

An evolutionary approach to incorporating intradepartmental flow into facilities design

B.A. Norman^{a,*}, A.E. Smith^b, E. Yildirim^a, W. Tharmmaphornphilas^a

^aDepartment of Industrial Engineering, University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA 15261, USA

^bDepartment of Industrial and Systems Engineering, Auburn University, 207 Dunstan Hall, Auburn, AL 36849-5346, USA

Accepted 17 October 2000

Abstract

Minimizing material transportation is an important design object for many facilities. In traditional facility design, departments and work centers have been located based on interdepartmental flow patterns. We introduce a methodology that extends the planning process to also consider intradepartmental flow patterns. A math programming approach is developed that uses an objective function and constraints that specifically addresses both the intradepartmental and the interdepartmental material flows together. The math programming approach can be used to solve small problems but it is difficult to scale this method to problems with realistic sizes. Therefore, we devise an evolutionary optimization methodology for this same problem. The evolutionary method can readily solve both large and small problems. The effectiveness of the evolutionary approach is demonstrated on a suite of test problems. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Facility design; Material flow; Genetic algorithms; Optimization

1. Introduction

Facility design problems generally involve the partition of a building into departments (work centers or cells) along with a material flow structure and a material handling system to link the departments. The primary objective of the design problem is to minimize the costs associated with production and materials movement within the facility. Such problems occur in many organizations, including manufacturing cell layout, hospital layout, semiconductor manufacturing and service center layout. They also occur in environmental management of forests, wetlands, etc. [1]. By any monetary measure, facilities design is an important problem and one that has assumed even greater importance as manufacturers strive to become more agile and responsive [2]. For US manufacturers, between 20 and 50% of total operating expenses are spent on material handling and an appropriate facilities design can reduce these costs by at least 10–30% [3]. Dr James A. Tompkins, a seminal researcher in the field, wrote in the August 1997 issue of *IIE Solutions*, “Since 1955, approximately 8 percent of the U.S. GNP has been spent annually on new facilities. In addition, existing facilities must be continually modi-

fied...These issues represent more than \$250 billion per year attributed to the design of facility systems, layouts, handling systems, and facilities locations...” [2].

An important aspect of facilities design is locating the input/output (I/O) points (also called pickup and delivery points) for each department within a facility. This is generally done by considering only the flows coming into and going out of each department and locating a specified number of I/Os per department (usually one and usually a combined input and output point). Flows between departments are significant in facilities that operate as job shops and also in facilities that utilize manufacturing cells if there is flow between cells or if the layout is a hybrid having both job shop and cellular manufacturing features. However, how material flows within a department can be equally important in determining I and O sites. In fact, there are certain manufacturing methods and production methodologies that demand a certain flow pattern within the department and these strongly affect the proper placement of I/Os. In this paper, we incorporate the manner of flow of material *within* departments with the flows *between* departments to optimally locate I/O points.

Specifically, we consider designs where there are both separate input (I) locations and output (O) locations and where there are combined I and O locations. Consider departments E, G and N in Fig. 1 where each exhibits a common type of intradepartmental flow pattern. Department

* Corresponding author. Tel.: +1-412-624-9841; fax: +1-412-624-9831.

E-mail addresses: banorman@engr.pitt.edu (B.A. Norman), aesmith@eng.auburn.edu (A.E. Smith).

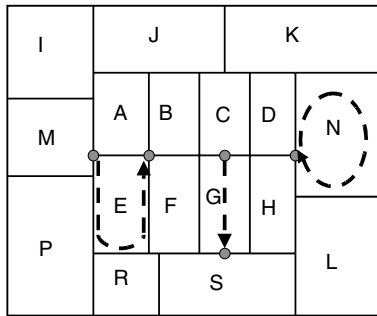


Fig. 1. Typical intradepartmental flows.

E has a *U-shaped* flow, department G has a *linear* (L) flow and department N has a *circular* (C) flow. To model this it is necessary to include constraints that require that the I location be on the opposite (or same) side of the department as the O location as would arise in a linear (or U-shaped) layout. Additionally, in departments E and G the input point for the department is distinct from the output point. Modeling this situation as one or two I/O locations would be incorrect and could lead to designs that would not perform well in practice.

We propose two solution methodologies for solving this design problem. The first uses math programming but is not readily applicable to large problems due to the computation time required. The second is a genetic algorithm methodology that provides optimal or near optimal solutions within a reasonable time frame.

The remainder of this paper is organized as follows. Section 2 discusses the relevant literature concerning facility design. Section 3 provides a formal statement and math programming formulation of the design problem. Section 4 describes the proposed genetic algorithm solution methodology. Section 5 presents computational test results on several problems. Section 6 discusses conclusions and possible extensions of this work.

2. Literature review

The facility design problem in the literature has usually been studied as either ‘block layout’ or ‘detailed design’ while a few researchers integrate the two. Block layout focuses on determining the relative locations of different departments within a facility. Detailed layout considers the material flow patterns within each department. Two surveys on block layout are by Kusiak and Heragu [4] and Meller and Gau [3], and a survey on discrete material movement through the facility is by Sinriech [5]. In this research we primarily focus on aspects of detailed layout and we now review the relevant work related to detailed layout design.

The primary approaches to detailed design have been concerned with placement of I/O stations, specification of material flow paths (aisles) and material handling control strategies, such as those required by automated guided vehicles (AGVs). Chhaged et al. [6] present a comprehen-

sive discussion on flow networks and distinguish between free flow (shortest distance paths) and flow which follows departmental contours, between capacitated and uncapacitated aisles, and between one way and bidirectional flowpaths. The authors then present both exact and heuristic algorithms to optimize a free flow path for a given block layout with pre-sited I/O points. The idea of *contour flow* has since been used in the literature related to discrete material handling systems such as AGVs, as found in the segmented flow topology (SFT) work of Sinriech and Tanchoco [7,8]. In this work, I/Os and the resulting flow paths are chosen based on both distance-based transport costs and on fixed costs for building an I/O. In Ref. [8], they further allowed material flow between two departments to be split heuristically among different paths if it is cost effective.

A number of papers have dealt with optimally choosing the I/O locations and/or aisles for a given block layout (for example, for free flow [9] and for the segmented flow topology flow along contours [10]). Palliyil and Goetschalckx [11] present exact and heuristic methods for optimally locating I/O stations and their flowpaths that can be applied to unidirectional or bidirectional flowpaths. Arapoglu et al. [12] consider the problem of locating I/Os in a facility where the locations of the departments are fixed. They develop a math programming formulation, a constructive heuristic, and a genetic algorithm methodology for the problem. However, they assume that in all departments material both enters and exits from a single combined I/O point. A similar problem is presented in Klein and Kim [13] and the authors propose constructive heuristics for solving the combined I/O location problem.

Benson and Foote [14] consider the placement of aisles and I/O points after the relative location of the departments and the general aisle structure have been selected. In the work of Nagi and co-workers [15,16], pre-defined departmental shapes are set on a grid covering the facility space, which is larger than the sum of the department areas. In Ref. [15] Dijkstra’s shortest path algorithm is used to calculate the rectilinear distance to and from pre-specified I/O points. In Ref. [16] I/O points are placed during the optimization and a corner constraint is imposed to encourage paths that are straight. Both papers use a simulated annealing metaheuristic to alter departmental placement. Chitratanaawat and Noble [17] use a QAP formulation of departments and simultaneously optimize placement, I/O location, and flowpaths using the shortest rectilinear distance metric and a tabu search metaheuristic.

The proposed method differs from the methods found in the literature because we explicitly consider a combination of linear, circular and U-shaped intradepartmental flow patterns for each of the departments. Combining different flow pattern types makes the problem much more complicated and necessitates the development of new solution methodologies. Additionally, much of the previous work related to locating I/O points only considers the rectilinear distance between departments rather than the contour or path distance.

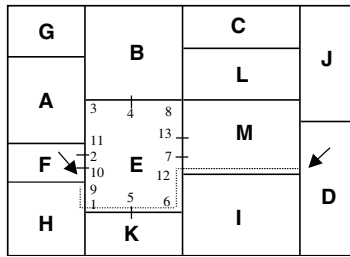


Fig. 2. Example of perimeter or contour distance travel.

3. Problem statement

In this research we consider the problem of determining the best location for the I/O points of each department within a facility. The objective is to find the set of locations that will minimize the total travel distance of all materials within the facility. We assume that material cannot pass through departments and must move around departments using the contour or perimeter metric as discussed in Norman et al. [18] In Fig. 2, the dashed line shows an example of the contour distance for moving from department F to department D if the I/O points are positioned at the locations indicated by the arrows in the figure.

We assume that the department locations have already been determined and that we know the flow pattern type for each department. That is, we know which departments will have U-shaped flows and which will have linear or circular flows. We also make assumptions about each of these three flow pattern types. We assume that for a U-shaped flow, material enters the department at one corner and exits at another corner. Given this assumption, there are eight different U-shaped flow possibilities. The first row of Fig. 3 shows four of these and the other four can be constructed by reversing the flow directions on each diagram in the first row of Fig. 3. In a similar manner we assume that the linear flows go from the midpoint of one side of the department as indicated in the second row of Fig. 3. Flows in a circular flow pattern enter and exit at the same location as indicated in the third row of Fig. 3. Theoretically, the I/O location for a circular flow pattern could be anywhere on the perimeter of a given department but using the results of Refs. [12,18] we can show that we only need to

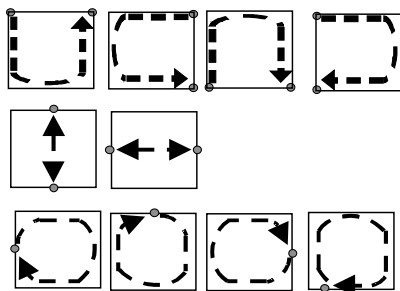


Fig. 3. Flow pattern types.

consider a discrete set of locations which constitutes a dominant set. This set of locations includes the four corners of the given department and all locations where the given department intersects either the corner or midpoint of one side of any other department. Consider the example given in Fig. 2. If department E has a circular flow pattern then the candidate locations include locations 1–13.

We now clarify how these 13 locations were selected and present our candidate I and O location numbering method. For any department, point 1 refers to the bottom left corner of the department, point 2 refers to the midpoint of the left side, point 3 refers to the top left corner, points 4 and 5 refer to the midpoints of the bottom and top edges, respectively, and points 6–8 refer to the bottom, midpoint, and top of the right side of the department. Points 9 and greater refer to any additional points where the department intersects either the midpoints or corner points of a side of another department. The numbering for points greater than or equal to 9 begins on the left side of the department, then continues to the bottom of the department, followed by the top and then the right side of the department. For a U-shaped flow the I/O points can only be at locations 1, 3, 6, and 8 because these represent the four corner points (note only certain pairs represent feasible U-shaped flows). For a linear flow the I/O points can only be at locations 2, 4, 5, and 7 because these represent the midpoint locations for each side. For a circular flow the I/O point can be at any of the locations 1–8 or at locations 9 to p_i where p_i is dependent on the physical layout of the departments (e.g. $p_i = 13$ for department E in Fig. 2).

The value of p_i can be reduced if we know the flow patterns for the departments adjacent to department i . Consider department E in Fig. 2 that has a circular flow. If department M has a U-shaped flow then location 13 does not need to be considered because department M can only have its I and O locations at its corners and not at location 13 and therefore it would never be advantageous to locate department E's I/O at location 13.

Once all of the candidate I and O locations have been determined for all of the departments, the material flow network can be constructed. This network consists of nodes at each department's candidate I and O locations and arcs consisting of the departmental edges connecting these nodes. It is possible to determine the distance between the I and O locations of each pair of departments by solving a shortest path problem on the underlying network. These distances represent the contour distances for moving material between departments.

Given the I and O location and material flow network information, the design problem is to determine the precise flow pattern for each department. The precise flow pattern for each department indicates where the I and O are located. This problem can be modeled using math programming as shown below. First we define the variables, then present the formulation followed by an explanation of the objective function and each of the constraints in the model.

3.1. Model notation and definition

3.1.1. Model parameters

I	Set of departments $i = 1, 2, \dots, n$
J	Set of departments $j = 1, 2, \dots, n$
K_i	Set of input/output points for department i $k = 1, 2, \dots, p_i$
L_i	Set of input/output points for department i $l = 1, 2, \dots, q_i$
A	Set of input/output points of flow type U $a = 1, 3, 6, \text{ and } 8$
B	Set of input/output points of flow type L $b = 2, 4, 5, \text{ and } 7$

$flow_{ij}$	Amount of flow from department i to department j
$dist_{ikjl}$	Distance from output point k of department i to input point l of department j

3.1.2. Decision variables

$$x_{ik} = \begin{cases} 1 & \text{if department } i \text{ has point } k \text{ as an output point} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{if department } i \text{ has point } k \text{ as an input point} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ikjl} = \begin{cases} 1 & \text{if there is a flow from output point } k \text{ of department } i \text{ to input point } l \text{ of department } j \\ 0 & \text{otherwise} \end{cases}$$

d_{ikjl} = Distance of the shortest path from output point k of department i to input point l of department j

d_{ij} = Distance of flow path between department i to department j

$$c_{ik} = \begin{cases} 1 & \text{if department } i \text{ has flow type C and uses point } k \text{ as an input/output point} \\ 0 & \text{otherwise} \end{cases}$$

$$u_{ia} = \begin{cases} 1 & \text{if department } i \text{ has flow type U and uses point } a \text{ as an output point} \\ 0 & \text{otherwise} \end{cases}$$

$$l_{ib} = \begin{cases} 1 & \text{If department } i \text{ has flow type L and uses point } b \text{ as an output point} \\ 0 & \text{otherwise} \end{cases}$$

$$uu_{ia} = \begin{cases} 1 & \text{If department } i \text{ has flow type U and uses point } a \text{ as an input point} \\ 0 & \text{otherwise} \end{cases}$$

$$ll_{ib} = \begin{cases} 1 & \text{If department } i \text{ has flow type L and uses point } b \text{ as an input point} \\ 0 & \text{otherwise} \end{cases}$$

$$c_i = \begin{cases} 1 & \text{If department } i \text{ has flow type C} \\ 0 & \text{otherwise} \end{cases}$$

$$u_i = \begin{cases} 1 & \text{If department } i \text{ has flow type U} \\ 0 & \text{otherwise} \end{cases}$$

$$l_i = \begin{cases} 1 & \text{If department } i \text{ has flow type L} \\ 0 & \text{otherwise} \end{cases}$$

$m_i = a \{0, 1\}$, binary variable for each department i

3.1.3. Objective function

$$\text{Minimize } \sum_{j=1}^n \sum_{i=1}^n \text{flow}_{ij} d_{ij} \quad (1)$$

3.1.4. Constraints

$$\sum_{k=1}^{p_i} x_{ik} = 1 \quad \forall i \quad (2)$$

$$\sum_{k=1}^{p_i} y_{ik} = 1 \quad \forall i \quad (3)$$

$$\sum_{l=1}^{q_j} \sum_{k=1}^{p_i} z_{ikjl} = 1 \quad \forall i, j \quad (4)$$

$$z_{ikjl} \leq x_{ik} \quad \forall i, k \quad (5)$$

$$z_{ikjl} \leq y_{jl} \quad \forall j, l \quad (6)$$

$$\sum_{l=1}^{q_j} \sum_{k=1}^{p_i} z_{ikjl} \text{dist}_{ikjl} = d_{ij} \quad \forall i, j \quad (7)$$

$$x_{i1} = c_{i1} + u_{i1} \quad \forall i \quad (8)$$

$$x_{i2} = c_{i2} + l_{i2} \quad \forall i \quad (9)$$

$$x_{i3} = c_{i3} + u_{i3} \quad \forall i \quad (10)$$

$$x_{i4} = c_{i4} + l_{i4} \quad \forall i \quad (11)$$

$$x_{i5} = c_{i5} + l_{i5} \quad \forall i \quad (12)$$

$$x_{i6} = c_{i6} + u_{i6} \quad \forall i \quad (13)$$

$$x_{i7} = c_{i7} + l_{i7} \quad \forall i \quad (14)$$

$$x_{i8} = c_{i8} + u_{i8} \quad \forall i \quad (15)$$

$$x_{ik} = c_{ik} \quad \forall i, k \geq 9 \quad (16)$$

$$y_{i1} = c_{i1} + uu_{i1} \quad \forall i \quad (17)$$

$$y_{i2} = c_{i2} + ll_{i2} \quad \forall i \quad (18)$$

$$y_{i3} = c_{i3} + uu_{i3} \quad \forall i \quad (19)$$

$$y_{i4} = c_{i4} + ll_{i4} \quad \forall i \quad (20)$$

$$y_{i5} = c_{i5} + ll_{i5} \quad \forall i \quad (21)$$

$$y_{i6} = c_{i6} + uu_{i6} \quad \forall i \quad (22)$$

$$y_{i7} = c_{i7} + ll_{i7} \quad \forall i \quad (23)$$

$$y_{i8} = c_{i8} + uu_{i8} \quad \forall i \quad (24)$$

$$y_{ik} = c_{ik} \quad \forall i, k \geq 9 \quad (25)$$

$$\sum_{a \in A} u_{ia} = u_i \quad \forall i \quad (26)$$

$$\sum_{a \in A} uu_{ia} = u_i \quad \forall i \quad (27)$$

$$\sum_{b \in B} l_{ib} = l_i \quad \forall i \quad (28)$$

$$\sum_{b \in B} ll_{ib} = l_i \quad \forall i \quad (29)$$

$$\sum_{k=1}^{p_i} c_{ik} = c_i \quad \forall i \quad (30)$$

$$u_i + l_i + c_i = 1 \quad \forall i \quad (31)$$

$$l_{i2} - ll_{i7} = 0 \quad \forall i \quad (32)$$

$$l_{i7} - ll_{i2} = 0 \quad \forall i \quad (33)$$

$$l_{i5} - ll_{i4} = 0 \quad \forall i \quad (34)$$

$$l_{i4} - ll_{i5} = 0 \quad \forall i \quad (35)$$

$$u_{i1} - uu_{i3} \leq m_i \quad \forall i \quad (36)$$

$$u_{i1} - uu_{i6} \leq 1 - m_i \quad \forall i \quad (37)$$

$$u_{i3} - uu_{i1} \leq m_i \quad \forall i \quad (38)$$

$$u_{i3} - uu_{i8} \leq 1 - m_i \quad \forall i \quad (39)$$

$$u_{i6} - uu_{i1} \leq m_i \quad \forall i \quad (40)$$

$$u_{i6} - uu_{i8} \leq 1 - m_i \quad \forall i \quad (41)$$

$$u_{i8} - uu_{i3} \leq m_i \quad \forall i \quad (42)$$

$$u_{i8} - uu_{i6} \leq 1 - m_i \quad \forall i \quad (43)$$

The model is formulated as an integer linear programming problem with the objective function of minimizing the product of flow times distance for the layout. The flow amount is fixed but the flow distances between the departments depend on where the I/O locations are placed in each

Chromosome	(4,4; 6,1; 3,8; 5,4; 7,2)				
Department	1	2	3	4	5
Input	4	6	3	5	7
Output	4	1	8	4	2

Fig. 4. Example GA representation.

department. The assignment constraints (2) and (3) require each department to have exactly one output point (constraint (2)) and one input point (constraint (3)). Constraint (4) ensures that a path is selected for moving material from department i to department j . Constraints (5) and (6) permit z_{ijkl} to equal 1 only if both x_{ik} and y_{jl} equal 1, meaning that the material flow path can only go from output point k of department i to input point l of department j if k and l are chosen as the I/O locations of departments i and j , respectively. Constraint (7) is used to evaluate the distance of the material flow path from department i to department j . Constraints (8)–(16) permit choosing only one flow type for the output point of each department. For example, if location 1 is selected as the output point, then constraint (8) ensures that the flow type must be either C or U. If location 2 is chosen as the output point, then constraint (9) requires that the flow type be either C or L. Constraints (17)–(25) are similar to constraints (8)–(16) but are applied to the input points. Constraints (26) and (27) ensure that department i has flow type U if the output and input points of department i have flow type U while constraints (28)–(30) perform a similar function for flow types L and C, respectively. Constraint (31) allows each department to have only one flow type. Constraints (32)–(43) guarantee that the same flow types are chosen for output and input points of the same department. Constraints (32)–(35) are applied for flow type L. For example, constraint (32) shows that if department i uses location 2 as an output point, only location 7 can be used as an input point and this department has a flow type L. Constraints (36)–(43) are similar for flow type U. However, for this kind of flow type, if location 1 is chosen to be the output point for department i either location 3 or location 6 can be chosen to be an input point so either—or constraints are used.

4. Solution methodologies

While it is possible to use the math programming formulation provided in Section 3 to solve small problems it is not a practical method to use on larger problems due to the long CPU times required for large problems (this will be discussed further in Section 5). The structure of the problem is such that the size of the search space grows exponentially as the number of departments increases. Therefore, we develop a genetic algorithm heuristic for solving this problem.

GAs are powerful heuristic computational procedures that are designed to search and find solutions to complex problems in situations where the number of possible alter-

native solutions is vast [19]. GAs, as described by Holland [20], utilize the principles of natural selection and survival of the fittest to ‘evolve’ a population of good solutions to a particular problem or set of problems. Solutions that are better or more ‘fit’ have a greater probability of surviving during the selection process. In this case, ‘fitness’ corresponds to the total travel distance for all materials in the facility. A good set of departmental I/O locations that results in a low total travel distance would have a good fitness score.

There are several important design issues associated with the development of a GA methodology for solving a problem. The first is the choice of problem encoding or representation. The GA manipulates solutions or chromosomes that utilize the problem encoding. There are two primary operators that are used to alter the chromosomes during the GA search process — crossover and mutation. Crossover is the mechanism that combines elements of two existing solutions in order to create new solutions. The traditional method for performing crossover is single-point crossover but many variants exist (see Refs. [21,22] for more details.) The method of selecting parents for crossover is also important. The selection process should ensure that solutions that have better fitness are more likely to be chosen to serve as parents in the crossover operation. Mutation is used to recover genetic material that may have been lost during the search and helps the algorithm to avoid converging prematurely. For details concerning different selection methods and mutation operators see Refs. [7,13].

Both the structure of the problem and the representation of the solution instances are crucial for an effective GA implementation. Our particular problem structure enabled us to make use of GAs without complicated transformations. In this problem, chromosomes consist of genes that carry the necessary information about the I/O points for each department. A chromosome has N genes for a layout with N departments where each gene corresponds to a department.

Fig. 4 shows an example chromosome with five genes corresponding to the I/O pairs for each of the five departments in the layout. The first department has its I point at node 4 in the layout and its O point at node 4 in the layout. Thus, any flow to department 1 will enter via node 4 and will exit via node 4. The other genes have the same structure and meaning.

Recall that each department must have one of the three flow pattern types — U, L or C. Thus, it is necessary to select I/O points in such a way that they form a feasible pair by representing one of the three flow pattern types. For example, the pair (1, 2) would never be permitted since this does not represent any of the flow pattern types.

Creating a set of feasible lists for each flow pattern type solves the feasibility problem that may arise when creating random I/O pair assignments. These lists, shown in Fig. 5, will be used in the GA when there is a need to create a random I/O assignment. For example, if department 2 has an L flow pattern type then it can only have one of the

U	1,3	1,6	3,8	3,1	6,1	6,8	8,3	8,6
L	2,7	4,5	5,4	7,2				
C	1,1	2,2	...	p_i	p_i			

Fig. 5. Feasible I/O pairs for each flow pattern type.

four I/O assignments shown in Fig. 5. Note that the number of C candidate I/O assignments will vary from department to department. In Fig. 2 department E would have 13 candidate C flow assignments while department M would have only 10 possibilities.

A key design component of any GA is the choice of operators and their definition. We now describe the operators utilized in the proposed GA. This is followed by a pseudocode of the GA.

Crossover is the main operator that directs the search into promising regions. Two parents are selected by choosing one via tournament selection with a tournament size of 2 and choosing the other one randomly. The parent with better fitness is labeled parent 1 and the other parent 2. One offspring is created by using parameterized uniform crossover where the allele values (both the I and O value) for each gene are taken from parent 1 with probability 0.7 and from parent 2 with probability 0.3. Thus, the parent with better fitness is more likely to provide allele values to the offspring. Note that this crossover mechanism will always provide feasible offspring since each I/O pair was feasible in the parents. The parameter *crossover_size* determines how many offspring will be generated via crossover.

Mutation is used to prevent premature convergence before reaching the optimal solution. The rate of mutation is governed by two parameters. First, the parameter *mutation-rate* determines what percentage of the population will undergo mutation. Then $mutation_size = popsize$ (the

population size) \times *mutation-rate* members of the current population are randomly selected without replacement for mutation. Second, the parameter *gene_mutation_probability* determines how many alleles are changed in a member of the population that undergoes mutation. Each gene has its I/O pair changed with probability equal to *gene_mutation_probability*. If the I/O pair is changed, for departments with flow type C or L the new I/O pair is randomly generated from the candidate list of I/O pairs as indicated in Fig. 5 and is required to be different from the current I/O pair. If a department with flow type U is mutated then with *U_gene_mutation_probability* either I or O but not both are changed and with probability $(1 - U_gene_mutation_probability)$ both I and O are changed. The mutated chromosome replaces the original chromosome in the new population. All of the mutation parameters are user-defined inputs that are entered at the beginning of the GA procedure.

The algorithm utilizes an elitist strategy where the best members are copied in exactly their current form from one generation to the next. The ratio of the elitist individuals to the whole population is a user-defined variable and is entered at the beginning of the GA procedure.

Here is a list of the key input parameters: *popsize*, *crossover_size*, *mutation_size*, *gene_mutation_probability*, *U_gene_mutation_probability* and *elitist_size*. The pseudocode for the GA is provided in Fig. 6.

We also developed a greedy local search improvement algorithm that was used to improve the final solution found by the GA. The algorithm looks at moving the I and/or O location for each department independently (while maintaining feasibility) and selecting the move with the most improvement in the objective function. For example, consider a department *i* that has a C flow, has 10 candidate I/O locations and the final GA solution has the I/O at

Step 1:

- 1.1. Construct an initial population with $2 * popsize$ randomly generated individuals
- 1.2. Sort the initial population based on fitness and retain only the *popsize* best solutions as the current population.

Step 2:

- 2.1. Copy the *elitist_size* best solutions directly into the new population
- 2.2. Repeat *crossover_size* times
 - Select two parents
 - Perform parameterized uniform crossover
 - Store the new chromosome in the new population
- 2.3. Repeat *mutation_size* times
 - Randomly select a chromosome from the new population
 - Mutate each department based on *gene_mutation_probability*
 - Replace the old chromosome with the mutated one
- 2.4. Sort the current population based on fitness

Step 3:

- 3.1. Set the current population equal to the new population
- 3.2. If there has been no improvement in the best fitness value for 100 generations stop. Otherwise, go to Step 2.

Fig. 6. GA pseudocode.

candidate location 8. The improvement algorithm would investigate moving department i 's I/O to locations 1–10 (excluding 8), while keeping the I and O locations of the other departments exactly as they are in the final GA solution, and evaluate the new cost associated with each move. A similar process is repeated for each of the other departments. After all of the moves for each of the departments have been analyzed, the move that yields the greatest improvement in the objective function is made and this becomes the new best solution. The greedy local search improvement process is repeated using this new solution until no more improvement is possible.

5. Computational test results

Extensive computational tests were run using the proposed solution methodologies. A total of 48 problems were constructed and tested using the math program and the GA. The problems were based on eight layouts from existing problems drawn from the facility layout literature that were modified to include flow pattern type. For each layout six problems were created. The first assumed that each department had a circular flow pattern, the second and third assumed all of the departments had linear and U-shaped flows, respectively. The fourth, fifth, and sixth problems represent problems with a mix of flow pattern types. These were created by randomly assigning a flow pattern type to each department.

The test problems are identified in Table 1 by the author's initials, the size of the problem, and the version (where necessary) VC-10 represents the 10-department problem of van Camp [23]. BAZ-14-1 and BAZ-14-2 represent two different layouts that utilize the flow data from the 14-department problem of Bazaraa [24]. AB-20-1 and AB-20-2 are two layouts that utilize the flow data from the 20-department problem of Armour and Buffa [25]. AA-50-1 and AA-50-2 are two 50-department problems and AA-60-1 is a 60-department problem first introduced in Ref. [12].

Math programming was applied to each of the problems but only the smaller problems could be solved consistently in less than 24 h of CPU time using CPLEX Version 6.5 on a Sun4u Sparc Ultra-Enterprise. The problems that CPLEX could solve required an average of 4 h of CPU time. Table 1 shows the optimal solution for those problems that could be solved using math programming. The remainder of the table presents comparisons to the best known solution. The best known solution represents the best solution found for each problem based on all computational testing that was done including testing done with different parameter values than those presented in the GA description.

In Section 4, several GA parameters were discussed. These parameters were tested and the GA was found to be fairly robust regarding them. Population sizes ranging from 50 to 150 were tested and the GA found good solutions

across this range. A population size of 150 generated more consistently good solutions, especially for the larger problems; therefore, the GA results presented in Table 1 use a value of 150. Another important factor was the number of elite solutions. Values ranging from 5 to 40% of the population size were tested. Empirical results indicated that values in the range of 20–30% performed the best. Larger values tended to cause premature convergence of the algorithm and with smaller values the search results were not as consistent. Ultimately, a value of 30% was used in the computational testing. The number of solutions mutated (*mutation_size*) was varied between 1 and 50% of the population. The algorithm was relatively insensitive to this value and ultimately a value of 10% was selected. The *gene_mutation_probability* was set to 10% based on our previous experience with similar GAs and this value performed well for the test problems. The *U_gene_mutation_probability* was set to 50% to balance the two types of mutation for U-shaped flow patterns.

The GA was run 10 times for each problem using a different random number seed in each run. Because a GA is a stochastic search algorithm the algorithm does find different solutions for different runs. Table 1 indicates the minimum, maximum, and average value from these 10 different runs. The data in Table 1 indicate that the GA was robust across runs and across the different problems. The maximum deviation from the best known solution for a single random number seed was 6.9%, which is very consistent considering a total of 480 runs were made. The average deviation is less than 3% and there were only three problems where the best GA solution was more than 1% above the best known solution. When the GA was combined with the greedy local search improvement routine then the average deviation from the best known solution was less than 1.8% across all of the problems and there are only five problems where the GA and greedy search did not find the best known solution (but it found solutions that were within 0.6% of the best known solution). The greedy local search improvement improves the GA's performance by 1.1% on average, with only a modest increase in computation time. Therefore, it seems to be a good idea to combine the two methods. Appendix A contains layout diagrams indicating the I and O locations and the flow paths for three sample problems.

The GA produces consistently good results and is much faster than the math program. Using only the GA requires less than 1 s of CPU time on a Sun Sparc 20 to solve the van Camp problems and this only increases to an average of 25 s for the largest problems. The improvement algorithm adds less than 1 s of CPU time for the smaller problems and up to 2 s for the largest problems. Concerning CPU time, the computational benefits of the GA versus the math program are clear. These benefits are especially significant if the I/O location problem discussed here is solved as a subproblem for each block layout evaluated during a block layout design process.

Table 1
Math programming and GA test results

Problem	Flow pattern	Optimal or best known	GA			GA with greedy search		
			MIN	MAX	AVG	MIN	MAX	AVG
VC-10-1	All C	7239.0 ^a	7239.0	7670.7	7368.5	7239.0	7670.7	7368.5
VC-10-1	All U	12929.8 ^a	12929.8	13941.0	13037.8	12929.8	12929.8	12929.8
VC-10-1	All L	13745.4 ^a	13745.4	13920.3	13762.9	13745.4	13745.4	13745.4
VC-10-1	Mixed 1	11890.9 ^a	11890.9	12712.9	11970.8	11890.9	12712.9	11952.6
VC-10-1	Mixed 2	12668.2 ^a	12668.2	13236.5	12704.0	12668.2	13097.1	12678.9
VC-10-1	Mixed 3	15185.9 ^a	15185.9	15925.9	15217.0	15185.9	15185.9	15185.9
BA-14-1	All C	4223.1 ^a	4223.1	4382.4	4266.0	4223.1	4223.1	4223.1
BA-14-1	All U	4008.4 ^a	4008.4	4069.9	4021.8	4008.4	4008.4	4008.4
BA-14-1	All L	5355.2 ^a	5355.2	5357.7	5355.4	5355.2	5355.2	5355.2
BA-14-1	Mixed 1	4855.9 ^a	4855.9	4902.6	4870.3	4855.9	4902.6	4865.2
BA-14-1	Mixed 2	4832.7 ^a	4832.7	4833.8	4833.0	4832.7	4832.7	4832.7
BA-14-1	Mixed 3	4561.7 ^a	4561.7	4582.6	4564.7	4561.7	4561.7	4561.7
BA-14-2	All C	4991.8 ^a	4991.8	5291.9	5111.7	4991.8	5187.8	5070.2
BA-14-2	All U	4963.3 ^a	4963.3	5258.3	5056.1	4963.3	5185.2	5029.9
BA-14-2	All L	7329.8 ^a	7329.8	7330.7	7329.9	7329.8	7329.8	7329.8
BA-14-2	Mixed 1	6072.1 ^a	6072.1	6213.5	6100.4	6072.1	6081.4	6078.1
BA-14-2	Mixed 2	5341.3 ^a	5341.3	5625.9	5414.7	5341.3	5547.1	5408.2
BA-14-2	Mixed 3	6732.6 ^a	6732.6	6970.3	6785.4	6732.6	6929.4	6766.1
AB-20-1	All C	344.8 ^a	347.5	357.8	350.7	345.4	350.8	347.8
AB-20-1	All U	440.1	440.1	454.9	445.4	440.1	451.8	443.9
AB-20-1	All L	673.8	673.8	681.9	676.7	673.8	681.8	676.2
AB-20-1	Mixed 1	507.2	507.2	511.3	508.6	507.2	510.2	507.8
AB-20-1	Mixed 2	489.5	489.5	495.8	492.4	489.5	489.5	489.5
AB-20-1	Mixed 3	477.8	478.3	484.4	479.9	477.8	484.4	479.2
AB-20-2	All C	334.9 ^a	335.1	344.5	339.5	334.9	339.7	336.7
AB-20-2	All U	432.2	432.2	434.0	432.4	432.2	432.2	432.2
AB-20-2	All L	712.6	712.6	719.0	716.8	712.6	718.9	716.6
AB-20-2	Mixed 1	518.2	518.2	525.2	520.3	518.2	522.3	518.6
AB-20-2	Mixed 2	535.2	535.2	541.9	536.3	535.2	535.3	535.2
AB-20-2	Mixed 3	543.0	544.3	549.7	546.0	543.0	547.6	544.9
AA-50-1	All C	39694.4	39898.4	40114.1	40013.2	39694.4	39694.4	39694.4
AA-50-1	All U	41103.9	41120.6	41438.2	41243.6	41103.9	41103.9	41103.9
AA-50-1	All L	46646.5	46646.5	46672.6	46654.8	46646.5	46646.5	46646.5
AA-50-1	Mixed 1	42951.6	43013.2	43234.2	43092.5	42951.6	42957.5	42954.6
AA-50-1	Mixed 2	42383.4	42422.6	42601.9	42506.6	42383.4	42387.3	42384.9
AA-50-1	Mixed 3	42392.5	42436.2	42622.5	42530.1	42392.5	42392.5	42392.5
AA-50-2	All C	38354.8	38749.8	40155.1	39313.6	38366.8	39098.8	38725.4
AA-50-2	All U	32600.6	32946.5	34296.6	33575.2	32690.1	33465.1	33036.1
AA-50-2	All L	39194.6	39223.6	39443.0	39322.1	39194.6	39364.4	39216.5
AA-50-2	Mixed 1	37614.9	38144.6	38681.7	38376.2	37848.6	38492.7	38021.6
AA-50-2	Mixed 2	38453.4	38832.1	39593.4	39055.0	38453.4	39055.4	38606.7
AA-50-2	Mixed 3	37139.6	37500.1	38285.1	37845.0	37238.6	37714.6	37463.3
AA-60-1	All C	50930.2	51201.9	51671.2	51462.9	50930.2	50930.2	50930.2
AA-60-1	All U	46660.5	46866.2	47202.0	47038.9	46660.5	46660.5	46660.5
AA-60-1	All L	50677.9	50677.9	50941.2	50732.3	50677.9	50677.9	50677.9
AA-60-1	Mixed 1	49913.3	49977.2	50363.0	50167.2	49913.3	49913.3	49913.3
AA-60-1	Mixed 2	49672.7	49775.9	50151.0	49927.7	49672.7	49672.7	49672.7
AA-60-1	Mixed 3	49412.2	49582.7	49960.7	49744.5	49412.2	49412.2	49412.2

^a The optimal solution was verified using CPLEX.

6. Conclusions and future work

In this paper we have considered the problem of locating input and output locations in facility design in order to minimize material transportation considering both inter-departmental and intradepartmental flow patterns. While the problem can be modeled using math programming, this is not a viable method for solving realistically sized problems.

The proposed GA methodology is an effective method for solving this problem both with regard to the quality of the solutions that are found and the CPU time that is required. While the GA is not guaranteed to find the optimal solution, it consistently finds optimal or near optimal solutions in only seconds of CPU time. In addition to its good performance on this problem, a key advantage of the GA methodology is that it can be adapted to handle the problem

extensions and modifications mentioned below. Because the ultimate use of this algorithm will be as a subroutine to a block layout optimization, its computational effort is critical. We envision a hierarchical GA, where the parent, or upper level, GA searches through block layouts (as in Ref. [18]) and the GA described in this paper (the lower level GA) accepts those block layouts and optimizes the I/O and flowpath structure. The I/O detail is returned to the parent GA so that the entire facility design can be properly assessed for fitness.

In future work we can extend these ideas in several ways. First, the flow patterns of some departments may be known with certainty while those of others may not. Therefore, the flow pattern type for each department could also be determined during the optimization process. Second, we can consider the cost of adding additional material transfer points to a department. In some cases there may be little to no cost. For example, if material transfer into or out of a department consists of simply placing pallets on the floor then adding additional material transfer points only involves clearing enough additional floor space for the pallets. However, if lift tables or shuttle tables (for example for AGVs) are required then there may be some fixed cost associated with having additional material transfer points.

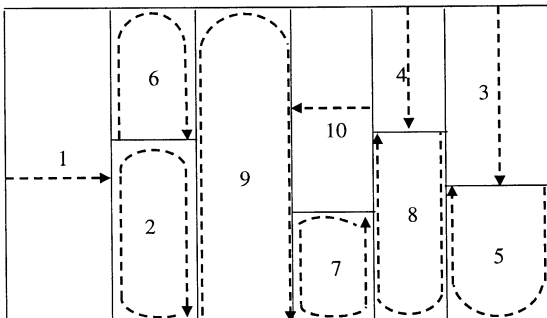
A third extension concerns the location of aisles in the floor plan. Aisle structure could be incorporated in several different ways. One way would be for the user to specify a general aisle structure and then the intradepartmental flow patterns would be constructed based on that aisle pattern. A second way would be to solve the I/O location problem prior to assuming any aisle structure and then based on these results determine what the aisle structure should be. Sensitivity analysis could be conducted by examining the material flow costs of using different aisle configurations and then a final aisle configuration could be selected.

Acknowledgements

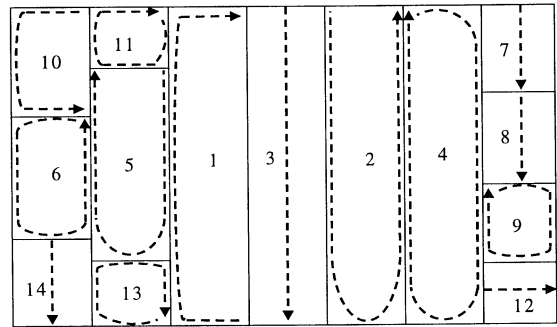
Part of this research has been supported by the US National Science Foundation grants DMI-9908322 and DMI-9908707. The authors would also like to thank Roseann Kuhns for her assistance in completing this paper.

Appendix A

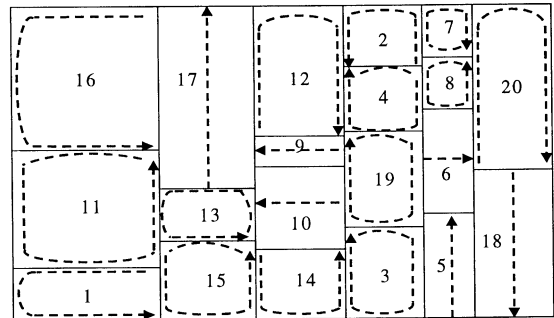
VC-10-01



BA-14-02



AB-20-02



References

- [1] Bos J. Zoning in forest management: a quadratic assignment problem solved by simulated annealing. *J Environ Manag* 1993;37:127–45.
- [2] Tompkins JA. *Facilities planning: a vision for the 21st century*. IIE Solutions 1997;August:18–19.
- [3] Meller RD, Gau K-Y. The facility layout problem: recent and emerging trends and perspectives. *J Manufacturing Syst* 1996;15:351–66.
- [4] Kusiak A, Heragu SS. The facility layout problem. *Eur J Oper Res* 1987;29:229–51.
- [5] Sinriech D. Network design models for discrete material flow systems: a literature review. *Int J Adv Manufacturing Technol* 1995;10:277–91.
- [6] Chhaged D, Montreuil B, Lowe TJ. Flow network design for manufacturing systems layout. *Eur J Oper Res* 1992;57:145–61.
- [7] Sinriech D, Tanchoco JMA. An introduction to the segmented flow approach for discrete material flow systems. *Int J Prod Res* 1995;33:3381–410.
- [8] Sinriech D, Tanchoco JMA. Design procedures and implementation of the segmented flow topology (SFT) for discrete manufacturing systems. *IIE Trans* 1997;29:323–35.
- [9] Montreuil B, Ratliff HD. Optimizing the location of input/output stations within facilities layout. *Engng Costs Prod Econ* 1988;14:177–87.
- [10] Sinriech D, Edouard S. A genetic approach to the material flow network design problem in SFT based systems. *Proceedings of the Fifth IE Research Conference*, 1996. p. 411–6.
- [11] Palliyil G, Goetschalckx M. A comprehensive model for the concurrent determination of aisles and load stations for aisle-based material handling systems. Technical Report, Georgia Institute of Technology, 1994.
- [12] Arapoglu RA, Norman BA, Smith AE. Locating input and output points in facilities design: a comparison of constructive, evolutionary and exact methods. *IEEE Trans Evol Comput* 2001 (in press).

- [13] Klein CM, Kim J. Location of departmental pickup and delivery points for an AGV system. *Int J Prod Res* 1996;34(2):407–20.
- [14] Benson B, Foote BL. Door FAST: a constructive procedure to optimally layout a facility including aisles and door locations based on an aisle flow distance metric. *Int J Prod Res* 1997;35:1825–42.
- [15] Harhalakis G, Lu T, Minis I, Nagi R. A practical method for design of hybrid-type production facilities. *Int J Prod Res* 1996;34(4):897–918.
- [16] Kane MC, Nagi R. Integrated material flow and layout in facilities design. *Proceedings of the Sixth Industrial Engineering Research Conference*, Miami, FL, May 1997. p. 919–24.
- [17] Chittratanawat S, Noble JS. An integrated approach for facility layout, P/D location and material handling system design. *Int J Prod Res* 1999;37:683–706.
- [18] Norman BA, Smith AE, Arapoglu RA. Integrated facilities design using a contour distance metric. *IIE Trans* 2001;33:337–43.
- [19] Davis L, editor. *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, 1991.
- [20] Holland JH. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [21] Goldberg DE. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley, 1989.
- [22] Michalewicz Z. *Genetic algorithms + data structures = evolutionary programs*. 2nd ed. New York: Springer, 1994.
- [23] van Camp DJ, Carter MW, Vannelli A. A nonlinear optimization approach for solving facility layout problems. *Eur J Oper Res* 1991;57:174–89.
- [24] Bazaraa MS. Computerized layout design: a branch and bound approach. *AIIE Trans* 1975;7:432–8.
- [25] Armour GC, Buffa ES. A heuristic algorithm and simulation approach to relative allocation of facilities. *Manag Sci* 1963;9:294–309.