ELSEVIER

Automation in Construction 9 (2000) 197–215

AUTOMATION IN CONSTRUCTION

www.elsevier.com/locate/autcon

Automated facilities layout: past, present and future

Robin S. Liggett [*]

Department of Architecture and Urban Design, UCLA, Box 951467, Los Angeles, CA 90095-1467, USA

Abstract

This paper reviews the history of automated facility layout, focusing particularly on a set of techniques which optimize a single objective function. Applications of algorithms to a variety of space allocation problems are presented and evaluated. Guidelines for future implementations of commercial systems are suggested. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Automated space layout; Floor plan layout; Facility layout

1. Introduction

With the growing demand for computerized facilities planning and management, there is the potential for automated space layout products to play a more significant role. As interest in such products rekindles and develops it seems appropriate to take another look at the relatively long history of approaches to automated facility layout. The author last reviewed the field (which had its origins in the early 1960's) in 1985 presenting an overview of alternative approaches to the layout problem and solution algorithms [39]. Since that time, commercial products have become available based on some of these original algorithms and on the research side new solution techniques such as simulated annealing and most recently genetic optimization have been applied to the problem.

Facility layout is concerned with the allocation of activities to space such that a set of criteria (for example, area requirements) are met and/or some objective optimized (usually some measure of communication costs). This paper reviews alternative formulations of the problem (e.g., how space is represented and methods of evaluating a plan) as well as existing solution algorithms. It identifies the specialized applications for which algorithms seem particularly useful as well as the particular needs of facilities layout that must be considered when applying algorithms.

Some commercial facilities management systems currently incorporate automated algorithms (usually within an interactive framework) to solve facilities planning problems of stacking and blocking activities. This paper also looks at the limitations of current commercial space allocation products and proposes a set of key requirements for implementing the next generation of such systems.

[*] Fax: +1-310-376-4936; e-mail: rligget@ucla.edu

2. Overview and history

Facility layout problems range in scale from the assignment of activities to cities, sites, campuses or buildings, to the location of equipment and personnel groups on a single floor of a building. A layout problem can surface in the design and allocation of space in a new building or the reassignment of space in an existing building. During the conceptual design phase, allocation of space within a new building can be used to test alternative options for building configuration. Plans can be evaluated with respect to best use of space in order to determine such things as the optimal number of floors, perimeter of the plan, etc. In an existing building, layout tools can be used for the on-going problem of space management. For example, as project groups increase or decrease in size, how should employees be located within an office so that group contiguity is maintained with a minimum number of workspace moves? How can unused space be consolidated effectively to minimize lease costs? More complex problems can involve issues of time-phased layouts based on projected changes in space needs [46].

Since the early 1960s numerous computer programs have been developed for the automated solution of such spatial layout problems. The objectives and scope of these programs have varied widely. Interest in this area has come from computer science/engineering researchers primarily looking at problems of plant or production facilities layout (or at the micro-scale, the layout of electronic circuits) as well as from architects and interior designers interested in the design of large facilities such as office buildings, universities, hospitals, or department stores. More recently there has been interest on the part of facilities managers concerned with reuse and rearrangement of space.

Most of the research and development has focused on what is known as the floor plan layout problem, the physical arrangement of space on a plan (referred to as a block plan). There are, however, other applications of the space allocation problem—for example, an important commercial application has been the assignment of activities to multiple floors of a building (known as the stack plan problem). Approaches to spatial allocation problems differ in terms of the type of problem addressed as well the criteria used to generate, compare and evaluate solutions.

3. Representation of space

All space planning problems consist of a set of activities to be located and a space in which to locate them. Space can be represented in different ways, thus providing a method of classifying alternative types of layout problems:

- Space as discrete objects (one-to-one assignment problem).
- Space as area (many-to-one assignment problem, for example, a stacking problem).
- Space as area and shape (blocking or floor plan layout problem).

Both the problem formulation and solution techniques are impacted by the way activity and physical space are represented.

The simplest layout problem is the assignment of a set of discrete activities to a set of discrete locations in such a way that each activity is assigned to a single location. This is called a one-to-one assignment problem (also known as an equal area layout problem) and has some very interesting applications on both the micro and macro level. For example, the assignment of buildings to sites or the assignment of employees to preexisting offices or work stations can be a one-to-one assignment. The issues of size and shape do not enter into the layout process.

Generally space planning applications are not as straightforward as one-to-one assignment. The areas required by activities are not necessarily equal, so it is not feasible to match activities and locations on a one-to-one basis. When assigning employees to existing offices, we might want to consider multiple occupants. In stacking plan problems (the assignment of activities to floors in a multi-storey building), more than one activity can be assigned to a single floor or a single activity can occupy multiple floors (many-to-one or one-to-many assignment). How the area of an activity is apportioned among floors can be an important consideration in generating and evaluating a plan. In both of these examples, however, activity size is still a relatively simple issue as actual activity shapes are not considered.

The most difficult problems to represent are those at the block-plan level. An activity is represented as a polygon on the plan. This polygon should be able to take on any shape and location while maintaining the required activity area. The method for handling unequal areas has a significant impact on the solution approach taken.

4. Approaches to automated layout

Automated space allocation algorithms require some method of evaluation in order to guide the layout process. There are three major paths that solution techniques have followed. The first involves the optimization of a single criterion function; specifically the minimization of costs associated with communication or flow of materials between activities. While there are numerous drawbacks to such approaches, they have quite widespread application with respect to types of plans that can be generated. This paper will focus primarily on this class of solution techniques.

A second path is based on a graph theoretic approach. It is concerned primarily with generating a layout that meets adjacency requirements between activities. This approach requires the construction of a planar adjacency graph where nodes represent activities to be located and edges (or links) represent a direct adjacency requirement. The dual of a planar graph determines the layout of the facility. A planar graph with its corresponding dual is displayed in Fig. 1. There is a long history of graph theoretic applications to the layout problem (see Grason's early work for example [20]). Many approaches which follow this path are based on Richard Muther's Systematic Layout Planning methodology [48], the most frequently used plant layout design methodology of the last 30 years. Muther's methodology results in the generation of a space relationship diagram that is considered a 'design skeleton' from which a layout can be generated. While graph theoretic approaches differ from those that optimize a single criterion, some of the heuristic solution techniques are similar. These will be noted later when such methods are discussed.

Approaches following a third path are not concerned with optimizing a single measure or value,

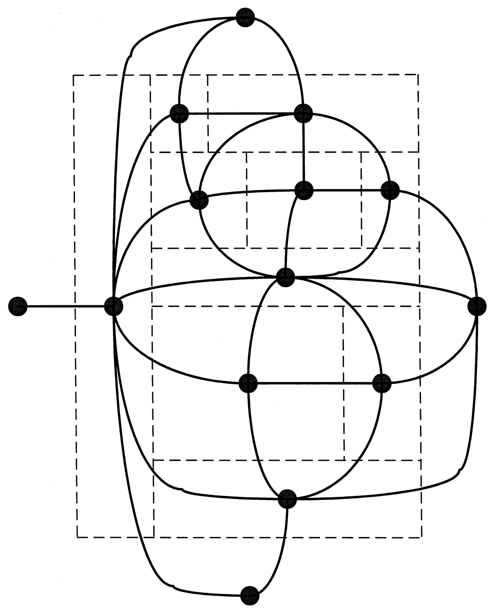
Fig. 1. A planar adjacency graph with corresponding dual [16].

but with finding an arrangement that satisfies a diverse set of constraints or relations. In this case the major criterion is feasibility. Early examples of this path are Eastman's General Space Planner [12] and Pfeffercorn's Design Problem Solver [51]. Both methods design layouts by placing objects so that they satisfy a set of constraints which involve such factors as position, orientation, adjacency, path, view, or distance. A more recent system of this type is the layout module of SEED [15], a software system to support the early phases in building design under development at Carnegie Mellon. This module generates schematic layouts of rectangular space under various constraints that include access, natural light and privacy. It is based on Flemming's LOOS/ABLOOS system [14]. Two other methods with a similar representation but with significantly different approaches to constraint satisfaction, HeGeL/HeGeL-2 [2] and WRIGHT [6], were also developed at Carnegie Melon (see Akin [3] and Flemming et al. [14] for a comparison of these three systems).

While to date this type of approach has not been the basis for commercial software, newer systems such as SEED have considerable potential, particularly as interactive aids to the layout of schematic

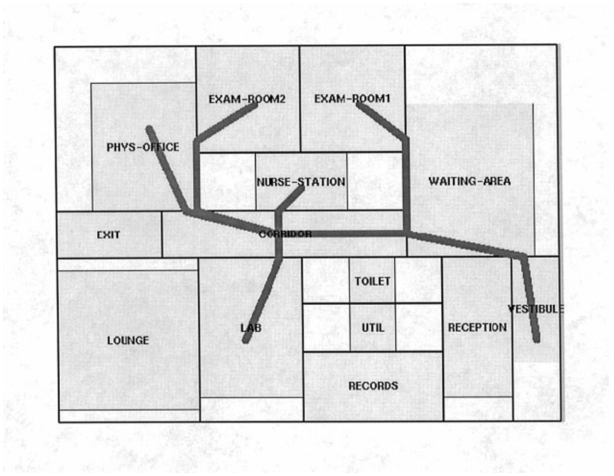

Fig. 2. Display of access paths in a SEED layout [15].

plans in the design development phase of new facilities. Some advantages of these systems are that they consider multiple criteria, maintain acceptable activity shape, and can usually handle issues of circulation space. Sample output from SEED with access paths displayed is shown in Fig. 2. A disadvantage of such systems is that they have not yet demonstrated the ability to handle large scale problems that are encountered in actual practice.

While a more detailed exploration of solution techniques for layout problems presented in the remainder of this paper focuses primarily on the first class of problems (the optimization of a single criterion function), solution methods of the other classes of problems will also be discussed as they relate to the overall classification framework.

5. One-to-one assignment: An early formulation as A Quadratic Assignment Problem

One of the most popular approaches to automated facility layout was first formulated by Koopmans and Beckmann [34] for problems concerned with the assignment of manufacturing plants to sites such that the cost of transportation of the flow of goods between plants is minimized. Known as the Quadratic Assignment Problem (QAP), it is concerned with finding optimal locations for a set of interrelated objects. The problem can be described as follows.

Consider the assignment of N activities to N or more sites, each of which can accommodate one and only one activity. Associated with each pair of activities (i,j) is a measure of interaction $Q(i,j)$ (e.g., intensity of communication, level of traffic, etc.). Associated with each pair of sites (k,l) is a measure of spatial separation $C(k,l)$ (e.g., distance, travel time, etc.). In addition, a fixed cost, $F(i,k)$, may be associated with the placement of activity i in site k. If $A(i)$ denotes the site to which activity i is assigned in a mapping A of activities to sites, the total cost of a mapping (solution) can be given by:

$$\mathrm{Cost}(A) = \sum_{\text{activity } i} F(i, A(i)) + \sum_{\text{activity } i} \sum_{\text{activity } j} \left[Q(i,j) C(A(i), A(j)) \right].$$

The objective is to find a mapping A, such that this cost function is minimized. The two parts of the function are known as the fixed cost term and the interactive cost term. The fixed cost term (which is equivalent to the criterion function of a linear assignment problem) is concerned with the costs of assigning a particular activity to a particular location. Fixed costs might represent rent, special facilities construction requirements, or some measure of preference for a particular site. The second term, which is the quadratic portion of the objective function, introduces costs caused by the interdependence of assignments. Interactive costs may represent a subjective judgment about grouping requirements or an actual measure of flow of goods or employees.

The floor plan layout problem was first formulated as a quadratic assignment problem by Armour and Buffa in 1963. They considered the layout of a manufacturing plant where the criterion to be minimized was the cost of product flow between departments. Their work resulted in a computer program called CRAFT (Computerized Relative Allocation of Facilities Technique) [9].

In the quadratic assignment formulation, floor plan layout can be viewed as a combinatorial problem in which indivisible activities (e.g. departments or individual employee work stations) are to be assigned to fixed locations on a plan. In principle, it is possible to solve this problem by exhaustive enumeration of all possible ways of assigning activities to locations, and by selection of a plan which satisfies given constraints and/or (as in the case of the quadratic assignment formulation) yields the minimum value for the criterion function. In practice this turns out to be infeasible for problems of realistic size (problems of over 15 activities) since the number of activity/location combinations involved is so vast.

It can be shown that quadratic assignment problems belong to a class of mathematical problems known as NP-complete. It is generally accepted that the efficient solution of NP-complete problems is impossible in principal. However a number of good approximate solution strategies do exist that produce high quality solutions to realistically sized problems at acceptable cost. This is particularly true for the special case of the one-to-one problem. As mentioned previously, one-to-one formulations have a number of commercial applications in terms of the assignment of activities to preexisting offices and workspaces. Fig. 3 shows a typical office plan where space can be allocated on a one-to-one basis. Links drawn on the plan represent adjacency requirements between activities assigned to rooms.

We will first review a number of algorithms which operate efficiently at the one-to-one level before considering the complexity added due to the unequal area requirements inherent in floorplan layout problems.

6. Solution procedures

Existing approximate solution strategies can generally be classified into two categories: constructive initial placement strategies and iterative improvement strategies. A constructive initial placement strategy locates activities one by one, building a

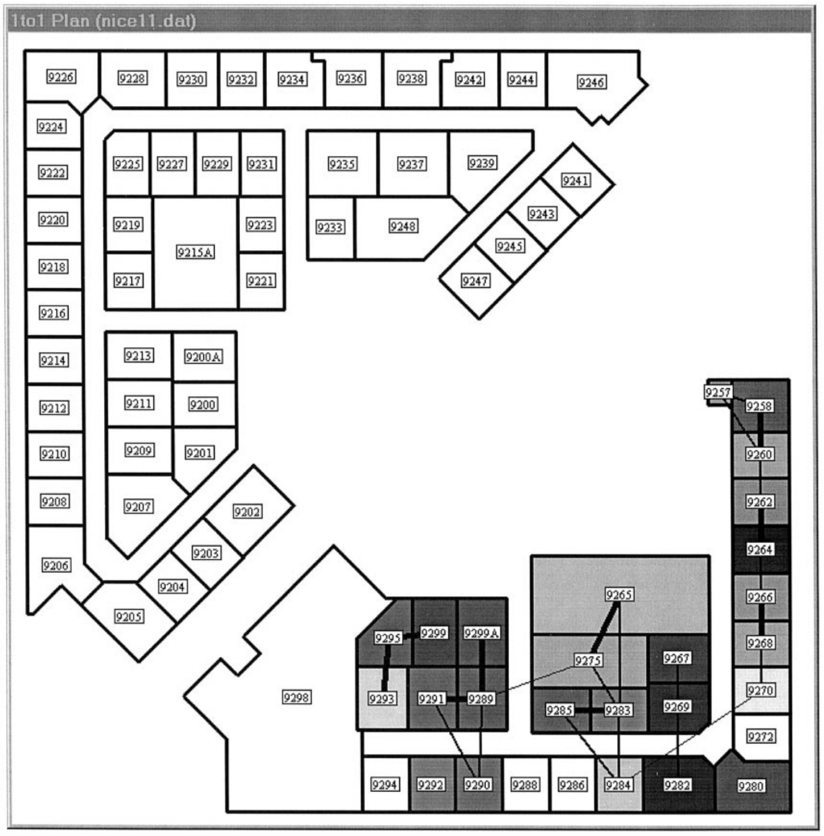

Fig. 3. One-to-one office layout.

solution from scratch in a step-by-step fashion. An iterative improvement (also known as hill climbing) strategy begins with some initial arrangement and attempts to improve it incrementally. Simulated annealing, can be viewed as a variant of an iterative improvement strategy. More recently, genetic algorithms have become of interest for the solution of combinatorial problems such as the QAP. Genetic algorithms begin with a set of possible solutions and use mutation and crossover techniques to evolve existing solutions into better solutions. The next sections will briefly cover basic improvement and constructive techniques (see Refs. [33,37–39] for a more detailed description and comparison of early techniques) and then focus on newer approaches which include simulated annealing and genetic algorithms.

6.1. Constructive procedures

Constructive procedures build a solution from scratch using an *n*-stage decision process. Some methods attempt to automate a set of 'rules of thumb' for making intelligent assignments at each stage, essentially modelling the thought process of a human designer. A simple activity selection rule might be: select the activity which has the highest connectivity to any activity already placed. A location can then be selected, either by a simple rule of thumb again (e.g., select the first empty location adjacent to a placed activity starting at the top left and working clockwise), or by more sophisticated criteria such as selecting that location which yields the minimum value of the criterion function considering only the activities already placed. (See Refs. [11,42,49] for early constructive procedures).

The constructive decision process can be viewed as a 'tree search' where at each branch the selection of an activity–location assignment is made. Two more sophisticated and computationally intensive approaches use mathematical bounds on the decision tree to guide the process. At each stage of the decision tree, Gilmore [18] and Hillier [25] calculate a lower bound for the objective function for each branch. The activity–location pair which yields the minimum lower bound is selected for the next assignment. Graves and Whinston [21] use probability theory to calculate the expected value of the objective function for each possible activity–location assignment at any step in the decision tree, and select the assignment which seems most likely to lead to an optimal solution.

Constructive methods generally adopt either a 'local' or 'global' orientation to a problem. Local methods consider only the assignments which have already been made; they tend to be less expensive but yield poorer solutions. Global techniques attempt to account for possible future moves in the evaluation of a particular assignment and, although more expensive, generally produce better solutions. Both the Gilmore–Hillier and Graves–Whinston algorithms are examples of global techniques.

6.2. Improvement procedures

Improvement procedures start with a single solution and attempt to incrementally improve it. The simplest version is the 'pair-wise' exchange. Starting from an initial solution, the procedure consists of systematically evaluating possible exchanges between pairs of activities and making an exchange if it improves the value of the criterion. There are a number of variants on the basic pair-wise exchange which focus on reducing the computational effort expended or on improving the quality of the solutions generated. These variants generally involve the method of selection of activities for possible exchange and which exchange to make (e.g., whether or not to make the first exchange that leads to an improvement or to evaluate all possible exchanges and select the exchange that results in the maximum cost improvement). Since this latter method can be very costly with respect to computation time, methods use different ways to limit evaluation of exchanges. For example, Elshafei [13] only evaluates all possible moves of a single activity. Hanan et al. [22] limit exchanges to immediate neighborhoods or activities. Volmann et al. [59] only considered the exchange of the two activities which contribute the most to the total cost of the current solution at each step. Other methods attempt to be intelligent about the order in which they evaluate potential exchanges [25,26]. A more expensive approach termed 'biased sampling' selects randomly from the set of possible exchanges showing improvement. The probability of selection associated with each exchange is proportional to its corresponding cost reduction [50]. Results have also been reported for experiments involving three-way, four-way and five-way exchanges [7,42], however, the minor improvements to solutions have generally not balanced the additional cost of the generation process.

Improvement procedures usually converge on local optima. Since all improvement procedures require an initial solution, a number of local optima can be generated and compared by using different starting configurations. Elshafei [13] experimented with a technique to retreat from a local optimum by selecting the move which results in the minimum cost increase. The exchange cycle is then repeated using this position as the starting solution which hopefully leads to a new local optima. A more recent solution technique taken from the area of statistical mechanics, simulated annealing, has been used to 'back out' of unattractive local optima.

6.3. Simulated annealing

Simulated annealing techniques eliminate many disadvantages of improvement (hillclimbing) methods. Solutions no longer depend on the starting point and are more likely to converge on a good (if not optimal) solution. The main departure from traditional improvement methods is that changes accepted at each stage of the optimization can actually increase the cost of the plan. In most hillclimbing methods, a new solution is accepted only if it yields an improved value of the objective function. In simulated annealing, an exchange can also be accepted if the probability of the resulting cost increase occurring is lower than a control parameter. This technique is derived from a method that simulates the cooling of a mass of vibrating atoms from a high temperature T. The probability of acceptance (p) of the exchange of a pair of activities equals one if the exchange provides a better value of the objective function. If, however, the cost change is positive (i.e., increases the cost), the probability of acceptance p is a function of the difference in values of objective function for the current solution and the new solution (Δ), and an additional control parameter, T (which represents temperature in the actual annealing process): $p = (\exp(-\Delta/T))$. In general, the lower the temperature T is, the smaller the chances for the acceptance of a new solution are. During execution of the algorithm, the temperature of the system, T, is lowered in steps. The algorithm terminates as T approaches zero. The use of simulated annealing techniques makes it less likely to fall into local optima, provided the annealing process is long enough.

Sharpe and Marksjo [53] show how an implementation of the simulated annealing method provides a relatively simple but powerful approach to facility layout optimization. They have applied it successfully to large scale problems with up to 200 locations. An additional advantage is that it can produce a number of near-optimal solutions from which designers can select. Similar advantages are shown by the author of a program called CLASS, Computerized Layout Solutions using Simulated annealing [30].

6.4. Genetic algorithms

Somewhat related to improvement procedures are a class of algorithms which rely on analogies to natural processes. This type of algorithm has been described by Michalewicz [45] as follows:

> Genetic algorithms are based on the principle of evolution (survival of the fittest). In such algorithms a population of individuals (potential solutions) undergoes a sequence of unary (mutation type) and higher order (crossover type) transformations. As these individuals strive for survival: a selection scheme, biased towards fitter individuals, selects the next generation. After some number of generations, the algorithm converges with the best individual hopefully representing the optimum.

Genetic algorithms share the following features (see Tate and Smith [56]):

- An initial population of solutions (can be randomly generated)
- A mechanism for generating new solutions by combining features from solutions in the existing population (reproduction).
- A mechanism for generating a new solution by operating on a single previously known solution (mutation).
- A mechanism for selecting the set of solutions from the population(s), giving preference to those with better objective function values (selection) and

- A mechanism for removing solutions from the population (culling).

Solutions can be selected to mutate or to reproduce. Selection is performed with a bias towards choosing the better solutions in the current population. For the facilities layout problem mutation can take the form of some variant of the pair-wise exchange. The key feature of the reproduction process used by Tate and Smith [56] is that any activity assigned the same location in both parents will occupy that location in the offspring. For the remaining locations, activity assignments are chosen randomly from one or the other parent. Any unassigned activities are then matched with the remaining unassigned locations. As children are created, solutions with the poorest values of the objective are eliminated (culled) to keep the population the same size. As for simulated annealing, excellent results have been reported for this type of algorithm and there is considerable interest in investigating further extensions. Any of the reported improvement procedures are candidates for mutation processes and there are numerous possibilities for reproductive transformations.

Gero and Kazakov [17], extending earlier work by Jo and Gero [29], take advantage of what they term 'superior' evolved genes. Solutions, called genotypes, are represented as a linear sequence of integers which can be interpreted as the order activities are placed on the plan. After a fixed number of generations, the top ten percent of genotypes (with respect to a specified objective) and the bottom ten percent of genotypes are searched to identify groups of genes that occur together (compact subsets of activities that appear together in the sequence). Compact gene groups found almost exclusively in the best solutions and almost never in the poorer solutions are declared new 'superior' evolved genes. These evolved genes are then represented as a single gene for further reproduction. Applications of this approach show excellent results for an office and a hospital layout problem found in the literature. The authors have also shown that evolved genes tend to represent design features that can be re-used later in similar layout problems.

Genetic search methods climb many peaks in parallel making it more likely to settle on a global or near-global solution than constructive or improvement procedures. Genetic algorithms are also attractive as the mutation and reproductive mechanisms can be visualized in terms of human design processes. We can conceive of a designer working with a set of possible solutions which evolve toward one or more preferred solutions. As these solutions evolve there may be portions of the design which are common to the best solutions. These are preserved when design options are combined to generate new solutions.

It is logical from a human perspective to think of the solution process as a hybrid of a number of approaches. This is also true for automating the layout process and there are a number of hybrid solution techniques.

6.5. Hybrid approaches

Since both iterative improvement (which includes simulated annealing) and genetic algorithms require initial solutions, it is generally preferable to begin with a reasonable solution rather than one which is randomly generated. On the other hand, while constructive procedures can produce good solutions, there is almost always room for improvement. Using a constructive procedure to generate an initial solution should reduce the number of iterations of the improvement procedure and improve the quality of the solution generated [38]. The coupling of a constructive procedure with an improvement procedure provides an effective combination of a global and local approach to a problem. The constructive procedure sets the general tone of the solution while the improvement procedure refines the details.

The author has shown good results with a hybrid approach which combines the Graves–Whinston constructive procedure with a pair-wise exchange improvement algorithm [37]. More recently Huntley and Brown [27] have combined a high-level genetic algorithm with a simulated annealing algorithm. Jo and Gero [29] improve upon Liggett's [39] solution to an office layout problem by using it as an initial population for their genetic search algorithm, EDGE (Evolutionary Design based on Genetic Evaluation). Heragu and Alfa [24] show that a hybrid method which uses a modified penalty algorithm to generate an initial solution which is then improved using simulated annealing produces superior results when compared to a two or three-way exchange procedure or just to simulated annealing alone. Kaku et al. [31] propose a hybrid heuristic that consists of three parts. In the first part, several partial assignments are generated for use as starting points for a constructive heuristic (a breadth first search tree is used to enumerate a set of good partial assignments). In the second part, these starting points are used to construct complete assignments. They experimented with both the Gilmore [18] and Graves–Whinston [21] constructive approaches for completing the solution. Finally, attempts at improving the constructed solutions are made by the application of both pair-wise and triple-exchange routines.

7. Unequal areas

Most of the approaches reviewed above can be used to generate acceptable solutions to the one-to-one assignment problem. However, space planning problems are generally more complex than the classical quadratic assignment formulation due to the imposition of activity area requirements. Since areas required by activities are not necessarily equal, it is not always feasible to match activities and locations on a one-to-one basis. Note, this is not true if location perimeters (e.g., existing offices) have been predefined since the assignment of activities to locations can be limited by size constraints. Such constraints are generally handled in the quadratic assignment formulation with the fixed cost function. Fixed costs can be set prohibitively high for the assignment of activities to offices of unacceptable size. At the block plan level, however, the problem becomes more difficult since we are locating activities on a plan where location boundaries have not been prefixed.

7.1. Modular approach

A typical approach to the unequal area block plan problem (for the class of algorithms discussed above) is to partition the plan into equal size modules. Each activity is then partitioned into modules of the same size according to required floor area. The problem is then one of assigning activity modules to location modules in a one-to-one fashion. For the quadratic assignment problem, artificially high interaction (adjacency) values can be specified between modules of the same activity to encourage contiguity of activity space. Heuristics in constructive procedures can also be used to ensure contiguity of space. For example, after the initial module is placed for an activity in the Liggett implementation of the Graves–Whinston algorithm [41], locations evaluated for the placement of subsequent modules of the same activity are limited to adjacent modules. The order of location module selection is based on the expected value of the objective function. For constructive procedures there is no guarantee, however, that activities will not be split (unless backtrack strategies are employed at perhaps a prohibitive cost) and modularization can produce irregular shapes that are undesirable or unworkable (see Fig. 4 for the effect of modularization on a block plan).

Some approaches do not require (or allow) a predefined plan perimeter making it easier to control activity shape. A key feature of a constructive procedure, SHAPE [23], is the order candidate location modules are considered. The shape of the layout evolves as departments are placed in the modules surrounding those already occupied. The collection of modules adjacent to a partial layout constitutes a potential activity location. The set of modules that minimizes the current value of the objective function is chosen for activity assignment. Earlier versions of this type of program include CORELAP [35] and ALDEP [52]. While such approaches guarantee activity contiguity, they are not generally effective given existing plan perimeters. These programs can, however, be valuable tools for exploring variations in the size and shape of a building during the schematic design phase. For the algorithms that require preset perimeters, layouts free from shape constraints can be generated by specifying a large square perimeter or a perimeter the size and shape of the site.

Buffa and Armour's early approach to the unequal area floorplan layout problem limited the exchange of activities to pairs of equal area [9]. This constrains the problem significantly in terms of exploration of the possible solution space. Other improvement procedures exchange individual activity modules, but must include constraints to maintain contiguity of activity shape. Exchange procedures can also be used to improve shapes resulting from constructive procedures.

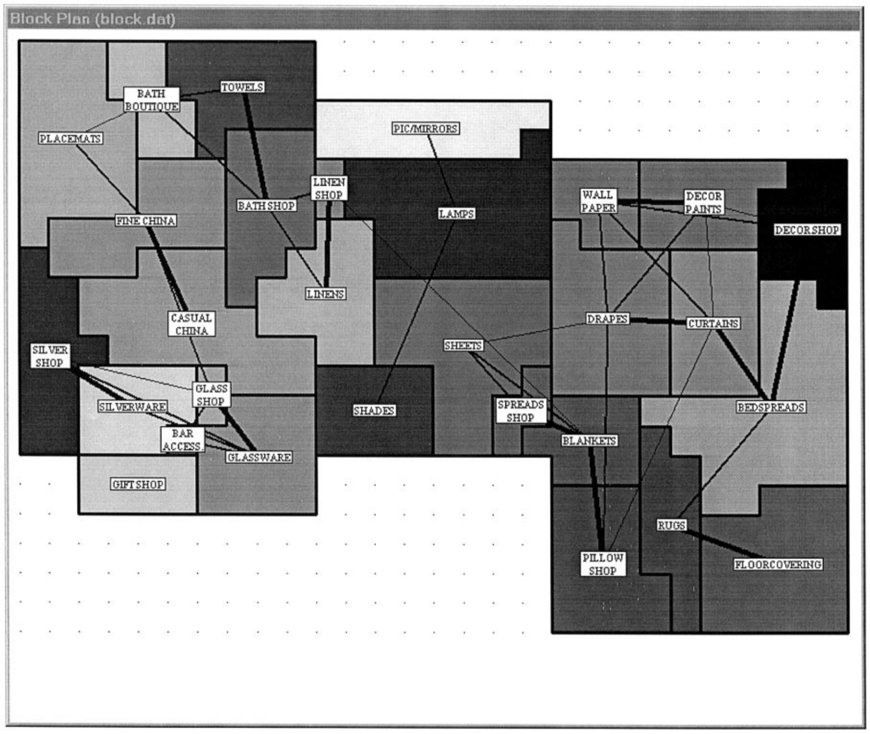

Fig. 4. Effect of modularization on a block plan.

The method of plan modularization has important implications for the automated algorithm. If an existing building has not been designed on a square planning grid or if the perimeter has other than right angled edges, the modularized plan will only approximate the actual perimeter. The smaller the grid size, the better the approximation of the existing plan. However, a small grid size increases the number of modules. This leads to increased computation time in generating a layout. It may also cause fragmented activity shapes.

7.2. Arrangement of rectangles

Other approaches to the unequal area layout problem partition rectangular shapes into smaller rectangles for assignment. Tam [54,55] represents the layout as a slicing structure that is constructed recursively by partitioning a rectangular block. Each rectangular partition in the slicing structure corresponds to the space allocated to an activity. Tam uses cluster analysis to generate a slicing tree which places activities with high interactions in close proximity to each other. The leaves on the tree represent activities to be placed, and interior nodes represent the slicing operation (left cut, right cut, bottom cut or top cut as shown in Fig. 5). A simulated annealing procedure is used to exchange slicing operators in the tree, thus generating different rectangular partitioning schemes. The objective function used to drive the annealing process includes the quadratic term of the QAP and a penalty term for geometric constraints. Tam shows good results with 20 and 30 activities.

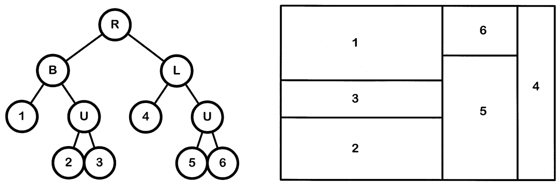

Fig. 5. Tam’s slicing free and associated layout [54].

Tate and Smith [57] use a genetic algorithm with a flexible bay structure. The prespecified rectangular area is divided in one direction into bays of varying width. Each bay is then divided into rectangular partitions of equal width but different lengths. The bays are flexible in that their widths will vary with their number and contents. In the Tate and Smith equal-area genetic algorithm discussed earlier [56], a solution was represented by a permutation of integers (one through n, where n is the number of activities) and the locations of the integers in the sequence correspond to locations on the plan. In this implementation, the locations correspond to bay partitions and an additional sequence is required to represent a solution. This second sequence defines the number of bays and the number of partitions per bay by indicating where breakpoints exist. Changing the location of break points can change the number of rectangles per bay and/or the number of bays. Solutions are evaluated and compared using the objective function of the quadratic assignment problem with the addition of adaptive penalty functions to handle area and proportion requirements for activities.

Van Camp et al. [58] use a nonlinear model. Activities are considered to be of fixed area but of variable dimensioned rectangular shape. Using a similar measure as the QAP, the objective function represents the cost of flow of material between activities multiplied by the distance between activity centroids. An additional term measures distance between activities and the outside wall. Constraints require that no two departments overlap, that departments must be contained in the facility and meet area as well as shape constraints (again using the penalty function approach). Nonlinear optimization is used to generate an initial feasible solution (note activity centroids and dimensions are viewed as continuous variables which can vary over the dimensions of the exterior boundaries). The nonlinear optimization generally results in local optima. The procedure then applies a pair-wise exchange improvement procedure to reduce the value of the interactive portion of the objective function. At this stage infeasible solutions (where an activity is too large for the available location) are accepted. A return is then made to the nonlinear algorithm to generate a new feasible solution.

8. Constraint based methods for unequal area layout

Methods which do not focus on a single objective but rather seek to generate solutions that meet a number of different (possibly conflicting) constraints generally deal with the unequal area layout problem in terms of arrangements of rectangles. Three methods mentioned earlier, LOOS [14], WRIGHT [6] and HeGeL [2], all formulate the problem as one of arrangement of rectangles with sides parallel to axes of an orthogonal system. They all attempt to satisfy two types of constraints: one set that is dependent on the structure or topology of the problem such as the requirement that the rectangles not overlap and fit within a given boundary; and a second set of constraints which are independent of structure and consider attributes such as area, dimension, orientation, and adjacency requirements.

LOOS and HelGeL are hierarchical generate-and-test methods that incrementally construct solutions by adding one rectangle at a time to a partial solution, testing for constraint satisfaction at each step. LOOS [14] employs a breadth first search, generating candidate solutions at each stage by enumerating all possible ways of adding a new rectangle to a partial solution. Each intermediate state is examined for constraint violation. If a structure dependent constraint is violated the intermediate solution is pruned. A count of other constraints violated is made for any partial state and only those with the lowest counts are selected for expansion. By sequentially expanding partial solutions and pruning less promising candidates, LOOS commits to a set of current globally best candidate solutions and avoids backtracking.

HeGeL/HeGeL2 [2] also constructs a solution in a step-by-step fashion, however, it follows a depth-first search. The solution procedure is based on a protocol analysis of the problem-structuring and problem solving behavior of designers. Layout requirements are expressed as relationships between objects to be located. These relationships, which are called 'predicates', are used as generative constraints or evaluative criteria. A generative predicate results in the selection of a design unit to be placed (for example, a direct access requirement). Alternative locations for the design unit are then generated based on the predicate. These locations are tested against the other predicates associated with the design unit. If none of the generated locations meets the test criteria, HeGeL will backtrack. If a single location meets the criteria, the placement is made. If there are multiple locations, they are presented to the user for selection. The user selects a final location and others are stored for possible later backtracking. While the original approach relied on the user to direct the search in terms of selection of the next generative predicate or unit to place, HeGeL2 implements an optimization methodology to make the best placement decision at each stage.

WRIGHT [6] implements a constraint-directed search called disjunctive constraint satisfaction in which constraints are incrementally satisfied. The layout is represented using algebraic equations and inequalities of variables that represent the border lines, dimensions, areas and orientation of the design units. These are called atomic constraints. Disjunctive constraints are Boolean combinations of atomic constraints and specify the ''structural alternatives considered by WRIGHT for satisfying the constraint.'' For example, a disjunctive constraint might represent the four alternatives of placing one unit directly adjacent to another (it can be north, south, east or west of the second). Solutions are generated by sequentially instantiating disjunctive constraints and solving the current constraint satisfaction problem (CSP). A CSP is consistent if there exists values for all variables that simultaneously satisfy all constraints. If a CSP is inconsistent, the propagation algorithm backtracks and selects an alternative disjunct. The full solution process will find all significantly different solutions. Since a problem can be underconstrained or overconstrained, the design process is characterized by the addition or relaxation of constraints by the designer.

Rather than formulating competing criteria as a diverse set of constraints, Jacobs [28] combines them into a single weighted objective function. The criteria considered include the distance between design units with respect to frequency of interaction (the typical single objective function criteria considered by the first class of solution techniques), as well as direct adjacency requirements. He also includes criteria related to the alignment of spaces with a goal of keeping the structure of the layout as simple as possible. A final consideration is the use of space. Designers can choose between minimizing empty space (making the most use of space) or maximizing empty space by compacting the layout. Jacobs uses a two-stage solution procedure where each stage combines both a constructive and improvement procedure. In the first stage units are ranked for order of placement by boundary preferences, adjacency preferences, and the distance measure. A solution is then constructed incrementally based on the priority ordering by generating all feasible locations for the next design unit to be placed. A location is selected for placement based on the objective function. If a unit will not fit on the layout, all units are removed and the process is restarted. Once a solution is constructed it is improved with a pair-wise exchange procedure. Since there is an element of randomness in the initial placement ordering as well as in the selection of a location for placement (if two locations yield the same value for the objective, the selection is made randomly), a different solution will result each time the process is restarted.

9. Graph theoretic approaches

Graph theoretic approaches also handle the unequal area block plan. In these approaches a block plan is constructed as the dual of a planar graph where nodes represent spaces and links represent required adjacencies. While it is always possible to construct a block plan from a planar graph which meets the given adjacency requirements between spaces and between spaces and the outside area, the resulting plan may not meet size and shape requirements imposed on each space. Constructing a block plan that meets size and shape requirements is a nontrivial problem.

A graph theoretic approach is a two stage process. In the first stage a planar graph which corresponds to the adjacency requirements is generated. A planar graph is one which can be drawn so that no two edges intersect. A planar graph is maximal if no edges can be added without losing planarity. It is possible that the adjacency requirements cannot be represented by a planar graph. In this case the problem is overconstrained and the solution procedure becomes one of generating maximal planar graphs which maximize the number of adjacencies or the weighted adjacencies met by the graph. Once a graph has been formed, the second stage involves generating the actual layout. See Baybars and Eastman [5] and Foulds [16] for a more detailed discussion of the graph theoretic approach and early applications.

A typical graph theoretic heuristic for the layout problem consists of the following steps:

Stage 1: Generating a planar graph

- Form a weighted graph of the relationships between facilities
- Identify a maximal planar subgraph of relatively high weight

Stage 2: Generating a block plan

- Construct the dual from the planar subgraph. The dual represents a layout apart from the fact that shapes and areas have not been taken into account.
- Attempt to accommodate shapes and areas in forming a block plan from the dual.

Much research focuses on just the first stage of the process-generating a maximal planar graph. A planar graph can be generated by adding or subtracting edges following a step-by-step process. Typically edges are added in a greedy fashion (local constructive method) where planarity is tested after each edge addition. Leung [36] presents such a constructive procedure which capitalizes on the fact that triangulated graphs are maximally planar. The method starts with a planar subgraph which is generated by enumerating all possible groups of four vertices. The group with maximum weight is selected. At each subsequent step either a single vertex or a triple of vertices which maximizes the additional edge-weight per vertex is added to a face. Once a maximal planar graph is constructed, many methods apply some type of improvement procedure such as pair-wise exchange or simulated annealing to improve the adjacency score while maintaining planarity.

By requiring a hexagonal structure for the adjacency graph, Goetschalckx [19] has developed an efficient method for generating a rectangular block plan that meets area requirements from the dual of a planar graph. The rectangular floor plan is divided into rows based on the number of rows of the hexagonal graph. Each row in the plan is then partitioned into the number of spaces in the corresponding row of the hexagonal graph. A feasible layout with respect to area is determined by adjusting the height of a row. Fig. 6 shows a hexagonal adjacency graph with the resulting layout. Goetschalckx's algorithm, SPIRAL, has been implemented in a commercial product called FactoryOPT by CIM TECHNOLOGIES of Ames, Iowa. Montreuil et al. [47] uses a linear programming model to generate a block layout of rectangular spaces from a planar adjacency graph. A limitation of both Goetschalckx's and Montreuil et al.'s approach is that the building perimeter must be rectangular.

The graph theoretic formulation differs from the traditional quadratic assignment approach to the layout problem in a number of ways. The fundamental difference is that the graph theory approach considers only direct adjacency requirements. No consideration is given to nonadjacent pairs of facilities with respect to communication costs, even if they are relatively close together on the plan. In addition,

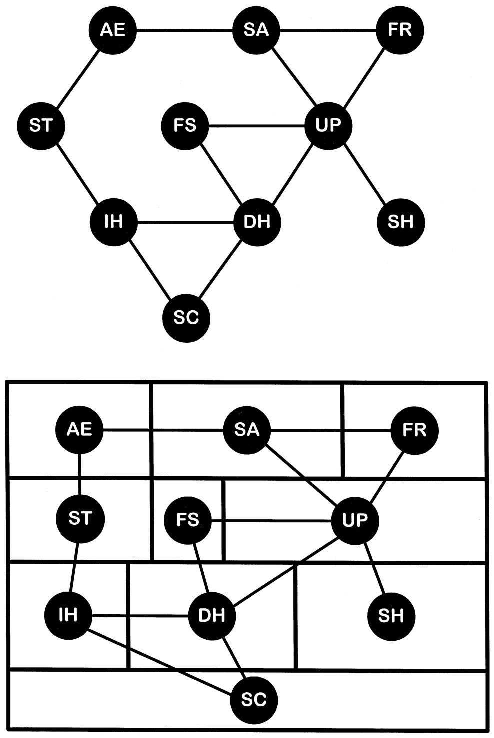

Fig. 6. Goetschkhs's hexagonal adjacency graph with resulting layout [19].

fixed costs are not included in the graph theoretic formulation nor are preassigned spaces accommodated. Foulds [16] views the graph theoretic approach as more appropriate for the design of a new facility where there is more design freedom, while the QAP formulation is more useful in a structured situation.

10. Many-to-one and one-to-many assignment

A problem that has not been addressed as much in academic research but is probably the most widely used commercial application is the stack plan. In stacking problems activities and locations may have unequal areas, however, there is not the added complexity activity shape brings to block plan problems.

Locations in a stack plan generally represent floors of a building (physically shown as a bar). Activities are represented by areas, colored bars assigned to the floors (see Fig. 7). Multiple activities can be assigned to a single floor or a single activity can require multiple floors. This can be modeled as a one-to-one module placement problem, where the floors and activities are comprised of equal area modules. However, it is more attractive to think in terms of assigning an entire activity to a floor in a single step (note, activities may also be allocated over multiple floors). Both constructive and improvement procedures can be used to generate a plan. Simple rules of thumb for locating groups on floors (local heuristics) can produce decent starting solutions to be modified by improvement procedures. A simple pair-wise exchange requires the exchange of relatively equal area activities unless there is free space on a floor. Alternatively, clusters of groups of similar size can be identified for exchange. It is also possible to exceed available floor area during the exchange process using some form of adaptive penalty function to ultimately converge on an acceptable solution. While we have not found applications of some of the newer approaches such as genetic algorithms to the stack problem, there could be a great deal of potential in this area.

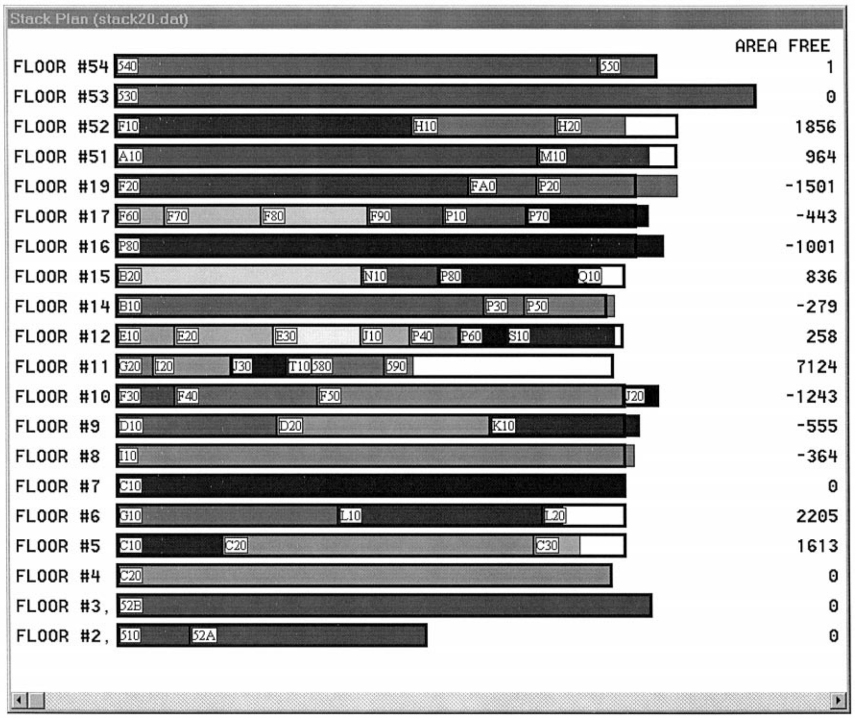
Stack Plan (stack20.dat)
AREA FREE
FLOOR #54 S40 S50 1
FLOOR #53 S30 0
FLOOR #52 F10 H10 H20 1856
FLOOR #51 A10 M10 964
FLOOR #19 F20 FA0 P20 -1501
FLOOR #17 F60 F70 F80 F90 P10 P70 -443
FLOOR #16 P80 -1001
FLOOR #15 B20 N10 P80 O10 836
FLOOR #14 B10 P30 P50 -279
FLOOR #12 E10 E20 E30 J10 P40 P60 S10 258
FLOOR #11 G20 I20 J30 T10 S80 S90 7124
FLOOR #10 F30 F40 F50 J20 -1243
FLOOR #9 D10 D20 K10 -555
FLOOR #8 I10 -364
FLOOR #7 C10 0
FLOOR #6 G10 L10 L20 2205
FLOOR #5 C10 C20 C30 1613
FLOOR #4 C20 0
FLOOR #3, S2B 0
FLOOR #2, S10 S2A 0

Fig. 7. A typical stack plan.

Mahdavi et al. [43], Zhang presents a new approach to the stacking problem. She clusters what are termed functional units (activities to be located) into groups and assigns groups to floors such that the weight between floors is minimized. To begin the process functional units are sorted by their total connected weights in descending order. Each of the first $nf-1$ units is then assigned to a separate group (where nf represents the number of floors in the stacking problem). The remaining units are assigned to the last group. Each group has an area constraint corresponding to one floor of the building. Functional units are moved from the last group to other groups such that intergroup weights are minimized and area requirements are satisfied. When no move will produce a gain or improve the area balance, the process is over. Groups are now assigned to actual floors. If the floors are of different size than it is obvious which cluster is assigned to which floor. If they are the same size, a dynamic programming algorithm is used to assign groups to floors with the objective of maximizing adjacent floor weights.

Another similar problem is the assignment of multiple occupants to a single office. In this case as well, an algorithm need not worry about the actual shape of space, only the activity area assigned to each location. The same solution techniques can be applied to the stack and the many-to-one office assignment problem. Only the graphic representation is different.

11. Multi-floor layout problems

Block and stack problems are considered simultaneously with multi-floor algorithms. Bozer et al. [8] have developed an algorithm called MULTIPLE (MULTI-floor Plan Layout Evaluation) which adopts the grid cell representation for the unequal area block layout problem. Each floor of the building is divided into grid modules. Spacefilling curves are used to layout activities on the grid. A space filling curve is a way of visiting neighbors on a grid by taking horizontal, vertical or diagonal steps to adjoining grid cells (see Fig. 8). The layout is controlled by the order activities are placed in the grid. An initial layout assigns activities to floors. A simulated annealing improvement algorithm is used to modify the order of layout by exchanging activity locations be-

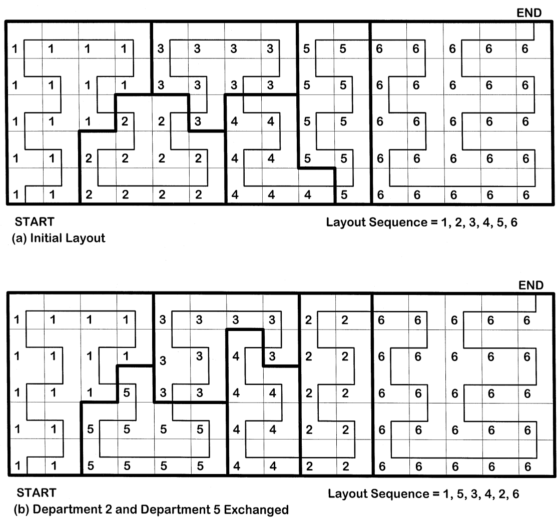
END
START
Layout Sequence = 1, 2, 3, 4, 5, 6
(a) Initial Layout
END
START
Layout Sequence = 1, 5, 3, 4, 2, 6
(b) Department 2 and Department 5 Exchanged

Fig. 8. Using spacefiling curves to construct layouts [8].

tween floors or within floors. The criterion function used to drive the improvement process is a function of horizontal travel between locations on the same floor and vertical travel which is a combination of horizontal travel from each activity to the lift and the travel time of the lift. The use of space filling curves ensures activity contiguity is always maintained and can help manage activity shape.

Kaku et al. [32] use a *K*-median heuristic to cluster departments into groupings in such a way that inter-group interaction is minimized where the number of floors determines the number of groups. This is similar to Mahdavi et al.'s [43] stacking process. Once clusters are created they are assigned to floors using a quadratic assignment objective function. Block plans for each floor are then generated based on distances between locations on the floor and between these locations and the elevator. Interactions with activities on other floors are assumed to flow through the elevator. Thus activities with strong connections to other floors will be placed next to the elevator. This algorithm is limited to equal area activities and a single elevator per floor.

12. Expert systems

While not true expert systems, many of the early heuristics for automated layout implemented rules of thumb that a designer might follow in generating a layout. Modelling the solution process of the human designer is dealt with more explicitly in the work by Akin et al. [2] on layout protocol analysis which resulted in the HeGeL system discussed earlier.

A more traditional expert system approach developed by Malakooti and Tsurushima [44] combines multiple-criteria decision making with an expert system. The expert system has four parts:

- A data base which expresses the problem to be solved. All raw data is treated as facts such as the number of activities, size, and flow.
- A knowledge base which stores domain-specific problem-solving knowledge such as rules of thumb for generating the layout.
- A priority base, which contains priorities for rules, adjacency, the order of assignment, etc.
- An inference engine, which controls the problem-solving structure.

The expert system interacts with the decision maker through the inference engine allowing the decision maker to change the priorities or rules. An interpretation of the layout, which includes all the rules that have been used to assign activities to sites, is displayed so that the decision maker can see why individual assignments have been made. A what-if analysis module allows the decision maker to change information in the data, the knowledge or priority bases and see the results. By giving priority to different criteria and comparing the resulting layouts, the system can automatically update priorities based on the decision maker's choices.

Recent references to expert systems tend to focus more on the integrated problem solving experience than on the actual layout process. Many expert systems use existing tools for the actual layout which include both constructive and improvement procedures mentioned earlier. Abdou and Dutta [1] use an expert system to derive the relationship chart from a set of multiple criteria that are fuzzy, non-quantifiable and apparently conflicting. Once the relationships are derived a standard layout generation package (such as ALDEP or CORELAP) is used to derive a suitable layout. The expert system is then used to examine the feasibility of the result.

FLEXPERT [4], a facility layout expert system based on the theory of fuzzy logic follows a similar scenario. It uses the expert system to generate a relationship chart to combine criteria on flow and closeness relationships between departments. A standard constructive algorithm is linked with an improvement component to generate the layout. Cleland and Hills [10] use a simulated annealing approach for the actual layout, but use a knowledge-based system as an intelligent editor to guide the designer in problem formulation and as an intelligent critic to assess the quality of the layout and to suggest ways to improve it.

13. Commercial applications

While there seems to be considerable interest in computer programs for facility layout, there are surprisingly few commercially available products. The so-called 'layout' features of many CAD systems simply provide a graphic interface for the user to layout a plan in manual mode with little or no access to information concerning the layout criteria. On the other hand, a solution generated by an automated algorithm that is based on a single cost function captures only one aspect of a designer's concerns in any realistic context. A system which meets commercial needs of today should provide interface capabilities ranging from complete user interaction, where the user interactively specifies the location of each activity, to complete automation, where an algorithm generates an initial solution [40]. Or as desired, a designer should be able to interactively locate some activities and use an algorithm to locate or suggest locations for others. Rather than generating a single least-costly plan, the designer with the aid of automated algorithms can make tradeoffs between competing criteria and converge on a solution that responds to a broad spectrum of complex and often ill-defined issues.

In order to meet the needs of facility designers and managers a number of factors must be present in a commercial product:

- The ability to handle large scale problems
- A modern interactive interface
- Support for an iterative design process
- Links to CAD and Facilities Management Databases

It is clear that realistic space allocation problems can involve the assignment of space for very large organizations. A single problem can include multiple buildings, numerous floors and hundreds if not thousands of activities. Automated solution procedures to date have not been tested for problems of this size.

The program interface should be able to prepare and present data at any desired level of aggregation and use output from one stage of the design process to generate subproblems at the next stage (for example, from stacking a multi-storey building to block plans of individual floors). An early commercial implementation, the Calcomp Facilities Planning and Management Application Package, allowed designers to select both the level of space aggregation and activity aggregation from a graphically displayed organization chart. Such an approach is an effective way of reducing the size of large scale problems to make both human and algorithmic problem solving possible. Newer drag and drop graphic interfaces, now expected by users, can be used to move activities from an organization chart to the graphic representation of a plan or to shift activities around a plan.

Experience has shown that layout tools are most effective when employed in an iterative fashion. For example, in a typical layout problem, the design process might start by automatically generating a plan where only information on activity interactions is considered. Generally the result produced will be unsatisfactory, prompting the designer and/or client to make their design requirements more explicit. This can be accomplished by adding information on activity–location preferences and activity preassignments. Location preferences, for example, can be added to the quadratic assignment objective function in the form of fixed costs. Most systems should also have the capability for preassigning activities to particular locations to account for preexisting conditions. Appropriate trade-offs can then be made as the problem is gradually transformed from one in which few locations are fixed to a complete solution.

One of the most important applications of computerized facility layout is in the area of ongoing space management. Here the link between a facilities management database and the layout program is critical. An inventory database of personnel, equipment and space provides information on the current layout of space in the building which can be displayed and evaluated by the layout program. The layout program can then be used to generate and test alternative configurations meeting new space requirements. Output resulting from a layout planner can be used to update the databases and even generate transactions such as move orders.

In spite of the long research history associated with automated layout and space allocation systems, in practice these systems have not been utilized to their full potential. We would expect this to change in the near future given the increasing interest in facilities planning and management, the increasing use of computer-aided design tools in the building design and management industry, and the improvements in computer hardware and software which make the solution of larger scale problems possible as well as facilitate human–computer interaction.

References

[1] G. Abdou, S. Dutta, An integrated approach to facilities layout using expert systems, Int. J. Prod. Res. 28 (4) (1990) 685–708.

[2] O. Akin, B. Dave, S. Pithavadian, Heuristic generation of layouts (HeGeL): based on a paradigm for problem structuring, Environ. Plan. B: Plan. Des. 19 (1992) 33–59.

[3] O. Akin, R. Sen, Navigation within a structure search space in layout problems, Environ. Plan. B: Plan. Des. 23 (1996) 421–442.

[4] A. Badiru, A. Arif, FLEXPERT: facility layout expert system using fuzzy linguistic relationship codes, IIE Trans. 28 (4) (1996) 295–309.

[5] I. Baybars, C. Eastman, Enumerating architectural arrangements by generating their underlying graphs, Environ. Plan. B 7 (1980) 289–310.

[6] C. Baykan, Formulating spatial layout as a disjunctive constraint satisfaction problem, Doctoral Dissertation, Department of Architecture, Carnegie Mellon University, 1991.

[7] T. Block, PLOP—Plant layout optimization procedure, University of Melbourne, Melbourne, 1978.

[8] Y. Bozer, R. Meller, S. Erelebacher, An improvement-type layout algorithm for single and multiple-floor facilities, Manage. Sci. 40 (7) (1994) 918–932.

[9] E. Buffa, G. Armour, Allocating facilities with CRAFT, Harvard Business Rev. 42 (2) (1964) 136–159.

[10] G. Cleland, W. Hills, A knowledge-based systems approach to the layout design of large made-to-order products, in: J.S. Gero, F. Sudweeks (Eds.), Artificial Intelligence in Design '94, Kluwer, The Netherlands, 1994, pp. 257–274.

[11] H. Edwards, B. Gillett, M. Hale, Modular allocation technique (MAT), Management Sci. 17 (3) (1970) 161–169.

[12] C. Eastman, Automated space planning, Artificial Intelligence 4 (1973) 41–64.

[13] A. Elshafei, Hospital layout as a quadratic assignment problem, Operations Res. Q. 28 (1) (1977) 167–179.

[14] U. Flemming, C. Baykan, R. Coyne, Hierarchical generate-

and-test versus constraint-directed search, in: J. Gero (Ed.), Proceedings of the Artificial Intelligence in Design Conference '92, Kluwer, Dordrecht, 1992, pp. 817–838.

[15] U. Flemming, R. Coyne, S. Fenves, J. Garrett, R. Woodbury, SEED—software environment to support the early phases in building design, Proc. IKM94, Weimar, Germany, 1994, pp. 5–10.

[16] L. Foulds, Techniques for facilities layout: deciding which pairs of activities should be adjacent, Management Sci. 29 (12) (1983) 1414–1426.

[17] J. Gero, V. Kazakov, Space layout problems using evolved design genes, Artificial Intelligence in Eng. 12 (3) (1998) 163–176.

[18] P. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem, J. Soc. Ind. Appl. Math. 10 (2) (1962) 305–313.

[19] M. Goetschalckx, An interactive layout heuristic based on hexagonal adjacency graphs, Eur. J. Operational Res. 63 (1992) 304–321.

[20] J. Grason, An approach to computerized space planning using graph theory, Proceedings of the Design Automation Workshop, June 28–30, Atlantis City, NJ, IEEE, New York, 1971, pp. 170–179.

[21] G. Graves, A. Whinston, An algorithm for the quadratic assignment problem, Manage. Sci. 17 (3) (1970) 453–471.

[22] M. Hanan, P. Wolff, B. Agule, Some experimental results on placement techniques, ACM Des. Automation Conf. Proc. 13 (1976) 214–224.

[23] M. Hassan, G. Hogg, D. Smith, SHAPE: a construction algorithm for area placement evaluation, Int. J. Prod. Res. 24 (5) (1986) 1283–1295.

[24] S. Heragu, A. Alfa, Experimental analysis of simulated annealing based algorithms for the layout problem, Eur. J. Operational Res. 57 (1992) 190–202.

[25] F. Hillier, Quantitative tools for plan layout analysis, J. Ind. Eng. 14 (1) (1963) 33–40.

[26] F. Hillier, M. Conners, Quadratic assignment problem algorithms and location of indivisible facilities, Manage. Sci. 13 (1) (1966) 42–57.

[27] C. Huntley, D. Brown, A parallel heuristic for quadratic assignment problems, Computers Ops. Res. 18 (3) (1991) 275–289.

[28] F. Jacobs, A layout planning system with multiple criteria and a variable domain representation, Manage. Sci. 33 (8) (1987) 1020–1034.

[29] J. Jo, J. Gero, Space layout planning using an evolutionary approach, Architectural Sci. Rev. 36 (1) (1995) 37–46.

[30] S. Jojodia, I. Minis, G. Harhalakis, J. Proth, CLASS: Computerized Layout Solutions using Simulated annealing, Int. J. Prod. Res. 30 (1) (1992) 95–108.

[31] B. Kaku, G. Thompson, T. Morton, A hybrid heuristic for the facilities layout problem, Computers Ops. Res. 18 (3) (1991) 241–253.

[32] B. Kaku, G. Thompson, I. Baybars, A heuristic method for the multi-story layout problem, Eur. J. Operational Res. 37 (1988) 384–397.

[33] A. Kusiak, S. Heragu, The facility layout problem, Eur. J. Operational Res. 29 (1987) 229–251.

[34] J. Koopmans, M. Beckmann, Assignment problems and location of economic activities, Econometrica 25 (1967) 53–76.

[35] R. Lee, J. Moore, CORELAP—computerized relationship layout planning, J. Ind. Eng. 18 (3) (1976) 195–200.

[36] J. Leung, A new graph-theoretic heuristic for facility layout, Manage. Sci. 38 (4) (1992) 594–605.

[37] R. Liggett, The quadratic assignment problem: an analysis of applications and solution strategies, Environ. Plan. B 7 (1980) 141–162.

[38] R. Liggett, The quadratic assignment problem: an experimental evaluation of solution strategies, Manage. Sci. 27 (1981) 442–460.

[39] R. Liggett, Optimal spatial arrangement as a quadratic assignment problem, in: J. Gero (Ed.), Design Optimization, Academic Press, 1985, pp. 1–40.

[40] R. Liggett, A designer-automated algorithm partnership, an interactive graphic approach to facility layout, in: Y. Kalay (Ed.), Evaluating and Predicting Design Performance, Wiley, 1992, pp. 101–124.

[41] R. Liggett, W. Mitchell, Optimal space planning in practice, Computer-Aided Des. 13 (5) (1981) 277–288.

[42] M. Los, The Koopmans–Beckmann problem: some computational results, Universite de Montreal, Centre de Recherche sur les Transports, 1976.

[43] A. Mahdavi, O. Akin, Y. Zhang, Formularization of concurrent performance requirements in building problem composition, Working Paper, School of Architecture, Carnegie Mellon University, 1998.

[44] B. Malakooti, A. Tsurushima, An expert system using priorities for solving multiple-criteria facility layout problems, Int. J. Prod. Res. 27 (5) (1989) 793–808.

[45] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1992.

[46] B. Montreuil, A. Laforge, Dynamic layout design given a scenario tree of probable futures, Eur. J. Operational Res. 63 (1992) 271–286.

[47] B. Montreuil, U. Venkatadri, H.D. Ratliff, Generating a layout from a design skeleton, IIE Trans., January, 1993.

[48] R. Muther, Systematic Layout Planning, Cahners Books, Boston, 1973.

[49] R. Muther, K. McPherson, Four approaches to computerized layout planning, Ind. Eng., February 1970, 39–42.

[50] C. Nugent, T. Vollmann, J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations, Operations Res. 16 (1) (1968) 150–173.

[51] C. Pfefferkorn, The design problem solver: a system for designing equipment or furniture layouts, in: C. Eastman (Ed.), Spatial Synthesis in Computer-Aided Building Design, Wiley, New York, 1975, pp. 98–146.

[52] J. Seehof, W. Evans, Automated layout design program, J. Ind. Eng. 18 (12) (1967) 690–695.

[53] R. Sharpe, B. Marksjo, Solution of the facilities layout problem by simulated annealing, Comput. Environ. Urban Syst. 11 (4) (1986) 147–154.

[54] K. Tam, Simulated annealing algorithm for allocating space to manufacturing cells, Int. J. Prod. Res. 30 (1991) 63–87.

[55] K. Tam, Genetic algorithms, function optimization and facility layout design, Eur. J. Operational Res. 63 (1992) 322–346.

[56] D. Tate, A. Smith, A genetic approach to the quadratic assignment problem, Computers Ops. Res. 22 (1) (1995) 73–83.
[57] D. Tate, A. Smith, Unequal-area facility layout by genetic search, IIE Trans. 27 (4) (1995) 465–473.
[58] D. van Camp, M. Carter, A. Vannelli, A nonlinear optimization approach for solving facility layout problems, Eur. J. Operational Res. 57 (1991) 174–189.
[59] T. Vollmann, C. Nugent, R. Zartler, A computerized model for office layout, J. Ind. Eng. 19 (7) (1968) 321–329.