

MINISTÉRIO DAS OBRAS PÚBLICAS

**LABORATÓRIO NACIONAL
DE ENGENHARIA CIVIL**

LAYOUT SCHEMES FROM ADJACENCY GRAPHS:
a case study in problem solving by theory building

Trabalho apresentado como tese de doutoramento na
Universidade de Brunel, integrado no PLANEAMENTO
DE ESTUDO NO DOMÍNIO DOS EDIFÍCIOS

LMP

INEC

Lisboa, Fevereiro de 1974

MINISTÉRIO DAS OBRAS PÚBLICAS
LABORATÓRIO NACIONAL DE ENGENHARIA CIVIL

SERVIÇO DE EDIFÍCIOS
DIVISÃO DE ARQUITECTURA

Proc.86/14/4030

Manoel Pereira

LAYOUT SCHEMES FROM ADJACENCY GRAPHS:
a case study in problem solving by theory building

Trabalho apresentado como tese de doutoramento na
Universidade de Brunel, integrado no PLANEAMENTO
DE ESTUDO NO DOMÍNIO DOS EDIFÍCIOS

Lisboa, Fevereiro de 1974

CONTENTS

Acknowledgements	IX
------------------------	----

CHAPTER 1

THE CASE FOR PROBLEM ORIENTED THEORY BUILDING

1. INTRODUCTION	1
2. DERIVATION PROBLEMS AND FORMATION PROBLEMS	5
3. GOAL - DIRECTED PROCEDURES	7
4. HEURISTIC, EXHAUSTIVE AND TRIAL-AND-ERROR SEARCHES..	10
5. PROBLEM-SOLVING POWER OF DEFINITIONS	15
6. GAME PLAYING AGAINST A THEORY	17
7. A NOTE ON MECHANICAL THEOREM PROVING	18
8. REFERENCES	18

CHAPTER 2

FROM AN ARCHITECT'S POINT OF VIEW

1. INTRODUCTION	19
2. METHODS	21
3. GENERAL COMMENT ON RESULTS	22
4. NOTES ON PREVIOUS WORK	25
5. REFERENCES	29

CHAPTER 3

GRAPH THEORETICAL NOTIONS

1. INTRODUCTION	32
2. INFORMAL PRESENTATION OF GRAPHS	33
3. FORMAL DEFINITIONS	35
Graph	35
Subgraph	35
Finite and Simple Graphs	35
Path	36
Polygon. Segment. Cycle	37
Connectedness. Separateness	37
Addition of Cycles	38
Example	38
Nets	41
Examples	41
Note on Notation	42
Removing a node from a graph	43
Realization. Planar Realization. Planar Graph	43
Theorem (Kuratowski).....	44
Planar Mesh	45
Theorem (Planar mesh)	45
Face. Contour	45
Example	45
Note on the Class of Graphs Considered	46
Equality Between Planar Realizations	46

Euler's Theorem	46
Uniqueness of the Contour. Boundary	47
External Face	47
Interior Node. Interior Edge. Exterior Nodes and Edges.....	47
Example	48
Dual Graph	48
Properties of a Graph and its Dual, for a Given Realization..	50
Avoiding Segment	51
Segments determined by V in P	51
Branch Graph	51
Bridges. Residual Segments. Peripheral Polygon	51
Three planarity Theorems	53
4. REFERENCES	54

CHAPTER 4

PROBLEM FORMULATION AND PROFILE OF ITS RESOLUTION

1. INTRODUCTION	55
2. INITIAL HYPOTHESES	55
3. INITIAL PROBLEM FORMULATION.....	58
4. CHOICE OF A PROBLEM REPRESENTATION SPACE	58
5. PROBLEM DECOMPOSITION	62
6. LAYOUT SCHEMES FROM GRAPHS	66
Conditions A	67
Conditions B	68

Justification of conditions A	68
Justification of conditions B	69
Comments on conditions A and B. Conditions C	71
Five "geometrical" properties	74
References	78

CHAPTER 5

OBTAINING ALL POSSIBLE LAYOUTS FROM A REALIZATION

OBEYING CONDITIONS 'A'

1. INTRODUCTION	77
2. THE COLOURING METHODS	80
General Considerations	80
Colour Properties and Colour Consequences	83
Components of interior nodes with four edges	97
Property R4	99
Components of quadrangular faces	99
Example	102
Property Q_R	103
General Components	103
Example	104
Property G	104
The colouring tests	107
Colour operations and colour options	108
Example	113
The colouring process	117
Example	122

The contour form	129
Theorem C0-2	131
Theorem of contour permissibility	135
Theorem of contour rectangularity	136
Imposing other contour forms	136
Other examples of contour forms	141

CHAPTER 6

HOW TO DRAW A MODULAR LAYOUT FROM A PERMISSIBLY
COLOURED REALIZATION

1. INTRODUCTION	145
2. RECTANGULAR CONTOUR MODULAR LAYOUT	145
Remark	151
Obtaining the modular dimensions	151
Another example	155
Drawing Modular Indented Layouts	158
Example of the workings of the automata	163

CHAPTER 7

HOW TO FIND PERMISSIBLE DESIRED DIMENSIONS FOR A
LAYOUT

1. INTRODUCTION	168
2. ASSIGNING DESIRED DIMENSIONS PERMISSIBLY	169
Informal comments on the distribution of dimensions	188

3. FORMAL PRESENTATION OF THE PROBLEM OF DIMENSION DISTRIBUTION AND OF METHODS FOR ITS RESOLUTION	193
Restatement of the problem	193
Abstract Problem Formulation	194
A Heuristic Algorithmic Resolution for the Distribution Problem	199
Purpose of HPR 1 and HPR 2	200
Description of HPR 1	200
Definition of procedure CONCILIATE	201
Definition of procedure DISTRIBUTE	201
Description of HPR 2	204

CHAPTER 8

HOW TO OBTAIN THE PLANAR REALIZATIONS OF A GRAPH AND MODIFY THEM TO MEET CONDITIONS ' A '

1. INTRODUCTION	208
2. FURTHER PROBLEM DECOMPOSITION	209
Theorem (realizations of 3-connected graphs)	210
3. EXAMPLES	211
4. RESOLUTION OF THE SUBPROBLEMS PRESENTED	219
Remark about the main program	221
5. THE PLANARITY AND PLANAR REALIZATIONS ALGORITHM: AN OUTLINE	222
Example	225

CHAPTER 9

SOME FURTHER DEVELOPMENTS AND LINES OF INVESTIGATION

1. INTRODUCTION	234
2. FURTHER DEVELOPMENTS AND LINES OF INVESTIGATION....	234
3. COMMENTS	239

CHAPTER 10

THE PROGRAM OUTPUT

1. INTRODUCTION	241
2. PROGRAM AND COMPUTER PARTICULARS	241
Program options	242
3. AN EXAMPLE	243

ACKNOWLEDGEMENTS

It is my kind obligation to acknowledge the stimulating interaction of my friends Luis Monteiro and Fernando Pereira, and its pervasiveness within and out of the bounds of this thesis.

My utmost recognition goes also to Prof. Gordon Pask, who always pursued his rôle, as a supervisor and teacher, into the realm of friendship.

To Architect Nuno Portas goes my deepest appreciation for having given me, unremittingly and from the very start, the opportunities and the advice which made possible the present work.

Thanks are also due to my superiors at one time or another at the National Laboratory for Civil Engineering, in Lisbon, where the greater part of this work was carried out. They are Architect Nuno Portas, Head of the Department of Architecture, Engineer Artur Ravara, Head of the Departments of Applied Mathematics and Applied Dynamics, and Engineer J. Ferry Borges, Sub-Director, all of whom have given me incentive and support during the course of this work.

Finally, I want to express my gratitude towards Calouste Gulbenkian Foundation, Lisbon, from whom I received, at various occasions during this work, the funds which permitted me to carry it out.

LAYOUT SCHEMES FROM ADJACENCY GRAPHS:

A Case Study in Problem Solving by Theory Building

LUIS MONIZ PEREIRA

Brunel University

Ph. D. Thesis

1974

The mental features
discoursed of as the analytical,
are, in themselves,
but little susceptible of analysis.

Edgar Allan Poe

To

Gordon Pask

Luis Monteiro

Fernando Pereira

Helder Coelho

with great esteem.

SUMÁRIO

Neste trabalho é construída uma teoria para resolver uma classe de problemas abstractos (definidos de modo preciso no capítulo 4), que são imediatamente interpretáveis em termos arquitecturais.

A teoria referida foi construída com vista a suportar processos de resolução de problemas susceptíveis de tradução quase directa numa linguagem de programação: de facto, foram escritos programas para computador que incorporam o espaço de representação da teoria e as estratégias de resolução de problemas definidas nesse espaço para resolver a classe de problemas considerada.

Em termos abstractos, a classe de problemas que foi encarada resultou do seguinte "desideratum": Dada uma lista de adjacências impostas e de outras não permitidas, num conjunto de espaços planos rectangulares, e dados intervalos dimensionais restringindo as extensões dos espaços e das suas adjacências, procurar e gerar todas as disposições no plano desse conjunto de espaços que satisfazem aos requisitos impostos, e elaborar programas para computador que executem os algoritmos desenvolvidos, bem como providenciar à interacção entre utilizador e programa com vista à alteração dos intervalos dimensionais estipulados, especialmente no caso em que surjam incompatibilidades.

Os capítulos 1 e 2 destinam-se a fornecer enquadramento adentro da metodologia de resolução de problemas e adentro da metodologia dos processos computacionais em arquitectura, a partir dos quais o restante trabalho pode perspectivar-se. O capítulo 3 fornece ao leitor as noções teóricas sobre grafos introduzidos em capítulos posteriores no desenvolvimento da teoria; o capítulo 4 apresenta a classe de problemas; os capítulos 5 a 8 mostram como a teoria foi construída para tratar com essa classe de problemas, e discutem vários aspectos da metodologia de resolução de problemas seguida; o capítulo 9 ilumina algumas linhas de investigação futura, e o capítulo 10 conclui o trabalho com uma demonstração de resultados do programa para computador elaborado.

ABSTRACT

In this thesis a theory is built to solve a class of abstract problems (defined precisely in chapter 4) readily interpreted in architectural terms. The referred theory was built so that it could support problem-solving procedures susceptible of an almost direct translation into a programming language; computer programs were in fact written which embody the theory representation space and the problem-solving strategies defined in that space for solving the class of problems under consideration.

In abstract terms, the class of problems envisaged results from the following "desideratum": Given a list of imposed and of non-permissible adjacencies among a set of planar rectangular spaces, and given dimensional intervals constraining each space and each adjacency, to search and generate all possible layouts on the plane of such a set of spaces which satisfy the requisites, and to provide for a computer implementation of the algorithms developed, as well as for user/machine interaction in view to the alteration of the dimensional intervals imposed, especially in case incompatibilities arise. These layouts must also respect restraints upon their contour.

Chapters 1 and 2 are meant to provide a problem-solving and a computational architecture background against which the work may be perspectivated. Chapter 3 furnishes the reader with the graph theoretical notions introduced in later chapters for developing the theory; chapter 4 presents the class of problems; chapters 5 to 8 exhibit how the theory was constructed to deal with that class of problems, and discuss various aspects of the problem-solving methodology followed; chapter 9 illuminates further developments, and chapter 10 concludes with a demonstration of program output.

CHAPTER 1

THE CASE FOR PROBLEM ORIENTED THEORY BUILDING

1. INTRODUCTION

Considerable experience has been accumulated in the last fifteen years on the construction and operation of relatively complex computer-based problem-solving systems, of a great variety of types.

Study of this body of experience is producing a set of concepts, methods, strategies, and insights that can be regarded as the beginnings of a METHODOLOGICAL BASIS for the design of problem-solving systems.

Today's computers - their hardware and software - provide us, furthermore, with a SUFFICIENT TECHNOLOGICAL BASIS for embodying problem-solving processes within problem-solving environments.

Integration of the methodological and technological resources should take place within a theoretical framework. The THEORY underlying the present thesis is employed in this way. It was developed using computer and problem-solving oriented concepts from the start. Theory building becomes then inextricably interwoven with problem-solving RESOURCES and not just problem oriented. Furthermore, such a theory becomes CONSTRUCTIVE in character, because its reasonings will be closely imaged by corresponding problem-solving actions.

In essence, a problem statement contains a description of the solution object in one form. A request is made to find a description of this object in another, specified form. In carrying out such a transformation, the choice of appropriate combinations of problem-solving control procedures and problem formulations becomes the core of the problem of designing a computer

-based problem-solving system.

The essential assumption here is that at some point, in between the initial and perhaps incomplete emergence of a problem and the attainment of its solution, a problem statement is formed which governs the rationale of the subsequent solution-seeking actions. Of course, stating a problem may require an entire new system of concepts and/or rules to deal with them. It may give rise to a theory.

More often than not, a problem statement is not communicated in its entirety once and for all. Often it is presented incompletely, without an explicit specification of a data base. The problem solver must then complete the problem statement by adjoining to it an appropriate data base, either communicated from previous knowledge or acquired anew and updated alongside the problem-solving process itself.

The data base D that enters into a problem statement contains general descriptive information about the problem environment (types of objects and properties of their relationships, general rules of operation in the environment, etc.) in terms of which the problem conditions can be interpreted, and problem solving actions generated. In general, the data base D is a SYSTEM OF CONCEPTS (a domain of knowledge, a theory, a formal system) within which the problem has a definite meaning. Most important to our purpose is the case where it is a theory, and one especially conceived for the problem or class of problems in question, and where, furthermore, its deductions are CONSTRUCTIVE in a computer oriented sense.

To the extent that the description of a problem in a problem statement is not unique, the choice of different descriptions usually influences the solution-seeking process for the problem in question, and it becomes impor -

tant to consider the notion of PROBLEM REPRESENTATION. The representation of a given problem P is the component of the problem statement for P which describes the desired solution of P, and also specifies the system of concepts within which this description is formulated.

Choice of a problem representation determines a SPACE where procedures will be devised for searching for a solution. It is up to such procedures to carry out an efficient search by using in the best possible way the representation specified.

To realize such a process it is necessary to define STATE, and the associated notions of TRANSITION between states. Moreover, a language has to be constructed for describing states and state configurations (later called SITUATIONS) as well as other essential entities: for example, problem specific information (initial and terminal conditions, conditions for applicability of operators and knowledge about properties of solutions).

Choice and construction of a description language (the key requirement of the last paragraph) involves:

- a) specification of the primitive objects in a universe of discourse together with certain properties, named by predicates in the language.
- b) appropriate functors, connectives and so on.
- c) a description scheme with paradigmatic "concepts" (i.e. classes of objects and compound objects).
- d) operators that act upon objects or concepts.
- e) criteria of relevance.

All this requires considerable knowledge about the problem or class of problems under consideration. Further there is a real difficulty with expressing this knowledge in a form expeditiously implemented by machine proce -

dures.

Surely, if such knowledge is integrated into the body of a THEORY, and if such a theory has been built with the procedural problem-solving resources in mind, one can expect the interaction between both to become facile and fruitfull.

Now, the formulation of an "initial" problem representation by human beings is rarely determined by consideration of problem-solving resources or, for that matter, of strictly relevant information. If the problem under consideration cannot be processed directly by a given computing system (because it is not formulated in a manner that can be easily interpreted by the system in question) then the "initial" problem statement must be reduced to an equivalent formulation which can be handled directly by the system.

Such a reduction entails the development of a METHOD OF SOLUTION for the problem, and its expression in the form of a PROCEDURE which consists of statements that are interpretable and executable by the computer according to a WELL-DEFINED program. The repertoire of statements that can be used in forming a procedure is determined by a PROCEDURAL LANGUAGE L , which, henceforward, we take to include the description language noted previously and for which the system is assumed to have the necessary linguistic and processing capabilities so it will respond effectively to any procedure stated within L . Such a system is characterized by L .

The closer the language L (and the procedures described in L) are to the supportive environment of a theory, the easier it is to interpret and the easier it is to implement a procedural solution in the theory. On the other hand, the richer the interconnections and the more direct the trans-

lation of the theory's system of concepts into a procedural language, the more CONSTRUCTIVE will be the proof of existence of a solution for some problem in the theory: i.e., if it shows that a solution does exist it will also show how to construct it.

2. DERIVATION PROBLEMS AND FORMATION PROBLEMS

We first consider a broad classification of problems which is based on significant distinctions between two main approaches to the construction of solutions in problem-solving systems. The two approaches correspond to two important families of problems: derivation problems and formation problems (Amarel, 1970). These families correspond closely to the two types of problems discussed by Polya, namely "problems to prove" and "problems to find".

In DERIVATION PROBLEMS, or "problems to prove" the situation is roughly as follows: We are given problem conditions in the form of properties that the solution BOUNDARIES must satisfy (that is, initial and final conditions as well as known conditions on some of the sub-problems, if there are any). We are then asked to find the solution by constructing some path, in an appropriate space, inside the known boundaries, by using various path building routines in accordance with given rules. The construction proceeds, by "anchoring" the solution path on some (or all) of the sub-problem or solution boundaries and by extending it piecewise to meet other boundaries or the path segments that develop from them. Deductive reasoning problems, where a derivation is sought within a formal system, are typical of this approach.

In FORMATION PROBLEMS, or "problems to find", the relationship between the problem conditions and the structure of a solution is more complex than in derivation problems. The determining feature of a formation problem is that the language of problem conditions is distinct from the language describing (or prescribing) resources and the possible methods of combining them to build a solution. Thus the language of conditions and the language of resources must be reconciled in an adequately structured problem representation space. The richer the known and exploitable relationships between the two languages the easier it is to solve a formation problem. The structuring of the representation space can be carried out by the development of a THEORY combining the necessity for structure WITH the problem-solving resources which make it operational.

In general, both problem-solving resources and methods (regarded as a body of knowledge) should be distinguished from the strategy used for their implementation on a given problem-solving system and, furthermore, discussion of the efficiency of the implementation kept apart from considerations on the efficiency of the problem-solving approach which permeates the representation space for a given problem.

To construct a solution is, in a formation problem, to build candidate objects with the available construction resources, and to TEST them against the given problem conditions. The candidate objects have the status of solution hypothesis. The essence of the problem-solving activity will be to generate a sequence of such hypotheses which converges to the desired solution. The subject matter of this thesis is an example of this type of approach.

3. GOAL-DIRECTED PROCEDURES

In the development of a path in a derivation problem, it is normally possible to go from a given state in the representation space to several other "next" states of the same space: the particular transition that is effected depends on a choice amongst applicable operators. Thus, there exists a NONDETERMINISTIC relationship between a given state in the development of a derivation and the subsequent state: the association between a given state and the operator-selecting decisions are nondeterministic.

"Nondeterministic association" does not have the connotation of a 'probabilistic coupling between problem states and the decisions that are made to change them". It is conceived in terms of the existence of several alternative decisions that are A PRIORI applicable if a state reached. Furthermore, it is assumed that a definite choice between decision routines is not possible until some or all of the decision routines are tried, and their EFFECTS are made explicit and explicitly assessed IN THE LIGHT OF GIVEN GOALS.

Given a set of nondeterministic associations, consider the decision-making mode whereby, in a given problem state a scan is carried out over alternative possible decision routines. In such a situation, some or all of the decision routines are pursued to a certain depth, ~~their~~ evaluated and, finally, one of them is chosen for ^{application} appreciation. Such a choice is based on an explicit scheme of purposeful reasoning that combines rules for ordering the scan over A PRIORI alternative decision methods, for controlling the depth of lookahead into the consequences of applying each method, for evaluating these consequences and for choosing one decision routine, on the evidence provided by the evaluations.

This scheme of reasoning is at the core of an important class of solu-

tion methods for derivation problems, where a controlled search is carried out for a SEQUENCE OF STEPS, each of which based on a given set of non-deterministic associations. These methods are suggestively called GOAL-DIRECTED METHODS, and the procedures that correspond to them GOAL-DIRECTED PROCEDURES. The class of goal-directed procedures includes simple backtracking algorithms, and other methods used in artificial intelligence. HEURISTIC SEARCH PROCEDURES are typical artificial intelligence methods. In this case the rationale for choosing some of the decision functions that determine the dynamics of search stems from various principles of plausibility, which can eventually be made precise with the assistance of a supporting theory.

A goal-directed procedure that is expressed in a relatively low-level procedural language (say ALGOL) when compared to the high-level processes involved in theory formation, produces, during its execution, a deterministic sequence of computational processes, one for each statement in the procedure. It is not distinguishable, in its symbol manipulating detail, from any other NON-GOAL-DIRECTED procedure. To make the argued distinction we must look at higher-level structural properties of such procedures to see the organizational principles that are characteristic of their underlying goal-directed method. In the end it is only recourse to the environmental theory in which the problem becomes defined and solved that can elucidate the goal-directedness of the procedure and the semantic content of the decision functions governing the search. In such cases the goal-directedness only becomes clear when a comparison is made, at the theory's higher level of discourse, between the successive states arrived at by the search and the (desired) states permitted by the theory. The disparity between them is what

provokes the necessary feedback mechanism for guiding the search. The disparity measures will motivate the choice between decision functions.

The more a procedure's search processes are governed by large numbers of non-deterministic associations and by heuristic rules of decision, the more important it is for the procedure designer to work with a suitable high-level system of ideas for shaping his procedures. That is especially true if the decision rules are to be readily modifiable by the procedure designer or by another procedure, incumbent upon exercising control over the process as a whole, in pursuit of its goals.

The notion of a goal, as applied to a "natural" system, has a completely different conceptual status for a designer whose objective is to specify an "artificial" system in such a way that it will behave purposively (preferably in the specific manner desired by the designer), when it is set to solve problems of a certain class. Given the statement of a problem selected from this class, the system's goal will be to satisfy the request conveyed by this statement, i.e., to find a solution for the problem. Thus, in the present context, a goal is a problem statement that the system "wants to satisfy". In other words, it is a problem statement that has control OVER the operation of the system. Unlike the situation with natural systems, here a goal has the status of an explicitly formulated control input.

There can coexist, of course, several concurrent, independent, or con conflicting goals, and the sub-goals they generate will not always be free of col lision. The disparity between one and another is yet another pretext for the intervention of theory-based rules aiming at "best" compatibilization.

Now, a goal-directed procedure proceeds - via the choice and application of operators - to reduce the initial goal to new goals, or sub-goals, that

are more easily attainable according to the evaluation functions it embodies, or which offer a marked increase in the overall efficiency in solving the problem. The process of SEARCH for sub-goals, thereby converting the problem into sub-problems or vice-versa, continues until: a) the initial goal is completely satisfied (giving a solution to the problem, along with solutions to the constituent sub-problems), or b) there is evidence that no subgoal can be satisfied (this gives a definite "no solution" answer unless the problem conditions can be modified and the search can be ^{resumed after} some form of compatibilization of requisites ~~further resumed~~), or c) a fixed amount of problem-solving effort is exceeded without a definite result ^{or} an inconclusive outcome or a failure, depending on one's attitude). During the search for solution it is usual to find that several operators are applicable to a given state. This reflects the non-deterministic associations induced, in the problem being handled, by the theoretical environment in which the solution process acquires a meaning.

4. HEURISTIC, EXHAUSTIVE AND TRIAL-AND-ERROR SEARCHES

The rules of selection between equally applicable operators allow a definite choice to be made by virtue of DESIRABILITY evaluation. Desirability is judged on the basis of assessments of values and costs for a situation (e. g. estimates of closeness to solution, expected efforts associated with the development of a solution path from the situation, prospect of easiness and of fitness of an inevitable compatibilization of requisites). In addition, there can be rules for ATTENTION CONTROL, i.e. to what "next" situation should the attention of the procedure be directed, so that further exploration from that situation can take place, and rules for determining the DIRECTION OF APPROACH to be used in the application of operators, i.e. once a situation

is chosen for continuation of search what direction the search should take (e.g. what sub-problems are to be explored, or what backtracking is to be made).

The selection of rules for attention control and for direction of search reflect specific doctrines of search, such as 'pursue the search in depth, and consider other alternatives only if there is sufficient evidence that the current line of search is fruitless', or 'work back from the terminal situation to the source situation', or 'extract all possible consequences of the latest trial before embarking on another'.

These rules are largely conceived of as heuristics that summarize EMPIRICAL (often statistical) evidence about the problem class under consideration, and also attitudes and doctrines about how to go about searching for a solution in this class. But heuristic rules can be of a more THEORETICAL origin, and dealt with in a more theoretically supported way. (Recent progress on heuristic evaluation functions in problem-solving AND/OR graphs is a case in point, as are heuristic resolution strategies in theorem-proving which have to be theoretically pondered as to their efficiency and their impingement on completeness.) Moreover, theoretically inspired heuristics may eventually become rigorously justified, once ~~the~~ ^{their} precise meaning is made clear through subsequent development of appropriate concepts within the theory itself. Such is the case for heuristic rules guiding strategies ^{when} ~~are~~ analysed with respect to efficiency of first solution, non-redundancy of search, or exhaustiveness in regard to search space. These properties, when occurring, can sometimes be PROVED, and the desirability of the corresponding heuristics ESTABLISHED within the framework of the given theory.

Heuristic rules provide a sensible balance between blind exhaustive systematic search and blind trial-and-error search. This result is achieved by

delimiting the regions of state space where trial-and-error search should be carried out, eventually delineating by such trial-and-error search further regions to be explored while disregarding others. The example of chess ~~might~~ is illustrative. In this game, critical regions of the board are usually selected, in the first place, for further analysis by the player.

This permits him to exclude from consideration in a methodical fashion board situations which would arise from some of the moves, whilst retaining for subsequent exploration those situations which require extra subdivision into possible outcomes.

Tichomirov and Poznyanskaya (1966) found that master chess players looked at boards for less time than novices and remembered positions more accurately, with the expert turning his gaze from one significant feature of the board to another, while the novice searched randomly. If the pieces are randomly arranged on the board, so that the expert has no meaningful search strategy, his performance on memorizing the board is similar to that of the novice.

While for both blind exhaustive and blind trial-and-error search strategies, no information is needed to carry out the search since there is no occasion to make selections before a path to a goal has been found, in heuristic search strategies the main concern is making the appropriate choices along the way, and relevant information becomes crucial.

If it is not empirically gathered, from where can such information be obtained?

The answer is "from the structure of the environment in which the problem acquires a meaning". The more structured the environment, the greater the advantage to be gained in accumulating such information, especially if the

environment is characterized by and gives an interpretation to a theory.

There are two reasons for this advantage: 1) Heuristic search (in contradistinction to blind exhaustive and blind trial-and-error search) makes use of all the information that can be mustered about the problem to be solved, especially information pertaining to GLOBAL properties. 2) In a structured environment it is possible to establish tighter coupling between the EVALUATION processes and the COMPUTATION processes which make up heuristic search procedures. Whilst evaluation normally precedes computation, some or all of the computations that are performed by the EVALUATION may already be, in part or in whole, the COMPUTATION subsequently required. In the extreme case, evaluation is absorbed as part of the computation, i.e. possible outcomes are weighed up in the course of problem solving or, vice-versa, computation becomes embodied within evaluation.

Close interaction between the two processes (problem solving and evaluation) leads to an overall computational efficiency. But close interaction is only feasible if the problem environment is computationally tractable within limits imposed by the available processor. This is a general precondition for subsequent computations to profit from the computing effort spent upon evaluation. By way of a compromise, some effort may be expended in weighing ^{out} ~~out~~ beforehand (by use of an overlooking procedure, for example) the proper balance between the two procedures.

An example is as follows. When solving a quadratic equation in one variable on the real numbers, $AX^2 + BX + C = 0$, its solutions can be computed by the well-known formula:

$$X = \left(-B \pm (B^2 - 4AC)^{1/2} \right) / 2A$$

Admit that B and C are not zero. What happens if $A = 0$? (question)

mark).

The equation then becomes a simple first-degree equation in one variable, $BX + C = 0$, but the above formula gives an undetermined value as solution. If we are to use the formula in a computer program we must be ready to first evaluate A , and only then compute the above formula in the case A is nonzero, or otherwise, in the case $A = 0$, to use the formula $X = -C/B$.

If we only want ONE solution, a way to obviate to the separateness of evaluation and computation, is to transform the initial equation with the substitution $Y = 1/X$. The equation becomes then $CY^2 + BY + A = 0$, and its solutions are $Y = (-B \pm (B^2 - 4AC)^{1/2})/2C$.

Therefore, $X = 2C/(-B \pm (B^2 - 4AC)^{1/2})$.

We can then choose as the wanted solution the one given by $X = 2C/(-B + (B^2 - 4AC)^{1/2})$, where, by hypothesis, B and C are nonzero.

Now when $A = 0$, X takes the appropriate value $-C/B$.

This example illustrates that whereas in the first instance one had to evaluate A before embarking on the computation leading to a solution, in the second, modified form, such evaluation is already BUILT-IN, through the agency of a theory, into the computational procedure itself.

As a further example, we present a similar trade-off between evaluation and action, occurring in control theory as a trade-off between system identification and system control (Arbib, 1972).

Identification procedures are very important for exercising control. For instance, supposing one wished to control a system, but did not know the exact values of the parameters figuring in the dynamic equations of the system. Their values might indeed vary according to circumstances.

Then, rather than build a controller specifically designed to control

any system specified by that set of values, *and*

~~Then~~ rather than hook up the controller directly to the system to be controlled one would interpose an identification procedure. The controller would thus work at any time upon the set of parameters which the identification procedure provides as the best estimate of the real system's only parameters at that time.

Such a controller, when coupled to an identification procedure, is precisely what is often referred to as an ADAPTIVE CONTROLLER: it adapts its control strategy to changing estimates of the dynamics of the controlled system.

We note, without elaboration here, that it might also be necessary to have the identification procedure apply test signals to try out various hypotheses about the parameters of the controlled system. One would then have to design a strategy to trade-off the loss of optimality we get by not having a very accurate estimate of the state parameters against the loss of optimality we get by having the controller relinquish control to the identification procedure from time to time.

5. PROBLEM-SOLVING POWER OF DEFINITIONS

Within a theoretical system a definition defines a concept (Hempel, 1952). Concepts, on the other hand, make classifications possible, and classifications have the astonishing power of dividing and sub-dividing.

Through definitions concepts can be engendered which classify a state space into regions, and these into further sub-regions. Opportune definitions may reduce a problem to some combination of sub-problems. The "enemy" becomes divided. If (as in the next section) we choose to regard problem

solving as a game like process, one move of the game playing opponent posing the problem is a propitious set of definitions (subdividing all possible problems) reducing the maximum variety of the other player's moves. Whether or not a particular definition will be rewarding is a heuristic question. Sets of definitions may be sometimes qualified in terms of the desirable or permissible levels or redundancy; in terms of other features of sub-problem classification, in terms of descriptive economy, or the facility with which the defined concepts can be manipulated.

For the present purpose the most important definitions are those which determine computable concepts. For these, a rule can be furnished for deciding whether or not an object belongs to the concept-class (the rule may arise from the nature of the definition itself).

Interest is primarily focussed upon problem-resources-oriented and theory-based systems of definition, leading to classifications that provide efficient problem reductions. In such cases, problem classification procedures may be regarded as a "look-ahead" instrument for directing problem-solving strategies. And the instrument is especially useful when the (hierarchical) classification can be closely (or exactly) mirrored by a hierarchy of problem reduction procedures. In these circumstances, problem perception by means of classifying criteria is intimately linked, AU FUR ET A MESURE, with problem reduction processing. That is, as problem classification is going on, certain corresponding problem reductions are already being executed. Perception is concomitant with action.

Definitions can be envisaged as active perception devices calling into attention the pertinent aspects of problem-solving situations for which further action is to be taken.

6. GAME PLAYING AGAINST A THEORY

Choice in a search strategy corresponds to moves in a game, the rules of which are theory determined. Consequences of a choice are the theory's inevitable reactions according to such rules. They are the opponent's moves. Thus, in this game, one plays against one's lack of information regarding the inescapable consequences of particular choices. Evaluation is the weighing of possible alternatives within the restriction of moves imposed by the theory.

It is thereby profitable to so structure the developed theory, on the occasion of its contrivance, in such a way that the number of possible alternatives is reduced, but without losing generality.

Such reduction has the counterpart meaning that knowledge gained about the structure of the problem space has been augmented.

In any case, it is a game "against nature" rather than a competitive game. The theory's moves are neutral, not malicious.

In this light, of problem-solving as an enterprise within a determinate theory where there are unknown outcomes to one's problem-solving moves, operator applicability acquires both A PRIORI and A POSTERIORI conditions. The former reflect what is already certain about the structure of the problem space. The latter inform us about that of which we are uncertain. Again, the first, a priori conditions, remind us of which moves are conceivable. The others, a posteriori conditions, show us which of them have a viable outcome.

Problem-solving search procedures become then strategies for playing a game to be won by finding solutions.

7. A NOTE ON MECHANICAL THEOREM PROVING

Mechanical theorem proving (Chang, Lee, 1973) strikes us as a paradigmatic example of current artificial intelligence work where most of the foregoing is symptomatic.

If we do not explore any further such conspicuity, the reason is that the specialization of the subject would oblige us to a rather lengthy exposition. But, above all, it is because the subject matter of this thesis aims at providing a detailed account of a more typical case study, in the sense that it was developed almost from scratch, without previous theorizations available.

8. REFERENCES

- Amarel, S. (1970) - On the representation of problems and goal-directed procedures for computers. pp. 179-244.
in "Theoretical Approaches to Non-numerical Problem Solving" (Banerji, R.; Mesarovic, M.D. (eds.)). Springer-Verlag.
- Arbib, M. (1972) - The metaphorical brain. pp. 80-82. J. Wiley.
- Chang, C-L.; Lee, R.C-T. (1973) Symbolic logic and mechanical theorem proving. Academic Press.
- Hempel, C.G. (1952) - Fundamentals of concept formation in empirical science. pp 1-20 The University of Chicago Press.
- Tichomirov, O.K.; Poznyanskaya, E.D. (1966) An investigation of visual search as a means of analysing heuristics.
in Soviet Psychology, vol. 5. pp.2-15.

CHAPTER 2

FROM AN ARCHITECT'S POINT OF VIEW

1. INTRODUCTION

An architect's work is beset by irregular and context specific conditions which may become obstrusive as a project develops. The overall effect at the level of problem solving and planning can be summed up by fine characteristics as follows.

First, there exists a very great complexity of decision levels, from the more strictly functional goals up to the most recondite symbolic expression.

Second, the non-existence or contemporary lack of knowledge of architectural laws, explaining and governing spatial organization of man-made "artificial" phenomena (like in classical tradition).

Third, the impossibility of systematically specifying a plan for a project before embarking on a tentative search for solutions.

Fourth, since the problems dealt with are essentially ill-defined, the non-existence of optimal solutions gives rise to the need for establishing "merely satisfactory" solutions (1).

Fifth (and last), the heuristic nature itself of the process for searching solutions, does not allow such a process to have a pre-determined trajectory but, instead, it must re-orient and feed itself upon a succession of tentative experiments.

Although these characteristics of architecture appear, at first sight, to thwart any attempt to apply new and computer oriented design techniques, they need not do so. It is only essential that candidate methods are GENUINE man/machine (or even wholly mechanized) design tools. The requirement to crea-

te such things is a spur to innovation in this area; and, it may be noted, the requirement is real enough (2). Traditional design methods are patently inadequate for dealing with problems arising from the linkage of several factors (3,9); for example, the following:

- a) Increasing acceptance of rational and socially relevant norms (for instance, floor space, access, and service requirements) as well as cost constraints giving rise to explicit comparison of alternatives and schemes of gradual decision, by means of step by step optimization methods and the application of means/ends analysis.
- b) Studies of architectural artifacts (houses, public buildings, etc.) whose form and organization is designed with respect to the behavioural patterns of their inhabitants.
- c) The possibility, in buildings design and urban planning at least, that the solutions may be adapted (whilst building or use is in progress) to information received during the process.

Of the factors, c) is perhaps the most likely to introduce dramatic modifications in architectural design, because any one building (at any rate with existing and foreseeable construction techniques) is planned, built and not altered. Hence, to realize a measure of adaptation, the architect is bound to employ predictive modelling: of the unstructured kind or through the mediacy of an appropriate system. The advantages of the computer approach have been analysed, for example by (4), and appear to be pragmatically justifiable on several grounds, e.g. communication between planners, greater possibility of interaction between the architect and the client by way of rapidly generat

ed graphical outputs and the like.

Henceforward we shall refer to this gamut of procedure modelling and adaptation methods, used "online", as MONITORING.

2.- METHODS

It is useful, for exposition at any rate, to distinguish between methods tailored to describe better formulations of problems to be solved (and the background of knowledge in which solutions are possible) and methods intended to find exact or optimal compromise solutions for well defined (sub) problems.

The difference between these points of view is identical with the distinction considered in chapter one, when PROBLEM REPRESENTATION was under discussion. From the perspective of computer implementation we require a capability for enunciating a problem within the framework of a PROBLEM REPRESENTATION and the ability of mustering procedures able to solve it.

If this distinction is agreed, any problem is automatically reduced to two subproblems. First, the problem of contriving a suitable problem representation. Second, the elaboration of efficient problem-solving strategies for the representation devised. Correspondingly, the architect, as a user of a program's problem-solving resources, is confronted with two problems. In the first place, he must be able to secure the formulation of HIS problems within the theoretical framework which serves as a structured problem-solving environment appropriate to the program's problem-solving capabilities. Secondly, he must be aware of the MEANING to be assigned to the program's formal solutions, or to its "no solution" indications, in terms of HIS interpretation of the theoretical objects, concepts and properties involved. Moreover, realistic interaction with the program and its proposals, depends upon a sha-

red (user/machine) usage of the program's (syntactical) capabilities and constraints and the architect's (semantical) desires. This "understanding" hinges, in turn, upon the versatility of the available interface and its ability to cope with the attempted transactions.

3. GENERAL COMMENT ON RESULTS

The main results considered in this thesis refer to a very limited subset of architectural problems. The methods developed help the architect, in a user/machine system, to handle topological arrangements, on the plane, of a set of conventional rectangular spaces with given dimensional limits, which obey a relationship of adjacency and respect given restraints upon the resulting contour.

Although dimensional constraints and control over the overall contour have been introduced (and for example T-shaped and L-shaped spaces can be envisaged in terms of a proper combination of rectangles) it is true that acute difficulties still persist at the level of PROBLEM FORMULATION, and in the previous specification of the design steps required to tackle that problem formulation.

In a user/machine interaction such as the one developed, which is still quite rigid, an appearedly better "architectural" output could be achieved as a result of very real information loss. The price to be paid for any facile realization would be a premature restriction of the problem and blind elimination of hypotheses which could show promise, once some adjustments or compromises were made, by dint of external heuristic processes, present in the designer himself. Such processes, as a means to evaluate and decide, are characteristically difficult to "translate" into a pro-

gram and their description hinges upon further knowledge about problem-solving, not only in a general sense but in particular design situations as well.

An important point can be made for the generative algorithms employed, in that not only "positive" information is output in the form of feasible layouts. "Negative" information is also provided to indicate that a proposed layout has incompatible components which are duly localised. Appropriate reformulation of such layouts can be a stimulus to the user which leads him to rethink his design requirements "online". Interaction with the computer makes this process less burdensome and more effective. Interaction also has a training effect, as a result of which the user becomes more familiar with his own mental processes, the problem solving operations of the program, and the possibilities of user/machine mutualism.

One major debility of present methods lies in the reductionism they impose on the formulation of problem statements, be that on a semiotic level or inclusively from a functionalist point of view.

In the methods we have developed, reductionism manifests itself insofar as the type of results obtained tends to be determined only by those semantical or practical contents which can be conveyed a priori in terms of adjacencies on the plane, etc.. Thus, strictly topological concepts like those of "compatibility" (within a space) and "connection" (contiguity) between elementary units of activity/space have to be explored, in order to show how the designer can express culturally significant contents by means of adjacencies. (This has been done to some extent in (5)).

Naturally, an input consisting almost only of adjacencies, implies a deep prior examination of the problem's requisites, especially in the sense that an all-or-nothing dependency between each pair of spatial elements is already a form of architectural output which carries an implicitly chosen (and unchosen)

decision about the pattern of spatial relationships. However, the formal conception of an architectural system can be distorted by giving priority to obtaining a set of optimum relationships among its internal component spaces. (It may not be optimum to optimize anything at all.)

The problem should be solved instead by reconciling a RELATIVELY satisfactory subset of geometric adjacency relationships with other principles of form corresponding to additional INPUT solicitations.

Let us for the moment assume that in architectural design there is a dialectic "composition" of two main processes, sometimes overlapping sometimes sequential, along the actual design process. The first, call it deductive, permits, in our case, to generate candidate solutions and test them against an explicitly defined grammar of spatial well-formedness according to very definite rules expressing what is FEASIBLE, at the articulation level of elementary or LOCAL requisites. The second one, call it inductive, tries to enact generic principles of DESIRABLE forms, underlying architectural typology, which can be input, in the programs, as GLOBAL properties of the layouts. These global properties, to be considered by a program in the form of INPUTS, must also be amenable to some grammar, as is the case for example with the permitted types of contour form that we already consider.

The inductive steps, i.e. the building up of grammars regulating global properties, are particularly important for ensuring the correct deployment of the layouts being generated, thus embodying in them formal structures which could not arise by simple deduction on the logic of elementary interrelations.

Local and global requirements are however diverse, and timing interaction is inevitably ambiguous. For one thing, a hierarchical relation of impor

tance among local and global input requirements is hard to establish, inasmuch it cannot be transposed into a pre-determined sequential ordering of design-process decisions: for another, as we have remarked, the architectural problem-formulation itself is neither independent nor necessarily previous to the automatic generation of hypotheses concerning the structure of would-be solutions. Especially because it is only by means of such proposed solutions that the local/global interaction becomes more fully deployed.

In this work, one aim was to establish an interplay between the designer and the program and vice-versa, as well as reconciling local requirements (input adjacencies, dimensions, and others) with the global topological inputs, i. e. pre-established principles of form embodying more complex and generic systems of requisites such as the contour form. On implementing these interactions, an attempt was made not to determine beforehand the precedence of one type of input over another, and also to explore the possibility of reconciling them in a cost effective manner (quantified in terms of a measure of disagreement between dimensional requisites and the actual values obtained). A step by step evaluation of alternative dimensional inputs allows this expedient, through appropriate user/machine interaction aimed at the re-orientation or reformulation of the initial and subsequent problem statements themselves.

4. NOTES ON PREVIOUS WORK

Until recently a widespread method in computer-aided architectural design had been the HIDECS (Hierarchical DEComposition of Systems) approach developed by Christopher Alexander (6). In this method, a design problem is subdivided into smaller problems through the identification of which subsets of the design requirements interact highly among themselves and minimally with the requirements of the other subsets. Then, in the synthesis phase of

the design, each subset of requirements having been "solved", is supposed to lead to some subcomponent that is reconciled into the complete design. (The same aim led to the development of the AIDA technique (7, 8)).

But just because a design problem may be decomposed according to certain subsets of requirements, it does not mean that it is effective to decompose a design program in the same (or in an isomorphic) way. Indeed, as it was argued in chapter one, a close relationship should exist between the problem representation and the problem solving resources which are viable within it.

A complementary point of view on representations, particularly applicable to the design process and expressed by Alexander (9), considers that, to be useful, a diagram representation should simultaneously be a requirement diagram and a form diagram (at least isomorphically so). That is to say, the design requirements should be embodied in the design diagram in terms of the form components (or their equivalent) which convey the design solutions that satisfy them.

Furthermore, if each design requirement maps directly, independently, and uniquely into a single form component (or a proper composition of them) of the design diagram, then the synthesis task amounts to carrying out the indicated mapping (or some derived mapping) of requirements to form. The designer is, of course, at liberty to experiment with various sets of requirements, by adding, deleting or composing independent form components. Some design requirements, however, are not independent of one another, and so cannot lead to independent form components. Thus, interacting requirements should be present in the design diagram, but those which should not interact should not be inadvertently represented as interacting.

Most programs used for computer implemented design of floor plan

layout do not use problem representations with the aforesaid characteristics. A common representation in the last ten years has been a square grid of modular areas, of usually small size. On this grid an activity space is located by assigning to it a number of contiguous squares, but the pattern of squares assigned to a given space need not itself be rectangular in shape.

Two distinct space allocation strategies are usually employed with this representation. The first, used for example in (10),(11)and(12), requires an initial assignment of squares to spaces, to be made by the program or the user in some arbitrary way. Next, the assignments of squares to spaces are methodically interchanged in some manner, and a space allocation optimization function is evaluated for each new arrangement. The arrangement which achieves the highest value of the optimization function until no further improvement is found, after some search effort, is the one adopted. This is the familiar "hill climbing" approach.

The second approach used in (13), (14) and(15), among others, starts by assigning a first activity space (usually the one which has strongest proximity requirements with the others) to some squares in the center of a grid. It then proceeds by assigning the other activity spaces to squares around the periphery of the first space, according to decision rules that ponder the relative strengths of the locational relationship extant among the activity spaces.

Both of these approaches deal with design requirements which are stated in terms of an area specification for each activity space, and one kind or another of relational matrix that expresses weights given to the degree of proximity desired for each pair of spaces. The area requirements are easily satisfied by assigning a number of squares on the grid to an activity space, and the relational matrix lends itself well to the evaluation of an optimiza-

tion function based on the communication costs within a given spatial arrangement.

A second type of representation considers each space to be located to have a rectangular form, but does not limit their location to any grid. Instead, the rectangles are drawn freely on the plane, sometimes partially overlapping one another, and usually conveying the impression of an approximate layout. Such representation is used in (14).

These representations do not have the desired characteristics described earlier because they deal "too directly" with planar representations of spaces having a definite size and shape and a definite location with respect to one another. Thus, if one wishes to satisfy the requirement "space A is adjacent to space B", one cannot indicate that fact unless one also assigns some arbitrary size and shape to each space, and places the two spaces next to one another so that they contact along some particular "wall" segment. But the sizes of the spaces and the "wall" segment along which they are adjacent, may not have been specified as design requirements, and determining them arbitrarily so that the two spaces can be drawn, may introduce unnecessary conflicts later in the design process. Similarly, it would be impossible to satisfy a size requirement on the design diagram without arbitrarily placing the space being sized somewhere on it. Thus, although these planar layout representations may allow a direct mapping of some of the requirements for a space on to form, the mapping is neither unique nor independent of the other design requirements for the same space.

The linear graph approach. The use of a linear graph as a design representation can overcome the major difficulties arising from layout representations which are "too literal". In this other representation, the spaces are

pictured as labelled nodes of a graph, possessing individual attributes such as intended use, area, perimeter, shape, and orientation. Adjacencies or proximity between spaces can then be indicated by edges connecting these nodes.

Thus, a space does not have to be physically drawn, since the indication that an adjacency requirements is satisfied or not is given by inspection of the edges connecting the nodes. A rudimentary application of this technique as a pencil and paper design method was described Levin in (16). Casella and Rittel (17), in an unpublished paper, also discussed the use of a linear graph representation for planar layouts.

The work of Krejcirik (18) and especially the work of Grason (19, 20, 21) should also be mentioned, inasmuch many of the aspects dealt with in their work were treated independently by the present author, although in quite different ways. This is not surprising since the parts in this work that are similar to their's had already been established and published (22, 23) before knowledge was gained of their own publications. A posterior publication is (24).

A more extensive reference to these two works can be found in (25).

5. REFERENCES

- (1) Simon, H. (1969) - "The sciences of the artificial", M. I. T. Press.
- (2) Negroponte, N. (1970) - "The architecture machine", M. I. T. Press.
- (3) Alexander, C.; Poyner (1967) - "The atoms of environment structure", Ministry of Public Buildings and Works, London.
- (4) Carter, J. (1973) - "Computers and the architect", in "Architect's Journal", October 3, 10, 24 and 31 numbers.

- (5) Portas, N. (1965) - "Inter-relações de funções no fogo", Laboratório Nacional de Engenharia Civil, Lisbon.
- (6) Alexander, C. (1966) - "HIDECS: Four computer programs for the hierarchical decomposition of systems which have an associated linear graph", Research Report R63-27, Dept. of Civil Engineering, Civil Engineering Systems Laboratory, M.I.T., Cambridge, Massachusetts.
- (7) Jessop, W.; Friend, J. (1969) - "Local government and strategic choice", Tavistock Publ., London.
- (8) Luckman (1969) - "An approach to the management of design", in Broadbent and Ward (eds.) "Design Methods in Architecture", Lund Humphries Publs., London.
- (9) Alexander, C. (1964) - "Notes on the synthesis of form", Harvard University Press, Cambridge, Massachusetts.
- (10) Armour, G.; Buffa, E. (1963) - "A heuristic algorithm and simulation approach to relative location of facilities", in "Management Science", vol. 9, no. 1, pp. 294 - 309, January.
- (11) Buffa, E.; Armour, G.; Vollman, T. (1964) - "Allocating facilities with CRAFT", "Harvard Business Review", March-April numbers.
- (12) Parsons, D. (1967) - "Planning by the numbers in performance design", in "Progressive Architecture", pp. 111, August.
- (13) Lee, R.C.; Moore, J. M. (1967) - "CORELAP - Computerized relationship layout planning", in "Industrial Engineering", vol. 18, no. 3, March.
- (14) Lee, K. - "COMSBUL - Computerized multi-storey building layout". For information on this program contact Kaiman Lee, 1 Rosa St., Hyde Park, Massachusetts 02136.
- (15) Whitehead, B.; Eldars, M.Z. (1964) - "An approach to the optimum layout of single-storey buildings", in "Architect's Journal", June 17.
- (16) Levin, P. H. (1964) - "Use of graphs to decide the optimum layout of buildings", in "Architect's Journal", October 7.
- (17) Casalaina, V.; Rittel, H. (1967) - "Morphologies of floor plans", "Conference on computer-aided building design".

- (18) Krejcirik, M. (1969) - "Computer-aided plant layout", in "Computer-Aided Design", pp. 7-19, Autumn.
- (19) Grason, J. (1970 a) - "A dual linear graph representation for space-filling location problems of the floor plan type", in Gary T. Moore (ed.) "Emerging Methods in Environmental Design and Planning", M. I. T. Press, pp. 170-178.
- (20) Grason, J. (1970 b) - "Fundamental description of a floor plan design program", in Sanoff, Henry, Cohn, and Sydney (eds.) "EDRA 1, Proceedings of the 1st Annual Environmental Design Research Association Conference", North Carolina State University, pp. 175-182.
- (21) Grason, J. (1970 c) - "Methods for the computer-implemented solution of a class of 'floor plan' design problems", Ph. D. thesis, Carnegie - Mellon University, Pittsburgh, Pa.
- (22) Pereira, L. M. et al. (1972) - "Agrupamentos de espaços a partir de grafos de adjacências", Laboratório Nacional de Engenharia Civil, Lisbon.
- (23) Pereira, L. M. et al. (1972) - "Geração de alternativas de agrupamentos no plano de espaços rectangulares com dimensões", Proceedings "II Simpósio sobre as Teorias da Informação e dos Sistemas", Instituto de Alta Cultura, Ministry of Education, Lisbon.
- (24) Pereira, L.M. et al. (1973) - "Interactive dimensional layout schemes from adjacency graphs", in Proceedings of the Design Activity International Conference, London.
- (25) Los, M. (1973) - "Spatial design and artificial intelligence", Dept. of Planning Sciences, University of Pennsylvania.

CHAPTER 3

GRAPH THEORETICAL NOTIONS

1. INTRODUCTION

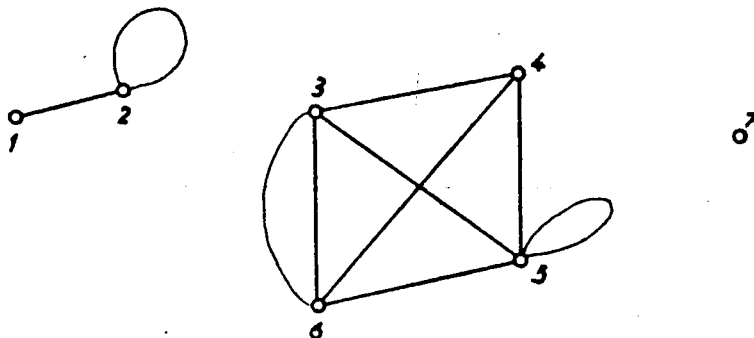
In this chapter the reader will find ^{most of} all the graph theoretical notions needed in the course of exposition in later chapters. These notions have been condensed in a single chapter for easier reference.

If some of the notions here expounded are standard, there are others whose introduction and development were motivated in the course of work by the necessity to deal with particular subproblems. They are, thus, of the present author's sole responsibility. Attention will be drawn to this particularity in the chapters where such notions are first exploited. Furthermore, all motivation for the notions in this chapter is relegated to the appropriate chapters of this work where the need for them is first encountered.

2. INFORMAL PRESENTATION OF GRAPHS

In this section we will consider that all sets referred to are finite.

Under certain conditions, a graph can be envisaged as a set of points on the plane, some of which, eventually none, are linked by lines on the plane also called edges. The following figure exemplifies one such graph.

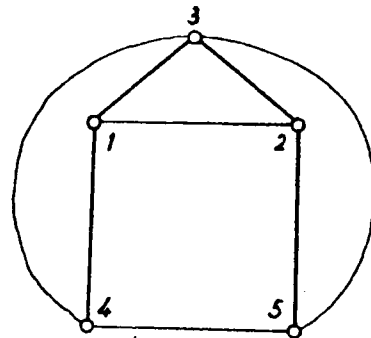
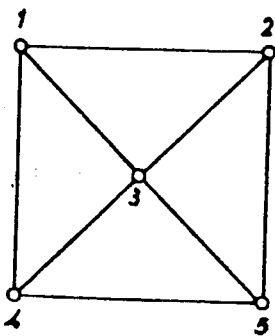


The points on this graph are represented by small circles, so that they may not be misinterpreted as edges eventually crossing one another or themselves. They are points 1, 2, 3, 4, 5, 6, and 7. The edges are consequently the lines linking these points, and may be represented as unordered pairs of points. Thus, (3, 5) means the edge linking point 3 with point 5. This edge may also be denoted by the pair (5, 3). (2, 2) is the edge linking point 2 to itself, etc.. Notice that point 7 is not linked to any point whatsoever.

This notation for the edges is ambiguous. It does not distinguish for example the two different edges linking points 3 and 6. The ambiguity is abolished however since we impose on the class of graphs we are interested in, the following two restrictions:

- 1) Given any two points there is never more than one edge linking them.
- 2) No edge links a point to itself.

Let us now consider the two next figures:



It can rightly be asked if these two figures are the same graph. The question becomes the more pertinent since we have not formally defined what we mean by a graph.

It can easily be seen that the points and edges of both figures can be put into correspondence by means of the numbers assigned to the points of each figure that are the same in both.

We shall not say that the two figures are the same graph but that they are two different realizations (or representations) of the same graph.

What is important for the definition of graph are the points which belong to it, and the edges joining some of them, independently of any particular form of representing or realizing them on the plane.

Because in this work we shall have to deal with different realizations of the same graph, it becomes advisable to make a clear cut distinction between "graph" and "realization of a graph".

3. FORMAL DEFINITIONS

Graph. By a graph G we mean a set $E(G)$ of edges, a set $V(G)$ of nodes (or vertices), and an incidence relation which assigns to each edge a (non-ordered) pair of nodes, distinct or not, which are its extremities. An edge is an arc if its extremities are different, and a loop if they are the same.

Subgraph. A subgraph of G is a graph H such that $E(H) \subseteq E(G)$, $V(H) \subseteq V(G)$, and each edge in $E(H)$ has the same extremities in H and in G : a subgraph H with $E(H) = \emptyset$ and $V(H) = \emptyset$ is denoted by \emptyset ; H is proper if $H \neq G$ and $H \neq \emptyset$. The union and intersection of subgraphs of a given graph are defined in the usual way. Given two subgraphs, H and K , of G , the intersection I of H and K , defined by $V(I) = V(H) \cap V(K)$ and $E(I) = E(H) \cap E(K)$, can easily be shown to be a subgraph of G . The union of H and K is also a subgraph of G , and can be similarly defined.

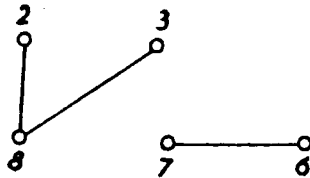
Any subset W of $V(G)$ determines a subgraph $G(W)$ of G , by defining $V(G(W)) = W$, and $E(G(W))$ as the set of all edges of G with both extremities in W . Any subset F of $E(G)$ determines a subgraph $G.F$ of G , by defining $V(G.F)$ as the set of the extremities of the edges of F , and $E(G.F) = F$.

Finite and Simple Graphs. A graph G is finite when $E(G)$ and $V(G)$ are both finite: it is simple if it contains no loops, and for each pair of nodes there exists at most one edge having them both as extremities. If a is an edge whose extremities are x and y , we shall indifferently represent it by (x, y) or (y, x) , and shall write $a = (x, y) = (y, x)$.

N.B. in what follows, "graph" will always be taken to mean "finite simple graph."

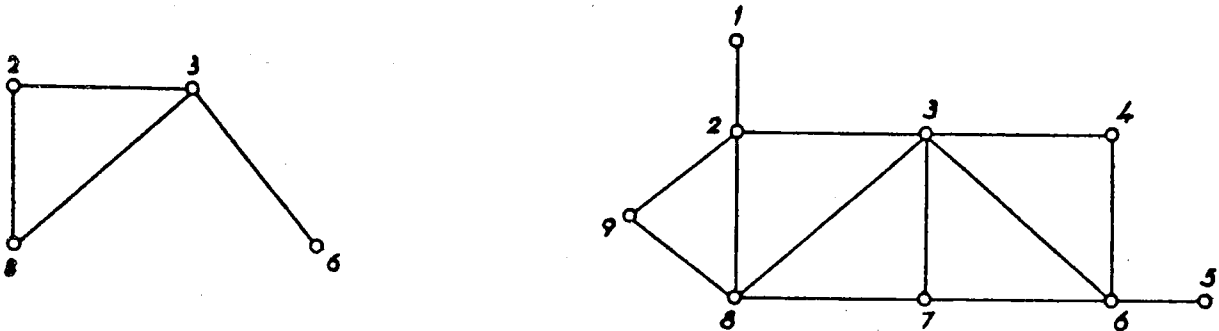
An edge (a, b) is an edge of node p if $p = a$ or $p = b$. By the edges of node p we mean the set of all edges of p .

Consider the two following figures:



The one on the right represents a subgraph of the graph represented on the left: in fact, it is the subgraph determined by the set of nodes of that graph which possess at least four edges.

The next figure shows a proper subgraph of the same graph:



Path. A path C in a graph G is a finite sequence of terms which are nodes and edges of G ,

$$C = (v_0, e_1, v_1, v_2, \dots, e_k, v_k),$$

with at least one term v_0 , and where

1) the terms of C are alternatively nodes v_i and edges e_j of G , and C starts and ends with nodes, its extremities.

2) v_{i-1} and v_i are the extremities of e_i in G , for $0 < i \leq k$.

A path is degenerate if it has only one term v_0 ; it is simple if all its terms are distinct; it is circular if it is not degenerate and all its

terms are distinct except for $v_0 = v_k$.

Polygon, Segment, Cycle. The edges and nodes of a path C define a subgraph $G(C)$ of G ; if C is circular, $G(C)$ is called a polygon; if C is simple and non-degenerate, $G(C)$ is called a segment, and v_0 and v_k are its extremities.

A cycle of G is a subset K of $E(G)$ such that the number of edges of K of each node in G is even. If C is a polygon then $E(C)$ is a cycle; if a cycle results from a polygon it is said elementary.

Intuitively, a cycle is a closed path which eventually passes more than once through the same node or edge. An elementary cycle is then a cycle which passes only once through any of its nodes or edges, except for its first (or last) node.

Connectedness. Separateness. Let G be a graph. Consider the (binary) relation in $V(G)$ determined by the following predicate: "there exists in G a path with x and y as its extremities". It is easily proved that it is an equivalence relation, and that it consequently induces in $V(G)$ a partition $P = (W_1, \dots, W_n)$; the subgraphs $G(W_1), \dots, G(W_n)$ have no nodes in common, and it is clear that, if $i \neq j$, no edge in G has one extremity in $G(W_i)$ and the other in $G(W_j)$. If there exists only one block W_1 in P , then G is said to be connected and $G(W_1)$ is the only connected component; otherwise, each $G(W_i)$ is a connected component of G . The number of connected components of a graph is called its connectivity.

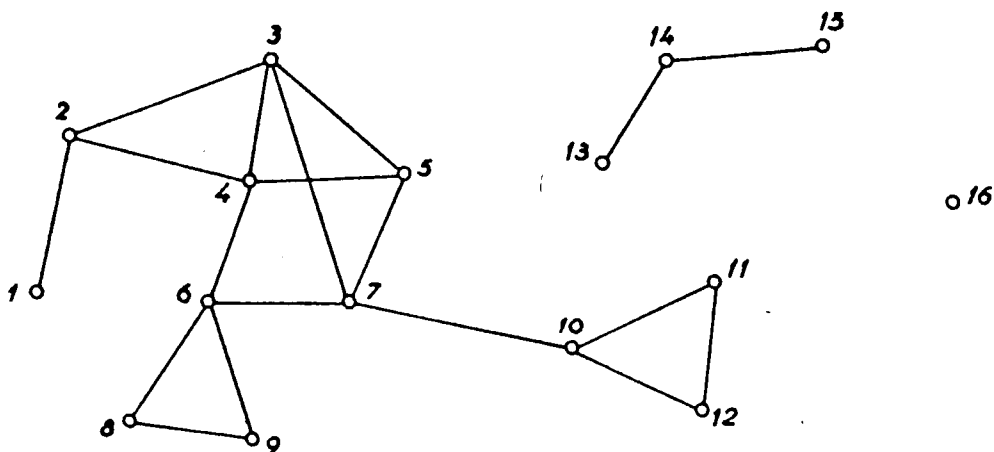
Let us now consider the following predicate, which determines an equivalence relation in $E(G)$: "there exists in G an elementary cycle containing edges a and b". The resulting partition $Q = (F_1, \dots, F_m)$ determines the subgraphs $G.F_1, \dots, G.F_m$ of G , which have no edges in common.

If the partition Q has only one block $G.F_1$, G is said non-separable or 2 - connected; otherwise, G is separable and each $G.F_i$ is a block of G .

Addition of Cycles. We will now define a binary operation on cycles, called addition, and represented by \oplus . Given two cycles C_1 and C_2 , their sum $C_1 \oplus C_2$, which is the result obtained by their addition, is a cycle C whose edges are those that occur once, and only once, in the union of C_1 and C_2 ; is indeed a cycle: for any node \underline{v} of $V(G)$, all its edges a_1, \dots, a_n which are terms of C_1 , and all its edges b_1, \dots, b_m which are terms of C_2 , are either different, in which case the number of edges of \underline{v} that are terms of $C_1 \oplus C_2$ is even, since \underline{m} and \underline{n} are even, or, there are pairs (i, j) with $a_i = b_j$, and in that case the number of edges of \underline{v} of $C_1 \oplus C_2$ is $m + n - 2k$, where \underline{k} is number of pairs (i, j) with $a_i = b_j$, and in that case the number of edges of \underline{v} in $C_1 \oplus C_2$ is $m + n - 2k$, where \underline{k} is the number of pairs (i, j) satisfying the above mentioned condition.

The operation \oplus is clearly associative and commutative: thus, it is legitimate to speak of the sum $C_1 \oplus \dots \oplus C_n$ of the list of cycles C_1, \dots, C_n . Note that in such a list the same cycle may figure more than once.

Example. Let us informally consider the following diagram,



as expressing a graph whose edges are,

(1, 2)	(5, 7)	(11, 12)
(2, 3)	(6, 7)	(13, 14)
(2, 4)	(6, 8)	(14, 15)
(3, 4)	(6, 9)	
(3, 5)	(7, 10)	
(3, 7)	(8, 9)	
(4, 5)	(10, 11)	
(4, 6)	(10, 12)	

and whose nodes are numbered from 1 to 16. $E(G)$ has then 19 edges and $V(G)$ 16 nodes. An example of a subgraph of this graph is given by the following set of edges

(3, 7)	(10, 12)
(7, 10)	(11, 12)
(10, 11)	

and a set of nodes comprised by nodes 3, 7, 10, 11 and 12.

An example of an edge of a node, node 7 say, is the edge (5, 7). The edges of node 7 are:

(3, 7), (5, 7), (6, 7), and (7, 10).

The following is a path in G :

(2, 4), 4, (4, 3), 3, (3, 7), 7, (7, 10), 10, (10, 12), 12, (12, 11), 11, (11, 10), 10).

Now, since G is simple and has more than one node, any non-degenerate path in G , as the one above, can more concisely be expressed by an ordered sequence of the extremities of the edges of the path. In this nota-

tion the above path is described as:

(2, 4, 3, 7, 10, 12, 11, 10).

The same can be said regarding cycles : in this graph the cycle ((3, 5), (5, 7), (7, 6), (6, 4), (4, 5), (5, 3), (3, 4), (4, 2), (2, 3)) is expressed less redundantly by the sequence of nodes

(3, 5, 7, 6, 4, 5, 3, 4, 2, 3).

In this notation, an elementary cycle of G is, for example,

(6, 4, 2, 3, 5, 7, 6).

In G, the set made up of edges

(3, 5), (5, 7), (7, 6), (6, 4), and (4, 3),

and the set made up of nodes 3, 5, 7, 6, 4, and 3 define a polygon P of G, since $P = G(C)$ where C is the circular path expressed concisely as

(3, 5, 7, 6, 4, 3).

Graph G is not connected. There is no path, for instance, between nodes 4 and 13.

The subgraph of G comprised of the edges of nodes 1 to 12 and these nodes is, however, connected. Connected are also the subgraph determined by nodes 13, 14, 15 and its edges, and the subgraph H with $V(H)$ as the set containing node 16 only, and with $E(H) = \emptyset$. Note that H is a connected component of G.

The union of these three subgraphs of G is G. Since they are also its connected components, the connectivity of G is three.

An example of a non-connected subgraph of G is given by the edges

(1, 2)

(4, 6)

(10, 11)

(2, 4)

(4, 5)

and the nodes 1, 2, 4, 5, 6, 10, and 11.

Nets. The notion of "connected component", or more simply, "component", can be usefully generalized.

According to previous definitions, the connected components of a graph G result from the partition induced in $V(G)$ by the predicate "there exists in $V(G)$ a path with x and y as its extremities". The generalization of the notion of component is established by imposing that the nodes on the path satisfy some given predicate P , such that the resulting partition defines in $V(G)$ an equivalence relation.

Given such a P , a component relative to P is a connected component whose nodes satisfy predicate P . P itself may be any logical combination of predicates.

Let P be one such predicate, with $V(G)$ as its domain. Let W_i be the block of a component relative to P .

Consider the set S_i of all edges of G having at least one extremity in W_i . Each S_i determines a subgraph $G.S_i$ of G , with $V(G.S_i)$ as the set of all extremities of the edges of S_i , and with $E(G.S_i) = S_i$. We shall call any such $G.S_i$ a net relative to P , or simply a net, whenever P is made clear from context.

Examples. Let P be defined as follows:

" P is true of node x if and only if x has four edges".

The components relative to P of the graph depicted in the last figure are only one: namely the one whose block is the set comprised of nodes 3, 4, 6, and 7.

Its edges are:

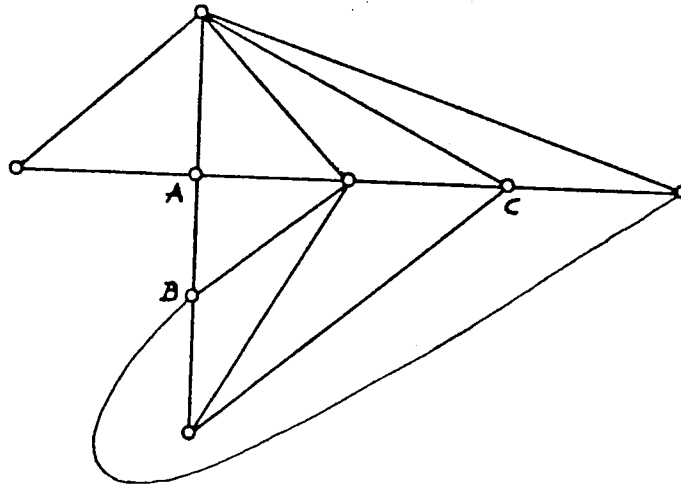
(3, 4), (4, 6), (6, 7), and (3, 7).

The only net relative to P has edges

(3, 2)	(4, 2)	(6, 8)
(3, 4)	(4, 5)	(6, 9)
(3, 5)	(4, 6)	(7, 5)
(3, 7)	(6, 7)	(7, 10),

and as nodes the set of extremities of the above edges.

In the graph depicted in the next figure, there exist two nets relative to P. One is determined by the block having nodes A and B as its elements; the other by the block having C as its only element. Their edges and nodes can be identified easily.



Note on Notation. If H is a subgraph of G, we write simply $H \subset G$.

If \underline{a} is an edge and \underline{v} is a node of G, we denote these facts by $a \in G$ and $v \in G$.

This slight abuse of language will not raise any problems whatsoever, but will contribute to easiness of communication.

Removing a node from a graph. Let G be a graph, and let $v \in G$.
the subgraph obtained from G by removing v we mean $H \subset G$ defined
 $H = G (V(G) - \{v\})$.

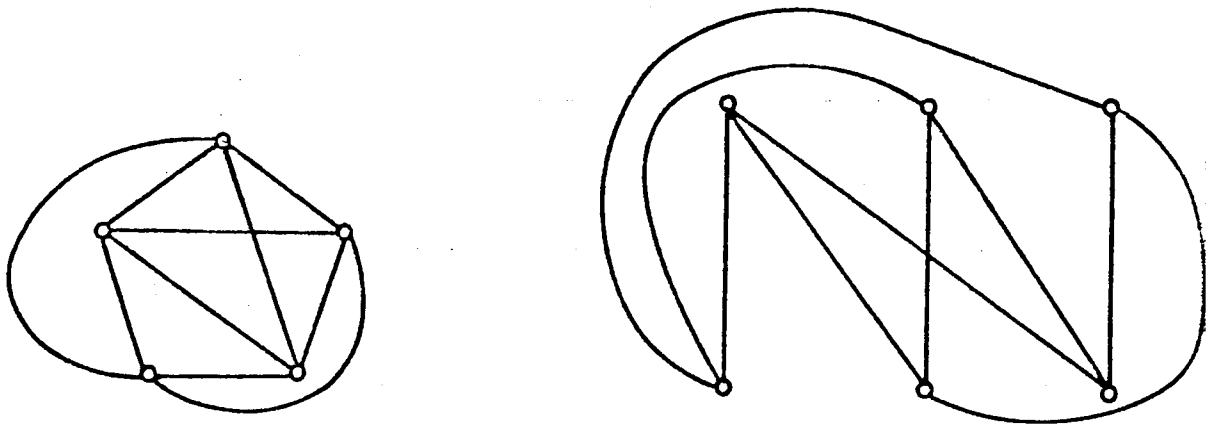
Realization. Planar Realization. Planar Graph. Often a graph is represented by a diagram on the plane as the ones we have been using. We are now going to give a precise definition of such "representations" so that we may define "planar graph".

A realization of a graph G is an assignment r defining a correspondence between each node $v \in G$ and a different point $r(v)$ on the plane, to each image, and to each edge $a \in G$ a different line $r(a)$ on the plane, which does not intersect itself, such that

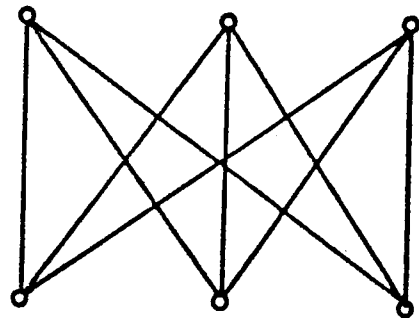
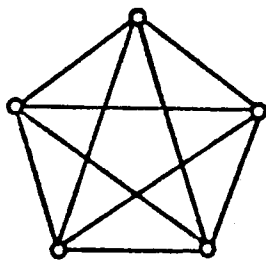
- 1) if $a = (x, y)$, $r(a)$ has endpoints $r(x)$ and $r(y)$;
- 2) for each edge a , $r(a)$ does not pass through any image of a node of G , besides the images of the extremities of a .

N.B. We will not always make a distinction between nodes and their images and edges and their lines. However where it may cause confusion, we will maintain the distinction.

It can be readily recognized that any graph has a realization. However, it is demanded that the realization be planar, i.e. that all lines should not meet except at the images of common extremities, then there are graphs which do not have any planar realization whatsoever: a planar graph is one having a planar realization. For example, the graphs realized by the following figures are not planar.



Other realizations of these two graphs are shown next.



Let us call Kuratowski graph to any graph obtained from these by substituting arbitrary segments for any edge.

The well-known Kuratowski theorem for planar graphs can then be phrased this way:

Theorem. A graph is not planar if and only if it has a Kuratowski graph as a subgraph.

However, this theorem is not in general propitious for detecting graph planarity. We shall rely on another way of characterizing planar graphs which lends itself better to algorithmic detection of graph planarity. It is due to Mac Lane (1, 2) and makes use of the concept of "planar mesh".

Planar Mesh. A planar mesh of a graph G is a list $R=(C_1, \dots, C_m)$ of elementary cycles of G , not necessarily distinct, such that:

1) if an edge belongs to a cycle of R then it belongs to exactly two cycles of R .

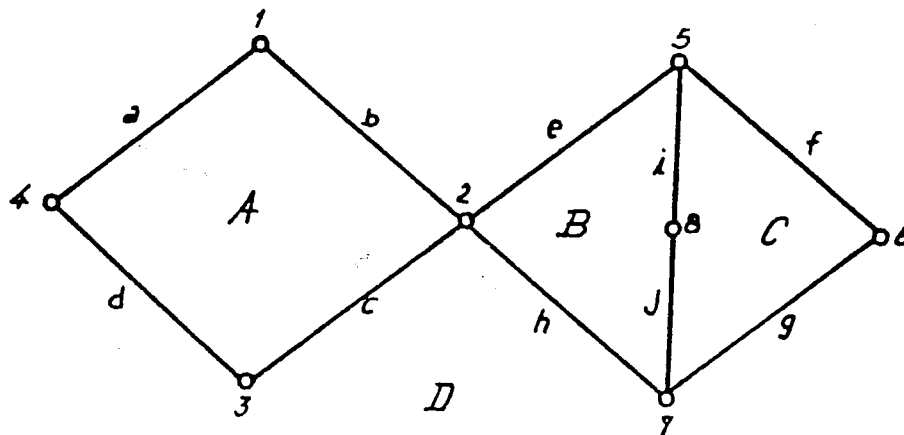
2) any cycle not belonging to R is the sum of cycles of R .

We can now enunciate the following theorem (Mac Lane):

Theorem. A graph is planar if and only if it has a planar mesh.

The proof can be found in (2).

Face. Contour. Consider the following planar realization of a graph.



Observe that the image of any cycle divides the plane into two or more disjoint regions, and that, furthermore, there are cycles (e.g. (a, b, c, d)) such that one and only one of its regions does not contain any (images of) nodes or (lines of) edges; we shall call such a cycle a face if the region referred to is bounded, and a contour if it is unbounded.

Example. In the last figure, the graph realized presents three faces. They are faces $C_1 = (a, b, c, d)$, $C_2 = (e, i, j, h)$, and $C_3 = (f, g, j, i)$, corresponding to regions A, B, and C, respectively. The only contour, $C_4 = (a,$

b, c, d, e, f, g, h), which is not an elementary cycle, corresponds to region D. C_4 is the sum the two elementary cycles C_1 and $C_5 = (e, f, g, h)$. A planar mesh for this realization is $R = (C_1, C_2, C_3, C_4, C_5)$.

Note on the Class of Graphs Considered. From here on we shall restrict our interest, within the class of finite simple graphs we have been considering, to the subclass of non-separable connected graphs of this class.

There are four main reasons for doing so:

- 1) it is known that "a graph is planar if and only if each of its blocks is planar" and "a graph decomposes uniquely into blocks" (3).
- 2) each connected component of a graph can be treated separately as a graph.
- 3) there are algorithms for determining the connected components and the blocks of a graph.
- 4) the semantics of the global problem dealt with in this work, of which graph planarity and other notions such as "dual graph" are only sub-problems, leads us, "in a natural way", to consider "connected components" and "blocks" as semantically more primary and relevant concepts rather than the general notion of graph.

N.B. Henceforth, when not explicitly stated otherwise, by "graph" we shall mean a "simple non-separable connected graph".

Equality Between Planar Realizations. By definition, two planar realizations R and S of the same graph G are equal if for every face F of R there exists a face H of S such that $H = F$, and vice-versa.

Euler's Theorem. For any planar realization of any graph, Euler's

theorem states that the following equality holds,

$$F + V = E + C,$$

where F , V , E , and C are, respectively, the number of faces, nodes, edges, and connected components. Note that no external face is included in F .

No proof is given here of this theorem; however, it can easily be proved by induction on F , V , E , and C .

Uniqueness of the Contour. Boundary. A connected graph is non-separable if we cannot break it at a single node into two graphs, each containing at least one edge; the equivalence of this definition with the previous definition for non-separable graphs is given in (3).

Now, the unbound outside region of a planar realization of a graph from the class of graphs we are considering, is separated from the regions inside by a closed path, the boundary. If this path had any repeated node (or edge) it would be possible to cut the graph, thus contradicting the hypothesis that it is non-separable. Hence the path gives a polygon, the edges of which define a unique cycle: the contour.

External Face. Call the contour of a planar realization R of graph G the external face of G in R .

Interior Node. Interior Edge. Exterior Nodes and Edges. An interior node of a given planar realization of graph G , is a node which is not an extremity of an edge belonging to the contour. An interior edge of the same realization is any edge with at least one interior node as an extremity.

If not interior, any edge or node will be called exterior. According

to this definition, an exterior edge does not necessarily belong to the contour.

Example. In the last figure, the graph there realized is separable. In fact, node two allows to break the graph.

$$F_1 = (a, b, c, d) \text{ and } F_2 = (e, f, g, h, i, j)$$

determine its two blocks $G.F_1$ and $G.F_2$.

G has a boundary in R which is not a polygon, and so does not give rise to a contour. $G.F_2$, on the other hand, has its contour formed by edges e, f, g, and h.

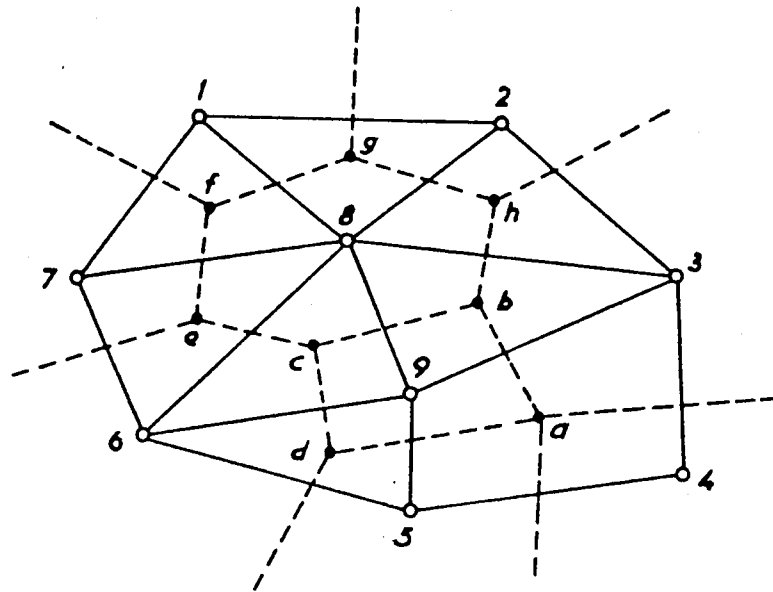
Node eight and edges i and j are the interior node and edges of $G.F_2$, all its other nodes and edges being exterior for this realization.

Suppose that in $G.F_1$ there was an edge with nodes two and four for extremities. This edge is an exterior edge in the realization shown of $G.F_1$. It would in fact be an exterior edge in any of its planar realizations.

Dual Graph. Given a simple connected non-separable planar graph G , and a planar realization R of G , we will next define, in a constructive way, what we mean by the planar realization relative to R of the dual graph G' of G relative to R , or more simply, the dual realization of R , or simpler still, the dual of R .

When no confusion is possible, we will speak of "the dual graph of G " or of "the dual of G " when referring to the dual of R .

Consider the following figure.



In it, the lines in full linking numbered points correspond to the edges of a planar graph G . The numbered points are the images of the nodes of G . The lines and points referred to, determine a planar realization R of G . It can easily be seen that G is simple, connected, and non-separable. Given these circumstances, we will now define the dual of G by means of three constructive steps:

1 - To each face of G in R make correspond a different point on the plane. Each such point is the image, in R' , the dual of R , of a node of a graph G' realized by R' .

2 - Consider one further point, also the image of a node of G' , anywhere in the unlimited region determined by the contour of G .

3 - Link by exactly n lines any two points above obtained, if and only if they correspond to a face and the external face, or to two faces, which have n edges in common; make sure the lines obtained by this process do not intersect themselves or one another.

These lines correspond to the edges of G' in R' .

N.B. Often, the lines linking a point corresponding to a face and the point corresponding to the external face are only partially depicted, since this last point is not always depicted itself. No confusion however arises from this.

The simple and usual way to carry out steps one to three, is to position each point corresponding to a face inside the empty region delimited by the face, and to draw the lines between points in such a way that they cross different common edges of the faces corresponding to the points they link (or the different common edges of a face and the external face). In our previous figure, the dashed lines linking lettered points have resulted from carrying out the above steps in this usual way.

Properties of a Graph and its Dual, for a Given Realization. Now, it is possible to prove (see (3)), that if a graph is planar, the dual graphs obtained from any planar realization are the same. This naturally leads (see also (3)), to state the following condition, relative to any graphs "dual" to one another, and which the previously defined dual graphs obey. Furthermore, it gives yet another characterization of planar graphs: A graph is planar if and only if it has a dual. It is not difficult to accept that a graph and its dual have the following properties:

1 - to each node of the dual there corresponds one and only one face of the original graph.

2 - to each node of the original graph there corresponds one and only one face of the dual.

3 - there is a one-to-one correspondence between the edges of both graphs.

4 - the dual graph of the dual graph is the original graph (3) (thus the reason for the word "dual").

5 - the dual of a non-separable graph is non-separable (3).

Avoiding segment. Let H be a subgraph of G , L a segment of G . L avoids H if at most the extremities of L belong to H .

Segments determined by V in P . Let P be a polygon, V a non-void set of nodes of P . It is easily seen that there exist segments S_1, \dots, S_n contained in P , such that:

$$(1) \quad \bigcup_i S_i = P,$$

(2) each S_i contains exactly two nodes of V , which are its extremities; the segments S_1, \dots, S_n are the segments determined by V in

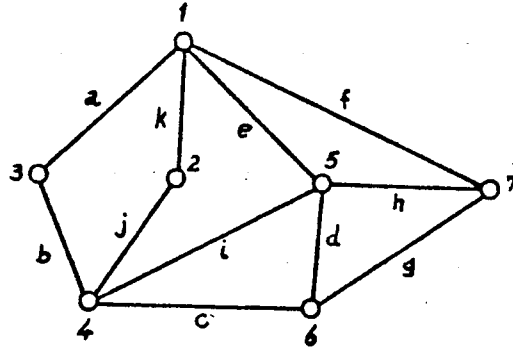
Branch Graph. A branch graph B of graph G is a segment of G such that the extremities of B are the only nodes of B which are possible extremities of edges of G not in B .

Bridges. Residual Segments. Peripheral Polygon. Let P be a polygon of G . We now define in $E(G) - E(P)$ the equivalence relation: " a relates to b " if and only if there exists in G a path avoiding P which contains a and b ". This equivalence relation induces in $E(G) - E(P)$ a partition E_i . Each subgraph $G.E_i$ is called a bridge of P (in G).

For example, in the next figure the polygon determined by cycle (a, b, c, d) has as bridges the subgraph determined by edges f, g , and h , the

one determined by edges j and k , and the one determined by i .

From here onwards, whenever ambiguity does not arise, we shall simply say "bridge of P ", being understood that it is "of G ".



Let \underline{u} and \underline{v} be nodes of a bridge B of polygon P ; there exist edges \underline{a} and \underline{b} with an extremity in \underline{u} and \underline{v} , respectively, belonging to $E(B)$, and so, there exists a segment in G avoiding P and containing \underline{u} and \underline{v} . Thus

- (1) Any bridge of P is connected;
- (2) Let B and B' be distinct bridges of P . If x is a node of B and B' then x belongs to P .

Proof. If $x \notin P$ and x belongs to B and B' , then there exist edges $\underline{a} \in E(B)$ and $\underline{b} \in E(B')$ with x as an extremity. Thus, $L = G \cdot \{a, b\}$ is a segment avoiding P and intersecting B and B' . Hence B and B' are not distinct, in contradiction with the hypothesis.

Let B be a bridge of P . The feet of B in P are the elements of $V(B \cap P)$; by definition $E(B \cap P) = \emptyset$. We shall leave without proof the fact that any bridge of P has at least two feet in P .

Now let V be the set of feet of B in P . We call residual segments of B to the segments determined by V in P .

If bridges B and B' of P obey one of the two following conditions we say that B and B' avoid each other, i.e. are avoiding bridges of P ; otherwise, they are called overlapping bridges of P . We will not prove the equivalence between these two conditions.

(1) the feet of one bridge are contained in a residual segment of the other;

(2) there exist residual segments R of B and S of B' such that $P=RS$.

A polygon of a graph is said peripheral if all its bridges avoid each other.

For the purpose of the graph planarity and graph realizations algorithm given in chapter eight, we will now state without proof, the three following theorems (see (4)).

Theorem. If a polygon has three mutually overlapping bridges, the graph is ~~not~~ planar.

Theorem. If a graph G is not planar, each peripheral polygon of G belongs to at least one planar mesh of G .

Theorem. If a graph G has two subgraphs H and K such that:

(1) H and K are planar, with planar meshes R and R' , respectively;

(2) there exists an elementary cycle C of G such that $C \cap R$ and $C \cap R'$ are symmetric differences of paths. Then G is planar and a planar mesh is obtained for G by determining the symmetric difference of R and R' .

4. References

- (1) MacLane, Saunders (1937) - "A structural characterization of planar combinatorial graphs", in Duke Mathematical Journal, vol. 3, pp.460-472.
- (2) MacLane, Saunders (1937) - "A combinatorial condition for planar graphs in Fundamenta Mathematica, vol. 28, pp.22-32.
- (3) Whitney, Hassler (1932) - "Non-separable and planar graphs", in Transactions of the American Mathematical Society, vol. 34, pp. 339-362.
- (4) Monteiro, L.; Pereira, F. (1974) - "An algorithm for testing planarity and generating planar realizations of graphs", Laboratório Nacional de Engenharia Civil, Lisbon. (To be published).

CHAPTER 4

PROBLEM FORMULATION AND PROFILE OF ITS RESOLUTION

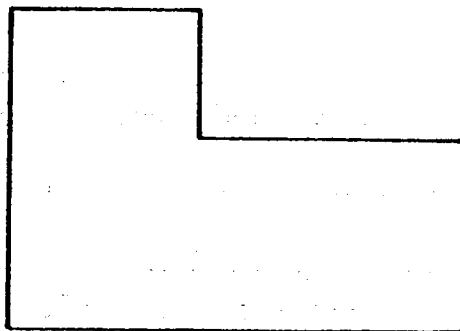
1. INTRODUCTION

In this chapter we state the initial problem formulation, and justify the way we divide it into subproblems within a given representation. In addition, we delineate the methods used for resolution of these subproblems.

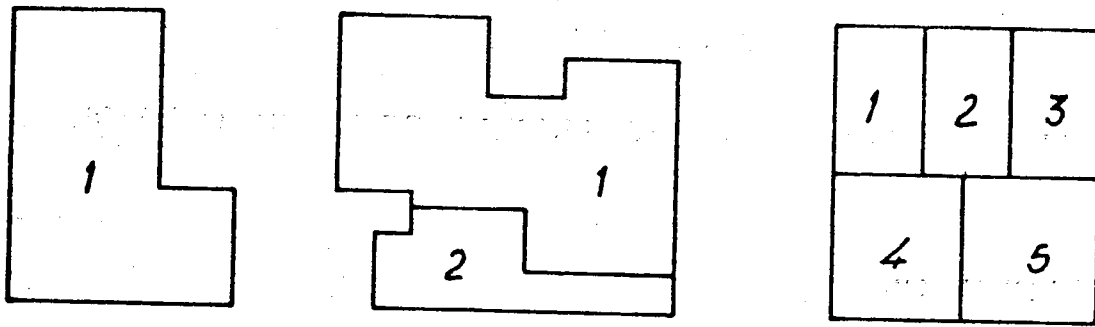
2. INITIAL HYPOTHESES

Hypothesis 1: By a space we mean the area inside a closed polygon on the plane, convex or not, whose consecutive sides form angles of 90 or 270 degrees between themselves.

Example of a space:



By layout scheme we mean any partition of a certain space, whose polygon defines the contour of the layout scheme, into a finite number of other spaces. Examples of layout schemes:



The spaces of the partition are identified by the first positive integers, and any two different spaces whose polygons or boundaries have a common side are said to be adjacent. Each adjacency is expressed by an unordered pair of the appropriate numbers.

The following is a complete list L of the adjacencies occurring in one of the above layout schemes:

$$L = ((1, 2), (1, 4), (2, 3), (2, 4), (2, 5), (3, 5), (4, 5)).$$

Hypothesis 2: It is given a list of pairs of positive integers, and each pair (a, b) is interpreted as meaning that a and b are spaces and that space a is adjacent to space b .

For our purpose $(a, b) = (b, a)$, and since we do not consider any space to be adjacent to itself, we note that all pairs of the form (c, c) , where c is a space, are absent from any given list.

The given list clearly defines a graph, whose nodes and edges are spaces and adjacencies, respectively.

Hypothesis 3: It is understood that all spaces figuring in any given list are rectangular, i.e. have a four-sided boundary.

Hypothesis 4: For each rectangular space, two tolerance intervals are specified on the positive integers, i.e. for each of its two dimensions an or-

dered pair consisting of a minimum and a maximum value is stipulated by means of two positive integers. When no tolerance intervals are given, these are understood to be any. Of course, a general condition is that any maximum value must be greater or equal than its corresponding minimum.

Furthermore, an excess integer is uniformly assigned to each tolerance interval, indicating by how much its maximum or minimum values may be exceeded, given the purpose of assigning dimension values to every space, within the limits imposed by the tolerance intervals and the excess.

Hypothesis 5: For each adjacency pair (a, b) , a tolerance interval is specified on the non-negative integers, which regulates the extent of contact between spaces a and b, along the adjacency expressed by (a, b) . When no particular tolerance interval for an adjacency pair is required, it should be considered to be an unlimited interval on the non-negative integers having zero as the minimum value.

Hypothesis 6: A class of forms is specified for the contour, by giving a description of the class of sequences of convex and concave angles allowed for the contour polygon. This description must be made through an appropriate grammar, developed to that effect.

If no such description is given, a choice has to be made whether the class of forms allowed for the contour is comprised of all possible polygons, or if it to be restricted to rectangular polygons alone.

Hypothesis 7: Some of the spaces, eventually all or none, may be marked as necessarily exterior spaces, i.e. as spaces having at least a point of their boundary in common with the contour.

(If the appropriate adjacency pairs and tolerance intervals are present, some of these exterior spaces may be semantically interpreted as "the north", "the south", "the east", "the west", "the garden", "the street", "the park", etc..)

Hypothesis 8: Since in a layout scheme any side of a space has one, and one only, of two possible relative directions (call them horizontal and vertical), two lists of adjacencies may be given: one with those referring to common sides of adjacent spaces which have necessarily to be horizontal; the other with those referring to necessarily vertical sides.

Note: There are no other initial hypotheses besides the ones explicitly stated until now. All other hypotheses, if any, will be outwardly expressed.

3. INITIAL PROBLEM FORMULATION

Initial Problem: Given the initial hypotheses one through eight, obtain all possible layout schemes in agreement with them, and with the particular data they convey.

4. CHOICE OF A PROBLEM REPRESENTATION SPACE

As was argued in chapter two, we should look for a representation

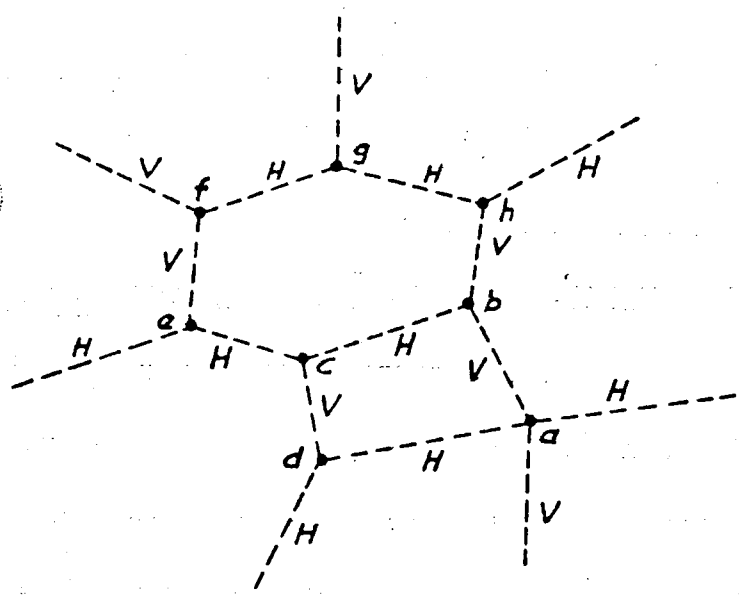
space expressing both requirements and forms.

In chapter three we introduced sufficient graph theoretical notions to show how to cope with the requirements of some of the initial hypotheses.

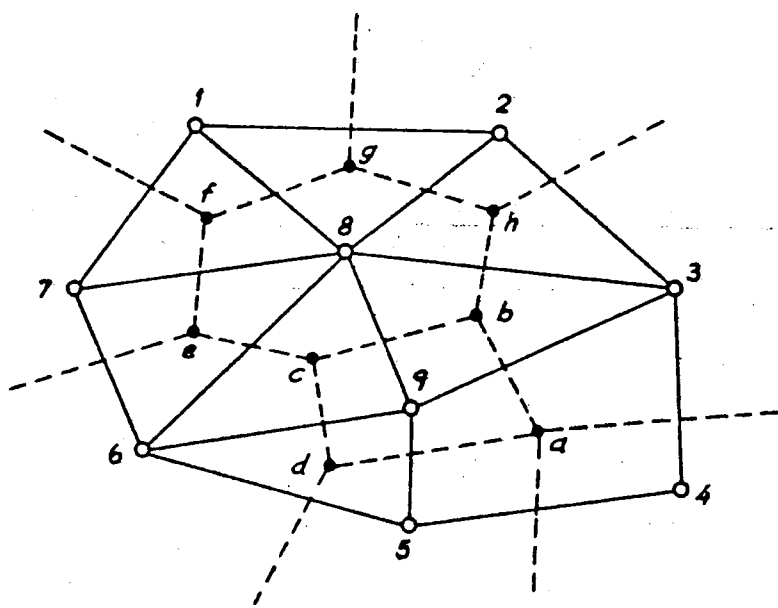
Indeed, the graphs determined by means of H2, have no isolated nodes (i.e. nodes without any edges), and no loops. From H3 it follows, however, that such a graph must also be simple since any two disjoint rectangles can only be adjacent along one common side at the most, and, according to H1, they are disjoint. Hypotheses 4 and 5 can be dealt with by considering flows within the graph in a way to be shown later on, when the problem of introducing the dimensions of the spaces is discussed.

The way hypotheses one, three, six, seven, and eight are accounted for, in a graph representation space, will now be mentioned. With that purpose in mind, consider the following figures.

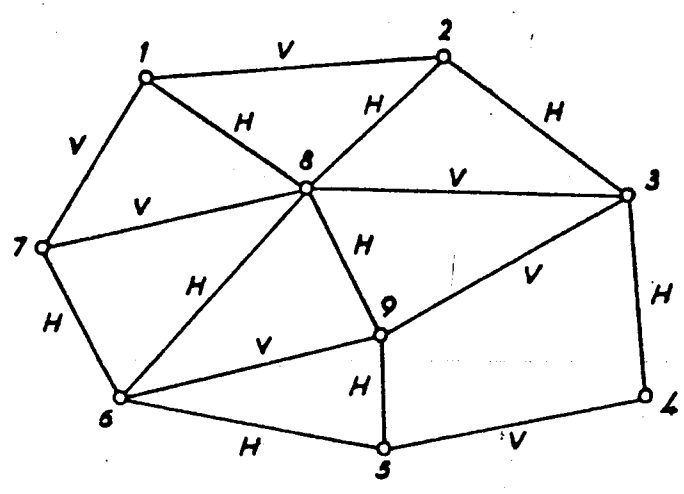
3

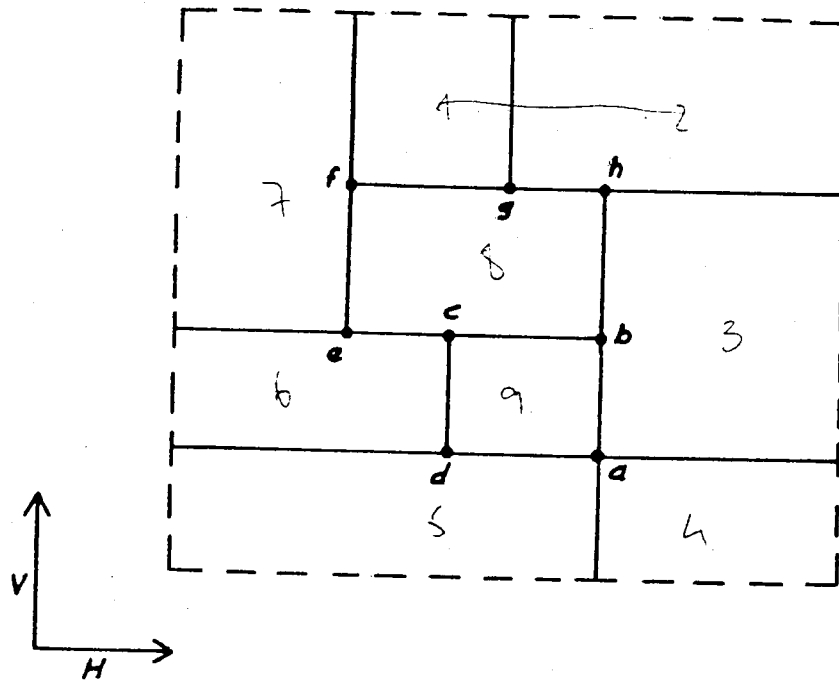


1



2





In the first one, a planar realization of a graph is shown (with numbered nodes and full edges), superimposed by its dual (with lettered nodes and dashed edges). This is an example of the graph referred to in H2.

In the second one, an assignment of H's and V's has been made to the edges of the numbered graph: call it an HV - assignment.

Since there is a one-to-one correspondence between the edges of the graph and its dual, there is a resulting assignment of H's and V's to the edges of the dual, which is shown in the third figure.

We now define, relative to the horizontal and vertical axes of the sheet of paper properties "horizontal" and "vertical" and interpret "H" and "V" as meaning that an edge must be drawn horizontally or vertically, respectively.

The dual can now be redrawn, attending to the assignment of H's and V's to its edges, in the unique way shown in the fourth figure, where a dashed rectangular contour has been added: this contour results from con

sidering all spaces rectangular. Note that the particular dimensions utilized for drawing the figure are at the moment, irrelevant; what is essential is the orientation of the edges, and the fact that they were redrawn attending only to their assignment value and the general condition that they could not have any common points except the extremities; this condition follows from the fact that the dual of a planar representation of a graph is itself planar.

We have this way obtained a layout scheme of rectangular spaces with a rectangular contour. Of course, not all HV - assignments in a given graph determine a layout scheme of rectangular spaces. There are also graphs for which there is no HV - assignment giving a layout scheme of rectangular spaces or a R-layout-scheme.

This leads us to state Problem 1 and Problem 2.

5. PROBLEM DECOMPOSITION

Problem 1: Given a graph G , under what necessary and sufficient conditions does there exist at least one HV - assignment leading to a layout scheme of rectangular spaces; i.e. a permissible HV - assignment.

Problem 2: When does a permissible HV - assignment, or PHV - assignment, lead to a R-layout-scheme with a specified type of contour, eventually, rectangular.

Now, these two problems immediately suggest several others.

One obvious necessary condition of Problem 1 is for graph G to have at least a planar realization, and hence to be planar. It may have however various planar realizations; for each such realization R , the question then arises if there exists at least one PHV - assignment giving rise to a

R - layout-scheme by means of the dual of R.

These comments lead us to decompose Problem 1 into the following five subproblems.

Subproblem 1.1: Given graph G, find out whether or not it is planar.

Subproblem 1.2: If G is planar, find all its planar realizations.

Subproblem 1.3: If G is not planar, how to alter G to make it planar. The different ways of transforming G, result into corresponding subproblems; such transformations however, to be efficient, should be closely matched to the conditions of Subproblem 1.4.

Similarly, any problem or subproblem arising from the necessity to transform initial data so as to comply with given conditions, should be solved by considering only those transformations which also comply to subsequent problem or subproblem conditions, if they are to be economic.

Subproblem 1.4: For any planar realization of G, under what conditions does it allow at least one PHV assignment.

Subproblem 1.5: For each realization of G allowing at least one PHV - assignment, find all its PHV - assignments.

Problem 2 is actually made up of two subproblems.

Subproblem 2.1: How to specify, using a graph theoretical representation, the types of contour forms to be allowed for the R-layout-schemes derived from the dual of a planar realization of a graph.

Subproblem 2.2: Once a given type of contour form is specified, in graph theoretical terms, for a given planar realization of a graph, derive only

those PHV - assignments leading to R-layout-schemes with that type of contour form.

We will relegate the statement of problems and subproblems pertaining to questions of dimension to a specific chapter, after having given solution to the merely topological problems stated up to this point. Indeed an important property of the solution method utilized for solving the Initial Problem, is that the merely topological questions can be and have been separated from the metrical ones. This is accomplished by having an overall generative mechanism producing topological solutions, which function as input candidates to the dimension giving mechanisms for producing metrical layout schemes.

Separateness is in fact a characteristic which manifests itself throughout the overall solution method. It occurs, for example, when Problem one and two are decomposed into the sequence of steps:

- a) finding if a graph is planar ;
- b) discovering one of its planar realizations;
- c) extracting from it all its other planar realizations;
- d) finding for each such realization those allowing at least one PHV - assignment ;
- e) obtaining all PHV - assignments, of any realization having at least one;
- f) restricting the PHV - assignments to only those which conform to a specified contour type.

Other situations where separateness also occurs are those of:

- a) dividing any given PHV - assignment of a given representation into the unique partial assignment present in all PHV - assignments

- , for that representation (its nucleus) and the particular completion of that partial assignment corresponding to the PHV - assignment considered.
- b) considering that the absolute orientation of each edge is composed of its orientation relative to the layout scheme plus the absolute orientation of the whole layout. It will be shown that in a PHV - assignment the relative orientation of every edge is simply determined by the "H" or "V" assigned to it. Thus, the edges of a graph need not be oriented; only the layout as a whole will have to be oriented, relative to an external referential, for the purpose of being drawn.
- c) since the spaces considered are rectangular the dimensional flow of each dimension through a given R-layout-scheme is independent of the other, and can be treated separately by considering an appropriate subgraph of the original one, which expresses only those adjacencies pertinent to the dimensional flow in question. Furthermore when no tolerance intervals are supplied (i.e. no dimensions are stipulated whatsoever), one can always draw a R-layout-scheme, given its PHV - assignment, by finding, for example, the minimum dimensions each space must have in terms of a unit module (i.e. any dimension will be some multiple of the module). Thus, a complete separateness is possible between any PHV - assignment defining the relative positions of the spaces of a R-layout-scheme, and the desired dimensions for those spaces conveyed by given tolerance intervals.

The import of separateness resides in the fact that diverse heuristics can be used for each separate subproblem, and more easily implemented. Moreover, interaction facilities become better located and defined.

Separateness also permits program modularity and easiness of reformulation, because the design of each component can be carried out with some degree of independence of the design of others, since each will affect the others largely through its function, and independently of the details of the mechanisms that accomplish its function. (This approach to the design of complex structures was explored by Christopher Alexander in (1)).

One way of considering the decomposition, but ^c acknowledging that the interrelations among the components cannot be ignored, is to think of the design process as involving first the generation of alternatives and then the testing of these alternatives against a whole array of requirements and constraints. There need not be merely a single generate-test cycle, but there can exist a whole interconnected series of such cycles. The generators implicitly define the decomposition of the design problem, and the tests guarantee that undesirable consequences will be noticed and pruned. Alternative decompositions correspond to different ways of dividing the responsibilities for the final design between generators and tests (2).

6. LAYOUT SCHEMES FROM GRAPHS

To solve Problem 1 completely we need to solve its five Subproblems. Subproblems 1.1 and 1.2 have already been treated in the chapter on graph theory, and Subproblem 1.3 is will not concern us in this work because we presume that alteration of a graph to make it planar will always be carried out by the user, although the planarity algorithm gives relevant in -

formation to that end.

The solution given to Subproblem 1.5 only becomes comprehensible after we state the conditions referred to in Subproblem 1.4, and show that they constitute indeed its solution.

However, the actual details of the solution contrived for Subproblem 1.5 will be dealt with later on, since they are rather complex to introduce at this stage.

We will now work out the conditions mentioned in Subproblem 1.4, dividing them into two groups -- conditions A and conditions B.

Group A expresses those conditions which are "rigid", i.e. that either are or are not obeyed by a graph, independently of any HV - assignment.

Group B refers to conditions stipulating which HV - assignments, are permissible, if any, once it is known that conditions A are met. For each realization of a graph, since there may be several HV - assignments obeying conditions B, finding all of them gives rise to the family of the R - layout - schemes obtainable from each given planar realization. This way, the set of all planar realizations produces the set of all such families.

Conditions A and B will first be presented jointly, and then justified.

Conditions A.

A0 - Graph G must be finite, simple, connected, non-separable, and planar, admitting at least one planar realization R which obeys all subsequent A and B conditions.

A1 - Every interior node of G in R must have four or more edges.

A2 - Every face of G in R must have either three or four edges;

i.e. each of them is either triangular or quadrangular in shape.

Conditions B.

B0 - Each edge of G in R must be assigned either an 'H' or a 'V', such that all other subsequent B conditions are met. We will often speak of 'H' and 'V' as two different colours, 'E' and 'V' respectively, and speak of an HV - assignment as a colouring of the graph.

B1 - No triangular face of G in R can have its three edges assigned the same colour.

B2 - All quadrangular faces of G in R must have opposite edges assigned the same colour, and non-opposite edges with different colours.

B3 - The edges of any interior node of G in R must be coloured in such a way that they may be grouped into four successive colour groups around the node, opposite groups having the same colour, and non-opposite groups having different colours.

Justification of conditions A.

A0 - Since the layout-schemes are to have rectangular spaces the graph must be simple, because two disjunct rectangular spaces cannot be adjacent on more than one side. Note, however, that the process of obtaining the dual would work as well with non-simple graphs, giving then rise in a natural way to non-rectangular spaces, as we shall show on the section on lines for further research (chapter 9).

We assume that the graph is connected since if it were not, each of its connected components could be separately considered without loss of generality. For the same reason, we also assume that the graph is non-separable, since each of its separate blocks could be considered separately, given a planar realization of the graph.

The graph must be planar if one of its realizations is to have a dual. Furthermore, only its planar realizations can eventually determine R-layout-schemes.

Finiteness is assumed by hypothesis one.

A1 - Each interior node of G in R is going to determine a space in a layout-scheme. If such a space is to be rectangular, and surrounded by rectangular spaces adjacent to it, then each interior node must have four or more edges since a rectangular space cannot be completely surrounded by less than four other rectangular spaces adjacent to it.

A2 - First of all, each face will necessarily have three or more edges because the graph is simple.

Now, in the dual R' of R, each node stands for a corner where several spaces meet, and corresponds to a face of G in R. Since, by hypothesis three, all spaces are rectangular, each of the spaces meeting in a node of R' will occupy, around the node, an angle of either 90 or 180 degrees. Because around the node only 360 degrees are available, this gives a maximum of $360/90 = 4$ spaces meeting in any corner; i.e. each face of G in R will have to have a maximum of four edges, otherwise it would give rise to a node in R' with more than four edges, thus dividing the region around that node into more than four regions, making it impossible for all those regions to be rectangular.

Justification of conditions B.

B0 - In a R-layout-scheme the edges form partitions between spaces. Since the spaces are rectangular, once an external reference direction is

provided, the whole layout scheme can be oriented in such a way that the partitions between the spaces are either parallel or perpendicular to that direction. Thus, we have adopted the designations horizontal and vertical to express the relative orientation of the edges of any layout scheme.

Up to this point, we have been using 'H' and 'V', or 'E' and 'V', when referring to the edges of G in R. Because a one-to-one correspondence exists between the edges of G in R and the edges of the dual R' of R, we may also refer to each edge of G in R as being necessarily either horizontal or vertical. Hence the reason for having to assign to each and every edge of G in R one of two different colours, denoted by 'E' and 'V'.

Whereas conditions A establish necessary properties G must have for at least one PHV - assignment to be possible, conditions B indicate the restrictions to which a HV - assignment must comply with for being a PHV - assignment, since it is clear from their justifications that not all HV - assignments are permissible.

B1 - To a triangular face of G in R there corresponds a node in the dual R' of R with three edges. These edges are the partitions dividing the region around the node into rectangular spaces. Now these partitions can only be horizontal or vertical, as has been argued. It is apparent that no more than two partitions of the same type (i.e. horizontal or vertical), can meet in any given node. It follows that the edges of a triangular face cannot all have the same colour.

B2 - The same way as before, each quadrangular face gives a node where four partitions meet; two of them will have to be horizontal and

the other two vertical. Furthermore, the horizontal and vertical partitions will have to alternate around the node. Condition B2 follows.

B3 - An interior node I of G in R , in any R -layout-scheme obtainable by a PHV - assignment, gives a rectangular space which is completely surrounded by other rectangular spaces. The partitions between that space S and the surrounding ones, are the four sides of its boundary. To each side of S there corresponds a number of edges of G in R , one for each surrounding rectangular space adjacent to S on that side. Thus, the four sides of S determine four groups of edges around node I , each group corresponding to the space adjacent to S on one of its sides.

Since the four sides of S are alternately horizontal and vertical, the groups of edges around node I are also alternately horizontal and vertical, with all the edges of any one group being of the same type. By a horizontal (vertical) group of edges of a node I , we mean a group of successive edges around I which are all horizontal (vertical). Of course, all edges of a group of edges have the same colour. Condition B3 follows.

Comments on conditions A and B. Conditions C.

It should be remarked that colouring conditions B, expressed above, take into account all necessary rules for colouring a realization R of a graph except the exterior nodes; i.e. there are no other rules besides those which derive from them. Thus, the set of necessary rules given is also a sufficient set of rules for colouring R , exception being made for the exterior nodes, whose colouring rules are dependent upon the contour description prescribed by Hypothesis 6.

In fact, conditions B apply to all edges through $B0$, to all faces, through

B1 and B2, and to all interior nodes through B3.

Since without a further condition on the exterior nodes we cannot properly speak of PHV - assignments, we shall give, without for the moment proving them, three contour conditions, C0, C1, and C2.

C1 is the most general condition on contour forms; i.e. it admits all contour forms complying with Hypothesis 1. Condition C2 expresses the requirement to be met if the contour is rectangular. C0 is a rectangularity condition on any exterior space of any contour.

Let N1 be the number of exterior nodes with exactly one group of edges. Let N2 be the number of exterior nodes with exactly two differently coloured groups of edges and with its two boundary edges of a different colour. Let N4 be the number of exterior nodes with exactly four alternately coloured groups of edges around them.

Intuitively, a node contributes to N1 if it gives rise to a rectangular space occupying exactly two convex corners of the contour of the layout scheme; to N2 if just one convex corner is occupied; to N4 if just one concave corner is occupied. If no corner is occupied then the node contributes to N0: i.e. all other nodes.

C0 - No exterior node may have more than four alternately coloured groups of edges around it. Thus it can only have one, two, or four groups, because three groups is impossible to have.

C1 - All contour forms allowed by Hypothesis 1, comply to the following equation:

$$(2 \times N1 + N2) - N4 = 4$$

This result means intuitively that the difference between the number of convex and concave rectangular corners of the contour of the layout scheme must be

four; i.e. the layout must be closed.

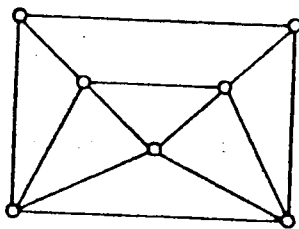
C2 - All rectangular contour forms and only them comply to the following equation, besides the general equation stipulated by condition C1:

$$N_4 = 0.$$

The introduction of this condition into the general equation of C1 gives the intuitive result that the total number of convex corners of a rectangular contour form is four.

Let us now proceed with our comments on conditions A e B.

First we show, by exhibiting a counter-example, that a planar realization satisfying conditions A, does not necessarily admit a PHV-assignment; i.e. as one proceeds to colour it respecting conditions B, one might find it impossible to complete the colouring process. A simple counter-example is the next figure.



This realization obviously obeys conditions A0; furthermore, all its exterior nodes have four edges, and all its faces are either triangular or quadrangular.

Since there are only two colours, two of the edges of the triangular face determined by the three interior nodes, will have to have the same colour. But both edges belong to one same interior node which, having two more edges, cannot have its edges coloured in compliance with

condition B3.

Thus, conditions A and B place a restriction on the classe of planar realizations that can produce R-layout-schemes through PHV-assignments. Some such restrictions can eventually be express in purely "geometrical" terms, as we have done with the five Properties enounced below, all of which are deducible from conditions A and B.

One is then confronted with the question of deciding to what extent should ever more sophisticated "geometrical" conditions be derived. If on the one hand they make it possible to "filter" unproductive realizations, on the other, being difficult to evaluate, they may well lead to a decrease in efficiency.

These comments are meant to call into attention two different perspectives from which to envisage conditions B:

(1) as originating the rules to be followed for the process of colouring a planar realization.

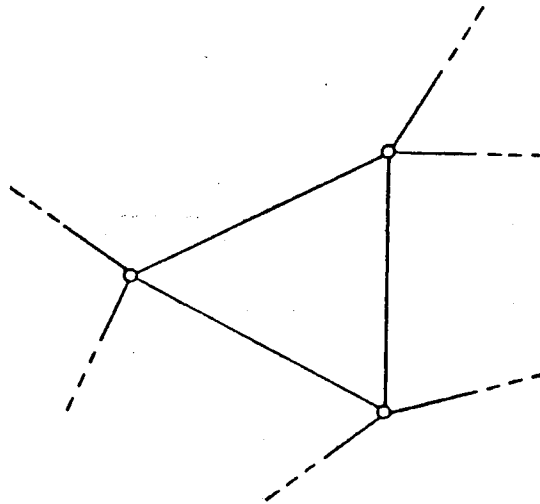
(2) as providing tests to be carried out at each step of that colouring process.

These two perspectives will be given further consideration when the colouring methods of the colouring process are expounded in detail.

Five "geometrical" properties. The five "geometrical" properties deducible from conditions A e B that we present, are effectively used, in the computer program developed, to "filter" input planar realizations.

Property 1: At least one of the nodes of every triangular face linking interior nodes must have five or more edges.

The situation referred to is depicted in the following figure:



where, by hypothesis, each node has only four edges, and is supposed to be interior.

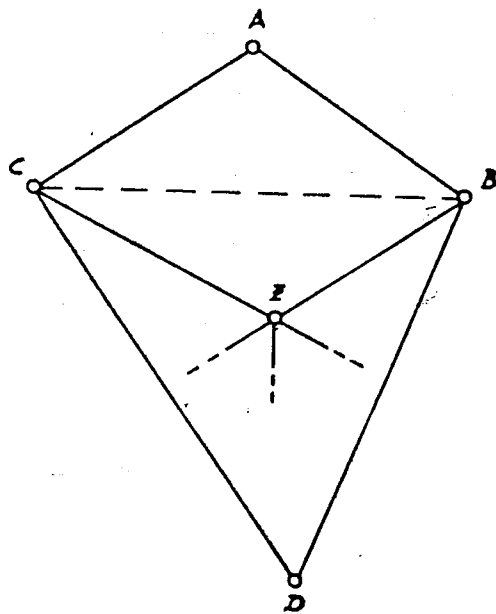
According to B0, each edge must have one of two colours. It follows that two of the edges of the triangular face will necessarily be of the same colour, in any colouring not infringing condition B1. But both edges belong to the same interior node which, having only two more edges, cannot have its edges coloured in compliance with condition B3. This proves the above property.

Property 2: No triangular polygon may enclose interior nodes.

Consider the three nodes of a triangular polygon and their corresponding rectangular spaces. Now, the three rectangular spaces are not sufficient to cover any combination of rectangular spaces corresponding to the interior enclosed. This proves property 2.

Property 3: There may not be a quadrangular polygon enclosing a quadrangular face having two consecutive edges in common with it.

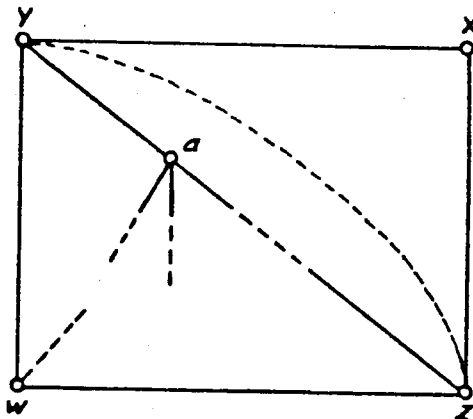
Consider the next figure with edge (C, B) absent. Suppose it was possible to colour it according to conditions B.



It would then be possible to obtain from it the above figure, with edge (C, B) present and with a legitimate colouring, since the colour assigned to (C, B) is irrelevant from the point of view of the previous coloured situation. But that would contradict Property 2. This proves the stated property.

Property 4: There may not be a quadrangular polygon enclosing a quadrangular face having one edge in common with it.

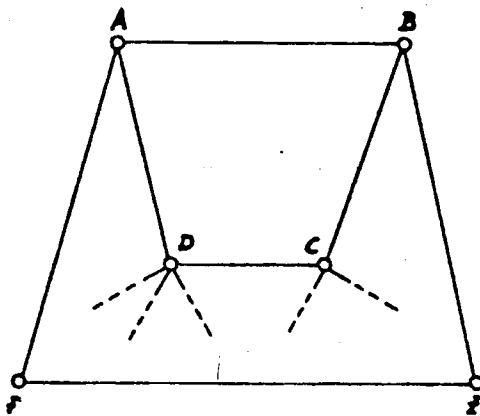
Consider the next figure. Suppose it was possible to colour it according to conditions B.



From it one could still obtain a duly coloured figure by fusing nodes A and B into one and abolishing edge (A, B), whatever colour it had. But then Property 2 would be contradicted. This proves Property 4.

Property 5: If there are interior nodes enclosed by a quadrangular polygon, then each node of the polygon must link to at least one such node.

Consider the following figure, where there is no edge from node X



to any interior node enclosed by the polygon. Nodes Y and Z cannot be linked by an edge interior to the polygon, as depicted by the dashed line, because that would infringe either the planarity

condition or Property 2. Thus, node X must belong a quadrangular face determined by it and nodes Y, Z, and one more node, a. Property 5 now follows from Property 3.

With these properties stated, we end our comments on conditions A and B.

References

- (1) Alexander, C. (1964) - "Notes on the synthesis of form", Harvard University Press.
- (2) Simon, H. (1969) - "The sciences of the artificial", M.I.T. Press.

CHAPTER 5
OBTAINING ALL POSSIBLE LAYOUTS FROM A REALIZATION
OBEYING CONDITIONS 'A'

1. INTRODUCTION

Once a given realization is known to obey conditions A, then, to obtain all its PHV - assignments, one must obtain all possible HV - assignments obeying conditions B, C0, and C1, stated in chapter four. Of course, it may happen that no PHV - assignment exists for the given realization. If that is the case, it will always be recognized by the above mentioned conditions; one or more of them will not hold for any of the possible ways of colouring the edges, as they are tried out.

Properties one to five of chapter four; can be used to weed out unproductive realizations, but from a strictly theoretical point of view, they are redundant and may be ignored.

It should now be remarked that, from a problem - solving perspective, once it is known that a given realization conforms to conditions A, the questions of determining if there exists at least one PHV - assignment and of finding them all, have been fused into one and the same process. Thus, the methods used are of a constructive character. This means that, instead of analysing in detail, by various calculations, if the said realization has or has not at least one PHV - assignment, running the risk it does not, and, in case it does, of then having to find them, we rather prefer to start at once by computing, in an efficient way, all possible "sensible" colourings, thereby compensating the risk of not finding any complete permissible

colouring (i.e. a PHV - assignment) with the information that can be gleaned from a partial colouring of the realization, and from detecting why the completion of that colouring is impossible. To sum up, the methods utilized are said constructive because, while determining whether or not there exists at least one PHV - assignment for a given realization, one already obtains, in the affirmative case, the possible PHV - assignment(s).

2. THE COLOURING METHODS

In this section we present and justify the methods that lead us to all PHV - assignments for a realization complying with conditions A.

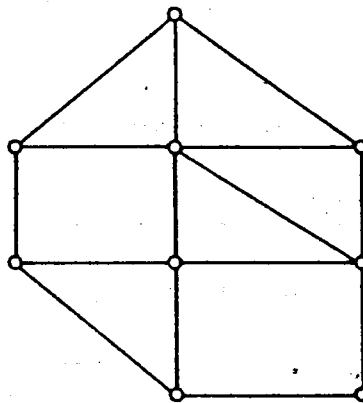
General Considerations. A first question that can be posed is if there really exists a systematic process of engendering all PHV-assignments. There is one easily conceived process we will now examine.

Knowing that there is only a finite number, say N , of edges, we know that the number of all different HV - assignments will also be finite, and in fact equal to 2^N . However we do not distinguish two HV - assignments resulting one from the other by exchanging the H's and the V's throughout. Such a change is immaterial because conditions B and C are symmetrical in regard to 'H' and 'V', and also because, taking an external direction as reference, that exchange amounts simply to a rotation of the correspondent R-layout-scheme by ninety degrees in any rotation sense.

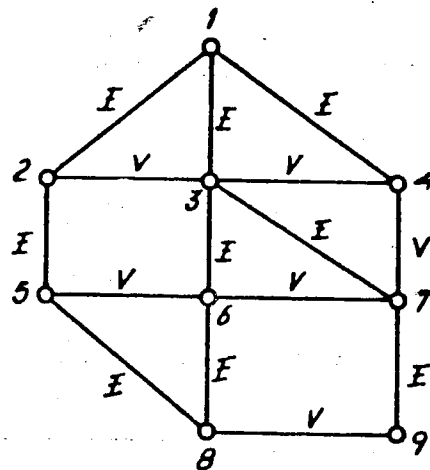
Thus, the number of "distinguishable" HV - assignments reduces to $2^{N/2} = 2^{N-1}$. The process itself consists in generating all HV-assignments and pruning them with the colouring and contour conditions. Its principal drawback is its lack of selectiveness, given that the number of PHV - assignments is always much smaller than the number of HV - assignments.

That is why we have used the phrase "all possible sensible colourings".

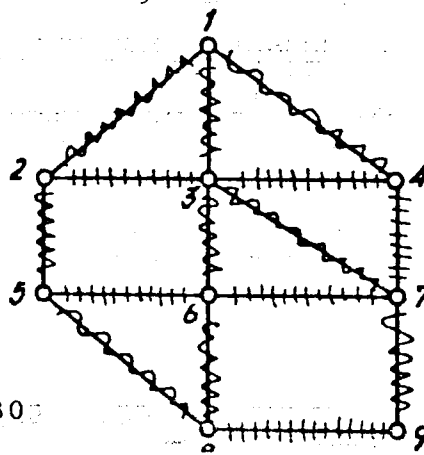
To give an idea of the disparity between the two numbers we present the following realization, where the edges are fifteen. The number of HV-assignments is $2^{14} = 16,384$, while the number of permissible ones with ~~any~~ contour is 40, and the number of permissible one with a rectangular contour is 32. Note that only one of the exterior nodes can give rise to a concave corner, since only one of them has four edges (as pointed out in the discussion preceding the statement of conditions C).



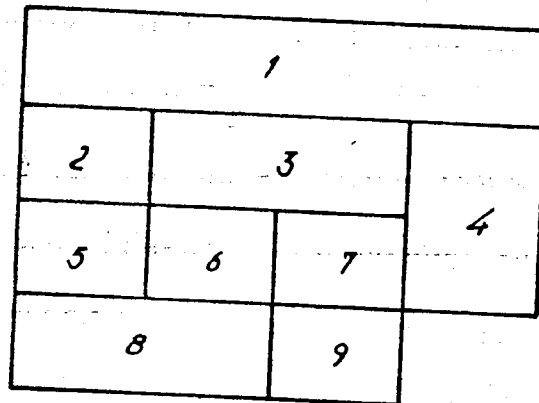
A permissible assignment is:



The same assignment will as well be represented as:



This assignment gives the layout-scheme below:



A sensible way of colouring will be one that explores the structure of the problem so as to:

- 1) avoid the repetition of assignments and, consequently, of solutions.
- 2) be efficient in terms of number of solutions relative to effort dispended; i.e. avoid as many as possible non-permissible HV - assignments.

Between the two extremes of systematically trying out all assignments and of randomly generating assignments, the methods for colouring will have to be able to profit from information about the problem structure. In this sense, they will be heuristic methods. As noted in chapter one, being heuristic does not necessarily mean that they cannot be sensible and thorough.

To gather information about the structure of the problem of colouring any particular realization, we will have to carry out a detailed analysis encompassing the generality of all possible colouring "situations"; and by colouring situation we mean any partially coloured stage of a realization. The set of all colouring situations of a realization defines the representation space for the problem of colouring that realization, where each state is a colouring situation.

The above referred analysis will consist, firstly, in discerning some relevant properties of a realization from the point of view of colouring it; secondly, in establishing a typology of colouring situations fundamented upon the properties encountered.

The importance of such a classification is that by defining appropriate sets of colouring situations, one can construct a hierarchy of problems and subproblems, from which the methods to be proposed will easily follow.

Indeed, it was precisely such a problem decomposition that lead us to enounce the said methods; i.e. by defining certain sets of colouring situations, we accomplished a problem decomposition leading to the stipulation and to the combination of a set of colouring methods satisfying our requirements.

Colour Properties and Colour Consequences

Two important notions will now be given: those of "colour property" and of "colour consequence".

By a colour property of a realization we mean a well - defined set of colouring situations which is specified in terms of the "geometric" components alone of the realization (i.e. nodes, edges, faces, and boundary).

By a colour consequence of a colouring situation, we mean a unique assignment of colour to a set of non-coloured edges of the colouring situation, which follows inevitably from the application of colouring conditions B. i.e. given a colouring situation, there is only one way of colouring some of the non-coloured edges without infringing colouring conditions B (even if conditions C are transgressed).

The reason for not taking conditions C into account, is that, since the contour forms desired may vary, we do not use them for extracting colour consequences in the general program elaborated. Instead, conditions C are used as tests for pruning undesirable or unpermissible contour forms of completed colour assignments. This decomposing of a realization into a contour and its interior is, after all, a way of dividing the colouring problem into two, making it easier, furthermore, to conceive and implement man/machine interaction facilities aimed at the reformulation of contour requirements, separate from those providing for the modification of constraints interior to a given realization. However it will be possible, and in certain cases desirable, to use conditions C to give colour consequences. This possibility shows that, similarly to conditions B, conditions C may also be envisaged either as generating or as testing devices.

Let us next examine some general colour consequences.

(1) Triangular faces. If a triangular face has two of its edges assigned one same colour, and the third edge uncoloured, then, a colour consequence of such a colouring situation is, by condition B1, that its remaining uncoloured edge be assigned the other colour.

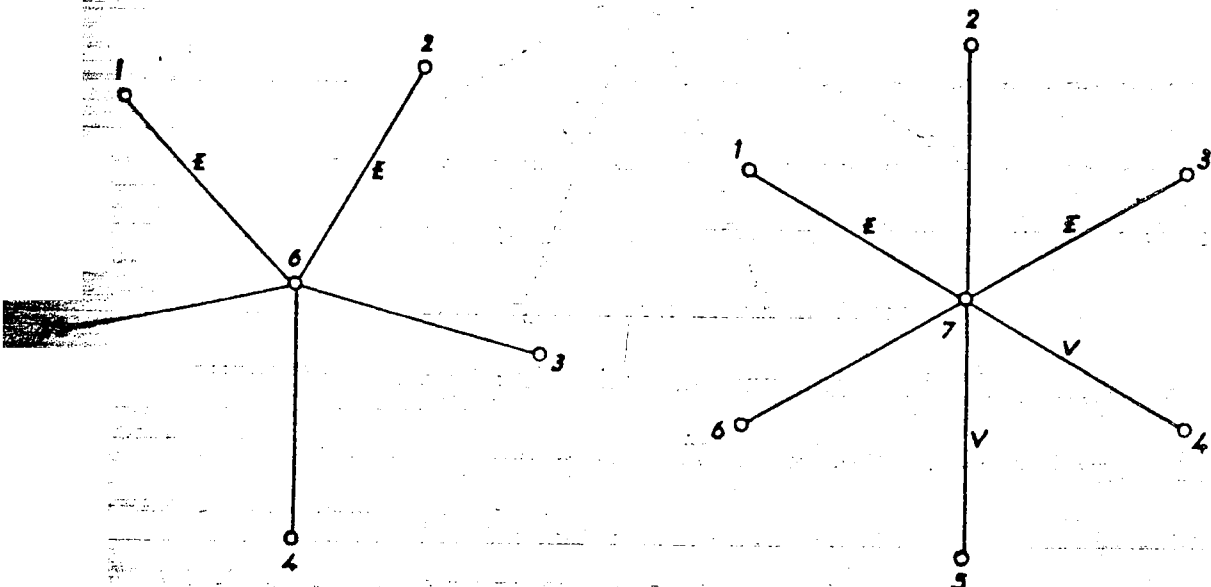
(2) Quadrangular faces. Any quadrangular face having one, two, or three of its edges already coloured in a permissible way, has, as a colour consequence, the assignment of colours to its remaining non-coloured edges in the way determined by condition B2.

(3) Interior nodes with four edges. An interior node with four edges, having one, two, or three of them already permissibly coloured, has, as a colour consequence, the assignment of colour to its remaining uncoloured edges, in the way determined by condition B3.

(4) Interior nodes with more than four edges. There exist a great variety of colour consequences for an interior node with more than four edges, having some but not all of its edges already permissibly coloured. All colour consequences result, however, from applying condition B3 to the node. Before proceeding to examine in detail the possible colouring situations that can arise, we will first show, by means of an example, one such situation. Next, we shall give a few definitions which make it possible to categorize all conceivable situations into 25 cases, where 9 present a non-empty colour consequence, 9 an empty colour consequence, and 7 are permissible colouring situations, so that the colouring process should regard them as part of a colouring situation.

The classification into categories of the colouring situation of an interior node with more than four edges, corresponds to the evaluation the computer program carries out before extracting any colour consequences, deciding to backtrack.

Let us now examine the colouring situation shown on the left of the following figure:

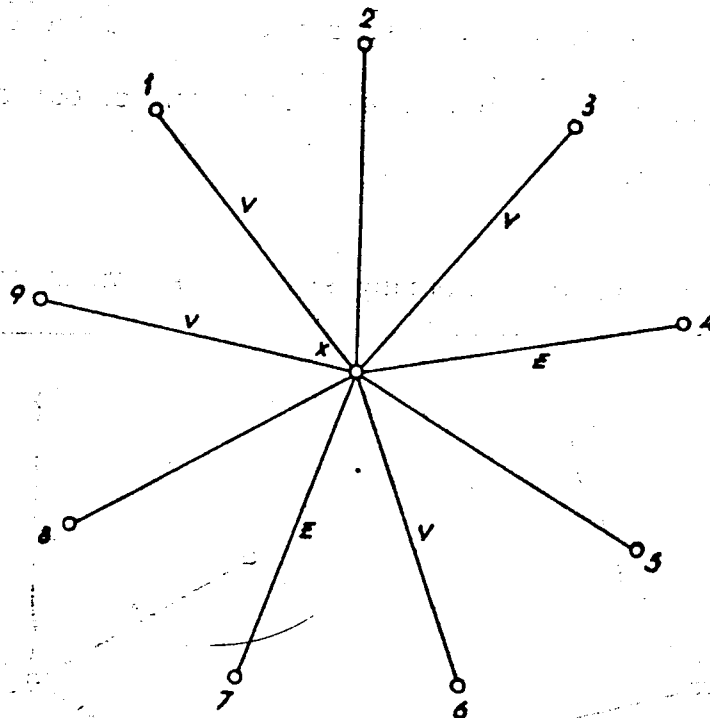


It is easily recognized that edges (5, 6) and (3, 6) will have to be coloured 'V', and that edge (4, 6) will have to be coloured 'E', so that the four groups of edges stipulated by B3 may be formed.

In the colouring situation shown on the right of the above figure, the only colour consequence is that of edge (2, 7) which will have to be assigned 'V'. Note that edge (6, 7) can have any colour, thus showing that colour consequences may not be exhaustive.

Let us now proceed to the detailed description of the colour consequences that may follow from colouring situations regarding the edges of interior nodes with five or more edges (or, simply, IN5 nodes).

Consider the following figure.



Six coloured edges are shown, and also three non-coloured ones, belonging to the same IN5, x . The arrow has the purpose of giving meaning to sentences such as, for example, "consider the edges between edges (3, x) and (6, x)". We are then referring to edges (4, x) and

(3, x), in the order given by the arrow, and starting with edge (3, x);
on the other hand, we would have said "the edges between (6, x) and
(3, x)", then we would be referring to edges (7, x), (8, x), (9, x), (1, x),
and (2, x).

N.B. In the following figures we shall omit the arrow but maintain
the convention.

We will now introduce four concepts. The first is that of number of
coloured edges of a IN5, which we denote by NA . In the previous figure
 $NA = 3$.

The second concept is that of number of groups of coloured edges of a IN5,
denoted by NG - a method for determining NG is now explicitly given. The
method is as follows:

- (1) if the IN5 has no coloured edge $NG = 0$;
- (2) if the edges of the IN5 all have the same colour, either E or V,
then $NG = 1$;
- (3) if both colours coexist in the edges of the IN5, we proceed in the
following way:
 - (i) take any coloured edge such that the first edge encountered
after it in the counter-clockwise sense has a different colour.
In the previous figure, any of edges (4, x), (6, x), (7, x),
or (9, x) could have been chosen. Make $NG = 1$;
 - (ii) next, rotate in sense indicated by the arrow and add one to
 NG every time a coloured edge is found with a colour diffe-
rent from the colour of the most recent coloured edge found.

This process ends when the edge immediately before the chosen in (i) is found.

The value of NG is the one it has after (ii) has been carried out.

This process of identifying IN5 nodes is the one used in the computer program developed. We will exemplify its application to the last figure.

Suppose that, in accordance with (i), edge (6, x) is chosen; NG is equal to 1 and, following (ii), the edges from edge (7, x) to edge (5, x) are considered in succession.

(7, x) has a colour, which is opposite to the one of the most recent edge encountered, (6, x) thus, $NG = 2$;

(8, x) is not coloured; $NG = 2$;

(9, x) has a colour different from the last found one; $NG = 3$;

(1, x) is coloured but has same colour as last one; $NG = 3$;

(2, x) is not coloured; $NG = 3$

(3, x) has same colour as last; $NG = 3$;

(4, x) has different colour from last; $NG = 4$;

(5, x) is not coloured; $NG = 4$.

There are, then, in the IN5 inspected, four groups of coloured edges. It can be easily recognized that NG can never be odd, except 1.

Now we arrive to the third concept to be defined, which only plays part when NG is different from zero and one.

In a IN5, an edge is between two groups if it is not coloured and the colour of the first coloured edge found in the sense of the arrow is

ifferent from the colour of the first coloured edge found in the other sense. The non-coloured edges that are not between two groups, are edges within a group.

Still referring to the last figure, of its three non-coloured edges only edge (2, x) is an edge within a group. The concept that interests us is, however, that of number of edges ^{between} within groups, denoted here by N_3 . In the previous example, one has $N_3=2$.

The fourth and last concept is that of maximum number of consecutive non-coloured edges, NAR . Observing the last figure, NAR is seen to be 1.

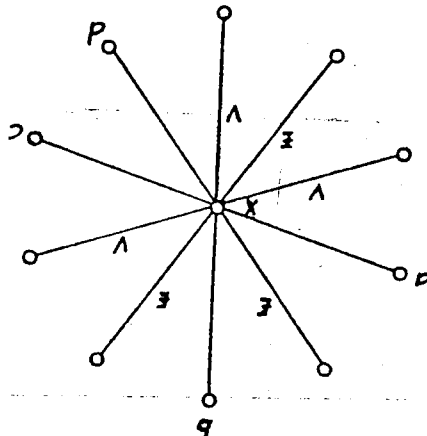
NA , NG , N_3 , and NAR are all that we need to decide the colour consequences of any IN5 colour situation, or that the colour situation does not admit continuation.

We now proceed to present all the various cases that may be.

I - NG greater than four. It is easily seen that, whatever the colour assigned to the remaining non-coloured edges, NG will keep being greater than four. Condition B3 is thus violated.

II - $NG = 4$. The edges within groups (if any) must necessarily be coloured with the group colour. The colour of other non-coloured edges is arbitrary.

Example:



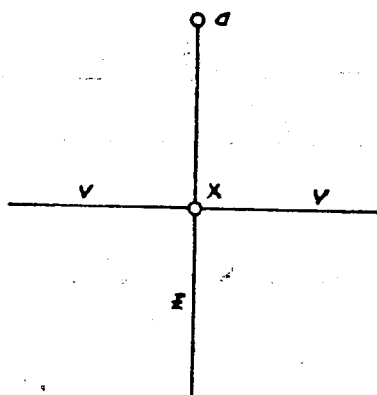
The colour of (a, x) is arbitrary; that of (b, x) must be E, that of (c, x) and (d, x) must be V.

III - NG = 2. (We know NG cannot be 3).

(1) NA = 0. Condition B3 is infringed.

(2) NA = 1.

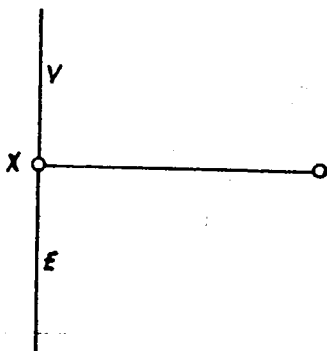
(a) N3 = 1.



N.B. In this figure and in the following, each line segment represents any positive number of edges, all of them with the assigned colour or all of them non-coloured, as the case may be.

In the previous figure, condition B3 is not respected.

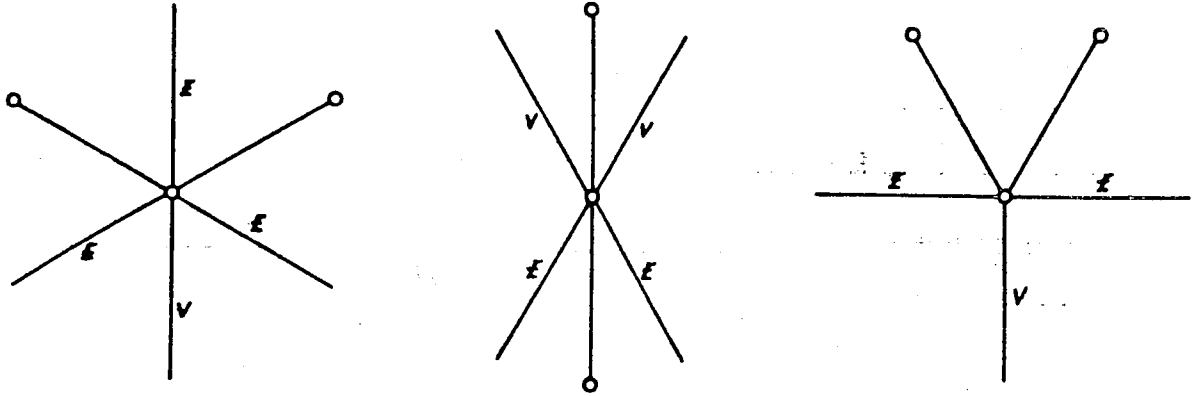
(b) N3 = 0.



The colour of (a, x) is necessarily E.

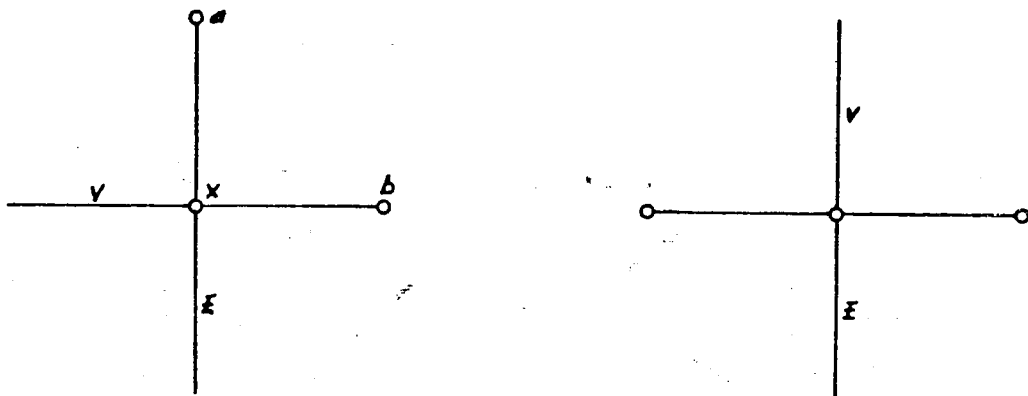
(3) $NA = 2$.

(a) $N3 = 0$. The following situations may occur:



There are no necessary consequences.

(b) $N3 = 1$. The following situation arises:

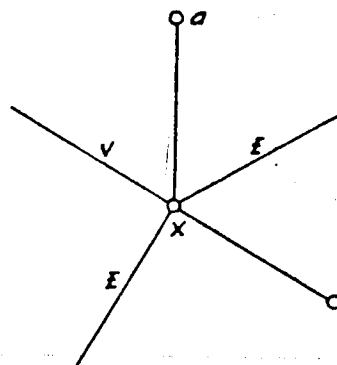


Edge (a, x) must have colour E.

(c) $N3 = 2$.

(i) $NAR = 2$.

(ii) $NAR = 1$.



(a, x) with E and

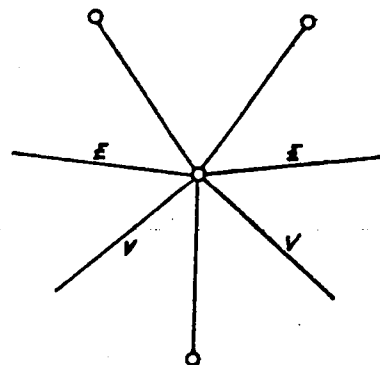
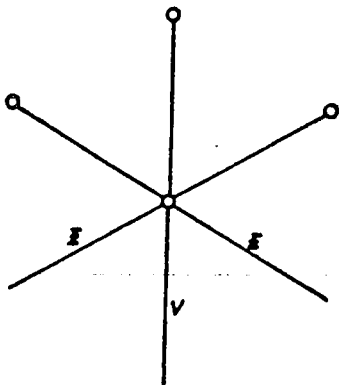
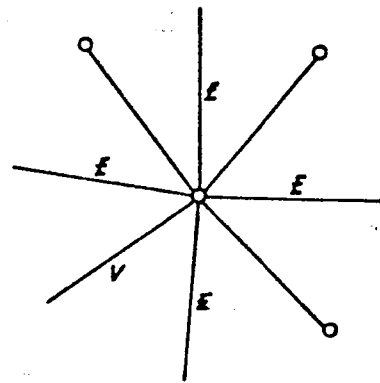
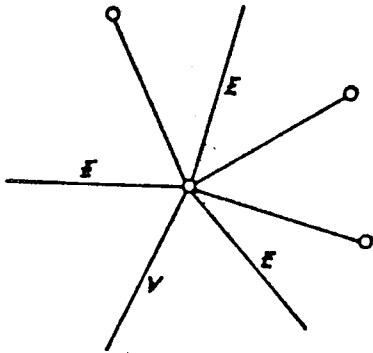
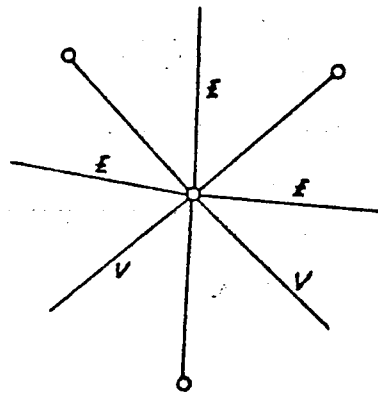
(b, x) with V.

Any colouring infringes
condition B3.

(4) $NA = 3$.

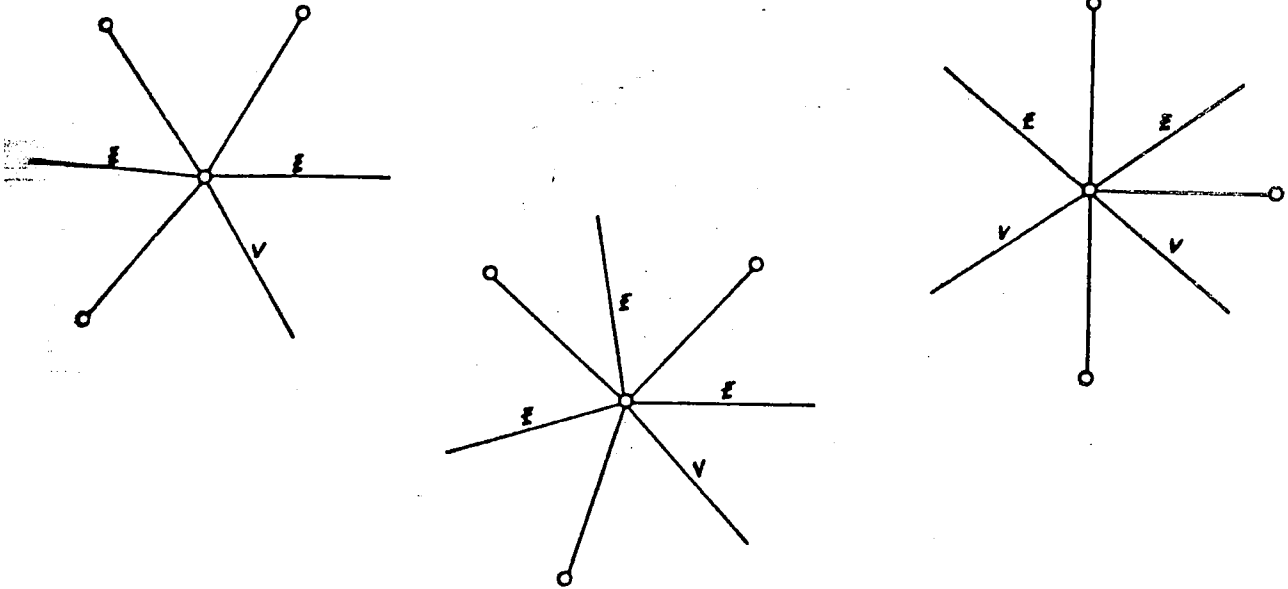
(a) $N2 = 0$.

The following situations occur:



In every case, there is no colour consequence.

(b) $N3 = 1$.

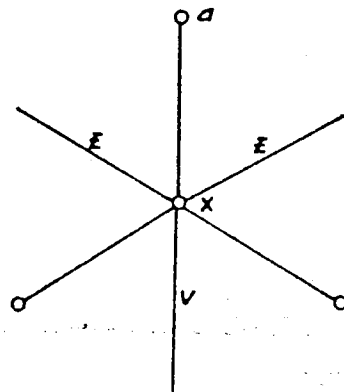


Here there is also no necessary colouring.

(c) $N3 = 2$.

(i) $NAR = 1$.

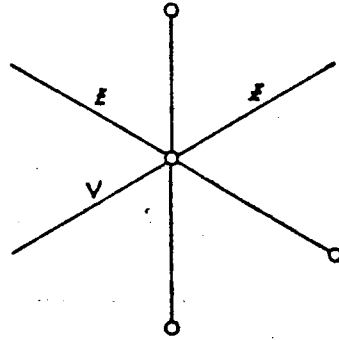
The only case is this one:



Edge (a, x) is necessarily coloured with V.

(ii) $NAR = 2$.

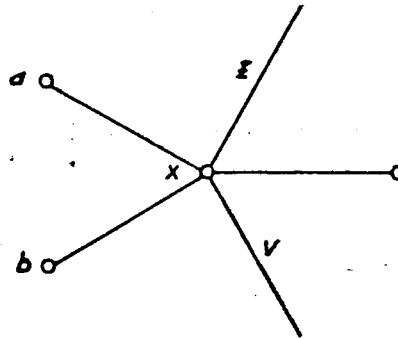
There is just one case.



There are no colour consequences.

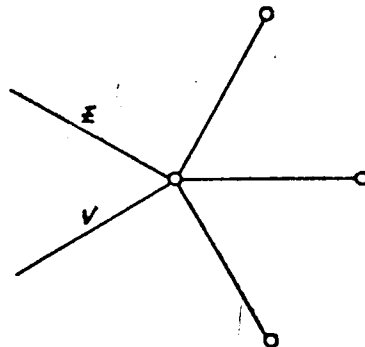
(d) $N3 = 3$.

(i) $NAR = 2$.



(a, x) must be V , and (b, x) must be E .

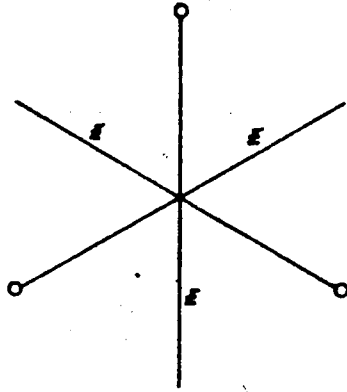
(ii) $NAR = 3$.



Condition B3 is not met.

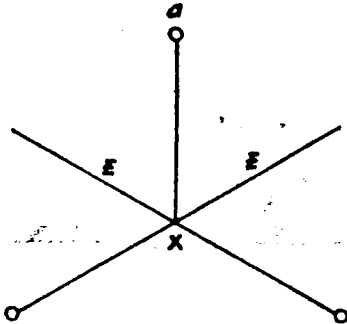
(4) $NA = 3$

(i) $NAR = 1$.



There are no consequences.

(ii) $NAR = 2$.



There are no consequences.

(5) $NA = 4$.

By joining, in all conceivable ways, a non-coloured edge to each of the cases with $NA = 3$, it is easily verified that none of them leads to colour consequences. By this reasoning, the same is also easily checked for NA greater than four.

to page 97

IV - NG = 1.

(1) $NA = 0.$ B3 is not respected.

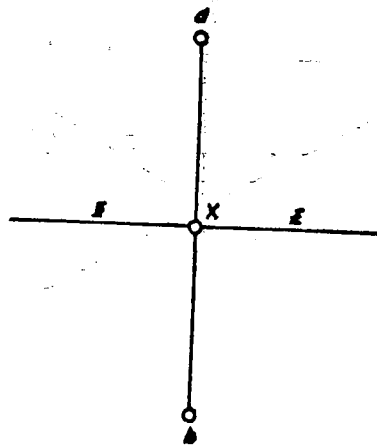
(2) $NA = 1.$



B3 is infringed.

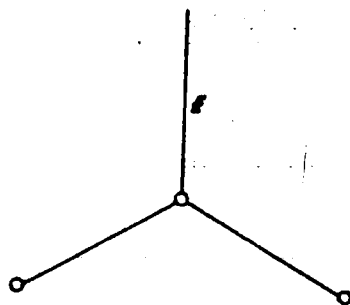
(3) $NA = 2.$

(i) $NAR = 1.$



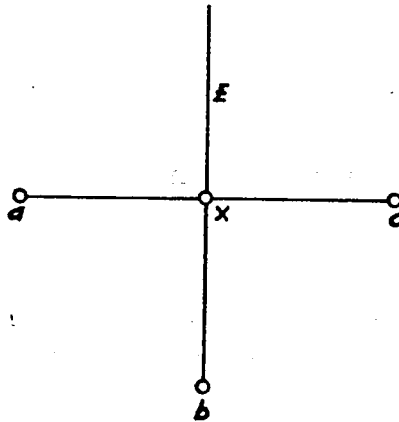
Edges (a, x) and (b, x) must be coloured V.

(ii) $NAR = 2.$



Edges (a, x) must be coloured V.

(iii) $NAR = 3$.



The colour consequences are: (a, x) and (c, x) with V and (b, x) with E.

→ from page 95

(5) $NA = 4$.

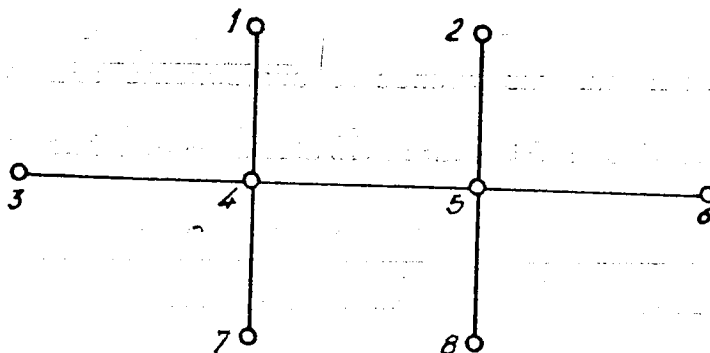
By joining, in all possible ways, a non-coloured edge to the cases where $NA = 3$, it is seen that no consequences follow. By the same reasoning, one verifies that a similar thing happens for the cases where NA is greater than four.

$V - \underline{NG} = 0$. Obviously, no colour consequences follow.

The possible colour situations for a $IN5$ node have now been all examined.

Components of interior nodes with four edges.

Consider the following figure, where nodes 4 and 5 are agreed to be interior.



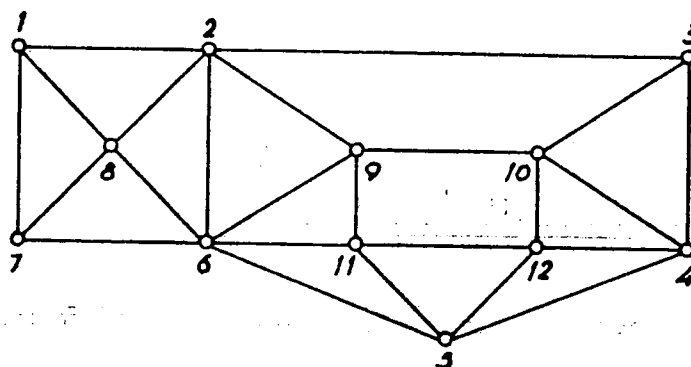
What consequences follow from colouring edge (3, 4), for example, with E? It can easily be verified that condition B3, applied to node 4, imposes the colour E to edge (4, 5). Thus, it makes it possible to apply condition B3 again, this time to node 5, and so to colour all its edges in the only permissible way.

Furthermore, as this example suggests, if any of nodes 1, 2, 3, 6, 7, or 8 were an interior node with four edges, all its edges would also be coloured in an inevitable way, after the assignment of any one colour to edge (3, 4) say.

In chapter three we defined the concepts of "net" and of "component relative to P", where P is a predicate, to deal with this and similar situations. In the examples given there, we used the predicate defined as follows:

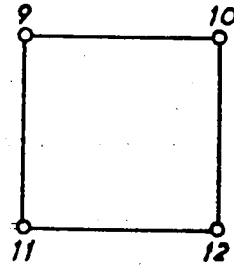
"P is true of node x if and only if x has four edges".

We will now apply it to the following planar realization R:



In this realization, the connected components relative to P define a sub-realization, called a R4 sub-realization, which has two components:

8



From condition B3, the following property is easily derived. The examples given will have convinced the reader of its validity. Furthermore, according to a previous definition, it is a colour property.

Property R4: Let R_4 be the sub-realization of a planar realization R , determined by the components of R relative to predicate P stated above. Let C be any one such component, and N its corresponding net relative to P . In these circumstances, the assignment of a colour to an edge of any node of C has, as a colour consequence, a unique assignment of colour to all the edges of N , in the case no colour conditions are violated; otherwise, realization R cannot be coloured (i.e., R has no HV - assignment conforming to all the colour conditions).

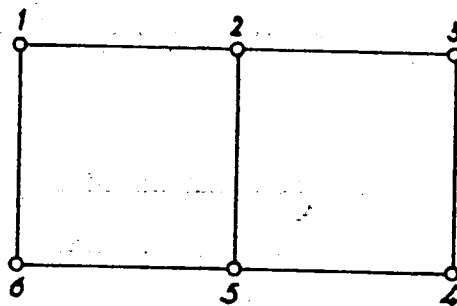
This property explains why we made the initial distinction, among interior nodes, of those having four and those having more than four edges.

Components of quadrangular faces

This section is similar to the previous one, because in it we will consider components of nodes with four edges, but in the dual R' of a planar realization R . To each such node of R' there corresponds in R a

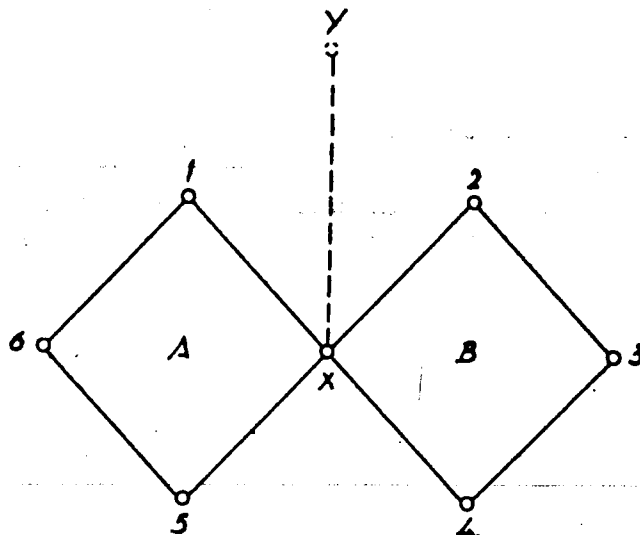
quadrangular face. Thus we shall be dealing in fact with connected sub-graphs of G in R , made up entirely of quadrangular faces. Furthermore, each face of any one such subgraph must have an edge in common with at least one other face of the same subgraph, since the two faces will correspond to two connected nodes of one same connected component in R' of nodes with four edges.

Let us start with an example. Consider the following figure:



Bearing in mind colouring rule B2, it can be easily ascertained that, for example, once edge (1, 6) is coloured, the assignment of colour to all other edges immediately follows.

Consider next the figure below.



Admit that x is an interior node. If edge $(1, 6)$ is coloured, the colour consequence relative to quadrangles determined by colour condition B2, imposes a definite colour to each of the edges of face A. Let us now distinguish between two cases:

(1) Suppose node x did not have any other edges besides those belonging to faces A and B; in particular, that edge (x, y) did not exist. In that case, x would be an interior node with four edges, with edges $(1, x)$ and $(5, x)$ already coloured, and edges $(2, x)$ and $(4, x)$ without any colour yet. The colours of edges $(1, x)$ and $(5, x)$, which were given by the colouring of face A, do not however go against condition B3. The colour consequence for this node is the assignment of colour to edges $(2, x)$ and $(4, x)$, which, in turn, imposes an inevitable colouring of face B. Thus, in this case, the colour consequences from face A to face B, are mediated by node x .

(2) Suppose edge (x, y) does exist, and that it is not coloured. After face A is coloured, x is a IN5 node with following characterization:

$$NG = 2$$

$$NA = 3$$

$$N3 = 3$$

$$NAR=3.$$

If the reader consults the section on the colouring of IN5 nodes, he will verify that for the case in question there follow no colour consequences for the edges of x . Thus, in this case, the colouring of A is prevented by node x from transmitting colour consequences to face B.

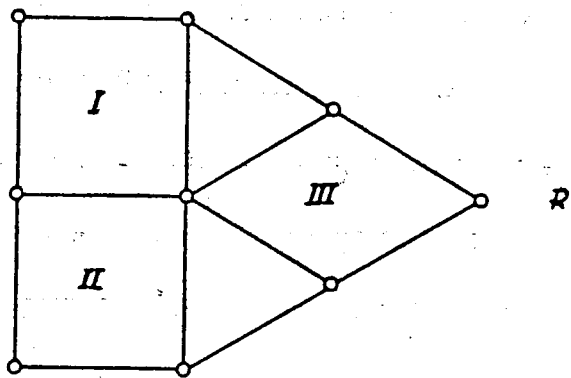
These two examples show that even if two quadrangular faces have a node in common, the colouring of one does not necessarily give consequen-

ces for the colouring of the other, unless they have also an edge in common.

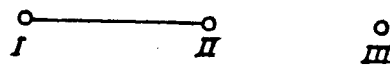
Now given a planar realization R of G , with at least one quadrangular face, define the subgraph Q_R of the dual G' of G relative to R as follows:

Q_R is the union of the components of G' relative to the property P' "node x' has four edges", which amounts to say "the face of R corresponding to x' is a quadrangle".

Example. Consider the following realization R :



The nodes of Q_R are I, II, and III; it has only one edge: (I, II). Thus, Q_R may be represented as:



It may be observed that Q_R has two connected components. These correspond to two subgraphs of G in R : the one made up of face I and II, and the one consisting in face III alone. The two components of Q_R are in fact the connected components in G' relative to property P' .

We shall now state a colour property relative to components of quadrangles, whose proof we omit, since its validity can be readily recognized after the previous examples.

Property Q_R : Let R be a planar realization, and C a connected component of Q_R . The assignment of colour to any edge of a face corresponding to a node of C has, as a colour consequence, a unique assignment of colour to all the edges of the faces corresponding to the nodes of C , in the case no colour conditions are violated; otherwise, realization R cannot be coloured (i.e., R has no PHV - assignment).

Now, to each component C of Q_R corresponds a subgraph C' of G whose nodes and edges are the nodes and edges of the quadrangular polygons dual to the nodes of C ; such a subgraph C' is called a component of quadrangles. Note that it is possible to have distinct components of quadrangles with nodes in common. For example, in the realization of the preceding figures, the component defined by quadrangles I and II has a node (the only interior node of the realization) in common with the component defined by quadrangle III.

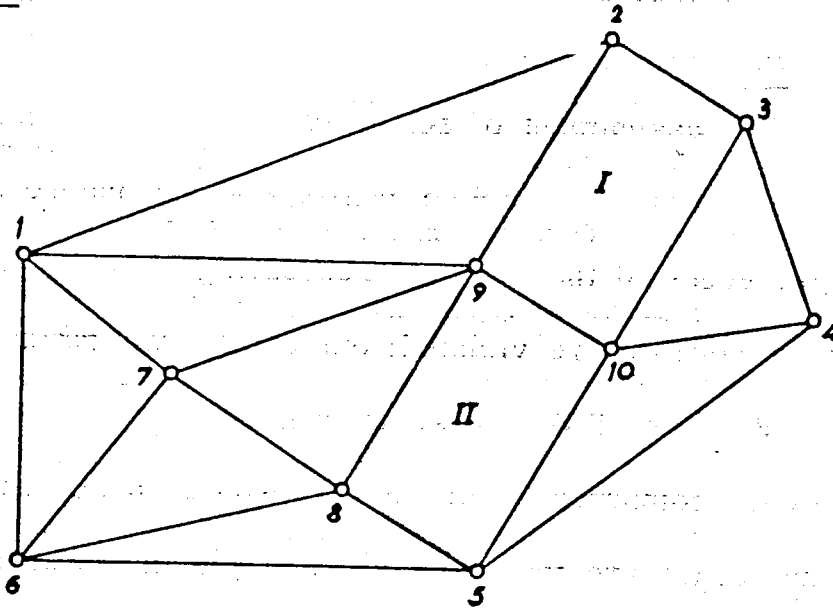
In the next section we will explore the fact that the two types of components, components of interior nodes with four edges and components of quadrangular faces, may have common nodes and/or edges, making it possible, in certain circumstances, to speak of the union of components of different type, to form what we call a "general component" or simply "G-component".

General Components.

Let us denote "an interior node with four edges" by IN4.

We shall say that a component C of quadrangles is linked to a component D of IN4s, if there exists at least one IN4 in D belonging to at least one quadrangle of C . In that case, the union of D with the net of C is called their G - component.

Example.



In this planar realization there exist two components of IN4s. Component D_1 has 7 and 8 as nodes, and component D_2 just node 10. There is only one component C of quadrangles, made up of quadrangles I and II. Both D_1 and D_2 are linked to C.

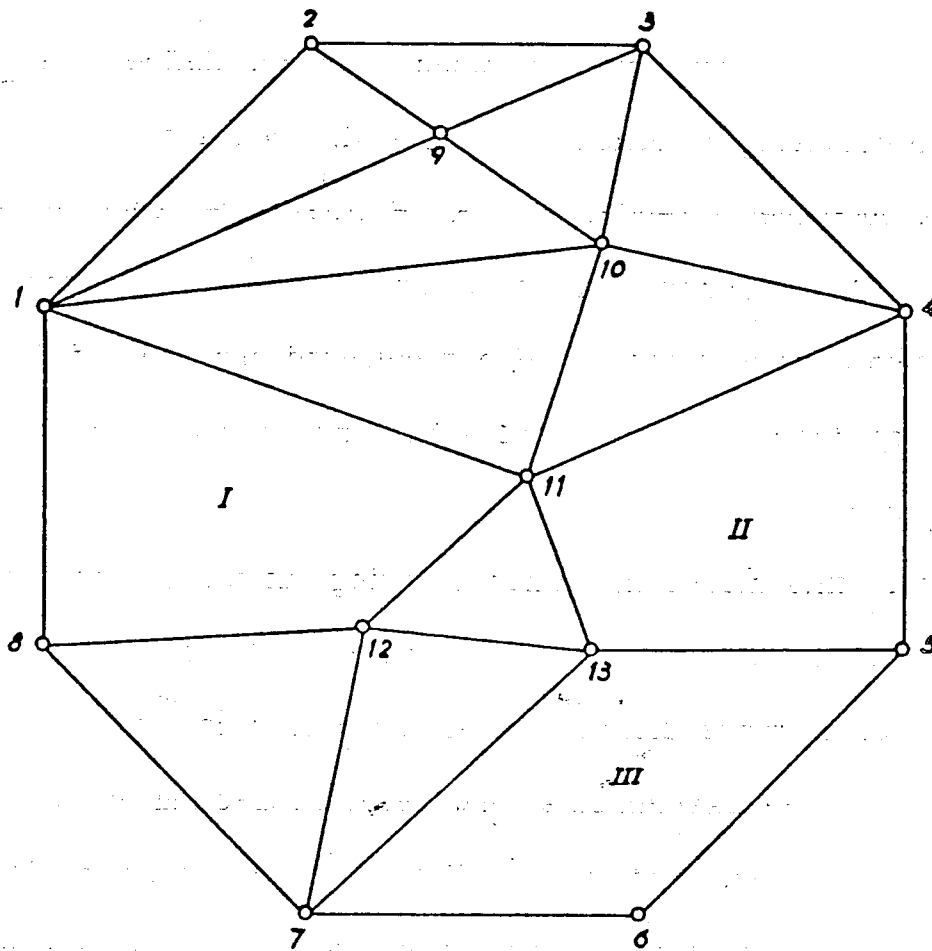
Property G:

Given a planar realization R, when a component C of quadrangles is linked to a component D of IN4s, thus determining a G-component F, the colouring of any one edge belonging to a quadrangle of D or to a node of C, i.e. an edge of F, has, as a colour consequence, the colouring of all the edges of the quadrangles of D and of all the edges of nodes of C, i.e. of all the edges of F, in the case no colour conditions are violated in the process of extracting consequences; otherwise, realization R cannot be coloured (i.e., R has no PHV - assignment).

This colour property will not be proved, since its validity follows easily from the validity of the two previous colour property^{ies}; it

can be seen in the examples given, and in the following one:

Example. Consider this realization:



The components of PI_4 's are:



Graph Q_R may be represented by:



The G - components of a planar realization R of G , provide structural information about the R - layout - schemes of R .

Take the largest G - component of R (i.e. the one with greater number of nodes), and obtain all its colour consequences within R . The coloured subgraph of G in R thereby obtained, when it exists, defines the nucleus of R , in case no colouring conditions are violated. Such a nucleus is invariant in every R -layout-scheme rendered by R . Furthermore, each of the other G -components of R , together with its colour consequences, in the case no colouring conditions are violated, is also a coloured subgraph of G in R , invariant in every R -layout-scheme; the subgraphs obtained are called the satellite nuclei of R .

Of course, this means that when carrying out the colour consequences for obtaining the nucleus or the satellite nuclei of R , if the nucleus and satellite nuclei are pairwise disjoint and if any colour condition is infringed then there are no PHV-assignments whatsoever respecting all the colour conditions, since the nucleus and the satellite nuclei are invariant from assignment to assignment, except perhaps for an exchange of one colour for the other. However, since all colouring conditions are symmetrical with respect to colour, any impermissible colouring situation transforms, by such an exchange, also into an impermissible colouring situation.

We have been considering the case where all the nuclei are disjoint; if that is not the case, it may be possible to ^{reconcile} ~~conciliate~~ two or more nuclei, whose colour assignments are incompatible, by exchanging the colours in only some of them. The process is in reality done pairwise. Everytime the colour consequences of the nucleus extend into some satellite nucleus, if an incompatibility arises between the two (since all the nuclei are coloured independently), an exchange of colours is tried for the satellite nucleus. If it

does not lift the colouring incompatibility, then the two nuclei are incompatible, whatever the way they are coloured. If they are compatible, then a partial subgraph has been permissibly coloured which may extend to some satellite nucleus, etc.. We shall deal with this process in greater detail in subsequent sections.

Of course, G - components (when they exist) reduce the total of assignments (2^{N-1}) to a much smaller number of sensible assignments. Each G-component determines a set of edges acting as a whole from the colouring point of view (i.e. as a colouring subproblem).

The colouring tests.

In chapter three we have already drawn attention to the double perspective from which colouring conditions B should be envisaged. In fact, they are utilized both as colouring rules and as colouring tests. Evaluation of colouring situations is mingled with the computation of consequences according to the colouring rules: i.e. colour consequences are carried out from colouring situations evaluated as legitimate, and their effect is accepted on the basis of the evaluations performed by the same colour conditions which were used for generating the colour consequences; thereby starting another cycle until no more consequences can be derived.

In chapter one, we have already had the occasion to comment upon this type of process. At the moment, we only want to point out that the referred cycle of testing-operating-testing is responsible for the outspread of colour consequences, given only purely local colour conditions on pure local colouring situations. G-components and other colour components are simply large colouring steps which have been built up from the colour con-

ditions' cycles.

Before we expound the colour conditions relative to the contour and its form, which we shall do in the section on contour forms, we will now state the process of colouring a realization so as to obtain the first PHV-assignment obeying colour conditions B, and contour conditions C, be they the already defined ones or newly defined.

Colour operations and colour options.

We shall now describe the colour operations and the colour options to be considered.

(I) The colour operations, which include the colour options, are enounced as follows:

(1) If, in a realization to be coloured, having already some coloured edges or not, there follow no more colour consequences, a colour operation takes place which consists in colouring any one non-coloured edge with an arbitrary colour; this type of colour operation is called a colour option; it will be further detailed in paragraph II.

(2) Otherwise, all colour consequences are carried out, recurrently. These may be:

- (a) to colour the edge of a triangular face;
- (b) to colour edges of a IN5;
- (c) to colour the edges of a G-component, (which, of course, may be made up only of quadrangular faces or only of IN4s).

(3) If at any stage of the process of colouring the realization, there arises an infraction of conditions B or conditions C, proceed as follows:

(a) detect the edge coloured when operation (1) was last applied; if no such edge exists stop: the realization given cannot be coloured permissibly;

(b) remove the colour from all edges coloured subsequently to the edge found in (a);

(c) colour the edge determined in (a) with the other colour; if this has already been done before for that edge, stop: the realization given cannot be coloured permissibly.

The colour options arise, as we have seen, in those colouring situations where colour operation (1) is applicable, and their purpose is to make effective and efficient the choice of the edge to be coloured as well as its colour, to be made in an arbitrary way from the set of candidate edges; i.e. they define the candidate set in such a way that the number of colour consequences is probably large.

The existence of such colouring situations explains how a given realization may give rise, by mediacy of the colouring process being defined, to various colouring solutions (i.e. PHV-assignments satisfying particular contour conditions, if any).

In fact, when an edge is chosen in (1), the colour to be given it is, in principle, arbitrary; which means that there may be (at least) two solutions that are different, if only on the colour assigned to that edge. The reason why we say "in principle", is that either one or both of the colours assigned to the edge may produce unpermitted colour situations while colour operation (2) is carried out.

II) The colour options we consider are described as follows.

(1) If there are no coloured edges, two cases are distinguished:

(a) There exists at least one G-component. In this case, any G-component is chosen which has the greatest number of nodes, and one of its edges is arbitrarily coloured with E. (We remark that all G-components may be coloured at this stage, which is more efficient. Then, at step (2) (c) of I) their colouring is either maintained, or the E_S and V_S are interchanged throughout the component).

(b) If there are no G-components, i.e. if there are no quadrangles or $IN4_S$, two cases are distinguished:

(i) If there exist interior nodes, one arbitrary edge, belonging to any of such nodes having the minimum number of edges, is coloured with E.

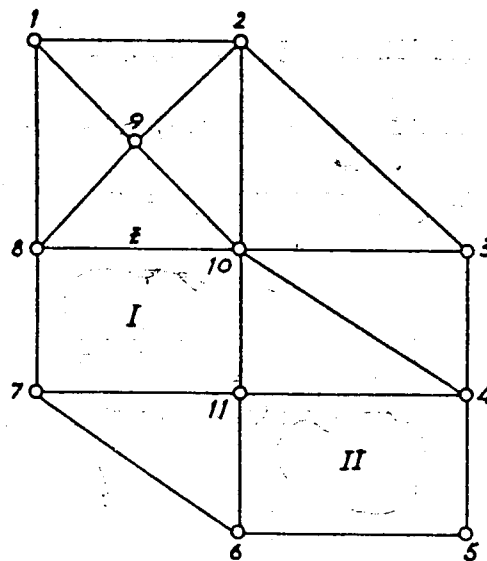
(ii) Otherwise, colour with E any arbitrarily chosen edge.

We remark that, since the edge to be coloured and its colour are in principle arbitrary, the colour options put forward are profiting from such arbitrariness by defining a candidate set of edges in each of the cases also defined at will. Of course, other cases and other candidate sets could have been defined. The reason for our particular choice of cases and sets is heuristic: it seems to us that they are the more fruitful ones, from the point of view that a greater probability exists, over all possible realizations, of a large number of edges being coloured through the colour consequences.

The cases considered cover the totality of situations that may occur

in a non-coloured realization. It is clear then that, if there are G-components, the first colour operation determines immediately the colouring of all the edges of the nucleus, without further application of colour operation (1); otherwise, to the set of edges formed by the first edge chosen and by all those edges belonging to the ensuing permitted colour consequences we also call nucleus of the realization, or simply nucleus. The nucleus is thus a permissible colouring situation common to all possible assignment solutions. Its determination is always carried out at the start of the colouring process, for the sake of efficiency.

To clarify these points we present an example. Consider this realization:

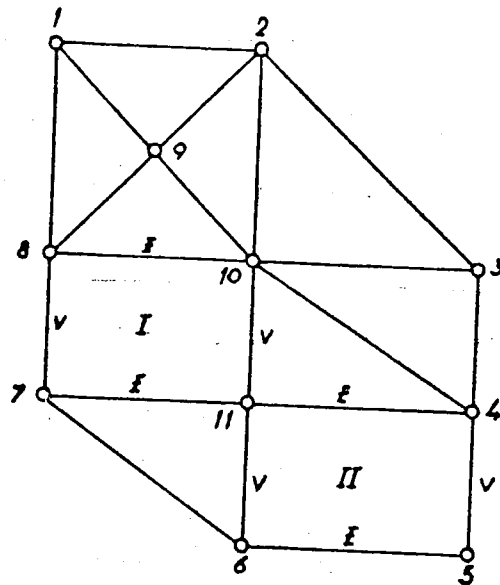


Its G-components are two. One is made up of quadrangular faces I and II, and node 11. The other, just of node 9.

Applying option (1) (a), we start by choosing the G-component with the greater number of nodes and, arbitrarily, edge (8, 10) belonging to it, which is then coloured with E.

The assignment of colour to this edge originates colour consequences that, according to property G, extend to all the edges of the component. Co-

our operation (2) (c) stipulates just that. Upon applying it one obtains the following colour situation:



Since there are no more colour consequences, the nucleus is constituted by edges (4, 5), (4, 11), (6, 5), (6, 11), (7, 8), (7, 11), (10, 8), and (10, 11), since this colouring situation presents no contradictions (i.e. one same edge having to have two colours by virtue of different colour conditions).

We now proceed to colour the remaining edges, but first we must continue to describe the colour options, for the case of a realization with already some coloured edges.

(2) Two cases will be distinguished:

(a) If there are no non-coloured interior edges, an exterior edge is arbitrarily chosen and the colour E is assigned to it. Preferably, the edges chosen in this case should be in succession around the boundary. We recall that an edge is interior if it has at least one interior node as an extremity and exterior otherwise.

(b) In case there exists at least one non-coloured interior edge, the colour E is given to any one such edge belonging to an interior node having

the minimum number of non-coloured edges, and preferably with already some coloured edges.

In the example that follows, showing how to obtain a colouring solution for a realization obeying conditions A, we will consider that the contour must be rectangular. We have previously stated conditions C0 and C2 pertaining to that case, although their proof will only be made in the section on contour forms.

However, we still need one more result, applicable to the case of rectangular contours. This result is an equivalent formulation of both conditions C2 and C0. Its usefulness derives from the fact that it translates the global rectangular constraint into a local constraint on every exterior node.

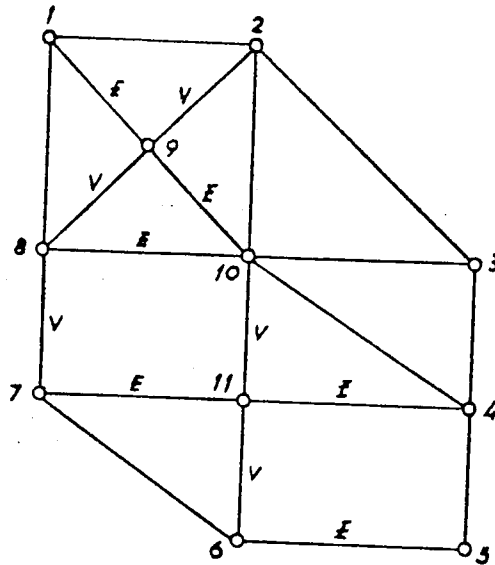
First recall that NG is the number of coloured groups of edges around any one node. Let the node in question be denoted by x ; we shall write $NG(x)$ to denote the number of groups of x .

Condition C0-2 may now be expressed through the following theorem:
C0-2 - It is a necessary and sufficient condition for a coloured realization obeying conditions A and B to have a rectangular contour form, that $NG(x) < 4$ for every exterior node and that condition C1 holds.

Example. Let us use the previous example and continue to colour it, from the point where the nucleus had been coloured.

Given that colouring situation, colour operation (2) (b) of II) is applicable. As a result, edge (9, 10) may be coloured with E. Since the edge belongs to a IN_4 , colour operation (2) (c) of I) is applicable to the component made up of node 9, its edges, and nodes 1, 2, 8, and 10. This

figure is obtained:



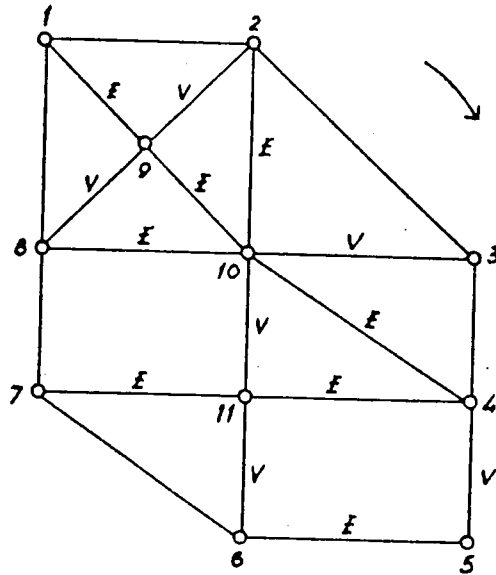
Note that, alternatively, we could have decided to apply colour operation (2) (a) of I, from which would result the colouring of edge (8, 9) with V. But that would not avoid the subsequent application of colour operation (2) (c) of I, so as to colour edges (1, 9) and (2, 9). This suggests that the order of application of the colour operations may profit from not being arbitrary.

We shall respect the rule that gives priority going from operation (2) (c) to operation (2) (a) of I.

N.B. The final result of a succession of colour operations, is independent of the particular sequence observed, as a little reflexion will convince the reader. Some sequences, however, are more efficient, in that some of its operations may become superfluous.

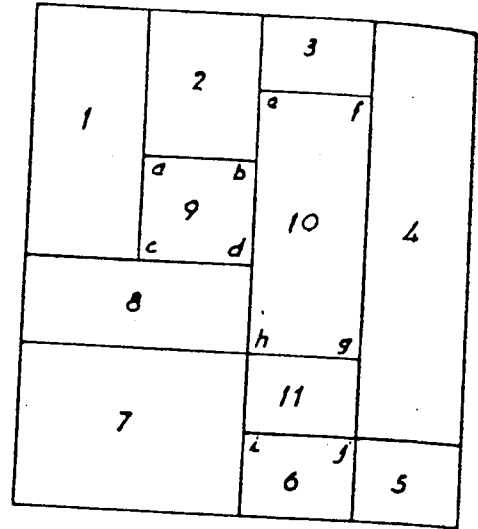
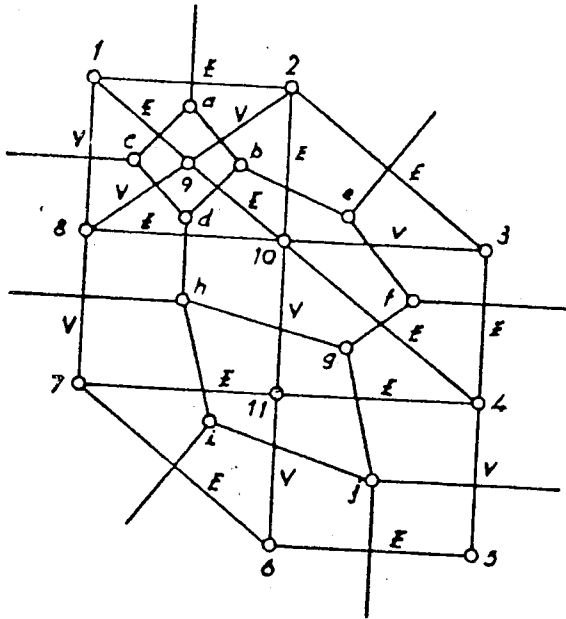
The figure shown last, still has some non-coloured interior edges: let us this time choose edge (2, 10) and give it colour E. We can now apply colour operation (2) (b) of I to IN5 10. Indeed, node 10 is characterized by $NG = 2$, $NA = 2$, $N3 = 2$, $NAR = 2$. Upon consulting the sec-

tion on the colour consequences for a IN5, it is easily seen that the following colouring situation arises:

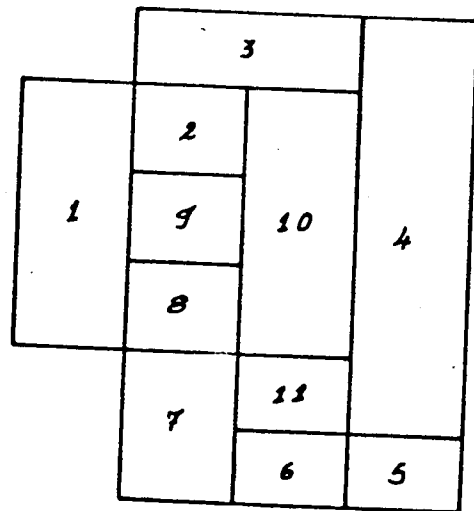
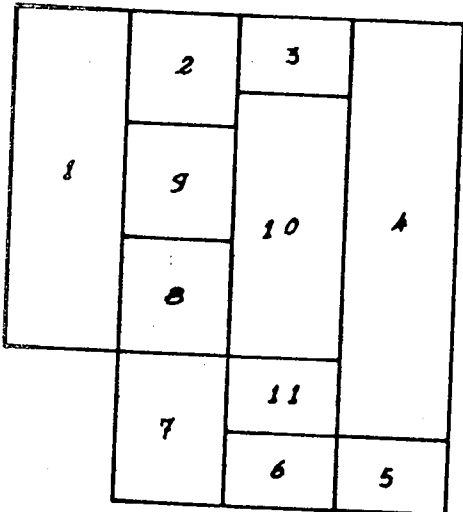


At this stage, only exterior edges remain to be coloured. Consequently, the only operation applicable is colour option (2) (a). Since, according to that option, we should choose the exterior edges in succession around the boundary, we shall agree to choose the edges in the sense indicated by the arrow in the previous figure. The first edge chosen is (1, 2), and its colour E. In this example we conclude that we may colour edges (2, 3), (3, 4), and (6, 7), successively with E, since the only colour conditions in case, conditions B1 and C0-2, are complied with. However, upon applying option (2) (a) to edge (8, 1), colouring it with E, condition C0-2 is infringed since $NG(8) = 4$. Next, to know if operation (3) of I) is applicable, we must detect which edge was most recently coloured by operation (1) of I) (in this case through option (2) (a)). Colour operation (3) (a) of I) gives us precisely edge (8, 1). Since no other edge was coloured after (8, 1), there is no need for carrying out operation (3) (b) of I). Finally, according to (3) (c) of I), edge (8, 1) is given the colour V. With this change in colour, the infraction of condition C0-2 no longer persists, and the R-layout-scheme (with rectangular contour) shown, is obtained by mediacy of the dual of the realiza-

tion.



If, however, we had not put the rectangular constraint on the contour, but only general conditions C0 and C1, we could have obtained the following two figures.



The one on the left differs from the first solution presented in that, according to the contour conditions now being considered, $NG(8) = 4$ is permissible. Thus, the only difference between this and the previous solution is the colour of edge (8, 1).

The one on the left differs from the first solution presented in that according to the contour conditions now being considered, $NG(8) = 4$ is permissible. Thus, the only difference between this and the previous solution is the colour of edge (8, 1).

The layout on the right differs from the one on the left only in the colour of (1, 2): in this third layout $NG(2) = 4$.

The colouring process.

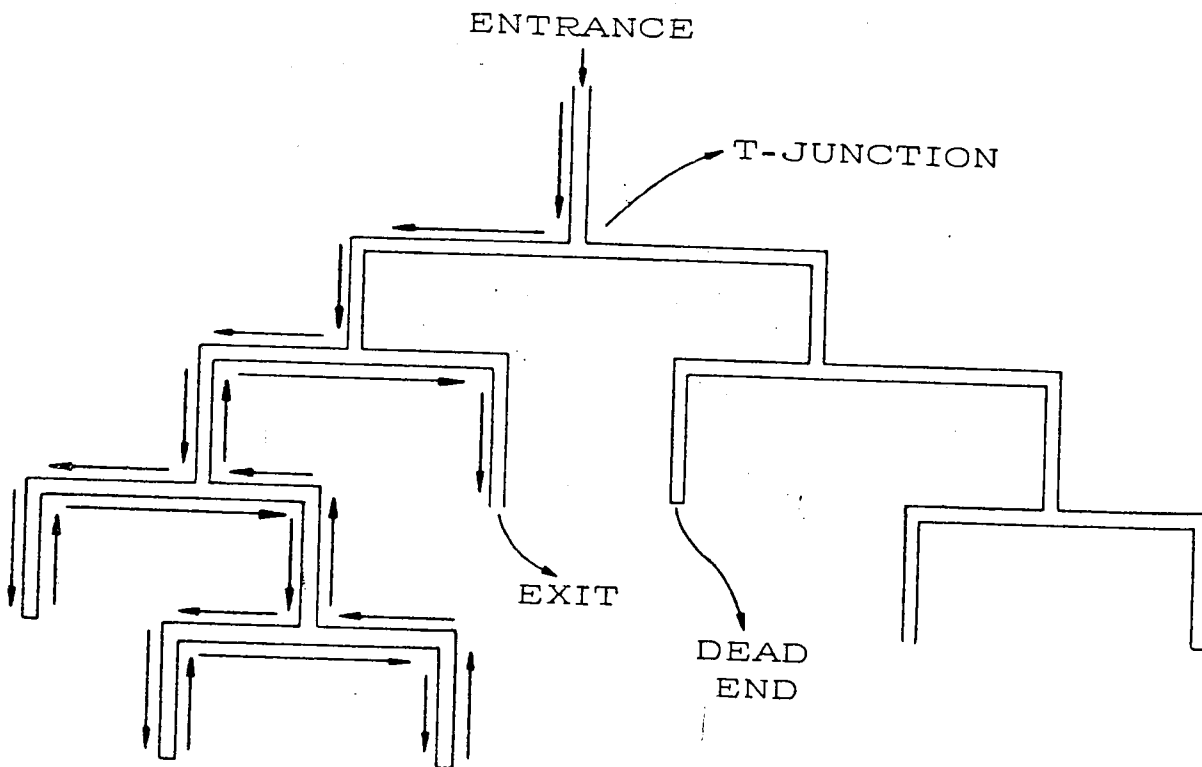
The example put forward in the previous section gives already a first idea of the process of colouring a realization satisfying conditions A. Although the example given constitutes a particular case (it has G-components, for example), we may easily abstract from it the essentials of the general process for obtaining a first solution. After all that has been said, we think the steps enounced below are self-explanatory.

- I - An edge of the realization is chosen, and E is assigned to it;
- II - Colour consequences are extracted (if any);
- III - In case contradictions arise, the realization cannot be completely and permissibly coloured, and the process stops; otherwise, the process continues with step IV;
- IV - If there are still non-coloured edges, one such edge is chosen and E is assigned to it; otherwise, the process stops and a first solution has been obtained;
- V - Colour consequences are extracted (if any);
- VI - If there are no contradictions the process returns to IV; otherwise, it proceeds to VII;

VII - The most recent edge coloured in IV is searched for. If no edge is found, the process terminates with the indication of "no solution". Otherwise, the colour is removed from every edge coloured after the edge detected; that edge is coloured with V, and the process continues with step five.

This process originates a simple backtracking mechanism to which we will devote some consideration, inasmuch its discussion will prove useful when we consider the problem of obtaining all PHV-assignments of a realization, subject to some contour condition.

Let us then consider a class of mazes called trees, a specimen of which is depicted in the following figure:



Informally this type of maze has an only entrance, located at the root and one or several exits, corresponding to some of the terminal nodes; to the other terminal nodes there correspond dead ends. At each T-junction one may choose between two branches, the one to the left and the one to the right, say. Furthermore, there are no cycles (i.e. if one always goes forward, one never passes the same point twice).

A systematic method for finding a way out consists in:

- (1) At any T-junction choose always the branch to the right, for example;
- (2) Whenever a dead end is reached, get back to the last T-junction whose left branch has not been followed, and take that branch.

It is easily ascertained that if there is an exit, it will most surely be found through this algorithm.

Of course, if some additional information is available, for example about the distribution of dead ends in the tree, one should try to profit from it. Suppose that we knew the tree to be symmetric; then we would need at most to explore half of it.

Let us now establish an analogy between the search in this type of tree, and the search of colour solutions for a realization which colours each edge at the time:

- (a) At its stage, the choice of a colour for an edge corresponds to choosing between the two branches of a T-junction, which is added to the tree when the edge is chosen. We may always choose colour E first, by analogy with step (1) of the tree searching algorithm;

(b) Whenever an infraction of a colour condition occurs, we search for the last edge coloured with E, colour it with V, and "uncolour" every edge coloured after it. This is similar to step (2) of the algorithm, where to "uncolour" an edge means to backtrack a T-junction. If no such edge can be found, the realization cannot be coloured; this is analogous with not finding any T-junction in step (2), which means that the maze has no exit.

Of course, the "colouring tree" of any realization is always symmetric since all colour conditions are symmetric with respect to the two colours. Consequently, if the edge determined in (b) is the first edge coloured the process should stop there; that is why we distinguished steps I and IV in the colouring process.

Additional information to be taken into account is that which refers to colour consequences. If a colour consequence is unpermissible that means that the T-junctions added by the edges belonging to that colour consequence would only lead to dead ends; thus, all those T-junctions should be ignored after the appropriate backtracking is made to the T-junction corresponding to the most recent edge coloured with E.

If, however, a colour consequence with \underline{n} edges is permissible, this means that in the corresponding T-junctions we always take only one of its branches, and so that \underline{n} branches may be ignored (since the edges of a colour consequence are always coloured as a group; i.e. they correspond to the construction of a definite path).

This way, every time an edge is coloured (by means of I, IV or VII), one should find out if there are any colour consequences, and, in the affirmative case, carry out such consequences; this justifies steps II and V, as well as the reference made to V in IV and in VII. The reason for III is the

fact the tree is symmetric. This fact justifies that hereafter we use "colouring tree" for referring to one of its symmetric parts only; namely to the one that corresponds to always choosing the colour E for the edge coloured first.

Now, the obtention of all PHV-assignments (subject to contour conditions) for a realization will consist in exploring the colouring tree so as to find all its exits (i.e. non-contradictory terminal nodes). This can be accomplished by the repeated application, in a controlled manner, of the colouring process already described; to do so, we will modify that process by substituting steps IV' and VII', described below, for steps IV and VII, respectively.

IV' - If there are still non-coloured edges, one such edge is chosen, it is assigned to it, and the process continues with step five; otherwise, a colour solution has been found and one proceeds to VII'.

VII' - Search for the most recent edge coloured in IV'. If no edge is found, two cases are distinguished:

(i) if one or more solutions have already been found then stop: there are no more;

(ii) if no solution has been found yet, stop: there are no solutions.

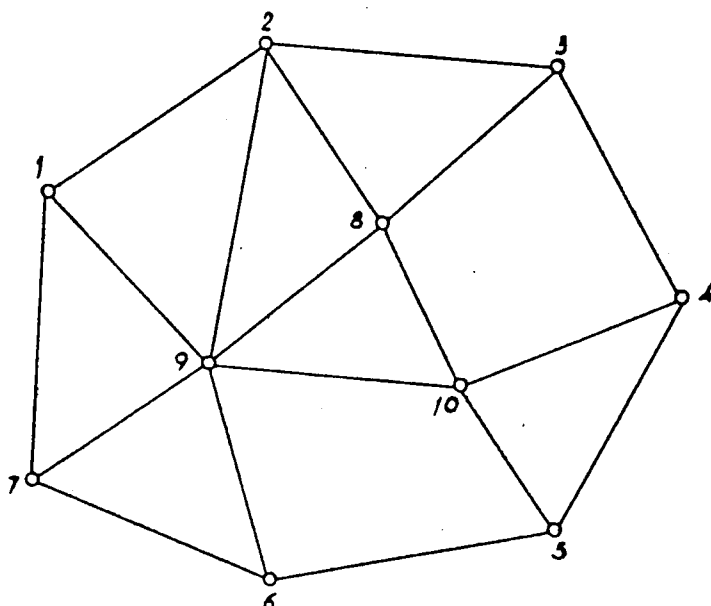
If, on the contrary, an edge is found, then the colour is removed from every edge coloured after that edge; the edge is coloured with V, and the

process continues with step five.

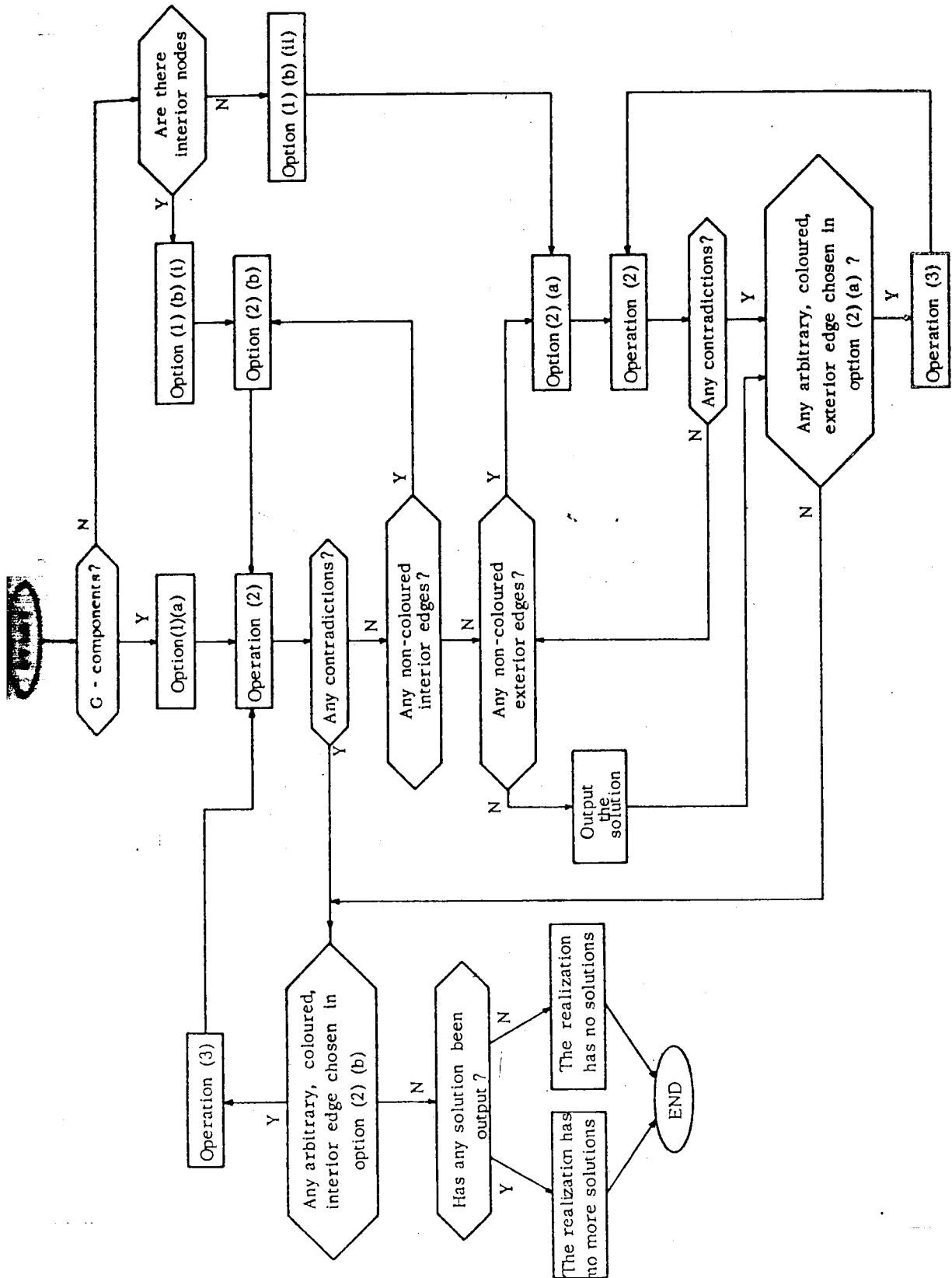
Note that in all steps any reference made to IV or VII is substituted by a corresponding reference to IV' or VII'.

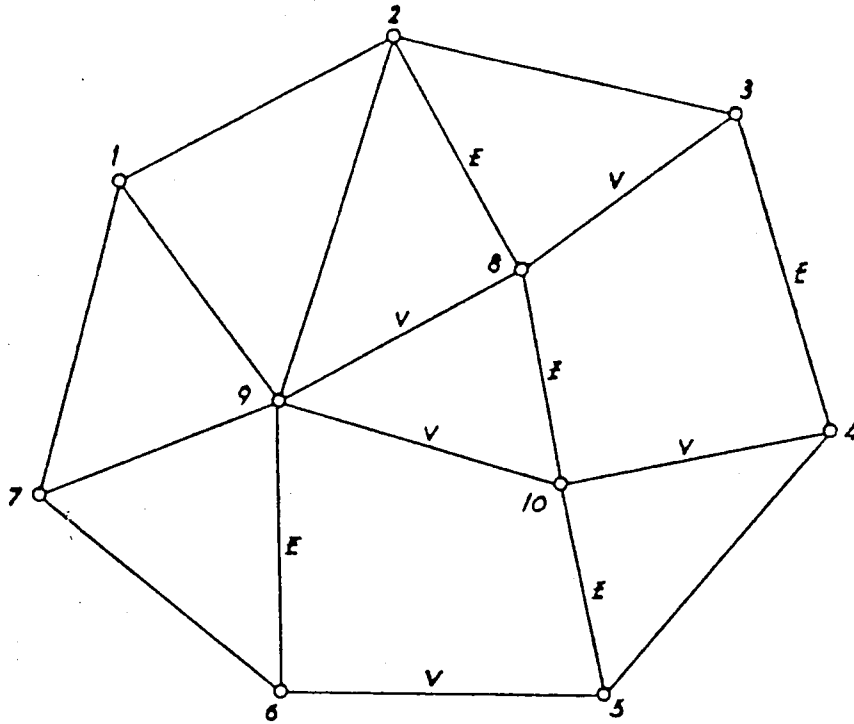
We will next express the whole colouring process in terms of colour operations and colour tests. It will be expressed in the form of a Blocks Diagram in next page. We think no further explications need be given; an example will however be presented going through the various stages.

Example . Consider the problem of obtaining all PHV-assignments of the realization shown, but subject to a rectangular contour. There are four such assignments.

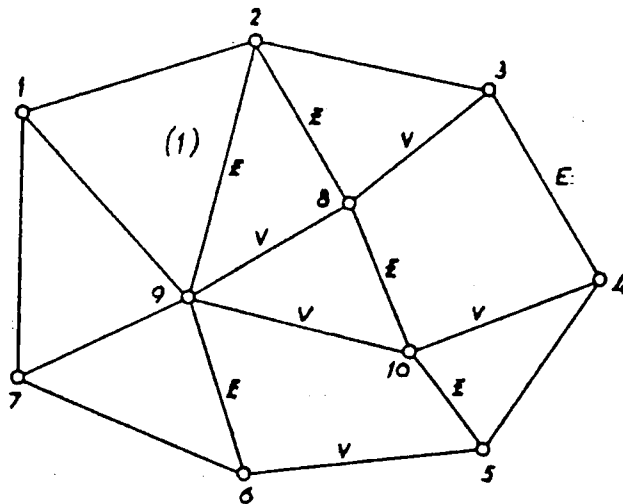


To apply option (1), we start by noting that there is only one G-component. Using option (1) (a), we may choose as initial edge (2, 8). There results the nucleus shown below:



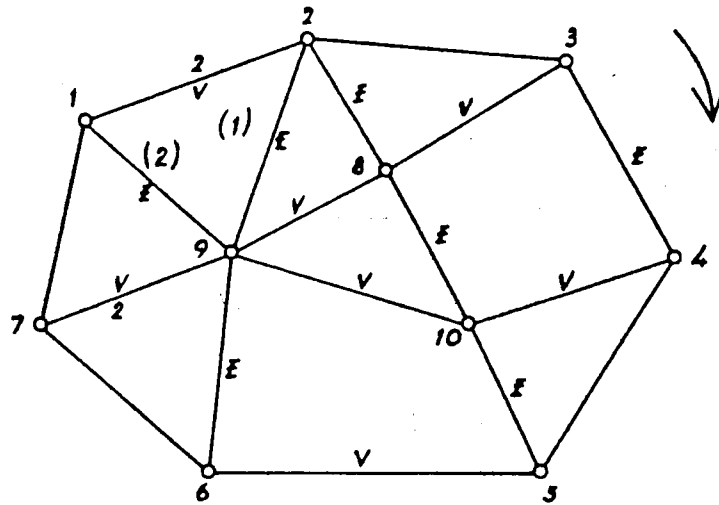


Since there are still some non-coloured interior edges, we next make option (2) (b) and choose edge (2, 9):



In this figure edge (2, 9) is signalled with "(1)". By that we mean to express that it is the first edge from option (2) (b). If edge (2, 9) had originated colour consequences, we would have identified the consequent edges with "(1)". We may now repeat the application of option (2) (b), and choose

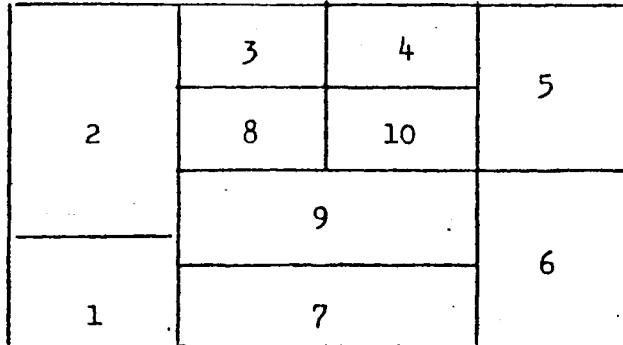
edge (1, 9). The following figure obtains:



In this figure, the colouring of (7, 9) was executed through operation (2) (b) of I, and that of (1, 2) through (2) (a) of I.

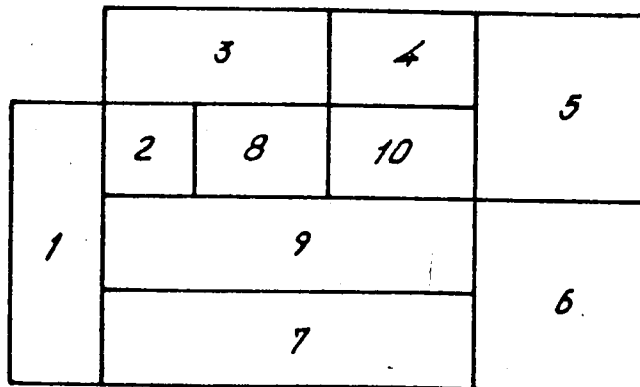
There now remain only non-coloured exterior edges. Applying in succession option (2) (a), we get the figure shown next.

The corresponding R-layout-scheme is:



If we did not impose a rectangular contour, only node 2 could give rise to a concave corner since it alone has at least four edges, among the exterior nodes. Let us examine how the previous assignment could be transformed regarding node 2, so as to make $NG(2) = 4$. Since the colour of (1, 8) is fixed because it belongs to the nucleus, there is actually only one way to alter the colours of the edges of node two, which is to colour edges (2, 3) and (2, 9) with V, and the others with E.

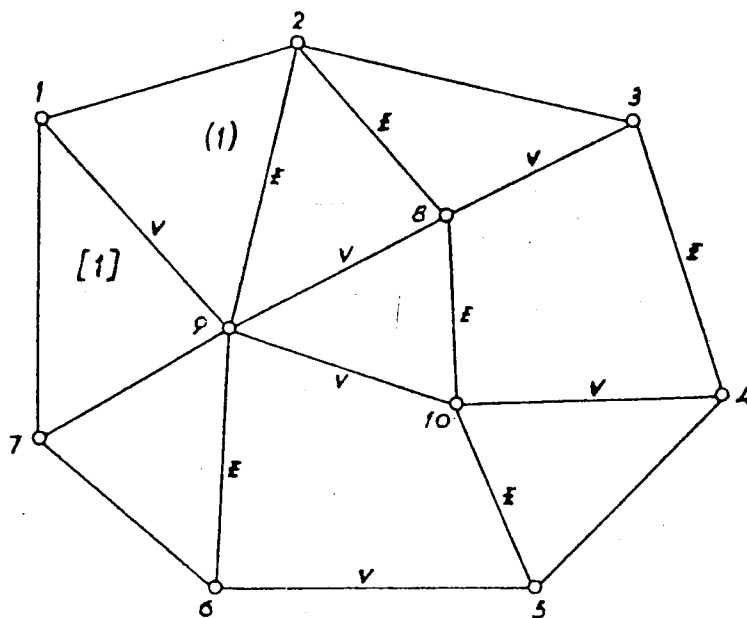
That modification originates the following R-layout-scheme, not having a rectangular contour:



To obtain the next solution with rectangular contour, we apply operation (3) of I. (3) (1) of I stipulates that one should look for the edge most recently coloured in operation (1) of II. That edge is (1,7); since no other edge was coloured after it, there results no consequence from applying (3) (b) of I; applying (3) (c) of I makes us colour edge (1,7) with V. Since a solution is obtained, we could return to operation (3) to search for the next one, edge (6,7) resulting from operation (3) (a) of I, etc..

In this example, however, we shall use a modified version of (3)(a) of I, which stipulates that we shall search for the last interior edge coloured as a result of (1) of I; i.e. the backtracking goes back to an interior edge because we expect a greater difference to arise between solutions differing at least in the colour of an interior edge than between solutions differing only in the colour of exterior edges. This will, of course, reduce the number of solutions, but will permit us to obtain faster the "more different" ones from a topological point of view.

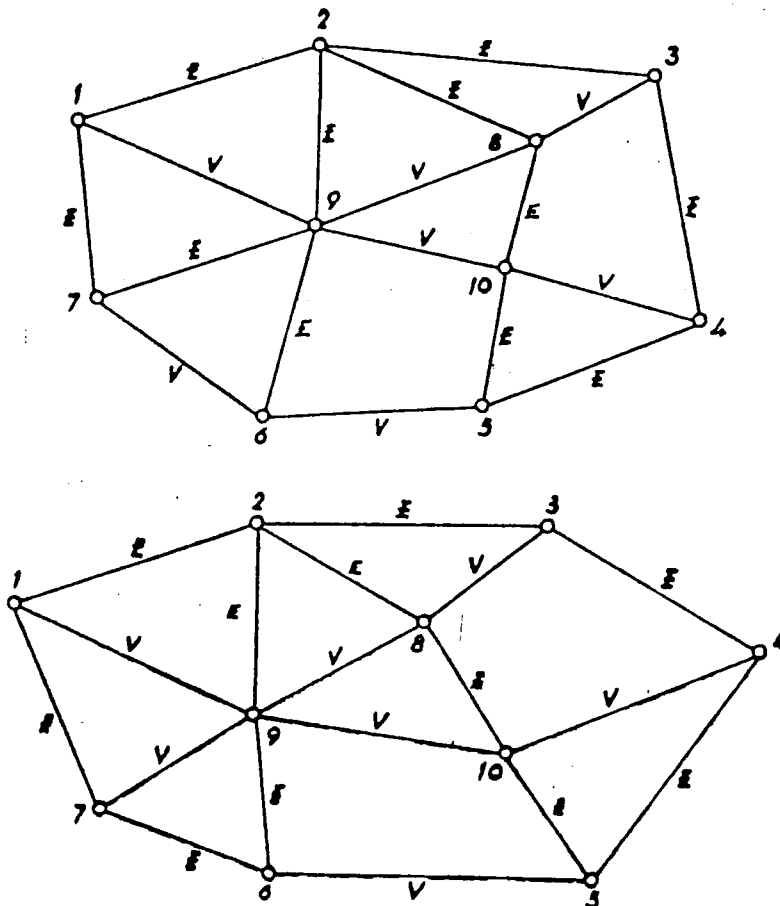
The interior edge detected by the new operation (3) (a) of I is (1,9). After applying (3) (b) and (3)(c) of I we obtain the next colour solution:

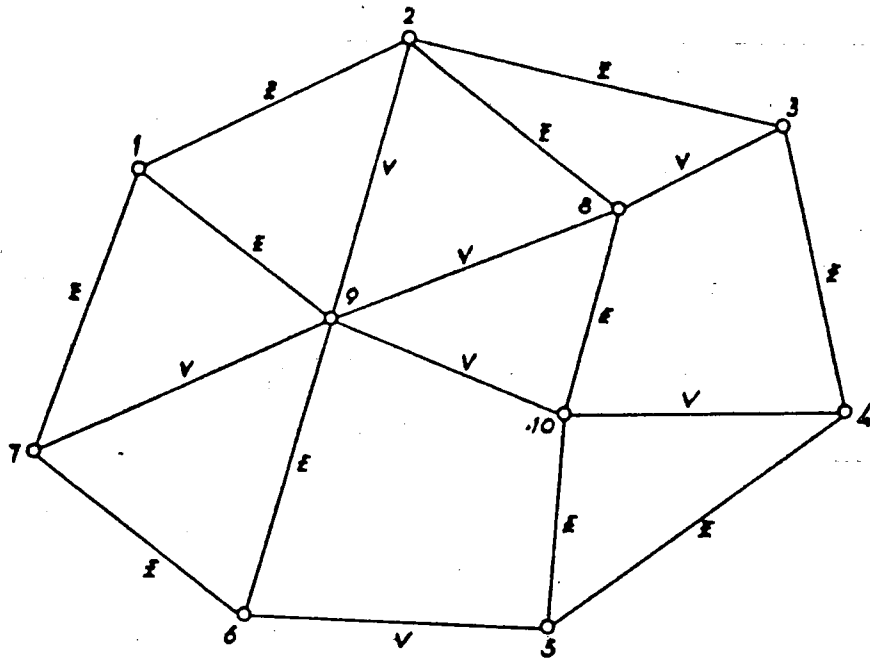


Edge (1, 9) has been coloured V, and the symbol "[1]" means that the colour of that edge will have to be removed if, in a subsequent step of the colouring process, the colour of edge (2, 9), marked with "(1)", is changed to V.

Now, since there are no more non-coloured interior edges, we can apply option (2) (b) and thus colour edge (7, 9) with E. We would next apply operation (2) (a) of I) thus colouring edge (6, 7) with V. The remaining non-coloured exterior edges would then be coloured by the repeated application of option (2) (a). When a solution would have been reached, we would once more use operation (3) of I) to search for another solution, thereby resulting, from (3) (a) of I), the choice of interior edge (7, 9), and so on.

Carrying out that routine, we would still obtain the following three colour solutions:





We shall leave to the reader the obtention of the R-layout-schemes corresponding to these solutions. However, after the next sections, on contour forms, we shall describe a simple method for obtaining the layouts from the coloured realizations. Indeed, that will be dealt with in chapter 6.

The contour form. In this chapter we have been concerned with solving Subproblem 1.5, given no conditions on the contour form except for general conditions C0 and C1. Nevertheless, we have already considered a specific type of contour form —the rectangle; that we have done by means of conditions C0 and C2, or equivalently, by means of condition C0-2. Up to now, none of the theorems resulting from such conditions have been proved, nor the equivalency of conditions, such as the one just mentioned.

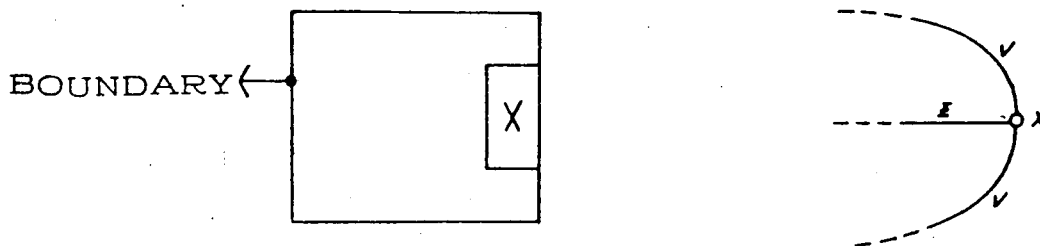
We will next engage in solving in detail Subproblem 2.1; the resolution of Subproblem 2.2 will result from imposing that the PHV-assignments obtained through the colouring process which solves Subproblem 1.5, must.

satisfy desirable contour form conditions. To be able to do so entails that such desirable contour forms be expressible in terms of colour conditions to be imposed upon the (exterior) nodes of the realization.

To fully solve Subproblem 2.2 we must also be able to specify a process for obtaining a R-layout-scheme with modular dimensions once a PHY-assignment is obtained satisfying any specific contour conditions (i.e. a CPHV-assignment). That problem shall be dealt with in the next chapter.

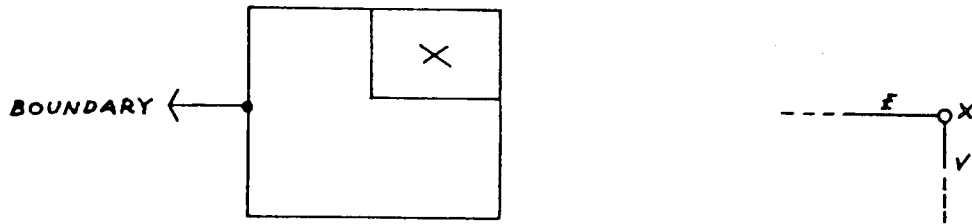
Let us start by remarking that any exterior rectangular space of a R-layout-scheme (derived from a non-separable realization) may only be adjacent to one, two, three, or four segments of the boundary. The latter case is of no interest since it corresponds to a layout scheme with just one space. Let us carry out an analysis of the remaining cases. In this analysis, X represents an exterior space, and x the exterior node that originates X .

Case I. X is adjacent to just one boundary segment.



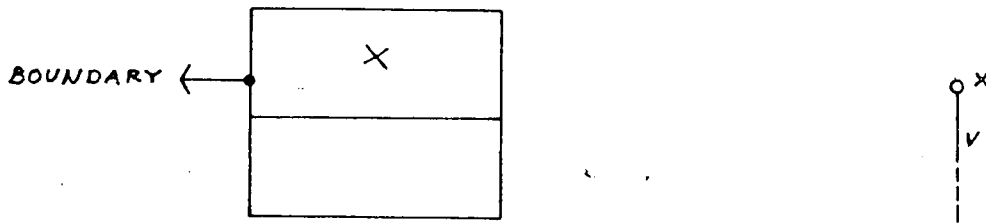
In the figure, we represent both X and x . Any of the coloured lines shown refers to a group of edges with the colour indicated. The relation between both figures is straightforward, if we recall that, by convention, "V" designates "horizontal" and "E" designates "vertical". This type of exterior space is thus characterized by $NG(x) = 2$, and the fact that the two boundary edges of x have the same colour.

Case II. X is adjacent to exactly two boundary segments.



In this case, x is characterized by $NG(x) = 2$, and the fact that the two boundary edges of x have a different colour.

Case III. X is adjacent to exactly three boundary segments.



In this case, x is characterized by $NG = 1$; i.e. all its edges have the same colour.

We shall now prove the theorem on condition C0-2, which states that:

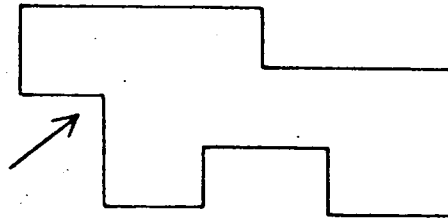
Theorem C0-2. It is a necessary and sufficient condition for a realization obeying conditions A and B to have a rectangular contour form, that $NG(x) < 4$ for every exterior node x , and that condition C1 holds.

Proof. We must show that the stated theorem is true for any of the three types of exterior nodes possible (we omit the case where the layout scheme has just one space because, since the corresponding node has no edges, any condition on its edges is true).

The fact that only the three types of exterior spaces noted, as well

as the types of groups of edges corresponding to them, are viable, already shows that CO-2 is a necessary condition. It now remains to be proved that it is also sufficient.

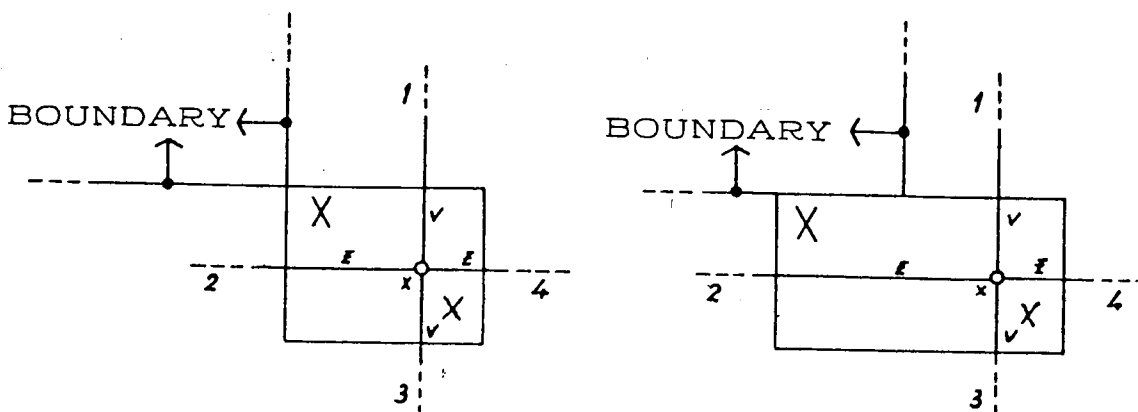
What does it mean to say that a closed contour is not rectangular? According to hypothesis one, the contour polygon can only have right angles or their complements to the circle ; i.e. in any non-rectangular contour form, something of this sort must happen:



The fact to bear in mind is that, in the non-rectangular case, there exist concave corners such as the one pointed out by the arrow.

Now, may a rectangular space be adjacent to the two consecutive boundary segments of a concave corner? No. In the vicinity of a concavity only two situations may in principle occur, in respect to a rectangular space.

They are depicted in the two following figures:



In either of them x is an exterior node with $NG(x) = 4$, and X is the corresponding exterior space. In the figure on the left, groups 1 and 2 of edges are indispensable for establishing the concavity. In the figure on the right, however, one might suppose that group two is not indispensable.

But, from the adjacencies' point of view, there is no reason why this figure should not reduce to the one on the left, by sliding the side corresponding to group 2 to meet point P of the concavity. In those circumstances, if group 2 did not exist, the concavity would not be genuine from a strictly non-metrical standpoint, since no particular dimensions would be then taken into account.

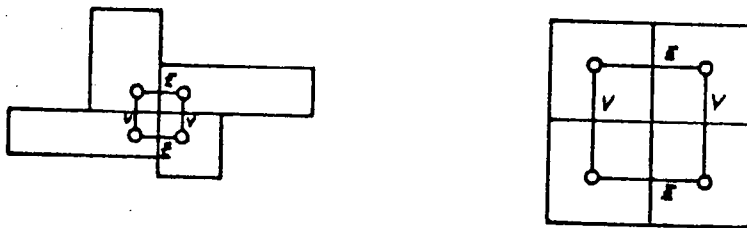
It thus seems to us clear that for a concavity to be "genuine", the existence of groups 1 and 2 is indispensable. On the other hand, if any of groups 3 or 4 did not exist, the realization would then be separable, thus contradicting condition A0.

Thus, in any non-rectangular contour, there must be at least one exterior node x , such that $NG(x) = 4$. But that amounts to saying that if $NG(x)$ is less than four for every node x of the boundary of a realization, then the contour of the corresponding layout-scheme cannot be non-rectangular.

Now, condition C1 stipulates in this case (since $N4 = 0$) that the total number of convex corners must be four. Thus, C0-2 is also a sufficient condition for a contour form to be rectangular. This ends the proof of theorem C0-2.

Note that, however, if dimensions are taken into account, we will always be able to obtain the figure on the right from the figure on the left, without endangering the existence of a concavity at node x .

This way, the figure on the left shown next reduces to the one on the right (unless some other spaces are created corresponding to, say, "north", "west"; "garden", "street", "river", etc., and if, in the layout-scheme obtained they are not drawn, as well as their corresponding nodes and its edges).



Next, we will have to show that C0 and C1 are necessary and sufficient conditions a (closed) contour must respect to comply to Hypothesis 1, and also that C0 and C2 (which includes C1) are necessary and sufficient conditions for a contour to be rectangular.

The equivalence of condition C0-2 to conditions C0 and C2 taken simultaneously, will follow from the proof that these conditions, taken in conjunction, are also necessary and sufficient for any realization obeying conditions A and B to have a rectangular contour form.

To sum it up, we will then have shown that:

- (1) C0 and C1 are, in conjunction, necessary and sufficient conditions for a contour to be permissible;
- (2) C0 and C2 (which includes C1) are, in conjunction, necessary and sufficient conditions for a contour to be permissible and rectangular;
- (3) C0-2 (which includes C0 and C1) is a necessary and sufficient condition for a contour to be permissible and rectangular;

(4) From (2), (3), C2, and C0-2, that C0-2 is equivalent to the conjunction of C0 and C2, and that condition " $NG(x) \leq 4$ for every exterior node x " is equivalent to condition " $N4=0$ " (i.e. "the contour has no concave corners").

Theorem of contour permissibility: It is a necessary and sufficient condition for the contour form a realization satisfying conditions A and B to comply to Hypothesis 1, that conditions C0 and C1 are both verified.

Proof. Note that:

- (a) C0 states that $NG(x) \leq 5$ for every exterior x ;
- (b) $NG(x) = 3$ is impossible;
- (c) $NG(x) = 2$ with the two boundary edges of exterior node x of the same colour does not give rise to a corner space;
- (d) According to the definitions of $N1$, $N2$, and $N4$, C1 takes into account all types of exterior nodes except the ones made irrelevant by (a) and (c).

Thus, all that remains to be shown is that equation $(2xN1+N2)-N4=4$ is a necessary and sufficient for contour permissibility.

That it is a necessary and sufficient condition can be seen from the fact that $(2xN1+N2)$ is the total number of convex corners and that $N4$ is the total number of concave corners; their difference must be 4 (convex corners) if the contour is to be closed and admit only right angles or their complements to the circle (i.e. if Hypothesis 1 is assumed), and vice-versa.

This ends the proof.

We shall now state and prove the following

Theorem of contour rectangularity: It is a necessary and sufficient condition for the contour form of a realization satisfying conditions A and B to be rectangular, that it verifies conditions C0 and C2.

Proof. Since C2 includes condition C1, it follows from the previous theorem that equation $(2 \times N1 + N2) - N4 = 4$ holds, which states that the contour is permissible and closed.

Because C2 stipulates that $N4 = 0$, that equation reduces to $2 \times N1 + N2 = 4$, stating that the number of convex corners is four. Since $N4 = 0$ means there are no concave corners, it follows that the contour is a rectangle and the theorem condition sufficient.

On the other hand, if the contour is a rectangle, it is obviously permissible, closed with no concave corners, and with exactly four convex corners; so, the theorem condition is necessary.

This ends the proof.

Imposing other contour forms.

In this section we describe how a closed contour form can be prescribed by use of the notions of number of convex corners, number of concave corners, and of sequences of both types of corner, with the help of the general equation of condition C1.

Let us first write that general equation in the form we have been used up to now:

$$(2 \times N1 + N2) - N4 = 4 .$$

Note that the expression within parentheses is the total number of convex corners, which we shall denote by cv . Furthermore, the total number of concave corners, N_4 , will henceforth be denoted by cc . Rewriting the equation one has:

$$cv - cc = 4 .$$

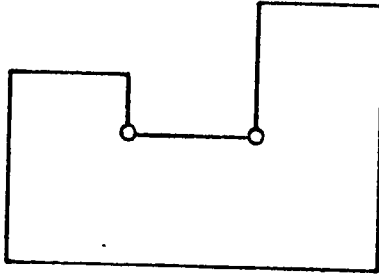
Suppose now that we are interested only in those layouts having two concave corners; i.e. those for which $cc = 2$. The two concave corners will separate the convex corners into two groups of consecutive corners: cv_1 and cv_2 . Thus, the above equation becomes:

$$cv_1 + cv_2 = 6 .$$

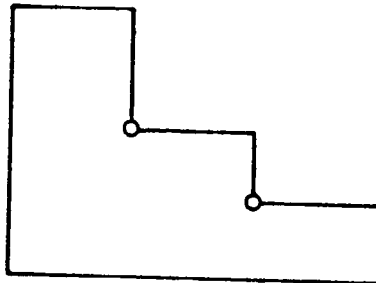
The various possible forms of layout schemes with two concave corners, become determined by all the different pairs of non-negative integers that satisfy this equation. They are four:

a	0	6
b	1	5
c	2	4
d	3	3

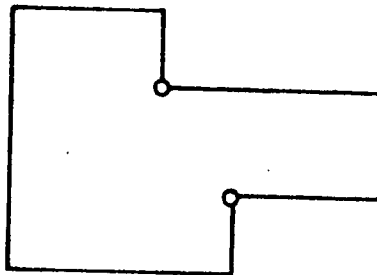
Form a :



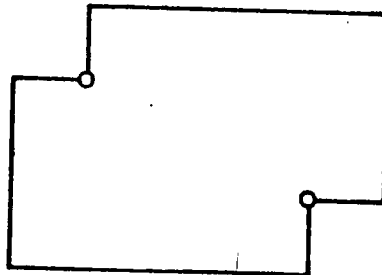
Form b :



Form c :



Form d :



Let $\overset{CC}{\cancel{CC}}$ and $\overset{CV}{\cancel{CV}}$ denote, respectively, a concave and a convex corner.

Define concave indentation as the sequence of corners expressed by:

$$CV - CC - CC - CV$$

Define convex indentation as the sequence of corners expressed by:

$$CC - CV - CV - CC$$

Define isolated concave corner as the sequence of corners expressed by:

$$CV - CC - CV$$

1) Let us next suppose that we wanted to characterize, in terms of sequences of corners, those contour forms obeying the following conditions:

- (1) it may have isolated concave corners, such as the ones exemplified in Form d ;
- (2) it may have concave indentations, as the one exhibited in Form a ;
- (3) it may have convex indentations, as the one illustrated in Form c ;
- (4) it may not have any other types of concave and convex corner arrangements, besides (1), (2), and (3).

It may be seen that the contour forms that interest us may have any number of concave corners as long as the following sequences of corners are always observed:

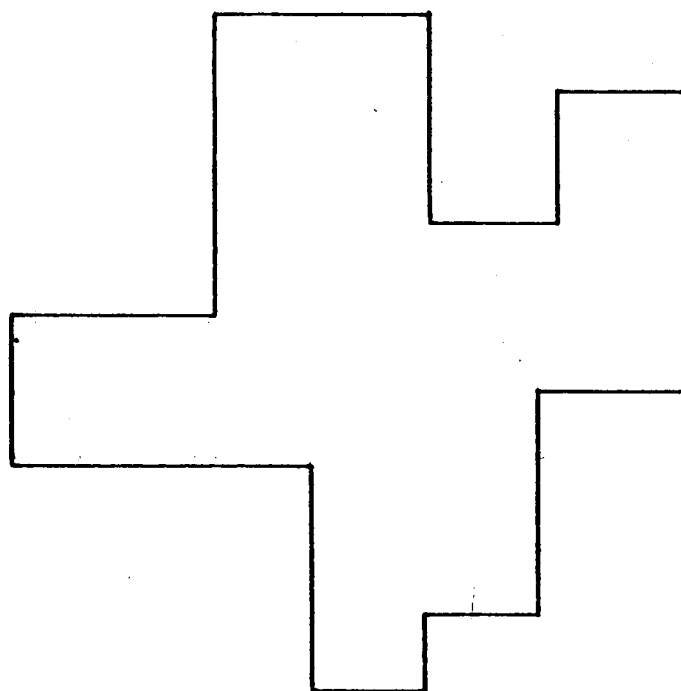
- (a) the sequence around the contour must verify: $cc - cv = 4$; this is the general condition on contours, and the only restriction on convex corners

(b) from (1), (2), (3), and (4), it follows that no sequence of more than two consecutive concave corners is permitted.

(c) it follows from (2) that any two sequences of consecutive concave corners must be separated by at least two consecutive convex corners. Intuitively, this means that any concave indentation must be "undone" before another is possible.

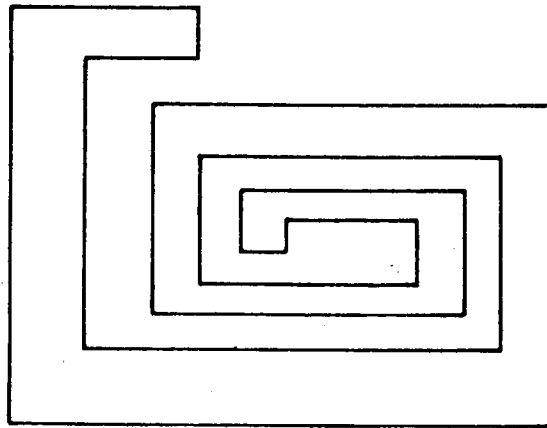
These three rules are the ones actually used in the computer program developed; we shall refer to the type of contour they define as the indented contour form.

An example of an indented contour form is:

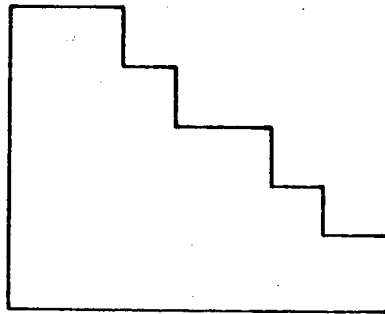


Other examples of contour forms:

II) A spiral is any sequence of consecutive concave corners followed by the same number of consecutive convex corners plus four:



III) A ladder is any sequence of concave corners alternating with convex corners, starting with a concave corner, and followed by four convex corners:



IV) Looked at from inside, a permissible contour form obeys this equation:

$$cc - cv = 4$$

This is so because in that case every concave corner becomes convex and every convex corner becomes concave. Thus, if one imposes on the contour of a representation the equation:

$$cv - cc = -4,$$

equivalent to the previous one, an interior patio is being specified, whose form can be controlled by supplementary rules.

V) Let us now derive all possible contour forms with three concave corners. First of all we have:

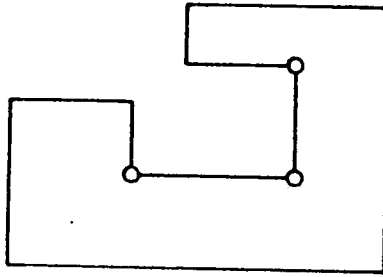
$$cv1 + cv2 + cv3 = 7.$$

The possible combinations of values are shown in the following table, disregarding all symmetries:

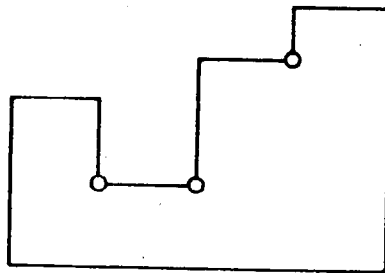
a	0	0	7
b	0	1	6
c	0	2	5
d	0	3	4
e	1	1	5
f	1	2	4
g	1	3	3
h	2	2	3

From it, we derive the eight possible contours forms exhibited below:

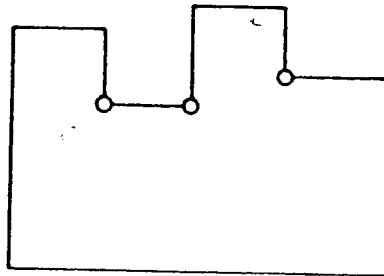
Form a:



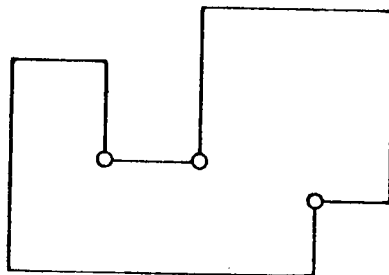
Form b:



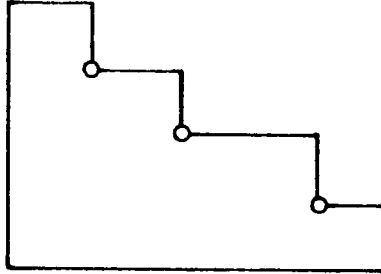
Form c:



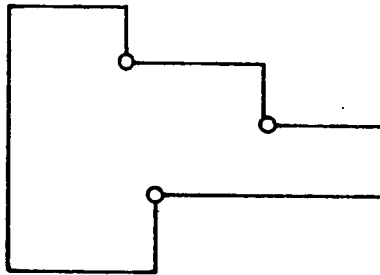
Form d :



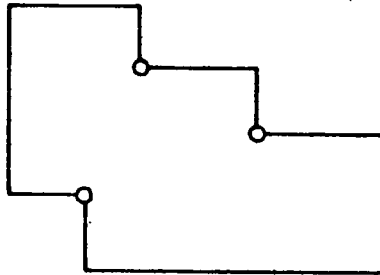
Form e :



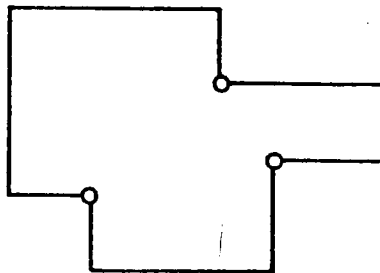
Form f :



Form g :



Form h :



CHAPTER 6

HOW TO DRAW A MODULAR LAYOUT FROM A PERMISSIBLY COLOURED REALIZATION

1. INTRODUCTION

In this chapter, a way is given for drawing any R-layout-scheme resulting from a permissibly coloured realization and having an indented contour form (eventually rectangular).

The process consists of two stages:

(1) Given a dimensional rectangular module (for example), each of the dimensions of every space is computed as a multiple of the correspondent modular dimension, such that the spaces can be juxtaposed on the plane in the unique way specified by their adjacencies.

(2) Given the absolute coordinates of the centre of one exterior space, as well as its orientation, relative to two orthogonal axis, the orientation and the centre coordinates of all other spaces are found, thus permitting to draw the whole layout scheme in the unique way specified by the adjacencies between spaces.

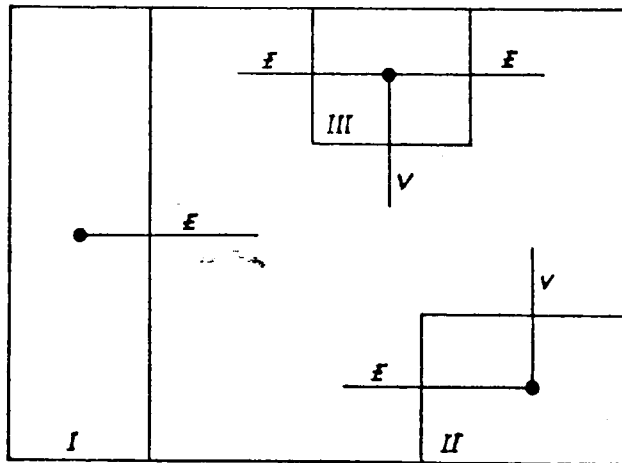
The process obtained for carrying out these two stages is said to solve Problem 3, and each of its two stages is said to solve Subproblem 3.1 and Subproblem 3.2, respectively.

2. RECTANGULAR CONTOUR MODULAR LAYOUT

First of all, we will treat the case where the contour is rectangular,

i.e. a special case of the general indented contour form.

The hypotheses of contour rectangularity and realization non-separability bring as a consequence that there can only be three types of exterior (rectangular) spaces, as we previously noted. Those three types are illustrated in the next figure, where each coloured line belongs to a group having the indicated colour and at least one edge. Furthermore, the type of an exterior node is the type of its corresponding space.

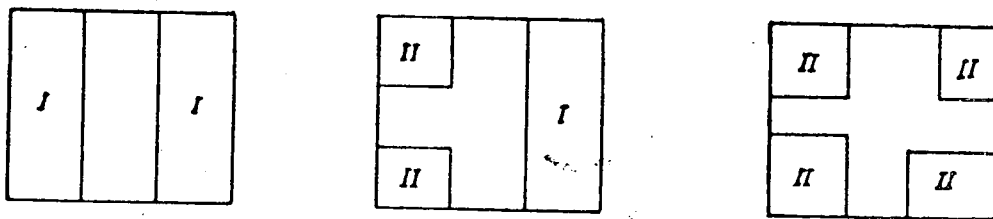


Thus, spaces of type I correspond to exterior nodes with just one group of edges; type II spaces to exterior nodes with just two groups of edges and having their two boundary edges of a different colour; type III spaces to exterior nodes with exactly two groups of edges, and having their two boundary edges belonging to the same group. N_1 , N_2 , and N_0 , are the number of spaces of type I, II, and III, respectively. Intuitively, a node contributes to N_1 if it gives rise to a space occupying exactly two convex corners of the contour of the layout; to N_2 if just one convex corner is occupied; to N_0 if no corner is occupied.

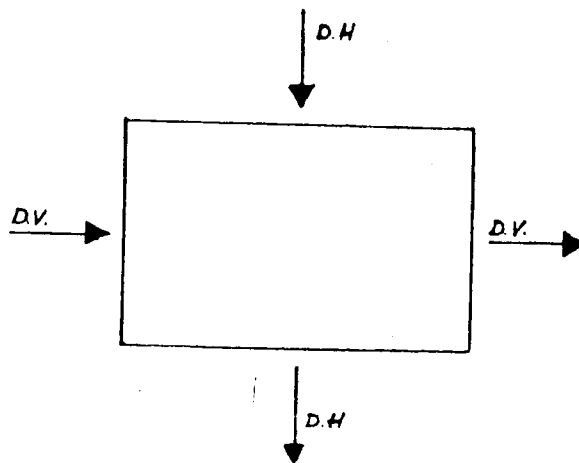
From the above figure it is easily seen that, if a permissibly coloured realization originates a rectangular layout scheme, the following facts hold:

- (1) there will not exist in it more than two nodes of type I;
- (2) there will not exist in it more than four nodes of type II;
- (3) only two nodes of type II may coexist with one node of type I, there not being any other possible combination of these two types of nodes.

These facts lead to the only three possible combinations of type I and II of spaces, coexisting with any number of type III spaces:



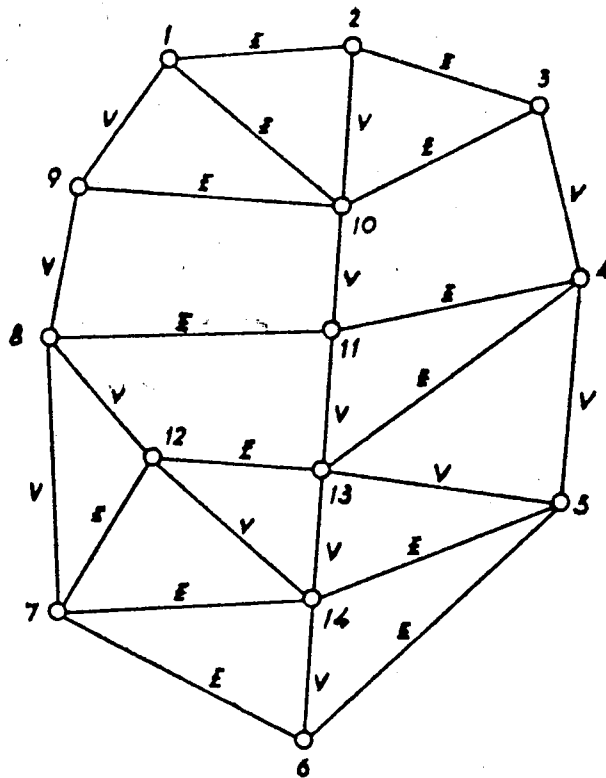
Consider next any rectangular contour:



It is apparent that however the horizontal dimension may be distribu-

ted within the layout, the total horizontal dimension (D.H.) "entering" the layout, must be equal to the total horizontal dimension which "exits" from it, as is also shown in the previous figure for the vertical dimension (D.V.).

Next, let there be the following permissibly coloured realization:



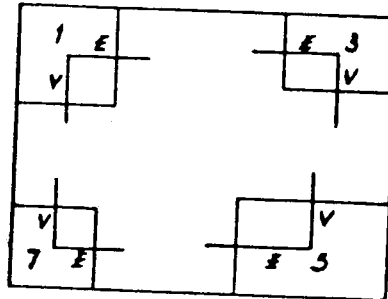
Let us now identify the exterior nodes according to the three types defined:

I - there are none

II - 1, 3, 5, and 7

III - the remaining ones

Consequently the layout scheme to be derived from the given realization will respect the form:



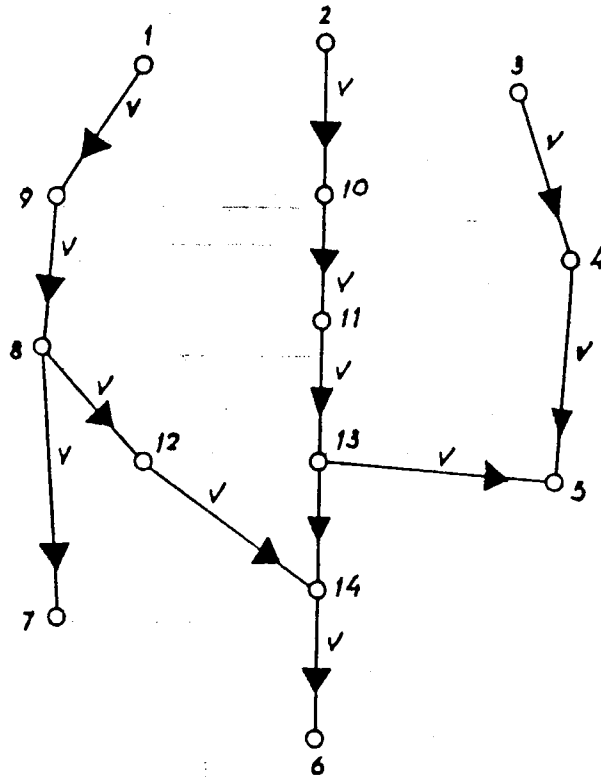
Where each line denotes a group of edges with the colour shown.

This way, the nodes in between nodes 1 and 3, as well as these nodes, delimit the total horizontal dimension of the layout. The same may be said of nodes in between nodes 5 and 7, these nodes included.

Similarly, the total vertical dimension is delimited by nodes 3, 4, and 5, or, equivalently, by nodes 7, 8, 9, and 1.

Let us now draw the following subgraph, obtained from the previous graph by taking only those edges that are coloured with V. Recall that to a V coloured edge there corresponds in the layout a horizontal partition segment.

We shall call this subgraph the horizontal subgraph of the realization. The vertical subgraph of the realization is similarly obtained. The sense of the arrows is justified as follows.



(1) For interior nodes. Since to each space and so to each node, there corresponds a horizontal (vertical) dimension "entering" equal to the horizontal (vertical) dimension "exiting", the edges of each of the two pairs of opposite groups of an interior node must have their arrows oriented differently with respect to the node. Recall that by condition B3, there are always two pairs of opposite groups of edges of the same colour in any interior node of a permissibly coloured realization.

(2) For exterior nodes . From the point of view of edge orientation, the two edges belonging to the boundary in a type III node are considered to belong to two different edge groups, although they are consecutive and have the same colour; furthermore, the two groups are considered to be

opposite groups. Bearing this in mind, the orientation of the two opposite groups of edges with the same colour of a type III node, is performed as for interior nodes; i.e. the edges of one group have an arrow sense different from the edges of the other group (relative to the node).

As for the orientation of the isolated colour groups in the three types of exterior nodes, the only condition is that in each group all the edges must be given the same arrow sense relative to the node; which arrow sense it is depends upon the orientation of adjacent nodes.

N.B. Note that in non-rectangular contours there is another type of exterior node, defined by $NG(x) = 4$. The orientation of its edges follows the rule used for interior nodes. We shall call them type IV nodes.

Remark. As can easily be seen, the foregoing rules are quite sufficient for orienting all the edges of a permissibly coloured realization, as long as any arbitrary orientation is first given to any two edges of different colour.

The choice of those two edges and of their orientation defines, thus, the orientation of the whole layout scheme, relative to an external orthogonal referential on the plane.

Note, moreover, that although orientation is dependent upon the colouring, the converse is not true.

Obtaining the modular dimensions. Consider again the previous figure showing the horizontal subgraph of a permissibly coloured realization.

In that subgraph the total horizontal dimension "enters" through nodes 1, 2 and 3, and distributes itself by all other nodes in the way indicated by

the arrows.

Thus the subgraph clearly shows that:

(1) space 9 will be immediately "below" space 1;

(2) space 8 will be immediately "below" space 9, and so, indirectly "below" space 1, and so on; space 14, for example, will be directly "below" both space 12 and space 13.

(3) note that directly "below" space 8, for example, will be spaces 7 and 12, which thus become spaces indirectly "below" spaces 9 and 1.

The information contained in these facts may be expressed as follows.

(1) is expressed by the column:

1
9

(2) is expressed by the column:

1
9
8

(3) is expressed by the two columns:

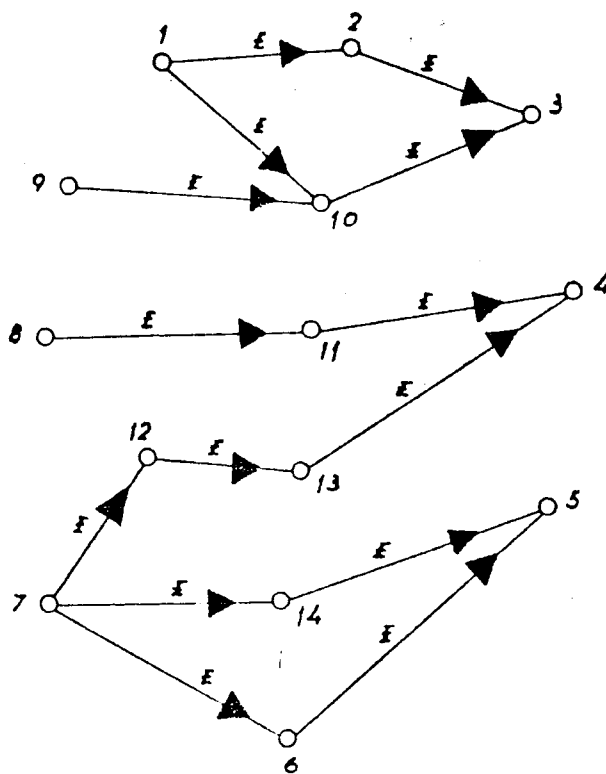
1	1
9	9
8	8
7	12

Now, the information contained in the *whole* subgraph is expressed by the following table of columns:

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

To obtain the table of rows relative to the vertical subgraph corresponding to the same realization, we proceed in the same fashion, considering in that case that the total vertical dimension "enters" through nodes 1, 9, 8, and 7.

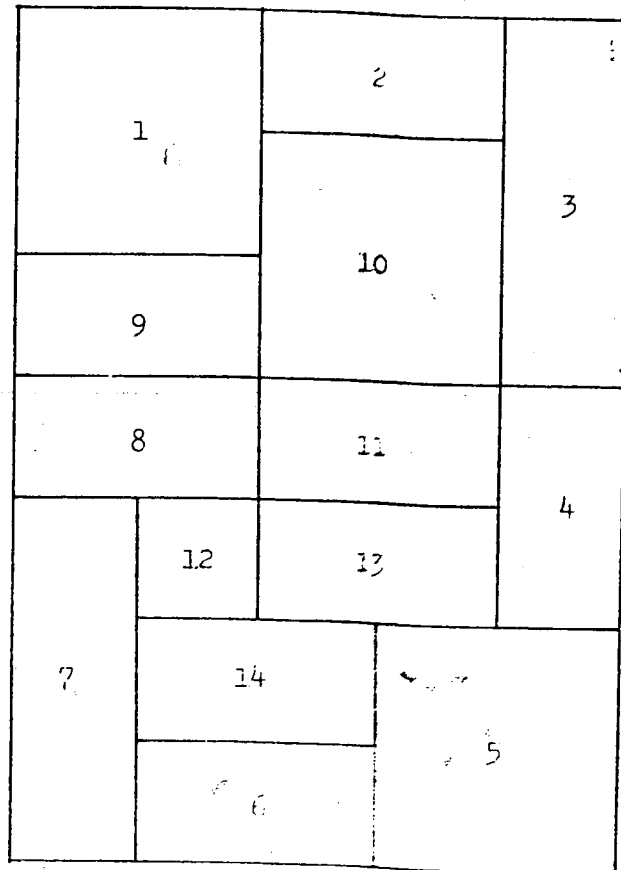
Next, we present the above mentioned subgraph and the table of rows that can be derived from it.



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

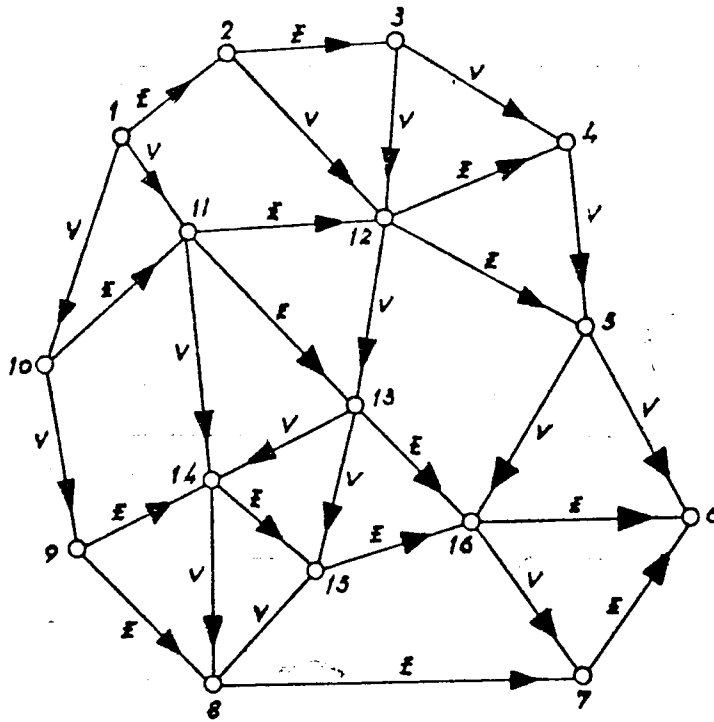
Once we obtain both tables, the dimensions of any space become specified by counting the number of occurrences of the space number in each of the two tables; i.e. the two numbers obtained for each space are the two multiples that define its dimension in terms of the dimensional module chosen.

This way, from the previously given coloured representation we may obtain, given the dimensional module, the following layout scheme, in which all stipulated adjacencies are satisfied:



Another example. We now present another example, where we have passed directly from the coloured oriented realization to the two dimensional tables; while the first is of columns and corresponds to the horizontal dimension, the second is of rows and corresponds to the vertical dimension.

Coloured Oriented Realization:



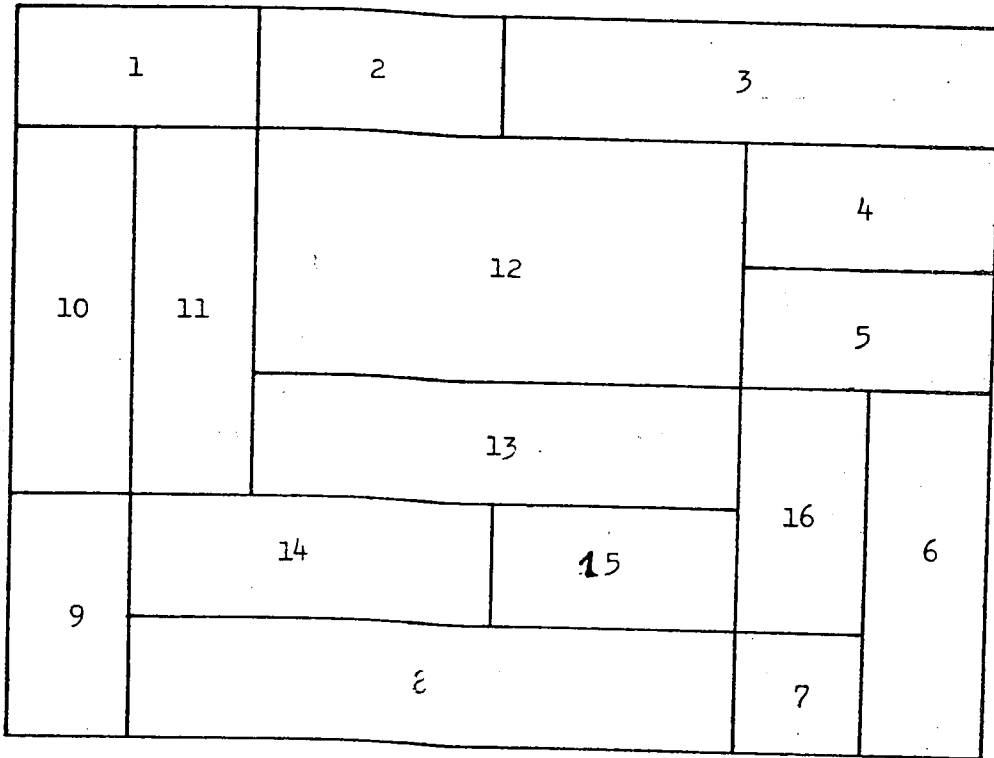
Horizontal Dimension Table (columns):

1	1	2	2	3	3	3	3
10	11	12	12	12	12	4	4
9	14	13	13	13	13	5	5
	8	14	15	14	15	16	16
		8	8	8	8	7	

Vertical Dimension Table (rows):

1	2	3		
10	11	12	4	
10	11	12	5	
10	11	13	16	6
9	14	15	16	6
9	8	7	6	

Layout Scheme Obtained:



In the modular dimension process for rectangular contours, the absolute coordinates of the spaces are easily obtained since we know which spaces occupy the four corners. In fact, one may start by drawing the spaces through which one same total dimension "enters", and proceed by juxtaposing to them all those spaces to which they are adjacent in the subgraph corresponding to the horizontal or vertical dimension chosen. Since all "entering" nodes for that dimension have a common absolute coordinate, namely the one corresponding to the direction of space juxtaposition indicated by the subgraph, any space having more than one "ancestor" exterior space, receives from them, by way of the oriented subgraph, exactly the same absolute coordinate.

However, in an indented contour, for example, the "entering" nodes of a chosen subgraph do not all necessarily have a common coordinate value

relative to the dimensional flow of the subgraph. The problem arising from this situation will be dealt with in the next section.

Drawing Modular Indented Layouts. The method we use for computing the absolute coordinates of the spaces of an indented contour layout will be expounded in this section.

We do not need another method, other than the previous one used, for computing the dimensional values of the spaces in a rectangular contour layout. In fact, the only new type of node is the exterior node with $NG(x) = 1$, and the way to orient its edges is exactly the same as for interior nodes, as noted in the previous section.

So, given that we already have the modular dimensions of every space, all that remains to be done is to calculate the absolute coordinates of their centres, granted that their orientation is made clear by the coloured adjacencies.

Now, the problem is solved if we are able to compute the coordinates of all exterior spaces, thus obtaining an "external skeleton" for the layout, because once these coordinates are computed, we may then use the same process as before for obtaining the coordinates of the interior spaces since, even though an interior node may have more than one "entering" node as an "ancestor", the coordinates of its "ancestors" will already have been compatibilized in the calculation of the "external skeleton" of the layout.

Thus, our only concern in this section is to provide a way for calculating the "skeleton" or "outer shell" of the layout. This is accomplished by means of a finite state machine (i.e. a finite automaton) that "goes around" the contour of the coloured oriented realization, with the purpose of recognizing the type of nodes encountered (its input), making the appropriate changes in "direction" (one of its internal states) and computing the two absolute coordinates of the space to be derived from the current node (its two other internal states).

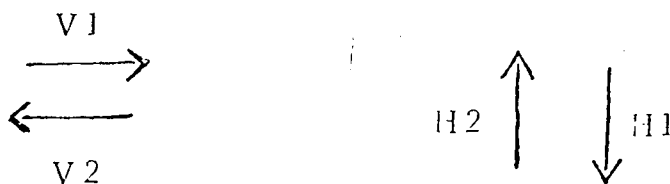
In reality, the sequence of node types around the contour is a string of numbers which has already been computed, and output to memory, by a previous automaton, responsible for ensuring an indented contour form for each colour solution of the realization.

Furthermore, the new automaton may be decomposed into two:

- (1) one that computes the changes in direction at each node;
- (2) another that computes a pair of absolute coordinates for each node encountered.

Let us now describe these two automata, and show how they work.

Let V1, V2, H1, and H2 be the senses and directions shown below.



The input to any of the two automata consists in one of the different types of nodes, I, II, III, or IV; these types of nodes are encountered clockwise, around the boundary of given coloured solution of a realization.

The state of automaton 1 is the direction and sense it had before a new input is encountered; it can be any of four senses of direction (H1, H2, V1 or V2).

The state of automaton 2 is an ordered triple, consisting of: the state S of automaton 1, the absolute horizontal coordinate attributed to the node visited last, and the absolute vertical coordinate for that node.

The outputs of any of the automata are their states.

Their initial state is defined by:

(1) the arbitrary coordinates, relative to two orthogonal axis, given to any one exterior node, and interpreted as those of the centre of its corresponding rectangular space;

(2) the initial direction and sense given by any one of the boundary edges of the node chosen in (1), as follows: if its colour is E then the initial direction and sense is V1, otherwise it is H1.

Note that the choice made in (2) does not guarantee the various solutions of a realization to become positioned in a similar way. For that purpose other rules are used in the program besides these, but they are not relevant to this exposition.

Now let CH and CV be the absolute coordinates, respectively horizontal and vertical, of the last node visited by the two automata. Let DH and DV

be the dimensions, respectively horizontal and vertical, of that node; and let NDH and NDV be the dimensions, respectively horizontal and vertical, of the input node type.

Define ΔH , ∇H , ΔV and ∇V , as follows:

$$\Delta H = (DH + NDH) / 2$$

$$\nabla H = (DH - NDH) / 2$$

$$\Delta V = (DV + NDV) / 2$$

$$\nabla V = (DV - NDV) / 2$$

These are the possible absolute increments of the coordinates present in the state of automaton 2 .

Now, the next state of automaton is 1 given by the combination of its input and its present state, according to the following double entry table:

	I	II	III	IV
V1	V2	H1	V1	H2
V2	V1	H2	V2	H1
H1	H2	V2	H1	V1
H2	H1	V1	H2	V2

The next state of automaton 2 is given by the combination of its input and its present state, according to the following double entry table:

	I	II	III	IV
V1	S	S	S	S
CH	$CH + \Delta H$	$CH + \Delta H$	$CH + \Delta H$	$CH + \Delta H$
CV	$CV + \nabla V$	$CV + \nabla V$	$CV + \nabla V$	$CV - \nabla V$
V2	S	S	S	S
CH	$CH - \Delta H$	$CH - \Delta H$	$CH - \Delta H$	$CH - \Delta H$
CV	$CV - \nabla V$	$CV - \nabla V$	$CV - \nabla V$	$CV - \nabla V$
H1	S	S	S	S
CH	$CH + \nabla H$	$CH + \nabla H$	$CH + \nabla H$	$CH - \nabla H$
CV	$CV - \Delta V$	$CV - \Delta V$	$CV - \Delta V$	$CV - \Delta V$
H2	S	S	S	S
CH	$CH - \nabla H$	$CH - \nabla H$	$CH - \nabla H$	$CH + \nabla H$
CV	$CV + \Delta V$	$CV + \Delta V$	$CV + \Delta V$	$CV + \Delta V$

This table may be reduced to:

	I, II, or III	IV
V1	S	S
CH	$CH + \Delta H$	$CH + \Delta V$
CV	$CV + \nabla V$	$CV - \nabla V$
V2	S	S
CH	$CH - \Delta H$	$CH - \Delta H$
CV	$CV - \nabla V$	$CV + \nabla V$
H1	S	S
CH	$CH + \nabla H$	$CH - \nabla H$
CV	$CV - \Delta V$	$CV - \Delta V$
H2	S	S
CH	$CH - \nabla H$	$CH + \nabla H$
CV	$CV + \Delta V$	$CV + \Delta V$

Note that minimizing and coding techniques could be applied next to transform this tableau into a simpler one with a simpler access. Although that has been carried out for the program developed, it need not concern us here.

We remark though, that the workings of the two automata can easily be followed with paper and pencil alone.

Of course, the two automata just defined work as well with dimensions not necessarily modular, as long as they are appropriately computed for providing a layout scheme respecting the given adjacencies and colours.

Example of the workings of the automata.

Consider the following permissibly coloured and oriented realization:

Types of exterior nodes:

I - 7

II - 1, 2, 10, 13, 14

III - 4, 6, 8, 9, 12, 15

IV - 3, 5, 11

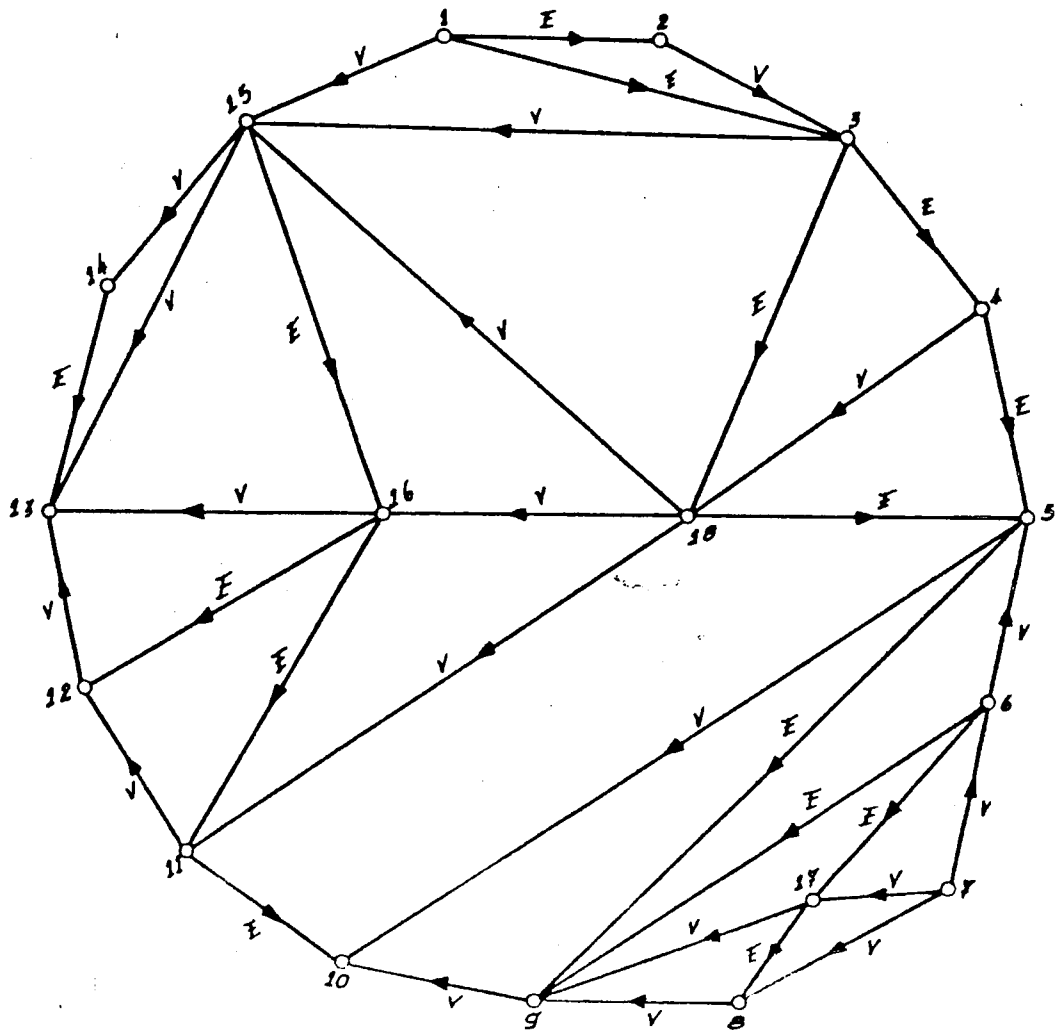
Interior nodes: 16, 17, 18 .

Horizontal Dimension Table (columns):

1	1	2	2	4	4	4	4	7	7	7
15	15	3	3	18	18	18	18	6	17	8
13	14	15	15	15	15	16	11	5	9	9
		14	13	13	14	13	12	10	10	10
							13			

Vertical Dimension Table (rows):

1	2			
1	3	4	5	9
1	3	18	5	9
15	16	12		
15	16	11	10	
14	13			
6	9			
6	17	8		
7				

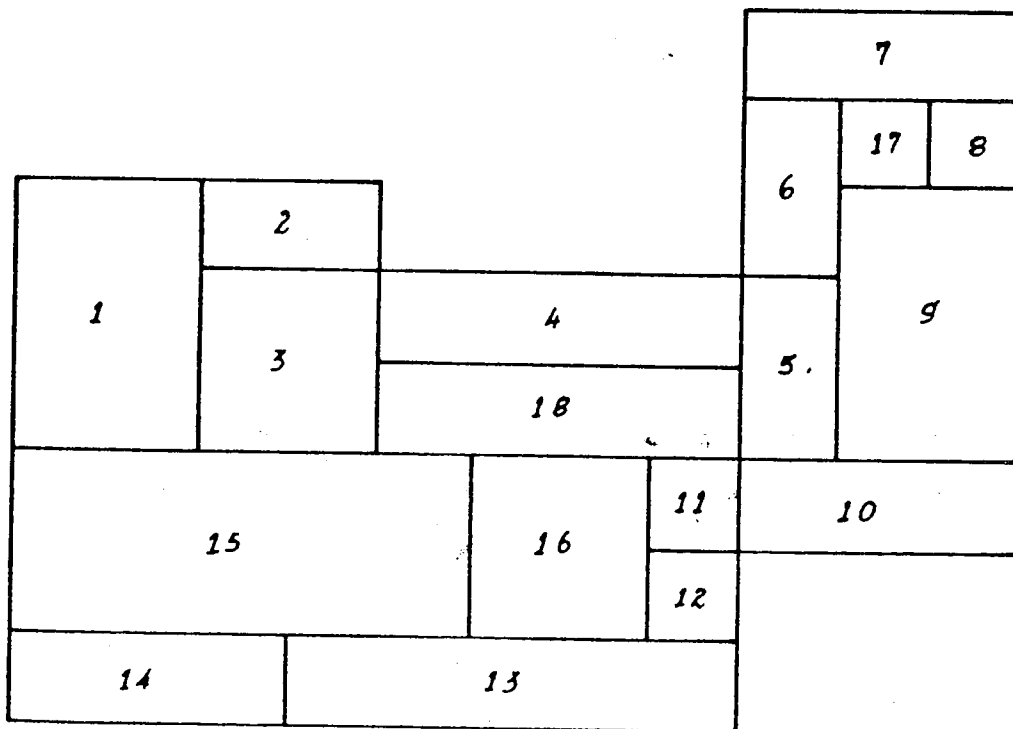


Dimensions taken from the tables:

<u>Node</u>	<u>Horizontal</u>	<u>Vertical</u>
1	2	3
2	2	1
3	2	2
4	4	1
5	1	2
6	1	2
7	3	1
8	1	1
9	2	3
10	3	1
11	1	1
12	1	1
13	5	1
14	3	1
15	6	2
16	1	2
17	1	1
18	4	1

Choose node 1 for getting the initial state; assign to it coordinates (0,0); obtain V1 as first sense of direction (corresponding to edge (1,2)).

Next, apply automata 1 and 2 and obtain the following layout scheme:



module:



CHAPTER 7

HOW TO FIND PERMISSIBLE DESIRED DIMENSIONS FOR A LAYOUT

1. INTRODUCTION

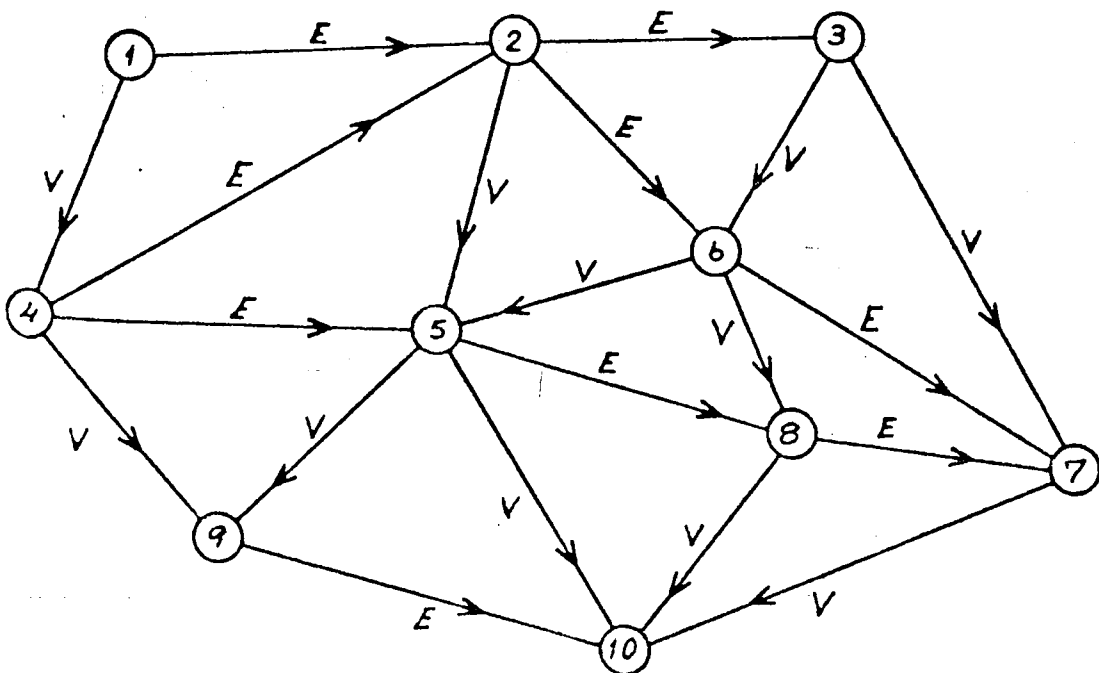
When the dimensions of the spaces of a layout, and the extent of contact between adjacent spaces, are not restricted to given values, it is apparent that it is possible to draw the layout corresponding to a given permissibly coloured and oriented realization such that all its spaces have a modular dimension. We have in fact shown a process whereby the spaces' dimensions can be obtained as multiples of any given module, for the case of layouts with an indented contour form. This restriction on the form of the contour arises because the process, as described, does not take into account the circumstance where the spaces may have such dimensions as to overlap, as for example in the case of a spiral contour, or, in general, as long as there are three consecutive concave corners. This circumstance may of course be detected, and, in the case of modular layouts, appropriate corrections of dimension may be made. We have not however dealt with the case of a general contour form, and so the dimension assigning process to be described in this chapter, since it makes use of the automata used for computing the absolute coordinates of the exterior spaces, remains restricted to indented contour forms, in the sense that if overlapping situations occur they will not be detected; i.e. layouts not belonging to the indented class may eventually be drawn by the program with overlapping non-adjacent spaces.

Note that in this chapter we will be taking Hypotheses 4 and 5 into account. Furthermore, the process devised for assigning permissible desired dimensions may also be used for modifying a solution already achieved by the process, into another viable one, thus also providing the means for an interaction facility between the program and the user. Moreover, partial solutions may be obtained when complete solutions are impossible within the dimensional and excess limits imposed through the above mentioned hypotheses.

2. ASSIGNING DESIRED DIMENSIONS PERMISSIBLY

To explain the process utilized, we shall first describe it by means of an example.

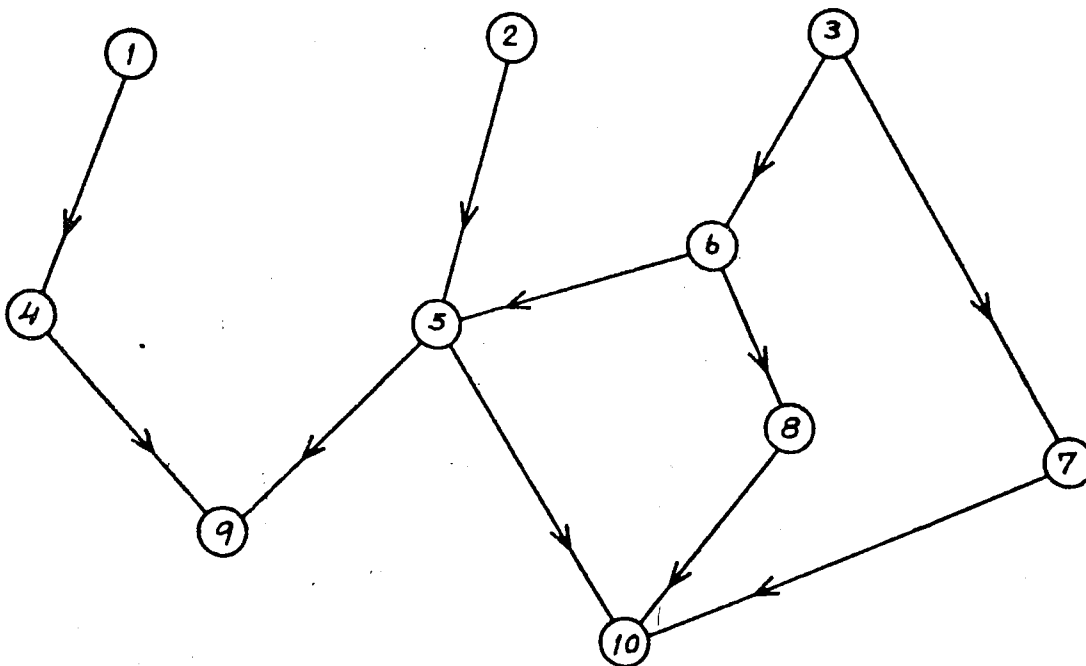
For that purpose, consider that the following permissibly coloured realization is given, together with the table of the dimensional tolerance intervals specified for the spaces. We further suppose that no special tolerance intervals are imposed in this example on the adjacencies between spaces, as foreseen by Hypothesis 5.



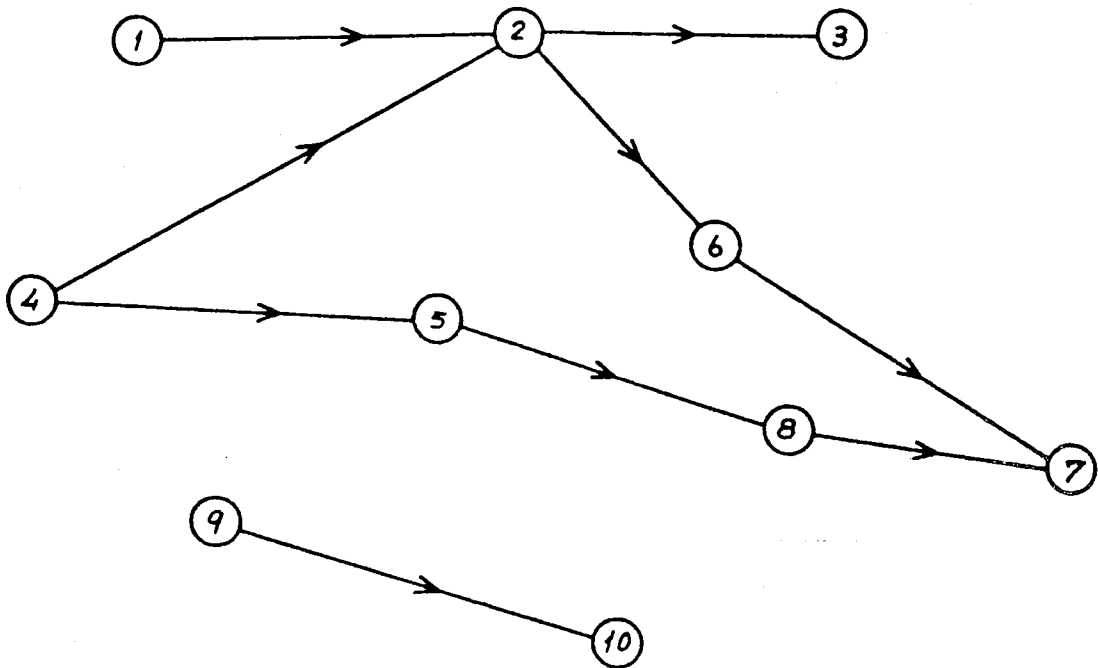
Tolerance Intervals:

<u>Node</u>	<u>Horizontal</u>	<u>Vertical</u>
1	2-3	1-2
2	2-3	2-5
3	3-5	1-2
4	1-3	2-4
5	2-4	1-2
6	1-3	4-6
7	1-2	2-6
8	1-2	1-2
9	2-5	1-3
10	3-4	2-3

The horizontal subgraph and the vertical subgraph of the given realization can be easily extracted from it; they are shown below.



Horizontal subgraph



Vertical subgraph.

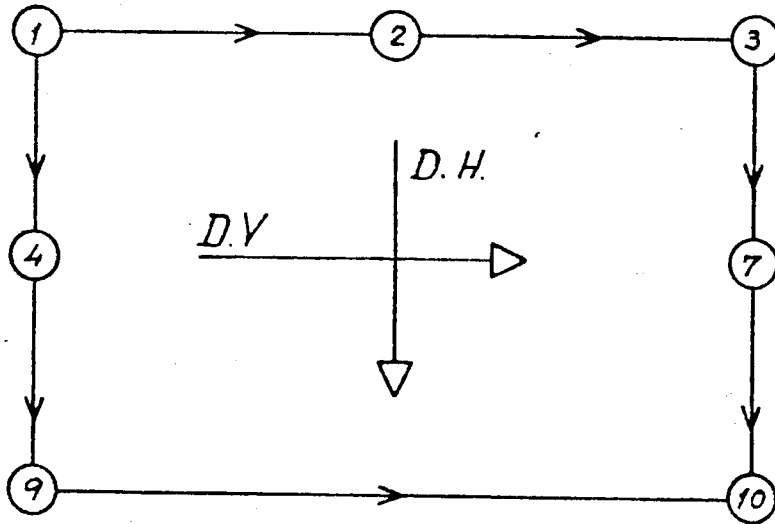
The exterior nodes of each type are:

- I - none
- II - 1, 3, 9, 10
- III - 2, 4, 7
- IV - none

The interior nodes are: 5, 6, 8.

It follows that the contour is rectangular, with the total horizontal dimension (D.H.) "entering" through nodes 1, 2, and 3, and "exiting" through nodes 9 and 10; the total vertical dimension (D.V.) "enters" through 1, 4, and 9, and "exits" through 3, 7, and 10.

The next figure shows these facts.



The method used consists, essentially, in obtaining successive node and edge dimension distributions, for each subgraph, such that, in the end, the total dimension "entering" each node through its set of in-going edges, is equal to the total dimension that "exits" through its set of out-going edges; if one of these sets is empty, this condition is not imposed on the node. Furthermore, the dimension of each node and the dimension conveyed by each edge must be within the tolerance intervals specified.

Consider any of the subgraphs above.

To start with, the process devised begins by assigning to each node an integer value from its tolerance interval; next, the dimensions so obtained for each of the "entering" nodes are distributed by their out-going edges, if possible proportionally to the assigned dimensions of the receiving nodes corresponding to those edges. Afterwards, a modification of dimension is carried out for those nodes for which the assigned dimension does not coincide with

the total dimension "entering" them and or with the total dimension "exiting",
ment, in case any one of these dimensions has already been defined. This is
done by use of certain non-ambiguously prescribed rules, to be shown further
on. If the assigned dimension of any node cannot be conveniently altered, i.e.
in such a way as to maintain it within the furnished limits, a well defined
procedure is initiated to modify the dimension values conveyed by its edges,
making all the indispensable subsequent adjustments of other nodes' and/or
edges' dimension values, such that all agreement possibilities are systema-
tically explored if it becomes necessary, and also in such a way that the
whole process converges to an equilibrium.

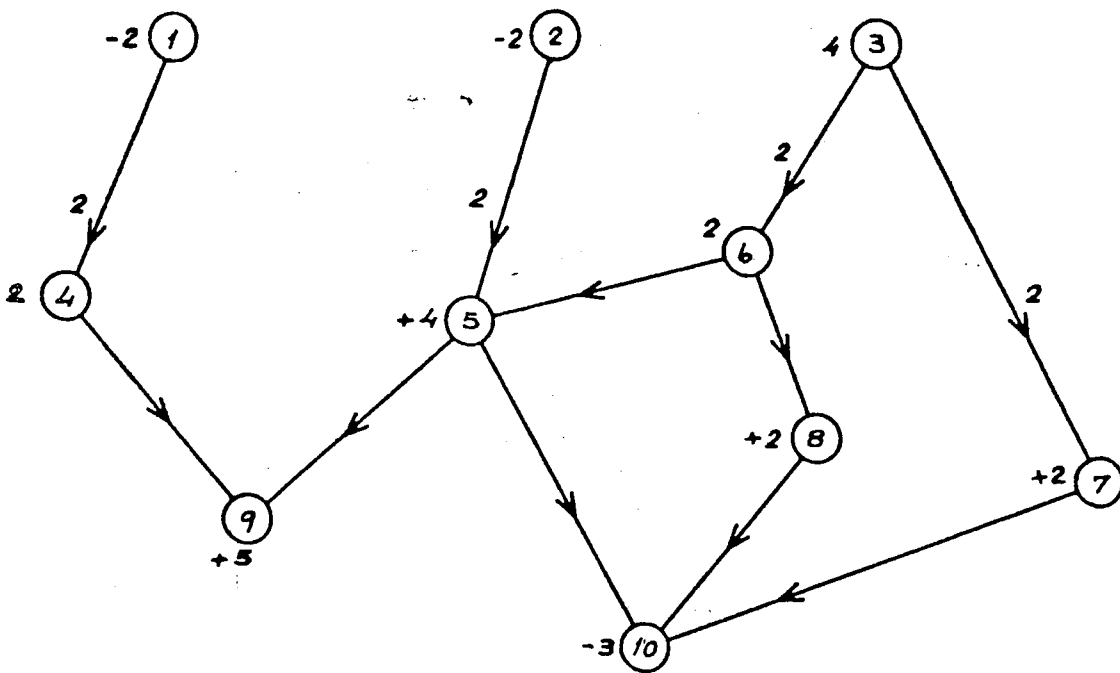
Only then a further distribution of dimensions is effected, now from
those nodes whose total "entering" dimension is known from its afferent ed-
ges, which is already compatible with their tolerance interval for the dimen-
sion under consideration; their dimension is distributed by its efferent edges,
by means of integer numbers as close as possible to the proportion of dimen-
sion values of the nodes receiving them, all the while respecting the edges'
dimension constraints.

Once these operations are gone through for each dimensional subgraph,
all nodes having been inspected, either a solution for the distribution of di-
mension will have been found, or the inexistence of such a solution proven.

What is more, once a solution is attained, the equilibrium thereby
achieved may be perturbed, either locally or globally, by imposing fixed
dimensions to some of the edges and/or nodes other than the ones which
had been found. This puts the dimension adjusting mechanisms in search of:

new equilibrium, and a malleable user/machine interaction becomes this way facilitated.

Let us now return to our example, and consider in the first place the horizontal subgraph. We start by attributing to its nodes any integer value belonging to their respective tolerance intervals. Further on, it shall be argued that an integer value closest to the mean interval value constitutes a best initial choice, but, for illustrating purposes, we make the initial choice of values shown in the figure below, where a permissible selection was made.



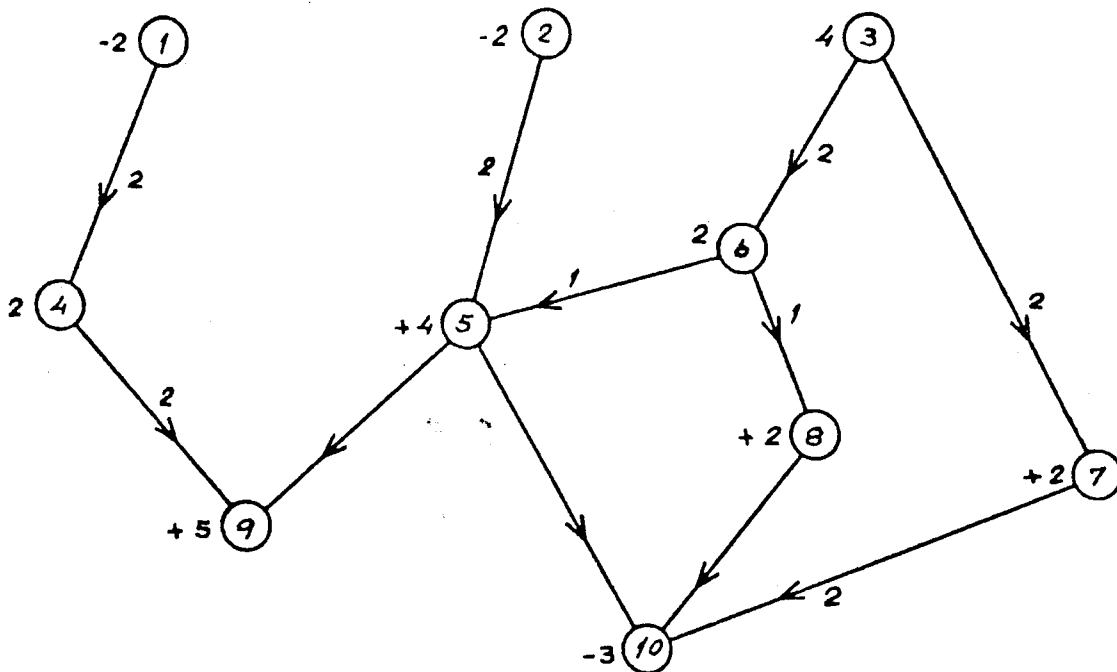
We shall observe the convention that whenever the maximum value of an interval is assigned it will be affected by a plus sign, and that whenever a minimum value is attributed a minus sign will be used; if the assigned value is simultaneously a minimum and a maximum, both signs will be utilized.

Furthermore, integer values will always be used since, if it need be, all dimension values may be multiplied by a positive integer, so as to enlarge the precision by as much as desired. Thus, whenever the need arises to augment or to diminish the dimension of a node or edge, that can be carried out by a succession of positive or negative unit increments. That way, the various unit increments may be achieved through different paths on the subgraph, and the maximum dimensional flexibility available, at the particular scale being used, is profited from. Moreover, the process may start by using a gross scale, so as to converge more rapidly to a solution region, and change to a finer scale if the need ever arises.

Observing the previous figure, note that the distribution of the dimension of the "entry" nodes 1, 2, and 3, by its efferent edges has already been made. In the case of node 3, the distribution was carried out proportionally to the dimensions assigned to nodes 6 and 7, since these are the nodes to which its efferent edges lead. This is the distribution strategy we adopt, keeping in mind that we must choose the closest integers to the real proportion values, and that, of course, their sum must equal the node dimension being distributed; besides that, a minimum of one unit must always be assured for each edge or, in general, whatever minimum value it tolerates. Recall that in this example the minimum is one unit and the maximum unlimited, for every edge.

Since a dimension has already been consigned to each of the edges affixed to nodes 4, 6, and 7, we are now in a position to compare, for those nodes, the sum of all dimension values "entering" them, with their previously assigned dimension:

1) Thus, the total dimension entering node 4, which is 2, coincides with the value 2 of its assigned dimension, making it unnecessary to effect any adjustment. On the other hand, the distribution of its dimension is made totally to node 9; this fact is indicated by inscribing the value 2 next to the edge from 4 to 9, in the subgraph below.



2) The total dimension entering node 6 coincides with its assigned dimension, and only its distribution by nodes 5 and 8 remains to be done. Now, when the dimension of a node is equal to the number of its efferent edges, there can be no question of proportionate distribution: it becomes necessary to give each edge a unit of dimension. Such is the case with node 6, as shown in the above subgraph.

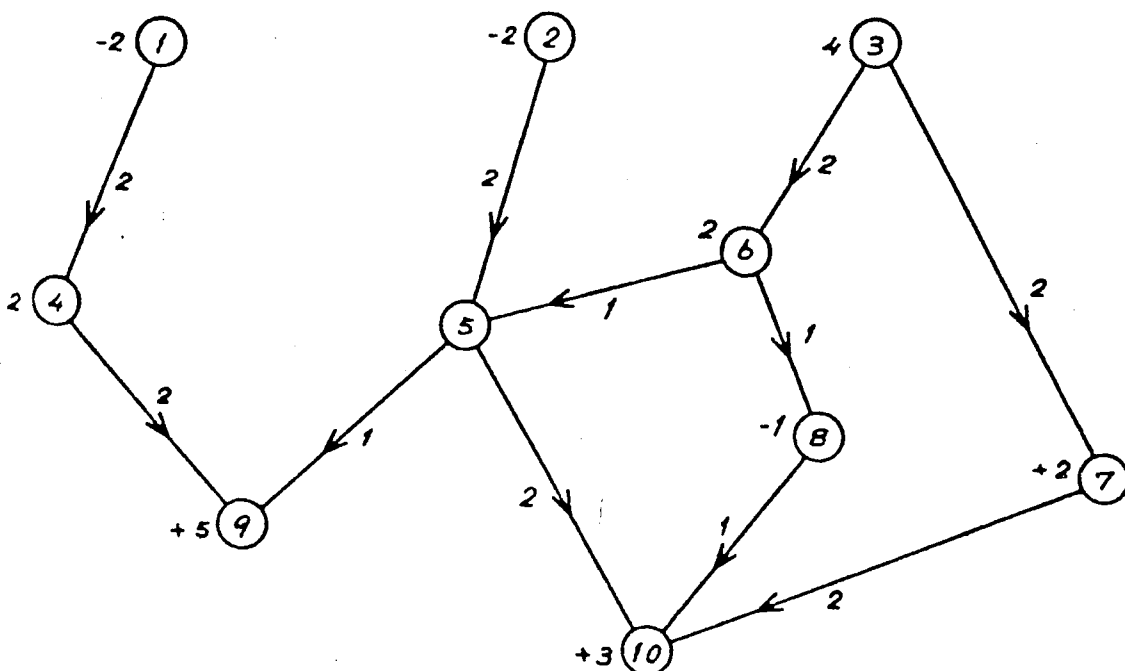
If it happens that the number of efferent edges of a node is greater than the available dimension at the node, it becomes necessary to augment its dimension by using a procedure expounded further on, where the situa-

tion arises of having to modify a node dimension.

3) Node 7 receives a total dimension of 2, equal to its assigned one, which is given out entirely through its only efferent edge.

The candidate nodes for dimensional conservation analysis are, at this stage, nodes 5 and 8, since all their afferent edges have had the dimension they convey specified:

1) The total dimension entering node 5, being 3, is less than its presently assigned dimension of 4. This makes us consider if it may virtually be reduced to 3. The answer is affirmative, upon looking up the tolerance intervals table. The value 3 is then assigned to node 5 and inscribed next to it, as shown in the subgraph that follows. On the other hand, the dimension 3 of node 5 is distributed, after ponderation, by nodes 9 and 10; to node 9 two units are given out, to node 10 the remaining ones.



2) In relation to node 8, its assigned dimension can be diminished to 1, since that is permitted by the tolerance table; furthermore, it is wholly distributed to its only out-going edge. If it had happened that the dimension of node 8 could not be reduced, a dimension increment to be conveyed by its only afferent edge would have to be obtained, using the convenient procedure for that purpose to be explained further on.

The fact, however, that its dimension is affected by a plus sign, is a guarantee that it can diminish at least one unit.

The above figure was obtained after nodes 5 and 8 were processed. Nodes 9 and 10 have now become apt for consideration:

1) Node 9 may have its dimension permissibly altered to 3, thus equalizing the total sum conveyed by its afferent edges. No distribution is made since it is an "exit" node.

2) Now, node 10 may only permissibly augment one unit, to the maximum value of 4, as expressed in the tolerance table. Thus, what has to be attempted is to reduce to 4 the sum of the edges afferent to node 10, which at this stage is 5, by reducing by one unit the value of one of those edges.

To make systematic the search for that unit, which has to be subtracted from a node having an afferent edge to node 10, we stipulate that the search for the appropriate edge will always be carried out clockwise around the node, either when "going up", i.e. in opposite sense of an edge's orientation, or "going down", i.e. following the arrows. Furthermore, those edges

will not be considered which lead to a node affected by (+) or (+), when "going up" looking for an extra dimensional unit, or "going down" trying to distribute an extra unit, in the first search mode; this mode is characterized by maintaining the sense of the search, relative to the edge orientation. Similarly, in the first mode, those edges are not considered which lead to a node affected by (-) or (+), when "going up" trying to get rid of an extra unit, or when "going down" trying to retrieve a unit given out.

When an appropriate edge cannot be found using the first search mode, the dimension finding process switches to the second search mode, and vice-versa. This mode is characterized by changing the sense of the search at each node; of course, every time the search sense is changed, the sign of the increment we were looking for also changes. For example, if we were trying to find an extra unit "going up" to a node, we will be looking, after having changed the search sense at that node, for a way to retrieve an extra unit given out from that node so that the original extra unit we were searching for can be found at the node.

Whereas we call this second search mode the reflexive mode, we call the first search mode the traversing mode, since in the former the nodes are used to reverse the sense search, and in the latter the nodes are traversed so as to maintain the search sense.

Let us next exemplify the traversing search mode by continuing the analysis of the subgraph we have been considering.

Since we have a unit too many entering node 10, we start by inspecting the first edge in the clockwise sense, of the set of its afferent edges as well as the node it links to. It is clear that that edge, as well as node 5.

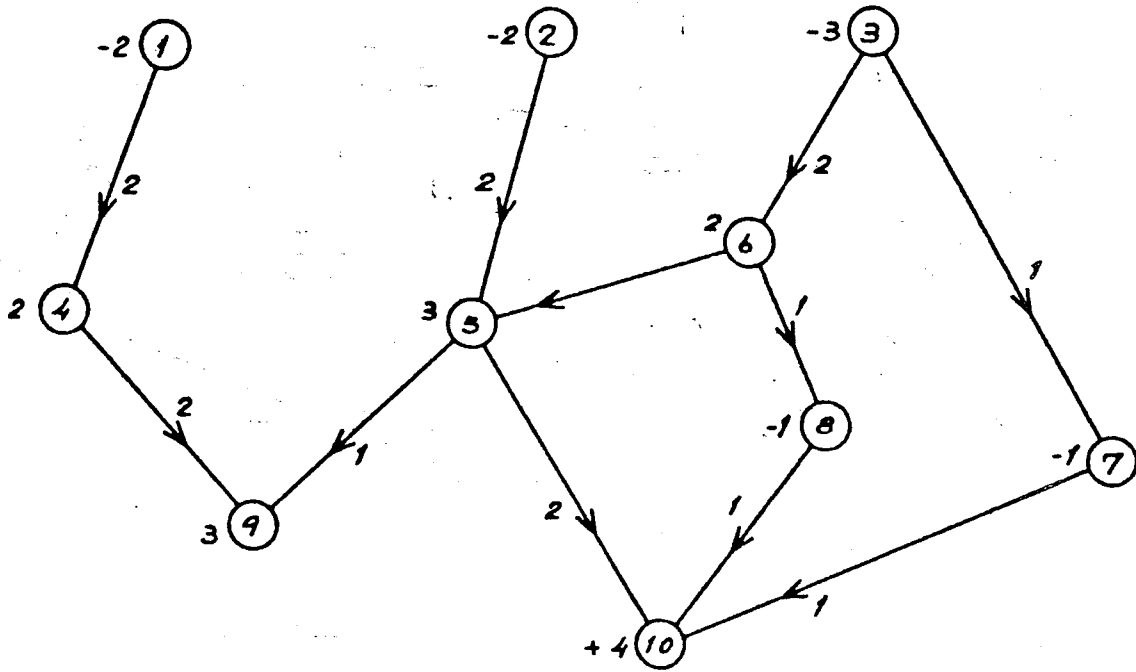
its extremity, may support the loss of one unit. We now have to examine if any of the afferent edges of node 5 may diminish the same one unit, as well as the node it links to. We are thus faced with an iterative process that will only come to a halt either if no complacent edge or node is found, or when an entering node is reached, since in this case we are "going up" and using only the traversing mode.

After inspection of the afferent edges of node 5, we verify that node 2 cannot have its dimension reduced, and that the edge linking to node 6 is at its minimum value of 1. It follows that we must give up the possibility of getting rid of the extra unit of node 10, through node 5, by means of the traversing search mode.

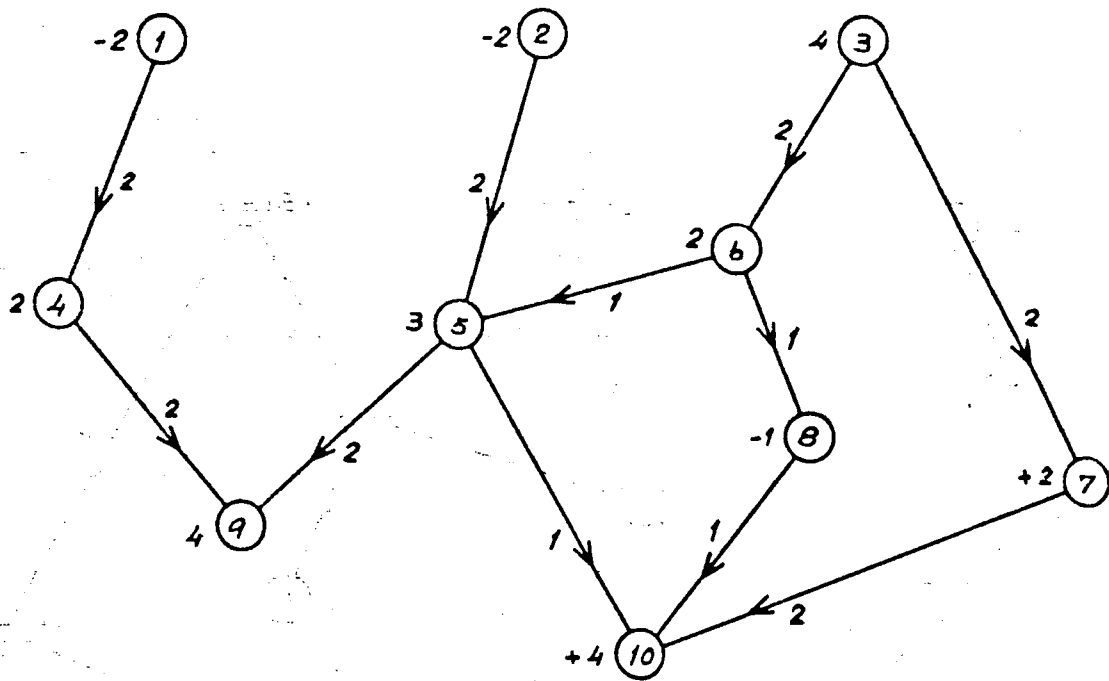
According to the clockwise sense around node 10, we are now reduced to exploring the edge to node 7 since the edge leading to node 8 is at its minimum. Upon inspection, we conclude that node 7 may diminish as well as entry node 3, from which emerges the only afferent edge to node 7. Furthermore, the two edges (10, 7) and (7, 3) are agreeable with diminishing both one unit.

This way, we have reached the end of our search for adequate horizontal dimensions for all nodes and edges of the realization submitted initially, since there are no more nodes to inspect, subsequent to the processing of node 10.

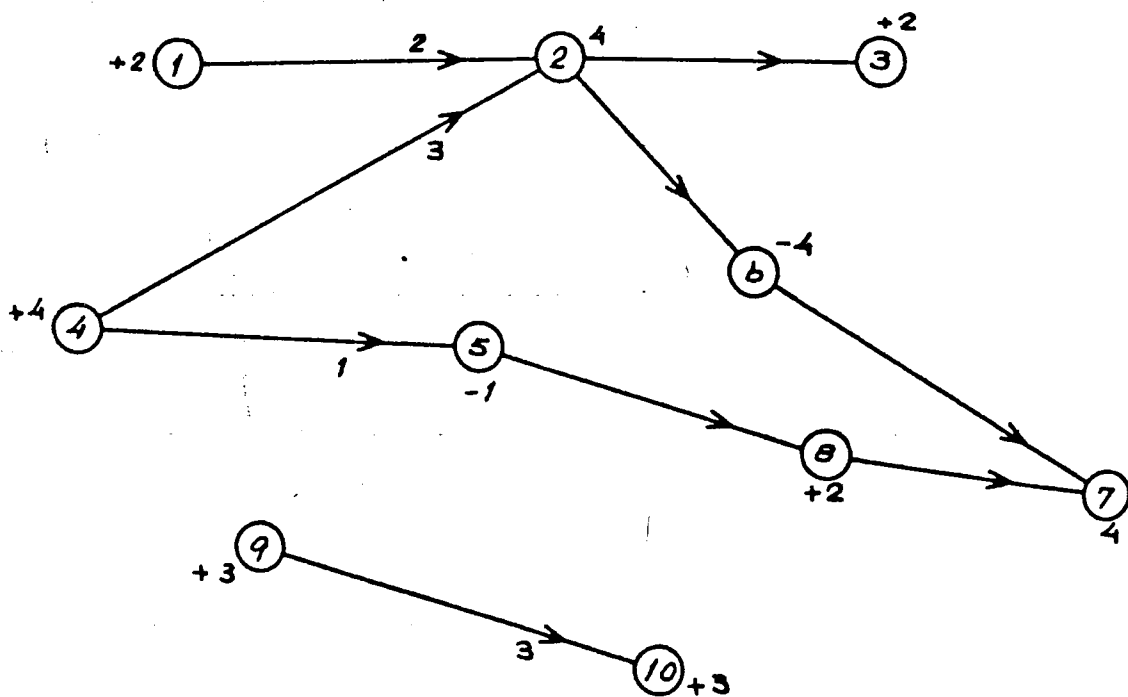
The final dimensional equilibrium reached is shown by the horizontal dimension subgraph below. This equilibrium represents one possible solution for the horizontal dimensions of all the nodes of the realization, which takes into account the dimension constraints expressed by all the relevant adjacencies, as well as by the given list of tolerance intervals.



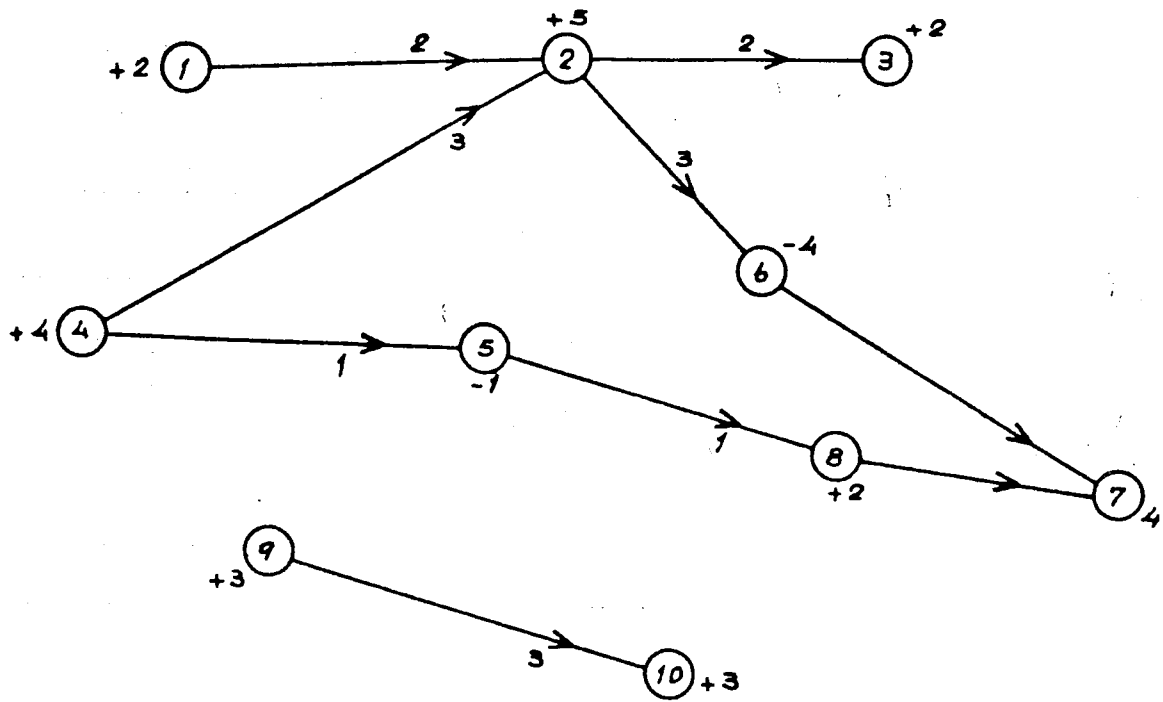
In the previous example, instead of trying to traverse node 5, we could have experimented the reflexive search mode, by considering if any of its efferent edges other than edge (5,10) could have its dimension augmented, such that the edge (5,10) could be diminished the desired unit increment, all the while keeping constant the dimension of node (5). Edge (5,9) may in fact support an extra unit as well as node 9, without violating any of the limits imposed. That way, we would have obtained another permissible solution, shown below, where both search modes were used.



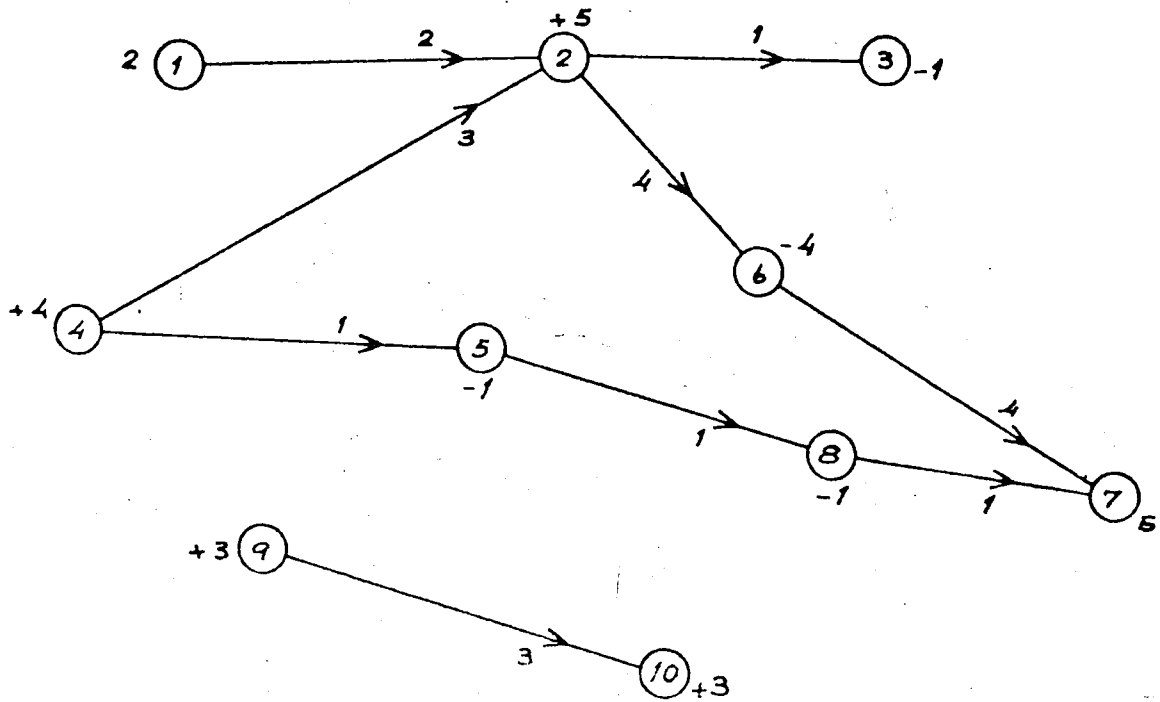
Before we comment further on both of the search modes, let us consider the vertical subgraph relative to the given realization, and the three stages, shown below, needed to achieved a permissible equilibrium for the vertical dimensions.



Stage one



Stage two

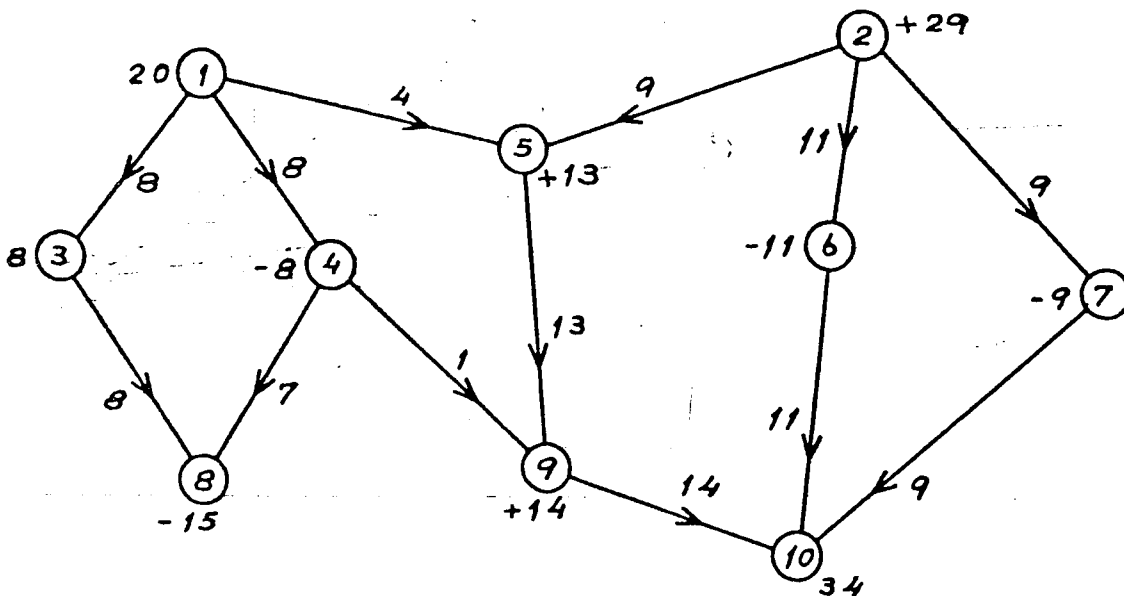


Stage three

The initial values are exhibited in the first diagram, where entry nodes 1, 4, and 9, have already been processed. In the second diagram, nodes 2, 5, and 10, have been dealt with, and nodes 3, 6, and 8, are being considered. Notice that the reflexive search mode had next to be utilized to deal with node 6, thus subtracting a unit increment to exit node 3, as shown in the last of three diagrams. Had it not been used, the distribution of vertical dimension throughout the subgraph would have been impossible to achieve.

We refrain from drawing the dimensional layout scheme corresponding to the dimensional solutions of the two complementary subgraphs, since the process of carrying it out has already been explained and exemplified in the last chapter.

We will next consider the following subgraph, relative to one of the dimensions of a given realization, and where a permissible overall distribution of dimension is already shown.



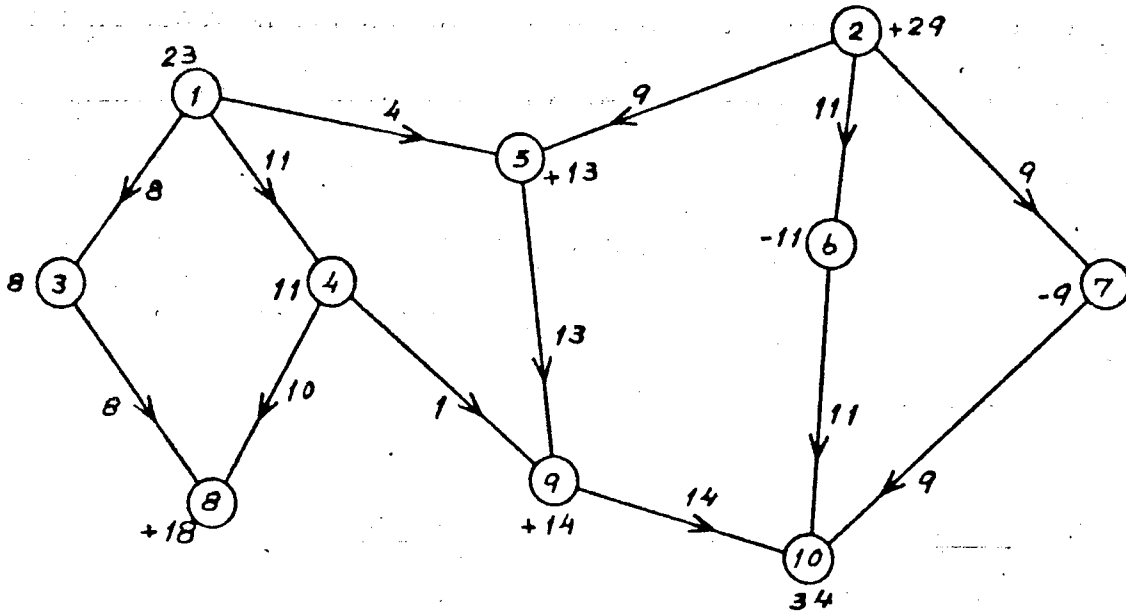
Our aim will be to increase as much as possible the dimension of node 4, without, nevertheless, trespassing any of the dimensional bounds expressed by the tolerance table presented below, with no constraint on the edges except that a minimum of a unit of dimension is required for each of them. By carrying this out, we will show how the two search modes interact, and also how the process can be used for user/machine interaction. Note that the process that will be used to alter the given solution makes use of the same search modes as the process for finding one, being actually a part of it.

<u>Node</u>	<u>Tolerance</u>
1	15-25
2	22-29
3	7-9
4	undefined
5	10-13
6	11-14
7	9-12
8	15-18
9	9-14
10	33-36

We will next begin to augment the dimension of node 4 one unit each time. Notice that to augment node 4 one must augment both the sum of its afferent and of its efferent edges.

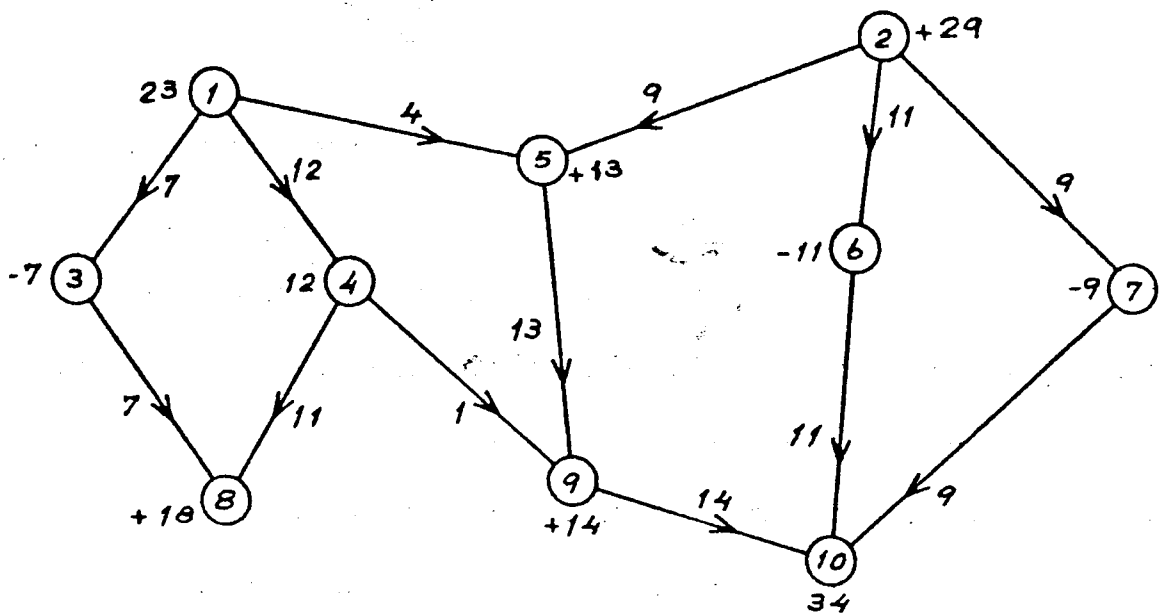
Now, using the traversing mode, and since node 1 as well as node 8 may go up at least one unit because neither of them carries a plus sign, node 4 can step up to 9, while nodes 1 and 3, as well as their edges leading to node 4, all step up one unit.

Since the same routine can be repeated two more times, we are now left with the following dimensional subgraph:



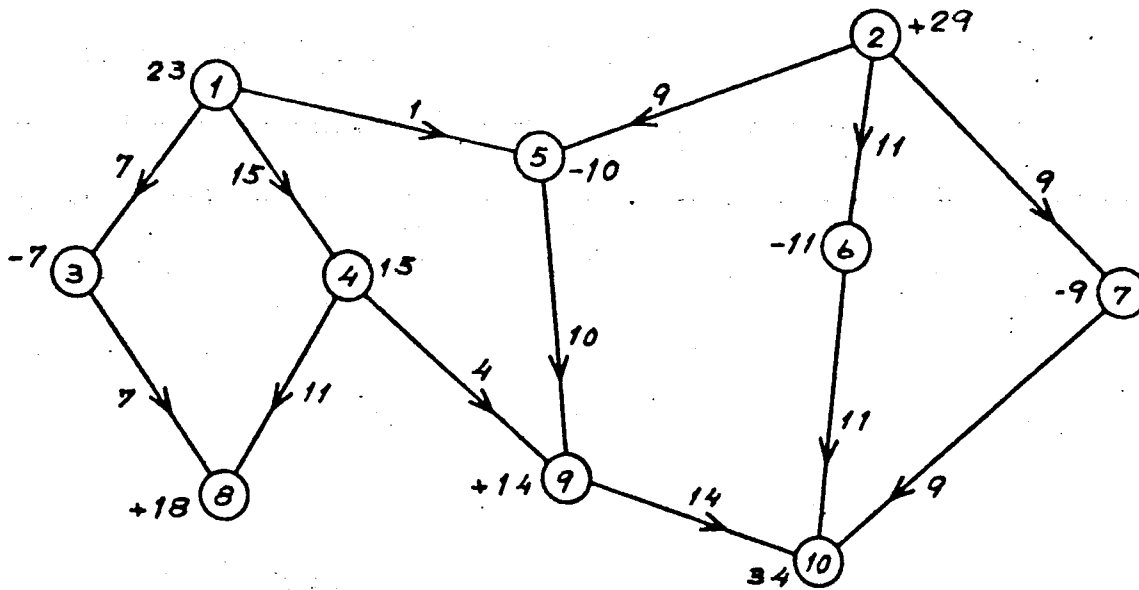
Now, because nodes 8 and 9 cannot go up anymore, we must resort to the reflexive mode for searching down from node 4. Taking edge (4, 8) first, instead of following the clockwise sense (since we are going to explore all possibilities it does not really matter), we reflect at node 8 and reach node 3. On account of the reflexion, the unit increment we are looking for changes its sign, and we are up to trying to diminish node 3 by one unit, along with entry node 1. It so happens that node 3 can step down a unit, along with entry node 1. We have this way completed the path that started down from node 4. Let us now look for a path starting up from that node that permits us to augment the sum of its afferent edges, so as to equal the sum of its efferent edges. It is clear that entry node 1 can augment one unit, thus returning to the value it had after having gone down one unit when the previous path was considered.

The two paths, 4-8-3-1 and 4-1, are an example of the formation of a circular path which means that a local compromise of dimensions was reached; notice that although node 1 is an entry node, it has not changed the in-coming dimension it conveys, in the new equilibrium found by virtue of the circular path 4-8-3-1-4. Now the same circular path cannot be used again since node 3 has reached its minimum. The stage arrived at is illustrated in the following diagram:



The next positive unitary increment of node 4 will be obtained, in a similar way, by reflection on node 9 followed by the traversing of node 3, and a reflection at node 1 back to node 4. The circular path, 4-9-5-1-4, thus formed, is composed of path 4-9-5-1, which starts down from node 4, and of path 4-1, which starts up from node 4. Since the circular path can still be used twice, at which occasion both node 5 and edge (5, 1)

reach their minimum possible value, we obtain the diagram below.



It is easily verified in the above diagram that the dimension of node 4 cannot possibly be further augmented; although node 1 has not reached its maximum, all paths starting downwards from node 4 are cut off from the possibility of reaching an entry or exit node, and from forming a circular path. Indeed, nodes 8 and 9 are at their maximum thus preventing the traversing mode, and nodes 3 and 5 are at their minimum thus preventing the reflexive mode since the only possible reflections are from nodes 8 and 9 to nodes 3 and 5, respectively.

Informal comments on the distribution of dimensions.

1) Since the impossibility of increment of a node always corresponds to the "saturation" of a node and/or edge for every conceivable path starting in one same or in both senses from that node (before the path reaches an entry or exit node or circulates to the starting node), it can easily be seen that the impossibility of further increment for a given node is always de-

etermined by the same set of "saturated" nodes and/or edges; to do so, it suffices to deny this hypothesis, thus reaching the absurd conclusion that there is at least one conceivable path that can become "saturated" at different nodes and/or edges than the ones found.

2) Similarly, the maximum dimension that can be reached permissibly by a node is always the same, independently of the sequence of paths used for carrying out unit increments. Note, however, that different final dimension distributions may be obtained, except for the values of the node in question and the "saturation" nodes and/or edges.

3) The various final distributions will be the end result of diverse sequences of paths chosen for unitary incrementation. On the other hand, one same set of paths will always have the same final distribution as an outcome, whatever the particular sequence of paths belonging to that set is chosen, provided each of the paths is really used the same number of times and that there does exist one sequence made up from the paths in the set which gives a permissible solution for the distribution of dimension.

4) It is not difficult to recognize that the two search modes are complementary, and that together they exhaust all possible positive or negative unit increments that may be obtained.

5) In general, when a unit increment is sought for a node, two paths have to be found: one which starts with an afferent edge, and one which starts with an efferent edge of that node, in case both types of edges exist otherwise, one or both of them will be the trivial path consisting only of that node.

Now, each of the two paths must reach an entry or an exit node, which can either be incremented or which the other path also reaches, unless the two paths meet before. In any circumstance, if the two paths meet, the visibility of obtaining a unit increment by means of the circular path formed will only depend upon the possibility of the node where they meet to be incremented, if the two paths arrive at the node with different senses; otherwise, it does not depend upon that node, since its dimension will remain the same.

6) If a circular path is found for obtaining a unit increment, we will speak of local compromise of dimensions, since the overall dimension of the layout is not altered by such a path; what happens in that case is that some of the partition segments are being moved parallel to themselves, thus changing their meeting points with other partition segments.

On the other hand, if no circular path is obtained, we will speak of a global adjustment of dimensions, since the overall dimension of the layout will be changed by the paths found. In that case some partition segments may also be moved, namely those which correspond to a reflexion at a node, and only those. This rule concerning the partition segments that are moved applies also to the case of a circular path.

7) Admitting that for each node and each dimension, a tolerance interval is specified by the user, a criterium for a good solution may be the one of obtaining the least number of nodes with an extreme dimension, either minimum or a maximum.

Another criterium might be the one of minimizing the sum of the squares of the difference between the dimensions of each node and the mean

value of their corresponding tolerance intervals.

Since, according to Hypothesis 4, an excess integer may be specified for the node dimensions, there may be found distributions of dimension which do not respect the tolerance intervals, in the case no solution respecting them exists. Criteria similar to the above ones may then be applied to compare such "compromise solutions" as to their relative desirability.

8) Initially, one may start by assigning to the nodes' dimensions the mean (integer) values of their tolerance intervals, with the hope that the final distribution of dimensions does not deviate much from those values; that is why the distribution of dimensions is made proportionately to the dimensions assigned, at any given stage.

In the absence of other criteria, such assignment of initial values is also made with the hope that that way a small number of searches for incremental units will need to be made.

In the case some nodes present extreme values, it may be tried to alter them by the methods expounded, such that they may lead to a better solution according to the above criteria. Of course, such modifications may be carried out automatically by the program and/or left to be commanded from outside, through the interaction of user and program. Through such interaction, moreover, the user may explore all sorts of heuristic rules for carrying out and modifying the distribution of dimensions.

9) Another way the program could act, if appropriately modified, would be the following. Since all solutions depend closely upon the initial values given, it would compute a solution starting with all the minimum values of the tolerance table, and another one starting with the maximum values. Now, it

can be readily recognized that the mean of the values of any two solutions is yet another solution, with values in between the values of the two original ones. Thus, the "mean value solution" of the two solutions obtained by starting with the minimum and the maximum values might hopefully be a better solution than any of the two, according to the criteria mentioned above.

10) As it is, the program starts with initial values proportional to the values obtained by first computing a modular dimensional solution. This is yet another heuristic rule for possibly obtaining a good solution fast.

11) The introduction of additional restrictions such as minimum and maximum values for the areas, gives rise to problems of interaction between the two dimensional subgraphs which we have not dealt with. Perhaps a first step in that direction would be to impose a linear relation between the two dimensions of a node, having both a minimum and a maximum value.

12) In its present form, the program developed makes a first check on the tolerance intervals given, with respect to the adjacencies to be obeyed, to ascertain that the minimum value permissible for each node is not greater than the sum of the maximum values permitted for the nodes adjacent to it, in any of the two subgraphs. Similarly, it also checks if the maximum value permitted for each node is not smaller than the sum of the minimum values of the nodes adjacent to it, in any of the two subgraphs. These two checks are made for every node because if any of them were to fail a solution would clearly be impossible. Of course, if an excess integer is also given, as foreseen by Hypothesis 4, it is taken into account when these checks are made.

We next make an abstract formulation of the problems we have been dealing with in this chapter, and of the methods used for their resolution.

3. FORMAL PRESENTATION OF THE PROBLEM OF DIMENSION DISTRIBUTION AND OF METHODS FOR ITS RESOLUTION

Let us start by restating the problem in question.

Restatement of the problem

Given a finite oriented graph with no loops, and given also for each node and each interval of permissible values, a distribution of dimensions to the set of edges is procured, such that, for every node, the sum of dimensions of the edges entering the node is equal to the sum of dimensions of its outgoing edges, while the total dimension at each node and edge remains within the limits individually prescribed.

To solve the problem thus formulated, a class of heuristic algorithms was defined, all of which can find the required compatible

distribution, if it exists. The algorithms of that class allow, furthermore, to modify, consistently, a given solution, so as to obtain one other compatible distribution with chosen characteristics, whenever such a distribution exists as a solution.

The choice of a particular algorithm from the class defined can moreover be effected in compliance with the specific characteristics of each problem formulation, so as to increase the efficiency of solution search. Furthermore, partial problem solutions may be offered in the case of non-existence of global solutions, indication being given on the reasons for the impossibility of a complete solution, and provision being made for user-machine interaction to the effect of partial reformulation of problem stipulations.

Abstract Problem Formulation

Let there be:

- 1) a non-empty finite set of nodes X .
- 2) a non-reflexive relation $G \subseteq X \times X$, such that $(x,y) \in G$ implies that there is no sequence x_0, \dots, x_n where $x_0 = y$, $x_n = x$ and $(x_{i-1}, x_i) \in G$ for all $i = 1, \dots, n$ (in particular $(y,x) \notin G$), i.e. G is an oriented graph with no loops.
- 3) a set L of intervals on the non-negative integers, indexed by X , where $L_x \in L$ is given by

$$L_x = [\min_x, \max_x].$$

Remark : The set of integer intervals E , relative to the edges, is considered in this first formulation to have only elements of the form $E_{x,y} = [1, \infty]$ that is, there is no upper limit on the restrictions on the edges. This restriction is not present in the computer programme developed.

Define :

1) a generating function on X , relative to G , $\Gamma_G : X \rightarrow 2^X$, expressed intensionally as :

$$\Gamma_G = \lambda x \{y \mid (x,y) \in G\}. (+)$$

2) a dual generating function on X , relative to G , $\Gamma_G^* : X \rightarrow 2^X$, expressed intensionally as :

$$\Gamma_G^* = \lambda x \{y \mid (y,x) \in G\}.$$

3) a set S_G of starting nodes :

$$S_G = \{x \mid \Gamma_G^*(x) = \emptyset\}.$$

4) a set T_G of terminal nodes :

$$T_G = \{x \mid \Gamma_G(x) = \emptyset\}.$$

Definition 1. A distribution D in G , is an assignment $D : G \rightarrow N$, from G to the non-negative integers N . The afferent value V_G^* of x in G , is then defined as :

$$V_G^*(x) = \sum_{y \in \Gamma_G^*(x)} D(y,x), \text{ for all } x \in X - S_G.$$

The afferent difference δ_G^* of x in G , between $V_G^*(x)$ and L_x takes

(+) This notation means that $\Gamma_G(x) = \{y \mid (x,y) \in G\}$; it is called the lambda notation.

the values :

zero, if $V_G^*(x) \in L_x$.

$V_G^*(x) - \max_x$, if $V_G^*(x) > \max_x$.

$V_G^*(x) - \min_x$, if $V_G^*(x) < \min_x$.

The efferent value V_G of x in G , is then defined as:

$$V_G(x) = \sum_{y \in \Gamma_G(x)} D(x,y), \text{ for all } x \in X - T_G.$$

The efferent difference δ_G of x in G , between $V_G(x)$ and L_x , is defined similarly to the afferent difference.

Definition 2. The distribution problem for G consists in searching for a distribution D in G such that it meets the following conditions:

- 1) $V_G^*(x) \in L_x$, for all $x \in X - S_G$.
- 2) $V_G(x) \in L_x$, for all $x \in X - T_G$.
- 3) $V_G^*(x) = V_G(x)$, for all $x \in X - (S_G \cup T_G)$.

Such a D is called a solution to the distribution problem in G , or so-
lution for G . When the above conditions are satisfied, the final value re-
lative to solution D in G $F_G(x)$ of x becomes defined for all $x \in X$, by
making :

$$F_G(x) = V_G(x) \text{ if } x \in X - T_G, \text{ and}$$

$$F_G(x) = V_G^*(x) \text{ if } x \in X - S_G.$$

Definition 3. A subset $Y \subseteq X$ gives rise to a relation $G' \subseteq Y \times Y \subseteq X \times X$, defined by the restriction of G to $Y \times Y$; i.e. :

$$(x,y) \in G' \iff (x,y) \in G \text{ and } x, y \in Y.$$

G' is then called a full subgraph of G . Such a G' has a generating function $\Gamma_{G'}$, and a dual generating function $\Gamma_{G'}^*$, which are the restrictions to Y of Γ_G and Γ_G^* , respectively :

$$\Gamma_{G'}(x) = \Gamma_G(x) \cap Y, \text{ for all } x \in Y.$$

$$\Gamma_{G'}^*(x) = \Gamma_G^*(x) \cap Y, \text{ for all } x \in Y.$$

$S_{G'}$ and $T_{G'}$ are defined for G' in the same manner as previously for G .

Definition 4. $G' \subseteq G$ is closed for G if and only if it is a full subgraph of G , and furthermore:

1) $\Gamma_{G'}$ satisfies :

$$\Gamma_{G'}(x) = \Gamma_G(x) \text{ for all } x \in Y - T_{G'}; \text{ or, equivalently,}$$

$$\Gamma_G(x) \subseteq Y \text{ for all } x \in Y - T_{G'}.$$

2) $\Gamma_{G'}^*$ satisfies :

$$\Gamma_{G'}^*(x) = \Gamma_G^*(x) \text{ for all } x \in Y - S_{G'}; \text{ or, equivalently,}$$

$$\Gamma_G^*(x) \subseteq Y \text{ for all } x \in Y - S_{G'}.$$

Definition 5. The distribution problem for $G' \subseteq G$ is a distribution subproblem for G if and only if G' is closed for G . G' is also said to be a subproblem for G.

Definition 6. A distribution D' in $G' \subseteq G$ is a partial distribution in G if the distribution problem for G' is a distribution subproblem for G .

Definition 7. A partial solution D' to the distribution problem for G is any solution D' of the distribution problem for $G' \subseteq G$ which is a distribution subproblem for G . D' is also said to be a partial solution for G.

Of course, a solution D for G is also a partial solution for G .

Definition 8. A heuristic production rule in G (HPR) is any rule for changing a partial solution D' for G in G' , into a partial solution D'' for G in G'' , if $G' \subseteq G'' \subseteq G$.

Definition 9. A heuristic algorithmic resolution (HAR) for a distribution problem in G is any finite sequence Σ :

$$\Sigma = (D'_1, D'_2, \dots, D'_j, \dots, D'_n)$$

of partial solutions for G , such that:

- 1) D'_n is a solution for G .
- 2) for $i = 1, \dots, n - 1$,

if D'_i is a solution for G'_i and D'_{i+1} is a solution for G'_{i+1} ,
then $G'_i \subseteq G'_{i+1}$.

3) for $i = 1, \dots, n - 1$,

D'_{i+1} is obtainable from D'_i through a finite number of applications
of the rules making up the set H of heuristic production rules in G .

Σ is called a resolution for the distribution problem in G , or simply
a resolution for G . Any non-empty subsequence Σ' of Σ is called
a partial resolution for the distribution problem in G , or simply a partial
resolution for G .

A Heuristic Algorithmic Resolution for the Distribution Problem .

Next we give (without further proof except that the programme developed works), one possible way of obtaining a HAR, if it exists, for a distribution problem in G . A set H of HPRs is indicated for such a HAR.

I - First, start with $G'_1 = \emptyset$ and

$$D'_1 = \emptyset \rightarrow N.$$

II - Apply $HPR1 \in H$, given below, once.

III - Apply $HPR2 \in H$, given below, a finite number of times, possibly none

IV - Repeat II and III in sequence any (finite) number of times until at least one solution for G is obtained, or until FAILURE is found.

Purpose of HPR1 and HPR2.

The purpose of HPR1 is to transform a partial solution D'_i into a partial solution D'_{i+1} such that, whenever possible, $G'_i \subset G'_{i+1}$ the condition $G'_i \subset G'_{i+1}$, is always possible if the problem has a solution and if $G'_i \subset G$.

The purpose of HPR2 is to transform a partial solution D'_i (which is eventually a solution) into a preferred partial solution D'_j such that $G'_i = G'_j$, if D'_j exists. If it does not exist, D'_i is maintained. Note that HPR2 is not indispensable, and the reason why it is considered is to propiciate eventual interaction of the user with the heuristic algorithm so that outside heuristics can be made effective by commanding the programme to carry out virtual alterations on a partial solution or solution already obtained.

Description of HPR1 .

To apply HPR1 proceed as follows:

1) if applying HPR1 for the first time, i.e. at step II of HAR, then:

Form sets $OPEN = S_G$ and $CLOSED = \emptyset$.

2) Consider all $x \notin (OPEN \cup CLOSED)$ and compute the set ADD given by:

$$ADD = \left\{ x \mid \prod_G^* (x) \subseteq (OPEN \cup CLOSED) \right\} .$$

3) put members of OPEN in set APRAISE.

4) put members of ADD in OPEN.

5) define

$$G' = \{ (\text{OPEN} \cup \text{APRAISE} \cup \text{CLOSED}) \times (\text{OPEN} \cup \text{APRAISE} \cup \text{CLOSED}) \} \cap G.$$

6) apply procedure CONCILIATE, defined below, to APRAISE. If SUCCESS is obtained, members of APRAISE have been put in CLOSED ; proceed to 7). If FAILURE is obtained, exit with NO SOLUTION EXISTS.

7) apply procedure DISTRIBUTE, defined below, to G' and D' obtained in 6).

Each time HPRI is applied successfully, a subproblem G'' of G has been solved, with

$$G'' = (\text{CLOSED} \times \text{CLOSED}) \cap G, \quad S_{G''} = S_G, \text{ and}$$

$T_{G''} = \text{APRAISE}$, and a partial solution D'' is added to a partial resolution Σ' for G.

Definition of procedure CONCILIATE.

a) if $\text{CLOSED} = \emptyset$, choose for each $x \in \text{APRAISE}$ a value from L_x , and exit to HPRI with SUCCESS.

b) if $\text{APRAISE} = \emptyset$, exit to HPRI with SUCCESS.

c) if $\text{APRAISE} \neq \emptyset$, pick any $x \in \text{APRAISE}$.

d) if $\delta_{G'}^*(x) = 0$, take x from APRAISE and put it in CLOSED ; go to b).

e) if $\delta_{G'}^*(x) \neq 0$, make $u = 1$ if greater than zero, else make $u = -1$.

f) search for a sequence of distinct elements

$$G = (x = c_0, c_1, c_2, \dots, c_n)$$

such that :

i) $c_1, \dots, c_{n-1} \in \text{CLOSED}$.

ii) $c_n \in S_{G'} \cup T_{G'}$.

iii) There correspond to it two sequences of length n . One, where each element is any one of the two generating function symbols $\Gamma_{G'}$ and $\Gamma_{G'}^*$.

$$\gamma = (\gamma_0, \gamma_1, \dots, \gamma_{n-1}).$$

Another, whose elements take the values u or, $-u$:

$$\omega = (u_1, u_2, \dots, u_n),$$

such that these sequences obey the next conditions.

iv) $c_1 \in \gamma_0(x)$ and γ_0 is $\Gamma_{G'}^*$.

v) $c_{i+1} \in \gamma_i(c_i)$ for $i = 1, \dots, n$.

vi) for $i = 1, \dots, n$:

$$u_i = u \quad \text{if } \gamma_{i-1} = \gamma_0 \quad \text{and } \gamma_i = \gamma_{i-1},$$

$$u_i = -u \quad \text{if } \gamma_{i-1} \neq \gamma_0 \quad \text{and } \gamma_i = \gamma_{i-1},$$

$$u_i = 0 \quad \text{if } \gamma_i \neq \gamma_{i-1}.$$

vii) for $i = 1, \dots, n-1$,

$$F_{G'}(c_i) + u_i \in L_{c_i}, \text{ and}$$

$$F_{G'}(c_n) + u_n \in L_{c_n} \text{ if } c_n \in \text{CLOSED}.$$

If no such sequences can be found exit to HPRI with FAILURE, if not, proceed to g).

g) Add u_i to $F_{G'}(c_i)$ for $i = 1, \dots, n-1$.

Subtract u from $\delta_{G'}^*(x)$, and, if $c_n \in \text{CLOSED}$, add u_n to $F_{G'}(c_n)$. If not, add u_n to $V_{G'}^*(c_n)$.

h) for $i = 0, \dots, n-1$, add u to $D'(c_i, c_{i+1})$ if $\gamma_i = \gamma_0$, add $-u$ if

$$\gamma_i \neq \gamma_0.$$

i) return to d).

Definition of procedure DISTRIEUTE.

a) for each $x \in \text{CLOSED}$, for every $Y \notin \text{CLOSED}$ such that $Y \in \square_{G'}(x)$, make $D'(x, Y)$ equal to any integer value belonging to L_Y , on condition that

$$\sum_{Z \in \square_{G'}(x)} D'(x, Z) = F_{G'}(x).$$

One possible, and perhaps propitious, way of doing it, i.e., choosing integer values for the several $D'(x, Y)$ relative to a given x , is to compute, for each one

$F_{G'}(x, Y)$, defined by:

$$p_{G'}(x,y) = \frac{\min_y + \max_y}{\sum_z (\min_z + \max_z)} (F_{G'}(x) - \sum_w D'(x,w))$$

where $z \in \square_{G'}(x)$ and $z \notin \text{CLOSED}$, and where $w \in \square_{G'}(x)$ and $w \in \text{CLOSED}$, and to make each $D'(x,y)$ equal to an integer value as near as possible to $p_{G'}(x,y)$ but subject to the condition expressed in a) for x .

Description of HPR2.

The application of HPR2 is always posterior to at least one application of HPR1, and may follow immediately or only after one or more previous applications of HPR2 itself. Since HPR1, if successful, always leaves us with some partial solution D' for a subproblem $G' \subseteq G$ of the form

$$G' = (\text{CLOSED} \times \text{CLOSED}) \cap G,$$

and since HPR2 does not alter G' but possibly only D' , changing it to a distribution D'' which is also a solution for G' since $G'' = G'$, on the occasion of applying HPR2 we are always confronted with a certain D'' which is a solution for a subproblem $G'' \subseteq G$. How can we alter D'' and still have a solution for G'' ?

One possible way is now given. It is shown how to alter one given node x by an amount $\varepsilon \neq 0$, which is an integer obeying to:

$$F_{G''}(x) + \varepsilon \in L_x.$$

Simultaneously, we can consider that L_x has also been changed, or

that it has remained the same. This gives us the liberty to alter the constraints imposed through the set of intervals L .

a) if $\varepsilon > 0$ make $u=1$, otherwise make $u=-1$.

b) Search for two sequences, each of them made from elements which are distinct from other elements in the sequence. Moreover, the first elements of each sequence are the same, and possibly also the last, all other elements of any of the sequences, being distinct.

The two sequences,

$$s = (x = s_0, s_1, s_2, \dots, s_n)$$

and

$$t = (x = s_0 = t_0, t_1, t_2, \dots, t_m)$$

are such that :

i) $s_0, s_1, \dots, s_n, t_1, t_2, \dots, t_m \in \text{CLOSED}$.

ii) $s_n = t_m$ or (inclusive or) $s_n, t_m \in S_{G'} \cup T_{G'}$.

iii) there correspond to each sequence two other sequences with length L equal to the length of the corresponding sequence plus one ;

i.e. $L = n + 1$ or $L = m + 1$.

One of the two sequences is made up of elements that are any of the two generating function symbols, $\square_{G'}$ and $\square_{G'}^*$. There are two of them:

$$\gamma = (\gamma_0, \gamma_1, \dots, \gamma_n)$$

and

$$\alpha = (\alpha_0, \alpha_1, \dots, \alpha_m)$$

The other sequence, takes the values u or $-u$ for its elements. There are also two of them:

$$w = (u_1, u_2, \dots, u_{n+1})$$

and

$$r = (r_1, r_2, \dots, r_{m+1}).$$

These two sequences are such that they obey the conditions which follow.

iv) $s_1 \in \gamma_0(x)$ and γ_0 is $\Gamma_{G^{11}}^*$.

$$t_1 \in \alpha_0(x) \text{ and } \alpha_0 \text{ is } \Gamma_{G^n}.$$

v) $s_{i+1} \in \gamma_i(s_i)$ for $i = 1, \dots, n$.

$$t_{i+1} \in \alpha_i(t_i) \text{ for } i = 1, \dots, m,$$

vi) for $i = 1, \dots, n$,

$$u_i = u \text{ if } \gamma_{i-1} = \gamma_0 \text{ and } \gamma_i = \gamma_{i-1},$$

$$u_i = -u \text{ if } \gamma_{i-1} \neq \gamma_0 \text{ and } \gamma_i = \gamma_{i-1},$$

$$u_i = 0 \text{ if } \gamma_i \neq \gamma_{i-1}.$$

for $i = 1, \dots, m$,

$$r_i = u \text{ if } \alpha_{i-1} = \alpha_0 \text{ and } \alpha_i = \alpha_{i-1},$$

$$r_i = -u \text{ if } \alpha_{i-1} \neq \alpha_0 \text{ and } \alpha_i = \alpha_{i-1},$$

$$r_i = 0 \text{ if } \alpha_i \neq \alpha_{i-1}.$$

vii) for $i = 1, \dots, n-1$,

$F_{G''}(s_i) + u_i \in L_{s_i}$ and

$F_{G''}(s_n) + u_n \in L_{s_n}$ if $s_n \neq s_0$.

for $i = 1, \dots, m-1$,

$F_{G''}(t_i) + r_i \in L_{t_i}$ and

$F_{G''}(t_m) + r_m \in L_{t_m}$ if $t_m \neq t_0$.

If no such sequences can be found exit from HPR2 with FAILURE, if not, proceed to g).

g) for $i = 1, \dots, n-1$ add u_i to $F_{G''}(s_i)$.

for $i = 1, \dots, m-1$ add r_i to $F_{G''}(t_i)$.

if $s_n \neq t_m$, add u_n to $F_{G''}(s_n)$

and r_m to $F_{G''}(t_m)$, otherwise add u_n to $F_{G''}(s_n)$ if $u_n = r_m$.

h) for $i = 1, \dots, n-1$,

add u to $D''(s_i, s_{i+1})$ if $\delta_i = \delta_0$,

add $-u$ if $\delta_i \neq \delta_0$.

for $i = 1, \dots, m-1$,

add u to $D''(t_i, t_{i+1})$ if $\alpha_i = \alpha_0$,

add $-u$ if $\alpha_i \neq \alpha_0$.

1) subtract u from \mathcal{E} and, if $\mathcal{E} \neq 0$, return to a).

2) Apply procedure DISTRIBUTE to G'' and D'' .

CHAPTER 8

HOW TO OBTAIN THE PLANAR REALIZATIONS OF A GRAPH AND MODIFY THEM TO MEET CONDITIONS 'A'

1. INTRODUCTION

In this chapter we shall be concerned with obtaining all possible planar realizations of a graph G given by the list understood in Hypothesis 2, and with modifying each of them so as to make it comply to conditions 'A'.

Thus, we will be searching for the solutions of:

(1) Subproblem 1.6: Given G , find out whether or not it is connected and non-separable; if not, how to alter it as to make it connected and non-separable.

(2) Subproblem 1.1: Given G , find out whether or not it is planar; in particular, only connected and non-separable graphs will have to be examined.

(3) Subproblem 1.3: If G is not planar, how to alter it to make it planar.

(4) Subproblem 1.2: If G is planar, find all its planar realizations; in particular, only connected and non-separable graphs will necessarily have to be considered. Furthermore, only those realizations obeying Property 2 of chapter four will necessarily have to be generated.

(5) Subproblem 1.7: Given the realizations of G found by solving the previous subproblems, modify each such realization so as to make it comply

with conditions A1 and A2, and with the following additional hypothesis:

Hypothesis 9: It may be given a list of non-adjacencies, in the form of pairs (a, b) of spaces, interpreted as meaning that spaces a and b may not be adjacent; i.e. pair (a, b) of the list must not be present in the list of Hypothesis 2, nor can it be added at any time to the list of adjacencies of graph G when it suffers modifications.

Now, paragraphs (1), (2), and (3), aim at reconciling G with condition A0, whereas paragraph (5) aims at securing that conditions A1 and A2 are met. Paragraph (4) pertains directly to the solution of the Initial Problem; it may be further divided into two subproblems:

(a) Subproblem 1.2.1: Given graph G, connected, non-separable, and planar, find one of its planar realizations R.

(b) Subproblem 1.2.2: Given an R obtained by solving the previous subproblem, find all the planar realizations of G; in particular, only those which obey Property 2 will necessarily have to be found, since the others cannot give rise to any layout scheme.

2. FURTHER PROBLEM DECOMPOSITION

Consider the following definition:

A graph G is 3-connected if, for any two subgraphs H and K of G such that

- 1) $H \cup K = G$;
- 2) $V(H \cap K)$ is a set, $\{u, v\}$, of two nodes;
- 3) $E(H \cap K) = \emptyset$,

either H or K are segments.

Now let V and E be the number of nodes and edges, respectively of graph G.

Theorem . If G is 3-connected and planar, the number of different planar realizations of G is given by $E-V+2$ (1).

Consider a planar realization R of G . The number of its faces, according to Euler's theorem, is $E-V+1$, since there is only one connected component. Taking into account the external face of G , then the total number of faces of G is $E-V+2$. Now, the external face corresponds to the boundary of the given realization R of G .

Consider any of the remaining faces of G in R . For each of them, there exists a planar realization of G obtained by assigning to the interior of that face all nodes and edges of G not belonging to the polygon which corresponds to that face, in such a way that all faces remain the same, except that the face considered now becomes the external face and the boundary of G in R becomes a face of the new realization.

Intuitively, from a given realization of G all other realizations are obtained by putting inside each face all other faces, thus transforming it into the external face. The number of different realizations is then $F=E-V+2$, where F is also the sum of the faces of G in R plus its external face.

It is easily recognized that the faces and external face of a given realization R of G define a planar mesh of G . In fact, each edge belongs exactly to two of those faces and, furthermore, any cycle of G can be obtained by the sum of those of such faces which are "interior" to the cycle.

What is more, the realizations of G are easily obtained by considering each of the faces of G in R as the contour in each of the remaining

realizations of G .

If G is not 3-connected, however, the obtention of all its planar realizations is not as simple; thus we shall divide Subproblem 1.2.2 into two others:

(i) Subproblem 1.2.2.1: Given an R obtained by solving Subproblem 1.2.1 for a 3-connected G , obtain all its planar realizations; in particular, only those which obey Property 2 will necessarily have to be found, as argued before.

This subproblem is solved as indicated above, but without considering those realizations having an external face with just three edges; i.e. those resulting from the triangular faces of R .

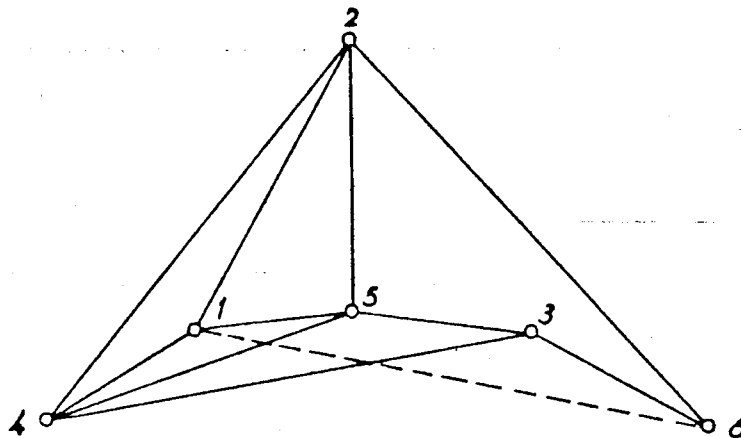
(ii) Subproblem 1.2.2.2: Given an R obtained by solving Subproblem 1.2.1 for a G which is not 3-connected, obtain all its planar realizations; in particular only those obeying Property 2 will be of interest, as argued before.

The solution of this subproblem will be considered further on.

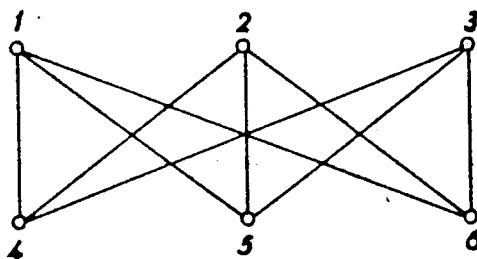
1. EXAMPLES

In this section we illustrate some of the above problems by means of examples.

I) Consider the following realization.



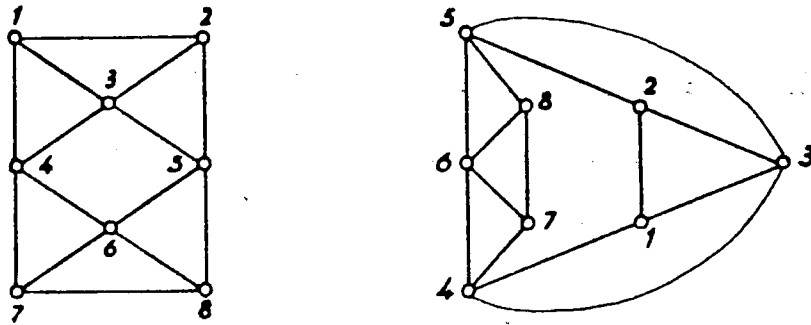
Now, taking the dashed edge into account, this realization could not be planar since it is a realization of a non-planar graph; the graph is not planar because it has the following Kuratowski graph as a subgraph:



The given graph is obviously connected. With the dashed edge present it is also non-separable (or 2-connected). Without the dashed edge it becomes planar, all the while remaining non-separable, as can be easily seen.

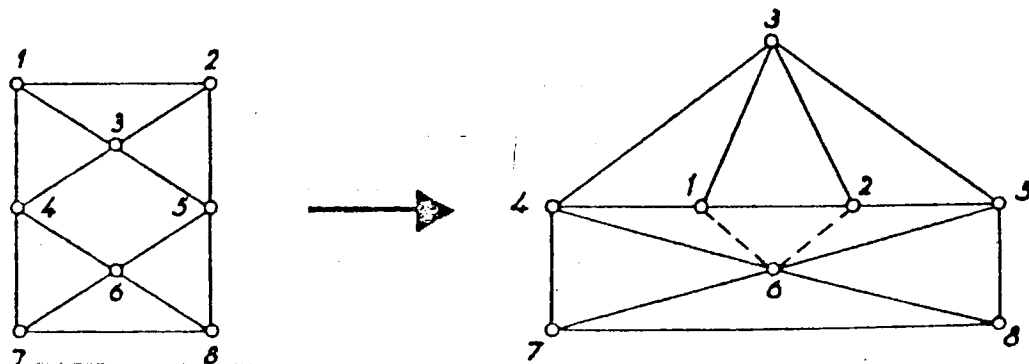
This example illustrates Subproblems 1.6, 1.1 and 1.3.

II) Consider the two following planar realizations of the same graph.



The one on the right was obtained from the one on the left by putting the whole graph inside face $(3, 4, 6, 8)$ of the realization on the left, thus turning it into the external face.

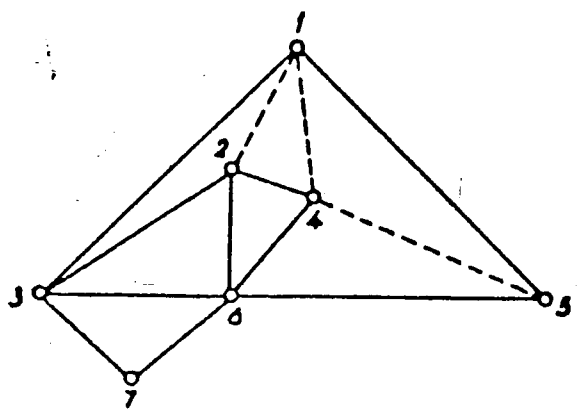
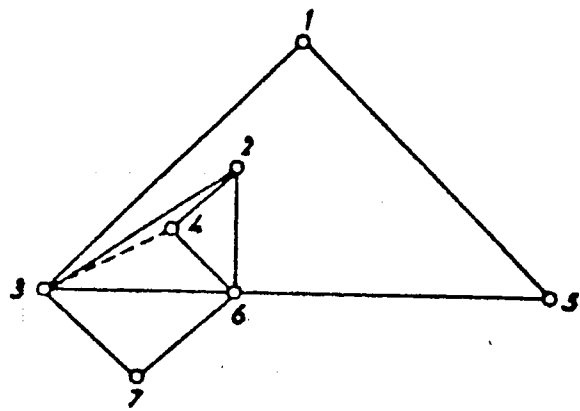
But from the realization on the left one may also obtain the next realization shown, where cycle $(3, 5, 8, 7, 4)$ of the realization on the left was turned into the external face of the new realization (for the moment do not consider its dashed edges). Note that cycle $(3, 5, 8, 7, 4)$ was not a face of the realization on the left. This can happen because the given graph is not 3-connected: by removing from it nodes 4 and 5 it becomes a disconnected graph with two components, both with cycles.

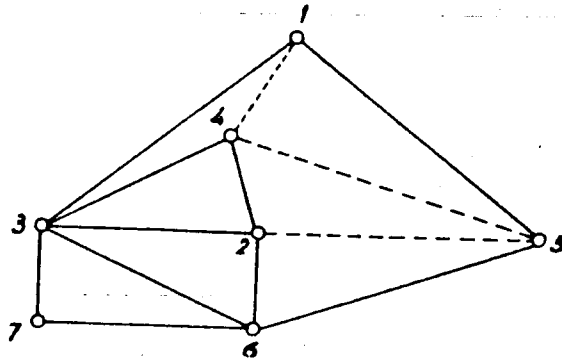


Now, if the last realization obtained is to obey conditions A1 and A2, we will have to add edges to nodes 1 and 2, and divide face (1, 2, 5, 6, 4) into triangles and/or quadrangles. The only way to do both things, however, turns out to be by creating the two dashed edges shown.

This example illustrates Subproblems 1.2.2 and 1.7. The list understood in Hypothesis 9 is considered to be the empty list.

III) Consider next the first two of following diagrams. They are two planar realizations of the same graph. But, whereas in the first it is not possible to meet requirement A1 relatively to node 4 since only the dashed edge may added to it without violating the planarity of the realization or making the graph non-simple, in the second realization a possible way of complying with A1 and A2 is shown by the dashed edges added.

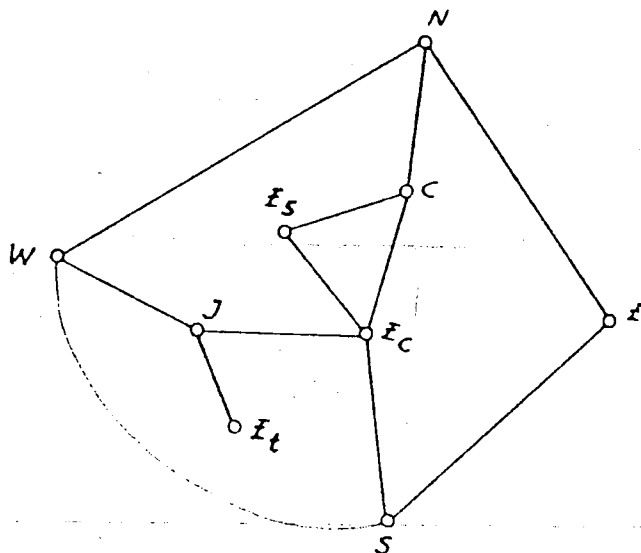




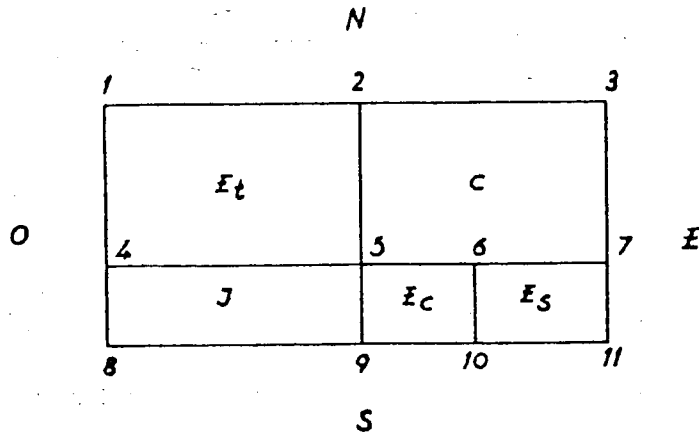
In the third figure edge (4, 6) has been deleted, corresponding to a redefinition, by the user, of the list of edges of Hypothesis 2. This way, the figure shows a viable completion of a planar realization of the new graph. Such completion would not have been possible were edge (4, 6) to persist. This third figure exemplifies a problem we leave to the user's attention and insight.

The other two figures illustrate again Subproblems 1.2.2 and 1.7.

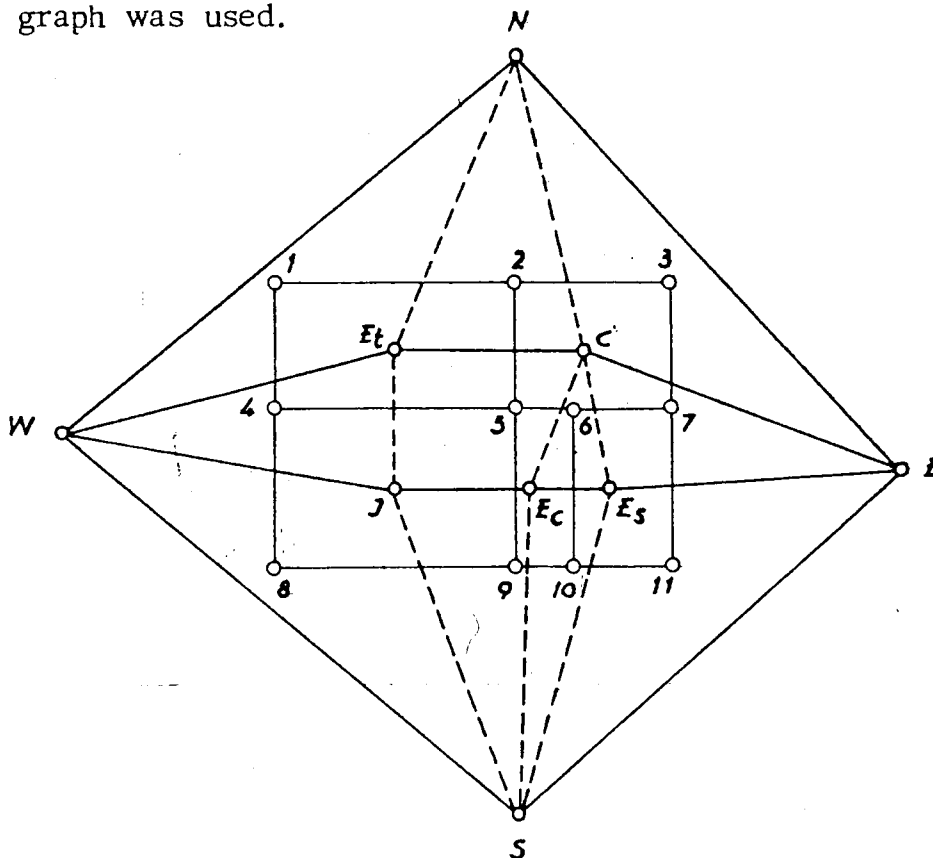
IV) Consider now the following planar realization of a graph with lettered nodes, where N, S, E, and W stand for "north", "south", "east" and "west", respectively.



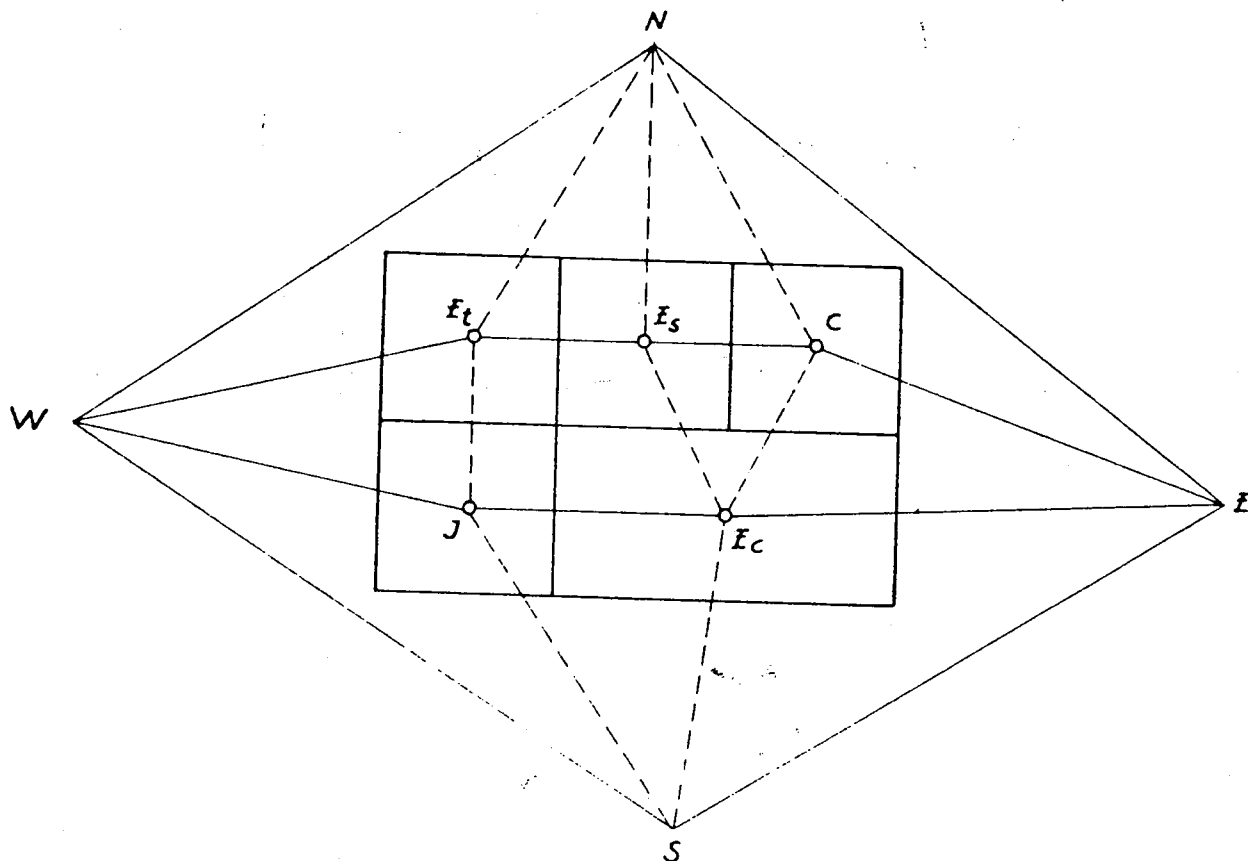
The layout scheme shown below respects all the imposed adjacencies; in it, partition segments between the four cardinal points have been abolished.



Of course, some adjacencies are present in this layout which were not imposed. These correspond to the dashed edges shown in the next figure, where the above layout and its dual realization are superimposed. From it we may observe that a realization different from the one used to specify the graph was used.

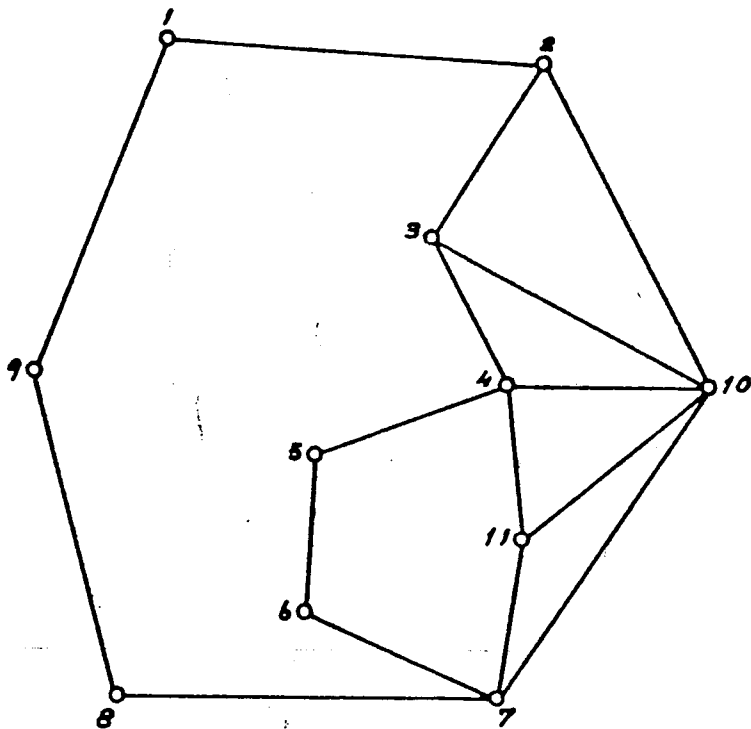
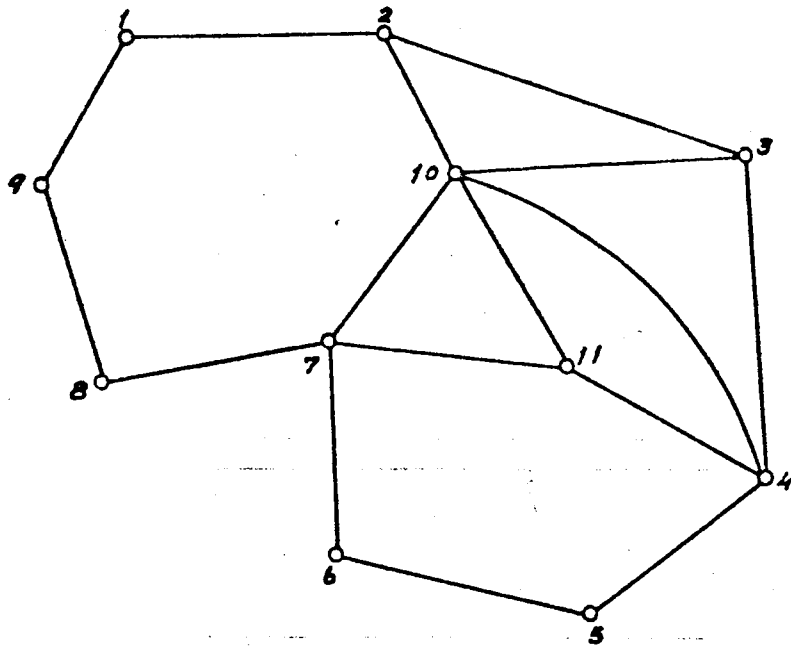


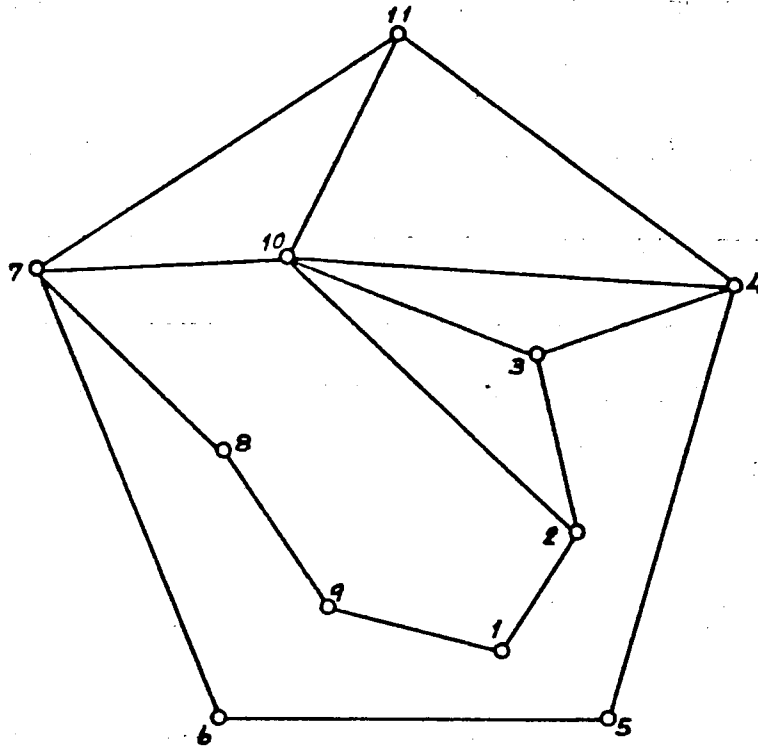
Yet another realization is shown next, where a viable completion of edges so as to meet conditions A1 and A2 is expressed by the dashed edges added.



This example illustrates how the various above subproblems interrelate, and what types of interventions the user may be confronted with, if the program is not to generate all possible solutions.

V) The following three figures are the only planar realizations of the 3-connected graph given by the adjacencies shown, which comply to Property 2. They correspond to consider as the external face, the contour cycle shown in each figure. Note that, according to Property 2, the boundary of a realization may not have only three edges if it has interior nodes.





This example illustrates Subproblem 1.2.2.1

4. RESOLUTION OF THE SUBPROBLEMS PRESENTED

In this section we indicate how the various subproblems arisen were dealt with, by using appropriate computer programs. As some of those programs were not developed by the present author and since the algorithms they execute belong more properly to graph theory, we will in some cases excuse ourselves from detailing their theoretical justification and their actual formulation.

(1) Subproblem 1.6 is solved by two programs: one which tests graph connectedness, and another which tests its separability. The information they output is conducive to the elimination both of unconnectedness and separability.

(2) Subproblems 1.1 and 1.3 are treated by means of a graph planarity program which finds out whether a graph is planar or not and, in the negative case, gives relevant information to change it into a planar graph. In the affirmative case, a planar realization for the graph is given in terms of a planar mesh.

(3) Thus, Subproblem 1.2.1 is also solved by the previous program.

(4) All the realizations of a planar graph are obtained by another program, which uses the planar realization given by the planarity program to compute the remaining ones. An outline the workings of this program, which really is part and parcel of the planarity program, will be delineated further on. Both the testing of graph planarity and the obtention of planar realizations will then be expounded.

(5) At present Subproblem 1.7 must be solved by the user himself. Of course, one could contrive a program to add missing edges in all conceivable ways so as to meet conditions A1 and A2, which would take into account Properties 1 to 5 of chapter four, so as to minimize subsequent pruning of alternatives. But to us, the issue here does not seem to be to generate all viable sets of additional edges, but to consider only those with a semantical significance. And at the moment, that can only be done by

the user himself, once he is confronted with the permissible planar realizations given by the planarity program. A fortiori, the question of edge deletion should also be left to the user's attention and insight; of course, the user will be helped in this task as in the previous one, by his knowledge of conditions A and B and their consequences.

We will return to this point in the chapter on future developments and lines of investigation.

(6) Subproblem 1.2.2.1 is also solved by the planarity program.

(7) Subproblem 1.2.2.2 is either solved by the planarity program treating the general case, or as indicated before, by relying on the theorem stated to that effect.

Remark about the main program. We remark that the program generating the PHV-assignments for a given realization, may test that realization in what regards conditions A, before engaging in the generation of assignments.

This means that when a realization is submitted to it, by means of the list of adjacencies understood in Hypothesis 2 and by giving the sequence of nodes around the contour, then, if that realization has not been previously processed by the above mentioned algorithms, the main program initially tests:

(a) graph connectedness;

(b) the planarity of the realization by identifying the faces of the given realization; this is accomplished by the program by "going around the contour" and "peeling off" the faces found; these faces are deleted, thereby

redefining the realization; the "peeling process" then continues until: 1) either one of Properties 1 to 5 of chapter four is infringed, or no more faces can be detected before Euler's condition, $F = E - V + 1$, is met, in which case the realization is rejected and relevant information is output regarding the reason(s) for failure, or, 2) the program accepts the given realization after having found all its faces according to Euler's formula; only then does it proceed with the assignment generation mechanisms.

5 THE PLANARITY AND PLANAR REALIZATIONS ALGORITHM: AN OUTLINE

Let G be a non separable graph, P a peripheral polygon of G , C the elementary cycle defined by P .

It may be shown that if G is planar, there is a planar mesh of G containing C .

Consider now the following splitting procedure, with respect to G and P :

Find a segment S of G , with end nodes u and v in P , determining in it segments R and T , such that

- (1) S avoids P ;
- (2) one of the segments, say T , determined by u and v in P is a branch graph.

It is easy to prove that such a segment always exists.

Let B_1, \dots, B_n be the bridges of the polygon $M = RUS$, other than T (which is a bridge by itself). If B_i has feet x_i and y_i such that x_i and y_i are interior nodes of R and S , respectively, it is then clear that B_i and

T overlap; call such a B_i an outer bridge. So if two outer bridges B_i and B_j overlap, G is non planar, ^{and also if} there are three mutually overlapping bridges, B_i, B_j, B_k ; in both cases stop.

If none of the above conditions apply, it is possible to find two disjoint subsets of the set of bridges $\{B_i\}, 1 \leq i \leq n$, such that:

- (1) if B_i and B_j belong to one of them, B_i and B_j avoid one another;
- (2) every outer bridge belongs to the same subset.

If there are no outer bridges, call inner part to one of these subsets and outer part to the other; else call outer part to the one which contains the outer bridges and inner part to the other. Furthermore, let the inner and outer parts be chosen in such a way that, if possible, the inner part is empty.

Now, if the inner part is empty, the polygon $N = TUS$ is clearly peripheral, and if G is planar, there is a planar mesh which contains both N and P .

Finally, call outer graph to the subgraph of G defined by M and the outer part, and inner graph to the subgraph defined by N and the inner part; one can see that M is peripheral in the outer graph and N is peripheral in the inner graph.

This splitting procedure is the core at the planarity algorithm, which works as follows:

Let G be a non separable graph.

(1) Choose an arbitrary polygon P_i of G .

(2) Determine the bridges B_1, \dots, B_n of P .

(3) If there are three mutually overlapping bridges, G is non planar, so stop; else it is possible to find a partition of $\{B_i\}, 1 \leq i \leq n$, into

two subsets such that any two bridges of the same subset avoid one another. Choose those subsets, H , K , in such a way that, if possible, one of them is empty.

(4) Let G_1 , G_2 be the subgraphs defined by P and H , and P and K , respectively.

(5) Consider the following recursive procedure, with arguments a graph X and a polygon Z peripheral in X : if $X = Z$, Z defines a cycle of the planar mesh to be generated; else apply the splitting procedure to Z , X and recursively apply the procedure twice:

the first time, with X the inner graph, and $Z = N$.

the second, with X the outer graph, and $Z = M$.

(6) Apply the procedure in (5) to G_1 , P , and to G_2 , P .

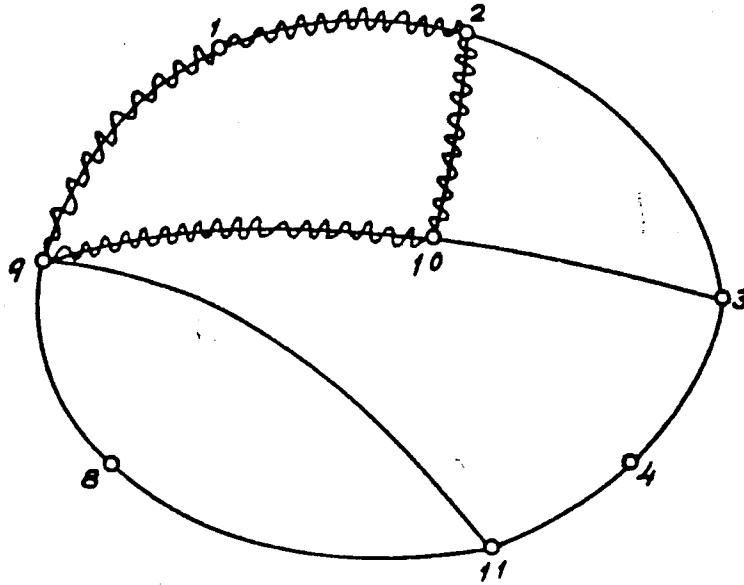
This ends the description of the planarity algorithm; note that the algorithm terminates either with a planar mesh for G or else with a statement of nonplanarity.

To obtain all the planar meshes for a given planar graph, it is sufficient to backtrack over all possible choices of M and N (if R and T are branch graphs) in the splitting procedure, and over all possible partitions of sets of bridges.

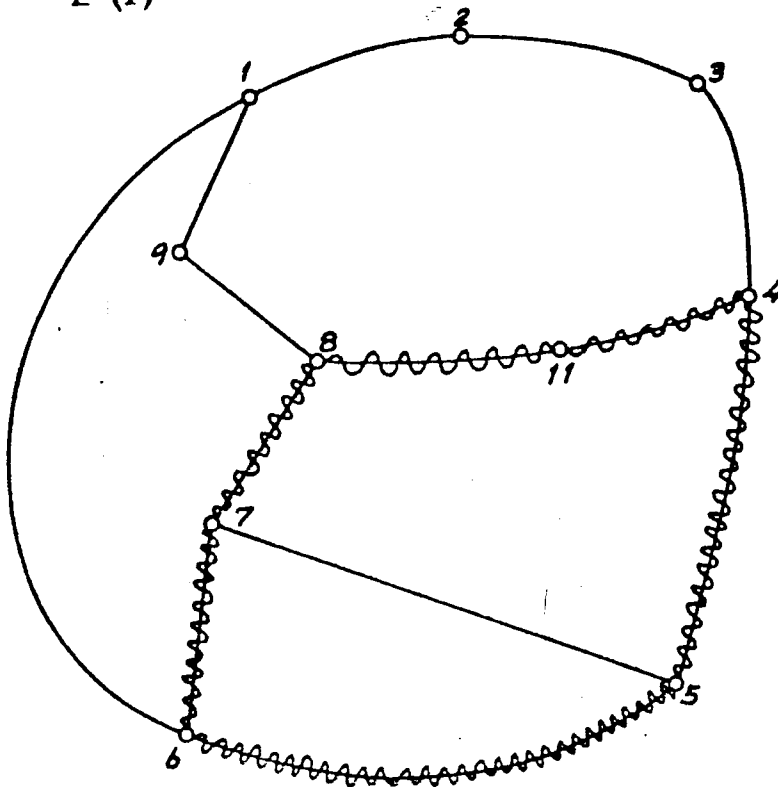
Since this graph is 3-connected, it has only one planar mesh; the number of planar realizations of the graph is thus equal to the number of cycles of its planar mesh, which is $E - V + 2$.

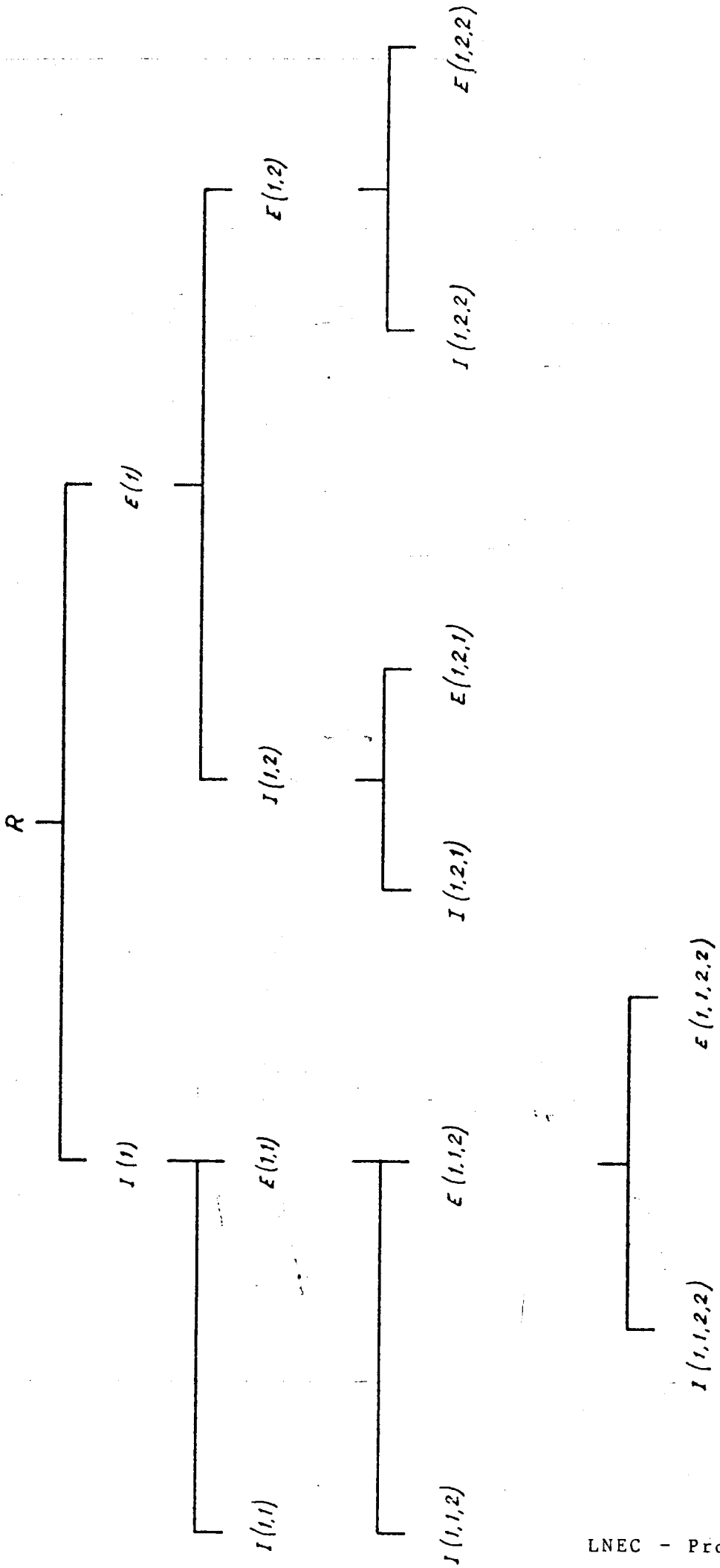
Let the polygon P in step (1) of the algorithm be the one defined by the nodes 1, 2, 3, 4, 11, 8 and 9. We write $E(1)$, $I(1)$ for the graphs G_2 , G_1 in step (4), and whenever in step (5) a subgraph $I(i_1, \dots, i_{n-1})$ (resp. $E(j_1, \dots, j_{n-1})$) splits, we call $I(i_1, \dots, i_{n-1}, 1)$ to the inner graph and $E(i_1, \dots, i_{n-1}, 1)$ to the outer graph (resp. $I(j_1, \dots, j_{n-1}, 2)$, $E(j_1, \dots, j_{n-1}, 2)$).

I (1)



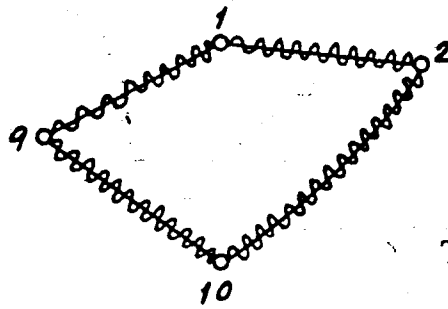
E (1)





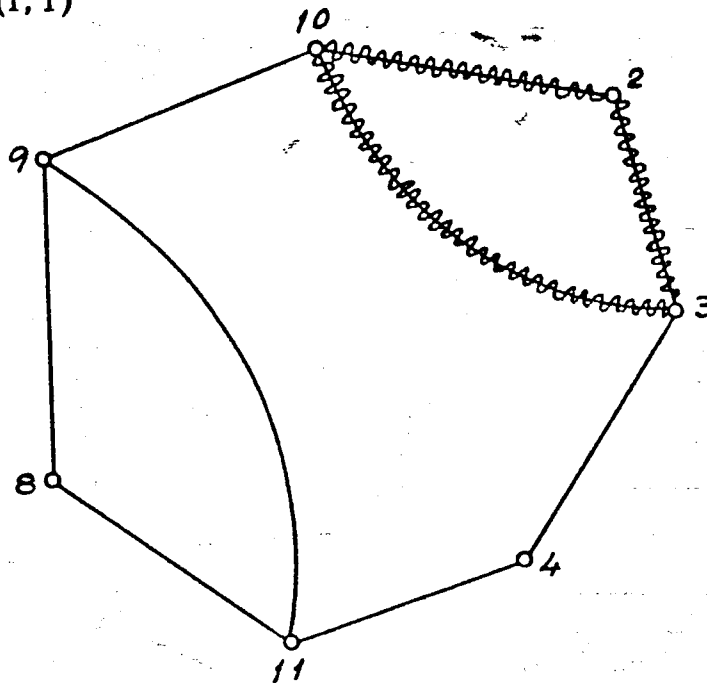
Now, $I(1)$ gives rise to the two following subgraphs and to the cycles shown in them, which were chosen according to the planarity algorithm.

$I(1,1)$



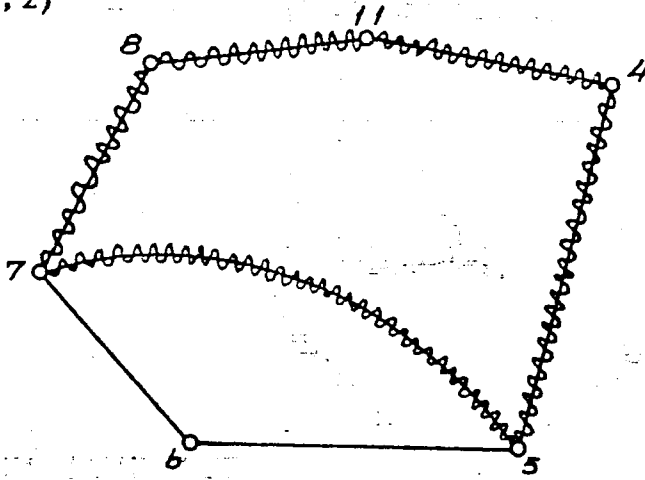
TERMINAL NODE

$E(1,1)$

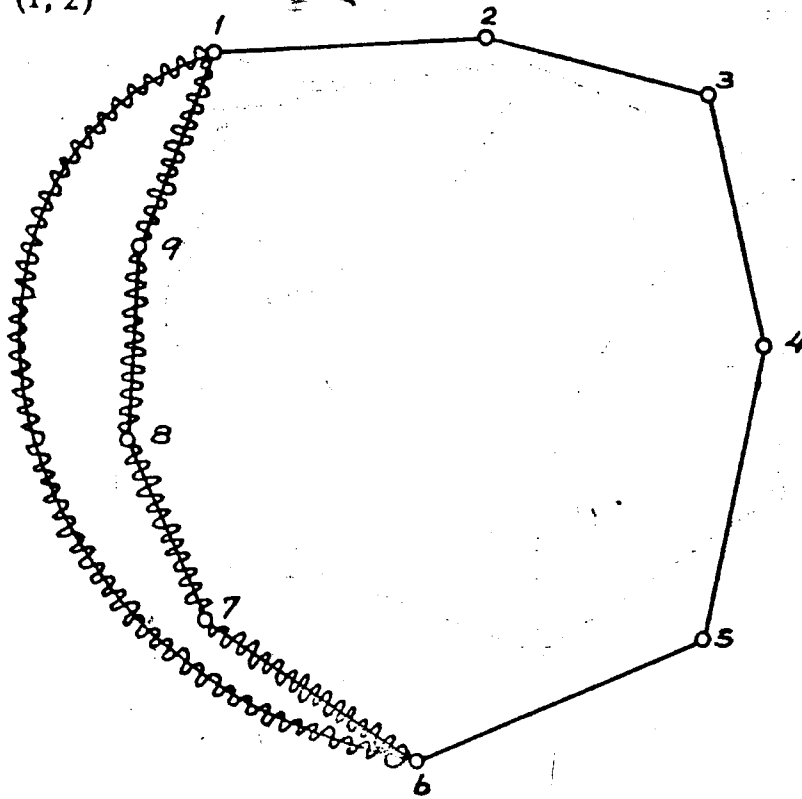


The algorithm continues as before, generating all the steps corresponding to the figures that follow.

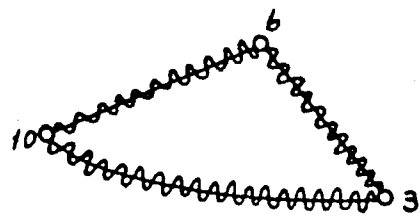
I (1, 2)



E (1, 2)

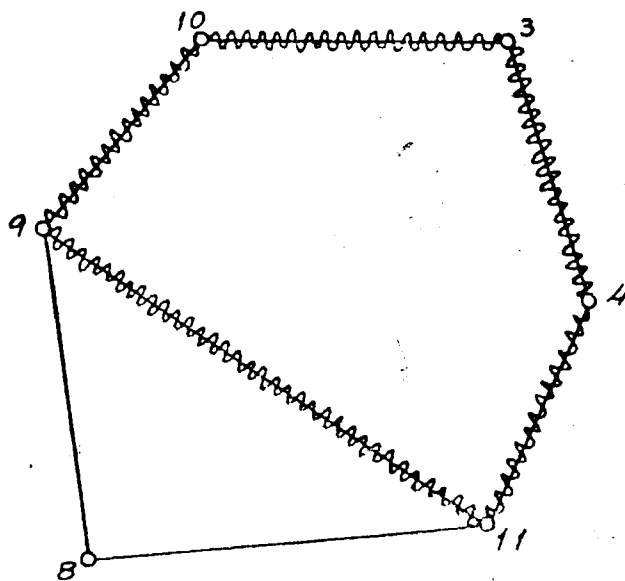


I (1, 1, 2)

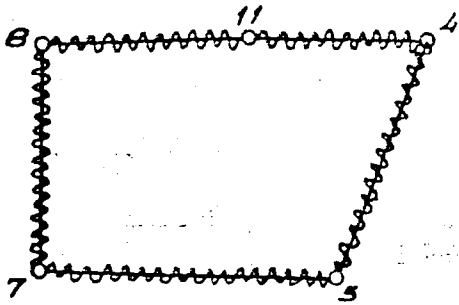


TERMINAL
NODE

E (1, 1, 2)

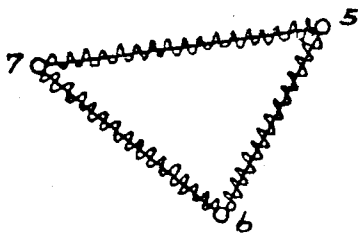


I (1, 2, 1)



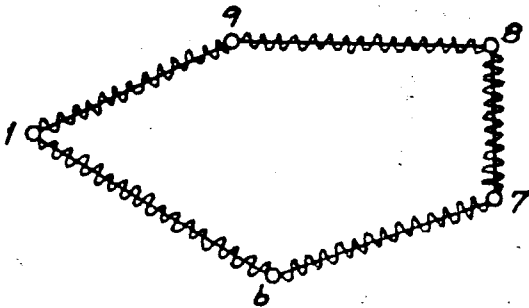
TERMINAL NODE

E (1, 2, 1)



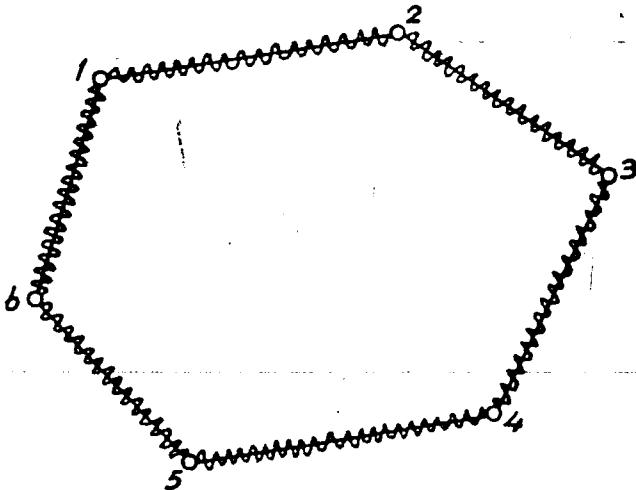
TERMINAL NODE

I (1, 2, 2)



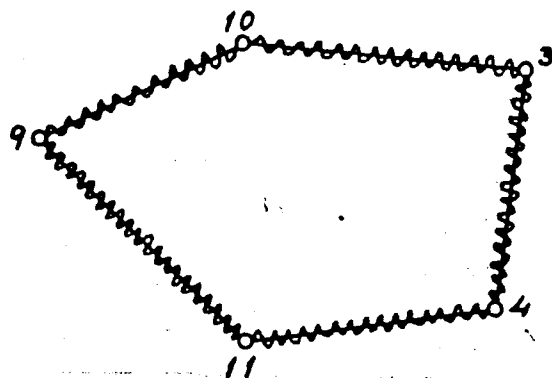
TERMINAL
NODE

E (1, 2, 2)



TERMINAL
NODE

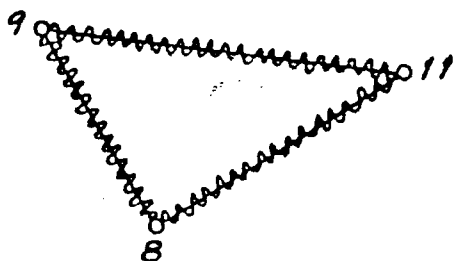
I (1, 1, 2, 2)



TERMINAL

NODE

E (1, 1, 2, 2)



TERMINAL

NODE

CHAPTER 9

SOME FURTHER DEVELOPMENTS AND LINES OF INVESTIGATION

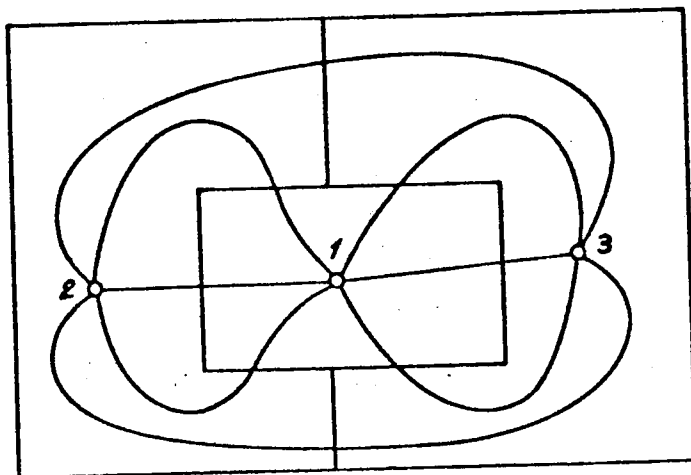
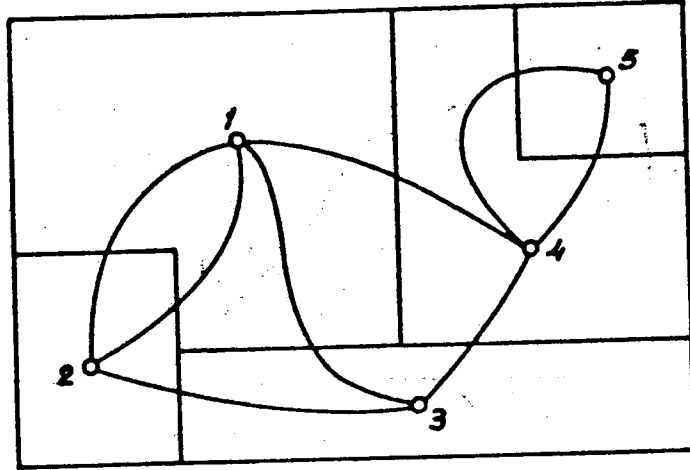
1. INTRODUCTION

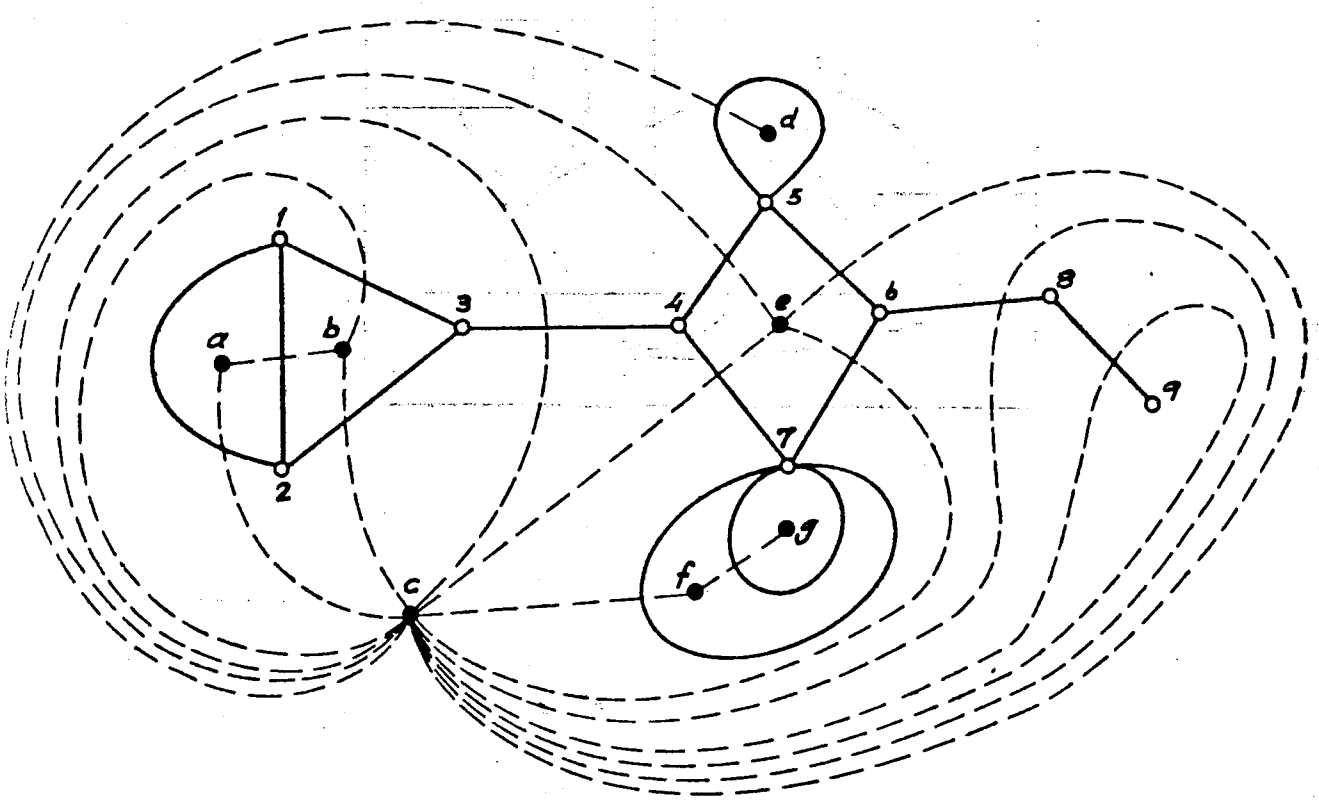
In this chapter we refer to further developments of the work presented and to lines of investigation which, though not immediate extensions of the theory are, nevertheless, worthy of exploration.

2. FURTHER DEVELOPMENTS AND LINES OF INVESTIGATION

These are:

(a) To consider other forms besides the rectangle for the individual spaces. The adjacencies between such spaces can be taken into account by the use of non-simple graphs, as shown in the next two figures. We remark that a dual of non-simple and/or separable planar realizations can be easily defined, as shown in the third figure.





(b) To describe three-dimensional layouts, by graphs of adjacencies with constraints on loads, access routes, piping, etc. .

(c) To take into account maximum and/or minimum distances between spaces, total area, individual and global perimeters, total length of wall, and other metric concepts.

(d) To envisage the arrangement of geometric forms within each space, taking various constraints into account which can be translated into metric terms.

(e) To consider that a graph may be used to express not only spatial adjacencies but also the linkage among activities that may (or should) occur in the same or in distinct places; for example, human activities like eating, meeting, sleeping, and so on. The problem is then to translate the graph of activities into a graph of adjacent and non-adjacent spaces by fusing some of the nodes or edges into one, and/or by splitting nodes and edges into various others, such that the requirements laid down for the adjacency graphs of spaces are satisfied.

(f) To increment the interaction facilities between user and programs. This should be a primary goal in all further developments.

(g) To develop a programming language especially apt for describing the manipulation of objects and concepts pertaining to spatial organization.

(h) To further formalize the theory developed and its extensions, so

as to free it from any particular semantical interpretation and thus it susceptible of semantical interpretations other than the particular considered (i. e. to turn it into a purely syntatic theory to which supplementary axioms, hypotheses or definitions may be added, and to which semantic interpretations may be assigned).

(i) To further explore appropriate grammars for talking about properties of the layouts, such as the types of architectural configuration that may be imposed (e. g. linear, cluster, focal, radial, grid, circular etc).

(j) To pursue the following line of investigation which is important enough to be specified in greater detail.

First note that:

(1) Conditions, A, B, and C have been identified for regulating the permissibility of a coloured realization, not only in respect of its "micro-structure" but also its "macro-structure".

(2) Algorithms for regulating the distribution of dimensions have been given.

(3) The processes, implicit in (1) and (2) which are used for regulating a given realization, possess a certain degree of "separateness", as marked before.

We now remark that:

(4) Each individual space can be thought of as being partially or totally surrounded by other spaces: i. e. as having a sequence of "contour" spaces to which "contour rules" may be applied.

(5) A space and its contour spaces determine a subrealization to which conditions A, B and C will apply, as well as the dimension distribution algorithms.

(6) The "subrealization" corresponding to each node can be individually considered for the purpose of adding edges so as to meet conditions A1 and A2. Significantly, the addition of the extra edges can be made in terms of a redefinition of the "contour" of the space corresponding to the node in question, by developing certain "contour specification rules for a node". Furthermore, as each node is considered in sequence, it either gives rise to constraints in the following nodes and their subrealizations, or in case incompatibilities develop demands reconsideration of the previous nodes.

(7) Since we are already aware of the individual processes stipulated for carrying out conditions A, B and C, as well as the dimension algorithms D, we suggest that these processes may be duly modified for being "simultaneously" carried out for each "subrealization".

(8) The particular succession of nodes and corresponding "subrealizations" would be controlled by the user, conflicts being resolved "on the spot" eventually through a redefinition of requirements. The user's insight or his inexplicit (heuristic) rules would also be exteriorized in this manner, by means of suitable interaction facilities, including a graphic display and a light-pen system.

3. COMMENTS

Of the paragraphs presented in the previous section, the ones more

readily implemented are (c) and (d); the ones which are more important for extending the scope of the work are, in order of important, (e), (i), (c) and (a). The ones permitting further facilities for dealing with the problems treated and leading to major developments are (j), (f), (g) and (h).

CHAPTER 10

THE PROGRAM OUTPUT

1. INTRODUCTION

In this chapter some of the layout-schemes pertaining to the planar realization given at the end of chapter six are exhibited. Not only modular solutions are shown but also dimensioned ones. Furthermore, interaction with the computer programs developed is carried out, so as to alter the initially specified dimensions for some of the layouts.

2. PROGRAM AND COMPUTER PARTICULARS

Two computer programs were written by the author, where the theory developed was embodied. The first tests conditions A and B for any given realization and generates all PHV-assignments complying to any specified permissible type of contour form. The second one takes these PHV-assignments, computes modular and/or dimensioned layout-schemes and outputs them. It also provides the interaction for reformulation of dimensional requirements.

The various planar realizations of any given planar graph are computed by the planarity algorithm mentioned in chapter eight. It was developed by Monteiro and F. Pereira (see reference in that chapter).

Now, the programs were implemented on an ICL-4130 computer with a 6 microsecond store cycle and the code of the first program occupies

roughly 11K 24 bit words, while that of the second occupies 8K. They were both written in ALGOL 60.

In the example to be given below a PHV-assignment was found every 60 seconds. It was then given dimensions and printed, on a 1,250 lines per minute lineprinter, by the second program, in 8 seconds.

Program Options.

In the generation of the PHV-assignments two main options are available. The first, is the specification of the type of contour form desired and the minimum and maximum number of convex corners it must respect. The second, controls whether the backtracking process should be done only on the interior edges, finding then the first permissible assignment for the exterior edges, or if, for each assignment on the interior edges, the backtracking mechanism should find all possible assignments on the remaining (exterior) edges. In the first case, successive solutions will in general differ more between themselves than in the second one. This option makes it possible to obtain only those layouts differing at least in the assignment given to an interior edge.

In the output of the layouts two groups of main options may be combined.

First group:

- to choose excess integer and scale
- to have interaction
- to accept new dimensions for nodes or edges
- to specify which spaces are not to be drawn (e.g. auxilliary spaces such as "north" or "patio" may be skipped).

- to change option
- terminate after next layout is drawn

Second group:

- to draw only modular layouts
- to draw only the dimensional layouts which do not exceed the excess integer.
- same as last, substituting the ones which do exceed by the corresponding modular
- all dimensional layouts, whether exceeding or not the excess integer
- get a particular layout given by its number.

Other interaction facilities permit:

- to alter the dimension requirements for any set of spaces of a layout and the extent of their contact.
- to keep the alterations made or to ignore them, for subsequent layouts

3. AN EXAMPLE

The planar realization shown at the end of chapter six was specified, by giving the program the list of its edges, according to Hypothesis 2.

Spaces corresponding to nodes 1 to 15 were specified as exterior spaces, according to Hypothesis 7. Since the given graph is 3-connected, once its contour is specified, the planar realization to be considered becomes uniquely defined. Thus the program "knows" what particular planar realization has been given.

The class of forms specified for the contour of the layouts was the indented class of forms. This class was given to the program by means of a procedure determining an automaton which recognizes just those forms.

In compliance with Hypothesis 4, two lists of tolerance intervals were also given for the dimensions of the spaces. Furthermore, an excess integer was supplied.

In compliance with Hypothesis 5 a list of tolerance intervals was specified for the edges.

The program generated 29 PHV-assignments using the option of backtracking on the interior edges once a complete assignment was found; after the 29th assignment was obtained, it was stopped.

It was also run using the option of backtracking on the interior as well as exterior edges, thereby generating 8 variations of exterior edges for the first layout found; after the 8th layout was produced it was stopped.

The group of 29 layouts was then drawn by the program, according to given dimensional tolerances and excess integer. These are not shown since the tolerance intervals and excess integer were modified on the evidence of the layouts produced, but the group of 29 layouts was then output according to the new specifications. These layouts are shown on the next pages, after the modular ones. Only the ones obeying the excess integer came out.

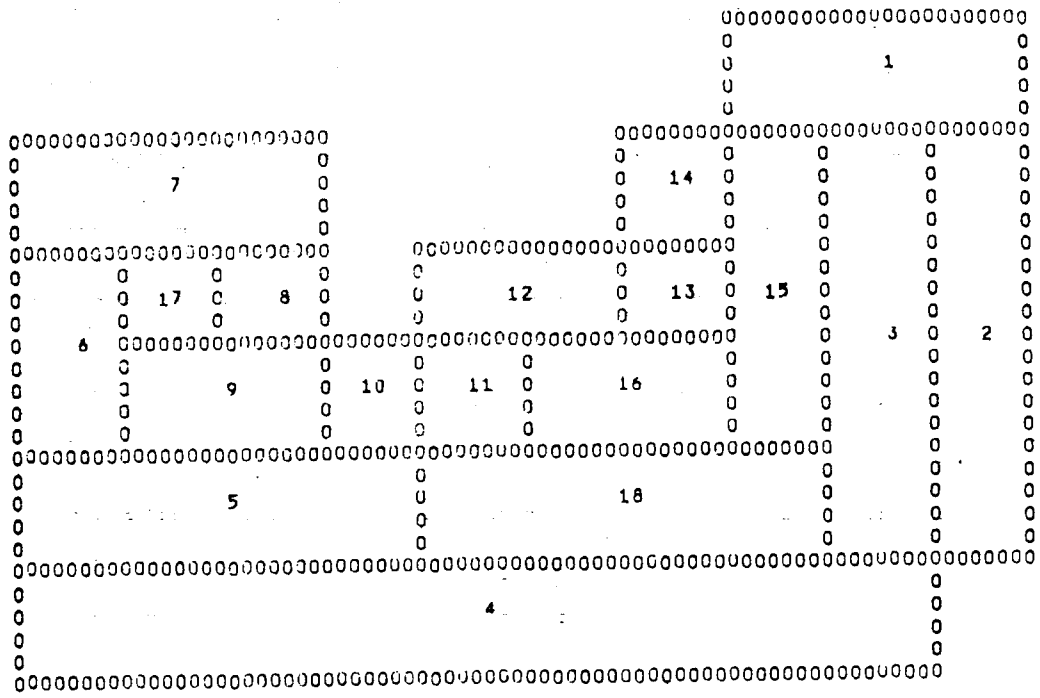
The final tolerance intervals used for the spaces are given now:

TOLERANCE INTERVALS

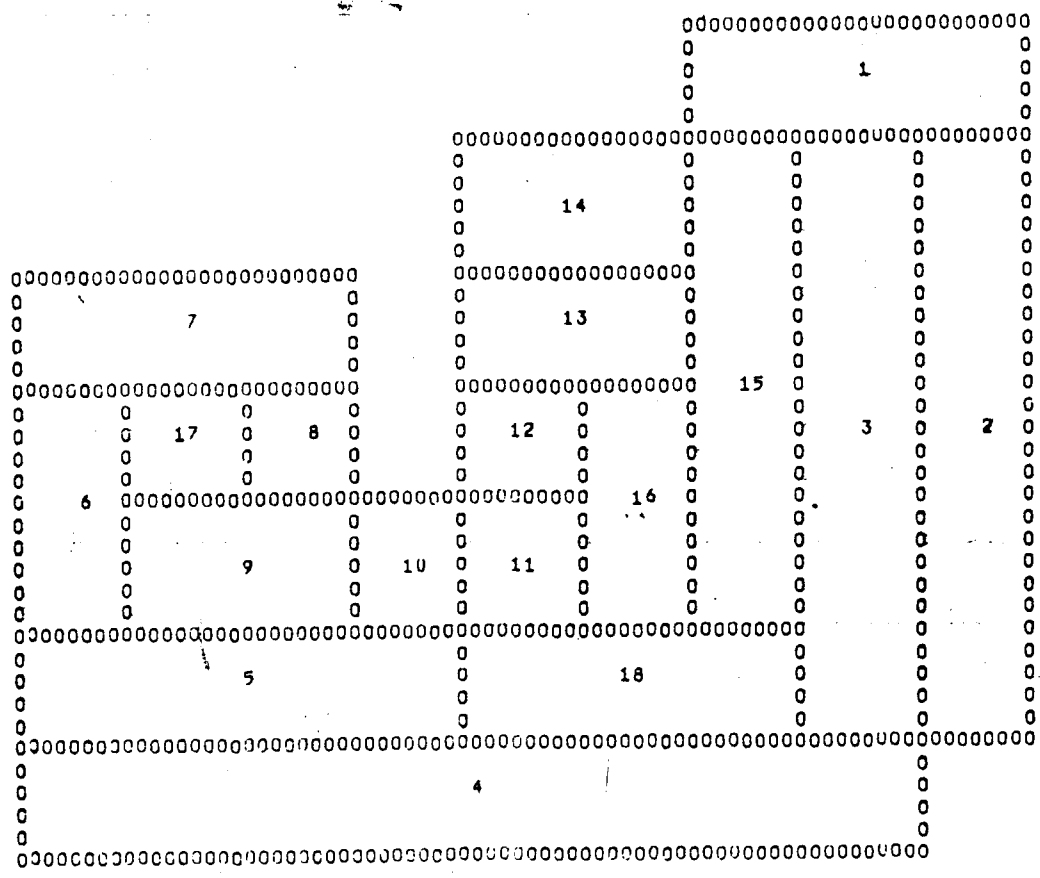
NODE	HORIZONTAL	VERTICAL
1	13 - 18	9 - 12
2	7 - 10	16 - 23
3	6 - 10	14 - 23
4	10 - 34	6 - 9
5	18 - 25	7 - 10
6	6 - 8	10 - 13
7	15 - 20	7 - 9
8	5 - 7	5 - 7
9	10 - 13	6 - 8
10	5 - 10	5 - 8
11	6 - 8	6 - 8
12	6 - 10	6 - 8
13	6 - 10	6 - 8
14	6 - 10	6 - 8
15	11 - 18	6 - 9
16	6 - 11	6 - 10
17	5 - 7	5 - 7
18	11 - 23	7 - 10

A uniform tolerance interval of 1-50 was given to the edges, and the excess integer 10 was supplied.

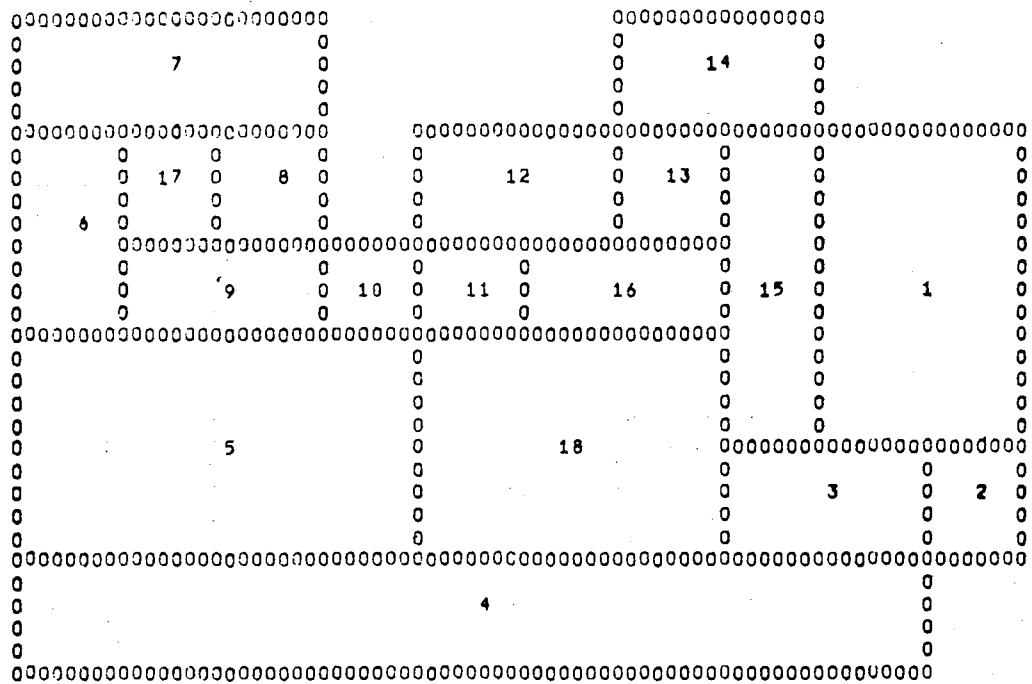
First the modular solutions are shown; then 8 exterior variants of the first layout; then the dimensioned layouts. The layouts resulting from interaction are shown further on.



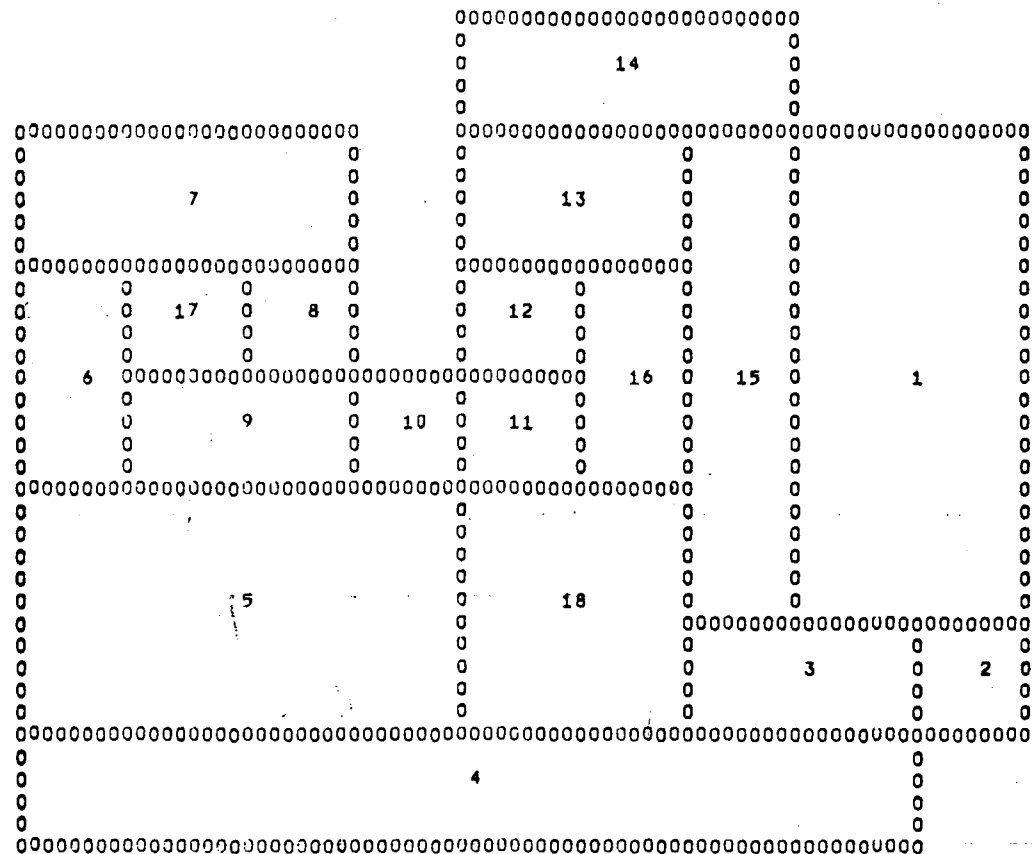
SOLUCAO N 1



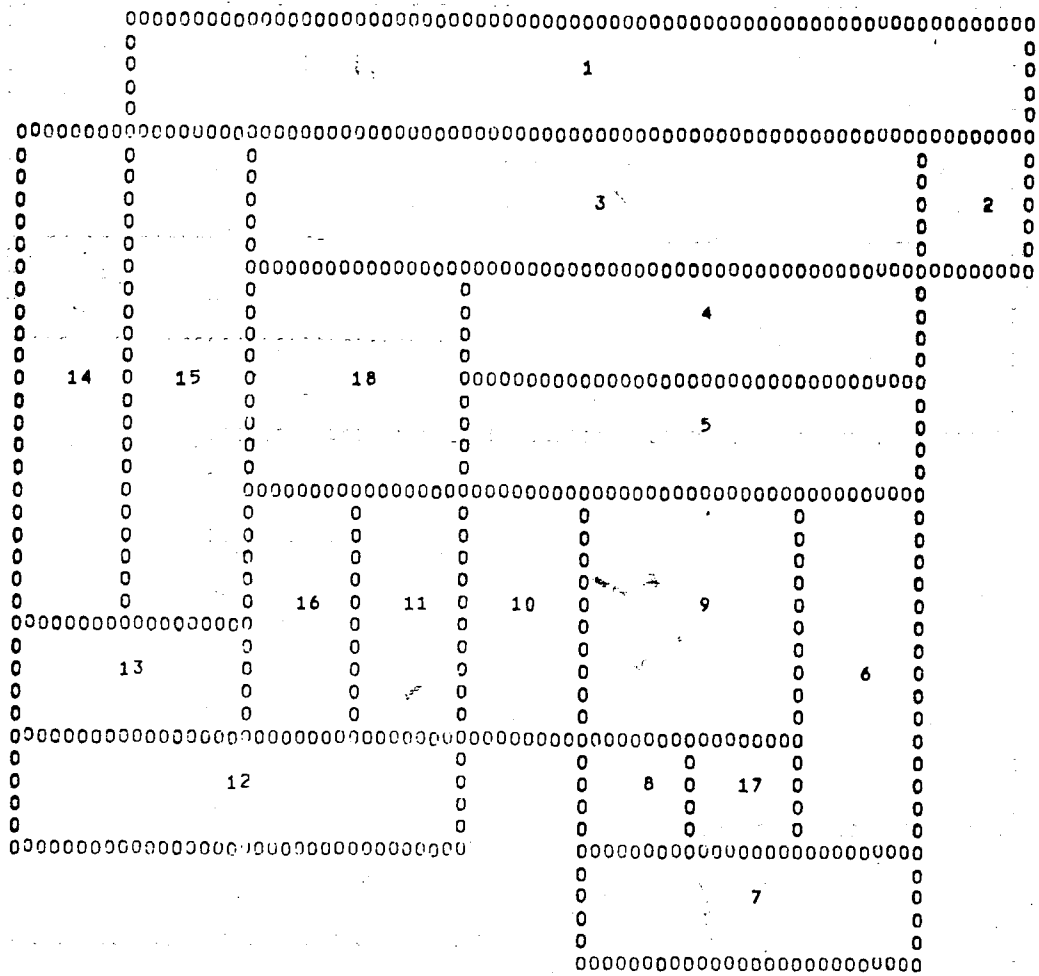
SOLUCAO N 2



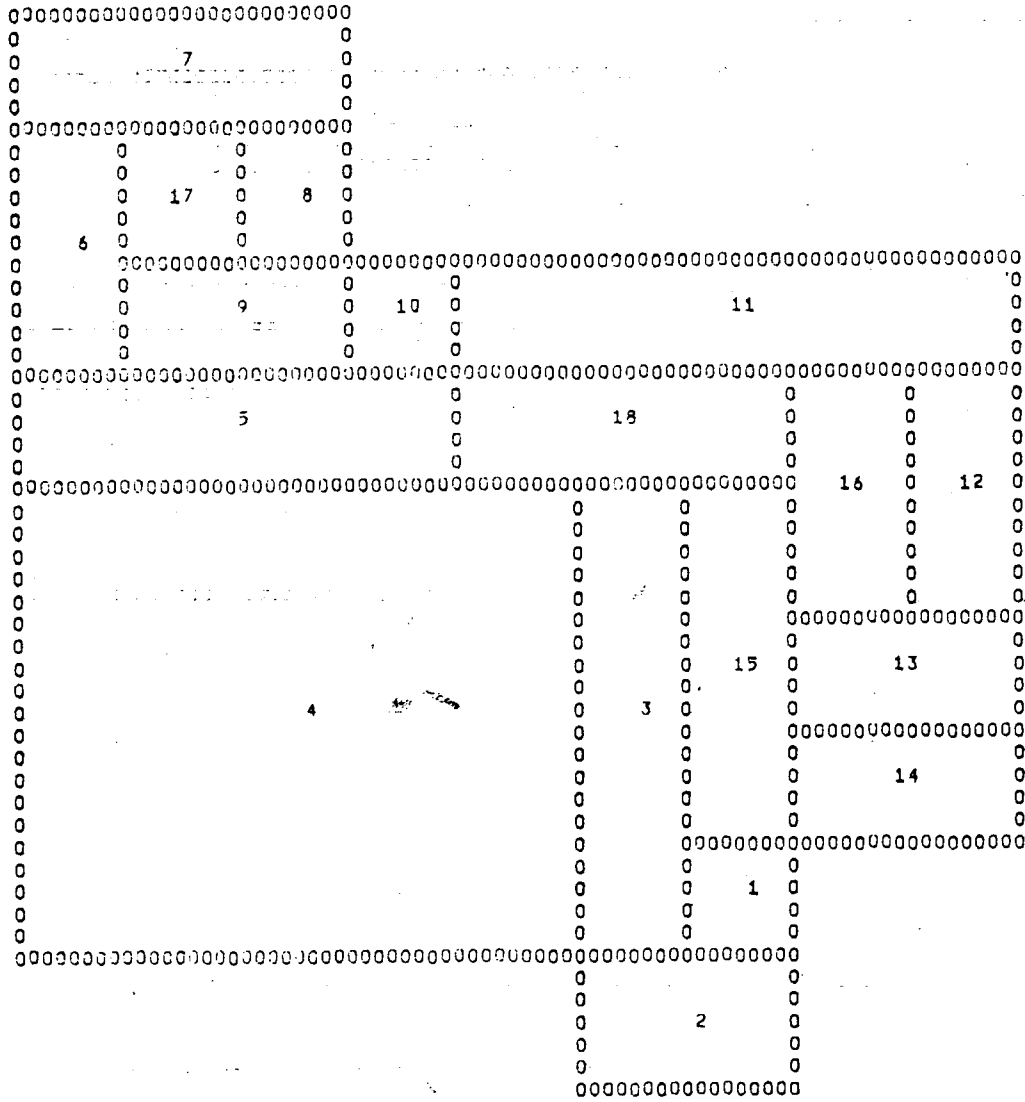
SOLUCAO N 9



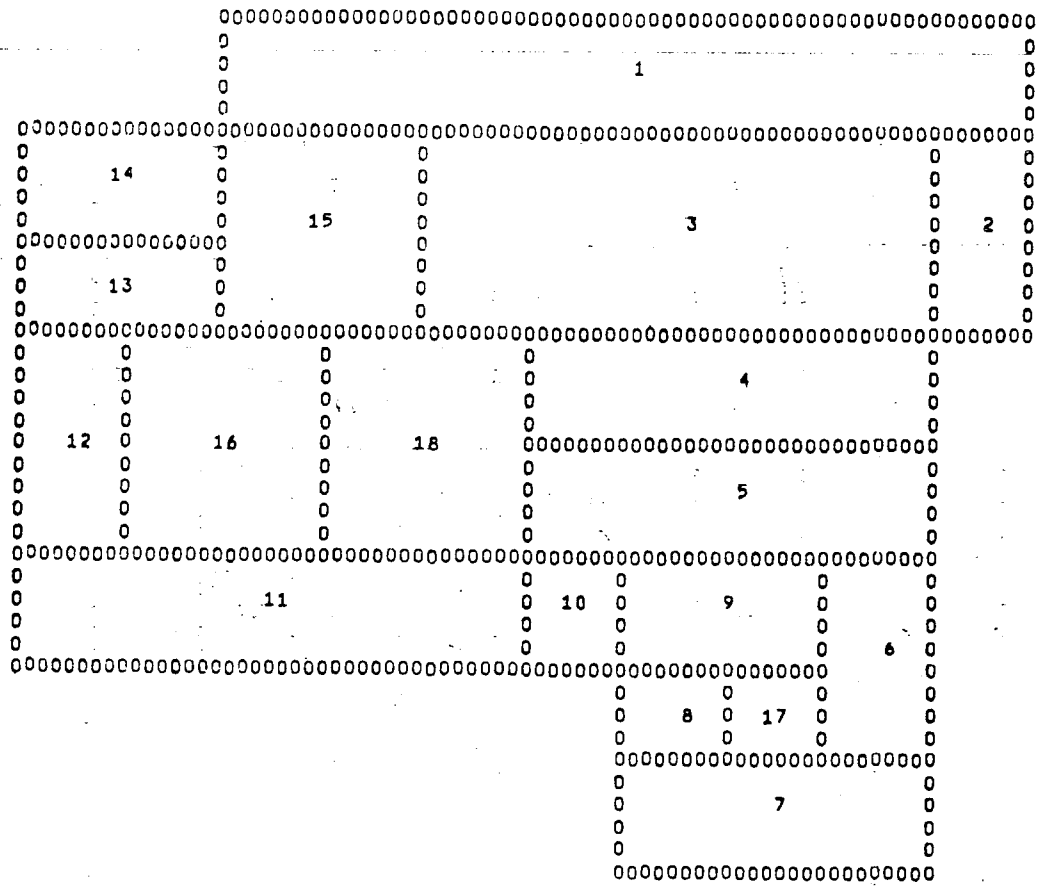
SOLUCAO N 10



SOLUCAO N 15



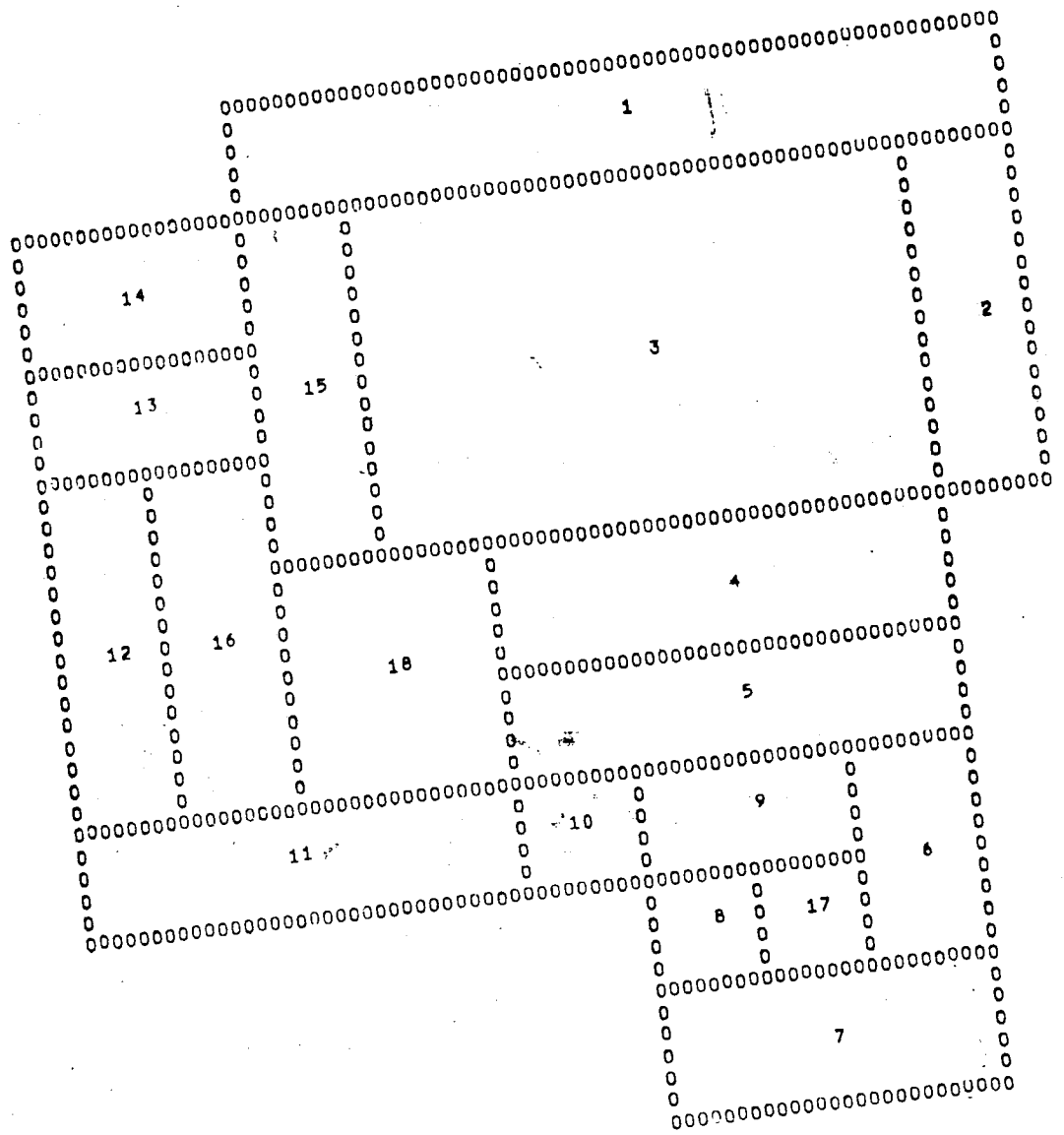
SOLUCAO N 20



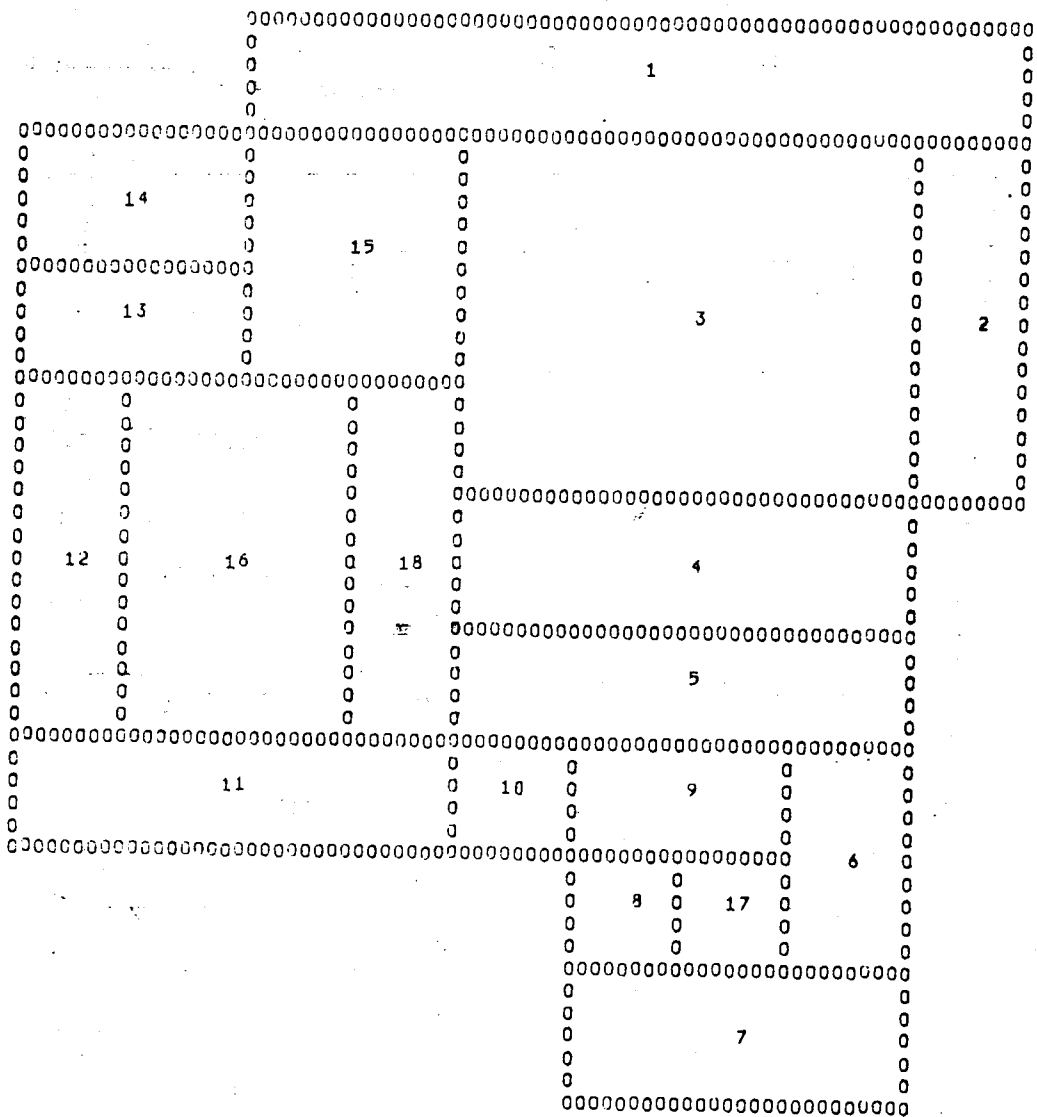
SOLUCAO N 21



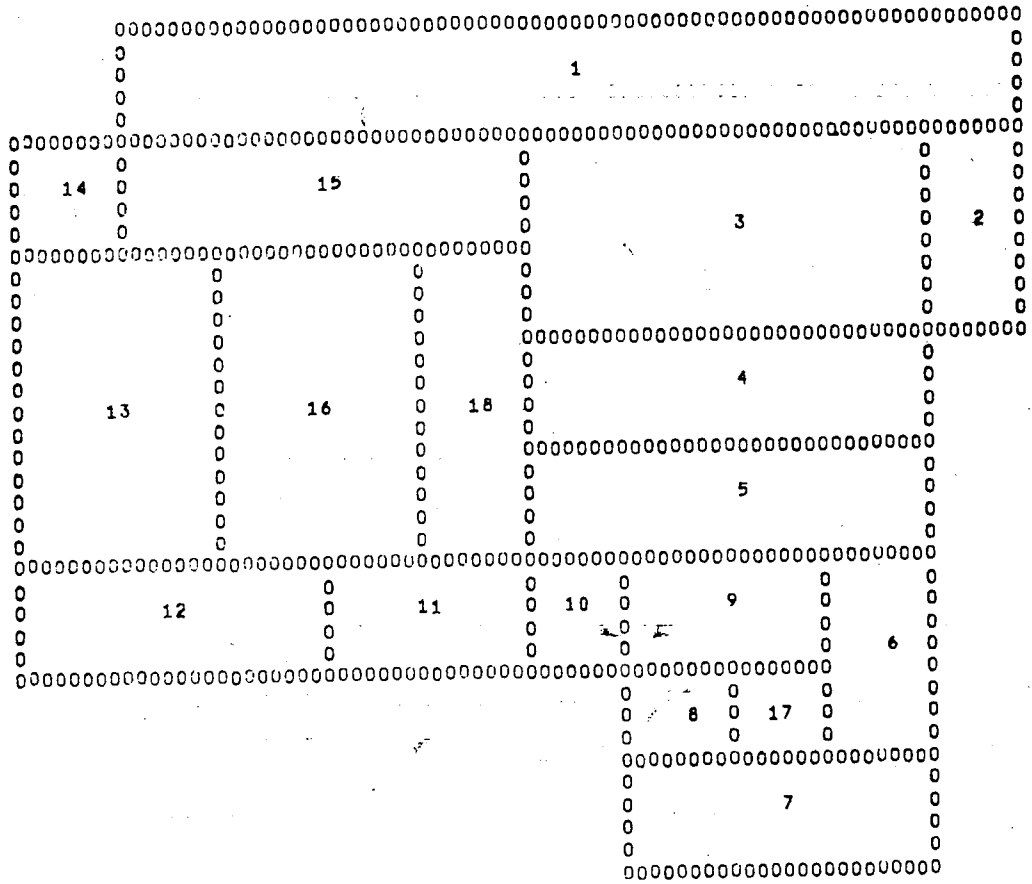
SOLUCAO N 22



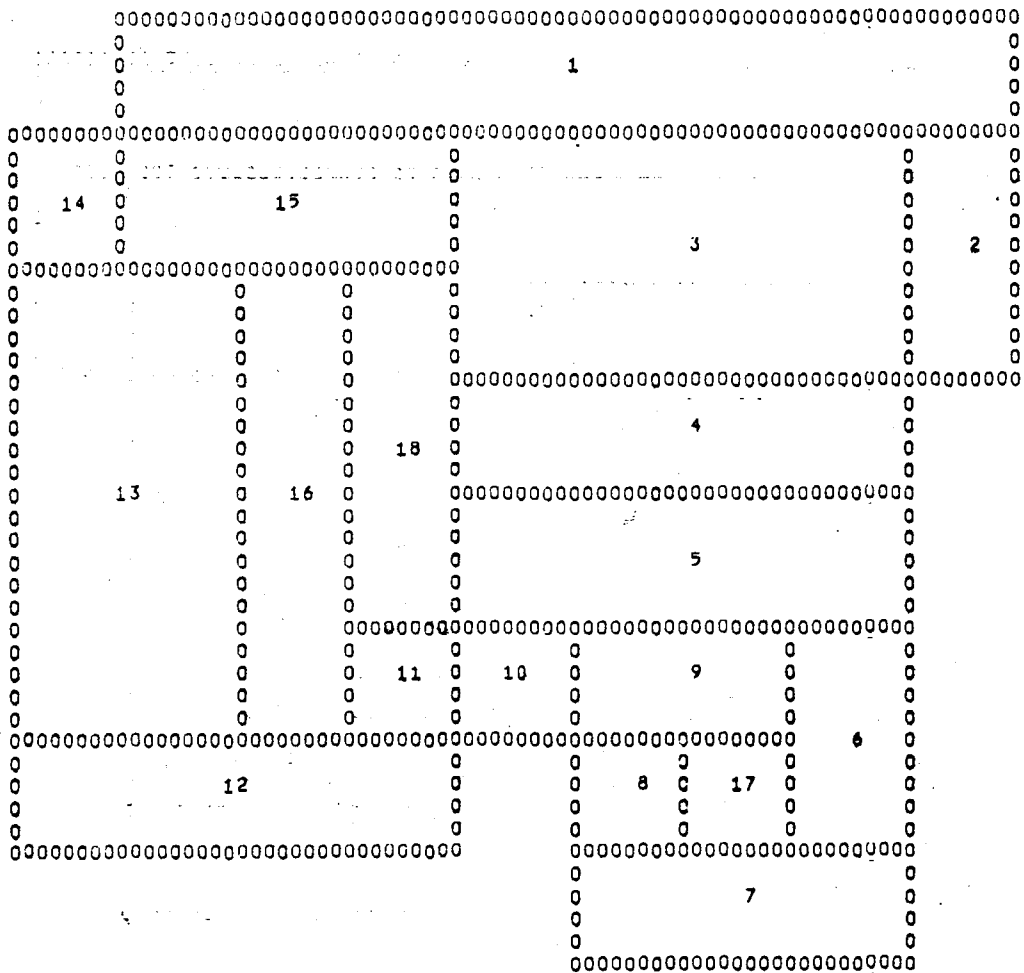
SOLUCAO N 25



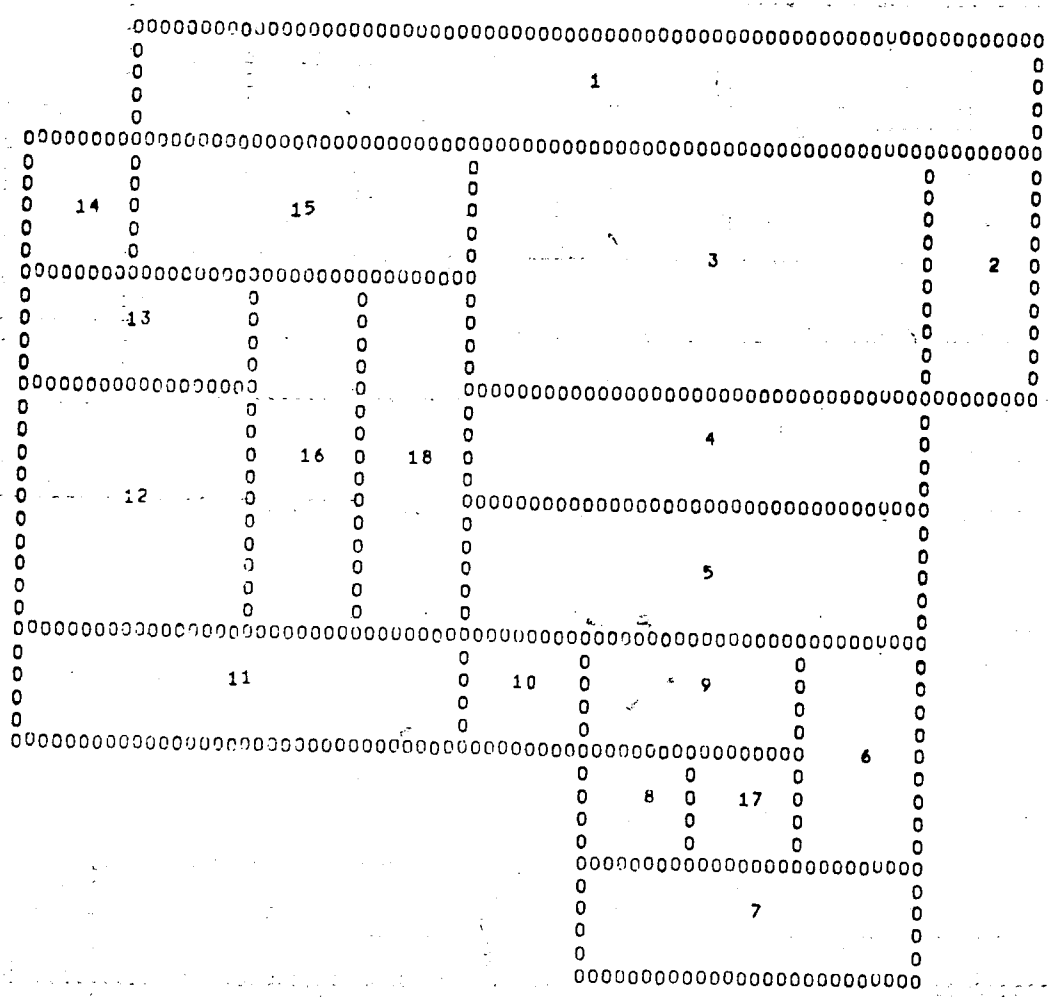
SOLUCAO N 26



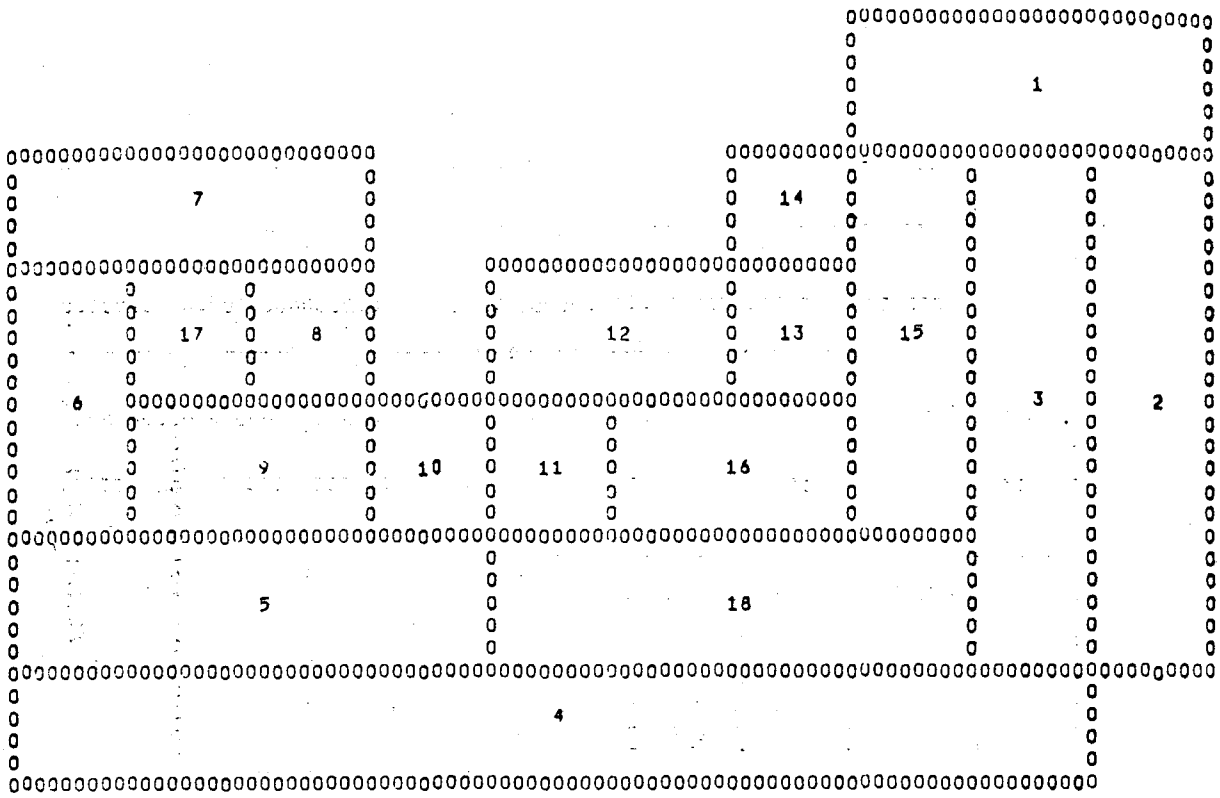
SOLUCAO N 27



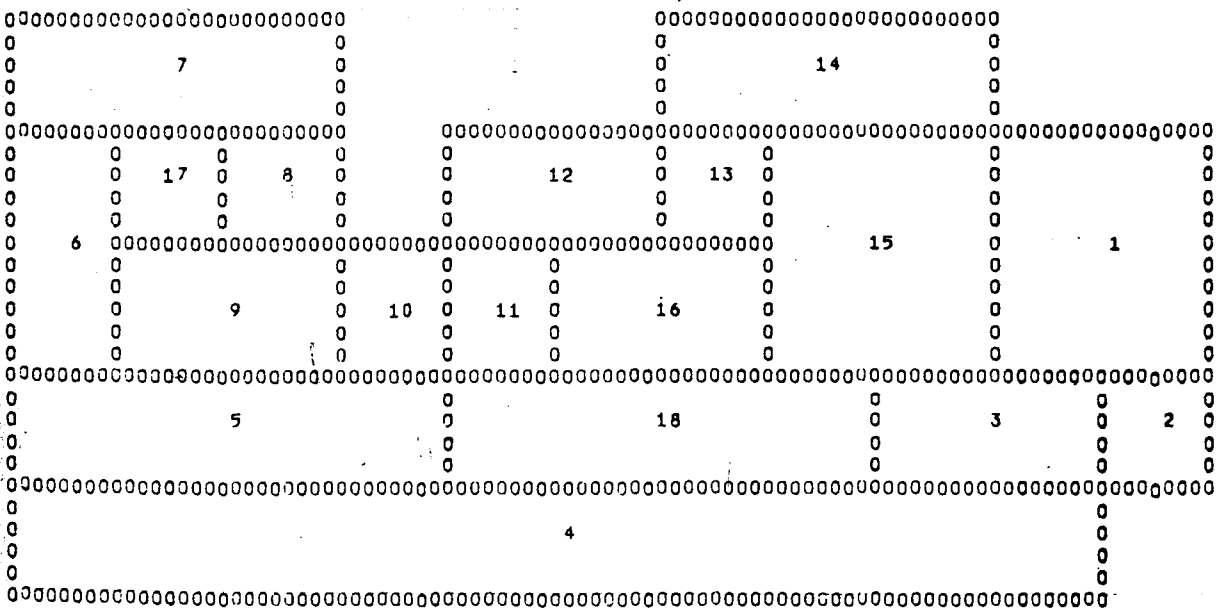
SOLUCAO N 28



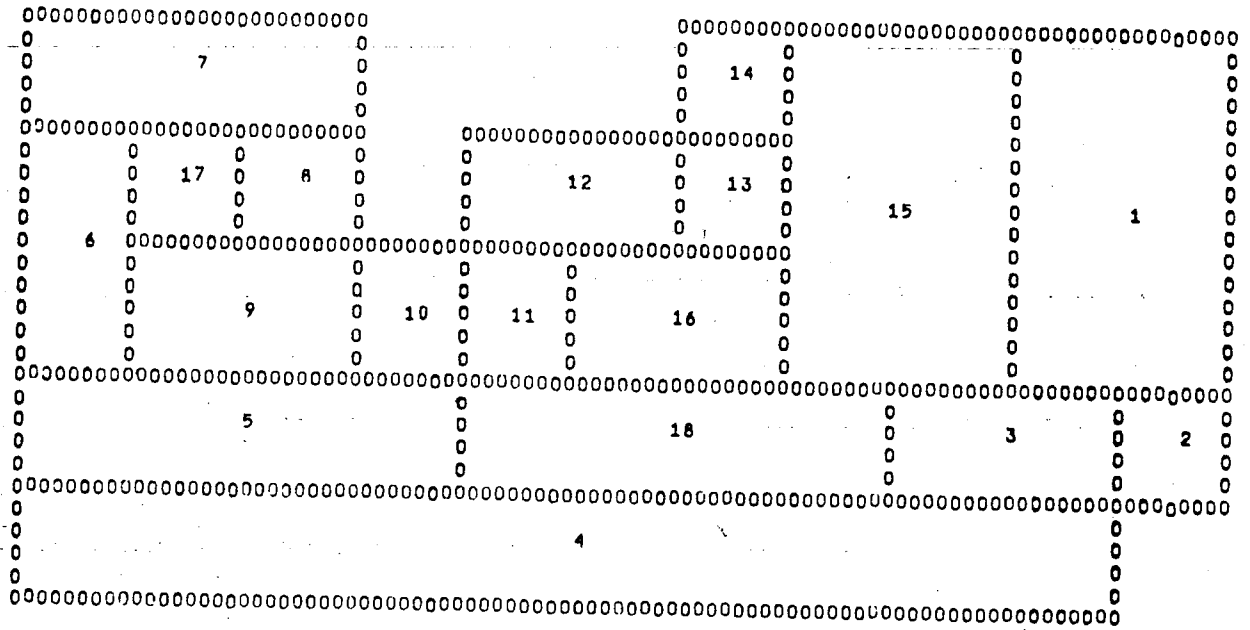
SOLUCAO N 29



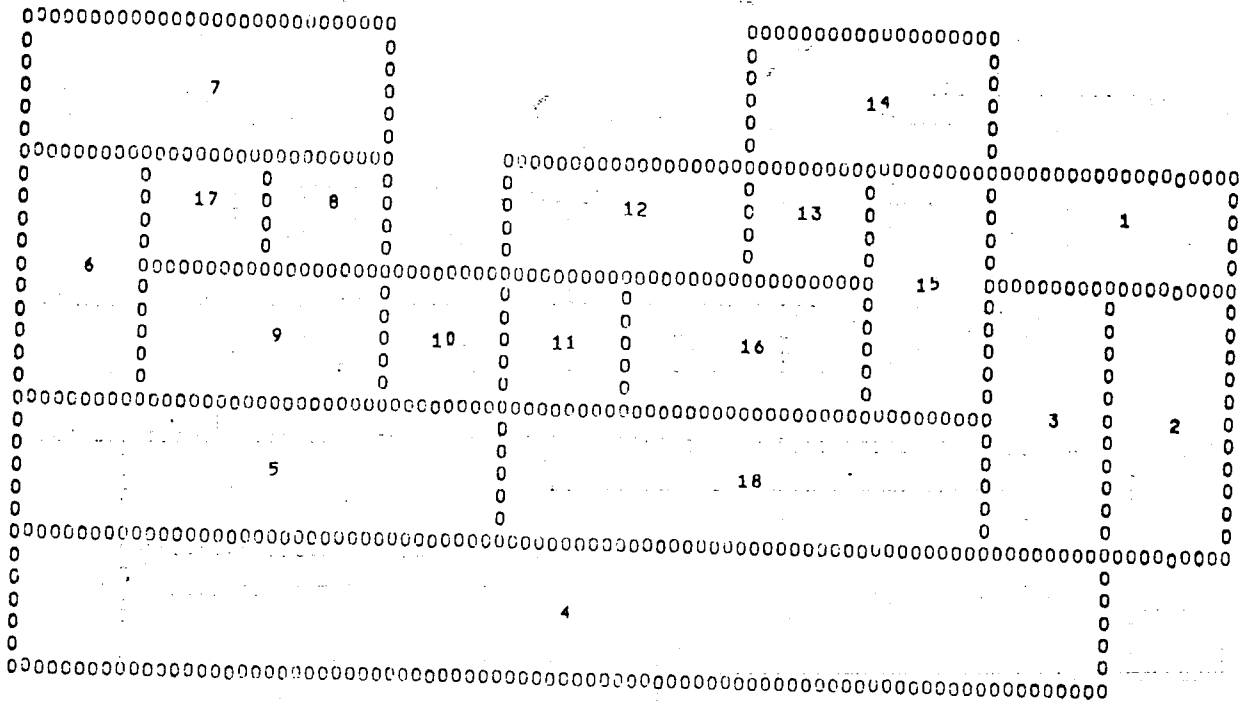
SOLUCAO N 1



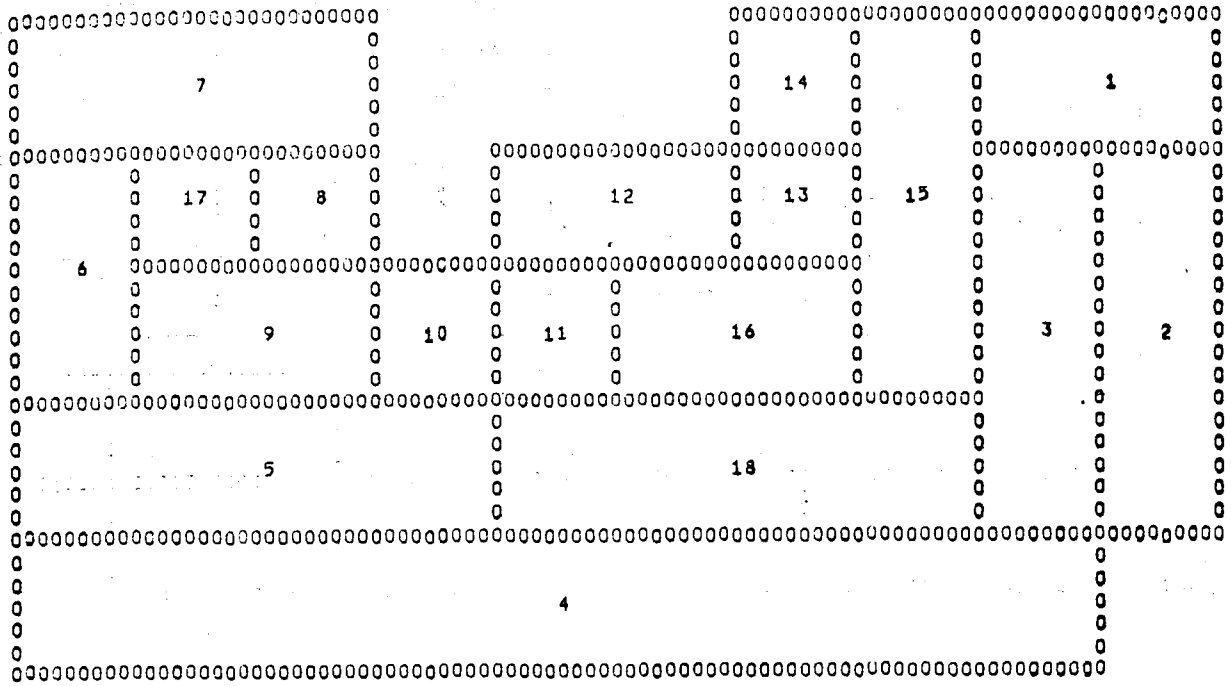
SOLUCAO N 2



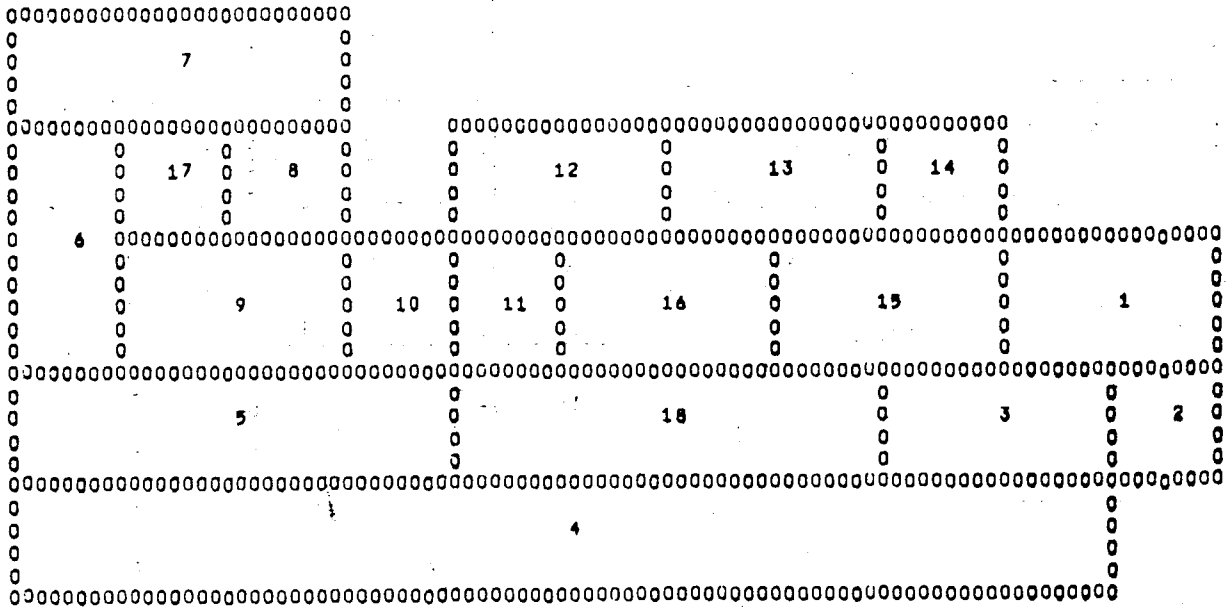
SOLUCAO N 3



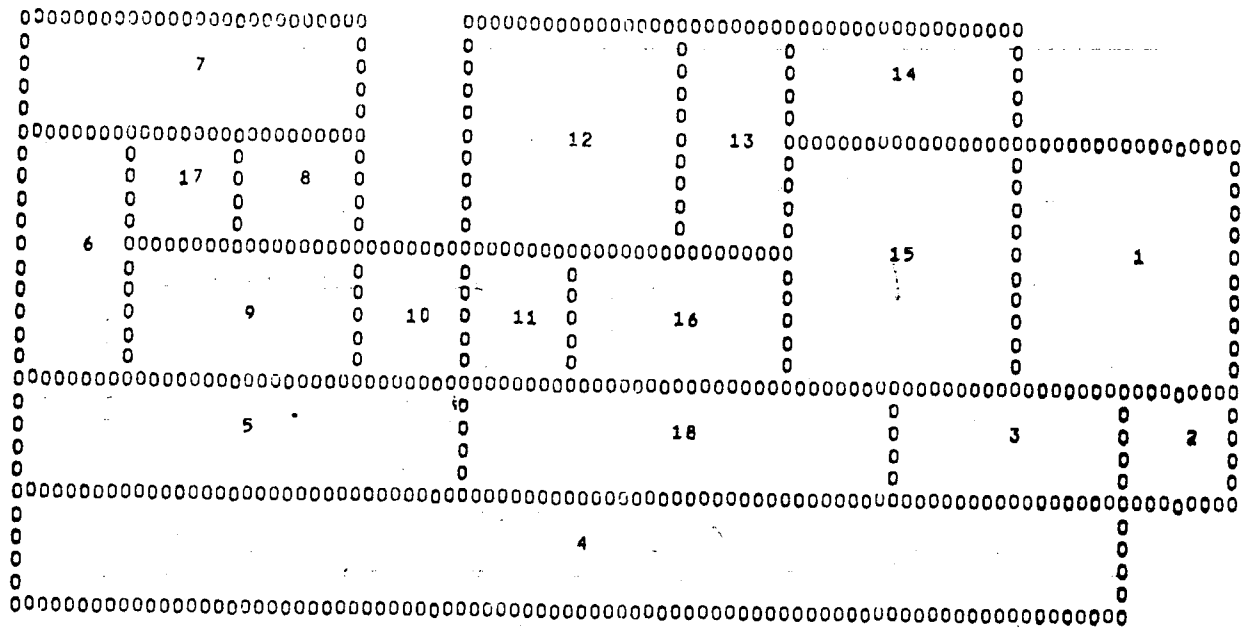
SOLUCAO N 4



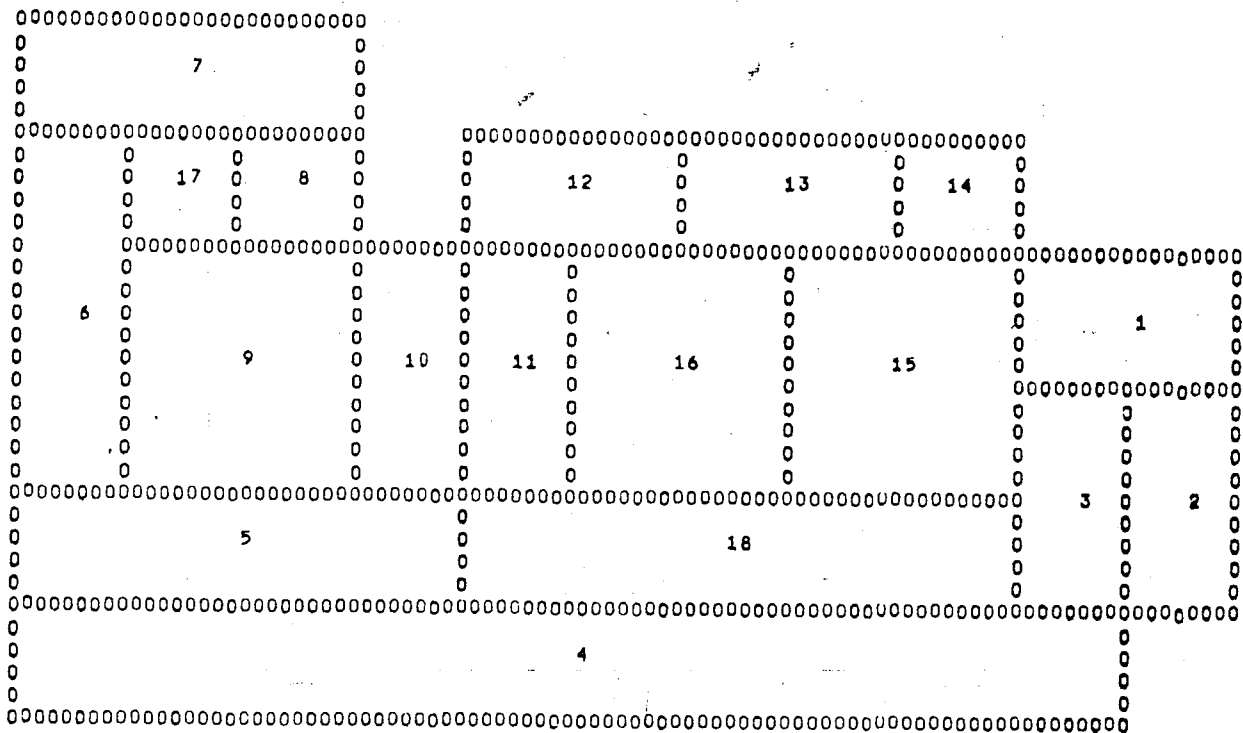
SOLUCAO N 5



SOLUCAO N 6



SOLUCAO N 7



SOLUCAO N 8


```

00000000000000000000000000000000000000000000000000000000000000000000
0
0 7 0 000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 14 0
0 0 0 0
0 0 0 0
00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 00 00 00
0 0 0 0 00 00
0 0 17 0 8 0 0 12 0 13 00 00 00 00
0 0 0 0 0 0 0 0 0 00 00 00
0 6 0 0 0 0 0 0 0 0 00 00
0 00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 15 00 1 0 0
0 0 0 0 0 0 00 00
0 0 0 9 0 10 0 11 0 16 00 00 00
0 0 0 0 0 0 0 00 00
0 0 0 0 0 0 0 00 00
00000000000000000000000000000000000000000000000000000000000000000000
0 0 000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 5 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
00000000000000000000000000000000000000000000000000000000000000000000
0
0 0 0 0
00000000000000000000000000000000000000000000000000000000000000000000

```

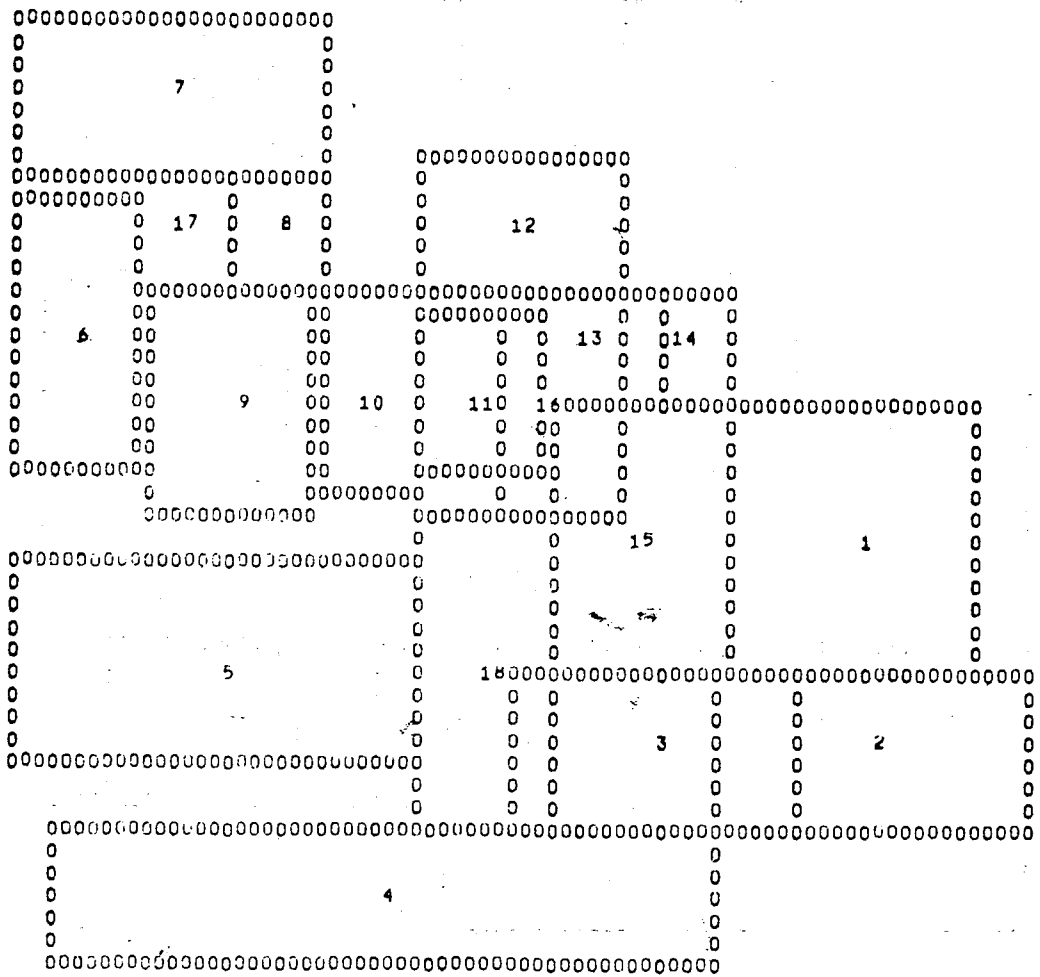
SOLUCAO N 9

```

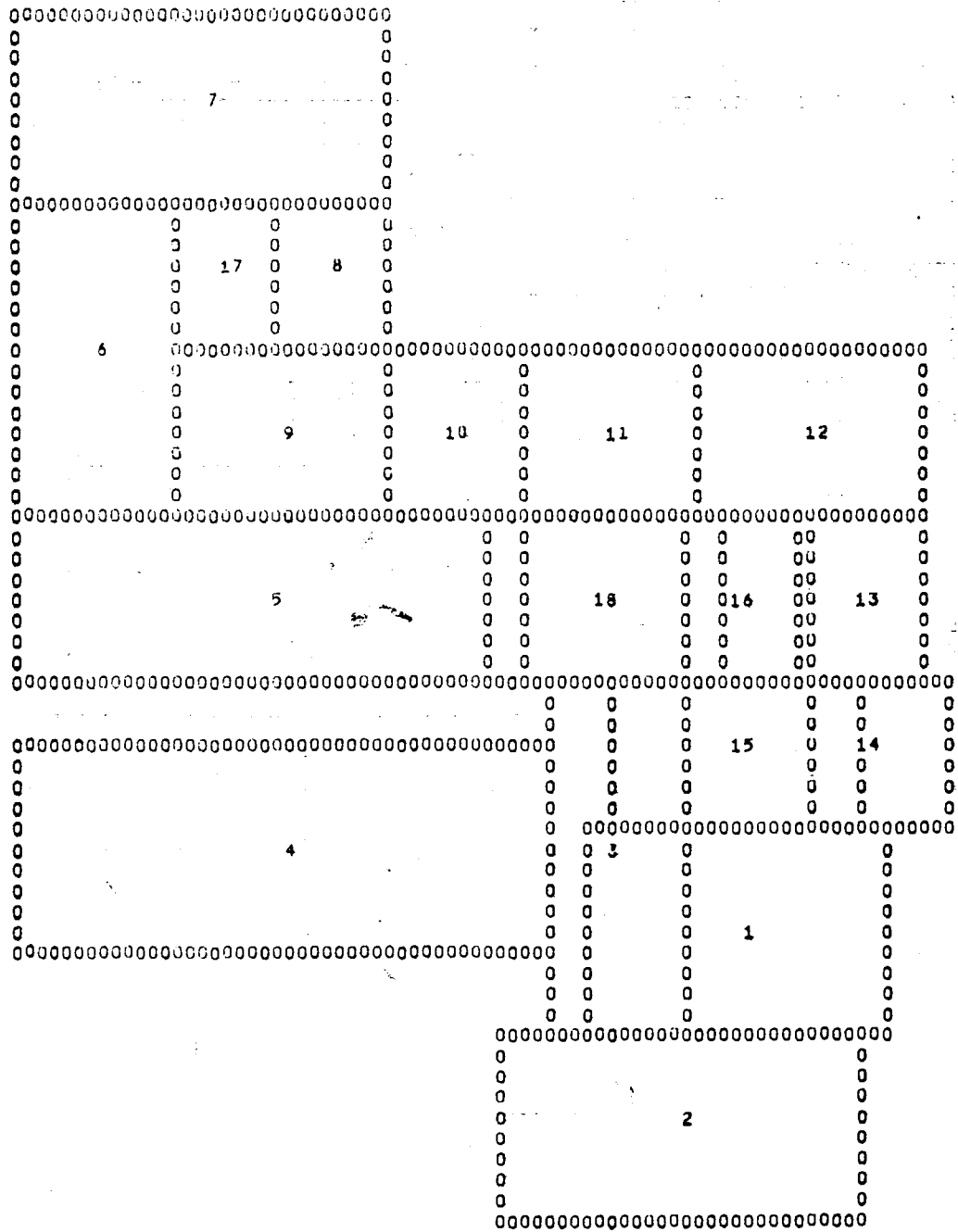
00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 14 0 0
0 0 0 0 0
00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 00 00 00
0 0 0 0 00 00
0 0 13 00 00 00 00
0 0 0 00 00
00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 00 0 00 0 00
0 0 0 0 00 0 00
0 0 17 0 8 0 0 12 00 150 00 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 16 0 00 0 0
0 6 0 0 0 0 0 0 0 0 00 00
0 00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 00 00
0 0 0 9 0 10 0 11 00 0 00
0 0 0 0 0 0 0 0 00 0 00
0 0 0 0 0 0 0 0 0 00 0 0
00000000000000000000000000000000000000000000000000000000000000000000
0 00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 5 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
00000000000000000000000000000000000000000000000000000000000000000000
0 0 0 0 0 0 0 0 0 0 0 0
00000000000000000000000000000000000000000000000000000000000000000000

```

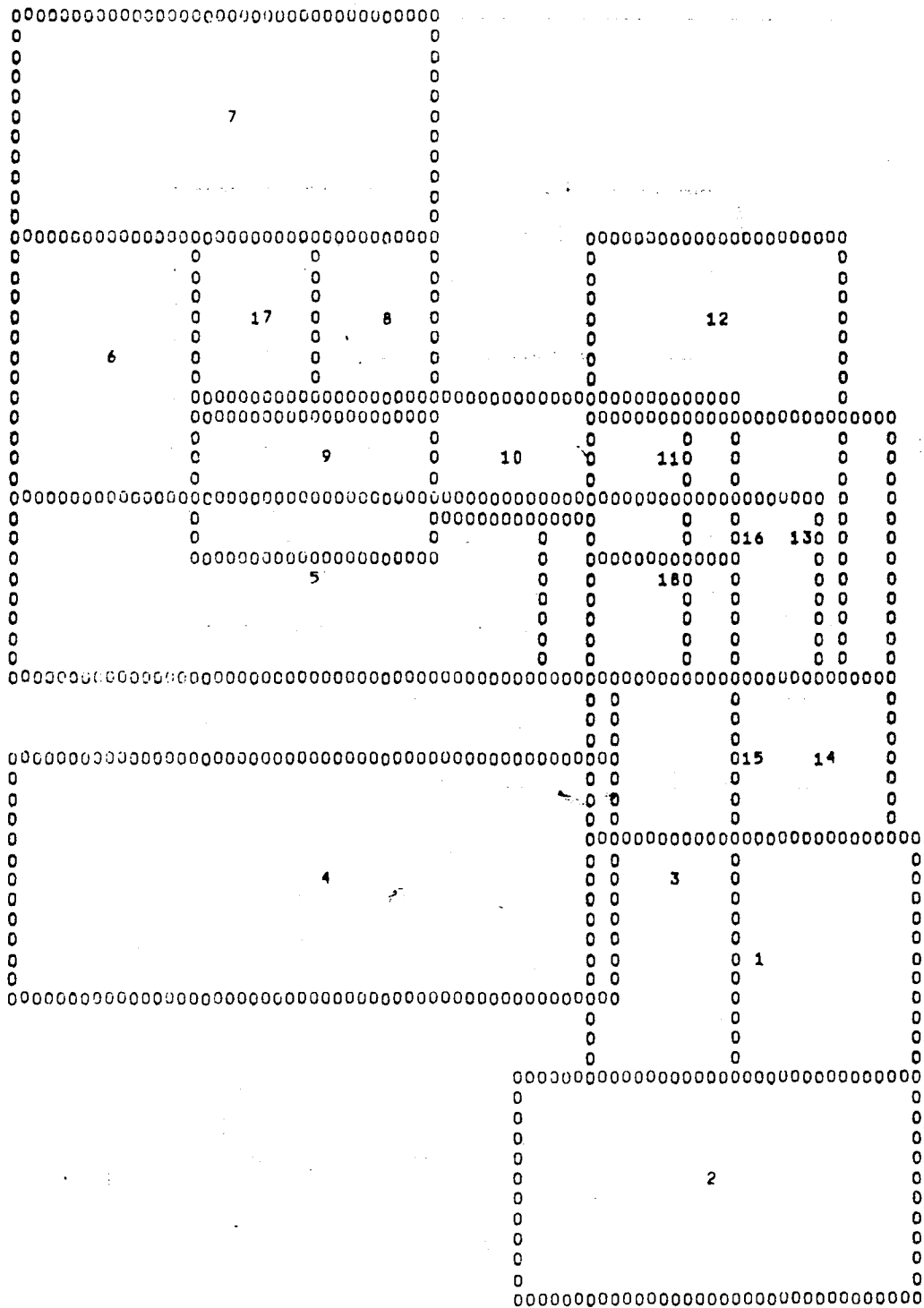
SOLUCAO N 10



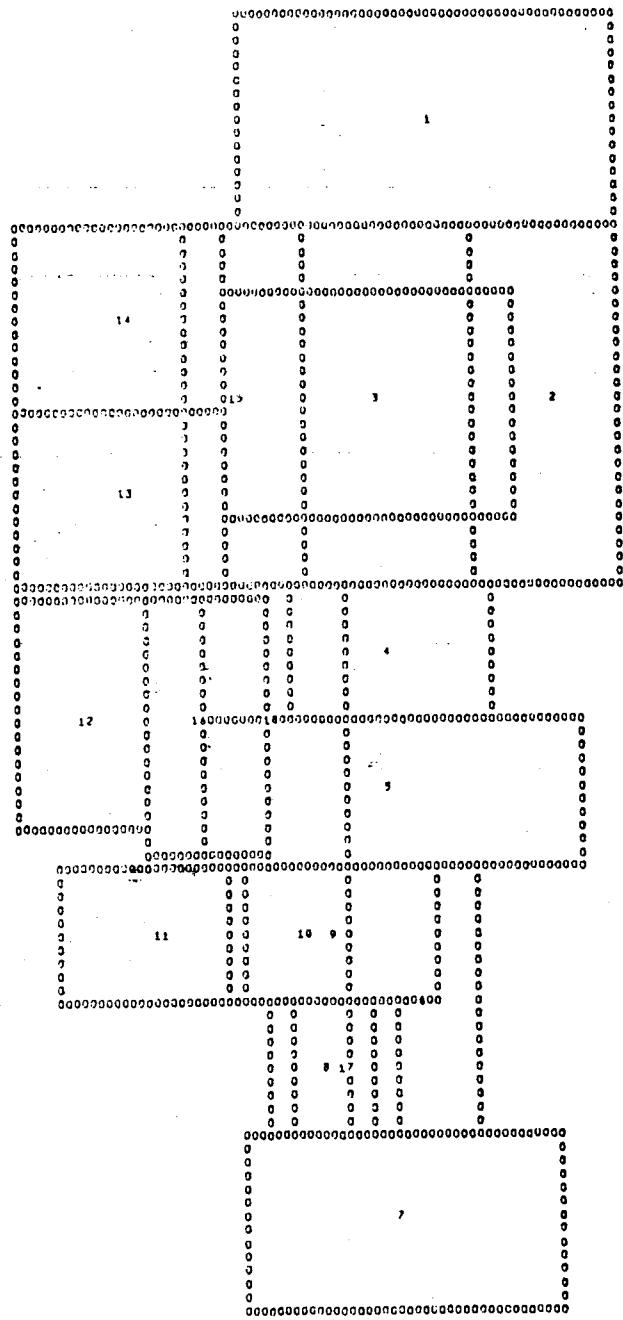
SOLUCAO N 11



SOLUCAO N 17



SOLUCAO N 18



SOLUCIÃO N 21

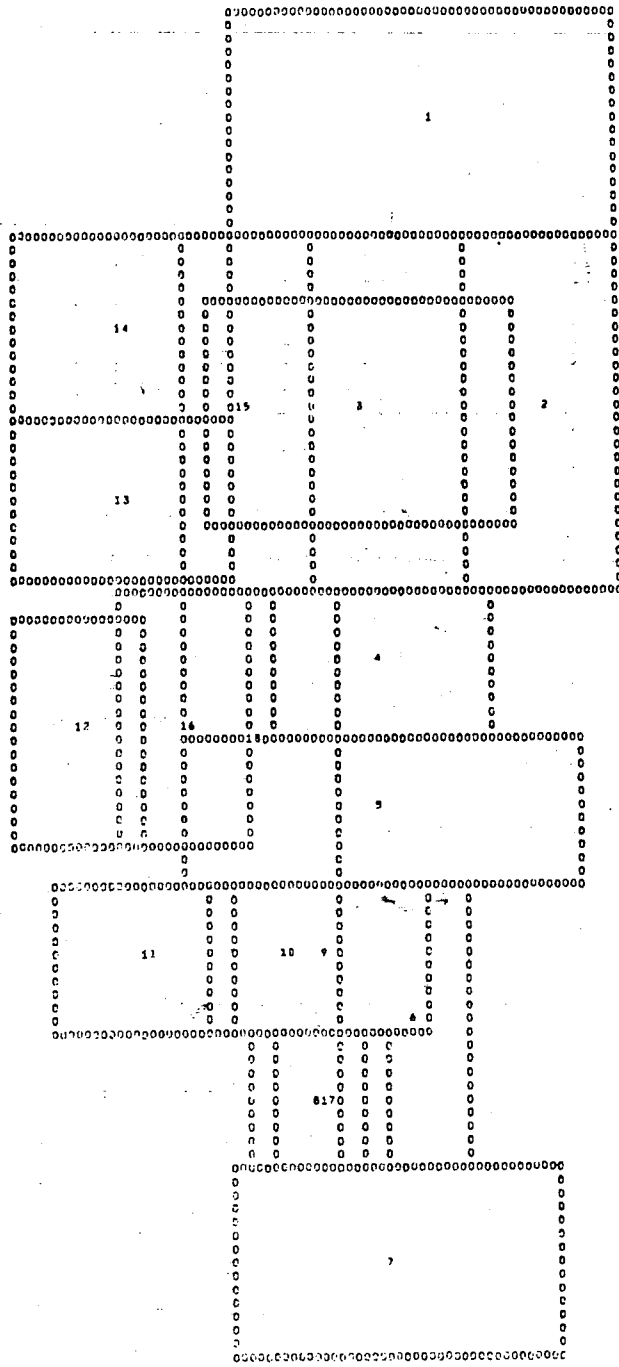
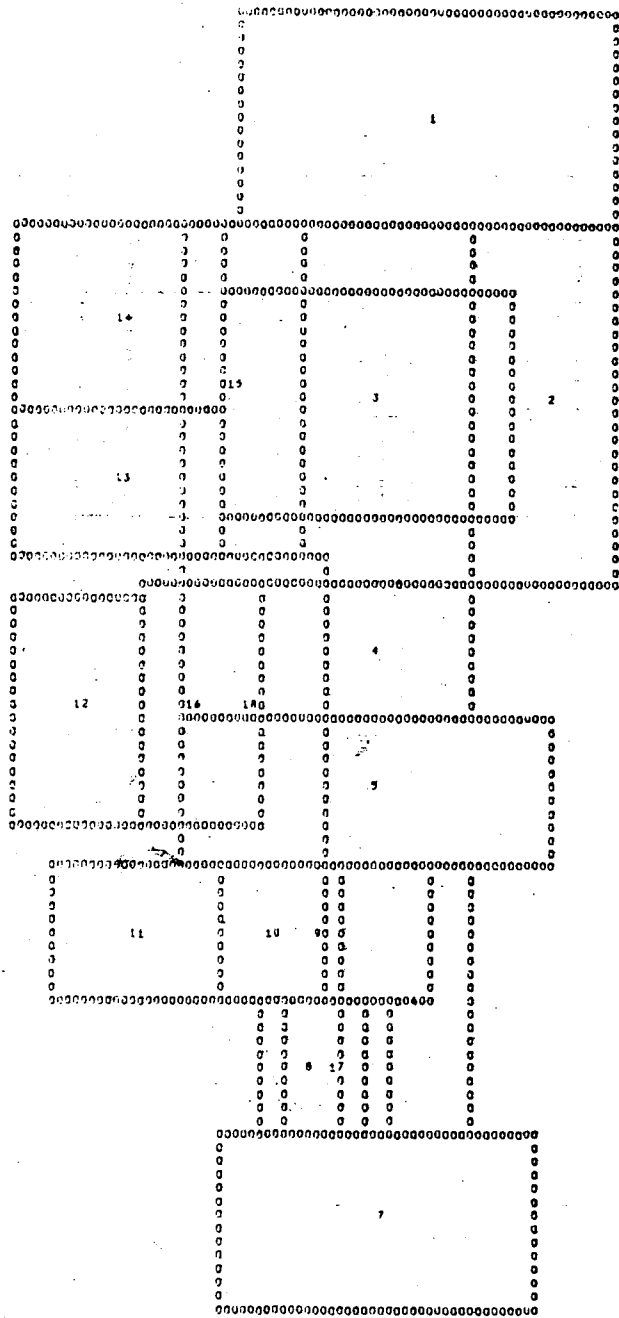
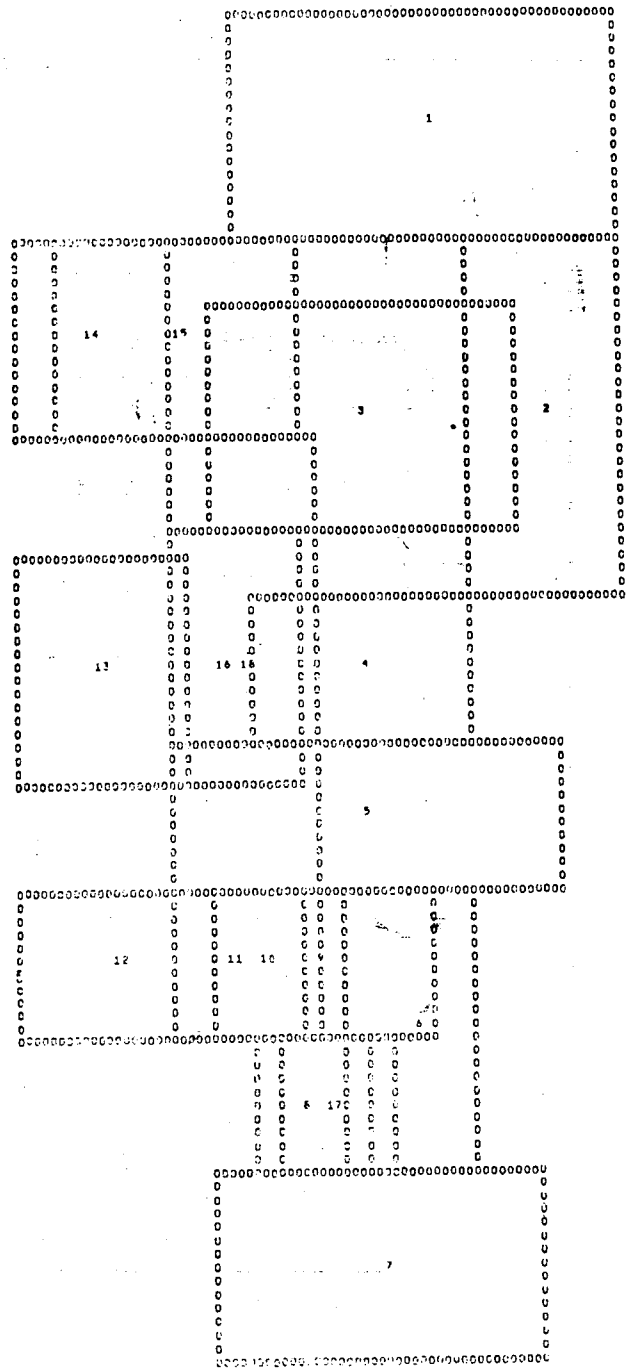


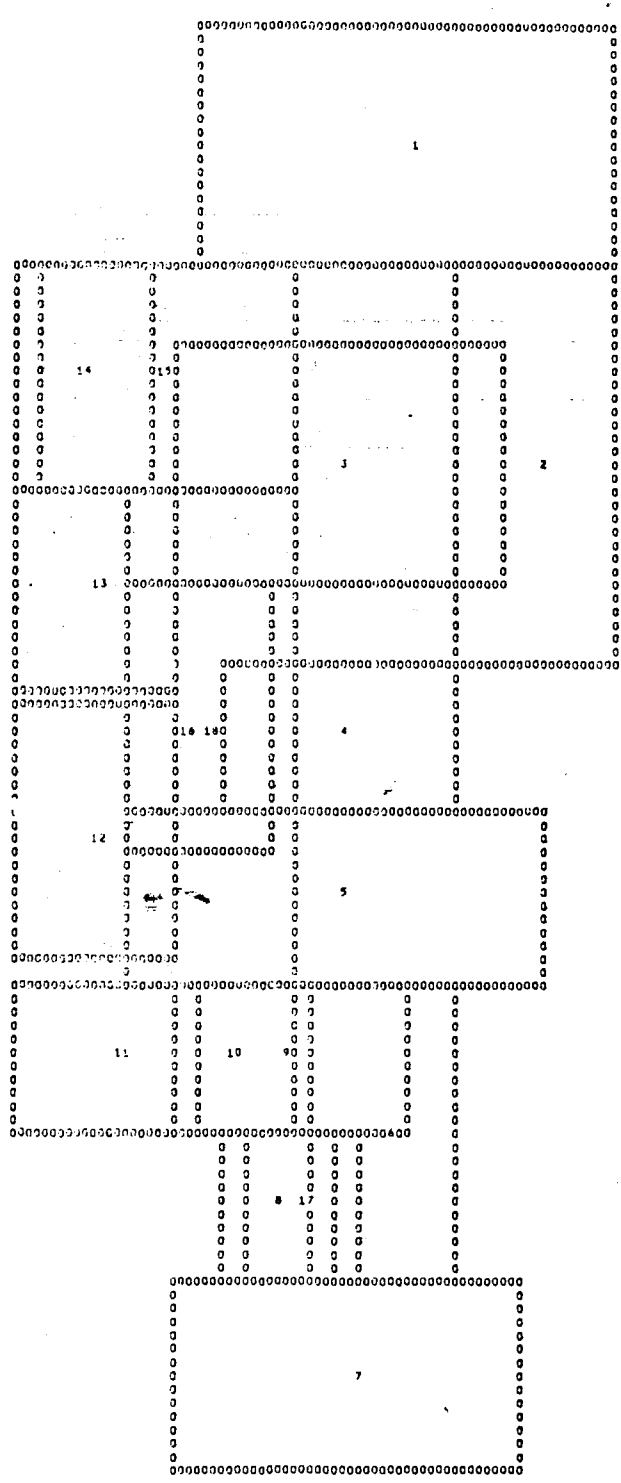
FIGURE 2



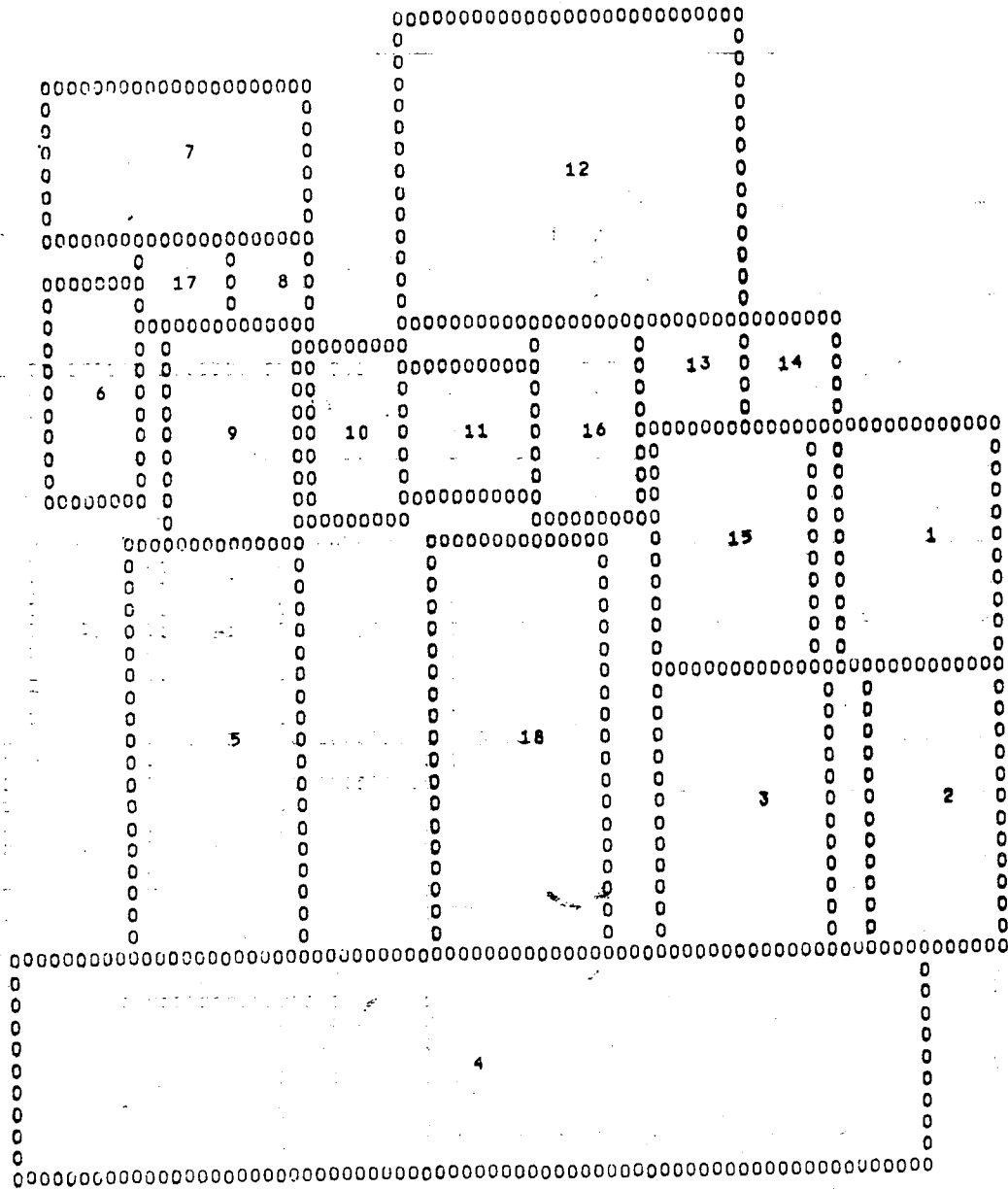
SOLUCIÓE N. 24



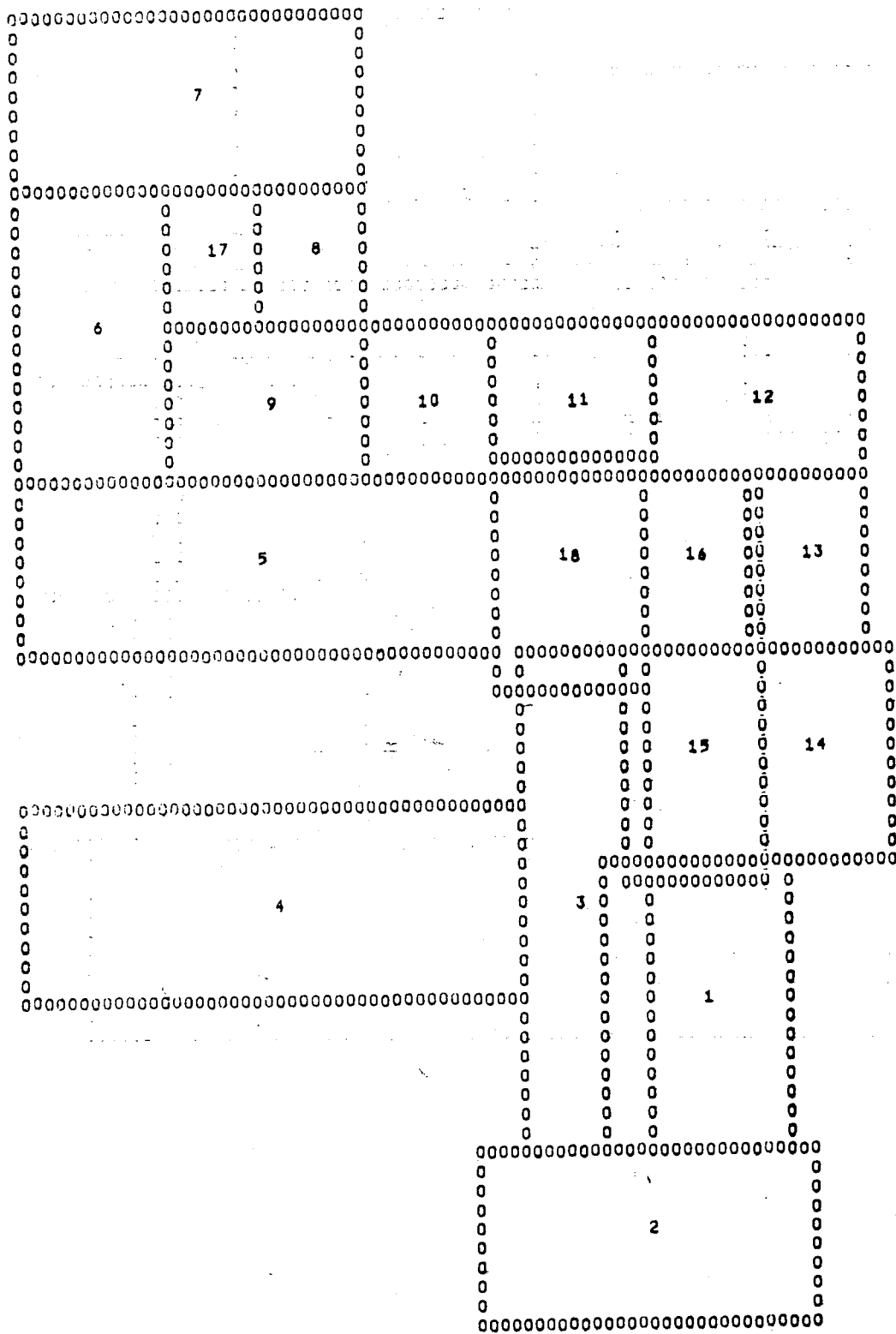
SOUCAS 27



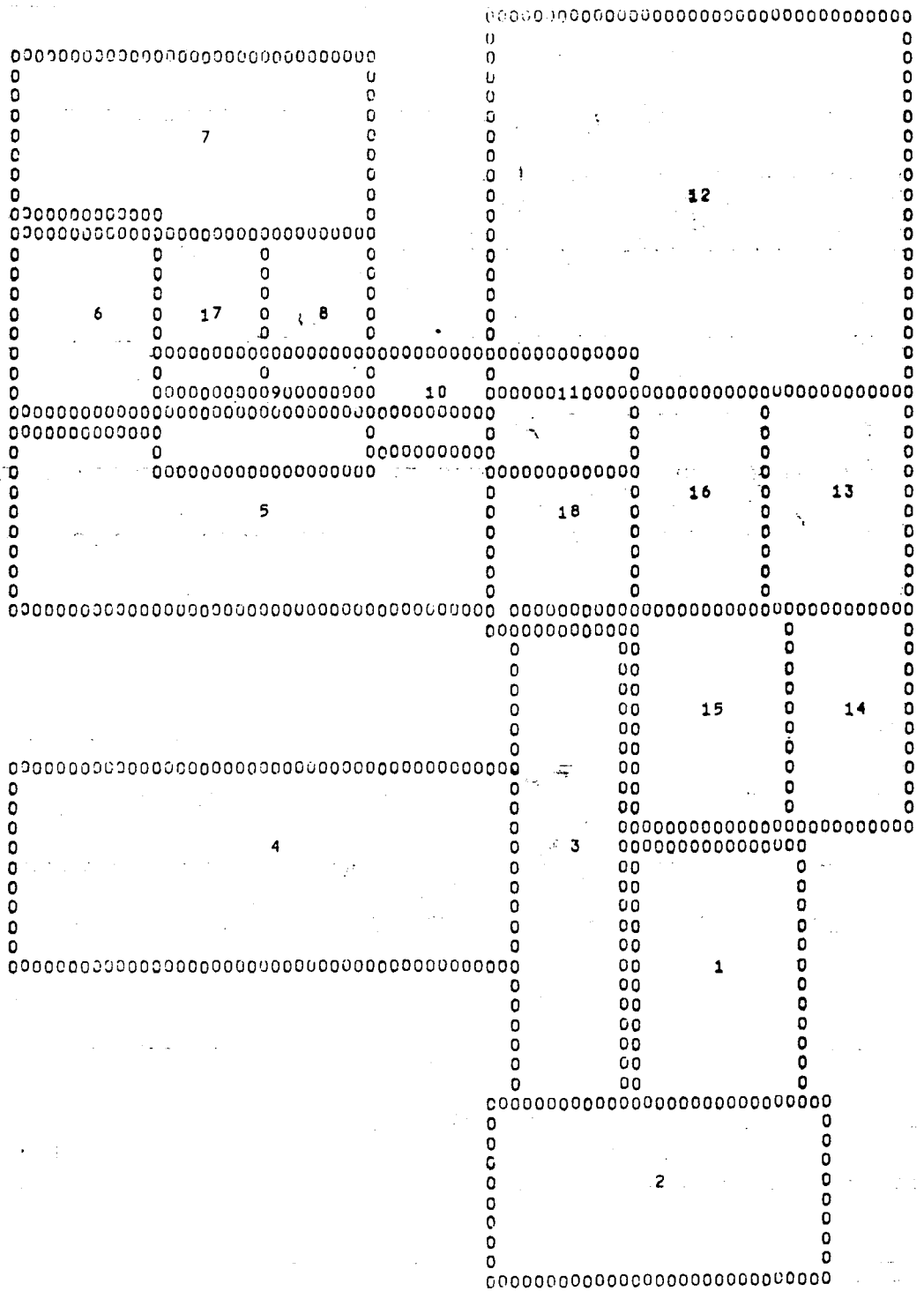
SOLUÇÃO 4 29



SOLUÇÃO N 11



SOLUCAO N 17



SOLUCAO N 18

After the precedent layouts were output interaction was initiated. Layouts numbered 11, 17, 18, 21, 25, and 26 were considered for alterations. For economy we only show the final layouts obtained, in the next series of layouts.

All alterations were made only local to each layout.

Layout 11. The alterations were made in two stages. On the whole, they consisted in exchanging the two tolerance intervals of each of spaces 2, and 5; of making then the two maximum dimensions of space 2 equal to 20 and 13; and of modifying the maximum horizontal and vertical dimensions of space 4 to 43 and 13, respectively.

Layout 17. Each of the two tolerance intervals of spaces 1, 15, and 8 were interchanged initially. Afterwards they were interchanged for space 12 also. The maximum dimensions, horizontal and vertical, of space 12 were augmented to 20 and 17, respectively. The final result is shown further on.

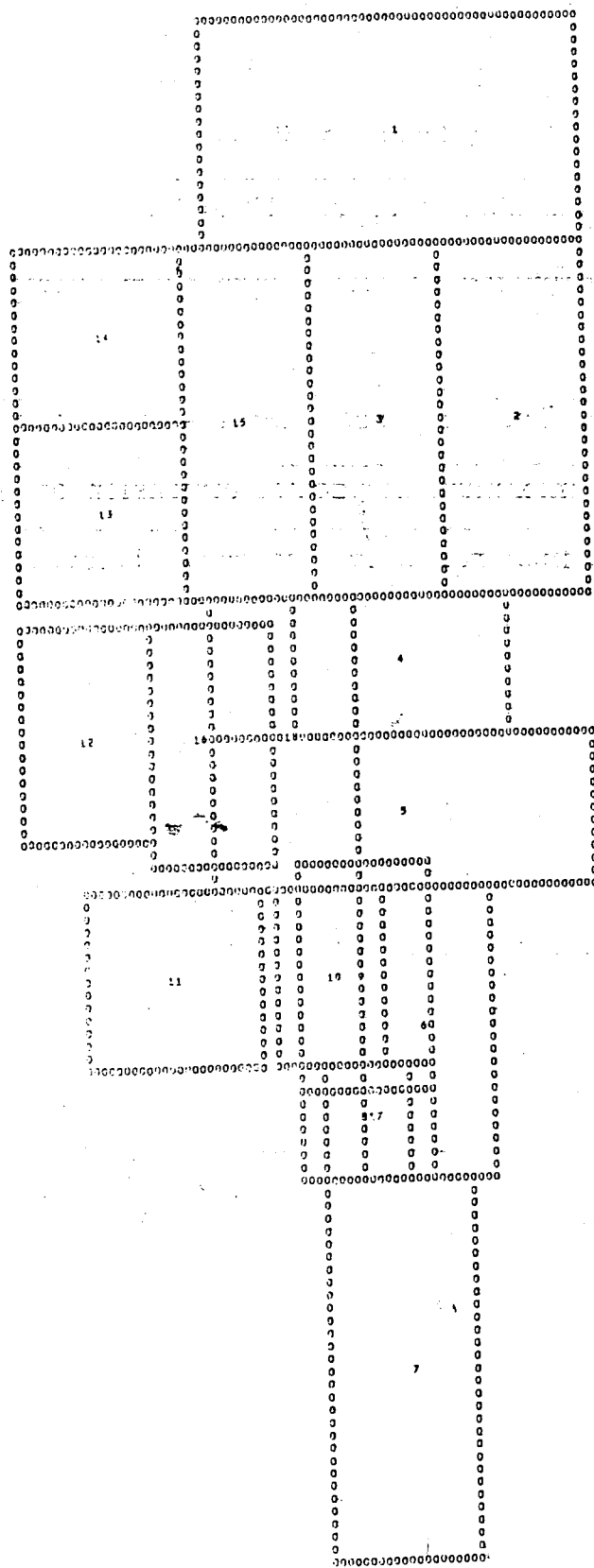
Layout 18. The tolerance intervals of spaces 1, 11, 15, and 18 were interchanged. The maximum dimensions, horizontal and vertical, of space 12 were made equal to 20 and 17, respectively. The final result is shown further on.

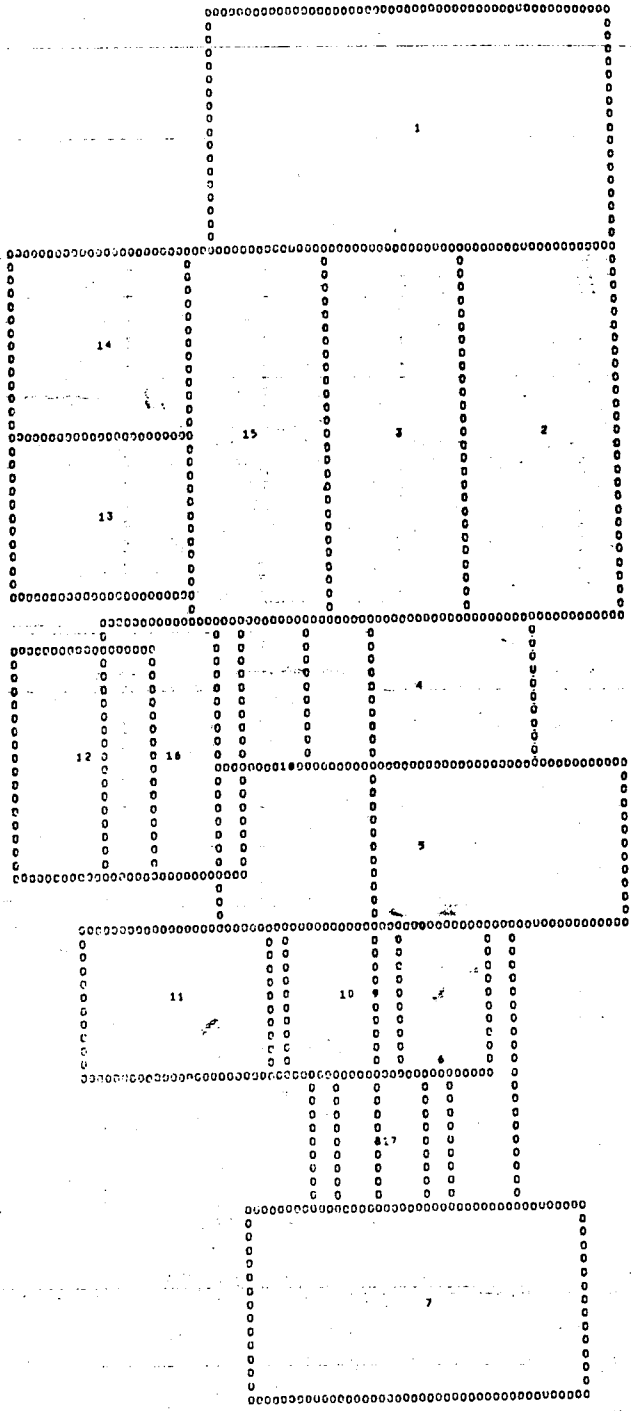
Layout 21. The tolerances of spaces 3, 4, and 5 were interchanged. The maximum horizontal dimension for spaces 13 and 14 was reduced to 8. The tolerances of spaces 4 and 5 were interchanged back to their initial assignment, and the tolerances of each of spaces 7 and 9 were interchanged. The

final resulting layout is shown further on.

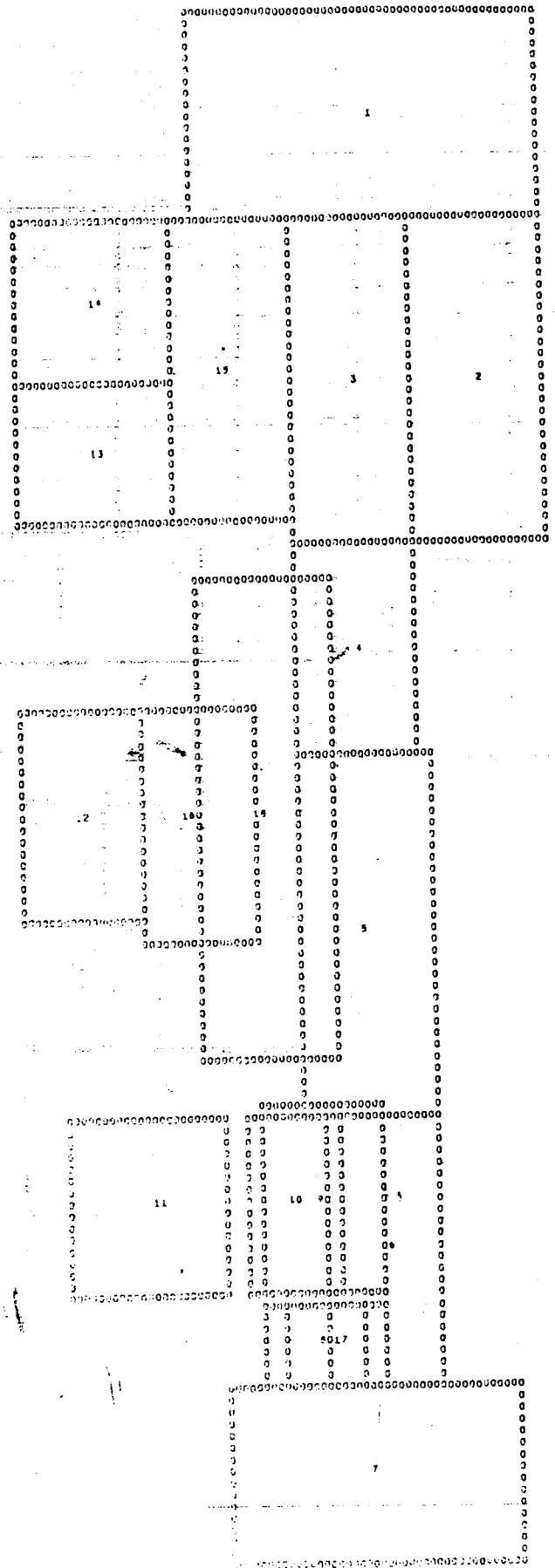
Layout 25. The tolerances of space 3 were interchanged. The maximum horizontal dimension of spaces 13 and 14 was reduced to 8. The final result is shown further on.

Layout 26. The tolerance intervals of spaces 3, 4, 5, and 9 were interchanged. The maximum horizontal dimension of spaces 13 and 14 was reduced to 8. The final result is also shown below.





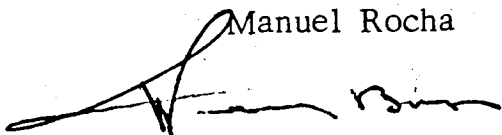
SOLUCAO N. 77



Lisboa, Laboratório Nacional de Engenharia Civil, 1 de Fevereiro de 1974.

○ ENGENHEIRO DIRECTOR,

Manuel Rocha



Eng^o. Ruy Gomes

P.6/ Chefe do Serviço de Edifícios

Luis Moniz Pereira

Luis Moniz Pereira

Engenheiro, Estagiário para Especialista na Divisão de Matemática Aplicada.

NUNO PORTAS

Arq^o. Nuno Portas

Chefe da Divisão de Arquitectura

Artur Ravara

Eng^o. Artur Ravara

Chefe da Divisão de Matemática

Aplicada