MACHINE LAYOUT: AN OPTIMIZATION AND KNOWLEDGE-BASED APPROACH

by

Sunderesh S. Heragu

A thesis
presented to the University of Manitoba
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Department of Mechanical Engineering and Industrial Engineering Program

Winnipeg, Manitoba

ISBN 0-315-48112-9

MACHINE LAYOUT:

AN OPTIMIZATION AND KNOWLEDGE-BASED APPROACH

BY

SUNDERESH S. HERAGU

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Sunderesh S. Heragu

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Sunderesh S. Heragu

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ACKNOWLEDGEMENTS

I wish to express deep appreciation to my thesis supervisor Dr. A.Kusiak, for his guidance. The members of my examination committee, Dr. A.S.Alfa, Dr. Y.P.Gupta, Dr. J.B.Mazzola, Dr. W.Pedrycz, and Dr. D. Scuse helped me during the preparation of the final draft of this dissertation. Their valuable advice is greatly appreciated.

I also wish to express my gratitude to the staff at the Computer Services department who provided me the services I required for conducting my research, and the University of Manitoba for awarding me a graduate fellowship to carry out my Ph.D studies.

I would like to thank my parents, sister, brother-in-law, brother and their families, who although miles away in India, have been a constant source of encouragement and help. Finally, I would like to express my gratitude to my sister Vijaya, and brother-in-law Sitaram, for their kindness and support during my stay at Winnipeg. I sincerely hope that some day I can repay at least a part of the kindness and love my family has shown to me.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this thesis, the machine layout problem in automated manufacturing systems is addressed. Four basic patterns of machine layouts which are frequently encountered in manufacturing systems are identified.

Three new models of the layout problem are presented: linear continuous with absolute values in the objective function and constraints, linear mixed integer and non-linear. The continuous models have a compact form. An advantage of the formulations presented in this thesis is that the location of sites need not be known a priori. More importantly, three of the formulations model the layout problem with machines of unequal area. Solving the models presented with a heuristic unconstrained optimization algorithm yields good quality suboptimal solutions in a relatively low computation time.

Two new heuristic construction algorithms for solving the machine layout problem are also presented. They generate solutions with acceptable quality in low computational time. One of them, called the Triangle Assignment Algorithm (TAA), is compared with existing algorithms for 8 test problems and is found to give solutions of better quality than any other construction algorithm published.

Since the models and algorithms developed in this thesis are efficient, they are embedded in a knowledge based system designed to solve the machine layout problem. The system, named KBML, combines the

optimization and expert system approaches and considers quantitative as well as qualitative factors while solving the machine layout problem. It is coded in Common LISP and implemented on a Symbolics 3650 machine.

Chapter I

INTRODUCTION

To date a large number of Flexible Manufacturing Systems(FMSs) have been implemented around the world. Some FMSs operate as independent manufacturing systems and some are integrated with the classical manufacturing systems. One of the problems encountered in the design of FMSs is the problem of layout of machines and stations, called for simplicity in this thesis, the Machine Layout Problem (MLP). Although there is a vast literature available on the facility layout problem, there are very few papers published on MLP in classical manufacturing systems. We are not aware of a single one addressing this problem in the flexible manufacturing environment.

The MLP involves the arrangement of machines on a factory floor in a way that minimizes the time (cost) required to transfer material between each pair of machines. Factors such as width of material handling carrier path, clearance between machines, etc., have to be considered while determining the layout.

The traditional approach used by layout analysts to solve the layout problem involves the following three steps:

i) formulating a model for the layout problem,

ii) solving the model using an optimal or heuristic algorithm,

iii) incorporating qualitative aspects not considered in the model and appropriate modification of the solution produced by the algorithm.

However, the models and algorithms available in the literature have certain limitations. For example, most of the models developed for the layout problem assume that the location of sites (to which facilities are to be assigned) are known a priori. The algorithms available to solve the layout problem require significantly high central processing unit (CPU) time. Moreover, the models and algorithms developed thus far are not applicable to the MLP in the flexible manufacturing environment. This is because the location of sites are not known a priori in the MLP. As a result, an attempt has been made in this thesis to develop efficient models and algorithms for the MLP. The models and algorithms developed as well as those available in the literature are embedded in a knowledge-based system designed to solve the MLP. Thus, the knowledge-based system developed combines the optimization and knowledge-based approaches to solve the MLP.

The thesis is organized as follows: In the remainder of this chapter, the MLP in a flexible manufacturing environment is addressed. The existing models, algorithms and expert systems developed for solving the layout problem are surveyed in the next chapter. In chapter 3, new models for the MLP are presented. The models are solved using a simple heuristic algorithm which is discussed in chapter 4. Another heuristic algorithm for solving the MLP is presented in chapter 5. The computational results of the two algorithms and a comparison with other well-known existing algorithms are also included in chapter 5. A

knowledge-based system named KBML is presented in chapter 6. Conclusions are drawn in the last chapter.

1.1 LAYOUT DESIGN IN FLEXIBLE MANUFACTURING SYSTEMS

Analysis of over 50 existing FMSs has shown that the layout of machines is determined by the type of material handling devices used (Kusiak, 1988). Matson and White (1982) have surveyed research in a number of material handling areas including robotics, transfer lines, warehouse layout, etc. The most commonly used material handling devices are (Heragu and Kusiak, 1988):

a) material handling robot,

b) Automated Guided Vehicle (AGV), and

c) gantry robot.

In an FMS served by the material handling robot, the arrangement of machines is determined by the robot envelope (figure 1). This type of layout has been discussed in Browne et al. (1985) in the context of an FMS cell. An AGV serves most efficiently while moving along a straight line (Muller, 1983). This technical limitation has forced designers of FMSs to arrange machines along straight lines (figure 2 and figure 3). When an AGV is to be used for material handling, it is important to consider the impact of the AGV on the track layout, material handling policy and production policy. Maxwell and Muckstadt (1982) present techniques for specifying the operational characteristics of an AGV.

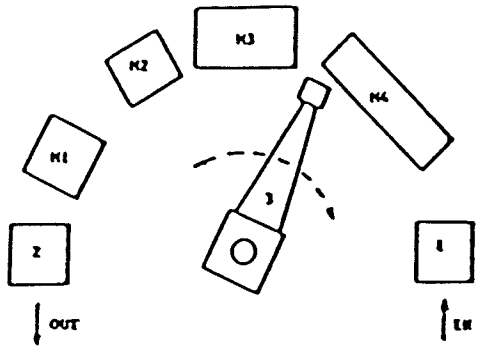
In some cases, especially where space is a limiting factor, gantry robots are used to transfer parts among the machines (figure 4). In such

cases, the geometry of the layout of machines is not important. The only limitations occuring here are of different nature, namely:

a) size of the machines,

b) working envelope of the gantry robot, and

c) access of the robot arm to the machines.

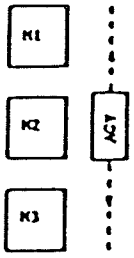


1 pallet with incoming parts
2 pallet with outgoing parts
3 material handling robot
Mi machine i

Figure 1: Circular machine layout

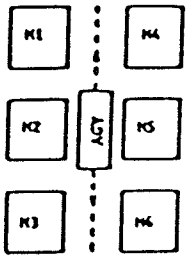
Since for each of the four discussed machine layouts, material is moved within the cell by a carrier (for eg., robot or AGV), it seems natural to arrange the machines according to the frequency of trips to be made by the carrier.

In order to determine the frequency of trips f_{ij} between two machines i and j, the following variables are defined:



AGV automated guided vehicle
Mi machine i

Figure 2: Linear single-row machine layout



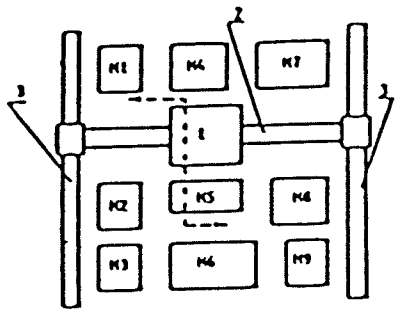
AGV automated guided vehicle
Mi machine i

Figure 3: Linear double-row machine layout

v_{ij}^{k} volume of part type k to be carried from machine i to machine j in a given time horizon (e.g., 1 year)

n_{ij} number of different part types to be carried from machine i to machine j in a given time horizon

u^{k} number of part type k to be carried in one trip of the carrier. (This number is typically determined by the capacity of the fixture, pallet, or AGV).



1 robot
2 gantry
3 gantry slides
Mi machine i

Figure 4: Multi-row machine layout

Based on the above notation,

$$f_{ij} = \left\lceil \sum_{k=1}^{n} (v_{ij}^{k} / u^{k}) \right\rceil \qquad (1)$$

where $\lceil \bullet \rceil$ is the smallest integer greater than or equal to $\bullet$.

The above formula (1) has been developed under the assumption that each batch size run in an FMS is always at least equal to the minimum value of one of the three typical limiting capacities: a fixture, a pallet or an AGV. In the case where the fixture creates the limiting capacity, this assumption almost always holds. This is due to the fact that running parts in batches of size smaller than the capacity of the fixture requires modification to the machining control program.

Chapter II

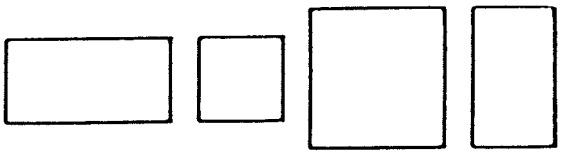
LITERATURE SURVEY

In this chapter, the models, algorithms and expert systems for solving the layout problem are surveyed. Since considerable research has been done on the related facility layout problem, various formulations of the facility layout problem and the algorithms for solving it are presented.

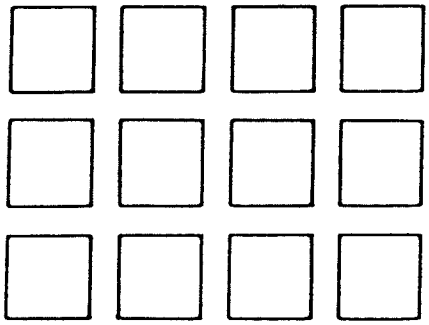
To date, a number of survey papers on the facility layout problem have been published. Wilson (1964) reviewed various facility design models applied to material flow network problems, communication network problems, etc. El-Rayah and Hollier (1970) discussed three types of facility layouts commonly seen in manufacturing plants and reviewed optimal and suboptimal algorithms for solving the Quadratic Assignment Problem (QAP). Another survey by Hanan and Kurtzberg (1972), reviewed algorithms for solving the QAP. Pierce and Crowston (1971) surveyed optimal algorithms for solving the QAP. Burkard and Stratman (1978) extended the survey of Pierce and Crowston (1971) to include suboptimal algorithms. Moore (1974) summarized the research done on the facility layout problem in Europe and North America. His survey was based on the response to a questionnaire sent to the authors of various facility layout algorithms. The survey of Foulds (1983) placed special emphasis on graph theoretic techniques but also reviewed other optimal and suboptimal algorithms. Levary and Kalchik (1985) compared some suboptimal algorithms on the basis of their characteristics and

features. As mentioned previously, four patterns of machine layout, namely: circular single-row, linear single-row, linear double-row and multi-row, can be identified in automated manufacturing systems. For the purpose of modelling the layout problem, however, only two patterns of layout, namely, single-row, in which machines are arranged in one row and multi-row, in which machines are arranged linearly in two or more rows, need to be considered. This is because, among the four patterns of layouut shown in figures 1-4, the circular single-row and linear single-row layouts are special cases of the single-row layout pattern. The linear double-row layout is a special case of the multi-row layout. A sample single-row and multi-row layout are shown in figure 5.

In the literature, the single-row and multi-row layout problem are also known as the one-dimensional and two-dimensional space allocation problem, respectively (Simmons, 1969). A special case of the single-row layout problem, i.e., when all machines are of the same length, is known as the linear ordering problem (Adolphson and Hu, 1973).



(a) Single-row layout



(b) Multi-row layout

Figure 5: Sample machine layout patterns

2.1 MODELS AND ALGORITHMS FOR THE SINGLE-ROW LAYOUT PROBLEM

Love and Wong (1976a) presented a linear mixed integer programming model for the single-row layout problem and solved it using the IBM MIP code (IBM, 1974). Simmons (1969) developed a branch-and-bound algorithm for the single-row facility layout problem. Dynamic programming algorithms have been developed by Karp and Held (1967) and Beghin-Picavet and Hansen (1982). Picard and Queyranne (1981) extended the dynamic programming algorithm of Karp and Held (1967). All the above algorithms have rather high computational time and memory requirements. Picard and Queyranne (1981) reported that a 11-facility layout problem required

less than a second of CPU time and 100k memory on an IBM 360/75. But for larger layout problems, for example, the 20-facility layout problem, they indicated that the dynamic programming algorithm would require excessively high computation time and memory.

2.2 MODELS FOR THE MULTI-ROW LAYOUT PROBLEM

The facility layout problem has been modelled as (Kusiak and Heragu, 1987):

- quadratic assignment problem
- quadratic set covering problem
- integer programming problem
- mixed integer programming problem
- graph theoretic problem.

2.2.1 Quadratic Assignment Model

Koopmans and Beckmann (1957) were the first to model the problem of locating plants with interplant flows between them. They modelled this problem as a QAP. The name was so given because the objective function is a second degree polynomial function of the variables and the constraints are linear functions of the variables. The following were defined:

n total number of plants/locations

a_{ij} net revenue from operating plant i at location j

f_{ik} flow of material from plant i to plant k

c_{jl} cost of transporting a unit material from location j to location l

$$x_{ij} = \begin{cases} 1 & \text{if plant } i \text{ is at location } j \\ 0 & \text{otherwise} \end{cases}$$

Koopmans and Beckmann (1957) assumed that:

- a_{ij} includes gross revenue minus cost of primary input but does not include the transportation cost of material between plants,
- f_{ik} is independent of the locations of the plants, and
- c_{jl} is independent of the plants and that it is cheaper to transport material directly from plant i to plant k than through a third location.

The QAP (as developed by Koopmans and Beckmann, 1957) is to:

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik} c_{jl} x_{ij} x_{kl} \qquad (1)$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \qquad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j=1,\ldots,n \qquad (3)$$

$$x_{ij} = 0 \text{ or } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,n \qquad (4)$$

However, if a_{ij} is the cost of locating and operating plant i at location j instead of the net revenue of operating plant i at location j, then (1) can be restated as:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik} c_{jl} x_{ij} x_{kl} \qquad (1a)$$

Equations (1a), (2)-(4) can be used to model the facility layout

problem by redefining a_{ij}, f_{ik} and c_{jl} as:

a_{ij} fixed cost of locating facility i at location j

f_{ik} flow of material between facility i and facility k

c_{jl} cost per unit flow of material between location j and location l.

Lawler (1963) introduced the parameter b_{ijkl}, where:

$$b_{ijkl} = \begin{cases} f_{ik} c_{jl} + a_{ij}, & \text{if } i=k \text{ and } j=l \\ f_{ik} c_{jl}, & \text{if } i \neq k \text{ or } j \neq l \end{cases}$$

and redefined the objective function (1a) as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} b_{ijkl} x_{ij} x_{kl} \qquad (1b)$$

In the above formulation $i \neq k$ implies $j \neq l$, $j \neq l$ implies $i \neq k$, $i=k$ implies $j=l$ and $j=l$ implies $i=k$ due to constraints (2) and (3). Also, the number of facilities is assumed to be equal to the number of locations. However, for some problems, as in the backboard wiring problem (Steinberg, 1961), the number of facilities m may be less than the number of locations n (i.e., $m<n$). Such problems can still be formulated as the QAP by introducing dummy facilities $1,\ldots,n-m$ and setting the flow values from these dummy facilities to all other facilities equal to zero.

If the a_{ij}'s are equal to zero or are identical, then the objective function (1a) reduces to:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} f_{ik}\, c_{jl}\, x_{ij}\, x_{kl} \qquad (1c)$$

Although a number of variations of the objective function have been proposed, the model involving objective function (1c) and constraints (2)-(4) is referred to as the QAP. A special case of a variant of the QAP, i.e., the model involving objective function (1a) and constraints (2)-(4), is shown below.

- Linear Assignment Problem:

If the f_{ik}'s are equal to zero or are identical then the objective function (1a) reduces to:

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

The equations (1d), (2)-(4) represent a linear assignment problem.

The QAP with objective function (1a), and constraints (2)-(4) has been frequently used to model the facility layout problem (Bazaraa 1975, Burkard and Stratman, 1983). However this does not mean that all facility layout problems can be formulated as a QAP. For example, consider the machine layout problem in which the locations of the machines are not known a priori. Such problems cannot be formulated as the QAP because the distance between the locations cannot be determined. The distance between two locations j and l depends on the sequence of arrangement of all the other machines.

This situation does not arise in layout problems in which the facilities are all of equal area, because the locations are all of the same area and hence the distance between any two locations is independent of the facilities assigned to those locations. Therefore, the distance between location pairs does not change from one facility arrangement to another.

There are two formulations for the layout problem with facilities of unequal area. The first is:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}^{K} c_{jl} x_{ij} x_{kl} \qquad (5)$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \qquad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j=1,\ldots,n \qquad (3)$$

$$x_{ij} = 0 \text{ or } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,n \qquad (4)$$

where c_{jl}^{K} is the transportation cost of a unit material from location j to location l under layout arrangement K.

Note that K ranges over the set of all potential layout arrangements. The number of potential layout arrangements depends upon the area of the facilities and need not necessarily be n! (Armour and Buffa, 1963). In fact, in most practical cases, the number of feasible layout arrangements to be evaluated is much less than n!. This is because, two or more layouts may be symmetrical and only one of these layouts needs to be evaluated.

2.2.2 Quadratic Set Covering Model

The second formulation for the general facility layout problem is a quadratic set covering problem (QSP) (Bazaraa, 1975). In the QSP formulation, the total area occupied by all the facilities is divided into a number of blocks. The following are defined:

q — number of blocks into which the total area occupied by all facilities is divided into

$I(i)$ — number of potential locations for facility i

$J_i(j)$ — set of blocks occupied by facility i if it is assigned to location j

$d(j_i, l_k)$ — distance between the centroids of locations j and l if facility i is assigned to location j and facility k is assigned to location l

$$x_{ij} = \begin{cases} 1 \text{ if facility } i \text{ is assigned to location } j \\ 0 \text{ otherwise} \end{cases}$$

$$p_{ijt} = \begin{cases} 1 \text{ if block } t \in J_i(j) \\ 0 \text{ otherwise} \end{cases}$$

The QSP is to:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{I(i)} a_{ij} x_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{I(i)} \sum_{k=1}^{n} \sum_{l=1}^{I(k)} f_{ik} d(j_i, l_k) x_{ij} x_{kl} \qquad (6)$$

$$\text{s.t.} \quad \sum_{j=1}^{I(i)} x_{ij} = 1 \qquad i=1,\ldots,n \qquad (7)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{I(i)} p_{ijt} x_{ij} \leq 1 \qquad t=1,\ldots,q \qquad (8)$$

$$x_{ij} = 0 \text{ or } 1 \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,I(i) \end{matrix} \qquad (9)$$

Constraint (7) ensures that each facility is assigned to exactly one location and constraint (8) ensures that each block is occupied by at most one facility.

Since the distance between locations is taken to be from centroids of the locations, Bazaraa (1975) suggested an alternate measure for the flow between facilities:

$f'_{ik} = f_{ik} / s_i s_k$, where:

s_i is the number of blocks occupied by facility i.

He also defined d'_{jl} as the distance between blocks j and l.

Using the above, the following generalized QAP is obtained.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{q} a_{ij} x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{q}\sum_{k=1}^{n}\sum_{l=1}^{q} f_{ik} d'_{jl} x_{ij} x_{kl} / s_i s_k \qquad (10)$$

$$\text{s.t.} \sum_{j=1}^{q} x_{ij} = s_i \qquad i=1,\ldots,n \qquad (11)$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \qquad j=1,\ldots,q \qquad (12)$$

$$x_{ij} = 0 \text{ or } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,q \qquad (13)$$

Although the above model can be used to formulate the layout problem with facilities of unequal area, a disadvantage is that the problem size increases as the total area occupied by all the facilities is divided into smaller blocks (Bazaraa, 1975). The same can be said about a suggestion of Hillier and Connors (1966) that for such layout problems, the facilities can be partitioned into subfacilities so that all the subfacilities are of equal area.

2.2.3 Equivalent Integer Programming Formulations of the QAP

In addition to the QAP and the QSP, there are several integer programming formulations for the facility layout problem. These are 0,1 integer programming models which are equivalent to the QAP. Lawler (1963) was the first to formulate the facility layout problem as an integer programming problem equivalent to the QAP.

By defining:

$$y_{ijkl} = x_{ij}\, x_{kl} \tag{14}$$

the QAP (1b), (2)-(4) can be represented as an integer programming problem.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} b_{ijkl}\, y_{ijkl} \tag{15}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j=1,\ldots,n \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} y_{ijkl} = n^2 \tag{16}$$

$$x_{ij} + x_{kl} - 2y_{ijkl} \geq 0 \qquad i,j,k,l=1,\ldots,n \tag{17}$$

$$x_{ij} = 0 \text{ or } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,n \tag{4}$$

$$y_{ijkl} = 0 \text{ or } 1 \qquad i,j,k,l=1,\ldots,n \tag{18}$$

Lawler (1963) proved that the above integer programming problem and the QAP are equivalent. Note that the QAP has n^2 variables x_{ij} and $2n$

constraints whereas the integer programming problem has n^2 variables x_{ij}, n^4 variables y_{ijkl} and n^4+2n+1 constraints. In the above and following comparisons, nonnegativity constraints have been excluded.

Kaufman and Broeckx (1978) developed a mixed integer linear program which has the smallest number of variables and constraints amongst all integer programming formulations of the QAP. They defined:

$$w_{ij} = x_{ij} \sum_{k=1}^{n} \sum_{l=1}^{n} b_{ijkl} x_{kl} \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,n \end{matrix} \qquad (26)$$

$$e_{ij} = \sum_{k=1}^{n} \sum_{l=1}^{n} b_{ijkl} \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,n \end{matrix} \qquad (27)$$

The objective function is:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} b_{ijkl} x_{ij} x_{kl} =$$

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \left(\sum_{k=1}^{n} \sum_{l=1}^{n} b_{ijkl} x_{kl} \right) =$$

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \qquad (28)$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \qquad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j=1,\ldots,n \qquad (3)$$

$$e_{ij} x_{ij} + \sum_{k=1}^{n} \sum_{l=1}^{n} b_{ijkl} x_{kl} - w_{ij} \leq e_{ij} \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,n \end{matrix} \qquad (29)$$

$$w_{ij} \geq 0 \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,n \end{matrix} \qquad (30)$$

$$x_{ij} = 0 \text{ or } 1 \qquad \begin{matrix} i=1,\ldots,n \\ j=1,\ldots,n \end{matrix} \qquad (4)$$

The equivalence of the above mixed integer programming problem (28)-(30) and (2)-(4) and the QAP is proved in Kaufman and Broeckx (1978) and Burkard (1984). Note that the above formulation involves n^2 zero-one and n^2 continuous variables and n^2+2n constraints. Other equivalent linear integer programs of the QAP have been given by Balas and Mazzola (1980), Bazaraa and Sherali (1980), Burkard and Bonniger (1983) and Frieze and Yadegar (1983). The mixed-integer linear program of Bazaraa and Sherali (1980) is discussed below. They defined:

$$g_{ijkl} = [a_{ij} + a_{kl}/(m-1)] + f_{ik}\, d_{jl} + f_{ki}\, d_{lj}$$

$$y'_{ijkl} = x_{ij}\, x_{kl} \qquad i=1,\ldots,n-1;\ k=i+1,\ldots,n;\ l,j=1,\ldots,n,\ l\neq j$$

The linear mixed-integer program equivalent to the QAP is:

$$\min \sum_{i=1}^{n-1}\sum_{j=1}^{n}\sum_{k=i+1}^{n}\sum_{l=1}^{n} g_{ijkl}\, y'_{ijkl} \tag{31}$$

$$\text{s.t.} \sum_{k=i+1}^{n}\sum_{\substack{l=1\\ l\neq j}}^{n} y'_{ijkl} - (n-i)x_{ij} = 0 \qquad i=1,\ldots,n-1;\ j=1,\ldots,n \tag{32}$$

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq l}}^{n} y'_{ijkl} - (k-1)x_{kl} = 0 \qquad k=2,\ldots,n;\ l=1,\ldots,n \tag{33}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j=1,\ldots,n \tag{3}$$

$$x_{ij} = 0 \text{ or } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,n \tag{4}$$

$$y'_{ijkl} \leq 1 \qquad i=1,\ldots,n-1;\ k=i+1,\ldots,n;\ j,l=1,\ldots,n,\ j\neq l \tag{34}$$

$$y'_{ijkl} \geq 0 \qquad \begin{matrix} i=1,\ldots,n-1 \\ k=i+1,\ldots,n \\ j,l=1,\ldots,n,\ j\neq l \end{matrix} \qquad (35)$$

Note that the above integer program has n^2 integer variables, $n^2(n-1)^2/2$ continuous variables, and $2n^2$ constraints. The equivalence of the above integer program and the QAP is given in Bazaraa and Sherali (1980).

2.2.4 Mixed Integer Programming Model

Love and Wong (1976) proposed a simple integer programming formulation for the layout problem in which:

- the locations are given as points on a two-dimensional plane,
- transportation costs are proportional to weighted rectangular distances.

They used the following notation to formulate the integer programming model (19)-(25) and (2)-(4):

h^r_{ik} = { horizontal distance between facilities i and k when facility i is located to the right of facility k; 0 otherwise }

h^l_{ik} = { horizontal distance between facilities i and k when facility i is located to the left of facility k; 0 otherwise }

v^a_{ik} = { vertical distance between facilities i and k when facility i is located above facility k; 0 otherwise }

v^b_{ik} = { vertical distance between facilities i and k when facility i is located below facility k; 0 otherwise }

$(\bar{x}_i, \bar{y}_i)$ location of facility i

$$x_{ij} = \begin{cases} 1 \text{ if facility } i \text{ is assigned to location } j \\ 0 \text{ otherwise} \end{cases}$$

Their linear integer programming formulation of the QAP is:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij} + \sum_{i=1}^{n-1} \sum_{k=i+1}^{n} f_{ik} (h_{ik}^{r} + h_{ik}^{l} + v_{ik}^{a} + v_{ik}^{b}) \tag{19}$$

$$\text{s.t. } h_{ik}^{r} - h_{ik}^{l} = \bar{x}_i - \bar{x}_k \qquad i=1,\ldots,n-1;\ k=i+1,\ldots,n \tag{20}$$

$$v_{ik}^{a} - v_{ik}^{b} = \bar{y}_i - \bar{y}_k \qquad i=1,\ldots,n-1;\ k=i+1,\ldots,n \tag{21}$$

$$\bar{x}_i + \bar{y}_i = \sum_{j=1}^{n} (\bar{x}_j + \bar{y}_j) x_{ij} \qquad i=1,\ldots,n \tag{22}$$

$$\bar{x}_i - \bar{y}_i = \sum_{j=1}^{n} (\bar{x}_j - \bar{y}_j) x_{ij} \qquad i=1,\ldots,n \tag{23}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j=1,\ldots,n \tag{3}$$

$$x_{ij} = 0 \text{ or } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,n \tag{4}$$

$$h_{ik}^{r},\ h_{ik}^{l},\ v_{ik}^{a},\ v_{ik}^{b} \geq 0 \qquad i=1,\ldots,n-1;\ k=i+1,\ldots,n \tag{24}$$

$$\bar{x}_i,\ \bar{y}_i \geq 0 \qquad i=1,\ldots,n \tag{25}$$

From the above formulation it can be seen that the locations of facilities are specified by rectangular coordinates. Also, constraints (22) and (23) uniquely specify the location of a facility. The above

formulation has n^2 integer variables and n^2+3n constraints. Computational experience for the above formulation indicates that it is not suitable for problems with nine or more facilities (Love and Wong, 1976).

Ritzman et al. (1979) formulated a large mixed-integer goal programming model for assigning offices in buildings. They also developed a computer program to evaluate the performance of solutions with respect to six conflicting objectives.

2.2.5 Graph Theoretic Model

In graph theoretic formulations it is assumed that the desirability of locating each pair of facilities adjacent to each other is known (Foulds and Robinson, 1976). In order to formulate the layout problem as a graph theoretic model, the following notation is used:

$G=(V,E)$	is a weighted graph with V as a nonempty set of vertices (facilities), E as a set of edges disjoint from V
w_{ij}	closeness rating indicating desirability of locating facility i adjacent to facility j
V	set of facilities
N	set of pairs of facilities which must be adjacent in any feasible solution
F	set of pairs of facilities which must not be adjacent in any feasible solution

$$E' = \{\{i,j\}: x_{ij} = 1, \{i,j\} \in E\}$$

$$x_{ij} = \begin{cases} 1 \text{ if facility i is adjacent to facility j} \\ 0 \text{ otherwise} \end{cases}$$

The graph-theoretic formulation is:

$$\max \sum_{i \in E} \sum_{j \in E} w_{ij} x_{ij} \qquad (36)$$

$$\text{s.t. } x_{ij} = 1 \qquad \{i,j\} \in N \qquad (37)$$

$$x_{ij} = 0 \qquad \{i,j\} \in F \qquad (38)$$

$$(V,E' \cup N) \text{ is a planar graph} \qquad (39)$$

A planar graph is such that it can be mapped onto a plane without any two of its edges intersecting. The reader not familiar with graph theory is referred to Harary (1969) or Bondy and Murty (1976).

In addition to the above mentioned models, Rosenblatt (1979) developed a model which minimizes the transportation cost of material and maximizes a closeness rating measure. Note that the objectives of minimizing transportation cost of material and maximizing a closeness rating measure are conflicting objectives. Rosenblatt (1979) and Dutta and Sahu (1982) developed heuristic algorithms to solve the model.

2.3 ALGORITHMS FOR THE MULTI-ROW LAYOUT PROBLEM

Since the late 1950's a number of algorithms have been developed to solve the facility layout problem. These algorithms may be classified as (Kusiak and Heragu, 1987):

- optimal algorithms
- suboptimal algorithms.

2.3.1 Optimal Algorithms

During the early 1960's a considerable amount of research was done in developing optimal algorithms for the QAP. These algorithms may be divided into two classes:

- branch-and-bound algorithms
- cutting plane algorithms.

The QAP is NP-complete. Moreover, computational experience with the QAP reported in the literature has indicated that it is a very difficult problem to solve. For example, the largest problem for which an optimal solution has been found is the layout problem with 15 facilities (Burkard, 1984).

2.3.1.1 Branch and bound algorithms

The first two branch-and-bound algorithms were independently developed by Gilmore (1962) and Lawler (1963). The main difference between the independent work of Gilmore (1962) and Lawler (1963) is in computing the lower bounds. Both the algorithms implicitly evaluate all potential solutions. Pierce and Crowston (1971) refer to this type of enumeration as controlled enumerative technique. If no bounds were considered for pruning the decision tree in the above two algorithms, then the procedure would have led to a computationally inefficient complete enumeration technique.

In addition to the Lawler (1963) and Gilmore (1962) algorithms, two other algorithms were developed by Land (1963) and Gavett and Plyter (1966). These algorithms assign pairs of facilities to pairs of locations whereas the algorithms of Gilmore (1962) and Lawler (1963) assign single facilities to single locations.

The above mentioned optimal branch-and-bound algorithms proceed on the basis of stage by stage assignment of facilities to locations. For

more details on branch-and-bound algorithms, the reader may refer to Balas (1965). Pierce and Crowston (1971) discussed an algorithm which proceeds on the basis of stage by stage exclusion of pairs of assignments from a solution to the problem. All the optimal algorithms discussed have high memory and computational time requirements (Burkard, 1984). Lavallee and Roucairol (1985) suggested the use of parallel branch-and-bound algorithms for solving the QAP optimally. Such branch-and-bound algorithms search in parallel through a number of parts of the decision tree. However, the computational results reported in Lavallee and Roucairol (1985) indicate that the parallel branch-and-bound algorithm requires high computation time for layout problems with twelve or more facilities.

Graves and Whinston (1970) developed a heuristic algorithm which is based on the fact that one can determine bounds using statistical properties of the objective function. These bounds are used in an enumerative procedure which develops suboptimal solutions.

Burkard (1973) proposed an optimal algorithm for solving the QAP based on the reduction of a square matrix. The reduction of a matrix refers to the transformation of a matrix A to another matrix A' of nonnegative elements in which there is at least one zero in each row and each column. Reductions were applied to the travelling salesman problem by Little et al. (1963). Reduction is applied to the QAP in order to improve the quality of the bound by reducing the magnitude of the quadratic term in the objective function and augmenting the influence of the linear term.

Bazaraa (1975) developed a branch-and-bound algorithm for the layout problem with facilities of unequal area. At each stage of the algorithm a partial layout P is available. A lower bound LB on the cost of all possible completions of the partial layout P is determined. If LB is less than the cost of the best available layout C', the algorithm proceeds forward with the assignment of a new facility and thereby increasing the size |P| of the partial layout. Otherwise the forward search along this path is terminated, the last assignment is prohibited and a new assignment is sought. The search continues by using the above procedure until a complete layout is obtained.

Bazaraa and Elshafei (1979) proposed a branch-and-bound algorithm for the QAP which is based upon the stage by stage assignment of single facilities to unoccupied locations. Kaku and Thompson (1986) provided another branch-and-bound algorithm which performs better than Lawler's (1963) algorithm, particularly for problems of larger size.

2.3.1.2 Cutting plane algorithms

Bazaraa and Sherali (1980) developed a cutting plane algorithm based on Benders' partitioning scheme. Burkard and Bonniger (1983) also developed a cutting plane method to solve the QAP.

The optimal branch-and-bound and cutting plane algorithms have a high CPU time and storage requirements. For example, the largest problem solved optimally by a cutting plane algorithm is the layout problem with eight facilities. A common experience with the optimal algorithms is that the optimal solution is found early in the branching process but is not verified until a substantially high number of solutions have been

enumerated (Burkard and Stratman, 1978 and Bazaraa and Kirca, 1983). This prompted researchers to terminate the branch-and-bound process prematurely without verifying optimality and resulted in heuristic branch-and-bound algorithms.

Burkard (1984) listed two criteria for the premature termination of the branch-and-bound process. They are premature termination based on:

- time limits, i.e., the enumeration process is stopped after a predetermined time limit is exceeded,
- quality of upper bounds, i.e., after a certain length of time if there is no improvement in the solution, the upper bound is decreased by a certain percentage.

2.3.2 Suboptimal Algorithms

The optimal algorithms discussed in the previous section have the following disadvantages:

- memory and CPU time requirement is high,
- large scale problems cannot be solved optimally.

The models presented in this chapter are computationally complex. As mentioned previously, the QAP, for example, is NP-complete (Sahni and Gonzalez, 1976). Burkard (1984) reported on computational results with the QAP. To find an optimal solution to the fifteen facility problem in Nugent at al. (1968), almost 50 minutes of CPU time was required on a CDC CYBER 76. Among the eight test problems in Nugent et al. (1968), the largest problem for which an optimal solution was found was the fifteen facility problem. Burkard (1984) also reported that the FORTRAN

branch-and-bound code for the QAP requires $n^3+5.5n^2+17.5n$ words of memory.

As a result, researchers concentrated on developing suboptimal algorithms for solving the layout problem. Some of the earlier methods used flow charts, process charts and the experience and knowledge of the facility analyst to determine layouts. Other methods used the relationship chart to determine the layout. The relationship chart shows the closeness desired between pairs of facilities and the concept was first introduced by Muther (1955). The closeness desired between pairs of facilities is represented in the relationship chart by values A, E, I, O, U, and X. For any pair of facilities (i,j), the values A, E, I, O, U and X indicate that the closeness between facilities i and j is absolutely necessary, especially important, important, ordinary, unimportant and undesirable, respectively. The relationship chart formed the basis for the development of a popular method called systematic layout planning (Muther, 1973). Wimmert (1958) presented a mathematical method for the facility layout problem which uses the criteria of minimizing the product of flow values and distances between all combinations of facilities. The theorem upon which Wimmert's method was based was proved to be incorrect using a counter example by Conway and Maxwell (1961). Buffa (1955) proposed another method called the sequence analysis which is based on the analysis of the sequence of operations of parts in a plant. In addition to the above, there were some other methods developed in the late 1950's and early 1960's which did not provide solutions of good quality. These methods are discussed in Foulds (1983) as schematic methods and systematic methods.

Since the earlier methods did not provide solutions of good quality, researchers began to develop new algorithms which can be classified into:

- construction algorithms
- improvement algorithms
- hybrid algorithms
- graph theoretic algorithms.

2.3.2.1 Construction algorithms

In construction algorithms a solution is constructed ab initio. In other words, facilities are assigned to a site, usually one at a time, until the complete layout is obtained.

In a survey, Moore (1974) found that there were twice as many construction algorithms as improvement algorithms. Some of the more popular construction algorithms are discussed below.

HC66: Hillier and Connors (1966) suggested a construction algorithm and an improvement algorithm based on an earlier algorithm by Hillier (1963). These three algorithms were termed as HC66, H63 and HC63-66 by Nugent et al. (1968). H63 and HC63-66 are discussed in the next section. HC66 is a modification of the Gilmore (1962) algorithm. In both the algorithms, at any stage k, k facilities are assigned to k locations. Given these k assignments, the Gilmore (1962) and HC66 algorithms calculate a lower bound associated with assigning each of the (n-k) unassigned facilities i to each of the unused locations j. Each of of these lower bounds is entered as the elements of a matrix H (whose rows represent unassigned facilities and columns represent unused

locations). While making the $(k+1)^{th}$ assignment, an element (i,j) of the matrix H is selected and facility i is assigned to location j. The difference between the Gilmore (1962) and HC66 algorithms is the criterion used for selecting an element (i,j) of the matrix H. HC66 uses the criterion suggested by Vogel's approximation method for solving transportation problems, whereas Gilmore (1962) suggests two criteria. In the first criteria, the minimum of each row and column of H is determined and the maximum of these minimums is selected. In the second criteria, the assignment problem for H is solved and the largest of the n-k elements of H appearing in the assignment problem solution is selected.

ALDEP: ALDEP (Seehof and Evans, 1967) randomly selects a facility and assigns it to the upper left corner of the layout. The next facility selected for assignment is the one which has a relationship that is greater than or equal to a user specified relationship, with a randomly selected first facility. If more than one such facility exists, then one of these is randomly selected for assignment. If there are no such facilities, the second facility to be assigned is selected randomly. This procedure is repeated until all the facilities have been assigned. Note that the facility to be assigned at the n^{th} step depends upon its relationship with the facility assigned at the $(n-1)^{th}$ step.

CORELAP: CORELAP (Lee and Moore, 1967) uses the total closeness rating of each facility to determine a layout. The total closeness rating of a facility i is equal to the sum of the numerical values of

the relationships of facility i with all other facilities, obtained from the relationship chart. Unlike ALDEP which randomly selects the first facility to be assigned, CORELAP selects the first facility depending upon its total closeness rating value. The facility with the highest total closeness rating is selected and assigned to the centre of the layout. The subsequent facilities are then added to the layout depending upon their relationships to the facilities already assigned. For example, at stage n in the assignment process, the relationship with the first assigned facility is selected for the n^{th} assignment. If no such facility exists, then the relationship chart is scanned again and the facility which has the highest relationship with the second assigned facility is selected for the n^{th} assignment and so on.

RMA Comp I: Like CORELAP, RMA Comp I (Muther and McPherson, 1970) selects the facility which has the highest closeness rating and places it in the centre of the layout. The subsequent facilities are then added to the layout depending upon their relationships to the assigned and unassigned facilities. For example, when placing a facility i in the layout, sufficient space is left for unassigned facility j which has a high closeness rating with facility i. At every stage, the relationship chart is scanned to make sure that the desirability (undesirability) of locating pairs of facilities adjacently (not adjacently) is satisfied.

MAT: MAT (Edwards et al., 1970) ranks pairs of facilities according to their flow values and location pairs according to their distance values and uses this information to determine a layout. It allows the

user to assign facilities to any desired location. The authors of MAT found that by combining the output of MAT with that of CRAFT (Armour and Buffa, 1963), which is an improvement algorithm, resulted in good quality solutions with less computational time (for eight test problems commonly used in the literature) when compared to the solution quality and computational time of CRAFT alone.

PLANET: The assignment of facilities in PLANET (Deisenroth and Apple, 1972) proceeds in three stages. In the first stage, the cost of unit flows between each pair of facilities is determined. Associated with each facility is a priority number ranging from 1 (highest) to 9 (lowest) which determines the order in which a facility can enter the layout. The priority number and the cost per unit flow between each pair of facilities form the basis for the selection of the order in which the facilities are to enter the layout. This selection of the order of facilities constitutes the second stage. For selecting this order, there are three algorithms and the user has the choice of selecting one of them. The third stage of PLANET consists of placing facilities in the layout in the order in which they were selected in stage 2.

LSP: LSP (Zoller and Adendorff, 1972) consists of a simulator which generates the sequence in which facilities are to be placed in a layout and a construction mode which determines a two-dimensional layout for the sequence generated by the simulator.

The simulator is a pseudo-random number generator and its output is converted into a biased-random sequence of facilities. The construction

mode converts the sequence of facilities into a layout. The layout is then evaluated and further checks determine whether to stop the program. In some respects LSP is similar to ALDEP and allows flexibility in terms of the applications to which it can be used. However, the flexibility is at the expense of a relatively higher computational effort.

Linear placement algorithm: Neghabat (1974) developed a Linear Placement Algorithm (LPA) for solving the facility layout problem in single-storey and multi-storey buildings. The algorithm begins by placing the two facilities which have the highest flow between them, at arbitrary locations such that the distance between the two facilities is minimized. The subsequent facilities are then selected one at a time, on the basis of their overall flows with the facilities already assigned. These facilities are assigned to locations such that the total cost of the partial layout is minimum. At the same time, space limitations are not violated. For example, at stage i of the iteration process, the facility i selected for assignment is such that it has the highest overall flow values with the facilities 1,...,i-1 which are already assigned. Then facility i is assigned to a location such that the cost is minimum and the ordering of the facilities 1,...,i-1 is not changed. This procedure is repeated until all the facilities are assigned to their locations. The above algorithm can solve layout problems in which facilities are of equal area only.

FATE: FATE (Block, 1978) was developed by extending the layout principles of MAT. As previously mentioned MAT ranks facility pairs only on the basis of their flow values. As a result, MAT is not able to differentiate between facility pairs which have identical flow values

and ranks such pairs of facilities in a random manner. Such a random ranking may often lead to solution results of poorer quality. FATE overcomes this problem by using two criteria to rank facility pairs, i.e., flow values and total closeness rating. Lewis and Block (1980) note that several versions of FATE, based on different ranking criteria, have been developed.

INLAYT: INLAYT is one of the algorithms used in a heuristic proposed by O'Brien and Abdel Barr (1980). This heuristic uses the construction algorithm INLAYT to generate an initial layout which is then improved by an improvement algorithm called S-ZAKY. The user can accept, reject or modify the output of both INLAYT and S-ZAKY by using a light-pen attached to a graphics terminal.

INLAYT groups facilities depending upon weighted flow values (i.e., number of units of flow multipled by the cost of transporting the unit flow) and displays the groups on a graphics terminal along with an array of possible locations. The user then responds by assigning the facilities in the first group to any desired location. The same procedure is repeated for the second group, third group, and so on until all the facilities have been assigned.

2.3.2.2 Improvement algorithms

In improvement algorithms there is always an initial solution, which is often randomly generated. To this initial solution, systematic exchanges between facilities are made and the results are evaluated. The exchange which produces the best solution is retained and the procedure is continued until the solution cannot be improved any further. Hence, the

solution quality of improvement algorithms depends upon the initial layout evaluated. In this subsection eight improvement algorithms are discussed briefly.

CRAFT: CRAFT was originally presented in Armour and Buffa (1963) and Buffa et al. (1964). The principle involved in CRAFT is so popular that it has been modified frequently. Examples of such modifications are COFAD (Tompkins and Reed, 1976), biased sampling technique (Nugent et al., 1968), COL (Vollmann et al., 1968), CRAFT-M (Hicks and Cowan, 1976), SPACECRAFT (Johnson, 1982) and CRAFT-3D (Cinar, 1975). SPACECRAFT (which was published later than CRAFT-3D) is very similar to CRAFT-3D (Jacobs, 1984).

CRAFT begins by determining the cost of the initial layout. It then evaluates all possible location exchanges between pairs of facilities which either are adjacent to each other or are of the same area. The location exchange which results in the greatest estimated cost reduction, is made. This procedure continues until there is no location exchange which results in a layout with a lower solution cost than that of the current layout. CRAFT can handle only forty facilities and does not perform well when the facilities are of unequal area (Foulds, 1983 and Scriabin and Vergin, 1976).

H63: Hillier (1963) developed a heuristic algorithm which is based on a move desirability table. This table consists of values (based on a given initial layout) which represent the cost changes that would result by moving a facility from its current location to an adjacent location.

The move desirability table is scanned and the maximum value is selected. Facility i corresponding to this maximum value is considered for a move to the location indicated in the move desirability table. If, after testing the move there is a positive reduction in cost, then the indicated move is made. Otherwise, other adjacent moves are considered and the move which results in a positive reduction in cost is made. If there is no positive reduction in cost when facility i is moved to any of its adjacent locations, then the second largest value in the move desirability table is selected and the above procedure is repeated. The algorithm considers only pairwise exchanges between adjacent facilities and solves problems with facilities of equal area only.

HC63-66: Hillier and Connors (1966) have suggested a modification of of H63. In the new algorithm, k-step moves ($1 \leq k \leq n$, where n is the number of facilities) of a facility are permitted. Unlike H63 which allows exchange of adjacent facilities only, HC63-66 allows the exchange of non-adjacent facilities as well. At the same time, it limits these exchanges only to facilities which lie on a horizontal, vertical or diagonal line. Beginning with the $(n-1)^{th}$ - step move, the algorithm proceeds by decreasing the step value k sequentially one by one whenever no reduction in cost is found. When k is equal to 1 and there are no moves which appear to reduce the cost, the procedure is terminated or repeated as is necessary. Like H63, HC66 can be used to solve problems with facilities of equal area only.

COL: COL (Vollmann et al., 1968) determines for each facility i, the cost p_i of flow from facility i to all other facilities which are

located α or more units from facility i (α may be set by the user). These costs p_i form the basis for selecting two facilities m,n which are the most promising candidates for exchange. Then, facility m is considered for exchange with all other facilities on the basis of the cost reductions possible. An exchange is made if a cost reduction is possible. After facility m has been considered for exchange with all the other facilities, facility n is considered for exchange with all the other facilities, again based on the cost reduction that is possible. As before, an exchange is made if a cost reduction is possible. The p_i's are then recomputed and the cycle is repeated until there is a set of p_i's for which the exchanges of the corresponding facilities m, n (obtained as described above) with the other facilities does not lead to improved solutions. When the above procedure is completed, a subroutine checks all possible pairwise exchanges twice to determine if further improvements can be found. If not, the program is terminated. COL produces good quality solutions, is twice as fast as HC66 and has lesser memory storage requirement.

Sampling algorithms: Two sampling algorithms have been proposed. The first sampling algorithm (called the biased sampling algorithm) by Nugent et al. (1968) generates random solutions. To each solution that is better than the previous solution, a non-zero probability is assigned. The algorithm permits the selection of any pairwise exchange which results in a cost reduction. However, the bias is towards the sampling of better solutions. In essence, the biased sampling procedure

introduces a probabilistic element to the CRAFT algorithm and searches the neighbourhood of CRAFT for a better solution. The authors of the biased sampling algorithm believe that in a sample of ten solutions, the best solution produced by the biased sampling algorithm may be better than that of CRAFT.

The second sampling algorithm was developed by Hitchings and Cottam (1976) and is called the Terminal Sampling Procedure (TSP). It uses principles from other algorithms such as CRAFT, COL, etc. The algorithm executes selective pairwise exchanges thereby reducing computation time. At the end of these selective exchanges, the iteration is terminated by a CRAFT loop.

FRAT: FRAT (Khalil, 1973) is an algorithm which uses principles from other well known algorithms such as HC63-66, CRAFT, COL, etc. The difference d between the longest and the shortest distances between two facilities in the initial layout, is determined. The algorithm then executes two procedures - the total cost determination procedure and the exchange procedure. In the total cost determination procedure, for each facility i, the total cost p_i, of flow from each facility i to all the other facilities which are d or more units apart, is calculated. The two facilities m,n corresponding to the highest and second highest costs among p_i, i=1,...,n are considered as possible candidates for the exchange procedure. Then the exchange procedure is done as follows: the total costs of exchanging the locations of each facility with that of facility m are considered. The exchange that results in the maximum

reduction in the total cost is made. The exchange procedure is repeated until no more cost reductions are possible. When no further cost reduction is possible, the exchange procedure is repeated as before, but this time the exchange is between facility n and other facilities. Then d is reset to another value equal to (d-l), where l is the shortest distance between the centres of two facilities. The total cost determination procedure and exchange procedures are applied to the current layout for the new value of d, until d is less than l. In the final stage of the algorithm, pairwise exchange of the "greedy" type as outlined in Parker (1976), are considered. In the greedy exchange procedure, if an exchange results in a positive savings in cost, the exchange is made immediately. This procedure terminates when there are no exchanges which will reduce the total cost. FRAT can only solve layout problems in which the facilities are of equal area. It produces solutions of good quality.

COFAD: COFAD (Tompkins and Reed, 1976) is a modification of CRAFT and includes move costs for all alternative material handling systems (MHSs), thereby integrating the material handling system selection problem with the layout problem. COFAD improves the initial layout using the CRAFT procedure. The algorithm then determines the cost of moving material between each pair of facilities using the feasible alternative MHSs. The move costs thus determined are used to select a minimum cost MHS. Then a 'model supervisor' determines whether the facility design has reached a steady state and then directs the model. Steady state is said to be reached when the cost of the MHS and the number of MHS changes vary by less than a certain percentage of the total MHS cost and

number of assignments, respectively. If the steady state condition is reached, then the model supervisor terminates the program (or performs sensitivity analysis, if desired). If the steady state condition is not reached, then the costs of the MHS is allocated to each move and the above procedure is repeated until a steady state solution is reached.

Shore and Tompkins (1980) have modified COFAD so as to incorporate flexibility in the design process. A facility design is said to be flexible if it has the least expected inefficiency over several production levels. The modified version of COFAD is termed COFAD-F.

Revised Hillier algorithm: The algorithm developed by Picone and Wilhelm (1984) uses H63 to improve a given initial solution and then further improves this solution by considering 4-way perturbations. (A k-way perturbation considers the exchange of the locations of k facilities at a time). If the application of the 4-way perturbation leads to an improved solution, then a method called PERTURB is applied. If not, an improved solution is sought using H63. PERTURB considers 3-way and 4-way perturbations. After the 3-way and 4-way perturbations, H63 is applied to the current solution. If the resulting solution meets a user specified criterion C, then the 4-way perturbation is applied once more. If, as a result, there is any cost reduction, then PERTURB is applied again to the current solution. If there is no reduction in cost, the program terminates. If the specified criterion C was not met as a result of applying PERTURB, a final effort is made to improve the current solution. If an improved solution is not forthcoming, the program is terminated.

Since the revised Hillier algorithm uses H63, it produces solutions which are at least as good as the solutions of H63. As can be expected, the revised Hillier algorithm requires more computation time than H63.

2.3.2.3 Hybrid algorithms

Bazaraa and Kirca (1983) classified algorithms which have the characteristics of optimal and suboptimal algorithms as hybrid algorithms. Examples of such algorithms can be found in Burkard and Stratman (1978), Bazaraa and Sherali (1980) and Bazaraa and Kirca (1983). In this thesis, this classification is extended to include certain algorithms such as those of Elshafei (1977) and Scriabin and Vergin (1985), which use the principles of construction and improvement algorithms.

Burkard and Stratman (1978) proposed a heuristic algorithm which uses a branch-and-bound algorithm and an improvement algorithm. An initial solution is obtained using a branch-and-bound algorithm which terminates after a preset time limit is exceeded. The initial solution is then improved by using an improvement algorithm called VERBES. VERBES uses pairwise and triple exchanges alternately until no further improvement can be found in the current solution. Then, the smallest level k_0 in the branching process at which VERBES obtains a better solution than the branch-and-bound algorithm, is determined. The above mentioned procedure is repeated from level k_0 until the current solution cannot be improved any further.

Burkard and Stratman (1978) proposed another algorithm which is similar to the above mentioned algorithm but uses the Gaschutz-Ahrens

algorithm (Gaschutz and Ahrens, 1968) instead of the branch-and-bound algorithm based on time limits.

Bazaraa and Kirca (1983) proposed heuristic algorithms which are modifications of an optimal algorithm presented in the same paper. The heuristic algorithms are based on a branch-and-bound algorithm which reduces the computation requirement by eliminating any branch which is a mirror image of a previously explored branch. In other words, if branch A is a mirror image of branch B whose lower bound has been previously computed, then no further search takes place along branch A. By using 2-way and 4-way improvement exchange algorithms and selective branching rules, the heuristic algorithms are shown to produce good quality solutions.

FLAC: FLAC (Scriabin and Vergin, 1985) is an algorithm which consists of three stages. In the first stage, facilities are located such that the distance between them are inversely related to the flow. In the second stage, the facilities are assigned using the principle in stage 1, but now, the space constraints are taken into consideration. The third stage consists of fine adjustment using an exchange algorithm similar to FRAT.

Elshafei (1977) proposed an algorithm which is a combination of a construction algorithm and an improvement algorithm. The construction algorithm employs two strategies. In the first strategy, locations are ranked in ascending order of R_j, where R_j is the sum of distances from location j to all other locations. Facilities are also ranked in ascending order of L_i, where L_i is based upon the number of facilities

having flow with facility i and the sum of the flow values to and from facility i. At any stage in the assignment process using the first strategy, the unassigned facility with greatest L is assigned to the unused location with minimum R . In the second strategy, at any stage k, the unassigned facility which has the maximum flow with the facility assigned in stage k-1, is assigned to an unused location that causes a minimum increase in the total cost. Using the above two strategies, a complete layout is obtained and improved (if possible) by an improvement algorithm.

DISCON: Drezner (1980) has modelled the facility layout problem as a nonconvex mathematical programming problem. This problem is solved using a two-phase algorithm called dispersion-concentration algorithm. In the dispersion (first) phase, using the Lagrangean differential gradient method, good initial conditions are found so as to obtain a satisfactory local minimum to the mathematical programming problem. The final solution in the dispersion phase provides good starting points for the concentration (second) phase. In the first phase, the solution is such that the facilities do not touch one another, i.e., they are not close enough. The second phase consists of concentrating the facilities so that they are as close as possible (without overlapping). This solution is a local minimum to the mathematical programming problem. Drezner (1980) points out that although the dispersion phase provides good starting points, it is difficult to justify this outcome.

2.3.2.4 Graph theoretic algorithms

Graph theoretic algorithms identify maximal planar subgraphs of a weighted graph which show the relationships between the facilities. The dual of a maximal planar subgraph determines a layout of the facilities. Note that although some of the graph theoretic algorithms can be classified as construction algorithms, all graph theoretic algorithms are discussed in this section.

Seppanen and Moore (1970) proposed the above mentioned graph theoretic solution procedure. A heuristic algorithm which uses this strategy was also presented (Seppanen and Moore, 1975). The algorithm determines the maximum spanning tree based upon the weighted graph. With the help of an edge adding process, the maximum spanning tree is then used to obtain a maximal planar subgraph. As mentioned before, the dual of the maximum planar subgraph determines a layout of the facilities.

Branch-and-bound algorithm: Foulds and Robinson (1976) presented a branch-and-bound algorithm for solving the facility layout problem. The algorithm begins by placing pairs of facilities {i,j} in descending order of their flow values in a list P. All the pairs of facilities ($\{i,j\} \in N$, where N is the set of pairs of facilities which must be adjacent) are placed in adjacent locations to get a partial graph (assignment). Then, the branch-and-bound process begins. At any stage k, k pairs of facilities (including the pairs of facilities in N) are chosen and included in the graph T. To obtain an optimal solution, a maximal planar graph is required. Since a maximal planar graph has 3n-6 edges, where n is the number of vertices or facilities, 3n-6-k more edges have to be added to the graph T to make it a maximal planar graph. Note that an edge {i,j} in the graph represents the relationship between facilities i and j.

Also, at every stage in the branch-and-bound process, the penalty of adding or not adding the next available pair of facilities in list P, is determined. The branching then takes place from the node which has the least penalty and which has less than 3n-6 edges. At the same time, it is determined whether an edge can be added to the current graph T without making it nonplanar. The nodes that are considered for branching are those that lead to maximal planar graphs with minimum penalty. The branch-and-bound process continues until all the nodes with a penalty less than that of the current minimum penalty have been considered. The last such node gives the optimum solution.

Deltahedron algorithm: Foulds and Robinson (1978) presented two heuristic algorithms which avoid the testing of planarity. (In graph theoretic algorithms, this is a difficult task, especially as the problem size increases). The algorithms initially determine a tetrahedron, i.e., a particular type of graph in which each of the four vertices is connected to the other three vertices (figure 6). Note that the tetrahedron has four faces including the external face - f_1, f_2, f_3, f_4.

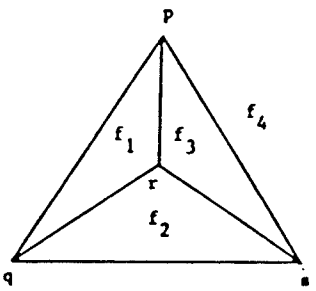


Figure 6: Tetrahedron

The remaining vertices are then inserted one at a time, in one of the faces of the graph. At any stage in the algorithm, a list of vertices V, edges E and faces F is maintained. For example, if vertex a is inserted in face f_1 which consists of edges pq, pr and qr (figure 6), then the corresponding edges ap, aq and ar are also added to the graph. The lists V, E and F are updated as follows:

V consists of vertices a, p, q, r, s

E consists of edges ap, aq, ar, pq, pr, qr, ps, rs, qs

F consists of faces f_1, f_2, f_3, f_4, f_5, f_6.

The two algorithms which employ the above mentioned strategy differ in the manner in which the initial tetrahedron is selected. Computational experience for the algorithms is presented in Foulds and Robinson (1978).

Carrie et al. (1978) developed four heuristic algorithms which follow the general solution procedure outlined in Seppanen and Moore (1975), but consider an additional step, that of redrawing the maximal planar graph based upon the relationship between the facilities. The four heuristic algorithms differ in the manner in which edges are added to the graph at each step. Note that edges are added at each step in order to obtain a maximal planar graph. The heuristic algorithms which were coded in FORTRAN and PL/1 are also compared in Carrie et al. (1978).

Wheel expansion algorithm: Eades et al. (1982) developed a heuristic algorithm which is similar to that of Foulds and Robinson (1978). The algorithm begins by determining a tetrahedron. Then a procedure known as wheel expansion takes place. A wheel on n vertices is a graph which consists of a cycle (known as the rim) of n-1 vertices, such that each

of the n-1 vertices is adjacent to an additional vertex p (known as the hub). A wheel on 7 vertices is shown in figure 7.

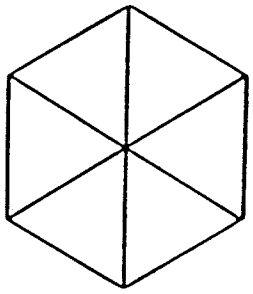


Figure 7: Wheel

In the wheel expansion procedure, an additional vertex q (which is not in the current wheel), is added to the graph such that:

i) p and q are the hubs of two wheels

ii) there are two vertices k,l which are on the rims of both the wheels

iii) each vertex previously adjacent to p is adjacent to at least one of p and q in the new graph.

By continuing in the above manner, planar graphs are obtained which can then be used to determine a layout of the facilities.

Foulds et al. (1985) compared the deltahedron algorithm, wheel expansion algorithm and another greedy algorithm in which edges are ordered on the basis of their weights and added to a graph if they do not make the graph nonplanar. It was found that the deltahedron

algorithm in combination with an improvement technique was the most successful with respect to solution quality and computation time.

In addition to the above mentioned algorithms, Moore (1976) proposed an algorithm which is similar to the algorithm of Seppanen and Moore (1975) except that it employs a different edge adding process to obtain a maximal planar graph from the maximum spanning tree. Green and Al-Hakim (1985) presented a matrix representation of a planar graph and its dual graph and used it to develop a heuristic algorithm for the facility layout problem. GASOL (Hammouche, 1983) is a heuristic algorithm based upon the string representation suggested by Carrie et al. (1978). It is compared with CRAFT, CORELAP and ALDEP for eight test problems in Nugent et al. (1968).

2.4 KNOWLEDGE-BASED SYSTEMS FOR THE LAYOUT PROBLEM

A number of knowledge-based systems have been applied for solving manufacturing problems. Heragu and Kusiak (1987) have surveyed some of them. In this section, two knowledge-based systems developed for the layout problem, i.e., FADES (Fisher and Nof, 1984) and IFLAPS (Kumara et al., 1985), are discussed.

2.4.1 FADES

Fisher and Nof (1984) have developed an expert system called FADES for the facility design problem. FADES aids not only in facility planning, but also in the selection of technology and economic investment analysis, generation of relationship charts, flow and distance matrices, acquisition of data (if necessary) from the user or a database

management system (DBMS). It addresses other unstructured problems which arise in the course of facility design, as well.

FADES consists of a knowledge base, a PROLOG interpreter, a DBMS (relevant to the concerned company) and task specific data. The knowledge base consists of algorithms, economic models and expert rules. First order predicate logic is used to represent knowledge. The knowledge representation includes assertions of facts, goals and procedures.

The PROLOG interpreter employs forward chaining depth first search in order to show that the negated goal does not match any of the assertions in the database.

The knowledge base consists of expertise for:

i) selection of equipment and economic investment analysis,
ii) developing relationship ratings between facilities,
iii) selecting and invoking the appropriate algorithm,
iv) solving the facility layout problem and also to prepare data that is necessary for solving the facility layout problem, and
v) retrieving appropriate data that may be required from a DBMS.

With the help of the above knowledge it is possible to design a manufacturing system. Initially the required equipment/technology level is identified and the available equipment is examined. Then a candidate list of the available equipment which will meet the required technology level, is prepared. In order to do this, production parameters such as parts per assembly, product volume, assembly time, number of different

styles and products, etc., are examined. Interaction with the user permits addition (deletion) of knowledge to (from) the database.

Once the candidate list of available equipment is prepared, a replacement analysis module performs economic analysis of the alternative equipment and recommends the appropriate equipment as also the inference procedure.

FADES is also capable of developing a relationship chart for a given set of facilities. The relationship chart provides closeness desired between each pair of facilities in the set. This is done using a series of expert rules which are obtained from human experts. These rules are subjective in nature but are important in determining the facility layout. For example, due to technological constraints, a forging and a heat treating station have to be located adjacently. Using the non-flow relationship ratings, facilities are put into groups of two. If this is not possible with the help of expert rules or knowledge in the DBMS, then the program asks the user about determining groups of facilities. Thus the relationship chart is constructed.

In order to solve the facility layout problem, flow (distance) data between pairs of facilities (sites) are required. The flow data are prepared with the help of data regarding product demands, operations performed by each facility, etc. To prepare the distance data, information regarding site descriptions is used. From the flow and distance data, a material handling cost matrix is constructed. This matrix and the relationship chart are used to solve the facility layout problem. To solve the facility layout problem, the linear assignment

algorithm is invoked. It should be noted that heuristic quadratic assignment algorithms may also be stored in the database and with suitable rules the appropriate algorithm (i.e., linear or quadratic) may be chosen depending on the problem at hand.

As mentioned before, FADES permits interaction with an external DBMS or with the user. However, it is desirable to keep interaction with the latter to a minimum.

2.4.2 IFLAPS

IFLAPS (Kumara et al., 1985) consists of two basic modules:

- an expert system module, and
- a syntactic pattern recognition module.

Both these modules can generate solutions for the layout problem.

The expert system module uses three types of assignment rules to assign machines to their respective sites. The first type of rule assigns a machine i to a site j if the resource required by machine i is available at site j. The second type of assignment rule assigns machines with high flow value between them to adjacent sites. The third type of assignment rule assigns machines which should not be located adjacently to non-adjacent sites.

The pattern recognition module consists of expert rules which determine which machine is to be assigned first in the floor plan. Then other machines are added to sites in the floor plan such that:

- hazardous machines are assigned to their corresponding designated sites,
- non hazardous machines are assigned based on their interaction with previously assigned machines.

Chapter III

MODELLING THE LAYOUT PROBLEM

A number of models, for example, the QAP (Koopmans and Beckmann, 1957), linear mixed integer programming problem (Love and Wong, 1976), nonconvex mathematical programming problem (Drezner, 1980), have been developed for the layout problem. All the formulations except that of Drezner (1980) and Neghabat (1974), require that the location of sites be known a priori. The formulations presented in this chapter, are more general than most of the existing models because the location of sites is not required to be known a priori.

3.1 MODELS FOR THE SINGLE-ROW MACHINE LAYOUT PROBLEM

In order to model the single-row machine layout problem, the following assumptions are made (Heragu and Kusiak, 1987a):

- machines are to be arranged along a straight line (figure 5a),
- machines are to be oriented in only one given direction.

The following notation is used in models M1, M1a and M1b:

f_{ij} frequency of trips between machines i and j

c_{ij} cost per trip between machines i and j

l_i length of machine i

d_{ij} minimum distance by which machines i and j are to be separated

x_i distance between center of machine i and vertical reference line vrl

The parameters l_i and d_{ij}, decision variable x_i and the reference line vrl are illustrated in figure 8.

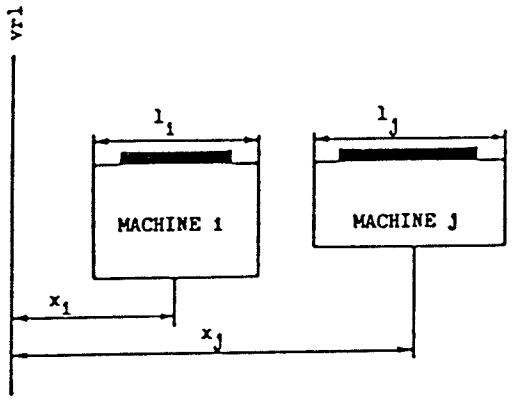


Figure 8: Illustration of parameters and decision variable for the single-row machine layout problem

Model M1

The objective function of model M1 minimizes the total cost involved in making the required trips between machines.

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} |x_i - x_j| \qquad (1)$$

$$\text{s.t. } |x_i - x_j| \geq 1/2(l_i + l_j) + d_{ij} \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (2)$$

$$x_i \geq 0 \qquad i=1,\ldots,n \qquad (3)$$

Constraint (2) ensures that no two machines in the layout overlap. Constraint (3) ensures nonnegativity.

Note that in the above and following models, the nonnegativity constraints have been provided to make the interpretation of the solutions easier. Omitting them does not affect the solution to the model.

Neghabat (1974) developed a model which is similar to model M1. Model M1 cannot be solved optimally by a standard linear programming code, as it includes absolute values in the objective function and constraints. In order to transform model M1 into an equivalent linear mixed integer programming model M1a, define:

$$x_{ij}^{+} = \begin{cases} (x_i - x_j) & \text{if } (x_i - x_j) > 0 \\ 0 & \text{if } x_i - x_j \leq 0 \end{cases} \tag{4}$$

$$x_{ij}^{-} = \begin{cases} -(x_i - x_j) & \text{if } (x_i - x_j) < 0 \\ 0 & \text{if } (x_i - x_j) \geq 0 \end{cases} \tag{5}$$

$$z_{ij} = \begin{cases} 1 \text{ if } x_i < x_j \\ 0 \text{ if } x_i > x_j \end{cases} \tag{6}$$

Based on the above, it is obvious that:

$$|x_i - x_j| = x_{ij}^{+} + x_{ij}^{-} \tag{7}$$

$$(x_i - x_j) = x_{ij}^{+} - x_{ij}^{-} \tag{8}$$

Model M1a

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (x_{ij}^{+} + x_{ij}^{-}) \qquad (9)$$

$$\text{s.t.} \quad x_i - x_j + Mz_{ij} \geq 1/2(l_i + l_j) + d_{ij} \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (10)$$

$$-(x_i - x_j) + M(1 - z_{ij}) \geq 1/2(l_i + l_j) + d_{ij} \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (11)$$

$$x_{ij}^{+} - x_{ij}^{-} = x_i - x_j \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (12)$$

$$x_{ij}^{+},\ x_{ij}^{-} \geq 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (13)$$

$$x_i \geq 0 \qquad i=1,\ldots,n \qquad (14)$$

$$z_{ij} = 0,1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (15)$$

Constraints (10) and (11) ensure that no two machines in the layout overlap. Since z_{ij} is a 0,1 variable, only one of the constraints (10) and (11) holds. Constraint (12) is identical to expression (8). Constraints (13) and (14) ensure nonnegativity and constraint (15) imposes integrality. In the above and other models presented in this thesis, the letter M denotes an arbitrarily large positive number.

Murty (1983) has shown that in any model which consists of absolute values in the objecvtive function, if the transformation similar to (7) is made in the objective function and transformation similar to (8) is made in the constraint, then at least one of x_{ij}^{+}, x_{ij}^{-} will always be zero, i.e.,

$$x_{ij}^{+} \; x_{ij}^{-} = 0 \qquad (16)$$

Observation: If transformation of the form (17) which is similar to (7), is made in the constraint:

$$|x_i - x_j| \geq b_{ij}, \text{ i.e., } x_{ij}^{+} + x_{ij}^{-} \geq b_{ij}, \qquad (17)$$

where b_{ij} is a real constant, then the solution to the model will not always satisfy (16). This is why constraint (2) in model M1 which is similar to (17), has been replaced by constraints (10) and (11) in model M1a.

It should be noted that the single-row machine layout problem can also be modelled as a non-linear continuous problem as shown below:

Model M1b

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (x_{ij}^{+} + x_{ij}^{-}) \qquad (18)$$

$$\text{s.t.} \quad x_{ij}^{+} + x_{ij}^{-} \geq 1/2 (l_i + l_j) + d_{ij} \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (19)$$

$$x_{ij}^{+}, x_{ij}^{-} \geq 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (20)$$

$$x_i \geq 0 \qquad i=1,\ldots,n \qquad (21)$$

and constraints (12), (16).

Constraint (19) ensures that no two machines in the layout overlap. Constraints (20) and (21) ensure nonnegativity.

3.2 MODELS FOR THE MULTI-ROW LAYOUT PROBLEM WITH MACHINES OF EQUAL AREA

Model M1 is used to formulate the single-row machine layout problem. In general, one finds that the machines have to be located in two or more rows. To model this problem, either the QAP or its equivalent linear transformations have frequently been used. Below, a linear program (22)-(24) which can be used to model the layout problem in which the machines are of equal area and square in shape, is presented. In addition to c_{ij}, d_{ij}, f_{ij}, defined in model M1, the following notation is used.

x_i vertical distance between facility i and horizontal reference line hrl

y_i horizontal distance between facility i and vertical reference line vrl

The above decision variables and the reference lines vrl, hrl are illustrated in figure 9.

Model M2

The objective function of model M2 is similar to that of model M1 and minimizes the total cost involved in making the required number of trips between the facilities.

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (|x_i - x_j| + |y_i - y_j|) \qquad (22)$$

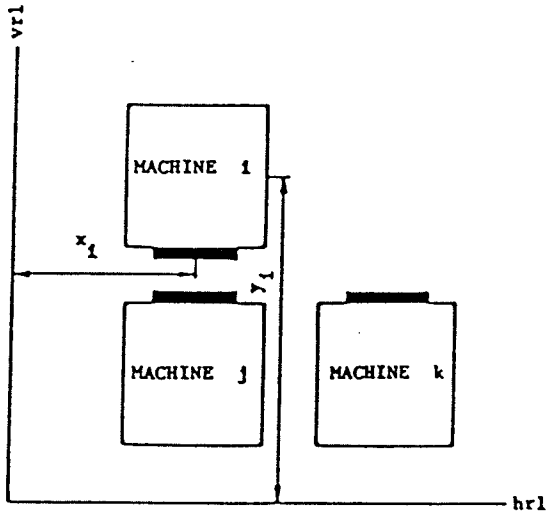


Figure 9: Illustration of decision variables and reference lines for the multi-row layout problem with machines of equal area

$$\text{s.t. } |x_i - x_j| + |y_i - y_j| \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (23)$$

$$x_i, y_i \geq 0, \text{integer} \qquad i=1,\ldots,n \qquad (24)$$

Constraints (23) and (24) ensure that no two machines in the layout overlap. Also, constraint (24) imposes nonnegativity and integrality.

If the horizontal and vertical dimensions of the floor plan are denoted as h and v respectively, then by adding constraints (25) and (26) provided below:

$$|x_i - x_j| \leq v - 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (25)$$

$$|y_i - y_j| \leq h - 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (26)$$

one can ensure that the machines are located within the boundaries of of the floor plan.

In order to transform model M2 into an equivalent mixed integer programming model M2a, the following decision variables are defined:

$$x^{+}_{ij} = \begin{cases} (x_i - x_j) & \text{if } (x_i - x_j) > 0 \\ 0 & \text{if } (x_i - x_j) \leq 0 \end{cases} \tag{27}$$

$$x^{-}_{ij} = \begin{cases} -(x_i - x_j) & \text{if } (x_i - x_j) < 0 \\ 0 & \text{if } (x_i - x_j) \geq 0 \end{cases} \tag{28}$$

$$y^{+}_{ij} = \begin{cases} (y_i - y_j) & \text{if } (y_i - y_j) > 0 \\ 0 & \text{if } (y_i - y_j) \leq 0 \end{cases} \tag{29}$$

$$y^{-}_{ij} = \begin{cases} -(y_i - y_j) & \text{if } (y_i - y_j) < 0 \\ 0 & \text{if } (y_i - y_j) \geq 0 \end{cases} \tag{30}$$

Based on (27)-(30), it can be easily verified that:

$$|x_i - x_j| = x^{+}_{ij} + x^{-}_{ij} \tag{31}$$

$$|y_i - y_j| = y^{+}_{ij} + y^{-}_{ij} \tag{32}$$

$$(x_i - x_j) = x^{+}_{ij} - x^{-}_{ij} \tag{33}$$

$$(y_i - y_j) = y^{+}_{ij} - y^{-}_{ij} \tag{34}$$

Model M2a

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (x_{ij}^{+} + x_{ij}^{-} + y_{ij}^{+} + y_{ij}^{-}) \qquad (35)$$

$$\text{s.t.} \quad x_i - x_j + Mp_{ij} + Mq_{ij} \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (36)$$

$$-(x_i - x_j) + Mp_{ij} + M(1 - q_{ij}) \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (37)$$

$$y_i - y_j + M(1 - p_{ij}) + Mq_{ij} \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (38)$$

$$-(y_i - y_j) + M(1 - p_{ij}) + M(1 - q_{ij}) \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (39)$$

$$x_{ij}^{+}, x_{ij}^{-}, y_{ij}^{+}, y_{ij}^{-} \geq 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (40)$$

$$x_i, y_i \geq 0 \qquad i=1,\ldots,n \qquad (41)$$

$$p_{ij}, q_{ij} = 0,1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (42)$$

and constraints (33), (34).

Constraints (36)-(39) ensure that no two machines in the layout overlap. Since p_{ij}, q_{ij} are 0,1 variables, only one of the constraints (36)-(39) holds. Constraints (40) and (41) ensure nonnegativity and constraint (42) imposes integrality.

A non-linear programming model which is equivalent to model M2a is provided below:

Model M2b

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (x_{ij}^{+} + x_{ij}^{-} + y_{ij}^{+} + y_{ij}^{-}) \qquad (43)$$

$$\text{s.t.} \quad x_{ij}^{+} + x_{ij}^{-} + Mz_{ij} \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (44)$$

$$y_{ij}^{+} + y_{ij}^{-} + M(1 - z_{ij}) \geq 1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (45)$$

$$x_{ij}^{+} \; x_{ij}^{-} = 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (46)$$

$$y_{ij}^{+} \; y_{ij}^{-} = 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (47)$$

$$z_{ij} \; (1 - z_{ij}) = 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (48)$$

$$x_{ij}^{+}, \; x_{ij}^{-}, \; y_{ij}^{+}, \; y_{ij}^{-} \geq 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (49)$$

$$x_i, \; y_i \geq 0 \qquad i=1,\ldots,n \qquad (50)$$

and constraints (33), (34).

Constraints (44)-(45) ensure that no two machines in the layout overlap. Constraint (48) imposes that only one of the constraints (44)-(45) holds. Constraints (46) and (47) ensure that one of the two decision variables x_{ij}^{+}, x_{ij}^{-} and one of y_{ij}^{+}, y_{ij}^{-} is always 0. Constraints (49) and (50) are nonnegativity constraints.

As mentioned before, if the dimensions of the floor plan are given, constraints (25) and (26) may be added in order to ensure that the machines are arranged within the boundaries of the floor plan.

3.3 MODELS FOR THE MULTI-ROW LAYOUT PROBLEM WITH MACHINES OF UNEQUAL AREA

The linear and non-linear models M2, M2a and M2b presented in the previous section can be used to formulate the layout problem in which the machines are of equal area. In many practical cases, one may observe that the area of the machines are not always equal. To model the layout problem in which the machines are of unequal area, a non-linear program is presented. It is assumed that the machines are square or rectangular in shape. Also, the physical orientation of the machines are assumed to be known. In addition to the notation for c_{ij}, f_{ij}, x_i, y_i, used in model M2, the following parameters are defined:

l_i length of the horizontal side of machine i

b_i length of the vertical side of machine i

The parameters, decision variables and reference lines vrl,hrl relevant to models M3, M3a and M3b are illustrated in figure 10.

Model M3

The objective function of model M3 is similar to that of models M1 and M2 and minimizes the total cost involved in making the required trips between the machines.

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (|x_i - x_j| + |y_i - y_j|) \qquad (51)$$

$$\text{s.t.} \quad |x_i - x_j| + Mz_{ij} \geq 1/2(b_i + b_j) + d_{ij} \qquad \begin{matrix} i=1,\ldots,n-1 \\ j=i+1,\ldots,n \end{matrix} \qquad (52)$$

$$|y_i - y_j| + M(1 - z_{ij}) \geq 1/2(l_i + l_j) + d_{ij} \qquad \begin{matrix} i=1,\ldots,n-1 \\ j=i+1,\ldots,n \end{matrix} \qquad (53)$$

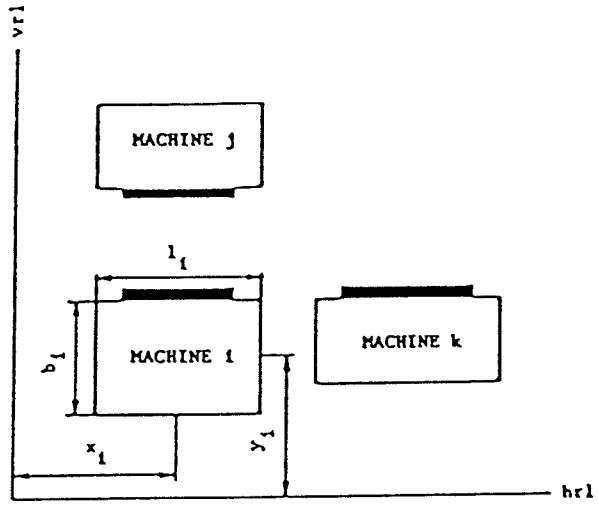


Figure 10: Illustration of decision variables and parameters for the multi-row layout problem with machines of unequal area

$$z_{ij}\,(1 - z_{ij}) = 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (54)$$

$$x_i,\ y_i \geq 0 \qquad i=1,\ldots,n \qquad (55)$$

Constraints (52)-(53) ensure that no two machines in the layout overlap. Constraint (54) ensures that only one of the two constraints (52)-(53) holds. Constraint (55) is a nonnegativity constraint.

Model M3 is transformed into an equivalent linear mixed integer programming model M3a as shown below.

Model M3a

The objective function of model M3a minimizes the total cost involved in making the required trips between the machines.

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (x_{ij}^{+} + x_{ij}^{-} + y_{ij}^{+} + y_{ij}^{-}) \qquad (56)$$

$$\text{s.t.}\quad x_i - x_j + M(p_{ij} + q_{ij}) \geq 1/2(b_i + b_j) \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (57)$$

$$-x_i + x_j + Mp_{ij} + M(1 - q_{ij}) \geq 1/2(b_i + b_j) \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (58)$$

$$y_i + y_j + M(1 - p_{ij}) + Mq_{ij} \geq 1/2(l_i + l_j) \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (59)$$

$$-y_i + y_j + M(1 - p_{ij}) + M(1 - q_{ij}) \geq 1/2(l_i + l_j) \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (60)$$

$$x_{ij}^+,\ x_{ij}^-,\ y_{ij}^+,\ y_{ij}^- \geq 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (61)$$

$$x_i,\ y_i \geq 0 \qquad i=1,\ldots,n \qquad (62)$$

$$p_{ij},\ q_{ij} = 0,1 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (63)$$

and constraints (33), (34).

Constraints (57)-(60) ensure that no two machines in the layout overlap. Integrality constraint (63) ensures that only one of the constraints (57)-(60) holds. For the sake of simplicity, the clearance, i.e., the minimum distance by which each pair of machines are to be separated, has not been included in the above and the next model. Constraints (61) and (62) are nonnegativity constraints.

An equivalent non-linear program M3b is provided below:

Model M3b

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} (x_{ij}^+ + x_{ij}^- + y_{ij}^+ + y_{ij}^-) \qquad (64)$$

$$\text{s.t.}\quad x_{ij}^+ + x_{ij}^- + Mz_{ij} \geq 1/2(b_i + b_j) \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (65)$$

$$y_{ij}^+ + y_{ij}^- + M(1 - z_{ij}) \geq 1/2(l_i + l_j) \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (66)$$

$$z_{ij}\,(1 - z_{ij}) = 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (67)$$

$$x^{+}_{ij},\ x^{-}_{ij},\ y^{+}_{ij},\ y^{-}_{ij} \geq 0 \qquad i=1,\ldots,n-1;\ j=i+1,\ldots,n \qquad (68)$$

$$x_i,\ y_i \geq 0 \qquad i=1,\ldots,n \qquad (69)$$

and constraints (33), (34), (46), (47).

As in the case of model M2, if the dimensions of the floor plan are known, one may add suitable constraints (as shown before) and ensure that the machines are arranged in such a manner that they fall within the boundaries of the floor plan.

The models presented in this thesis have the least number of integer variables when compared to other models published. A summary of the number of constraints, continuous and integer variables, for the linear models presented in this thesis and the existing most compact models is provided in table 1.

TABLE 1
Summary of models developed for the layout problem (continued on next page)

Layout problem	Type of model	Number of constraints[a]	Number of variables[b]	Number of nonnegativity constraints[c]	Number of integrality constraints	Number of integer variables	Reference
	Linear[d]	$n(n-1)/2$	n	n	0	0	M1[h]
Single-row	Linear mixed integer	$3n(n-1)/2$	n^2	n^2	$n(n-1)/2$	$n(n-1)/2$	M1a[h]
	Linear mixed integer	$n(3n-1)/2$	n^2	n^2	$n(n-1)/2$	$n(n-1)/2$	Love and Wong (1976a)
	Linear mixed integer[d]	$n(n-1)/2$	$2n$	$2n$	n	n	M2[h]
Multi-row[e]	Linear mixed integer	$3n(n-1)$	$2n^2$	$2n^2$	$n(n-1)$	$n(n-1)$	M2a[h]
	Linear mixed integer[d]	$n(n-1)$	$2n$	$2n$	$n(n-1)/2$	$n(n-1)/2$	M3[h]
Multi-row[f]	Linear mixed integer	$3n(n-1)$	$2n^2$	$2n^2$	$n(n-1)$	$n(n-1)$	M3a[h]

TABLE 1
Summary of models developed for the layout problem

Layout problem	Type of model	Number of constraints[a]	Number of variables[b]	Number of nonnegativity constraints[c]	Number of integrality constraints	Number of integer variables	Reference
Multi-row[g]	Linear mixed integer[d]	$3n(n-1)/2$	$2n$	$2n$	n	n	M2[i]
	Linear mixed integer	$4n(n-1)$	$2n^2$	$2n^2$	$n(n-1)$	$n(n-1)$	M2a[j]
	Linear mixed integer	n^2+4n	$2n^2$	$2n^2$	n^2	n^2	Love and Wong (1976)
	Linear mixed integer[k]	n^2+2n	$2n^2$	n^2	n^2	n^2	Kaufman and Broeckx (1978)
	Linear mixed integer[k]	n^2+2n+1	0	0	n^2+n^4	n^2+n^4	Lawler (1963)
	Linear mixed integer[k]	$2n^2$	$n^2(n-1)^2/2$	$n^2(n-1)^2/2$	n^2	n^2	Bazaraa and Sherali (1980)
	QAP	$2n$	0	0	n^2	n^2	Koopmans and Beckmann (1957)

a Excluding nonnegativity and integrality constraints
b Excluding integer variables
c Excluding integrality constraints
d Model consists of absolute values in the objective function and constraints
e Layout problem with machines of equal area only; dimension of floor plan not considered
f Layout problem with machines of unequal area; dimension of floor plan not considered
g Layout problem with machines of equal area only; dimension of floor plan considered
h Presented in this chapter
i Model M2 presented in this chapter with constraints (25) and (26)
j Model M2a presented in this chapter with constraints (25) and (26)
k Linearization of the quadratic assignment problem

Chapter IV

HEURISTIC ALGORITHM FOR SOLVING THE LAYOUT MODELS

In this chapter, computational experience with the models M1 and M2 developed in the previous chapter, is provided. To demonstrate the efficiency of the models, a number of single-row and multi-row layout problems available in the literature were solved. The models were solved using the Powell method of conjugate direction for unconstrained minimization (Press et al., 1986). Since models M1 and M2 are constrained models, they were transformed into unconstrained programs using the penalty method (Bazaraa and Shetty, 1979). In the penalty method, each constraint is squared, multiplied by a penalty parameter β, and placed in the objective function. Thus any violation of the constraints in the original model results in an objective function of higher value than the optimal. The heuristic algorithm used to solve the models, called the Modified Penalty Algorithm (MPA), is presented below.

4.1 MODIFIED PENALTY ALGORITHM (MPA)

Step 0: Initialization

Set β = penalty parameter

P = initial solution vector (may be an arbitrary feasible or infeasible solution)

z = objective function value corresponding to initial solution vector P

XI = direction matrix (a unit matrix of dimension n x n, where n is the number of variables in the problem)

Step 1: Multiply each squared linear inequality and equality constraint of the constrained minimization problem by the penalty parameter β and place them in the objective function.

Step 2: Solve the unconstrained minimization problem (obtained in Step 1) using the Powell algorithm;

Set $P^* =$ solution vector

$z^* =$ objective function value corresponding to P^*

If $z^* \leq z$, set $z=z^*$.

Step 3: Modify solution vector P^* so that a feasible solution is obtained.

There are three points regarding the above algorithm which are worth mentioning. First, computational experience has shown that if β is set to a high value, then its influence becomes less significant relative to the value of the elements in the flow matrix. Hence, the user has to exercise care and judgement in setting the value of β. Second, the quality of the solution produced by the algorithm depends to an extent on the initial solution provided. In general, the better the initial solution, the better the final solution. Third, the solution produced by the Powell algorithm (in step 2) may not always be feasible, i.e., the value of the variables may be such that the corresponding machines overlap. In such cases, the values of the variables (corresponding to the overlapping machines) are to be modified to make the solution feasible. This is done in step 3.

In order to be able to evaluate the solution quality of the above algorithm, certain standards were adopted in:

- setting the value of β, and
- providing the initial solution.

4.1.1 Computational Results with Model M1

The single-row layout problems were solved twice using the heuristic algorithm presented above. An infeasible initial solution (in which the value of each variable was set at 1) was provided the first time and a feasible initial solution was provided the second time. The way feasible initial solutions were provided was standard for each problem - machine 1 was placed in the left extreme position; machine 2 was placed to the right of machine 1, machine 3 to the right of machine 2 and so on. For all the single-row layout problems solved, the value of β was set at 1. Furthermore, a "greedy" pairwise exchange algorithm was used to improve the quality of the solution produced by MPA. The greedy algorithm considers pairwise exchange between the positions of machines. If the exchange between the positions of any two machines results in an improvement in the solution cost, then the exchange is made, and the above procedure is repeated until there is no further improvement in the solution cost.

In order to assess the performance of the modified penalty algorithm, 8 single-row layout problems were solved (see table 3). The flow and machine length data for problem 1 is provided in Beghin-Picavet and Hansen (1982); for problems 2 and 6 in Love and Wong (1976a); for problems 3,4 and 5 in Simmons (1969). Since the largest problem

available in the literature is the 11-machine layout problem (problems 5 and 6), we have introduced the 20-machine and 30-machine layout problems (problems 7 and 8) respectively, in order to demonstrate that model M1 can be used to solve large layout problems in a reasonable computation time. The flow data for problems 7 and 8 in table 3 are taken from Nugent et al. (1968); the corresponding machine dimension data are provided sequentially in table 2, begining from machine 1. For example, the dimension of machines 1,...,20 in problem 7 are 20,3,9,3,7,3,7,5,9,6,5,3,9,3,7,3,7,5,9,6 respectively. Computation results for model M1 are provided in table 3.

TABLE 2

Machine length data for problems 7 and 8 in table 3

Problem number	Machine length
7	20,3,9,3,7,3,7,5,9,6, 5,3,9,3,7,3,7,5,9,6
8	3,9,3,7,3,7,5,9,6,5, 3,9,3,7,3,7,5,9,6,5, 3,9,3,7,3,7,5,9,6,5

All the computation with MPA reported in this chapter, has been performed on an AMDAHL 5870 computer. As can be seen from table 3, MPA produces optimal solutions for 3 out of 4 problems for which optimal solutions are known. For problems 3, 6 and 7, the algorithm produced better solutions than those available in the literature. It should be noted that the dynamic programming algorithm of Picard and Queyranne

TABLE 3

Computational results with model M1 for the single-row machine layout problem

Problem number	Number of machines	Optimal or best known solution		MPA[1] with infeasible initial solution		MPA[1] with feasible initial solution	
		OFV	CPU[2]	OFV	CPU[3]	OFV	CPU[3]
1	4	78.0	0.18	78.0	0.08	78.0	0.09
2	5	151.0	3.13[4]	151.0	0.08	151.0	0.13
3	8	2348.5[6]	0.96	2324.5	0.36	2341.5	0.59
4	10	2781.5	5.40	2781.5	1.11	2781.5	0.84
5	11	6933.5	9.80	7041.5	0.96	7274.5	2.18
6	11	7021.5[6]	1200.00[5]	6933.5	0.98	6933.5	0.95
7	20	17244.0[6]	150.39[3]	16265.0	10.68	16109.0	7.82
8	30	n.a.	n.a.	46139.0	36.43	46454.0	35.74

OFV Objective function value
CPU Central processing unit time in seconds
n.a. Data not available

1 Each solution produced by MPA was improved by a greedy pairwise exchange algorithm once; the corresponding OFV and CPU times are reported

2 CPU time in seconds on an IRIS 80 C.I.I.-H.B. computer for problems 1,3,4 and 5 (Beghin-Picavet and Hansenm 1982)

3 CPU time in seconds on an AMDAHL 5870 computer (Heragu and Kusiak, 1987a)

4 CPU time in seconds on an IBM 360/65 computer for problem 2 (Love and Wong, 1976a)

5 CPU time in seconds on an IBM 370/158 computer for problem 6 (Love and Wong, 1976a)

6 Objective function value of the best known solution (Heragu and Kusiak, 1987a)

(1981) or Beghin-Picavet and Hansen (1982) can solve problems 3 and 6 optimally, but not problem 7 or 8. However, since they have not provided the objective function values corresponding to the optimal solution to problems 3 and 6, only the best known solution is reported in table 3.

The algorithm presented has low computation time. Unlike the dynamic programming algorithm of Picard and Queyranne (1981) or Beghin-Picavet and Hansen (1982), MPA can be used to solve large single-row layout problems. A major disadvantage of dynamic programming algorithms is that they have a high memory requirement. Picard and Queyranne (1981) have reported that their dynamic programming algorithm requires about $O(n2^n)$ memory locations, where n is the number of machines.

The computation results with model M1a which was solved using the branch-and-bound enumerative method of LINDO (Schrage, 1984), were not encouraging. For example, the optimal solution either could not be found or could not be verified for problems with 8 or more machines, even after 30 minutes of CPU time. This is because of the large number of integer variables in the model. Hence, computational results with model M1a are not included in this thesis.

4.1.2 Computational Results with Model M2

For all multi-row layout problems solved, the value of ß was set at 3. As before, each problem was solved twice, once with a standard infeasible initial solution and once with a feasible initial solution. The way in which feasible initial solutions were provided was also

standard for all problems, i.e., machine 1 was assigned to site 1, machine 2 was assigned to site 2, and so on.

Computational results using model M2 for the multi-row machine layout problem are provided in table 4. The flow and distance data for the multi-row layout problems solved in this chapter, are provided in Nugent et al. (1968). The performance of the modified penalty algorithm presented is compared with that of revised DISCON (Drezner, 1988). The reason for comparing MPA with revised DISCON is that both the algorithms are designed to solve models (for the layout problem) which do not require the location of sites to be known a priori. The revised DISCON algorithm applies CRAFT exchange algorithm (Armour and Buffa, 1963) ten times, to improve the solution. However, the objective function values (OFV) corresponding to the solutions produced by revised DISCON and MPA reported in table 4, indicate the OFV of the solution produced by the algorithms before the application of the CRAFT exchange algorithm, and "greedy" exchange algorithm respectively. This was done so as to provide a meaningful comparison of MPA with revised DISCON. Note also that the objective function values reported for revised DISCON are the average OFVs provided in Drezner (1987).

From table 4 it can be seen that with a feasible initial solution, the objective function values of the solutions generated by MPA were lower than the average objective function values of ten solutions (obtained by using ten different initial solutions) generated by revised DISCON, for 5 of the 6 test problems presented in Nugent et al. (1968). Thus the use of a simple, easily available algorithm such as the Powell algorithm to solve model M2, produces good quality solutions. This to a

TABLE 4

Computational results with model M2 for the multi-row machine layout problem

Problem number	Number of machines	Revised DISCON[1]		MPA with infeasible initial solution		MPA with feasible initial solution	
		OFV[2]	CPU[3]	OFV	CPU[4]	OFV	CPU[4]
1	6	47.5	0.06	43.0	0.63	43.0	0.30
2	8	118.8	0.08	113.0	1.40	131.0	1.35
3	12	322.2	0.16	332.0	5.01	320.0	3.47
4	15	630.8	0.32	658.0	9.27	630.0	5.00
5	20	1416.4	0.86	1407.0	17.81	1398.0	13.83
6	30	3436.4	4.86	3371.0	82.49	3418.0	49.90

OFV Objective function value
CPU Central processing unit time in seconds
[1] The OFVs reported for revised DISCON correspond to the solutions obtained before applying the CRAFT exchange algorithm 10 times
[2] Objective function values reported for revised DISCON are the average values of solution costs obtained using 10 different starting solutions (Drezner, 1987)
[3] CPU time required on an AMDAHL 470/v8 computer; the reported CPU time includes the computation time required by revised DISCON and the CRAFT exchange algorithm that was applied 10 times to the solution produced by revised DISCON
[4] CPU time required on an AMDAHL 5870 computer

degree demonstrates that model M2 is an efficient formulation of the layout problem. It appears that the use of more specialized algorithms to solve model M2 may produce solutions of even better quality. Although the algorithm presented generates solutions of better quality than

revised DISCON for the test problems, it appears that the latter has lower CPU time requirement. However, since the computer systems used are different, no conclusive inference may be drawn.

MPA combined with the FRAT (Khalil, 1973) exchange algorithm is compared with a branch-and-bound based heuristic algorithm developed by Bazaraa and Kirca (1983) and the objective function value, CPU time required to solve the test problems in Nugent et al. (1968) using these algorithms are provided in table 5. The reason for comparing MPA with the branch-and-bound based algorithm developed by Bazaraa and Kirca (1983) is that the latter is known to produce solutions of very good quality for the layout problem.

From table 5, it can be observed that MPA combined with FRAT produces good quality solutions in an acceptable computation time. The reason for not obtaining optimal solutions can be partly attributed to the limitations of the penalty method, in which the constrained model is transformed into an unconstrained one. This observation was also supported by the fact that the same solutions were obtained when we solved the unconstrained models using the Rosenbrock algorithm (Bazaraa and Shetty, 1979). In industrial applications where the deviation of the estimated flow data is usually less than 10% of the actual flow data, solutions whose objective function values deviate less than 10% from that of the optimal solution may be acceptable. If not one must use algorithms suited for solving constrained optimization models.

TABLE 5

Comparison of the objective function values and CPU time of the solutions generated by MPA combined with FRAT with the algorithm presented in Bazaraa and Kirca (1983)

Problem number	Number of machines	Branch-and-bound based algorithm		MPA + FRAT with infeasible initial solution		MPA + FRAT with feasible initial solution	
		OFV	CPU[1]	OFV	CPU[2]	OFV	CPU[2]
1	6	43.0	n.a.	43.0	0.64	43.0	0.31
2	8	107.0	n.a.	113.0	1.42	107.0	0.96
3	12	289.0	n.a.	321.0	5.09	300.0	3.56
4	15	575.0	30.59	622.0	9.35	600.0	5.09
5	20	1285.0	156.03	1329.0	17.90	1308.0	13.95
6	30	3064.0	320.25	3154.0	83.75	3147.0	50.04

OFV Objective function value
CPU Central processing unit time in seconds
n.a. Data not available
[1] CPU time required on a CDC Cyber 70 model 74-28/CDC 6400 computer
[2] CPU time required on an AMDAHL 5870 computer

It should also be noted that by suitably changing the penalty parameter β and providing different initial solutions, one may obtain solutions of better quality than those presented in tables 3, 4 and 5.

As mentioned previously, model M3 is suitable for solving the machine layout problem with machines of unequal area. Unfortunately, no such problem has been solved optimally in the literature. Computational experience with model M3 indicates that large layout problems can be

solved in a reasonable computational time. For example, the 30-facility layout problem can be solved in less than 2 minutes.

Chapter V

HEURISTIC ALGORITHMS FOR SOLVING THE LAYOUT PROBLEM

In practice, the decision regarding type of material handling system to be used is typically made at the equipment selection stage. Once this is done, the structure of the layout is determined based on the number of machines, space limitations and type of material handling system used. The actual layout is prepared using a heuristic algorithm. In this chapter, two algorithms for solving the machine layout problem are presented (Heragu and Kusiak, 1988). Each algorithm is applicable to a particular layout structure. The Modified Spanning Tree Algorithm (MSTA) is to be used when the layout pattern is single-row and the Triangle Assignment Algorithm (TAA) is to be used when the layout pattern is multi-row. Throughout this thesis, it is assumed that the cost of assigning a machine to any site is the same. This assumption is realistic because in an FMS, the site preparation and the machine location costs are independent of the sites. The required clearance between machines depends on which machines are adjacent and need not be a constant. For example, the clearance between a milling machine and a drilling machine may be more than the clearance between a milling machine and a lathe in order to allow easy loading and unloading. The required clearance between each pair of machines may be entered in a matrix as shown in matrix (3) presented later in this chapter. It is also assumed that the machines can be oriented in only one particular direction, irrespective of their locations. This assumption is only to

make the presentation simpler, and if necessary, can be relaxed. The number of machines is denoted as n. In addition, the algorithms use the data in an adjusted flow matrix which is constructed as follows:
The frequency of trips between the machines are entered in a flow matrix (for example see matrix (1) presented later in this chapter). Using the information about the dimensions and the orientation of all machines, an adjacency time matrix is also constructed. A value in the i^{th} row and j^{th} column of such a matrix indicates the time required to travel between machines i and j when they are adjacent to each other. The value in the i^{th} row and j^{th} column of the flow matrix is then multiplied by the corresponding entry in the i^{th} row and j^{th} column of the adjacency time matrix to obtain a new matrix called the adjusted flow matrix F.

It is to be noted that travel time rather than travel distance has been used to compute the adjusted flow matrix. This factor requires that the AGV motion characteristic be considered.

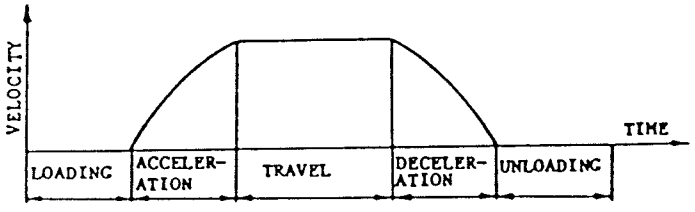


Figure 11: Components of the AGV travel time

As shown in figure 11, the AGV travel time between any two machines consists of five different components. Since the AGV velocity is a nonlinear function of time rather than distance between sites, the AGV travel times should be considered in the machine layout problem. This can be done due to the fact that for a given type of AGV:

- loading time is constant,
- acceleration time can be assumed to be constant with very small error,
- travel time can be calculated as a function of the distance between the sites travelled,
- deceleration time can be assumed to be a constant with very small error, and
- unloading time is constant.

5.1 MODIFIED SPANNING TREE ALGORITHM (MSTA)

Step 0. From the adjusted flow matrix $[f_{ij}]$ compute:

$f_{i^*j^*} = \max \{f_{ij} : i=1,....n, j=1,....n\}$.

Connect i*,j* and include them in the partial solution.

Set $f_{i^*j^*} = f_{j^*i^*} = -\infty$.

Step 1. Compute

$f_{p^*q^*} = \max \{f_{i^*k}, f_{j^*l} : k=1,...n, l=1,...n\}$ and

(i) connect q* to p* and add q* to the partial solution

(ii) delete row p* and column p* from $F = [f_{ij}]$

(iii) if p*=i*, set i*=q*; otherwise, set j*=q*.

Step 2. Repeat Step 1 until the final solution has been obtained (i.e., all the machines have been included in the solution).

There are four factors regarding the modified spanning tree algorithm which are worth mentioning. First, note that the solution generated by the algorithm does not produce the layout but only the sequence in which the machines have to be placed in the layout. The actual layout depends on the type of equipment selected for material handling, the required clearance between machines and their orientation. If a robot is used for material handling, then based on the sequence generated by MSTA, the machines are arranged along the circumference of a circle whose diameter is equal to twice the reach of the robot (see figure 1) and the robot is positioned in the centre of the circle. If an AGV is to be used for material handling, then the machines are arranged along a straight line as shown in figure 2. To determine the orientation of the machines, factors such as machine shape, type of loading device used, etc., need to be considered. The clearance between machines and their orientations are known to the layout analyst.

Second, in some manufacturing situations, a condition that a particular machine be placed in a particular site (say the beginning, the end or in the middle of a production line), may be imposed. It may also be desirable to locate machines with maximum flow value between them, near the battery charging station of the AGV. The reason is that typically AGVs are charged when they are not in use. Hence, in order to reduce travel time it may be worthwhile to locate machines which have high flow value between them near the battery charging station. Such

conditions can be easily incorporated in the proposed algorithm. The execution of the algorithm for problems with such conditions would be faster than the execution of problems without such conditions, as the number of machines to be assigned is smaller than the number of machines in the layout problem.

Third, one can easily prove that MSTA provides optimal results when the number of machines in the problem is less than four. But when the number of machines is four or more, it does not provide optimal results.

Fourth, MSTA is similar to the maximum spanning tree algorithm (Bondy and Murty, 1976). The difference between the two is that the former generates a spanning tree with the condition that:

- every vertex (machine) except the end vertices (machines) has degree two (i.e., adjacent to two other vertices),
- the end vertices (machines) have degree one and there are only two such vertices (machines),

whereas the maximum spanning tree algorithm generates a maximum spanning tree with no conditions.

The modified spanning tree algorithm may also be thought of as a heuristic algorithm for the "open" travelling salesman problem. By "open" is meant that the solution does not form a closed loop as in the general travelling salesman problem.

The use of MSTA is illustrated using the numerical example presented below.

5.1.1 Numerical Example with MSTA

Given the frequency of trips between 6 pairs of machines (matrix (1)), adjacency time matrix (2), the machine sizes (table 6) and the clearance between machines (matrix 3), determine the single-row machine layout assuming that an AGV has been selected as the material handling carrier. The orientation of each machine is such that the longer side of each machine is parallel to the AGV path. Furthermore, one of the longer sides of each machine must be equidistant from the AGV path.

$$[\tilde{f}_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 40 & 80 & 21 & 62 & 90 \\ 40 & 0 & 72 & 12 & 24 & 28 \\ 80 & 72 & 0 & 14 & 41 & 9 \\ 21 & 12 & 14 & 0 & 21 & 12 \\ 62 & 24 & 41 & 21 & 0 & 31 \\ 90 & 28 & 9 & 12 & 31 & 0 \end{bmatrix} \end{array} \qquad (1)$$

$$[t'_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 4 & 4 & 6 & 4 & 5 \\ 4 & 0 & 2 & 5 & 2 & 3 \\ 4 & 2 & 0 & 5 & 3 & 3 \\ 6 & 5 & 5 & 0 & 5 & 8 \\ 4 & 2 & 3 & 5 & 0 & 4 \\ 5 & 3 & 3 & 8 & 4 & 0 \end{bmatrix} \end{array} \qquad (2)$$

TABLE 6

Machine Sizes for the Example Problem

Machine No.	Dimension
1	50 x 30
2	20 x 20
3	25 x 20
4	60 x 35
5	30 x 15
6	40 x 40

$$[d_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 3 & 1 \\ 2 & 1 & 1 & 3 & 0 & 2 \\ 2 & 1 & 1 & 1 & 2 & 0 \end{bmatrix} \end{array} \tag{3}$$

From the flow and adjacency time matrices, the adjusted flow matrix (4) is constructed.

$$F = [f_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 160 & 320 & 126 & 248 & 450 \\ 160 & 0 & 144 & 60 & 48 & 84 \\ 320 & 144 & 0 & 70 & 123 & 27 \\ 126 & 60 & 70 & 0 & 105 & 96 \\ 248 & 48 & 123 & 105 & 0 & 124 \\ 450 & 84 & 27 & 96 & 124 & 0 \end{bmatrix} \end{array} \tag{4}$$

Step 0. Machines 1,6 are connected and included in the partial solution.

Step 1. Machine 3 is connected to machine 1 and is added to the partial solution.

Row 1 and column 1 are deleted in matrix (4).

Step 1. Machine 2 is connected to machine 3 and is added to the partial solution.

Row 3 and column 3 are deleted in matrix (4).

Step 1. Machine 5 is connected to machine 6 and is added to the partial solution.

Row 6 and column 6 are deleted in matrix (4).

Step 1. Machine 4 is added to machine 5 and is added to the partial solution.

Row 5 and column 5 are deleted in matrix (4).

The sequence in which machines are to be placed in the layout (obtained from MSTA) is (2,3,1,6,5,4).

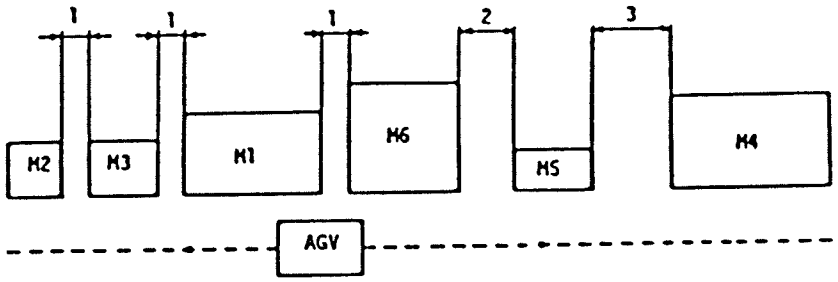


Figure 12: Single-row layout for the example problem

Since an AGV is to be used for material handling, the layout as shown in figure 12 is developed. Note that the sequence of machines generated by MSTA is maintained and a longer side of each machine is equidistant from the AGV path. Also, the clearance between adjacent machines is maintained as indicated in matrix (3).

5.2 TRIANGLE ASSIGNMENT ALGORITHM (TAA)

Now consider the arrangement of machines as presented in figures 3 and 4. To solve these machine layout problems a triangle assignment algorithm is developed. It consists of the following two phases.

Phase 1

Phase 1 involves the generation of triangles of maximum weight. The weight of a triangle is the sum of the weights of the edges of the triangle. In the algorithm, the vertices of triangles represent machines. To generate triangles based on the adjusted flow matrix F, a maximum spanning tree T is constructed. Then for the maximum spanning

tree, the adjacency matrix is set up. Some rows i of the adjacency matrix have pairs of columns j and k which have a "1" entry. This particular combination of vertices i,j,k indicates that a triangle (i,j,k) may be constructed by adding one of the edges {i,j} or {j,k} or {i,k} which is not in the spanning tree. All such possible triangles (i,j,k) in which the adjusted flow between any two vertices is greater than or equal to a threshold value q_0, are selected. Note that the value q_0 may be set by the user. Experience has shown that with a value of:

$$q_0 \leq (1/4)[\max\{f_{ij} : i=1,\ldots n, j=1,\ldots n\}],$$

the algorithm produces good solutions. The weights of these triangles are determined and the triangle with the maximum weight is chosen and denoted as Δ^*. Now, there is an edge of Δ^* which is not in the spanning tree, but if added would form the triangle Δ^*. This edge is added to the spanning tree T and triangle Δ^* is thus formed. Note that T is no more a tree as it has a cycle of length 3. The adjacency matrix is updated to represent this new graph.

The above procedure is repeated until all but one machine appear as vertices of one of the triangles at least once. For example, for a problem with n machines, if the triangles generated have at least n-1 vertices appearing at least once, then phase 1 of algorithm stops. The triangles generated are arranged in descending order of their weights in a list L and control then passes over to phase 2 of the algorithm.

Phase 2

Phase 2 of the algorithm consists of assigning machines to sites. The sites are created such that:

- there is one site for each machine, and
- all the sites are of equal area.

The sites created depend on the structure of the layout and are independent of the machine sizes. They are numbered sequentially from 1 through n, where n is the number of machines. The distance between the sites are entered in a matrix and used only in steps 5 and 6 of the algorithm. They are not used to calculate the solution cost of the layout.

Step 5 consists of two assignment rules. Assignment rule 1 selects two vertices of the first triangle in list L, based on the adjusted flow values. The first vertex selected is assigned to site p and the second vertex is assigned to a site that is closest to site p. It is to be noted that the value of p ranges from 1 to n and the assignment of the first vertex determines the assignment of the other vertices of the triangles in list L. Thus, it can be seen that TAA generates n sets of assignments. The third vertex in the first triangle in list L is assigned using assignment rule 2.

Assignment rule 2 involves determining the unassigned vertices in a triangle and assigning them to sites such that vertices with high adjusted flow value between them are as close to each other as possible. Step 7 uses assignment rule 2 to assign the vertices of the other triangles in list L.

Each of the n sets of assignments generated by TAA indicates the assignment of n machines to their corresponding sites. The actual layout is then constructed depending on the required clearance between machines

and their orientation. The orientation of the machines will depend on the type of loading device used, reach of the robot arm and other technological considerations.

Thus n different layouts are constructed and for each of the n layouts, the time t_{ij} required to travel between machines i and j is determined for each pair of machines and entered in a matrix. The frequency of trips f_{ij} and t_{ij} are used to calculate the solution cost as follows:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij} t_{ij}$$

Thus the solution cost for each layout is determined and the layout with minimum cost is selected.

Triangle Assignment Algorithm

Phase 1

Step 1. Set counter l = 1; q = q_0 (q_0 is set by user).

From the adjusted flow matrix $[f_{ij}]$, determine the adjacency matrix for the maximum spanning tree T_l.

Step 2. For each row i of the adjacency matrix of T_l, determine all the pairs of columns {j,k} which have a "1" entry in row i;

Determine the weight of each triangle (i,j,k):

(i) which is not in T_l and

(ii) in which the flow value between any two of its vertices is $\geq q$.

Step 3. Determine the triangle (i^*, j^*, k^*) with maximum weight. Break ties arbitrarily.

Add edge $\{j^*, k^*\}$ to T_l, and label this new graph as T_{l+1};

Store (i^*, j^*, k^*) as Δ_l.

If the number of different vertices in the l triangles, $N \geq n-1$ and $l > (n/3)$ then go to Step 4;

otherwise set $l=l+1$ and go to Step 2.

Step 4. Arrange the l triangles in descending order of their weights.

Phase 2

Step 5. Set counter $t=1$ and site number $p=1$.

Assign the vertices of Δ_l using assignment rule 1 and assignment rule 2 below.

Assignment Rule 1: Choose from Δ_t, the edges $\{i,j\}$, $\{j,k\}$ such that the weight of $\{i,j\}$ and $\{j,k\}$ is greater than or equal to the weight of $\{i,k\}$. Assign vertex j to site p.

Determine site p_1 such that the distance between sites p and p_1 is minimum.

If weight of $\{i,j\}$ is greater than weight of $\{j,k\}$, assign vertex i to site p_1; otherwise assign vertex k to site p_1.

Assignment Rule 2: Determine the unassigned vertices of Δ_t and assign them to sites which are as close as possible to the sites of the previously assigned vertices of

Δ_t. At the same time, pairs of vertices of Δ_t which have greater flow should be closer than pairs of vertices which have lesser flow;

Set t=t+1.

Step 6. Examine Δ_t. If the number of previously assigned vertices of Δ_t is:

(i) 0 or 3, then go to Step 8

(ii) 1 or 2, then assign the remaining vertices using assignment rule 2.

If any of these vertices are also vertices of a triangle, say Δ_m ($m \leq t$) some of whose vertices are unassigned, then set t=m and repeat Step 6; otherwise go to Step 7.

Step 7. If the total number of assigned vertices is less than n-1, then set t=t+1 and go to Step 6; otherwise assign the last unassigned machine to the last unassigned site.

Determine the solution cost; Set p=p+1 and t=1.

Step 8. If p ≤ the number of sites then go to Step 5; otherwise select the solution with the minimum cost.

The use of TAA is explained using the numerical example presented below.

5.2.1 Numerical Example with TAA

Given the frequency of trips between 5 machines (matrix (5)), the adjacency time matrix (6), machine sizes (first five rows of table 6) and the clearance between machines (first five rows and columns of

matrix (3)), determine a double row machine layout assuming that an AGV has been selected as the material handling carrier. The orientation of the machines are to be such that the longer side of each machine is parallel to the AGV path. The machines are to be aligned so that their nearest longer sides are equidistant from the AGV path. Note that the entry (i,j) in matrix (3) indicates the required clearance between the shorter sides of machines i and j when they are adjacent to each other. The clearance between the longer sides of machines i and j is determined by the width of the AGV path.

$$\tilde{[f_{ij}]} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 5 & 1 & 4 & 1 \\ 5 & 0 & 3 & 0 & 2 \\ 1 & 3 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 5 \\ 1 & 2 & 0 & 5 & 0 \end{bmatrix} \end{array} \qquad (5)$$

$$[t'_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & 3 & 4 & 3 \\ 3 & 0 & 3 & 4 & 3 \\ 3 & 3 & 0 & 4 & 3 \\ 4 & 4 & 4 & 0 & 4 \\ 3 & 3 & 3 & 4 & 0 \end{bmatrix} \end{array} \qquad (6)$$

From the flow and adjacency time matrices, the adjusted flow matrix F is constructed (see matrix (7)).

$$F = [f_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 15 & 3 & 16 & 3 \\ 15 & 0 & 9 & 0 & 6 \\ 3 & 9 & 0 & 0 & 0 \\ 16 & 0 & 0 & 0 & 20 \\ 3 & 6 & 0 & 20 & 0 \end{bmatrix} \end{array} \qquad (7)$$

Phase 1

Step 1. Set l=1; q_0= 2

The maximum spanning tree T_1 is as follows:

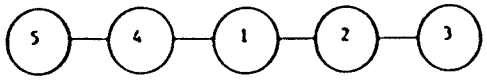


Figure 13: Maximum spanning tree for data in matrix (7)

Step 2. The adjacency matrix for the spanning tree T_1 is as follows:

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

Step 3.

i	{j,k}	Triangle (i,j,k)	Weight of Triangle (i,j,k)
2	{1,3}	(2,1,3)	27
4	{1,5}	(4,1,5)	39

Step 4. Triangle (4,1,5) is selected;

edge {1,5} is added to T_1 and the new graph is labelled as T_2;

(4,1,5) is stored as Δ_1;

Since N=3 < 5-1, and l=1 ≤ (5/3) go to Step 2.

Step 2. The adjacency matrix for T_2 is as follows:

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

Step 3.

i	{j,k}	Triangle (i,j,k)	Weight of Triangle (i,j,k)
1	{2,5}	(1,2,5)	24
2	{1,3}	(2,1,3)	27

Step 4. Triangle (2,1,3) is selected;

edge {1,3} is added to T_2 and the new graph is labelled as T_3;

(2,1,3) is stored as Δ_2;

Since N=5 > 5-1 and l=2 > (5/3) go to Step 5.

Step 5. Arrangement of the triangles in descending order of their weights:

Triangle No.	i	j	k	Weight
1	4	1	5	39
2	2	1	3	27

Phase 2

Five sites of equal area are constructed as shown in figure 14.

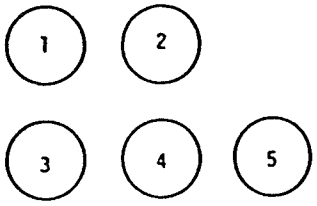


Figure 14: Construction of sites for the example problem

Step 6. Set t=1; p=1.

Assignment using Rule 1:

edges {1,4},{4,5} are selected; vertex 4 is assigned to site 1;

site 2 is selected; vertex 5 is assigned to site 2.

Assignment using Rule 2:

vertex 1 is assigned to site 3;

t=2.

Step 7. Δ_2 is examined;

number of previously assigned vertices of Δ_2 is 1

Assignment of vertices 2,3 using Rule 2:

vertex 2 is assigned to site 4; vertex 3 is assigned to site 5.

Step 8. Solution cost is 52.

Set t=1; p=2.

Step 9. Go to Step 6.

Step 6. Assignment using Rule 1:

edges {1,4},{4,5} are selected; vertex 4 is assigned to site 2;

site 1 is selected; vertex 5 is assigned to site 1

Assignment using Rule 2:

vertex 1 is assigned to site 4;

t=2.

Step 7. Δ_2 is examined;

number of previously assigned vertices of Δ_2 is 1

Assignment of vertices 2,3 using Rule 2:

vertex 2 is assigned to site 3; vertex 3 is assigned to site 5.

Step 8. Solution cost is 58.

Set t=1; p=3.

Step 9. Go to Step 6.

Step 6. Assignment using Rule 1:

edges {1,4},{4,5} are selected; vertex 4 is assigned to site 3;

site 2 is selected; vertex 5 is assigned to site 1.

Assignment using Rule 2:

vertex 1 is assigned to site 4;

t=2.

Step 7. Δ_2 is examined;

number of previously assigned vertices of Δ_2 is 1

Assignment of vertices 2,3 using Rule 2:

vertex 2 is assigned to site 2; vertex 3 is assigned to site 5.

Step 8. Solution cost is 66.

Set t=1; p=4.

Step 9. Go to Step 6.

Step 6. Assignment using Rule 1:

edges {1,4},{4,5} are selected; vertex 4 is assigned to site 4;

site 1 is selected; vertex 5 is assigned to site 3.

Assignment using Rule 2:

vertex 1 is assigned to site 2;

t=2.

Step 7. Δ_2 is examined;

number of previously assigned vertices of Δ_2 is 1

Assignment of vertices 2,3 using Rule 2:

vertex 2 is assigned to site 1; vertex 3 is assigned to site 5.

Step 8. Solution cost is 62.

Set t=1; p=5.

Step 9. Go to Step 6.

Step 6. Assignment using Rule 1:

edges {1,4},{4,5} are selected; vertex 4 is assigned to site 5;

site 2 is selected; vertex 5 is assigned to site 4.

Assignment using Rule 2:

vertex 1 is assigned to site 2;

t=2.

Step 7. Δ_2 is examined;

number of previously assigned vertices of Δ_2 is 1

Assignment of vertices 2,3 using Rule 2:

vertex 2 is assigned to site 1; vertex 3 is assigned to site 3.

Step 8. Solution cost is 55.

Set t=1; p=6.

Step 9. Select assignment with minimum solution cost 52.

The corresponding layout is shown in figure 15.

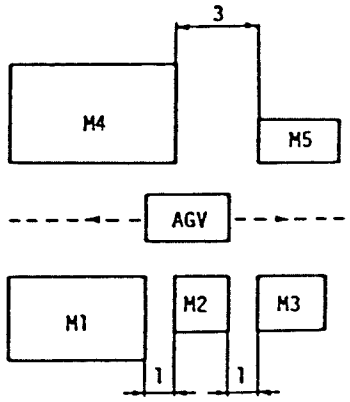


Figure 15: Double-row layout for the example problem

Note that the clearance between the shorter sides of the adjacent machines corresponds to the entries in matrix (3). Also, the clearance between the longer sides of the adjacent machines is equal to the width of the AGV path. The machines have been aligned such that the nearest longer side of each machine is equidistant from the AGV path.

5.3 RESULTS AND DISCUSSION

The modified spanning tree and triangle assignment algorithms were coded in VS FORTRAN and run on an AMDAHL 5870 computer. To test the two algorithms, 36 problems were solved; 16 of these were used to test MSTA and the remaining 20 were used to test TAA.

As mentioned previously, travel time is a better measure of closeness than travel distance, for an FMS. However, we have used travel distance as a measure of closeness only because it enables the comparison of the algorithms presented with the other existing algorithms.

Nine of the sixteen problems used to test MSTA are four-machine layout problems and use the flow data in table 7 and the machine size data in table 8. The clearance between each pair of machines was assumed to be one unit. For the remaining 7 problems, the flow data was taken from Nugent et. al (1968). The machine sizes were assumed to be unequal and are provided in table 9. For the n-machine layout problem, the machines are numbered sequentially from 1 through n ($5 \leq n \leq 20$). Thus, the machine sizes for machines 1,2,3,4 and 5 in the 5-machine layout problem can be obtained from rows 4,2,5,3 and 1 respectively in table 9. The clearance between each pair of machines was assumed to be 0.01 unit.

In the case of TAA, 8 of the test problems were assumed to have machines of equal sizes and the flow and distance data for these problems were obtained from Nugent et al. (1968). The layout pattern was also taken to be the same as in Nugent et al. (1968). The remaining 12 test problems were assumed to have machines of unequal sizes. The machine sizes are provided in table 9 and are to be read as mentioned

above. Eight of the 12 test problems were assumed to have a double-row layout, the number of machines on each of the two rows being as equal as possible. The other 4 test problems were assumed to have a multi-row layout pattern. The number of rows of machines for each of these 5 problems have been provided in table 13. For all the problems used to test TAA, the flow data was taken from Nugent et al. (1968). Also, for the layout problems in which the machine sizes were unequal, the clearance between the machines was assumed to be 0.01 unit. For the layout problems in which the machine sizes are equal, the clearance between the machines was assumed to be zero in order to enable a comparison of TAA with other existing algorithms. For the sake of simplicity, the machines in all the test problems were assumed to be square in shape.

Since single row machine layout problems have not been solved in the literature, the results for the single row four-machine layout problem were compared to the optimal solution (obtained from complete enumeration). As can be seen from table 10, the optimal solution was obtained in six out of nine problems. Table 10 also shows the percentage deviations of the solutions of MSTA from the optimal solutions and the flow dominance for each of the problems. Flow dominance can be defined as the coefficient of variation of the flow data, computed from the flow matrix elements as (Herroelen and Gils, 1985):

100 x standard deviation / mean, where

$$\text{mean} = \{ \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} / n^2 \}$$

$$\text{standard deviation} = \sqrt{\{[\sum_{i=1}^{n} \sum_{j=1}^{n} (f_{ij} - \text{mean})^2]/[n^2 - 1]\}}$$

MSTA appears to provide optimal results when the flow dominance is above 125 %. However, since table 10 considers only four-machine layout problems, no conclusive inference may be drawn. Table 11 provides the solution results of MSTA for 7 more problems in which the number of machines range from 5 to 20. The CPU time and the solution cost for the single-row machine layout problems are also reported in table 11.

Tables 12 and 13 present the solution results of TAA for the double row and multi-row machine layout problems. MSTA and TAA generate good quality solutions. The inaccuracy of data, especially the flow data, does not justify spending too much effort to improve the quality of solutions. However, the biggest advantage of TAA is that the computational time requirement is low. Note that TAA requires 1.27 seconds of CPU time on an AMDAHL 5870 computer, whereas the revised Hillier procedure and FLAC require 22.86 and 23.4 seconds of CPU time on a Prime 750 computer and IBM 4341 computers respectively for the same problem.

Another advantage of TAA is that, being a construction algorithm, it does not require an initial solution unlike most other methods. A comparison of tables 12 and 14 shows that the CPU time for problems with equal machine sizes is almost the same as those for problems with unequal machine sizes.

As mentioned before, most of the existing heuristic algorithms are not designed to solve layout problems with facilities of unequal area. Also, the ones that are designed to do so alter the shape of facilities in the final layout and hence are not applicable to machine layout problems.

In order to further improve TAA's solution quality, it was combined with two improvement algorithms (Heragu and Kusiak, 1986). The first improvement algorithm is a pairwise exchange algorithm of the "greedy" type, discussed in chapter 4. The second algorithm combined with TAA is an in-house version of FRAT developed by Khalil (1973). In addition, TAA was combined with both the "greedy" pairwise exchange algorithm and FRAT. The CPU times and solution costs of the layouts for 10 problems (with facilities of equal area) which were tested using:

- TAA
- TAA combined with "greedy" exchange algorithm
- TAA combined with FRAT
- TAA combined with "greedy" exchange algorithm and FRAT,

are shown in table 14. Among the ten problems, eight are from Nugent et al. (1968), one is from Elshafei (1977) and one is from Steinberg (1961).

By setting q to different values, one can generate different sets of triangles in phase 1 of TAA. Since the layout is based on the sets of triangles generated, a different layout may be generated for each value of q. Thus a layout with a better solution quality than those reported in table 14 may be obtained by suitably selecting the values of q. However, for all problems reported in this thesis, a standard was adopted in setting the value of q, so as to estimate the quality and reliability of the solutions provided by TAA.

TABLE 7

Flow Data for the 4-Machine Layout Problems

Problem # 1

Machine	1	2	3	4
1	0	10	5	0
2	10	0	0	20
3	5	0	0	8
4	0	20	8	0

Problem # 2

Machine	1	2	3	4
1	0	10	15	15
2	10	0	0	5
3	15	0	0	40
4	15	5	40	0

Problem # 3

Machine	1	2	3	4
1	0	40	40	10
2	40	0	0	0
3	40	0	0	10
4	10	0	10	0

Problem # 4

Machine	1	2	3	4
1	0	10	15	20
2	10	0	10	15
3	15	10	0	10
4	20	15	10	0

Problem # 5

Machine	1	2	3	4
1	0.0	0.3	0.0	0.31
2	0.3	0.0	1.0	0.5
3	0.0	1.0	0.0	1.0
4	0.31	0.5	1.0	0.0

Problem # 6

Machine	1	2	3	4
1	0	1	2	39
2	1	0	2	0
3	2	2	0	40
4	39	0	40	0

Problem # 7

Machine	1	2	3	4
1	0	3	8	2
2	3	0	2	2
3	8	2	0	40
4	2	2	40	0

Problem # 8

Machine	1	2	3	4
1	0	2	2	2
2	2	0	2	2
3	2	2	0	0
4	2	2	0	0

Problem # 9

Machine	1	2	3	4
1	0	4	2	4
2	4	0	0	0
3	2	0	0	40
4	4	0	40	0

TABLE 8

Machine Sizes for the 4-Machine Layout Problems

Machine Number	Machine Dimension
1	2 x 2
2	4 x 4
3	6 x 6
4	2 x 2

TABLE 9

Machine Sizes

Machine Number	Machine Dimension
5,8,12,16,22,23,28	0.01 x 0.01
2,14,17,24,29	0.02 x 0.02
4,13,15,18,25,30	0.03 x 0.03
1,10,19,26,27	0.04 x 0.04
3,6,9,11,21	0.05 x 0.05
7	0.08 x 0.08
20	0.09 x 0.09

TABLE 10

MSTA Solution Results for Nine 4-Machine Single-Row Layout Problems

Problem No.	TAA Solution Result	Optimum Result	Percentage Deviation	Flow Dominance
1	225.0	225.0	0	128.52
2	535.0	440.0	+21.59	122.56
3	510.0	510.0	0	135.45
4	465.0	465.0	0	68.31
5	22.4	19.7	+13.57	104.05
6	359.0	359.0	0	164.88
7	318.0	318.0	0	183.27
8	82.0	60.0	+36.67	80.00
9	244.0	244.0	0	212.55

TABLE 11

MSTA Solution Results for Single-Row Machine Layout Problems

No. of Machines	Solution Cost	CPU Time [secs]
5	1.165	0.04
6	2.085	0.04
7	5.420	0.04
8	7.995	0.04
12	31.525	0.04
15	62.624	0.05
20	178.149	0.05

TABLE 12

TAA Solution Results for Double-Row Machine Layout Problems

No. of Machines	Solution Cost	CPU Time [secs]
5	1.14	0.04
6	2.01	0.04
7	3.98	0.05
8	4.95	0.05
12	17.91	0.07
15	34.98	0.12
20	91.47	0.21
30	228.30	1.01

TABLE 13

TAA Solution Results for Multi-Row Machine Layout Problems

No. of Machines	No. of Rows	Solution Cost	CPU Time [secs]
12	4	15.77	0.08
15	5	29.09	0.10
20	5	70.86	0.23
30	6	144.58	1.06

TABLE 14
Objective function values and CPU times of TAA, TAA combined with "greedy" pairwise exchange algorithm, TAA combined with FRAT, and TAA combined with "greedy" pairwise exchange algorithm and FRAT

Problem number	Number of machines	TAA		TAA + "greedy" exchange algorithm		TAA + FRAT		TAA + "greedy" exchange algorithm + FRAT	
		OFV	CPU	OFV	CPU	OFV	CPU	OFV	CPU
1	8	116	0.05	107	0.09	107	0.09	107	0.11
2	12	314	0.07	295	0.14	297	0.16	295	0.16
3	15	596	0.11	580	0.18	584	0.19	584	0.21
4	19	12,245,437	2.40	11,297,537	3.20	10,274,259	3.46	10,274,259	3.51
5	20	1,414	0.23	1,324	0.64	1,340	0.52	1,324	0.67
6	30	3,326	1.27	3,124	3.35	3,133	2.93	3,124	3.47
7	36	5,098	9.05	5,098	10.51	5,032	9.49	5,064	11.10

OFV Objective function value
CPU CPU time on an AMDAHL 5870 computer
Problem numbers 1,2,3,5 and 6 are from Nugent et al. (1968), 4 from Elshafei (1977) and 7 from Steinberg (1961)

5.4 COMPARISON OF COMPUTATIONAL RESULTS OF MPA AND TAA

Nugent et al. (1968) presented eight test problems which have been frequently used for comparing the performance of various algorithms. The backboard wiring problem (Steinberg, 1961), hospital layout problem (Elshafei, 1977) and Krarup (1972) problem have also been used to compare algorithms, but to a lesser extent. However, one common feature of all the above mentioned problems is that the facilities are all assumed to be of equal area and hence the distance between locations is known a priori.

There are very few problems in the literature in which the facilities are of unequal area. An example is the problem presented in Armour and Buffa (1963). In order to make the comparison of algorithms more valid, more layout problems with facilities of unequal area must be included in the comparison.

In this section, the performance of MPA and TAA presented in chapters 4 and 5 is compared with that of 9 heuristic algorithms for six test problems in Nugent et al. (1968). Three algorithms from each of the following classes of algorithms: construction, improvement and hybrid algorithms, are included for comparison purposes. Graph theoretic algorithms could not be included in the comparison because, in the literature, the graph theoretic algorithms with the exception of GASOL, have not been applied to the test problems in Nugent et al. (1968). However, the computational results of GASOL published in Hammouche (1983) do not provide either the layouts or their solution costs.

The selection of the algorithms from each of the three classes is based upon the quality of the solution produced and computation time required by them. Among construction algorithms, MAT (Edwards et al., 1970), the linear placement algorithm LPA (Neghabat, 1974), and FATE (Block, 1978) were selected. CRAFT (Armour and Buffa, 1963), Revised Hillier (Picone and Wilhelm, 1984), and TSP (Hitchings and Cottam, 1976) were selected among the various improvement algorithms. FLAC (Scriabin and Vergin, 1985), the heuristic algorithms in Bazaraa and Kirca (1983) and revised DISCON (Drezner, 1987) were the hybrid algorithms selected for comparison purposes.

The criterion used for comparison is the solution quality and computation time. Solution quality of an algorithm is the ratio of the objective function value of the solution produced by the algorithm and a lower bound expressed as a percentage (Ritzman, 1972). The lower bound is calculated by adding the n products of the largest flow value with the smallest distance, the second largest flow value with the second smallest distance and so on. The computation time provided in table 15 cannot be directly used for comparison because the computation time for each of the algorithms depends upon factors such as programmer's efficiency, computer system used, etc., and these factors are different for each algorithm.

The solution quality and computation time for each of the eleven algorithms are presented in table 15.

From table 15, it can be seen that MPA (combined with the FRAT exchange algorithm) and TAA (combined with the "greedy" exchange algorithm) produce solutions of better quality than:

TABLE 15
Comparison of solution quality and CPU time of TAA and MPA with construction, improvement and hybrid algorithms for six test problems in Nugent et al. (1968)

Number of Machines	MAT SQ	MAT CPU[4]	LPA SQ	LPA CPU[4]	FATE[1] SQ	FATE[1] CPU[4]	CRAFT[1] SQ	CRAFT[1] CPU[4]	RH[1] SQ	RH[1] CPU[4]	MPA[2] SQ	MPA[2] CPU[4]
6	134.1	1.26	104.9	0.10	123.4	2.40	107.8	2.00	104.9	0.24	104.9	0.31
8	140.7	2.10	129.7	0.74	139.2	3.30	124.6	10.00	118.0	0.44	117.6	0.96
12	138.7	6.12	137.5	1.50	134.3	4.80	121.9	70.00	122.6	1.35	123.5	3.56
15	154.7	13.20	129.0	2.50	138.0	6.10	126.5	160.00	121.7	2.38	125.3	5.09
20	143.0	47.49	136.6	3.60	141.6	11.90	132.1	528.00	130.7	5.83	129.0	13.95
30	165.8	346.71	145.7	4.50	151.5	32.40	142.5	3150.00	138.9	22.86	140.6	50.04

Number of Machines	TSP[3] SQ	TSP[3] CPU[4]	FLAC[3] SQ	FLAC[3] CPU[4]	BK[3] SQ	BK[3] CPU[4]	RD[1] SQ	RD[1] CPU[4]	TAA[2] SQ	TAA[2] CPU[4]
6	104.9	0.30	104.9	n.a.	104.9	n.a.	115.9	0.06	104.9	0.05
8	117.6	1.23	117.6	n.a.	117.6	n.a.	130.6	0.08	117.6	0.09
12	118.9	6.84	118.9	4.30	118.9	n.a.	132.6	0.16	121.4	0.14
15	122.1	20.13	122.1	5.00	120.0	30.59	131.7	0.32	121.1	0.18
20	127.9	59.87	128.5	14.60	126.7	156.03	139.7	0.86	130.6	0.64
30	137.9	383.27	137.6	45.00	136.9	320.25	153.6	4.86	139.6	3.35

SQ Solution quality, i.e., the ratio of the objective function values of the solutions produced by the algorithm and a lower bound expressed as a percentage (Ritzman, 1972)

CPU Central processing unit time

n.a. Data not available

1 Solution quality for: FATE is based on the average objective function values reported in Lewis and Block (1978); CRAFT is based on the average objective function values obtained by using randomly selected initial solutions provided in Nugent et al. (1968); RH (Revised Hillier) is based on the average objective function values using randomly selected initial solutions provided in Picone and Wilhelm (1984); RD (Revised DISCON) is based on the average objective function values reported in Drezner (1987).

2 The solutions produced by MPA and TAA were improved once using FRAT (Khalil, 1973) and the "greedy" exchange algorithm, respectively

3 Solution quality for TSP, FLAC and BK (the branch-and-bound based algorithm presented in Bazaraa and Kirca, 1983) are based on the objective function values of the best solutions reported in Hitchings and Cottam (1976), Scriabin and Vergin (1985), and Bazaraa and Kirca (1983), respectively.

4 CPU time for MAT, LPA and CRAFT are from a GE 265 computer, FATE on an ICL 1903T computer, RH (Revised Hillier) on an Prime 750 computer, MPA and TAA on an AMDAHL 5870 computer, TSP on an ICL 470 computer, FLAC on an IBM 4341 computer, BK (the branch-and-bound based algorithm presented in Bazaraa and Kirca, 1983) on a CDC Cyber 70 model 74-28/CDC 6400 computer, RD (Revised DISCON) on an AMDAHL 470/v8 computer, respectively.

- all the three construction algorithms MAT, LPA and FATE,
- Revised DISCON (hybrid) algorithm, for the 6 test problems presented in Nugent et al. (1968).

TAA produced solutions of better quality than CRAFT while MPA produced solutions of better quality than CRAFT for 5 out of the 6 test problems. When compared to the Revised Hillier (improvement) algorithm, MPA (combined with the FRAT exchange algorithm) produced: solutions of better quality than the former for 2 problems; solutions of inferior quality for 3 problems; and same quality solution for one problem. On the other hand, TAA (combined with "greedy" exchange algorithm) produced solutions of better quality than revised DISCON for 5 of the 6 test problems. From table 15 it can also be verified that TAA combined with the "greedy" exchange algorithm produced solutions of equal or better quality than FLAC for 3 test problems and solutions of inferior quality for the other 3 problems.

Based upon the relative performance of the computer systems used for running TAA, FLAC and revised DISCON (Drezner, 1987) which are 620, 40 and 310 respectively, it can be estimated that the AMDAHL 5870 computer used for running MPA and TAA is 16 times faster than the IBM 4341 computer used for running FLAC and twice as fast as the AMDAHL 470/v8 computer used for running revised DISCON (see Ein-Dor, 1985). Considering the above and the CPU times of TAA and FLAC shown in table 15, one can observe that the computational time requirement of TAA is lower than that of FLAC and revised DISCON. However, since the computation time depends upon the programmer's efficiency, program compiler used, etc., no conclusive inference may be drawn. Furthermore,

TAA can solve layout problems in which the machines are of unequal area, whereas FLAC cannot.

The other two algorithms (TSP and BK) listed in table 15 appear to provide better quality solutions than TAA and MPA. However, they cannot solve unequal area machine layout problems. Also, the solution quality reported for TSP is based on the objective function values of the best solution generated by it. Since TSP is an improvement algorithm, the solution generated depends upon the initial solution provided. Hence, based on the solutions provided in Hitchings and Cottam (1976), it cannot be determined whether TSP is superior to MPA and TAA.

The only algorithm which consistently produces solutions of equal or better quality than TAA or MPA is the branch-and-bound based algorithm developed by Bazaraa and Kirca (1983). However, it has a very high computation requirement and cannot be used to solve large scale layout problems. By computation requirement is meant both memory and computation time requirement.

The main advantage of TAA and MPA is that they can solve large scale layout problems, require low computational time and can be used to solve unequal area machine layout problems as well.

For the application presented (i.e., solving the machine layout problem), where it is desired to determine the locations of hundreds of machines of unequal area, it seems that the algorithms MPA and TAA presented in this thesis will provide solutions of good quality in low computational time.

Chapter VI

KNOWLEDGE-BASED SYSTEM FOR MACHINE LAYOUT

In this chapter, a knowledge based system designed to solve the machine layout problem is presented. The data requirement and input format required by the system are discussed in the next section. The problem solving approach of KBML is presented and the system is illustrated with a numerical example. Experience has shown that for most industrial layout problems, the two algorithms presented in chapters 4 and 5 are likely to be used in KBML (Heragu and Kusiak, 1988a).

6.1 DATA INPUT IN KBML

KBML obtains the declarative knowledge, i.e. data for the problem to be solved, from the user, in an interactive mode. The user is provided with the exact format in which data is to be input. The following data are required by KBML:

i) number of machines to be assigned
ii) flow matrix
iii) clearance matrix
iv) relationship indicator matrix
v) machine dimensions
vi) location restrictions (if any) for the machines
vii) type of layout
viii) type of material handling carrier

ix) dimensions of the floor plan.

Details regarding the above data are provided below:

Number of machines to be assigned: The number of machines to be assigned is the total number of machines in the layout problem minus the number of machines whose locations are restricted to certain sites (item vi above).

Flow matrix: The elements of the flow matrix indicate the frequency of trips to be made by the material handling carrier between each pair of machines in a given time horizon.

Clearance matrix: Elements of the clearance matrix indicate the minimum distance by which machines i and j are to be separated if they are located adjacently in the layout.

Relationship indicator matrix: KBML uses three relationship indicators namely: A_{ij}, O_{ij} and X_{ij}, which indicate the adjacency requirement that is to be satisfied while placing machines i and j in the layout. An entry A_{ij} (X_{ij}) in row i and column j of the relationship indicator matrix means that corresponding machines i, j are (not) to be located in adjacent sites. Entry O_{ij} indicates that the location of machine i with respect to machine j is to be determined by the algorithm which solves the layout problem.

The relationship indicator matrix is somewhat similar to the relationship chart which was first suggested in Muther (1973). The

relationship chart shows the closeness desired between pairs of machines and consists of entries A, E, I, O, U or X. For any pair of machines (i,j), the values A, E, I, O, U and X indicate that the closeness desired between facilities i and j is absolutely necessary, especially important, important, ordinary, unimportant and undesirable, respectively. In contrast, the relationship indicator matrix used in KBML consists only of A, O and X entries whose interpretation was provided in the previous paragraph.

The reason for using the relationship indicator matrix as opposed to the relationship chart is as follows: KBML uses the relationship indicator matrix not to determine the closeness desired between machines but to determine whether a pair of machines must:

i) be located in adjacent sites,

ii) not be located in adjacent sites, and

iii) be located as suggested by the algorithm which solves the layout problem.

The closeness desired between each pair of machines can be obtained from the flow matrix and it was therefore decided not to use the relationship chart in KBML.

Machine dimensions: Machine dimensions refer to the length and breadth of each machine and are used to determine whether space constraints are violated in a layout.

Location restrictions: It may sometimes be desirable to restrict the location of a particular machine(s) to a particular site(s). Such information may be easily recorded in KBML.

Type of layout: Type of layout refers to the type of arrangement of machines on the floor plan. As shown in chapter 1, there are four types of machine layouts in automated manufacturing systems.

Type of material handling carrier: The type of material handling carrier selected has an impact on the type of layout. In order to determine the type of layout, KBML requires the user to input the type of material handling carrier selected. On the other hand, if the type of layout is provided, KBML suggests a suitable material handling carrier.

Dimensions of the floor plan: This information is required so that KBML can determine whether the arrangement of machines violates space constraints. It is assumed that the floor plan is rectangular in shape and the user is required to input the length and breadth of the floor plan.

Since LISP is an efficient language for list processing, the declarative knowledge in KBML is mostly represented in the form of lists. Usually flow, clearance, distance and relationship indicator data are in matrix form. But in KBML they are entered in the form of lists. The flow, clearance, distance and relationship indicator data are subsequently stored in matrix form. The machine dimension and location restriction data for all the machines are also entered in list form. The number of machines to be assigned, type of layout, type of material handling carrier and dimensions of the floor plan, are entered as single elements. A sample user-system session is shown in figure 19 (presented later in this chapter).

If there is a conflict among the data entered by the user, the system immediately notifies the user and requests the correct data to be entered. For example, if the user has specified that the number of machines in the layout problem is 8 and does not provide 8x8 = 64 flow matrix elements while entering the flow data, the system notifies the user and requests the flow data to be re-entered. On the other hand, if there is no conflict in the data entry but the user has entered the data incorrectly, the error can be rectified towards the end of the data input session when the system asks if there are any corrections to be made. The user then responds appropriately by specifying which data type has to be re-entered, for example, machine dimension, and then enters the corresponding data. The system consists of 12 production rules which determine if the data provided by the user is consistent and are shown in the appendix.

6.2 PROBLEM SOLVING APPROACH

KBML has been implemented using the tandem architecture discussed in Kusiak (1987). The tandem architecture and its variants can be used for many practical problems arising in the manufacturing environment. They are capable of solving ill-structured as well as well-structured problems. A tandem architecture combines the expert system and optimization approaches. It can be thought of as an expert system linked to a data base of models and algorithms. For a given problem, the expert system first selects an appropriate model and algorithm. The problem is solved by the algorithm and the solution produced is evaluated. If the solution is implementable, the expert system accepts it. For example, in the case of the machine layout problem, the solution

(layout) is implementable if space constraints are satisfied and adjacency requirements are met in the layout produced by the expert system. If the solution is not implementable, then the expert system may take one of the following actions:

i) modify certain parameters in the algorithm (if possible) and apply the algorithm again to the problem in order to generate a new solution, check whether it is implementable and repeat the above procedure until an implementable solution is obtained,

ii) modify the solution in order to make it implementable.

Of course, alternative (i) may not be applicable to all algorithms. Even if it is applicable to a particular algorithm, the corresponding parameter can be modified only to a certain extent, beyond which any modification fails to produce solutions. In such a case, i.e., when the parameter(s) in the algorithm cannot be modified any further, and if the solutions produced thus far are not implementable, the expert system adopts alternative (ii) mentioned above. Note that the system may use alternative (i) to also improve the current solution. KBML which is a variant of the tandem system discussed above, uses alternative (i) to improve the current solution and alternative (ii) to make a solution implementable (if necessary).

6.3 STRUCTURE OF KBML

The structure of KBML is shown in figure 16 and its four main components are discussed briefly.

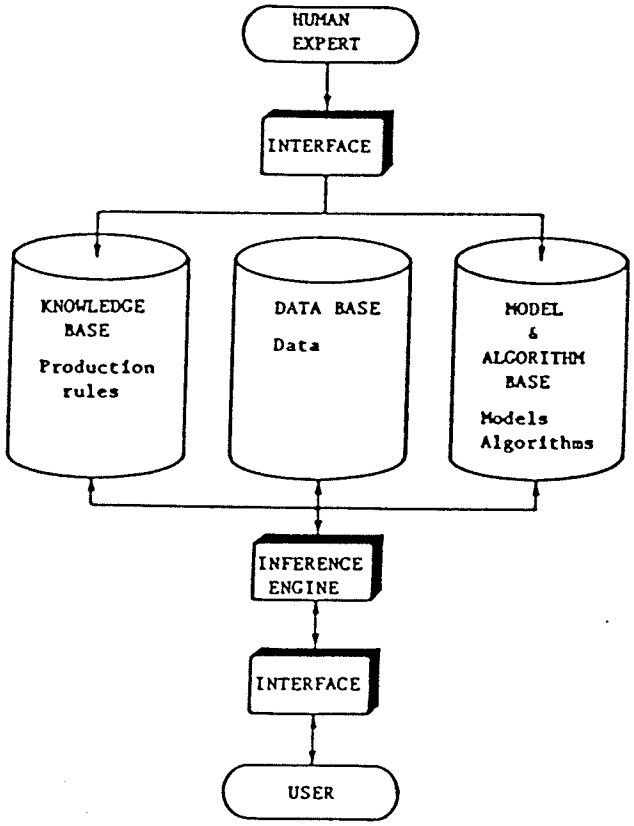


Figure 16: Structure of KBML

Data base: The data base consists of data related to the machine layout problem. KBML interacts with the user and obtains the required data and stores them in the data base.

Model and algorithm base: The models and algorithms related to the layout problem are stored in the model and algorithm base. Each model is represented as a frame. The model representation scheme in KBML is illustrated in figure 17. In the figure, OBJ_FUN denotes the objective function of model M1. LHS_i and RHS_i denote the left and right hand sides of constraint i respectively. IC_i indicates whether constraint i is an equality or inequality constraint. If IC_i is an inequality constraint, its sign is also indicated.

```
((MODEL M1) (OBJ_FUN O) ((LHS  LHS  IC )
                           1    1   1

                         (LHS  LHS  IC )
                            2    2   2

                           .    .    .

                           .    .    .

                           .    .    .

                         (LHS  LHS  IC )))
                            n    n   n
```

Figure 17: Model representation in KBML

Knowledge Base: The knowledge base consists of rules for solving the machine layout problem. There are five classes of rules in KBML:

i) Class 1 rules for determining the type of layout or the type of material handling carrier,

ii) Class 2 rules for selecting an appropriate model and algorithm for the layout problem,

iii) Class 3 rules for making initial assignments based on input data,

iv) Class 4 rules for varying parameters within the algorithm (if applicable), and

v) Class 5 rules for checking whether the layout is implementable.

The above 5 classes of rules are provided in the appendix. To solve the layout problem, the five classes of rules are applied sequentially beginning from Class 1 rules.

KBML requires the user to indicate the desired type of layout. Based on this data, KBML can suggest a suitable material handling carrier depending upon the dimensions of the floor plan. If the user is not able to provide the type of layout and if the type of material handling carrier is known, then based on dimensions of the floor plan, KBML can suggest a suitable type of layout. Two sample rules which do so are shown below.

Rule R06: **IF type of layout is single-row**

AND one of the dimensions of the floor plan is > twice the reach of the robot

AND the other dimension of the floor plan is > the reach of the robot

THEN use robot as material handling carrier and adopt a circular layout and apply Rule R16.

(Note: Rule R16 is presented later in this chapter).

Rule R12: **IF type of material handling carrier is robot**

THEN use circular single-row layout and apply Rule R16.

From Rule R06 above, it can be observed that if one of the dimensions of the floor plan is greater than the reach of the robot and the other dimension of the floor plan is greater than twice the reach of the robot, a circular single-row layout is suggested. If the dimensions of the floor plan are such that either a linear single-row layout or a circular single-row layout can be accommodated, KBML suggests the latter because an AGV required by the linear single-row layout is more expensive than a handling robot (of comparable capacity) required by the circular layout.

Thus Class 1 rules allow KBML to determine either:

i) the type of layout given the type of material handling carrier to be used and the dimension of the floor plan, or

ii) the type of material handling carrier given the type of layout.

When the type of layout and type of material handling carrier are both unknown, the system uses a default value of "single-row" for the layout and determines if such a layout can be accommodated within the boundaries of the floor plan. If a single-row layout can be accommodated, the system determines whether a circular single-row layout is possible. If it is possible, then a robot is suggested as the material handling carrier. If not, an AGV is suggested as the material handling carrier. If a single-row layout cannot be accommodated, the system determines if a double-row is possible. If not, a multi-row layout is suggested.

Class 1 rules consist of 5 meta-rules and 20 first-order rules. A sample meta-rule and first-order rule are shown below. The meta-rules activate the first-order rules. The first-order rules are further categorized into three classes of rules namely: Class 1A, 1B and 1C rules. If the type of layout is known and the type of material handling carrier is not, Class 1A rules are activated. If the type of material handling carrier is known and the type of layout structure is not, Class 1B rules are activated. If the type of material handling carrier and type of layout structure are both unknown, Class 1C rules are activated.

Meta-rule R02:

IF type of layout is unknown and type of material handling carrier is known

THEN apply rule R14 of Class 1B.

First-order rule R14:

IF type of material handling carrier is gantry robot

THEN use multi-row layout.

Class 2 rules are capable of selecting an appropriate model and algorithm for solving the given problem. As was demonstrated in chapter 3, the machine layout problem can be modelled as a linear or a non-linear program. In the past, the machine layout problem has been modelled as a quadratic assignment problem, quadratic set covering problem, linear mixed-integer program, etc. The latter models cannot be solved optimally in an acceptable time if the number of machines in the layout problem is greater than 8. Moreover, the QAP is applicable only when the machines are of equal sizes. Thus, it can be seen that each model is applicable to a particular problem scenario. In table 16, the model and algorithm selected by KBML for twelve problem scenarios are provided. An X entry in table 16 and algorithm in a row in which X appears, can be used for the layout problem in the corresponding column. References to the models and algorithms are also provided in table 16. For example, it can be observed that for a multi-row layout problem involving less than 15 machines of equal sizes, KBML selects model M5, i.e., the quadratic assignment problem (Koopmans and Beckmann, 1957) and uses the heuristic algorithm presented in Heragu and Kusiak (1987a).

A sample rule which selects the model and algorithm for a given problem is provided below.

Rule R16: **IF** **number of machines to be assigned is ≥ 8**
AND **the type of machine layout is single-row**
THEN **select model M2 and solve the model using algorithm A1.**

From table 16 it can be observed that model M2 and algorithm A1 in rule R16 refer to the linear program with absolute values in the objective

TABLE 16
Model and algorithm selected by KBML for twelve problem scenarios

		Problem Scenario										
Model Number	Reference	Machines of equal sizes	Machines of unequal sizes	NMP < 8	NMP 8-15	NMP >15	Single-row layout	Multi-row layout	Algorithm Number	Reference	Heuristic	Optimal
M1 (NLP)	*		X	X	X			X	A1	Heragu and Kusiak (1987a)	X	
M1 (NLP)	*		X			X		X	A2	Heragu and Kusiak (1988)	X	
M2 (LCP)	**	X	X		X	X	X		A1	Heragu and Kusiak (1987a)	X	
M3 (LMIP)	Heragu and Kusiak (1987a)	X	X	X			X		A3 ***	Press et al. (1986)		X
M4 (LMIP)	Heragu and Kusiak (1987a)	X				X		X	A1	Heragu and Kusiak (1987a)	X	
M5 (QAP)	Koopmans and Beckmann (1957)	X		X	X			X	A4	Heragu and Kusiak (1987a)	X	

NMP - Number of machines in the layout problem
NLP - Non-linear program
LCP - Linear program with continuous variables
LMIP - Linear program with mixed {0,1} integer variables
QAP - Quadratic assignment problem
* - Model M1b presented in chapter 3
** - Presented in chapter 3
*** - Simplex based branch-and-bound algorithm

function and constraints (model M1 presented in chapter 3) and the modified penalty algorithm presented in chapter 4.

Thus it can be seen that the model and algorithm selected by KBML depend upon the nature of the problem, namely, machine sizes, number of machines in the layout problem and type of machine layout.

Class 3 rules are used to make initial assignments. The initial assignments may be specified by the user or decided by KBML. For example, if AGV is used as the material handling carrier, then it requires a battery charging station. It is advantageous to assign machines with maximum flow value between them to adjacent sites near the battery charging station. For, if this is done, the AGV spends less time in travel to the battery charging station.

User desired assignments have priority over the assignments done by KBML. For example, if the user desires to locate machines with maximum flow value between them to sites which are not near the battery charging station, the system does not attempt to relocate these machines near the battery charging station. Class 3 consists of 6 rules.

As mentioned in section 5, it is possible to modify certain parameters in some of the algorithms. Class 4 rules are used for changing these parameters. For every modified value of the parameter, the algorithm often provides a different solution (layout). The solution generated by the algorithm is evaluated for each value of the modified parameter. If the modification of the parameter in the algorithm leads to a better solution, the process is continued; otherwise it is terminated and the layouts obtained are evaluated for implementability by Class 5 rules. A layout is implementable if:

i) adjacency requirements (between pairs of machines) specified by the user are met,

ii) location restrictions of machines specified by the user are satisfied, and

iii) space constraints are not violated.

Class 5 rules check whether the machine layout is implementable. If a layout is implementable, its solution cost is computed and provided to the user. If not, the solution is modified to make it implementable.

Inference Engine: KBML uses a forward-chaining inference strategy. The inference engine attempts to match the data concerning type of material handling carrier and type of layout with the IF part of the meta-rules in Class 1. If the match with the IF part of a rule is successful, then the rule fires other first-order rules. The first-order rules suggest either the type of layout or the type of material handling carrier to be used depending upon which rule has been fired. The control is then directed to Class 2 rules. The inference engine attempts to match the data provided by the user (number of machines to be assigned) and the data created by the first-order Class 1 rules (type of layout) with the IF part of Class 2 rules. If a successful match is found in any rule, the THEN part indicates the model and algorithm that are to be used to solve the given layout problem.

Similarly, using the forward-chaining strategy, the inference engine uses Class 3 rules to perform the user desired assignments and also some assignments based on the domain knowledge stored in the knowledge base. As mentioned before, such knowledge is represented in the form of production rules in KBML. A sample production rule is provided below:

Rule R34: **IF** **type of material handling carrier used is AGV**

AND **the assignment of machines i,j with maximum flow value between them are not restricted to any particular site**

THEN **locate battery charging station near one end of the layout and assign machines i and j to sites which are adjacent to the battery charging station.**

Rule R34 ensures that the AGV spends less time in travel to the battery charging station, by assigning machines with maximum flow value between them near the battery charging station. The inference engine applies Class 4 and Class 5 rules in a way similar to the other rules.

The control flow from Class 1 rules to Class 5 rules in KBML is illustrated in figure 18. As shown in the figure, control is directed back and forth between Class 4 and Class 5 rules. Using Class 4 rules, the expert system modifies a parameter of the algorithm selected, invokes the algorithm and calls Class 5 rules. Class 5 rules evaluate the solution produced by the algorithm (for the new value of the parameter) for implementability. If the solution is implementable, its cost is computed. If not, the solution is modified in order to make it implementable and the corresponding cost is computed. If there is an improvement in the solution cost, the current solution is stored as the best solution. The system uses Class 4 rules to vary the modifiable parameter of the algorithm. This process of modifying the parameter, solving the problem for its new value, evaluating the solution produced for implementability, computing the solution cost and checking the current solution cost with that of the best available solution is repeated until the current solution cannot be improved any further.

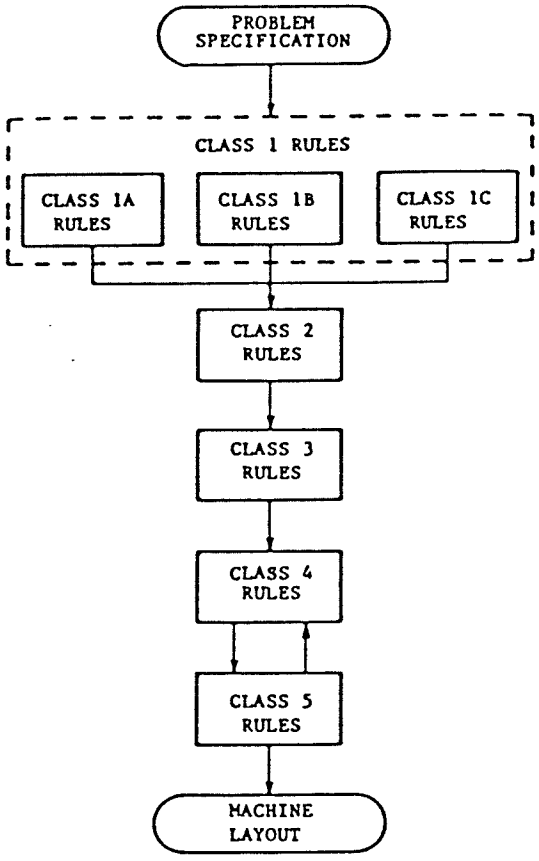


Figure 18: Control flow in Knowledge-based System for Machine Layout (KBML)

The solution provided by the system depends on the algorithm selected for solving the layout problem. For example, if TAA is selected, then the solution provided by the system is in the form of a list. This list indicates the sites to which each machine is assigned. On the other hand, if MPA is selected for solving the layout problem, then the solution provided is also in the form of a list which indicates the coordinates of the center of each machine.

The knowledge base in KBML consists of 59 rules. New rules can be easily added when required. KBML is coded in Common LISP and implemented on a Symbolics 3650 machine.

KBML is illustrated using the numerical example presented below.

6.4 NUMERICAL EXAMPLE

Determine a machine layout for the following data:

i) number of machines to be assigned is 8

ii) flow matrix

Machine	1	2	3	4	5	6	7	8
1	0	2	8	1	1	0	0	2
2	2	0	3	0	2	2	2	0
3	8	3	0	0	0	0	0	0
4	1	0	0	0	5	2	2	10
5	1	2	0	5	0	10	0	0
6	0	2	0	2	10	0	1	1
7	0	2	0	2	0	1	0	10
8	2	0	0	10	0	1	10	0

(1)

iii) clearance matrix

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

(2)

iv) relationship indicator matrix

$$
\begin{array}{c} \\ \text{Machine} \end{array}
\begin{array}{c}
\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & A & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & X \\
0 & 0 & 0 & 0 & 0 & 0 & X & 0 \\
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & A & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & X & 0 & A & 0 & 0 & 0 \\
0 & X & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\qquad \text{Machine} \qquad (3)
$$

v) machine dimensions

Machine Number	Dimension
1	20 x 20
2	10 x 10
3	15 x 15
4	10 x 10
5	15 x 20
6	15 x 25
7	10 x 10
8	10 x 15

vi) location restriction for machines is as follows:

machine 6 is to be located at site 6

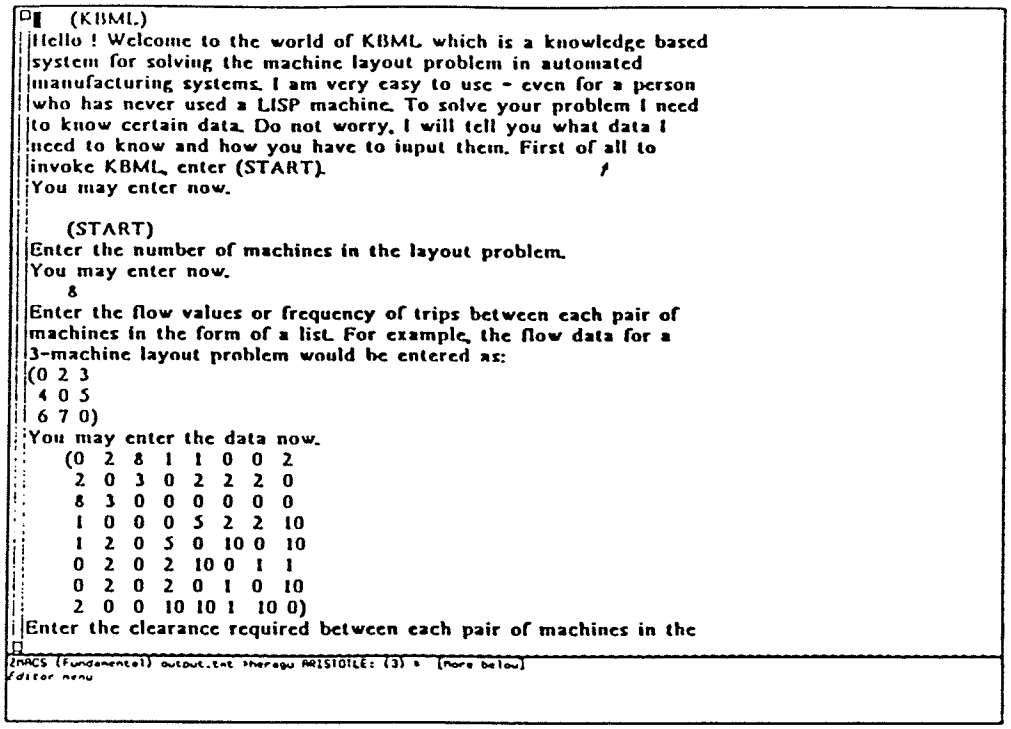
vii) type of layout is single-row

viii) type of material handling carrier is unknown

ix) dimensions of floor plan are 115 x 30

The data entry for the above example problem is shown in figure 19.

Based on the dimensions of the floor plan and type of layout provided by the user, the system suggests that an AGV is to be used as the material handling carrier and that a linear layout be adopted. The final solution obtained at a cost of 2006.50 is shown in figure 20.



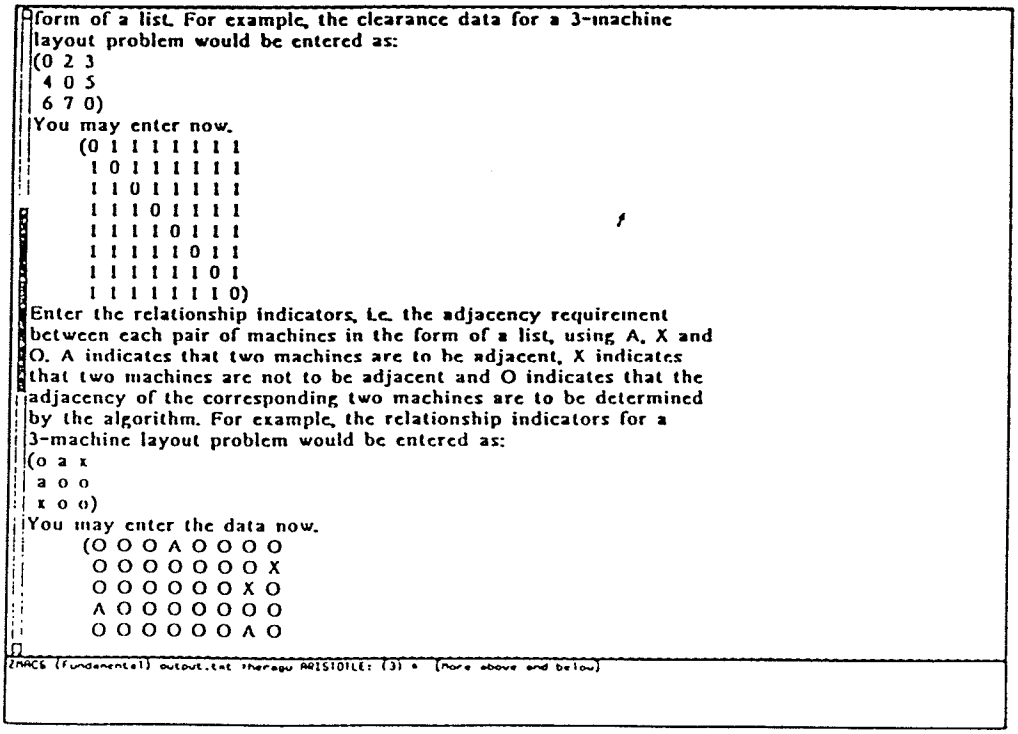
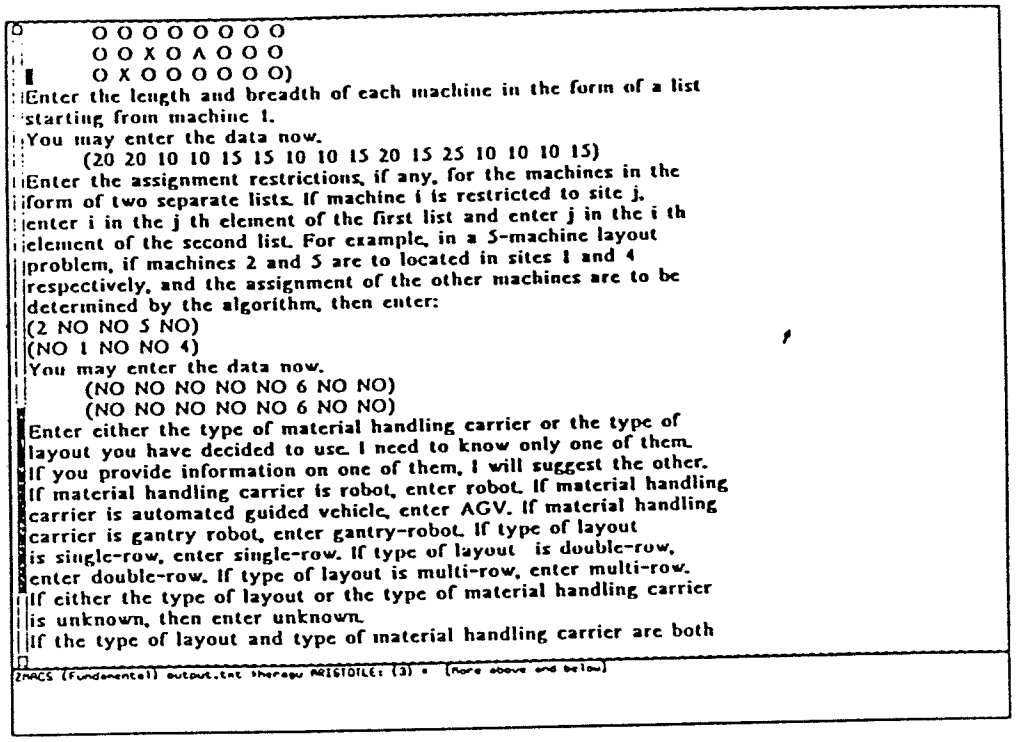


Figure 19: Sample user-system session in KBML (continued on next page)



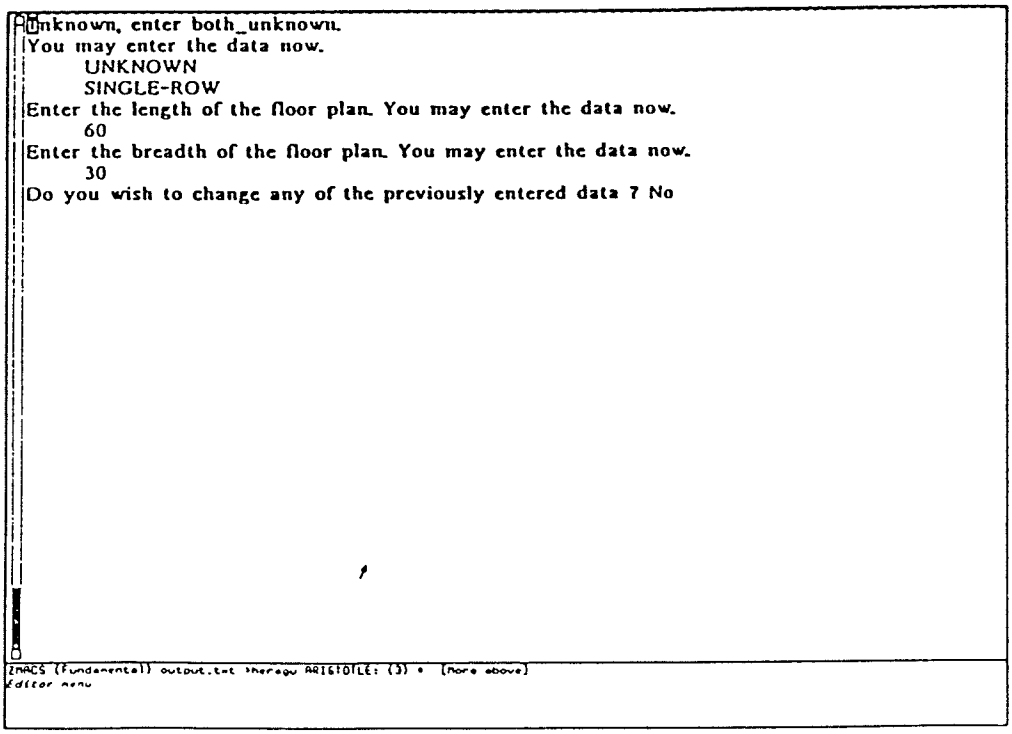
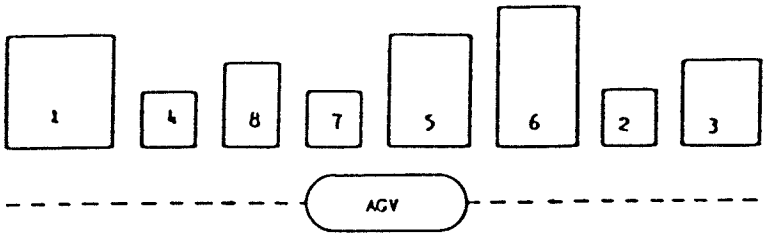


Figure 19: Sample user-system session in KBML



1,...,8 machines
AGV automated guided vehicle

Figure 20: Layout generated by KBML

Chapter VII

CONCLUSION

In this thesis, the machine layout problem in automated manufacturing systems was addressed. Four patterns of layout were identified. The motion characteristic of an AGV was also analyzed. A new approach to modelling the layout problem was presented. The main intent was to explore a continuous model that appears to be computationally easier to solve than the QAP (which has been traditionally used to model the layout problem). As shown in chapter 4, the models may be solved using commercial computer codes. The use of specialized algorithms for solving the models presented will more likely produce solutions of even better quality than those reported in this thesis. The models developed have the following advantages:

- models M3, M3a and M3b are perhaps the first models which formulate the machine layout problem in which the machines are of unequal area,
- the linear models have a compact form and can be used to solve large scale layout problems
- for the models presented in this thesis the location of sites need not be known a priori as in the case of many other existing models for the layout problem.

The computational results provided for the single-row and multi-row layout problem indicate that MPA produces solutions of good quality. It

should be noted that for some problems solved, optimal solutions were not obtained because of the limitations of the penalty method that was used. Using more sophisticated codes for solving the unconstrained minimization problem, or codes for solving the constrained minimization problem, one may be able to obtain better quality solutions. With the development of more efficient integer programming algorithms, the linear mixed integer models may become useful as well.

It was also discussed that the quadratic assignment problem can be used to formulate only certain types of layout problems, i.e., problems in which the location of sites are known a priori. It cannot be used to formulate the machine layout problem because, in general, the machine sizes are not equal and hence the location of sites, which depend upon the sequence of machines, are not known a priori. To solve the MLP, two new algorithms were presented. The algorithm for solving multi-row layout problems, TAA, was shown to provide solutions of better quality than other construction algorithms published to date for six test problems commonly used in the literature.

There is scope for improving the solution quality of the triangle assignment algorithm at the expense of slightly higher computation costs. The algorithm has the following features:

- it considers flow as well as non-flow factors,
- it has very low computational time requirement,
- it generates good quality solutions when compared to CRAFT, ALDEP, PLANET and MATCH (Montreuil et al., 1987) for problems in which facilities are of unequal area (Heragu and Kusiak, 1986). (CRAFT

has been considered to be one of the most efficient algorithms for solving the facility layout problem),

- TAA combined with the greedy exchange algorithm provides solutions of better quality than those obtained by other construction algorithms which have been tested on the problems in Nugent et al. (1968),
- TAA has no restriction on the problem size,
- TAA can be used for problems with high flow dominance as well as for problems with low flow dominance,
- it can be used for problems with machines of equal and unequal area,
- no initial solution is required, and
- the CPU time is almost the same for problems with equal and unequal machine sizes.

TAA combined with a "greedy" exchange algorithm produces solutions of better quality than many other algorithms for the layout problem and also requires low computation time.

The flow data in a machine layout problem is usually not accurate. This is because the flow between machines depends on the production schedule and the production schedule cannot be predicted accurately, due to changing market demand, unexpected repairs, etc. In such cases, one might ask if it is worthwhile to use algorithms which require significantly higher CPU time to obtain a slightly better solution. For most practical purposes, what is required is a reasonably good solution (not necessarily the optimal one), with low computation time requirement. TAA is capable of producing good quality solutions and requires low CPU time.

The heuristic algorithms MPA and TAA designed for solving the layout problem, are easy to follow and implement. The main intent of developing these algorithms was to incorporate them in a knowledge-based system.

During the last thirty years considerable effort has been invested in research on the layout problem. Optimization techniques have been widely used for solving the machine layout problem. If knowledge-based systems are to be successfully used for solving the machine layout problem, it is clear that they have to take advantage of the optimization approach as well. KBML is an effort in that direction (Kusiak and Heragu, 1989).

Since lists are easily and efficiently manipulated in Common LISP, KBML requires the user to input most of the data in list form. It should be noted that KBML is easy to implement. New rules can be easily added to the knowledge base. Since the number of rules is relatively small, the computation time required by KBML is low.

KBML has the potential to produce solutions of good quality when compared to the two existing knowledge-based systems for machine layout - IFLAPS and FADES. The reason is that KBML uses tested efficient models and algorithms for solving the layout problem whereas IFLAPS uses simple rules of thumb in determining the machine layout. FADES, on the other hand, can solve small scale layout problems in which the machines are of equal area only.

The advantages of KBML are as follows:

- KBML can solve large scale industrial layout problems and requires low computation time,

- it can be used to solve layout problems with machines of equal or unequal sizes, single-row or multi-row layout problems, etc.,
- it uses efficient models and algorithms available to solve the layout problem,
- it allows modification of parameters within an algorithm in order to generate new solutions, and
- it considers quantitative as well as qualitative data while solving the layout problem.

REFERENCES

1. D. Adolphson and T.C. Hu (1973), Optimal linear ordering, SIAM Journal of Applied Mathematics, Vol. 25, pp. 403-423.

2. G.C. Armour and E.S. Buffa (1963), A heuristic algorithm and simulation approach to relative allocation of facilities, Management Science, Vol. 9, No. 2, pp. 294-300.

3. E. Balas (1965), An additive algorithm for solving linear programs with zero-one variables, Operations Research, Vol. 13, pp. 517-546.

4. E. Balas, and Mazzola, J.B., (1980), Quadratic 0-1 programming by a new linearization, presented at the TIMS/ORSA meeting, Washington, D.C.

5. M.S. Bazaraa (1975), Computerized layout design: A branch and bound approach, AIIE Transactions, Vol. 7, No. 4, pp. 432-437.

6. M.S. Bazaraa and A.N. Elshafei (1979), An exact branch and bound procedure for quadratic assignment problems, Naval Research Logistics Quarterly, Vol. 26, pp. 109-121.

7. M.S. Bazaraa and O. Kirca (1983), A branch-and-bound-based heuristic for solving the QAP, Naval Research Logistics Quarterly, Vol. 30, pp. 287-304.

8. M.S. Bazaraa and M.D. Sherali (1980), Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem, Naval Research Logistics Quarterly, Vol. 27, No. 1, pp. 29-41.

9. M.S. Bazaraa and M.D. Sherali (1982), On the use of exact and heuristic cutting plane methods for the quadratic assignment problem, Journal of Operations Research Society, Vol. 33, No. 11, pp. 991-1003.

10. M.S. Bazaraa and C.M. Shetty (1979), Nonlinear Programming: Theory and Algorithms, John Wiley, New York, NY.

11. M. Beghin-Picavet and P. Hansen (1982), Deux problemes d'affectation non lineaires, R.A.I.R.O Recherche Operationelle, Vol. 16, No. 3, August, pp. 263-276.

12. T.E. Block (1977), A note on "Comparison of computer algorithms and visual based method for plant layout" by M. Scriabin and R.C. Vergin, Management Science, Vol. 24, No. 2, pp. 235-237.

13. T.E. Block (1978), FATE: A new construction algorithm for facilities layout, Journal of Engineering Production, Vol. 2, pp. 111-120.

14. T.E. Block (1979), On the complexity of facilities layout problems, Management Science, Vol. 25, No. 3, pp. 280-285.

15. J.A. Bondy and U.S.R. Murty (1976), Graph Theory With Applications, Elsevier, New York, N.Y.

16. J. Browne, W.W. Chan and K. Rathmill (1985), An integrated FMS design procedure, Annals of Operations Research, Vol. 3, pp. 207-237.

17. E.S. Buffa (1955), Sequence analysis for functional layouts, The Journal of Industrial Engineering, Vol. 6, pp. 12-13, 25.

18. E.S. Buffa (1976), On a paper by Scriabin and Vergin, Management Science, Vol. 23, No. 1, pp. 104.

19. E.S. Buffa, G.C. Armour and T.E. Vollmann (1964), Allocating facilities with CRAFT, Harvard Business Review, Vol. 42, pp 136-158.

20. J.L. Burbidge (1975), The Introduction of Group Technology, Halsted Press and John Wiley, New York, N.Y.

21. R.E. Burkard (1973), Die storungsmethode zur losung quadratisches zuordnungsprobleme, Operations Research Verfahren, Vol. 16, pp. 84-108.

22. R.E. Burkard (1984), Locations with spatial interaction - Quadratic assignment problem, in R.L. Francis and P.B. Mirchandani (Eds.), Discrete Location Theory, Prentice-Hall Inc., Englewood Cliffs, NJ.

23. R.E. Burkard (1984a), Quadratic assignment problems, European Journal of Operational Research, Vol. 15, pp. 283-289.

24. R.E. Burkard and T. Bonniger (1983), A heuristic for quadratic boolean program with applications to quadratic assignment problems, European Journal of Operational Research, Vol. 13, pp. 374-386.

25. R.E. Burkard and K.H. Stratman (1978), Numerical investigations on quadratic assignment problems, Naval Research Logistics Quarterly, Vol. 25, pp. 129-144.

26. A.S. Carrie, J.M. Moore, M. Roczniak, J.J. Seppanen (1978), Graph theory and computer-aided facilities design, OMEGA, Vol. 6, No. 4, pp. 353-361.

27. U. Cinar (1975), Facilities planning: a systems analysis and space allocation approach, in C.M. Eastman, Ed., Spatial Synthesis in Computer-Aided Building Design, John Wiley, New York, N.Y.

28. D.R. Coleman (1977), Plant layout: computer versus humans, Management Science, Vol. 24, No. 1, pp. 107-112.

29. R.W. Conway and W.L. Maxwell (1961), A note on the assignment of facility location, The Journal of Industrial Engineering, Vol. 12, pp. 34-36.

30. M.P. Deisenroth and J.M. Apple (1972), A computerized plant layout analysis and evaluation technique, Technical Papers, 1972, Annual AIIE Conference, Norcross, GA.

31. D.R. Dolk and B.R. Konsynski (1984), Knowledge representation for model management systems, IEEE Transactions on Software Engineering, Vol. SE-10, No. 6, pp. 619-628.

32. Z. Drezner (1980), DISCON: a new method for the layout problem, Operations Research, Vol. 25, No. 6, pp. 1375-1384.

33. Z. Drezner (1987), A heuristic procedure for the layout of a large number of facilities, Management Science, Vol. 33, No. 7, pp. 907-915.

34. K.N. Dutta and S. Sahu (1982), A multigoal heuristic for facilities design problems : MUGHAL, International Journal of Production Research, Vol. 20, No. 2, pp. 147-154.

35. P. Eades, L.R. Foulds and J. Giffin (1982), An efficient heuristic for identifying a maximal weight planar subgraph, In Lecture Notes in Mathematics No. 952 (Combinatorial Mathematics IX), Springer-Verlag, Berlin.

36. H.K. Edwards, B.E. Gillett and M.C. Hale (1970), Modular allocation technique (MAT), Management Science, Vol. 17, No. 3, pp. 161-169.

37. P. Ein-Dor (1985), Grosch's law re-revisited: CPU power and the cost of computing, Communications of the ACM, Vol. 28, No. 2, pp. 142-151.

38. T.E. El-Rayah and R.H. Hollier (1970), A review of techniques, International Journal of Production Research, Vol. 8, No. 3, pp. 263-279.

39. A.N. Elshafei (1977), Hospital layout as a quadratic assignment problem, Operations Research Quarterly, Vol. 28, No. 1, pp. 167 -179.

40. G. Finke, R.E. Burkard and F. Rendl (1985), Quadratic assignment problems, Working Paper, Dept. of Applied Mathematics, Technical University of Nova Scotia, Halifax, N.S.

41. E.L. Fisher and S.Y. Nof (1984), FADES: knowledge-based facility design, Proceedings of the International Industrial Engineering Conference Chicago, Il, May 6-10, pp. 74-82.

42. L.R. Foulds (1983), Techniques for facilities layout: deciding which pairs of activities should be adjacent, Management Science, Vol. 29, No. 12, pp. 1414-1426.

43. L.R. Foulds, P.B. Gibbons and J.W. Giffin (1985), Facilities layout adjacency determination: an experimental comparison of three graph theoretic heuristics, Operations Research, Vol. 33, No. 5, pp. 1091-1106.

44. L.R. Foulds and D.F. Robinson (1978), Graph theoretic heuristics for the plant layout problem, International Journal of Production Research, Vol. 16, No. 1, pp. 27-37.

45. R.L. Francis and J.A. White (1974), Facilities Layout and Location: An Analytical Approach, Prentice-Hall Inc., Englewood Cliffs, NJ.

46. A.M. Frieze and J. Yadegar (1983), On the quadratic assignment problem, Discrete Applied Mathematics, Vol. 5, pp. 89-98.

47. G.K. Gaschutz and J.H. Ahrens (1968), Suboptimal algorithms for the quadratic assignment problem, Naval Research Logistics Quarterly, Vol. 15, p. 49-62.

48. J.W. Gavett and N.V. Plyter (1966), The optimal assignment of facilities to locations by branch and bound, Operations Research, Vol. 14, pp. 210-232.

49. P.C. Gilmore (1962), Optimal and suboptimal algorithms for the quadratic assignment problem, Journal of the Society for Industrial and Applied Mathematics, Vol. 10, pp. 305-313.

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

51. R.H. Green and L. Al-Hakim (1985), A heuristic for facilities layout planning, OMEGA, Vol. 13, No. 5, pp. 469-474.

52. R.M. Gupta and M.P. Deisenroth (1981), Comments on the complexity rating factors for layout problems, Management Science, Vol. 27, No. 12, pp. 1460-1464.

53. A. Hammouche (1983), Evaluation of an application of graph theory to the layout problem, in Proceedings of ICPR 83, Vol II, August 22-24, 1983, Windsor, Ontario.

54. M. Hanan and J.M. Kurtzberg (1972), A review of the placement and quadratic assignment problems, SIAM Review Vol. 14, No. 2, pp. 324-342.

55. F. Harary (1969), Graph Theory, Addison-Wesley, Reading Mass.

56. S.S. Heragu and A. Kusiak (1988), Machine layout problem in flexible manufacturing systems, Operations Research, Vol. 36, No. 2, March-April 1988.

57. S.S. Heragu and A. Kusiak (1988a), KBML: A knowledge-based system for machine layout, in Proceedings of the International Industrial Engineering Conference, May 22-25, Orlando, Florida, pp. 159-164.

58. S.S. Heragu and A. Kusiak (1987), Analysis of expert systems in manufacturing design, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-17, No. 6, pp. 898-912.

59. S.S. Heragu and A. Kusiak (1987a), Efficient models for the facility layout problem, Working Paper #11/87, Department of Mechanical and Industrial Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.

60. S.S. Heragu and A. Kusiak (1986), A construction algorithm for the facility layout problem, Working Paper #14/86, Department of Mechanical and Industrial Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.

61. W. Herroelen and A.V. Gils (1985), On the use of flow dominance in complexity measures for facility layout problems, International Journal of Production Research, Vol. 23, No. 1, pp. 97-108.

62. P.E. Hicks and T.E. Cowan (1976), CRAFT-M for layout rearrangement, Industrial Engineering, May 1976, 30-35.

63. F.S. Hillier (1963), Quantitative tools for plant layout analysis, Journal of Industrial Engineering, Vol. 14, pp. 33-40.

64. F.S. Hillier and M.M. Connors (1966), Quadratic assignment problem algorithms and the location of indivisible facilities, Management Science, Vol. 13, pp. 42-57.

65. G.G. Hitchings and M. Cottam (1976), An efficient heuristic procedure for solving the layout design problem, Omega, Vol. 4, No. 2, pp. 205-214.

66. IBM (1974), Mathematical Programming System Extended (MPSX/370), Mixed Integer Program/370 (MIP/370), Program Reference Manual, Program Product 5740-XM3, 1st Edition.

67. R.F. Jacobs (1984), A note on SPACECRAFT for multi-floor layout planning, Management Science, Vol. 30, No. 5, pp. 648-649.

68. F.R. Jacobs (1987), A layout planning system with multiple criteria and a variable domain representation, Management Science, Vol. 33, No. 8, pp. 1020-1034.

69. R.V. Johnson (1982), SPACECRAFT for multi-floor layout planning, Management Science, Vol. 28, No. 4, pp. 407-417.

70. B.K. Kaku and G.L. Thompson (1986), An exact algorithm for the general quadratic assignment problem, European Journal of Operational Research, Vol. 23, pp. 382-390.

71. L. Kaufman and F. Broeckx (1978), An algorithm for the quadratic assignment problem using Bender's decomposition, European Journal of Operational Research, Vol. 2, pp. 204-211.

72. L. Kaiman (1970), Computer Architecture Programs, Centre for Environmental Research, 955 Park Square Building, Boston, MA.

73. R.M. Karp and M. Held (1967), Finite-state processes and dynamic programming, SIAM Journal of Applied Mathematics, Vol. 15, pp. 693-718.

74. T.M. Khalil (1973), Facilities relative allocation technique (FRAT), International Journal of Production Research, Vol. 11, No. 2, pp. 183-194.

75. T.C. Koopmans and M. Beckmann (1957), Assignment problems and the location of economic activities, Econometrica, Vol. 25, No. 1, pp. 53-76.

76. J. Krarup (1972), Quadratic assignment, Data, Vol. 3, pp. 12-15.

77. S.R.T. Kumara, R.L. Kashyap and C.L. Moodie (1985), Artificial intelligence techniques in facilities layout planning: The development of an expert system, Technical Report # TR-ERC 86-1, December 1985, School of Industrial Engineering, Purdue University, West Lafayette, Indiana, USA.

78. A. Kusiak (1988), Artificial Intelligence and CIM systems, in A. Kusiak (Ed.), Artificial Intelligence: Implications for CIM, IFS Publications, Kempston, U.K., and Springer-Verlag, New York, N.Y., pp. 3-23.

79. A. Kusiak (1987), Artificial intelligence and operations research in flexible manufacturing systems, Information Processing and Operations Research (INFOR) Vol. 25, No. 1, pp. 2-12.

80. A. Kusiak and S.S. Heragu (1989), Expert systems and optimization, IEEE Transactions on Software Engineering, (forthcoming).

81. A. Kusiak and S.S. Heragu (1987), The facility layout problem, European Journal of Operational Research, Vol. 29, No.3, pp. 229-253.

82. A.H. Land (1963), A problem of assignment with interrelated costs, Operations Research Quarterly, Vol. 14, pp. 185-198.

83. I. Lavallee and C. Roucairol (1985), Parallel branch and bound algorithms, presented at Euro VIII, Bologna, Italy.

84. E.L. Lawler (1963), The quadratic assignment problem, Management Science, Vol. 9, No. 4, pp. 586-599.

85. R. Lee and J.M. Moore (1967), CORELAP - computerized relationship layout planning, Journal of Industrial Engineering, Vol. 18, pp. 195-200.

86. R.R. Levary and S. Kalchik (1985), Facilities layout - a survey of solution procedures, Computers and Industrial Engineering, Vol. 9, No. 2, pp. 141-148.

87. W.P. Lewis and T.E. Block (1980), On the application of computer aids to plant layout, International Journal of Production Research, Vol. 18, No. 1, pp. 11-20.

88. R.S. Liggett (1981), The quadratic assignment problem: an experimental evaluation of solution strategies, Management Science, Vol. 27, No. 4, pp. 442-458.

89. J.D.C. Little, K.G. Murty, D.W. Sweeney and C. Karel (1963), An algorithm for the travelling salesman problem, Operations Research, Vol. 11, pp. 972-989.

90. R.F. Love and J.W. Wong (1976), Solving quadratic assignment problems with rectilinear distances and integer programming, Naval Research Logistics Quarterly, Vol. 23, pp. 623-627.

91. R.F. Love and J.Y. Wong (1976a), On solving a one-dimensional space allocation problem with integer programming, Information Processing and Operations Research (INFOR), Vol. 14, No. 2, pp. 139-143.

92. J.O. Matson and J.A. White (1982), Operations research and material handling, European Journal of Operational Research, Vol. 11, No. 4, December, pp. 309-318.

93. W.L. Maxwell and J.A. Muckstadt (1982), Design of automated guided vehicle systems, IIE Transactions, Vol. 14, No. 2, pp. 114-124.

94. B. Montreuil, H.D. Ratliff and M. Goetschalckx (1987), Matching based interactive facility layout, IIE Transactions, Vol. 19, No. 3, pp. 271-279.

95. J.M. Moore (1974), Computer aided facilities design: an international survey, International Journal of Production Research, Vol. 12, No. 1, pp. 21-44.

96. J.M. Moore (1976), Facilities design with graph theory and strings, Omega, Vol. 4, No. 2, pp. 193-203.

97. T. Muller, (1983),Automated Guided Vehicles, IFS Publications Ltd., Kempston, U.K.

98. K.G. Murty (1983), Linear Programming, John Wiley, New York, N.Y.

99. R. Muther (1955), Practical Plant Layout, McGraw-Hill, New York, N.Y.

100. R. Muther (1973), Systematic Layout Planning, Cahers Books, Boston, MA.

101. R. Muther and K. McPherson (1970), Four approaches to computerized layout planning, Industrial Engineering, February, pp. 39-42.

102. F. Neghabat (1974), An efficient equipment layout algorithm, Operations Research, Vol. 22, pp. 622-628.

103. C.E. Nugent, T.E. Vollmann and J. Ruml (1968), An experimental comparison of techniques for the assignment of facilities to locations, Operations Research, Vol. 16, No. 1, pp. 150-173.

104. C. O'Brien and S.E.Z. Abdel Barr (1980), An interactive approach to computer aided facility layout, International Journal of Production Research, Vol. 18, No. 2, pp. 201-211.

105. C.S. Parker (1976), An experimental investigation of some strategies for component placement, Operations Research Quarterly, Vol. 27, No. 1, pp. 71-81.

106. J. Picard and M. Queyranne (1981), On the one-dimensional space allocation problem, Operations Research, Vol. 29, No. 2, pp. 371-391.

107. C.J. Picone and W.E. Wilhelm (1984), Perturbation scheme to improve Hillier's solution to the facilities layout problem, Management Science, Vol. 30, No. 10, pp. 1238-1249.

108. J.F. Pierce and W.B. Crowston (1971), Tree - search algorithms for quadratic assignment problems, Naval Research Logistics Quarterly, Vol. 18, pp. 1-36.

109. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling (1986), Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, New York, NY.

110. V. Raju (1986), FMS Around the World: A Status Report, Proceedings of the 1986 SME Seminar on Flexible Manufacturing Systems, Chicago, IL, pp. 1-13, (also SME Paper No. MS-86-163).

111. L.P. Ritzman (1972), The efficiency of computer algorithms for plant layout, Management Science, Vol. 18, No. 5, pp. 240-248.

112. L.P. Ritzman, J. Bradford and R. Jacobs (1979), A multiple objective approach to space planning for academeic facilities, Management Science, Vol. 25, No. 9, pp. 895-906.

113. M.J. Rosenblatt (1979), The facilities layout problem: a multigoal approach, International Journal of Production Research, Vol. 17, No. 4, pp. 323-332.

114. S. Sahni and T. Gonzalez (1976), P-complete approximation problem, Journal of Association for Computing Machinery, Vol. 23, No. 3, pp. 555-565.

115. L.E. Schrage (1984), Linear, Integer and Quadratic Programming with LINDO, The Scientific Press, Palo Alto, CA.

116. M. Scriabin and R.C. Vergin (1975), Comparison of computer algorithms and visual based methods for plant layout, Management Science, Vol. 22, No. 2, pp. 172-181.

117. M. Scriabin and R.C. Vergin (1976), Computer and visual methods for plant layout - a rejoinder, Management Science, Vol. 23, No. 1, pp. 105.

118. M. Scriabin and R.C. Vergin (1985), A cluster-analytic approach to facility layout, Management Science, Vol. 31, No. 1, pp. 33-49.

119. J.M. Seehof and W.O. Evans (1967), Automated layout design program, The Journal of Industrial Engineering, Vol. 18, No. 12, pp. 690-695.

120. J.J. Seppanen and J.M. Moore (1970), Facilities planning with graph theory, Management Science, Vol. 17, No. 4, pp. B242-B253.

121. J.J. Seppanen and J.M. Moore (1975), String processing algorithms for plant layout problems, International Journal of Production Research, Vol. 13, No. 3, pp. 239-254.

122. R.H. Shore and J.A. Tompkins (1980), Flexible facilities design, AIIE Transactions, Vol. 12, No. 2, pp. 200-205.

123. D.M. Simmons (1969), One-dimensional space allocation: an ordering algorithm, Operations Research, Vol. 17, pp. 812-826.

124. L. Steinberg (1961), The backboard wiring problem: a placement algorithm, SIAM Review, Vol. 3, No. 1, pp. 37-50.

125. J.A. Tompkins and J.M. Moore (1984), Computer Aided Layout: A User's Guide, AIIE, Norcross, GA.

126. J.A. Tompkins and R. Reed, Jr. (1976), An applied model for the facilities design problem, International Journal of Production Research, Vol. 14, No. 5, pp. 583-595.

127. T.W. Trybus and L.D. Hopkins (1980), Human vs. computer algorithms for the plant layout problem, Management Science, Vol. 26, No. 6, pp. 570-574.

128. T.E. Vollmann and E.S. Buffa (1966), The facility layout problem in perspective, Management Science, Vol. 12, No. 10, pp. B450-B468.

129. T.E. Vollmann, C.E. Nugent and Zartler (1968), A computerized model for office layout, The Journal of Industrial Engineering, Vol. 19, pp. 321-327.

130. R.C. Wilson (1964), A review of facility design models, The Journal of Industrial Engineering, Vol. 15, pp. 115-121.

131. R.J. Wimmert (1958), A mathematical method for equipment location, The Journal of Industrial Engineering, Vol. 9, pp. 498-505.

132. K. Zoller and K. Adendorff (1972), Layout planning by computer simulation, AIIE Transactions, Vol. 4, No. 2, pp. 116-125.

Appendix A

RULE BASE IN KBML

CLASS 1 RULES

Rule R1

```
IF   type of layout is known
 AND type of material handling carrier is not
THEN apply Class 1A rules.
```

Rule R2

```
IF   type of material handling carrier is known
 AND type of layout is not
THEN apply Class 1B rules.
```

Rule R3

```
IF   type of  material handling  carrier and  type of  layout are  both
     unknown
THEN apply Class 1C rules.
```

CLASS 1A RULES

Rule R4

IF type of layout is circular single-row
THEN use robot as the material handling carrier.

Rule R5

IF type of layout is linear single-row
THEN use AGV as the material handling carrier.

Rule R6

IF type of layout is single-row
AND one of the dimensions of the floor plan is greater than twice the reach of the robot
AND the other dimension of the floor plan is greater than the reach of of the robot
THEN use robot as material handling carrier and adopt a circular layout.

Rule R7

IF type of layout is linear double-row
THEN use AGV as the material handling carrier.

Rule R8

IF type of layout is multi-row
THEN use gantry robot as the material handling carrier.

CLASS 1B RULES

Rule R9

IF type of material handling carrier is AGV
AND length of the floor plan is large
AND breadth of the floor plan is small
THEN adopt linear single-row layout.

Rule R10

IF type of material handling carrier is AGV
AND length of the floor plan is large
AND breadth of the floor plan is medium
THEN adopt linear double-row layout.

Rule R11

IF type of material handling carrier is gantry robot
THEN adopt multi-row layout.

Rule R12

IF type of material handling carrier is robot
THEN adopt circular single-row layout.

CLASS 1C RULES

Rule R13

IF type of layout and material handling carrier are both unknown
AND length of the floor plan is large
AND breadth of the floor plan is small
THEN adopt linear single-row layout and use AGV as the material handling carrier.

Rule R14

IF type of layout and material handling carrier are both unknown
AND length of the floor plan is medium
AND breadth of the floor plan is medium
THEN adopt circular single-row layout and use robot as the material handling carrier.

Rule R15

IF type of layout and material handling carrier are both unknown
AND length of the floor plan is large
AND breadth of the floor plan is medium
THEN adopt linear double-row layout and use robot as the material handling carrier.

Rule R16

IF type of layout and material handling carrier are both unknown
AND length of the floor plan is large
AND breadth of the floor plan is large
THEN adopt multi-row layout and use gantry robot as the material handling carrier.

CLASS 2 RULES

Rule R17

IF type of layout is single-row
AND number of machines in the layout problem is less than 8
AND machines are of equal sizes
THEN select model M3 and algorithm A3.

Rule R18

IF type of layout is single-row
AND number of machines in the layout problem is between 8 and 15
AND machines are of equal sizes
THEN select model M2 and algorithm A1.

Rule R19

IF type of layout is single-row
AND number of machines in the layout problem is greater than 15
AND machines are of equal sizes
THEN select model M2 and algorithm A1.

Rule R20

IF type of layout is single-row
AND number of machines in the layout problem is less than 8
AND machines are of unequal sizes
THEN select model M3 and algorithm A3.

Rule R21

IF type of layout is single-row
AND number of machines in the layout problem is between 8 and 15
AND machines are of unequal sizes
THEN select model M2 and algorithm A1.

Rule R22

IF type of layout is single-row
AND number of machines in the layout problem is greater than 15
AND machines are of unequal sizes
THEN select model M2 and algorithm A1.

Rule R23

```
IF   type of layout is multi-row
 AND number of machines in the layout problem is less than 8
 AND machines are of equal sizes
THEN select model M5 and algorithm A4.
```

Rule R24

```
IF   type of layout is multi-row
 AND number of machines in the layout problem is between 8 and 15
 AND machines are of equal sizes
THEN select model M5 and algorithm A4.
```

Rule R25

```
IF   type of layout is multi-row
 AND number of machines in the layout problem is greater than 15
 AND machines are of equal sizes
THEN select model M4 and algorithm A1.
```

Rule R26

```
IF   type of layout is multi-row
 AND number of machines in the layout problem is less than 8
 AND machines are of unequal sizes
THEN select model M1 and algorithm A1.
```

Rule R27

```
IF   type of layout is multi-row
 AND number of machines in the layout problem is between 8 and 15
 AND machines are of unequal sizes
THEN select model M1 and algorithm A1.
```

Rule R28

```
IF   type of layout is multi-row
 AND number of machines in the layout problem is greater than 15
 AND machines are of unequal sizes
THEN select model M1 and algorithm A2.
```

CLASS 3 RULES

Rule R29

IF machines i and j are to be located in adjacent sites
THEN set R(i,j)=A.

Rule R30

IF machines i and j are not to be located in adjacent sites
THEN set R(i,j)=X.

Rule R31

IF the adjacency of machines i and j is to be determined by the algorithm which solves the layout problem
THEN set R(i,j)=0.

Rule R32

IF machine i is to be located at site j
THEN set V(i,j)=1.

Rule R33

IF type of material handling carrier used is AGV and type of layout is linear single-row
AND the assignment of machines i,j with maximum flow value between them are not restricted to any particular site
THEN locate battery charging station near one end of the layout and assign machines i and j to horizontally adjacent sites close to the battery charging station.

Rule R34

IF type of material handling carrier used is AGV and type of layout is linear double-row
AND the assignment of machines i,j with maximum flow value between them are not restricted to any particular site
THEN locate battery charging station near one end of the layout and assign machines i and j to vertically adjacent sites close to the battery charging station.

CLASS 4 RULES

Rule R35

IF algorithm selected is A1
THEN modify penalty parameter β and apply the algorithm.

Rule R36

IF algorithm selected is A1
THEN set the value of each variable in the initial solution to 1 and apply the algorithm.

Rule R37

IF algorithm selected is A1
THEN increase the value of each variable in the initial solution by 1 and apply the algorithm.

Rule R38

IF algorithm selected is A1
THEN modify value of parameter α and apply the algorithm.

Rule R39

IF algorithm selected is A2
THEN modify value of parameter q_0 and apply the algorithm.

Rule R40

IF algorithm selected is A4
THEN modify penalty parameter β and apply the algorithm.

Rule R41

IF algorithm selected is A4
THEN set the value of each variable in the initial solution to 1 and apply the algorithm.

Rule R42

IF algorithm selected is A4
THEN increase the value of each variable in the initial solution by 1 and apply the algorithm.

Rule R43

IF algorithm selected is A4
THEN modify value of parameter α and apply the algorithm.

CLASS 5 RULES

Rule R44

IF R(i,j)=A
AND machines i and j are not in adjacent locations
THEN assign machines i and j to adjacent locations and compute the solution cost.

Rule R45

IF R(i,j)=X
AND machines i and j are in adjacent locations
THEN arbitrarily assign machines i and j to nonadjacent locations and compute the solution cost.

Rule R46

IF V(i,j)=1
AND machine i is not assigned to site j
THEN assign machine i to site j and compute the solution cost.

Rule R47

IF space constraints are violated
THEN modify layout so as to obtain an implementable layout and compute the solution cost.

RULES USED DURING DATA INPUT

Rule RD1

IF number of elements in the flow matrix is less than the square of the number of machines in the layout problem
THEN inform user that more flow matrix elements are to be entered.

Rule RD2

IF number of elements in the flow matrix is greater than the square of the number of machines in the layout problem
THEN inform user that the number of flow matrix elements entered is greater than the required number.

Rule RD3

IF number of elements in the clearance matrix is less than the square of the number of machines in the layout problem
THEN inform user that more clearance matrix elements are to be entered.

Rule RD4

IF number of elements in the clearance matrix is greater than the square of the number of machines in the layout problem
THEN inform user that the number of clearance matrix elements entered is greater than the required number.

Rule RD5

IF number of elements in the relationship indicator matrix is less than the square of the number of machines in the layout problem
THEN inform user that more relationship indicator matrix elements are to be entered.

Rule RD6

IF number of elements in the relationship indicator matrix is greater than the square of the number of machines in the layout problem
THEN inform user that the number of relationship indicator matrix elements entered is greater than the required number.

Rule RD7

IF number of elements entered in the vector representing machine dimension is less than twice the number of machines in the layout problem
THEN inform user that more elements are to be entered in the machine dimension vector.

Rule RD8

IF number of elements entered in the vector representing machine dimension is greater than twice the number of machines in the layout problem
THEN inform user that the number of elements entered in the machine dimension vector is greater than the required number.

Rule RD9

IF the number of machines to be assigned is not equal to the number of machines in the layout problem minus the number of machines whose location are restricted to certain sites
THEN inform the user accordingly.

Rule RD10

IF the relationship indicator matrix consists of elements other than A, O or X
THEN inform user that the relationship indicator matrix must include only A, O or X entries.

Rule RD11

IF the flow matrix consists of non-numerical entries
THEN inform user that the flow matrix must include only numerical entries.

Rule RD12

IF the clearance matrix consists of non-numerical entries
THEN inform user that the clearance matrix must include only numerical entries.