding the value of the objective function at each node in the tree, both for eliminating dominated paths and for selecting a next branch for evaluation and elaboration. These procedures possess the following three important attributes according to Pierce & Crowston [1971]:

1) Termination at any usable solution prior to the ultimate completion of the problem solving process.

2) Exploiting in an efficient manner the information that is available beforehand pertaining to the value of an optimal solution. For instance, when feasible solution is known from past experience or has been derived with the aid of a heuristic procedure, it is used to discard the solutions having higher costs. Thus, prior knowledge of upper or lower bounds reduces the region that need to be searched.

3) With slight modifications, these algorithms can be employed to find all the optimal or most preferred solutions.

The branch and bound methods developed for solving the QAP have been classified as follows:

a) Single Assignment Algorithm. This approach was first used by Gilmore [1962]and Lawler [1963]. Both presented essentially the same

algorithm for solving a Koopmans and Beckmann type QAP. Their approach employed a search strategy which elaborates the tree shown in Figure 2.1 from left to right. The ordering of the location is taken arbitrarily or with some heuristic ordering rule such as decreasing sums $\sum_{q \neq j} (d_{ij}+d_{jq})$. At each level the node is chosen, for elaboration to the next level, based on the least lower bound among the nodes not yet elaborated. These lower bounds are obtained by solving linear assignment problems. The process of selecting a node in a tree continues level by level, or until a node is reached at the level n or else lower bound exceeds the current upper bound, and the tree evaluation process backtracks to the lowest node on the path for which all branches have not been elaborated. The process then selects the next node and the procedure is repeated; when all the branches have been enumerated, the problem solving is complete.

Thus, it is seen that the bounding operation at each node is the key to the algorithm. Mathematically, this can be described by considering the objective function of the QAP:

$$f(x) = \sum_{ij} \sum_{pq} c_{ijpq}\, x_{ij}\, x_{pq}$$

$$= \sum_{ij} x_{ij} \sum_{pq} c_{ijpq}\, x_{pq}$$

Since X is a permutation matrix, we have

$$x_{ij}\, x_{iq} = 0 \text{ if } j \neq q$$

and $$x_{ij}\, x_{pj} = 0 \text{ if } i \neq p$$

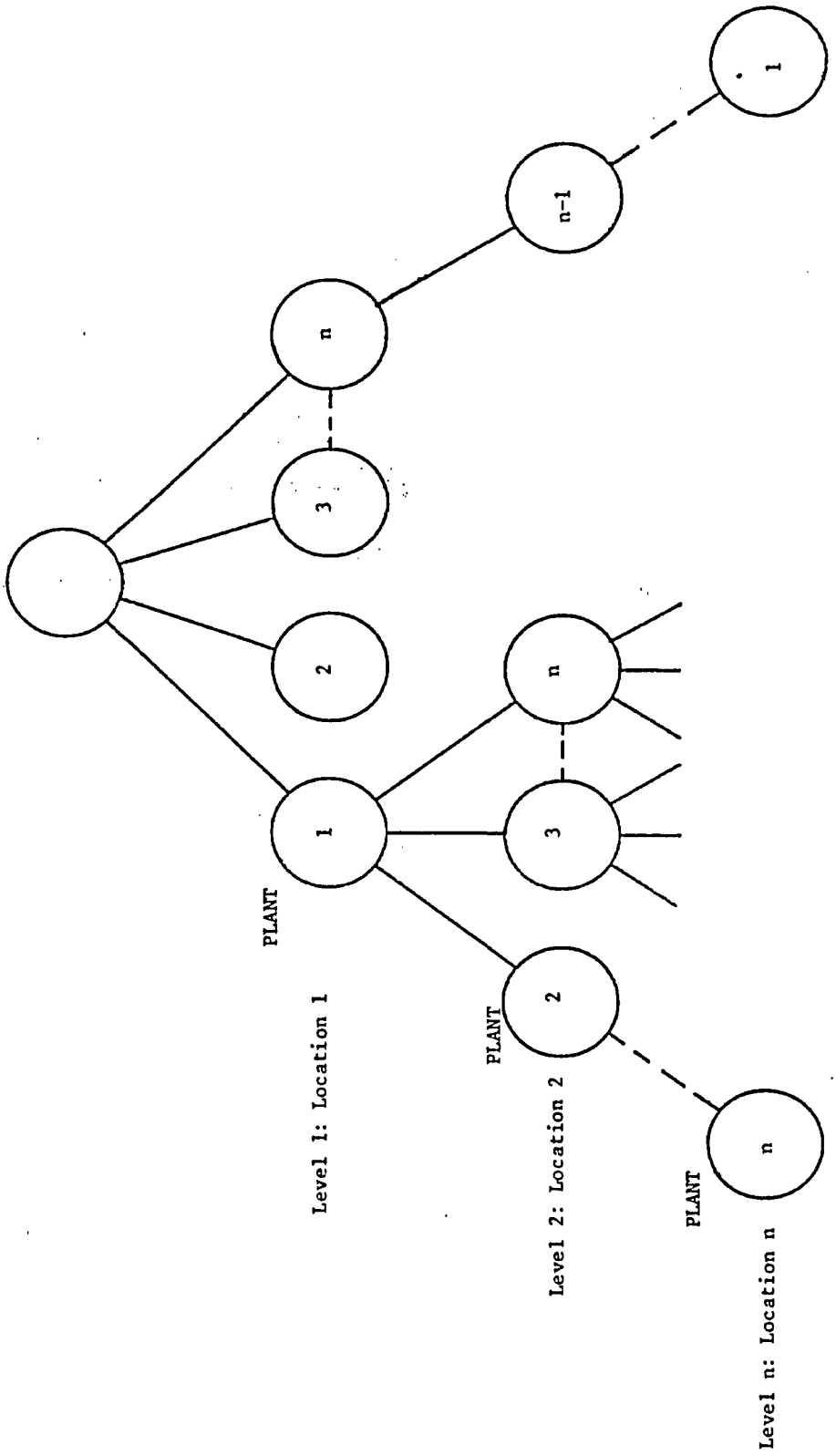


Figure 2.1 Search Tree With Each Level Representing A Unique Plant.

The above objective function may now be written as

$$f(x) = \sum_{ij} x_{ij} \left(c_{ijij} + \sum_{pq} c_{ijpq} x_{pq}\right) \quad p \neq i,\ q \neq j$$

Now, define an nxn matrix $[b_{ij}]$ in which each element represents the optimal value of LAP whose objective function is

$$\text{minimize } b_{ij} = c_{ijij} + \sum_{pq} c_{ijpq} x_{pq} \quad p \neq i,\ q \neq j$$

Thus for given values of i and j, b_{ij} represents a lower bound on the sum of n cost terms from the cost matrix. Now, if we solve the LAP:

$$\text{minimize } f(x) = \sum_{ij} b_{ij} x_{ij},$$

the solution to this problem would represent a lower bound on the sum of n^2 cost terms and therefore a lower bound for any feasible solution to the QAP.

If the lower bound is the same as the objective function value of the QAP, the optimal solution is the solution to the LAP. Otherwise the branch and bound procedure starts by finding lower bounds at each node. The method of finding the lower bound is essentially the same as discussed above. The only difference is that the elements of the lower bound cost matrix are composed of the linear cost contributions of the fixed variables of the node in addition to the lower bound costs associated with solving the LAP's. For further details of this

procedure the reader is referred to Cabot and Francis [1978].

Pierce and Crowston have also suggested several alternative ways of branching and bounding. For example, in Fig. 2.1 the plants and locations may be interchanged to obtain the tree as shown in Fig. 2.2. This arrangement would give rise to the elaboration of different partial trees of solutions with quite differing number of nodes. Thus the time required to elaborate and evaluate a single node in a tree can differ markedly. They further suggest that the lower bound for the Koopmans and Beckmann formulation can be obtained by simply sequencing the relevant flow and distance values and forming the inner product. For finding bounds at intermediate nodes several methods are similarly suggested. Each of them would lead to the elaboration and evaluation of different branches of the tree. It is therefore stated that the relative efficiency may be highly dependent on the particular form of the QAP being solved.

Burkard [1973] presented a branch and bound algorithm for the general QAP. The lower bound cost matrix is first formed. As in Lawler's approach, an LAP is then solved on this matrix to establish an initial lower bound. The lower bounds at intermediate nodes are obtained by augmenting the lower bound matrix to include the linear cost contributions of the fixed variables of a node. Only one LAP is solved to develop a lower bound for an intermediate node. This results in larger search trees because of weaker lower bounds developed

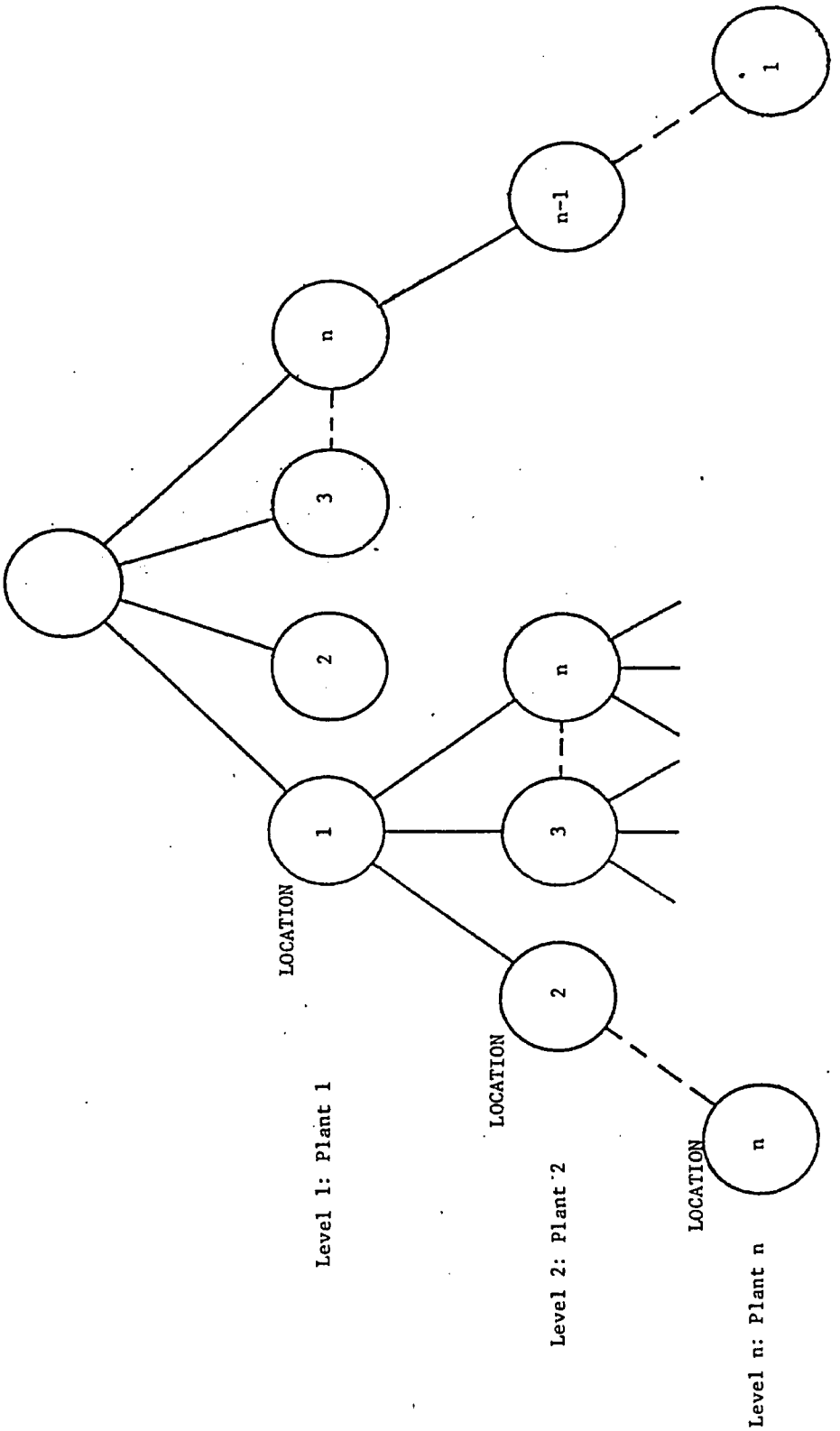


Figure 2.2 Search Tree With Each Level Representing A Unique Location

at intermediate nodes. The superiority of this procedure over Lawler's may be due to significant differences in computational requirements for finding the lower bounds at each node.

Maybee [1978] developed an efficient branch and bound technique based on the iterative process of matrix reduction. It is derived from the idea of adding a skew symmetric matrix to a quadratic cost matrix c_{ijpq}. This was first suggested by Murty [1970] in conjunction with assignment ranking algorithms. The skew symmetric matrix selected for addition was such as to produce an upper triangular matrix. All the entries in the lower triangular portion were zero. The advantage of triangularization was to capture in one element, the two quadratic costs associated with pairs of allocations. This reduces the subsequent cost manipulation by a factor of 2. It is estimated that the computational superiority of this procedure over Lawler's algorithm is two orders of magnitude for 12 x 12 problems. But still its applicability is limited to $n \leq 15$.

Bazaraa and Elshafei [1979] developed an exact branch and bound scheme similar to the algorithm of Gilmore [1962]. They incorporated the concept of "stepped fathoming" given by Bazaraa and Elshafei [1977]. It is shown that the algorithm speeds up the search of the decision tree. However, it failed to solve problems of size $n \geq 15$.

b) Pair-Assignment Algorithms. Gavett and Plyter [1966] and Land [1963] described such a branch and bound technique. Like a single-

assignment algorithm, this is viewed as a LAP where pairs of plants j and q are located at locations i and p. Algorithms are developed for symmetric Koopmans and Beckmann problem with $c_{ijpq} = c_{iqpj} = f_{iq} d_{ip}$.

The algorithm in general considers a cost matrix of size n(n-1)/2 where every distinct pair of locations and every distinct pair of facilities have an appropriate cost entry, which is the product of the distance between the location pair and total flow between the facility pairs in both directions. Operationally both the algorithm of Land and that of Gavett and Plyter commence by determining an optimal linear assignment solution to the initial cost matrix, A_0. Thereafter Gavett and Plyter employ a specified method of successive reduction method, whereas Land employs a column-reduced matrix at each node. Both algorithms proceed level by level in the tree, adding one new pair to the solution at each level, and backtracking to the lowest level in the tree having an unevaluated branch. In selecting the pair to be added at a given level in the tree, Gavett and Plyter use the alternate cost method of Little et al. [1963], while Land always selects from the column having the fewest number of feasible elements in the column-reduced matrix A_v.

As an extension to this procedure, Pierce and Crowston [1971] formulated a linear problem for non-symmetric QAP as:

$$\text{minimize} \quad \sum_{(ip),(jq)} (c_{ijpq}\, t_{ijpq} + c_{iqpj}\, t_{iqpj})$$

$$\text{subject to} \quad \sum_{jq}^{n(n-1)/2} (t_{ijpq} + t_{iqjp}) = 1 \qquad \text{all } (i,p)$$

$$\sum_{(ip)}^{n(n-1)/2} t_{ijpq} + \sum_{(ik)}^{n(n-1)/2} t_{iqjp} = 1 \quad \text{all } (j,q) \qquad (2.1)$$

$$t_{ijpq},\ t_{iqjp} = 0,1 \quad \text{for all } i,j,p,q.$$

Further feasibility constraints which must be satisfied, are

$$\text{if} \quad t_{ijpq} = x_{ij}\, x_{pq} = 1$$

$$\text{then} \quad t_{ickl} = 0,\ t_{vikl} = 0,\ t_{ujkl} = 0,\ t_{ivkl} = 0$$

$$t_{uvpl} = 0,\ t_{uvlp} = 0,\ t_{uvkq} = 0,\ t_{uvkq} = 0 \qquad (2.2)$$

$$\text{where} \quad i \neq u \neq j \quad i \neq v \neq j \quad p \neq k \neq q \ \&\ p \neq l \neq q$$

It is reported that the Gavett and Plyter algorithms require a great deal of computational time; an eight-plant location problem, which would be computationally equivalent to $\frac{8(7)}{2} = 28$ city travelling salesman problem, takes 42 minutes on IBM 7074.

c) Pair-Exclusion Algorithms. Pierce and Crowston [1971] describe such an algorithm, which is similar to the solution procedure discussed

in the preceding Section. The algorithm starts with an optimal solution for the linear assignment portion of the problem. If the feasibility constraints given by Equations (2.2) are satisfied for all t_{ijpq}, the solution obtained is an optimal solution to the QAP. Otherwise, there are one or more conflicting assignments in the solution rendering infeasibility to the QAP. The procedure therefore subdivides the total set of feasible quadratic assignments into those that do not include the partial assignments indicated by the optimal solution to linear assignment problem. For example, if the optimal assignment is (AB,14), (AC,24), (AD,13), (BC,34), (BD,23), (CD,13) this does not satisfy the feasibility condition. It means, at least one of the assignments will not be present in the solution. This results in a tree of nodes as shown in Fig. 2.3. Each node is then evaluated and the one which has the least lower bound is chosen for further elaboration. The resulting assignment at this node is checked for its feasibility. If the solution is not feasible, the result is another tree with a new level of nodes. This is continued until a node is reached for which the optimal linear assignment is a feasible quadratic assignment. The process is complete when no node is available whose lower bound is less than the value of the quadratic assignment solution.

From the survey of above exact solution procedures, it can be concluded that the QAP is an extremely difficult combinatorial problem. Undoubtedly, the branch and bound integer programming approach preclude

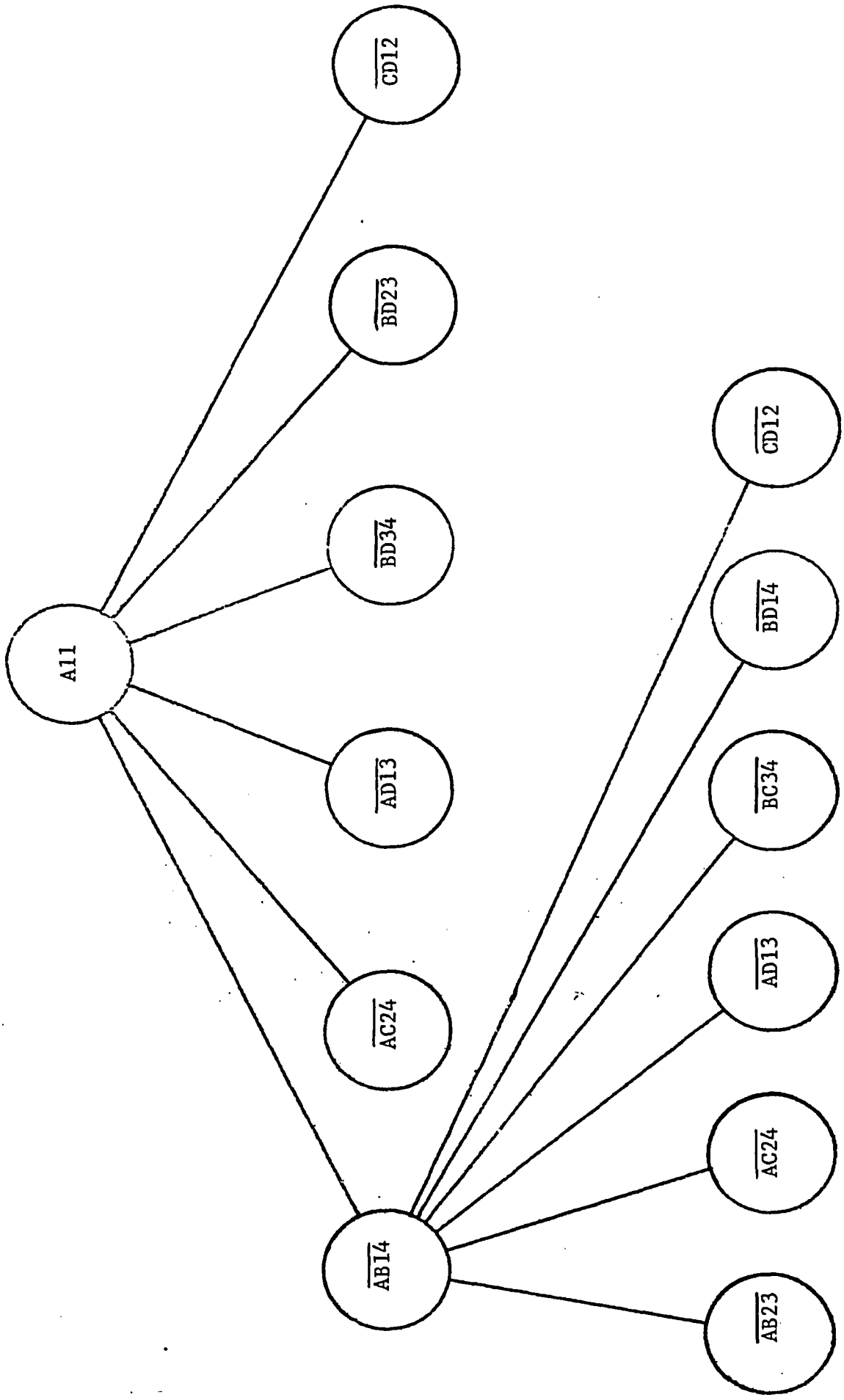


Figure 2.3 Tree Diagram for Pair Exclusion Algorithm

several classes of assignments from consideration. But still these procedures become computationally infeasible if n, the number of facilities to be located, is greater than 12, in which case it might be no better than an exhaustive search. The question of what is theoretically possible is yet to be proven.

Because of the obvious difficulties experienced in the development of exact solution procedures, several researchers have considered this problem from the point of view of developing heuristic procedures. These are described in the next section.

2.2 Heuristic Solution Procedures

A heuristic technique, as defined by Nicholson [1971] may be stated as a method for solving problems by an intuitive approach, in which the structure of the problem can be interpreted and exploited intelligently to obtain a reasonable solution. Heuristic methods lend themselves ideally to fast computing methods, which usually involve the scanning of many alternative solution attempts, and selecting the better or best of these solutions according to specified criteria [Hitchings and Cottam,1976]. The heuristic procedures which have been developed in the past, to solve the QAP may be classified under the following groups.

(i) Construction Methods.

These methods start with a null solution and proceed to build a complete permutation. For a problem involving n facilities, the

method terminates in (n-1) steps by making successive assignment of facilities to locations and adding them to the null solution.

There are several computerized layout programs based on construction methods, such as PLANET by Apple and Deisenroth [1972], "RMA comp I" by Muther and McPherson [1970], CORELAP by Lee and Moore [1967], ALDEP by Seehof and Evans [1967], LSP by Zoller and Adendorf [1972] and LAYOPT by Matto [1969]. However, as Francis and White [1974] pointed out, ALDEP and CORELAP are the most representative of the construction methods. CORELAP chooses facilities in terms of their relatedness to those which have already been assigned. First, it assigns the facility with the most interaction to a central location. It then chooses the 'most related' facility, that is, the facility which has the most interaction with the assigned facility, and assigns it as nearby as possible. In a similar way, it successively chooses the most related facility, among those unassigned, and assigns it nearby the assigned facilities. ALDEP is similar to CORELAP, since it also chooses the facilities in terms of their relatedness to those already assigned. However, its first choice is made randomly, as well as subsequent choices when there is little relatedness.

Gilmore [1962] presented two construction algorithims, one requiring on the order of n^4 elementary operations, and the other on the order of n^5 operations. These are based on the stage decision process, where each stage facility assignment not chosen earlier is added.

The process is repeated until all the facilities are located. The first algorithm chooses the facility assignment according to some criteria based on the cost in the lower bound cost matrix. The second algorithm, in addition, solves one linear assignment problem at each stage. The facility assignments are chosen according to some criteria on the basis of cost elements appearing in the assignment problem solution.

Graves and Whinston [1970] presented an algorithm based on the mean value consideration. At each stage, it chooses that facility assignment which minimizes some mean value function of all remaining assignments. Thus, if k facilities have been located, at (k+1)th stage, it calculates the expected final influence of the remaining n-k facilities. This procedure compares favourably to Gilmore's n^4 and n^5 algorithms.

Edward,et al.[1970] proposed the Modular Allocation Technique (MAT). This is based on the theorem that the sum of pairwise products of two sequences of real numbers is minimized if one sequence is arranged in increasing order and the other in decreasing order. Given the matrix of flows among the facilities and the matrix of distances between locations, MAT arranges all pairwise flows and distances in descending and ascending orders, respectively. It then selects the pair of facilities which has the highest interaction and locates them to the pair of locations which have the lowest distance between them. Next it selects the pair of facilities which has one of the assigned facilities. The pair of

locations are also selected in such a way that one of the locations is already occupied. These selected pairs give rise to the location of a new facility. Suppose facilities i and j are selected initially and facilities m and j are selected next. And suppose locations p and q are selected initially and locations q and r selected next. The assignment is then made as follows:

Facilities	Locations
i	p
j	q
m	r

This procedure is repeated until all the facilities are assigned.

Weingarten [1972] developed a construction procedure termed as p algorithm. It is based on ranking the facilities and locations. Each facility is ranked according to the total number of interactions between itself and other facilities. Each location is ranked according to sum of the distances from itself to the rest of the locations. The complete solution is then obtained by assigning the facilities, ranked in a descending order, to locations, ranked in an ascending order. Weingarten further discusses the optimality and non optimality of the p algorithm.

Neghabat [1974] presented an algorithm in which two facilities having the greatest amount of interaction are first arranged arbitrarily at their minimum allowable distance. Next the facility having the largest

interaction with the first two facilities is located as close as possible. Once the relative distances are established, the overall cost of each partial arrangement can be computed. The arrangement that corresponds to the minimum cost is selected for the next iteration. In general, at each stage of the process, the oncoming facility having the largest overall interaction with previously selected facilities is located relative to the already established configuration (i.e., previous ordering remains the same) such that the objective function up to the present stage is minimized. The process terminates when all n facilities have been considered individually.

Parker [1976] in his comparative study discusses RAND and BEST MATCH in addition to above construction procedures. RAND generates random assignments. BEST MATCH is similar to the P-algorithm developed by Weingarten [1972].

(ii) Improvement Methods.

These methods start with an arbitrary assignment and iterate from one assignment to the next, changing pairs or triples of facilities until no more improvement is possible. Several heuristics have been developed using this concept.

Hillier [1963] developed an improvement method, based on a Move Desirability Table (MDT). In the literature it is referred to as H63.

The MDT calculations are based upon the cost benefits accrued by unilaterally moving a facility to an adjacent location (left, right,

up or down). In this procedure an initial assignment is chosen randomly, then the facility with the highest MDT value is investigated for adjacent interchanges. If it improves the cost, the exchange is executed. Otherwise, the facility having the next highest MDT value is examined for the exchange. The process is continued till no further improvement is possible.

Hillier and Connors [1966] later revised the H63 procedure by permitting exchanges among non-adjacent facilities, and thus allowing a large number of facilities to be investigated. In the literature, the revised procedure is termed as Hc63-66.

Armour and Buffa [1963], and Armour and Buffa [1964] developed the computerized relative allocations of facilities technique (CRAFT). This technique starts with an arbitrary assignment and interchanges all $n(n-1)/2$ pairs (or else all $\frac{n(n-1)(n-2)}{3!}$ triples), choosing the assignment which has the lowest cost. In this manner, it goes from assignment to assignment, until no improvement is possible. Thus, the process explores all the solutions in the neighbourhood of a given solution, and chooses the best one as the next starting point.

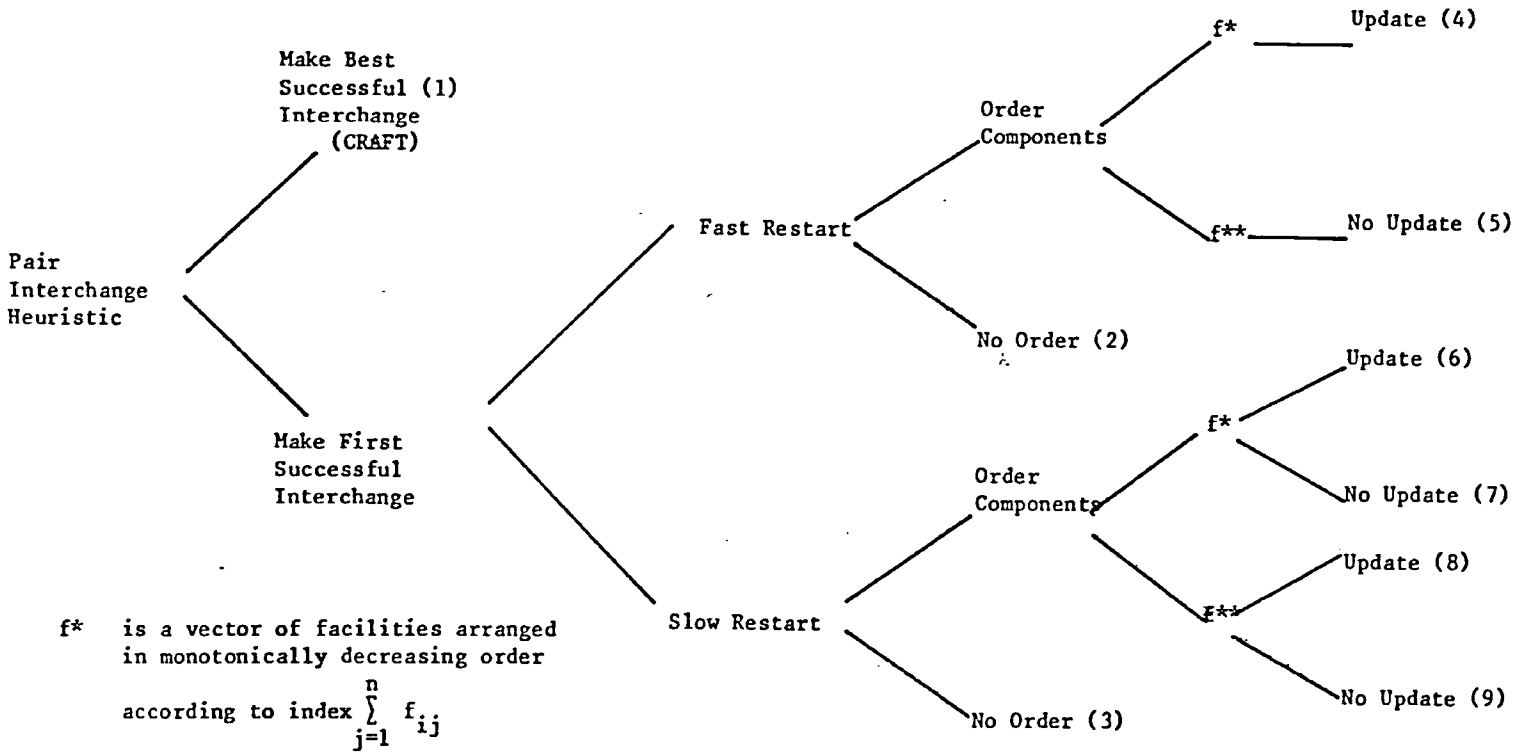
In order to reduce the computational effort required by CRAFT, Vollmann et al. [1968] have devised an alternative procedure, which is sometimes referred to as COL or VNZ procedure. This procedure consists of two phases. Phase 1 identifies two facilities, say M_1 and M_2 that have the highest and second highest total costs. Then a prescribed

interchange procedure is followed to improve the assignment. The choice of facilities is motivated by the fact that interchanging these two facilities with others will lead to a greater reduction in total cost than that obtained by most other choices of two facilities. In phase 2 all pairwise interchanges of facilities are checked twice, and interchanges are made when the total cost is reduced.

Khalil 1973 proposed the method of Facilities Relative Allocation Technique (FRAT) which combines the basic ideas of several heuristic techniques, mainly, those of Hillier and Connors [1966], Vollmann, et al. [1968], Armour and Buffa [1963], Buffa and Armour [1964].

Hitchings and Cottam [1976] presented the Terminal Sampling Procedure. Like Khalil [1973], they also embrace many of the desirable elements from previous heuristics which would tend to contribute towards the efficiency and quality of the solution. In this procedure, the pairwise exchange is carried out on a selective basis, similar to COL. In case of ties, the iteration is continued for all the tied assignments.

Parker [1976] proposed nine improvement algorithms based on pairwise interchange methods for his comparative study. Except for CRAFT, all methods considered select as the new basis, on any given cycle through the facility interchanges, the first interchange resulting in a decrement of the objective function value. These procedures differ from each other with respect to three parameters: restart, ordering of



f* is a vector of facilities arranged in monotonically decreasing order according to index $\sum_{j=1}^{n} f_{ij}$

f** is a vector of facilities arranged in monotonically decreasing order according to current contribution to the objective function $\sum_{}^{n} d_{iv}\, d(a(i),a(j))$ for V i

Figure 2.4 Improvement Algorithms Based On Interchange Procedure

component pairs, and update of component ordering. The nine procedures thus developed having the above characteristics, can be described as in Fig. 2.4.

Elshafei [1977] developed a heuristic technique, which is a combination of construction and improvement procedures. After developing an initial solution, the pairwise interchange procedure is used to improve the solution. To obtain an initial solution, two methods are proposed. One is similar to Weingarten's [1972] except that the ranking of a facility is based on the number of facilities having interaction with this facility. The second method constructs the initial solution, at any stage k, by choosing the facility which has the maximum interaction with the most recently located facility, and locating it at a location which causes minimum increase in the total cost. This procedure is continued until all facilities are located. In the development of his algorithm, Elshafei combines both methods to develop the initial solution.

(iii) Algebraic Methods.

These methods generally seek the solution in some simultaneous fashion. In most cases, a relaxation is first employed which gives lower bound on the optimal solution. Then subsequent operations seek to perturb this solution in the least damaging manner to the objective function value so that the initial restrictions to the problem are met.

One such approach is due to Gaschutz and Ahrens [1968]. The

method, which is based on the procedure proposed by Kodres [1959], relaxes the indivisibility constraint so that facilities are assigned on a continuous grid. A linear assignment algorithm is then used to locate the facilities at appropriate discrete positions.

The procedure assumes that the distances are specified monotonic functions, and that the set of possible locations is a set of points which can be embedded into a rectangular array of locations in the plane. Initially the facilities are placed randomly or according to a prescribed procedure. The coordinates are then transformed linearly such that the sums $\sum_i x_i$ and $\sum_i y_i$ become zero and the sums $\sum_i x_i^2$ and $\sum_i y_i^2$ become equal to the corresponding sums for the coordinates of given locations to which facilities would be finally assigned. This ensures that the initial random collection of points in the plane covers approximately the same area that is occupied by the given rectangular array of locations, centered at (0,0). The following transformation is then applied to all initial placements simultaneously.

$$Z_i^* = (1 - t \sum_p f_{ip}) Z_i + t \sum_p f_{ip} Z_p$$

where Z_i^* and Z_i are new and old coordinates, and t is a constant factor, chosen freely.

The above transformation is repeated a given number of times with linearly decreasing factors t until f reaches zero.

This procedure leaves the placement of facilities into the positions that correspond only approximately to given locations. It

is then discretized to definite locations by the Hungarian method developed by Kuhn [1957].

(iv) Stochastic Methods.

These methods employ some random methods as an adjunct to other major solution procedures. This usually occurs when a random solution is generated as a starting point for some improvement algorithms, or when ties are broken randomly.

CRAFT, proposed by Armour and Buffa [1962] begins with a random assignment, but derives each successive assignment deterministically. Nugent, et al. [1968] developed a biased sampling technique, which modifies CRAFT by choosing interchanges on a probabilistic basis. If S_i is the amount by which the cost of an assignment has been reduced, by the ith interchange which has a cost reduction, the probability of choosing S_j is then given by

$$P_j = \frac{S_j^c}{S_i^c + \ldots + S_k^c}$$

where P_j = probability of selecting jth pairwise exchange

c = a parameter to vary the effect of cost reduction

k = number of pairwise interchanges with cost reduction.

The procedure is to associate high probabilities to interchanges in relation to their cost reduction. This, in effect, explores the neighbourhood of the CRAFT solution, the size of the neighbourhood

being determined by c.

2.3 Computational Experiences and the Comparative Studies

In the preceding section, the basic concepts of different algorithms, developed in the past, were summarized. No mention was made of their applications and relative superiority. The purpose of this section is to summarize the available experiences on different algorithms and their comparative performance.

The exact solution procedures are reported to be feasible for solving only small size problems ($n < 15$). Obviously, such procedures consume a great amount of time; it may therefore not be surprising that comparative studies are not available in the literature. However, Maybee [1978] reported his algorithm to be efficient as compared to Lawler's [1963] algorithm; for a problem of size 12 x 12, he showed his algorithm to be superior by two orders of magnitude. Pierce and Crowston [1971] have given the experiences on branch and bound procedures. They state that it is difficult to assess the relative efficiency of the different algorithms, because it is highly dependent on the particular form of the QAP being solved.

Heuristic solution procedures have been developed from the point of view of solving moderate and large size problems in the light of (i) solution efficiency; (ii) solution quality; (iii) solution diversity. Solution efficiency means the time it takes to apply a specific approach to obtain a solution to a specific problem.

Solution quality is defined as the proximity to the optimal solution. Solution diversity indicates the capability of generating different solutions. If the algorithm always produces the same solution (which need not be optimal) to a problem, it is much less valuable than a second one producing many solutions.

The first comparative study of heuristic procedures is given by Nugent, et al. [1968]. They compared H63, HC63-66, and biased sampling in the light of the first two solution standards. Eight test problems were used for this study. It was concluded that H63 is inferior to HC63-66 on both counts. They also reported that CRAFT produces somewhat higher quality solutions than HC63-66, but the computation time was found to increase by a factor of n^3, whereas HC63-66's computational time increases by a factor of n^2. Biased sampling produces better quality solutions over CRAFT but its computational time increases by a factor of n^4.

Edward, et al. [1970] summarized the results of CRAFTand MAT. They discussed the superiority of MAT over CRAFT from the point of view of computational time but at the expense of the solution quality. It was further suggested that MAT could be used to construct the starting solution for the improvement methods.

Ritzman [1972] made a comparative study on CRAFT, HC63-66, ALDEP, and Wimmert's procedure. It was concluded that the best performer, in terms of solution quality, solution efficiency and solution diversity,

was CRAFT. The HC63-66 method was found to be competitive with CRAFT, the differences being small.

Neghabat [1974] compared his construction procedure with other existing heuristics and concluded that his procedure was capable of solving large size problems that were interactable from the computational view point. However, no emphasis was given to the solution quality. Thus, the solution may not be better than a randomly, chosen assignment.

Francis and White [1974] reported that no construction procedures developed to date such as ALDEP, CORELAP, PLANET, etc., have been shown to be clearly superior to the best improvement procedures, given by Nugent, et al. [1968].

Khalil [1973] compared FRAT with HC63, HC63-66 CRAFT, COL and Biased Sampling, and concluded that Biased Sampling method provides favourable results from the point of vie of solution quality. But with respect to the solution efficiency, COL was found to produce the solution, using the least amount of computational time. If Biased Sampling is discarded (because of excessive computational time) FRAT becomes next on the list for higher quality solutions (FRAT was reported to take slightly more computational time than COL). This indicates the difficulties involved in comparing the procedures on time basis of a single criterion.

Parker [1976] tested four construction and nine improvement procedures. The construction procedures tested were RAND, BEST MATCH, MAT, GW (Graves and Whinston). It was concluded that, among the

construction procedures, the GW procedure produced high quality solutions, but the computational times for these heuristics varied inversely with solution quality. Among the improvement procedures, CRAFT produced higher quality solutions, on the average, than other improvement procedures. However, this superiority was attained at the expense of additional computation time, as all greedy interchange methods tested yielded solutions more quickly than CRAFT. It was further concluded that the GW procedure, when used for constructing the starting solution for improvement procedures, produced higher quality solutions as compared to the different starting solutions obtained by other construction procedures such as RAND, BEST MATCH and MAT. It was thus concluded that the GW procedure may be used to generate the starting solution for improvement procedures. GW method though restricts the algorithm to explore the vicinity of only one particular assignment. It doesnot have the capability to generate more solutions unless it is modified.

Elshafei [1977] compared his method with HC63, HC63-66, CRAFT, Biased Sampling and Neghabat's, and showed its superiority over other algorithms. He demonstrated the capability of the algorithm to construct different starting solutions and generate improved solutions. He further applied the algorithm to a practical problem and showed the superiority of the algorithm by achieving 19.2% improvement in the result as compared to Khorshid and Hassan [1974].

Hitchings and Cottam [1976] gave some computational experiences on their sampling method over H63, HC63-66, MAT, FRAT, CRAFT, COL and Biased Sampling. They showed that terminal sampling procedure produces equivalent or higher quality solutions in less computational time as compared to Biased

Sampling (which is known to produce the higher quality solution). They emphasized on embracing several elements from different heuristics so as to generate better starting solutions for further improvement.

Bazaraa and Sherali [1980] demonstrated the use of their exact solution procedure as a heuristic. It produces high quality solutions; however, it consumes a great deal of computational time as compared to the other heuristics.

For ease of reference the results of various heuristics which produce reasonably good solutions, such as HC63-66, MAT, CRAFT, Biased Sampling, Terminal Sampling, COL, FRAT, Elshafei and Bazaraa and Sherali, are summarized in the Tables 2.1 and 2.2. The quality of solution here is not a well defined term. These are the solutions which are produced by several heuristics which do not guarantee the closeness of the solution to the optimal solution. However, the heuristics which produce closer solutions to optimal solutions are referred to as high quality solution producing methods. Since the aim of this study is to develop high quality solutions in an efficient manner the comparative timings given here are only for those methods which produce high quality solutions. Other methods are either found to be inferior or so not emphasize the quality of the solution.

2.4 Motivation

The previous discussion indicates that each procedure has its own advantages as well as limitations, thus making it difficult to select the "best" procedure. However, using three main criteria, namely (1) solution quality (2) solution efficiency and (3) solution diversity, it appears appears that the best solution procedures are those of Bazaraa and

Table 2.1 Summary of Comparative Results of Various Heuristic Procedures

Problem Size	Average Final Objective Function Value *			Best Final Objective Function Value *						Best Objective Function Value
	CRAFT 1963	HC-63-66 1966	Biased Sampling 1968	COL 1968	MAT 1970	FRAT 1973	Terminal Sampling 1976	Elshafei 1977	Bazaraa &Sherali 1980	
5	28.2	29.4	26.8	25	25	25	25	26	25	25
6	44.2	44.2	43.6	43	55	43	43	43	43	43
7	79.6	78.4	74.8	74	80	74	74	75	74	74
8	113.4	110.2	107.0	107	128	107	107	114	107	107
12	296.2	310.2	293.0	293	337	295	289	295	289	289
15	606.0	600.2	580.2	583	741	575	575	614	575	575
20	1339.0	1345.0	1313.0	1312	1450	1300	1296	1299	1285	1285
30	3189.0	3206.8	3124.8	3152	3711	3129	3086	3125	3095	3077

* Objective function values are in cost units.

Table 2.2 Summary of Computational Times per Solution for Various Heuristic Procedures

Problem Size	CRAFT 1963		HC-63-66 1966		Biased Sampling 1968		COL† 1968	MAT 1970		FRAT 1973		Terminal[$] Sampling 1976	Bazaraa[$$] & Sherali 1980
	sec*	sec**	sec*	sec**	sec*	sec**		sec**	sec**	sec[I]	sec**	sec	sec
5	1	.042	10	.42	1.1	.046	3	1.02	.0425	.11	.050	.25	0.69
6	2	.080	9	.38	2.1	.088	4	1.26	.0520	.15	.075	.30	1.30
7	5	.210	12	.50	5.7	.238	5	1.62	.0680	.25	.125	.75	2.25
8	10	.420	14	.58	10.9	.454	6	2.10	.0880	.33	.165	1.23	3.56
12	70	2.900	19	.79	65.8	2.742	8	6.12	.2550	1.25	.625	6.84	13.69
15	160	6.700	40	1.67	219.2	9.133	19	13.20	.5500	2.96	1.480	20.13	33.93
20	528	22.00	75	3.13	691.5	28.813	32	47.49	1.9790	10.87	5.430	59.87	100.61
30	3152	131.300	285	11.88	4272.4	178.000	118	346.71	14.4460	58.30	29.150	383.27	393.96

* GE 265 computer timings
GE 635 computer timings
I IBM 360 computer timings
$ ICL 470 computer timings
** equivalent IBM 370/3031 timings
$$ CDC cyber 70 Model 74-28/CDC 6400 computer timings

Sherali [1980], Hitchings and Cottam [1976], and Elshafei [1977].

Bazaraa and Sherali's procedure is based on the exact solution procedure, while Hitchings and Cottam's, and Elshafei's procedures are based on the improvement method. Comparative studies of these procedures are not available in the literature. However, considering the three criteria mentioned above, all three procedures have the capability to generate improved starting solutions for further improvement. The solution efficiencies can not be compared, as exact time comparisons are not easy to make due to differences in computing systems and programming techniques. From the point of view of quality of solution, Bazaraa and Sherali demonstrated the ability of their procedure in producing high quality solutions but at the expense of computational time. On the other hand, Elshafei's and Hitchings and Cottam's procedures produce solutions with reasonable quality in a comparatively efficient manner. Thus, although improvement methods are the best solution procedures developed so far, they are dependent on the starting solution. The neighbourhood surrounding this assignment is questionable and may not contain any improved solutions. Moreover, this neighbourhood may be just a collection of random assignments with no structure. In such cases it can not be said that these procedures are more effective than an arbitrary choice of assignments [Weingarten, 1972]. Also, from the viewpoint of efficiency, various starting solutions need

to be investigated before reaching a "good" solution. As problem size increases, the solution space increases drastically, with the result that an unproportionally large number of starting solutions would need to be investigated.

Despite these difficulties in developing both heuristic and exact solution procedures, there is continuing interest in both directions, that is, the development of exact as well as heuristic methods. In many instances, such as layout of plants, offices or departments in a building, the decisions are of a permanent nature. Therefore, it is desirable to have exact or high quality solutions available; otherwise, under the adopted configuration, the system would be operating inefficiently, and as a result, there would be cumulative losses overtime. The interest in high quality solutions, even for moderate size problems ($12 \leq n \leq 30$), is evident from the fact that the test problems reported by Nugent, et al. [1968] have become a challenge to the researchers.

On the other hand, heuristic procedures have been developed so as to provide practical methods of analyzing and solving many problems encountered in manufacturing industry, such as line balancing, site location, scheduling, routing, design problems, etc. However, as Hitchings and Cottam [1976] point out, the heuristic methods developed in the past are being criticized because of the excessive computational times entailed in their applications. They further

suggest that these procedures be re-examined and suitably modified in the light of previous findings so as to improve their efficiency and solution quality.

Having recognized the necessity and importance of the solution to the QAP, it is the purpose of this study to develop efficient heuristic procedures so as to obtain high quality solutions for the QAP.

CHAPTER III

DEFINITION OF PROBLEM AND OUTLINE OF SOLUTION METHODOLOGY

3.1 Introduction

In this Chapter, the quadratic assignment problem is defined, and the outlines of the solution methodology proposed are described. The problem studied here is the Koopmans-Beckmann problem, which is formulated in section 3.2. In the next section, the iterative solution procedure is identified and the solution methodology is proposed, followed by a summary of the steps necessary for the development of the algorithm.

3.2 Formulation of the Problem

Consider the objective function of the QAP written as

$$f(X) = \sum_{ij} \sum_{pq} c_{ijpq}\, x_{ij}\, x_{pq}$$

$$= \sum_{ij} x_{ij} \sum_{pq} c_{ijpq}\, x_{pq} \qquad (3.1)$$

where c_{ijpq} is defined as the "cost" per unit time of having facility j located at site i and facility q located at site p, and where the permutation matrix $X = [x_{ij}]$ satisfies the following constraints:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i=1,..,n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for } j=1,..,n$$

$$x_{ij} = 0 \text{ or } 1$$

We therefore have

$$x_{ij} \cdot x_{iq} = 0 \qquad \text{if } j \neq q$$

$$x_{ij} \cdot x_{pj} = 0 \qquad \text{if } i \neq p$$

Equation (3.1) may now be written as

$$f(X) = \sum_{ij} x_{ij} \left[c_{ijij} \left(x_{ij} \mid p = i,\ q = j\right) + \sum_{\substack{pq \\ p\neq i, q\neq j}} c_{ijpq}\, x_{pq}\right]$$

$$= \sum_{ij} x_{ij} \left[c_{ijij} + \sum_{\substack{pq \\ p\neq i, q\neq j}} c_{ijpq}\, x_{pq}\right]$$

$$= \sum_{ij} b_{ij}\, x_{ij} \qquad (3.2)$$

where

$$b_{ij} = c_{ijij} + \sum_{\substack{pq \\ p\neq i, q\neq j}} c_{ijpq}\, x_{pq} \qquad (3.3)$$

The term b_{ij} may be defined as the "cost" of locating facility j at site i. The equivalent Koopmans and Beckmann problem can be formulated by defining the coefficients as

$$c_{ijpq} = \begin{cases} a_{ij} + w_{ij}\, d_{ii} & \text{if } i=p, j=q \\ w_{jq} \cdot d_{ip} & \text{otherwise} \end{cases}$$

Hence the problem reduces to the determination of the permutation matrix X so as to

$$\text{Minimize } f(X) = \sum_{ij} b_{ij}\, x_{ij} \qquad (3.4)$$

where

$$b_{ij} = a_{ij} + \sum_{\substack{pq \\ p\neq i, q\neq j}} w_{jq}\, d_{ip} x_{pq}$$

The new matrices, $A = [a_{ij}]$, $W = [w_{jq}]$ and $D = [d_{ip}]$ are defined in Chapter 2.

In many practical formulations, including the present study, a_{ij} is assumed to be zero for $\forall_{ij}$.

3.3 Outlines of the Solution Methodology

The combinatorial nature of the problem indicates that for a case involving n facilities, there are $n!$ possible permutation matrices $X = [x_{ij}]$, that is, there are n! possible assignments of facilities to locations. It is computationally infeasible to generate all possible permutation matrices for moderate to large values of n. Therefore, the procedure proposed in this study generates certain sets of permutation matrices for further investigation. The methodology behind this procedure may be explained as follows.

In Equation (3.4), it is noted that if $b_{ij}=a_{ij}$, i.e., w_{jq} are zero for all (j,q), the problem reduces to that of linear assignment, and any of the existing techniques could be used to solve it. In QAP, however, the b_{ij}'s are dependent on the relative locations of facilities. This implies that, depending on the configuration of facilities, each b_{ij} may assume $(n-1)!$ values. The problem is how to compute the values of b_{ij}'s so that the solution of a linear assignment problem would minimize the function given by Equation (3.4).

The solution method proposed in this study is based on the following approach. As a preliminary trial, we may set the values of b_{ij}'s at their lowest level, and solve the resulting LAP. If the optimal objective function value of the LAP equals the objective function value of the QAP, the optimal solution is obtained. However, in most of the cases this is not realized because of the fact that the

permutation matrix may not have actual (optimal) values of b_{ij}'s associated with its assignment. The next step therefore is to determine the values of b_{ij}'s which are optimal. The amount by which the current value of b_{ij} (set at its lowest level) deviates from its actual value, derived from the solution to LAP, is defined as the "discrepancies". For example, consider the cost of locating facility j at site i given by

$$b_{ij} = \sum_{\substack{pq \\ p\neq i, q\neq j}} w_{jq}\ d_{ip}\ x_{pq} \qquad (3.5)$$

Let the values of b_{ij}'s computed such that the function given by Equation (3.5) is minimized. These values of b_{ij} may or may not be the results of the assignments, which minimize Equation (3.4) and have the allocation of facilities j at sites i for all values of i and j respectively. The deviations in the values of b_{ij}'s thus obtained are referred to as discrepancies. The values of b_{ij}'s must therefore be adjusted such that

$$b_{ij} = \sum_{\substack{pq \\ p\neq i, q\neq j}} w_{jq}\ d_{ip}\ x'_{pq} \qquad (3.6)$$

where $[x'_{pq}]$ is the permutation matrix which minimizes the function

$$f_{ij}(X) = \sum_{\substack{pq \\ p\neq i, q\neq j}} b_{pq}\ x_{pq} \qquad (3.7)$$

The solution methodology thus seeks to remove the discrepancy in the values of b_{ij}'s in a number of iterations. Once all the values of b_{ij}'s have been adjusted, a permutation matrix X is obtained which minimizes Equation (3.4). The objective function value corresponding to this permutation matrix represents a new, improved value for the solution to the QAP. The procedure is then repeated until a desired accuracy in the solution to the QAP is obtained. The basic steps used in the development of this procedure can be summarized as follows:

Step 1: construction of matrix B of lower bounds on b_{ij}'s, the costs of locating facilities at different sites.

Step 2: Determination of permutation matrix $X = [x_{ij}]$ or solution to the LAP which minimizes the function given by Equation (3.4) with respect to current values of b_{ij}'s as obtained in matrix B. If the actual objective function value corresponding to this permutation matrix is equal to the lower bound, stop; otherwise proceed further.

Step 3: Improving the elements of matrix B.

3.1 Fix facility q at site p such that

$$B_{pq} = [b_{ik}] \quad i \neq p,\ q \neq k$$

3.2 Determine the permutation matrix by solving the LAP which minimizes the function given by Equation (3.4), and which

includes the allocation of facility q at site p. Let the reduced matrix be B'_{pq} and the resulting permutation matrix be $X^t = [x^t_{ij}]$. Adjust the value in the cell (p,q) of matrix B such that

$$b_{pq} = \sum_{\substack{ij \\ i \neq p, j \neq q}} w_{jq} d_{ip} x^t_{ij}$$

3.3 Repeat steps 3.1 and 3.2 for p = 1,..,n; q = 1,..,n.

Step 4: Determine X which minimizes Equation (3.4) with respect to current values of b_{ij}.

Repeat the whole procedure as long as the objective function value is improving.

CHAPTER IV

FACTORS INFLUENCING THE SOLUTION QUALITY

4.1 Introduction

In general, in step 3.2 of Section 3.3, there may result different permutation matrices X^t. In such cases evaluation of partial cost b_{pq} needs further investigation, due to the fact that different values of b_{pq}, obtained by using different permutation matrices, might affect the objective function value as well as the total number of iterations. In this respect, three criteria for the selection of the permutation matrix are proposed for further investigation in conjunction with the algorithm.

4.2 The Best Assignment Criterion

The partial cost b_{pq} is improved according to permutation matrix X^{t*}, which minimizes the objective function given by Equation (3.4). The matrix consists of the admissible cells in the reduced matrix B'_{pq} obtained in step 3.2 of Section 3.3.

The idea of improving b_{pq} with respect to the best assignment is to compute all the b_{ij}'s from the least costly assignments. It is, therefore, expected that the final permutation matrix comprising these b_{ij}'s would result in a solution of high quality.

The investigation of this criterion is, thus, based on the evaluation of the best assignment obtained from the permutation matrices in step 3.2 of Section 3.3. This requires a systematic procedure for identification and evaluation of these assignments. In this regard, Murty [1968] has described a procedure which ranks all the assignments in the order of increasing cost. The method requires the solution of at most (n-1) different assignments to obtain just one additional assignment having the same or higher cost. Since in the course of investigating the present criterion, we only need to generate assignments which have the same costs, it would be computationally infeasible to use Murty's algorithm without the necessary modifications. It is the purpose of this section to develop a systematic and efficient procedure for generating iSO cost assignments, based on Murty's algorithm.

The basic steps used by Murty in ranking the assignments can be summarized as follows [Murty, 1968]:

(1) Find the permutation matrix X^t which minimizes the given objective function for the LAP. Assume this matrix is:

$$X^t_{i_1 j_1} = 1, \quad X^t_{i_2 j_2} = 1, \ \ldots, \ X^t_{i_n j_n} = 1$$

where i and j are site and facility indices, respectively. Place the optimal solution in the list. Initialize the parameters t = 1 and d = 0.

(2) Remove the least costly solution from the list and output this solution as X^t.

(3) If X^t is obtained by fixing facilities $j_1,..,j_d$ at sites $i_1,...,i_d$, respectively, then leave these facilities fixed as they are, and create (n-d) new nodes or problems by fixing the rest of the facilities as follows:

$$M_{d+1} = \{X_{i_1 j_1} = 1,...,X_{i_d j_d} = 1,\ X_{i_{d+1} j_{d+1}} = 0\}$$

$$M_{d+2} = \{X_{i_1 j_1} = 1,...,X_{i_d j_d} = 1,\ X_{i_{d+1} j_{d+1}} = 1,\ X_{i_{d+2} j_{d+2}} = 0\}$$

$$\vdots \qquad\qquad \vdots$$

$$M_{n-d} = \{X_{i_1 j_1} = 1,...,X_{i_{n-d-1} j_{n-d-1}} = 1,\ X_{i_{n-d} j_{n-d}} = 0\}$$

Compute the optimal solutions to each of these (n-d) nodes and place each solution in the list together with the record of facilities which were fixed for each of them. Set t=t+1 and go to step 2.

When this basic algorithm is applied to find several solutions having the same cost, it requires solving (n-d) different assignment problems, one each of size 2,3,..,(n-d), with respect to the corresponding nodes, just to obtain one additional assignment. If there are K solutions having the same cost, it requires solving

K(n-d) assignment problems. Here d is a variable which depends upon the number of facilities fixed for a particular node.

Modification to Murty's Method

Several assignments having the same cost can be obtained by just solving one assignment problem using Hungarian or any other improved methods. Other assignments follow by branching the first assignment into different nodes exactly as in Murty's method. But all these nodes are not solved in the assignment; instead they are checked for feasibility to qualify for an assignment having the same cost. If the node is feasible, the assignment is made and recorded in the list together with facilities which were fixed for that node. Infeasible nodes are not recorded.It is further noted that in this procedure, to determine the assignments, which are made with respect to feasible nodes, it is not required to carry out all the computations; some steps, such as the reduction of the cost matrix may be saved.

The modified procedure for generating the permutation matrices in connection with the proposed procedure in Chapter 3 can be outlined as follows:

(1) Store the permutation matrix X^t identified by the LAP algorithm in step 3.2 of Section 3.3. Place the permutation matrix $X^t = [x^t_{ij}]$ in the list. Find the QAP's objective function value C_1 according to this permutation matrix, and store it in X^{t*}.

Output the solution X^t, t=1, and initialize parameter d=0.

(2) If X^t is obtained by fixing facility j, $j=j_1,..,j_d$, at site i, $i=i_1,...,i_d$, respectively, then leaving these facilities as they are, find another permutation matrix which satisfies the following:

$$X_{i_1 j_1} = 1,\ X_{i_2 j_2} = 1,\ldots,X_{i_d j_d} = 1,\ X_{i_{d+1} j_{d+1}} = 0.$$

(3) If permutation matrix in (2) consists of only the admissible cells of the matrix B^1_{pq}, go to step (4), otherwise go to step (5).

(4) Find the QAP's objective function value, C_2 according to the x permutation matrix obtained in step (3). C_2 is less than or equal to the minimum cost cost found so far; store the permutation matrix in X^{t*}; add this solution to the list, i.e., t=t+1 and then go step (2), otherwise go to step (5).

(5) Set d = d+1. If $d \leq (n-1)$, go to (2); otherwise set d = 0. Delete the permutation matrix just considered from the list, and pick the next permutation matrix. If list is empty, stop, otherwise go to step (2).

At the end of the application of above steps we have a permutation matrix X^{t*}. This matrix is then used to evaluate b_{pq} in step 3.2 of Section 3.3.

A computer code, developed by Metrick and Maybee, with a number of

modifications, listed above, has been used for generating best assignment in this study. A listing of this computer code is given in the Appendix IV.

4.3 The Least-Allocation-Cost Criterion

The cost b_{pq} is improved according to the permutation matrix X^{t*} which minimizes b_{pq} and consists of admissible cells in the reduced matrix B'_{pq} obtained in step 3.2 of Section 3.3

The idea of improving b_{pq} with respect to the least-allocation cost is to compute all the b_{ij}'s such that they assume their smallest values. It is therefore expected that the final permutation matrix comprising these b_{ij}'s would result in a lower value of the objective function.

The investigation of this criterion requires the evaluation of least allocation cost assignment in the reduced matrix resulting from step 3.2 of Section 3.3. This is done by formulating an LAP, which is discussed below.

Formulation of LAP to Find the Least-Allocation-Cost Assignment

The new LAP is formulated based on the reduced matrix $[b'_{ij}]$ in step 3.2 of Section 3.3. The formulation is:

$$\text{Minimize } b_{ij} = \sum_{pq} w_{jq}\, d_{ip}\, x_{pq}$$

$$p \neq i,\ q \neq j$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1,..,n)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1,..,n)$$

$$x_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j$$

$$b'_{pq} = \infty \text{ for all } b'_{pq} \neq 0$$

The solution to this LAP results in the permutation matrix X^{t*} which gives the least-allocation cost assignment. This is then used to find the value of b_{pq} in step 3.2 of Section 3.3.

4.4 The Psuedo-Random-Assignment Criterion

The cost b_{pq} is improved according to the permutation matrix X^{t*} identified by the LAP algorithm within the admissible cells in the reduced matrix obtained in step 3.2 of Section 3.3. The investigation of this criterion does not need any extra manipulation other than that described in Section 3.3.

It is noted that the criteria presented here are in the order of increasing computational efficiency. But the first two criteria are expected to contribute to the solution quality more than the last one. However, if there are no alternative solutions at each stage for the evaluation of partial costs b_{pq}, all three criteria are performed equally well.

CHAPTER V

FACTORS INFLUENCING THE SOLUTION EFFICIENCY

5.1 Introduction

In developing a solution procedure to QAP, computational efficiency and quality of the solution are equally important. However, most of the methods developed in the past, emphasize one criterion at the expense of the other. That is, they produce either good quality solutions in inordinate amounts of computer times, or inferior solutions in relatively short times.

As mentioned earlier, the aim of this study is to develop a solution procedure which takes into consideration both the quality and efficiency criteria. The factors that influence the solution quality were discussed in Chapter 4. The effects of these various factors on computational time are different, depending upon the choice of the criterion. The purpose of this Chapter is to study the factors that affect the efficiency of the solution procedure in such a way that the basic steps of the algorithm are unaffected. These factors are explained in the following sections.

5.2 Linear Assignment Problem (LAP)

The necessity of the study of LAP is obvious. There is a multitude of LAPs which need to be solved at various steps of the algorithm. The efficiency of the algorithm, therefore, depends to a great extent on the efficiency of the LAP code used. There are several solution techniques which have been developed in the past. Kuhn [1955] developed the Hungarian algorithm. Munkers [1957] presented an algorithm which is a variant of Kuhn's Hungarian algorithm. Branch and bound, and linear programming approaches have also been used to solve the LAP [Hillier and Liebermann, 1967].

Kuhn's Hungarian algorithm, which is used in this study, is known to be the most efficient. But computer codes are not easily accessible. The LAP computer code used in this study was developed by Metrick and Maybee [1973], and is essentially based on the Hungarian method. It contains a number of modifications to the Hungarian method in order to make the computer code more efficient. A listing of the computer code is given in Appendix IV.

5.3 Determination of Several Solutions to LAP Simultaneously

It is seen that for a problem of size nxn, the algorithm, in step 3 of Section 3.3, solves n linear assignment problems in order to improve n elements (B(i,k), k=1,..,n) of a given row i of matrix B. The cost matrices corresponding to these n linear assignment problems differ only in one column from one another, which

points out the need for an efficient procedure to find the n solutions in a more efficient way. The following description outlines a procedure to find n solutions simultaneously with less computational effort.

The method of finding simultaneous solutions which contain one of the assignments $x_{ij}^{(0)} = 1$, $j = 1,2,..,n$, is based upon the reduced coefficient matrix. This is a matrix of non-negative elements which is obtained at the end of the reduction procedure in solving linear assignment problems (LAP). Suppose $B^{(0)} = b_{ij}^{(0)}$ is the matrix of reduced coefficients; then the elements satisfy the following conditions:

$$b_{ij}^{(N)} \geq 0 \qquad \text{for } i = 1,2,..,n \quad j = 1,2,..,n$$

$$b_{ij}^{(N)} = 0 \qquad \text{if } x_{ij} = 1$$

For any assignment problem, there is a reduced coefficient matrix $B^{(N)}$, obtained from $B^{(0)}$ in N steps, which has the following properties [Kreuzberger and Weiterstadf, 1971]:

$$b_{ij}^{(N)} \geq 0 \qquad i = 1,2,..,n; \quad j = 1,2,..,n \tag{5.1}$$

$$b_{ij}^{(N)} = 0 \qquad \text{if } x_{ij}^{(0)} = 1 \tag{5.2}$$

$$b_{Nj}^{(N)} = \Delta L_{Nj} \qquad N \in n \quad j = 1,2,..,n \tag{5.3}$$

where

$$\Delta L_{Nj} = L_{Nj} - L^{(0)}; \quad \Delta L_{Nj} \geq 0 \tag{5.4}$$

and

L_{Nj} = the minimum value of the objective function sought in the assignment of facility j to site N.

$L^{(0)}$ = the optimal value of the objective function sought.

From (5.3) and (5.4) it results that

$$L_{Nj} - L^{(0)} = b_{Nj}^{(N)}$$

Thus the elements $b_{Nj}^{(N)}$ of row N of the reduced coefficients matrix specify the amount by which the minimum objective function value sought in an assignment containing $x_{Nj} = 1$ exceeds the value $L^{(0)}$.

The method of finding the reduced coefficients matrix $B^{(N)}$, which satisfies the conditions of (5.1) through (5.3) is described below. This matrix would then be used to determine all the assignment vectors which contain one of the assignments $x_{kj}^{(0)} = 1$, $j = 1,2,..,n$ for row k of the matrix.

(1) Find the reduced coefficients matrix $B^{(0)} = [b_{ij}^{(0)}]$ and the permutation matrix $X^{(0)}$ of the assignment problem.

(2) Add a constant $M_0 > 0$ to all the elements $b_{kj}^{(0)}$, $j \in n$, of row k of matrix $B^{(0)}$.

(3) Substract M_0 from all the elements of column ℓ. The index ℓ is obtained such that $x_{k\ell}^{(0)} = 1$. The results of steps 2 and 3

are the following matrix:

$$b_{ij}^{(1)} \begin{cases} b_{ij}^{(0)} & \text{for } i=k,\ j=\ell \\ b_{ij}^{(0)} + M_0 & \text{for } i=k,\ j\neq\ell \\ b_{ij}^{(0)} - M_0 & \text{for } i\neq k,\ j=\ell \\ b_{ij}^{(0)} & \text{otherwise} \end{cases}$$

(4) Compute the parameter M_1 for the next iteration:

$$M_1 = \max_{\substack{j\in n \\ i\in n}} \{ -b_{ij}^{(1)} \mid b_{ij}^{(1)} < 0\} = -b_{rs}^{(1)}$$

(5) Add M_1 to all the elements of row r of $B^{(1)}$, and substract M_1 from all the elements of column t of $B^{(1)}$. The index t is determined such that $x_{rt}^{(0)} = 1$.

(6) The results are the following matrix of reduced coefficients:

$$B^{(2)} \begin{cases} b_{ij}^{(0)} & \text{for } i=r,\ j=t \\ b_{ij}^{(1)} + M_1 & \text{for } i=r,\ j\neq t \\ b_{ij}^{(1)} - M_1 & \text{for } i\neq r,\ j=t \\ b_{ij}^{(1)} & \text{otherwise} \end{cases}$$

(7) The procedure is repeated until matrix $B^{(n)}$ is obtained. This matrix contains n new permutation matrices which corresponds to the assignments $x_{kj} = 1$, $j = 1,2,..,n$. The identification of these assignment vectors may be carried out using any labelling

scheme.

The mathematical proof of this procedure is given in Kruezberger and Weiterstadf [1971]. A summary of this paper is prepared in English and appears in Appendix I.

CHAPTER VI

SUMMARY OF SOLUTION PROCEDURE

6.1 Introduction

It is well known that the QAP is combinatorial in nature. There are n! possible assignment vectors in its solution space. As noted in the literature, it is computationally infeasible to generate all the assignment vectors even for moderate values of $n(10 \leq n \leq 30)$. The algorithm summarized here, therefore, generates only improved assignment vectors using sequential search procedure. The motivation behind the solution procedure is given in Chapter 3. The solution procedure starts by constructing a matrix of lower bounds on the costs of locating facilities at different sites. It then seeks to remove the discrepencies in the elements of this matrix by solving a succession of linear assignment problems. Different criteria could be used for removing the discrepencies in the elements of this matrix, depending upon the quality of solution desired as discussed in Chapter 4. The efficiency of the solution using each criterion is dependent on the computer code used for solving the LAP. The efficiency is further improved by attaining n solutions to LAP simultaneously, hence removing the discrepencies in n elements of this matrix in less

computational time. This idea was discussed in Chapter 5.

The various criteria for improving the quality of the solution require varying amounts of computational effort. The purpose of each criteria is to determine an assignment to be used in Step 3.2 of Chapter 3. If each criteria select the same assignment for improving b_{pq} in Step 3.2 of Chapter 3, the quality of the solution obtained would be the same for each criteria. The algorithm summarized in this Chapter, therefore incorporates only Pseudo-Random assignment criterion, which obviously is the most efficient criterion from the view point of efficiency. This algorithm would need to be modified, if other criteria are used. If the Best assignment or least-allocation cost criterion is used, the computer program given in Appendix IV for generating best assignment or least-allocation cost assignment must be incorporated in the algorithm.

6.2 The Algorithm

Step 1: Development of the matrix of lower bounds on the costs of locating facilities at different sites [Francis and White, 1974].

1.1 Let w(k) be the row vector obtained from row k of matrix $W = [w_{ij}]$ by deleting the element in column k of row k of w. Then, let $\bar{w}(k)$ be the row vector obtained by ordering the elements of w(k) so that they are non-decreasing.

1.2 Let d(i) be the row vector obtained from row i of the distance matrix $D = [d_{ij}]$ by deleting the element in column i of row i of matrix D. Then, let $\bar{d}(i)$ be the row vector obtained by ordering the elements of d(i) so that they are non-increasing.

1.3 For i = 1,..,n and k = 1,..,n, find the lower bound matrix B whose elements are calculated as

$$b_{ik} = [\bar{w}(k)][\bar{d}(i)]'$$

where b_{ik} is a lower bound on the "cost" of locating facility k at site i, as given by equation (4).

1.4 Solve the linear assignment problem (LAP) having the cost matrix $B = [b_{ik}]$. Let the resulting permutation matrix be $X^{(0)} = [x_{ij}^{(0)}]$. If the lower bound corresponding to the assignment represented by this permutation matrix is equal to its actual cost, stop; otherwise set the parameter p=0, and go to step 2.

Step 2: It was observed that for a problem of size n, the algorithm would normally solve n linear assignment problems in order to improve the n elements (b_{ik}, k=1,..,n) of a given row i of matrix B [Lashkari and Jaisingh, 1980]. The cost matrices corresponding to these n assignment problems differ in only one column with one another, which points out the possibility of obtaining the n

solutions in a more efficient way. The procedure to find n solutions simultaneously with less computational efforts has been discussed in Section 5.3 of Chapter 5. The following steps outline the method of generating n solutions simultaneously. Let the resulting matrix after solving the LAP in step 1.4 be $B^{(0)} = [b_{ij}^{(0)}]$ and the permutation matrix be represented by $X^{(0)}=[x_{ij}^{(0)}]$. The matrix $B^{(0)}$, known as the reduced coefficients matrix, consists of non-negative elements having specified properties; it is possible to reduce this matrix further in N steps, to a matrix $B^{(N)}$ such that it contains all the solutions which have one of the following allocations $x_{m1}^{(0)} = 1$, $x_{m2}^{(0)} = 1,..,x_{mn}^{(0)} = 1$ corresponding to a given row m of matrix $X^{(0)}$ (see Section 5.3, Chapter 5). This matrix can be determined as follows:

2.1 Set the parameters p = p+1, k = p, N = 0, q = 1 and select ℓ such that $x_{k\ell}^{(0)} = 1$.

2.2 Increment parameter N by 1. For current value of N find $B^{(N)}$ such that

$$B^{(N)} = \left[b_{ij}^{(N)}\right] = \begin{cases} b_{ij}^{(0)} & \text{for } i=k,\ j=\ell \\ b_{ij}^{(N-1)} + M_{N-1} & \text{for } i=k,\ j\neq\ell \\ b_{ij}^{(N-1)} - M_{N-1} & \text{for } i\neq k,\ j=\ell \\ b_{ij}^{(N-1)} & \text{otherwise} \end{cases}$$

where $M_0 >> 0$.

2.3 Find the parameter

$$M_N = \max_{\substack{i\in n \\ j\in n}} \left\{ -b_{ij}^{(N)} \mid b_{ij}^{(N)} < 0 \right\} = -b_{rs}^{(N)}$$

2.4 Set the parameters k=r and ℓ=t, where t is such that $x_{rt}^{(0)} = 1$.

2.5 Repeat steps 2.2, 2.3 and 2.4 for N=1,2,..,n. The final matrix $B^{(n)}$ thus obtained contains n solutions which have one of the following assignments

$x_{p1} = 1$, $x_{p2} = 1,..,x_{pn} = 1$.

Step 3: Improving the elements of matrix B.

3.1 For improving the element corresponding to any cell (p,q) of matrix B, delete row p and column q in the current matrix $B^{(n)}$ and denote the resulting matrix by B_{pq}. Thus $B_{pq} = [b_{ik}^{(n)}]$, $i\neq p$, $k\neq q$. This step amounts to assigning facility q to site p.

3.2 Identify the assignment of the rest of the facilities, without further reduction of the matrix B_{pq}, using any labelling scheme. Let the resulting permutation matrix be $X^{(t)} = [x_{ij}^{(t)}]$.

3.3 Replace the elements in cell (p,q) of matrices $B^{(n)}$ and B by the new elements computed as follows: =

$$b_{pq}^{(n)}(U) = \sum_{j=1}^{n} w_{qj}\, d_{pv} - b_{pq} + b_{pq}^{(n)}$$

and $b_{pq} = \sum_{j=1}^{n} w_{qj}\, d_{pv}$ where v is obtained such that $x_{vj}^{(t)} = 1$.

3.4 Repeat steps 3.1, 3.2 and 3.3 for q=1,..,n. This updates the elements of the row p in the matrices $B^{(n)}$ and B.

3.5 Replace the elements in row p of matrix $B^{(n)}$ such that $b_{pq}^{(n)} = b_{pq}^{(n)}(U) - M^*$. M^* = minimum number of row p of matrix $B^{(n)}$. This is done to achieve at least one admissible cell in row p.

3.6 Repeat steps 2.1 through 3.4 for p = 1,2,..,n.

3.7 Solve the linear assignment problem having the updated cost matrix $B^{(n)} = [b_{ij}]$. The cost of the resulting assignment is an improved objective function value of the QAP.

Steps 2 and 3 are repeated as long as the objective function value of the QAP in step 3.7 is improving.

6.3 Stopping Criteria

The solution procedure summarized above seeks to remove infeasibilities in the elements of matrix $B = [b_{ij}]$. Each b_{ij} is improved by finding a permutation matrix. If the element b_{pq} is improved corresponding to the assignment vector defined by permutation matrix X^t, and the element b_{rs} corresponding to assignment vector defined by permutation matrix $X^{t'}$, both assignment vectors may

have one or more common elements (say (u,v)) . But this element (u,v) may be improved according to a completely different assignment obtained from the solution to LAP in step 3. These situations may change the direction of search and in some cases the procedure may tend to oscillate with steadily decreasing rate of improvement in the solution. It is therefore essential to stop the computation at some point where

$$\frac{|TC_i - TC_{i-1}|}{TC_{i-1}} \leq \epsilon$$

where TC_i is the objective function value of the assignment obtained at iteration i. If this criterion is not satisfied within a reasonable number of iterations, the procedure could be stopped after a pre-specified number of iterations.

CHAPTER VII

DISCUSSION OF RESULTS

7.1 Introduction

The purpose of this Chapter is to discuss the performance of the proposed algorithm from the view point of the efficiency and quality of the solution. In this context, the algorithm is applied to various test problems given by Nugent, et al. [1968], Elshafei [1977] and randomly generated problems of sizes ranging from 5x5 to 60x60. The superiority of the algorithm is demonstrated and results are discussed in section 7.2. Further, the sensitivity of the algorithm to variations in the parameters of the distance or the flow matrix are discussed in section 7.3.

7.2 Application of Algorithm to Test Problems

(i) Test Problems Suggested by Nugent, et al. [1968]. In recent years the eight test problems given by Nugent, et al. [1968] have become a challenge to researchers. These problems have been solved several times in the past using different approaches - and every year some improvements are being reported. The proposed algorithm is applied to these problems, and the computational results of the present

algorithm, as well as those of Hillier [1963], Hillier and Connors [1966], Armour and Buffa [1963], Nugent, et al. [1968], Khalil [1973], Hitchings and Cottam [1976], Elshafei [1977], and Bazaraa and Sherali [1979] are summarized in Tables 7.1 and 7.2. The data for these problems are given in Appendix II.

From Table 7.1 it is seen that the proposed algorithm produces high quality solutions for the problems under consideration. The results are reasonably close to the best solutions known so far. The"best cost assignment" criterion, as expected, is found to be the most time consuming. The efficiency of this criterion is dependent upon the number of alternate solutions resulting in step 3.2 of the algorithm. It is further observed that these 8 test problems have numerous alternate solutions. The investigation of this criterion for large size problems is therefore dropped. The "least-allocation Cost" criterion was found to produce slighly higher quality solutions as compared to "pseudo-random-assignment" criterion, although this is at the expense of compuational time. But the differences in the quality are not significant. In contrast to the improvement methods, the proposed procedures is independent of the starting solution and also converges rapidly. In all cases, the number of iterations required to satisfy the stopping criterion at $\epsilon = 0.05$ is not more than 10.

The computational efficiency of the proposed algorithm, as compared with other heuristic procedures, is shown in Table 7.2.

Table 7.1 Results (Cost Units) of Different Heuristic Procedures

Problem Size, n	Hillier, 1963	Hiller & Connors, 1966	Armour & Buffa 1963	Nugent et al. 1968	Khalil, 1973	Hitchings & Cottam 1976	Elshafei, 1977	Bazaraa & Sherali 1980	Proposed Procedure			Best Objective Function Value (Cost Units)
	Average Cost*				Best Cost†			Cost	Best Cost Asgmt	Least Allocation Cost Asgmt.	Pseudo-Random Asgmt.	
5	27.6	29.4	28.2	27.6	25	25	26	25	25	25	26	25
6	44.2	44.2	44.2	44.6	43	43	43	43	43	43	43	43
7	78.8	78.4	79.6	77.2	74	74	75	74	74	74	76	74
8	114.4	110.2	113.4	111.6	107	107	114	107	107	107	118	107
12	317.4	310.2	296.2	304.6	295	289	296	289		292	296	289
15	632.6	600.2	606.0	603.0	575	575	614	575		576	585	575
20	1400.4	1345.0	1339.0	1339.0	1300	1296	1299	1285		1294	1320	1285
30	3267.2	3206.8	3189.6	3189.6	3129	3086	3125	3095		3094	3088	3077

* Average costs evaluated as a result of several starting solutions

† Highest quality solutions produced. In general, the solution would be dependent on starting solution Asgmt. assignment

Table 7.2 Summary of Computational Times (secs) Per Solution

Problem	Hillier, 1963*		Hillier & Connors, 1966*		Armour & Buffa 1963*		Nugent, et al., 1968*		Khalil, 1973††		Hitchings& Cottam 1977#	Bazaraa & Sherali 1980°		Proposed Procedure $		
														Best Cost Asgmt.	Least allocation Cost Asgmt.	Pseudo Random Asgmt.
Size, n	sec*	sec**	sec*	sec**	sec*	sec**	sec*	sec**	sec†	sec**	sec#	sec°	sec**	sec	sec	sec
5	6.0	.25	10.0	.42	1.0	.04	1.1	.05	.11	.06	,25	.69		0.31	.06	.026
6	7.0	.29	9.0	.38	2.0	.08	2.1	.09	.15	.08	.30	1.30		2.81	.12	.04
7	15.0	.63	12.0	.50	5.0	.21	5.7	.24	.25	.13	.75	2.25		2.99	.21	.06
8	14.0	.58	14.0	.58	10.0	.42	10.9	.45	.33	.17	1.23	3.56		4.52	.36	.10
12	55.0	2.29	19.0	.79	70.0	2.92	65.8	2.74	1.25	.63	6.84	13.69			1.73	.36
15	78.0	3.25	40.0	1.67	160.0	6.67	219.2	9.13	2.96	1.48	20.13	33.93			4.23	.77
20	168.0	7.00	75.0	3.13	528.0	22.0	691.5	28.81	10.87	5.43	59.87	100.61			16.95	2.11
30	398.0	16.58	285.0	11.88	3152.0	131.33	4272.4	178.0	58.3	29.15	383.27	393.96			87.71	9.30

* ran on GE 265
\# ran on ICL 470 (Equivalent IBM 370/3031 times were not available & therefore not included)
†† ran on IBM 360
° ran on CDC Cyber 70 model 74-28/CDC 6400
$ ran on IBM 370/3031
** computational time normalized to IBM 370/3031

The times listed for Bazarra & Sherali's procedure are total time to attain the final solutions after specified number of cuts.

Under each procedure, the left column shows the amount of time spent to obtain the solution on the respective computer system, and the right column indicates the equivalent time (wherever obtainable) on IBM 370/3031 computer, the system we have used. The last column shows the computational times of the proposed algorithm. In comparing these times it should be noted that:

1. the conversion times are approximate.
2. the time reported for other heuristic procedures is only the average time per solution. We recall that these procedures try various starting solutions until a reasonable, final solution is obtained. Thus the total computer time spent to obtain the final results is many times larger than the reported time. Since these total times are not available, it is indeed very difficult to compare the computational time of our procedure with that of others.

(ii) Practical Problem Suggested by Elshafei [1977]. This is the problem of relative location of clinics within a hospital department. The objective is to decide upon the location of the various clinics so as to reduce the total effort spent by the patients while moving from one clinic to another. Thus the objective is to locate the clinics within the given building, so as to minimize the total distance travelled per year. The estimates of the patient flows among the 19 clinics and the distances among their locations are given in Appendix III.

This problem has been solved in the past, using three different approaches. The original layout, with an objective function value of 13,973,298, is given by Khorshid and Hassan [1974]. Elshafei [1977] solved this problem using his algorithm and achieved an objective function value of 11,281,887, a 19.2% improvement over the original layout. Bazaraa dn Sherali [1979] used their exact solution procedure with premature termination to solve this problem, and achieved further improvement in the layout, resulting in an objective function value of 8,606,274.

In this study, this problem is solved using the proposed algorithm, with the pseudo random assignment criterion. The objective function value of the best layout is found to be 8,683,664, which represents a 37.85% improvement over the original layout and a 23.03% improvement over the revised layout given by Elshafei. The difference in the quality of the solution between Bazaraa & Sherali and the proposed procedure is less than 1%.

The detailed results of computational time using the proposed algorithm are summarized in Table 7.3. It is seen from the Table that the algorithm is terminated after iteration #3 using a stopping criterion of $\epsilon \leq .0005$. The total time required to solve the problem completely is 6.497 seconds of CPU time. Elshafei reported that his procedure took 136 seconds of CPU time on IBM 360/40 to obtain full solution. Using a conservative conversion factor of 16 between

TABLE 7.3

Summary of Results for Practical Problem.

Iteration Number [A]	Objective Function Value [B]	Operations Performed [C]	Time for the Operations in [C](sec) [D]	Cumulative Time (sec) [E]
0	12,178,865	Construction of the lower bound matrix	0.936	0.936
1	8,712,748	Updating the lower bound matrix	1.846	2.782
2	8,687,067	"	1.849	4.631
3	8,683,664	"	1.866	6.497

IBM 360/40 and IBM 370/3031, the equivalent IBM 370/3031 time for Elshafei's procedure would be 8.5 seconds. It means that the proposed procedure reduces the computational time by 25%. Although Bazaraa and Sherali's procedure produced higher quality solution, it did so at the expense of the computational time; it took about 96 seconds on a CDC Cyber 70 model 74-28/CDC 64W computer. Thus in comparison, the proposed algorithm produces reasonably high quality solutions in considerably less computational time.

(iii) Randomly Generated Large Problems. In the preceding sections the results of various test problems were summarized. Although the relative performance of the proposed algorithm is evidence from these results, it is not possible to judge the efficiency of the algorithm for large size problems, since the results are limited to problem sizes up to 30x30. To alleviate this shortcoming, and in order to examine the efficiency of the algorithm for large problems, a number of problems varying in size from 30x30 to 60x60 were randomly generated and solved using pseudo random assignment criterion. The average computational time per iteration and the number of iterations required to solve these problems completely are summarized in Table 7.4.

From Table 7.4 it is seen that the proposed algorithm can handle large size problems efficiently. In most cases the number of iterations required is not more than 10 to satisfy the stopping criterion at $\epsilon = .002$. For example, consider the problem of size 60x60; it takes

TABLE 7.4

Summary of Computational Experiences on Large Problems.

Problem Size	Average Time per Iteration (sec)	Number of Iterations Required for ≤ 0.002
30x30	9.8659	8
35x35	18.6460	7
40x40	29.3430	7
45x45	43.7560	9
50x50	65.8180	7
55x55	93.6120	10
60x60	131.0840	9

131.08 secs of CPU time per iteration and a total of 9 iterations to obtain the final solution. In contrast to this, the heuristic procedures based on improvement method would need to investigate several hundred starting solutions before achieving a reasonably good solution. This is because of the fact that the solution space for QAP increases tremendously with the increase in problem size. The plot of average time per iteration for various problem sizes ranging from 5x5 to 60x60 is given in Fig. 7.1, which indicates the exponential nature of the solution times as the problem size increases.

As for the improvement in the solution quality of large size problems, it is observed that the rate of improvement in the objective function value is very rapid in the first few iterations, but it slows down in the subsequent iterations. This is shown in Fig. 7.2 which demonstrates the improvement rate for a problem of size 40x40.

Further, in order to obtain an indication of the optimality of the solution, the algorithm was applied to a number of small size problems whose optimal solution could be obtained by the branch and bound or total enumeration method. The problems of size 4,5 and 6 were considered for this purpose. Fifty problems each of the three different sizes were randomly generated and then solved. The results are summarized in Tables 7.5, 7.6 and 7.7. Out of these 150 problems, the proposed algorithm resulted in optimal solution for 109 problems. For the rest, the results are very close to the optimal solutions. This amounts to a probability of almost 73% of obtaining optimal

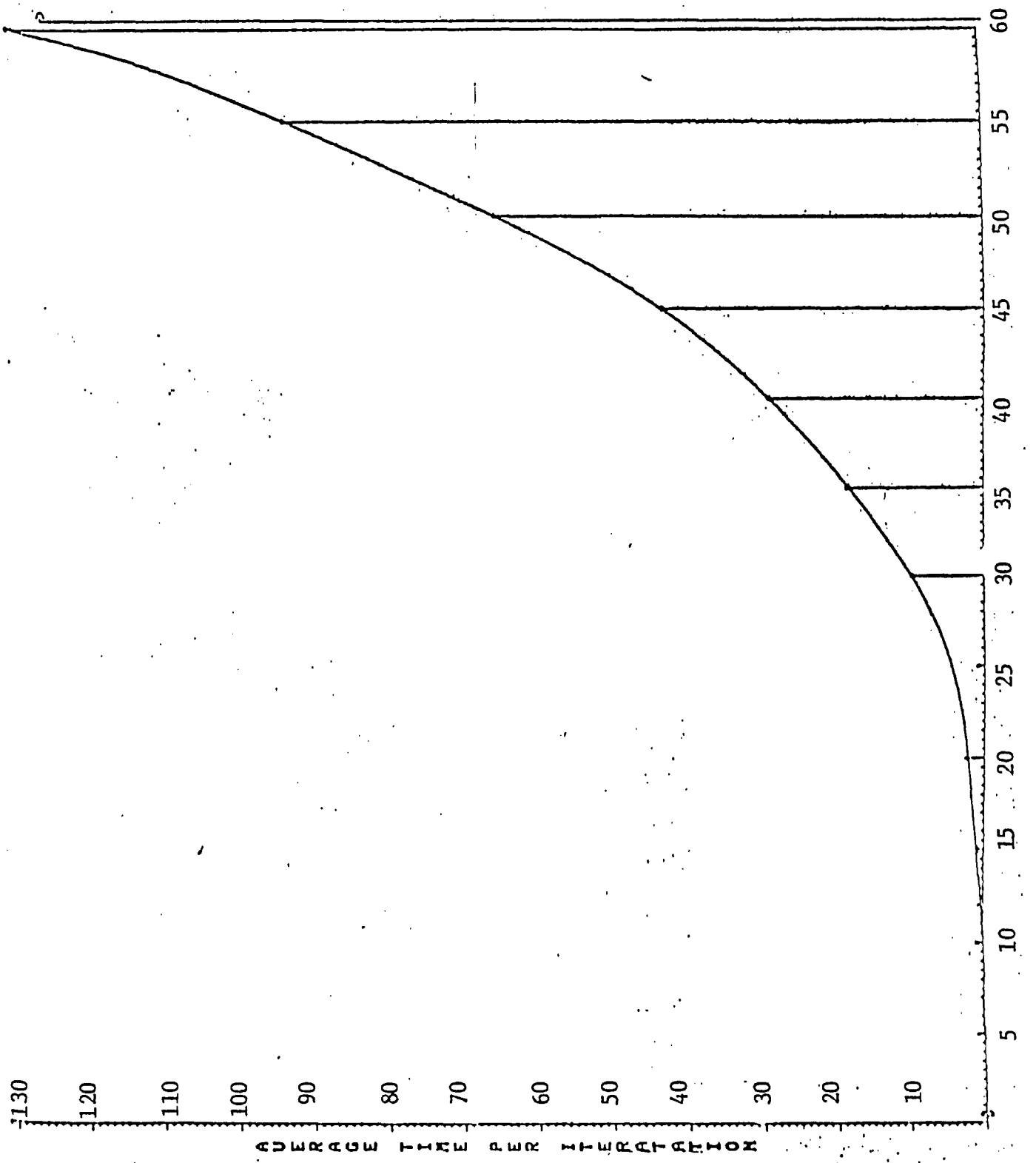


Fig. 7.1 Increase in Average Time per Iteration vs. Problem Size

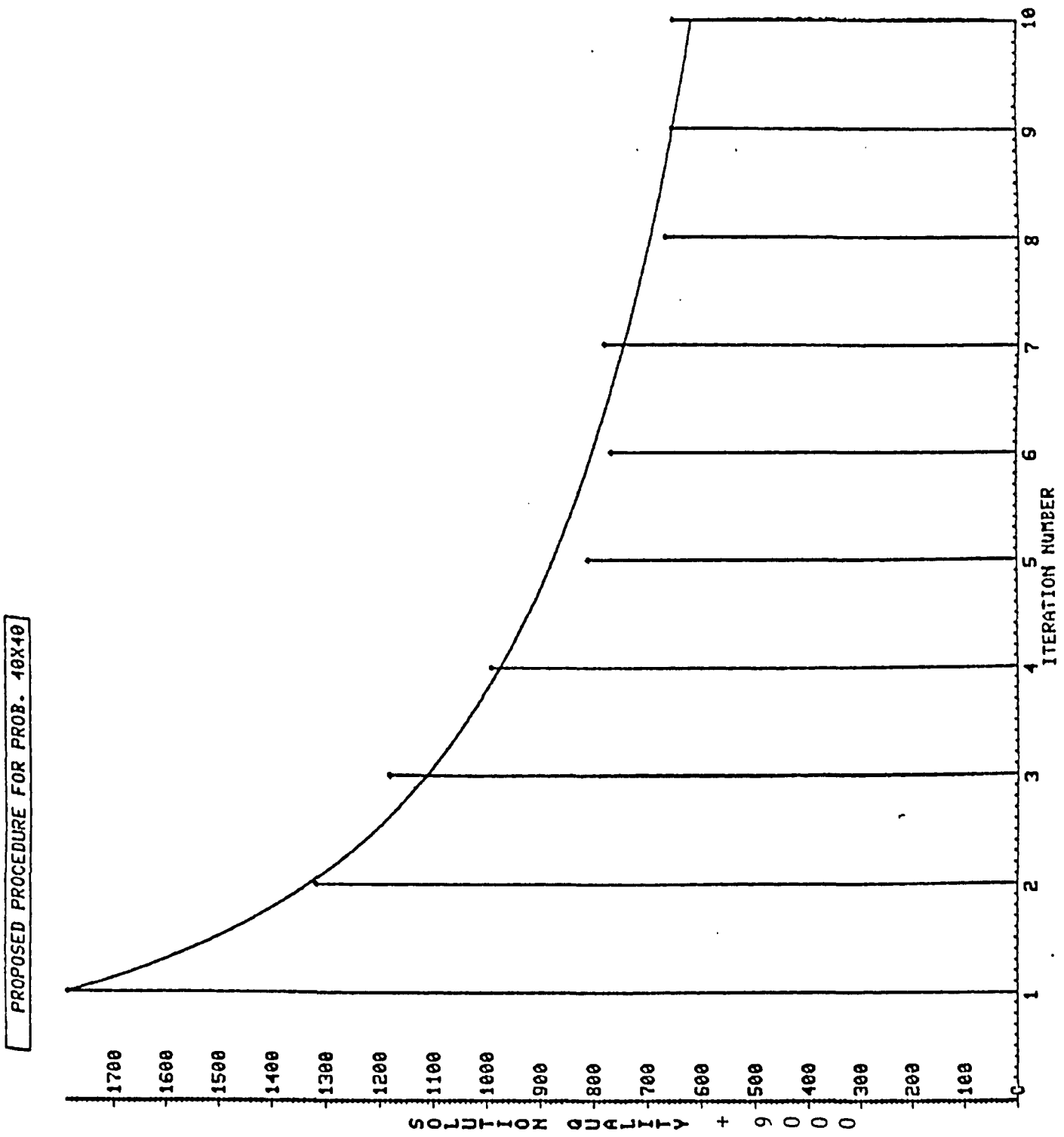


Fig. 7.2 Improvement in the Quality of Solution vs. Iteration Number

Table 7.5: Results of Random Generation of Problem Size 4x4

Problem Number	Optimal Cost Achieved or Not	% Deviation from the Optimum Solution	Total Time Consumed to Solve the Problem	Problem Number	Optimal Cost Achieved or Not	% Deviation from the Optimum Solution	Total Time Consumed to Solve the Problem
1	yes	0	.025	26	yes	0	.026
2	yes	0	.022	27	yes	0	.029
3	yes	0	.029	28	no	2.228	.026
4	yes	0	.026	29	yes	0	.029
5	yes	0	.049	30	yes	0	.029
6	yes	0	.029	31	no	1.912	.039
7	yes	0	.029	32	yes	0	.029
8	yes	0	.042	33	yes	0	.029
9	yes	0	.032	34	yes	0	.029
10	no	0.558	.029	35	yes	0	.026
11	yes	0	.029	36	yes	0	.029
12	yes	0	.029	37	no	0.982	.029
13	yes	0	.051	38	yes	0	.052
14	yes	0	.032	39	yes	0	.029
15	no	0.217	.026	40	yes	0	.026
16	yes	0	.029	41	yes	0	.026
17	no	0.295	.029	42	yes	0	.026
18	yes	0	.025	43	yes	0	.026
19	yes	0	.026	44	yes	0	.022
20	yes	0	.029	45	yes	0	.022
21	yes	0	.026	46	yes	0	.029
22	no	2.895	.039	47	yes	0	.029
23	yes	0	.032	48	yes	0	.026
24	yes	0	.029	49	yes	0	.029
25	yes	0	.032	50	yes	0	.026

Table 7.6: Results of Random Generation of Problem Size 5x5

Problem Number	Optimal Cost Achieved or Not	% Deviation from the Optimum Solution	Total Time Consumed to Solve the Problem	Problem Number	Optimal Cost Achieved or Not	% Deviation from the Optimum Solution	Total Time Consumed to Solve the Problem
1	yes	0	.049	26	yes	0	.075
2	yes	0	.104	27	no	0.898	.078
3	yes	0	.056	28	yes	0	.046
4	no	0.704	.058	29	no	2.953	.049
5	yes	0	.055	30	yes	0	.101
6	yes	0	.055	31	yes	0	.046
7	no	2.280	.055	32	no	4.991	.078
8	yes	0	.055	33	no	1.907	.049
9	yes	0	.049	34	yes	0	.069
10	no	2.727	.098	35	yes	0	.052
11	yes	0	.052	36	yes	0	0.49
12	yes	0	.052	37	no	2.517	.046
13	no	2.422	.081	38	yes	0	.049
14	no	1.477	.059	39	yes	0	.049
15	yes	0	.059	40	yes	0	.075
16	yes	0	.049	41	yes	0	.046
17	yes	0	.130	42	yes	0	.049
18	no	1.136	.055	43	yes	0	.052
19	no	1.215	.049	44	yes	0	.046
20	no	0.633	.049	45	yes	0	.052
21	yes	0	.062	46	yes	0	.078
22	yes	0	.052	47	yes	0	.081
23	yes	0	.052	48	no	1.103	.046
24	yes	0	.062	49	no	2.073	.049
25	no	6.308	.046	50	no	4.500	.049

Table 7.7: Results of Random Generation of Problem Size 6x6

Problem Number	Optimal Cost Achieved or Not	% Deviation from the Optimum Solution	Total Time Consumed to Solve the Problem	Problem Number	Optimal Cost Achieved or Not	% Deviation from the Optimum Solution	Total Time Consumed to Solve the Problem
1	yes	0	.135	26	no	0	.297
2	no	1.967	.263	27	yes	0	.160
3	no	0.566	.096	28	yes	0	.078
4	no	2.561	.086	29	yes	0	.082
5	no	0.736	.082	30	yes	0	.086
6	yes	0	.088	31	yes	0	.082
7	yes	0	.086	32	no	0.90	.128
8	yes	0	.082	33	yes	0	.210
9	yes	0	.089	34	yes	0	.121
10	no	5.709	.220	35	no	.891	.125
11	yes	0	.181	36	yes	0	.078
12	yes	0	.211	37	no	4.739	.085
13	yes	0	.164	38	yes	0	.079
14	yes	0	.124	39	no	3.536	.128
15	no	0.950	.082	40	yes	0	.118
16	yes	0	.085	41	yes	0	.086
17	no	1.118	.220	42	yes	0	.125
18	yes	0	.086	43	yes	0	.078
19	yes	0	.128	44	no	1.754	.078
20	yes	0	.088	45	yes	0	.078
21	no	1.814	.086	46	yes	0	.174
22	yes	0	.121	47	yes	0	.211
23	no	0.978	.171	48	yes	0	.078
24	no	2.430	.171	49	no	1.932	.085
25	yes	0	.141	50	yes	0	.214

solution when the algorithm is applied to a similar problem. However, it should be noted that this observation is based on the results obtained from problems which are small enough as to lend themselves to exact solution. The results may not be extended to large problems whose exact solutions are not readily available.

7.3 Sensitivity Analysis

The preceding sections were devoted to a discussion of the applicability of the algorithm to various problems. No mention was made of the effects of varying the parameters of the distance or flow matrix. A change in the parameters might result in a completely new layout. For this reason, it is important to perform a sensitivity analyses to investigate the effect on the layout, provided by the proposed algorithm, if the parameters assume other possible values. In some cases there may be parameters that can assume any value without affecting the layout. For instance, two facilities can be located independent of each other if there is no flow between them. However, when there are interflows among facilities, a change in distance and/or flow parameters would result in a change in the value of the objective function, but the layout may or may not remain the same. It is noted that sensitivity analysis would be expensive computationally, if it is necessary to reapply the proposed algorithm to investigate the effects of a changed parameter on the layout. Moreover, it is difficult to state, a priori, if the

information derived by the algorithm at the final iteration could be useful in any way to the process of sensitivity analysis. Therefore, investigation is carried out on several randomly generated problems of varying sizes. The original problems are solved using the proposed algorithm with the "random assignment" criterion. The effect of a parameter change in the distance or flow matrix, on the original layout, is studied in two ways:

(i) By reapplying the proposed algorithm; and

(ii) By using the information derived by the algorithm at the final iteration of the original problem. To elaborate, suppose matrix B^N is obtained at the end of the problem solving process in step 3.7 of the algorithm. The information contained in this matrix is then used to perform the sensitivity analysis. This is achieved by reapplying the algorithm to matrix B^N starting from step 2. The effect of the change in the parameter is taken into account in step 3.3 of the algorithm.

It is shown in the previous section, that the quality of the solution improves rapidly in the first few iterations, but the rate of improvement decreases subsequently. It is, therefore, expected that if sensitivity analysis is carried out as explained above, the number of iterations required to solve the problem completely, and thus the computational efforts, would be reduced.

As an example, consider the effect of varying parameter w_{18} for a problem of size 10x10. The data for this problem are given in

Table 7.8. The problem is first solved using the random assignment criterion. The layout thus obtained and the corresponding objective function value are listed in Table 7.9. The parameter w_{18} is then varied from 0 to 205 in steps of 5. The investigation is then carried out by reapplying the algorithm as well as using the information available in the last iteration of the original problem solving process. The results are depicted in Figures 7.3 and 7.4 and Table 7.9. It is found that when the algorithm is reapplied, the value of objective function increases with an increase in the value of w_{18}. However, the increase in the objective function value does not necessarily mean that the layout would be different. For example, from Table 7.9 it is observed that a change in the value of w_{18} from 0 to 25 results in different layouts. This is represented by points A through G_1 in Figure 7.3. As w_{18} increases from 25 to 30, the layout remains the same, but the objective function value increases. This is shown by joining points G_1 and G_2 on the graph. At $w_{18} = 35$, the layout changes again; point I on the graph shows this new objective function value. However, as w_{18} increases from 35 to 40, a new layout results, which remains the same for values of w_{18} greater than 40. The objective function value, however, increases correspondingly. This fact is depicted by the straight line joining points $J_1, J_2 \ldots$ on the graph.

On the other hand when the sensitivity analysis is carried out using the information available in the final iteration of the original

Table 7.8 Data For Sample Problem of Size 10x10 For Sensitivity Analysis

Distance Matrix

0	9	8	5	11	7	8	8	1	10
9	0	4	1	8	10	7	8	4	5
8	4	0	3	10	7	1	8	10	12
5	1	3	0	5	8	5	4	3	8
11	8	10	5	0	9	4	11	1	3
7	10	7	8	9	0	1	4	1	12
8	7	1	5	4	1	0	9	4	10
8	8	8	4	11	4	9	0	8	9
1	4	10	3	1	1	4	8	0	8
10	5	12	8	3	12	10	9	8	0

Flow Matrix

0	20	0	2	11	9	18	23	7	21
20	0	29	14	5	23	0	5	0	16
0	29	0	29	29	6	19	25	7	5
2	14	29	0	20	16	6	12	19	4
11	5	29	20	0	10	23	13	19	29
9	23	6	16	10	0	2	26	12	21
18	0	19	6	23	2	0	15	24	7
23	5	25	12	13	26	15	0	7	5
7	0	7	19	19	12	24	7	0	2
21	16	5	4	29	21	7	5	2	0

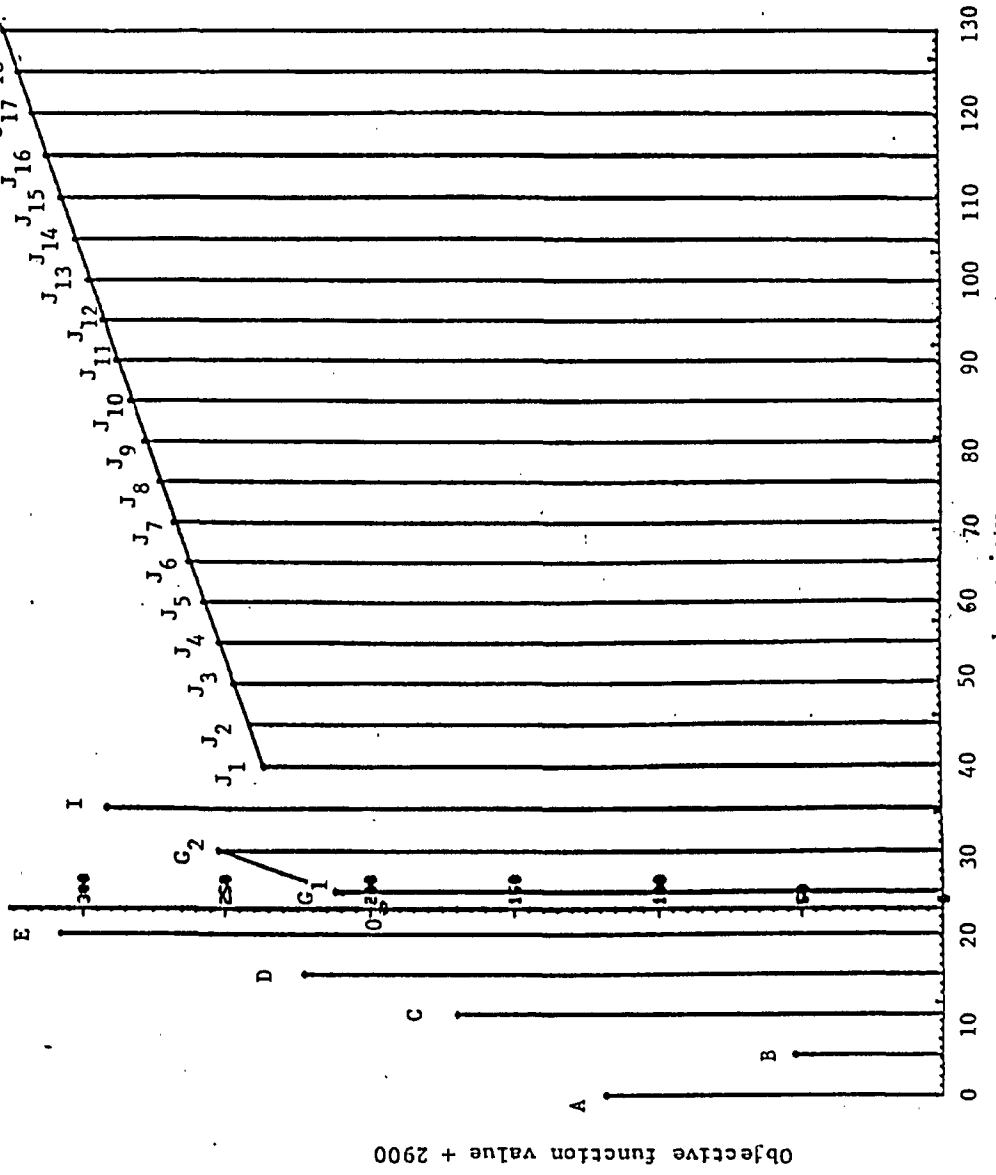


Fig. 7.3 Effect of varying parameter w_{18} on layout when reapplying the algorithm.

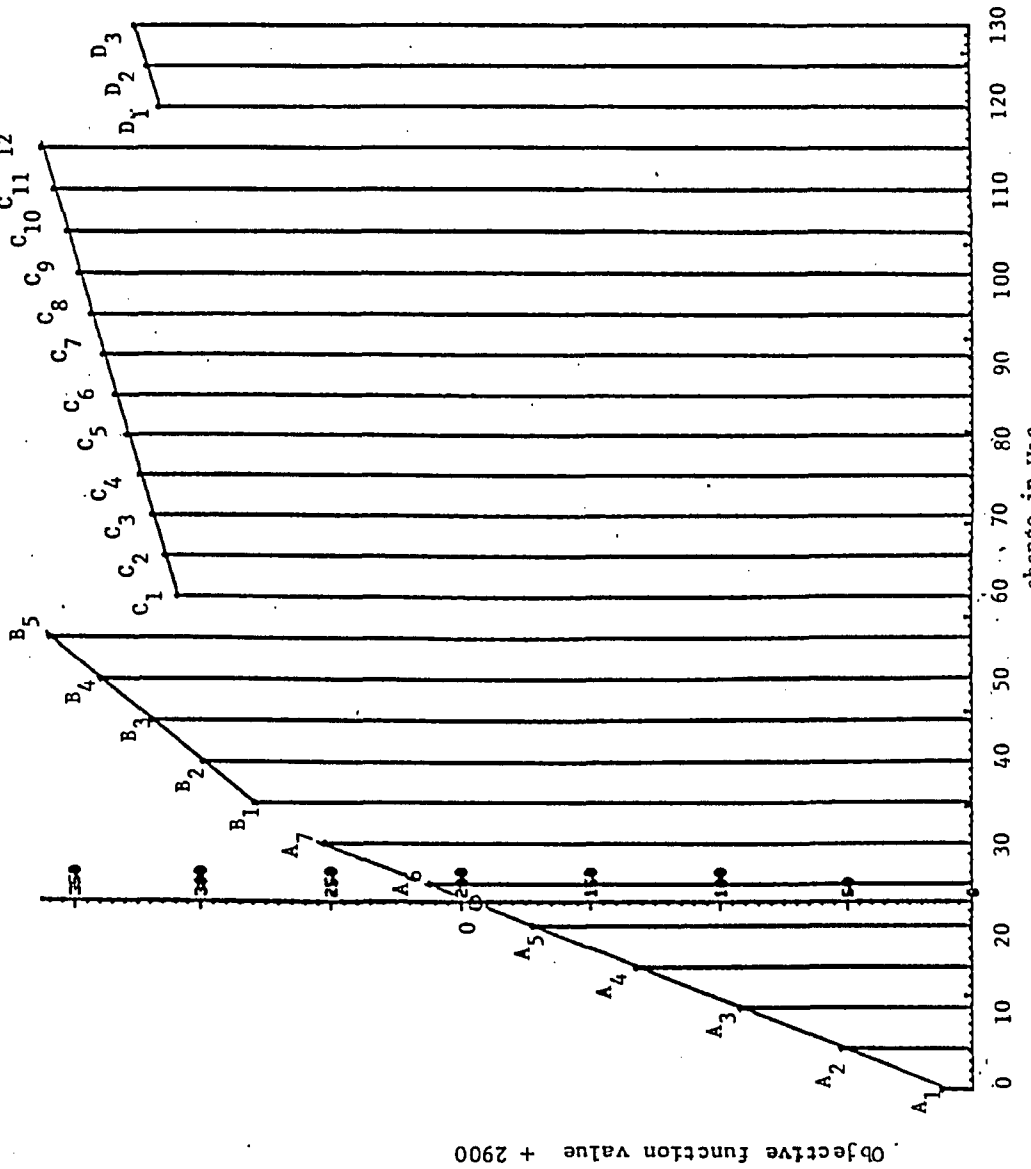


Fig. 7.4 Effect of varying parameter w_{18} on layout when using the previous information.

Table 7.9 Effect of Varying the Parameter w_{18} in a Problem of Size 10x10

Current Value of w_{18}*	Percentage change in w_{18}	[A] Reapplying the Algorithm		[B] Using Previous Information	
		Layout	Cost	Layout	Cost
0	-100	8,10,3,5,2,9,4,7,6,1	3018	8,7,10,5,4,2,6,1,3,9	2912
5	- 78.3	8,7,10,5,4,2,6,1,3,9	2952	"	2952
10	- 56.5	10,9,4,5,8,2,3,1,6,7	3070	"	2992
15	- 34.8	8,4,10,5,7,2,1,6,3,9	3123	"	3032
20	- 13.0	8,2,10,6,4,7,5,1,3,9	3208	"	3072
25	8.7	8,7,10,5,4,2,6,1,3,9	3112	"	3112
30	30.4	"	3152	"	3152
35	52.2	4,7,1,8,5,2,10,6,3,9	3191	2,7,10,5,4,8,6,1,3,9	3179
40	73.9	2,7,1,5,4,6,8,10,3,9	3136	"	3199
45	95.7	"	3141	"	3219
50	117.4	"	3146	"	3239
55	139.1	"	3151	"	3259
60	160.9	"	3156	2,4,10,5,7,8,1,6,3,9	3208
65	182.6	"	3161	"	3213
70	204.3	"	3166	"	3218

* original value of w_{18} = 23

Table 7.9 Continued ...

Current Value of w_{18}	Percentage Change in w_{18}	A Reapplying the Algorithm		B Using Previous Information	
		Layout	Cost	Layout	Cost
	226.1	2,7,1,5,4,6,8,10,3,9	3171	2,4,10,5,7,8,1,6,3,9	3223
75	246.8	"	3176	"	3228
80	269.6	"	3181	"	3233
85	291.3	"	3186	"	3238
90	313.0	"	3191	"	3243
95	334.8	"	3196	"	3248
100	356.5	"	3201	"	3253
105	378.3	"	3206	"	3258
110	400.0	"	3211	"	3263
115	421.7	"	3216	2,7,1,5,4,6,8,10,3,9	3216
120	443.5	"	3221	"	3221
125	465.2	"	3226	"	3226
..	...	"	..	"	..
..	...	"	..	"	..
..	...	"	..	"	..
205	791.3	"	3301	"	3301

Final cost of the original layout = 3096

Original layout FAC. 8 7 10 5 4 2 6 1 3 9
Site 1 2 3 4 5 6 7 8 9 10

problem solving process, the results obtained are summarized in Figure 7.4. When the parameter w_{18} is changed from 0 to 30, the layout remains the same. This is repeated by a line joining points A_1 to A_7. There is a change in the layout when w_{18} is changed to 35. This layout is represented by B_1. This layout then remains the same until the parameter w_{18} reaches a value of 60, at which point the layout changes, and the objective function value decreases. This is represented by point C_1. The next change in the layout occurs at point D_1 which corresponds to a w_{18} value of 115.

Similar results are obtained when a parameter in the distance matrix is changed. Figures 7.5 and 7.6 and Table 7.10, summarize the results of a change in parameter d_{18} in the distance matrix. From these graphs and the Table, it is obvious that the nature of the results are similar to those discussed above and can therefore, be interpretted likewise.

As a further example, a problem of size 20x20 is considered, in which the spread between the parameter values of the flow and distance matrices is large. The problem data are given in Table 7.11. It is assumed that parameter d_{12} varies. The results, which are depicted in Figures 7.7 and 7.8 and summarized in Table 7.12, are similar to those for the problem of size 10x10 and may, therefore, be interpreted similarly.

As the above results indicate, no obvious pattern governing the changes in the layout or in the objective function value is observed.

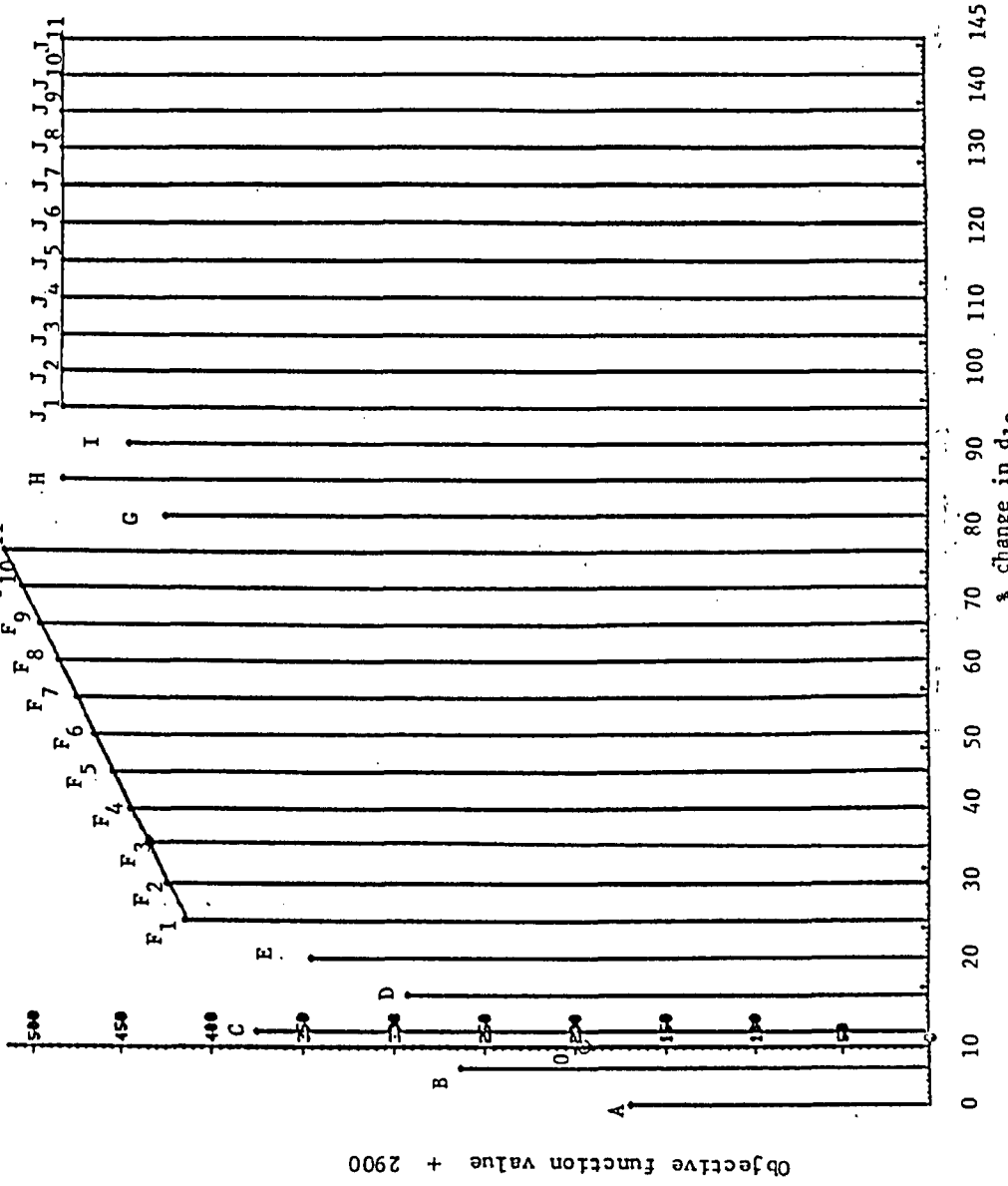


Fig. 7.5 Effect of varying parameter at d_{18} on layout when reapplying the algorithm.

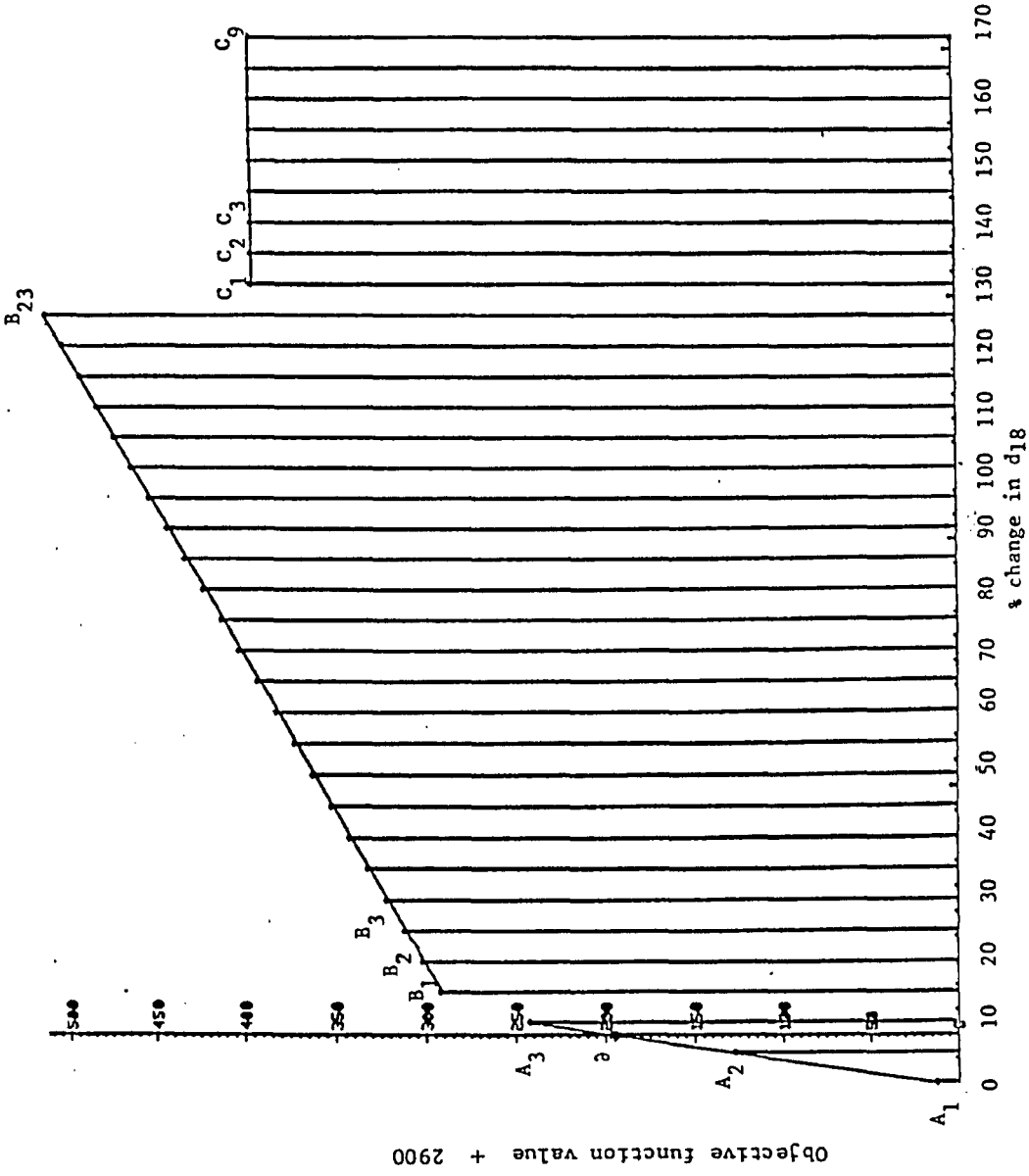


Fig. 7.6 Effect of varying the parameter d_{18} on layout when using previous information

Table 7.10 Effect of Varying the Parameter d_{18} in a Problem of Size 10x10

Current Value of d_{18}*	% Change in d_{18}	[A] Reapplying Algorithm		[B] Using Previous Information	
		Layout	Cost	Layout	Cost
0	-100.0	5,7,1,8,4,2,6,10,3,9	3069	8,7,10,5,4,2,6,1,3,9	2912
5	- 37.5	8,2,10,6,4,7,5,1,3,9	3163	"	3027
10	25.0	4,10,9,5,2,8,6,7,3,1	3275	"	3142
15	87.5	4,7,10,5,8,2,6,1,3,9	3192	4,7,10,5,8,2,6,1,3,9	3192
20	150.0	8,7,1,5,4,2,6,10,3,9	3245	"	3202
25	212.5	4,5,9,7,2,8,6,1,3,10	3314	"	3212
30	275.0	"	3324	"	3222
35	337.5	"	3334	"	3232
40	400.0	"	3344	"	3242
..			..		..
..			..		..
..			..		..
75	837.5	"	3414	"	3312

* original value of d_{18} = 8

Table 7.10 Continued...

Current Value of d_{18}	% Change in d_{18}	[A] Reapplying Algorithm		[B] Using Previous Information	
		Layout	Cost	Layout	Cost
80	900.0	4,7,10,5,8,2,6,1,3,9	3322	4,7,10,5,8,2,6,1,3,9	3322
85	962.5	3,6,4,8,10,7,9,1,5,4	3380	"	3332
90	1025.0	4,7,10,5,8,2,6,1,3,9	3342	"	3342
95	1087.5	3,6,4,8,10,7,9,1,5,4	3380	"	3352
100	1150.0	"	3380	"	3362
..			..		..
..			..		..
125	1462.5	"	3380	3,5,10,8,9,2,6,1,4,7	3412
130	1525.0	"	3380	"	3294
135	1587.5	"	3380	"	3294
..			..		..
..			..		..
205	2462.5	"	3380	"	3294

Final cost of the original layout = 3096
Original layout FAC 8 7 10 5 4 2 6 1 3 9
Site 1 2 3 4 5 6 7 8 9 10

Table 7.11 Data For Sample Problem Of Size 20x20 For Sensitivity Analysis

Distance Matrix

0	76	58	61	42	5	52	61	97	30	5	56	90	29	65	26	71	94	22	83
76	0	97	27	93	11	23	39	23	83	88	77	65	97	93	83	61	15	41	6
58	97	0	67	43	60	66	57	44	49	98	49	5	87	78	81	79	43	45	83
61	27	67	0	94	15	36	81	65	55	43	65	97	97	7	64	19	38	52	68
42	93	43	94	0	35	2	90	22	22	32	90	49	81	45	38	16	55	81	89
5	11	60	15	35	0	5	23	91	37	4	89	1	96	74	75	85	31	18	26
52	23	66	36	2	5	0	93	23	92	48	56	1	99	86	21	48	99	57	50
61	39	57	81	90	23	93	0	88	74	46	15	71	89	88	28	73	82	32	50
97	23	44	65	22	91	23	88	0	11	20	14	5	2	61	52	54	59	65	55
30	83	49	55	22	37	92	74	11	0	43	51	74	92	84	80	20	95	95	6
5	88	98	43	32	4	48	46	20	43	0	85	50	36	64	53	46	92	18	96
56	77	49	65	90	89	56	15	14	61	85	0	30	16	27	18	59	94	27	21
90	65	5	97	49	1	1	71	5	74	50	30	0	74	55	66	98	93	72	94
29	97	87	97	81	96	99	89	2	92	36	16	74	0	18	57	79	55	20	24
65	93	78	7	45	74	86	88	61	84	64	27	55	18	0	58	32	70	26	22
26	83	81	64	38	75	21	28	52	80	53	18	66	57	58	0	1	5	13	34
71	61	79	19	16	85	48	73	54	20	46	59	98	79	32	1	0	87	11	83
94	15	43	38	55	31	99	82	59	95	92	94	93	55	70	5	87	0	99	45
22	41	45	52	81	18	57	32	65	95	38	27	72	20	26	13	11	99	0	70
83	6	83	68	89	26	50	50	55	6	96	21	94	24	22	34	83	45	70	0

Table 7.11 Continued ...

Flow Matrix

0	671	23	95	361	307	586	754	249	701	964	475	173	761.	5	179	23	527	951	963
671	0	217	629	819	256	162	665	530	194	394	616	150	354	768	425	630	960	94	860
23	217	0	406	694	508	803	242	228	187	64	700	617	405	872	585	657	676	144	774
95	629	406	0	352	140	673	778	607	638	362	422	274	845	604	18	664	826	974	407
361	819	694	352	0	678	403	312	243	648	700	366	889	41	242	82	312	132	981	696
307	256	508	140	678	0	342	786	634	730	672	458	693	36	979	551	492	993	531	249
586	162	803	673	403	342	0	708	8	671	951	663	417	531	431	802	932	373	849	732
754	665	242	778	312	786	708	0	751	918	742	190	459	42	115	312	837	215	753	582
249	530	228	607	243	634	8	751	0	717	59	896	844	998	394	379	727	949	154	381
701	194	187	638	648	730	671	918	717	0	893	932	547	897	456	665	827	65	949	100
964	394	64	362	700	672	951	742	59	893	0	61	464	235	231	266	518	714	619	284
475	616	700	422	366	458	663	190	896	932	61	0	127	210	110	774	646	912	651	701
173	150	617	274	889	693	417	459	844	547	464	127	0	344	750	407	687	460	573	294
761	354	405	845	41	36	531	42	998	897	235	210	344	0	607	994	500	51	800	340
5	768	872	604	242	979	431	115	394	456	231	110	750	607	0	840	972	275	901	924
179	425	585	18	82	551	802	312	379	655	266	774	407	994	840	0	433	284	803	259
23	630	657	664	312	492	932	837	727	827	518	646	687	500	972	433	0	328	632	840
527	960	676	826	132	993	373	215	949	65	714	912	460	51	275	284	328	0	350	545
951	84	144	974	981	531	849	753	154	949	619	651	573	800	901	803	632	350	0	111
963	860	774	407	696	249	732	582	381	100	284	701	294	340	924	259	840	545	111	0

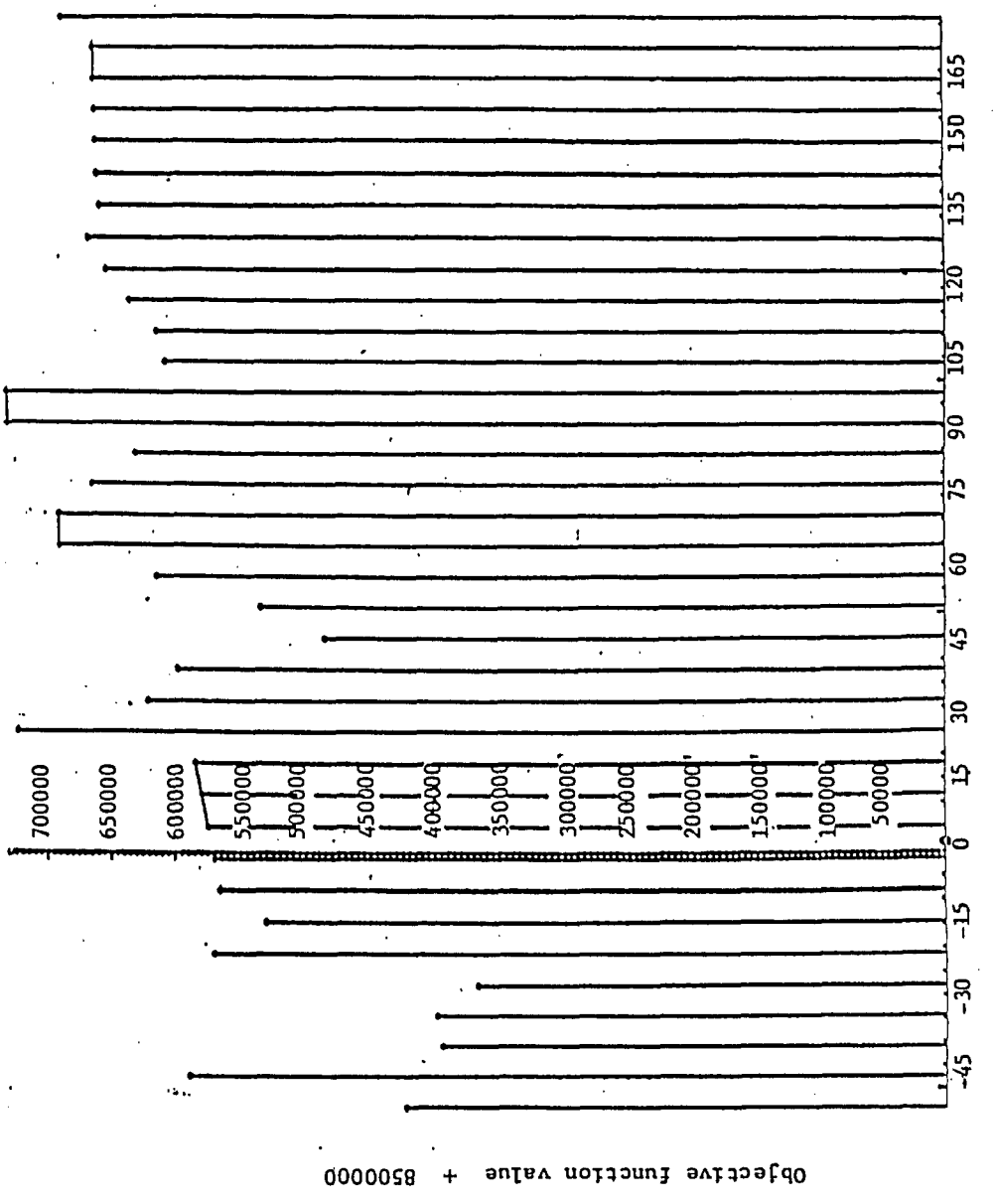


Fig. 7.7 Effect of Varying the Parameter d_{12} on Layout when Reapplying the Algorithm in a Problem of Size 20x20

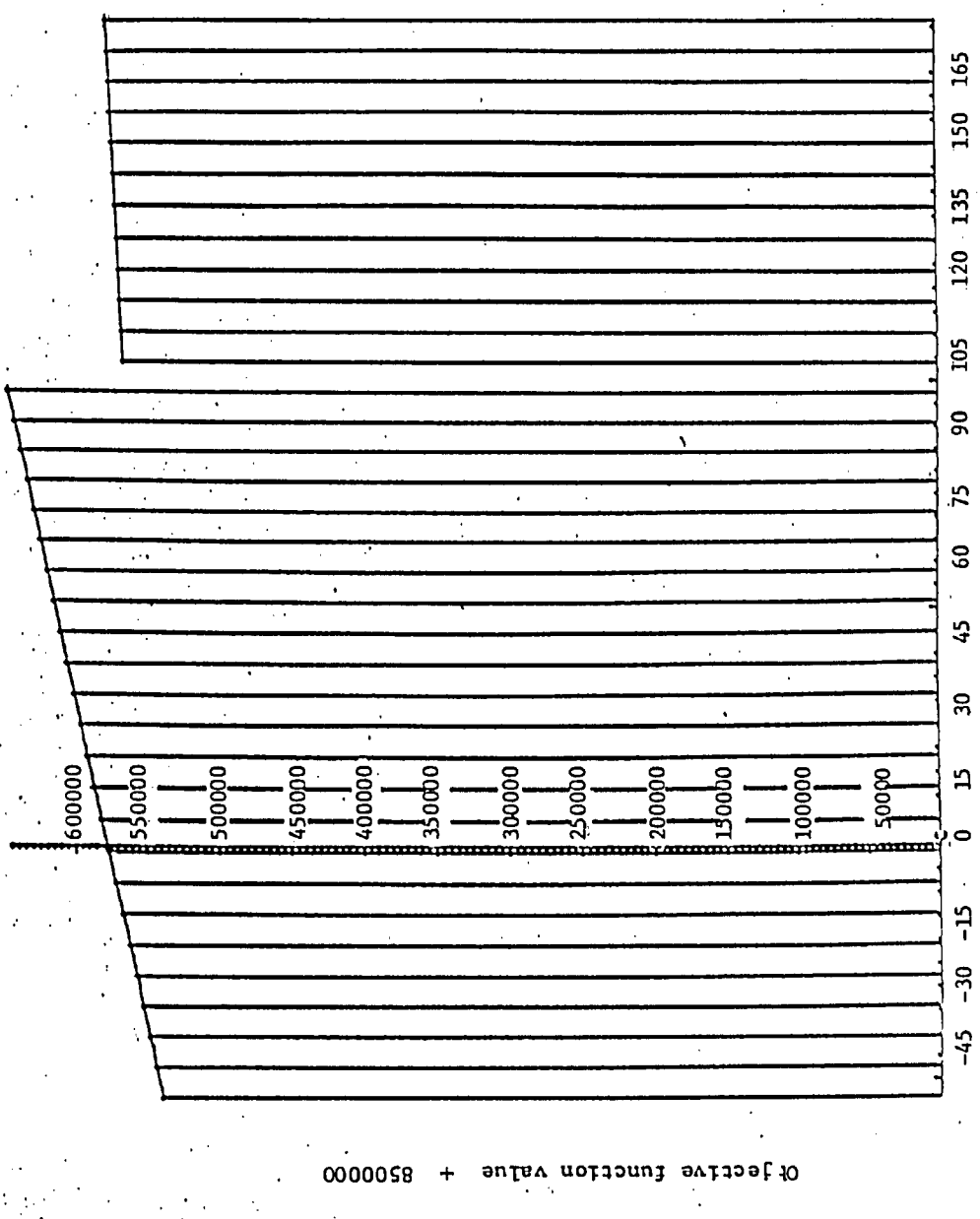


Fig. 7.8 Effect of Varying the Parameter d_{12} on layout When Using the Previous Information on a Problem of Size 20x20

Table 7.12 Effect of Varying the Parameter d_{12} in a Problem of Size 20x20

Current Value of d_{12}	Percentage Change in d_{12}*	[A] Reapplying Algorithm		[B] Using Previous Information	
		Layout	Cost	Layout	Cost
0	-100	5,6,16,11,19,7,10,3,9,13, 4,12,14,2,1,17,8,15,20,18	9034436	8,13,1,5,18,9,6,14,20,2,4,16,12,17 19,7,11,3,10,15	9007744
5	- 93.4	12,17,16,8,19,9,10,2,15,18,4, 20,14,3,1,5,11,13,2,6	9015808	8,13,6,5,18,9,6,14,20,2,1,17,12,4, 19,7,11,3,10,15	9005602
10	- 86.8	7,16,4,6,11,19,10,14,2,5,1,9,8, 20,18,7,13,15,12,3	8928824	8,13,1,5,18,9,6,14,20,2,4,16,12, 17,19,7,11,3,10,15	9016924
15	- 80.2	"	8936844	"	9021514
20	- 73.7	11,16,4,3,20,19,7,14,2,5,1,9,8 18,6,17,12,15,10,13	9004814	"	9026104
25	- 67.1	10,15,18,13,12,17,16,4,20,9,8,11,2, 1,5,7,6,3,19,14	8875264	"	9030694
30	- 60.5	10,15,18,17,12,16,7,6,20,9,14,11,3, 1,5,8,4,2,19,13	9008672	12,13,4,5,18,9,6,14,20,2,1,17,8,16, 19,7,11,3,10,15	9035284

* original value of d_{12} = 76
Final cost of the original layout = 9070780
Original layout = FAC. 8 13 16 5 18 9 6 14 20 2 1 17 12 4 19 17 11 3 10 15
Site 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Table 7.12 Continued ...

Current Value of d_{12}	Percentage Change in d_{12}	[A]		[B]	
		Layout	Cost	Layout	Cost
35	- 53.9	10,15,18,13,12,16,17,11,20,9,14,8,2, 1,5,7,4,3,19,6	8921732	8,13,1,5,18,9,6,14,20,2,4,16,12 19,7,11,3,10,15	9039874
40	- 47.3	10,15,12,13,2,6,3,16,20,8,5,7,18, 1,11,14,4,9,19,17	9088724	"	9044464
45	- 40.7	10,15,18,13,12,17,16,4,20,9,8,11,2, 1,5,7,6,3,19,14	8893504	"	9049054
50	- 34.2	10,15,18,13,12,17,16,4,20,9,8,11,2, 1,5,7,11,3,10,20	8898064	"	9053644
55	- 27.6	19,15,14,6,4,13,17,16,2,5,8,12,9, 1,18,7,11,3,10,20	8865768	"	9058234
60	- 21.0	11,16,13,14,18,19,2,7,20,6,1,17,5 8,4,12,9,3,10,15	9069464	"	9062824
65	- 14.4	11,7,13,14,18,19,4,10,3,6,1,17,5, 20,2,12,9,16,8,15	9030158	"	9067414
70	- 7.8	8,13,16,5,18,9,6,14,20,2,1,17,12 4,19,7,11,3,10,15	9065272	"	9072004

Table 7.12 Continued ...

Current Value of d_{12}	Percentage Change in d_{12}*	[A]		[B]	
		Layout	Cost	Layout	Cost
75	- 1.3	8,13,16,5,18,9,6,14,20,2,1,17,12, 4,19,7,11,3,10,15	9069862	8,13,15,18,9,6,14,20,2,4,16,12, 17,19,7,11,3,10,15	9076594
80	5.2	8,13,16,5,18,9,6,14,20,2,1,17,12, 4,19,7,11,3,10,15	9074452	"	9081184
85	11.8	"	9079042	"	9085774
90	18.4	"	9083632	"	9090364
95	25.0	11,9,15,14,5,18,13,16,20,2,1,17,6, 8,10,7,4,3,19,12	9222334	"	9094954
100	31.5	15,16,13,3,11,6,7,9,19,1,8,10,5,4, 2,17,20,15,12,14	9119002	"	9099544
105	38.1	12,13,16,5,17,6,3,9,20,2,18,14,15, 1,19,7,4,11,10,8	9095820	"	9104134
110	44.7	19,18,5,4,16,9,12,13,20,15,14,11,2 1,8,7,17,3,10,6	8983310	"	9108724
115	51.3	14,2,3,17,13,9,4,12,11,6,10,7,18,1 20,19,15,5,16,8	9031924	"	9113314

Table 7.12 Continued ...

Current Value of d_{12}	Percentage Change in d_{12}	[A]		[B]	
		Layout	Cost	Layout	Cost
120	57.8	6,2,16,18,15,9,4,13,7,3,17,19, 14,11,1,10,20,12,8,5	9110838	8,13,1,5,18,9,6,14,20,2,4,16,12, 17,19,7,11,3,10,15	9117904
125	64.4	10,18,13,5,14,6,16,17,2,9,8,20, 15,1,19,7,4,3,11,12	9187316	"	9122494
130	71.0	"	9187966	"	9127084
135	77.6	14,18,15,6,13,9,17,4,20,16,8,19, 2,1,5,7,11,3,10,12	9161846	"	9131674
140	84.2	14,2,20,12,13,9,4,1,17,3,10,7,8, 11,6,19,15,5,16,18	9127452	"	9136264
145	90.7	15,8,16,10,19,17,11,3,1,14,20,13, 7,2,5,18,4,12,6,9	9229486	"	9140854
150	97.3	"	9230636	"	9146555
155	103.9	16,5,1,3,19,6,8,15,10,14,7,2,11, 12,18,17,4,13,20,9	9104422	12,13,4,5,18,9,6,14,20,2,1,17,8, 16,19,7,11,3,10,15	9062372
160	110.5	9,11,20,19,2,18,5,6,7,12,4,16,3,14, 15,8,17,1,13,10	9110886	"	9063642

"

Table 7.12 Continued ...

Current Value of d_{12}	Percentage Change in d_{12}	[A]		[B]	
		Layout	Cost	Layout	Cost
165	117.1	13,11,3,2,14,18,4,6,10,16,9,5,12, 8,20,19,15,1,17,7	9132190	12,13,4,5,18,9,6,14,20,2,1,17, 8,16,19,7,11,3,10,15	9064912
170	123.6	9,5,16,6,14,18,2,3,19,13,4,7,12, 8,10,20,15,1,17,11	9150756	"	9066182
175	130.2	12,5,16,6,4,9,2,3,19,10,18,7,14, 1,11,17,15,13,20,8	9164300	"	9067452
180	136.8	9,5,16,6,14,18,2,3,19,13,4,7,12, 8,10,20,15,1,17,11	9155616	"	9068722
185	143.4	12,13,19,6,2,18,9,5,14,16,3,10,4, 7,8,20,15,1,17,11	9157598	"	9069992
190	150.0	"	9158868	""	9071262
195	156.5	12,13,19,6,2,18,9,5,14,16,3,10,4, 7,8,20,15,1,17,11	9160138	"	9072532
200	163.1	"	9161408	"	9073802
205	169.7	"	9162678	"	9075072
210	176.3	15,8,16,11,14,19,10,7,9,13,6,3, 12,18,2,20,17,1,4,5	9187464	"	9076342

The changes are dependent on the structure of the problem data. However, in most cases, when sensitivity analysis is carried out using previous information, it is observed that the value of objective function increases but as the layout changes, the rate of increase in the objective function value decreases. It is interesting to note that the average deviation in the objective function value obtained by the two methods (that is, reapplying the algorithm, and using the previous information) is within $\pm$ 5%. This suggests that the effects of varying the parameters in the distance or flow matrix can be studied in a much more efficient manner and without significant loss in the quality of the solution, if information available in the last iteration of the problem solving process is utilized. However the layouts would be different in each case.

7.4 Effect of Using Pairwise Interchange Procedure in Conjunction With the Algorithm

In preceding sections, the algorithm was applied independently to various problems given in the literature as well as some randomly generated problems. It was observed that the algorithm produced high quality solutions for almost all the problems in a reasonable amount of computer time. Because of the computational efficiency of the algorithm, it was felt that if extra computer time is available, other heuristic procedures could be used in conjunction with the algorithm to further improve the quality of the solution. In this study, the

investigation is carried out to study the effect of pairwise interchange procedure at various stages of the algorithm.

In general, there are $N.n^2+N$ assignments which are generated by the algorithm, N being the number of iterations. Some of these assignments could be generated several times at various stages of the algorithm; therefore, the actual number of new assignments generated by the algorithm would be less than $N.n^2+N$. No attempt was made to sort these assignments, and the main emphasis was on improving the quality of the solution using some or all the assignments generated. Therefore, the following alternative investigations were carried out:

1) Application of pairwise interchange procedure to the final solution produced by the algorithm, A(1).
2) Application of pairwise interchange procedure to the intermediate solutions generated at the end of each iteration, A(N).
3) Application of pairwise interchange procedure to the n^2 assignments generated in the first iteration of the algorithm, $A(n^2)$; the algorithm is terminated at the end of the first iteration.
4) Application of pairwise interchange procedure to the $N.n^2$ assignments generated during the N iterations of the problem solving process, $A(N.n^2)$.

It is obvious that the steps listed above are in the order of decreasing efficiency. The comparative study of the results, from the view point of quality and efficiency of the solution, is given in Tables 7.14 and 7.15. It is seen that the quality of the solution can be improved and the best known solutions are obtainable in almost all cases.

It is therefore concluded that high quality solutions can be achieved by using other heuristics in conjunction with the proposed algorithm.

Table 7.13 Effect of using pairwise interchange procedure on the quality of the solution (objective function value in unit costs).

Problem Size, n	A(1)	A(N)	$A(N^2)$	$A(N.n^2)$	Pseudo Random Cost	Best Known Objective Function Value
5	26	26	25	25	26	25
6	43	43	43	43	43	43
7	76	76	74	74	76	74
8	118	107	107	107	118	107
12	296	296	289	289	296	289
15	585	584	576	576	585	575
20	1320	1320	1301	1285	1320	1285
30	3088	3105	3105	3077	3088	3077

Table 7.14 Average Increase in Computation Time Per Iteration (secs) When Using Pairwise Interchange Procedure

Problem Size, n	Number of Iterations, N	Time*/ Iteration	Average Increase in Computation time/iteration			
			A(1)**	A(N)	$A(n^2)$	$A(N.n^2)$
5	2	.026	.003	.003	.01	.01
6	2	.04	.001	..005	.06	.05
7	3	.06	.003	.005	.16	.075
8	5	.10	.003	.009	.33	.15
12	4	.36	.009	.020	1.65	.94
15	8	.77	.019	.067	6.94	4.19
20	8	2.11	.093	.151	25.68	13.79
30	8	1.523	1.523	1.799	291.00	141.63

* These times are for Pseudo-Random Criterion (see Table 7.2)

** Times listed are per solution (only one solution is considered for improvement)

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

This research was carried out from the point of view of developing a heuristic procedure for solving the QAP, which could handle large size problems efficiently and produce high quality solutions. The algorithm developed in this study is an iterative scheme, which removes the discrepancies in the cost of locating facilities at different sites in a number of iterations. Unlike the improvement procedures, the proposed algorithm is independent of the starting solution and converges in a finite number of iterations.

The algorithm was applied to various test problems given in the literature. The superiority of the algorithm was demonstrated by comparing the results with those of various heuristics given in the literature. It was concluded that the proposed algorithm produces high quality solutions in reasonable amounts of computational time. The applicability of the algorithm for large size problems was demonstrated by solving randomly generated problems ranging in size from 30 to 60. It was shown that the rate of improvement in the value of objective function is very rapid in the first few iterations, but it slows down in subsequent iterations. The study of the performance of the algorithm concerning the quality of the solution showed that, for small size problems, the results are optimal approximately 65% of the time. These results may not be extended to large size problems due to unavailability of optimal

solutions. But the experience with problems given in the literature indicates that high quality solutions are obtainable by the proposed algorithm.

Different methods of removing the discrepancies in the costs of locating facilities at different sites were used, and their effect on the quality of the solution was investigated. However, the applicability of these methods is dependent on the availability of extra computational time.

In order to investigate the possibility of improving the solution quality by using other heuristics in conjunction with the algorithm, the pairwise interchange procedure was employed to improve on the solutions generated by the algorithm in a number of iterations. The results seem to be encouraging as better solutions are obtained in some cases. However, a criterion must be developed to investigate only selected solutions generated by the algorithm which would lead to the most improvement in the quality of the solution.

It was further noted that the algorithm solves several linear assignment problems in order to improve the elements in the lower bound matrix. The efficiency of the algorithm is therefore dependent on the efficiency of the computer code used for solving the linear assignment problem. It is therefore suggested that the development of efficient codes for solving linear assignment problems be considered.

APPENDICES

APPENDIX I

"A METHOD FOR FINDING SEVERAL SOLUTIONS TO THE ASSIGNMENT PROBLEM SIMULTANEOUSLY"

The paper originally appeared in

Angewandte Informatki, Vol. 13, No. 9,

p. 404-414, 1971. (Germany)

A METHOD FOR FINDING SEVERAL SOLUTIONS TO THE ASSIGNMENT PROBLEM SIMULTANEOUSLY

This paper describes an efficient method for finding several solutions for the assignment problem simultaneously. Given a problem of size nxn the algorithm finds all solutions which have one of the following allocations (K,1), (K,2),...(K,n).

Because of the obvious difficulties in understanding the paper and its applicability, the important sections of the paper were translated into English. For the east of reference these sections are summarized below:

1. Problem Statement:

Consider the problem of Linear Assignment Problem, which is defined as the minimization of the function:

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} X_{ij} \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} X_{ij} = 1, \ \forall \ i \epsilon M, \ M = \{1,2,\ldots,n\} \tag{2}$$

$$\sum_{i=1}^{n} X_{ij} = 1, \ \forall \ j \epsilon N, \ N = \{1,2,\ldots,n\} \tag{3}$$

$$X_{ij} \in (0,1) \quad \forall\, i \in M,\ j \in N \tag{4}$$

The problem is to determine all solutions which have one of the following allocations (K,1), (K,2), ..., (K,n).

2. Method of Solution:

Let $x^{(0)} = (x_{ij}^{(0)})$ be the optimal solution of the assignment problem given by relations (1) to (4) and let $L^{(0)}$ be the optimum value of objective function. If L_{ij} is the minimum value of the objective function, assuming that the assignment (i,j) is in the solution, ΔL_{ij}, the difference, would then be

$$\Delta L_{ij} = L_{ij} - L^{(0)} \tag{5}$$

It is easy to see that $\Delta L_{ij} \geq 0$. The basis of an assignment problem is defined by the variables for which $x_{ij} = 1$. The method for finding simultaneous solutions which contain one of the assignments (K,1),(K,2),...,(K,n) $K \in M$, is based upon the reduced coefficient matrix. Let a matrix $D^{(0)} = (d_{ij}^{(0)})$ describe a reduced coefficient matrix of an assignment problem. Then for the elements of this matrix $d_{ij}^{(0)}$, the following is true:

$$\begin{aligned} &d_{ij}^{(0)} \geq 0 \quad \forall\, i \in M, \quad \forall\, j \in N \\ &d_{ij}^{(0)} = 0 \quad \text{if } x_{ij}^{(0)} = 1 \end{aligned} \tag{6}$$

The Hungarian method and the Marking algorithms of FORD and FULKERSON [1] are systematic methods for developing the reduced coefficient matrices with the property formulated in (6). In general, there are always several matrices with the property of $D^{(0)}$ for every assignment problem of the type in (1) to (4).

In particular, for every assignment problem there is a reduced coefficient matrix $D^{(K)}$ with the property:

$$d_{ij}^{(K)} \geq 0 \quad \forall\ i\epsilon M, \quad \forall\ j\epsilon N, \tag{7}$$

$$d_{ij}^{(K)} = 0, \text{ if } x_{ij}^{(0)} = 1 \tag{8}$$

$$d_{Kj}^{(K)} = \Delta L_{Kj} \quad K\epsilon M, \quad \forall\ j\epsilon N \tag{9}$$

From (9) and (5) it follows immediately that $L_{Kj} - L^{(0)} = d_{Kj}^{(K)}$ $K\epsilon M$, $\forall\ j\epsilon N$. The coefficients $d_{Kj}^{(K)}$ of row K in the reduced matrix $D^{(K)}$, thus specify the amount by which the minimum value of the objective function of an assignment problem, is greater than the value $L^{(0)}$, when assignment (K,j) is in the solution. The method of achieving a reduced matrix $D^{(K)}$ with the properties of (7), (8) and (9) is described below:

it can be shown that the solution of an assignment problem will remain unchanged if constants are added along row(s) or column(s) to the coefficient matrix $D^{(0)}$. (Refer to [4],p.12). The addition of constants is carried out in such a way that the properties (6)

$$d_{ij}^{(0)} \geq 0 \quad \forall\, i \in M, \quad \forall\, j \in N,$$

and

$$d_{ij}^{(0)} = 0, \text{ if } x_{ij}^{(0)} = 1$$

of the matrix $D^{(0)}$ are preserved. Let M_0 be constant such that

$$M_0 >> 0. \tag{10}$$

We add M_0 to all elements $d_{Kj}^{(0)}$, $\forall\, j \in N$, a fixed row K of the matrix $D^{(0)}$. In order to preserve property (6), M_0 must be subtracted from all elements of the column ℓ of the matrix $D^{(0)}$. The index ℓ is determined by the fact that in the optimal solution of the assignment problem the variable $x_{K\ell}$ has the value $x_{K\ell}^{(0)} = 1$. Because of (6), $d_{K\ell}^{(0)} = 0$. The result of these operations are retained in a matrix D^1:

$$d_{ij}^1 = \begin{cases} d_{ij}^{(0)}, \text{ for } i = K, \ j = \ell \\ d_{ij}^{(0)} + M_0, \text{ for } i = K, \ j \neq \ell \\ d_{ij}^{(0)} - M_0, \text{ for } j = \ell, \ i \neq K \\ d_{ij}^{(0)} \quad \text{otherwise} \end{cases} \tag{11}$$

Because of (6) it is in particular true that

$$d_{K\ell}^1 = d_{K\ell}^{(0)} = 0$$

Certainly the matrix D^1 has negative elements in column ℓ, since

$M_0 >> 0$. We determine the constants

$$M_1 = \max_{\substack{i\in M \\ j\in N}} \{-d^1_{ij} \mid d^1_{ij} < 0\} = -d^1_{rs} \tag{12}$$

which we add to all elements of row r of D^1, and subtract from all elements of column t of D^1. The index t is again determined by the fact that in the optimal solution of the assignment problem the variable x_{rt} has the value $x^{(0)}_{rt} = 1$.

From D^1 can be achieved a matrix D^2 with elements

$$d^2_{ij} = \begin{cases} d^{(0)}_{ij} & \text{for } i = r,\ j = t, \\ d^1_{ij} + M_1 & \text{for } i = r,\ j \neq t \\ d^1_{ij} - M_1 & \text{for } i \neq r,\ j = t \\ d^1_{ij} & \text{otherwise} \end{cases} \tag{13}$$

In general the matrix D^{q+1} is derived from the D^q, which precedes it, as shown. If

$$M_q = \max_{\substack{i\in M \\ j\in N}} \{-d^q_{ij} \mid d^q_{ij} < 0\} = -d^q_{uw} \tag{14}$$

and $x^{(0)}_{uv} = 1$, (15)

then for the elements of the matrix D^{q+1}

$$d_{ij}^{q+1} = \begin{cases} d_{ij}^{(0)} & \text{for } i = u,\ j = v, \\ d_{ij}^{q} + M_q & \text{for } i = u,\ j \neq v \\ d_{ij}^{q} - M_q & \text{for } i \neq u,\ j = v \\ d_{ij}^{q} & \text{otherwise} \end{cases} \tag{16}$$

After a total of n iterations we have the matrix D^n. We define

$$D^{(K)} = D^n, \tag{17}$$

because D^n proceeds out of $D^{(0)}$, since first a constant was added to the elements of row K of $D^{(0)}$.

It remains to be shown that the reduced coefficient matrix $D^{(K)}$ has the properties (7), (8), and (9) formulated above. It is easily seen that from (6) in relation to (11) to (14) it is always true that

$$M_0 \geq M_1 \geq \ldots \geq M_{n+1} \tag{18}$$

In general it is true for an element $d_{ij}^{(K)}$ that

$$d_{ij}^{(K)} = d_{ij}^{(0)} + M_\nu - M_{\nu+\mu}, \tag{19}$$

where $\nu \in \{0,1,..,n-1\}$ and $\mu \geq 0$, $\mu \in \{1,2,..,\nu+ n-1\}$.

From $d_{ij}^{(0)} \geq 0$ and $M_\nu - M_{\nu+\xi} \geq 0$ it follows that $d_{ij}^{(K)} \geq 0$.

In particular, it is true that

$$d_{ij}^{(K)} = 0, \text{ if } x_{ij}^{(0)} = 1, \tag{20}$$

which follows immediately from (11), (13), and (16).

Somewhat more difficult to prove is property (9), from which

$$d_{Kj}^{(K)} = \Delta L_{kj} \qquad K \epsilon M, \quad \forall j \epsilon N$$

First there would be a modified assignment problem to define, through the fact that instead of the variables $x_{K\ell}^{(0)}$ the variable x_{Kj} with $x_{Kj} = 1$ must be in the solution. ΔL_{Kj} then describes the amount which the optimal solution of the modified problem deviates from the optimal solution of the original problem. The optimal solution of the original assignment problem which was defined through the relations (1) to (4) would have the assignment (K,ℓ). Since from (20)

$$d_{K\ell}^{(K)} = 0,$$

then (9) is trivially true for $j = \ell$

Since

$$d_{ij}^{(K)} \geq 0 \qquad \forall\, i \epsilon M, \ \forall\, j \epsilon N$$

it is true that

$$\Delta L_{Kj} \geq d_{Kj}^{(K)}$$

Assume that

$$\Delta L_{Kj} > d_{Kj}^{(K)} \tag{21}$$

In the optimal solution of the modified problem there must be, besides the variables x_{Kj} with $d_{Kj}^{(K)} \geq 0$, at least a further basis

variable x_{uv} with $d_{uv}^{(K)} \geq 0$.

Without limiting generality it is in the following assumed that in the solution matrix of the modified problem only to the variables x_{Kj} and x_{uv} a positive amount $d_{Kj}^{(K)}$, $d_{uv}^{(K)} > 0$ is assigned. It is also true that

$$\Delta L_{Kj} = d_{Kj}^{(K)} + d_{uv}^{(K)} \tag{22}$$

We set

$$T_0 = d_{uv}^{(K)} \geq 0 \tag{23}$$

By a reordering of the matrix $D^{(K)}$ it can always be arranged that the reduced coefficients of the optimal assignment of the original assignment problem stand on the major diagonal. For example, the following arrangement can be effected:

Spalte / Zeile	l	j	t	. . .	v	w
k	[0]	$j_{kj}^{(k)}$				
u		[0]			$j_{uv}^{(k)}$	
r			[0]			
⋮				⋱		
s					[0]	
q						[0]

Fig. 1. Matrix $D^{(K)}$ after reordering of rows and columns.

In the above diagram the optimal solution of the original problem is marked with little boxes. The optimal solution of the modified problem is differentiated, among other elements, through the

assignments marked with circles. We add the constants

$$T_0 = d_{uv}^{(K)}$$

to all elements of row K and subtract T_0 from all elements of column ℓ. It is easy to verify that thus the solution of the original assignment is not affected. In the field (K,j) now stands the coefficient

$$d_{Kj}^{\ell} = d_{Kj}^{(K)} + d_{uv}^{(K)} = \Delta Z_{Kj}$$

In the field (u,ℓ) in particular there can be no negative value. If it were true that

$$d_{u\ell}^{\ell} = d_{u\ell}^{(K)} - T_0 < 0,$$

then the optimal solution of the modified problem would appear as follows:

instead of $x_{K\ell}(-)$ there would be $x_{KJ}(+)$ in the basis, instead of $x_{uj}(-)$ there would be $x_{u\ell}(+)$ in the basis, with $\Delta L_{Kj} < d_{Kj}^{(K)} + d_{uv}^{(K)}$, which would contradict the assumptions formulated above. From the elements of column was substracted the constant T_0. The smallest negative element of column ℓ is simultaneously the smallest element in the matrix. Let

$$T_1 = \max_{i\in M} \max_{\substack{i\in M\setminus\{K,u\} \\ p\in N}} \{-d_{ip}^{1} \mid d_{ip}^{1} < 0\} = -d_{r\ell}^{1} \qquad (24)$$

We add T_1 to the elements of row r and subtract T_1 from the elements of column t. Now

$$d^2_{r\ell} = d^{(K)}_{r\ell} - T_0 + T_1 = 0$$

Since $d^{(K)}_{ip} \geq 0 \quad \forall\ i\epsilon M, \quad \forall\ p\epsilon N$, it is true that $T_1 < T_0$;

it follows from this that

$$d^2_{Kt} = d^{(K)}_{Kt} + T_0 - T_1 \geq 0$$

Furthermore, $d^2_{ut} = d^{(K)}_{ut} - T_1 \geq 0$

If $d^2_{ut} < 0$ were true, then the assignment (u,v) could not be contained in the optimal solution of the modified problem, since the sequence $x_{Kj}(+)$, $x_{uj}(-)$, $x_{ut}(+)$, $x_{rt}(-)$, $x_{r\ell}(+)$, $x_{K\ell}(-)$ would be assigned a value $L_{Kj} < d^{(K)}_{Kj} + d^{(K)}_{uv}$

We determine that

$$T_2 = \max_{\substack{i\epsilon M\setminus\{K,u,r\} \\ p\epsilon N}} \{-d^2_{ip} \mid d^2_{ip} < 0\} = -d^2_{qt} \qquad (25)$$

By the respective addition or subtraction of T_2 to the elements of row q, or from the elements of column w, the original optimal solution is not affected. In particular the equation $d^3_{qt} = d^{(K)}_{qt} - T_1 + T_2 = 0$ is preserved.

Furthermore, the elements d^3_{Kw}, d^3_{rw}, and d^3_{qw} remain non-negative. Thus the element d^3_{uw} must also remain non-negative, since otherwise

there would occur a more conducive result to a solution for the basis of the modified problem, as is seen in Figure 2.

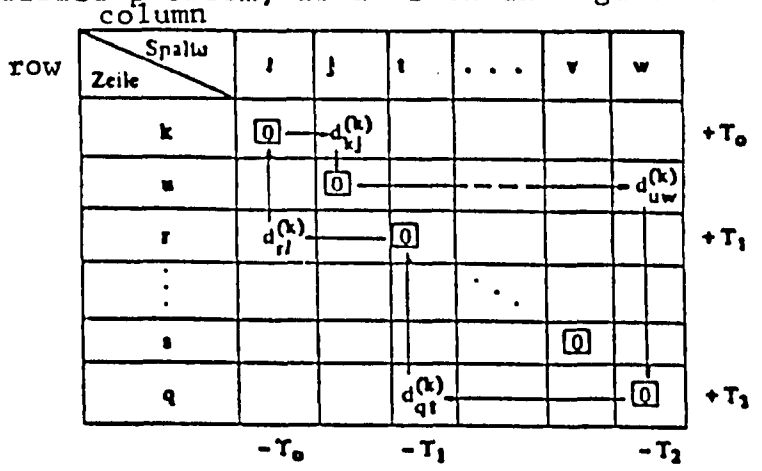


Fig. 2. Determining the path of exchanges

In general, it can be said that by the column-wise subtraction of constants

$$T_\mu = \max_{\substack{i\in\overline{M}\\ p\in\overline{N}}} \{ -d_{ip} \mid d_{ip} < 0 \} \tag{26}$$

in the described manner in row u a negative element can never appear, since otherwise there would immediately exist a better solution for the modified problem. The several indices $\overline{M}$ or $\overline{N}$ should contain those indices which are assigned to rows or columns with negative elements. It can easily be seen that in row to which T_μ is added, at least one distinguishable element will be exactly 0, and in fact this element can be found in one of the columns from which a constant T_σ, $\sigma \in \{0,1,..,\mu-1\}$ is subtracted. Since to every row or column at most one constant is added once, these distinguishable zero elements cannot be further

altered by subsequent operations. So from every element of row u, from which a constant T_σ is substracted, it is possible to arrive at a solution of the modified assignment problem through the distinguishable zero elements. Since by the above assumption, the optimal solution of the modified problem contains the assignment (u,v) all the coefficients d_{up}, $u\varepsilon M$, V $p\varepsilon N$, remain non-negative. That is, a constant can be subtracted from the elements of column j. In this manner we find a matrix $\overline{D}^{(K)}$ with $\overline{d}_{ip}^{(K)} \geq 0$ V $i\varepsilon M$, V $p\varepsilon N$

In particular,

$$\overline{d}_{Kj}^{(K)} = d_{KJ}^{(K)} + d_{uv}^{(K)} \qquad (27)$$

and

$$\overline{d}_{uv}^{(K)} = 0.$$

$\overline{D}^{(K)}$ could also have been determined by beginning with the development of the matrix $D^{(K)}$ from the matrix $D^{(0)}$ with the addition of a constant $M^1_{(0)} = M_0 + T_0$. Thus we have contradicted the above assumptions. By (10), $M_0 >> 0$, that is, in the solution of the modified assignment problem there can be no variable x_{uv} with $d_{uv}^{(K)} > 0$.

So it is shown that the matrix $D^{(K)}$ has the above formulated property (9), by which $\Delta L_{Kj} = d_{Kj}^{(K)}$ $K\varepsilon M$, V $j\varepsilon N$.

Thus, with the help of the reduced coefficient matrix, all the optimal solutions of an assignment problem can be determined, which have the alternative assignment (K,1),(K,2),..,(K,n).

3. Example

The above described method for the simultaneous determination of several solutions of an assignment problem will be demonstrated with an example using numbers. In Figure 3, the coefficient matrix $D = (d_{ij})$ of the assignment problem to be solved is repeated (see [4], page 17, slightly modified). The assignments (i,i) V $i \in M$ are not permissible.

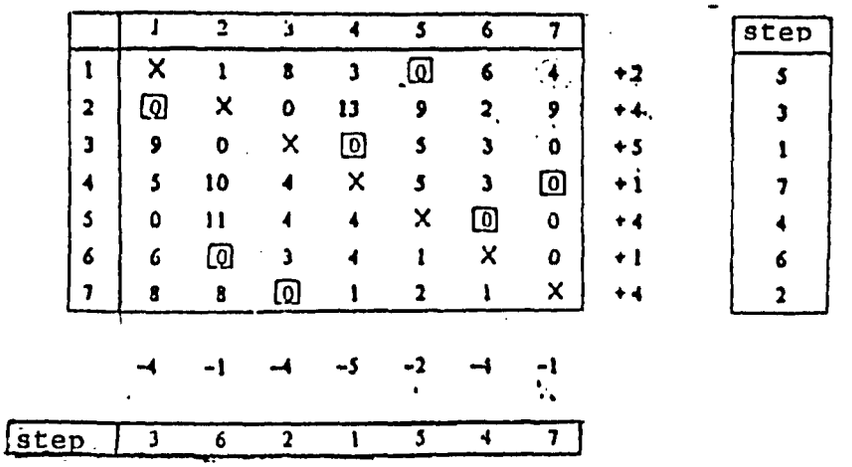
	1	2	3	4	5	6	7
1	X	20	33	23	12	19	16
2	19	X	24	32	20	14	20
3	35	25	X	26	23	22	18
4	24	28	28	X	16	15	11
5	13	23	22	17	X	6	5
6	20	13	22	18	7	X	6
7	20	19	17	13	6	6	X

Fig. 3. Coefficient matrix

Figure 4 shows the reduced coefficient matrix $D^{(0)} = (d_{ij}^{(0)})$ after the solving of the assignment problem. The optimal basis solution of the problem is denoted by the little boxes, and the value of the solution sought is $L^{(0)} = 104$.

	1	2	3	4	5	6	7
1	X	1	8	3	[0]	6	4
2	[0]	X	0	13	9	2	9
3	9	0	X	[0]	5	3	0
4	5	10	4	X	5	3	[0]
5	0	11	4	4	X	[0]	0
6	6	[0]	3	4	1	X	0
7	8	8	[0]	1	2	1	X

Fig. 4. Reduced coefficient matrix $D^{(0)} = (d_{ij}^{(0)})$



	1	2	3	4	5	6	7		step
1	X	1	8	3	[0]	6	4	+2	5
2	[0]	X	0	13	9	2	9	+4	3
3	9	0	X	[0]	5	3	0	+5	1
4	5	10	4	X	5	3	[0]	+1	7
5	0	11	4	4	X	[0]	0	+4	4
6	6	[0]	3	4	1	X	0	+1	6
7	8	8	[0]	1	2	1	X	+4	2
	-4	-1	-4	-5	-2	-4	-1		
step	3	6	2	1	5	4	7		

Fig. 5. Development of matrix $D^{(3)}$ from $D^{(0)}$

Proceedings from the reduced coefficient matrix $D^{(0)}$, for example, the matrix $D^{(3)}$ can now be determined. It gives information as to how great an amount ΔL_{3j} the value of the function sought increases, if some variable x_{3j}, V $j \in N$, is taken into the basis solution. We add a constant M_0, and for simplicity's sake we set $M_0 = 5$, to the elements of the third row of the matrix $D^{(0)}$. We expect that the optimal assignment in the matrix $D^{(3)}$ just as in the matrix $D^{(0)}$ would be assigned the value 0. This condition will be filled if we add M_0 to the elements of row 3 (Fig. 4) and substract it from the elements of column 4. Figure 5 gives information about the further mechanical operations. By substracting M_0, the fields (1,4), (5,4), (6,4), and (7,4) become negative assigned reduced coefficients. By (12), the constant M_1 is thus determined:

$$M_1 = \max \{-d^1_{ij} \mid d^1_{ij} < 0\}$$

$$M_1 = \max \{-d^1_{14}, -d^1_{54}, -d^1_{64}, -d^1_{74}\}$$

$$= \max \{2, 1, 1, 4\} = 4 = -d^1_{74}$$

$M_1 = 4$ is to be added to the elements of row 7 and subtracted from the elements of column 3. By (14) we find

$$M_2 = \max \{-d^2_{ij} \mid d^2_{ij} < 0\}$$

$$M_2 = \max \{-d^2_{23}, -d^2_{63}, -d^2_{14}, -d^2_{54}, -d^2_{64}\}$$

$$= -d^2_{23} = 4$$

M_2 is added to the reduced distances d^2_{ij} of row 2, and subtracted from the elements of column 1. In step 4, M_3 is to be determined:

$$M_3 = \max \{-d^3_{ij} \mid d^3_{ij} < 0\}$$

$$M_3 = -d^3_{51} = 4$$

and to be added to the elements of row 5, as to be subtracted from column 6. The further arithmetic operations proceed in a similar manner. Matrix $D^{(3)}$ is repeated in Figure 6.

	1	2	3	4	5	6	7
1	X	2	6	0	[0]	4	5
2	[0]	X	0	12	11	2	12
3	10	(4)	X	[0]	8	4	4
4	2	10	1	X	4	0	[0]
5	0	14	4	3	X	[0]	3
6	3	[0]	0	(0)	0	X	0
7	8	11	[0]	0	4	1	X

Fig. 6. Reduced coefficient matrix $D^{(3)}$

Matrix $D^{(3)}$ delivers 5 new solutions to the assignment problem, which can be determined with the aid of a simple marking process. The coefficients of row 3 of this matrix give the amounts

$$\Delta L_{3j} = d_{3j}^{(3)} \quad \forall\, j \in N$$

For example, the best solution of the assignment problem, which contains the assignment (3,2), has for its solution sought the value

$$L_{32} = L^{(0)} + \Delta L_{32}^{(3)}$$

$$= 104 + 4 = 108.$$

Instead of (3,4)(-) there is substituted (3,2)(+) in the basis, so that (6,2)(-) disappears from the solution and (6,4)(+) is brought into the solution. The path of substitution is marked by arrows in Figure 6. In the same manner the remaining solutions can be determined, and they alternatively have an assignment (3,j) $\forall$ $j \in N$.

REFERENCES

[1] Ford, L.R. and D.R. Fulkerson: A Simple Algorithm for Finding Maximal Network Flows and an Application to Hitchcock Problem, Can. J. Math. 9 (1957), p. 210-218.

[2] Hadley, G.: Linear Programming, 3 Aufl. London, Sidney-Manila, 1969.

[3] Kuhn, H.W.: The Hungarian Method for Assignment Problem, Nav. Res. Log. Quart. 3 (1956), p. 253-258.

[4] Muller-Merbach, H.: Optimale Reiheu Folgen, Berlin-Heidelberg - New York, 1970.

APPENDIX II

DATA FOR TEST PROBLEMS

(NUGENT, ET AL., 1968)

```
                    PROBLEM SIZE  5x 5
  703.
  704.
  705.
  706.
  707.              DISTANCE MATRIX
  708.
  709.                0  1  1  2  3
  710.
  711.                1  0  2  1  2
  712.
  713.                1  2  0  1  2
  714.
  715.                2  1  1  0  1
  716.
  717.                3  2  2  1  0
  718.
  719.
  720.
  721.
  722.              FLOW MATRIX
  723.
  724.                0  5  2  4  1
  725.
  726.                5  0  3  0  2
  727.
  728.                2  3  0  0  0
  729.
  730.                4  0  0  0  5
  731.
  732.                1  2  0  5  0
  733.
  734.
  735.
...
?
```

```
                    PROBLEM SIZE  6X 6
  753.
  754.
  755.
  756.
  757.              DISTANCE MATRIX
  758.
  759.                0  1  2  1  2  3
  760.
  761.                1  0  1  2  1  2
  762.
  763.                2  1  0  3  2  1
  764.
  765.                1  2  3  0  1  2
  766.
  767.                2  1  2  1  0  1
  768.
  769.                3  2  1  2  1  0
  770.
  771.
  772.
  773.
  774.              FLOW MATRIX
  775.
  776.                0  5  2  4  1  0
  777.
  778.                5  0  3  0  2  2
  779.
  780.                2  3  0  0  0  0
  781.
  782.                4  0  0  0  5  2
  783.
  784.                1  2  0  5  0 10
  785.
  786.                0  2  0  2 10  0
  787.
  788.
  789...
 ?
```

```
                    PROBLEM SIZE  7X 7
807.
808.
809.
810.
811.          DISTANCE MATRIX
812.
813.            0  1  2  3  2  3  4
814.
815.            1  0  1  2  1  2  3
816.
817.            2  1  0  1  2  1  2
818.
819.            3  2  1  0  3  2  1
820.
821.            2  1  2  3  0  1  2
822.
823.            3  2  1  2  1  0  1
824.
825.            4  3  2  1  2  1  0
826.
827.
828.
829.
830.          FLOW MATRIX
831.
832.            0  5  2  4  1  0  0
833.
834.            5  0  3  0  2  2  2
835.
836.            2  3  0  1  0  2  5
837.
838.            4  0  1  0  5  2  2
839.
840.            1  2  0  5  0 10  0
841.
842.            0  2  2  2 10  0  5
843.
844.            0  2  5  2  0  5  0
845.
846.
...
?
```

866.
867.
868.
869.
870. DISTANCE MATRIX
871.
872. 0 1 2 3 1 2 3 4
873.
874. 1 0 1 2 2 1 2 3
875.
876. 2 1 0 1 3 2 1 2
877.
878. 3 2 1 0 4 3 2 1
879.
880. 1 2 3 4 0 1 2 3
881.
882. 2 1 2 3 1 0 1 2
883.
884. 3 2 1 2 2 1 0 1
885.
886. 4 3 2 1 3 2 1 0
887.
888.
889.
890.
891. FLOW MATRIX
892.
893. 0 5 2 4 1 0 0 6
894.
895. 5 0 3 0 2 2 2 0
896.
897. 2 3 0 0 0 0 0 5
898.
899. 4 0 0 0 5 2 2 10
900.
901. 1 2 0 5 0 10 0 0
902.
903. 0 2 0 2 10 0 5 1
904.
905. 0 2 0 2 0 5 0 10
906.
907. 6 0 5 10 0 1 10 0
908.
909.
910.
?

```
PROBLEM SIZE 12X12

 972.
 973.
 974.
 975.
 976.
 977.
 978.
 979.
 980.
 981.
 982.
 983.
 984.
 985.
 986.
 987.
 988.
 989.
 990.
 991.
 992.
 993.
 994.
 995.
 996.
 997.
 998.
 999.
1000.
1001.
1002.
1003.
10...
?
```

DISTANCE MATRIX

0	1	2	3	1	2	3	4	2	3	4	5
1	0	1	2	2	1	2	3	3	2	3	4
2	1	0	1	3	2	1	2	4	3	2	3
3	2	1	0	4	3	2	1	5	4	3	2
1	2	3	4	0	1	2	3	1	2	3	4
2	1	2	3	1	0	1	2	2	1	2	3
3	2	1	2	2	1	0	1	3	2	1	2
4	3	2	1	3	2	1	0	4	3	2	1
2	3	4	5	1	2	3	4	0	1	2	3
3	2	3	4	2	1	2	3	1	0	1	2
4	3	2	3	3	2	1	2	2	1	0	1
5	4	3	2	4	3	2	1	3	2	1	0

```
1006.
1007.                FLOW MATRIX
1008.
1009.                     0  5  2  4  1  0  0  6  2  1  1  1
1010.
1011.                     5  0  3  0  2  2  2  0  4  5  0  0
1012.
1013.                     2  3  0  0  0  0  0  5  5  2  2  2
1014.
1015.                     4  0  0  0  5  2  2 10  0  0  5  5
1016.
1017.                     1  2  0  5  0 10  0  0  0  5  1  1
1018.
1019.                     0  2  0  2 10  0  5  1  1  5  4  0
1020.
1021.                     0  2  0  2  0  5  0 10  5  2  3  3
1022.
1023.                     6  0  5 10  0  1 10  0  0  0  5  0
1024.
1025.                     2  4  5  0  0  1  5  0  0  0 10 10
1026.
1027.                     1  5  2  0  5  5  2  0  0  0  5  0
1028.
1029.                     1  0  2  5  1  4  3  5 10  5  0  2
1030.
1031.                     1  0  2  5  1  0  3  0 10  0  2  0
1032.
:...
?
```

PROBLEM SIZE 15X15

1011.
1012.
1013.
1014.
1015. DISTANCE MATRIX
1016.
1017. 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6
1018.
1019. 1 0 1 2 3 2 1 2 3 4 3 2 3 4 5
1020.
1021. 2 1 0 1 2 3 2 1 2 3 4 3 2 3 4
1022.
1023. 3 2 1 0 1 4 3 2 1 2 5 4 3 2 3
1024.
1025. 4 3 2 1 0 5 4 3 2 1 6 5 4 3 2
1026.
1027. 1 2 3 4 5 0 1 2 3 4 1 2 3 4 5
1028.
1029. 2 1 2 3 4 1 0 1 2 3 2 1 2 3 4
1030.
1031. 3 2 1 2 3 2 1 0 1 2 3 2 1 2 3
1032.
1033. 4 3 2 1 2 3 2 1 0 1 4 3 2 1 2
1034.
1035. 5 4 3 2 1 4 3 2 1 0 5 4 3 2 1
1036.
1037. 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4
1038.
1039. 3 2 3 4 5 2 1 2 3 4 1 0 1 2 3
1040.
1041. 4 3 2 3 4 3 2 1 2 3 2 1 0 1 2
1042.
1043. 5 4 3 2 3 4 3 2 1 2 3 2 1 0 1
1044.
1045. 6 5 4 3 2 5 4 3 2 1 4 3 2 1 0
1046.
10

```
1048.
1049.
1050.
1051.
1052.
1053.
1054.
1055.
1056.
1057.
1058.
1059.
1060.
1061.
1062.
1063.
1064.
1065.
1066.
1067.
1068.
1069.
1070.
1071.
1072.
1073.
1074.
1075.
1076.
1077.
1078.
1079.
1080.
1081.
1082.
108...
?
```

FLOW MATRIX

0	10	0	5	1	0	1	2	2	2	2	0	4	0	0
10	0	1	3	2	2	2	3	2	0	2	0	10	5	0
0	1	0	10	2	0	2	5	4	5	2	2	5	5	5
5	3	10	0	1	1	5	0	0	2	1	0	2	5	0
1	2	2	1	0	3	5	5	5	1	0	3	0	5	5
0	2	0	1	3	0	2	2	1	5	0	0	2	5	10
1	2	2	5	5	2	0	6	0	1	5	5	5	1	0
2	3	5	0	5	2	6	0	5	2	10	0	5	0	0
2	2	4	0	5	1	0	5	0	0	10	5	10	0	2
2	0	5	2	1	5	1	2	0	0	0	4	0	0	5
2	2	2	1	0	0	5	10	10	0	0	5	0	5	0
0	0	2	0	3	0	5	0	5	4	5	0	3	3	0
4	10	5	2	0	2	5	5	10	0	0	3	0	10	2
0	5	5	5	5	5	1	0	0	0	5	3	10	0	4
0	0	5	0	5	10	0	0	2	5	0	0	2	4	0

PROBLEM SIZE 20X20

1107.
1108.
1109.
1110.
1111. DISTANCE MATRIX
1112.
1113. 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7
1114.
1115. 1 0 1 2 3 2 1 2 3 4 3 2 3 4 5 4 3 4 5 6
1116.
1117. 2 1 0 1 2 3 2 1 2 3 4 3 2 3 4 5 4 3 4 5
1118.
1119. 3 2 1 0 1 4 3 2 1 2 5 4 3 2 3 6 5 4 3 4
1120.
1121. 4 3 2 1 0 5 4 3 2 1 6 5 4 3 2 7 6 5 4 3
1122.
1123. 1 2 3 4 5 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6
1124.
1125. 2 1 2 3 4 1 0 1 2 3 2 1 2 3 4 3 2 3 4 5
1126.
1127. 3 2 1 2 3 2 1 0 1 2 3 2 1 2 3 4 3 2 3 4
1128.
1129. 4 3 2 1 2 3 2 1 0 1 4 3 2 1 2 5 4 3 2 3
1130.
1131. 5 4 3 2 1 4 3 2 1 0 5 4 3 2 1 6 5 4 3 2
1132.
1133. 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4 1 2 3 4 5
1134.
1135. 3 2 3 4 5 2 1 2 3 4 1 0 1 2 3 2 1 2 3 4
1136.
1137. 4 3 2 3 4 3 2 1 2 3 2 1 0 1 2 3 2 1 2 3
1138.
1139. 5 4 3 2 3 4 3 2 1 2 3 2 1 0 1 4 3 2 1 2
1140.
1141. 6 5 4 3 2 5 4 3 2 1 4 3 2 1 0 5 4 3 2 1
1142.
1143. 3 4 5 6 7 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4
1144.
1145. 4 3 4 5 6 3 2 3 4 5 2 1 2 3 4 1 0 1 2 3
1146.
1147. 5 4 3 4 5 4 3 2 3 4 3 2 1 2 3 2 1 0 1 2
1148.
1149. 6 5 4 3 4 5 4 3 2 3 4 3 2 1 2 3 2 1 0 1
1150.
1151. 7 6 5 4 3 6 5 4 3 2 5 4 3 2 1 4 3 2 1 0
1152.
1153.
1154.
:...
?

```
$
 1154.
 1155.
 1156.          FLOW MATRIX
 1157.
 1158.            0  0  5  0  5  2 10  3  1  5  5  5  0  0  5  4  4  0  0  1
 1159.
 1160.            0  0  3 10  5  1  5  1  2  4  2  5  0 10 10  3  0  5 10  5
 1161.
 1162.            5  3  0  2  0  5  2  4  4  5  0  0  0  5  1  0  0  5  0  0
 1163.
 1164.            0 10  2  0  1  0  5  2  1  0 10  2  2  0  2  1  5  2  5  5
 1165.
 1166.            5  5  0  1  0  5  6  5  2  5  2  0  5  1  1  1  5  2  5  1
 1167.
 1168.            2  1  5  0  5  0  5  2  1  6  0  0 10  0  2  0  1  0  1  5
 1169.
 1170.           10  5  2  5  6  5  0  0  0  0  5 10  2  2  5  1  2  1  0 10
 1171.
 1172.            3  1  4  2  5  2  0  0  1  1 10 10  2  0 10  2  5  2  2 10
 1173.
 1174.            1  2  4  1  2  1  0  1  0  2  0  3  5  5  0  5  0  0  0  2
 1175.
 1176.            5  4  5  0  5  6  0  1  2  0  5  5  0  5  1  0  0  5  5  2
 1177.
 1178.            5  2  0 10  2  0  5 10  0  5  0  5  2  5  1 10  0  2  2  5
 1179.
 1180.            5  5  0  2  0  0 10 10  3  5  5  0  2 10  5  0  1  1  2  5
 1181.
 1182.            0  0  0  2  5 10  2  2  5  0  2  2  0  2  2  1  0  0  0  5
 1183.
 1184.            0 10  5  0  1  0  2  0  5  5  5 10  2  0  5  5  1  5  5  0
 1185.
 1186.            5 10  1  2  1  2  5 10  0  1  1  5  2  5  0  3  0  5 10 10
 1187.
 1188.            4  3  0  1  1  0  1  2  5  0 10  0  1  5  3  0  0  0  2  0
 1189.
 1190.            4  0  0  5  5  1  2  5  0  0  0  1  0  1  0  0  0  5  2  0
 1191.
 1192.            0  5  5  2  2  0  1  2  0  5  2  1  0  5  5  0  5  0  1  1
 1193.
 1194.            0 10  0  5  5  1  0  2  0  5  2  2  0  5 10  2  2  1  0  6
 1195.
 1196.            1  5  0  5  1  5 10 10  2  2  5  5  5  0 10  0  0  1  6  0
 1197.
 1198.
 1199.
...
?
```

PROBLEM SIZE 30X30

DISTANCE MATRIX

0	1	2	3	4	5	1	2	3	4	5	6	2	3	4	5	6	7	3	4	5	6	7	8	4	5	6	7	8	9
1	0	1	2	3	4	2	1	2	3	4	5	3	2	3	4	5	6	4	3	4	5	6	7	5	4	5	6	7	8
2	1	0	1	2	3	3	2	1	2	3	4	4	3	2	3	4	5	5	4	3	4	5	6	6	5	4	5	6	7
3	2	1	0	1	2	4	3	2	1	2	3	5	4	3	2	3	4	6	5	4	3	4	5	7	6	5	4	5	6
4	3	2	1	0	1	5	4	3	2	1	2	6	5	4	3	2	3	7	6	5	4	3	4	8	7	6	5	4	5
5	4	3	2	1	0	6	5	4	3	2	1	7	6	5	4	3	2	8	7	6	5	4	3	9	8	7	6	5	4
1	2	3	4	5	6	0	1	2	3	4	5	1	2	3	4	5	6	2	3	4	5	6	7	3	4	5	6	7	8
2	1	2	3	4	5	1	0	1	2	3	4	2	1	2	3	4	5	3	2	3	4	5	6	4	3	4	5	6	7
3	2	1	2	3	4	2	1	0	1	2	3	3	2	1	2	3	4	4	3	2	3	4	5	5	4	3	4	5	6
4	3	2	1	2	3	3	2	1	0	1	2	4	3	2	1	2	3	5	4	3	2	3	4	6	5	4	3	4	5
5	4	3	2	1	2	4	3	2	1	0	1	5	4	3	2	1	2	6	5	4	3	2	3	7	6	5	4	3	4
6	5	4	3	2	1	5	4	3	2	1	0	6	5	4	3	2	1	7	6	5	4	3	2	8	7	6	5	4	3
2	3	4	5	6	7	1	2	3	4	5	6	0	1	2	3	4	5	1	2	3	4	5	6	2	3	4	5	6	7
3	2	3	4	5	6	2	1	2	3	4	5	1	0	1	2	3	4	2	1	2	3	4	5	3	2	3	4	5	6
4	3	2	3	4	5	3	2	1	2	3	4	2	1	0	1	2	3	3	2	1	2	3	4	4	3	2	3	4	5
5	4	3	2	3	4	4	3	2	1	2	3	3	2	1	0	1	2	4	3	2	1	2	3	5	4	3	2	3	4
6	5	4	3	2	3	5	4	3	2	1	2	4	3	2	1	0	1	5	4	3	2	1	2	6	5	4	3	2	3
7	6	5	4	3	2	6	5	4	3	2	1	5	4	3	2	1	0	6	5	4	3	2	1	7	6	5	4	3	2
3	4	5	6	7	8	2	3	4	5	6	7	1	2	3	4	5	6	0	1	2	3	4	5	1	2	3	4	5	6
4	3	4	5	6	7	3	2	3	4	5	6	2	1	2	3	4	5	1	0	1	2	3	4	2	1	2	3	4	5
5	4	3	4	5	6	4	3	2	3	4	5	3	2	1	2	3	4	2	1	0	1	2	3	3	2	1	2	3	4
6	5	4	3	4	5	5	4	3	2	3	4	4	3	2	1	2	3	3	2	1	0	1	2	4	3	2	1	2	3

7	6	5	4	3	4	6	5	4	3	2	3	5	4	3	2	1	2	4	3	2	1	0	1	5	4	3	2	1	2
8	7	6	5	4	3	7	6	5	4	3	2	6	5	4	3	2	1	5	4	3	2	1	0	6	5	4	3	2	1
4	5	6	7	8	9	3	4	5	6	7	8	2	3	4	5	6	7	1	2	3	4	5	6	0	1	2	3	4	5
5	4	5	6	7	8	4	3	4	5	6	7	3	2	3	4	5	6	2	1	2	3	4	5	1	0	1	2	3	4
6	5	4	5	6	7	5	4	3	4	5	6	4	3	2	3	4	5	3	2	1	2	3	4	2	1	0	1	2	3
7	6	5	4	5	6	6	5	4	3	4	5	5	4	3	2	3	4	4	3	2	1	2	3	3	2	1	0	1	2
8	7	6	5	4	5	7	6	5	4	3	4	6	5	4	3	2	3	5	4	3	2	1	2	4	3	2	1	0	1
9	8	7	6	5	4	8	7	6	5	4	3	7	6	5	4	3	2	6	5	4	3	2	1	5	4	3	2	1	0

FLOW MATRIX

0	3	2	0	0	2	10	5	0	5	2	5	0	0	2	0	5	6	3	0	1	10	0	10	2	1	1	1	0	1
3	0	4	0	10	4	0	0	2	2	1	0	5	0	0	0	0	2	0	1	6	1	0	1	2	2	5	1	10	5
2	4	0	3	4	0	5	5	5	1	4	1	0	4	0	4	0	6	3	2	5	5	2	1	0	0	3	1	0	2
0	0	3	0	0	0	0	2	2	0	6	0	2	5	2	5	1	1	1	1	2	2	4	0	2	0	2	2	5	5
0	10	4	0	0	5	2	0	0	0	0	2	0	0	0	0	2	1	0	0	2	0	5	1	0	2	1	0	2	1
2	4	0	0	5	0	1	2	2	1	4	10	10	2	5	5	0	5	0	0	0	10	0	0	0	4	0	10	1	1
10	0	5	0	2	1	0	10	10	5	10	10	6	0	0	10	2	1	10	1	5	5	2	3	5	0	2	0	1	3
5	0	5	2	0	2	10	0	1	3	5	0	0	0	2	4	5	2	10	6	0	5	5	2	5	0	5	5	0	2
0	2	5	2	0	2	10	1	0	10	2	1	5	2	0	3	0	2	0	0	4	0	5	2	0	5	2	2	5	2
5	2	1	0	0	1	5	3	10	0	5	5	6	0	1	5	5	0	5	2	3	5	0	5	2	10	10	1	5	2
2	1	4	6	0	4	10	5	2	5	0	0	0	1	2	1	0	2	0	0	0	6	6	0	4	5	3	2	2	10
5	0	1	0	2	10	10	0	1	5	0	0	5	5	2	0	0	0	0	2	0	4	5	10	1	0	0	0	0	1
0	5	0	2	0	10	6	0	5	6	0	5	0	2	0	4	2	2	1	0	6	2	1	5	5	0	0	1	5	5
0	0	4	5	0	2	0	0	2	0	1	5	2	0	2	1	0	5	3	10	0	0	4	2	0	0	4	2	5	5
2	0	0	2	0	5	0	2	0	1	2	2	0	2	0	4	5	1	0	1	0	6	0	2	0	0	5	1	1	0

?

```
3.
1324.   0  0  4  5  0  5 10  4  3  5  1  0  4  1  4  0  0  3  0  2  2  0  2  0  5  0  5  2  5 10
1325.
1326.   5  0  0  1  2  0  2  5  0  5  0  0  2  0  5  0  0  2  2  0  0  0  6  5  3  5  0  0  5  1
1327.
1328.   6  2  6  1  1  5  1  2  2  0  2  0  2  5  1  3  2  0  5  1  2 10 10  4  0  0  5  0  0  0
1329.
1330.   3  0  3  1  0  0 10 10  0  5  0  0  1  3  0  0  2  5  0  0  5  5  1  0  5  2  1  2 10 10
1331.
1332.   0  1  2  1  0  0  1  6  0  2  0  2  0 10  1  2  0  1  0  0  5  2  1  3  1  5  6  5  5  3
1333.
1334.   1  6  5  2  2  0  5  0  4  3  0  0  6  0  0  2  0  2  5  5  0  4  0  1  0  0  0  5  0  0
1335.
1336.  10  1  5  2  0 10  5  5  0  5  6  4  2  0  5  0  0 10  5  2  4  0  5  0  4  4  5  0  2  5
1337.
1338.   0  0  2  4  5  0  2  5  5  0  6  5  1  4  0  2  6 10  1  1  0  5  0  0  4  4  1  0  2  2
1339.
1340.  10  1  1  0  1  0  3  2  2  5  0 10  5  2  2  0  5  4  0  3  1  0  0  0  5  5  0  1  0  0
1341.
1342.   2  2  0  2  0  0  5  5  0  2  4  1  5  0  0  5  3  0  5  1  0  4  4  5  0  1  0 10  1  0
1343.
1344.   1  2  0  0  2  4  0  0  5 10  5  0  0  0  0  0  5  0  2  5  0  4  4  5  1  0  0  0  0  0
1345.
1346.   1  5  3  2  1  0  2  5  2 10  3  0  0  4  5  5  0  5  1  6  0  5  1  0  0  0  0  0  0 10
1347.
1348.   1  1  1  2  0 10  0  5  2  1  2  0  1  2  1  2  0  0  2  5  5  0  0  1 10  0  0  0  2  2
1349.
1350.   0 10  0  5  2  1  1  0  5  5  2  0  5  5  1  5  5  0 10  5  0  2  2  0  1  0  0  2  0  2
1351.
1352.   1  5  2  5  1  1  3  2  2  2 10  1  5  5  0 10  1  0 10  3  0  5  2  0  0  0 10  2  2  0
1353.
1354.
1355.
...
?
```

APPENDIX III

DATA FOR PRACTICAL PROBLEM

(ELSHAFEI, 1977)

PROBLEM SIZE 19X19

DISTANCE MATRIX

0	12	36	28	52	44	110	126	94	63	130	102	65	98	132	132	126	120	126
12	0	24	75	82	75	108	70	124	86	93	106	58	124	161	161	70	64	70
36	24	0	47	71	47	110	73	126	71	95	110	46	127	163	163	73	67	73
28	75	47	0	42	34	148	111	160	52	94	148	49	117	104	109	111	105	111
52	82	71	42	0	42	125	136	102	22	73	125	32	94	130	130	136	130	136
44	75	47	34	42	0	148	111	162	52	96	148	49	117	152	152	111	105	111
110	108	110	148	125	148	0	46	46	136	47	30	108	51	79	79	46	47	41
126	70	73	111	136	111	46	0	69	141	63	46	119	68	121	121	27	24	36
94	124	126	160	102	162	46	69	0	102	34	45	84	23	80	80	69	64	51
63	86	71	52	22	52	136	141	102	0	64	118	29	95	131	131	141	135	141
130	93	95	94	73	96	47	63	34	64	0	47	56	54	94	94	63	46	24
102	106	110	148	125	148	30	46	45	118	47	0	100	51	89	89	46	40	36
65	58	46	49	32	49	108	119	84	29	56	100	0	77	113	113	119	113	119
98	124	127	117	94	117	51	68	23	95	54	51	77	0	79	79	68	62	51
132	161	163	104	130	152	79	121	80	131	94	89	113	79	0	10	113	107	119
132	161	163	109	130	152	79	121	80	131	94	89	113	79	10	0	113	107	119
126	70	73	111	136	111	46	27	69	141	63	46	119	68	113	113	0	6	24
120	64	67	105	130	105	47	24	64	135	46	40	113	62	107	107	6	0	12
126	70	73	111	136	111	41	36	51	141	24	36	119	51	119	119	24	12	0

FLOW MATRIX

0	76687	0	415	545	819	135	1368	819	5630	0	3432	9082	1503	0	0	13732	1368	1783
76687	0	40951	4118	5767	2055	1917	2746	1097	5712	0	0	0	268	0	1373	268	0	0
0	40951	0	3848	2524	3213	2072	4225	566	0	0	404	9372	0	972	0	13538	1368	0
415	4118	3848	0	256	0	0	0	0	829	128	0	0	0	0	0	0	0	0
545	5767	2524	256	0	0	0	0	47	1655	287	0	42	0	0	0	226	0	0
819	2055	3213	0	0	0	0	0	0	926	161	0	0	0	0	0	0	0	0
135	1917	2072	0	0	0	0	0	196	1538	196	0	0	0	0	0	0	0	0
1368	2746	4225	0	0	0	0	0	0	0	301	0	0	0	0	0	0	0	0
819	1097	566	0	47	0	196	0	0	1954	418	0	0	0	0	0	0	0	0
5630	5712	0	829	1655	926	1538	0	1954	0	0	282	0	0	0	0	0	0	0
0	0	0	128	287	161	196	301	418	0	0	1686	0	0	0	0	226	0	0
3432	0	404	0	0	0	0	0	0	282	1686	0	0	0	0	0	0	0	0
9082	0	9372	0	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1503	268	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	972	0	0	0	0	0	0	0	0	0	0	0	0	99999	0	0	0
0	1373	0	0	0	0	0	0	0	0	0	0	0	0	99999	0	0	0	0
13732	268	13538	0	226	0	0	0	0	0	226	0	0	0	0	0	0	0	0
1368	0	1368	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1783	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

?

APPENDIX IV

COMPUTER PROGRAMS

```
C  ********************************************************
C  *     A HEURISTIC ALGORITHM FOR THE SOLUTION OF        *
C  *     QUADRATIC ASSIGNMENT PROBLEMS.                   *
C  ********************************************************
C
C      THIS IS AN ENCODING OF THE ALGORITHM BASED ON THE
C      PAPER "A HEURISTIC APPROACH TO QUADRATIC ASSIGNMENT
C      PROBLEM" BY R.S.LASHKARI AND S.C.JAISINGH,PUBLISHED
C      IN THE JOURNAL OF OPERATIONAL RESEARCH SOCIETY,
C      VOL.31,NO.9,1980.
C
C
C      THIS PROGRAM IS CURRENTLY DIMENSIONED TO HANDLE
C      PROBLEMS WITH UP TO 60 FACILITIES AND 60 SITES.
C      FOR PROBLEMS HAVING MORE THAN 60 FACILITIES AND
C      SITES,ALL DIMENSION STATEMENTS MUST BE ADUSTED
C      ACCORDINGLY.
C
C     VARIABLES
C  ***************
C     SIZE      -DIMENSIONED SIZE OF THE MATRICES
C     N         -SIZE OF THE COST MATRIX
C     STO1      -ARRAY USED TO STORE THE COST OF THE ASSIGNMENT
C                AT THE END OF EACH ITERATION
C     EPS       -A PARAMETER USED WITH STOPPING CRITERION
C     RLAB      -ARRAY FOR ROW LABELS FOR ASSIGNMENT ROUTINE
C     CLAB      -ARRAY FOR COL LABELS FOR ASSIGNMENT ROUTINE
C     MOROWS    -ARRAY FOR LABELLED ROWS FOR ASSIGNMMENT ROUTINE
C     MORCOL    -ARRAY FOR LABELLED COLUMN FOR ASSIGNMENT ROUTINE
C     RP,RK     -ARRAY FOR ROW POINTERS FOR ASSIGNMENT ROUTINE
C     U         -ARRAY FOR ROW DUAL COST FOR ASSIGNMENT
C     ID1       -DISTANCE MATRIX
C     IW1       -FLOW MATRIX
C     MATC      -LOWER BOUND MATRIX ON THE COST OF LOCATING
C                FACILITY J AT SITE I
C     MATR1     -CURRENT REDUCED MATRIX
C     MAT       -CURRENT UPDATED MATRIX
C     CT        -REDUCED MATRIX OBTAINED AT THE BEGINNING OF
C                THE IMPROVEMENT PROCESS IN EACH ROW
C     IAS       -ASSIGNMENT VECTOR OBTAINED AT THE BEGINNING
C                OF THE IMPROVEMENT PROCESS IN EACH ROW
C     IAS1,IASP -CURRENT ASSIGNMENT VECTORS
C     KRC        -ITERATION COUNT
C     KUP       -COST CORROSPONDING TO BEST ASSIGNMENT IASM(I)
C     IASMAT    -ASSIGNMENT AT THE END OF EACH ITERATION
C     ICOST     -COST OF THE ASSIGNMENT AT THE END OF EACH
C                ITERATION
C
C
      IMPLICIT INTEGER(A-Z)
      REAL STO1(100),EPS
      INTEGER*4 RLAB(60),CLAB(60),MOROWS(60),MORCOL(60),RP(60)
      DIMENSION U(60),V(60)
      DIMENSION ID1(60,60),IW1(60,60),MATC(60,60)
      DIMENSION MATR1(60,60),MAT(60,60)
      DIMENSION CT(60,60)
?
```

```
      INTEGER*4 IAS1(60)
      INTEGER*4 IASMAT(60),IASM(60),IAS(60)
      INTEGER RK(60),IASP(60)
      SIZE=60
      NPROB=0
  676 CONTINUE
      KSS=2**31-1
      KODE=0
C     READ AND PRINT THE DATA
      READ(5,1)N
    1 FORMAT(I5)
      WRITE(6,315)
      WRITE(6,1201)N,N
 1201 FORMAT(30X,'PROBLEM SIZE ',I2,'X',I2)
      DO 1001 I=1,N
 1001 READ(5,1000)(ID1(I,J),J=1,N)
      WRITE(6,315)
      WRITE(6,1003)
 1003 FORMAT(10X,'DISTANCE MATRIX')
      DO 301 I=1,N
  301 WRITE(6,302)(ID1(I,J),J=1,N)
  302 FORMAT(/,10X,30I3)
      WRITE(6,315)
  315 FORMAT(///)
      DO 1004 I=1,N
 1004 READ(5,1000)(IW1(I,J),J=1,N)
 1000 FORMAT(20I4)
      WRITE(6,1005)
 1005 FORMAT(10X,'FLOW MATRIX')
      DO 303 I=1,N
  303 WRITE(6,302)(IW1(I,J),J=1,N)
      WRITE(6,315)
      KTOT=0
      CALL TINIT
C     OBTAIN THE LOWER BOUND MATRIX
      CALL MATCH(N,ID1,IW1,MATC)
      DO 678 I=1,N
      DO 678 J=1,N
  678 MAT(I,J)=MATC(I,J)
      WRITE(6,315)
C     OBTAIN THE LOWER BOUND ON THE COST
      KODE=0
      CALL ASSIGN(SIZE,N,MATC,RLAB,CLAB,MOROWS,MORCOL,RP,IASMAT,U,V,TOTAL
     *L,KODE,CT)
      CALL COST(IASMAT,N,ID1,IW1,ICOST)
      TOTAL=TOTAL/2
      WRITE(6,1203)TOTAL
 1203 FORMAT(10X,'LOWER BOUND ON THE COST=',I9)
      WRITE(6,315)
      WRITE(6,320)
      WRITE(6,1202)
 1202 FORMAT(10X,'OBJECTIVE',/,10X,'FUNCTION',/,4X,'ITER.',2X,'VALUE'
     1,12X,'LAYOUT')
      WRITE(6,320)
  320  FORMAT(1X,'-----------------------------------')
      WRITE(6,319)
?
```

```
      KRC=0
      WRITE(6,1006)KRC,ICOST,(IASMAT(I),I=1,N)
 1006 FORMAT(5X,I2,2X,I6,4X,30I3)
      WRITE(6,319)
      KUP=ICOST
      STO1(1)=ICOST
      IF(ICOST.EQ.TOTAL)GO TO 680
      DO 546 I=1,N
  546 IASM(I)=IASMAT(I)
C     IMPROVE THE ELEMENTS OF THE LOWER BOUND MATRIX
      DO 111 I=1,N
      IASP(I)=IASMAT(I)
      IAS(I)=IASMAT(I)
      IAS1(I)=IAS(I)
      DO 111 J=1,N
  111 MATR1(I,J)=MATC(I,J)
   14 CONTINUE
      CALL TINIT
      DO 570 IB=1,N
      IF(IB.EQ.1)GO TO 1061
      IB1=IB-1
      JK=KMT
C
C     OBTAIN THE ADMISSIBLE CELLS IN THE UPDATED ROW
C
      CALL REDUCE(IB1,JK,MATR1,N)
 1061 CONTINUE
      DO 59 KK=1,N
   59 IAS(KK)=IAS1(KK)
C
C     REDUCE THE MATRIX SO AS TO OBTAIN N SOLUTIONS
C     SIMULTANEOUSELY FOR THE ROW 'IB' WHOSE ELEMENTS
C     ARE TO BE IN THE SOLUTION.
C
      CALL SIMULT(IB,N,MATR1,SIZE,IAS,KSS)
      KODE=3
      CALL ASSIGN (SIZE,N,MATR1,RLAB,CLAB,MOROWS,MORCOL,RK,IAS,U,V,
     *TOTAL,KODE,CT)
      KMT=2**31-1
      DO 580 J=1,N
      DO 58 IC=1,N
      RP(IC)=RK(IC)
      IASMAT(IC)=IAS(IC)
      KJ=IASMAT(IC)
      IASP(IC)=0
      RLAB(IC)=KJ
   58 CLAB(KJ)=IC
      DO 55 IT=1,N
      DO 55 IT1=1,N
      IF(IB.EQ.IT.AND.IT1.EQ.J)GO TO 56
      IF(IB.EQ.IT)GO TO 57
      MATC(IT,IT1)=CT(IT,IT1)
      GO TO 55
   56 MATC(IT,IT1)=0
      GO TO 55
   57 MATC(IT,IT1)=2**31-1
?
```

```
      55 CONTINUE
         K1=IASMAT(IB)
         K2=CLAB(J)
         CLAB(K1)=0
         RLAB(K2)=-1
         CLAB(J)=IB
         RLAB(IB)=J
         IASMAT(K2)=0
         IASMAT(IB)=J
         RP(IB)=J
         IF(K1.EQ.J.AND.K2.EQ.IB)GO TO 681
         DO 555 IP=1,N
         KMN=IASMAT(IP)
         IF(KMN.EQ.0)GO TO 555
         IASP(KMN)=IP
     555 CONTINUE
         KODE=2
         CALL ASSIGN(SIZE,N,MATC,RLAB,CLAB,MOROWS,MORCOL,RP,IASP,U,V,TOTA
        *L,KODE,CT)
         GO TO 682
     681 CONTINUE
         DO 683 IL=1,N
     683 IASP(IL)=IASMAT(IL)
     682 CONTINUE
         YK=0
         DO 423 IK=1,N
         IK1=IASP(IK)
     423 YK=YK+ID1(IB,IK)*IU1(J,IK1)
         MATR1(IB,J)=MATR1(IB,J)-MAT(IB,J)+YK
         MAT(IB,J)=YK
         IF(KMT.GT.MATR1(IB,J))GO TO 1125
         GO TO 580
    1125 DO 1127 IN1=1,N
    1127 IAS1(IN1)=IASP(IN1)
         KMT=MATR1(IB,J)
     580 CONTINUE
     570 CONTINUE
         IB=N
         JK=KMT
         CALL REDUCE (IB,JK,MATR1,N)
         DO 1075 I=1,N
    1075 IASMAT(I)=IAS1(I)
         KODE=2
         CALL COST(IASMAT,N,ID1,IU1,ICOST)
         KRC=KRC+1
         IF(ICOST.LE.KUP)GO TO 343
         GO TO 341
     343 DO 345 IP=1,N
     345 IASM(IP)=IASMAT(IP)
         KUP=ICOST
     341 CONTINUE
         CALL TUSED(M)
         KTOT=KTOT+M
C        PRINT THE RESULT AT THE END OF EACH ITERATION
         WRITE(6,1006)KRC,ICOST,(IASMAT(I),I=1,N)
         WRITE(6,319)
?
```

```
  319 FORMAT(/)
      STO1(KRC+1)=ICOST
      EPS=ABS(STO1(KRC+1)-STO1(KRC+0))/STO1(KRC+0)
      IF(EPS.LE.0.0025.OR.KRC.GT.15)GO TO 688
      GO TO 14
  680 WRITE(6,687)
  687 FORMAT(10X,'SOLUTION IS OPTIMAL')
  688 CONTINUE
      WRITE(6,1011)KUP
 1011 FORMAT(10X,'FINAL COST OF ASSIGNMENT=',I9)
      WRITE(6,319)
      WRITE(6,1012)KTOT
 1012 FORMAT(10X,'TOTAL TIME=',I7,' MSEC')
      NPROB=NPROB+1
      IF(NPROB.LT.8)GO TO 676
      STOP
      END
      SUBROUTINE SIMULT(IB,N,MATR1,S,IASMAT,KSS)
C
C  ***********************************************************
C  *   THIS SUBROUTINE FINDS SEVERAL SOLUTIONS TO          *
C  *   A LINEAR ASSIGNMENT PROBLEM SIMULTANEOUSELY.        *
C  ***********************************************************
C
C     THIS ENCODING IS BASED ON THE PAPER 'EINE METHODE ZUR
C     BESTIMMUNG MEHRERER LOSUNGEN FUR DAS ZUORDNUNGS PROBLEM',
C     BY KREUZBERGER,H. AND WEITERSTADT IN THE JOURNAL OF
C     ANGEWANDTE INFORMATIC,1971.THE ENGLISH TRANSLATION
C     OF THIS PAPER CAN BE REFERRED TO IN THE APPENDIX.
C
C
C     VARIABLES
C   ****************
C     IB         =ROW NUMBER WHOSE ELEMENTS ARE TO BE IN THE
C                 SOLUTION
C     IASMAT(I) =MINIMUM COST ASSIGNMENT VECTOR INDICATING
C                 THE LOCATION OF FAC J(J=IASMAT(I)) AT SITE I.
C     MATR1(I,J)=REDUCE COEFFICIENT MATRIX WHICH IS
C                 USED TO FIND SEVERAL SOLUTIONS TO LINEAR ASSIGNMENT
C                 PROBLEM SIMULTANEOUSLY.
C
      IMPLICIT INTEGER(A-Z)
      DIMENSION MATR1(S,S)
      INTEGER IASMAT(S)
      INTEGER KROW(100),KCLN(100),KCOL(100)
      KST=2**31-1
      KR=IB
      DO 1 I=1,N
    1 KROW(I)=0
      KRS=IASMAT(KR)
      DO 572 I=1,N
      DO 569 J=1,N
C
C     SUBTRACT A CONSTANT KST >> 0 FROM ALL ELEMENTS
C     OF COLUMN KRS SUCH THAT IASMAT(IB)=KRS.
C
```

```
      MATR1(J,KRS)=MATR1(J,KRS)-KST
C
C     ADD CONSTANT KST TO ALL ELEMENTS OF ROW IB
C
  569 MATR1(KR,J)=MATR1(KR,J)+KST
      KCLN(I)=KRS
      KD=99999999
      DO 530 IF=1,I
      DO 530 IF1=1,N
      IF2=KCLN(IF)
      IF(MATR1(IF1,IF2).GE.0)GO TO 530
      IF(KD.GE.MATR1(IF1,IF2))GO TO 531
      GO TO 530
  531 IF(KROW(IF1).NE.0)GO TO 530
      KD=MATR1(IF1,IF2)
      KJ=IF1
      IF(KD.EQ.KSS)GO TO 189
  530 CONTINUE
  189 CONTINUE
      KCOL(KRS)=KST
      KROW(KR)=-KST
      KR=KJ
      KRS=IASMAT(KR)
      KST=-KD
      KSS=KD
  572 CONTINUE
      RETURN
      END
      SUBROUTINE MATCH(N,ID1,IW1,MATC)
C   ***********************************************************
C   *  THIS SUBROUTINE FINDS THE LOWER BOUNDS ON THE     *
C   *  COST OF LOCATING FACILITIES AT DIFFERENT SITES    *
C   *  BY MATCHING HIGHEST FLOWS WITH LEAST DISTANCES    *
C   ***********************************************************
C
C     VARIABLES
C   *************
C     KA(I)=GIVEN ROW VECTOR OF THE ELEMENTS IN A DISTANCE
C     MATRIX ARRANGED IN ASCENDING ORDER.
C     KBC(I)=GIVEN ROW VECTOR OF THE ELEMENTS OF A  FLOW
C     MATRIX ARRANGED IN ASCENDING ORDER.
C
      DIMENSION KA(60),KBC(60),ID1(60,60),IW1(60,60),MATC(60,60)
      DO 612 I=1,N
      MK=0
      DO 600 J=1,N
      IF(J.EQ.I)GO TO 600
      MK=MK+1
      KA(MK)=ID1(I,J)
  600 CONTINUE
      CALL SORT(MK,KA)
      DO 601 KK1=1,N
      MK1=0
      DO 603 KK2=1,N
      IF(KK2.EQ.KK1)GO TO 603
      MK1=MK1+1
?
```

```
      KBC(MK1)=IW1(KK1,KK2)
  603 CONTINUE
      CALL SORT(MK1,KBC)
      MATC(I,KK1)=0
      DO 604 I1=1,MK
      I2=MK-I1+1
  604 MATC(I,KK1)=KA(I1)*KBC(I2)+MATC(I,KK1)
  601 CONTINUE
  612 CONTINUE
      RETURN
      END
      SUBROUTINE REDUCE (IB,K,MATR1,N)
C     ************************************************************
C     * THIS SUBROUTINE REDUCES THE CURRENT UPDATED ROW          *
C     * IN THE LOWER BOUND MATRIX SO AS TO ATTAIN THE            *
C     * ADMISSIBLE CELLS (WHICH WERE LOST DURING UPDATING        *
C     * PROCESS).                                                *
C     ************************************************************
C
C     IB=ROW NUMBER WHOSE ELEMENTS ARE TO BE UPDATED
C     MATR1(I,J)=UPDATED MATRIX.
C
      IMPLICIT INTEGER(A-Z)
      DIMENSION MATR1(60,60)
      IF(K.LT.0)GO TO 3
      DO 2 I=1,N
    2 MATR1(IB,I)=MATR1(IB,I)-K
      RETURN
    3 K=-K
      DO 4 I=1,N
    4 MATR1(IB,I)=MATR1(IB,I)+K
      RETURN
      END
      SUBROUTINE COST (IASMAT,N,ID1,IW1,ICOST)
C     ************************************************************
C     *  THIS SUBROUTINE DETERMINES THE COST OF A GIVEN          *
C     *  ASSIGNMENT.                                             *
C     ************************************************************
C       ICOST=COST OF AN ASSIGNMENT REPRESENTED BY
C       IASMAT(I)
C
      INTEGER IASMAT(60)
      DIMENSION ID1(60,60),IW1(60,60)
      ICOST=0
      DO 125 I7=1,N
      K8=IASMAT(I7)
      DO 125 J7=I7,N
      K9=IASMAT(J7)
  125 ICOST=ICOST+ID1(I7,J7)*IW1(K8,K9)
      RETURN
      END
      SUBROUTINE SORT (N,KA)
C     ************************************************************
C     *   THIS SUBROUTINE ARRANGES THE NUMBERS  IN ASCENDING     *
C     *   ORDER..                                                *
C     ************************************************************
?
```

```
      DIMENSION KA(N)
      I=1
   10 I=I*2
      IF(I.LE.N)GO TO 10
      M=I-1
   20 M=M/2
      IF(M.EQ.0)GO TO 100
      L=N-M
      DO 40 J=1,L
      DO 30 J1=1,J,M
      I=J+1-J1
      IM=I+M
      IF(KA(IM).GE.KA(I))GO TO 40
      TEMP=KA(I)
      KA(I)=KA(IM)
      KA(IM)=TEMP
   30 CONTINUE
   40 CONTINUE
      GO TO 20
  100 CONTINUE
      RETURN
      END
      SUBROUTINE ASSIGN(S,N,B,RL,CL,NR,NC,RP,X,U,V,OBJ,KODE,CC)
C
C  ***************************************************
C     THIS SUBROUTINE FINDS THE SOLUTION TO LINEAR
C     ASIGNMENT PROBLEM.
C  ***************************************************
C
C      THIS IS AN ENCODING OF THE HUNGARIAN ALGORITHM
C      PROVIDED BY DR.K.G.MURTY.,TECHN. REPORT NO.73-77,
C      DEPT. OF INDUSTRIAL AND OPERATIONS ENGINEERING,
C      UNIVERSITY OF MICHIGAN.
C
      IMPLICIT INTEGER (A-Z)
      INTEGER*4 X(S),RL(S),CL(S),NR(S),NC(S),RP(S)
      DIMENSION U(S),V(S),B(S,S)
      DIMENSION CC(S,S)
C
C     VARIABLES
C     *********
C     X(J)      = ASSIGNMENT 'I TO J' INDICATED BY X(J)=I
C     U(I),V(J),= DUAL VARIABLES * ROW & COL
C     RL(I)     = ROW LABEL *-1 IF ROW I NOT ASSIGNED 0 OR POSITIVE
C                 OTHERWISE
C     CL(J)     = COL LABEL  0  OR POSITIVE
C     RP(I)     = ROW POINTER * PTS TO COL OF FIRST ADMISSABLE
C                 CELL OF ROW I
C     NNR       = NUMBER OF NEW ROWS (JUST LABELLED)
C     NR(I)     = NEW ROWS (JUST LABELLED)
C     NNC       = NUMBER OF NEW COLUMNS (JUST LABELLED)
C     NC(J)     = NEW COLUMNS (JUST LABELLED)
C     LAC       = LAST ADMISSABLE CELL (COLUMN INDEX FOR ROW BEING
C                 CONSIDERED).
C     KODE      = THE PARAMETER TO IDENTIFY THE VARIOUS SEGMENTS OF
C                 THE ALGORITHM.
?
```

```
      C     KODE      = 3:SKIP THE REDUCTION OF INITIAL COST MATRIX
      C               = 2:GO TO THE LABELLING SEGMENT OF THE ALG.
      C
      C     GENERATE INITIAL DUAL SOLUTION AND RELATIVE COST MATRIX
      C     ******************************************************
      C     SET U(I) TO MIN COST OF ROW I & ADJUST B MATRIX
            IF(KODE.EQ.3)GO TO 1051
            IF(KODE.EQ.2)GO TO 30
            DO 5 I=1,N
          5 U(I)=B(I,1)
            DO 7 I=1,N
            DO 7 J=2,N
          7 IF(B(I,J).LT.U(I))U(I)=B(I,J)
            DO 9 I=1,N
            DO 9 J=1,N
          9 B(I,J)=B(I,J)-U(I)
      C     SET V(J) TO MIN COST (ADJUSTED ABOVE) OF COL J
      C     ADJUST B MATRIX TO REFLECT RELATIVE COSTS: B(I,J)=B(I,J)-U(I)-V(J)
            DO 11 J=1,N
         11 V(J)=B(1,J)
            DO 13 J=1,N
            DO 13 I=2,N
         13 IF(B(I,J).LT.V(J))V(J)=B(I,J)
            DO 15 I=1,N
            DO 15 J=1,N
         15 B(I,J)=B(I,J)-V(J)
      C     FROM INITIAL PRIMAL SOLUTION
      C     ****************************
      C     SET INITIAL X(),CL(),RL(),RP(),ADMISSABLE CELL PTRS & NNR/NR() VALUES
      C     ********************************************************************
       1051 IF(KODE.NE.3)GO TO 1050
            DO 1052 I=1,N
            V(I)=0
       1052 U(I)=0
       1050 CONTINUE
            DO 17 J=1,N
            CL(J)=0
         17 X(J)=0
            NNR=0
            DO 25 I=1,N
            RP(I)=0
            RL(I)=-1
      C     SEARCH FOR ADMISSABLE CELL IN ROW I: B(I,J)=0
            DO 22 J=1,N
            IF(B(I,J).NE.0)GO TO 22
            IF(RP(I).NE.0)GO TO 18
            RP(I)=J
            GO TO 20
         18 B(I,LAC)=-J
         20 LAC=J
      C     IF ROW I NOT ASSIGNED (RL()=-1) AND J NOT ASSIGNED (X()=0),
      C     MAKE ASSIGNMENT
            IF(RL(I)+X(J).GE.0)GO TO 22
            X(J)=I
            RL(I)=0
         22 CONTINUE
?
```

```
      C     IF ROW I HAS NOT ASSIGNED, CONSIDER IT LABELLED (RL(I).NE.0)
            IF(RL(I).EQ.0)GO TO 25
            NNR=NNR+1
            NR(NNR)=I
         25 CONTINUE
            GO TO 35
      C     RESET VALUES FOR RL(),CL(),NNR, & NR() AFTER PRIMAL CHANGE
      C     ************************************************************
         30 NNR=0
            DO 33 I=1,N
            IF(RL(I))31,33,32
         31 NNR=NNR+1
            NR(NNR)=I
            GO TO 33
         32 RL(I)=0
         33 CL(I)=0
      C     CHECK FOR OPTIMALITY
      C     ********************
         35 IF(NNR.EQ.0)GO TO 100
      C     ATTEMPT COLUMN LABELLING FROM NEW LABELLED ROWS
      C     ***********************************************
         40 NNC=0
            DO 45 I=1,NNR
            J=RP(NR(I))
            GO TO 43
         42 J=-B(NR(I),J)
         43 IF(J.EQ.0)GO TO 45
            IF(CL(J).NE.0)GO TO 42
            CL(J)=NR(I)
      C     UNASSIGNED COL WAS LABELLED,GO TO BT
            IF(X(J).EQ.0)GO TO 70
            NNC=NNC+1
            NC(NNC)=J
            GO TO 42
         45 CONTINUE
      C     IF NO COLUMNS WERE LABELLED,GO TO NBT
            IF(NNC.EQ.0)GO TO 80
      C     ATTEMPT ROW LABELLING FROM NEW LABELLED COLUMNS
      C     ***********************************************
         50 NNR=0
            DO 55 J=1,NNC
            I=X(NC(J))
      C     IF ROW I IS ALREADY LABELLED, GO TO END OF LOOP
            IF(RL(I).NE.0)GO TO 55
            RL(I)=NC(J)
            NNR=NNR+1
            NR(NNR)=I
         55 CONTINUE
            IF(NNR.EQ.0)GO TO 80
            GO TO 40
      C     BREAKTHROUGH: INCREASE PRIMAL FLOW
         70 X(J)=CL(J)
            IF(RL(X(J)).LT.0)GO TO 75
            J=RL(X(J))
            GO TO 70
         75 RL(X(J))=0
?
```

```
      GO TO 30
C     NON BREAKTHROUGH: DUAL CHANGE
   80 MIN=2**31-1
C     IDENTIFY MINIMUM RELATIVE COST AMONG LABELLED ROWS (RL.NE.0)
C     AND UNLABELLED COLUMNS (CL.EQ.0)
      DO 811 I=1,N
      IF(RL(I).EQ.0)GO TO 811
      DO 81 J=1,N
      IF(CL(J).NE.0)GO TO 81
      IF(B(I,J).LT.MIN)MIN=B(I,J)
   81 CONTINUE
  811 CONTINUE
      NNC=0
      DO 90 I=1,N
      IF(RL(I).EQ.0)GO TO 85
C     LABELLED ROW
      U(I)=U(I)+MIN
      DO 84 J=1,N
      IF(CL(J).NE.0)GO TO 84
C     AND UNLABELLED COLUMN * ADJUST COST DUE TO DUAL CHANGE
      B(I,J)=B(I,J)-MIN
      IF(B(I,J).NE.0)GO TO 84
C     NEW ADMISSABLE CELL IDENTIFIED
      IF(NNC.EQ.0)GO TO 813
      DO 812 NCK=1,NNC
      IF(NC(NCK).EQ.J)GO TO 814
  812 CONTINUE
  813 NNC=NNC+1
      NC(NNC)=J
      NR(NNC)=I
C     UPDATE POINTERS
  814 NJ=J-1
      IF(NJ.EQ.0)GO TO 83
      DO 82 JJ=1,NJ
      IF(B(I,J-JJ).GT.0)GO TO 82
      B(I,J)=B(I,J-JJ)
      B(I,J-JJ)=-J
      GO TO 84
   82 CONTINUE
   83 B(I,J)=-RP(I)
      RP(I)=J
   84 CONTINUE
      GO TO 90
C     UNLABELLED ROW
   85 DO 89 J=1,N
      IF(CL(J).EQ.0)GO TO 89
C     AND LABELLED COL
      IF(B(I,J).LE.0)GO TO 86
C     ADJUST POSITIVE COSTS TO CONFIRM TO NEW DUAL SOLUTION
      B(I,J)=B(I,J)+MIN
      GO TO 89
C     ADMISSABLE CELL LOST HERE * ADJUST POINTERS AND COSTS
   86 NJ=J-1
      IF(NJ.EQ.0)GO TO 88
      DO 87 JJ=1,NJ
      IF(-B(I,J-JJ).NE.J)GO TO 87
?
```

```
            B(I,J-JJ)=B(I,J)
            B(I,J)=MIN
            GO TO 89
         87 CONTINUE
         88 RP(I)=-B(I,J)
            B(I,J)=MIN
         89 CONTINUE
         90 CONTINUE
C           ADJUST DUAL SOLUTION FOR LABELLED COLUMNS AT NBT
            DO 91 J=1,N
         91 IF(CL(J).NE.0)U(J)=U(J)-MIN
C           LABEL NEW COLUMN OR COLUMNS & CHECK FOR BT
            DO 92 JJ=1,NNC
            J=NC(JJ)
            CL(J)=NR(JJ)
            IF(X(J).EQ.0)GO TO 70
         92 CONTINUE
            GO TO 50
C           OPTIMALITY
C           **********
C           DETERMINE OBJECTIVE FUNCTION VALUE
C           PUT X IN ROW FORMAT & I.E. X(I)=J REPRESENT ASSIGNMENT
C            OF I TO J.
        100 OBJ=0
            DO 103 I=1,N
        103 RL(X(I))=I
            DO 105 I=1,N
            OBJ=OBJ+U(I)+V(I)
        105 X(I)=RL(I)
C           SET ADMISSABLE CELL POINTERS TO ZERO
            IF(KODE.EQ.2)RETURN
            DO 301 I=1,N
            DO 301 J=1,N
        301 CC(I,J)=B(I,J)
            DO 120 I=1,N
            J=RP(I)
        110 IF(B(I,J).EQ.0)GO TO 120
            JJ=-B(I,J)
            B(I,J)=0
            J=JJ
            GO TO 110
        120 CONTINUE
            RETURN
            END
?
```

Computer Code for Finding Least-Allocation Cost Assignment

The listing of this code is given in the succeeding pages as subroutine 'BP-COST'. Consider the reduced matrix named as 'MATRIX' obtained at the end of step 3.1 of Chapter 6. This matrix is then transformed to another matrix named 'MAT'. This is obtained by replacing the inadmissible cells by ∞ and admissible cells by the corresponding linear components of the costs when facility J is located at site I. Finally, the matrix 'MAT' is solved for all LAP which results in least-allocation cost assignment represented by IASP.

Thus, whenever, least-allocation cost assignment criterion is to be used, lines 185, 186, 187 and 188 in the main program are deleted and lines 190-193 are replaced by the statement which calls the subroutine 'BP-COST'. Further subroutine, 'BP-COST' defined by lines 1-31 in succeeding pages is added.

```
      SUBROUTINE BPCOST(N,MATRIX,IASP,ID1,IW1,I,J)
C  ***************************************************************
C     THIS SUBROUTINE FINDS THE LEAST COST-ALLOCATION ASSIGNMENT
C  ***************************************************************
C
C     N      -SIZE OF THE COST MATRIX
C     MATRIX -REDUCED MATRIX WHICH HAS SEVERAL SOLUTIONS
C     IASP   -LEAST COST-ALLOCATION ASSIGNMENT VECTOR WHEN FACILITY
C              J IS LOCATED AT SITE I
C     ID1    -DISTANCE MATRIX
C     IW1    -FLOW MATRIX
C
      DIMENSION MAT(60,60),MATRIX(60,60),IASP(60)
      INTEGER RLAB(60),CLAB(60),MOROWS(60),MORCOL(60),RK(60)
      DIMENSION U(60),V(60),CT(60,60)
      DO 1 I1=1,N
      DO 1 J1=1,N
      IF(I1.EQ.I.AND.J1.EQ.J)GO TO 5
      IF(I1.EQ.I)GO TO 6
      IF(J1.EQ.J)GO TO 6
      IF(MATRIX(I1,J1).EQ.0)GO TO 3
    6 MAT(I1,J1)=2**31-1
      GO TO 1
    5 MAT(I1,J1)=0
      GO TO 1
    3 MAT(I1,J1)=ID1(I,J1)*IW1(J,J1)
    1 CONTINUE
      CALL ASSIGN(SIZE,N,MAT,RLAB,CLAB,MOROWS,MORCOL,RK,
     1IASP,U,V,TOTAL,KODE,CT)
      RETURN
      END
?
```

Computer Code for Finding the Best Assignment

One of the computer code for determining the assignments in the order of increasing cost is developed by Metrick and Maybee [1973]. This is based on the ranking algorithm of Murty [1968]. This computer code would be quite inefficient if used for generating the assignments having the same cost. This is discussed in detail in section 4.2 of Chapter 4. Several modifications have therefore been made to use this code efficiently. A listing is given in the succeding pages.

Two parameters KODE and CC have been added. The purpose of these two parameters is to identify the nature of solution to LAP. In some cases we need a complete solution of a node and in other cases we need to utilize the previous information and skip the unneccessary or repeated computations. For example, consider the generation of permutation matrices which are within the admissible cells of the reduced matrix. In case the reduced matrix needs further reduction the generation of the permutation matrices must be seized instead of storing high cost permuation matrices and sorting the list for their cost for further generation of the matrices.

The listing of the computer code given in the succeding pages generates the alternate solutions one by one. These are then evaluated for best cost assignment using the distance and flow data for QAP. Thus

when best cost assignment criterion is to be used, lines 32, 33 and 34 from control program for finding best cost assignment are added in the main program after the line 59. Lines 185, 186, 187 and 188 of the main program are deleted and lines 190-193 are replaced by lines 53 to 83 of the control program for finding the best cost assignment. Further the subroutine RNKINT, listed in lines 89 to 343, is included.

```
      C  ************************************************************
      C     CONTROL PROGRAM FOR FINDING THE BEST COST ASSIGNMENT
      C  ************************************************************
      C     THIS ENCODING IS BASED ON ASSIGNMENT RANKING,
      C     TECHN. REPORT NO.73-77,DEPT. OF INDUSTRIAL AND
      C     OPERATIONS ENGINEERING,UNIVERSITY OF MICHIGAN.
      C
      C
      C     VARIABLES
      C  ****************
      C     S       =DIMENSIONED SIZE OF THE MATRICES
      C     M       =SIZE OF THE COST MATRIX
      C     MATC    =REDUCED COST MATRIX WHICH HAS ALTERNATE SOLUTIONS
      C     B       =COST MATRIX TO BE USED BY ASSIGNMENT ROUTINE
      C     X       =CURRENT SOLUTION VECTOR DETERMINED BY RANKING
      C     IASP    =BEST COST ASSIGNMENT.
      C     OUTCOL  =LOGICAL ARRAY FOR COLUMNS INCLUDED IN A NODE
      C     OBJECT  =COST OF ASSIGNMENT RETURNED BY RANKING
      C     UPPER   =CURRENT UPPER BOUND ON ALLOWABLE ASSIGNMENT
      C     CSHIFT  =ARRAY FOR MAPPING SUBPROBLEMS FOR ASSIGNMENT
      C              FOR RANKING
      C     EXCLUD  =LOGICAL ARRAY FOR COLUMNS DELETED FROM NODE
      C              FOR RANKING
      C
            IMPLICIT INTEGER (A-Z)
            INTEGER*4 RLABEL(60),CLABEL(60),MOROWS(60),MORCOL(60),RP(60)
            DIMENSION U(60),V(60)
            DIMENSION ID1(60,60),IU1(60,60),MATC(60,60)
            DIMENSION CT(60,60)
            INTEGER IASP(60)
            DIMENSION EXCLUD(60),OUTCOL(60)
            INTEGER X(60),B(60,60),CSHIFT(60)
            LOGICAL*1 EXCLUD,OUTCOL,COUNT
            COMMON/CONTRL/TIMES,COUNT
      C
            READ(5,12)N,NFAC
         12 FORMAT(I3,I5)
      C     READ THE DISTANCE MATRIX
            DO 102 I=1,N
            READ(5,3)(ID1(I,J),J=1,N)
          3 FORMAT(20I4)
        102 CONTINUE
      C     READ THE FLOW MATRIX
            DO 103 I=1,N
            READ(5,3)(IU1(I,J),J=1,N)
        103 CONTINUE
      C     READ THE MATRIX WITH ALTERNATE SOLUTION.
            DO 4 I=1,N
            READ(5,18)(MATC(I,J),J=1,N)
         18 FORMAT(13I5)
          4 CONTINUE
            SIZE=60
            TIMES=10000
            UPPER=0
            COUNT=.TRUE.
            NFAC=10000
?
```

```
      IF(N.GE.8)GO TO 21
      NFAC=1
      DO 20 I=1,N
   20 NFAC=NFAC*I
   21 IF(TIMES.GT.NFAC.OR.TIMES.LE.0)TIMES=NFAC
      KODE=0
      CALL ASSIGN (SIZE,N,MATC,RLABEL,CLABEL,MOROWS,MORCOL,
     *RP,X,U,V,OBJECT,KODE,CT)
      SOL=0
      UPP=2**31-1
      CALL RNKINT(SIZE,N,MATC,B,RLABEL,CLABEL,MOROWS,MORCOL,RP,
     *X,U,V,OUTCOL,OBJECT,UPPER,CSHIFT,EXCLUD,KODE,CT)
    6 SOL=SOL+1
      IF(OBJECT.GT.UPPER)GO TO 200
      CALL COST(X,N,ID1,IW1,ICOST)
      IF(ICOST.LE.UPP)GO TO 501
      GO TO 502
  501 UPP=ICOST
      DO 503 I=1,N
  503 IASP(I)=X(I)
  502 CONTINUE
      TIMES=TIMES-1
    7 IF(TIMES.EQ.0)GO TO 500
    8 CALL RANK(SIZE,N,MATC,B,RLABEL,CLABEL,MOROWS,MORCOL,RP,
     1X,U,V,OUTCOL,OBJECT,UPPER,CSHIFT,EXCLUD,KODE,CT)
      GO TO 6
  200 CONTINUE
  500 WRITE(6,5)SOL,UPP,(IASP(I),I=1,N)
    5 FORMAT(10X,'NO. OF SOL. EXPLORED',I3,2X,'BEST COST=',I3,'LAYOUT'
     1,25I3)
      STOP
      END
      SUBROUTINE RNKINT (S,N,A,B,RLABEL,CLABEL,MOROWS,MORCOL,RP,
     *X,U,V,OUTCOL,OBJECT,UPPER,CSHIFT,EXCLUD,KODE,CC)
C
C  **********************************************************
C      THIS SUBROUTINE FINDS THE BEST COST ASSIGNMENT
C  **********************************************************
      IMPLICIT INTEGER (A-Z)
      COMMON/MEMORY/FWORD(6000)
      INTEGER*2 HWORD(12000)
      INTEGER FWORD
      EQUIVALENCE (HWORD(1),FWORD(1))
      COMMON/CONTRL/TIMES,COUNT
      INTEGER*4 TIMES,X(S),RLABEL(S),CLABEL(S),MOROWS(S),MORCOL(S)
      INTEGER*4 RP(S),CSHIFT(S)
      DIMENSION A(S,S),B(S,S),U(S),V(S)
      DIMENSION CC(S,S)
      LOGICAL*1 OUTCOL(S)
      LOGICAL*1 EXCLUD(S),COUNT
C
C     INITIALIZATION OF FIRST ASSIGNMENT
C
      CALL LISINT(N,NODES)
      IF(COUNT)GO TO 8
      COUNT=.TRUE.
```

```
      TIMES=NODES
      GO TO 9
    8 IF(TIMES.GT.NODES)TIMES=NODES
    9 DO 10 I=1,M
      DO 10 J=1,N
   10 B(I,J)=A(I,J)
      OBJECT=0
      CALL GETSPC(RNKHED)
      FWORD(RNKHED)=OBJECT
      HPTR=2*RNKHED+1
      HWORD(HPTR)=0
      DO 11 J=1,N
   11 HWORD(HPTR+J)=X(J)
      HWORD(HPTR+N+1)=0
      HWORD(HPTR+N+2)=0
      GO TO 501
C
      ENTRY RANK(S,N,A,B,RLABEL,CLABEL,MOROWS,MORCOL,RP,
     *X,U,V,OUTCOL,OBJECT,UPPER,CSHIFT,EXCLUD,KODE,CC)
C
C     NOW GET THE LEAST COSTLY SOLUTION,PARTITION ITS NODE
C             AND RETURN TO SENDER
C
      INTERS=0
      NUMINC=HWORD(BESPTR+N+1)
      NUMEXC=HWORD(BESPTR+N+2)
      R=NUMINC
      NODES=N-R-1
      IF(NODES.EQ.0)GO TO 500
C
C          NOW TO GENERATE THE NODES
C
      DO 420 J=1,NODES
      INCNUM=NUMINC+J-1
      DO 404 I=1,N
      POINTR=BESPTR+I
      IF(I.GT.INCNUM)GO TO 402
      OUTCOL(HWORD(POINTR))=.TRUE.
      GO TO 404
  402 OUTCOL(HWORD(POINTR))=.FALSE.
  404 CONTINUE
      CALL GETSPC(NEWNOD)
      NEWPTR=2*NEWNOD+1
      PARTAL=NEWPTR+N+1
      HWORD(PARTAL)=INCNUM
      HWORD(PARTAL+1)=NUMEXC+1
C
C         COPY THE INCLUDED CELLS
C
      IF(INCNUM.EQ.0)GO TO 4149
      DO 415 I=1,INCNUM
      OUTCOL(HWORD(BESPTR+I))=.TRUE.
  415 HWORD(PARTAL+I+1)=HWORD(BESPTR+I)
 4149 DO 4150 I=1,N
 4150 EXCLUD(I)=OUTCOL(I)
      IOUTS=INCNUM
?
```

```
      C
      C          COPY THE EXCLUDED CELLS
      C
 4151 DEL=0
      IF(NUMEXC.EQ.0)GO TO 4171
      ROW=NUMINC+1
      DO 417 I=1,NUMEXC
      TO=PARTAL+I+INCNUM-DEL+1
      FROM=BESPTR+2+N+NUMINC+I
      HWORD(TO)=HWORD(FROM)
      COL=HWORD(TO)
      IF(ROW.LE.INCNUM)GO TO 416
      IF(OUTCOL(COL))GO TO 416
      IOUTS=IOUTS+1
      EXCLUD(COL)=.TRUE.
      GO TO 417
  416 DEL=DEL+1
  417 CONTINUE
 4171 TO=PARTAL+2+INCNUM+NUMEXC-DEL
      HWORD(TO)=HWORD(BESPTR+INCNUM+1)
      COL=HWORD(TO)
      IF(OUTCOL(COL))GO TO 418
      IOUTS=IOUTS+1
      EXCLUD(COL)=.TRUE.
      GO TO 419
  418 DEL=DEL+1
  419 HWORD(PARTAL+1)=NUMEXC-DEL+1
      IF(IOUTS.NE.N-1)GO TO 41950
      DO 41910 I=1,N
      IF(EXCLUD(I))GO TO 41910
      GO TO 41915
41910 CONTINUE
      GO TO 41950
41915 HWORD(PARTAL)=INCNUM+1
      HWORD(PARTAL+1)=0
      HWORD(PARTAL+INCNUM+2)=I
      OUTCOL(I)=.TRUE.
41950 R=HWORD(PARTAL)
      R1=R+1
      SIZE=N-R
      DO 41951 I=1,N
41951 CSHIFT(I)=0
      DEL=0
      DO 41954 COL=1,N
      IF(OUTCOL(COL))GO TO 41953
      CSHIFT(COL)=DEL
      DO 41952 ROW=R1,N
41952 B(ROW-R,COL-DEL)=A(ROW,COL)
      GO TO 41954
41953 DEL=DEL+1
41954 CONTINUE
      C
      C             SET HIGH COST FOR EXCLUDED CELLS
      C
      EXCNUM=HWORD(PARTAL+1)
      IF(EXCNUM.EQ.0)GO TO 41956
?
```

```
      FROM=PARTAL+R+1
      DO 41955 I=1,EXCNUM
      COL=HUORD(FROM+I)
      COL=COL-CSHIFT(COL)
      B(1,COL)=2**31-1
41955 CONTINUE
C
C           CHECK THE NODE FOR FEASIBILITY
C
      KODE=3
41956 CALL ASSIGN(S,SIZE,B,RLABEL,CLABEL,MOROWS,MORCOL,RP,
     *X,U,V,OBJECT,KODE,CC)
C        PLACE SOLUTION IN NODE
C
      IF(R.EQ.0) GO TO 42958
      DO 41957 I=1,R
      COL=HUORD(PARTAL+1+I)
      HUORD(NEUPTR+I)=COL
41957 CONTINUE
42958 IF(OBJECT.LE.UPPER) GO TO 41958
      CALL FRESPC(NEUNOD)
      GO TO 420
C
C       PUT SOLUTION CELL IN NODE
C
41958 PLACE=1
      DO 41959 I=1,N
      IF(OUTCOL(I))GO TO 41959
      CSHIFT(PLACE)=CSHIFT(I)
      PLACE=PLACE+1
41959 CONTINUE
      DO 41960 I=R1,N
      COL=X(I-R)+CSHIFT(X(I-R))
41960 HUORD(NEUPTR+I)=COL
      FUORD(NEUNOD)=OBJECT
      HUORD(NEUPTR)=0
C
C           LINK NODE TO RANKED LIST
C
      IF(RNKHED.NE.0) GOTO 41962
41961 RNKHED=NEUNOD
      GOTO 41970
41962 IF(FUORD(NEUNOD).GT.FUORD(RNKHED)) GOTO 41963
      HUORD(NEUPTR)=RNKHED
      GO TO 41961
41963 LAST=RNKHED
41964 NEXT=HUORD(LAST*2+1)
      IF(NEXT.EQ.0) GO TO 41966
      IF(FUORD(NEUNOD).GT.FUORD(NEXT)) GO TO 41967
41965 HUORD(NEUPTR)=HUORD(LAST*2+1)
41966 HUORD(LAST*2+1)=NEUNOD
41970 IF(.NOT.COUNT) GO TO 420
C
C     PRUNE SIZE OF LIST TO TIMES IF COUNT IS TRUE
C
      NEXT=RNKHED
?
```

```
            RANKED=1
      41969 LAST=NEXT
            NEXT=HWORD(NEXT*2+1)
            IF(NEXT.EQ.0.AND.RANKED.LE.TIMES) GO TO 420
            RANKED=RANKED+1
            IF(RANKED.GT.TIMES) GO TO 41968
            GO TO 41969
      41968 HWORD(LAST*2+1)=0
      41971 IF(NEXT.EQ.0) GO TO 420
            IF(UPPER.GT.FWORD(NEXT)) UPPER=FWORD(NEXT)
            CALL FRESPC(NEXT)
            NEXT=HWORD(NEXT*2+1)
            GO TO 41971
      41967 LAST=NEXT
            GO TO 41964
      420   CONTINUE
      500   CALL FRESPC(BEST)
      C
      C         FINALLY WE RETURN THE SOLUTION
      C
      501   BEST=RNKHED
            IF (BEST.NE.0)  GO TO 502
            OBJECT=UPPER+1
            RETURN
      502   BESPTR=2*BEST+1
            OBJECT=FWORD(BEST)
            RNKHED=HWORD(BESPTR)
            DO 510 I=1,N
      510   X(I)=HWORD(BESPTR+I)
            RETURN
            END
            SUBROUTINE LISINT(N,NODES)
      C
      C     THIS ROUTINE SIMULATES A STORAGE MANAGEMENT
      C     SYSTEM USING THE MEMORY CALLED FWORD
      C
            IMPLICIT INTEGER(A-Z)
            COMMON/MEMORY/FWORD(6000)
            INTEGER*2 HWORD(12000)
            INTEGER FWORD
            EQUIVALENCE(FWORD(1),HWORD(1))
      C
            NODSIZ=N+2
            NODES=6000/NODSIZ
            DO 2 I=1,NODES
            FPTR=(I-1)*NODSIZ+1
          2 FWORD(FPTR)=FPTR+NODSIZ
            FWORD(FPTR)=0
            FRELIS=1
            RETURN
      C
      C
            ENTRY GETSPC(PTR)
      3     PTR=FRELIS
            FRELIS=FWORD(FRELIS)
            RETURN
?
```

```
      C
      C
            ENTRY FRESPC(PTR)
            FWORD(PTR)=FRELIS
            FRELIS=PTR
            RETURN
            END
?
```

APPENDIX V

PROGRAM USER'S GUIDE

DATA CARDS: These are described in the order in which the main program reads them.

Problem size - A single card with a single number in I5 format specifying problem size.

Distance Matrix - N number of cards with max. of 20 values on each card in I4 format specifying distance between the sites.

Flow Matrix - N number of cards with 20 values on each card in I4 format specifying the flow between the facilities.

It should be noted that the data for distance and flow matrix are read in integer numbers. If the data is available as real number, it must be changed to integer number before using the algorithm. Further, if necessary the read format I4 can be changed accordingly.

APPENDIX VI

OUTPUT FOR THE SAMPLE PROBLEM

OF SIZE 30x30, AS GIVEN IN APPENDIX II

LOWER BOUND ON THE COST= 2269

ITER.	OBJECTIVE FUNCTION VALUE	LAYOUT
0	3723	26 25 23 1 2 5 14 6 8 3 19 21 17 11 10 22 30 27 12 18 7 13 29 24 15 20 16 9 4 28
1	3453	26 23 11 17 28 4 14 29 8 30 19 25 6 13 18 22 16 27 12 20 9 7 10 24 5 2 3 1 21 15
2	3352	23 4 14 11 27 25 15 16 29 30 8 20 6 3 18 7 19 28 12 21 22 10 1 17 5 2 13 9 24 26
3	3274	28 4 16 11 30 15 20 14 27 22 8 25 5 3 18 23 19 26 6 29 13 10 7 17 21 2 9 12 1 24
4	3204	28 4 30 11 25 15 20 27 19 16 8 17 14 3 9 7 18 23 21 29 13 10 22 12 5 2 6 26 24 1
5	3127	4 14 16 28 25 15 20 27 30 11 8 23 3 29 19 7 1 17 21 9 10 22 18 26 5 2 13 6 12 24
6	3109	28 4 16 11 25 15 14 27 30 8 23 17 20 3 19 7 22 1 21 29 9 10 18 24 5 2 13 26 6 12
7	3088	28 4 16 11 25 15 14 27 30 8 23 18 20 3 19 7 22 1 21 29 9 10 26 17 5 2 13 6 12 24
8	3088	28 4 16 11 25 15 14 27 30 8 23 18 20 3 19 7 22 1 21 29 9 10 26 17 5 2 13 6 12 24

FINAL COST OF ASSIGNMENT= 3088

TOTAL TIME= 72.642 SECS

?

REFERENCES

(1) Apple, J.M. and Deisenroth, M.P. [1972]. "A Computerized Plant Layout Analysis and Evaluation Technique" (PLANET). Proceedings, American Institute of Industrial Engineers, 23rd Annual Conference and Convention, Anaheim, California.

(2) Armour, G.C. and Buffa, E.S., [1963]. "A heuristic algorithm and simulative approach to relative location of facilities". Management Science, Vol. 9, No. 2, p. 294-309.

(3) Bazaraa, M.S. and Elshafei, A.N., [1977]. "On the use of ficitious bounds in tree search algorithms", Management Science, Vol. 23, No. 8, p. 904-908.

(4) _____ and _____ [1979]. "An exact branch and bound procedure for the quadratic assignment problem", Nav. Res. Log. Quart., Vol. 26, No. 1, p. 109-122.

(5) _____ and Sherali, H.D., [1979]. "New approaches for solving quadratic assignment problem", ORV (Germany, F.R.), Vol. 32, p. 29-46.

(6) _____ and _____ [1980]. "Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem", Nav. Res. Log. Quart., Vol. 27, No. 1, p.29-41.

(7) Beale, E.M.L. and Tomline, J.A. [1972]. "An integer programming approach to a class of combinatorial problems", Mathematical Programming, Vol. 3, p. 339-344.

(8) Buffa, E.S.; Armour, G.C. and Vollmann, T.E., [1964]. "Allocating facilities with CRAFT", Harvard Business Review, Vol. 42, No. 2, p. 136-159.

(9) Burkard, R.E., [1973]. "A perturbation method for solving quadratic assignment problems, paper presented at VIII Symposium for Mathematical Programming, Stanford, California.

(10) _____ and Stratmann, K.H., [1978]. "Numerical investigations on quadratic assignment problems", Nav. Res. Log. Quart., Vol. 25, No. 1, p. 129-148.

(11a) Cabot, A., and Francis, R.L., [1970]. "Solving certain quadratic minimization problems by ranking the extreme points", Ops. Res., Vol. 18, No. 1.

(11) Catherine Roucairol [1979]. "A reduction method for quadratic assignment problems", ORV (Germany, F.R.), Vol. 32, p. 183-187.

(12) Christofides, Nocos and Gerrard, M., [1976]. "Special cases of the quadratic assignment problem", Management Sciences Research Report No. 391, Carnegie-Mellon University, Pennsylvania.

(13) Conway, R.W. and Maxwell, W.L., [1961]. "A note on the assignment of facility location". The J. of Ind. Engg., Vol. 11, No. 1, p. 34-36.

(14) Cole, L.C., [1972]. "The Quadratic Capacitated Facilities Location Problem", Ph.D. Dissertation, Dept. of Industrial Engineering, State University of New York at Buffalo.

(15) Edward, H.K., Gillett, B.E. and Hale, M.E., [1970]. "Modular allocation technique" (MAT), Management Science, Vol. 17, No. 3, p. 161-169.

(16) El-Rayah, T.E. and Hollier, R.H., [1970]. "A review of plant design techniques", Int. J. Prod. Res.,Vol. 8, No. 3, p. 263-279.

(17) Elshafei, A.N., [1977]. "Hospital layout as a quadratic assignment problem", Opl. Res. Quart., Vol. 28, No. 1, p. 167-179.

(18) Ford, L.R. Jr. and Fulkerson, D.R., [1962]. "Flows in networks", Princeton University Press, New Jersey.

(19) Francis, R.L. and Goldstein, J.M., [1974]. "Location theory: a selective bibliography". Ops. Res., Vol. 22, No. 2, p. 400-410.

(20) Francis, R.L. and White, J.A., [1974]. "Facility Layout and Location; an analytical approach", Prentice Hall, Inc., Englewood Cliffs, New Jersey.

(21) Gaschutz, G.K. and Ahrens, J.H., [1968]. "Suboptimal algorithms for the quadratic assignment problem", Nav. Res. Log. Quart., Vol. 15, No. 1, p. 49-62.

(22) Gavett, J.W. and Plyter, N.V., [1966]. "The optimal assignment of facilities to locations by branch and bound", Ops. Res.,Vol. 14, No. 2, p. 210-232.

(23) Gilmore, P.C., [1962]. "Optimal and suboptimal algorithms for quadratic assignment problem". J. Soc. Ind. and Appl. Math., Vol. 10, No. 2, p. 305-313.

(24) Graves, R.J., [1974]. "Implicit Enumeration and Bounding Approach to Subclass of Location-Allocation Problems", Ph.D. Dissertation, Dept. of Operations Research, State University of New York at Buffalo.

(25) Graves, G.W. and Whinston, A.B., [1970]. "An algorithm for the quadratic assignment problem", Management Science, Vol. 17, No. 7, p. 453-471.

(26) Hannan, M. and Kurtzberg, J., [1972]. "A review of the placement and quadratic assignment problems", SIAM Review, Vol. 14, No. 2, p. 324-342.

(27) Hillier, F.S., [1963]. "Quantitative tools for plant layout analysis", J. Ind. Eng., Vol. 14, No. 1, p. 33-40.

(28) _____ and Connors, M.M., [1966]. "Quadratic assignment problem algorithms and the location of indivisible facilities", Management Science, Vol. 13, No. 1, p. 42-57.

(29) _____ and Lieberman, G.J., [1967]. "Operations Research", Holden-Day Inc., 500 Sansome Street, San Francisco, California, 94111.

(30) Hitchings, G.G. and Cottam, M., [1976]. "An efficient heuristic procedure for solving the layout design problem", Omega, The Int. J. Mgmt. Sc., Vol. 4, No. 2, p. 205-214.

(31) Karp, R.M., [1972]. "Reducibility Among Combinatorial Problems, in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, N.Y., p. 85-104.

(32) Khalil, T.M., [1973]. "Facilities relative allocation technique (FRAT)", Int. J. Prod. Res., Vol. 11, No. 2, p. 183-194.

(33) Khorshid, M.H. and Hassan, M.F., [1974]. "Computerized technique to hospital layout", A paper presented at the Annual Conference on Statistics, Cairo University.

(34) Kodres, V.R., [1959]. "Geometrical positioning of circuit elements in computer", Conference paper 1172, AIIE Fall General Meeting.

(35) Kohler, W.H., [1974]. "Characterization and theoretical comparison of branch and bound algorithms for permutation problems", J. Assoc. for Computing Machinery, Vol. 21, No. 1, p. 140-156.

(36) Koopmans, T.C. and Beckmann, M., [1957]. "Assignment problems and the location of economic activities", Econometrica, Vol. 25, No. 1, p. 53-76.

(37) Kreuzberger, H. and Weiterstadt, [1971]. "Eine methode zur bestimmung mehrerer losungen furdas zuordnangsproblem", Angewandte Informatki, Vol. 13, No. 9, p. 407-414.

(38) Kuhn, H.W., [1955]. "The Hungarian method for solving linear assignment problem", Nav. Res. Log. Quart., Vol. 2, No. 2, p. 83-98.

(39) Land, A.H., [1963]. "A problem of assignment with inter-related costs", Opl Res. Quart., Vol. 14, No. 2, p. 185-198.

(40) Lashkari, R.S. and Jaisingh, S.C., [1980]. "A heuristic approach to quadratic assignment problem", J. Opl. Res. Soc., Vol. 31, No. 9, p. 845-850.

(41) .... and _____ [1979]. "An efficient algorithm for quadratic assignment problem", ORSA/TIMS Joint National Meeting, Milwaukee, Wisconsin.

(42) Lawler, E.L., [1963]. "The quadratic assignment problem", Management Science, Vol. 9, No. 4, p. 586-599.

(43) Lee, R.C. and Moore, J.M., [1967]. "CORELAP - computerised relationship layout planning", J. Ind. Engg., Vol. 18, No. 3, p. 195-200.

(44) Little, J.D.C., Murty, K.G., Sweeney, D.W., and Karel, C., [1963]. "An algorithm for the travelling salesman problem", Ops. Res., Vol. 11, No. 6, p. 972-989.

(45) Love, R.F. and Wong, J.Y.,[1976]. "Solving quadratic assignment problems with rectangular distances and integer programming", Nav. Res. Log Quart., Vol. 23, No. 4, p. 623-627.

(46) Maybee, J.D., [1978]. "Efficent Branch and Bound Algorithms for Permutation Problems", Ph.D. Dissertation, Industrial and Operations Engineering, University of Michigan.

(47) McCormick, E.J., [1970]. Human Factors Engineering, McGraw Hill, New York.

(48) Metrick, L.B. and Maybee, J.D., [1973]. "Assignment Ranking", Technical Report No. 73-7, Dept. of Industrial and Operations Engineering, University of Michigan.

(49) Munkers, J., [1957]. "Algorithms for assignment and transportation problems", J. Siam, Vol. 5, p. 32-38.

(50) Murty, K.G., [1970]. "Some applications of the algorithm for ranking extreme points", unpublished paper, Dept. of Industrial and Operations Engineering, University of Michigan.

(51) _____ [1968]. "An algorithm for ranking all assignments in order of increasing costs", Ops. Res., Vol. 16, No. 3, p. 682-687.

(52) Muther, R. and McPherson, K., [1970]. "Four approaches to computerized layout planning", J. Ind. Engg., Vol. 2, No. 1, p. 39-42.

(53) Neghabat, F., [1974]. "An efficient equipment layout algorithm", Ops. Res., Vol. 22, No. 3, p. 622-628.

(54) Nicholson, T., [1971]. "Optimization Techniques", Longman Press, London, Business School Series.

(55) Nugent, C.E., Vollmann, T.E. and Ruml, J. [1968]. "An experimental comparison of techniques for the assignment of facilities to locations", Ops. Res., Vol. 16, No. 1, p. 150-173.

(56) Parker, C.S., [1976]. "An experimental comparison of some heuristic strategies for component placement", Opl Res. Quart., Vol. 27, No. 1, p. 71-81.

(57) _____ [1974]. "A Theoretical and Emperical Investigation of Strategies to Some Permutation Problems", Ph.D. Dissertation, Business Administration, University of Colorado.

(58) Pegels, C.C., [1966]. "Plant layout and discrete optimizing", Int. J. Prod. Research, Vol. 5,No. 1, p. 81-92.

(59) Pierce, J.V. and Crowston, W.B., [1971]. "Tree search algorithms for quadratic assignment problems", Nav. Res. Log. Quart., Vol. 18, No. 1, p. 1-36.

(60) Reiter, S. and Sherman, G.R., [1965]. "Discrete optimizing", J. Soc. for Ind. and Appl. Math., Vol. 13, No. 3, p. 864-889.

(61) Ritzman, L.P., [1972]. "The efficiency of computer algorithms for plant layout", Management Science, Vol. 18, No. 5, p. 240-248.

(62) Ross, G.T. and Soland, R.M., [1977]. "Modelling facility location problems as generalized assignment problems", Management Science, Vol. 24, No. 3, p. 345-357.

(63) Sasieni, M., Yaspan, A. and Friedman, L., [1959]. Operations Research, Wiley and Sons, New York.

(64) Seehof, J.M., and Evans, W.O., [1967]. "Automated layout design program", J. Ind. Engg., Vol. 18, No. 12, p. 690-695.

(65) Steinberg, L., [1961]. "The background wiring problem: a placement algorithm", Soc. Ind. Appl. Review, Vol. 3, No. 1, p. 37-50.

(66) Vollmann, T.E., Nugent, C.E. and Zartler, R.L., [1968]. "A oomputerized model for office layout", J. Ind. Engg., Vol. 19, No. 7, p. 321-329.

(67) Wallace, H., Hitchings, G.G. and Towill, D.R., [1976]. "Parameter estimation for distributions associated with the facility design problems", Int. J. Prod. Res., Vol. 14, No. 2, p. 263-274.

(68) Weingarten, A., [1972]. "The Analytical Design of Facilities Layout", Ph.D. Dissertation, School of Engineering and Science, New York University.

(69) Whitehead, B. and Elders, M.Z., [1964]. "An approach to the optimum layout of single storey buildings", Architect's Journal, Vol. 139k p. 1373-1380.

(70) Wimmert, R.J. [1958]. "A mathematical method of equipment location", J. Ind. Engg., Vol. 9, No. 6, p. 498-505.

(71) Zoller, K. and Adendorff, K., [1972]. "Layout planning by computer simulation", AIIE Transactions, Vol. 4, No. 2, p. 116-125.

VITA AUCTORIS

1951	Born on July 1, Varanasi, Uttar Pradash, India.
1966	Completed Secondary School Education at Central School, Banaras Hindu University, India.
1967	Completed Pre-University course at Science College, Banaras Hindu University, India.
1972	Graduated from Institute of Technology, Banaras Hindu University, India, with degree of Bachelor of Science in Mechanical Engineering with Honours.
1973	Worked as an Officer Trainee in Electricity Project and Planning Circle, Lucknow, Uttar Pradesh, India.
1975	Graduated from University of Windsor, with a degree of Master of Applied Science in Industrial Engineering.
1979 - present	Working as an Industrial Engineer with Viscount Machine and Tool, Windsor, Ontario.
1981	Candidate for the degree of Ph.D. in Industrial Engineering at the University of Windsor, Windsor, Ontario.