COMPUTER-GENERATED PRELIMINARY DESIGN OF ROOM AND CORRIDOR ARRANGEMENTS UNDER GEOMETRIC CONSTRAINTS

by

ULRICH JOHANNES WALTER FLEMMING

Dipl. Ing., Technische Universitat Berlin
(1968)

submitted in partial fulfillment
of the requirements for the degree of

Master of Architecture

at the

Massachusetts Institute of Technology
(January, 1972)

Signature of Author . . . . . .
Department of Architecture, January 21, 1972

Certified by . . . . . .
Thesis Supervisor

Accepted by . . . . . .
Chairman, Departmental Committee
on Graduate Students

Abstract

COMPUTER-GENERATED PRELIMINARY DESIGN OF ROOM AND CORRIDOR ARRANGEMENTS UNDER GEOMETRIC CONSTRAINTS

by

ULRICH JOHANNES WALTER FLEMMING

Submitted to the Department of Architecture on January 21, 1972, in partial fulfillment of the requirements for the degree of Master of Architecture at the Massachusetts Institute of Technology.

This paper outlines a computer program for the preliminary design of floorplans of rectangular shape in which all spaces are allocated along or around a common circulation area such that each space is accessible from this area. The program consists of a sequence of steps in which the global geometric constraints imposed on the floorplans are satisfied first on the basis of the global significance of the constraints specified for each of the required spaces in the problem at hand. Some of these constraints are then considered a second time, but now with respect to their local significance: the program tests whether there are local configurations of constraints which render a globally feasible floorplan infeasible. A solution is found when all constraints are satisfied.

Due to the specific order in which the constraints are considered, the number of possibilities to be tried during each step in the sequence is limited, except for certain cases which will be described that make a revision of the present version of the program necessary. The sequence of operations, furthermore, is 'tight': the order of operations is predetermined; and it is a 'single thread' sequence: there are no branches.

The program is specialized. More complicated floorplans can be handled if they can be decomposed into parts to which the sequence of operations is sequentially applicable. Entirely different floorplans cannot be introduced.

It is proposed to view the floorplans that can be generated by the described program as representations of a common building type, or of a model of an 'environmental system', and to implement programs of this kind in an experimental environment in which these models are considered hypotheses that can be tested with respect to the class of problems to which they are applicable, with respect to their performance, and with respect to effective design procedures. The results of these tests are considered relevant outside the field of computer-generated design: they might increase the knowledge about the properties and implications of specific models, or suggest generalizations useful during the generation of new models.

Thesis Supervisor: Timothy E. Johnson

Title: Assistant Professor of Architecture

Acknowledgements

The support for this thesis came from a 2-year research grant awarded to the Massachusetts Institute of Technology by the National Science Foundation.

I am grateful to Professor Timothy Johnson and Professor William Porter for their interest in this work and their support.

Contents

I. Introduction

T. Markus suggests a model for the relations between a building and its use which consists of four parts (Markus, 1969):

(1) the building system as a combination technical sub-systems (constructional system, services system, contents system);

(2) the environmental system as a combination of two sub-systems: the spatial system (geometry of spaces and overall layout), and the physical system (visual, thermal characteristics etc.);

(3) the activity system as generated by the occupants;

(4) the objectives of the organization which initiates and sustains the activity system.

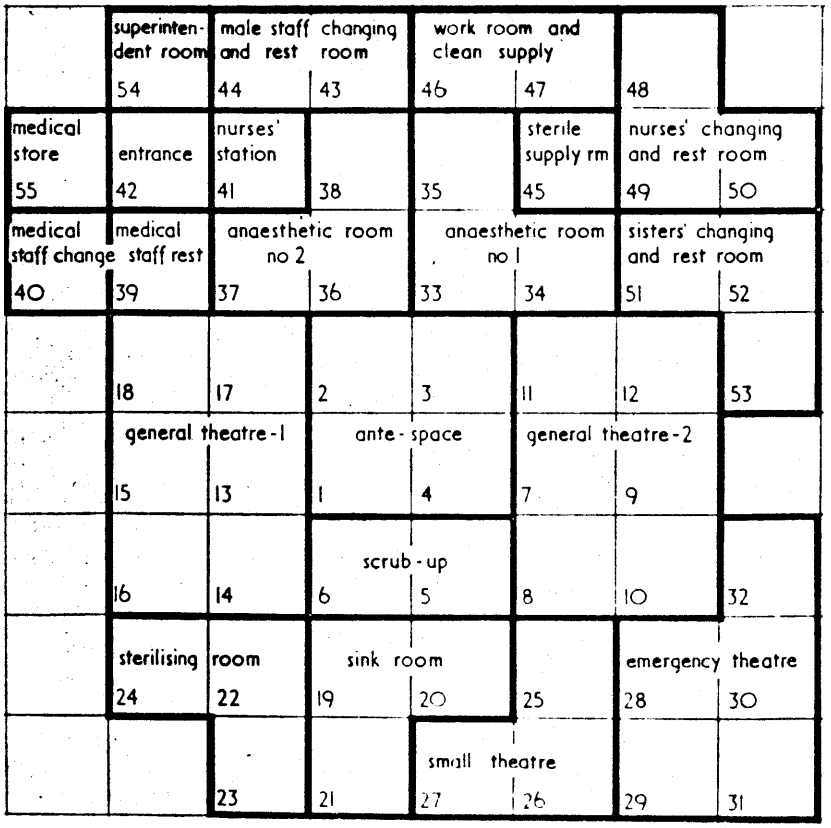
Relations exist between pairs of systems as indicated by the arrows, and each of these relations becomes a topic at some phase in the design process: the relations between (4) and (3) are analyzed during the programming phase; the relations between (1) and (2) are defined, at various levels or detail, during the stages of the design development, and at each stage and for each solution under consideration, the relations between (2) and (3) have to be anticipated and evaluated. Due to the interrelation between these parts, the design problem becomes constrained from two sides: the activity system and the building system. Its goal is to define, within these limits, a satisfactory or optimum environmental system.

Markus distinguishes between the "design morphology" and the

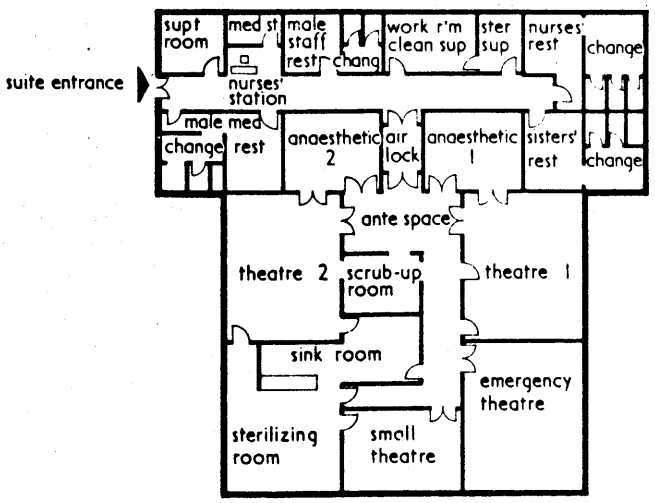
"design process": The design morphology consists of the sequence of operations and decisions which contribute to the development of a project over time, and the design process consists of a series of problem-solving procedures which are applied during each phase in the design morphology. In the following, the term preliminary design refers to a particular phase in the design morphology in which the general shape and layout of a building are defined on the basis of information collected in the previous programming phase. The term design process refers to the methods used in order to achieve this goal.

Consider now the layout for a part of a hospital as suggested by an early preliminary design program and shown in figure 1(a) (Whitehead and Eldars, 1964). It is based on an analysis of the activity system the characteristics of which are represented as a set of required areas and the frequency of traffic between these areas. The layout is generated by a sequential allocation routine which tries to minimize the total cost of traffic.

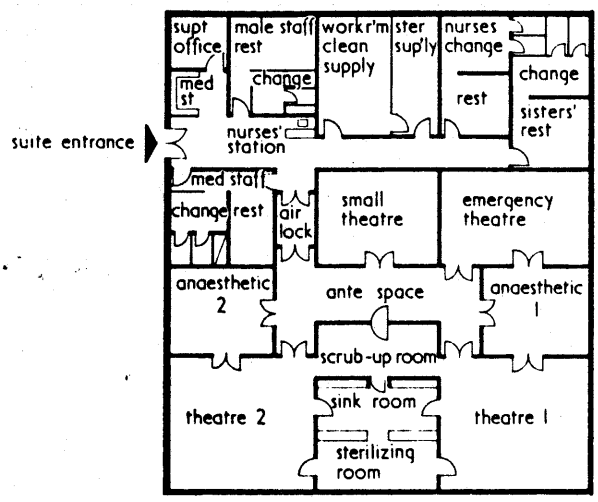
As Whitehead and Eldars observe, this arrangement, in order to become usable, has to be converted into a "practicable form". Two stages of the conversion process, as suggested by the authors, are shown in figure 1(b) and 1(c). One of the main criteria that govern the conversion is stated explicitly: the problem requires a certain circulation pattern which groups the spaces around two circulation areas one of which has to be introduced into the layout. Other criteria can be deduced from the drawings: there is a strong interest in regular shapes both for the single areas and the overall layout. This interest might reflect structural considerations, functional requirements connected with the



1(a)



1(b)



1(c)

Figure 1: Layout for a part of a hospital

use of the areas, a formal interest, or most likely, all of them. The circulation pattern and the overall shape appear as global constraints: they influence the dimensions and relations of almost all the areas, and it seems indeed difficult to consider them in a sequential allocation procedure. More local constraints, like area shape, were perhaps excluded from the program for simplicity reasons. Local and global, functional, formal and technical constraints, as well as the optimization criterion, interact, and the final combination of relations and dimensions reflects this interaction in a specific way. The authors are sceptical of the possibility of writing a computer program which considers all of these constraints simultaneously. In light of the computer programs I know, this scepticism seems to be well founded.

The problem seems to be less difficult for cases where the shape and the dimensions of an area within which the spaces have to be allocated are given. But the preliminary design process is characterized precisely through the absence of such an area. Neglecting exceptional cases, the definition of such an area is one of the main tasks of this phase.

Even if, at the present time, it is not possible to design a computer program which generates floorplans as shown in figure 1(c), it might nevertheless be worthwhile to write a program which solves simpler problems. On the basis of this experience, it might be possible to suggest extensions of the simple problems or reductions of the more complex ones. At the same time, the set of examples is enlarged against which different approaches can be evaluated and compared. Finally, the results might contribute to an understanding of the constraints under which the design process takes place and the methods that can be used. To conduct

such a study on the basis of a computer program has an immediate advantage: all relevant assumptions have to be completely explicit and can subsequently be tested through a series of examples. This, in fact is the main interest behind this paper.

More specifically, I try to achieve two objectives:

(1) to outline a computer program for the preliminary design of a certain class of floorplans;

(2) to draw conclusions both for the problem at hand and the design process in general.

The selected floorplans are simpler versions of the floorplan shown in figure 1(c). They are characterized by the following properties:

(1) The overall shape is rectangular. In the present program, this assumption appears as a purely formal constraint. It should be noted, however, that it has implications for the technical subsystems which can be used.

(2) Each required space is adjacent to a common circulation area such that all spaces form a "chain" along or around its sides.

(3) The dimensions of adjacent spaces are equal along shared walls, except in corners.

Properties 1 and 2 are global constraints on the problems which can be solved by the program. Furthermore, they are used, together with property 3, as important means for directing the search for a solution and cutting down the possible number of combinations of spaces. They are not treated as external constraints in a trial-and-error procedure. The relations between global and local constraints will be dealt with expli-

citly, and to arrive at an understanding of these relations is the main purpose for the program's implementation.

Aside from these global constraints, the program depends on local constraints which (a) must be specified by the user for each space: min. dimension, min. area, max. area, daylight required or not; (b) can be specified by the user: adjacency relations between pairs of spaces, clusters of spaces. They are necessary if the program is to be applied to realistic cases.

The buildings which can be designed under these constraints belong to a certain class which, in the terminology of this paper, represents a model of an environmental system, although in a crude way. It is applicable to apartments, office buildings, schools and the like. I selected this particular model for two reasons:

(1) It seems easy enough to define and program the way in which global and local, functional and formal constraints interact.

(2) The program can be applied to realistic cases such that the validity of the assumptions on which it is based can be tested.

The program was programmed in LISP which turned out to be a very convenient language for the problem at hand; yet the possibilities provided by the language were not fully utilized. The program is, at the present time, only partially implemented.

II. Program Description: Problem Statement

A problem statement consisting of the following parts is sufficient for the class of problems the program is intended to solve:

(1) names of required spaces

(2) attributes of the required spaces

(3) relations between the required spaces

(4) external constraints

In the following, each of these parts will be discussed in turn.

Spaces and their attributes

The values of the following attributes have to be specified for each space:

(1) minimum dimension (in meters)

(2) minimum area (in square meters)

(3) maximum area (in square meters)

(4) daylight required (yes/no)

Exceptions are circulation spaces for which only the value of attribute 1 has to be specified. For all other spaces, it is in addition assumed that they should be of rectangular shape.

Attributes 1 and 2 specify lower bounds below which a space becomes inappropriate for its intended use; attribute 3 indicates an upper bound above which the allocation of areas becomes wasteful. Maximum dimensions are computed as the quotients of maximum areas and minimum

dimensions. Neither of the two dimensions implies a particular orientation of the space for which it is specified. Attributes 1, 2 and 3 are quantitative attributes: their values are expressed as numbers. Attribute 4 is a qualitative attribute. A space for which its value is "yes" must be placed along an external wall; if the value is "no", the space can be placed in the inner core of a building or along an external wall.

Table 1 shows a subset of the spaces required for a university building as listed by Bareither and Schillinger which will function as an example throughout the paper (Bareither and Schillinger, 1968). The table shows also the assumed values of the attributes for the required spaces. Among those, only the minimum areas are explicitly given by the authors. Plausible values for the other attributes were assumed in a rather ad-hoc way. Table 2 lists the required spaces and the values of their attributes for a 1-bedroom apartment which will serve as a second example in the following chapters.

Table 1 also shows that some spaces occur repeatedly, and that for others the values of their attributes are identical. In both cases, the attributes do not have to be specified more than once The input routine, therefore, expects two lists of spaces: a list of space types containing their code names and the values of their attributes, and a list of required spaces containing their code names, the code names of the space types to which they belong, and, if appropriate, an integer number which indicates how often they occur in the problem at hand. Required spaces of the same type which differ in their relations to other spaces have to be listed separately in order to avoid ambiguities.

These spaces and their attributes are part of the data base to which

TABLE 1: Example 1 - Attributes of Spaces

	Name	No.	Min. Area	Max. Area	Min. Dimension	Daylight
A-1	Classroom	4	39	43	4.50	no
A-2	Seminar Room	3	26	29	3.60	no
A-3	Phonetics Classroom	1	39	43	4.50	no
A-4	Observation Room	1	39	43	4.50	no
A-5	Meeting Room	1	93	103	7.80	no
A-6	Duplicating Room	1	37	41	4.35	no
B-1	Office	1	22	25	3.30	yes
B-2	Office	1	17	19	2.90	yes
B-3	Office	2	11	13	2.25	yes
B-4	Office	1	11	13	2.25	yes
B-5	Conference Room	1	17	19	2.90	no
B-6	Library	1	22	25	3.30	no
B-7	Storage	1	11	13	2.25	no
B-8	Office	12	11	13	2.25	yes
B-9	Office	4	11	13	2.25	yes
COR	Public Corridor	1	—	—	2.25	—

TABLE 2: Example 2 - Attributes of Spaces

	Name	No.	Min. Area	Max. Area	Min. Dimension	Daylight
LIV	Livingroom	1	20	30	4.00	yes
KIT	Kitchen	1	7	10	2.25	no
BED	Bedroom	1	12	16	3.00	yes
BAT	Bathroom	1	4	5	1.75	no
COR	Private Corridor	1	—	—	1.25	—

the program refers at various stages. In its present version, the program represents this part of the data base as a set of "property lists". Property lists are a standard feature of LISP. They are associated with each literal atom and structured as lists in which "indicators" (names) and values alternate (Weissman, 1967). An example is shown in figure 2.

A property list is set up for each space type and each required space; it contains, in the first case, the names and values of all attributes, and in the second case, a pointer to the appropriate space type. The property list for the space type "classroom" is shown in figure 2 as an example. Since the elements of a list can be numbers, literal atoms,

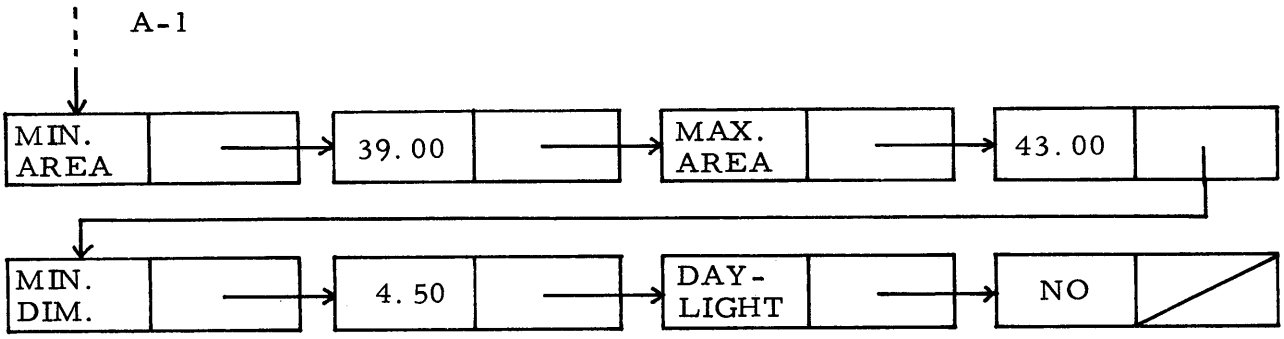


Figure 2: Property list

or themselves lists, this set of property lists represents a general, and, as it turns out, very handy way of storing information of this kind.

Relations between Spaces

Bareither and Schillinger represent the "functional relationships" between the subset of spaces in example 1 through graphs which are shown in figure 3. A decomposition into 3 non-overlapping clusters suggests

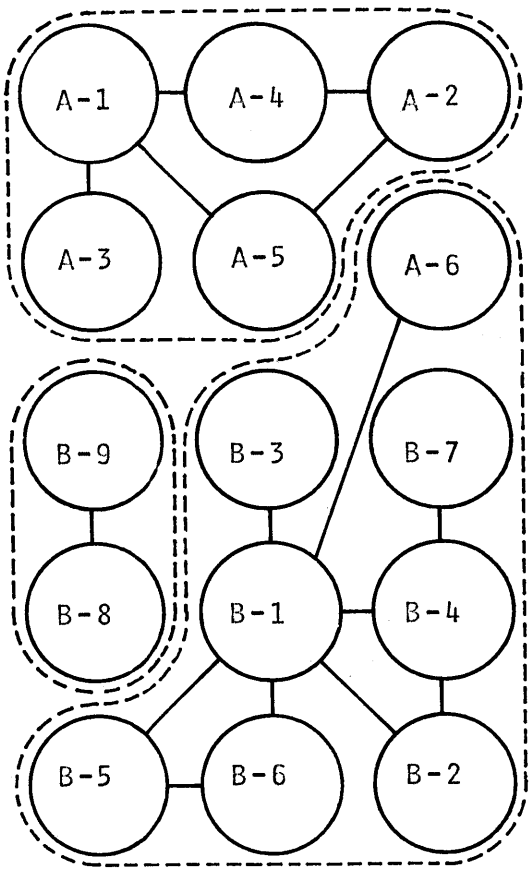


Figure 3: Example 1 – Clusters

itself and is indicated through dotted lines. In this example, the clusters represent the departments and sub-departments within a university, a way of grouping which occurs frequently in design problems. For cases which exhibit more complex interrelationships between spaces, one might use one of the available clustering routines in order to arrive at a similar decomposition (Miller et al., 1969).

Once the clusters are established, they can be incorporated into the problem statement by appropriate parentheses in the list of required spaces. Spaces of the same type which occur in different clusters have to be listed under different names in each cluster where they occur. This representation of clusters neglects the relations within a cluster. It will be indicated below how they could be added without changing the framework of the program.

A stronger functional relationship between spaces can be expressed by a required adjacency relation. In the present program, two spaces are considered adjacent if they are allocated at the opposite sides of a wall-segment the length of which is at least equal to the width of a door.

In example 1, adjacency relations are required between the public corridor and all other spaces. In the terminology of this paper, these spaces form a chain of spaces. The same holds for example 2: the private corridor has to be adjacent to all other spaces.

Definition:

A chain of spaces is a set of spaces all of which have to be adjacent to the same circulation space.

The present program handles only chains of this kind. All other

patterns of adjacency relations are considered as nets and, once they are discovered, cause the program to terminate. Under this restriction, the space allocation procedure becomes extremely simple as will be shown below.

Adjacency relations can be specified also between the members of a chain. In example 1, the observation room has to be adjacent to a classroom and a seminar room. They form, in the terminology of this paper, a strong chain. All members of a chain which do not belong to a strong chain form a weak chain. Two restrictions hold for adjacency relations between the members of a chain:

(1) No member of a chain can be adjacent to more than two members of the same chain; otherwise, the shape of these spaces cannot be rectangular.

(2) Except for cases where four spaces form a strong chain, no two members of a chain can be adjacent to the same two spaces in the same chain.

Adjacency relations between pairs of spaces are specified in the problem statement as lists and stored redundantly in the property lists of the required spaces for which they are specified. Again, required spaces of the same type, but with different adjacency relations, have to be listed separately.

In figure 4(a), the adjacency relations specified for example 1 are indicated by arrows, clusters by dotted lines. The graph for example 2 is shown in figure 4(b). In both cases, the spaces are either circulation spaces or members of the same chain. The program in its present

version handles only simple chains of this type.

Figure 5, on the other hand, shows the adjacency relations in an apartment-building which contains a series of apartments as specified in example 2. The whole set of relations can be viewed as a hierarchical structure of chains: the spaces within an apartment form a chain in relation to the private corridors, and the apartments form a chain in relation to the public corridor. The program, in a slightly extended version, could handle such a case by applying sequentially the procedures for simple chains.

Figure 6, finally, shows a possible interpretation of the floorplan shown in figure 1(c) in terms of adjacency relations (slightly simplified). The air lock is a member of two distinct chains: both chains overlap. The program in its present version cannot handle this case.

External constraints

The program, in its present form, can handle only restrictions of the overall dimensions of a building as external constraints.

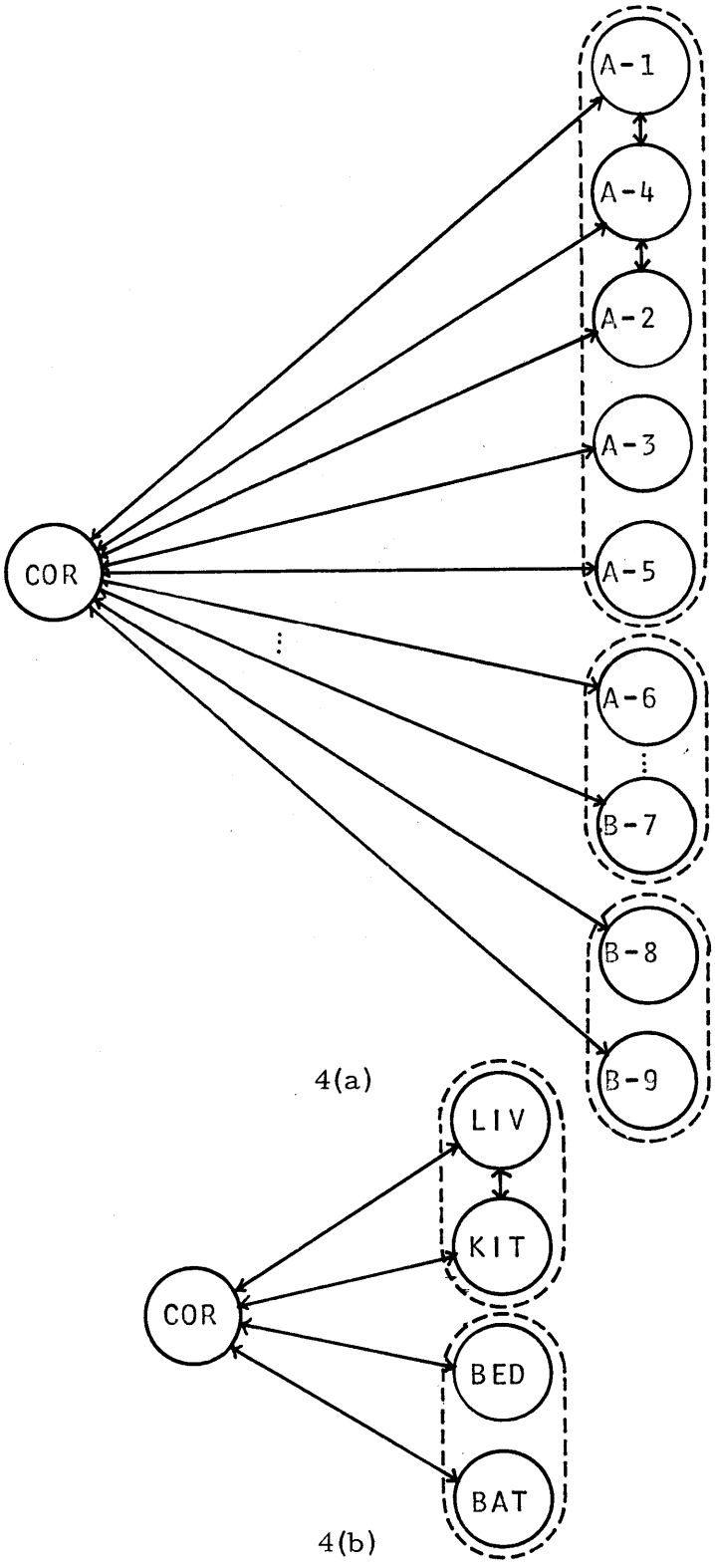


Figure 4: Simple chains

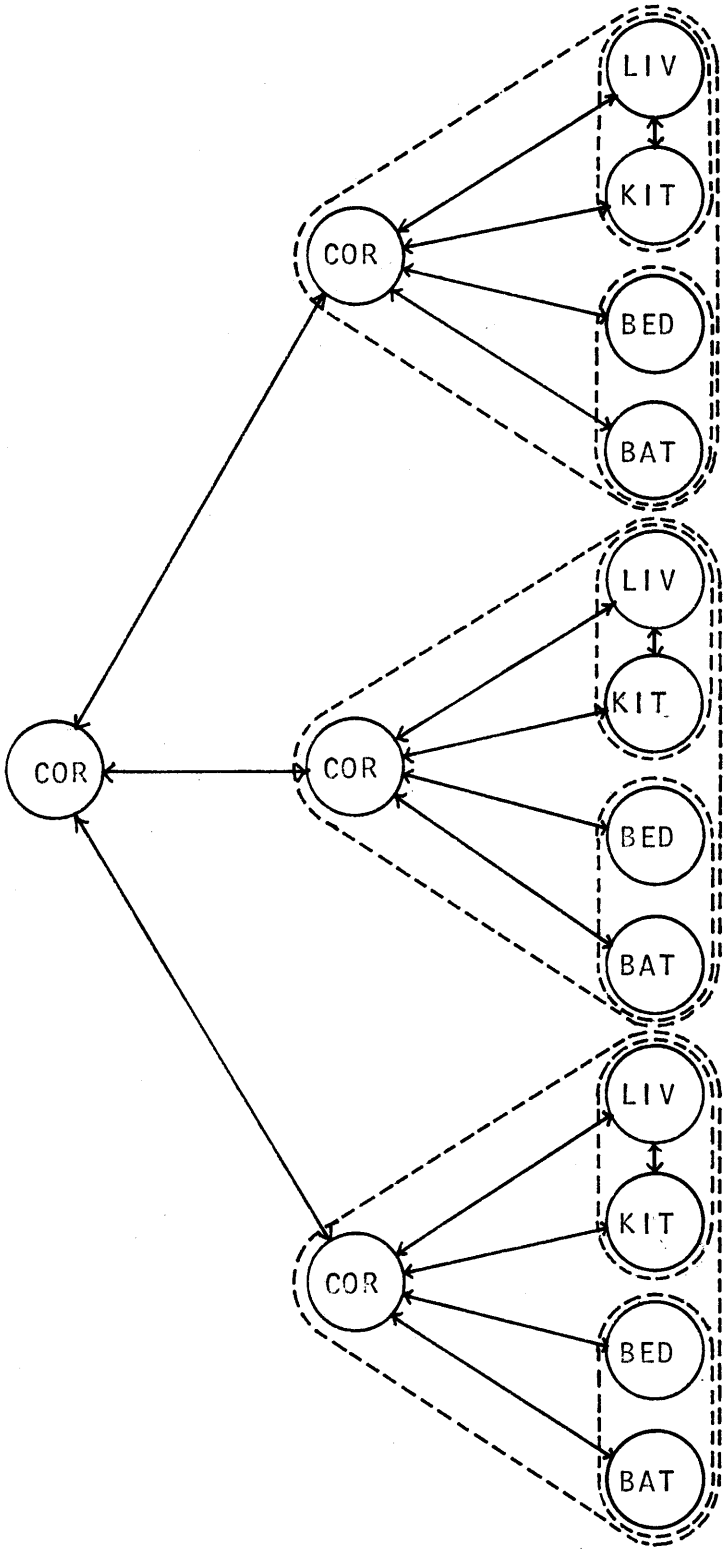


Figure 5: Nested chain

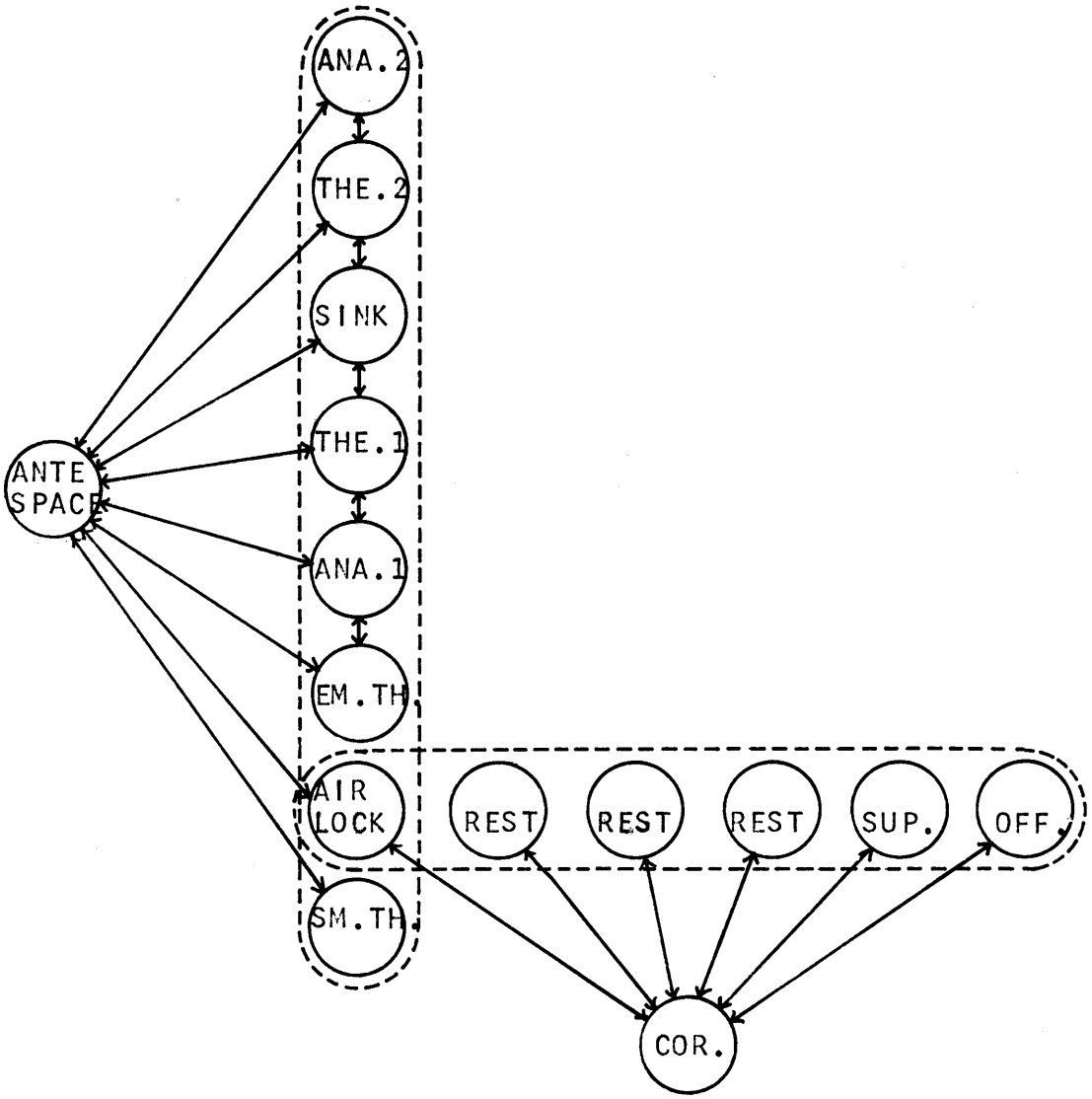


Figure 6: Overlapping chains

III. Program Description: Sequence of Operations

Figure 7 shows a first class of floorplan schemes which the program is able to generate. They all contain a circulation area (CA) and one or more zones, labelled from A to D, within which spaces can be allocated. Since it is required that plans have to be of rectangular shape, certain combinations of zones have to be complemented by corner zones, numbered from 1 to 4 in figure 7. Zones and corner zones are assumed to be of rectangular shape, a restriction which does not necessarily hold for circulation areas, as will be shown later. I shall also explain how the procedures that handle these schemes can be extended to a second class of floorplans which contain an inner core of spaces.

Zones, corner zones and circulation areas are the components out of which legal floorplans can be formed. The possible combinations of these components are known, as well as a condition under which these combinations can be realized: components can only be adjacent to each other along sides of equal dimensions. The dimensions themselves are undefined. It is the task of the program to find possible combinations of these components, to define their dimensions, and to allocate the required spaces into zones and corner zones such that the constraints specified in the problem statement are satisfied. This is done in a sequence of steps. A flow chart for this sequence is shown in figure 8. It has to be kept in mind during the following chapters that this sequence depends on the assumed representation of the floorplans. It would have a different structure if the floorplans were represented differently.

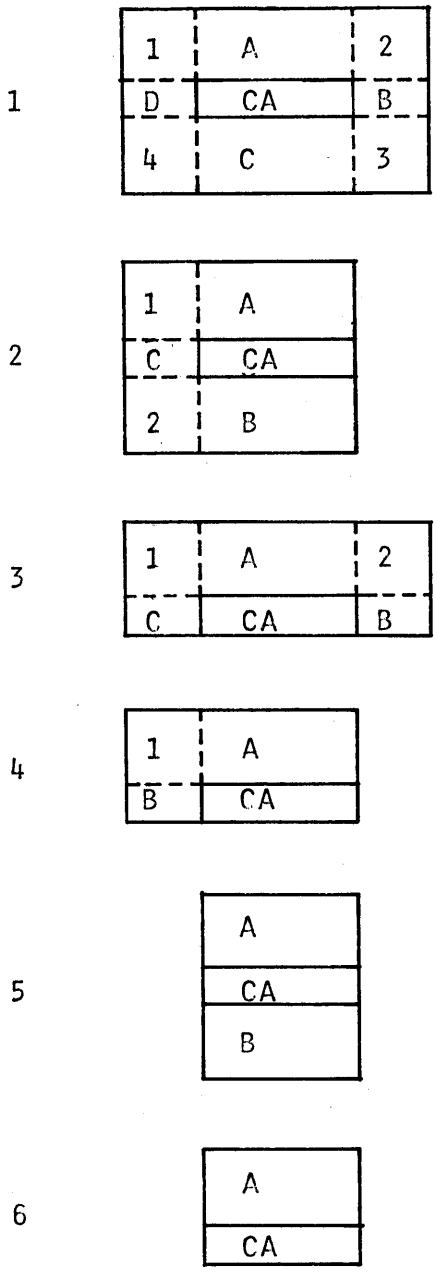


Figure 7: Floorplan schemes

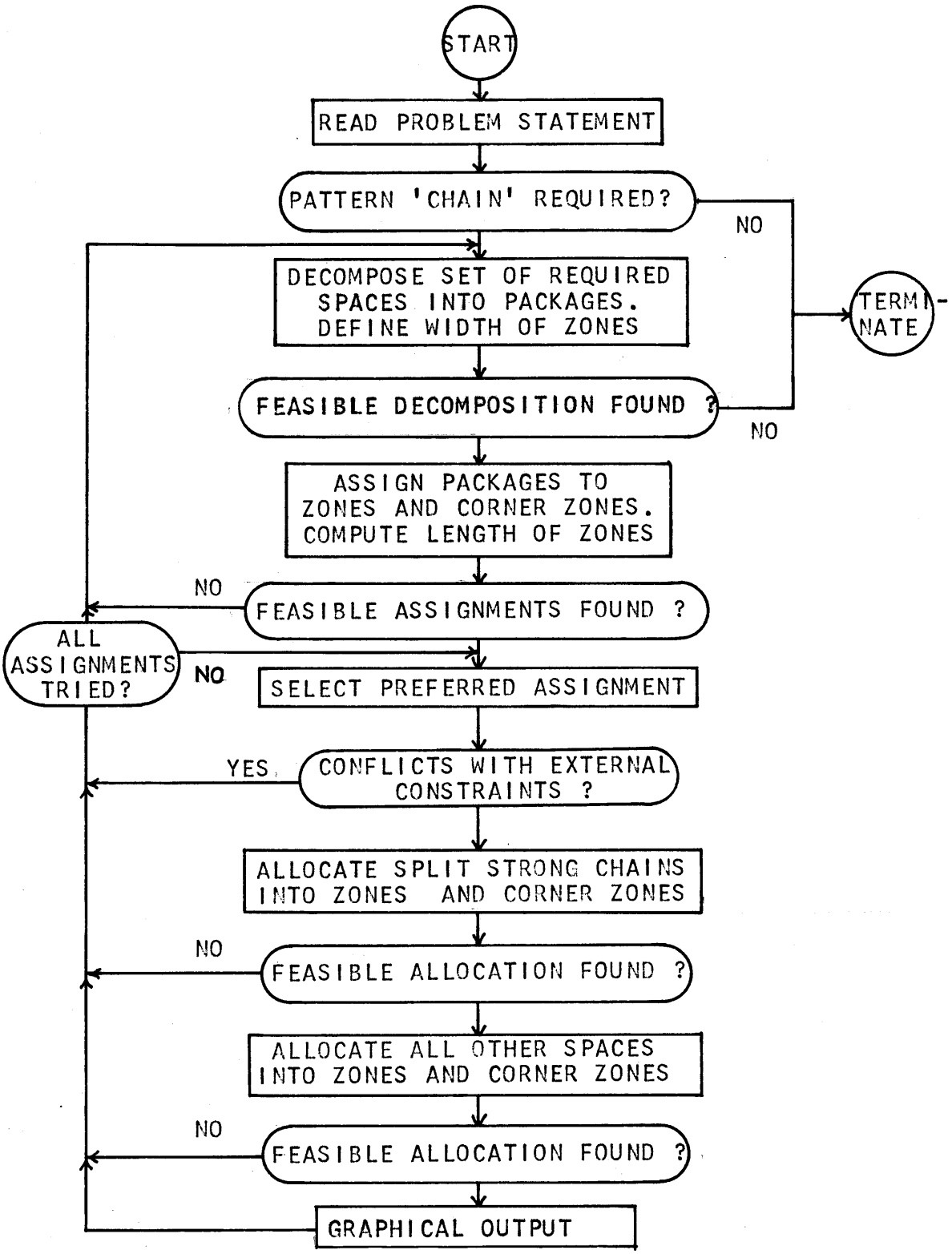


Figure 8: Program flow chart

Step 1: Decomposition of Spaces into Packages

It was assumed that spaces form a simple chain around a circulation area, that they have to be allocated in zones and corner zones, and that these zones are rectangular. It follows that the dimensions of two spaces X and Y which are allocated in the same zone must be compatible with each other such that

max. (min. dim. of X, min. dim. of Y) $\leq$
min. (max. dim. of X, max. dim. of Y).

The maximum of the minimum values and the minimum of the maximum values define a range for the width of a zone in which X and Y can be allocated, where the width is measured at right angles to the circulation area (cf. figure 7).

The compatibility of dimensions provides a criterion for a second decomposition of the set of required spaces into subsets which will be called packages in the following. Figure 9 shows the dimensions of the spaces required for example 1 in increasing order. If, for example, (B-3, . . . , B-6) and (A-5) are regarded as two packages, the remaining package (A-2, . . , A-4) is compatible with either one, but not with both at the same time The compatibility criterion, therefore, does not necessarily partition a set of spaces uniquely, and due to the way in which the packages are used in the next step, it might be necessary to consider more than one partition. For the present examples, I was content with a single partition into packages which is generated by the following rule: order the set of spaces as shown in figure 9; group spaces into packages starting with the biggest space; after each space which is

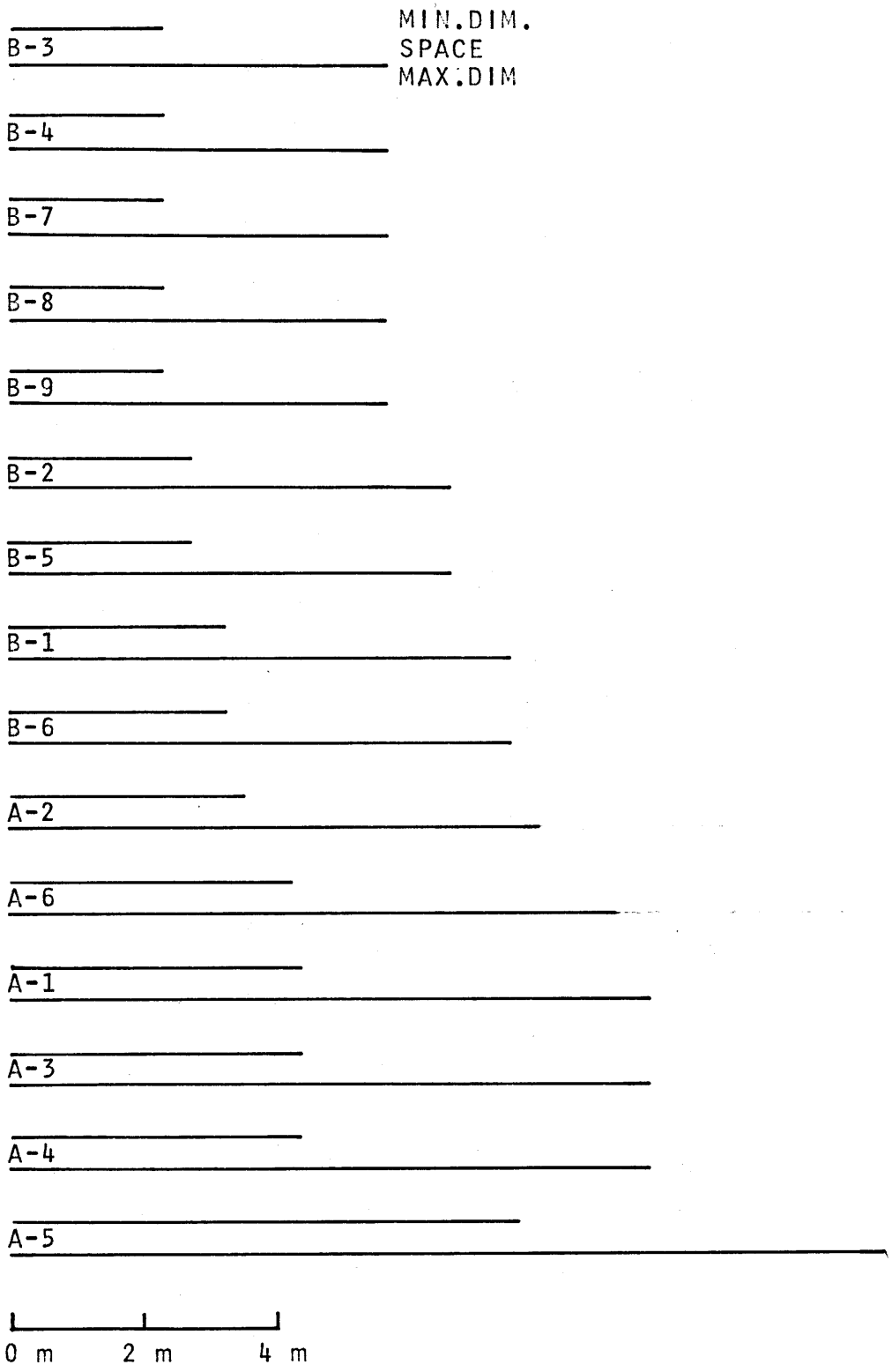


Figure 9: Example 1 – Minimum and maximum dimensions

compatible with the last one, re-compute the range of compatible dimensions; start a new package if the dimensions are incompatible. The resulting decompositions for examples 1 and 2 are shown in tables 3 and 4. Spaces which must be adjacent to each other or belong to the same cluster can belong to different packages. The decomposition into packages neglects these relations; the reason will be explained below.

By definition, the maximum number of zones is 4 for the first class of floorplans. Since the spaces in different packages have to be allocated in different zones, the maximum number of packages is also 4. Decompositions which yield more than 4 packages are infeasible. In this case, a different decomposition for the first or a second class of floorplans has to be tried.

Step 2: Assignment of Packages to Zones

A different zone is needed for each package. For each of the present examples two packages are found during step 1. A possible combination of zones and corner zones, therefore, must consist of at least two zones. There are two ways in which two zones can be combined under the conditions introduced at the beginning of this chapter. They are shown in the first row of columns 1 and 2 in figure 1o. Row 2 shows how two packages A and B can be assigned to these zones. These diagrams should be considered as schemes: rotations and exchanges of packages are omitted, but have to be considered during step 2. Combination 2 has to be complemented by a corner zone. Spaces can extend into this corner zone from either zone and can, therefore, belong to either package. The possible assignments of packages to zones and corner zones are shown

TABLE 3

Example 1 – Decomposition into packages

package	zone width	spaces
A	3.30 – 5.77	B-1 B-2 B-3 B-4 B-5 B-6 B-7 B-8 B-9
B	7.80 – 8.05	A-1 A-2 A-3 A-5 A-6 A-4

TABLE 4

Example 2 – Decomposition into packages

package	zone width	spaces
A	4.00 – 4.44	LIV KIT BED
B	1.75 – 2.85	BAT

TABLE 5

Example 2 – Second decomposition into packages

package	zone width	spaces
A	2.25 – 2.85	KIT BAT
B	4.00 – 5.33	LIV BED

TABLE 6

Example 1 – Second decomposition into packages

package	daylight	zone width	spaces
A	yes	3.30 – 5.77	B-1 B-2 B-3 B-4 B-8 B-9
B	no	3.30 – 5.77	B-4 B-5 B-6
C	no	7.80 – 8.05	A-1 ... A-6

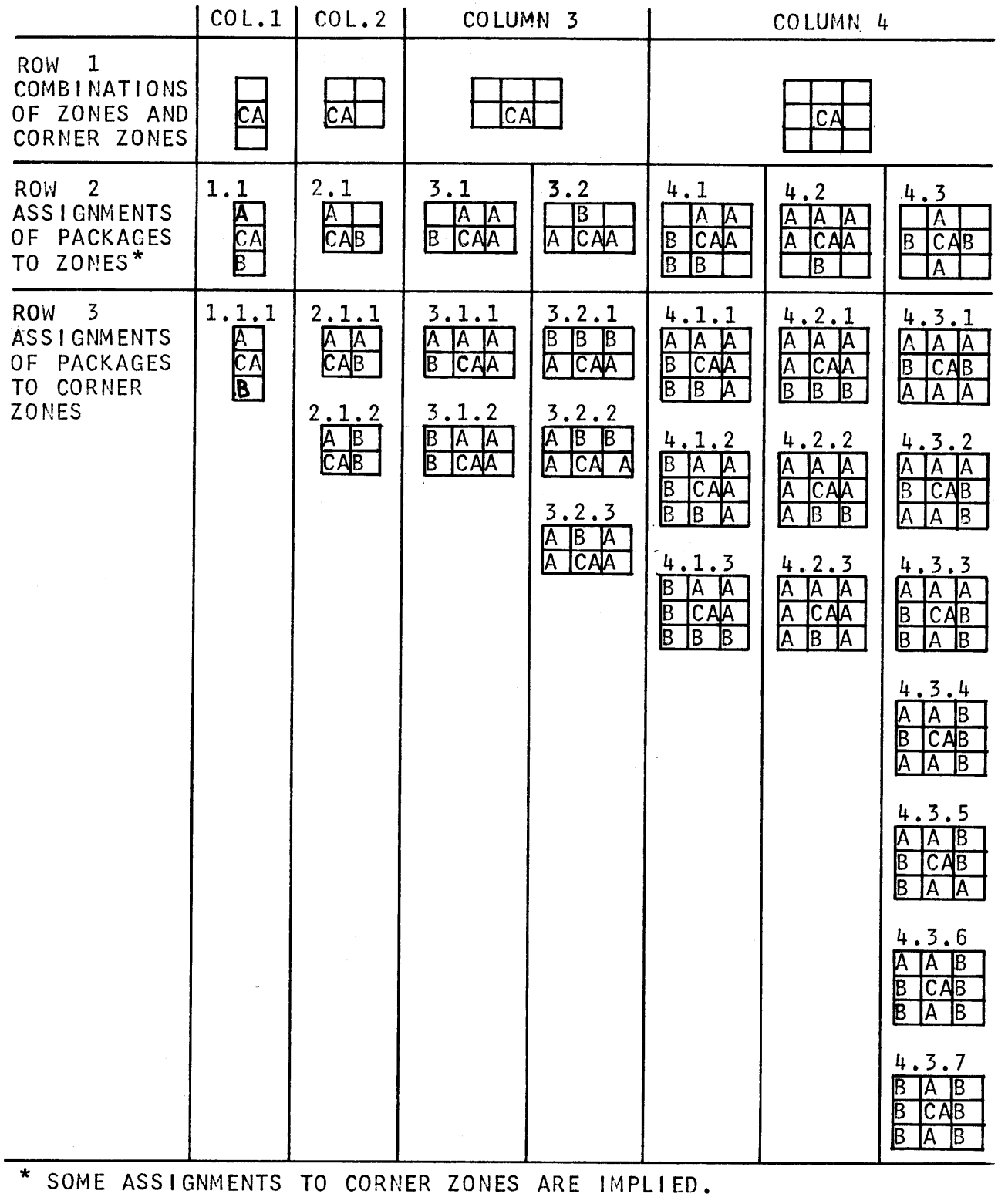


Figure 10: Assignments of packages to zones

in row 3 of figure 10.

The range within which the width of these zones can vary is known after the packages are assigned. The length of the zones can be computed, where the length is measured parallel to the circulation area. This area has a long and a short side. The minimum dimension of the short side is specified in the problem statement; its maximum dimension is assumed to be the sum of the minimum dimension and some constant (1 m for the present examples). These dimensions, at the same time, define a range within which the length of the zones at the short sides of the circulation area can vary. For each assignment, therefore, the minimum and maximum dimensions for the width of all zones, for the length of the zones at the smaller sides of the circulation area, and consequently, for the sides of the corner zones are known. The areas of the smaller zones and the corner zones can be computed. The program tests whether there is a length x for the longer zones such that

(1) the total area which can be allocated for each package equals the total area required for the spaces in these packages;

(2) the dimensions of all zones and corner zones are compatible with each other.

Each assignment which passes this test is tentatively considered a feasible assignment. The length x is again defined within a range: it is a function of the area required for the zone under consideration and its width, where both can vary; the length of the zone can vary accordingly. Whenever I refer to dimensions in the following paragraphs, it is implicitly assumed that these dimensions can vary between a minimum

and a maximum value.

I assumed for the present version of the program that packages can be split and the pieces assigned to different zones. As a result, the number of possible combinations of zones and corner zones and consequently the number of possible assignments increases. The number of ways in which this can be done depends on both the number of packages and the number of desired zones. Once packages are assigned to zones, it has again to be decided how the corner zones are to be filled. The principal possibilities for two packages are shown in row 3 of columns 3 and 4 in figure 10. The total number of possible assignments for two packages is 21. I counted 4 assignments for 1 package, 28 for 3 packages, and 16 for 4 packages.

For most of the assignments, the program has to consider rotations such that packages which were assigned to small sides become assigned to long sides and vice-versa. Also, packages might have to be exchanged in an assignment. Thus, the number of possibilities that have to be considered increases; I counted, for example, 52 possibilities for two packages. On the other hand, the assignments can be considered in an order which reduces the number of possibilities which actually have to be tested: some assignments can be skipped if previous ones failed. In any case, the combinatorial problem remains manageable: the number of possible assignments of packages to zones and corner zones is known and limited, and the number of feasible assignments which have to be considered in the following steps is much smaller. The total number of feasible assignments is 7 for example 1, and 2 for example 2.

A further simplification is possible, if the feasible assignments are considered in an order of preference according to some heuristic rule such that the search terminates if solutions for the preferred assignments are found. The rule adopted for the present version of the program is based on an implicit efficiency criterion and favours compact floorplans. The program always tries the combinations 1 and 4. For each feasible assignment within combination 4, the overall dimensions are computed and the assignment with the smallest overall length is tried first; only if conflicts occur at later stages, the next best assignments are tried. If both combination 1 and combination 4 fail, combination 3 is tried, and if it fails, combination 2. Since for each feasible assignment only one solution will be generated, the maximum number of solutions for the first class of floorplans is 2. According to this rule, the program considers, within the first class of floorplans, two feasible assignments for example 1: assignment 1.1.1 and assignment 4.1.2. For example 2, assignment 3.1.1 succeeds.

Feasible assignments are represented in the following way: an internal name is generated for each zone and corner zone, and a property list is associated with each of these names containing the appropriate dimensions and pointers to the names of the adjacent areas and to the appropriate packages. During the following steps, the program tests whether the required spaces can be allocated in these areas such that the constraints specified in the problem statement are satisfied.

Step 3: Comparison of Overall Dimension and External Constraints

At the end of step 2, the overall dimensions of the assignment under consideration are known or can be computed. The program tests whether

they are in conflict with the external constraints specified in the problem statement. If a conflict occurs, a different assignment has to be tried.

Step 4: Allocation of Split Strong Chains

So far, relations between spaces were neglected. But the decomposition into packages might split strong chains, and they can only be recombined at corners where these packages meet. The program generates a list of split strong chains and tests whether there is, within the assignment under consideration, a sufficient number of corner zones where the appropriate packages meet. If the test fails, a different assignment has to be tried. Otherwise, the spaces in the list of split strong chains are allocated in the appropriate corner zones and adjacent zones such that two conditions are fulfilled:

(1) Spaces in a corner zone have to extend far enough into one of the adjacent zones such that they share a wall segment of at least door-width with the circulation area; otherwise, the required adjacency relations cannot be satisfied. In case of failure, a different assignment has to be tried.

(2) Spaces in the same strong chain can be allocated in a corner zone and its adjacent zones only if the areas available in these zones are not exhausted before all spaces are allocated: strong chains cannot be split and allocated in distant zones. If a zone is adjacent to two corner zones, its area can be exhausted even faster: both corner zones might be needed, and spaces can be allocated in the same zone from two directions. On the other

hand, strong chains can be exchanged. Again, a limited number of possibilities has to be tried, depending on the number of split strong chains, which is never larger than 4, and the number of appropriate corner zones. An assignment fails if all possibilities are exhausted.

Some back-up mechanism is needed which tries the different possibilities in turn. This mechanism has not been implemented since it is not needed for the present examples: the list of split strong chains is empty in both cases, and step 4 can be skipped.

The allocation procedure itself is extremely simple. For each zone and corner zone, the spaces are listed in the order of allocation as required by the adjacency relations; two lists are needed for zones in which spaces are allocated from two directions. After each allocated space, the length of the remaining area is re-computed. At the end of the allocation procedure, the lists of allocated spaces and the length of the remaining areas are added to the property lists of the zones and corner zones under consideration.

Step 5: Allocation of All Other Spaces

The remaining spaces have now to be allocated in the remaining areas. For example 1 and assignment 1. 1. 1 the allocation procedure is especially simple. There are two packages and two zones, and the zones are still empty since step 4 was skipped. The spaces are allocated linearly in the appropriate zones as described above: they are listed in the order of allocation and the length of the remaining areas is re-computed; the results are stored in the appropriate property lists. A

floorplan which represents the generated solution is shown in figure 11. In this floorplan and the following ones, arabic numbers refer back to figure 7 and indicate the floorplan scheme on which the floorplan under consideration is based.

The order in which spaces are allocated is based on the clusters which were specified in the problem statement: spaces in the same cluster are kept together, unless a stronger constraint makes a splitting of clusters necessary. Split clusters can occur if strong chains were allocated at distant corners during step 4 such that the zones between these corners have to be filled with spaces from different clusters, or if spaces have to be exchanged due to local conflicts as will be shown in the next paragraphs. In general, clusters are treated as rather weak constraints: they are realized only by default. The order of allocation within a cluster is arbitrary and depends on the order in which the required spaces are listed in the problem statement.

The space allocation procedure is less simple if corner zones are part of an assignment. When a space has to be allocated in a corner zone, the program has to test, as in step 4, whether this space extends far enough into an adjacent zone. If the test fails, two possibilities can be tried:

(1) The critical space can be exchanged for a bigger space in the same package;

(2) if this is impossible, the program can try to utilize the tolerances within the dimensions of a zone and insert a piece of circulation area between zone and corner zone such that the corner zone becomes directly accessible.

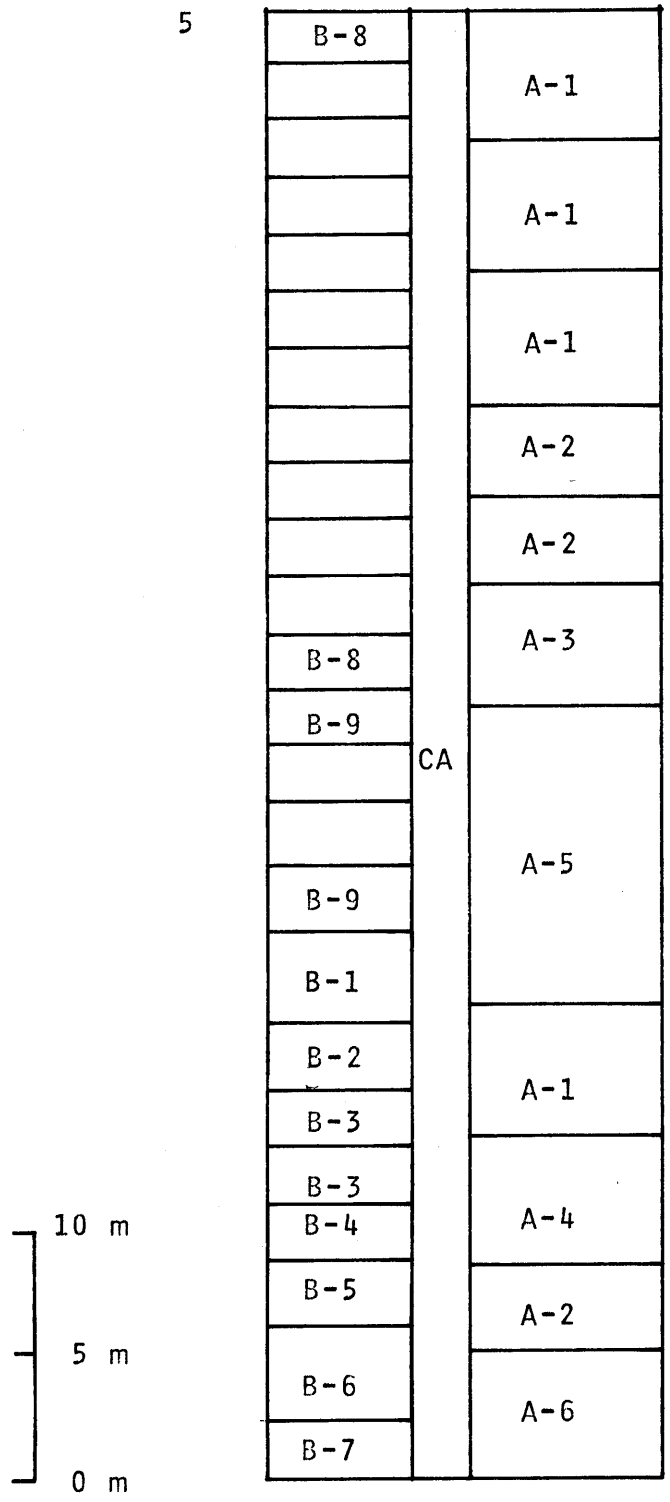


Figure 11: Example 1 – Solution 1

Neither of the two procedures is implemented; but it can easily be done for assignments in which no packages are aplit. In these cases, the necessity for an exchange can only occur for the first or last spaces in a package. The critical spaces are either put back into the list of spaces not yet allocated and a better space is used, or the better space is taken out of the list of spaces already allocated and put at the end of the same list. The addition of a circulation space is equally simple: its name has to be added to the front or the end of the list of spaces allocated in the appropriate zone.

Both procedures are needed for assignment 4.1.2 and example 1. Figure 12 shows a simulated version of the resulting floorplan. Circulation spaces are added in the two lower corners; space A-5 is taken out of its original position (cf. figure 11) and placed in the upper right corner. But in this assignment, both packages are split and assigned to two different zones. In these cases, spaces in the same package can easily be exchanged only if the better space is not yet allocated or allocated in the same zone in which the critical space is to be allocated. If the better space is allocated in a different zone, it has to be exchanged for a group of spaces which require the same area; and in order to find such a group, combinations of spaces might have to be tried among all spaces which are not yet allocated or allocated in the critical zone. This problem remains unresolved in the present version of the program.

As a consequence, the order of allocation is important whenever corner zones are part of an assignment. But this order can be easily reversed only if all spaces in a package have to be allocated in the same

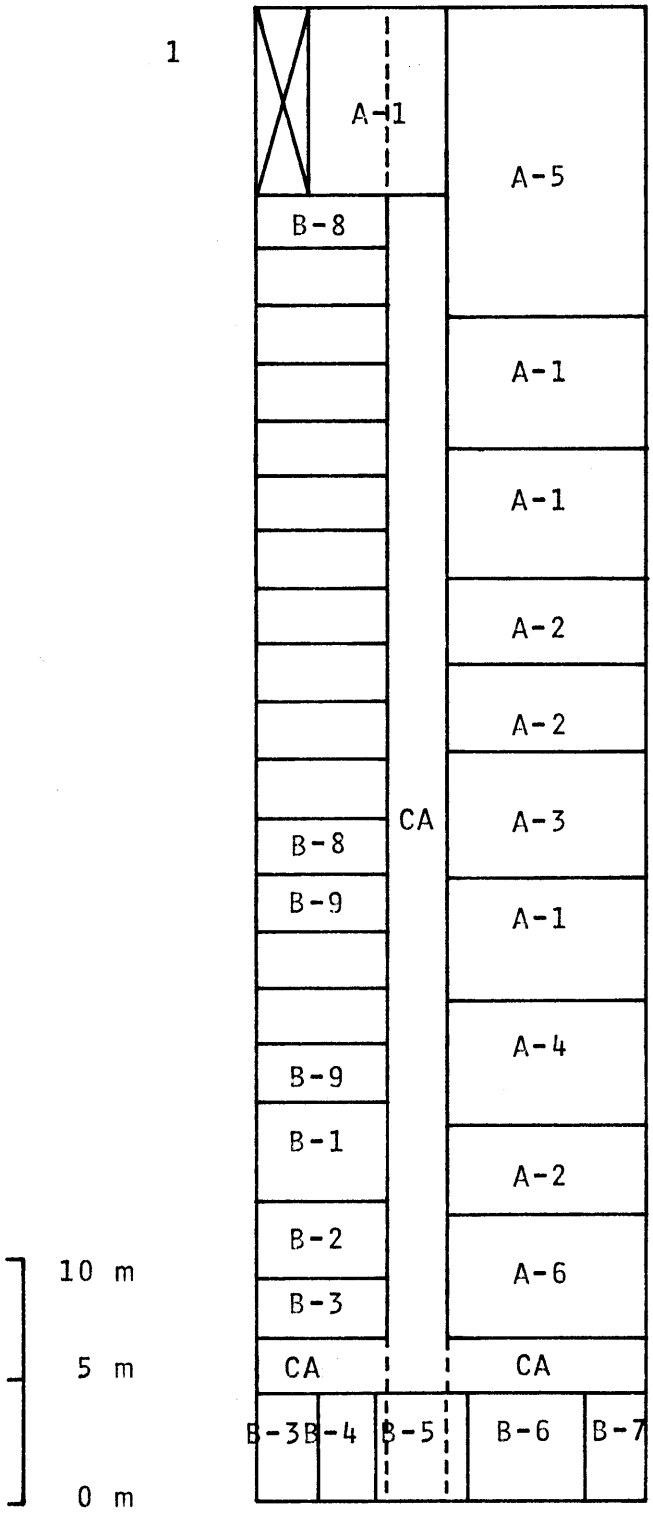


Figure 12: Example 1 – Infeasible Solution

zone. If packages are split, combinatorial difficulties might occur.

In the latter case, an additional problem arises which renders the floorplan shown in figure 11 infeasible. During the generation of feasible assignments it was implicitly assumed that packages can be split in a continuous way, neglecting the fact that spaces are discrete units and that for each sequence of spaces within a package, this package can be split only between spaces. Adjustments are nevertheless possible since the dimensions of a space and the length of a zone are variable within limits.

In general, the problem seems to become more significant as the zones become shorter and/or the spaces become bigger. This is the case in the present example: space A-1 is the last space to be allocated in the upper zone and upper left corner zone as shown in figure 12; but it is not big enough to fill both areas. Different assignments consisting of four zones and four corner zones in which package B is not split would succeed. But the program is very close to a feasible solution based on the present assignment, yet does not discover it due to a flaw in its set up. The minimum width of the zone in which the critical space is to be allocated depends on space A-5 which is allocated in a different zone. A re-computation of the width of the critical zone would render the order of allocation as shown in figure 12 feasible. But these re-computations depend also on the order in which spaces are allocated: if space A-5 is allocated in the upper short zone, the width of the long zone at the right side of the floorplan has to be re-computed. In any case, it seems unreasonable to determine the width of a zone with respect to a space which is allocated elsewhere. It is a mistake to split

packages indiscriminately.

No problems occur during the space allocation procedure for example 2. A floorplan which represents the generated solution is shown in figure 13.

Figure 14, on the other hand, shows an equally feasible solution for example 2 which will not be generated since it is based on a different decomposition into packages which starts with the smallest space as shown in table 5. From this decomposition, the solution shown in figure 14 can be found on the basis of assignment 3.2.3 which splits package A into two parts. This example emphasizes points which have been discussed before: (a) different decompositions have to be tried in order to generate a representative set of solutions; (b) the odd proportions of the floorplan stem from the mistake mentioned before: the width of the zone in which the space BED is allocated is determined by the space LIV allocated in the opposite zone; both spaces belong to a package which was split during step 2.

Figure 15 shows a floorplan based on an assignment which is tried before the succeeding one. It fails for the same reason which made assignment 4.1.2 fail for example 1: an essentially feasible configuration of spaces is not discovered due to unreasonable restrictions imposed on the dimensions of zones through spaces which are allocated elsewhere. The floorplan shown in figure 15 can be produced by a decomposition in which each space is treated as a package in itself. But the resulting solution is nevertheless infeasible in a practical sense since the circulation area is not accessible from the outside. The program considers at the present time only internal circulation patterns. A set

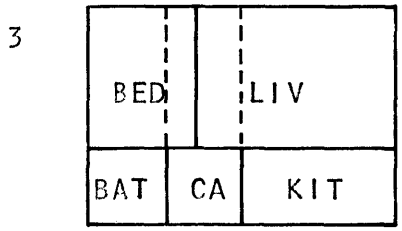


Figure 13: Example 2 – Solution

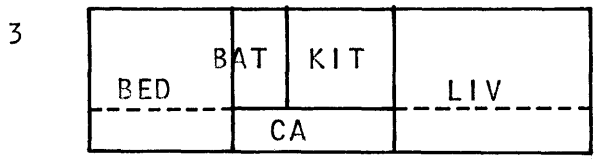


Figure 14: Example 2 – Solution

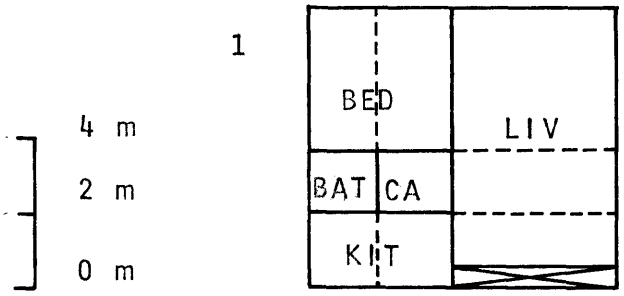


Figure 15: Example 2 – Infeasible solution

of rules is needed which assure their proper connection with external patterns.

In any case, it seems to be extremely inefficient to combine few spaces into packages if these packages have to be split in later steps as illustrated by the last examples. There might be some threshold for the number of required spaces below which the sequence of steps which was described above can be circumvented and the possible combinations of spaces tried directly.

These points indicate loose ends which have to be pursued further. I shall return to the problem of split packages in the next chapter.

At the end of step 5, feasible solutions are represented as a structure of inter-connected property lists. The representation of the solution generated for example 2 is shown in figure 16. Indicators are represented by labelled arrows pointing to the appropriate values which can be numbers, literal atoms with associated property lists, or lists themselves. The width of the zones is re-computed; the range within which these dimensions can vary becomes narrower due to the inter-relations between all dimensions.

Step 6: Graphical Output

The internal representation of a feasible solution is difficult to interpret if printed out literally. In a last step, the program translates the internal representation into a set of 2-dimensional coordinates which can be passed to a plotting or display routine This step is not implemented at all. The floorplans shown in figures 11-15 were drawn by hand.

During this translation, a last decision has to be made At the end

of step 5, dimensions are still defined within a range; but for the output routines, these dimensions have to assume single values. In the absence of any additional rule, the program might again select the most compact floorplan such that its overall length is defined by the smallest possible value. The floorplans which were shown above are based on this rule.

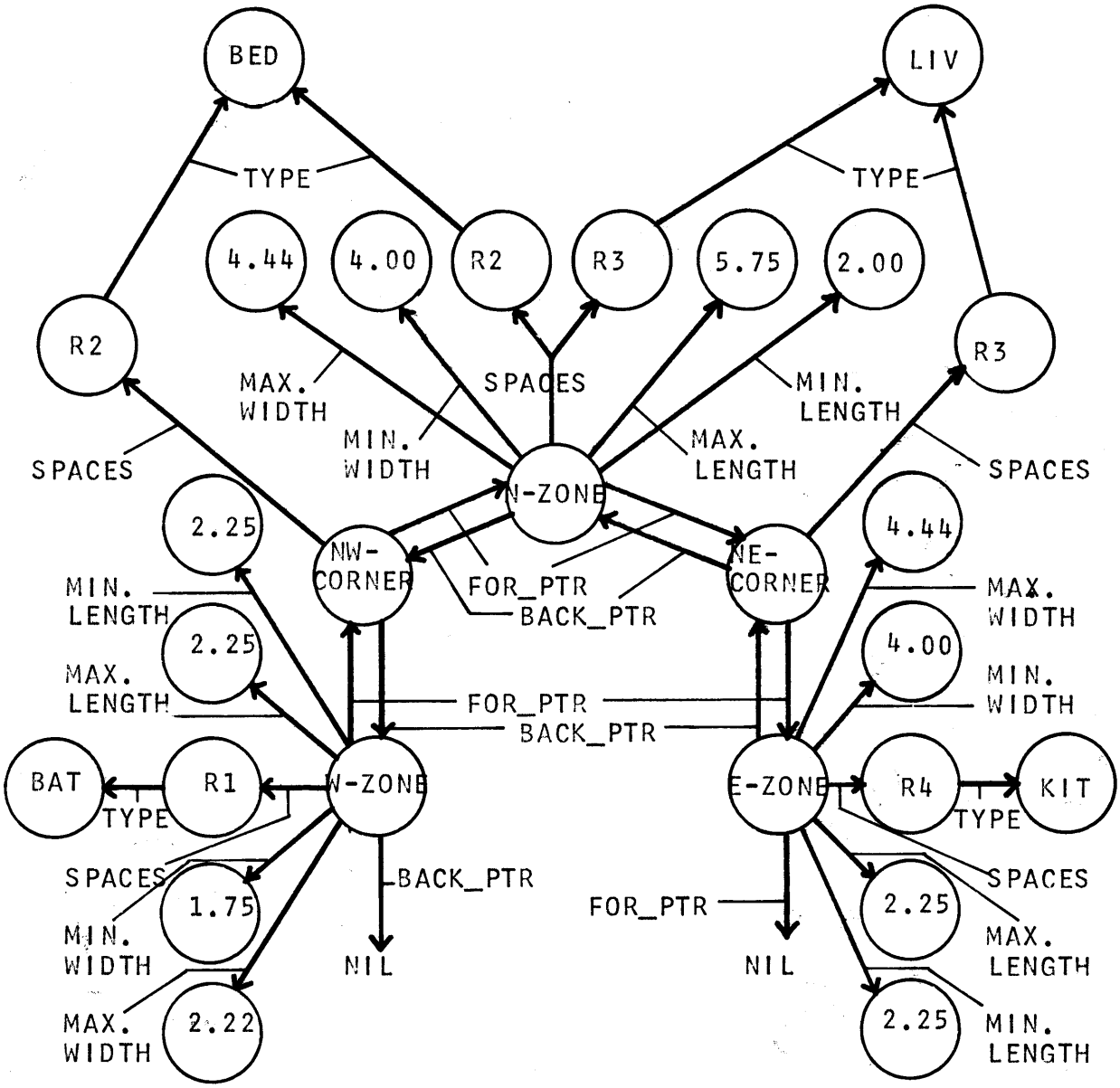


Figure 16: Example 2 – Internal representation of solution

IV. Discussion: Basic Characteristics of the Program

The problems which can be handled by a complete version of the present program are constraint from two sides:

(1) by the constraints specified in the problem statement: attributes of single spaces, relations between pairs or groups of spaces, external constraints;

(2) by the constraints which restrict the class of problems to which the problem can be applied in general: the floorplans are always of rectangular shape and are appropriate only for "chains" of spaces.

A class of legal floorplans was introduced within which constraints of the second kind can be satisfied provided that specific configurations of constraints in the problem statement do not render these floorplans infeasible. The floorplans were represented by a limited number of combinations of zones and corner zones in which spaces can be allocated and which can be grouped around a circulation area such that the pattern "chain" is satisfied, and which can be adjacent to each other only at sides of equal dimensions such that the overall shape of the floorplan is rectangular. The dimensions themselves are undefined, and their definition depends on the constraints specified in the problem statement at hand.

These two sets of constraints cannot be satisfied independently. On the one hand, the range of the possible dimensions and combinations of zones and corner zones is restricted with respect to the constraints specified in the problem statement; on the other hand, the range of the possible combinations and dimensions of the required spaces is restricted

with respect to the global constraints imposed on the floorplans.

The first characteristic feature of the sequence of operations described in the last chapter is the order in which these constraints are considered. In figure 17, the sequence of operations is drawn vertically as a single pass without loops, and the constraints specified in the problem statement are listed horizontally. Arrows indicate where these constraints enter the sequence, and short comments are added which explain the aspects under which the constraints are considered in a particular step.

There is, first of all, a preliminary test which checks whether all spaces have to be adjacent to the circulation area. These relations, taken together, form the circulation pattern "chain", and this pattern represents the global significance of the adjacency relations. On the basis of the existence or non-existence of this pattern, the program decides whether an allocation of spaces in zones and corner zones around a circulation area is appropriate, i. e. whether the class of floorplans which can be generated is adequate for the problem at hand.

In step 1, the dimensional constraints for all spaces are compared with respect to their compatibility, and this compatibility represents the global significance of the dimensional constraints: it determines a range for the width of the zones; those constraints which define the limits of these ranges assume global significance. In addition, the minimum number of required zones can be defined which makes certain possible combinations of zones and corner zones infeasible. The number of possible combinations can be enlarged if the daylight constraints

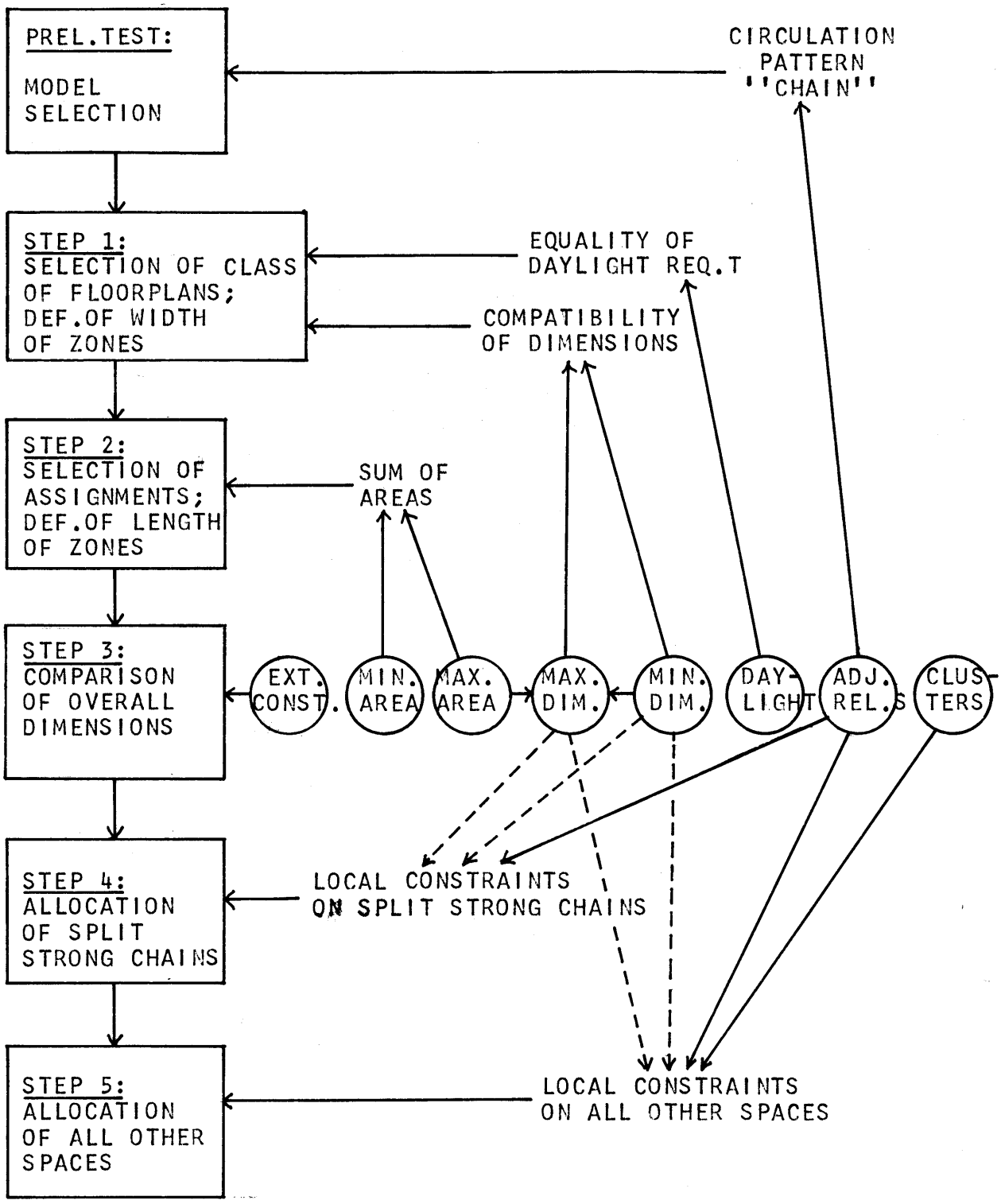


Figure 17: Global and local significance of constraints

which have been neglected so far are considered together with the dimensional constraints, as will be shown in the next chapter. The daylight constraints, again, will be considered with respect to their global significance: they will be used in order to determine whether a second class of floorplans can be introduced.

In step 2, the length of the zones is determined on the basis of their width and the sum of the area constraints specified for each space. This sum, again, represents the global significance of the area constraints. As a result, feasible combinations of zones and corner zones can be found such that the global geometric constraints imposed on the class of floorplans are satisfied.

Their overall dimensions can be computed and, in step 3, compared with the external constraints specified in the problem statement which are global by definition.

During each of these steps, only few of the constraints in the problem statement are considered, and they are considered solely with respect to their global significance. Each step, furthermore, depends on the results of the previous step: the global implications of particular constraints can only be determined if the implications of other constraints were considered before.

In the last two steps, the program tests whether there are certain configurations between the various constraints in the problem statement which are incompatible with these global constraints. Some of the constraints specified in the problem statement are considered a second time, but now solely with respect to their local significance.

If no packages are aplit, the program can concentrate exclusively on adjacency relations. It has to test whether pairs of spaces that have

to be adjacent can be adjacent in a given assignment. The areas and dimensions of each zone are adequate for the spaces which have to be allocated in these zones due to the way in which they are determined in the previous steps.

If, on the other hand, packages are split, dimensional and area constraints have to be considered a second time together with the adjacency relations. As shown above, this might introduce problems which are not resolved in the present version of the program.

The distinction between the global and the local significance of the constraints specified in the problem statement is the principle on which the sequence of steps is based. The order in which these constraints are considered, however, does not imply a weighting of the constraints: assignments can fail totally due to global or local conflicts. The order is purely operational: as the sequence is set up, local conflicts can only be detected after the global constraints are known.

The second characteristic of the sequence follows from this order. Except in cases where packages are split, the number of possibilities that have to be considered during each step is small, and the combinatorial problem remains manageable. Furthermore, the number of feasible combinations that have to be considered in the following steps is much smaller, and can be further reduced by heuristic rules.

The operations themselves consist of simple computations, comparisons of numbers, and list operations. This operational simplicity is caused by three factors: (a) the restrictions imposed on the problem make (b) a representation of the space arrangements through linear lists possible which (c) can easily be handled with the help of a language

like LISP.

Finally, the number of steps which are necessary in order to generate a solution is known: the sequence of steps is limited and actually very short.

To use S. A. Gregory's terminology (Gregory, 1969): the sequence is "tight" since the order of operations is pre-determined, as opposed to a "slack" sequence in which operations can be exchanged; and it is a "single thread" sequence since there is only one type of operation that can be performed at each step, as opposed to a "multiple thread" sequence which contains branches. The sequence is, furthermore, specialized since its application is restricted to a very narrowly defined class of problems. The latter characteristic is, in fact, a prerequisite for the former ones. The sequence can be simply structured because the constraints which can be expected and their interrelations are explicitly known. These interrelations are represented by the sequence in an operational way which is summarized in figure 17.

I do not know, however, whether this is the only sequence of operations that can be found for the present problem. It depends largely on the way in which legal floorplans are represented. These floorplans do not have to be interpreted as combinations of zones and corner zones; and the sequence changes as this interpretation changes.

V. Discussion: Revisions and Extensions of the Program

Revisions of the program are necessary. First of all, packages should not be split indiscriminately. A rule is needed, for example, which prevents in step 2 packages from being split if the width of the zones to which they can be assigned is determined by single spaces. Rather than splitting packages, different decompositions should be tried.

I propose as a general directive for these revisions to keep the operations in steps 4 and 5 simple, to resolve problems as early as possible, and not to postpone their resolution to the very last step. These revisions should concentrate on the rules which govern the decomposition in step 1 and the assignments of packages to zones in step 2. I expect these rules to have a distinctly heuristic character: they will work in certain cases and fail in others.

These rules, furthermore, can be found in a heuristic way and determined in a series of experiments, provided steps 4 and 5 are completely implemented. I suggest, in fact, such an experimental approach for the necessary revisions since programs like the present one do not have to consider all logically possible cases and difficulties as long as they work for a sufficiently large number of realistic design problems.

I have to mention here a feature of the program which has not been discussed before. The program terminates if all constraints are satisfied. But at this point, the dimensions are still defined within a range, and within this range they can be suboptimized. The floorplans suggest certain constructional, service and contents systems; if it is possible to specify a set of preferred dimensions or dimensional modules for

these systems, the final dimensions of zones, corner zones and spaces can be determined accordingly In addition, it might be possible to utilize such technical background information already during step 1 as a basis for the decomposition of the set of required spaces into packages . Those decompositions which yield preferred dimensions for the width of zones can be generated and tested first. The introduction and appropriate utilization of information of this kind seems to indicate a possible starting point for the revision of step 1.

There remains, as mentioned briefly in the last chapter, an additional inconsistency: clusters can be specified in the problem statement, but are realized only by default. They have certainly a more global significance than the local constraints considered during steps 4 and 5, but, as a look at figure 17 shows, they enter the sequence of operations too late. They are not only over-ridden by more local constraints; in certain assignments and under certain circumstances, they cannot be realized in the first place, for example, when packages containing only spaces which belong to the same cluster are split and allocated at distant ends in an assignment. The problem was caused by a first revision of the initial version of the present program. I started with a decomposition of the required spaces into subsets which reflected both clusters and strong chains on the one hand, and dimensional compatibilities on the other hand, and ended with an unknown number of elements that had to be combined. This introduced immediately the possibility of a combinatorial explosion, and it was an important decision to treat the constraints in the problem statement separately as described in the last chapter. During the following revisions, clusters were neglected.

Looking back at figure 17, I suggest considering clusters right after step 2, or in this step, as an additional criterion for the selection of an assignment, although I do not know at the present time how this could be programmed.

The heuristic rule used in step 2 to further reduce the number of assignments that have to be considered was introduced for simplicity reasons in an ad-hoc way. The rule implies that all feasible assignments are generated and compared before the preferred assignments can be passed to the following steps. I suggest turning such a "breadth-first-search" during step 2 into a characteristic feature of the program such that feasible assignments are ordered according to the likelihood that clusters are kept together, and tested in that order

In the present version of the program, the final dimensions and relations of spaces are determined in an arbitrary way. It was shown above how the final dimensions can be suboptimized provided the appropriate background information is available. A similar procedure can be applied to the relations of spaces within a cluster. Once an initial arrangement is known, the relations of spaces can be improved through successive exchanges until the total cost of traffic reaches a suboptimum (Pack et al., 1966), provided the frequency of traffic between spaces is known. It is consistent with the present approach to use clusters first with respect to their global significance for a selection of feasible assignments and to improve the relations of spaces within a cluster during a later step.

Given the suggested revisions of steps 1 and 2 can be implemented, it seems possible to regard the entire sequence of steps as a basic set

of operations which can be executed sequentially such that the class of solvable problems extends.

A second class of floorplans provides an example for such an extension. The decompositions shown in tables 3 and 4, as well as the floorplans based on them, neglect attribute 4 (daylight required or not). The resulting solutions are nevertheless feasible due to a loose interpretation of the daylight requirement. Yet the specification of this attribute suggests a very common second class of floorplans which combine the spaces for which no daylight is required in an inner core. A decomposition reflecting both dimensions and attribute 4 is shown in table 6 for example 1.

If now the circulation area is first collapsed into an area of zero width, the procedures outlined above can be applied to packages B and C alone. For the present example, assignment 4.2.1 succeeds. The dimensions of the circulation area plus core can then be re-computed and the same procedures applied to the remaining package A. An assignment similar to assignment 1.1.1 for one package succeeds. The resulting floorplan is shown in figure 18. The short supervisory routine which is needed to control these operations is not implemented at the present time; the operations and the resulting solutions were simulated.

A second possible extension was mentioned in Chapter 2. Figure 5 shows a chain of apartments consisting of chains of spaces. The sequence can be applied first to the spaces in an apartment as demonstrated for example 2; these apartments can then be regarded as larger spatial units which again can be allocated in zones around a corridor. In this case, however, the spatial units are oriented: they can be adjacent to the public

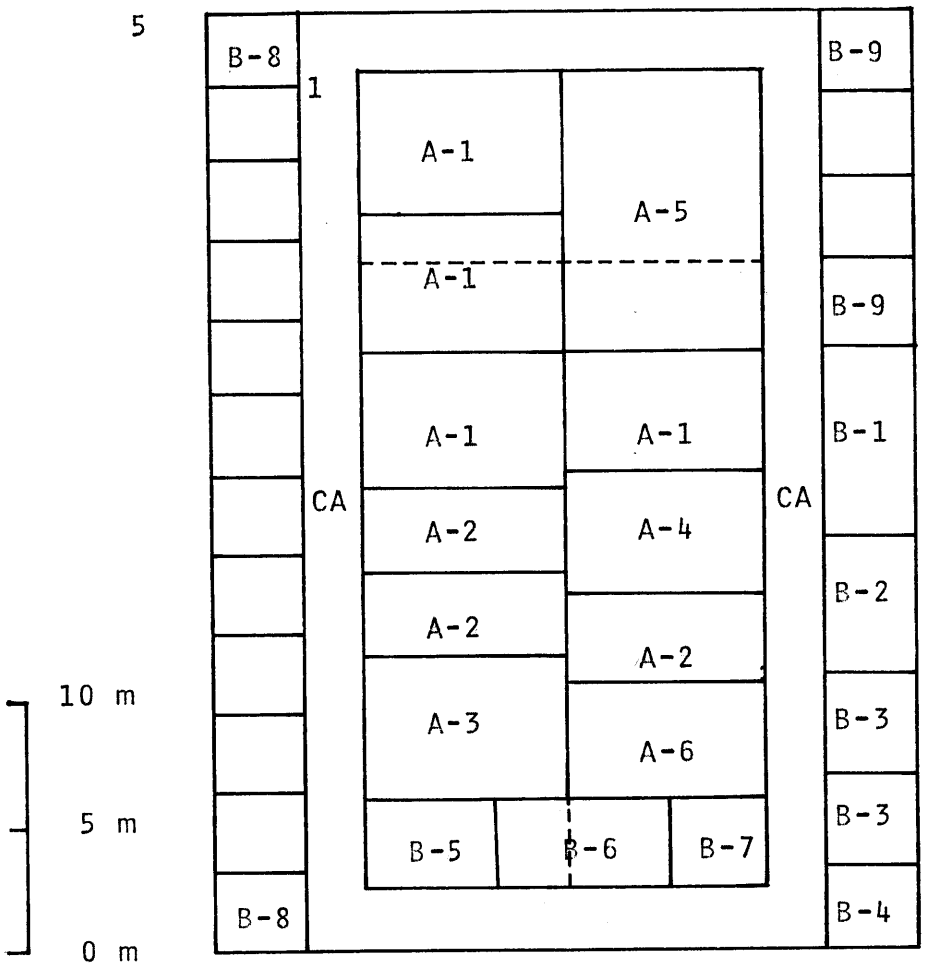


Figure 18: Example 1 — Solution

corridor only along one side which necessitates the introduction of a new constraint.

I also think, but this is not more than a speculation, that the problem shown in figure 6 could be handled, provided that the pattern "overlapping chains" can be recognized by the program and a third class of floorplans together with new decomposition rules are introduced.

But these extensions are based on the reduction of a given problem to a set of sub-problems which can be solved by a basic routine. They represent no generalization of the approach. The sequence, as it stands, cannot be generalized to models of completely different environmental systems: model representation and program execution are inseparable. I even do not know whether the sequence is a good paradigm for a study of different models since I tried only one model which, in addition, is perhaps the simplest model one can think of. But I do suggest that the basic principles which characterize the present approach: to distinguish between the global and the local significance of constraints, to consider these constraints separately, and to use the order in which they are considered as means of reducing the combinatorial difficulties of the problem, point into directions worthy of further investigations.

VI. Discussion: The Program Within an Experimental Environment

The question arises whether the considerations of the last chapter are only a matter of internal program organization, performance and efficiency, or whether programs possessing the described characteristics, provided they work for a sufficiently large class of problems, can have some importance outside the field of computer-generated design. In this chapter, I shall offer some speculations which suggest the latter possibility.

Amos Rapoport emphasizes the fact that design problems are not always "started from scratch" (Rapoport, 1969). As design problems occur repeatedly, a body of standard solutions and partial solutions emerges; building types, or models of environmental systems, belong to this body of knowledge.

He furthermore proposes to view the design process as a hypothesis-formation (design generation) and testing (design evaluation) sequence. T. Markus makes the same suggestion (Markus, 1969). Within this framework, models of environmental systems can be regarded as hypotheses which have been generated in the past.

Each of these models represents a specific way in which an activity system interacts with a building system. The global properties in which these models can be described reflect simultaneously general characteristics of the activity system and general characteristics of the building system, and they indicate how the constraints imposed by both systems can be reconciled.

If the interactions between both systems in a model are known, the

model's adequacy for a given design task can be evaluated before the design process starts. On the basis of this evaluation, a model can be rejected or accepted. In both cases, an understanding of the interactions is essential: adequate models should be considered, inadequate models should not be accepted.

To take the simple model represented by the floorplans that can be generated by the present program as an example: It is applicable to a certain class of problems (characterized by the circulation pattern "chain"); among all possible combinations of spaces that fit into this pattern, it specifies a particular arrangement (in rectangular zones and corner zones) which results in a floorplan of rectangular shape. For floorplans of this kind, certain combinations of constructional, service and contents systems are adequate, others are excluded. Knowing these properties, a designer who is confronted with a particular task can first decide whether the model is applicable. He can then evaluate whether a linear allocation of spaces in zones around a circulation area is in conflict with other requirements of the task. He can evaluate whether the overall shape which results from this allocation is desirable. And he can decide whether the appropriate technical subsystems are available and can be used efficiently. Aside from these global implications, he can furthermore evaluate specific side-effects of the global properties for the activity system under consideration. In the first class of floorplans, for example, all spaces are allocated at external walls. For spaces which do not require daylight, the designer has a choice: he can omit windows in the external wall or provide adequate means of covering them. In the latter case, the use of the spaces is less

restricted. The resulting flexibility is an implication of this class of floorplans which can be anticipated and compared with the implications of different classes of floorplans.

The acceptance of a model not only pre-determines the global properties of the resulting design; it has implications for the design process itself (Rapoport, 1969). During this process, the global properties of the model appear as global constraints which reduce the search space considerably: large numbers of possible configurations of areas and combinations of technical sub-systems are excluded from the beginning and do not have to be explored.

Furthermore, if the relations between the global constraints imposed by the model and the specific constraints imposed by the problem at hand are well understood, a decomposition of the design problem into sub-problems can be found which is process-oriented and specifies which sub-problems should be solved at which state in the design process or the design morphology. Such a decomposition is efficient only if unresolvable conflicts are detected as early as possible. It depends indeed on a precise knowledge of the interactions of constraints.

I consider a study of these models worthwhile under three aspects:

(1) the applicability of the models to specific tasks can be studied; this might lead to a set of criteria for the initial selection of appropriate models;

(2) the implications of the models can be studied, and this might lead to a set of criteria for an initial evaluation of the models;

(3) the relations between various constraints can be analyzed, and

this might lead (a) to the discovery of particular regularities on which a model-specific decomposition of a problem into sub-problems can be based, and (b) to generalizations which are relevant even for new models since some types of constraints can be found in almost any design problem.

These considerations are based on a general interest in the systematic organization of past experience.

I suggest considering these models as hypotheses of environmental systems which can be tested. I furthermore propose to carry out these tests in an experimental environment in which computers play an important part. Computer programs require a level of complete explicitness as regards assumptions which underlie the problem representations and the solution procedures. These assumptions, as well as the models themselves, can then be tested in a series of experiments.

During these tests, initial problem classifications can be changed. The present program, for example, is assumed to be applicable to problems characterized by the circulation pattern "chain". A series of tests, on the other hand, would demonstrate that certain configurations of local constraints prevent feasible solutions, for example, if there are too many spaces the dimensions of which are incompatible, although all spaces belong to the same chain. Thus, classification criteria can be found which determine more precisely when a model is applicable to certain tasks.

These results can be used outside the field of computer-generated design. They will be model-specific and different for the different models under consideration.

Once solutions are generated, they can be evaluated provided the appropriate criteria are specified. Although the problem of design evaluation falls outside the scope of this paper, I mention briefly some known procedures which can assist in these evaluations. If the programs produce solutions of a certain "practicability", floorplans, for example, which do not have to be substantially re-designed, the geometric properties of these solutions can be passed to routines which simulate the construction process (Daniels et al., 1968), or a "traffic history" (Souder et al., 1964), i. e. the movement of occupants or the transport of material in a given time period, or to routines which estimate construction costs (Barnett, 1967), or maintenance costs (heating costs for example).

A systematic documentation of the relations between the global properties of a model and its implications can be of general use.

The results, again, will first of all be model-specific. But as certain configurations of constraints occur repeatedly in different models and different tasks, a comparison of the results might lead to the discovery of regularities which can be generalized and used even in design processes where new hypotheses, i. e. new models of environmental systems, are generated.

Finally, the solution methods for different models can be compared. These comparisons, again, might lead to generalizations which are first of all important for the programs themselves. But once generalized procedures are available, they might suggest ways of describing the global properties of new models which then can be included in the series of experiments. The distinction between new and old models disappears

within the experimental context as soon as both can be programmed and tested. The tests of new models, again, are of general interest.

A close look at the way in which global and local constraints are interrelated in each model might even suggest process-oriented decompositions of design problems into sub-problems. A first hint for such a decomposition, for example, was found for the first class of floorplans: the daylight requirements can be neglected during the design phase proper.

The order in which all other constraints are considered in the present program, on the other hand, has primarily internal significance. It depends on the way in which floorplans are represented; the results demonstrate that these constraints actually have to be considered together in the same design phase. Global and local dimensional constraints and global and local relations between spaces are interrelated. Taken together, they define a single sub-problem, and the program can be considered an operational unit which tries to solve it. It has to be seen in which cases and to which degree this sub-problem is independent of sub-problems introduced by constraints which are not considered in the present program. A general process-oriented decomposition for the present model can be outlined only after such an analysis.

VII. Summary

The floorplans that can be generated by the program described in Chapter 3 are represented as combinations of zones, corner zones and circulation areas. Spaces are allocated in the zones and corner zones which are grouped around the circulation area such that the circulation pattern "chain" can be satisfied; and the zones and corner zones can be adjacent only at sides of equal dimensions such that the overall shape of the floorplan is rectangular. These are global constraints imposed on the floorplans, and it is the task of the program to reconcile these constraints with the specific constraints imposed on the required spaces in a problem at hand.

The suggested solution procedure reflects the chosen representation of the floorplans and depends on it. It consists of a sequence of operations in which the global geometric constraints imposed on the floorplans are satisfied first on the basis of the global significance of the specific constraints imposed on each of the required spaces. The program then tests whether there are local configurations of constraints which render a globally feasible combination of zones and corner zones unfeasible. The interrelations between all constraints are expressed through the sequence of steps in an operational way.

There are inconsistencies in the present version of the program which have to be eliminated. Except for these cases, the number of possibilities to be tried during each step is limited. The sequence itself is "tight" and follows a "single thread". It is, furthermore, specialized: extensions, but no generalizations, are possible.

It is proposed to view the floorplans that can be generated by the program as a representation of a model of an environmental system, and to implement programs of this kind in an experimental environment in which these models are considered hypotheses that can be tested with respect to the class of problems to which they are applicable, with respect to their performance, and with respect to effective design procedures. The results of these tests are considered relevant outside the field of computer-generated design: they might increase the knowledge about the properties and implications of specific models, or suggest generalizations useful during the generation of new models.

References

Bareither, Harlan D., and Jerry L. Schillinger, University Space Planning, Chicago: University of Illinois Press, 1968.

Barnett, Jonathan, "Computerized Cost Estimating", Architectural Record (March, 1967), 163-166.

Daniels, Robert L., et al., ICES PROJECT-I: Engineering User's Manual, Cambridge: M. I. T , Dep. of Civil Engineering, 1968.

Gregory, S. A. , "Morphological Analysis: Some Simple Explorations", Design Methods in Architecture, eds. Geoffrey Broadbent and Anthony Ward, London: Lund Humphries, 1969.

Markus, Thomas A., "The Role of Building Performance Measurement and Appraisal in Design Method", Design Methods in Architecture, eds. Geoffrey Broadbent and Anthony Ward, London: Lund Humphries, 1969.

Miller, William R., et al., "Matrix Method for Grouping an Interrelated Set of Elements", Proceedings of the First Annual Environmental Design Research Association Conference, Chapel Hill, N. C., 1969.

Pack, Ludwig, et al., "Space-Allocation and Lay-Out", Management International (October, 1966), 24-35.

Rapoport, Amos, "Facts and Models", Design Methods in Architecture, eds. Geoffrey Broadbent and Anthony Ward, London: Lund Humphries, 1969.

Souder, James J., et al., Planning for Hospitals: A Systems Approach Using Computer-Aided Techniques, Chicago: American Hospital Association, 1964.

Weissman, Clark, LISP 1.5 Primer, Belmont, Calif : Dickenson, 1967.

Whitehead, B., and M. Z. Eldars, " An Approach to the Optimum Layout of Single-Storey Buildings", The Architects' Journal (June 17, 1964) 1373-1380.