SOLVING THE FACILITY LAYOUT PROBLEM WITH SIMULATED ANNEALING

Russell D. Meller
Yavuz A. Bozer

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

Technical Report 91-20

July 1991
Revised March 1992

Solving the Facility Layout Problem with Simulated Annealing

Russell D. Meller
Yavuz A. Bozer

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

March 23, 1992

Abstract

In this paper we present an application of simulated annealing to facility layout problems with single and multiple floors. The facility layout problem is highly combinatorial in nature and generally exhibits many local minima. These properties make it a suitable candidate for simulated annealing. Using spacefilling curves and a compact representation of the layout, we develop a new improvement-type layout algorithm based on simulated annealing. This algorithm offers increased flexibility when considering the exchange of departments, and it achieves low-cost solutions that are much less dependent on the initial layout. We compare the performance of the simulated-annealing based algorithm with two steepest descent heuristics, namely, CRAFT and MULTIPLE, which also attempt to improve a layout through department exchanges. We also compare our results with those obtained from a simulated-annealing based layout algorithm presented by Tam. Unlike other simulated annealing papers which typically present a statistical experiment to evaluate the effect of numerous control settings, all the experiments presented in this paper were conducted with control settings that are constant or easily specified. This approach facilitates the application of the proposed algorithm to real-life facility layout problems in both single and multiple floor facilities. Although the algorithm presented here can be applied to many types of facilities, our primary focus is on production facilities.

1 Introduction

A common problem encountered in industrial engineering is the determination of department locations in a facility, that is, the determination of a layout. Each department has given area requirements and certain interactions that occur with other departments. These interactions may be in the form of material, resource, or information flow. The overall objective is to develop an efficient and feasible layout. The efficiency of a layout is measured by the physical arrangement of the departments and how the above interactions occur. The feasibility of a layout, on the other hand, is established by satisfying the area requirements of the departments, while ensuring that departments do not overlap and that none of the constraints imposed by the site plan or the building (including fixed and unusable areas) are violated.

The problem of determining the layout of a facility while considering the above constraints is generally defined as the *facility layout problem*, which can be formulated as a quadratic assignment problem (QAP) or a nonlinearly constrained mixed integer programming (MIP) problem [17]. The QAP is *NP-complete* [7] and solving problems with 18 or more (equal area) departments is currently not possible. The nonlinear MIP formulation, which was developed specifically for the facility layout problem, is fundamentally different than previous QAP linearizations. However, it has been proposed only recently and it contains many binary variables. Hence, it is likely to be difficult to solve unless a specialized procedure is developed.

The first solution obtained from the facility layout problem is generally referred to as a *block layout*, which shows the departmental borders (or departmental "footprints") that satisfy the area and shape requirements. A block layout generally does not specify the aisles and the arrangement of the objects within each department. (Such particulars are usually shown later in a *detailed layout.*) Obviously, in a block layout the departments must neither overlap nor be divided (or "split") into two or more pieces. Also, those departments that are fixed at particular locations, or those areas of the building that are unusable, must be accurately represented in the solution.

In this paper we consider both single-floor and multi-floor facilities. Multi-floor facility layout differs from single-floor facility layout in two fundamental ways. First, the time to travel between two departments can no longer be represented only by the horizontal distance between the two departments since traveling between two departments on different floors requires access to a *lift.* (Here we define a lift to be any material handling device that can be used to transport material between two or more floors.) Hence, if two departments are located on separate floors, not only may the horizontal travel increase, but additional vertical travel will be necessary. In the proposed algorithm we assume that the existing (or potential) lift locations, which are an important consideration in a multi-floor facility, are given and fixed.

The second principal difference between single-floor layout and multi-floor layout is the determination of area feasibility. According to Bozer, Meller and Erlebacher [2], "in the single-floor layout problem, as long as the total usable floor space is greater than or equal to the total area required by all the departments, there are no feasibility issues associated with space availability. However, in a multi-floor layout problem, even if there is sufficient usable floor space in the facility, the number of departments that can be assigned to any floor is limited by available space on that floor." Hence, in multi-floor facilities, some layout alternatives may not be area feasible.

The paper is organized as follows. In §2 we present our assumptions and some of the notation we use throughout the paper, as well as related previous work on the facility layout problem. The level of influence the initial layout has on the cost of the final solution is explored in §3. The simulated-annealing based algorithm is described in §4. The results of the comparison between simulated annealing and other heuristics developed for single and multiple floor layout problems are detailed in §5 and §6, respectively. Lastly, in §7 we state our conclusions.

2 Problem Description and Previous Work

Without exact procedures to solve facility layout problems of more than a few non-equal area departments, most of the research has been aimed at developing heuristic procedures. After we present some assumptions, we describe the procedures that are compared in §5 and §6. Previous work in simulated annealing is then presented in the remainder of the section.

2.1 Previous Work on the Facility Layout Problem

A review of the literature indicates that there is no general consensus on the type of objective function to use in facility layout problems. In some algorithms, the objective function value is based on the number of departments that are adjacent in the layout, while others consider the sum of weighted distances where the weights are proportional to the unit cost of flow between two departments. All the algorithms considered in this paper, as well as the proposed simulated-annealing based algorithm, utilize a well-known objective function that is based on (flow) x (unit cost) x (rectilinear distance between department centroids). This decision was made, in part, because adjacency-based objective functions ignore department placement if any two departments are not adjacent, whereas a weighted distance objective function does not.

Note that the exact travel distance between two departments is difficult to determine

at this stage since the actual location of the input/output (I/O) points of a department (and sometimes the flow associated with individual I/O points) is usually not known until the block layout is completed and a detailed layout is generated for each department. Thus, approximating the distance between two departments by the distance between their centroids is a generally conservative approach, in that it does not allow for any of the benefits that may be derived from a more appropriate placement of the I/O points. We must stress, however, that the simulated annealing algorithm we present here can be used with alternative objective functions as well. These include adjacency-based objective functions with or without multiple I/O points for each department.

A fairly extensive review of *improvement-type* layout algorithms for single-floor and multi-floor facilities is presented by Bozer, Meller and Erlebacher [2]. The authors show that, on single-floor problems, MULTIPLE is superior to CRAFT [1]. Unlike CRAFT, MULTIPLE can exchange any two departments even if they are neither adjacent nor equal in area. On multi-floor problems, they show that MULTIPLE yields better solutions than SPACECRAFT, which is another multi-floor improvement-type layout algorithm developed by Johnson [12]. In [2] it is noted that SPACECRAFT is likely to split departments across two or more floors, and that SPACECRAFT considers exchanging only adjacent or equal area departments just like CRAFT. Other multi-floor algorithms, such as ALDEP [20], BLOCPLAN [6], SPS [16], and MSLP [15], are also reviewed in [2]. These algorithms have limited use in most multi-floor plant layout problems because they have *one or more* of the following shortcomings: (i) the lift locations are either not considered at all or only one centrally located bank of lifts is allowed; (ii) all the departments are assumed to be equal in area; (iii) once the departments are assigned to floors, the layout of each floor is developed independently of the layout of the other floors.

In addition to the work reviewed in [2], Tam recently [21] introduced an improvement algorithm that incorporates a simulated-annealing search strategy. The layout is represented in a continuous fashion, rather than the discrete fashion adopted by MULTIPLE and CRAFT. The approach developed in [21] is based upon a similar objective function and assumes rectilinear travel.

In this study, for single-floor problems we will compare our simulated annealing algorithm with CRAFT, Tam's approach, and MULTIPLE. For multi-floor problems, we will compare our results with those obtained from MULTIPLE. Therefore, in the remainder of this section we will focus on CRAFT [1], Tam's approach [21] and MULTIPLE [2] only.

2.1.1 Single-Floor Algorithms

As described by Ritzman [19], a number of computer-based, heuristic layout algorithms have been developed over the years for single-floor facilities, "but few have been as success-

ful as CRAFT." CRAFT attempts to improve a given initial layout in a steepest descent manner through two-way and three-way exchanges of departments that are equal in area and/or adjacent in the layout. The objective function is based on (flow) x (unit cost) x (rectilinear distance between department centroids). That is, letting the flow between departments i and j be denoted by f_{ij}, and the cost to move one unit load, one distance unit between the two departments be denoted by c_{ij}, the objective used by CRAFT is to

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

where N denotes the number of departments and d_{ij} denotes the rectilinear horizontal distance between the centroids of departments i and j. The interdepartmental flow and cost values are given as data, while the distances between the departments, d_{ij}'s, are the decision variables obtained directly from the layout.

MULTIPLE [2], when utilized in a single-floor setting, uses the objective function of equation (1) and has data requirements similar to CRAFT. With the use of spacefilling curves, MULTIPLE considers the exchange of any two departments, thus improving upon CRAFT (MULTIPLE's strength over CRAFT was originally presented in [2]; further empirical results are given in §5). An example may help demonstrate how spacefilling curves are used to determine the layout of a floor in MULTIPLE. Consider the single-floor initial layout shown in Figure 1a. The layout of the floor is determined by assigning each grid square to a particular department *according to the sequence dictated by the spacefilling curve.* Of course, the number of grid squares assigned to a department is dictated by the area of that department.

Note that, using this approach, a layout is defined only by the spacefilling curve and the *sequence* in which the departments appear on the curve. Hence, any two departments can be exchanged simply by interchanging their position in the above sequence. Following an exchange, the new layout is obtained simply by reassigning each grid square to the appropriate department. For example, in Figure 1a, if departments 2 and 5 are exchanged, one will obtain the layout shown in Figure 1b. Given the appropriate spacefilling curve, MULTIPLE will not split departments and it can accurately represent fixed departments and restricted areas. (If a department is fixed, the curve simply "bypasses" all the grid squares assigned to that department when the spacefilling curve is generated; that is, the spacefilling curve is not routed through that department.)

The approach introduced in [21] is considerably different than previous heuristic approaches. A layout is represented as a collection of rectangular partitions that are specified by a "slicing tree." A slicing tree consists of branches and branching operators, which specify if departments on opposite sides of the branch are to the east(west) or to the north(south) of one another. Given a slicing tree and area values, the department

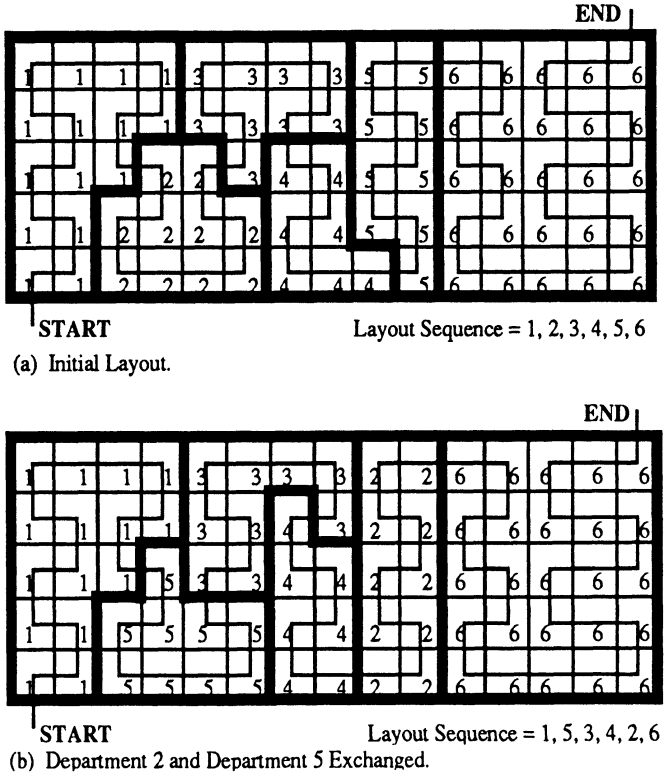

Figure 1: Using Spacefilling Curves to Construct Layouts.

locations are determined by recursively partitioning a rectangular block and placing departments within the partitions according to the branching operators. A characteristic of the this approach is that all departments are limited to rectangular shapes due to this rectangular recursive partitioning. Exceptions to the rectangular department shapes only occur in the presence of fixed departments, which improves the algorithm's application domain.

An initial layout is defined by first generating a slicing tree. The generation of slicing trees is not the subject of [21], but guidelines are included. Once an initial slicing tree has been generated, two-way department exchanges within the slicing tree or exchanges in branching operators, leads to a new layout. The approach uses a simulated-annealing search heuristic to find an improved layout.

Since the approach presented in [21] may produce undesirable department shapes, two shape constraints are defined and incorporated into the formulation using penalty methods. That is, when a department's shape violates the user defined constraint, a penalty is added to the objective function. Since departments are restricted to rectangular shapes (when there are no obstacles), shape measurement is relatively straightforward.

2.1.2 Multi-Floor Algorithm

Multi-floor problems require the user to specify, in addition to the single-floor data requirements, potential or existing lift locations and a spacefilling curve for each floor. Furthermore, along with the horizontal material handling costs (c_{ij}^H), the cost to move one unit load, one vertical distance unit, between two departments i and j must be specified as c_{ij}^V. The objective used in multi-floor problems is to

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

where d_{ij}^H (d_{ij}^V) denotes the horizontal (vertical) distance between department i and department j. Distances are measured rectilinearly between two department centroids, or between a department centroid and a lift when the flow occurs between departments on different floors. In the latter case, the flow is assumed to go through the lift which minimizes the total horizontal distance traveled to and from the lift.

The number of algorithms available for solving multi-floor facility layout problems, as compared with single-floor algorithms, is quite small. As remarked earlier, if the problem has non-equal area departments, multiple lift locations, and departments are not to be split across floors, then MULTIPLE is the only improvement algorithm that may be used effectively.

MULTIPLE exchanges departments within a floor in the manner specified in the previous section. However, when departments on different floors are exchanged, MULTIPLE simply exchanges the positions of departments in the sequence of their respective spacefilling curves. In the process, MULTIPLE considers a user defined *range of acceptable area values* for each department and any slack area on each floor of the facility. This increases the program's flexibility and ability to model the problem realistically [2].

2.2 Previous Work in Simulated Annealing

Both CRAFT and MULTIPLE are steepest descent algorithms. Consequently, they are both "path dependent" heuristics, influenced by the initial layout and the exchanges performed at each iteration. Furthermore, since both algorithms are greedy in nature and will not consider department exchanges that may temporarily increase the objective function, they are subject to terminating at the first local minimum encountered in the path. Therefore, although there are other factors that influence the final solution obtained by CRAFT or MULTIPLE, altering the initial layout alone leads to a wide variation in the quality of the final layout. The extent of this variation will be discussed in §3.

The relationship between combinatorial optimization problems and statistical mechanics was first suggested by Kirkpatrick, Gelatt and Vecchi [13], where a feasible solution and the associated objective function value is characterized by a *representation.* Small changes are made to the representation of the current solution to obtain a *candidate representation.* The objective (that is to be minimized) is evaluated with the candidate representation and measured relative to the current representation. The change in the objective function value (current minus candidate) is denoted by ΔE. If $\Delta E > 0$ (i.e., the candidate layout produces a lower cost), the candidate representation becomes the new current representation. If $\Delta E \leq 0$, the candidate representation is accepted with probability $P(\Delta E)$, where

$$P(\Delta E) = \exp(\Delta E / k_b T). \tag{3}$$

In the above expression, T is the *temperature* and k_b is Boltzmann's constant. In applying simulated annealing to combinatorial problems, Boltzmann's constant has no meaning and the temperature setting, T, is used to control the annealing schedule. Of course, in addition to the current and candidate representations, the *current best* representation is maintained throughout the execution of the algorithm.

Simulated annealing was introduced as a heuristic approach to solve numerous combinatorial optimization problems in place of schemes "which could get 'stuck' on inferior solutions" [22]. The relative success of simulated annealing has been explored by [13], [8], and [11], among others. Heragu and Alfa [10] applied simulated annealing to the problem of determining the relative location of departments within a facility when both the dimensions and the orientation of each department are given *a priori.* As mentioned previously, in [21] simulated annealing was applied to the facility layout problem. The representation used in [21] is different than the grid-square representation to be described in §4.

In addition, Burkard and Rendl [4] and Wilhelm and Ward [22] have implemented simulated annealing algorithms to solve the QAP. In fact, Wilhelm and Ward compared their simulated annealing algorithm with CRAFT. However, since they considered only *equal area* departments, CRAFT was used simply as a two-way or three-way exchange heuristic applied to the QAP. Nevertheless, the research of Wilhelm and Ward influenced the development of the simulated-annealing based layout algorithm presented in this paper. For a more complete review of simulated annealing applications the reader may refer to Collins, Eglese and Golden [5].

3 Initial Layout Bias and the Objective Value

For layout algorithms, the bias due to path dependency in the steepest descent approach was first examined by Nugent, Vollman and Ruml [18] when they developed a

non-deterministic approach to CRAFT that is known as the Biased Sampling Technique. Instead of selecting the exchange that is estimated to reduce the layout cost the most, each exchange (that is estimated to improve the layout cost) is assigned a non-zero probability. (The sum of these probabilities is equal to one.) A random number sampled between 0 and 1 and the aforementioned probabilities are used to select the exchange to be performed in a given iteration. Repeating the biased sampling approach many times (10 times in [18]) yielded solutions of slightly higher quality than CRAFT, but with a substantial increase in runtime. The bias due to path dependency was evaluated by executing the algorithm with alternative initial layouts.

Examination of the results presented in [18] yields the following statistics. Five initial layouts, with all *equal area* departments, were given as input to CRAFT and Biased Sampling. (Biased Sampling was run ten times with ten different random number streams to select the exchange at each iteration). For every problem set, the difference between the maximum and the minimum observed cost divided by the minimum observed cost was used as an indication of the effect that the change in the initial layout had on the quality of the solutions. For problems with 5, 6, 7, 8, 12, 15, 20, and 30 equal area departments, the above measure for CRAFT was 16.0%, 7.0%, 7.7%, 15.9%, 6.6%, 9.8%, 2.3%, and 4.0%, respectively. Likewise, for Biased Sampling, the percentages were 16.0%, 7.0%, 2.7%, 0.0%, 2.8%, 2.3%, 0.8%, and 2.2%, respectively.

The initial layout bias generally decreased when Biased Sampling was used to solve the above problems. The runtimes were approximately ten times longer than CRAFT. Simulated annealing may also lead to runtimes that are much longer than CRAFT. However, simulated annealing represents a more formalized and generalized approach than the Biased Sampling Technique. Moreover, the Biased Sampling Technique is still "greedy" in nature in that it will never select an exchange that will increase the estimated objective function value. In this regard, unlike simulated annealing, the Biased Sampling Technique will always terminate at the first local optimal solution that it encounters during the improvement process.

In short, CRAFT and MULTIPLE, like other improvement-type algorithms, are influenced by the initial layout, the exchanges considered at each iteration, and their greedy nature. Moreover, in multi-floor layout problems, if one uses only two-way or three-way exchanges, the initial assignment of departments to floors may make it impossible to obtain a good solution. Ideally, simulated annealing should reduce the bias due to the initial layout, remove some of the exchange restrictions, and yield better solutions. The simulated-annealing based layout algorithm outlined in §4 is shown to accomplish these tasks on the test problems detailed in §5 and §6.

4 Simulated-Annealing Based Layout Algorithm

Before describing the algorithm in detail, we must discuss the representation utilized throughout the simulated-annealing based algorithm. We also need to establish a procedure for generating a candidate representation (i.e., a candidate solution) from a current representation. As shown earlier in Figure 1, a common representation used in facility layout problems is to divide the given area into *grids*, where the area of a grid is equal to a common denominator of the departmental areas. Generally speaking, a smaller grid size yields higher "resolution," but at the expense of increased computational effort. (The appropriate grid size to use depends on the particular application and the specific values of the departmental area requirements.) Since the actual area values that a department assumes in the layout affects the distances between departments, and hence, the objective function, it would be difficult to make technically accurate comparisons of layout objective function values. Thus, in the test problems, all department areas are set equal to the minimum acceptable value specified by the user. However, to allow for flexibility, slack area was provided within the facilities.

For a *single-floor* problem, a sequence of departments such as the one shown in Figure 1a, coupled with a spacefilling curve and departmental area requirements, uniquely defines a layout. Given that the spacefilling curve and the area requirements remain the same throughout the algorithm, in a single-floor problem a layout can be represented as a sequence of department numbers, which we refer to as a "layout sequence." In a single-floor problem, any layout sequence is feasible.

In a *multi-floor* setting, a layout sequence alone does not completely define a layout since such a sequence does not indicate the floor number of a department. However, if "dividers" are used to divide the given sequence into floors, one can still use the layout sequence as a unique representation. The use of such "dividers" can perhaps be best described through an example.

Consider a 3-floor, 10-department problem, where there are 12 units of space available on each floor. The departmental area requirements are given in Table 1. Note that the total available space is equal to 36 units while the departmental area requirements sum up to 33 units. Consider next the following layout sequence with no dividers: 1-2-3-4-5-6-7-8-9-10. To determine the divider positions, we start with the first department in the layout sequence and assign each department, one at a time, to the first floor. When the remaining space on the first floor is not large enough to accommodate the next department in the sequence, we place a divider in the layout sequence. Starting with this next department, we then repeat the above process for the second floor and subsequently for the third floor.

For the example problem, the above procedure yields the following layout sequence

Dept	1	2	3	4	5	6	7	8	9	10
Area	2	1	3	5	4	3	2	1	6	6

Table 1: The departmental area requirements for an example problem with 10 departments.

with dividers: 1–2–3–4 / 5–6–7–8 / 9–10. That is, departments 1, 2, 3, and 4 are on the first floor, 5, 6, 7, and 8 are on the second floor, and 9 and 10 are on the third floor. The sequence is feasible since the total floor space occupied on the first, second, and third floors is equal to 11, 10, and 12 units, respectively. Suppose two alternative sequences (without dividers) are given as follows: 9–4–1–2–7–5–8–3–10–6 and 10–6–4–3–1–5–8–2–7–9. Applying the above procedure to the first sequence we obtain 9–4 / 1–2–7–5–8 / 3–10–6, which is also feasible since the first, second, and third floors have 11, 10, and 12 units of floor space occupied, respectively. However, for the second layout sequence we obtain 10–6 / 4–3–1 / 5–8–2–7–9, which is infeasible since the departments assigned to the third floor require a total of 14 units of floor space.

Obviously, with the above procedure, only the last floor may cause infeasibility. If our only objective was to find a feasible solution, the above problem can be solved as a bin packing problem with a given number of bins. However, within the context of simulated annealing, our objective is to generate an alternative feasible solution from the current solution which is, of course, feasible. To accomplish this, we use the following procedure (which is formally presented as part of the overall algorithm we show in section 4.1): given an initial layout sequence, we first randomly sample a number between zero and one for each department. We refer to this number as the "department address." We then sort the departments according to their address and obtain a new layout sequence.

In a single-floor problem, we need not check the new layout sequence for feasibility. Rather, we can directly construct the new layout by using the new sequence and the given spacefilling curve. In a multi-floor problem, however, we first apply the aforementioned procedure to identify the dividers and check the new layout sequence for feasibility. (Note that the position of the dividers in the new sequence does not necessarily match the position of the dividers in the current sequence.) If the sequence is feasible, then we simply construct the new layout by using the new sequence (with dividers) and the spacefilling curve on each floor. If the sequence is not feasible, we sample new department addresses and sort the departments according to their new addresses to obtain another layout sequence.

Obviously, one may develop more sophisticated schemes to generate feasible layout sequences. However, the above scheme (based on department addresses) is "fast" and easy to implement. Furthermore, in multi-floor problems, coupled with the approach we use to determine the divider positions, the above scheme may vary the number of departments on each floor from one feasible layout to the next. In this regard, it is superior to two-way and three-way exchanges, where the number of departments on each floor remains the same. Furthermore, the above scheme may "automatically" generate 2–for–1 or 2–for–3 exchanges (or other combinations) without the specialized code required for implementing such exchanges.

From an annealing standpoint, however, a more systematic approach is required to vary the layout sequence. Generally speaking, the change in the layout from one (feasible) layout sequence to the next is not a random process. Rather, the "difference" between the current solution and the candidate solution (which is generated from the current solution) is generally controlled by a user-specified parameter.

In order to control the "difference" between the current and the candidate solutions, we modified the above procedure as follows. Recall that, in the current layout sequence each department has an address between zero and one. Starting with the first department, we sample a random number between zero and one. If this random number is *less than* a certain user-defined *critical value*, then we change the department address by sampling a second random number (between zero and one). Otherwise, the department address remains the same. To generate a candidate solution, we repeat this procedure for all the departments in the current layout sequence and, if one or more departments have a new address, we re-sort the departments. With such an approach, if the critical value is close to one, the candidate layout sequence is likely to be considerably different than the current sequence. In contrast, if the critical value is close to zero, the candidate sequence is likely to be similar to the current one. That is, the relative position of each department in the sequence will not change considerably. As a result, even after the new divider positions are determined, the candidate layout is unlikely to be considerably different from the current one.

Prior research ([8], [13], and [22]) in simulated annealing indicates that the *annealing schedule* and the *determination of equilibrium at each temperature* are two factors that are critical for its success. The annealing schedule involves the determination of an initial temperature and its systematic reduction. (The temperature is reduced when the system has reached equilibrium at the current temperature.) In this study, unlike the approach used by Wilhelm and Ward [22], we use an initial temperature setting that is based on the initial value of the objective function. (See §4.2.) The justification for using such an initial temperature setting is presented in [11]. The temperature reduction scheme we use, on the other hand, has been used before in other simulated annealing algorithms. To reduce the current temperature we simply multiply it by α, where $0 < \alpha < 1$. The

value of α, which remains the same throughout the algorithm, is input by the user. Our preliminary results indicate that an α value of 0.80 generally works quite well.

To determine whether the "system" has reached equilibrium at the current temperature, we use the approach described in [22]. Under this approach, the algorithm proceeds from one *epoch* to the next, *within the same temperature setting.* For our problem, an epoch is defined by the number of accepted candidate layout sequences. To detect equilibrium, the mean value of the objective function is maintained for each epoch. When the mean objective function value for the current epoch is within a given percentage of the overall mean of the previous epochs at the current temperature, the "system" is assumed to be in equilibrium at the current temperature, which is then reduced by a factor of $1-\alpha$. (By definition, a temperature consists of two or more epochs.) The number of accepted candidate layout sequences per epoch and the above percentage difference between the two means are specified *a priori* by the user.

In the next section, we develop the necessary notation to formally present the procedure to generate candidate layouts and the simulated annealing algorithm itself.

4.1 Notation and the Algorithm

The following notation will be useful when discussing the Simulated-Annealing Based Layout Evaluation algorithm, that is, SABLE. When appropriate, variables were used in a manner similar to their use in [22].

s^0 = initial sequence of departments — when combined with the spacefilling curve and area data, it uniquely represents the initial layout.

s^* = "current best" sequence of departments which corresponds to the lowest cost layout identified by the algorithm.

s = the current sequence of departments.

s' = the candidate sequence of departments.

T = $\{t_1, t_2, t_3, \dots\}$ — a set of annealing schedule temperatures, where $t_i = t_0(0.8)^{i-1}, \forall i > 1$.

t_0 = initial temperature.

e = epoch length — fixed number of candidate sequences that will be accepted during each epoch.

$f_j(s)$ = objective value of the jth accepted candidate solution, s.

$\overline{f}_e$ = $\left(\sum_{j=1}^{e} f_j(s)\right)/e$ — the mean objective function value of an epoch with e accepted candidate sequences.

$\overline{f}'_e$ = the overall mean of objective function values for the sequences accepted during the epochs previous to the current one for a given temperature.

ϵ_i = an error constant used to determine whether the system is in equilibrium at temperature i.

$\boldsymbol{a}$ = a vector of (0,1) random variables that designates the address for each department in the layout sequence.

$\boldsymbol{b}$ = a vector of (0,1) random variables that determines if the department addresses change.

β = critical value to determine whether a new address is assigned to a department, $\beta \in (0,1)$.

M = a number that is specified *a priori* to control the maximum number of epochs considered.

I = a counter to record the temperature setting which produced the "current best" sequence, s^*.

N = the maximum number of successive temperature settings that do not produce a new s^*, $I \leq N$.

The variables $t_1, e, \epsilon, \beta, M$ and N are control variables that are set by the user. The settings used for the control variables will be described in subsequent sections. A general outline of SABLE follows.

For a given annealing schedule of temperatures, $T = \{t_1, t_2, \ldots\}$,

Step 1 Determine an initial sequence that is area feasible, s^0, which may represent the existing layout. Set $s = s^0$, $I = 1$, $i = 1$ and $\boldsymbol{a}$ according to s^0.

Step 2 Compute the layout cost of the current assignment, $f(s)$, from equation (2).

Step 3
- a Select a vector of random variables $\boldsymbol{b}$, where each element of $\boldsymbol{b}$ is distributed $U(0,1)$.
- b For each element of $\boldsymbol{b} \leq \beta$, generate a new element of $\boldsymbol{a}$ distributed $U(0,1)$. Sort $\boldsymbol{a}$ to determine the new sequence. If the sequence is area feasible then set the sequence to s' and to to step 3c; otherwise go to step 3a.
- c Evaluate the resulting change in layout cost, $\Delta f = f(s) - f(s')$. If $\Delta f \leq 0$, then go to step 3d; otherwise go to step 3e.
- d Select a random variable $x \sim U(0,1)$. If

$$x < P(\Delta f) \equiv \exp(\Delta f / t_i),$$

  then go to step 3e; otherwise go to step 3a.
- e Accept the candidate sequence, set $s = s'$ and $f(s) = f(s')$, then test to see if $f(s) < f(s^*)$. If true, then $s^* = s$, $f(s^*) = f(s)$ and $I = i$. If we have accepted e sequences go to step 4; otherwise go to step 3 a.

Step 4 If equilibrium has not been reached at temperature t_i, i.e., if $|\overline{f}_e - \overline{f}'_e|/\overline{f}'_e \geq \epsilon$, then go to step 3 a; otherwise, set $i = i+1$ and $t_i = t_0(0.8)^{i-1}$. If $(i-I) < N$ go to step 5; otherwise **STOP** because the maximum succession of non- improving temperatures has been reached.

Step 5 If the total number of epochs is less than M then go to step 3a; otherwise, **STOP.**

4.2 Determination of the Initial Temperature

The selection of an initial temperature can have a strong influence on the quality of the solutions obtained with a simulated annealing algorithm. Furthermore, Johnson *et al* [11] observe that "it may not be necessary to spend much time at very high temperatures." They suggest that, when setting the initial temperature, the user should consider the probability that a non-improving candidate solution will be accepted. To avoid making many runs with different temperature settings, we opted for the following expression to set the initial temperature, t_0, in all the test problems:

$$t_0 = z_0(1/40), \tag{4}$$

where z_0 denotes the initial objective function value. Obviously, using an alternative initial temperature setting for a particular problem may outperform the above setting we suggest. However, the initial temperature setting given by (4) performed quite well on all the test problems presented in the following sections.

In our algorithm we attempted to choose an epoch length so that a high percentage of the total improvement was realized within the first few temperature settings, but also that the total improvement was achieved over several temperature settings. Therefore, based on trial and error, an epoch length of 30 accepted layouts was used throughout the test problems. It is interesting to note that longer epoch lengths did not necessarily outperform shorter epoch lengths. Another method for varying the performance of the algorithm is to run the algorithm longer. Since candidate solutions are generated in a *pseudo-random* manner, longer runtimes are likely (but not guaranteed) to improve the solution obtained from a short run.

5 Single Floor Problem Comparisons

First, we evaluate the performance of the three heuristics, CRAFT, MULTIPLE, and SABLE on three single-floor test problems. The three data sets have 11, 15 and 25 departments, respectively. (The data sets will be referenced according to the number of

Data Set	CRAFT		MULTIPLE		SABLE	
	$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ
11	1686.51	121.76	1526.90	124.45	1402.40	38.64
15	41925.96	4119.12	38373.91	2231.06	34746.95	1565.03
25	2218.60	179.46	1970.71	146.30	1776.86	80.23

Table 2: The average and standard deviations of the final layout cost for CRAFT, MULTIPLE, and SABLE.

departments in the test problem, e.g., data set 25 refers to the example problem with 25 departments. The complete data sets for all single-floor test problems are presented in Appendix A and include the minimum departmental area requirements, the flow matrix, the 10 initial layout sequences and associated costs, and the spacefilling curve which shows the location of any fixed departments.) Second, we compare the performance of SABLE against the algorithm presented in [21]. For this comparison we use the data sets shown in [21].

5.1 CRAFT, MULTIPLE and SABLE Comparisons

The example problem flow matrices were generated by a method much like that used in materials requirement planning (MRP). Production quantities, department routings and unit loads were specified for a number of products and then the flow matrix was generated in the same fashion as a MRP program would determine the number of subassemblies that were needed for production based on a final product production target. We used the following values for the control settings in parenthesis: 30 (e), 0.25 (ϵ_1), 0.05 ($\epsilon_i, i \geq 2$), 0.10 (β), 37 (M) and 5 (N). The initial temperature was specified according to (4). Each heuristic was run with 10 initial layouts for the three data sets. The results obtained for the three single-floor problem comparisons are presented in Table 2 where the average final cost (Avg) and the standard deviation of the final cost (Stnd) are reported for the three heuristics.

As shown in Table 2, SABLE achieves lower average cost solutions than both MULTIPLE and CRAFT for all the single-floor problems presented. To test the statistical significance of the final cost differences exhibited in Table 2, the Wilcoxon signed-rank test was used. (Golden and Stewart [9] suggest the Wilcoxon signed-rank test - a non-parametric test - when comparing empirical data obtained from heuristics.) The test results indicate that the difference observed between SABLE and MULTIPLE is significant ($\alpha = 0.05$) for each problem. Likewise, MULTIPLE produced solutions that were lower than CRAFT and this difference is statistically significant ($\alpha = 0.05$) for the three data sets. Furthermore, SABLE has sample standard deviations that are lower than the

Data Set	Best Soln	CRAFT Max	MULT Max	SABLE Max
11	1372.73	1859.47	1693.53	1453.90
15	32319.30	50618.20	42478.99	36843.50
25	1641.00	2560.43	2204.67	1883.80

Table 3: A worst-case analysis of the single-floor test problems based on minimum-cost solutions.

sample standard deviations of CRAFT and MULTIPLE for all problems.

The test problem with 11 departments was also solved "optimally." Since a layout is represented by a sequence of departments, total enumeration of all possible sequences will determine the "optimal" cost *for a given spacefilling curve.* SABLE terminated with the "optimal" cost for 6 of the 10 initial layouts while MULTIPLE and CRAFT equaled the "optimal" cost 1 and 0 times out of 10, respectively. (The "optimal" cost was equal to 1372.73 units. Thus, CRAFT, MULTIPLE and SABLE produce solutions that averaged 22.9%, 11.2% and 2.2% above "optimal," respectively.) For single floor problems, SABLE appears superior to MULTIPLE which appears superior to CRAFT if the algorithms are compared by the average final layout cost or the number of "optimal" layouts obtained.

The initial layout used by a layout algorithm is usually dictated by the current layout. That is, the analyst may not have the advantage of trying many initial layouts. Furthermore, a typical decision-maker does not solve the facility layout problem frequently enough to rely only on the "average performance" of an algorithm. Therefore, a worst-case analysis seems to be well-justified for the facility layout problem. An examination of the worst-case performance is presented in Table 3. For each data set, the highest cost solution (Max) obtained by each of the three heuristics (when ran with the 10 initial layouts) is compared to the overall best solution (Best Soln) obtained by the three heuristics. These three data sets show that SABLE outperforms both MULTIPLE and CRAFT in this worst-case analysis and that MULTIPLE outperforms CRAFT.

Another worst-case analysis was performed because of the initial layout bias. Table 4 includes the number of times (Prod Min) that an algorithm produced the lowest cost solution observed (Min) for each of the ten initial layouts. The percentage that the two other final solutions are above Min is computed for each initial layout, and the maximum of this measure (Max % > Min) is shown for the three algorithms and each data set in Table 4. Once again, SABLE appears to be a superior algorithm if judged by this criteria. SABLE produces the lowest cost layout in approximately 90% (26 out of 30) of the initial layouts and, in the worst-case, is no more than 10% above the lowest cost layout obtained by the other two algorithms for the same initial layout. It is also important to note that if this worst-case analysis is used as a criteria to compare CRAFT and MULTIPLE, the

Data Set	CRAFT		MULTIPLE		SABLE	
	Prod Min	Max % > Min	Prod Min	Max % > Min	Prod Min	Max % > Min
11	0	35.46	2	23.37	9	2.09
15	0	48.96	1	25.01	9	0.58
25	0	47.21	2	27.39	8	9.51

Table 4: A worst-case analysis of the single-floor test problems based on initial layouts.

Data Set	CRAFT			MULTIPLE			SABLE		
	$\overline{x}$	σ	Max	$\overline{x}$	σ	Max	$\overline{x}$	σ	Max
11	0.54	0.14	0.80	7.96	3.19	12.02	15.39	2.59	20.70
15	1.43	0.40	2.20	41.23	7.47	52.95	66.00	14.46	94.30
25	4.44	1.39	6.60	243.52	53.55	331.42	382.27	34.27	419.58

Table 5: The average and standard deviations of the runtimes (expressed in seconds) for the single-floor test problems.

solutions produced by MULTIPLE are better than the solutions produced by CRAFT for these data sets.

SABLE and MULTIPLE were implemented on a 25 MHz 386 DOS-based personal computer (PC) with an 80387 math coprocessor. Both algorithms are written in PASCAL. CRAFT was obtained from a mainframe computer program written in FORTRAN and adapted to the PC described above. The runtimes (RunT) and standard deviation of the runtimes (Stnd) are presented in Table 5 for the three algorithms. Rewriting CRAFT in PASCAL is highly undesirable and "risky" since the code is considerably long and has no internal documentation. Hence, some of the difference we observe in runtimes in Table 5 is due to compiler differences. Test programs we prepared to quantify this difference indicate that the FORTRAN compiler we used generates object codes that run approximately 2.25 times faster than those generated by the PASCAL compiler.

MULTIPLE is indeed a more powerful heuristic than CRAFT, obtaining solutions that are approximately 11% lower on the average. Furthermore, SABLE is an effective extension of MULTIPLE in single-floor problems, in which it averaged solutions that are 10% lower than those solutions obtained with MULTIPLE. Both MULTIPLE and SABLE have significantly longer runtimes than CRAFT. However, since the facility layout problem is a high-level planning problem which is not solved frequently, the absolute value of the runtimes is as important as the relative runtimes. The absolute runtimes reported for both MULTIPLE and SABLE are clearly acceptable; the longest runtime we observed for SABLE is less than 7 minutes on the above PC which no longer represents state-of-the-art hardware.

Data Set	Tam Value	SABLE AVG	SABLE RTime
20	11756.39	9850.70	157.54
30	23416.51	21900.98	351.36

Table 6: A comparison between the estimated layout cost achieved in [21] and the SABLE average layout cost. SABLE runtimes are given in seconds.

5.2 Comparison of Simulated-Annealing Based Algorithms

Since SABLE clearly outperforms CRAFT and MULTIPLE, in this section we will compare only SABLE with Tam's algorithm [21]. We will use the two data sets (taken from [21]) with 20 and 30 departments. Since the objective function values reported in [21] include the penalties incurred due to department shape constraint violations, we had to estimate the layout cost (without any penalties) from the layouts shown in [21]. Furthermore, in [21] only the minimum-cost layouts are presented; i.e., alternative layouts obtained from different initial layouts are not shown. Therefore, we only compared the solutions obtained from SABLE against the minimum-cost layouts obtained in [21]. Please note, however, that since the layouts obtained in [21] are based on an objective function that includes the shape-based penalties, the minimum-cost layout may no longer be the best layout when the penalties are removed. Therefore, the comparison remains an approximate one. Given the fundamental differences between SABLE and [21], we believe exact comparisons are not possible. Recall that SABLE tends to generate non-rectangular departments while in [21] departments are assumed to be rectangular in general.

For each data set, 10 initial layouts were randomly generated and used as input for SABLE. The estimated final layout cost for [21], the SABLE average final layout cost for the 10 initial layouts, and the average runtimes (for SABLE) on the PC described earlier over the two data sets, are shown in Table 6. It is worth noting that, for each data set, all of the ten SABLE final layout costs were lower than those obtained from [21]. We must emphasize that accepting non-rectangular department shapes works in favor of SABLE. The results indicate that SABLE is able to fully capitalize on this difference.

6 Multiple Floor Problem Comparisons

SABLE and MULTIPLE were compared via numerous multi-floor example problems. Two 11-department, two-floor problems and one 12-department, three-floor problem were solved to represent small problems. In addition, four different 21-department, four-floor problems and one 40-department, four-floor problem were also used to examine the per-

Data Set	MULTIPLE		SABLE	
	$\bar{x}$	σ	$\bar{x}$	σ
11-1	16702.07	3026.15	8958.67	2107.60
11-2	2910.67	267.92	2558.09	50.59
12	2153.09	223.97	1640.34	129.51
21-1	18553.50	2152.90	16310.40	847.63
21-2	14410.57	1792.44	12745.74	663.05
21-3	11787.28	775.84	10788.89	352.33
21-4	11252.08	1125.08	9636.51	877.04
40	23348.29	2355.15	21622.69	423.17

Table 7: The average and standard deviations of final layout cost for MULTIPLE and SABLE.

formance of SABLE on medium and large problems. (All multi-floor problem data is presented in Appendix B.) The example problems will be referenced by data set numbers which correspond to the number of departments and an index to differentiate the two 11-department problems and the four 21-department problems, e.g., the second 21-department problem will be referenced as data set 21-2. All data sets have slack area of 7 - 10% on each floor (with the exception of the 11- and 12-department data sets, which have no slack to facilitate obtaining the "optimal" solution) because in practice there is often slack area in a facility.

SABLE was run on these data sets using the same control settings as reported in §5 with one exception. In data set 40, the objective function tended to "cool" very slowly due to the large number of departments and the relatively high value of β, which determines how much the department sequence will change when generating a candidate representation. In response to this slow cooling rate, we decreased β by 3% with each temperature reduction.

The results for the eight multi-floor problem comparisons are presented in Table 7 where the average (Avg) and standard deviation (Stnd) of the final costs achieved with 10 initial layouts are reported for the two heuristics. SABLE achieves lower average cost solutions than MULTIPLE for each multi-floor data set. To test the statistical significance of the final cost differences exhibited in Table 7, the Wilcoxon signed-rank test was again used. Test results indicate that the above differences are statistically significant ($\alpha = 0.02$). Furthermore, SABLE had sample standard deviations that were much lower than those produced with MULTIPLE, which indicates the SABLE final solutions are less biased by the initial layout than MULTIPLE.

The three test problems with 11 and 12 departments were solved "optimally" (for spacefilling curves generated by the authors). An examination of the final costs indicate that for the three data sets, SABLE terminated with the "optimal" layout 7, 1 and 3 times out of 10, while MULTIPLE terminated with the "optimal" cost 0, 1 and 0 times

Data Set	MULTIPLE		SABLE		Data Set	MULTIPLE		SABLE	
	$\bar{x}$	σ	$\bar{x}$	σ		$\bar{x}$	σ	$\bar{x}$	σ
21-1 H	38148.60	5744.11	30403.60	2449.06	21-1 L	12301.75	804.06	12147.35	415.65
21-2 H	29001.60	8494.35	21702.53	1043.80	21-2 L	11129.10	771.01	10280.80	469.10
21-3 H	27710.40	2958.58	24403.70	898.46	21-3 L	7252.45	225.79	6998.80	149.89
21-4 H	28925.77	4415.34	22354.03	1115.90	21-4 L	6104.38	367.85	5775.62	188.76

Table 8: The average and standard deviations of final layout cost for MULTIPLE and SABLE with the high and low V/H Ratio data sets.

out of 10. (The "optimal" cost for the three data sets are 8275.70, 2493.92 and 1513.15 units, respectively. Thus, averaging over the three data sets, MULTIPLE and SABLE produce solutions that are 53.6% and 6.4% above "optimal," respectively.) Therefore, SABLE appears superior to MULTIPLE for multi-floor problems if the algorithms are compared by the average final layout cost or the number of "optimal" layouts obtained.

SABLE was primarily developed to improve the performance of MULTIPLE in multi-floor layout problems where department exchanges across floors plays an important role. We expect SABLE to generally outperform MULTIPLE because SABLE is based on a more generalized and powerful "exchange routine" driven by the candidate solution concept in simulated annealing. As such, SABLE has more potential to reduce vertical handling costs. The relative unit cost of horizontal and vertical travel is likely to influence the performance differences between SABLE and MULTIPLE. In the above 21-department data sets, the ratio of the vertical cost per distance unit to the horizontal cost per distance unit (henceforth referred to as the V/H Ratio) is 5 to 1. To explore the effect that the V/H Ratio might have on the relative performance of the two algorithms, it was first increased from 5 to 20, and later decreased from 5 to 1, for the four 21-department data sets, yielding eight new data sets.

The results of the two algorithms on the four new data sets with the high V/H Ratio (H) are shown in Table 8. The difference between the average algorithm performance is significant ($\alpha = 0.005$) for the four data sets. The corresponding results for the low V/H Ratio (L) are also shown in Table 8. The differences in Table 8 are significant ($\alpha = 0.005$) for data sets 21-2 L and 21-3 L, but are not significant for the other two data sets. In addition, the percentage difference between the averages of the two algorithms is substantially higher for the data sets with high V/H Ratios. Therefore, it appears that the relative performance of SABLE over MULTIPLE improves as the V/H Ratio increases. This is perhaps due to MULTIPLE's limited two-way exchange routine and its inability to vary the number of departments on each floor. These eight additional data sets will be included in the following worst-case performance analyses of the two algorithms.

A worst-case performance analysis was performed to compare the worst cost solution

Data Set	Best Soln	MULT Max	SABLE Max
11-1	8275.70	21982.17	14956.30
11-2	2493.92	3332.41	2662.70
12	1513.15	2538.05	1842.50
21-1	14970.00	21408.00	17774.40
21-2	11854.70	17262.33	14230.70
21-3	10263.50	13072.00	11273.70
21-4	8633.00	13062.33	10828.00
21-1 H	27298.00	49888.00	35014.00
21-2 H	19155.33	47317.33	22573.33
21-3 H	22899.00	33536.00	25788.33
21-4 H	20524.67	36467.33	24841.00
21-1 L	11241.00	13995.50	12788.50
21-2 L	9765.00	12345.33	11278.00
21-3 L	6778.83	7512.00	7276.00
21-4 L	5466.33	6786.00	6012.00
40	20441.46	27730.50	22072.50

Table 9: A worst-case analysis for the multi-floor test problems comparing the maximum and best cost solutions.

obtained with each algorithm and the best solution obtained with either algorithm. For each data set, the maximum solution obtained over ten initial layouts by each of the heuristics (Max) is compared to the best solution (Best Soln) obtained by either of the two heuristics. The results are presented in Table 9, where the sixteen data sets show that SABLE outperforms MULTIPLE by this worst-case performance measure.

Another worst-case analysis was performed because of the initial layout bias. Table 10 includes the number of times (Prod Min) that an algorithm produced the lower of the two algorithm final costs (Min) for each of the ten initial layouts. The percentage that the other algorithm's final solution is above Min is computed for each initial layout, and the maximum of this measure (Max % > Min) is shown for the two algorithms and each data set in Table 10. (Note that this statistic is 0.00 by definition when SABLE outperforms MULTIPLE for all 10 initial layouts of a data set.) Once again, SABLE appears to be a superior algorithm if judged by this criteria. SABLE produces the lowest cost layout in approximately 90% (142 out of 160) of the initial layouts and, in the worst case, the SABLE layout cost was never more than 9.4% above the MULTIPLE layout cost for the same initial layout. (This statistic decreases to 5.4% if the low V/H Ratio data sets are not included.)

SABLE and MULTIPLE were implemented on the PC described above in §5. The runtimes (RunT) and standard deviation of the runtimes (Stnd) are presented in Table 11 for the two algorithms. The SABLE runtimes are approximately two to four times as long as the MULTIPLE runtimes for small problems, two to three times as long for medium

Data Set	MULTIPLE		SABLE	
	Prod Min	Max % > Min	Prod Min	Max % > Min
11-1	0	165.62	10	0.00
11-2	1	30.41	9	2.67
12	0	67.72	10	0.00
21-1	2	43.01	8	2.19
21-2	1	35.34	9	5.38
21-3	1	24.94	9	1.62
21-4	0	32.44	10	0.00
21-1 H	1	46.34	9	0.49
21-2 H	2	117.92	8	2.15
21-3 H	1	35.69	9	3.20
21-4 H	0	66.48	10	0.00
21-1 L	4	13.18	6	9.40
21-2 L	1	18.57	9	2.19
21-3 L	1	7.93	9	1.97
21-4 L	1	20.69	9	7.32
40	2	25.63	8	2.64

Table 10: A worst-case analysis for the multi-floor test problems based on initial layouts.

problems, and 30% less for the 40-department problems. Regardless of their relative difference, the magnitude of the SABLE and MULTIPLE runtimes are not unreasonable when considering the scope of a facility layout problem.

With low, medium and high V/H Ratios, SABLE averaged solutions that are 5%, 13%, and 25% lower, respectively, than the solutions obtained with MULTIPLE for the four 21-department problems, and 6% lower for the 40-department problem. For the small problems, the solution differences ranged from 16% to 84%, which is quite a dramatic range, but can be attributed, in part, to the limited number of feasible exchanges possible at any iteration. In general, SABLE improves upon MULTIPLE, which may not perform well if the floor space on each floor is very "tight" and little or no flexibility in departmental areas are provided. MULTIPLE also does not perform well when the V/H Ratio is high, whereas SABLE performs well if the V/H Ratio is low and very well for higher V/H ratios. In addition, SABLE also performs better than MULTIPLE if evaluated by worst-case performance criteria. Moreover, SABLE clearly reduces the bias due to the path dependency while maintaining runtimes that are reasonable in an absolute sense.

7 Conclusions

The layout algorithm presented in this paper represents a successful integration of simulated annealing and spacefilling curves (introduced in [2]). The non-greedy nature of simulated annealing and the technique we used for generating candidate solutions, com-

Data Set	MULTIPLE			SABLE		
	$\overline{x}$	σ	Max	$\overline{x}$	σ	Max
11-1	8.05	1.76	11.37	17.78	3.70	21.31
11-2	3.40	1.03	5.55	14.26	2.07	16.54
12	4.94	1.66	7.20	20.37	3.01	26.63
21-1	146.86	21.34	192.79	283.52	65.76	372.74
21-2	142.88	31.14	173.13	249.57	58.92	369.32
21-3	126.54	13.99	150.72	277.13	41.85	310.72
21-4	110.00	21.07	152.47	191.48	43.34	238.93
21-1 H	116.73	20.70	142.80	283.99	42.20	364.26
21-2 H	105.81	25.66	148.85	208.17	33.44	253.32
21-3 H	122.03	33.57	198.12	219.85	56.08	289.34
21-4 H	101.75	24.02	135.61	184.81	29.51	235.03
21-1 L	104.08	22.82	132.48	309.06	53.82	371.96
21-2 L	95.35	25.37	148.35	247.88	43.64	292.48
21-3 L	92.11	21.02	125.40	314.46	29.45	362.61
21-4 L	87.21	14.85	117.65	236.70	51.28	293.58
40	3236.93	874.10	4901.03	2174.84	241.12	2594.25

Table 11: The average and standard deviations of the runtimes (expressed in seconds) for the multi-floor test problems.

bined with the speed and flexibility offered by spacefilling curves, resulted in an algorithm that outperforms "comparable" algorithms developed for single and multiple floor facility layout problems. In the process, we were also able to determine the "optimal" layout for small, single and multi-floor layout problems. Although we have not performed extensive numerical tests on single-floor problems, our empirical results indicate that the simulated-annealing based layout evaluation algorithm, SABLE, performs better than CRAFT and MULTIPLE, both in an average sense as well as a worst-case sense. SABLE is fairly robust to changes in the vertical to horizontal cost ratio, and it seems to considerably reduce the bias induced by the initial layout. Also, for small test problems, we observed that SABLE generated near-optimal solutions.

As far as the simulated annealing scheme is concerned, we used the same control settings for each problem, except for the initial temperature setting which is based on the cost of the initial layout. Such initial temperature settings generally perform better than arbitrary settings. Obviously, for a particular problem, one may find an annealing schedule that outperforms ours. However, for all of the 210 problems we tested in this study, we found that the annealing schedule we present here consistently performs well.

Appendix A:
Data Sets for the Single-Floor Test Problems

A1: Data Set 11

The minimum departmental area (Area) requirements for the 11 departments are given in Table A1. The flow matrix given in Table A2 was generated by the method described in [2]. The data is included in Table A3 where multiplier 1 (Mult1) is defined to be the number of subassemblies included in one final product, and the number in parentheses after any department defines the unit load size that is moved between that department and the next. Product 1 is made of one main assembly and two subassemblies. Since Product 1 is assumed to have a production quantity of 100 units, and the second subassembly of product 1 has a multiplier 1 value of 2 and is moved in unit load sizes of 20 between department 6 and 3, therefore the flow matrix entry from 6 to 3 is $(100 * 2)/20$ which equals 10. All cost data is assumed to be \$1.00 per unit distance, per unit time. The spacefilling curve is shown in Figure 2 and note that since department 11 is fixed, the spacefilling curve does not pass through any of the grids assigned to department 11. Finally, Table A4 contains the 10 initial sequences of departments for the single-floor test problem and their associated costs (in dollars). The optimal solution to this test problem (\$1372.73) is characterized by the following sequence: 6-9-10-1-5-7-2-3-4-8.

Dept	1	2	3	4	5	6	7	8	9	10	11
Area	3	2	4	5	2	3	4	5	1	1	6

Table A1: The departmental area requirements for the single-floor test problem with 11 departments.

From/To	1	2	3	4	5	6	7	8	9	10	11
1	0	10	0	0	140	90	20	0	40	0	0
2	0	0	10	0	0	0	0	0	0	0	0
3	0	0	0	10	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	4
5	0	10	0	0	0	0	40	0	0	20	0
6	0	0	10	0	0	0	0	0	20	0	0
7	0	0	0	0	0	0	0	10	0	0	0
8	0	0	0	0	0	0	0	0	0	0	10
9	0	0	0	0	0	0	0	0	0	20	0
10	0	0	0	0	0	0	0	0	0	0	20
11	146	0	0	0	0	0	0	0	0	0	0

Table A2: The flow matrix for data set 11 of the single-floor test problem.

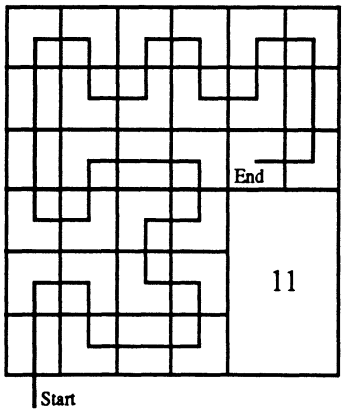

Figure 2: Spacefilling Curve and Fixed Department Location for Test Problem 11.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Production
1	1	11(25)	1(10)	2(10)	3(10)	4(25)	11	100
1	2	11(20)	1(10)	5(20)	2			
1	2	11(25)	1(20)	6(20)	3			
2	1	11(25)	1(10)	7(20)	8(20)	11		200
2	2	11(10)	1(5)	5(10)	7			
3	1	11(25)	1(10)	9(20)	10(20)	11		400
3	1	11(10)	1(5)	6(20)	9			
3	1	11(20)	1(10)	5(20)	10			

Table A3: The flow matrix generator for data set 11 of the single-floor test problem.

Init	Sequence	Cost	Init	Sequence	Cost
1	9 8 6 7 4 5 1 3 10 2	2517.67	6	4 3 8 1 10 2 7 6 9 5	3310.87
2	10 9 6 8 3 2 5 4 1 7	2743.87	7	10 2 7 4 1 9 3 5 6 8	2641.53
3	4 7 8 2 3 10 5 1 6 9	1874.87	8	6 9 10 1 7 5 3 2 4 8	1647.40
4	9 2 7 8 1 10 3 5 6 4	2693.40	9	3 2 6 9 8 4 5 7 10 1	2115.87
5	1 8 10 6 9 4 7 3 5 2	3142.67	10	7 3 8 5 1 9 4 10 2 6	2893.47

Table A4: The sequences to generate the initial layouts for data set 11 of the single-floor test problem and their associated costs.

A2: Data Set 15

The minimum departmental area (Area) requirements for the 15 departments are given in Table A5. The flow matrix generator is included in Table A6. All cost data is assumed to be \$1.00 per unit distance, per unit time except for flows to and from department 15, which are \$0.50

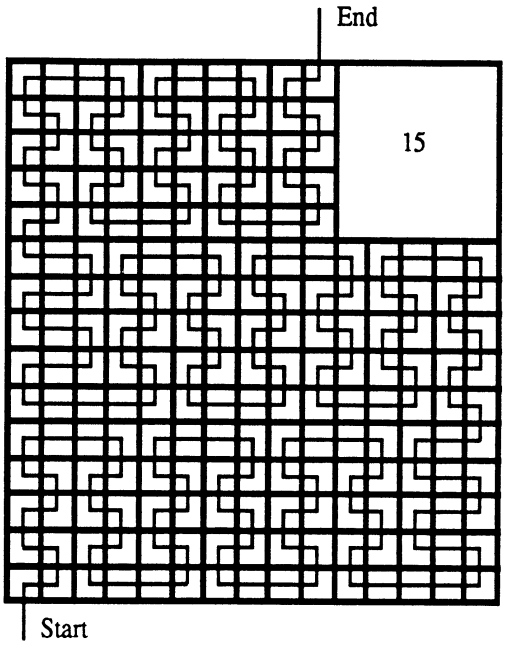

Figure 3: Spacefilling Curve and Fixed Department Location for Test Problem 15.

per unit distance, per unit time. The spacefilling curve is shown in Figure 3 as well as the fixed location of department 15. Finally, Table A7 contains the 10 initial sequences of departments for the single-floor test problem and their associated costs (in dollars). The best known solution to this test problem ($32,319.30) is characterized by the following sequence: 13-7-8-6-1-2-14-5-4-11-3-9-10-12.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Area	8	4	5	3	6	2	3	5	7	9	5	1	2	4	8

Table A5: The departmental area requirements for the single-floor test problem with 15 departments.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Production
1	1	15(10)	2(20)	3(5)	4(5)	5(10)	15	100
1	2	15(25)	6(50)	7(10)	4			
2	1	15(50)	2(50)	8(10)	9(10)	5(100)	15	200
2	1	15(25)	10(10)	11(5)	9			
2	2	15(100)	12(25)	13(50)	11			
3	1	15(10)	2(20)	14(20)	1(10)	5(10)	15	400

Table A6: The flow matrix generator for data set 15 of the single-floor test problem.

Init	Sequence	Cost	Init	Sequence	Cost
1	1 2 3 4 5 6 7 8 9 10 11 12 13 14	61422.98	6	10 8 5 1 4 6 3 7 12 13 9 14 2 11	73198.39
2	5 11 9 4 10 1 14 12 3 6 2 13 7 8	52247.90	7	9 13 14 8 5 7 2 12 10 1 4 6 11 3	71859.61
3	10 9 3 12 6 4 5 8 11 1 14 2 13 7	80030.61	8	10 7 12 9 6 11 13 5 4 2 14 3 8 1	69378.53
4	9 13 1 8 14 5 6 2 11 3 10 7 12 4	74387.68	9	7 10 8 6 14 5 13 1 2 4 12 11 9 3	79264.57
5	3 9 12 4 6 5 7 8 1 13 10 11 2 14	80837.69	10	10 3 4 13 8 12 6 5 1 7 14 11 9 2	73053.77

Table A7: The sequences to generate the initial layouts for data set 15 of the single-floor test problem and their associated costs.

A3: Data Set 25

The minimum departmental area (Area) requirements for the 25 departments are given in Table A8. The flow matrix generator is included in Table A9. All cost data is assumed to be \$1.00 per unit distance, per unit time. The spacefilling curve is shown in Figure 4 as well as the fixed location of department 25. Finally, Table A10 contains the 10 initial sequences of departments for the single-floor test problem and their associated costs (in dollars). The best known solution to this test problem (\$1641.00) is characterized by the following sequence: 7-13-23-1-4-8-14-5-3-18-24-15-20-17-22-2-12-11-16-10-21-19-9-6.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Area	2	1	5	3	1	3	5	3	2	1	3	2	4	5	2
Dept	16	17	18	19	20	21	22	23	24	25					
Area	4	1	5	3	4	1	4	5	2	4					

Table A8: The departmental area requirements for the single-floor test problem with 25 departments.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Production
1	1	25(25)	1(15)	4(5)	7(10)	13(20)	25	100
1	1	25(20)	2(40)	17(20)	23(10)	13		
1	1	25(25)	3(20)	5(10)	23			
2	1	25(50)	8(10)	24(20)	21(20)	10(10)	25	200
2	2	25(25)	6(10)	9(10)	19(20)	10		
2	1	25(25)	16(10)	11(25)	19			
3	1	25(25)	12(10)	17(10)	22(25)	25		400
3	1	25(50)	14(10)	18(20)	22			
3	2	25(100)	15(10)	20(20)	22			

Table A9: The flow matrix generator for data set 25 of the single-floor test problem.

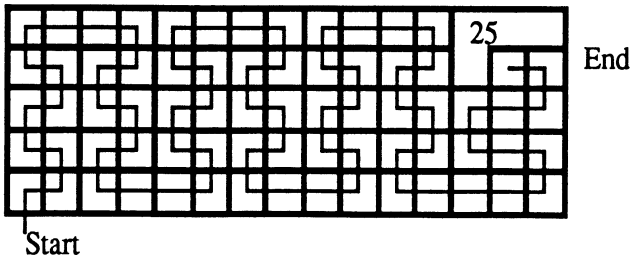

Figure 4: Spacefilling Curve and Fixed Department Location for Test Problem 25.

Init	Sequence	Cost
1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	3789.23
2	16 15 7 14 3 2 23 19 22 10 20 12 6 9 21 8 11 24 17 8 4 1 5 13	4265.50
3	2 18 8 23 5 4 17 3 14 21 13 19 16 9 15 7 1 12 22 20 10 11 6 24	4148.33
4	17 6 7 23 5 24 18 1 22 3 10 21 13 14 15 4 9 12 19 8 11 2 16 20	4369.53
5	5 6 12 7 15 17 9 23 19 8 14 16 13 22 4 3 18 20 10 2 21 24 1 11	4178.20
6	6 20 2 16 1 22 4 23 21 15 19 14 12 7 8 17 5 24 10 3 9 11 13 18	4245.40
7	4 20 19 16 3 9 1 10 21 6 15 14 22 2 17 13 5 18 23 7 24 8 11 12	3832.80
8	17 21 16 6 15 23 10 20 8 11 7 19 12 3 18 13 2 4 14 24 5 22 9 1	4919.37
9	2 21 1 12 15 19 9 23 4 24 5 7 8 16 6 14 17 22 18 20 10 11 13 3	4203.07
10	4 1 11 7 15 20 22 24 14 18 6 13 17 10 16 9 8 23 12 5 2 3 21 19	3171.77

Table A10: The sequences to generate the initial layouts for data set 25 of the single-floor test problem and their associated costs.

Appendix B:
Data Sets for the Multi-Floor Test Problems

B1: Data Set 11-1

The minimum departmental area (Area) requirements are equivalent to data set 11 of the single-floor test problem (refer to Table A1 in Appendix A). The flow matrix is also equivalent to the flow matrix used by data set 11 of the single-floor test problem (refer to Table A2 in Appendix A). All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 2.5 square distance units and the interfloor distance is 10.0 distance units. The spacefilling curve is shown in Figure 5 and note that since department 11 is fixed, the spacefilling curve does not pass through any of the grids assigned to department 11. Figure 5 also indicates the location of the two lift locations. Finally, Table B1 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The optimal solution to this test problem (\$8275.70) is characterized by the following sequence: 9-6-2-5-1-10 / 3-4-8-7.

Init	Sequence	Cost	Init	Sequence	Cost
1	3 7 2 9 10 6 8 5 4 1	23003.50	6	3 4 6 2 7 10 1 9 5 8	19910.00
2	7 2 9 8 3 4 5 10 6 1	22014.00	7	4 1 3 7 2 10 9 6 8 5	22783.00
3	1 3 8 5 4 10 9 7 6 2	24725.17	8	7 1 9 5 2 10 3 4 8 6	15712.83
4	5 4 8 9 6 1 7 2 10 3	26038.83	9	4 9 3 5 2 6 10 7 8 1	30603.00
5	4 10 7 5 3 1 6 9 8 2	24543.00	10	3 1 2 9 5 4 8 10 7 6	19613.00

Table B1: The sequences to generate the initial layouts for data set 11 of the single-floor test problem and the associated initial layout costs.

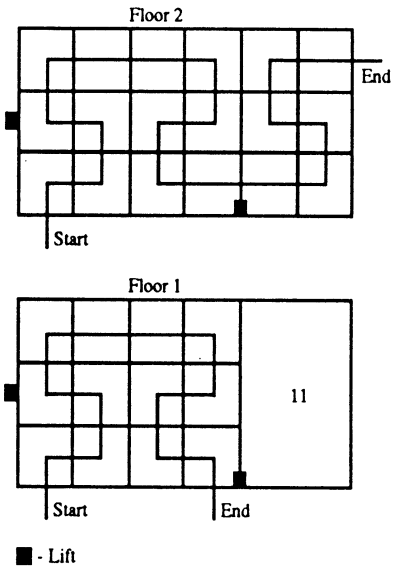

Figure 5: Spacefilling Curve and Fixed Department Location for Test Problem 11.

B2: Data Set 11-2

The minimum departmental area (Area) requirements are given in Table B2. The from-to flow values are included in Table B3. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 1.0 square distance units and the interfloor distance is 1.0 distance units. The spacefilling curve is shown in Figure 6 and the figure also shows the fixed location of department 11. Figure 6 also indicates the location of the two lift locations. Finally, Table B4 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The optimal solution to this test problem (\$2493.92) is characterized by the following sequence: 9-7-3 / 4-8-1-10-6-5-2.

Dept	1	2	3	4	5	6	7	8	9	10	11
Area	1	3	2	2	1	2	1	1	5	2	4

Table B2: The departmental area requirements for the second multi-floor test problem with 11 departments.

From	To	Flow	From	To	Flow
1	4	30.5	3	11	32.6
1	6	4.6	4	5	6.3
1	7	14.0	4	10	34.7
1	9	28.5	5	6	15.4
2	4	34.1	5	8	33.1
2	5	33.1	5	11	42.1
2	6	26.6	6	8	4.4
2	8	30.7	6	9	11.8
2	9	34.0	6	10	21.0
3	5	6.2	7	8	36.6
3	6	9.2	7	9	43.1
3	7	49.1	8	10	20.4
3	8	11.4	9	10	16.3

Table B3: The from-to flow values for data set 11-2 of the multi-floor test problem.

Init	Sequence	Cost	Init	Sequence	Cost
1	5 9 8 7 6 1 3 10 2 4	3793.58	6	10 7 9 2 3 1 4 6 5 8	3671.00
2	5 9 3 8 2 1 6 4 7 10	3386.15	7	4 5 6 3 1 9 2 8 10 7	3603.39
3	5 4 2 3 7 1 8 10 9 6	24725.17	8	9 6 5 3 2 10 8 1 7 4	3519.16
4	5 3 10 6 7 9 2 1 8 4	3527.59	9	6 5 8 3 4 9 7 1 2 10	3677.56
5	2 1 4 3 9 7 8 5 10 6	3527.62	10	1 4 3 5 10 7 6 2 9 8	3547.66

Table B4: The sequences to generate the initial layouts for data set 11-2 of the multi-floor test problem and their associated costs.

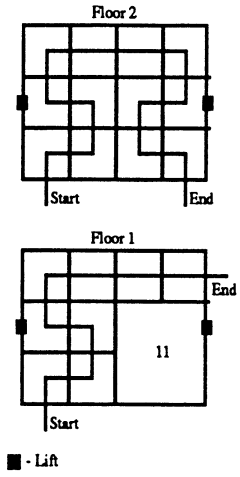

Figure 6: Spacefilling Curve and Fixed Department Location for Test Problem 11-2.

B3: Data Set 12

The minimum departmental area (Area) requirements are given in Table B5. The from-to flow values are included in Table B6. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 1.0 square distance units and the interfloor distance is 1.0 distance units. The spacefilling curve is shown in Figure 7 and the figure also shows the fixed location of department 12. Figure 7 also indicates the location of the two lift locations. Finally, Table B7 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The optimal solution to this test problem (\$1513.15) is characterized by the following sequence: 7-6-10-1 / 4-5-8-2 / 11-9-3.

Dept	1	2	3	4	5	6	7	8	9	10	11	12
Area	1	1	2	4	2	1	1	1	2	1	4	4

Table B5: The departmental area requirements for the multi-floor test problem with 12 departments.

From	To	Flow	From	To	Flow
1	2	10.0	10	11	12.9
1	4	15.3	4	5	39.6
1	7	25.4	4	6	6.7
1	8	18.0	5	7	15.5
1	10	31.6	5	8	45.5
1	12	19.9	5	11	3.4
2	3	17.1	7	10	5.9
2	8	31.3	7	11	4.9
3	5	26.3	8	11	5.6
3	9	14.0	9	11	42.4
3	12	16.0	10	12	43.8

Table B6: The from-to flow values for data set 12 of the multi-floor test problem.

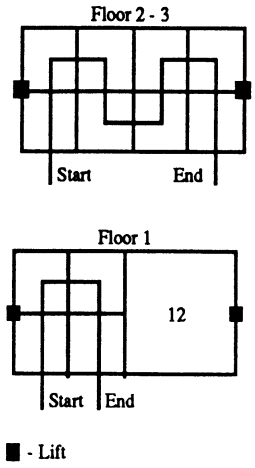

Figure 7: Spacefilling Curve and Fixed Department Location for Test Problem 12.

Init	Sequence	Cost	Init	Sequence	Cost
1	11 2 3 7 6 10 9 4 1 5 8	2898.95	6	3 5 6 4 9 7 10 1 11 8 2	3412.95
2	10 1 3 4 2 7 9 11 6 5 8	2817.95	7	4 1 3 11 7 8 10 5 9 2 6	3337.90
3	5 2 10 6 8 4 1 7 11 9 3	2661.25	8	5 10 8 4 3 1 6 7 11 2 9	2852.50
4	11 6 7 1 3 9 10 8 2 5 4	2339.55	9	6 3 7 2 9 11 10 5 4 8 1	2839.50
5	11 8 1 9 4 10 2 7 6 3 5	3162.05	10	11 1 9 6 4 3 7 8 5 10 2	2863.85

Table B7: The sequences to generate the initial layouts for data set 12 of the multi-floor test problem and their associated costs.

B4: Data Set 21-1

The minimum departmental area (Area) requirements are given in Table B8. The flow matrix generator is included in Table B9. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 4.0 square distance units and the interfloor distance is 2.5 distance units. The spacefilling curve is shown in Figure 8 and the figure also shows the fixed location of department 21. Figure 8 also indicates the location of the two lift locations. Finally, Table B10 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The best known solution to this test problem (\$14970.00) is characterized by the following sequence: 15-16-14 / 18-17-20-19-9-13 / 11-8-5-3-4-1-2 / 7-6-12-10.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Area	2	1	2	2	4	2	4	2	2	2	1	2	2	1	1	2	2	4	2	2	8

Table B8: The departmental area requirements for the first multi-floor test problem with 21 departments.

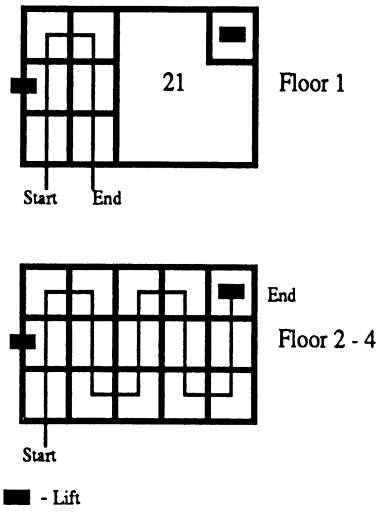

Figure 8: Spacefilling Curve and Fixed Department Location for Test Problem 21-1.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Dept 7	Production
1	1	21(20)	1(20)	2(10)	3(1)	4(10)	10(50)	21	100
1	1	21(20)	1(10)	11(10)	8(5)	4			
1	2	21(20)	1(10)	5(5)	8				
2	1	21(50)	12(10)	6(10)	7(5)	10(50)	21		200
2	1	21(20)	13(20)	9(10)	17(10)	18(10)	7		
2	1	21(20)	19(10)	20(10)	17				
3	1	21(50)	14(5)	16(5)	15(50)	21			400
3	1	21(20)	13(10)	9(5)	17(5)	18(10)	16		
3	2	21(20)	1(10)	5(5)	8(10)	17			

Table B9: The flow matrix generator for data set 21-1 of the multi-floor test problem.

Init	Sequence	Cost
1	16 11 2 20 13 10 6 12 9 3 19 5 17 18 14 1 7 8 4 15	38674.00
2	3 4 15 12 9 13 7 1 8 6 18 20 14 17 11 16 2 10 19 5	35626.00
3	16 11 15 7 6 10 20 13 3 17 12 4 18 14 8 1 9 2 5 19	32985.00
4	13 3 5 4 1 7 14 9 20 2 18 12 15 19 17 6 10 8 11 16	36451.50
5	8 10 16 6 4 14 15 12 2 5 19 3 9 7 18 13 1 11 20 17	44036.00
6	15 11 6 10 7 1 5 8 13 17 16 20 4 9 12 2 19 18 14 3	30190.50
7	8 16 3 9 1 17 6 20 13 19 5 18 2 14 7 12 15 11 4 10	40462.50
8	20 7 11 16 12 5 9 19 1 6 10 17 2 15 13 4 14 3 8 18	45170.00
9	1 7 12 8 17 14 6 2 4 16 11 10 15 3 13 19 18 20 9 5	40652.50
10	8 6 15 5 20 10 2 14 7 9 16 19 3 18 4 17 1 13 11 12	38275.00

Table B10: The sequences to generate the initial layouts for data set 21-1 of the multi-floor test problem and their associated costs.

B5: Data Set 21-2

The minimum departmental area (Area) requirements are given in Table B11. The flow matrix generator is included in Table B12. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 4.0 square distance units and the interfloor distance is 2.5 distance units. The spacefilling curve is shown in Figure 9 and the figure also shows the fixed location of department 21. Figure 9 also indicates the location of the two lift locations. Finally, Table B13 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The best known solution to this test problem (\$11854.70) is characterized by the following sequence: 18-20-17 / 10-13-8-1-2-14-12-16 / 3-5-19-7-15 / 4-9-11-6.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Area	1	2	3	4	2	3	4	1	2	3	4	1	2	3	2	1	2	3	4	1	8

Table B11: The departmental area requirements for the second multi-floor test problem with 21 departments.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Dept 7	Production
1	1	21(100)	15(10)	4(25)	12(10)	18(10)	20(10)	21	100
1	1	21(100)	15(25)	7(10)	19(25)	18			
1	2	21(100)	3(10)	5(10)	7				
2	1	21(100)	15(25)	6(10)	9(10)	11(10)	20(10)	21	200
2	1	21(100)	15(10)	7(1)	19(10)	11			
2	2	21(100)	3(10)	5(1)	19				
3	1	21(100)	1(25)	8(10)	10(20)	13(20)	20(10)	21	200
3	2	21(100)	3(20)	5(25)	10				
4	1	21(100)	16(10)	14(20)	2(20)	18(10)	17(10)	21	400

Table B12: The flow matrix generator for data set 21-2 of the multi-floor test problem.

Init	Sequence	Cost
1	10 14 20 1 4 7 15 19 8 17 3 12 18 16 6 5 2 11 9 13	40201.33
2	7 1 19 3 10 16 17 9 18 2 4 6 15 13 14 20 8 12 11 5	47002.33
3	2 20 17 4 9 16 11 3 8 13 19 1 7 6 18 14 5 10 15 12	33261.33
4	18 15 5 12 17 4 20 9 8 7 13 3 16 6 19 2 10 14 1 11	43972.00
5	6 2 4 14 10 15 5 11 1 16 3 20 7 8 19 9 18 12 17 13	43564.00
6	19 13 8 6 9 16 15 7 8 14 17 12 10 20 3 4 1 2 5 11	46872.33
7	16 10 1 4 13 11 3 9 8 6 7 12 18 20 17 5 19 15 2 14	29004.33
8	20 13 5 9 8 18 19 1 6 10 16 7 3 17 4 11 2 14 12 15	38337.33
9	17 13 5 15 11 20 16 1 17 12 9 14 4 6 18 10 19 8 3 2	51863.0
10	17 9 6 20 19 1 18 4 7 3 5 2 8 14 10 11 15 13 16 12	33054.67

Table B13: The sequences to generate the initial layouts for data set 21-2 of the multi-floor test problem and their associated costs.

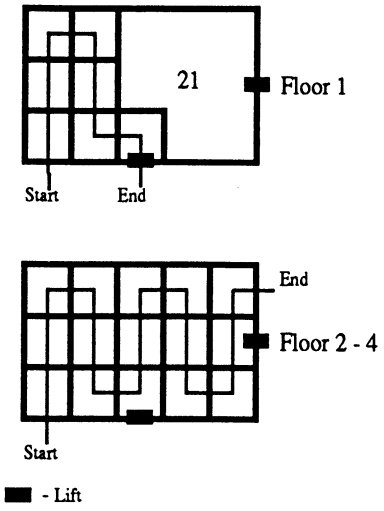

Figure 9: Spacefilling Curve and Fixed Department Location for Test Problem 21-2.

B6: Data Set 21-3

The minimum departmental (Area) area requirements are given in Table B14. The flow matrix generator is included in Table B15. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 4.0 square distance units and the interfloor distance is 2.5 distance units. The spacefilling curve used for data set 21-2 is also used for data set 21-3 and Figure 10 exhibits the new lift locations. Finally, Table B16 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The best known solution to this test problem (\$10263.50) is characterized by the following sequence: 10-3-4-13 / 18-19-12-14-1 / 5-6-16-20-17-11 / 15-9-7-8-2.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Area	2	1	1	1	3	2	3	4	3	2	2	3	2	2	3	2	2	4	2	2	8

Table B14: The departmental area requirements for the third multi-floor test problem with 21 departments.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Dept 7	Dept 8	Production
1	1	21(20)	1(10)	2(10)	8(20)	15(20)	10(20)	21		100
1	1	21(10)	4(10)	7(10)	9(5)	15				
1	2	21(20)	13(20)	18(5)	9					
2	1	21(50)	1(10)	17(20)	16(25)	20(25)	10(20)	21		200
2	1	21(20)	3(20)	5(25)	6(10)	20				
2	1	21(20)	13(20)	18(5)	6					
3	1	21(25)	1(25)	11(25)	12(5)	14(10)	18(20)	10(25)	21	400
3	1	21(50)	1(25)	19(20)	12					
3	2	21(25)	13(20)	18(5)	19					

Table B15: The flow matrix generator for data set 21-3 of the multi-floor test problem.

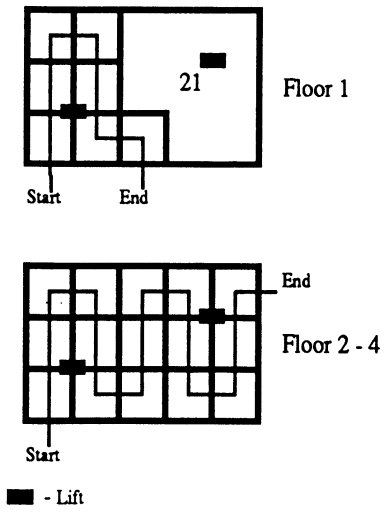

Figure 10: Spacefilling Curve and Fixed Department Location for Test Problem 21-3.

Init	Sequence	Cost
1	5 9 1 14 20 11 16 18 8 10 13 2 6 7 12 19 15 3 17 4	24550.83
2	13 2 17 3 14 19 9 20 7 4 12 6 10 16 15 5 8 1 18 11	24704.00
3	16 20 6 10 12 15 9 5 3 13 2 4 7 11 19 18 1 14 17 8	18035.00
4	7 17 16 11 2 18 19 13 9 20 6 5 8 14 4 1 12 10 3 15	19180.83
5	19 1 14 4 2 10 7 17 5 13 15 11 3 20 9 8 16 6 18 12	27408.00
6	3 8 20 13 14 7 10 15 5 16 6 12 11 1 19 18 17 4 2 9	21219.33
7	17 11 8 20 13 19 10 16 4 1 14 2 7 18 6 3 15 5 12 9	20073.50
8	18 13 5 15 11 20 12 2 10 6 16 8 19 4 9 1 14 7 17 3	24582.17
9	12 6 16 9 15 5 1 13 19 17 20 11 8 7 3 2 18 10 14 4	26608.50
10	14 8 6 1 11 18 19 17 16 3 12 4 10 13 15 7 2 9 20 5	19619.17

Table B16: The sequences to generate the initial layouts for data set 21-3 of the multi-floor test problem and their associated costs.

B7: Data Set 21-4

The minimum departmental area (Area) requirements are given in Table B17. The flow matrix generator is included in Table B18. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 4.0 square distance units and the interfloor distance is 2.5 distance units. The spacefilling curve used for data set 21-1 is also used for data set 21-4 and Figure 11 exhibits the new lift locations. Finally, Table B19 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The best known solution to this test problem (\$8633.0) is characterized by the following sequence: 10-5 / 17-16-20-13-19-14-3 / 18-11-12-8-6-4 / 1-2-9-15-17.

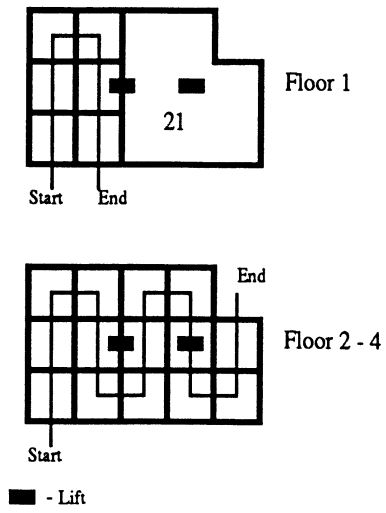

Figure 11: Spacefilling Curve and Fixed Department Location for Test Problem 21-4.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Area	2	2	2	2	2	2	4	4	3	4	2	1	2	1	2	1	2	3	3	2	8

Table B17: The departmental area requirements for the fourth multi-floor test problem with 21 departments.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Dept 7	Production
1	1	21(25)	14(10)	7(20)	1(20)	2(10)	9(20)	21	100
1	1	21(10)	14(10)	8(20)	15(25)	2			
1	2	21(10)	5(10)	10(25)	15				
2	1	21(25)	14(20)	3(20)	4(10)	6(10)	8(25)	21	200
2	1	21(20)	11(20)	18(20)	12(25)	4			
2	1	21(10)	5(10)	10(25)	18				
3	1	21(10)	14(20)	16(10)	17(10)	13(20)	8(20)	21	400
3	1	21(25)	14(10)	19(20)	20(20)	17			
3	2	21(20)	5(10)	10(25)	20				

Table B18: The flow matrix generator for data set 21-4 of the multi-floor test problem.

Init	Sequence	Cost
1	3 7 10 19 14 2 15 11 6 13 16 9 8 4 20 12 1 17 18 5	18636.33
2	18 1 14 10 17 4 2 15 13 9 5 11 7 16 3 6 20 12 8 19	20007.50
3	5 20 13 1 4 12 15 8 17 3 2 11 16 6 7 19 14 18 9 10	23295.50
4	10 19 4 1 2 16 11 13 15 14 5 12 6 3 17 20 8 7 18 9	22225.00
5	3 1 9 10 17 19 2 4 18 16 5 11 8 6 13 7 14 12 20 15	21171.67
6	8 15 11 9 2 16 18 13 19 1 17 3 20 6 14 4 5 12 10 7	17153.83
7	17 11 14 12 9 8 19 18 1 4 7 3 5 20 16 10 15 13 2 6	20253.67
8	10 18 14 2 12 13 19 5 17 16 9 3 11 1 20 8 7 15 6 4	17501.00
9	16 8 14 2 4 5 6 7 10 9 20 19 11 3 17 15 12 1 13 18	16304.50
10	14 17 3 1 5 9 6 16 18 20 8 7 13 11 15 2 10 12 4 19	21456.50

Table B19: The sequences to generate the initial layouts for data set 21-4 of the multi-floor test problem and their associated costs.

B8: Data Set 40

The minimum departmental area (Area) requirements are given in Table B20. The flow matrix generator is included in Table B21. All cost data is assumed to be \$1.00 and \$5.00 per unit distance, per unit time for horizontal and vertical travel, respectively. The grid size is 4.0 square distance units and the interfloor distance is 2.5 distance units. The spacefilling curve used for data set 40 is shown in Figure 12, which also exhibits the lift locations. Finally, Table B22 contains the 10 initial sequences of departments for the multi-floor test problem and their associated costs (in dollars). The best known solution to this test problem (\$20441.46) was obtained with MULTIPLE and initial layout 6 and is characterized by the following sequence: 1-5-6-34-30-11-2 / 3-4-9-10-23-29-28-8-38 / 21-36-22-31-37-15-32-33-39-20-26-35-12 / 27-25-17-16-14-13-24-7-19-18.

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Area	2	4	3	3	4	2	2	4	1	4	4	1	4	4	2	2	1	1	1	1
Dept	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Area	4	4	2	2	1	1	2	4	3	1	1	4	4	1	1	1	1	2	1	8

Table B20: The departmental area requirements for data set 40 of the multi-floor test problem.

Product	Mult1	Dept1	Dept2	Dept3	Dept4	Dept5	Dept6	Dept 7	Dept8	Dept9	Production
1	1	40(25)	1(10)	9(10)	10(25)	11(25)	2(50)	40			400
1	1	40(25)	1(20)	3(20)	4(10)	10					
1	1	40(10)	1(20)	5(10)	11						
1	1	40(100)	1(20)	5(10)	6(20)	11					
2	1	40(25)	1(25)	12(50)	13(10)	14(20)	15(25)	2(50)	40		100
2	1	40(25)	16(10)	17(20)	13						
2	1	40(25)	18(20)	19(20)	13						
2	1	40(50)	1(20)	3(20)	4(20)	14					
2	2	40(100)	1(20)	5(20)	15						
3	1	40(25)	20(10)	21(10)	22(20)	23(20)	8(10)	40			150
3	1	40(20)	7(10)	24(20)	25(20)	22					
3	1	40(25)	1(20)	3(10)	4(20)	25					
3	1	40(25)	1(10)	5(25)	23						
3	2	40(100)	1(20)	5(10)	6(25)	23					
4	1	40(25)	26(10)	27(20)	28(20)	29(20)	8(10)	2(20)	40		175
4	1	40(20)	1(20)	3(10)	4(10)	28					
4	1	40(100)	1(20)	5(20)	6(10)	29					
5	1	40(50)	1(10)	30(20)	31(20)	32(10)	33(10)	34(20)	2(50)	40	250
5	1	40(20)	1(20)	36(25)	37(10)	32					
5	1	40(25)	1(25)	38(10)	39(10)	33					
5	1	40(25)	1(10)	3(20)	4(10)	33					
5	1	40(100)	1(20)	5(10)	34						

Table B21: The flow matrix generator for data set 40 of the multi-floor test problem.

Init	Sequence	Cost
1	3811 20 32 36 34 39 8 26 30 23 17 7 37 29 16 19 31 5 35 27 3 9 15 10 2 24 33 1 21 13 22 6 18 14 25 28 4 12	49334.13
2	37 2 26 28 1 5 9 34 31 27 38 21 11 24 12 17 4 16 32 30 36 25 7 39 35 23 22 6 3 15 19 13 20 14 8 10 18 29 33	39107.25
3	30 26 22 11 1 36 28 13 7 39 20 19 15 24 23 29 35 10 21 38 17 12 32 25 2 8 33 31 6 18 27 9 5 37 14 4 3 34 16	45582.38
4	1 16 38 18 5 2 24 32 17 25 27 15 13 3 6 29 23 9 12 11 26 14 19 36 22 7 8 37 21 30 34 35 28 10 33 39 31 4 20	39361.71
5	27 21 32 9 3 29 12 33 1 35 36 38 13 11 7 26 18 34 8 25 30 23 15 22 6 19 5 37 10 17 20 31 28 32 2 14 24 16 4	45973.54
6	11 34 12 39 5 6 28 1 4 2 22 10 9 38 8 3 29 21 20 37 25 16 17 32 23 33 36 18 7 27 15 30 26 13 31 24 14 35 19	36048.63
7	35 6 2 9 32 12 38 10 34 7 20 14 22 24 17 21 37 3 18 25 26 1 33 27 19 36 28 15 31 29 16 4 39 8 23 30 11 13 5	48148.04
8	5 23 38 33 3 36 8 35 18 14 7 25 16 32 1 11 21 17 9 27 37 30 34 19 39 4 15 6 2 29 10 20 28 12 24 13 31 26 22	45500.87
9	11 23 13 30 1 7 37 15 5 29 10 21 38 39 12 28 35 8 32 34 18 31 20 14 3 27 16 24 33 25 36 26 22 4 2 9 19 17 6	47111.50
10	28 13 4 1 11 27 3 10 15 30 23 25 31 22 34 38 20 32 8 33 26 36 14 12 24 21 19 6 2 17 16 18 5 39 9 29 7 35 37	45287.17

Table B22: The sequences to generate the initial layouts for data set 40 of the multi-floor test problem and their associated costs.

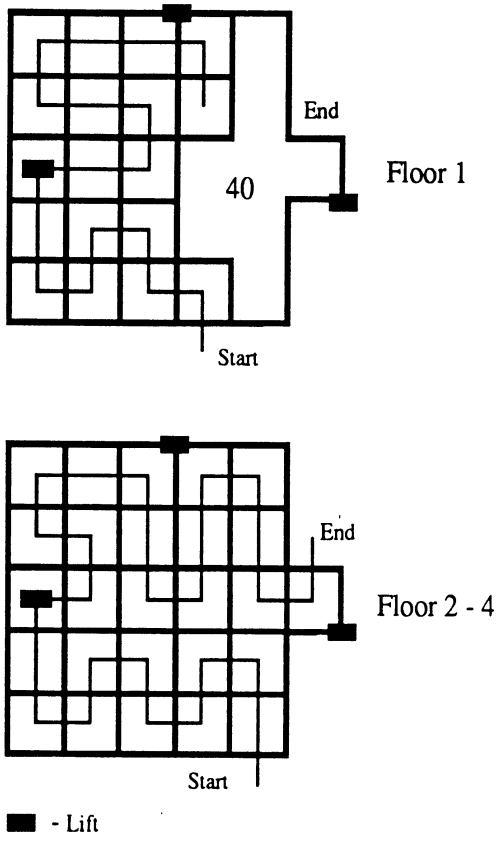

Figure 12: Spacefilling Curve and Fixed Department Location for Test Problem 40.

References

[1] Armour, G. E. and E. S. Buffa, "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," *Management Science*, Vol. **9**, 1963, pp. 294-309.

[2] Bozer, Y. A., Meller, R. D., and S. J. Erlebacher, "An Improvement-Type Layout Algorithm for Multiple Floor Facilities," Technical Report 91-11, The University of Michigan, Ann Arbor, Michigan, 1991.

[3] Burkard, R. E. and U. Derigs, *Assignment and Matching Problems: Solution Methods with FORTRAN Programs*, Lecture Notes in Economics and Mathematical Systems # 184, Springer-Verlag, New York, 1980.

[4] Burkard, R. E. and F. Rendl, "A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems," *European Journal of Operational Research*, **17**, 1984, pp. 169-174.

[5] Collins, N. E., Eglese, R. W. and B. L. Golden, "Simulated Annealing - An Annotated Bibliography," Report No. 88-019, College of Business and Management, University of Maryland, College Park, MD, 1988.

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

[7] Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, New York, 1979.

[8] Golden, B. L. and C. C. Skiscim, "Using Simulated Annealing to Solve Routing and Location Problems," *Naval Research Logistics Quarterly*, Vol. **33**, 1986, pp. 261- 279.

[9] Golden, B. L. and W. R. Stewart, "Empirical Analysis of Heuristics," *The Traveling Salesman Problem*, John Wiley & Sons Ltd., New York, New York, 1985, pp. 207-214.

[10] Heragu, S. S. and A. S. Alfa, "A Hybrid Simulated Annealing Based Algorithm for the Layout Problem," Working Paper # 06/90, School of Business and Economics, State University of New York, Plattsburgh, NY, 1990.

[11] Johnson, D. S., Aragon, C. R., McGeoch, L. A. and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning," *Operations Research*, Vol. **37**, 1989, 865-892.

[12] Johnson, R. V., "SPACECRAFT for Multi-Floor Layout Planning," *Management Science*, Vol. **28**, 1982, 407-417.

[13] Kirkpatrick, S., Gelatt, C. D., Jr. and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. **220**, 1983, 671-680.

[14] Koopmans, T. and M. Beckmann, "Assignment Problems and the Location of Economic Activities," *Econometrica*, Vol. **25**, 1957, 53-76.

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

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

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

[18] Nugent, C. E., Vollman, R. E. and J. Ruml, "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," *Operations Research*, Vol. **16**, 1968, pp. 150-173.

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

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

[21] Tam, K. Y., "A Simulated Annealing Algorithm for Allocating Space to Manufacturing Cells," *The International Journal of Production Research*, Vol. **30**, 1991, pp. 63-87.

[22] Wilhelm, M. R. and T. L. Ward, "Solving Quadratic Assignment Problems by 'Simulated Annealing'," *IIE Transactions*, Vol. **19**, 1987, pp. 107-119.