AN IMPROVEMENT-TYPE LAYOUT ALGORITHM FOR MULTIPLE FLOOR FACILITIES

Yavuz A. Bozer

Russell D. Meller

Steven J. Erlebacher

Department of
Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 91-11

April 1991
Revised December 1991

AN IMPROVEMENT-TYPE LAYOUT ALGORITHM FOR MULTIPLE FLOOR FACILITIES†

Yavuz A. Bozer

Russell D. Meller

Steven J. Erlebacher

Department of
Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109, USA

ABSTRACT

In this paper we extend a well-known facility layout algorithm (CRAFT) to facilities with multiple floors, and we introduce the use of spacefilling curves in facility layout. Such curves make it possible to exchange any two departments and to use more powerful exchange routines than two-way or three-way exchanges. Also, for both single and multiple floor facilities, we considerably enhance CRAFT by allowing "flexible" departmental area requirements and by controlling department shapes. Although the algorithm we present can be used for any multi-floor facility layout problem, its primary target is production facilities. A tailored version of the algorithm was successfully tested and used in a large, multi-floor production facility. The algorithm differs significantly from two previous extensions of CRAFT to multi-floor facilities.

1. INTRODUCTION

According to Tompkins and White [29], the "generation of layout alternatives is a critical step in the facilities planning process, since the layout selected will serve to establish the physical relationships between activities." Physical relationships such as distances between activities, material flow patterns, entry and exit points, etc. that result from a layout largely determine its "cost" and "desirability." Given certain interactions that occur among the activities, generally speaking, the facility layout problem is concerned with determining the "most efficient" arrangement of the activities (i.e., departments) subject to constraints imposed by the site plan, the building, the departmental area/service requirements, and the decision-maker.

†This study was partially supported by Dr. Bozer's Presidential Young Investigator Award under NSF Grant DMC-8858562 and a research grant from General Electric Company, DRDA 90-0083.

Obtaining optimal solutions to the facility layout problem is not straightforward primarily for two reasons. First, there are no generally accepted objective functions which capture all the relevant aspects of the facility layout problem. Second, with commonly accepted objective functions, finding the optimal solution is currently near-impossible since it often leads either to a large-scale Quadratic Assignment Problem (QAP) or a large-scale mixed integer programming problem (see Montreuil [24]).

Without exact procedures to solve the facility layout problem, most of the research has been aimed at developing heuristic procedures. Generally speaking, the multi-floor layout problem is more complicated than the single-floor layout problem since the former involves vertical flow and lifts. Also, in single-floor layout problems, as long as the total usable floor space is greater than or equal to the total area required by all the departments, any layout is area feasible. However, in a multi-floor layout problem, the number of departments that can be assigned to any floor is limited by available space on that floor. Consequently, certain layouts may not be area feasible. Of course, there are some additional constraints imposed by a multi-floor building. We will discuss some of these constraints in section 4.2.

Most production facilities constructed today are single-story buildings. However, multi-floor production facilities are still in use in many parts of the United States, and some new production facilities are constructed as multi-floor buildings (see, for example [30], among others). According to [30], the company "saved on land and construction costs by going up three floors." Also, quite often an older facility is renovated to avoid building a new facility. According to [32], "... one major (factor) which recommends renovation is its low cost compared to that of a new building ... (and) ... most old buildings are multi-story." In fact, in [19] it is stated that "a refurbished old industrial plant ... is likely to cost as little as a tenth as much per square foot of building as a new factory today." We are not arguing that renovating a multi-floor plant is always the preferred alternative. Rather, we are stressing that there are many older multi-floor industrial buildings that have been (and are being) renovated. Coupled with recent advances in vertical material handling technology (see, for example, [31], [33], [34], [35], and [36]), many firms are likely to consider renovating or constructing multi-floor buildings.

In this paper we present an *improvement-type* algorithm based on spacefilling curves to solve the multi-floor facility layout problem. More specifically, we extend CRAFT ([1] and [4]) to multi-floor facilities. In the process, we enhance some of the features of CRAFT by increasing the number of department exchanges considered at each iteration and allowing flexible departmental area requirements. We also add new features such as shape control for departments and the consideration of additional constraints which are described later. The remainder of the paper is organized as follows. In section 2 we present the literature review. In sections 3 and 4 we present our layout improvement algorithm for multi-floor facilities. In sections 5 and 6 we demonstrate the new algorithm with an example layout problem in a multi-floor and a single-floor setting, respectively. In section 7 we present certain extensions which would further enhance the algorithm. Lastly, in section 8 we present our conclusions and some of the practical issues we encountered in implementing the proposed algorithm in a real-life multi-floor production facility.

2. LITERATURE REVIEW

A number of computer-based heuristic layout algorithms, such as ALDEP [27], BLOCPLAN [8], COFAD [28], CORELAP [20], CRAFT [1],[4], and PLANET [7], have been developed over the years. There are also a number of algorithms based on graph theory (see [9], [10], and [11], among others), where the area and shape of the departments are initially ignored since each department is first represented as a node. After a planar graph is developed to identify adjacent departments, a heuristic procedure is applied to construct a block layout that satisfies these adjacency relationships (see, for example, Montreuil, Ratliff and Goetschalckx [23]).

As remarked in the previous section, CRAFT ([1] and [4]) — Computerized Relative Allocation of Facilities Technique — is a well-known algorithm which has become one of the most often referenced improvement-type layout algorithms in the literature. Despite some of its shortcomings, CRAFT represents the cornerstone of improvement-type layout algorithms. It has been shown to give reasonably good solutions to a wide range of (single-story) layout problems (see Ritzman [26]). Also, CRAFT can capture in reasonable detail the building shape and related constraints (such as unusable areas) as well as the current layout including fixed departments.

CRAFT begins with an initial layout and performs two-way and/or three-way exchanges of *department centroids* to identify those that potentially reduce the layout cost, which is based on (flow) $\times$ (unit cost) $\times$ (rectilinear distance between department centroids). At each iteration, the exchange that leads to the largest *estimated* reduction in total cost is selected and new centroids are computed after the department locations are actually exchanged. The process of seeking improvements is restarted with the new layout and it continues until there are no 2-way or 3-way exchanges that lead to a reduction in the estimated cost. Of course, like most steepest descent improvement-type algorithms, CRAFT is a "path dependent" heuristic; that is, the initial layout as well as the number and types of exchanges considered at each iteration largely determine the quality of the final solution obtained.

It is well-known that CRAFT can exchange only those departments that are either adjacent or equal in area. If two non-adjacent departments (with unequal areas) are exchanged, all the other departments in the layout must "shift" to accommodate the exchange; otherwise, one of the departments being exchanged will be "split." CRAFT is not capable of shifting the other departments, and splitting a department would not be acceptable in a production facility. Also, in most real-life layout problems, only a few, if any, of the departments will have exactly the same area requirements. Consequently, the above constraint significantly reduces the number of exchanges CRAFT would consider at each iteration. Due to the path dependency of steepest descent algorithms, this is likely to adversely affect the overall quality of the final solutions obtained by CRAFT.

The remainder of the literature review is limited to layout algorithms that can be used in multi-floor facilities. SPACECRAFT is the first multi-floor, improvement-type layout algorithm that appeared in the refereed literature (see Johnson [17]). Except for two modifications, SPACECRAFT is similar to CRAFT. First, the non-linearity imposed by vertical travel is incorporated in the objective function, and second, the facility is "transformed" into a single floor to identify department exchanges. The transformation

is performed (at each iteration) by "appending" each floor, one at a time, to the first floor. (Note that, without such a transformation, SPACECRAFT will not exchange any two departments located on different floors, unless they are equal in area.) Once the layout is transformed, potential exchanges are identified as in CRAFT. To evaluate each exchange, the exchange is performed on the single-floor layout and subsequently the layout is separated back into multiple floors to compute the objective function.

When the layout is separated into multiple floors, a department may be split across two or more floors. Although this may be acceptable for some office buildings or banks, in a production facility a department represents an indivisible entity, by definition. Also, since the *detailed layout* has not yet been determined, it is not clear how one would allocate the incoming flow and the outgoing flow among two or more pieces of a split department. Furthermore, if a department is split across two or more floors, it creates additional *vertical handling within the department*, which is not captured in the objective function.

In evaluating the objective function for the multi-floor layout, SPACECRAFT captures the non-linear nature of vertical travel by assigning an expected waiting time for each lift. If two departments are located on separate floors, the flow between them is assumed to go through the lift which minimizes the total travel time. However, SPACECRAFT does not explicitly consider the utilization of the lifts and the expected waiting time associated with each lift as a function of the workload. Rather, it is assumed that the user enters the expected waiting times *a priori* for each lift.

A similar adaptation of CRAFT to multiple floor facilities, namely, CRAFT-3D, is presented by Çınar [6]. Some of the details of CRAFT-3D are not explained in [6] and we have not located a refereed, archival publication by the author on multi-floor layout. Nevertheless, SPACECRAFT and CRAFT-3D appear to be similar. For details, the reader may refer to Jacobs [16] who provides a brief comparison of the two algorithms.

Other layout algorithms developed for multi-floor facilities are of the *construction type*; i.e., they start with an empty building and construct a layout "from scratch." Four algorithms fall into this category: Automated Layout Design Program (ALDEP) by Seehof and Evans [27], Space Planning Systems (SPS) by Liggett and Mitchell [22], Multi-Story Layout Program (MSLP) by Kaku, Thompson and Baybars [18], and BLOCPLAN by Donaghey and Pire [8]. Typically, these algorithms are based on a two-stage approach: in the first stage, a heuristic procedure is used to assign each department to a particular floor, and in the second stage, a layout is developed for each floor, one floor at a time.

With SPS, each stage of the problem is formulated as a QAP, which is solved by applying an expected value approach proposed by Graves and Whinston [13]. At the end of the first stage, a department may be split between two floors. Also, the authors assume that there is only one lift (or a centrally located single group of lifts). MSLP, on the other hand, is similar to SPS except that: 1. it is assumed that all the departments are equal in area; 2. the assignment of departments to K floors is solved as a K-median problem; and 3. an improvement heuristic (which considers exchanging only those departments located on different floors) is applied after the QAPs have been solved for each floor. As acknowledged by the authors [18], due to the QAP and the K-median problem, it is not straightforward to extend MSLP to the case where departments have unequal area requirements.

ALDEP, which to our knowledge is the first construction-type multi-floor layout al-

gorithm reported, can generate layouts for up to three floors. It is not clear how ALDEP is used in a multi-floor setting since neither the original paper [27] nor subsequent publications address the topic in detail. In fact, in [27] it is not specified how the departments are assigned to floors. Once the assignment is made, however, ALDEP uses the "sweep method" to layout each floor independently of the others. All interactions that occur between departments on different floors are ignored (Çınar [6], p. 25). BLOCPLAN [8] uses an adjacency-based heuristic method to assign departments to floors (see Pire [25]). All the departments on the same floor are considered adjacent while those on different floors are considered non-adjacent. The score does not reflect the number of floors that separate non-adjacent departments. After the departments have been assigned to floors, BLOCPLAN determines the layout of each floor independently (with no provisions for existing or potential lift locations).

In short, although a few improvement or construction-type layout algorithms that can be used in multiple floor facilities exist, certain factors would limit their use in a multi-floor production facility. Except for the throughput capacity of the lifts and the resulting waiting times, the improvement-type algorithm we present here overcomes these limitations and seems to generate reasonably good layouts for single and multiple floor facilities. Before we formally present the algorithm, in the next section we present the use of specefilling curves, flexible areas, and departmental shape constraints in facility layout. These concepts and their integration into one algorithm represent our main contribution.

3. SPACEFILLING CURVES, FLEXIBLE AREAS, AND SHAPE CONTROL

In this section we describe basic concepts which lead to the development of the new improvement algorithm, namely, **MULTIPLE** (*MULTI*-floor *P*lant *L*ayout *E*valuation). After discussing layout representation with spacefilling curves, we illustrate how spacefilling curves and flexible department areas allow MULTIPLE to increase the number of exchanges within a floor and the number of exchanges across floors (without splitting departments). We also show how MULTIPLE controls department shapes.

3.1. Spacefilling Curves and the Facility Layout Problem: Spacefilling curves (or Peano curves) were first viewed as mathematical oddities since a spacefilling curve is a continuous function with no unique derivative at any point. More recently, however, spacefilling curves were proposed as a Traveling Salesman Problem (TSP) heuristic by Bartholdi and Platzman [2], who also used such curves to locate items in a storage rack (see [3]).

In most computer-based layout algorithms (including MULTIPLE), the layout is represented as a matrix. Each element of the matrix corresponds to a grid square (or grid) of specified area, and the space required by each department is expressed as an integer number of grids. To construct and manipulate the layout, we propose to use a spacefilling curve which simply visits all the grids on a floor (except for the grids that correspond to fixed departments and/or unusable areas). To ensure that a department is not split (within a floor or across two floors), all the grids assigned to a department must be contiguous, i.e., each grid must be adjacent to another grid that has been assigned to the same department. By its very construction, a spacefilling curve can guarantee that no departments

will be split because a separate curve is used for each floor and, within each floor, the curve visits the "neighbors" of a grid before visiting other grids. As we demonstrate later, using spacefilling curves offer significant advantages.

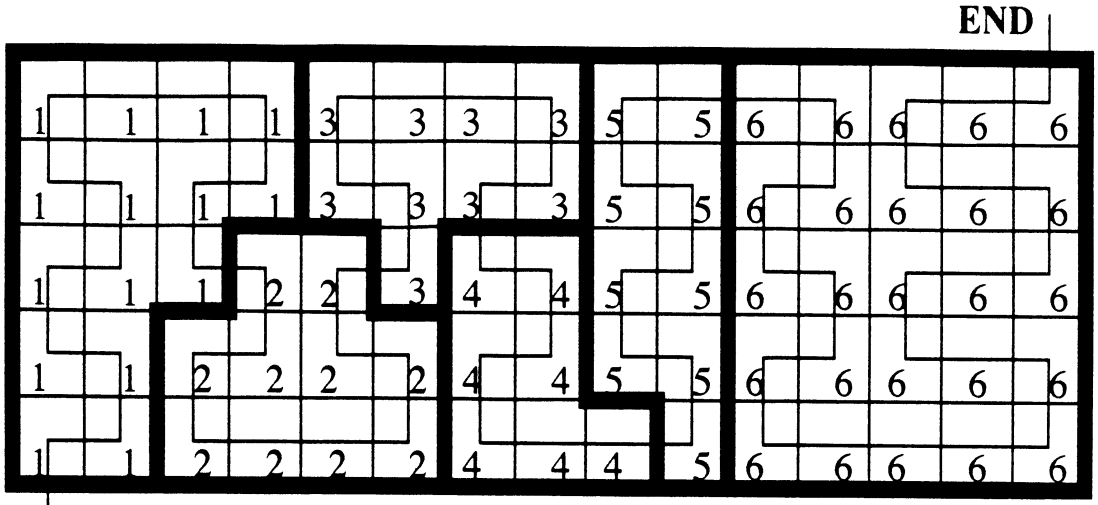
A six-department example with no fixed departments or unusable areas is shown in Figure 1a. Note that, with a given spacefilling curve and department area values, a layout is uniquely determined only by a sequence of department numbers. Suppose departments 1 through 6 require 15, 10, 9, 7, 9 and 25 grids, respectively. Then, a possible layout is shown in Figure 1a where the sequence of department numbers, say, the *layout sequence*, is given by 1–2–3–4–5–6. The layout was constructed by allocating the first 15 grids visited by the spacefilling curve to department 1, the next 10 grids to department 2, and so on.

3.2. Exchanging Departments Within a Floor: Unlike CRAFT, spacefilling curves allow MULTIPLE to exchange any two departments whether or not they are adjacent and/or equal in area. For example, in Figure 1a, to exchange departments 2 and 5 we simply exchange their positions in the layout sequence and reconstruct the layout to obtain the new layout shown in Figure 1b. Note that, since departments 1 and 6 are not between departments 2 and 5 in the original layout sequence, their locations are not affected by the exchange. In contrast, the locations of departments 3 and 4 have shifted since they fall between departments 2 and 5.

3.3. Generating Spacefilling Curves: In a rectangular building with no fixed departments or unusable areas, the spacefilling curve can be obtained by using the recursive procedure presented in the Appendix. This procedure is based on the Hilbert curve (shown in Figure A3 and described in [15]). If the building shape is quite irregular and many obstacles (such as load-bearing walls) and/or fixed departments are present, one may consider generating a spacefilling curve by hand. Although such a curve is not necessarily a spacefilling curve in a mathematical sense, *functionally* it may be viewed as one. Using MULTIPLE in a large, four-floor production facility, for example, we opted for hand-generated curves. In doing so, we fully captured the current layout and all the obstacles as well as the exact building shape (although it was not rectangular).

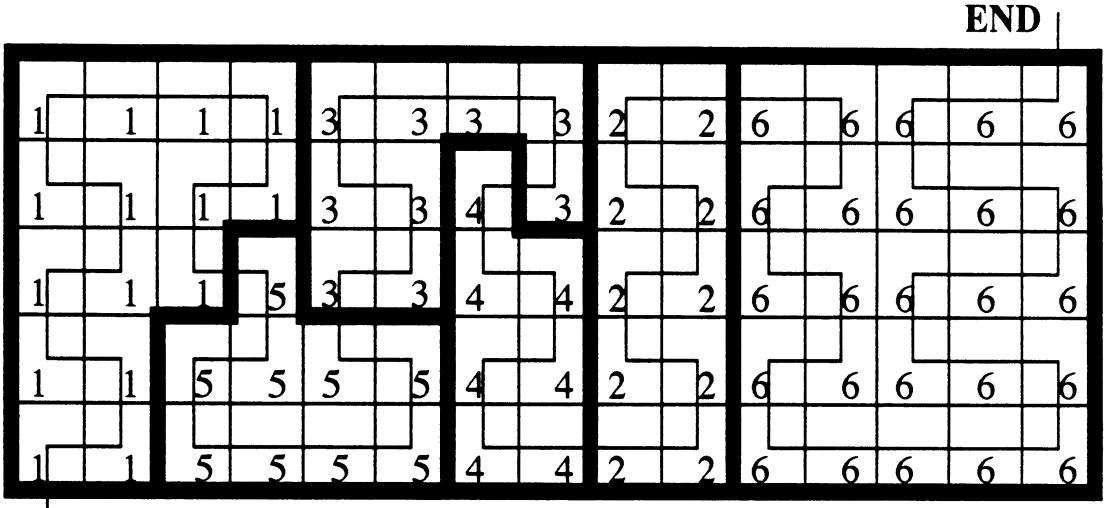
In fact, any curve (hand-generated or otherwise) which visits all the grids by taking horizontal, vertical, or diagonal steps (from one grid to an adjoining grid) can be used with MULTIPLE. Such alternate curves can be generated with relative ease on a personal computer using the "cursor keys" or a "mouse." We have also generated some curves by solving a TSP, where one can minimize the number of diagonal steps in favor of horizontal and vertical steps. Regardless of the method employed, we must stress that spacefilling or alternate curves are generated once for each floor only at the start.

A spacefilling curve may be tailored to a particular building. If some departments are fixed, the curve simply bypasses all the grids assigned to those departments; that is, when we generate the curve we do not route it through fixed departments. Otherwise, even if we disallow all pairwise exchanges that involve a fixed department, it may still shift when other departments on that floor are exchanged. In reference to Figure 1a shown earlier, if department 2 is fixed, we simply reroute the spacefilling curve so that none of the grids assigned to department 2 are visited. On the other hand, if department 5 is fixed, it would divide the floor into two disjoint sections. Consequently, we would generate two spacefilling



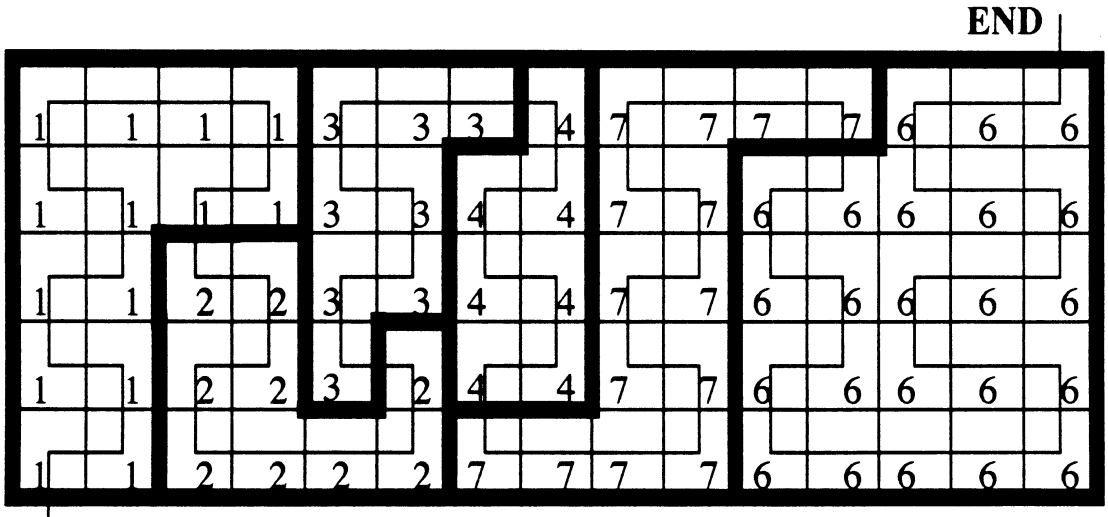
Layout Sequence = 1, 2, 3, 4, 5, 6

(a) Initial Layout.



Layout Sequence = 1, 5, 3, 4, 2, 6

(b) Department 2 and Department 5 Exchanged.



Layout Sequence = 1, 2, 3, 4, 7, 6

(c) Department 5 and Department 7 Exchanged.

Figure 1. Using Spacefilling Curves to Construct Layouts.

curves, one for each section of the floor. Of course, with such a fixed department, some within-floor exchanges may no longer be area feasible.

The initial layout may also affect curve generation. Curves that fully conform to the initial layout can be generated by solving a *clustered* TSP [5], where all the grids assigned to a particular department in the initial layout will be visited before the salesman is permitted to visit any other grid. Lastly, it is instructive to note that the *sweep method* used by ALDEP [27] can also be viewed as a "spacefilling curve." Although the sweep method can determine a layout quickly and represent it as a sequence of department numbers, it may split departments around fixed departments or obstacles. The sweep method is too rigid; it does not possess the flexibility of spacefilling curves.

3.4. Flexible Departmental Areas and Multiple Floors: Perhaps the most challenging task in extending CRAFT to multi-floor facilities is to exchange unequal area departments across two floors without splitting the departments. According to Tompkins and White [29], primarily due to uncertainty, "... perhaps the most difficult determination in facilities planning is the amount of space required in the facility." A similar observation is made in architecture by Lew and Brown [21] who stated that "in any architectural design process, area requirements have a range of acceptable values." Hence, instead of supplying a single estimate for the departmental area requirements (as in other layout algorithms), we propose to use a range of acceptable values specified for each department. That is, we let A_i^L and A_i^U designate the minimum acceptable and the maximum allowable floor space to be allocated to department i, respectively. The area of a department in the initial/current layout, say, A_i, is assumed to fall within the range specified by A_i^L and A_i^U.

Consider exchanging two departments, say, i and j, located on different floors. Without loss of generality, suppose $A_i > A_j$. If $A_i^L \leq A_j$ and $A_j^U \geq A_i$, then the exchange is area feasible and the two departments represent an *even exchange*. That is, A_i simply assumes the value of A_j (and vice versa) and the two departments can be exchanged without having to relayout either floor. If the above two conditions are not met, one might still be able to exchange departments i and j after "compressing" all the departments currently located on the same floor with department j by setting their areas equal to their lower limits. (We need not compress the departments currently located on the same floor with department i since $A_i > A_j$.) If department i fits in the space that becomes available, then the exchange is area feasible. Otherwise, departments i and j cannot be exchanged in the current layout. Of course, an exchange will never be area infeasible due to upper limits on department areas. Following an exchange, any additional floor space will be left unused at the end of the spacefilling curve. The above procedure is formally presented in Figure 2, where $k(j)$ denotes the floor number of department j, $S_{k(j)}$ denotes the set of departments located on floor $k(j)$, and $T_{k(j)}$ denotes the total area available on floor $k(j)$.

Consider the previous example shown in Figure 1a. Suppose the lower, current, and upper limits on the area for each of the six departments have been specified as follows:

$$12 \leq A_1\,(=15) \leq 17 \quad 8 \leq A_2\,(=10) \leq 12 \quad 7 \leq A_3\,(=9) \leq 12,$$
$$6 \leq A_4\,(=7) \leq 10 \quad 6 \leq A_5\,(=9) \leq 12 \quad 20 \leq A_6\,(=25) \leq 30.$$

Consider next exchanging departments 5 and 7, where the latter is currently located on a different floor and $12 \leq A_7\,(=14) \leq 17$. Since $A_5 < A_7$ we need not be concerned

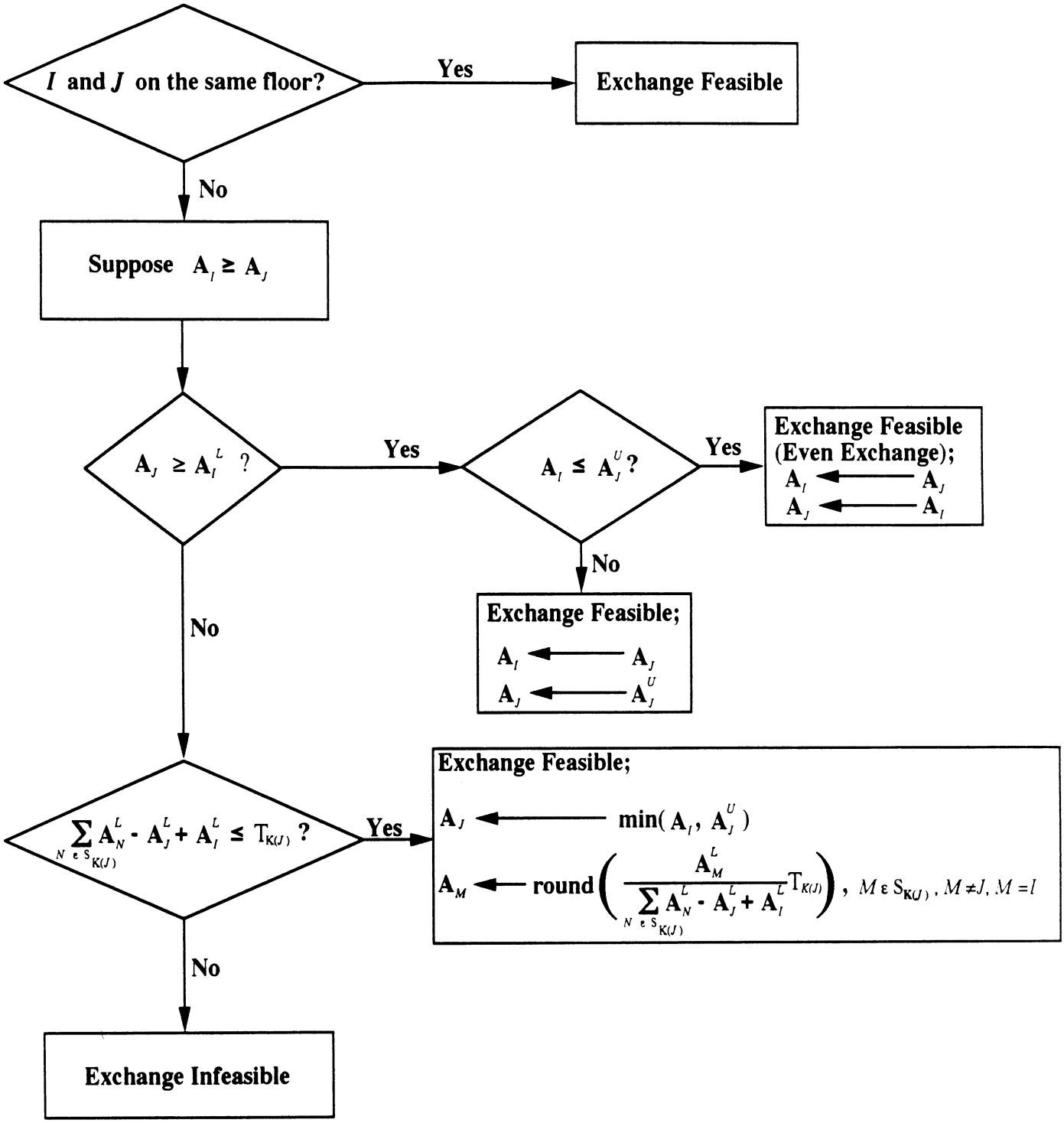


Figure 2. Procedure to Determine Area Feasibility and Department Areas.

with space availability on the floor where department 7 is currently located. However, since $A_7^L > A_5$, an even exchange is not possible and we must try compressing other departments. If each department is reduced to its lower limit, the total area required by departments 1, 2, 3, 4, 7, and 6 would be equal to 65 grids, which implies that exchanging departments 5 and 7 is area feasible. Expanding each department to fill the floor is accomplished by multiplying the lower area limit of each department with $75/65 = 1.1538$ and rounding it off to the nearest integer (while observing the upper limit on each department). The resulting layout is shown in Figure 1c. Note that, without spacefilling curves, compression would not be possible. Many of the departments which are compressed must shift to accommodate the entering department which is larger than the exiting department.

3.5. Controlling Department Shapes: As a department becomes more irregular in shape, it becomes impractical or more difficult to develop an efficient arrangement of work-stations within that department. That is, constructing the *detailed layout* becomes more difficult. With MULTIPLE, after exchanging two departments, the new shape assumed by some departments may not be acceptable. (The same problem has also been reported for CRAFT; see [14] and [21], among others. Presently, there is no mechanism to control department shapes in CRAFT.) For example, in Figure 1c, although one can easily smooth the border between departments 2 and 3 when the grid structure is replaced by the actual layout, the resulting shape of department 2 may not be acceptable.

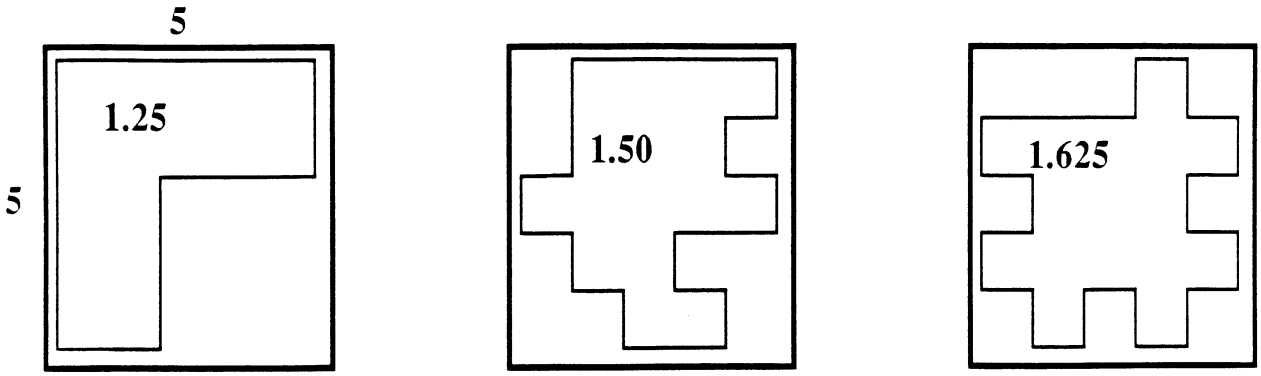
While the human eye is very adept at making judgments concerning shape, a computer program requires a formal measure, which must also be easy to compute since shape measurements must be performed after each possible exchange. Freeman [12] notes that, for a fixed area, the perimeter of an object increases as it becomes more irregular in shape. Letting P_i denote the perimeter of department i, we propose to use P_i/A_i as a measure of shape irregularity. (With the matrix representation of the layout, it is possible to compute P_i without much effort.) Since a circle is not an acceptable department shape, the perimeter of an object would be minimized if the object is square shaped. Hence, for a given area of A_i, we can compute the minimum perimeter for department i, say, P_i^*, simply from $P_i^* = 4\sqrt{A_i}$.

To compare department shapes, it would be more desirable to use a unitless measure of irregularity that improves as the department shape approaches an ideal shape. Assuming that a square represents the least irregular, ideal shape for a department, the unitless (or normalized) measure of irregularity for department i, say, Ω_i, is given by:

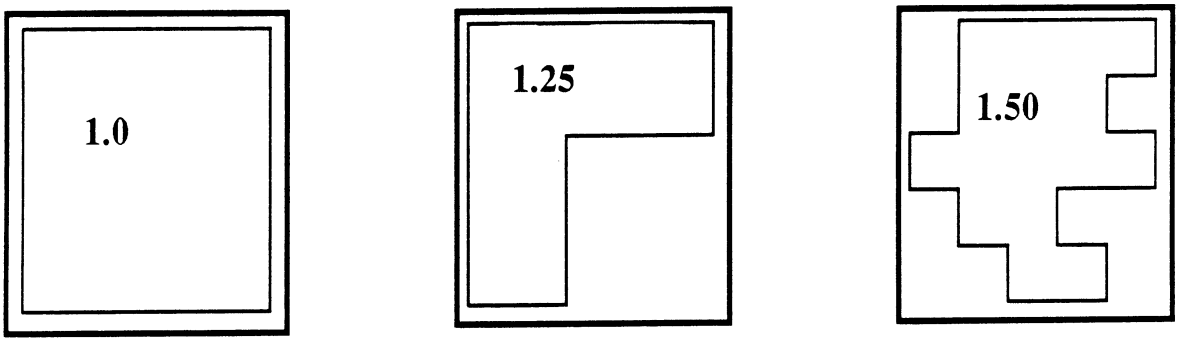
$$\Omega_i = \frac{P_i/A_i}{P_i^*/A_i} = \frac{P_i}{P_i^*} = \frac{P_i}{4\sqrt{A_i}} = \frac{1}{4} P_i A_i^{-0.5}. \tag{1}$$

With the above normalized shape measure, as the shape of a department becomes more irregular, its Ω_i value increases. Hence, the analyst must specify an upper limit on Ω_i that he/she is willing to tolerate for each department. Our experience suggests that reasonable shapes are generally obtained if the upper limit on Ω_i is kept under 1.50. If the ideal shape for a department is not a square, one needs to redefine the minimum perimeter and impose a lower limit on Ω_i if square shaped departments are to be avoided.

Two alternative shape measures were proposed by Liggett and Mitchell [22]. The first one is obtained by dividing the area of the smallest rectangle that fully encloses a



(a) Measure 1: $\frac{\text{Encl. Rect. Area}}{\text{Department Area}} = 25/16 = 1.5625$



(b) Measure 2: $\frac{\text{Encl. Rect. Length}}{\text{Encl. Rect. Width}} = 1.0$

Figure 3. Comparison with the Liggett and Mitchell Shape Factors.

department by the area of the department itself. The second measure is obtained by dividing the length of the smallest enclosing rectangle by its width. As shown in Figure 3, neither measure is as effective as Ω_i. Consider first Figure 3a, where the department area is assumed to be fixed at 16 grid squares and the ideal shape is assumed to be a square. Using the first shape measure proposed in [22], we obtain $25/16 = 1.5625$ for *all three shapes* shown in Figure 3a. In contrast, the Ω_i value (which is shown within each department) starts at 1.250 for the L–shaped department and increases to 1.625 to correctly capture the deteriorating shape of the department. In Figure 3b, under the same assumptions stated above, a similar observation is made. In this case, the second shape measure proposed in [22] is equal to 1.00 for *all three shapes*, while Ω_i starts from 1.00 for a square and increases to 1.50 as the department shape becomes more irregular.

Of course, the measure we propose will not guarantee that a department will attain its ideal shape in the final layout since shapes other than the ideal one may have perimeter values that are close to P_i^*. In that sense, Ω_i does not allow one to directly and explicitly specify the shape of a department. (That would be quite difficult to implement within an improvement-type algorithm). However, in general, the proposed measure will enable the program to reject some exchanges which result in irregular department shapes that are likely to have perimeter values far greater than P_i^*. Also, the proposed measure is a general one; that is, it can be used to control department shapes in any layout algorithm as long as department perimeters can be computed with relative ease.

The spacefilling curve may also affect the final department shapes. However, the shape of a department is jointly determined by the area of the department itself and the total area of the departments that precede it in the layout sequence. Therefore, until all the department areas are computed and the layout is constructed, it is not possible to predict department shapes from the spacefilling curve alone. Nevertheless, as far as department shapes are concerned, one advantage of using spacefilling curves is that the department shapes do not necessarily deteriorate from one iteration to another. (As mentioned earlier, the department shapes in CRAFT have a tendency to deteriorate fairly rapidly with the number of iterations.)

4. THE LAYOUT IMPROVEMENT ALGORITHM

The new layout algorithm we developed, MULTIPLE, is presented in this section. The algorithm is based on integrating the concepts described in section 3 within the framework of a steepest descent search heuristic. After presenting additional assumptions and definitions relevant to this section, we discuss additional constraints that apply to multi-floor facilities. We then present MULTIPLE in flow chart form.

4.1. Assumptions and Definitions: The overall approach we use is similar to CRAFT. That is, the objective function is a distance-based measure (rather than an adjacency-based measure) and we attempt to improve the layout through departmental exchanges. Let f_{ij} denote the flow from department i to department j, and let c_{ij}^H (c_{ij}^V) denote the horizontal (vertical) cost of moving one unit load from department i to department j by one distance unit. (It is assumed that both the f_{ij}'s and the c_{ij}'s are supplied by the analyst.) The

objective is to

$$\min \sum_{i=1}^{N}\sum_{j=1}^{N}(c_{ij}^{H}d_{ij}^{H} + c_{ij}^{V}d_{ij}^{V})f_{ij}, \tag{2}$$

where N denotes the number of departments and d_{ij}^{H} (d_{ij}^{V}) denotes the horizontal (vertical) distance from department i to department j. Note that the d_{ij}'s are the decision variables and their values are obtained from the layout.

Like CRAFT, all horizontal distances are assumed to be measured rectilinearly between department centroids. However, when two departments are located on different floors, the flow is forced to go through one of several lifts. (Here we define a lift as a generic vertical handling device.) The location of each existing or potential lift is assumed to be specified in the initial layout. Letting ℓ designate a lift, the flow is assumed to go through the lift which minimizes total horizontal travel; that is,

$$d_{ij}^{H} = \min_{\ell}(d_{i\ell}^{H} + d_{\ell j}^{H}), \tag{3}$$

where $d_{i\ell}^{H}$ designates the horizontal distance from the centroid of department i to lift ℓ. We do not consider the throughput capacity of each lift. Rather, we allow each f_{ij} that involves vertical flow to use the lift that minimizes d_{ij}^{H}. Also, we implicitly assume that c_{ij}^{V} does not vary from one lift to another; however, this assumption can be easily relaxed. In addition to the f_{ij}, c_{ij}^{H}, and c_{ij}^{V} values, for each floor k, the analyst is assumed to supply the total usable floor space, T_k (in grids), an initial layout (in matrix form), and a spacefilling curve. The analyst must also supply the range of acceptable area values (in grids) for each department; that is, A_i^{L}, A_i^{U}, and A_i for $i = 1, 2, \ldots, N$.

4.2. Additional Constraints: In addition to the area and shape constraints discussed in section 3, there may be certain constraints associated with the location of a (non-fixed) department. One such constraint is concerned with the particular floor a department is assigned to. It can take one of the following two forms: department i must be assigned to floor k, or department i cannot be assigned to floor k. A good example for the first case is the receiving/shipping department which (almost always) must be assigned to the ground floor. (Of course, if there are existing docks, one can also designate the receiving/shipping department as a fixed department.) The second case may arise if the floor loading capacity and/or the floor-to-ceiling distance varies from one floor to another. In this case, a particular department may not be located on some floors due to heavy equipment or the vertical clearance required by some workstations. Both cases can be easily treated by disallowing an exchange if one or both of the departments involved violates either constraint as a result of that exchange. Note that exchanges that occur within a floor will not violate the above constraint.

Another type of constraint that one may encounter is concerned with the particular region (or section of a floor) a non-fixed department may not be assigned to. For example, it may be acceptable to locate a particular department anywhere on the second floor, except for, say, the west wing. If such *regional constraints* are in effect for a particular department, the analyst simply indicates which grid squares these regions cover. After an

exchange, if a department occupies a grid square that falls into its forbidden region, the exchange is disallowed.

In a multi-floor setting, being able to accommodate the above constraints adds considerable realism to the proposed algorithm. In the real-life, multi-floor production facility where we applied MULTIPLE, we encountered all of the above constraints. (In fact, the regional constraint was proposed by the plant manager.) Provided that it starts at a feasible point, MULTIPLE is able to observe the above constraints as it attempts to improve the layout.

4.3. The Algorithm: MULTIPLE is presented in Figure 4, where r is the iteration counter and DEP designates the number of non-fixed departments. Starting with a given initial or current layout, at each iteration MULTIPLE considers all two-way, area feasible exchanges between non-fixed departments. The impact on layout cost is measured according to equations (2) and (3). The current-best layout cost for each iteration is maintained under MIN. When all exchanges have been evaluated, the algorithm selects the feasible exchange which yields the maximum reduction in the layout cost and the exchange procedure is restarted with the new layout. The search procedure terminates when no feasible cost improving exchanges are identified in the current layout.

Note that, at any given iteration, MULTIPLE considers all exchanges within each floor as well as all area feasible exchanges across two or more floors. Hence, the type of exchange selected may vary from one iteration to the next. In general, we believe this is a better strategy than a two-stage approach where one would first consider all exchanges across two or more floors, followed by all possible exchanges within each floor.

5. NUMERIC EXAMPLE FOR MULTIPLE FLOORS

In this section we present a simple example to demonstrate MULTIPLE. The example problem is based on a three-floor facility composed of fifteen departments and six existing (or potential) lift locations. The current area requirements and the minimum area requirements for each department are presented in Table 1. We assume there are no upper bounds imposed on the floor space occupied by any department. Department 15 represents the receiving/shipping department and is fixed in its current location in the initial layout, which is presented in Figure 5.

The spacefilling curve used for the second and third floors is shown in Figure 1a. (Note that the layout shown in Figure 1a corresponds to the third floor in Figure 5.) The spacefilling curve for the first floor, on the other hand, is identical to the other two except that it does not cover the last 25 grids of the layout because department 15 is fixed. The flow matrix for the problem is shown in Table 2. For simplicity, the cost to travel a distance unit for any product is assumed to be \$1.00 if the distance is horizontal and \$5.00 if it is vertical. This is true for all pairs of departments except the receiving/shipping department, which assumes a cost of \$0.25 for horizontal travel and \$1.25 for vertical travel due to, say, more efficient material handling equipment and larger unit loads. The distance between adjacent floors is assumed to be 10 distance units.

The cost of the initial layout is \$281,702.35. Using MULTIPLE on a 25 MHz DOS-based 386 personal computer (PC) with a 387 coprocessor, we obtained the final layout

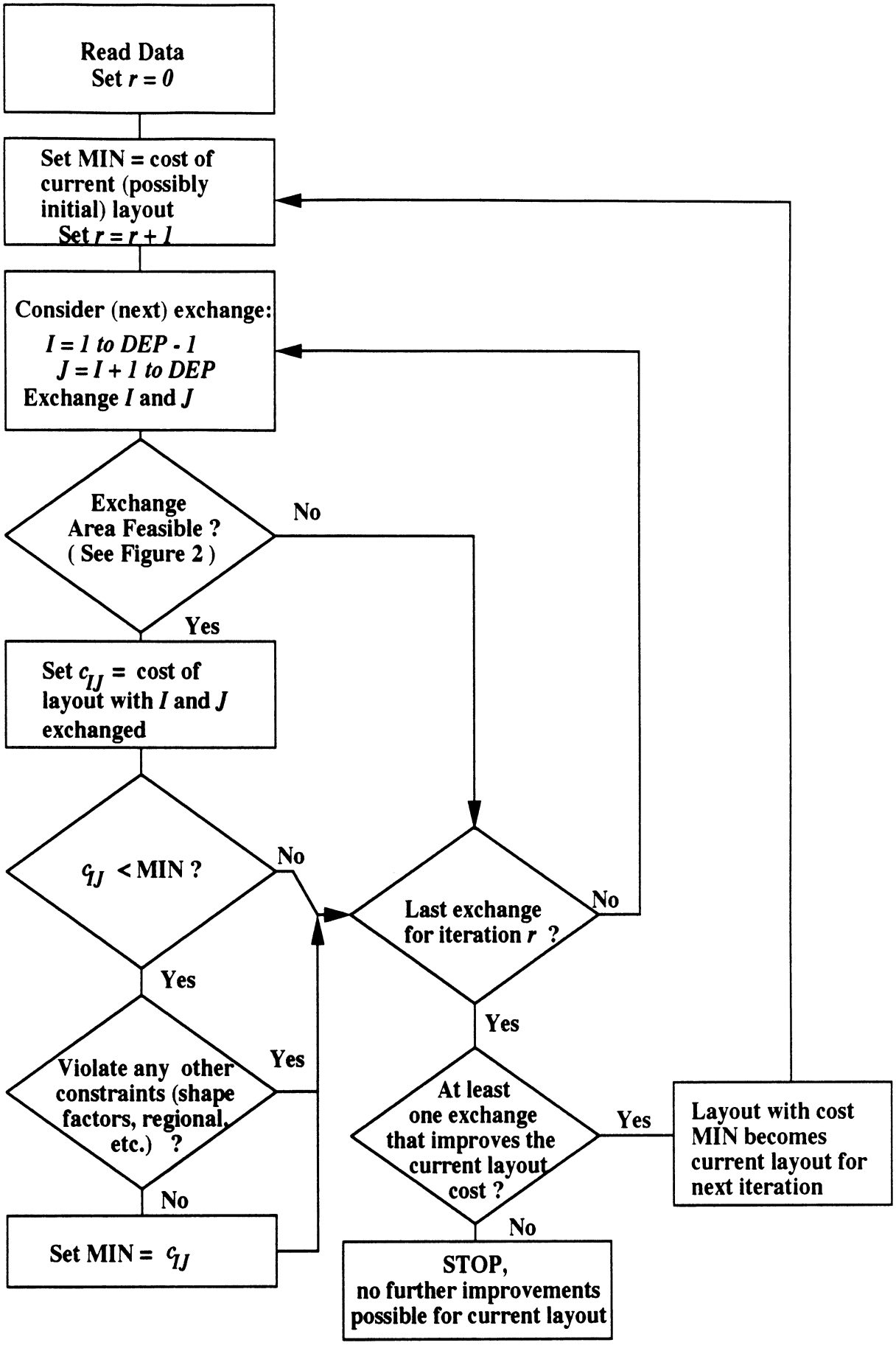


Figure 4. Flow Chart of MULTIPLE.

Table 1. Area Requirements for the Multi-Floor Example Problem.

Department	Current Area	Minimum Area
1	15	12
2	10	7
3	9	6
4	7	5
5	9	7
6	25	22
7	25	22
8	15	13
9	10	7
10	25	22
11	10	9
12	15	13
13	6	4
14	19	17
15	25	25

Table 2. Flow Matrix for the Multi-Floor Example Problem.

From/ To	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	240
2	240	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0
6	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
9	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0
11	0	0	0	0	0	0	480	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600
13	0	0	0	0	0	0	480	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0
15	0	10	25	0	25	40	0	0	25	0	40	0	20	0	0

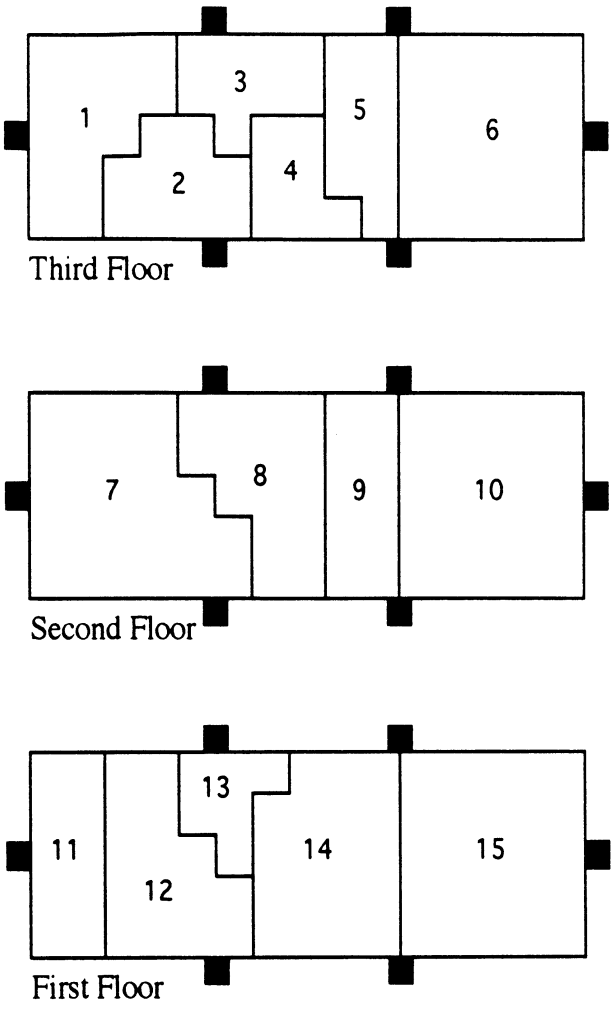


Figure 5. Initial Layout for the Multi-Floor Example Problem.

shown in Figure 6. The final cost is $125,822.50, which represents a decrease of $155,879.85 (or a 55.33% reduction in layout cost). The final layout was obtained in seven iterations, which took 37.9 seconds.

The effect of department compression was explored by using the above example problem with the compression feature removed from the algorithm. The final layout (obtained in eight iterations) is shown in Figure 7. The final cost is $126,733.92, which represents a decrease of $154,968.43. Hence, department compression accounts for an improved savings of only $911.42. The runtime without compression was 20.16 seconds, which represents a decrease of 47% over the runtime with compression enabled. Although the number of iterations increased without compression, the runtime decreased. This is primarily due to a smaller number of feasible exchanges at each iteration with compression disabled. With the example problem, compression does not greatly improve the final layout cost. However, with larger and more realistic problems, compression may be essential.

Suppose we now impose a shape constraint on department 10 and assume that its ideal shape is a square. Inspecting the final layout shown in Figure 7, it is clear that department 10 violates this constraint. To obtain a final layout that meets this constraint, we set the upper limit on Ω_{10} equal to 1.0 and rejected any layout that violated this constraint. (In general, setting the upper limit on any Ω_i value equal to 1.0 is too restrictive since near-square shapes will not be acceptable. This may severely limit the number of exchanges considered by the algorithm.) The final layout with the above shape constraint imposed is presented in Figure 8. Clearly, for the example problem the shape constraint had little impact on the cost savings. The percentage reduction in cost fell from 55.01% to 54.32%, which is a net difference of only $1,935.37. The final layout was still obtained in eight iterations. The runtime was 20.09 seconds (without compression).

We now consider the impact of imposing additional constraints. Suppose department 9 may not be located on the third floor due to the weight of the machines involved. Further suppose department 14 may not be located on the left-hand side of the first floor because of the unusually low floor-to-ceiling distance in that section of the building. Both of these constraints are violated in Figure 7, and they are both representative of location restrictions that may arise in practice. The final layout obtained from MULTIPLE is shown in Figure 9. The algorithm reduces the cost of the layout by $99,293.27 (or 35.25% of the initial layout cost) in 11 iterations. The runtime was 24.96 seconds (without compression). No shape constraints were imposed on any department.

For comparison purposes, we ran SPACECRAFT with the same data and obtained the layout shown in Figure 10. (Setting the expected waiting time at each lift equal to zero forces SPACECRAFT to use the closest lift.) The layout shown in Figure 10 was obtained in 12 iterations. (We will not report the runtime since SPACECRAFT currently runs on a mainframe.) In the final layout, SPACECRAFT split departments 8 and 10, and the cost is $129,168.00. (The cost of the final layout obtained by MULTIPLE is $125,822.50 and $126,733.92 with and without compression, respectively.) Note that, when a department is split, the objective function of SPACECRAFT underestimates the cost of the layout since vertical flow *within* the split department is not considered in computing the layout cost.

We would have preferred to make a more equitable comparison between MULTIPLE and SPACECRAFT. However, if one attempts to stop SPACECRAFT from splitting a

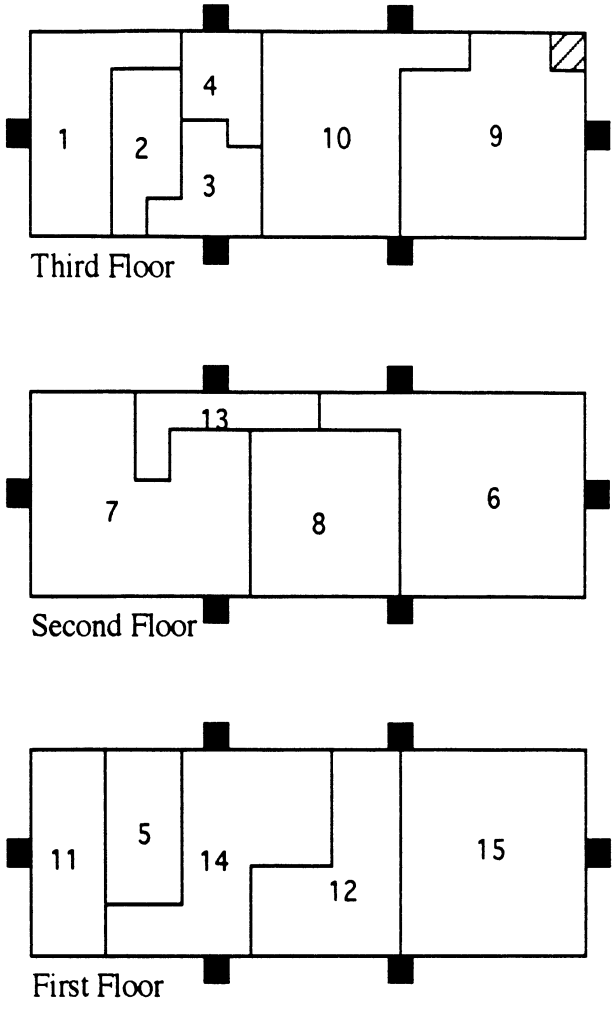


Figure 6. Final Layout Obtained by MULTIPLE with Compression.

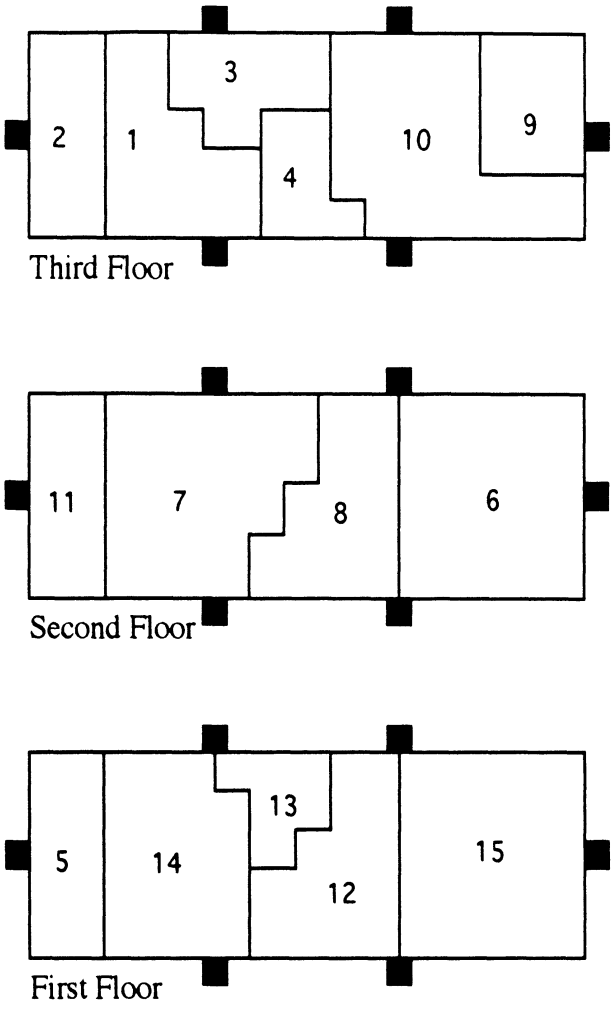


Figure 7. Final Layout Obtained by MULTIPLE without Compression.

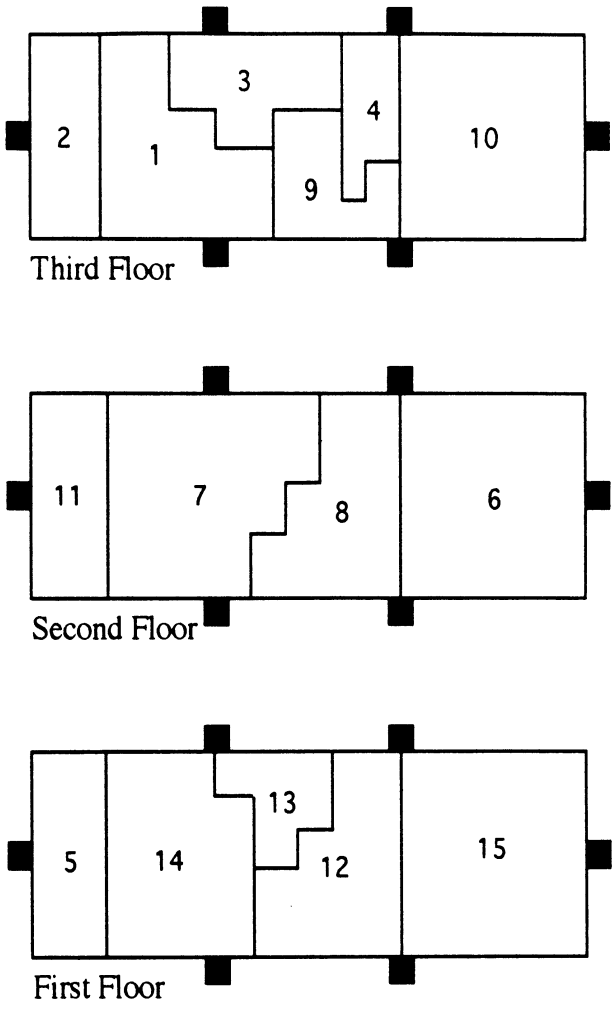


Figure 8. Final Layout Obtained by MULTIPLE with Shape Constraint.

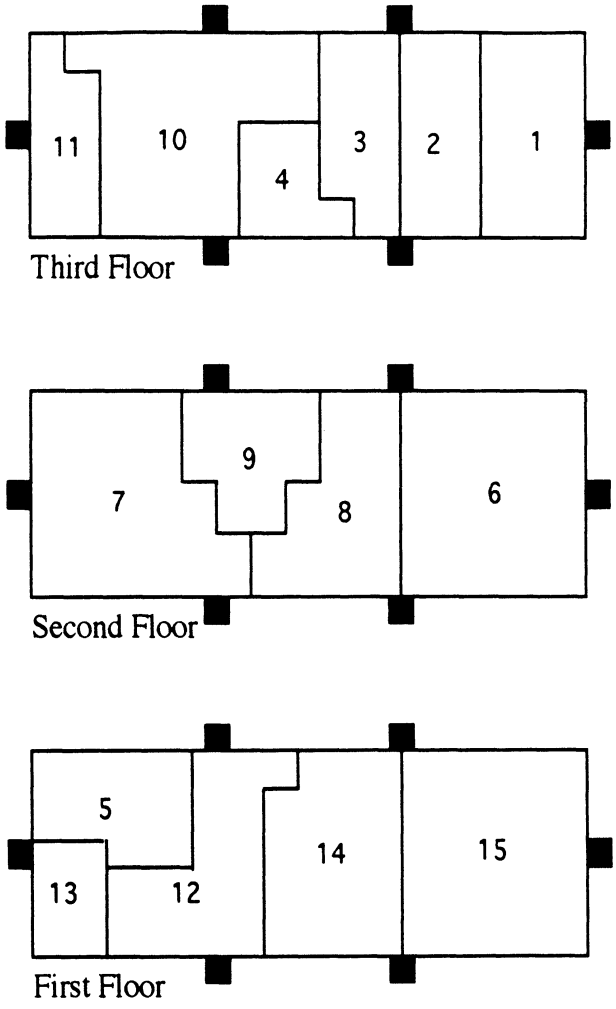


Figure 9. Final Layout Obtained by MULTIPLE with Regional Constraints.

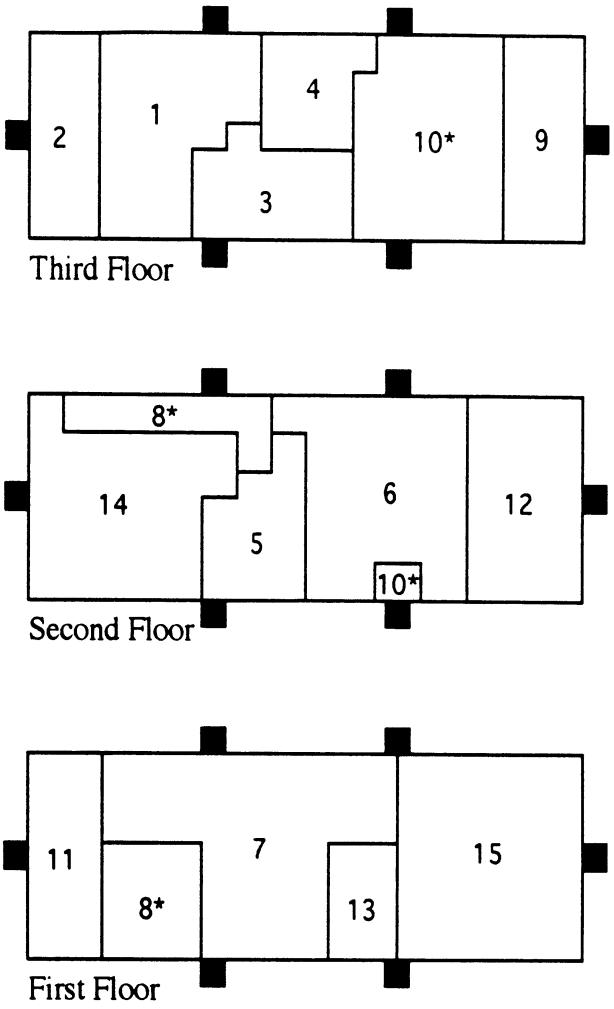


Figure 10. Final Layout Obtained by SPACECRAFT.

department across two or more floors, it essentially transforms the algorithm back to CRAFT. That is, no two departments will be exchanged across floors unless they are equal in area. Also, SPACECRAFT cannot handle flexible areas. Therefore, we had to run it with the A_i values fixed. (These values were obtained from the initial layout.)

6. NUMERIC EXAMPLE FOR SINGLE FLOOR

Although MULTIPLE's principal strength lies in multi-floor facility layout problems, in this section we use it on a single-floor example problem and compare the final layout with the one obtained by CRAFT. Recall that MULTIPLE can exchange any two departments whether or not they are adjacent and/or equal in area. Consequently, the set of exchanges considered by CRAFT at each iteration is a subset of those considered by MULTIPLE. However, since both algorithms are path dependent heuristics, one cannot conclude that MULTIPLE will always outperform CRAFT. Rather, provided that both heuristics are started with the same initial layout, in general, MULTIPLE is *likely to obtain a lower-cost layout simply because it considers more exchanges at each iteration.* In other words, MULTIPLE represents a substantive improvement of CRAFT because it relaxes a significant constraint. Due to the heuristic nature of the search process, this improvement is likely but not guaranteed to produce a lower-cost solution, and its impact on the final layout cost may show significant variation from one problem to another.

The following 20-department problem was taken from [1], where CRAFT was originally proposed. The spacefilling curve used by MULTIPLE is shown in Figure 11. (Note that the spacefilling curve fully conforms to the initial layout to ensure that both algorithms start with the same layout.) The cost of the initial layout is \$101,643.37. In seven iterations CRAFT reduced the layout cost to \$78,620.90 whereas MULTIPLE reduced the layout cost to \$68,578.83 in 13 iterations. Hence, in this particular case, MULTIPLE increases the savings in material handling costs by 44%. The final layout obtained from MULTIPLE is shown in Figure 12. The shapes of departments V and M in the final layout are probably not acceptable. This is due, in part, to the conforming spacefilling curve we used. One may readily generate alternative spacefilling curves and/or impose shape constraints to obtain better department shapes in the final layout.

We also present the runtimes to give both a relative and an absolute measurement of the solution times required by both algorithms. Both programs were run on a 25 MHz DOS-based 386 PC with a 80387 coprocessor. The runtime for CRAFT was 2.1 seconds while MULTIPLE required 2.5 minutes. There are four reasons behind the above significant difference in runtimes. First, at each iteration, MULTIPLE considers more exchanges than CRAFT. Second, unlike CRAFT, MULTIPLE evaluates the *actual cost* of each exchange. (CRAFT first obtains a *cost estimate* for each exchange by exchanging only the department centroids.) Third, although we did not impose any shape constraints in MULTIPLE (to keep the comparison equitable), the algorithm automatically computes the shape measure for each department in evaluating each exchange. Lastly, our copy of CRAFT was written in FORTRAN while MULTIPLE was implemented with PASCAL. Benchmark problems we ran (on the above PC) indicate that our FORTRAN compiler generates object codes that run about 2.25 times faster than those generated by our PASCAL compiler. (Rewriting CRAFT in PASCAL — or MULTIPLE in FORTRAN – is not a feasible avenue to pursue.)

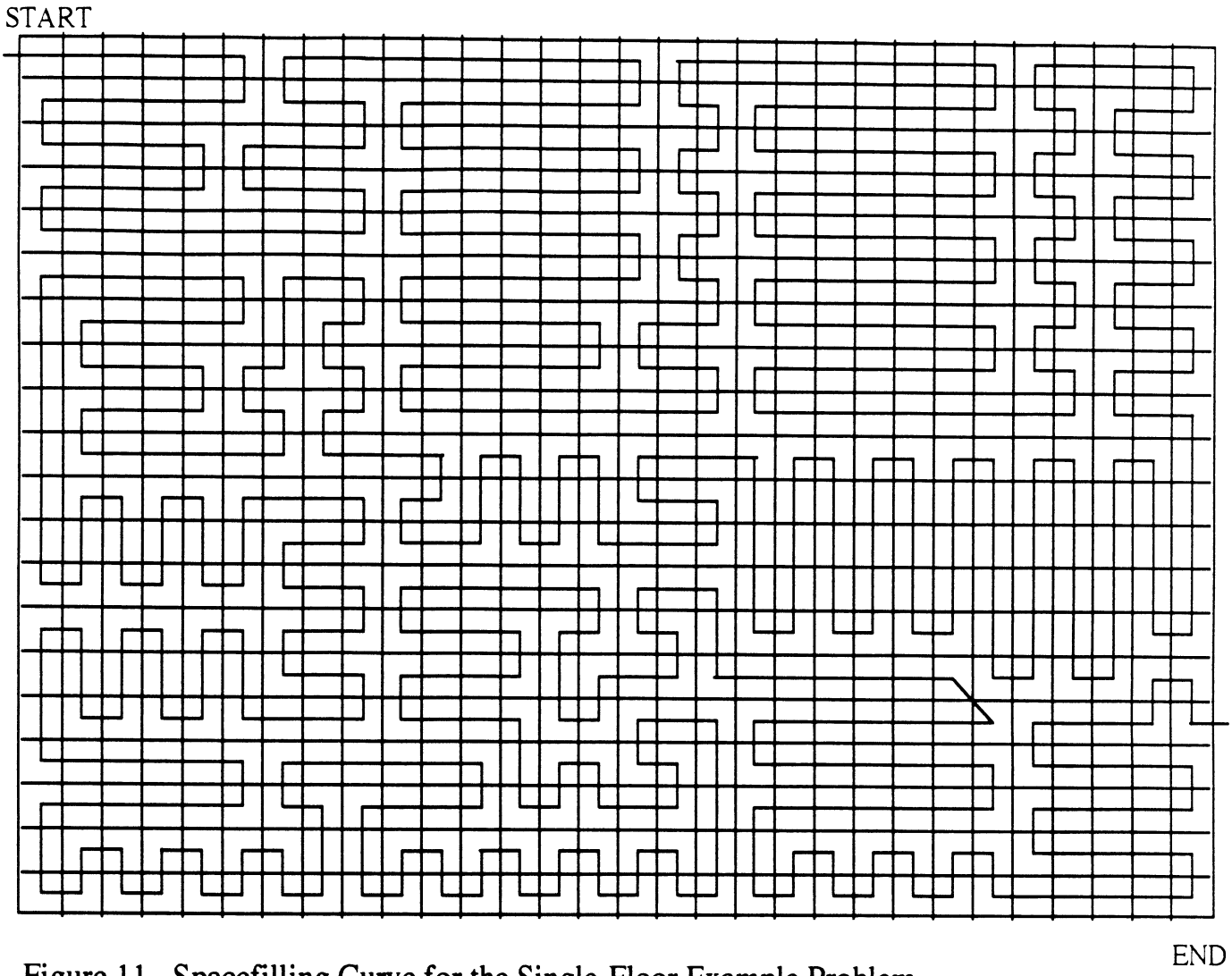


Figure 11. Spacefilling Curve for the Single-Floor Example Problem.

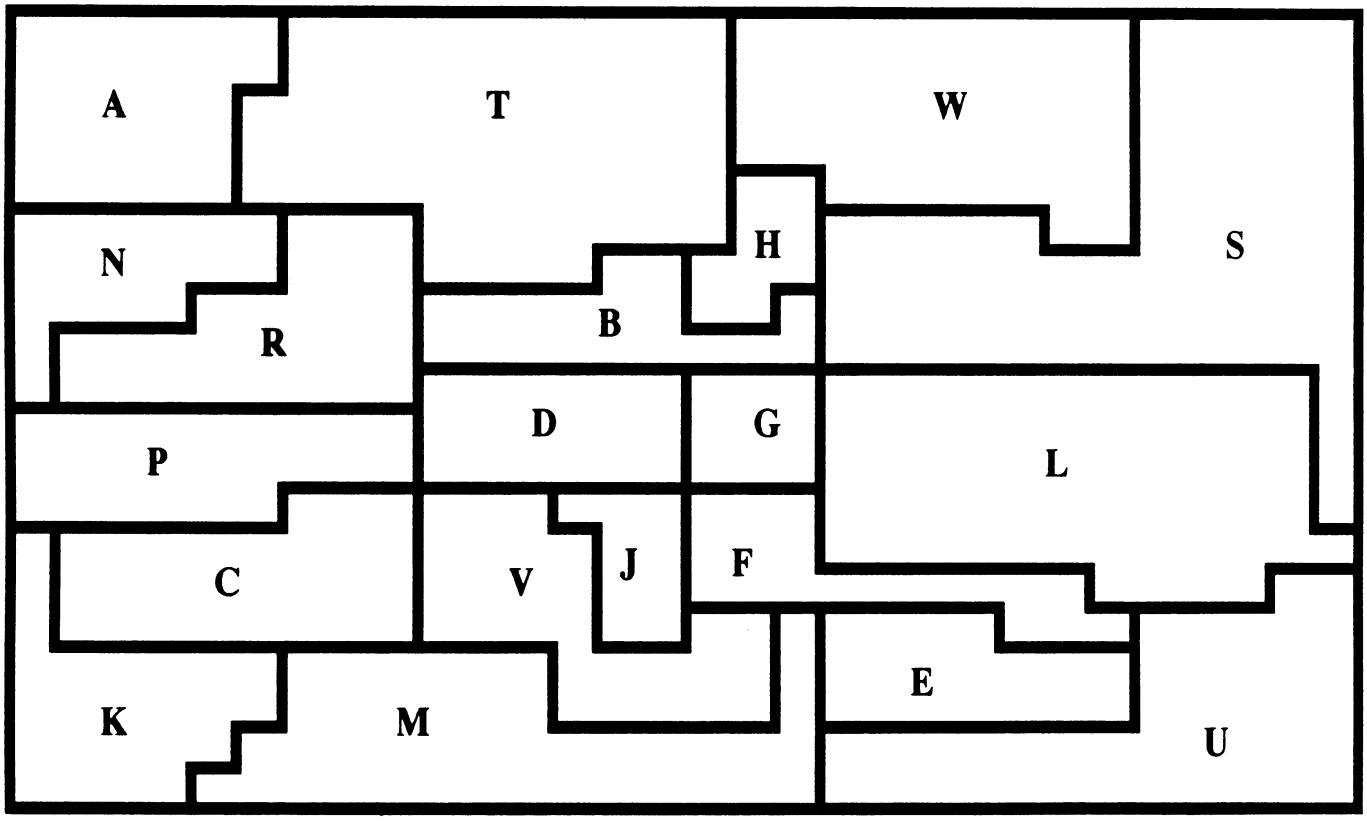


Figure 12. Final Layout Obtained by MULTIPLE for the Single-Floor Example Problem.

After adjusting for this compiler/language difference, solving the above example problem with MULTIPLE would require approximately 30 times more CPU time than CRAFT. Although this is a substantial increase in runtime, obtaining a solution for a 20-department layout problem in 2.5 minutes on a PC is well within reason.

CRAFT has the ability to consider three-way as well as two-way exchanges. To maintain equity in the above comparison, CRAFT was restricted to consider two-way exchanges only. MULTIPLE could be modified to consider three-way exchanges like CRAFT. In fact, the use of spacefilling curves allows MULTIPLE to consider the exchange of any three departments, while CRAFT is still limited to exchange only adjacent or equal area departments. Therefore, including three-way exchanges is likely to further improve the performance of MULTIPLE over CRAFT. However, this would also increase the runtime, and as we discuss in the next section, there are more promising extensions to MULTIPLE than adding three-way exchanges.

7. EXTENSIONS TO MULTIPLE

MULTIPLE may be extended in several directions. One extension to MULTIPLE concerns the number of (real) departments on each floor. Due to the pairwise exchange routine, the number of departments on each floor remains constant. Using dummy departments (which have negligible area requirements and no flow), MULTIPLE can vary the number of (real) departments on a floor. We have found dummy departments to be quite effective in most problems and recommend placing several dummy departments on each floor in the initial layout. Another method that would allow the number of departments on a floor to vary is based on exchanging two departments on a particular floor for one department on another floor; that is a "two–for–one" exchange.

In general, even if dummy departments are used, some two–for–one exchanges will not be possible to achieve through two or more two-way exchanges. Therefore, the cost of the final layout obtained with two-way and two–for–one exchanges is very likely to be less than that obtained with two-way exchanges only. We must stress that the potential improvement obtained by two–for–one exchanges is not limited to exchanges across floors. Although any two–for–one exchange *within a floor* may also be achieved with two two-way exchanges, since MULTIPLE is a steepest descent algorithm, if either one of the above two-way exchanges does not reduce the cost of the layout, the two–for–one exchange (which may still reduce the layout cost) will never be realized. Unfortunately, the number of two–for–one exchanges increases rapidly with the number of floors and the number of departments on each floor. For example, in a two floor facility, the number of two–for–one exchanges to be considered is equal to $\binom{n_1}{2}n_2 + \binom{n_2}{2}n_1$, where n_1 (≥ 2) and n_2 (≥ 2) denote the number of departments on floors 1 and 2, respectively. As a result, in general, the runtime with two–for–one exchanges will increase considerably.

The cost reductions that might be obtained with two–for–one exchanges, combined with the inability of MULTIPLE to temporarily accept solutions that do not improve the layout cost, suggests the use of a fundamentally different search procedure. We believe that extending MULTIPLE in a *simulated annealing* framework may be a very effective approach to the multi-floor facility layout problem. Simulated annealing may reduce the path dependency of MULTIPLE which is due, in part, to the steepest descent nature of

the algorithm. Moreover, since we use spacefilling curves to rapidly revise the layout, any (feasible) exchange can be accommodated simply as a new layout sequence on each floor. Spacefilling curves allow the analyst to use a very general exchange structure. Not only may two–for–one exchanges be considered along with three–for–one or three–for–two, but combinations of n_1–for–n_2 exchanges may be performed in a single iteration. Of course, the exact type of exchanges to consider at each iteration and how to generate them is a topic for future research in multi-floor layout.

8. IMPLEMENTATION AND CONCLUSIONS

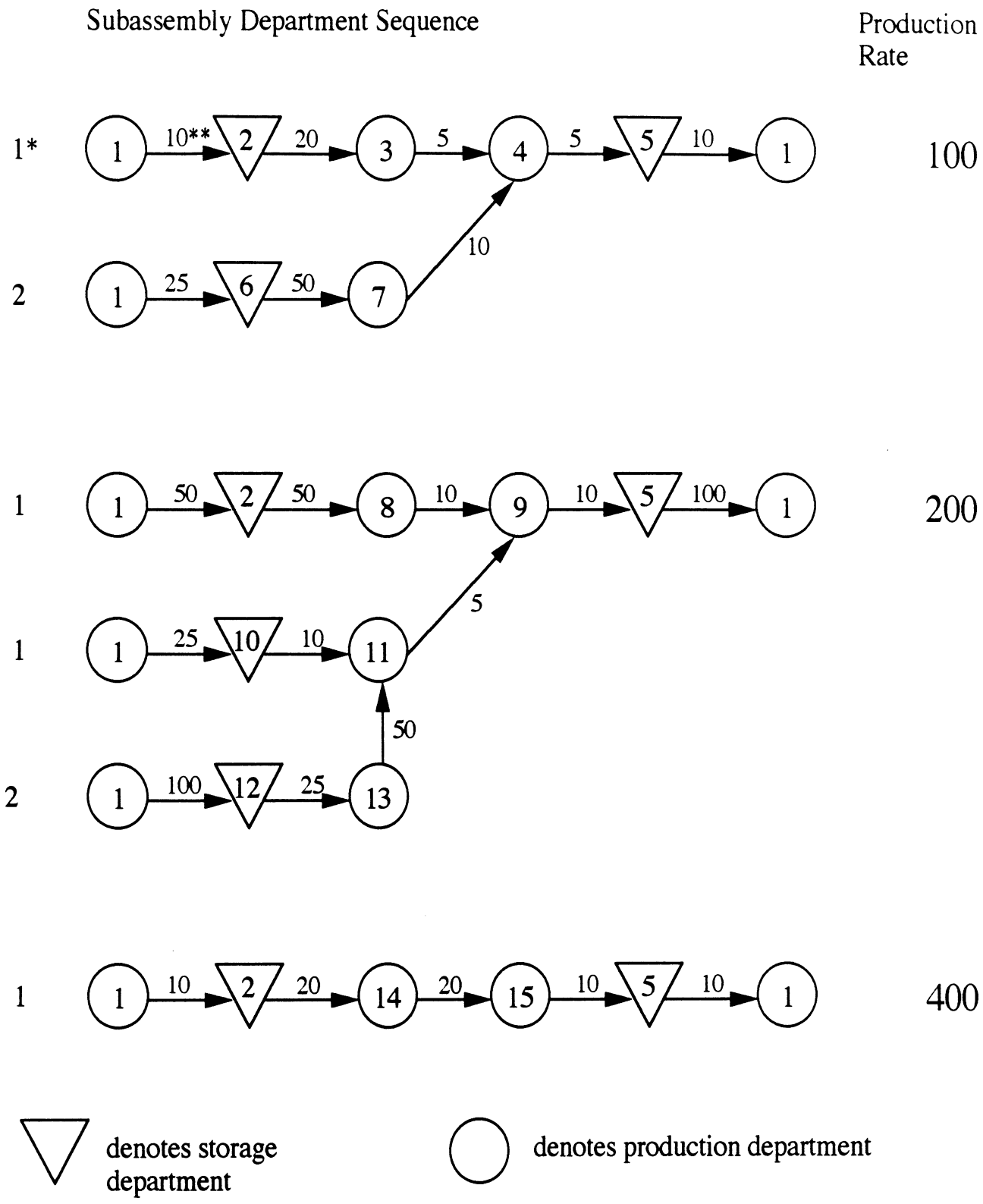
A tailored version of MULTIPLE was implemented in a large, four-floor production facility that produced a variety of products in large quantities. The facility had nearly 70 departments some of which were fixed. There were certain obstacles and some unused areas as well. The building was non-rectangular and had several adjoining sections built at different time periods. Hence, the floor loading capacity as well as the floor-to-ceiling distance varied from one floor to another, and from one section of the building to another. There were five elevators located at various points in the building.

Although management understood the data requirements of MULTIPLE, they were reluctant to generate a 70 by 70 flow matrix "from scratch" and preferred to have a more convenient way of converting their production figures to a flow matrix. To accommodate this need, we developed a "flow matrix generator" for MULTIPLE which we believe is a unique application of a well-known method used in production planning.

Suppose there are three types of finished products produced in a facility. As shown in Figure 13, we first determine the production route (i.e., the sequence of departments visited) for each finished product and its subassemblies. For each subassembly we next determine the number of subassemblies required for each unit of the finished product. For example, in Figure 13, the first product type has only one subassembly type and each finished product requires two subassemblies of that type. Lastly, for all the flows, we determine the basic units of flow. For example, in Figure 13, the subassembly of the first product flows from department 1 to department 6 in 25 pieces per container. Hence, the material flow from department 1 to department 6 (due to the first product) is equal to $2(100/25) = 8$ containers per time unit since the first product is assumed to be produced at a rate 100 units per time unit. Repeating the same computation for each flow, we automatically generate the entire flow matrix.

Of course, the data required for the above production facility was more extensive than that shown in Figure 13. However, once the data was collected, the flow matrix generator became a very useful tool. Simply by varying the planned production rates for all or some of the products, the plant manager was able to gauge the impact of the production plan on material flow within the facility. Subsequently, the flow matrix generator was also used to quantify vertical flow in the facility as a function of planned production rates and the current layout. Note that generating a flow matrix with the above approach is similar to the well-known "backward explosion" technique used in MRP systems.

We also collected data on the area requirements of each department. We were encouraged by the plant manager to compress the storage departments by up to 50%. Several shape and regional constraints were also considered. As mentioned earlier, given the na-



* Number in front of subassembly department sequence corresponds to the number of subassemblies in the final product.

** Number on top of the arrow between two departments corresponds to the size of the unit load moved between the two departments.

Figure 13. Sample Product Flow Chart to Generate a Flow Matrix.

ture of the building, we opted for a hand-generated "spacefilling curve" for each floor of the facility. Management also expressed their concerns over the steepest descent nature of MULTIPLE. Given their concern for department relocation costs, simply picking only the best exchange at each iteration was not the preferred approach. Therefore, instead of picking the best exchange and proceeding to the next iteration automatically, the program displayed the best twenty exchanges in decreasing order of layout cost savings. This presented the analyst with an opportunity to evaluate each exchange in terms of relocation costs and practical considerations.

In [14] relocation costs are considered within the model. Such an approach was not considered to be realistic by management since specifying the relocation cost *a priori* for each department would be time consuming and not possible without knowing where a department is to be relocated. With our approach, since the analyst observes the changes required to implement a particular exchange, he/she can evaluate its desirability based on layout cost savings and relocation costs. This feature was viewed as essential by management who had to economically justify major changes in the layout.

The tailored version of MULTIPLE was implemented on a 20 MHz DOS-based 386 PC (located at the site of the study) with a 387 coprocessor and a DOS-Extender. (The latter was required due to the large problem size.) Each iteration of the tailored algorithm required about 25 minutes. Given that layout problems are not solved on a daily basis, and that the PC offers portability and ease–of–use, the above runtime seems very reasonable. Using the tailored version of MULTIPLE and working jointly with the technical staff at the site, we were able to identify several cost-justified improvements in the layout.

In conclusion, MULTIPLE offers several advantages over CRAFT in single-floor applications. It also generates reasonably good layouts for multiple floor facilities without splitting departments. MULTIPLE's primary strength is derived from its use of spacefilling curves to rapidly re-layout one or two floors after an exchange. These curves provide considerable speed and flexibility. They also make it possible to consider more powerful exchange routines than two-way or three-way exchanges.

Two other key features of MULTIPLE is its ability to effectively handle a range of area requirements for each department and its use of a "new" shape measure to avoid irregular department shapes. Also, MULTIPLE explicitly considers the location of each existing and potential lift in the facility; however, it does not consider the throughput capacity of each lift. This aspect of the problem is subject to further research. Lastly, MULTIPLE is based on the "conventional" assumption of travel between department centroids. We would like to stress that MULTIPLE can be easily modified to accommodate alternative measures of layout efficiency. For example, one may use the "adjacency score" instead of the layout cost used by CRAFT. With appropriate modifications, it seems possible to use MULTIPLE in conjunction with other recent approaches such as the one proposed in [24], where a department may have several input/output (I/O) points and the layout design is integrated with the design of the flow network (which connects all the I/O points).

ACKNOWLEDGEMENT

The authors would like to thank Professor Roger V. Johnson for providing an updated copy of SPACECRAFT which was used in section 5.

APPENDIX

A spacefilling curve for a rectangle may be generated by the procedure shown in Figure A1. The rectangle is defined by r rows and c columns, while n (restricted to integer values) is determined such that the 2^{n-1} by 2^n rectangle (shown in Figure A2) is the largest rectangle to be contained in the r by c area. (If c is not greater than or equal to r, the rectangle should be reoriented.) Once n is determined, rectangles A, B and C are defined. If $r = c = 1$, both rectangles B and C are undefined and the procedure simply connects the appropriate points.

The original call to the SFC procedure will fill rectangle A with the appropriate Hilbert Curve (see [15] and Figure A3). Once rectangle A is filled, it will mirror image itself onto rectangle B, and then call the SFC procedure recursively to fill rectangle C. The difference between r and 2^{n-1} determines whether the spacefilling curve that begins at point 1, ends at point 2 or point 3. If the curve ends at point 2, then the SFC procedure will be called recursively to fill the remainder of rectangle A. Finally, each successive recurvsive call to the SFC procedure will link point 2 (or 3) from the previous rectangle to point 1 in the recursively defined rectangle. Figure A4 shows the spacefilling curve generated when the SFC procedure is applied to an 11 by 18 rectangle. It also shows rectangles A, B and C obtained with $n = 4$.

```
Procedure SFC ( r x c )                                  | c must be ≥ r; if not reorient;
    find max k such that ( 2^k ≤ c )                     | defining areas A, B & C
    find max m such that ( 2^(m-1) ≤ r )                 | in Figure A2;
    n = min ( k, m )                                     |
                                                         |
    if n = 0, then                                       | degenerate SFC (1 x 1) - point;
            only one possible solution, Return           |
    else                                                 |
            Continue                                     |
    draw 2^(n-1) x 2^(n-1) Hilbert Curve beginning at point 1 (Figure A2)   | decide where curve will end
            if r - 2^(n-1) > 0, then                     | based on r and c values;
                    end Hilbert Curve at point 2 (Figure A2)   |
                    SFC ( min {r - 2^(n-1), 2^(n-1) } ; max {r - 2^(n-1), 2^(n-) } )   | recursively call SFC with additional
            else                                         | area above curve;
                    end Hilbert Curve at point 3 (Figure A2)   |
    mirror image r x 2^(n-1) A onto r x 2^(n-1) B        | now A and B filled;
                                                         |
    if c > 2^n, then                                     | recursive call to SFC for C;
            SFC ( min {r , c - 2^n } ; max {r , c - 2^n } )   |
    else                                                 | if C is empty, then
            Return                                       | return to last SFC call;
End Procedure SFC                                        |
```

Figure A1. Procedure to Generate a Spacefilling Curve for an r by c Rectangle.

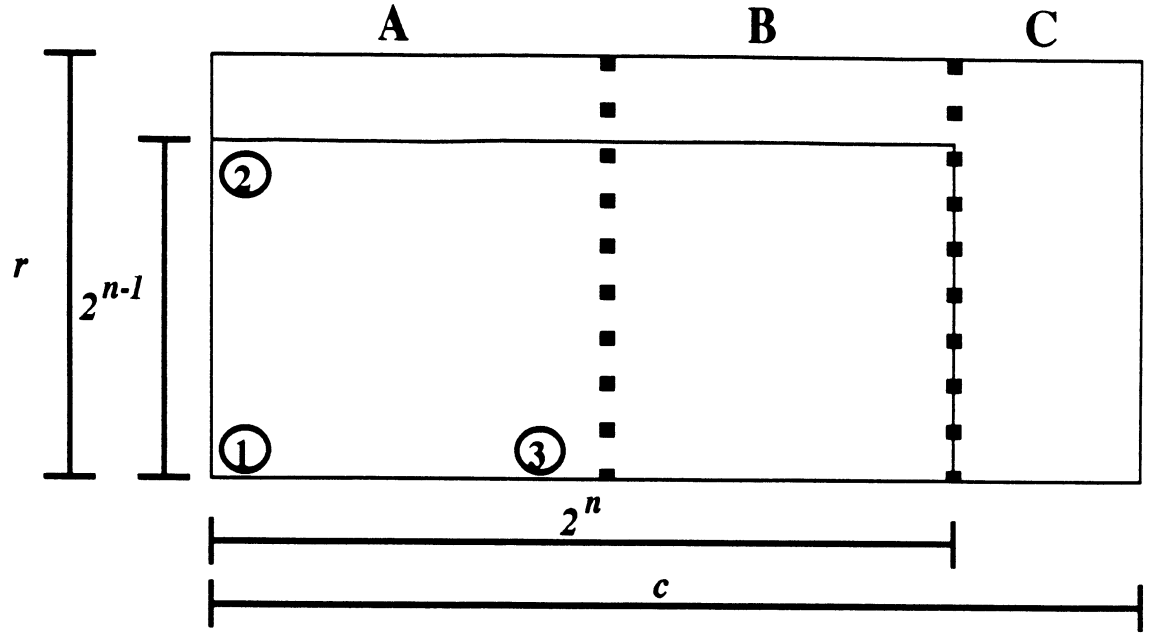
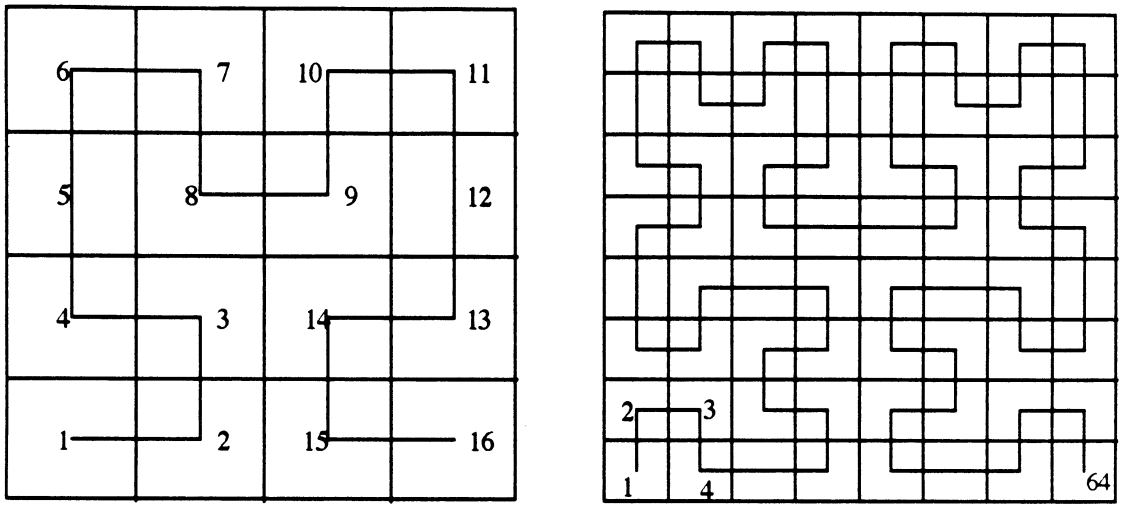
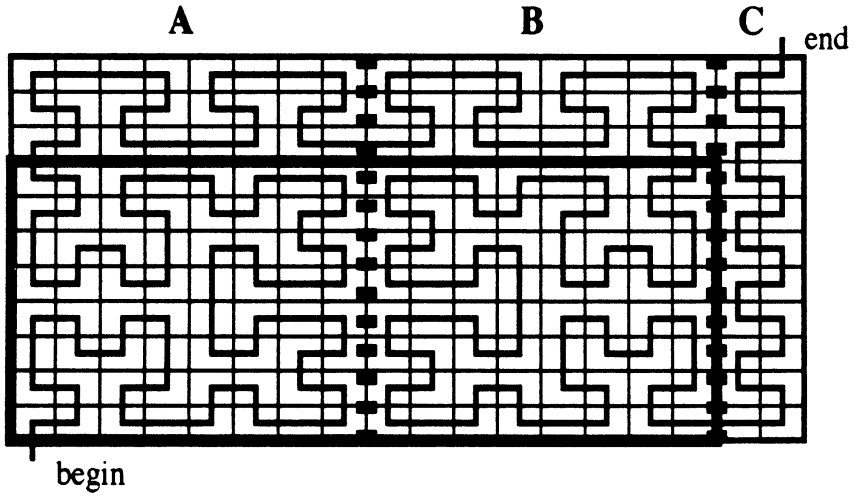


Figure A2. The Rectangles A, B and C as Defined by the SFC Procedure.



(a) Area = 16 (b) Area = 64

Figure A3. Examples of the Hilbert Curve (1891).



- Originally, $n = 4$

Figure A4. Spacefilling Curve Generated for an 11 by 18 Rectangle.

BIBLIOGRAPHY

[1] Armour, G. C. and Buffa, E. S., "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," *Management Science*, Vol. 9, No. 2, 1963, pp. 294–309.

[2] Bartholdi, J. J. and Platzman, L. K., "An $O(n \log n)$ Planar Traveling Salesman Heuristic Based on Spacefilling Curves," *Operations Research Letters*, Vol. 1, No. 4, 1982, pp. 121–125.

[3] Bartholdi, J. J. and Platzman, L. K., "Design of Efficient Bin-Numbering Schemes for Warehouses," *Material Flow*, Vol. 4, 1988, pp. 247–254.

[4] Buffa, E. S., Armour, G. C., and Vollman, T. E., "Allocating Facilities with CRAFT," *Harvard Business Review*, Vol. 42, 1964, pp. 136–158.

[5] Chisman, J. A., "The Clustered Traveling Salesman Problem," *Computers and Operations Research*, Vol. 2, 1975, pp. 115–119.

[6] Çınar, Ü., "Facilities Planning: A Systems Analysis and Space Allocation Approach," *Spatial Synthesis in Computer-Aided Building Design*, Charles M. Eastman (Ed.), Wiley, 1975, pp. 19–40.

[7] Deisenroth, M. P. and Apple, J. M., "A Computerized Plant Layout Analysis and Evaluation Technique (PLANET)," *Technical Papers 1962*, American Institute of Industrial Engineers, Norcross, Georgia, 1972.

[8] Donaghey, C. E. and Pire, V. F., "Solving the Facility Layout Problem with BLOCPLAN," Industrial Engineering Department, University of Houston, Houston, TX, 1990.

[9] Drezner, Z., "DISCON: A New Method for the Layout Problem," *Operations Research*, Vol. 20, 1980, pp. 1375–1384.

[10] Foulds, L. R., "Techniques for Facilities Layout: Deciding Which Pairs of Activities Should be Adjacent," *Management Science*, Vol. 29, 1983, pp. 1414–1426.

[11] Foulds, L. R. and Robinson, D. F., "Graph Theoretic Heuristics for the Plant Layout Problem," *International Journal of Production Research*, Vol. 16, 1963, pp. 27–37.

[12] Freeman, H., "Computer Processing of Line- Drawing Images," *Computing Surveys*, Vol. 6, 1974, pp. 57–97.

[13] Graves, G. W. and Whinston, A. B., "An Algorithm for the Quadratic Assignment Problem," *Management Science*, Vol. 17, 1982, pp. 453–471.

[14] Hicks, P. E. and Cowen, T. E., "CRAFT-M for Layout Rearrangement," *Industrial Engineering*, Vol. 8, 1976, pp. 30–35.

[15] Hobson, E. W., *The Theory of Functions of a Real Variable and the Theory of Fourier's Series, Volume I*, 3rd Edition, Cambridge University Press and Harren Press, Washington D.C., 1950.

[16] Jacobs, F. R., "A Note on SPACECRAFT for Multi-Floor Layout Planning," *Management Science*, Vol. 30, No. 5, 1984, pp. 648–649.

[17] Johnson, R. V., "SPACECRAFT for Multi-Floor Layout Planning," *Management Science*, Vol. 28, No. 4, 1982, pp. 407–417.

[18] Kaku, K., Thompson, G. L., and Baybars, I., "A Heuristic Method for the Multi-Story Layout Problem," *European Journal of Operational Research*, Vol. 37, 1988, pp. 384–397.

[19] King, J. and Johnson, R. E., "Silk Purses From Old Plants," *Harvard Business Review*, Vol. 61, No. 2, 1983, pp. 147–156.

[20] Lee, R. C. and Moore, J. M., "CORELAP– Computerized Relationship Layout Planning," *Journal of Industrial Engineering*, Vol. 18, No. 3, 1967, pp. 194–200.

[21] Lew, P. and Brown, P. H., "Evaluation and Modification of CRAFT for an Architectural Methodology," *Proceedings of the Design Methods Group First International Conference*, Gary T. Moore (Ed.), 1968, pp. 155–161.

[22] Liggett, R. S. and Mitchell, W. J., "Optimal Space Planning in Practice," *Computer Aided Design*, Vol. 13, 1981, pp. 277–288.

[23] Montreuil, B., Ratliff, H. D., and Goetschalckx, M., "Matching Based Interactive Facility Layout," *IIE Transactions*, Vol. 19, 1987, pp. 271–279.

[24] Montreuil, B., "A Modeling Framework for Integrating Layout Design and Flow Network Design," *Proceedings of the Material Handling Research Colloquium*, Hebron, Kentucky, 1990, pp. 43–58.

[25] Pire, V. F., "Automated Multistory Layout System," Unpublished Master's thesis, Industrial Engineering Department, University of Houston, Houston, TX, 1987.

[26] Ritzman, L. P., "The Efficiency of Computer Algorithms for Plant Layout," *Management Science*, Vol. 18, 1972, pp. 240–248.

[27] Seehof, J. M. and Evans, W. O., "Automated Layout Design Program," *Journal of Industrial Engineering*, Vol. 18, 1967, pp. 690–695.

[28] Tompkins, J. A. and Reed, R. Jr., "An Applied Model for the Facilities Design Problem," *International Journal of Production Research*, Vol. 14, No. 5, 1976, pp. 583–595.

[29] Tompkins, J. A. and White, J. A., *Facilities Planning*, John Wiley & Sons, New York, N.Y., 1984.

[30] "Candy Maker 'Sweetens' Efficiency with Pflow Lifts," *Industrial Engineering*, Vol. 22, No. 10, 1990, p. 60.

[31] "Pflow Conveyors Are an Alternative for Material Handling Needs," *Industrial Engineering*, Vol. 21, No. 1, 1989, pp. 63.

[32] "Plant Renovation: The Low-Cost Road to Success," *Material Handling Engineering*, Vol. 43, No. 2, 1988, pp. 41 – 51.

[33] "Vertical Conveyor Eliminates Need for New Construction," *Material Handling Engineering*, Vol. 40, No. 7, 1988, pp. 84–86.

[34] "Vertical Conveyors Increase Plant's Efficiency 33%," *Modern Materials Handling*, Vol. 40, Casebook Directory, 1985, pp. 113.

[35] "Vertical Lift Grows with New Distribution Center," *Industrial Engineering*, Vol. 22, No. 9, 1990, p. 81.

[36] "Vertical Reciprocating Conveyors: Flexible Handling Devices," *Material Handling Engineering*, Vol. 45, No. 9, 1990, pp. 81–85.