A WORKING STRATEGY
FOR GENERAL
SCHOOL SCHEDULING
PROCEDURE PTC(q, s); SET(n) q, integer (i)
    BEGIN
        integer j;
        FOR j = 0 TO n DO IF q(j) <= k THEN 
            BEGIN
                PTC(q AND k, s, j);
                PTC(q 0 s, E(j), j);
            END;
        END FOR
    END PROCEDURE

END
HARALD MICHAelsen

A WORKING STRATEGY FOR GENERAL SCHOOL SCHEDULING

Regnesenteret NTH
SINTEF
Til Bodil

Å legge en timeplan er dessverre ikke det samme som å kunne følge en timeplan.
PREFACE

The purpose of this work is to demonstrate a successful strategy for the master-schedule problem for schools while simultaneously defending the claim that the strategy is general for the problem in question. Just as a schedule itself is a compromise, this is also valid for this book:

a. It is desirable to achieve as general and exact formulations as possible but at the same time being able to transform these to an operating program. Elegant and simple formulations are esthetically attractive but they might easily result in very inefficient and poor algorithms.

b. The major part of any large programs system consists of relatively straightforward techniques. One must achieve a fair balance between necessary explanations of trivial procedures and simultaneously concentrate the main effort on the fundamental and more complicated principles.

c. One might easily be tempted to attack too large a problem area such that one goes astray without achieving a solution to any realistic problem.

These dilemmas result in a difficult "tightrope walking", and I do not expect that this is always successfully solved. It has been necessary to restrict this work entirely to the basic principles for school scheduling without bothering about other applications. If the choice between a theoretical or a somewhat pragmatic approach is problematic, the latter is usually preferred.

Any large application program ought to be seen in a broader perspective to get some understanding of the comprehensive concept called "computer methods". A particular program can of course only show some aspects (or facets) of this concept, and this work may serve as an example on the following:

a. Heuristic programming is not an arbitrary collection of ad hoc procedures, but a conscious effort to combine exact and intuitive principles.
b. To be able to formulate a complex problem, it is usually necessary to define a particular data structure and a corresponding set of operators typical for the problem area in question.

c. Man-machine interaction is a frequently used word, and two kinds of interaction are demonstrated by this work.

1. Time consuming and trivial operations are performed by the computer while central decisions and possibly final touches are performed by man.

2. Intermediate results from the computer may lead to new impulses which may have a large impact on the choice of solution method.

The former is an ordinary form of interaction, only pinpointing the fact that there is a practical limit for which functions ought to be done by a computer. The latter implies however, that the computer acts as an incentive on the problem solving capacity of man which among other things indicates that a number of complex problems ought to be solved in a strongly computer-oriented environment.

Obviously, alternatives to the strategy discussed here might result in a successful solution of the school scheduling problem. Even if these approaches are apparently quite different from the current one, I am completely convinced that they will reflect the same basic attitude towards the problem: A general and working strategy will be a synthesis of a readiness to refrain from mathematical completeness and a reluctance to use heuristic principles.

The assumed previous knowledge of the reader will be mentioned: The mathematical notation to be used will be defined independent of other works using logical operators. (But there is of course a similarity). It is assumed that the reader is familiar with conventional programming. Only summary mention is given of administrative procedures, and detailed explanation of the various
algorithms is usually not given. The scheduling problem is to be defined in a rather abstract way and some knowledge of practical structures is desirable (See [1]). The large number of parameters might probably seem confusing and thorough work is necessary to get a complete understanding.

Chapter 1 delimits the problem to be discussed and gives together with chapter 2 some general background. Chapter 3 gives a precise verbal formulation of the problem and a rough outline of the basic strategy. Even if the strategy is only roughly sketched, chapter 3 serves as a basis for the rest of this presentation. Chapter 4 defines the data structure and corresponding set of operators. Chapter 5 is a detailed discussion of the central conditions, and defines the framework for the strategy. Chapters 6-10 discuss other elements of the strategy, and chapters 5-10 define integrated the entire strategy for the time allocation. Chapter 11 gives a summary mention of other important features of the program. Chapter 12 is a short verbal summary of this work.

The development of the program in question has lasted for almost 5 years, and a promising idea is merely a valuable part of the accomplishment of a large project. Just as important are financial backing, programming, management of production-runs, discussions of practical problems, typing of manuscripts etc. In all these aspects a lot of people have given valuable help for which I am extremely grateful, but the list becomes too long to mention anyone in particular.

Instead I will address my thanks to SINTEF for making possible such unconventional and stimulating environments as REGNESENTRET NTH. This has been a major incentive in carrying out the task.

Trondheim, December 1970

Harald Michalsen
## CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PREFACE</td>
<td>II</td>
</tr>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>HISTORICAL SURVEY</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Brief survey of other methods and results</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Development of the scheduling project at Regnesentret NTH</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Information about the program system</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>VERBAL FORMALIZATION OF PROBLEM AND SOLVING METHOD</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Definitions basic to the scheduling problem</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>An outline of the basic strategy for solution of the scheduling problem</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>DATA STRUCTURE AND OPERATORS</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>Notation for the data structure</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>Definition of the data structure</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Operators for logical variables</td>
<td>46</td>
</tr>
<tr>
<td>4.4</td>
<td>Procedure set for the data structure</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>THE CENTRAL CONFLICT TYPES</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>Direct resource conflicts</td>
<td>69</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Necessary conditions</td>
<td>69</td>
</tr>
<tr>
<td>5.1.2</td>
<td>The TCs of the system</td>
<td>77</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Evaluation of the utility, G</td>
<td>86</td>
</tr>
<tr>
<td>5.1.4</td>
<td>A comment to the necessary and sufficient conditions for solution of the scheduling problem</td>
<td>98</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Modification of the conflict vector due to direct resource conflicts</td>
<td>102</td>
</tr>
<tr>
<td>5.2</td>
<td>Complex resource conflicts</td>
<td>106</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Necessary conditions</td>
<td>106</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Practical modifications</td>
<td>122</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Common features of complex resource conflicts</td>
<td>124</td>
</tr>
</tbody>
</table>
5.3. Variable period length (The period requirement) 124
5.3.1. Representation of the period requirement 125
5.3.2. Necessary conditions 130
5.4. Day-conflicts 137
5.4.1. Representation of day-conflicts 139
5.4.2. Necessary conditions 142
5.5. The form of the conflict picture 144

Chapter 6. THE ALLOCATION SEQUENCE 147

Chapter 7. CRITERIA FOR ALLOCATION OF AN ACTIVITY 157
7.1. Global criteria 157
7.2. Local criteria 161

Chapter 8. MODIFICATION OF DATA STRUCTURE DUE TO CHOSEN ALLOCATION 169
8.1. Modification of the prehistory 169
8.2. Generation of new conditions 172
8.2.1. Modification of data structure 179

Chapter 9. DESIRABLE REQUIREMENTS 184
9.1. Examples of representation of desirable requirements 193

Chapter 10. SPECIAL CONFLICT TYPES 200
10.1. Special complex conflicts 200
10.2. The continuity requirement 205
10.3. The split requirement 210
10.4. Other particular conflict types 211

Chapter 11. ADDITIONAL PROCEDURES TO THE TIME ALLOCATION 214
11.1. Error situations and manual adjustment 214
11.2. Room allocation 220
11.2.1. Principles for room distribution 221
11.2.2. Principles for room balancing 222

Chapter 12. SUMMARY AND CONCLUSION 229
APPENDIX 1: Abbreviations for variables, operators and concepts 234

APPENDIX 2: Literature references 245
1. INTRODUCTION

A good schedule is of prime importance for the pursuit of educational intentions, and school administrators have expressed a strong desire that the scheduling should partly be done by means of a computer program. The possibility of achieving considerable savings in this way has been known for quite a long time, but the results have been long in coming. Larger savings are anticipated than only those accruing from the automation of a tedious function which otherwise must be executed by well qualified and expensive personnel. It is just as important that one expects a saving of school resources such as teachers and classrooms, and that the students can be offered more nuanced education. From such objectives the importance of a good schedule is readily realized, and from a more humane point of view, it may be mentioned that most of the scheduling work must be done during the summer months, which makes it difficult for school principals to get their vacation.

During the last decade much work has been done on introducing new and more nuanced education structures. Consequently school administration has become more complicated, and it is often difficult to work out a satisfactory schedule. The current dynamic school structure indicates an increasingly complicated scheduling problem. Accordingly a computer program is not only a desirable alternative to manual scheduling, but in some cases it is actually a necessity to have a reasonable chance of satisfying actual requirements.

Simultaneously, the scheduling problem is attractive from a computer point of view. The problem is basically non-algorithmic; i.e., in practice no general and well-defined method exists. An analysis of the problem reveals that relatively complicated techniques are needed, and that a complete representation of a realistic problem must be done by means of a respectable number of parameters. Regardless of purely economical evaluations the problem is professionally and methodically challenging; since a successful solution of the problem will offer great possibilities in applying new methodologies. The school scheduling problem is a special case of the more general resource allocation problem, and this is one of the areas where the computer is expected to be particularly valuable. Methods for a general resource allocation problem will not
be analysed here. The scheduling problem for schools in itself is large enough, and even for this simplified problem there is a lack of formalization. The frame of this work is to find a general strategy for the scheduling problem, and it will simultaneously be formulated such that the imaginative reader may find other application areas for the strategy.

There is an ever increasing amount of literature on the scheduling problem, pointing out that the concept "school-scheduling" is being used to describe two rather different problems:

1. The master schedule problem
The students to be included in the different education groups (classes) is given beforehand. Furthermore, which teachers and room types are included in the various subjects are given. The actual problem is to allocate time and place to the various subjects consistent with the conditions to be satisfied by the schedule.

2. The sectioning problem
Time and place for the various activities are given beforehand. The schedule for the individual teacher is also given. The problem in this case is to determine which students shall be included in the various education groups (classes) in order to give as many as possible the opportunity of taking part in the education of their choice, and to utilize the resources of the school in the best way possible.

The former problem is the most common one, while the sectioning problem is especially relevant for large schools or universities where there is large freedom in designing the schedule of the individual student. The master schedule problem is regarded as definitely the most complicated of these, and in the following only that will be discussed. (It would lead too far to discuss differences and similarities between the two problems, and several other papers claim to have solved the sectioning problem satisfactorily.)

One of the great obstacles has been the lack of definition and formalization of the scheduling problem. The various school structures apparently differ considerably, which necessitates numerous special considerations. Experiments with alternative school structures
complicate the problem still further, and when various pedagogical motivations for a new school structure are introduced, it does not always seem that regard for what is realistically possible from the existing resources, has been evaluated. During the period we have been working with scheduling, we have had numerous discussions with school administrators about the requirements for a schedule. It has proved extremely difficult to find uniform requirements and still harder to define a so-called "optimal" schedule. This is comprehensible for two reasons; every schedule is a compromise, and a school principal evaluates a schedule from his subjective opinion of education and the requirements he can allow to be made from the freedom present in his own school. This implies that the concept "optimal schedule" changes from year to year for the same school. Provided that several schedules are made for the same basic data and that all absolute requirements are satisfied, it is often difficult to decide which schedule is the best one. This evaluation will be done from more qualitative criteria; i.e., the "optimization criteria" are different from the primary requirements for the schedule. In the opinion of the author the concept of an optimal schedule is a fiction; it would be far more meaningful to say that one seeks an acceptable schedule. A somewhat cynical and supercilious definition of an acceptable schedule is: A schedule comparable in quality with the most successful results of manual methods. Such a thought is opportunism; however, it points out the important fact that one should not expect to achieve the so-called ideal schedule, which seldom exists in practice.

It has been pointed out that the scheduling problem is challenging both from a methodical and economic point of view. The existing literature reflects this, as it roughly can be divided into two parts:

1. Design of pure mathematical models.
2. Simulation of manual methods or analog heuristic principles.

The advantage of a purely mathematical model is that there is a possibility of using familiar methods, and that the limitations of the chosen method may be evaluated. On the other hand, the model might be unrealistic, so that the real problem may be to find a corresponding school structure.
For a given practical problem the use of heuristic rules will ordinarily be more successful. This may lead to an operative program, but the disadvantage is that little is known about the general value of the program for other school structures, and in the case of failure it may be difficult to determine whether this is caused by the method or the structure of the problem.

Paradoxically, there is hardly any problem for which it is easier to find some sort of a method than just the scheduling problem. To accentuate this, two alternatives will be mentioned: A method may be defined which implies that all possible schedules are generated, and from some optimization criterion the best schedule(s) is found. This is irrep.achable from a theoretical point of view. Alternatively, if a given problem has very few constraints, a procedure may be employed which generates random numbers as a "heuristic" method for scheduling. Both principles are equally value-less from a practical viewpoint (but there are suggested methods suspiciously similar to one of the mentioned principles). More seriously, if a particular or simplified problem formulation is chosen, then it may be simple to design an operative program, and one could easily draw the wrong conclusion that the methodology is general.

The above argumentation leads to an important problem: Which criteria should be used for evaluating a scheduling method?

For a number of new computer applications it has turned out that the most suitable approach is not the purely mathematical one, but a combination of exact methods and more heuristic principles, due to the complexity of the problem. It should be apparent that several excellent ways of applying the computer in fact are based on transformation of qualitative and intuitive principles to "algorithms" where one cannot show that the "best" solution exists. The program system as a whole will then be heuristic. Accordingly, the only correct criterion for a method ought to be the operative value, defined particularly for the scheduling problem as follows:

1. How general is the method; i.e., which sets of requirements can be treated?
2. What practical results have been achieved, and how is the quality of the schedules compared to manually constructed ones?

3. Which are the possibilities for defining a problem? (In principle it is uninteresting how the schedules are printed out, but in practice this is also important.)

4. What does it cost to have a schedule made, and how much time is needed? Not only computer time, but also the necessary manual work must be taken into consideration.

5. Can the program easily be modified to consider new conflict types?

An abstract of the above criteria is:

**Will the schools use the program?** This rather pragmatic objective, in the opinion of the author, is far more descriptive of the value of a method than possible quantitative measures. The reason for this is the complexity of the problem, and analogies to the mentioned measure can easily be found, for instance for games such as chess and bridge. From a practical viewpoint neither of them are algorithmic, and the value of a strategic principle is naturally a function of the achieved results.

The frame of this work is in short:

1. To formulate the scheduling problem as generally as possible. To a large extent exact methods will be employed, but a combination of these and heuristic principles is necessary.

2. A large number of the conditions relevant for realistic school structures will be discussed. The primary goal is to clarify the integration of the different principles, whereas the actual design of an algorithm is of secondary interest.

3. Theoretical concepts must be transferred to algorithms which are economically justifiable and sufficient in practice.

4. The measure of the program system as a whole is the operative value.

This aim obviously implies that an operative program system must be designed, the development of which requires the solution of a number
of other problems in addition to finding the basic principles of scheduling. A summary of the various phases is given below:

1. How should the scheduling problem be formalized from the viewpoint of the school?

2. Which rules should be used for finding the self-contradictions occurring in the problem specification?

3. Which are the necessary data structure and operators?

4. Which strategic principles should be used for allocating time and place to the various activities?

5. Which (manual) rules should be used for adjusting an impossible problem definition or for compensating an incomplete strategy?

6. How should the final result be presented?

Each of the above phases represent a considerable amount of work, but in principle phases 1 and 4 are the most interesting. Phase 1 is discussed in [1], a book which in combination with the present one forms a unit. [1] is, however, based on some particular school structures and a number of conditions are not discussed. Nevertheless, it should give a fair impression of realistic school structures.

The "external" formalization of the scheduling problem defines which part of the scheduling will be done by the school, and which function will be taken over by the program. One important principle is that the school management still must exercise complete control of the vital decisions, while a program system deals with the most time consuming part of the work; i.e., allocation of time and place for the various subjects. The school management must still be able to decide necessary compromises; this may be done "iteratively" if the original specification is self-contradictory.

The present work concentrates mainly on phase 4; a condition for this is that phase 3 is also defined. It will be shown later that room allocation may be done approximately algorithmically when time allocation is done. Room allocation is therefore only given a summary mention, as the problem is solved by means of relatively
straightforward principles. Correspondingly, phases 2 and 6 involve a considerable amount of conventional programming, which is not mentioned here. Phase 5 is much based on practical experience, and the rules vary for the different school structures. The rules are intimately connected with the requirements for an acceptable schedule, which has more in common with phase 1 than phase 4. A complete discussion of manual adjustment possibilities is outside the scope of this work.
2. HISTORICAL SURVEY

2.1 Brief survey of other methods and results

The first papers on computer made scheduling appeared during the late 50's, and the related sectioning problem was the first one to be treated. The developed programs had to solve a considerable administrative problem, while the sectioning technique itself was relatively straightforward. The ability of the computer to cope with enormous amounts of information resulted in programs being at least equal to manual technique, and much work was saved. Eventually several methods for solution of the sectioning problem have been introduced. This is somewhat beyond the frame of the master scheduling problem, and references [9], [48], [49], [51] may serve as examples of the employed techniques.

The first mention of operative programs for the master scheduling problem is found in Appleby, Blake and Newman [4], Akerman [57], Berghuis, Heiden and Bakker [8], Holtz [29], Barraclough [6]. These papers discuss a purely heuristic approach, and eventually they have been succeeded by numerous similar methods. A detailed discussion of the various papers would lead too far afield. The formalization and complexity of the treated problems vary to a rather great extent, and characteristic basic elements of a heuristic solution method are pointed out below:

1. Administrative routines.

During the actual scheduling one must keep track of which teachers, classes and rooms have been blocked due to previous assignments. The most efficient way of doing this is by using logical matrices, and by means of a set of logical operations, possible periods for the various subjects are found.

2. Allocation rules (for the assignment-sequence of the subjects).

These rules are often defined by each subject being given an empirically determined parameter which shall describe the "difficulty" of allocating a subject. The most difficult subjects are allocated first. The allocation rules may also be directives, e.g.: double periods are assigned before all single ones, or all physical education is assigned first.
3. **Optimization functions (i.e., rules for which period will be chosen for the subjects).**

A number of different principles are employed. Some methods choose the first available period, or use random numbers for choosing periods, and others choose periods from certain qualitative criteria. A more quantitative way of choosing periods is by trying to minimize functions of the form:

\[
Q = \text{MIN}( \sum_{i \in K} \frac{P_i}{a_i})
\]

where:
- \(K\): The set of subjects getting new blockings when a subject is allocated.
- \(P_i\): Unassigned periods for subject \(i\).
- \(a_i\): Available periods for subject \(i\).

In practice several different (and complicated) versions of this function have been used.

4. **Back-tracking mechanisms.**

When assigning the subjects, sooner or later the situation will arise where it is no longer possible to allocate certain subjects. It is then assumed that one of the previous allocations is wrong, and rules are defined for returning to a previous situation for another attempt. (In principle an enormous branch structure of possibilities will be investigated, eliminating the branches which are unable to offer solutions. Obviously this implies extensive administrative routines. The process is time consuming and leads to extensive use of "trial and error"-methods.)

5. **Simple necessary conditions for solution.**

Such conditions require for instance that one must secure that the number of available hours for any class (teacher) at any time is at least equal to the number of unassigned hours for the class (teacher). These rules may be expanded to a certain degree, but on the whole the mentioned rules are simple and not generally formulated.

Most papers on heuristic methods report encouraging results as one has (possibly approximately) succeeded in scheduling realistic problems. However, with a few exceptions none of these schedules have been utilized. Here are summarized some objections against these purely heuristic methods:
Most decisions are made based on purely local criteria and empirical parameters. This may function well in particular cases, but not much is known about the generality. In case of an error situation occurring, it may just as well be due to the method as to a self-contradictory problem definition, since little is known about potential error situations for later assignments.

Parallel to the pragmatic treatment of the scheduling problem several theoretical approaches were made. Using a simplified model as a basis, attempts were made at setting up the necessary and sufficient conditions for the existence of a solution. The best known of such models is discussed by Gotlieb and Csima in [19] and [20]. Several papers present further work along this line, and attempts have been made at fitting the model to realistic models. See [14], [38], [39], [40]. Integrated, these papers represent an essential contribution to the formulation of a general strategy, and the following is a brief summary:

Gotlieb's model assumes that a schedule consists of a number of events where one teacher will meet one class (it is disregarded that a room is also included in the event). The number of time units where class i will meet teacher j is given by component $r_{ij}$ of matrix R (Requirement matrix). The schedule will be made within a frame of t time units, and by introducing a number of fictitious classes (teachers) and fictitious incidents it is provided that the system is symmetrical and tight (i.e., the number of teachers is n and the number of classes is n). Each teacher (class) partakes in t incidents.). The schedule is represented by a three-dimensional logical matrix A (Availability matrix) where classes, teachers, and time units are the three coordinate axes. A is defined as:

$$a_{ijk} = \begin{cases} 1, & \text{if class } i \text{ can meet teacher } j \text{ in period } k \\ 0, & \text{otherwise} \end{cases}$$

When one of the three mentioned parameters is constant, the following conditions may be verbally formulated:
1. The number of time units where class $i$ can meet a selection of teachers $J$ is denoted $a(i,J)$, and the number of time units where class $i$ must meet these teachers is denoted $r(i,J)^K$. A necessary condition for existence of a solution is that $a(i,J) \geq r(i,J)$ at any time. If particularly $a(i,J)=r(i,J)$, then class $i$ cannot meet teachers not included in selection $J$ for the time units represented by $a(i,J)$.

2. An identical condition must be valid for teacher $j$ and a selection $I$ of the classes to be met by him.

3. For any time unit it must be valid that any selection of classes $I$ must be able to meet a corresponding selection of teachers $J$.

The mentioned conditions are given a mathematical formulation in [12] and [19] while [20] and [38] define methods for the extensive search process necessary to find out whether the mentioned conditions are satisfied. (This process is denoted tight-set search.) By means of known theorems [12], it was possible to prove that there always exists a solution, and it was further assumed that the defined method ensured that both the necessary and sufficient conditions for solution were satisfied. In [19] a rather hypothetical example of the method not being adequate was pointed out. This example did not correspond with the assumptions for the method, but in [39] it was decisively proved how one could, by means of an arbitrary allocation sequence, provide for the satisfaction of Gotlieb's conditions for each stage, and that it was possible to perform an allocation which would imply that a solution no longer existed. Accordingly, Gotlieb's conditions are only necessary.

However, the method is still of practical value, as only exceptional error situations occurred. [40] discusses a practical application of the method. (These results are encouraging, but as far as is known the work has not been continued.)

Certain aspects of Gotlieb's method have been criticized:

1. A complete tight set search is very time consuming, and extensive computing is done for choosing a period for a subject

$^K r(i,J)$ is the number of currently unassigned periods.
even if the actual choice is rather indifferent.

2. The model is much too simple to be able to consider the problems really causing difficulties in practice. One assumes too easily that the "week problem" may be divided into a number of "day problems", one partly neglects room problems, and attempts to avoid the considerable practical difficulties like for instance parallel assignments of subjects by preallocation, thus solving the real problem before actually applying the method.

One should be able to avoid the first objection by slightly resigning from the requirement of a complete tight set search and taking conscious risks, but still cover the vital situations. The author is in agreement with the objections to the generality of the method, and furthermore regards the chosen representation to be unsuitable for realistic problems. However, the method is very important in spite of the objections. It introduces necessary conditions in a lucid way, and makes it possible to limit the solution space globally. (Even if all resources included in an event are available in one certain time unit, there is no guarantee that a solution exists which incorporates the mentioned possibility. By means of Gotlieb's method one is frequently able to make definite statements about this.) Gotlieb's method replaces the approximately local viewpoint for heuristic methods with global conditions. An attempt at generalizing these principles in order to define a method for realistic structures seems appealing. To be able to do that one must be ready to sacrifice some of the mathematical completeness possible for a simple model.

Gotlieb's method is based on combinatorial principles. (By that is meant methods where the major part of the data structure are logical matrices, and the various possibilities and consequences are found by means of logical operations. It is not a method where by means of combinatorics all possibilities are generated.) Most of the literature deals with heuristic or combinatorial methods or a combination of these. In view of achieved results this seems to be the only reasonable approach. The problem representation changes, but most methods use the conflict matrix as the fundamental data structure (the conflict matrix states which events cannot occur simultaneously.
in pairs). A natural consequence of the data structure is the need for a suitable representation of logical variables in the computer, and that an operator set is defined for the chosen data representation.

Attempts have been made at solving the scheduling problem by means of known mathematical methods; e.g., in some papers versions of linear programming are introduced (see for example [5], [30], [33]). It is highly desirable that the scheduling problem could be connected to a known formalism; in that way one might be able to determine both the necessary and sufficient conditions for solution. It has not been definitely shown that linear programming leads to operative program systems, and it seems that this formalism is both artificial and of limited importance to the scheduling problem. The most obvious objections are:

1. The problem must be represented by an enormous number of variables (e.g. $10^8$) and such huge models are unsolvable at present. The only way of avoiding the problem is to transfer the difficulties to the problem definition (see for instance [33]). By combining a large number of different subjects and considering them as a unit, one may reduce the number of variables. However, in doing this, an initial limitation of the solution space has been done due to the chosen method and not due to the problem itself.

2. The simplest (and most important) conditions may be formulated by means of a set of equations to be solved by linear programming. However, as will be shown later, the scheduling problem consists of numerous different conditions, and it is difficult to understand how one should be able to represent all of these within a practical frame. Furthermore, the various conditions must be ranked; a problem that cannot directly be treated by means of linear programming.

3. Linear programming employs an optimization function, and the ones mentioned in literature are far too simple and have little relevance to the requirements made in practice.
A number of papers mention use of principles from the graphtheory (see [12], [55]), but none of these go any further than treating the very simplest conditions.

The examination problem may be considered as a simplified case of the scheduling problem, and it is discussed in several papers (see for instance [10], [11], [18], [28]). The techniques employed for solving it are in principle similar to those used for the scheduling problem, and special mention appears superfluous.

Extensive literature references about the scheduling problem are offered for instance in [17] and [21]. Creating a clear image of the various principles may be complicated; the general impression is however as follows:

Many of the papers concentrate on formalizing the problem situation. This reflects the fact that the scheduling problem must be represented by a large number of parameters, that the schools themselves have difficulties in defining the problem, and that the current school structures are not particularly "stationary". The various papers are relatively independent. It is easy to lose track of the basic principles for solving the problem and get lost in extensive explanations of necessary administrative routines. Nobody has succeeded in formulating a complete mathematical method for a realistic problem, neither is documented a program system operative on a large scale. From a principle point of view the most interesting goal is possibly to design a method which is able to make one schedule, but from a practical and economic viewpoint such a program is uninteresting until it can be applied to handle a large number of schools. The practical difficulties involved in this must not be underestimated.

2.2 Development of the scheduling project at Regnesentret NTH
(The Computing Centre at the Technical University of Norway).

Various versions of the program system have been in operative use for the past five years. The original objective was to develop a program to satisfy the local NTH requirements. This work started in January 1966, and as early as summer 1966 a computer made schedule was introduced. It was considered quite acceptable and in several ways superior to a manual schedule.
The first program version was based on relatively simple heuristic principles (they were, however, similar to the more general strategy eventually developed). The objective was reformulated to design an operative program system for any Norwegian school structure and if possible find general principles for solution of the scheduling problem. Initially it was essential to examine as many different school structures as possible and mainly concentrate on the most complicated ones; during the development phase the number of schedules actually produced was of secondary interest. The model structure was the Norwegian "ungdomsskole"; a relatively complicated streamed structure.

From the start it was chosen to use data representing real schools, and the quality of the computer made schedules were compared to the manual schedules used by the schools. The figures below show the development of the project; i.e., the number of schools actually scheduled per year:

- 1966: 2 schedules
- 1967: 4 "
- 1968: 27 "
- 1969: 83 "
- 1970: 100 "

Grouping the various schools in school structures is rather complicated since a number of schools represent several types of schools. Schedules have been made for university (NTH only), technical schools (tekniske skoler), teachers' colleges (lærerskoler), high-schools (gymnas, realskoler), comprehensive schools (ungdomsskoler), elementary schools (barneskoler), and several combinations of these types of schools. In 1970 schedules have also been made for Swedish and Danish school structures corresponding to the Norwegian comprehensive school. These structures vary considerably, and totally, a wide spectrum of school structures has been studied. (Most of the scheduling has been done for Norwegian comprehensive schools.)

---

* This survey does not include schedules made only for comparison with manual schedules nor the data sets designed for investigating particular properties of the program. If these were included the program system has been used for making 750-1000 schedules.

** The English terms used here do not necessarily characterize the actual Norwegian types of schools (in parentheses).
A schedule has always been obtained when applying the program; i.e., usually the schedule is not found by means of the program system alone, but the problem is in some cases modified manually when self-contradictory, or the found (partial) solution is adjusted manually. A manual adjustment very seldom requires more than 8 hours work for any of the examined school structures.

More than 95% of the computer made schedules have been used (possibly modified). The majority of cases where a schedule was not accepted were due to factors outside the program. Naturally, during the development period weaknesses of the program have been discovered, but at present it is considered completely satisfactory from an operative point of view.

To facilitate a qualitative evaluation of the program the school principals were asked to evaluate their schedules. They made a rough qualitative classification of a number of criteria characterising a good schedule. The most important of these parameters are presented in a table, which represents average figures for all types of schools:

<table>
<thead>
<tr>
<th>Quality of computer made schedules</th>
<th>Good</th>
<th>Medium</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student schedules for the entire school</td>
<td>50%</td>
<td>45%</td>
<td>5%</td>
</tr>
<tr>
<td>Teacher schedules</td>
<td>45%</td>
<td>45%</td>
<td>10%</td>
</tr>
<tr>
<td>Room utilization</td>
<td>75%</td>
<td>20%</td>
<td>5%</td>
</tr>
<tr>
<td>Room distribution</td>
<td>50%</td>
<td>40%</td>
<td>10%</td>
</tr>
</tbody>
</table>

The evaluation is based on figures from 160 schedules. The result is satisfactory, in particular since some of the schedules for 1970 are not included, which would probably have improved the result. (Several quality improvements of the program were introduced in 1970.)

One should not expect all objections eventually to vanish. It is always possible that the program system does not make the compromises most readily accepted by the schools, and qualitative evaluations always imply some dissatisfaction. If the dissatisfaction can be limited to less than 5%, the program system should be considered satisfactory from an operative viewpoint. It would lead too far afield to compare the quality of manual and computer made schedules.
The main impression is that by means of the program system the requirements considered essential are more easily satisfied, while a manual scheduler has better possibilities for utilizing particular circumstances. (To compensate for this the computer made schedules can be adjusted manually.) Newspapers tend to characterize the program system as "studentfriendly" (as opposed to "teacherfriendly"). It would be more appropriate to say that the program gives higher priority to the most important requirements. As a natural consequence the advantages of computer made schedules increase as the problem becomes more complicated.

Unmodified versions of the program system have been successfully applied to new types of schools. In other cases certain simple program modifications have been necessary. However, not enough experience has been obtained to claim that the program system will work for all schools. The basic limitations of the methodology will be shown later, but within these limitations it is hopefully not preposterous to claim that the methodology is general. (There are, however, schools where the schedule is determined from numerous, rather particular, conditions. As a result a "tailor-made" version of the program system is required, and whether this is appropriate is an economical question.)

On November 1st, 1969, Regnesentret NTH sent a status report on the scheduling project to Det norske forsøksrådet for skoleverket (The Norwegian school ministry), which agreed to the conclusion of the report:

> Some years ago the question was: Is it possible to use computers for scheduling? We claim to have given a definite positive answer to that question, in spite of the fact that one could raise certain objections to the current program system. However, these ought to be easily solvable, and a large majority of schedules are satisfactorily solved by the current program system.

> In our opinion the computer is the only realistic alternative for future scheduling. The current problem is not whether the computer should be used, but how soon the schools will utilize this alternative entirely. This depends primarily on the
education offered to school administrators on computer made scheduling, and also on how quickly the program can be improved plus an environment for management of a heavy production load.

The reservations concerning the quality of the program system mentioned in the above conclusion apply to a previous version of the program. At present the quality of the program is considered satisfactory. The cited conclusion shows the primary aim of this work.

2.3 Information about the program system.

Totally the program consists of some 15,000-20,000 FORTRAN instructions. In addition a procedure set for treating logical variables (see chapter 4) has been designed. The program is written for UNIVAC 1108 (or 1107). The total number of data for a medium sized school (some 20 classes) is 1-1 million variables. However, the majority is binary variables, and the main part of data can be stored in core.

Run time for a schedule today is between 3 and 5 minutes processing time (N.B. not clock time) on UNIVAC 1108. One must expect that most data specifications are wrongly or self-contradictorily specified. As a result some reruns are necessary, which increases the total run time by a factor of 1.5-2. (The early versions of the program were far from optimal as regards time usage. This was eventually corrected, which resulted in a modest time usage. During the last years the program has gradually become more general, and the run time has increased significantly. On the other hand the quality of computer made schedules has considerably improved, and the amount of manual work has been reduced.)

The largest schedules made hitherto consisted of 50 classes (NTH is somewhat larger but also somewhat special; it consists of some 3500 students and some 500 teachers). This is, however, no upper limit for how large schedules can be handled by the program system. Run time for a schedule increases approximately by a factor $k_1 c^{k_2}$ where $c$ symbolizes the number of classes and $k_1$ and $k_2$ are constants ($1 < k_2 < 2$).
3. VERBAL FORMALIZATION OF PROBLEM AND SOLVING METHOD

Summary

The most important concepts are defined, and the problem area is delimited. The basic strategy for solving the scheduling problem is outlined.

3.1 Definitions basic to the scheduling problem

Schedule: A record of the relative time sequence of a limited and defined set of events.

System: The set of events for which a schedule is required.

Resources: The set of basic elements in a system for which a schedule is required.

For a school this will be the set of teachers, student groups, teaching rooms, and mobile teaching equipment which make up the school. By a student group is meant a number of students for whom one can prescribe all simultaneous education, i.e., usually a student group will be equivalent to a class.

Activity: An event which in a given number of time periods of given length, requires that a given part of the total resources is available for this event.

An activity makes claims on the resources. There are two kinds of requirements:

Direct resource requirements: A certain resource is included in the activity.

Complex resource requirements: A resource from a defined subset of the total resources is included in the activity.

The teachers and student groups forming an activity are usually defined by direct resource requirements; i.e., a certain teacher is prescribed for the various subjects for a student group. The rooms included in an activity may be defined by requirements of both kinds. In those cases where an activity requires one or more specified rooms, this would be a direct resource requirement, and when an activity requires a room of a particular room type this would be a complex resource requirement. By room type is meant a group of...
rooms which may be considered equally suited (or acceptable) for certain activities.

It might be tempting to consider a direct resource requirement as a special case of a complex resource requirement, but a direct resource requirement has a number of important consequences not directly valid for a complex resource requirement. These consequences are important for the choice of solution method. Typically, the schedule of every school includes a large number of direct resource requirements.

Time unit: The highest common measure for the length of the time periods defined for the activities of the system.

In a school, the time unit is usually identical with a teaching hour, and these will be used as synonymous concepts or the time unit will simply be called an hour (this does not necessarily mean a clock hour).

Period: The period of a time sequence will be the relation between the length of the time sequence and the time unit. Per definition this is an integer number.

For a school, the concepts single period, double period, triple period will be used if the periods are 1, 2, or 3. The largest period for a school is a small number. This is an important factor relevant for storage space, computer time, and limiting of possibilities for allocation.

Period division: The set of periods for the time sequences connected with an activity.

Interval: A set of contiguous time units to which it is necessary to have a reference due to the structure of the problem.

One interval for a school schedule will be all time units in a week, another interval will be all time units in a day, and morning and afternoon lessons might be defined as intervals of their own. To distinguish between the various intervals, names as week, day, morning, etc. are used.

Time frame: A hierarchy of intervals so that no subinterval is a component of more than one interval on a higher level. The structure of the problem defines the design of the hierarchy.
There will be two main types of schedules: **Cyclic plan** which is repeated after a given time frame, and **non-cyclic plan** which is only used once. The length of the time frame is usually given beforehand when a cyclic plan is used, whereas when a non-cyclic plan is used, a requirement may be that the length of the time frame shall be minimized. An ordinary school schedule is cyclic, whereas an examination plan for example may be non-cyclic. Whether the length of the time frame is constant or variable is no essential restriction, although only constant time frame will be considered here.

It might be assumed as an axiom that no solution exists which satisfies all requirements for a realistic schedule. A schedule is therefore a compromise solution between a large number of conflicts. **These conflicts are of a hierarchical nature; i.e., all requirements are not equally strict.** One may clearly distinguish between a set of absolute requirements and a set of desirable requirements. By an absolute requirement is meant a requirement which has to be satisfied, and by a desirable requirement a requirement which ought to be satisfied. A desirable requirement is usually formed as a qualitative criterion, and it must be assumed that it is possible to rank the desirable requirements.

The main kinds of conditions are:

1. **Direct resource conflicts**: No resource can be part of more than one activity simultaneously.

   This condition is a consequence of the definition of an activity, and for a school this may be transformed into the following conditions:

   - **Student conflicts**: No student can be taught more than one subject simultaneously.
   - **Teacher conflicts**: No teacher can teach more than one subject simultaneously.
   - **Room conflicts**: No room can be used for teaching of more than one subject simultaneously.

2. **Complex resource conflict**: The number of resources from a subset of the total resources which may be allocated to the same time unit will be at most the number of resources of which the subset consists.
A typical complex resource conflict for a school is:

**Room requirement:** Each subject must be taught in sufficiently large rooms of correct (or acceptable) room type.

This requirement implies that one does not completely decide beforehand which rooms are to be included in an activity, and in addition to time allocation of activities, room allocation will also be necessary. A complex resource conflict is a far weaker condition than a direct resource conflict, and any complex resource conflict necessitates allocation of more factors than time for the activities.

(3.3) **Interval conflict:** Within a defined time interval one will forbid allocation of more than one period of a defined subset of the total number of activities, or one will alternatively demand that all periods of a given set of activities are allocated within the same time interval.

Usually allocation of more than one period of a set of activities to the same interval is forbidden, and typical conditions for a school would be:

**Period requirements:** No activity can have more than one period in one day, and all activities must have the period division determined beforehand.

**Day conflicts:** A set of activities makes various claims on the resources, but for one or more student groups these represent the same subject. Consequently, it is either forbidden to allocate more than one of these on one day, or it is demanded that the set of activities is allocated sequentially on the same day.

**Pedagogic requirements:** Activities within a set are considered closely related, and one forbids allocation of more than one of these on one day (for example, double hours in written subjects are considered heavy, and are not to be taught on the same day. Such conditions may also be formulated as desirable requirements.).

In those cases where an interval conflict forbids allocation of several activities within the same interval, this has the same relation for a defined set of intervals (days) as a direct resource conflict has for the time units. The latter states which activities are conflicting during the same hour, whereas the former states which activities are conflicting during the same day.
Continuity requirements: If one or several resources from a given subset of the total resources are allocated to the time units \( j-1 \) and \( j+1 \) and both of these belong to the same interval, one requires that the resources also are to be among the activities being allocated to time unit \( j \).

In other words, certain resources must be used continuously within certain time intervals, for example:

**Continuity requirement for students:** All students shall have continuous education every day, i.e., no in-between hours.

This requirement is not absolute for all school structures, and for tight systems (i.e., the number of hours for each student equals the number of hours for the time frame) this requirement is obvious. For slack systems this requirement may lead to certain problems.

Initial requirements: Any acceptable solution for a schedule must be designed so that a defined set of activities is allocated to given time intervals, or alternatively that a given set of resources cannot be used in certain time intervals.

These requirements are due to external factors, i.e., the interaction between the system and its environment, and the requirements will prescribe a partial plan for a subset of the total resources. Typical examples of initial requirements are:

**Preassignment:** Certain activities require allocation to certain hours, for example in those cases where some resources are shared between several schools, or a teacher is bound by work elsewhere. The possibilities allowed for data specification may also necessitate preassignment.

**Blocking:** Some of the resources are not available in certain intervals, or allocation of certain activities in certain intervals is forbidden due to pedagogical reasons.

**Intermission:** For some school structures it will be required that within certain intervals each student must have at least one hour free for lunch break.

The motivation for the various initial requirements may be different, whereas the practical consequences are nearly alike. The above initial requirements are absolute requirements. Initial requirements
may also be formulated as desirable, and this is classified as a quality criterion.

(3.6) **Sequence requirements**: For certain subsets of the activities the relative time sequence is prescribed, or alternatively it is demanded that certain activities must be allocated to the same time.

An activity may accordingly have precedence, incidence, or subsedence relation to other activities. In a cyclic schedule, this will be a rare requirement, whereas for other resource allocation problems it might be a dominating requirement. The precedence and subsedence relations found in practice for schools have been simple. The sequence requirement will therefore be treated superficially. However, relatively complicated incidence relations have proved to be necessary. These are usually avoided by combining different activities into one because these are to be allocated to the same time. In some cases it cannot be defined exactly how the activities shall be coupled, and a conflict may develop with the period requirement for the individual activities.

**Split requirements**: For some subjects, the students of a class are divided into two or several groups for individual education. For a tight system it must be required that when one of the groups has lessons, the other groups must be instructed in some subject simultaneously. The criteria for splitting will change from subject to subject, for example division according to sex or because of a common choice of subjects. It must be assumed that the same grouping principle is valid for at least two subjects to make a split requirement meaningful.

**Parallel requirements**: For tight streamed school structures it must be required that certain subjects are taught simultaneously, and that students from several classes take part. This is due to level-divided education and combination of the optional subjects. Usually it will appear explicitly from the data specification (i.e. definition of the activities) that these subjects are taught simultaneously.

**Alternating education**: Some schools have subjects which are only taught every other week. This may be group- as well as class education. This may lead to incidence relations, due to either a tight system or to the continuity requirement.
Precedence relations: In a few cases education in one activity depends on education in another activity being given shortly beforehand; e.g., certain laboratory exercises depend on the laboratory being made ready the day before the actual education takes place.

The above incidence relations may lead to complex requirements, which preferably should be avoided. When it is possible to prescribe unambiguously all simultaneous lessons for all students in a number of classes, incidence relations are avoided by suitable activity definitions.

(3.7) Quality criteria: The relative time sequence of all activities which include a certain resource is called a pattern. For the set of possible patterns for a resource, a number of properties are defined as desirable. These properties are qualitatively defined, and they vary for the different resources. It must be assumed that a certain ranking of the various properties may take place.

These requirements will be the desirable requirements of the system, and for the structure in question these will be qualitatively defined, for example: Certain sets of activities should be as sequential as possible, some activities should be allocated as evenly as possible, or in such a way that they are in accordance with a continuity requirement. For a complex resource requirement, it may often be desired that a certain resource is used as much as possible, and for various activities certain time intervals may be more desirable than others. It is difficult (or impossible) to find a quantitative measure for the various properties; in other words, an unambiguous "optimal criterion" cannot be defined. However, the various properties may be ranked, and on this basis a method is defined trying to fulfil as many as possible of the desirable requirements, and at the same time give priority to the essential requirements. For the various systems, it must be possible to make a subjective ranking, and it is important that the methodology should be flexible on this point; i.e. the solution process may be iterative.

For a school, the quality criteria will represent educational and administrative aspects. Here are a few examples:
Pedagogical Considerations: Some principles defining properties of a good schedule are: Even work load every day, difficult subjects allocated to the best education time, the same subject not getting only bad education time, every class must get the desired special rooms as often as possible, etc. The pedagogical considerations are compound, and as there may be special views upon this, it must be possible to make subjective rankings.

Teacher Considerations: A teacher will often have particular desires about his teaching time, and as a rule it is desirable to have continuous schedules and at the same time to avoid overloading one particular day. The teachers' desires may often be in conflict with more central requirements, and some teachers will probably always be discontent with their schedules. The teacher's range of subjects decides to which extent his desires can be fulfilled. Particularly awkward teacher schedules ought to be modified manually.

Hour Priority: Some hours during the day are considered good teaching hours. Difficult subjects are usually allocated to these hours, and with otherwise equal conditions one will take maximum advantage of them. For all structures, the situation will be that some subjects are more critical than others regarding allocation, e.g. some subjects should not be allocated to Monday, and if one is forced to use boundary hours, they should preferably be used for some special subjects.

Room Allocation: The subjects should as often as possible be given required special rooms. Some rooms are overloaded, and the use of these should be assigned to the classes as evenly as possible; at the same time one would also want to maximize the utilization of the rooms. Moving between classrooms should be minimized, and one class should preferably have the same classroom for all subjects.

Interrupts in Education: For those schools which do not have continuity requirements, one will usually try to minimize interrupts in education. This is analogous to the main rule of the teachers' schedule.

Customs: If possible, the different subjects should keep the same rooms as used the previous year. This may reduce spreading of the education over a large area. At the same time one eliminates the
criticism which always results if an existing system is changed too much. Corresponding arguments may be claimed for utilization of teachers, but this will often be formulated as an absolute requirement.

A school structure may contain other quality criteria. But if more requirements are introduced, there will be more conflict possibilities, and the ranking of the quality criteria becomes vague. Accordingly, the concept "acceptable schedule" becomes diffuse, and a large number of quality criteria may lead to neglect of more essential quality requirements.

From the set of requirements and activity definitions for the various systems, these may be classified into school structures. There will be no attempt to make a nuanced grading, but certain school structures may differ characteristically, and this will have consequences for the choice of a suitable method. The following rough division will be used here:

1. **Class structure**: Most of the activities are defined by a teacher teaching one class (student group) in one room; i.e., each activity consists of three resources. A certain regrouping and coupling of student groups may be done for a small number of the activities.

2. **Streamed structure**: A stream is defined as a set of classes subdivided into a number of student groups. The classes of one stream are mostly taking part in the same activities. In some activities there is strictly class education, whereas in other activities the students are regrouped within the same stream. This is due to level-divided education, optional courses or division of sexes. There may also occur couplings of students belonging to different streams, but this is rare. A level-divided structure leads to the activities being defined by a large number of resources.

3. **University structure**: In principle, this is analogous to a streamed structure, but there are few teacher conflicts and a large number of student conflicts. The student schedules are a slack system, and there is a large percentage of special-rooms. On the whole the university structure is more slack than the streamed structure, but complicating factors will be: A large number of special considerations for each activity, variable room size, geographical
considerations, etc. The qualitative criteria must therefore be formulated in a different way.

The most general and complicated of these structures is the streamed structure. (The other structures are, technically speaking, special cases of the streamed structure.)

The following terms will be used for the way an activity is usually defined:

**Simple activity:** The activity consists of one class, one teacher, and one room.

**Compound activity:** The activity consists of several educational groups.

**Class activity:** All student groups which take part in the activity belong to the same class.

**Parallel activity:** All student groups forming a stream take part in the activity.

**Linked activity:** Two (or several) successive activities.

**X-activity:** An activity which is not taught every week. An X-activity may for example be taught every other week or only part of the school year.

The classification of the most important kinds of conflicts characterizes and limits the problem area which is being treated, and this may be defined as:

\[
\text{THE SCHEDULING PROBLEM OF A SCHOOL IS:}
\]

\[
\begin{align*}
\text{Within a given time frame, time (and place) shall be allocated to a defined set of activities in such a way that all absolute requirements are fulfilled, and all desirable requirements are fulfilled in the best way possible based on a given ranking.}
\end{align*}
\]

The above definitions indicate that the scheduling problem is a special case of a more general problem area called resource allocation. It is beyond the scope of this work to judge the value of the methods developed here for an analogous approach to other areas. One may hope that the basic principles and methods are more general than the area treated here.
The most important limitations to a more general approach involve the special assumptions for a school schedule, these are:

a. Direct resource conflicts play a dominating part.
b. The sequence requirement is of secondary importance.
c. The maximum period length is a small figure (this limitation is not as important as a and b).

3.2 An outline of basic strategy for solution of the scheduling problem

It has been mentioned that the primary goal is to define a strategy which can be realised through an operative program system for general and realistic school structures.

The methods that have been developed are based partly on combinatorial and partly on heuristic principles. Axiomatically it may be assumed that:

It is pointless (and probably impossible) to define the necessary and sufficient conditions for the existence of a solution of a general scheduling problem.

It will be demonstrated later that the methodology mentioned here only gives the most important necessary conditions for the existence of a solution based on the absolute requirements. When, in addition, the qualitative requirements are to be considered, the above assumption should be self-evident.

This is, however, no objection to defining a method which is completely satisfactory from an operative viewpoint. This should, of course, be thoroughly documented by the achieved practical results. (See chapter 2.)

Naturally it is important that the limitations of a method are examined as thoroughly as possible. This will be done in two ways:

1. Necessary requirements for solution of the scheduling problem are defined in as general a formulation as possible. By quantitative and/or qualitative methods, the limitations of a given formulation will be evaluated.
2. A general formulation is transformed to operative (and approximate) algorithms. This is done from an evaluation of the practical consequences of a requirement, consideration for the necessary storage space and computer time, and the economic competitive ability of a method.

The second part of the task is just as difficult as the first, because it requires good knowledge of realistic models. It has been pointed out that combinatorial principles easily lead to computer time increasing exponentially with the number of parameters in the system. The most elegant formulations may easily turn out to be illusions, and a method which lacks realistic proportions is without value. This fact plays an important role regarding the presentation, and implies among other things that the two parts of the task are not clearly separated. Practical limitations may lead to the basic principles becoming less distinct. The author realizes this danger, but is willing to sacrifice elegance and esthetic principles to clarify the most important operative principles.

When accepting that complete sufficiency conditions are not known, it follows naturally that the activities must be allocated sequentially (or step by step). From the definition of a problem a partial schedule may be known, and all requirements which are to be fulfilled in the final schedule are known.

To reach this goal, these final requirements are transformed to other requirements which must be fulfilled for each step of the allocation process. It may simply be assumed:

A solution to a given problem exists, and a set of activities is allocated in such a way that the solution still is existing. By allocation of an arbitrary activity from those still left, to a possible period for this, one of two things will happen: Either a solution still exists, or it does not.

This trivial conclusion is conceptually important. The set of conditions valid for the final schedule may partly be transformed to necessary conditions which the allocation in question must fulfil. This limits the possible solution space, and such conditions are called global conditions. A large set of necessary conditions implies large probability for making a correct allocation. There is,
however, a practical limit to how many global conditions it is useful
to define, and it may be better to define a set of local conditions
to maximize the probability of the existence of solution. Local
conditions are not necessary conditions for solution, and they are
defined from analogies with the global conditions, probability eva-
luations and heuristic principles.

The local conditions are combined with the qualitative requirements,
and in that way a local "optimization" is defined which tries to
maximize the probability of the existence of solution, at the same
time seeking that the final schedule shall fulfil as many as possi-
ble of the quality criteria.

Experience has shown that for most steps of the allocation the fol-
lowing is valid: Assuming that a solution exists in a given situ-
ation, it is only guaranteed that this solution still exists for a
small number (often only one) of the allocation possibilities in
question. For these possibilities the choice is indifferent regard-
ing absolute conditions, and local conditions are used to distingui-
sh between these. (A typical example of this is the case where
the blockings resulting from one possibility are a subset of the
blockings from another possibility.) Originally it was assumed that
this always was the case, but this proved to be wrong. For tight
school structures there may occur a number of interactions among the
conflict types which makes the above invalid, thus stressing the
importance of avoiding unnecessary limitations of the solution space
because of local optimization; it has proved particularly important
that qualitative criteria do not play a too dominating role (for
slack structures the qualitative criteria may be given more prio-
ритy).

The final schedule is represented by a logical matrix FFT, the allo-
cation matrix. A new step (or stage) in the allocation is charac-
terized by a modification of FFT. The time frame consists of TMAX
time units and the system consists of N activities. The dimension
of FFT is \([TMAX,N]\):

(3.9) \[ FFT_{ij} := \begin{cases} 
1, & \text{if activity } j \text{ is allocated to time unit } i \\
0, & \text{otherwise} \end{cases} \]
The situation for the various steps of the allocation will be represented by the freedom matrix (or freedom picture) LEDIG. It has the same dimension as FFT and defines which time units are still available for the various activities:

\[
(3.10) \quad \text{LEDIG}^g_{ij} := \begin{cases} 
1, & \text{if activity } j \text{ may be allocated to time unit } i \\
0, & \text{after } s \text{ steps in the allocation}
\end{cases}
\]

A section of the allocation and freedom matrices is shown in fig. 3.1.

FFT

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

Activity 12 is allocated to time unit 4.

TMAX

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

Activity 15 may be allocated to time unit 6 after s steps in the allocation process.

Fig. 3.1

FFT is constructed gradually, and FFT$^g$ represents the prehistory of LEDIG$^g$. The prehistory will be a result of the conditions to be fulfilled by the final FFT and the allocation strategy.

For each completed allocation a number of new conflicts arise. These will be represented by the matrix TAFT, the conflict picture:
\[
(3.11) \quad TAP^S_{ij} := \begin{cases} 
1, & \text{if activity } j \text{ is available for time unit } i \\
\text{after } s \text{ steps in the allocation process, but} \\
\text{is blocked after } s+1 \text{ steps} \\
0, & \text{otherwise} 
\end{cases}
\]

The allocation process may be described as follows:

\[
LEDIG^0 \rightarrow LEDIG^1 \rightarrow LEDIG^2 \rightarrow \ldots \rightarrow LEDIG^p \\
\text{LEDIG}^{s+1} := LEDIG^s + TAPT^S 
\]

where \( p \) is the number of periods to be allocated.

(3.12) shows two natural strategic principles for transformation of the requirements to the final schedule:

1. For each step in the allocation the conflict picture \( TAPT^S \) will be generated as completely as possible.

2. For each possible allocation the resulting \( LEDIG^{s+1} \) should satisfy as complete as possible a set of necessary conditions for the existence of a solution.

In practice, these principles must be modified on the basis of the following observations:

a) Not all necessary conditions can be transferred to \( TAPT^S \).

b) The complexity of the attached algorithms; i.e., necessary computing time and storage space.

c) How frequently a condition has consequences for \( TAPT^S \).

The above considerations apply to allocation of an arbitrary activity, and by allocating the activity which has the smallest number of possibilities, a considerable reduction in computing time may be obtained. However, due to the fact that the conflict pictures are not always complete, a number of activities might have illusory allocation possibilities. The activity which has the smallest number of real possibilities is called the critical activity, and has a minimum degree of freedom. One chooses to allocate the most critical activity at any time, and this has important consequences not only for computing time:

1. A number of necessary conditions will be implicitly fulfilled when at any time the most critical activity is allocated. This is motivated by the fact that possible unfortunate consequences

\[\text{The various operators are defined in the following chapter.}\]
for other activities may be more easily accepted because these have several allocation possibilities, and that a critical situation may be avoided later in the allocation.

2. If no solution exists this will be discovered early. This creates good perspective, and renders it easy to find the reason for a self-contradiction. (It also has important consequences for total computing time. Experience has shown that all self-contradictions usually are discovered before 10% of the activities are allocated.)

The minimum degree of freedom is a relatively complicated function of the conditions which are defined for the system. The principles chosen for the order in which the activities are to be allocated (or the allocation sequence) is of prime importance. To use a metaphor: it may be a question of finding a needle in a haystack or finding a way for the needle out of the haystack. The analogy with the scheduling problem is good, because if one starts at the right end of the problem it has a tendency to dissolve by itself. A new strategic principle therefore includes the rules chosen for the allocation sequence.

It is in no way obvious that the importance of the various conditions is the same throughout the allocation. On the contrary, it is simple to show that this is not the case, and the "weight function" for the conditions must be modified. In practice one will also be forced to use an insufficient set of conditions, and it may be necessary to redefine the set of conditions which is most suitable as the allocation progresses; i.e., "new" conditions will be generated. Several factors necessitate this:

1. An approximate method may be improved by completing the existing conditions with new ones.

2. On account of prohibitive computing time requirements, the consequences of certain conditions are only investigated for the conflict picture of the actual allocation. It is assumed that this limitation of solution space still provides for the existence of a solution.

3. 1) and 2) may lead to new rules for the allocation sequence, and forced allocations may occur; i.e., typical chain reactions
where allocation of one activity forces allocation of several other activities.

The strategy must take such factors into account, and these are named by the collective term: Generating of new conditions.

Strategy is here defined as the actual possibilities for transforming the requirements to the final goal (the final schedule) to operative goals; i.e., goals which can be reached from the existing situation. The basic principles for this have been mentioned, and (3.13) gives a survey:

<table>
<thead>
<tr>
<th>Known information</th>
<th>Transformation possibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Activity definitions for the system.</td>
<td>1. Generating of complete conflict pictures for each allocation.</td>
</tr>
<tr>
<td>2. Requirements to be fulfilled by the final schedule.</td>
<td>2. Necessary global conditions to be fulfilled by the resulting freedom picture.</td>
</tr>
<tr>
<td>(3.13)</td>
<td>3. Analogous local criteria maximizing the probability for solution, combined with an evaluation of qualitative criteria.</td>
</tr>
<tr>
<td></td>
<td>4. Rules for the allocation sequence.</td>
</tr>
<tr>
<td></td>
<td>5. Generating of new requirements.</td>
</tr>
</tbody>
</table>

The frame of this work is given in (3.8) and (3.13).

Characteristic properties of the suggested methodology will be discussed. (3.12) gives

(3.14) \[ \text{TAPTS} \in \text{LEDIGS} \]

\text{LEDIGS} is a result of the prehistory \text{FFT}S, and it may be stated:

(3.15) \[ \text{TAPTS} = f(\text{FFT}S) \]

i.e., the current conflict picture is a result of the prehistory. This is a reformulation of possibly the most important property of the scheduling problem:
No activity can get a worse relation to another activity than to be in conflict; i.e., it does not matter whether the reason why two activities cannot be allocated to the same time is due to one or several factors.

The course of events is anticipated, but the form of the conflict pictures will be suggested. This is most easily done by plotting $y_1 = \Sigma EDIQ_{i}$ and $y_2 = \Sigma TAPT_{i}$ as functions of the time unit.

(The expressions indicate the sum of 1-components in a row vector.) This is outlined in fig. 3.2, where allocation is done to time unit $t_i$.

![Graph](image)

Fig. 3.2

$y_2$ shows an important property of the conflict picture: There occur conflicts for other time units than those where an allocation is done, although most of the new conflicts arise there. (It is exceptional that the conflict pictures get a simpler form.)

A natural main principle for the allocation is that at any time, one tries to combine as many reasons for conflicts as possible; in that way trying to save as much freedom as possible for the rest of the activities; i.e., the activities are allocated in such a way that:
\[
T_{\text{MAX}} \sum_{i=1}^{N} \sum_{j=1}^{T_{\text{APT}}} = \text{MIN}
\]

for the chosen allocation.

Provided that global conditions limit the solution space and with certain considerable modifications, this is of course a correct principle. (It was originally assumed that local optimization conditions could be formulated analogous to (3.16). Practical experience proved that this was not the case, and it is easy to show that (3.16) is wrong.)

The requirements for FFT may result in a large number of conditions for LEDIG. The relation between the conditions may cause a number of interactions reducing the resulting freedom picture. A conflict which is due to one condition is called a first order conflict, interaction between two conditions is called a second order conflict, etc. One is interested in finding the conditions and combinations of conditions which are most likely to modify the conflict picture. (If this probability is small, consideration for computing time might lead to neglect of the possible consequences for certain combinations.)

Certain possible modifications of the conflict picture have little influence on the existence of a solution, and can be neglected. (In practice this means that for simple activities a few fictitious allocation possibilities may be allowed.)

Generation of the conflict picture is in a way analogous to expansion in series for approximation of ordinary functions: The conflict picture is generated from the set of conditions which most likely will change it. By conventional expansion so many terms are included that the deviation from the exact answer may be tolerated. For generation of the conflict picture no quantitative error margin is used; instead more qualitative measures are used for the frequency of how often a "term" (a combination of conditions) modifies the conflict picture and how important possible modifications would be. Most of the new conflicts are of a low order; i.e., the interaction between many conditions is unlikely to create new conflicts.
A schematic block diagram for the allocation process is presented in fig. 3.3.

Start

The rules for the allocation sequence find the critical activity

If all activities are allocated, the process is terminated

The conflict pictures for the various possibilities are generated and the solution space is limited due to global conditions

If there is no possible allocation, the schedule must be modified manually later

From local conditions and quality criteria one decides to which period the activity is to be allocated

LEDIG_{S+1}^{S} = LEDIG_{S}^{S} + TAPT_{S}

New conditions are generated, the conflict picture due to chosen allocation is expanded and the rules for the allocation sequence are modified

Principle outline for time allocation

Figure 3.3
It is simple to point out objections to this method, for example:

1. The prehistory, $FT^S$ does not have to be consistent with the new requirements which are being generated.

2. If one does not generate complete conflict pictures, one risks that some activities get fictitious allocation possibilities with the result that one does not find the most critical activity.

All similar objections may be reduced to stating that the method does not evaluate the future conflict possibilities sufficiently. This may be blunted by mentioning the practical results which have been achieved.

It is likely that due to economic competitive ability it is not suitable to have a standard program which covers every school structure; but any program may be based on a general strategy to which new requirements may be added by simple means.
4. DATA STRUCTURE AND OPERATORS

Summary

The operators to be defined will be useful in a much wider context than just the scheduling problem. Ordinary auto-code languages such as ALGOL and FORTRAN have an inconvenient representation of logical variables. Operators are defined only for scalar logical variables. The problem in question absolutely requires a suitable representation of logical vectors and matrices, and operators associated with these. To conserve storage space it is also necessary to represent arithmetic variables in partial words. (To simplify matters all scalar values will be represented in the same way as the auto-code language in question.)

The most important consequences of the data structure and operators are:

1. The chosen data structure is only vaguely based on the conventional concept of word length. This permits compressed storage of information. For logical matrix-and vector-operations are in addition achieved a reduction of computing time with factor approximately equal to the word length.

2. The ordinary logical operators are generalized and expanded. There are two classes of operators:
   a) Operators of class 1 are used to modify and transform logical matrices and vectors.
   b) Operators of class 2 are used to make logical assertions about vectors and matrices.

3. To be able to refer to a subset of a larger data structure, a selection operator has been defined. The definition implies that operations with a selection as operand are executed selectively. A selective operation is realised in practice by the operation being done in a loop where a selection vector guides the controlled variable. These loop structures are generated by a NEXT-operation, a generalization of conventional loop mechanisms.

4. The definitions of the various operations are far from being an attempt at defining a new formalism; the definitions have been chosen for practical purposes only, and should be regarded as a
set of rules with the following functions:

a) Simplify the definition of the various conditions, operations, etc. necessary for the problem area.

b) The various operations are to use minimal computing time.

Because of the chosen data structure most of the definitions thus refer to the rows of a matrix and not to the columns.

4.1 Notation for the data structure

The main part of the data structure will be logical variables (matrices, vectors, or scalar figures). Arithmetic variables will also be necessary. Different notations for the two kinds of variables will not be used. The variable type will appear either from the context or definitions. The following notation is used:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td>Capital Roman letter (or a combination of capital letters and numbers). E.g. A, B, TAPT, FFT.</td>
</tr>
<tr>
<td>Vector</td>
<td>Small Roman letter (or a combination of small Roman letters and numbers). E.g. a, b, c, q1, w9.</td>
</tr>
<tr>
<td>Row vector in matrix</td>
<td>The name of the matrix in capital letters and the row number as index. E.g. Aᵢ, Bᵢ, Cᵢ, FFTᵢ.</td>
</tr>
<tr>
<td>Column vector in matrix</td>
<td>The name of the matrix in capital letters and the column number as index; at the same time, the vector gets the letter T as superscript. E.g. Aᵢ, Bᵢ.</td>
</tr>
<tr>
<td>Scalar</td>
<td>Small Greek letter. E.g. α, δ, σ.</td>
</tr>
<tr>
<td>Component in matrix</td>
<td>The name of the matrix in capital letters, row number as first index and column number as second. E.g. Aᵢₗ.</td>
</tr>
<tr>
<td>Component in vector</td>
<td>The name of the vector in small letters and the component number as index. E.g. aᵢ, bᵢ, cᵢ.</td>
</tr>
</tbody>
</table>

To distinguish between various states or parts of a data structure is used a superscript analog to the definition of a column vector, e.g.:

1. LEDIGᵢ: may describe LEDIG after allocation of s activities.
2. LEDIG$^d$ : may describe that part of LEDIG where all time units belong to day $d$.

3. LEDIG$^{T,j,s,d}$ : may describe column vector $j$ of LEDIG after $s$ assignments, and only the time units belonging to day $d$ are of interest (a selection).

What the various superscripts symbolize should otherwise appear from the context. A superscript may often be so obvious that it is omitted in the text (the superscript is implicit).

The aforementioned notation is a main rule. Due to historical reasons there may be exceptions (e.g. scalar figures are written in Roman letters and vectors in capital letters). Such exceptions should appear from the context. Later names for various data structures will be introduced. These names might possibly seem deceptive, meaningless, or vague. The reason for this is a desire to use symbols corresponding to the existing program system. Furthermore, it may be difficult to find suitable names for very special data areas. However, the important thing is not what a data area is named, but that the name is used consistently. This work is an important part of the program documentation itself, and a double set of symbols may be confusing. Appendix 1 offers a short survey of the information in the various data areas.

4.2 Definition of the data structure

The physical representation of data influences the design of the algorithms. The requirements of the internal data representation may shortly be formulated as follows:

1. The data structure must be very compressed on account of necessary storage space.

2. One should be able to define fast and lucid algorithms.

3. An unambiguous representation of data should be secured.

4. The data structure must be flexible to facilitate inclusion of all relevant information and ease definitions of new requirements.

5. If possible, conventional ways of data representation should be used so that algorithms may be formulated directly in an auto-code language.
A medium scheduling problem requires direct access to between \( \frac{1}{2} \) and 1 million variables, and this is too much for most computers. The time usage of an algorithm is usually in a sense, inversely proportional in relation to the storage space used, but it will be shown that the requirement for small storage space and fast algorithms may be combined.

Most computers use fixed word length (with possible use of half words, double accuracy, etc.). Any scalar variable is represented as in an auto-code language, whereas for matrices and vectors the conventional word length concept does not apply. A few definitions:

**Bit:** The smallest storage space necessary to distinguish between the values 0 and 1, alternatively the statements FALSE and TRUE. (0 and FALSE, 1 and TRUE are used as synonymous concepts for logical variables.)

**Byte:** A set of bits. The number of bits in a byte is its dimension. To distinguish between the various bytes is written byte \((q)\) for a set of \(q\) bits. Note that byte \((1)\) is one bit.

**Register:** A set of \(n\) bytes where \(n\) is an arbitrary number. The dimension of a register is defined as follows: Register \((n,q)\) indicates \(n,q\)-bytes. Note that register \((n,1)\) is a logical vector, register \((1,q)\) a \(q\)-bits word, and register \((n,q)\) an ordinary one-dimensional array whose greatest component is less than \(2^{q} - 1\).

**Area:** A set of \(m\) registers where \(m\) is an arbitrary number. Area \((m,n,q)\) represents a set of \(m,n\)-registers in their turn constructed of \(q\)-bytes. Area \((m,n,1)\) represents a logical matrix of dimension \((m,n)\). Area \((m,1,q)\) is a one-dimensional array with word length \(q\). Area \((m,n,q)\) is an ordinary two-dimensional integer array.

In practice the figure \(q\) is not chosen randomly due to the actual word length of the computer and the instruction repertoire available for dealing with partial words. Univac 1107 has a 36-bit word length, and typical values for \(q\) would be: 1, 6, 12, 18, 36, 72. (For the last two values ordinary arrays might have been used as
well.) Other values of q have been used, even variable q for the same area, but this has not proved very useful.

Assume that the word length is w, and necessary storage space for an area \((m,n,q)\) may be estimated:

\[ 1: \quad q = 1 \]

In this case a register will be represented physically by an integer multiple of w, i.e.:

Necessary storage space in words = \(m \cdot ((n-1)/w+1)\)

\[ 2: \quad w/q = \text{Integer number} \quad \text{(and q=1)} \]

It is assumed that an instruction repertoire is defined for byte (q), and the various components will follow consecutively physically, i.e.:

Necessary storage space in words = \((m \cdot n -1)/(w/q)+1\)

In practice this means that necessary storage space is approximately reduced with a factor \(w/q\), i.e., for a logical matrix with a factor w.

In principle the representation will be as shown in fig. 4.1.

**Fig. 4.1**

Physical storage space for logical matrix \([m, n]\).

Physical storage space for integer matrix \([m, n]\).
The data representation is in itself well known, but is seldom used, since there are few problems which make it worth while to compress data in this way. In principle, the outlined data structure may be used for any kind of data. Logical vectors, matrices, and integer matrices (with the highest possible value definitely known) are dealt with here. For integer variables the representation primarily means considerable storage savings, whereas for logical variables an additional reduction of computing time with a factor \( w \) (sometimes considerably more) is achieved. One may safely assert that without the chosen data representation one would still have been very far from an economically justifiable solution of realistic scheduling problems.

Naturally, the data structure also has some disadvantages, e.g. there are very limited possibilities for dynamic allocation of data, and if one should want to redefine the dimensions of various areas, this usually necessitates recompiling of certain program elements. However, static allocation of data also has great advantages.

To be able to use the mentioned data structure one must make some declarations analogous to variables in Algol. The method used for this may be partially computer dependent. This will not be discussed in detail, but the main principles will be shown.

The information needed about an area is:

1. The dimension of \( m, n, q \).
2. To which storage the area is allocated.

Assume that one wants to define the following areas:

- **A** \((n,n,1)\)
- **B** \((\text{TMAX}, n, 1)\)
- **C** \((n, \text{UD}, 6)\)
- **D** \((\text{KM}, 4, 12)\)

20 registers with dimension \((n, 1)\)

It is given:

\[
\begin{align*}
  n &= 300 \\
  \text{TMAX} &= 48 \\
  \text{UD} &= 6 \\
  \text{KM} &= 200
\end{align*}
\]
One wants to store A in secondary storage (drum). In Univac 1107 the primary storage is divided in two banks, and one wants to store B in bank 1 and the rest of the data in bank 2. In principle, the declaration for this will be as in fig. 4.2.

**CONSTANTS**

<table>
<thead>
<tr>
<th></th>
<th>EQU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td>300</td>
</tr>
<tr>
<td>TMAX</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>UD</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>KM</td>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>

**MAIN AREA**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DRUM</td>
<td>N</td>
<td>N</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>BNK1</td>
<td>TMAX</td>
<td>N</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>BNK2,6</td>
<td>N</td>
<td>UD</td>
<td>.</td>
<td>2</td>
</tr>
<tr>
<td>BNK2,12</td>
<td>KM</td>
<td></td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>DO 20, BNK2</td>
<td>1</td>
<td>N</td>
<td>.</td>
<td>4-23</td>
</tr>
</tbody>
</table>

Fig. 4.2

The first element defines necessary constants, and the next where the areas are to be allocated and their dimensions. Dependent on the computer configuration other factors may be defined, e.g. common buffer for large data areas in primary storage, allocation of the same data area to primary as well as to secondary storage, etc. Such technical details will not be discussed here.

The program system assumes that a data structure analogous to fig. 4.2 is defined, and reference to the various areas is done by constants given by the sequence of the MAIN AREA components. (The need for much storage space also leads to a compressed "packing" of conventional data areas.)

**4.3 Operators for logical variables**

The ordinary logical operations for scalar values are unambiguously defined by the operation symbol. They may be generalized to operations for vectors and matrices, and an attached procedure set may be defined. The ordinary way to define a procedure name is the operator name plus a prefix and/or suffix defining for which data type the operation will be done (i.e., it has proved useful that the same
operation on a matrix row or a vector uses two different procedure calls; for example due to different numbers of formal parameters. Some operations will also depend upon a logical unit value (e.g. a selection operator will depend upon whether the following operations shall be done for the logical value 0 or 1 in the selection vector.) This may also be defined by a suitable prefix or suffix to the procedure call.

Table 4.1 outlines necessary operators. Operator names and notation symbols for the basic operations are listed to the left. Some of these are frequently used in special contexts, which makes it suitable to use special operator names (to clarify the program).

Table 4.2 presents the prefixes and/or suffixes which will appear. (not all of these symbols are equally meaningful, but symbols used in the program system are also used here, thus "meaningless" symbols are historically founded.)

Table 4.1 Logical operators

<table>
<thead>
<tr>
<th>Operator name</th>
<th>Notation</th>
<th>Special operator names</th>
<th>Verbal definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>∧</td>
<td></td>
<td>Logical multiplication</td>
</tr>
<tr>
<td>COIN</td>
<td>⊕</td>
<td></td>
<td>Coincidence</td>
</tr>
<tr>
<td>OR</td>
<td>∨</td>
<td></td>
<td>Logical addition</td>
</tr>
<tr>
<td>XOR</td>
<td>⊕</td>
<td></td>
<td>Logical exclusion</td>
</tr>
<tr>
<td>SUB</td>
<td>+</td>
<td></td>
<td>Logical subtraction</td>
</tr>
<tr>
<td>NEG</td>
<td>¬</td>
<td></td>
<td>Negation</td>
</tr>
<tr>
<td>EQL</td>
<td>≡</td>
<td></td>
<td>Equivalence of two logical variables</td>
</tr>
<tr>
<td>BIT</td>
<td></td>
<td></td>
<td>Equivalence of a logical scalar and a given logical value (0 or 1)</td>
</tr>
<tr>
<td>IMPL</td>
<td>⊃</td>
<td></td>
<td>Logical implication</td>
</tr>
<tr>
<td>EQUI</td>
<td>:=</td>
<td></td>
<td>Dynamic equal sign</td>
</tr>
<tr>
<td>TRF</td>
<td></td>
<td></td>
<td>A vector (matrix) is assigned to another vector (matrix)</td>
</tr>
<tr>
<td>LOAD</td>
<td></td>
<td></td>
<td>A row (column) of a matrix is assigned to a vector. The dimension of the vector is made equal to the dimension of the row (column).</td>
</tr>
</tbody>
</table>
Table 4.1 continued

<table>
<thead>
<tr>
<th>Operator name</th>
<th>Notation</th>
<th>Special operator names</th>
<th>Verbal definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQU</td>
<td>STORE</td>
<td>(:=1)</td>
<td>A vector is assigned to a row (column) of a matrix.</td>
</tr>
<tr>
<td></td>
<td>SET</td>
<td>(:=0)</td>
<td>All elements of a more closely defined part of a logical variable are given the value 1</td>
</tr>
<tr>
<td></td>
<td>CLR</td>
<td></td>
<td>Like SET, but the elements are given the value 0</td>
</tr>
<tr>
<td>SEL</td>
<td>s</td>
<td></td>
<td>Selection, i.e. a given operation is only done for a defined set of values</td>
</tr>
<tr>
<td></td>
<td>NEXT</td>
<td></td>
<td>From a given position of a vector (or row/column of a matrix) is found the first component with a defined logical unit value. The result of the operation will be an integer number, and an operation will be done for this value.</td>
</tr>
<tr>
<td>SUM</td>
<td>Σ</td>
<td></td>
<td>The arithmetic sum of all components with the same value as a given logical unit value</td>
</tr>
<tr>
<td>LSUM</td>
<td>Σ</td>
<td></td>
<td>The logical sum of a number of variables</td>
</tr>
<tr>
<td>LPRO</td>
<td>Π</td>
<td></td>
<td>The logical product of a number of variables</td>
</tr>
</tbody>
</table>

Table 4.2 Prefix and/or suffix

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Characteristic parameter is 1</td>
</tr>
<tr>
<td>N</td>
<td>Characteristic parameter is 0</td>
</tr>
<tr>
<td>R</td>
<td>The operation is done on a matrix row. This suffix is often used in connection with B and N.</td>
</tr>
<tr>
<td>1</td>
<td>The operation uses one vector as parameter</td>
</tr>
<tr>
<td>2</td>
<td>The operation uses two vectors as parameters</td>
</tr>
<tr>
<td>D</td>
<td>The first parameter is stored on drum</td>
</tr>
<tr>
<td>H</td>
<td>The first parameter is a matrix row</td>
</tr>
<tr>
<td>V</td>
<td>The first parameter is a matrix column</td>
</tr>
<tr>
<td>REG</td>
<td>The first and second parameters to an operation are vectors. In those cases where the operation has three parameters, the third will also be a vector.</td>
</tr>
</tbody>
</table>
Table 4.2 continued

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Suffix</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRE</td>
<td></td>
<td>The first parameter is a matrix row, the second (and possibly third) a vector.</td>
</tr>
<tr>
<td>RRT</td>
<td></td>
<td>The first parameter is a matrix row, the second a vector, and the result of the operation is assigned to the first parameter.</td>
</tr>
<tr>
<td>TRL</td>
<td></td>
<td>The operation is done on a matrix column</td>
</tr>
</tbody>
</table>

Logical operations for scalar figures are defined by a truth table. For those operators in table 4.1 where this is meaningful, a truth table is presented in table 4.3.

Table 4.3 Truth table for scalar logical operations

<table>
<thead>
<tr>
<th></th>
<th>NEG</th>
<th>AND</th>
<th>OR</th>
<th>XOR</th>
<th>SUB</th>
<th>COIN</th>
<th>EQL</th>
<th>IMPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>β</td>
<td>ᾱ</td>
<td>αβ</td>
<td>α∨β</td>
<td>α↔β</td>
<td>α⊙β</td>
<td>α∈β</td>
<td>α∉β</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The definitions used in table 4.3 are well known but for a few exceptions. A few comments:

1. Operation NEG is a unary operator, the others are binary.

2. Operations AND and COIN are identical for scalar operations, i.e. COIN is apparently superfluous, but these operations will get different definition for vectors and matrices.

3. Note: $α + β = α + β$

4. For all operations, except SUB and IMPL, the commutative law is valid, i.e.

   $aAβ ≡ βAα$

   whereas:

   $(a+β)α = (β+a)α$

   $(a·β)α = (β·a)α$
The operators in table 4.3 will be generalized to include matrices and vectors. The operators are divided into two classes:
Class 1: NEG, AND, OR, XOR, SUB
Class 2: COIN, EQL, IMPL
OP now symbols a class 1 operation, and the following rules have been chosen:

a. Vectors
(4.1) \[ a \; OP \; b = \{ a_1 \; OP \; b_1, \, a_2 \; OP \; b_2, \, \ldots, \; a_n \; OP \; b_n \} \]

b. Matrix and vector
(4.2) \[ (A)\; OP \; b = \begin{bmatrix} \begin{bmatrix} A_{11} \; OP \; b_1, \, A_{12} \; OP \; b_2, \, \ldots, \, A_{1n} \; OP \; b_n \end{bmatrix} \\ \vdots \\ \begin{bmatrix} A_{m1} \; OP \; b_1, \, A_{m2} \; OP \; b_2, \, \ldots, \, A_{mn} \; OP \; b_n \end{bmatrix} \end{bmatrix} \]

b has the same dimension as a row in A, and the result of the operation is a matrix with the same dimension as A.

c. Matrices
The operations of class 1 are seldom used as independent operations for matrices, but when this happens, two kinds of rules are necessary:

Rule No. 1:
(4.3) \[ A \; OP \; B = \begin{bmatrix} A_{11} \; OP \; B_{11}, \, A_{12} \; OP \; B_{12}, \, \ldots, \, A_{1n} \; OP \; B_{1n} \\ A_{21} \; OP \; B_{21} \; \ldots \\ \vdots \\ A_{m1} \; OP \; B_{m1}, \, A_{m2} \; OP \; B_{m2}, \, \ldots, \, A_{mn} \; OP \; B_{mn} \end{bmatrix} \]

Rule No. 2:
(4.4) \[ A|\; OP|\; B = \begin{bmatrix} A_{11} \; OP \; B_1, \, A_{12} \; OP \; B_2, \, \ldots, \, A_{1q} \; OP \; B_q \\ A_{21} \; OP \; B_1 \; \ldots \\ \vdots \\ A_{p1} \; OP \; B_1, \, A_{p2} \; OP \; B_2, \, \ldots, \, A_{pq} \; OP \; B_q \end{bmatrix} \]
To distinguish between the two rules, $||$ is used around the operation symbol for rule 2, whereas the operation symbol is used alone for rule 1.

Rule 1 assumes that dimension of row and column is equal for the two operands, and the result is a two-dimensional matrix with the same dimension.

Rule 2 assumes that the row dimension is equal for the two operands, and the result is a three-dimensional matrix. If the dimension of $A$ is $[p,n]$ and $B$ is $[q,n]$, the final dimension will be $[p,q,n]$.

The operations on the individual elements follow table 4.1 (although operation NEG is unary it may be said to belong to class 1). Figure 4.3 gives an example for two arbitrary vectors $a$ and $b$:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>1100100011</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>1010111101</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0011011100</td>
<td></td>
</tr>
<tr>
<td>$a\land b$</td>
<td>1000100001</td>
<td></td>
</tr>
<tr>
<td>$a\lor b$</td>
<td>1110111111</td>
<td></td>
</tr>
<tr>
<td>$a\oplus b$</td>
<td>0110011110</td>
<td></td>
</tr>
<tr>
<td>$a+b$</td>
<td>0100000010</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4.3

From figure 4.3 it should appear clearly how the rules are expanded to matrices. Note that logical matrix multiplication $A\land B$ or $A\lor B$ has nothing in common with the corresponding arithmetical operation.

For operators of class 2 different rules are defined. OP now symbols such an operator and OP$_1$ one of the operators LSUM, LPRO. The rules for vectors and matrices will be:

$$A \text{ OP } B = \text{ OP}_1(A_{ij}, \text{ OP } B_{ij})$$

This will be shown explicitly for each of the operators.
COIN

a. Vectors

\[(4.4)\quad a\Lambda b = (a_1 A_1) V (a_2 A_2) V \cdots (a_n A_n)\]

\[= \bigvee_{i=1}^{n} (a_i A_i) \bigvee_{i=1}^{n} (a_i A_i)\]

The result of this operation is a scalar logical figure. (The operator LSUM is defined simultaneously.)

b. Matrix and vector

\[(4.5)\quad A\Lambda b = \left\{ \begin{array}{c}
\bigvee_{i=1}^{n} (A_{1i} A_{i1}), \quad \bigvee_{i=1}^{n} (A_{2i} A_{i1}), \quad \cdots \quad \bigvee_{i=1}^{n} (A_{ni} A_{i1})
\end{array} \right\}\]

The result of this operation is a vector.

c. Matrices

\[
(4.6)\quad A \Lambda A = \left\{ \begin{array}{c}
\bigvee (A_{11} A_{11}), \quad \bigvee (A_{21} A_{11}), \quad \cdots \quad \bigvee (A_{p1} A_{11})
\end{array} \right\}
\]

The result of this operation is a matrix, and it has been assumed that the dimension of A is \([p,n]\) and of B \([q,n]\). This operation is very much like an ordinary matrix multiplication. Each element is then the scalar product of a row vector and a column vector, whereas the elements for \(A \Lambda B\) are the scalar product of two row vectors. The reason for this is the way in which the data structure is defined; if at all possible, definitions demanding reference to a column vector should be avoided to reduce computing time.

Figure 4.4 is an example of the operation \(A \Lambda B\).

\[
\begin{array}{c|c}
A & B \\
\hline
\begin{bmatrix}
101110 \\
110000 \\
000000 \\
111100
\end{bmatrix} & \begin{bmatrix}
111111 \\
000011 \\
001100 \\
110110
\end{bmatrix} \\
\end{array} = \begin{bmatrix}
1101 \\
1000 \\
1001
\end{bmatrix}
\]

Figure 4.4
EQL

a. Vectors

\[(4.7)\]  
\[a \equiv b = (a_1 \equiv b_1) \land (a_2 \equiv b_2) \land \cdots \land (a_n \equiv b_n)\]

\[= \bigwedge_{i=1}^{n} (a_i \equiv b_i)\]

b. Matrices

\[(4.8)\]  
\[A \equiv B = (A_{11} \equiv B_{11}) \land (A_{12} \equiv B_{12}) \land \cdots \land (A_{1n} \equiv B_{1n})\]

\[\land (A_{21} \equiv B_{21}) \land (A_{22} \equiv B_{22}) \land \cdots \land (A_{mn} \equiv B_{mn})\]

\[= \bigwedge_{j=1}^{m} \bigwedge_{i=1}^{n} (A_{ji} \equiv B_{ji})\]

IMPL

a. Vectors

\[(4.9)\]  
\[a \in b = (a_1 \in b_1) \land (a_2 \in b_2) \land \cdots \land (a_n \in b_n)\]

\[= \bigwedge_{i=1}^{n} (a_i \in b_i)\]

b. Matrices

\[(4.10)\]  
\[A \in B = (A_{11} \in B_{11}) \land (A_{12} \in B_{12}) \land \cdots \land (A_{1n} \in B_{1n})\]

\[\land (A_{21} \in B_{21}) \land (A_{22} \in B_{22}) \land \cdots \land (A_{mn} \in B_{mn})\]

\[= \bigwedge_{j=1}^{m} \bigwedge_{i=1}^{n} (A_{ji} \in B_{ji})\]

EQL and IMPL are defined in the same way. The result of these operations is always a scalar logical figure, and the two operands must always have the same dimension. If \(A \in B\), one says that \(A\) is included in \(B\) or that \(A\) is a subset of \(B\). COIN is defined somewhat
differently from EQL and IMPL. These could of course have been defined in the same way, but the above definitions are most suitable for the actual problem area.

When only using scalar logical values, logical operators are usually used to decide whether a statement is TRUE or FALSE, and arithmetic operators to calculate arithmetic expressions. When introducing logical vectors and matrices the situation changes. One may say that the operators of class 1 replace the arithmetic operators, so that calculations with logical matrices and vectors can be done. The operators of class 2, however, are used to make logical assertions about matrices and vectors (COIN has a kind of intermediate position, but is mainly used in connection with logical assertions). The above is a somewhat simplified presentation, but it gives a certain impression of the use of the operators.

**EQU (:=)**

The dynamic equal sign is well known from other literature and should be self-explanatory.

**SUM, LSUM, LPRO**

These are not independent operators, but abbreviations for a set of basic operations. Definitions:

**SUM (Σ)**

\[ \sum_{i=1}^{n} a_i = a_1 + a_2 + a_3 \ldots + a_n \]

Note that + signifies the arithmetic sum of logical figures (TRUE=1, FALSE=0).

A frequent way of using the sum symbol is:

\[ \sum_{i \in b} a_i = a_1 \cdot b_1 + a_2 \cdot b_2 \ldots + a_n \cdot b_n \]

Where b is a logical vector, and a an arithmetic or logical vector.
If the summation limits appear from the context, one will substitute
\[ \sum_{i=1}^{n} a_i \quad \text{or} \quad \sum_{i \in b}^{\infty} a \] with \(|a|\).

If this operator is used for a two-dimensional or three-dimensional matrix, only \(\Sigma A\) is written, implying a summation over the last index. For a two-dimensional matrix \(A[n,m]\) the resultant will be a vector of dimension \(n\), and each element will be the arithmetic sum of the elements of a row. An example is the following:

\[ C = \Sigma(A|n|B) \]

\[ \text{DIM}[A] = (p,n) \]

\[ \text{DIM}[B] = (q,n) \]

The resultant \(C\) gets dimension \([p,q]\), and each element is the arithmetic sum of \(n\) elements.

**LSUM (\(\mathbf{V}\))**

(4.13) \[ \mathbf{V} a_i = a_1 V a_2 V a_3 \cdots V a_n \]

**LPRO (\(\mathbf{A}\))**

(4.14) \[ \mathbf{A} a_i = a_1 A a_2 A a_3 \cdots A a_n \]

The operations \(\mathbf{V} a_i\) and \(\mathbf{A} a_i\) are defined analogously to (4.12).

These operations are also defined for matrices:

(4.15) \[ \mathbf{V} A i \in b = (A_1 V d^1)(A_2 V d^2) \cdots V(A_m V d^m) \]

where \(d^i = \begin{cases} 1 & \text{vector if } b_i = 1 \\ 0 & \text{vector if } b_i = 0 \end{cases} \)

and \(m\) is dimension of \(b\) (equal to number of rows in \(A\)).

(4.15) \[ \mathbf{A} A i \in b = (A_1 V d^1)(A_2 V d^2) \cdots A(A_m V d^m) \]

where \(d^i = \begin{cases} 1 & \text{vector if } b_i = 0 \\ 0 & \text{vector if } b_i = 1 \end{cases} \)
The vector defined by (4.15) is later called the selection sum of \( A \) with regard to \( b \), and the vector defined by (4.16) is called the selection product of \( A \) with regard to \( b \).

**SEL**

Selection is an operation used to define a subset of a given data structure. This subset is called a selection of the complete data structure, the selection object. The data structure defining a selection is called the selector (or selection vector since this data structure usually is a logical vector). A number of problems require a selection operator, but formalization is lacking, and definitions suitable for the actual problem are chosen. A selection is defined as follows:

\[(4.17) \quad A := b \cap C \quad \text{or} \quad A := b \cap C \quad q, T\]

where:

- \( A \) = the selection
- \( C \) = the selection object
- \( b \) = the selection vector. It is always a logical vector.
- \( q \) = a logical unit value (0,1). If \( q \) has the value 1, it may be omitted:
  \[ b \cap C \equiv b \cap C \quad 1 \]

(4.17) is identical with:

\[(4.19) \quad A^1 := \begin{cases} C^1 & \text{if } b^1 = q \\ 0 & \text{if } b^1 = \bar{q} \end{cases}\]

(4.18) is identical with:

\[(4.20) \quad A^T := \begin{cases} C^T & \text{if } b_i = q \\ 0 & \text{if } b_i = \bar{q} \end{cases}\]

(4.17) states that \( A \) is a row selection of \( C \), and (4.18) that \( A \) is a column selection of \( C \). The criterion for a row (column) to be included in a selection is called the selection criterion.
In some cases a selection is defined in a form different from (4.17) and (4.18):

(4.21) \( A := \bigcap_{i \in k} q(i) \) or

\( A := \bigcap_{i \in k} \{ q(i) \}, T \)

where \( q(i) \) is a logical function of \( i \), so that \( C_i \) (or \( C_i^T \)) are included in the selection \( A \) if \( q(i) = 1 \).

Examples:

\[
A := \bigcap_{i = k} \{ i \in k \} \quad : \quad A_i := \begin{cases} C_i & \text{for } i = k \\ 0 & \text{for } i \neq k \end{cases}
\]

\[
A := \bigcap_{i = 1, 3, 5} \{ i \in 1, 3, 5 \} \quad : \quad A_i := \begin{cases} C_i & \text{for } i = 1, 3, 5 \\ 0 & \text{for } i \neq 1, 3, 5 \end{cases}
\]

\[
A := \bigcap_{\text{MOD}(i-1,k)+1=k} \{ \text{MOD}(i-1,k)+1=k \} \quad : \quad A_i := \begin{cases} C_i & \text{for } i = k, 2k, 3k \cdots \\ 0 & \text{for } i \neq k, 2k, 3k \cdots \end{cases}
\]

Selection \( A \) may also be a result of several selection objects, e.g.:

\[
A := (b \cup C) \cap (d \cup E) \quad : \quad A_i := \begin{cases} C_i \land E_i & \text{for } b_1 \land d_1 = 1 \\ 0 & \text{for } b_1 \land d_1 = 0 \end{cases}
\]

The general rule is:

\[
A := (b_1 \cup B_1) \cup P_1 (b_2 \cup B_2) \cup P_2 (b_3 \cup B_3) \cup P_3 \cdots \quad : \quad A_i := \begin{cases} C_i \land P_i & \text{for } b_1 \cup B_1 \cdots = 1 \\ 0 & \text{for } b_1 \cup B_1 \cdots = 0 \end{cases}
\]

In (4.23) \( P_1, P_2, P_3 \) etc. are operators of class 1, and it is assumed that the sequence of the operations is unambiguous (this may be done with the help of a parenthesis structure).

The definition of a selection implies more than just giving reference to a subset of a larger data structure. A selection also implies that all operations containing a selection will be done selectively, i.e., the operations will be done only for a subset of the possible values. This is shown in (4.23); one example:
(4.24) \[ A := F(bfC, D, E \quad \cdot \cdot \cdot ) \]

One may say:

\[
\begin{align*}
A_1 &: = bfA \\
A_2 &: = bfA
\end{align*}
\]

(4.25) \[ A = A_1VA_2 \]

\(A_1\) is the subset of \(A\) present in the selection \(bfC\), thus:

(4.26) \[ A_1 := bf(C, D, E \quad \cdot \cdot \cdot ) \]

\(A_1\) only necessitates the operations \(F(C, D, E \quad \cdot \cdot \cdot )\) for a set of the possible parameters. This has practical consequences for necessary computing time. An operation of the form (4.26) is called a selective operation.

Some examples of selective operations are:

1. Any logical multiplication may be considered a selective operation where one of the operands is a selector defining the values for which an operation on the second operand is done. (This example is rather artificial, but it shows a selective operation in its purest form.)

2. The above example may be expanded to become important in practice: Assume that the logical product of a set of vectors defines the values for which a more complex operation will be done. After doing the multiplication (finding the selector), the operation is done for a subset of the values given by the final vector (such a technique is often called masking and the selector a mask). A different organization of the operations may have consequences for the computing time.

3. The constructions \(<for>\ <step>\ <until>\ <do>\ <while>\ <do>\) are selectors for the operations defined inside the loop. The controlled variable defines the values for which the operations will be done. This point of view shows that a selector is used on an operation which itself is constructed of selective suboperations, by double loops, triple loops, etc.

This last example shows how reference to a selection is achieved and how an operation is done:
A selection or a selective process is defined by referring to the selection object inside a loop structure where the controlled variable is guided by the selection vector.

The NEXT-operator is used in practice for selective operations. Its importance to the scheduling problem necessitates a special mention of the operator:

\[ j := \text{NEXTB}(a, j, n, L) \quad (\text{where DIM}[a] = n) \]

should be read as follows:

\[
\text{for } j := j+1 \text{ while } a(j) \equiv 0 \land j \leq n \text{ do;}
\]

\[
\text{if } j > n \text{ then goto L;}
\]

i.e., the new \( j \) becomes the nearest index, greater than the current \( j \), where the element \( a_j \) has the value 1. If no index in \( a \) satisfies the requirement, then \( j := n+1 \), and one gets exit to label \( L \).

This shows that a NEXT-operation is often used to define a loop structure, e.g.:

\[
\text{for } j := 1 \text{ step 1 until } n \text{ do}
\]

\[
\text{if } a(j) = 1 \text{ then}
\]

\[
\begin{align*}
\text{begin} \\
\text{(code goes here)} \\
\text{end;}
\end{align*}
\]

An identical loop structure would be:

\[
j := 0;
\]

\[
L1: j := \text{NEXTB}(a, j, n, L2);
\]

\[
\text{begin}
\]

\[
\text{(code goes here)}
\]

\[
\text{go to L1;}
\]

\[
L2: \text{end;}
\]

The NEXT-operation is in a way more general than ordinary loop-structures, since it in a simple way allows a variable increment. The sequence of the values for the controlled variable are usually assumed to be a monotonously increasing (or decreasing) series of numbers. This also applies to a NEXT-loop, but it is so flexible that
this may easily be deviated from. The disadvantage of the NEXT-operation is that the selection vector requires storage space, and usually has to be generated before entering a loop. With the chosen data structure the additional storage space is insignificant, and complex selective operations necessitate using much time to generate (reduce) the selection vector. (One of the important principles in programming is that as much as possible of a process should be executed in the outer loops of a multi-looped structure. This is equivalent to minimize the selection vector for the inner loops.) The opinion of the author is that an operation analogous to a NEXT-operation ought to be implemented in an ordinary autocode language. It facilitates the understanding and documentation of complicated loop structures, and makes possible an estimate of necessary computing time for analogous algorithms.

Example:

\[(4.27) \quad d := (a/fR)_c \quad \text{for } a_i = 1\]

\[(4.28) \quad d_i := \begin{cases} \quad B_i A_c & \text{for } a_i = 1 \\ 0 & \text{for } a_i = 0 \end{cases}\]

is identical with:

\[(4.29) \quad d := \text{aA(BAc)}\]

\[(4.29)\] may be written:

\[
\text{for } i := 1 \text{ step } i \text{ until } n \text{ do } \\
\quad \text{REGAND}(a,d,d); \\
\quad \text{RCOIN}(B_i,i,c);
\]

\[(This \ is, \ however, \ very \ inconvenient.)\]

\[(4.27)\] may be written:

\[
\text{i} := 0; \text{REGCLR}(n,d); \\
\text{L1:} \text{if } \text{NEXTB}(a_i,n,L2) \\
\quad \text{d(i)} := \text{RCOIN}(B_i,i,c); \\
\quad \text{goto L1; }
\]

\[
\text{L2: }
\]

The procedures which are used are defined on the following pages.
Assume that the time for finding the controlled variable is approximately equal in both cases, and the time quotient will be:

\[
\frac{\text{Time}(4.29)}{\text{Time}(4.27)} = \frac{n}{\Sigma a_i} \geq 1
\]

and if \( \Sigma a_i << \Sigma a_i \) then \( \frac{n}{\Sigma a_i} \gg 1 \)

The example is trivial and the result evident. However, one important thing should be noted: Necessary computing time is approximately a function of the selection vector, and a main principle for using the NEXT-operation is that one should always try to maximise (or minimize) the proportion \( \Sigma a_i/\Sigma a_i \) for the selection vector. The example shows the significance of operations being selectively defined. This has primarily consequences for necessary computing time, and furthermore it will be difficult to transfer more complex selective operations to a form analogous to (4.29). The following is usually valid for selective operations: That part of the final result not "included" in the actual selection of a selection object, is independent of it, and the values of these elements may be directly stipulated.

Finally a simple example of the NEXT-operation when using double loops will be shown:

(4.30) \( A:=\Sigma(b/B|A|c/C) \)

\[
\begin{align*}
\text{DIM}[B] &= (n,p) \\
\text{DIM}[C] &= (m,p) \\
\text{DIM}[b] &= n \\
\text{DIM}[c] &= m \\
\text{DIM}[A] &= (n,m)
\end{align*}
\]

\[
A_{ij} := \begin{cases} 
\Sigma (b_i \land c_j) & \text{for } b_i \land c_j \neq 0 \\
0 & \text{for } b_i \land c_j = 0
\end{cases}
\]

An algorithm for (4.30) is:

**comment** q1 and q2 are auxiliary registers (vectors);

```
RRECLR(A);
i:=0;
L1:
i:=NEKTBB(b,i,n,L3);
LOADH(B,i,q1);
j:=0;
```
L2: \( j := \text{NEXTB}(c, j, m, L1); \)
\( \text{comment note exit to label L1;} \)
\( \text{RREAND}(C, j, q1, q2); \)
\( A[i, j] := \text{SUMB}(q2); \)
\( \text{goto L2}; \)

L3:

Observe that an operation with two selections as operands leads to a double loop structure. Should one want to transfer (4.30) to a form analogous to (4.29), the result will be:

\[(4.31) \quad A := \sum_{i \in D} (B[A]C) \]

where \( D[i,j] = b_{i,j} \cdot A_{i,j} \)

An algorithm for (4.31) where the form of \( D \) (the selector, in this case a matrix) is not primarily considered, is of course time consuming.

4.4 Procedure set for the data structure

Data structure and operators are defined. Based on this a procedure set has been designed. Only procedures in current use will be mentioned, but the set may of course easily be expanded.

It would have been simple to limit the number of procedure names by defining formal parameters such that they distinguish between the various contexts of analogous procedure usage. From a practical point of view one would expect the program development to be simpler and clearer. However, the opposite has proved to be the case: If the same procedure name is used in a number of various contexts, and the various cases are distinguished by complex definitions of the formal parameters, debugging is complicated due to lack of lucidity. If, however, different prefix and/or suffix are used for a procedure (operation) depending on the context where the operation will be done, a better syntax check is indirectly achieved, since the number of current parameters must correspond to actual prefix (suffix).

Furthermore, a disadvantage of the general procedures is that they easily lead to a large and unnecessary increase of computing time. (By development of compilers the opposite way is usually employed, i.e., one tries to gather I/O operations into a few large procedures.)
Provided that one has a sufficient syntax check this is excellent, but this is rather complicated for the current data structure.

In addition to previously mentioned logical operations, arithmetic operations and procedures for communication with external media (secondary storage, high speed printer, etc.) are needed. Such operations are well known, and the procedure definitions are assumed to be self-explanatory. The scheduling problem implies some particular operations without obvious general value, and these are defined in due course.

### Logical Operations

<table>
<thead>
<tr>
<th>Operator</th>
<th>Procedure call</th>
<th>Definition</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>REGAND(a,b,c)</td>
<td>c := a\land b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRLAND(A,i,b,c)</td>
<td>c := A_1 \land b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRTAND(A,i,b)</td>
<td>A_1 := A_1 \land b</td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td>REGOR(a,b,c)</td>
<td>c := a\lor b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RREOR(A,i,b,c)</td>
<td>c := A_1 \lor b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRTOR(A,i,b)</td>
<td>A_1 := A_1 \lor b</td>
<td></td>
</tr>
<tr>
<td>XOR</td>
<td>REGXOR(a,b,c)</td>
<td>c := a\oplus b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RREXOR(A,i,b,c)</td>
<td>c := A_1 \oplus b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRTXOR(A,i,b)</td>
<td>A_1 := A_1 \oplus b</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>REGSUB(a,b,c)</td>
<td>c := a+b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRESUB(A,i,b,c)</td>
<td>c := A_1 + b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRTSUB(A,i,b)</td>
<td>A_1 := A_1 + b</td>
<td></td>
</tr>
<tr>
<td>NEG</td>
<td>REGNEG(a,b)</td>
<td>b := \neg a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRENNEG(A,i,b)</td>
<td>b := \neg A_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRTNEG(A,i)</td>
<td>A_1 := \neg A_1</td>
<td></td>
</tr>
<tr>
<td>COIN</td>
<td>c := COIN1(a)</td>
<td>c := a \cdot a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c := COIN2(a,b)</td>
<td>c := a \cdot b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c := RCOIN(A,i,b)</td>
<td>c := A_1 \cdot b</td>
<td></td>
</tr>
</tbody>
</table>

*Note: Some operations use boolean functions (c is a scalar logical variable).*
<table>
<thead>
<tr>
<th>Operator</th>
<th>Procedure call</th>
<th>Definition</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQL</td>
<td>c:=RESEQL(a,b)</td>
<td>c:=a\equiv b</td>
<td>boolean functions (c is a scalar logical variable)</td>
</tr>
<tr>
<td></td>
<td>c:=RREQL(A,i,b)</td>
<td>c:=A_i \equiv b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=BITB(a,i)</td>
<td>c:=a_i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=BITN(a,i)</td>
<td>c:=\overline{a}_i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=BITBR(A,i,j)</td>
<td>c:=A_{ij}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=BITNR(A,i,j)</td>
<td>c:=\overline{A}_{ij}</td>
<td></td>
</tr>
<tr>
<td>IMPL</td>
<td>c:=REGIMP(a,b)</td>
<td>c:=a \oplus b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=RREIMP(A,i,b)</td>
<td>c:=A_i \oplus b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=IMPL1(a,b)</td>
<td>c:=(a \oplus b) V(a \equiv b)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c:=IMPL2(A,i,b)</td>
<td>c:=(A_i \oplus b) V(A_i \equiv b)</td>
<td></td>
</tr>
<tr>
<td>EQU</td>
<td>RERTRF(a,b)</td>
<td>b:=a</td>
<td>The difference between RRETRF and LOADH is that the first procedure assumes corresponding dimensions, whereas the latter redefines the dimension of b.</td>
</tr>
<tr>
<td></td>
<td>RRETRF(A,i,b)</td>
<td>b:=A_i</td>
<td>Slow procedure</td>
</tr>
<tr>
<td></td>
<td>LOADH(A,i,b)</td>
<td>b:=A_i \top</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOADV(A,i,b)</td>
<td>b:=A_i \top</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STOREH(A,i,b)</td>
<td>A_i := b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STOREV(A,i,b)</td>
<td>A_i := b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SETB(a,i)</td>
<td>a_i := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SETN(a,i)</td>
<td>a_i := 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SETBR(A,i,j)</td>
<td>A_{ij} := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SETNR(A,i,j)</td>
<td>A_{ij} := 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REGSET(n,a,b---)</td>
<td>a:=b::---:=1</td>
<td>The procedures have a random number of parameters. First param. redefines the dimension of the following vectors. All components of these get the value 1(or 0).</td>
</tr>
<tr>
<td></td>
<td>DIM[a]:=n</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIM[b]:=---:=n</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REGCLR(n,a,b---)</td>
<td>a:=b::---:=0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIM[a]:=n</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIM[b]:=---:=n</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRESET(A)</td>
<td>A:=1</td>
<td>All components of matrix A get the value 1 (or 0).</td>
</tr>
<tr>
<td></td>
<td>RRECLR(A)</td>
<td>A:=0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRLSET(A,i)</td>
<td>A_i := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRLCLR(A,i)</td>
<td>A_i := 0</td>
<td></td>
</tr>
<tr>
<td>Operator</td>
<td>Procedure Call</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>SEL</td>
<td>( p := NEXTB(a,i,n,L) )</td>
<td>( p \subseteq a \text{ and } i \leq n \text{ and } \sum_{k=i+1}^{n} a_k = 1 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( p := NEXTN(a,i,n,L) )</td>
<td>( p \subseteq a \text{ and } i \leq n \text{ and } \sum_{k=i+1}^{n} a_k = 1 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( p := NEXTBR(A,i,j,n,L) )</td>
<td>( p \subseteq A_i \text{ and } j \leq n \text{ and } \sum_{k=i+1}^{n} A_k = 1 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( p := NEXTNR(A,i,j,n,L) )</td>
<td>( p \subseteq A_i \text{ and } j \leq n \text{ and } \sum_{k=i+1}^{n} A_k = 1 )</td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td>( p := SUMB(a) )</td>
<td>( p := \Sigma a =</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>( p := SUMP(a) )</td>
<td>( p := \Sigma A =</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>( p := SUMBR(A,i) )</td>
<td>( p := \Sigma A_i =</td>
<td>A_i</td>
</tr>
<tr>
<td></td>
<td>( p := SUMNR(A,i) )</td>
<td>( p := \Sigma \overline{A_i} =</td>
<td>\overline{A_i}</td>
</tr>
<tr>
<td>LSUM</td>
<td>SLSUM(A,b,c)</td>
<td>( c := \bigwedge_{i \in B} A_i )</td>
<td></td>
</tr>
<tr>
<td>LPRO</td>
<td>SLPRO(A,b,c)</td>
<td>( c := \bigvee_{i \in B} A_i )</td>
<td></td>
</tr>
</tbody>
</table>

**Arithmetic operations**

<table>
<thead>
<tr>
<th>Procedure call</th>
<th>Definition</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD(SP,i,j,p)</td>
<td>( SF_{ij} := SF_{ij} + p )</td>
<td></td>
</tr>
<tr>
<td>ASSIGN(SP,i,j,p)</td>
<td>( SF_{ij} := p )</td>
<td></td>
</tr>
<tr>
<td>SUBTR(SP,i,j,p)</td>
<td>( SF_{ij} := SF_{ij} - p )</td>
<td></td>
</tr>
<tr>
<td>p := VERDI(SP,i,j)</td>
<td>( p := SF_{ij} )</td>
<td>integer function</td>
</tr>
</tbody>
</table>

Except from input/output operations these are the only standard procedures operating on partial words.
### I/O-operations (inclusive drum operations)

<table>
<thead>
<tr>
<th>Procedure call</th>
<th>Comment</th>
</tr>
</thead>
</table>
| **DMP(p,A,B,C...N)** | The procedure has an arbitrary number of parameters and transfers from core to an external media those vectors, matrices and partial word-areas listed from the second parameter and on. The first parameter, p, is defined as follows:  
- p=1 transfer to drum  
- p=0 " " printer  
- p=1 " " tape unit A  
- p=2 " " tape unit B , etc.  
Printing is done in a special format. DMP does not destroy the core areas. |
| **LOAD(p,A,B,C...N)** | The complementary routine of DMP, i.e. transfer from an external media to core. If the external media is a tape unit, the LOAD-call must correspond to a DMP-call, otherwise this is unnecessary. |
| **DLOAD(A,i,b)** | b:=A,. Matrix A is allocated on drum. It is assumed that DIM[b]=DIM[A]. |
| **DSTORE(A,i,b)** | A[i]:=b |
| **INTDMP(A)** | Printing of logical matrix (vector) A. Instead of a complete printing of all logical values, only the position of all 1-values of a row in A is printed. This routine is often very useful. |
| **EXTDMP(A,EXTERN)** | Analogous to INTDMP but instead of printing the position of 1-values, the contents of corresponding index in an integer array EXTERN is printed. (The external representation of the 1-values.) |
| **DMP1(p,A,t)** | The first parameter, p, is defined as follows:  
- p=2 transfer to drum area 1  
- p=3 " " drum area 2, etc.  
Second parameter states which core area is to be stored, and third parameter, t, is an identifier indicating to which item no. in the drum area, the data area is to be transferred. The item length of the various drum areas is constant. The purpose of the procedure is temporary storage of consequences of various possibilities, and later it will be decided which possibility to use. There is random access to a drum area. |
| **LOAD1(p,A,t)** | The complementary routine of DMP1. (In this case t is the chosen possibility.) |
| **CLEAR(A,B,C...N)** | The procedure has a random number of parameters, and is used as an initialization procedure. All data areas are nullified except those mentioned in the call which are given the value 1. |
Other standard functions and operators.

These are not particular for the current data structure, but certain frequently occurring operations will be mentioned:

1. \( a := \text{MAX}(x,y,z) \)
   \( a := \text{MIN}(x,y,z) \)

\( a \) gets the highest (or lowest) value of the variables given in the call. These symbols are also used in a slightly different way:

   \( a := \text{MAX}(f(i)) \) or \( a := \text{MIN}(f(i)) \)

\( f \) is a function of \( i \), and \( a \) symbols the maximum or minimum \( f(i) \).

2. // symbols integer division.
5. THE CENTRAL CONFLICT TYPES

Summary

(3.13) outlines the strategic principles. These will be applied on some conflict types. This chapter discusses two of the strategic principles:
1. Generation of the conflict picture.
2. Definition of global conditions for LEDIG\textsuperscript{S+1}.

By central conflict types is meant the absolute requirements common to any school structure. These are:
1. direct resource conflict,
2. complex resource conflicts (room conflicts),
3. variable period-length, and
4. day conflicts.

3) and 4) above are two main forms of an interval conflict. The treatment of the above conflict types outlines how other conflict types should be handled, and these will be mentioned later. (The continuity requirement might also reasonably be regarded as a central conflict, but the most interesting school structures are tight, and this requirement is then obvious.)

For each of the mentioned conflict types one will evaluate the possible modifications of the conflict picture and find global conditions. This will be connected to the data structure, and approximations and simplifications will be done to define suitable algorithms. Certain qualitative estimates of the limitations will be done.

The most important conflict type is direct resource conflicts, and based on these two central concepts are defined: Terminal combination \([TC]\) and utility \([G]\). These concepts will be repeated in some form for almost all conflict types or interaction of conflict types.

It will be demonstrated that the strategy finds necessary global conditions. These are not simultaneously sufficient, but from an operative point of view this is relatively uninteresting.
5.1 Direct resource conflicts

5.1.1 Necessary conditions

The requirement is defined in (3.1). Any scheduling method discusses this requirement, but a presentation for a general school structure is lacking. The requirement is trivial, whereas the practical consequences are large and not always evident. Generating of the conflict picture and the conditions for the freedom picture are the results of an interaction between the requirements for the system, but due to the dominating role of the direct resource conflicts one may say that other requirements are "superposed" and act as a modifying factor for the resource conflicts. The conflict picture will then be a result of all requirements, whereas one assumes existence of a solution if the resulting freedom picture satisfies the conditions which are a consequence of the resource conflicts. This simplified presentation will gradually be modified.

Direct resources included in the various activities are represented by a logical matrix KOMB, called resource matrix. (Later to be called TC-matrix.) Assume that the system consists of n activities and m sets of direct resource-conflicts. KOMB gets the dimension \([m,n]\), see fig. 5.1.

```
  1 1 1
  1 1 1
  1 1 1
  1 1 1
```

Fig. 5.1

Resource matrix KOMB

KOMB is generated directly from the specification of the problem, and is defined as follows:

\[
KOMB_{ij} = \begin{cases} 
1, & \text{if resource i is included in activity j} \\
0, & \text{otherwise} 
\end{cases}
\]

Accordingly:

\[KOMB_i = \text{All activities where resource i is included. The vector is called the combination for resource i.}\]
$KOMB_i^T$ = All direct resources included in activity $i$. The vector is called the descriptor of activity $i$.

(3.1) may therefore be written as follows:

\[
0 \leq \sum_{k=1}^{n} (KOMB_i AFFT_j)_k \leq 1 \quad i = 1,2,\ldots,m \quad j = 1,2,\ldots,TMAX
\]

A matrix, $KOLMA$, with dimension $[n,n]$ is defined as follows:

\[
KOLMA := \sum_{i=1}^{m} (KOMB_i f(KOMB_i \vee KOLMA))
\]

$KOLMA$ is called the conflict matrix of the system, and it is a "superposition" of the direct resource conflicts. An identical definition for $KOLMA$ is:

\[
KOLMA_{ij} = KOMB_i^T \land KOMB_j^T
\]

i.e., $KOLMA_{ij}$ is 1 if activity $i$ and activity $j$ cannot be allocated to the same time and 0 otherwise. Relation (5.4) shows that $KOLMA$ is symmetrical ($KOLMA_i = KOLMA_i^T$).

A simple algorithm for (5.3) is:

```
for i:=1 step 1 until m do
begin
  LOADH(KOMB,i,q1);
  j:=0;
L1:  j:=NEXTB(q1,j,n,L2);
  RRTOR(KOLMA,j,q1);
  goto L1;
L2:  end;
```

If the resource matrix, $KOMB$, is as in fig. 5.1, an evaluation of $KOLMA$ from (5.3) will be as in fig. 5.2.

Note that $KOLMA_i$ says nothing about the relation among other activities, i.e., if $KOLMA_{ij}=1$ and $KOLMA_{ik}=1$, nothing can be concluded about the value of $KOLMA_{jk}$.

Generally one will find the set of activities, $a$, which conflicts with all activities of a given set, $b$. Assume that $b$ consists of the activities $i$ and $j$. $a$ will then be:
\[ a := \bigwedge_{i \in b} KOLMA_i \bigwedge_{i \in b} KOLMA_i \]

If \( b \) consists of the activities: 1, 2, ---, \( p \), one gets:
\[ a := \bigwedge_{i=1}^p KOLMA_i \]

This may be written generally:

\[(5.6) \quad a := \bigwedge_{i \in b} KOLMA_i \]

\( a \) is defined as the selection product of the selection \( b \mid KOLMA \), and
\[(5.6) \] will be read as follows: \( a \) is the logical product of the
vectors making up the selection \( b \mid KOLMA \). Accordingly:

*If the conflict matrix is the selection object, all activities in a selection product will be in conflict with all activities in the corresponding selection vector.*

This means that if activity \( k \) is allocated to time interval \( t \) all (other) activities included in a selection vector where \( k \) is included in the selection product, cannot be allocated to time interval \( t \).

Important parameters for a system are:

\[(5.8) \quad \text{Subcombination (SC): If } b \in \bigwedge_{i \in b} KOLMA_i, \ i.e., \text{ the selection vector is a subset of the selection product.} \]

\[(5.9) \quad \text{Terminal combination (TC): If } b \in \bigwedge_{i \in b} KOLMA_i, \ i.e., \text{ the selection vector is identical with the selection product.} \]

A somewhat simpler formulation may be:

The vector \( b \) is a SC if \( KOLMA_{ij} \equiv 1 \) for all \( i \) and \( j \in b \). For \( b \) to be a TC too, demands that there is no activity \( k \) which is not included in \( b \) and which satisfies the condition \( KOLMA_{ik} \equiv 1 \) for all \( i \in b \).

Since the activities of a TC (or SC) are included in a selection product as well as in the corresponding selection vector, all activities in a TC (SC) are conflicting. Any SC is necessarily a subset of one (or several) TCs.

Accordingly:
The smallest number of time units necessary to allocate a TC (or SC) will be the arithmetic sum of time units for each activity included in a TC (or SC).

This is maybe the most important sentence applying to a schedule. Assume that those time units not yet allocated for the various activities are specified in an arithmetic vector $SP$ (the original $SP$ is known from the specification of the problem, and it may easily be modified for each step in the allocation process).

$$\sum_{i=1}^{TMAX} (LEDIG^{S} \cap b)^{i} \geq \sum_{i \in b}^{SP^{S}}$$

for all $b$ satisfying (5.8).

The number of time units making up a set of activities, and which are not yet allocated, is symbolized by the operator $TT$. In this case:

$$TT(b):= \sum_{i \in b}^{SP^{S}}$$

Another way of writing (5.11) is:

$$|LEDIG^{S} \cap b| \geq TT(b)$$

for all $b \in \mathcal{KOLMA}_{i}$

Index $s$ in (5.13) indicates how far the allocation has come, and (5.13) must be satisfied for each step in the process; i.e., it must be demanded that the rest of the allocation possibilities for a TC (or SC) are larger or equal to the number of time units not yet allocated.

Based on (5.13) a TC or SC may be classified as follows:

1. **Slack terminal combination (STC) or slack subcombination (SSC):** $|LEDIG \cap b| > TT(b)$
2. **Tight terminal combination (TTC) or tight subcombination (TSC):** $|LEDIG \cap b| = TT(b)$
3. **Broken terminal combination (BTC) or broken subcombination (BSC):** $|LEDIG \cap b| < TT(b)$
If a TC consists of \( q \) activities, one gets \( 2^q \) conditions from (5.13) for each TC. It is desirable to transform (5.13) to such a form that when the conditions for the TCs of the system are satisfied, all conditions for the SCs of the system are implicitly satisfied.

Assume that the number of TCs is \( km \) and that these are represented by a matrix \( B[km,n] \). One writes:

\[
(5.14) \quad F_i = LEDIG \land B_i
\]

\( F_i \) is called the freedom vector for \( B_i \) and \( |F_i| \) is called freedom for \( B_i \). \( F_i \) is divided into a set of linearly independent vectors \( h^1, h^2, h^3, \ldots h^P \); i.e.:

\[
(5.15) \sum_{j=1}^{P} h_j^i = F_i \quad \text{and} \quad h_j^i h_j^i = 0 \quad \text{for} \quad i \neq j
\]

One wants to know the maximum number of time units from an interval \( h_j^i \) which can be utilized by \( B_i \), and this parameter is called the utility of the interval, \( g_j \). The SC of \( B_i \) which may be allocated to \( h_j^i \) is:

\[
(5.16) \quad c_j^i := \sum_{k \in h_j^i} (LEDIG \land B_i)_k
\]

\[
(5.17) \quad g_j := \text{MIN}(|h_j^i|, \ TT(c_j^i))
\]

i.e., the utility of \( h_j^i \) is the smallest of the values:

1. The number of time units in \( h_j^i \).
2. The number of time units from \( B_i \) which may be allocated to \( h_j^i \).

If \( |h_j^i| \geq \ TT(c_j^i) \), then all activities in \( c_j^i \) must be allocated to \( h_j^i \) to maximize utilization of \( h_j^i \). One writes:

\[
(5.18) \quad r_j := \text{MAX}(0, |h_j^i| - \ TT(c_j^i))
\]

\( r_j \) is called the freedom loss of \( h_j^i \).

The vectors \( h_j^1, h_j^2, \ldots h_j^P \) shall be found so that:

\[
(5.19) \quad G(\text{LEDIG}, B_i) := \text{MIN} \left( \sum_{j=1}^{P} (|h_j^i| - r_j) \right) = \text{MIN} \left( \sum_{j=1}^{P} g_j \right)
\]

\( G(\text{LEDIG}, B_i) \) symbolizes the utility of the freedom matrix, LEDIG, with regard to TC, \( B_i \).
G is an operator of the form $G(A, b)$ defining utility of matrix $A$ with regard to vector $b$. Later, the most common use of $G$ will be $G(\text{LEDIG}, \text{KOMB})$, abbreviated to $G_{\bar{1}}$. 

(5.13) may be rewritten as:

$$G(\text{LEDIG}^g, B_{\bar{1}}) \geq \text{TT}(B_{\bar{1}})$$

for all $B_{\bar{1}} = \begin{bmatrix} \mathbf{A} & \mathbf{KOLMA} \end{bmatrix}_j$

(5.20) is the most important necessary condition for the satisfaction of (5.2). It may limit the solution space, and lead to certain activities getting forced assignments.

Assume:

$$\begin{align*}
\text{a. } & G(\text{LEDIG}, B_{\bar{1}}) \leq \text{TT}(B_{\bar{1}}) \\
\text{b. } & \text{TT}(c_j) \leq \lfloor h_j \rfloor \times \text{TT}(B_{\bar{1}}) \\
\text{c. } & r_j = 0
\end{align*}$$

(5.21) says that $B_{\bar{1}}$ is a TTC (or BTC), (5.21.b) and (5.21.c) say that $c_j$ must be allocated to $h_j$ to be able to allocate the maximum number of activities of $B_{\bar{1}}$. These forced assignments may be represented by a limitation of the solution space:

$$\begin{align*}
\text{(5.22) } \text{LEDIG}_{\bar{1}} := \\
\begin{cases} \\
\text{LEDIG}_{\bar{1}} + (\mathbf{A} \cdot \mathbf{c}_j \cdot \mathbf{KOLMA} + c_j) \text{ for } i \in h_j \\
\text{LEDIG}_{\bar{1}} + c_j \text{ for } i \in (F_{\bar{1}} \cdot h_j)
\end{cases}
\end{align*}$$

If (5.21.a) and (5.21.b) are satisfied, but not (5.21.c), then (5.22) can only be done for the interval $(F_{\bar{1}} \cdot h_j)$. This involves a danger, since LEDIG contains illusory freedom due to the forced assignment of $c_j$ which cannot be represented completely in LEDIG.

(5.21) and (5.22) define an iterative process: (5.22) modifies LEDIG, which may lead to (5.21) being satisfied for a new $h_j$ for an arbitrary $B_{\bar{1}}$, and so on. This iteration expresses the interaction between the various SCs within a TC and the interaction between the various TCs. Sooner or later this iteration stops, and (5.20) must still be satisfied.
It will be shown that when (5.20) is satisfied, (5.13) is also satisfied. Assume:

(5.23) \[ |\text{LEDIG} \land s| < TT(s) \quad \text{for SC, s} \]

TC, \( t \) satisfies:

\[ s \in t \]

which gives:

\[ G(\text{LEDIG}, t) \leq |\text{LEDIG} \land s| + TT(t+s) < TT(t) \]

i.e., if (5.20) is satisfied for \( t \), then (5.23) cannot be satisfied for any \( s \in t \).

(5.20) may be formulated verbally:

To maintain the existence of a solution, it must be required for each step in the allocation process that:

1. The utility of a TC must be larger than or equal to the total number of hours for activities in this TC not yet allocated.

2. The conflict picture for each allocation must be so complete that no new conflicts can occur as a result of interaction between the TCs of the system.

3. Should there arise interactions between the TCs of the system, the allocation possibilities for a set of activities will be limited, which may lead to forced assignments.

Note that the requirements are trisected, which may be connected to three phases of the allocation strategy. 2 is used to find the consequences of a possible allocation. 1 is used to decide whether the resulting freedom picture is acceptable. 3 is taken care of by the rules for the allocation sequence. This trisection is maintained as other requirements than direct resource conflicts are introduced.

The concept TC allows a more general formulation of the original requirement (5.2). This is equivalent to:

(5.24) \[ |\text{FFT}_j \land B_i| \leq 1 \]

for all \( B_i \subseteq \bigwedge_{k \in B_j} \text{KOLMA}_i \)

and \( j = 1, 2, \ldots, \text{TMAX} \)

A special case of (5.20) are the simple but important requirements formulated in [19].
In principle these are: The sets of activities for a class, a teacher or a room must satisfy (5.20).

These sets of activities will be TCs, but due to the activity definition for a general structure, a TC is a much broader concept. (5.20) is a substantial generalization and a much stricter requirement than the above. It will be shown that other requirements may be suited to (5.20), and that interaction between various conflict types may be taken into account.

5.1.2 The TCs of the system

A TC is given by relation (5.9), and an algorithm will be defined to find the vectors satisfying this requirement. Prior to this, one should state the reasons why one uses TCs of activities which cannot be allocated to the same time as parameters for a system instead of generating TCs of activities which can be allocated to the same time. It should be simple to realize that the number of the first set of vectors increases approximately linearly with the resources of the system, whereas the other set of vectors increases exponentially with the resources of the system. For realistic problems, this will exceed practical limits for available storage space, and one is forced to define the TCs of the system as in the previous section. These implicitly define the combinations which can be allocated to the same time (relation (5.24)).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

TC:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC₁:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₂:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₃:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₄:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₅:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₆:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₇:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₈:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₉:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₁₀:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC₁₁:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 5.3

Fig. 5.4
Fig. 5.3 is an arbitrary conflict matrix, and fig. 5.4 shows the TCs describing it. In principle it is simple to generate the TCs of a system, but computing may be extensive, and the demand for storage space large. An effective algorithm assumes a certain transformation of basic data. Until now, it has been assumed that the various activities are given by the original specification, which may of course contain activities with the same descriptor (e.g., a teacher teaches a class in two subjects, and these are defined as two activities). (5.4) shows that all activities which have the same descriptor must necessarily be included in the same TCs. This is used to combine activities with common descriptor into one activity in the conflict matrix, and by means of another matrix keep track of which activities this represents externally. This administrative problem will not be discussed further. It is now assumed that KOLMA is constructed of activities with different descriptors, and the number of hours connected with each activity is the sum of time units for the external activities with the corresponding descriptor.

The TCs will be generated in an approximately lexiographical order (i.e., at first all TCs containing activity 1 are generated. The first of these are all TCs containing activities 1 and 2, etc.). The TCs of a system are characterized by:
1. If \( b \) is a TC where activity \( k \) is included, then \( b \in \text{KOLMA}_k \).
2. Assume that one wants to generate the TCs starting with activity \( k \). Activities with lower number than \( k \) may then be disregarded. (TCs containing these are already generated.)
3. If \( q_1 \) is a selection vector (with regard to KOLMA) with corresponding selection product \( q_2 \) so that \( q_1 \in q_2 \), it may be concluded that \( q_1 \) will form one or several TCs with the activities included in \( q_2 \).
4. If \( q_2 \in q_2' \) where \( q_2' \) is a selection product of \( q_1' \), then all TCs formed by \( q_1 \) and \( q_2 \) may also be evaluated from \( q_1' \) and \( q_2' \).
5. If \( \Xi(q_2+q_1)\equiv 1 \), then \( q_2 \) might be a new TC. If \( q_2 \) is a subset of an already generated TC, then \( q_2 \) is a SC, otherwise it is a new TC.
6. Assume activity \( i \in (q_2+q_1) \), and \( \text{KOLMA}_i \cap aq_2 = q_2 \). One may then conclude that all new TCs where \( q_1 \) is included also contain
activity i. If this condition is not satisfied, i is said to be a junction for selection vector q1. (The new branch q2 ∈ KOLMA_i and junction i are temporarily stored, and one proceeds to find those TCS included in (q2[i]). These do not contain activity i, which is eliminated from the current q1.)

From these elements an algorithm for finding the TCS of a system may be defined. Temporary storage of new branches and junctions is necessary. This is done in matrix TEMP and integer vector ST, whose dimension is assumed to be sufficiently large.

**Algorithm for calculation of the TCS of a system**

```
REGCLR(N,q3);
for k:=1 step 1 until N do
begin
  LOADV(KOMB,k,q1);
  q1 is already generated TCS where activity k is included
  LOADH(KOLMA,k,q2);
  REGSUB(q2,q3,q2);
  eliminates TCS which do not start with activity k
  STOREH(TEMP,1,q2);
  S:=T:=T1:=1; S1:=2; ST(1):=k;

Necessary initiation for area TEMP for generating of those TCS
starting with activity k.
L0: for i:=S step 1 until T do
begin
  LOADH(TEMP,i,q2);
  j:=ST(i);
L1: j:=NEXTB(q2,j,N,L4);
  RREAND(KOLMA,j,q2,q4);
q4 is the selection product for activities in q2 with lower number
  than j.
  if REGQL(q2,q4) then goto L1;
if q2=qi then all new TCS containing q2 also include activity j; i.e.,
this activity is no junction.
for j1:=S1 step 1 until T1 do
if RREIMP (TEMP,j1,q4) then goto L31;
```

if \(q^4\) is implied in another branch, then \(q^4\) cannot generate any
TC not derived from this.

\[
\text{l1} := \text{NEXTB}(q^4, j, N, L2);
\]
\[
\text{l2} := \text{NEXTB}(q^4, j_l, N, L3);
\]
\[
\text{l3} := \text{if } j_l > N \text{ then goto L4;}
\]

if \(\Sigma(q^4-q^2)\equiv 1\) a new TC has possibly been generated.

\[
\text{T1} := \text{T1}+1; \quad \text{ST(T1)} := j;
\]
\[
\text{STOREH(TEMP, T1, q^4));}
\]

\(q^4\) is stored in \(\text{TEMP}\) since more TCs may be generated from \(q^4\).

\[
\text{l3l} := \text{SETN(q2, j);}
\]
\[
\text{goto l1;}
\]

\(q^2\) is modified, and return is done to find more branches.

\[
\text{l4} := \text{j1} := 0;
\]
\[
\text{l5} := \text{NEXTB}(q1, j_1, K_{M}, L6);
\]
\[
\text{if RREIMP(KOMB, j1, q^4) then goto L7;}
\]

this TC is then already generated

\[
\text{goto L5;}
\]
\[
\text{KM := KM} + 1;
\]
\[
\text{STOREH(KOMB, KM, q^4);}
\]
\[
\text{SETB(q1, KM);}
\]
a new TC is generated and stored in \(\text{KOMB}\)

\[
\text{l7} := \text{end;}
\]
\[
\text{if T1 > T then begin}
\]

There are still unexamined branches.

\[
S := T + 4;
\]
\[
T := T1;
\]
\[
S1 := T1 + 1;
\]
\[
\text{goto L0;}
\]

\end;
All TCs for the system are stored in KOMB when the loop terminates. The outlined algorithm is fast, but for large conflict matrices it may demand unduly storage due to inefficient use of area TEMP. A different administration of storage, analogous to what is done by means of S, T, S1 and T1, can be done.

Assumed that the conflict matrix is as shown in fig. 5.3, fig. 5.5 shows the calculation by means of snapshots of q2 and q4 for each step of the j-loop. The comments to fig. 5.5 outline the consequences of the vectors in question.

<table>
<thead>
<tr>
<th>No</th>
<th>k</th>
<th>q2</th>
<th>j</th>
<th>q4</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>TEMP₂ = q4 (ST₂ = 2)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>KOMB₂ = q4 (New T)</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>TEMP₃ = q4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>q₄ ∈ TEMP₃</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>10</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>TEMP₂ = q₄</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>TEMP₂ = q₄</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>q₄ ∈ TEMP₂</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>10</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>TEMP₂ = q₄</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>10</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>KOMB₂ = q₄</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>10</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>10</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>q₄ ∈ KOMB₂</td>
</tr>
</tbody>
</table>

Fig. 5.5
A detailed study of fig. 5.5 reveals that after step 19 nothing new occurs, and furthermore there is a previous temporary storing not resulting in a new TC. These unnecessary operations could have been avoided if for each step it had been investigated whether q2 or q4 were implied in a previously generated TC. Whether this ought to be done, depends on the frequency of this occurrence.

Furthermore, it is known beforehand that the combinations for the various resources is a TC (only exceptionally this is not the case); thus a certain initiation of KOMB may be done. Such improvements are not discussed here.

It is desirable to have an estimate of how many TCs a system can consist of. The theoretical limit may be established by an expression of the form:

\[ k_m = \sum_{i=0}^{d} \binom{d}{i} p_1(i) p_2(i) \]

where \( k_m \) = the number of TCs
\( d \) = the number of descriptors
\( p_1(i) \) = the probability for \( i \) descriptors forming a SC
\( p_2(i) \) = the probability for this SC also being a TC

An estimate for \( k_m \) is:

\[ k_m = \sum_{i=0}^{d} \binom{d}{i} p^i (1-p)^{d-i} \]

(5.25)

where \( p \) = the probability for two arbitrary descriptors conflicting

This upper limit assumes that the descriptors are defined stochastically. It does not take into account that the school structure defines the descriptors according to definite rules, and the limit is too large to have practical interest. Thus the number of TCs will be estimated on the basis of properties of practical structures:

1. The descriptors of the system are called \( D_1, D_2 \ldots D_p \). To describe which resources form \( D_i \) is written \( D_i(r_{i1}, r_{i2}, \ldots) \).
2. Assume that a class (student group) is included in all descriptors, and that the system consists of \( c_l \) classes. Most of the descriptors of a system consist of 2 (or 3) resources. A class
is on the average included in \( r^k \) descriptors. A descriptor which includes class \( k \) is called \( b^k \).

3. The probability is \( p(j) \) for \( j \) \( D^k \)-descriptors satisfying:

\[
\begin{align*}
D_1^k \land D_x & \equiv 1 \\
D_2^k \land D_x & \equiv 1 \\
& \quad \vdots \\
D_j^k \land D_x & \equiv 1
\end{align*}
\]

where class \( k \) is not included in \( D_x \)

4. The requirements for forming of a new \( TC, b \) which is not a direct resource conflict:

\[
\begin{align*}
D_i \land D_j & \equiv 1 \quad \text{for all } i \text{ and } j \in b \\
D_i \land D_j & \equiv 0 \quad \text{for at least one value of } i \in b \text{ and } j \in b \quad \text{(otherwise } b \text{ is a SC)} \\
D_i \land D_j & \equiv 0 \quad \text{otherwise } b \text{ is a direct resource conflict}
\end{align*}
\]

5. The largest number of resources (except from student groups) in a descriptor is \( rm \).

The simplest way to satisfy (5.26) is by using the activities defined by the descriptors.

\[
D^k(r_i) + b^k(r_j) + D(r_i, r_j, \ldots)
\]

(For each of these descriptors there must exist a set of activities in order to form a new \( TC \). Example: A new \( TC \) is: the activities of teacher \( r_i \) in class \( k \) plus activities of teacher \( r_j \) in class \( k \) plus activities in which both teachers \( r_i \) and \( r_j \) take part outside class \( k \).)

Another form of a new \( TC \) is:

\[
D^k(r_i) + D^k(r_j) + b^k(r_x) + D(r_i, r_j, r_x, \ldots)
\]

etc.

The number of new \( TCs \) with at least two \( D^k \)-descriptors are:

\[
km_1 = \sum_{i=2}^{rm} (p(i) \cdot r^k)
\]

It is obvious that \( p(2) \gg p(3) \gg p(4) \) etc.; accordingly
\[ k_{m_1} \leq c_i(r_k)p(2) \]

Note particularly:

If there is a pure class structure (i.e., all descriptors consist of two resources: one class and one teacher), then \( p(i)=0 \) for \( i \geq 2 \); i.e., for a pure class structure the number of TCs (maximum) equals the sum of classes and teachers.

For most school structures, \( k_{m_1} \) defines most TCs of a higher order, but when the system consists of a number of descriptors with several resources, (5.26) may be satisfied for several sets of descriptors, none of them consisting of two resources. This depends on the school structure. Assume that DX is the number of descriptors with many resources. (5.25) may be used to estimate new TCs, but the following empirical rule for a streamed school structure is simpler:

\[ k_{m_2} \approx k_0 \cdot L \cdot DX \]

where \( k_{m_2} \) = TCs of higher order where no descriptor consists of two resources
\( k_0 \) = a constant
\( L \) = the number of streams

(A complete motivation for the expression for \( k_{m_2} \) would lead too far.)

Both L and DX are relatively independent of school size, and \( k_{m_2} \) increases slowly with increasing number of classes.

For a given school structure it is easy to estimate values for \( r_k \) and \( p(2) \), whereas \( k_0 \) must be decided empirically. Usually \( k_{m_2} \ll k_{m_1} \), and the number of higher order TCs is:

\[ k_m = k_{m_1} + k_{m_2} < 2k_{m_1} < c_1 \cdot r_k q \cdot p(2) \]

(For a Norwegian comprehensive school it may be assumed that \( p(2) \approx 0.3 \) and \( r_k \approx 10 \); i.e., \( k_m \approx 30 \cdot c_1 \).

Based on this estimate the number of TCs may be large, and usually one is forced to neglect a number of TCs due to computing time. The set of TCs used should have such qualities that the probability of (5.20) limiting the solution space is high. In part this must be based on practical knowledge of a school structure, but it may also be derived from more general principles:
Assume that a system will be described by maximum $km$ TCs. These should be defined to satisfy the following conditions:

1. The set of TCs should be as "tight" as possible.
   A weight for a TC is: 
   $$\omega_1 = \frac{TT(B_i)}{G_i}, \text{ where } B_i \text{ is a TC}$$
   The probability for (5.21) and (5.22) giving new conflicts increases with increasing $\omega_1$.

2. The set of TCs should be as "orthogonal" as possible. This is defined as:
   $$\omega_2 = \frac{TT(B_i \cap B_j)}{TT(B_i) \cdot TT(B_j)}$$
   If $\omega_2 = 1$ then $B_i$ and $B_j$ are independent, and if $\omega_2 \approx 0$ it is very probable that if $B_i$ satisfies (5.20), then $B_j$ also satisfies this condition ($B_i$ and $B_j$ are almost identical activities in this case).

3. In addition to requirements of tightness and orthogonality it is important to include TCs with descriptors consisting of a large number of resources. Assume that vector $a$ defines such descriptors. By finding TCs from matrix $a$/KOLMA instead of KOLMA, this matter is taken into account. The following weight describes that TC-property:
   $$\omega_3 = \frac{TT(B_i \cap a)}{TT(B_i)}$$
   The reason why these activities are specially considered is that when the system is contradictory, the activities with many resources should be allocated to render it possible to adjust the schedule manually.

The $km$ TCs describing the system are determined by a maximizing of the "weight" of the various TCs. This may be formulated as:

$$S := \text{MAX} \prod_{i=1}^{km} f_1(\omega_1(i)) \cdot f_3(\omega_3(i)) \cdot \prod_{j=1}^{km} f_2(\omega_2(i,j))$$

Approximately $S$ is determined by:
(5.27) \[ S := \text{MAX} \left( \sum_{i=1}^{km} \left( \frac{TT(B_iA)}{G_i^0} \right) \sum_{j=1}^{km} TT(B_iW_j) \right) \]

where \( G_i^0 \) = initial utility

For most school structures the direct resource conflicts have proved to be sufficient to describe the system. Particularly difficult structures should in addition consider those TCs where many activities consist of more than 3 resources.

The above considerations indicate that a practical method necessarily must contain heuristic elements. The expressions are more or less qualitative, but they are important because they indicate which heuristic principles ought to be used.

Relation (5.20) (or analogous relations) are of almost no value without realistic proportions, and vice versa, heuristic methods should be motivated from fundamental conditions and not pure intuition. A combination of this creates a general and operative method.

It has been pointed out that activities with a common descriptor may be combined. Conditions between the various periods of activities with common descriptor are called internal conditions, and conditions between different activities are called external conditions. By combining activities the problem may be compressed, but the internal conditions become more complex thus leading to increased computing time and storage space. Except from evaluation of TCs this work is not based on combining activities. On the contrary, one tries to simplify the internal conditions, but the alternative is interesting.

The number of TCs for a system is a factor stating approximately how complicated the problem is, which should have consequences for the cost of making a schedule. Usually the price is a function of the number of classes, whereas it should be a function of the number of classes as well as school structure.

### 5.1.3 Evaluation of the utility \( U_G \)

\( G_i = G(\text{LEDIG}, B_i) \) is given by (5.19). Assume that freedom vector \( F_i \) consists of \( T \) time units. There are \( 2^T \) vectors \( h^i \) satisfying the
requirement \( h^j \in F^j_i \). In principle it is simple to define a method examining all possibilities for an evaluation of (5.19). This is uninteresting for realistic models.

Assume: \( m \) allocations will be done, each with \( p \) possibilities. The system consists of \( km \) TCs, and for each possibility \( G \) is modified for \( k_0 \cdot km \) TCs (0 < \( k_0 \) < 1). The freedom vectors consist on the average of \( T \) time units. The number of times (5.19) must be estimated is:

\[
q = m \cdot km \cdot k_0 \cdot p \cdot 2^T
\]

Even for a small system \( q \) becomes a considerable figure. Chain reactions as a result of (5.21) and (5.22) will increase \( q \) still further. A drastic reduction of \( q \) is the most important requirement to a method, but simultaneously the probability of finding correct \( G_i \) for all TCs should be large. There are several ways to reduce \( q \): \( m \) and \( km \) are given by the size of the system, whereas the average \( k_0 \) and \( p \) are reduced by making complete conflict pictures, and \( p \) is further reduced by the rules for the allocation sequence. \( G \) will be estimated by a method where computing time increases approximately linear to \( T \) and not exponentially.

One will evaluate conditions to render this possible, and give a qualitative estimate of the completeness of the method. The freedom vector \( F^j_i \) for TC\(_j\)B\(_i\) will be divided into three vectors (time unit sets):

**F1:** This set of time units always has maximum utility, independent of which blockings and forced assignments may arise for other time units in \( F^j_i \).

**F2:** For this set of time units the utility will be calculated, its number of time units will be gradually reduced, and relations will be defined which easily find the utility for the various subvectors in \( F^j_i \).

**F3:** This set of time units is constructed of a number of subvectors, for each of which is known the utility independent of blockings and forced assignments.

An auxiliary matrix, \( C \), is defined:

\[
C := \text{LEDIG} \land B^j_i
\]
(C is the part of LEDIG consisting of activities from TC, B_i. To simplify matters one may imagine the dimension of a row vector in C as |B_i|; i.e., the columns where all components are 0 are eliminated.)

One may say:

\[(5.28) \quad G^*_i := G(F_1 \setminus C, B_i) + G(F_2 \setminus C, B_i) + G(F_3 \setminus C, B_i)\]

where \( F_i \equiv F_1 \cup F_2 \cup F_3 \)

Abbreviated \((5.28)\) becomes:

\[G^*_i := \varepsilon_{F_1} + \varepsilon_{F_2} + \varepsilon_{F_3} = (|F_1| - r_{F_1}) + (|F_2| - r_{F_2}) + (|F_3| - r_{F_3})\]

Note that for the following calculations the vectors F1, F2, and F3 are modified when new conditions are satisfied.

Maximum \(G^*_i\) is called \(GMAX_i\) and the corresponding freedom loss \(RMIN_i\).

It is known:

\[(5.29) \quad \begin{cases} 
GMAX_i := \text{MIN}(|F_i|, TT(B_i)) \\
RMIN_i := \text{MAX}(0, |F_i| - TT(B_i))
\end{cases}\]

\(a(j)\) symbolizes the number of hours for activities of TC, B_i which may be allocated to time unit \(j \in F_i\).

\[(5.30) \quad a(j) := TT(\text{LEDIG}_j \cap B_i) = TT(C_j)\]

A vector \(f \in F_i\) consists of time units \(f_1, f_2, \ldots, f_p\)

Assume:

\[(5.31) \quad \begin{cases} 
a(f_1) \geq 1 \\
a(f_2) \geq 2 \\ \vdots \\
(\vdots) \\
a(f_p) \geq p
\end{cases}\]

The vectors satisfying \((5.31)\) might lead to new blockings in C, but these modifications alone will never give any freedom loss for \(f\), and the freedom loss must be determined by vector \((F_i \setminus f)\). When vector \((F_i \setminus f)\) has maximum utility, vector \(f\) has minimum utility.

Assume:

a. \(|F_i| \leq TT(B_i)|
A vector $f$ satisfies:

$$
\begin{align*}
\text{a}(f_p) & \geq |F_i| \\
\text{a}(f_{p-1}) & \geq |F_i|-1 \\
\vdots & \\
\text{a}(f_2) & \geq |F_i|-(p-2) \\
\text{a}(f_1) & \geq |F_i|-(p-1)
\end{align*}
$$

(5.32)

The rest of the time units ($F_i+f$) are assumed to have maximum utility, i.e., ($|F_i|-p$) time units of $B_i$ are allocated to ($F_i+f$). By subtracting ($|F_i|-p$) from each $a(f_j)$ in relation (5.32), (5.31) is still satisfied; i.e., vector $f$ can get no freedom loss.

b. $|F_i| > \text{TT}(B_i)$

A vector $f$ satisfies:

$$
\begin{align*}
\text{a}(f_p) & \geq \text{TT}(B_i) \\
\vdots & \\
\text{a}(f_1) & \geq \text{TT}(B_i)-(p-1)
\end{align*}
$$

(5.33)

Vector ($F_i+f$) is again assumed to have maximum utility, and subtraction gives:

$$
\begin{align*}
\text{a}(f_p) & \geq \text{TT}(B_i) - |F_i|+p = p - \text{RMIN}_i \\
\text{a}(f_{p-1}) & \geq \text{TT}(B_i) - 1 - |F_i|+p = (p-1) - \text{RMIN}_i \\
\vdots & \\
\text{a}(f_1) & \geq \text{TT}(B_i) - (p-1) - |F_i|+p = 1 - \text{RMIN}_i
\end{align*}
$$

The utility of $f$ is then:

$$
g_f:=\text{MAX}(0,p-\text{RMIN}_i)
$$

since $p-\text{RMIN}_i$ time units still satisfy (5.30).

If ($F_i+f$) has a freedom loss $r_{(F_i+f)}$, utility of $f$ increases accordingly:

$$
g_f:=\text{MAX}(0,p-\text{MAX}(0,\text{RMIN}_i-r_{(F_i+f)}))
$$

(5.34)

Combination of (5.32) and (5.33) gives that vector $F_i$ may be divided into two vectors $F_1$ and $F_2$:
\[ F_{i} := F_{1} \land F_{2} \]
\[ |F_{1}| := p \]
\[ |F_{2}| := q = |F_{1}| - p \]

where \( F_{1} \) satisfies:

\[
\begin{cases} 
    a(F_{1}^p) \geq G_{\text{MAX}}_i \\
    a(F_{1}^{p-1}) \geq G_{\text{MAX}}_i - 1 \\
    \vdots \\
    a(F_{1}^1) \geq G_{\text{MAX}}_i - (p-1) 
\end{cases}
\]

(5.35)

Utility of \( F_{1} \) is always given by (5.34), and by setting

\[ G_{i} := \varepsilon_{F_{1}} + \varepsilon_{F_{2}} \]

results in:

(5.36) \[ G_{i} := \text{MAX}(0, p - \text{MAX}(0, R_{\text{MIN}}_i - r_{F_{2}})) + \varepsilon_{F_{2}} \]

where \( r_{F_{2}} = \text{MAX}(0, q - \varepsilon_{F_{2}}) \)

The process of finding \( G_{i} \) is reduced to finding \( \varepsilon_{F_{2}} \); i.e., possible subvectors of \( F_{1} \) are reduced by a factor \( 2^P \).

Each time unit \( j \in F_{2} \) satisfies:

(5.37) \[ a(j) < q \]

(otherwise time unit \( j \) would have been included in vector \( F_{1} \))

The set of vectors minimizing \( \varepsilon_{F_{2}} \) is called \( \text{FMIN} \). Each time unit \( j \in F_{2} \) is characterized by vector \( C_{j} \). Assume:

(5.38) \[ C_{j} = C_{k} \]

Time unit \( j \) and \( k \) must necessarily be included in the same vector in the set \( \text{FMIN} \). (This may be shown by means of (5.18) and (5.19).)

Each time (5.38) is satisfied, possible subvectors are reduced by a factor 2. Time units satisfying (5.38) are combined:

In an arithmetic auxiliary vector \( S \) every component has initially the value 1. Each time (5.38) is satisfied the following is done:

(5.39) \[
\begin{cases} 
    S_{j} := S_{j} + S_{k} \\
    F_{2}^k := 0 
\end{cases}
\]

This modification is done iteratively until (5.38) is no longer satisfied for any of the remaining components in \( F_{2} \). If \( F_{2,j} \equiv 1 \),
then \( S_j \) in other words defines how many time units are characterized by the same vector \( C_j \). (Special administrative processes must be defined to keep account of which time units are represented by \( S_j \).) When these modifications have been done, the number of subvectors is reduced by a factor \( 2^{|F_2|} \) compared with a complete search process.

Necessary conditions to achieve maximum utility of \( g_{F_2} \) are:

(5.40) \( a(j) \geq S_j \) for all \( j \in F_2 \)

If \( a(j) \leq S_j \) for \( j \in F_2 \), then \( C \) must be modified due to (5.21) and (5.22) to achieve maximum utility for the corresponding time units (\( C \) replaces LEDIG in those relations). Assume that \( S_j \) represents time units \( f^j \). Modifications resulting from (5.21) and (5.22) lead to:

(5.41) \( C_k A C_j \equiv 0 \) where \( k \in f^j \) and \( j \in (F_2 + f^j) \)

(5.41) leads to \( f^j \) not having any further freedom loss due to forced allocations to other time units in \( F_2 \). Thus \( f^j \) may be eliminated from \( F_2 \) and transferred to \( F_3 \):

(5.42)

\[
\begin{align*}
F_2^j & := 0 \\
C_k^j & := C_k + C_j^j \\
F_3 & := F_3 + S_j - a(j) \\
F_3 & := F_3 \lor f^j
\end{align*}
\]

When introducing vector \( S \), (5.37) changes to:

(5.43) \( a(j) < \sum_{k \in F_2} S_k \) for \( j \in F_2 \)

(5.42) modifies \( a(j) \), and if (5.43) is no longer satisfied for some \( j \in F_2 \), the corresponding set of time units may be eliminated from \( F_2 \) and transferred to \( F_1 \).

A new auxiliary vector \( S_1 \) is defined:

(5.44) \( S_1^j := \sum_{k \in F_2} \delta_{jk} S_k \)

where \( \delta_{jk} = \begin{cases} 1, & \text{if } C_k \cap C_j \\ 0, & \text{otherwise} \end{cases} \)

A new necessary requirement for maximum utility of \( F_2 \) is:

(5.45) \( a(j) \geq S_1^j \) for all \( j \in F_2 \)
If \( a(j) \leq S_{1j} \), C is modified due to (5.21) and (5.22). Time units represented by \( S_{1j} \) are eliminated from F2 by means of (5.42) etc.

The requirements (5.40) to (5.45) define an iterative process ending when the inequality is satisfied for (5.40), (5.43), and (5.45). It is assumed that the remaining time units in F2 have maximum utility.

\[
\begin{align*}
G_{F2} &:= \min \left( \sum_{k \in F2} S_k \cdot TT(\mathbf{v}, C_k) \right) \\
R_{F2} &:= \max \left( 0, \sum_{k \in F2} S_k \cdot TT(\mathbf{v}, C_k) \right)
\end{align*}
\]

(5.46)

By means of (5.36) is found:

\[
G_{1} = \max(0, |F1| - \max(0, R_{MIN1} - (R_{F2} + R_{F3}) + G_{F2} + |F3| - R_{F3})
\]

(5.47)

(5.46) is an approximate expression, but the following claim is ventured:

For a realistic school structure the probability for \( G_{F2} \) being correctly estimated by (5.46) is approximately 1.

This assumption is important for the outlined method, and it can be verified from the following considerations:

1. Any vector \( f \in F2 \) satisfying the condition:

\[
a(j) \not\equiv a(k) \quad \text{for } j \text{ and } k \in f \text{ (and } j \not\equiv k)\]

must have maximum utility, as the corresponding time units always will satisfy (5.31).

For a realistic structure it is very probable that:

\[
C_j \in C_k \quad \text{or} \quad C_k \in C_j \quad \text{for } j \text{ and } k \in F2
\]

This probability is called \( p_0 \), and the number of vectors satisfying (5.48) are called \( n_1 \). It is simple to prove:

\[
(1 + p_0)|F2| < n1 < 2|F2|
\]

(5.50)

(The probability is rather small for \( LEDI_{G} \not\in LEDI_{C} \), whereas it is large for \( (LEDI_{G} \not\in B_1) \in (LEDI_{C} \not\in B_1) \), and with the shown iteration this probability is approximately 1.)

2. If \( C_j \not\equiv C_k \equiv 0 \), the time units \( S_j \) cannot lead to any freedom loss for time units \( S_k \). A new characteristic feature of realistic structures is that when (5.49) is not satisfied, the probability is large for \( C_j \not\equiv C_k \equiv 0 \). This considerably reduces the number
of subvectors which may lead to (5.46) no longer being correct.

3. The condition for (5.46) being wrong is that based on the resulting C, a vector \( d \in \sum_{k \in F_2} C_k \) may be found, such that:

\[
|CA \sum d| < TT(d)
\]

(5.51)

There is a minimum probability for (5.51) being satisfied when the mentioned iteration is finished.

[An exact formulation of the requirements to be satisfied by \( F_2/C \) are: For each \( f \in F_2 \), a set of \( l \)-components in \( C \) must be found, in such a way that from each row vector one component is included in the set, and from each column vector is included at the most as many components as the number of hours for corresponding activities. It is not difficult to define an algorithm to find out whether this condition is satisfied. (A method for an analogous condition is discussed in [38].) From a practical point of view this is not very interesting due to necessary computing time and because the outlined method is completely sufficient. The principal objections against it is that the interaction between column vectors in \( C \) are not sufficiently considered, but by means of relations (5.40) and (5.45) they are implicitly considered. As a curiosity it is shown how (5.40) can give a wrong \( y_{F_2} \):

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 1 & 1 &  &  &  &  \\
3 & 1 & 1 &  &  &  &  \\
4 & 1 & 1 &  &  &  &  \\
5 & 1 & 1 &  &  &  &  \\
6 & 1 & 1 &  &  &  &  \\
7 & 1 & 1 &  &  &  &  \\
\end{array}
\]

Assume that the hour number for each activity is 1. By means of (5.46) utility is 7, whereas the correct value is 6 (due to the vector consisting of activities 5, 6, and 7 satisfying (5.51)).]

4. By analyzing realistic problems it can be explained qualitatively why the outlined method is sufficient:

Assume that \( TC \), \( B_1 \) are activities for a certain class. A vector \( d \in B_1 \) satisfying (5.51) is usually those hours that only one teacher (exceptionally a small number of teachers) can have together with the class. These hours are likely to be represented by the same \( S_1 \), by means of (5.44), and (5.45) is then a sufficient requirement for finding possible freedom loss.
The outlined method will later be completed with other principles to improve the probability of (5.46) being correct.

The most important points concerning the outlined method for finding $G_i$ are:

1. Initially $F_1$ is found from (5.35).

2. $a(j)$ is found from (5.30), $S$ is found from (5.38), (5.39), and $S_1$ from (5.44).

3. $F_2$ is gradually reduced by means of (5.40), (5.43), and (5.45) ($S$ and $S_1$ must partly be reestimated when $C$ is modified).

4. $G_i$ is determined by (5.46) and (5.47).

This will be shown by an example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5.6

Figure 5.6 shows the initial value of $C$, and each activity is assumed to consist of 1 hour. To the right in fig. 5.6 are shown important parameters after the first execution of (5.44). Note that time units 10 and 11 satisfy (5.35), and that time units 4, $S$, and 6 satisfy (5.38) and are represented by $S_n$. $C$ is modified due to $S_{14}=a(4)$. Execution of (5.42) results in figure 5.7:
Time units 2, 4, 5, 6 are now represented by F3, and columns 1, 4, 5, 6 in C are modified. Now is found a(1)=S_1 and a(2)=S_2, which gives new modifications of C, etc. The resulting C is:

F2 is eliminated, and (5.47) gives S_1=ll; i.e., for fig. 5.8 B_1 has maximum utility.

This example is somewhat more complicated than realistic freedom pictures, just to show which iterations may occur. Ordinarily the inequalities (5.40) and (5.45) are fulfilled, thus leaving C unmodified. Accordingly:
An estimate of $G_i$ leads to an \textbf{approximately linear} increase in computing time when $|F_i|$ increases. (More correctly: computing time increases proportional to $|F_i|^\alpha$ where $1<\alpha<2$.) Computing time increases more than linear in those cases where the freedom picture is really modified. This property of the method is favorable, since the really important thing is to reduce computing time to a minimum when no new conflicts occur.

It has been pointed out that the utility of the various TCs will be determined quite a large number of times, and the outlined method quickly distinguishes between the utilities which must be determined approximately completely, and the less important ones. A method lacking this ability will either use too much computer time or be too inaccurate.

Certain simplifications may be done in practice; e.g. it may be assumed that the inequalities of (5.40) and (5.45) always are satisfied if $a(i)$ is larger than a certain value. Of course such empirical rules depend on which school structure is being examined. It has been assumed for the outlined method: The conflicts of the system consist only of direct resource conflicts, i.e., complex resource conflicts, variable period length, day conflicts, etc. are disregarded.

This limitation is far more important than the approximation done in (5.46). One may safely forget all hypothetical situations where (5.46) is wrong, and instead concentrate on estimating $G_i$, when interaction between the conflict types occur. The utility is important as a theoretical concept. It is an excellent guide for the more approximate and heuristic principles used in practice.

For each allocation, utility will be estimated for many TCs of the system. These are defined by the row vectors in KOMB. If $KOMB_i$ is a TTC (or the actual allocation leads to $KOMB_i$ becoming a TTC), possible modifications of C due to (5.40) and (5.45) must be transferred to LEDIGS. This might in its turn lead to the utility being modified for other TCs; accordingly, utility for the various TCs must be determined iteratively. The consequences of this iteration may be summed up:
a. Advantages
By making the conflict pictures as complete as possible at any time, solution space is limited, and the probability decreases for allocation of an activity so that no solution exists. This implies that qualitative criteria may get more influence.

b. Disadvantages
A considerable increase of computing time does not necessarily imply any important limitations of solution space. When no complete schedule exists, blockings due to (5.40) or (5.45) can often be done in several ways. By choosing a random one, one risks that the final (and incomplete) schedule is manually inadjustable, and from this viewpoint an unnecessary limitation of solution space is done.

For tight systems where the existence of a solution is known, a) is the most important consideration. For tight systems with no solution, manual adjustment possibilities should be considered, and for simple systems an increase of computing time should be avoided. These considerations lead to certain compromises by definition of a method which among other things influences the rules for the allocation sequence. In practice the following rules have proved useful:

For a TTC in KOMB, G is determined from a more complete method than what is necessary for a STC. Blockings due to forced assignments are partly represented by modification of the freedom picture and partly by giving priority to those activities which have got forced allocation, for the next stages. New blockings for activities which are complicated to adjust manually are seldom introduced, and due to computing time, interaction between TCs is not considered completely.

This has proved sufficient even for very tight structures. The above may be considered as a purely empirical rule, but it is really a practical approximation of the more exact principles discussed previously, and the frequency of new limitations of solution space as a function of computing time has been estimated.

It has been pointed out that the rules for the allocation sequence may compensate for incomplete conflict pictures. This can be shown by returning to fig. 5.6. Assume that $G_1$ is determined from simplified relations so that the only forced assignments discovered are that activities 1, 4, 5, and 6 must be allocated to time units 2,
4, 5, and 6. A rule is defined stating that these allocations must immediately succeed the current allocation. Thus the existence of a solution is assumed (as in fig. 5.8), but there is no guarantee. To improve probability of the existence of a solution, activities with limited allocation possibilities are immediately allocated.

The above is a rough outline of the rules to be defined for the allocation sequence. In practice one is forced to use far more complete rules, but they are based on analogous principles. This simple example shows that rules for the allocation sequence limit the number of possibilities to be examined as well as reduce the number of necessary computing operations for each possibility. In those cases where the rules "compensate" for incomplete conflict pictures, one assumes a solution to be existing. Practice has proved this principle to be useful.

For the calculation of G there is shown a gradual transition from an exact method to more empirical principles. It is easy to modify the calculation of G depending on the complexity of the problem. Another useful principle is: A simpler calculation of G is used for possible allocations than for a chosen allocation; in the latter case the conflict picture is more completely generated. It is assumed that these more complete conflicts preserve the existence of a solution.

5.1.4 A comment to the necessary and sufficient conditions for solution of the scheduling problem

a. It is known that when all the activities of a system consist of only one student group and one teacher (i.e., pure class structure) and room conflicts can be disregarded, then a solution always exists for a system consisting of only direct resource conflicts. (5.20) together with an appropriate strategy for the allocation sequence will be the necessary and sufficient conditions for providing the existence of a solution for each step in the allocation process.

A similar approach to the problem is known from other literature [12], and will not be discussed here. The result is important, but unfortunately it reveals little about sufficient conditions for solution of realistic problems. (The formalism introduced
here, is unnecessarily complicated for dealing with such a simple problem.)

b. One has hoped that analogous conditions as (5.20) would also suffice for realistic problems. [20] provides one of the first indications that this was not the case.

This simple example consists of 3 classes and 3 teachers meeting in pairs in one hour. Activity 1 consists of class 1 and teacher 1, activity 2 of class 1 and teacher 2, etc. If the example is transferred to the representation in question, then one gets:

<table>
<thead>
<tr>
<th>KOLMA</th>
<th>KOMB</th>
<th>LEDIG</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Table" /></td>
<td><img src="image" alt="Table" /></td>
<td><img src="image" alt="Table" /></td>
</tr>
</tbody>
</table>

Fig. 5.9

KOLMA, KOMB, and LEDIG being as in fig. 5.9, condition (5.20) is satisfied and no new conflicts are possible as a result of (5.21) and (5.22). An attempt to schedule the 9 activities within given time frame is impossible. The original assertion was that freedom pictures of the same form as LEDIG in fig. 5.9 will not arise for realistic problems, among other things due to the fact that reductions of the freedom picture as a result of (5.21) and (5.20) cannot lead to freedom pictures as in fig. 5.9. This is true, with certain modifications:

1. The freedom picture LEDIG in fig. 5.9 cannot possibly be a result of direct resource conflicts alone, since all classes and teachers have all hours free; neither can (5.22) with the given conflict matrix lead to the freedom picture in question. (This can most easily be seen by examining a random blocking in LEDIG, and finding that it cannot possibly be a result of the resources of the activity being part of forced assignments. The only
possible explanation might be that LEDIG is a result of incomplete conflict pictures.)

2. If a scheduling problem consists of other requirements in addition to direct resource conflicts, a freedom picture with the same properties as LEDIG in fig. 5.9 is perfectly possible.

Accordingly, the example in fig. 5.9 says nothing about (5.20) not being the necessary, as well as the sufficient, conditions for solution of a scheduling problem consisting of only direct resource conflicts and having pure class structure. However, it does offer the important practical information that interaction between direct resource conflicts and other requirements reduces (5.20) to just necessary conditions.

The main conclusion from fig. 5.9 is that a random prehistory destroys the sufficiency of (5.20). In [39] there is given a conclusive example which shows that conditions equivalent to (5.20) are not sufficient when the allocation sequence is arbitrary. However, this objection might possibly be avoided by defining suitable rules for the allocation sequence. There has been made no attempt to prove this, but the claim is verified by mentioning that the strategy in question has never failed to find a solution to a pure class structure schedule.

c. One may still hope that (5.20) might be a basis for defining sufficient conditions if a system consists only of direct resource conflicts; i.e., that (5.20) are sufficient conditions for a general conflict matrix. It is simple to show that this is not the case. In fig. 5.10 a small change from the conflict matrix in fig. 5.9 has been done, whereas LEDIG contains no blockings, i.e., the figure may be regarded as the initial situation for a system only consisting of direct resource conflicts.

Fig. 5.10 satisfies (5.20), and no reduction of LEDIG is possible. However, it is soon discovered that it is impossible to allocate all activities. Thus:

The conditions (5.20) are necessary, but not sufficient for a general conflict matrix.
d. The above arguments drastically reduce the possibilities of making sufficient and necessary conditions for the solution to a general scheduling problem because:

**Argument b** shows that the interaction between the various conflict types easily has the result that it becomes impossible to make sufficient conditions even for the simplest conflict matrices. Furthermore the rules for the allocation sequence will be of prime importance.

**Argument c** shows that even complicated conditions like (5.20) are only necessary conditions for a general conflict matrix.

Consequently, at the most, one may hope to achieve necessary and sufficient conditions for a very restricted set of conflict types, and simultaneously one cannot treat a general conflict matrix. From a practical point of view the value of this is very limited. Another possibility is to try more complex conditions than (5.20). But this is almost impossible due to consideration of computing time, and regardless of the complexity of the conditions, it is easy to define a new conflict type leading to the conditions being insufficient.

The consequences pointed out here are apparently paradoxical:

1. The previous paragraphs show that a completely combinatorial method may lead to enormous computing time and large demand for storage space, so that the method must be simplified in practice.
2. This paragraph shows that even if the method is complete, the conditions employed are insufficient.

However, the practical results obtained are surprisingly good, and the conclusion is that the only sensible measure for a method is the operative value discussed in chapter 1.

5.1.5 Modification of the conflict vector due to direct resource conflicts

The conflict vector, $TAP^S_j$, defines the set of new activities blocked for time unit $j$, due to allocation No. s. Below is assumed that activity $k$ is allocated to time unit $j$. Thus:

$$TAP^S_j = (LEDIG^S_j \land KOLMA_k)$$

$$= (LEDIG^S_j \land (\forall i \in KOMB^S_k))$$

These conflicts might introduce several more conflicts by means of (5.21) and (5.22). These conditions imply extensive search processes. It is desirable to find a simple expression covering those cases where (5.21) and (5.22) are usually fulfilled; i.e., one wants to make $TAP^S_j$ so complete that the consequences which (5.21) and (5.22) may have for (5.20) are implicitly fulfilled. New conflicts arising in this way are symbolized as $KV^S_j$; i.e.:

$$KV^S_j = (LEDIG^S_j \land KOLMA_k) \lor KV_j$$

Vector $W_9$ defines which TCs of the system are tight:

$$W_9 := \begin{cases} 
1, & \text{if } KOMB_{1} \text{ is a TTC (BTC)} \\
0, & \text{otherwise} 
\end{cases}$$

Assume:

$$q_1 := KOMB_k^T$$

$$q_2 := KOMB \land (LEDIG^S_j \land KOLMA_k)$$

i.e., $q_1$ is the TCs containing activity $k$, and $q_2$ is the TCs getting new conflicts (i.e., $q_1 \in q_2$).

$$q_3 := W_9 \land (q_2 \land q_1)$$

The TTCs defined by $q_3$ must also, after the assignment in question, still have activities available for time unit $j$. Thus:
\[(5.54) \quad q^i = A\ KOLMA^p_i \cap KOMB^i_{p} \quad \quad \quad p \in (KOMB^i_{1} \cap LEDIG^S_{j})\]

where \(i \in q^3\)

$q^i$ is the selection product with regard to KOLMA for activities of KOMB$^i_{1}$ still available for time units $j$, except those activities contained in KOMB$^i_{1}$. However, since KOMB$^i_{1}$ is a TTC none of the activities $q^i$ can be allocated to time unit $j$, because this implies that (5.20) is no longer satisfied for KOMB$^i_{1}$. Accordingly:

\[(5.55) \quad KV^j_i = \bigvee_{i \in q^3} q^i\]

The practical problem resulting from the above is:

*On which assumptions is vector $q^i$ unequal to the $0$-vector?*

It ought to be simple to realize that the above is satisfied when:

\[(5.56) \quad (KOMB^i_{1} \cap LEDIG^S_{j}) \epsilon KOMB^p_i\]

where $i \not\in p$

i.e., the activities of KOMB$^i_{1}$ still available for time unit $j$ must be a subset of activities available on the same time unit for another TC of the system. When (5.56) is satisfied, one may say:

\[(5.57) \quad q^i = \bigvee_{p \in 1} \quad (KOMB^p_i \cap \delta^p \cap KOMB^i_{1}) \cap LEDIG^S_{j}\]

where \(\delta^p = \begin{cases} 
1\text{-vector, if (5.56) is satisfied} \\
0\text{-vector, otherwise}
\end{cases}\)

The most common cases where (5.56) is satisfied are:

1. All activities for TTC, KOMB$^i_{1}$ available for time unit $j$ have 2 (or more) resources in common. Activities where only a subset of these resources is included are blocked for time unit $j$.

2. The activities for (KOMB$^i_{1} \cap LEDIG_{j}^S$) have one resource in common, furthermore at least one of a limited group of resources is included in all activities. In this case activities where the common resource is not included, but where the whole group of resources is included, are blocked.

This can be made more concrete by leaving the abstract TC-concept, and using a streamed school structure for example. The activities of a class are assumed to be tight:
1. If the class shall meet the same teacher (or use the same room) for all activities available for a certain time unit, then other activities including the teacher (or the room) must be blocked for that time unit. The same rule is valid for the activity set including a teacher or a room, if one of these combinations is tight.

2. If the activities available for a class for a certain time unit only consist of parallel activities (i.e., student groups from several classes are included), then the activities where a subset of these student groups is included must be blocked for that time unit.

3. The activities available for the class are a combination of the above mentioned possibilities; i.e., parallel activities and pure class activities where the same teacher (or room) is included. In that case the activities where the teacher teaches other student groups within the same stream must be blocked.

Most of the new conflicts arise for the time units where an allocation is done, and in practice it has proved sufficient to determine KV for those time units, and that the above points are sufficient.

The above rules may appear empirical and heuristic, but they show the gradual transformation from a general formulation to operative principles:

1. (5.20) is a necessary condition for the existence of a solution. (5.21) and (5.22) result from that condition and lead to a limitation of the solution space.

2. It is time-consuming to estimate the utility, and it is done by means of approximate expressions. It is desired to generate the conflict picture as completely as possible, to implicitly satisfy the most important limitations due to (5.20). (5.54) and (5.55) are formulations making this possible. (These might be formulated more generally by considering sets of time units, but due to computing time some of the purpose would then be lost.)

3. The most common ways of getting new conflicts from (5.54) are found by means of an analysis of the activity concept.
4. This is transformed to operative rules based on characteristic features of various school structures.

5. From a large number of realistic models is found that new conflicts due to (5.54) need only to be examined for those time units where an allocation is tried, since the frequency for this having consequences for other time units is low.

It is not always so simple to show the connection between general formulations and operative principles, and due to lack of space such analyses must partly be omitted. An objection to defining operative rules is that they will depend too much on the current school structure, but it is often possible to define the rules generally enough to cover any school structure.

It has been pointed out that it is not useful to generate a complete set of TCs for the system, and that the reduction is done based on orthogonal considerations and an evaluation of which activities are the most important for a system. Further, a complete calculation of (5.20) is not done. This might be dangerous since the critical conditions are not necessarily the same throughout the allocation process, and important conditions may be disregarded. To compensate this a certain approximation is done:

For important interactions between direct resource conflicts certain SCs for the system are regarded as if they were TCs.

The consequence of this is that if any of these SCs are found to be critical, new "TCs" are generated for the system, in other words, as a result of certain allocations a number of new "TCs" can be generated which shall satisfy (5.20). for the rest of the allocation process.

(5.21) may be formulated verbally as:

If a TC is tight, and a SC of this TC is also tight, the TSC must be allocated to the interval where the TSC is available. If the utility of the TSC in question is less than the length of the interval, LEDIG cannot be modified completely. However, it is desirable to represent the information that a TSC has been found. It is done in two ways:
1. The activities forming a TSC modify the allocation sequence so that they are allocated as quickly as possible.

2. The found TSC is considered an independent TTC and becomes a "new" condition for the rest of the allocation process.

The probability is much larger for the interaction between two resources leading to a TSC than for the interaction between several resources leading to a TSC.

These consequences of direct resource conflicts will be discussed later. Generally speaking one tries to secure the existence of a solution based on interactions between classes, teachers, rooms, and time units. (The latter factor will prove to be important. It is not sufficient for a schedule to exist from the "viewpoint" of each resource; it must also exist from the "viewpoint" of each time unit.)

5.2 Complex resource conflicts

5.2.1 Necessary conditions

The conflict type is defined by (3.2), and a direct resource conflict is a special case of this. These conflict types will be treated analogously; there is, however, an important difference:

A direct resource conflict defines explicitly which activities cannot be allocated to the same time unit, whereas a complex resource conflict states how many activities of a set of activities can be allocated to the same time unit. E.g.: activities i, j, and k may form a complex resource conflict so that they in pairs may be allocated to the same time unit, but all three activities cannot be allocated to the same time unit.

The reason why the two types of resource conflicts are used, is the specification possibilities necessary for a correct formulation of the problem. Several of the resources included in an activity are unambiguously defined. For example, it is required that a certain student group or a certain teacher are included in the activity, whereas one is often content when an arbitrary room from a more closely defined group is included in the activity. If for the latter resource requirement, it is specified that a certain room is included, it means a limitation of the set of activities which can be
allocated to the same time unit and which make up a complex resource conflict. This possibility drastically limits the solution space, and it should not be used unnecessarily.

For the complex resource conflicts it is impossible to define a conflict matrix and generate the corresponding TCs, since a complex resource conflict is no relation between the activities in pairs. This implies that a condition like (5.20) cannot be used directly; which would have been very desirable since that condition limits the solution space considerably. It is, however, desirable to complete the conflict picture $\text{TAPT}^5$ with the consequences of complex resource conflicts. For each stage in the allocation process it is known that a new complex resource conflict necessarily arises only in the time interval where an allocation is done, because this conflict is a relation between activities being allocated to the same time unit.

These conflicts will be represented by the vector $\text{RR}_j^5(\text{FFT}_j^5)$. (5.52) may be transformed to:

$$\text{TAPT}_j^5 = (\text{LEDIO}_j^5 \land \text{KOLMA}_j^k) \lor \text{KV}_j^5 \lor \text{RR}_j(\text{FFT}_j^5)$$

Complex resource conflicts are symbolized $\text{RR}_j(\text{FFT}_j^5)$ to indicate that they are approximately a pure function of the activities allocated to time unit $j$; i.e., the prehistory. ($\text{RR}_j$ is of course also a function of the room resource requirement of the activity in question.) The resulting freedom picture from (5.58) must still satisfy (5.20), and in that way the combination of the two conflict types is considered.

Direct and complex resource conflicts are treated equally; however, the complex resource conflicts are not found from the conflict matrix but from more complexly defined conditions.

A principal objection to this is that one does not consider possible interactions particularly valid for complex resource conflicts, e.g. forced assignments may lead to new conflicts due to relations between the activities which shall be allocated to the same time unit.

Rules will be defined to avoid this, but it is not suitable to introduce as extensive conditions as for the direct resource conflicts. They would be very complex, and regard for computing time forbids the use of them. All practical experience shows that the frequency is
very small for interactions between complex resource conflicts alone leading to neglect of important conflicts. (This is the case for the scheduling of schools, but it does not of course have to be valid for other scheduling). Maybe the most important thing is that one tries to define the allocation sequence such that interaction between complex resource conflicts alone does not arise. The reason why this interaction occurs is that certain groups of activities get forced allocations to certain time intervals within the total time frame. Provided that such conditions are valid for some activities, these will be allocated as quickly as possible.

The direct resource conflicts express via the conflict matrix explicitly existing conflicts, whereas complex resource conflicts (and other conditions) only implicitly define conflicts. By means of the rules for the allocation sequence implicit conflicts are tried to be made explicit (possibly eliminated) as quickly as possible, so that the conditions for the direct resource conflicts can be used with more validity. This is a strong argument for the importance of the allocation sequence, and for other problem types where complex resource conflicts are more dominating than for the scheduling problem for schools, these rules may be of vital importance.

A schedule may include a number of different complex resource conflicts. Room conflicts are the most important of these, and in the following they are used as an example of the treatment of complex resource conflicts. A discussion of the specification possibilities required by a school is necessary. A room resource requirement is defined by:

Room type: A set of rooms for which an unambiguous reference is needed.

The set of room types for which reference is necessary is represented by matrix \( R \), so that a row vector defines a room type. Assume that the system consists of \( R \text{MAX} \) rooms and that the number of different room types is \( \text{ATY} \). The dimension of \( R \) is \( [\text{ATY},R\text{MAX}] \).

\[
R_{ij} := \begin{cases} 
1, & \text{if room } j \text{ belongs to room type } i \\
0, & \text{otherwise}
\end{cases}
\]

Unfortunately, it can seldom be assumed that the different room types are linearly independent; i.e., it cannot be assumed that
$R_i \star R_j = 0$ for $i \neq j$. (If this were the case, treating room conflicts would have been simple.)

The information in $R$ will be transformed, and a matrix $AR$ is defined as follows:

$$AR_{ij} := \begin{cases} 1, & \text{if } R_i \in R_j \\ 0, & \text{otherwise} \end{cases}$$

$AR$ is called the **room type hierarchy**.

The room types may satisfy:

$$\begin{cases} (R_j \in R_i) \lor (R_j \in R_i) = 0 \\ R_i \in R_j \lor R_j \in R_i = 1 \end{cases}$$

i.e., $R_i$ and $R_j$ are neither linearly independent nor subsets of each other. (5.61) being satisfied, $R$ must be modified:

(5.62) If no room type $R_k = R_i \lor R_j$ is defined, then it is introduced as a **fictitious** room type.

(5.61) and (5.62) are iteratively defined. (5.61) being satisfied between fictitious room types, further room types might be introduced as a result of (5.62), etc. When all new room types have been generated, $AR$ is found from (5.60). The number of fictitious room types is called RFK, and quantity $\text{RTMAX} = \text{ATY} + \text{RFK}$ is the sum of all room types. $AR$ gets dimension $[\text{ATY, RTMAX}]$. An example shall be given:

The different room types are not linearly independent because one wants to specify alternative possibilities for room allocation for certain activities; i.e., a certain group of rooms is preferred, but if this is impossible, alternative rooms will be accepted. The various alternatives of a room type are ranked. Another factor is that each room (or room group) does not have the same relation to all activities, since some activities require a certain room explicitly, whereas for other activities the same room is equivalent to other rooms; i.e., typing usually requires a certain room, but the room is regarded as an ordinary classroom for those time units where it is not used for typing.
Assume that a school consists of the following room groups:

<table>
<thead>
<tr>
<th>Room group</th>
<th>Room No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: 5 classrooms</td>
<td>1-5</td>
</tr>
<tr>
<td>02: 1 geography room</td>
<td>6</td>
</tr>
<tr>
<td>03: 1 film room</td>
<td>7</td>
</tr>
<tr>
<td>04: 1 chemistry room</td>
<td>8</td>
</tr>
<tr>
<td>05: 1 physics room</td>
<td>9</td>
</tr>
<tr>
<td>06: 1 gymnasion boys</td>
<td>10</td>
</tr>
<tr>
<td>07: 1 gymnasion girls</td>
<td>11</td>
</tr>
<tr>
<td>08: 1 swimming pool</td>
<td>12</td>
</tr>
<tr>
<td>09: 1 typing room</td>
<td>13</td>
</tr>
<tr>
<td>10: 1 art room</td>
<td>14</td>
</tr>
<tr>
<td>11: 2 crafts rooms</td>
<td>15-16</td>
</tr>
<tr>
<td>12: 2 needlework rooms</td>
<td>17-18</td>
</tr>
</tbody>
</table>

RMAX=18

Fig. 5.11

Note the difference between room group and room type:

1. **Room group**: A set of identical rooms.

2. **Room type**: A room group and the set of room groups which from the definition of the room type are acceptable alternatives. (And the alternatives of the alternatives, etc.)

In those cases where a room group has no acceptable alternatives, room group and room type are synonymous concepts. Further, it appears from the definitions that all rooms of a room group belong to the same room types. (Next paragraph will discuss an exception, important in practice.) Based on the resource requirements, it is assumed that the room type references shown in figure 5.12 are needed.

Figure 5.12 are the room types given by a school, and from the definitions in [1], figure 5.12 is a result of the specification shown in figure 5.13.
### Room Survey

<table>
<thead>
<tr>
<th>Room type</th>
<th>Room groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01, 02, 03, 09, 05</td>
</tr>
<tr>
<td>02</td>
<td>02, 03</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
</tr>
<tr>
<td>06</td>
<td>06, 08</td>
</tr>
<tr>
<td>07</td>
<td>07, 08</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
</tr>
<tr>
<td>09</td>
<td>09</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>04, 05</td>
</tr>
<tr>
<td>14</td>
<td>11, 10</td>
</tr>
<tr>
<td>15</td>
<td>12, 10</td>
</tr>
<tr>
<td>16</td>
<td>10, 11, 12</td>
</tr>
</tbody>
</table>

**Fig. 5.12**

*ATY = 16*

### Room Survey

<table>
<thead>
<tr>
<th>ROOM TYPE</th>
<th>ROOM DESCR.</th>
<th>NAME</th>
<th>NO. ALTERNATIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>CL</td>
<td>01</td>
<td>05</td>
</tr>
<tr>
<td>02</td>
<td>SE</td>
<td>03</td>
<td>01, 02, 03</td>
</tr>
<tr>
<td>03</td>
<td>MOV</td>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>04</td>
<td>CHE</td>
<td>05</td>
<td>01</td>
</tr>
<tr>
<td>05</td>
<td>PHY</td>
<td>06</td>
<td>01</td>
</tr>
<tr>
<td>06</td>
<td>GB</td>
<td>07</td>
<td>01, 02</td>
</tr>
<tr>
<td>07</td>
<td>GG</td>
<td>08</td>
<td>01</td>
</tr>
<tr>
<td>08</td>
<td>SW</td>
<td>09</td>
<td>01, 02, 03</td>
</tr>
<tr>
<td>09</td>
<td>TYP</td>
<td>10</td>
<td>01, 02</td>
</tr>
<tr>
<td>10</td>
<td>ART</td>
<td>11</td>
<td>01, 02</td>
</tr>
<tr>
<td>11</td>
<td>WIK</td>
<td>12</td>
<td>01, 02</td>
</tr>
<tr>
<td>12</td>
<td>SEW</td>
<td>13</td>
<td>01, 02</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>14</td>
<td>00, 04, 05</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>15</td>
<td>00, 11, 10</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>16</td>
<td>00, 12, 10</td>
</tr>
</tbody>
</table>

**Fig. 5.13**
The rooms are numbered consecutively in figure 5.11, and R is found from figure 5.12, (5.61), and (5.62):

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

Note that ATY=16 whereas RTMAX=18. Two fictitious room types are defined. Room type 17 is found from room types 1 and 13, whereas room type 18 is a combination of room types 6 and 7. Note that for example, room types 14 and 15 satisfy (5.61), but not (5.62) due to room type 16. Based in figure 5.14 and (5.60) is found the room type hierarchy, AR - see figure 5.15.

AR contains the following information:

\[
AR_{ij} = \begin{cases} 
1, & \text{if an activity requiring room type } i \text{ blocks a room belonging to room type } j. \\
0, & \text{otherwise}
\end{cases}
\]

Assume that (5.61) is satisfied for \( R_i \) and \( R_j \). It is known that the sum of rooms belonging to \( R_i \) and/or \( R_j \) will be less than \( \Sigma (R_i \cap R_j) \). This is represented by introducing a fictitious room type \( R_k \) from (5.62), and if an activity requires \( R_i \) or \( R_j \) to be included, then a room belonging to room type \( R_k \) will also be included; in other words, any fictitious room type implies a new limitation on the set of activities which can be allocated to the same time unit.
An activity requiring a room of room type $i$, will block a room for a number of room types given by $AR_i$, which is represented by introducing a number of fictitious rooms so that each room is duplicated as many times as the number of room types to which the room belongs, e.g., the 5 classrooms in figure 5.11 are substituted by 10 fictitious rooms from $AR_1$, and the geography room is substituted by three fictitious rooms from $AR_2$. If an activity requires a classroom, one says instead that the activity shall have a fictitious room of room type 1 and a fictitious room of room type 17. AR defines the fictitious room requirements for the various physical room requirements. This point of view implies that the various room types may be considered linearly independent; i.e., if matrix $R'$ represents the fictitious rooms in the same way as $R$ represents the physical rooms, it may be concluded that $R'_i + R'_j = 0$ for $i$,$j$. The number of fictitious rooms belonging to the various room types is represented by the arithmetic vector $RT$.

The number of fictitious rooms used for the different time units is represented by the arithmetic matrix $RF$ as follows:
RF\textsubscript{ij} = the number of fictitious rooms of room type j used for time unit i.

The necessary and sufficient condition for all room conflicts to be consistent is:

\[(5.63) \quad RT\textsubscript{j} \geq RF\textsubscript{ij} \quad \text{for } i = 1,2 \quad \text{---TMAX}
\]

\[\text{and } j = 1,2 \quad \text{---RTMAX}\]

(5.63) would be an obvious condition when the defined room types are linearly independent, since the condition in this case is identical with (3.2).

Assume that p room types are pure subsets of each other, i.e.

\[(5.64) \quad R_1 \subset R_2 \subset R_3 \cdots \subset R_p \]

Any activity requiring room type \(R_i\) also requires a fictitious room of room type \(R_2, R_3 \cdots R_p\). Any activity requiring room type \(R_2\) also requires a fictitious room of room type \(R_3, R_4 \cdots R_p\), etc.

Assume that room type requirements to the various room types for time unit i are represented by vector \(RK\textsubscript{i}\). From that follows:

\[(5.65) \quad RF\textsubscript{ij} = \sum_{k=1}^{p} RK\textsubscript{ik} \cdot \delta\textsubscript{ik}\]

where

\[\delta\textsubscript{ik} := \begin{cases} 1, & \text{if } R_k \in R_i \\ 0, & \text{otherwise} \end{cases}\]

In other words, \(RF\textsubscript{i}\) is defined equivalent to vector \(3l\) from relation (5.44), and (5.63) is the necessary and sufficient condition for vectors satisfying (5.64) having maximum utility. In this case it can be shown that the concept maximum utility implies that all room type requirements can be satisfied. Further, it can be shown that the actual definition of fictitious room types makes it impossible to satisfy relations analogous to (5.51) when (5.63) is fulfilled. This will not be shown here, but it will be identical with estimating the utility of a TC where the freedom vectors have certain properties.

It is always possible to satisfy the room requirements defined by \(RK\textsubscript{i}\) if (5.63) is satisfied.

The above has an important consequence:
If for each step of the allocation process one provides for (5.63) being satisfied, then time allocation may be done independently of the room allocation; i.e., the room allocation may be evaluated globally after the time allocation has been done.

This is a main point of the strategy, and for any complex resource conflict found in practice, conditions may be formulated such that a global evaluation of these is done after terminating the time allocation. Thus unnecessary limitations of solution space are avoided, reduction of computing time is achieved, and qualitative requirements may be fulfilled in a better way; e.g., the number of room changes can be minimized for the various classes.

However, (5.63) requires much computing time and large storage space, and a data structure will be defined which simplifies this condition and in a simple way finds the new room conflicts arising from a possible allocation.

The number of primary rooms for the room types is represented by an arithmetic vector ROM. Primary rooms are defined as the number of rooms primarily belonging to a room type, and not included in it as an alternative. (column NO if figure 5.13 is the primary rooms for the various room types. Note that this figure may be 0.)

The number of fictitious rooms, AMAX, for a system is:

\[
AMAX_i = \sum_j (\sum_k ROM_{ij}) = \sum_i \sum_j R_{ij}
\]

In the example shown in figure 5.13 AMAX=51.

Assume that a vector q has dimension AMAX, and q is a common representation of RTMAX vectors \( q_1, q_2 \ldots q_{RTMAX} \), so that vector \( q_1 \) has dimension \( |R_1| \). For figure 5.14 vector q is indicated in figure 5.16, where the various bars show the vectors \( q_1 \ldots q_{RTMAX} \).

![Figure 5.16](image-url)
The representation in fig. 5.16 is very compressed, but far more important is that one can do operations with the various subvectors as if they were a unit. In addition to the vector \( q \) an arithmetic vector \( \text{ROTY} \) is formed, defining first and last components of the various subvectors \( q^i \).

\[
(5.67) \quad \text{ROTY}_i = \sum_{j=1}^{i} \sum_{k=1}^{\text{RMAX}} R_{jk}
\]

Accordingly:

First component to vector \( q^i \) is component \( \text{ROTY}_{i-1} + 1 \) in \( q \)

(\( \text{ROTY}_0 = 0 \)). Last component to \( q^i \) is component \( \text{ROTY}_i \) in \( q \).

For figure 5.16 \( \text{ROTY} \) has the following components:

\[9,11,12,13,14,16,18,19,20,21,23,25,27,30,33,38,48,51.\]

Two standard operations will be defined:

**Right justifying \( RJ \)**

This operation has an arbitrary number of integers as argument, each integer being an identifier of a room type. The same integer may occur as argument several times. Operation \( RJ \) performs the following:

For each room type component \( R_i \) being an argument, the corresponding room type hierarchy vector \( AR_i \) is allocated as right justified as possible in the corresponding subvector \( q^i \); i.e.,

If \( AR_{ij} = 1 \) and \( q_{\text{ROTY}_i} = 0 \) then \( q_{\text{ROTY}_i} := 1 \)

otherwise if \( AR_{ij} = 1 \) and \( q_{\text{ROTY}_i} = 1 \) and \( q_{\text{ROTY}_{i-1}} = 0 \) then \( q_{\text{ROTY}_{i-1}} := 1 \)

etc.

\( q \) is constructed gradually, each modification is superposed on the previous one. Formally may be stated:

\[
(5.68) \quad q := RJ(R_{i1}, R_{i2}^{--}, R_{iP})
\]

A few examples of this operation will be shown. \( R \) and \( AR \) are given by figures 5.14 and 5.15.
The operation $q := RJ(01, 01, 01, 05, 07)$ gives $q :=$

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

The operation $q := RJ(05, 10, 11, 12)$ gives $q :=$

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Fig. 5.17

Left Justifying $LJ$

The operation has two vectors as argument, both of which are subdivided analogously to fig. 5.16. The first vector is formed by the operation $RJ$ whereas the second one in constructed by successive $LJ$-operations. The operation is written formally as:

\[(5.59) \quad LJ(q, ql)\]

and it performs the following:

Each component in $q$ with value 1 (and right justified) is allocated as left justified as possible in corresponding subvector $rqi$ for vector $ql$; i.e.:

For each value $rqi = 1$ is done:

If $ql_{ROTY(i-1)+1} = 0$, then $ql_{ROTY(i-1)+1} := 1$;

otherwise if $ql_{ROTY(i-1)+2} = 0$, then $ql_{ROTY(i-1)+2} := 1$

e tc.

An example of operation $LJ(q, ql)$ will be shown.

Assume that $q$ and $ql$ have the following form:
The operation \( \text{LJ}(q, q_1) \) then gives \( q_1 = \)

\[
\begin{array}{cccccccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Operations \( \text{RJ} \) and \( \text{LJ} \) are apparently complicatedly defined, but by
means of the NEXT-operations fast algorithms are easily made.

Assume that activity \( k \) requires that a room of room types \( R_{k1}, R_{k2}, \ldots, R_{kp} \) are included in the activity. This set of room types is
called \( \mathcal{R}_{k} \).

If the following operation is performed

\[
\mathcal{F}_k = \text{RJ}(\mathcal{R}_k)
\]

then \( \mathcal{F}_k \) will define the number of fictitious rooms required by
activity \( k \) from the various room types. Vector \( \mathcal{F}_k \) is called the
room resource requirement for activity \( k \).

\( \mathcal{F}_k \) will of course be constant during the whole allocation process,
and all room resource requirements are generated initially and
represented in a matrix \( \mathcal{F} \). Realistic problems always have the
property that a number of different activities present the same room
resource requirements, accordingly a certain compression of the
data structure may be done:

Matrix FA is a representation of the different room resource require-
ments defined by the activities of the system. An arithmetic vector
SP₄ states which room resource requirements correspond to each acti-
vity. Preferably one should also have a data structure defining
which activities represent the same room resource requirements. This
is represented by matrix FA₁ so that FA₁ₓ are the activities with
room resource requirements FAₓ. (FA₁ is usually stored in secondary
storage since it is seldom referred to.)

Figure 5.19 outlines the connection between SP₄, FA, and FA₁.

(NA = number of different room resource requirements.)
Each allocated activity is represented in the matrix FFT. Correspondingly, a matrix TA is defined representing room blockings resulting from the allocations. Each row vector in TA is subdivided in the same way as a row vector in FA, and the dimension of TA is $[\text{TMAX}, \text{AMAX}]$.

If activity $k$ is allocated to time unit $j$, operation $\text{LJ}(\text{FA}_{SP_k}, \text{TA}_j)$ is performed, and TA is a representation of the fictitious rooms blocked for each step of the allocation.

The general expression for $\text{TA}_j^S$ is:

$$\text{(5.70)} \quad \text{TA}_j^S := \text{LJ}(\text{FA}_{SP_k}, \text{TA}_j^0)$$

for all $i \in \text{FFT}_j^S$, and $\text{TA}_j^0$ is the initial value of $\text{TA}_j$.

One wants to know:

When considering only room conflicts, what is the condition for being able to allocate activity $k$ to time unit $j$?

Based on the given data structure a very simple expression results:

$$\text{(5.71)} \quad \text{FA}_{SP_k} \ A \ \text{TA}_j \equiv 0$$

All activities $k$ satisfying (5.71) may, when considering only room conflicts, be allocated to time unit $j$, since (5.63) will still be satisfied. In other words, (5.71) and (5.63) are identical conditions.

The simple form of (5.71) is the motivation for the chosen data structure TA and TA; i.e., that it is simple to determine whether (5.63) is satisfied. This condition could of course be evaluated by means of arithmetic matrices. Some more storage space would be necessary, but the main objection is that a condition as simple as (5.71) cannot be made.

Operation RJ is used only for making the initial data structure, and operation LJ is seldom done, whereas a large number of operations based on (5.71) are necessary.

New complex resource conflicts $\text{RR}_j(\text{FFT}_j^S)$ will be determined. It is known:
(5.72) \[ T_{i,j}^{s+1} = L_{ij}(F_{i}^{s}P_{i,k}^{s} + T_{i,j}^{s}) \]

if activity \( k \) is allocated after \( s \) steps.

Room resource requirements which can no longer be satisfied for time unit \( j \) are called \( t_{j}^{i} \).

(5.73) \[ t_{j}^{i} = FA \cdot (TA_{j}^{s+1} + TA_{j}^{s}) \]

New room conflicts:

(5.74) \[ RR(FTT_{j}^{s}) := LEDIG_{j} \land \left( \bigvee_{i \in t_{j}^{i}} FA_{i} \right) \cup KOLMA_{i} \]

(In (5.74) \( KOLMA_{i} \) eliminates new blockings which may also be due to direct resource conflicts.)

If conflict vector \( TAPT_{j} \) is defined as in (5.58) and (5.74), this is the necessary and sufficient condition for (5.63), i.e. (3.2), being satisfied for each step in the allocation. This means that each activity available for a time unit may be allocated to this when considering only room conflicts; however, these conflicts may be the result in not all activities currently available for the time unit in question can be allocated to it.

More extensive conditions have been defined for direct than for complex resource conflicts. From (5.74) may arise new "unexpected " blockings, which may have unpleasant consequences for forced assignments. It is therefore highly desirable for complex resource conflicts to be transformed to direct resource conflicts, which is possible in some cases.

If for the two room resource requirements \( FA_{k} \) and \( FA_{j} \) there are corresponding subvectors \( r_{i,k}^{i} \) and \( r_{i,j}^{i} \) satisfying the condition:

(5.75) \[ \Sigma_{i} r_{i,k}^{i} + \Sigma_{i} r_{i,j}^{i} > ROT_{i} + ROT_{i-1} \]

it may be concluded that all activities in \( FA_{k} \) are in conflict with all activities in \( FA_{j} \). If particularly \( k = j \), it may be concluded that \( FA_{k} \) forms a direct resource conflict.

Defining an algorithm for (5.75) is simple, and this modification of \( KOLMA \) is done initially.
5.2.2. Practical modifications

The reason why fictitious room types must be introduced is easily shown. Consider the room types defined in figure 5.12. Assume that two activities not in direct conflict both require a room of type 06 and a room of type 07 to be included in the activity. From subvectors $r_6^q$ and $r_7^q$ it is apparently possible to use two rooms of each of these room types. What does not appear from $r_6^q$ and $r_7^q$ is that room type 08 is an alternative to both room types 06 and 07. Together these two room types consist of three physical rooms. This information is represented by room type 18, consisting of three fictitious rooms. When allocating activities which consist of room types 06 and/or 07 a corresponding number of rooms of type 18 are blocked; i.e., maximum three room resource requirements of types 06 and 07 to the same time unit are allowed.

For the room type definitions necessary for a streamed school, it has been perfectly acceptable to evaluate all consequences of (5.61) and (5.62). The number of fictitious rooms $\text{AMAX}$ is usually less than $2 \cdot \text{RMAX}$. For university structures with variably sized rooms, connections between different room types can be complex and the number of fictitious room types large; accordingly, $\text{AMAX}$ may become a factor 10-20 times larger than $\text{RMAX}$. This implies increases in computing time and storage space, and a sensible compromise would be to disregard interactions between fictitious room types possible from (5.61) and (5.62). The practical consequence of this is in short: connections between room groups not closely "related" are disregarded; i.e., the room groups which are usually used in different connections, and the probability for the type of error allocation mentioned above is small.

Till now it has been presumed that all rooms of a room group belong to the room types where the room group is included. There is one important exception: A room type (ordinary class rooms) is used as a kind of "buffer" for certain special rooms; i.e., a classroom may be accepted as an alternative, but only a certain number of classrooms per time unit. The motivation for this is to secure large utility of the special room and avoid too strong a concentration of activities using the special room for the same time units. (Assume for example that each student group should actually use a certain special room
for two time units and that this room is overloaded. By accepting one classroom as an alternative, the room allocation can always be done such that each student group gets at least one hour of education in the special room, whereas this is not necessarily the case if several classrooms are used as alternatives for the same time unit.)

The condition mentioned is represented as follows: For the special rooms accepting a certain number of classrooms as alternatives a corresponding number of fictitious rooms are introduced in the corresponding subvector in $\mathbf{FA}$. If the special room itself is an alternative classroom nothing more has to be done, and the mentioned conditions are still correct. If the special room is not used as a classroom, the classroom alternatives are represented in a vector $\mathbf{Wl3}$:

$$\mathbf{Wl3}_i = \begin{cases} 1, & \text{if the alternative room is a classroom and the special room is no alternative for a classroom} \\ 0, & \text{otherwise} \end{cases}$$

Operation $LJ$ is modified as follows:

1. If the last classroom is blocked, a room for all room types with a 1-component in $\mathbf{Wl3}$ is blocked.

2. If the corresponding component in $\mathbf{Wl3}$ has value 1 for a blocked room, a classroom and a room of the room types where the classrooms form a unity are also blocked.

When using classrooms as alternatives in this way, another special rule for room type definitions is also employed: A classroom must be defined explicitly as an alternative to a room type and not indirectly via another alternative.

It is possible to construct situations where the above modifications may lead to wrong allocations, but this has not happened in practice, and there is always the possibility of manual adjustment.

As a matter of form, another objection against the principles on which (5.74) is based, shall be mentioned. These principles were going to secure that the room resources were sufficient for any time unit. Double- and triple periods will also require allocation to the same physical room. This requirement might lead to the room resources for a time unit being insufficient. However, this possibility is rather hypothetical, and for the examined school structures not even theoretically possible.
5.2.3 Common features of complex resource conflicts

Room conflicts play an important role in scheduling. Below are some points of more general value:

1. Direct and complex resource conflicts are treated equally. This is done by defining necessary and sufficient conditions to make the complex resource conflicts consistent for each time unit, and these conditions may modify the conflict vector for each step of the allocation.

2. The total conflict picture is evaluated from the relations valid for direct resource conflicts. Therefore complex resource conflicts should, if possible, be transformed to direct resource conflicts. Global conditions for complex resource conflicts are difficult to handle, as it implies conditions in "two dimensions": time and resource type.

3. Point 1 indicates an approximate way of treating "two-dimensional" conditions, and a very important result is that the evaluation of the complex resource requirements may be done globally when all activities have been allocated.

4. The data structure defined by FA and TA may have a general value: The possibilities for specification of room types may be transformed so that any room resource requirement is done by reference to a list structure. (If an activity requires a certain room this is defined by a certain sublist, whereas a more loosely formulated requirement is defined by referring to a list on a higher level.) This possibility for definition of complex resource requirements is also characteristic for other problem areas. Data structures TA and FA and their resulting conditions may be useful for resource requirements defined by list structures.

5.3 Variable period length (The period requirement)

The condition is formulated verbally as:

\[(5.76) \text{ An activity must be allocated to } p \text{ contiguous time units in a day interval where } p \text{ is the period length of the activity.}\]

This is the one main type of an interval conflict where it is prescribed which sets of activities shall be allocated to the same day
interval. This condition will be adjusted to the "standard pattern" for the conflict types: The consequences of (5.76) are transferred to the conflict picture, and the condition modifies global conditions for direct resource conflicts. In this case it is more meaningful to say conflict picture than conflict vector, since (5.76) leads to new conflicts for time units close to those where an allocation is done.

5.3.1 Representation of the period requirement

The period division of the activities can be represented by means of a matrix, but for certain reasons another representation is better:

Any externally defined activity with variable period length is represented internally by a number of activities equal to the number of different period lengths.

This means that the number of activities to be allocated (internal activities) will be larger than the number of specified activities. This increase of storage space is justifiable, since any (internal) activity has equal period length for all periods, which is highly advantageous. The period length is represented by an arithmetic vector PL. Usually the period length is 1, 2, or 3, and only this will be discussed here. An externally defined activity can in other words have a maximum representation of three internal activities. (It occurs that activities have period lengths larger than 3, but so seldom, that one prefers to pre-assign these. If it appears that a school structure consists of many activities with large period length, it is simple to expand the principles mentioned here.)

The vectors $W_3$, $W_4$, $W_5$, and $W_6$ are defined as:

\[
\begin{align*}
W_{3,j} &= \begin{cases} 
1, & \text{if activity } j \text{ is not allocated} \\
0, & \text{otherwise}
\end{cases} \\
W_{4,j} &= \begin{cases} 
1, & \text{if } W_{3,j} = 1 \text{ and period length of act. } j \text{ is 1} \\
0, & \text{otherwise}
\end{cases} \\
W_{5,j} &= \begin{cases} 
1, & \text{if } W_{3,j} = 1 \text{ and period length of act. } j \text{ is 2} \\
0, & \text{otherwise}
\end{cases} \\
W_{6,j} &= \begin{cases} 
1, & \text{if } W_{3,j} = 1 \text{ and period length of act. } j \text{ is 3} \\
0, & \text{otherwise}
\end{cases}
\]
W4, W5, and W6 are linearly independent and W4 V W5 V W6 = W3.

Freedom matrix LEDIG defines to which time units the various activities can be allocated. Due to variable period length this concept is ambiguous. Assume that activity j has period length 3 and that it can be allocated to time unit i. This time unit may be part of three different triple periods; i.e., the triple hours starting respectively in time units i-2, i-1, or i. Which of these periods can be used by activity j, does not appear from LEDIG^T_jj alone. (It will be shown that this cannot be found from LEDIG^T_jj either.) To keep track of which combinations of hours can be used for double- and triple periods, a matrix LEDPER is introduced. A mention of the notation for the time units is necessary.

The time units are not numbered consecutively from how they succeed in time, but: First hour on the first day of the week is called time unit 1, first hour on the second day of the week is called time unit 2. If the week consists of UD days, then time unit UD+1 symbolizes the second hour on the first day. Double- and triple periods are numbered analogously: Combination of the two first hours on first day is called double period 1, two first hours on day 2 double period 2. Time units UD+1 and 2UD+1 are called double period UD+1, etc. Triple periods are numbered correspondingly.

The time frame of a schedule is characterized by two parameters:

UD = the number of days

DT = the number of hours per day. This figure does not have to be constant for all days. (Usually DT is constant, and in all cases a maximum DT can be given, and eventually all activities can be blocked for certain time units.)

From the given notation day D for period j will be found. The period length is called TYPE, and one wants to find which time units on day D are the first and last belonging to period j. These are called respectively T1 and T2. One finds:

$$D = \text{MOD}(j-1, \text{UD})+1$$

$$T1 = (j-1)/\text{UD}+1$$

$$T2 = T1+(\text{TYPE}-1)$$

Note that D and T1 are independent of DT and TYPE. This has some technical advantages, which is the motivation for the chosen
notation. This has consequences for some computing rules, and figures 5.20 and 5.21 are examples of the notation. In figure 5.20 UD=6 and DT=8, and in figure 5.21 UD=5 and DT=9. First column for each day states which single period each time unit is included in. Second column states which double periods the time unit is included in, and third column the triple periods.

The physical time units for a period j with period length TY1, are represented by a vector q^j. Generally, one wants to find which set of periods with period length TY2, where at least one of the time units in q^j is included in the physical representation of the period. This set of periods is called A, and period i of the set is represented by a vector a^i. In other words, one wants to find the set A which satisfies:

(5.79) \( q^j \cdot a^i = 1 \) for given j, TY1, and TY2.

From figures 5.20 or 5.21 is found:

\[
\begin{array}{ccccccc}
TY1 & TY2 & A_1 & A_2 & A_3 & A_4 & A_5 \\
1 & 1 & 1 & & & & \\
1 & 2 & 1-UD & 1 & & & \\
1 & 3 & 1-2UD & 1-UD & 1 & & \\
2 & 1 & 1 & 1+UD & & & \\
2 & 2 & 1-UD & 1 & 1+UD & & \\
2 & 3 & 1-2UD & 1-UD & 1 & 1+UD & \\
3 & 1 & 1 & 1+UD & 1+2UD & & \\
3 & 2 & 1-UD & 1 & 1+UD & 1+2UD & \\
3 & 3 & 1-2UD & 1-UD & 1 & 1+UD & 1+2UD \\
\end{array}
\]

(5.80)

if A_i for given j satisfies one of the conditions:

\[ A_i < 0 \]
\[ A_i > UD \cdot DT - UD \cdot (TY2-1), \]

then this A_i does not exist.

(5.80) is used to define standard operation KONV:

\[ KONV(TY1, TY2, j, A) \]: This operation has period j of length TY1 as an argument, and one wants to find the set of periods with length TY2 satisfying (5.79). This is represented by an arithmetic
vector A. (The dimension of A must be at least 6, when maximum period length is 3.) The components in A are generated from (5.80), and significant components are left-justified in A.

Operation KONV occurs frequently, and for the scheduling problem it is a standard operation, just like logical operations. Operation KONV may be regarded as a coordinate transformation of the time units: From a set of physical time units one can easily find the set of single-, double-, or triple periods connected with them. Further, it is easy to transform a set of periods with a given length to the set of periods with another length connected with it. This means that when allocations or blockings are defined by one coordinate representation of the time units, it is simple to transform the time units to a new representation to find other consequences.

A new data structure, LEDPER, is defined:

\[
\text{LEDPER}_{ij}^T = \begin{cases} 
1, & \text{if activity } i \text{ can be allocated to period } j \\
0, & \text{otherwise}
\end{cases}
\]

\( LEDG_j^T \) represents the physical time units which activity \( j \) can use, whereas \( LEDPER_j^T \) represents the set of periods available for activity \( j \). \( LEDPER_j^T \) defines the allocation possibilities for activity \( j \).

The relation between the two vectors will be shown:

For all activities with period length 1, \( LEDPER_j^T \equiv LEDG_j^T \). Generally may be said: For each \( i \in LEDPER_j^T \) transformation \( KONV(PL_j^T, 1, i, A) \) is done. The components in the successive As are superposed in a vector \( q2 \). The final \( q2 \) will then be identical with \( LEDG_j^T \). This can be done as below:

```
TYPE:=VERDI(PL,j);
LOADV(LEDPER,j,q1);
REGCLR(TMAX,q2);
i:=0;
L1:i=NEXTB(q1,i,TMAX,L2);
KONV(TYPE,1,i,A);
    for k:=1 step 1 until TYPE do SETB(q2,A(k));
goto L1;
L2:
```

At exit from the loop \( q2 \equiv LEDG_j^T \);
The algorithm defines \( \text{LEDIG}^T_j = f(\text{LEDPER}^T_j) \), and one may want to determine \( \text{LEDPER}^T_j = g(\text{LEDIG}^T_j) \). This operation may be ambiguous; e.g., assume that double and triple periods of an activity cannot be allocated over the lunch break; i.e., certain combinations of time units cannot be used for some activities. By themselves these time units can be used for the activities, and the condition has no consequence for LEDIG. The mentioned condition is externally defined, and is represented by suitable blockings in LEDPER. In order to make the transformation \( \text{LEDPER}^T_j \leftrightarrow g(\text{LEDIG}^T_j) \), such special requirements must be known, and the operation cannot be given a general form.

However, it is simple to find new modifications for LEDPER as a result of a modification of LEDIG; in other words,

\[
\text{LEDPER}^{S+1} = g_1(\text{LEDIG}^{S+1}, \text{LEDPER}^S)
\]

Important points of the outlined data structure will be summarized:

1. By splitting an external activity into a number of internal activities a certain increase in storage space occurs. This is fully compensated due to the information in LEDIG and LEDPER being completely represented by these matrices. If each internal activity could have variable period length, the corresponding information would require considerably more space.

2. LEDIG represents time units available for the activities. This is the relevant information to estimate the utility for the TCs of the system. LEDPER accounts for the allocation possibilities for each activity; it also represents conditions particular for variable period length, but without consequences for LEDIG.

3. The coordinate transformation KONV makes it simple to transfer consequences from LEDIG to LEDPER and vice versa.

5.3.2 Necessary conditions

The consequences of (5.76) for the conflict picture will be evaluated. (5.76) alone only has consequences for the time units belonging to the actual day of the allocation. Due to this property an auxiliary matrix \( \text{LEDIG}^d \) is introduced; it is a selection of LEDIG so that \( \text{LEDIG}^d \) only represents the time units in LEDIG belonging to the same day, \( d \). Formally is written:
(5.8c) \[ \text{LEDIG}^d = \sum_{i=1}^{\text{LEDIG}_i} (\text{MOD}(n-1, 4k)+1) \]

Data structure \(\text{LEDIG}^d\) does not get the same dimension as LEDIG, but is compressed so that \(\text{LEDIG}^d\) gets dimension \([\text{DT}, N]\). This is indicated in figure 5.22, where \(\text{UD}=3\), \(\text{DT}=6\), and \(d=2\).

The broken lines show which vectors of \(\text{LEDIG}\) are represented by \(\text{LEDIG}^d\) for \(d=2\).

Fig. 5.22

\(\text{LEDIG}^d\) is called the freedom picture for day \(d\). One wants to know the modification of \(\text{LEDIG}^d\) due to \(\text{TAPT}_j\), where \(j\) symbols one time unit in \(\text{LEDIG}^d\). \(\text{TAPT}_j\) is found from (5.58) or a more complex relation.

In the same way as (5.74) modifies conflict vector due to room conflicts, (5.83) modifies the conflict picture due to variable period length. If the actual activity has period length larger than 1, one gets a superposition of the activities resulting from (5.33). (5.83) is technically inconvenient, and a certain optimization is done when designing an algorithm.
1. Activities with period length = 1
New conflicts arise for: \( q_1 := T_A T^{j - T_A W_4} \). None of these activities can be blocked for other time units than \( j \). For other time units only \( T_A T^{j - T_A W_4} \) is interesting.

2. Activities with period length = 2
New conflicts arise for:
\( q_2 := T_A T^{j}_j \wedge W_5 \)

\( q_2 \) is blocked for time unit \( j \), and in addition these activities can be blocked for the time units:
\( j + \delta \quad \text{where} \quad \delta \neq 1 \)
The activities which can be blocked for one of the time units \( j + \delta \) are:
\( q_2' := q_2 \wedge LEDIG^{d}_j + \delta \)

If an activity in \( q_2' \) is still available for time unit \( j + 2 \delta \), it is not blocked in \( LEDIG^{d}_{j + 2 \delta} \), since combination of time units \( j + \delta \) and \( j + 2 \delta \) is still acceptable for period length 2. Activities \( q_2'' \) being blocked for time units \( j + \delta \) are:
\( q_2'' := q_2' + LEDIG^{d}_{j + 2 \delta} \)

\( (T_A T^{j}_j \wedge W_5) \wedge LEDIG^{d}_{j + \delta} \) + LEDIG^{d}_{j + 2 \delta} \)

(5.83)

3. Activities with period length = 3
In an analogous way is found:

a. Activities \( q_3'' \) being blocked for time units \( j + \delta \) are:
\( q_3'' := ((T_A T^{j}_j \wedge W_6) \wedge LEDIG^{d}_{j + \delta}) + (LEDIG^{d}_{j + 2 \delta} \wedge LEDIG^{d}_{j + 3 \delta}) \)

b. Activities \( q_3''' \) being blocked for time units \( j + 2 \delta \) are:
\( q_3''' := (q_3'' \wedge LEDIG^{d}_{j + 2 \delta}) + (LEDIG^{d}_{j + 3 \delta} \wedge LEDIG^{d}_{j + 4 \delta}) \)

(It is simple to generalize the relations forming the basis for definition of \( q_2'' \), \( q_3'' \), and \( q_3''' \).)

4. If one of the indexes in the expressions above is not within the frame of \( LEDIG^{d} \), i.e.,
\( j + y \cdot \delta > DT \) or \( j + y \cdot \delta < 1 \) (\( y \)-positive integer),
the corresponding vector is made equal to the 0-vector.
(5.83) verifies the assertion that the probability for (5.46) being correct is large. Any conflict vector arising for time units other than those where an allocation is done, will be a subset of the conflicts there according to (5.83). Since the conflict vectors for the various time units are defined from subset relations and not stochastic processes, the probability for (5.46) being correct is increased. From the day conflicts discussed in the next section it becomes very probable that time units belonging to the same vector in set \( h^1, h^2 \ldots h^p \) (see relation (5.19)) also belong to the same day, and the above arguments get still more weight.

(5.76) may have consequences for utility \( G_i = G(LEDG, KOMB_i) \). Situations may be constructed where interactions between variable period length and resource conflicts lead to calculation of \( G_i \) from relation (5.47) being wrong, and that condition must be modified. On account of computing time it cannot be done completely. For most school structures the majority of activities have equal period length (for streamed structures period length is usually 1, whereas university structures usually have period length 2). This factor implies that the period requirements do not have large consequences for the utility. The most important consequences can be considered by simple means.

By introducing variable period length by means of vector \( PL_i \), the number of hours (TT) for a TC defined by (5.12) must be modified. Furthermore, the requirement that no activity may have more than one period on the same day, makes it convenient to split the operator TT, in two new operators TT and TD:

\[
\begin{align*}
\text{a. } TT(b) &= \sum_{i \in b} SF_i \cdot PL_i \\
\text{b. } TD(b) &= \sum_{i \in b} PL_i
\end{align*}
\]

TT defines the number of unallocated time units in \( b \), whereas TD defines the maximum number of time units in \( b \) which may be allocated to a day interval.

Utility \( g_j \) for an interval \( h^j \) is given by (5.17), but due to variable period length that relation must be modified:
\[(5.85)^* \quad g_j = \text{MIN}(|h^3_j|, (\text{MIN}(|h^3_j|, |c^3_jW4|) + \text{MIN}(|h^3_j|/2, |c^3_jW5|) \cdot 2
+ \text{MIN}(|h^3_j|/3, |c^3_jW6|) \cdot 3)) \]

where \(c^3_j\) is given by \((5.16)\).

\((5.85)\) is a more correct expression than \((5.17)\), since one considers only the number of time units being an integer multiple of the periods; i.e., \(|h^3_j| = 5\) and \(c^3_j\) consists of two activities with period length 3. \((5.17)\) gives \(g_j = 5\), whereas \((5.85)\) gives \(g_j = 3\) which is of course the correct value.

Relation \((5.20)\) can now be evaluated by means of \((5.85)\). This may result in \(G_j\) from \((5.47)\) being wrong, since the time sequence of the time units must necessarily be considered. Further, sufficiently small time intervals will imply that activities with period length larger than 1 cannot be allocated to those. This is of course an artificial limitation, and condition \((5.15)\) for vectors \(h^1_j, h^2_j, \ldots, h^P_j\) must be modified. The time sequence must be considered, which gives complicated relations between the vectors. No attempt will be made to define those conditions, since they are prohibited by consideration of computing time.

It will later be shown that all time units belonging to an \(h^3_j\) essentially belong to the same day, and to compensate for variable period length, \(G_j\) is estimated in two ways:

1. Utility for a vector \(h^3_j\) is determined by means of \((5.17), (5.16), \text{and } (5.84.a)\), and the principles discussed in section 5.1.3. The time sequence of the time units is considered to a certain degree, but situations may be designed where this gives errors. It seldom occurs in practice, and the method evaluates the interaction between the time units.

2. Utility for the time units where a TC is available on a day is determined by means of \((5.85)\). In this way the time sequence for the period requirements is considered, but interaction between time units within the day is partly neglected.

\*(5.85) may be improved by substituting \(|h^3_j|\) in the three terms with the number of time units actually available for single-, double-, and triple periods, but this is a time consuming operation.
One uses the smallest of these values, which is usually found by the first mentioned principle when the majority of activities in a TC have period length 1. The TCs with several activities of larger period length get a more correct expression for the utility by means of the second principle.

In practice is found the favourable property that the period length of the activities in a TC is approximately constant. This is simply explained, since the activities in a TC represent approximately the same kind of subjects, and the probability of these having the same period length is high. The allocation sequence will imply early allocation of double- and triple periods, and after a number of steps (5.47) is a very good approximation for the utility. For certain groups of TCs with activities of period length 2 or 3, the consequences of (5.85) act as a stabilizing factor for the estimate of utility for the first part of the allocation. (Important subject groups are ability subjects such as crafts and cooking.)

One writes:

$$h^{d} = \text{LEDIG}^{d} \cdot \text{KOMB}_{1}$$

An approximate expression for $G_{1}$ is:

$$G_{1} = \sum_{d=1}^{U} g_{d}$$

where $g_{d}$ is given from (5.86) and (5.85)

(5.87) has proved to be a good approximation for $G_{1}$, and one avoids the extensive computing operations which may be necessary when $G_{1}$ is determined by (5.47). ((5.87) also considers day conflicts.)

In practice a combination of (5.47) and (5.87) is used:

(5.87)$^{\text{K}}$ is used to determine whether a TC (or SC) is tight. When this is satisfied, the more time-consuming relation (5.47) is used for further calculations of $G_{1}$ for TTC, KOMB$_{1}$.

$G_{1}$ is not changed for all KOMB$_{1}$ for each step in the allocation, and to avoid unnecessary operations some new data structures are defined: TTR, SP3, and SP10.

$^{\text{K}}$ In a simple way (5.87) can be modified to give an even better approximation for $G_{1}$. 

Such an extravagance cannot usually be defended. The mentioned "two-stage process" really implies dislocation of the difficulties, instead of solving the most critical conflicts first. Certain manual methods with approximate day by day scheduling clearly show the weakness of this. For the first days a schedule is easily found, whereas all difficulties are dislocated to the last days, which is a serious mistake.

Accordingly, day- and hour conflicts must be treated simultaneously. However, for practical structures it is possible to make considerable simplifications due to the form of the conflict picture. The conflict types hitherto discussed have primarily had consequences for the time units of the actual day where an allocation is done. When introducing more conflict types this becomes still more distinct. The number of day-conflicts is small compared to the hour-conflicts, and the day-conflicts are approximately independent (i.e., interactions between these rarely result in day-TCs of a higher order).

The practical consequence of these factors is:

The number of new conflicts for the actual day of an allocation is of a quite different magnitude than the number of new conflicts for other days, and the probability of other days still having the same utility for the various TCs is very high. The interactions between the days are usually determined by day conflicts, and when these are independent, simple conditions may be employed to determine whether the day conflicts are solvable.

A natural heuristic principle is:

1. The interaction between new conflicts for time units of the actual day of an allocation is examined by means of an extensive set of conditions.

2. The interactions between the days are examined for a simplified set of conditions. Further consequences of these conditions are superficially examined, but it must of course be found out whether new blockings or forced allocations are consistent with the pre-history.

This is a marked difference from a separate treatment of the two conflict types. The day allocation will now be a result of the prehistory of each time unit; thus the artificial limitation of solution
space mentioned above is avoided. (A common feature of both methods is that a somewhat superficial treatment of the day conflicts is necessary or useful.)

The day conflicts are divided into two groups:

1. **External day conflicts**: These are given from the problem definition, and are for example due to different activities representing the same subject for certain student groups, or absolute requirements regarding uneven work load for certain resources (usually the student groups).

2. **Internal day conflicts**: These are a consequence of the data structure discussed in section 5.3.1, where an externally defined activity could be divided into several internal activities. The condition that no external activity can be taught for more than one period in one day is represented as a day conflict between internal activities. (An implicit rule is of course that an internal activity is in day conflict with itself.) This form of day conflict is the most common one, and the two groups of day conflicts can be treated identically.

5.4.1 Representation of day conflicts

It has been pointed out that this conflict type can be represented in the same way as a direct resource conflict. In order to make this necessary, the day conflicts must be of a complex nature, and the data structure to be defined assumes certain simplifications to be acceptable. The following pages show tables of definitions of some important data areas. Not all of these are necessary for the treatment of day conflicts alone. Rather, they are a consequence of the division of the time frame into days, and it seems natural to define these data areas together.

The definitions assume that activity k is allocated to period ti on day d after s steps in the allocation.
<table>
<thead>
<tr>
<th>Notation for data area</th>
<th>Data type</th>
<th>Dimension</th>
<th>DEFINITION</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>Logical vector</td>
<td>N</td>
<td>( W_{ij} = \begin{cases} 1, &amp; \text{if activity } j \text{ is included in a day conflict} \ 0, &amp; \text{otherwise} \end{cases} )</td>
<td></td>
</tr>
<tr>
<td>DABL</td>
<td>Logical matrix</td>
<td>DMA,N</td>
<td>DABL is a representation of the day-TCs (DTC) of the system just like a regular TC is defined by KOMB</td>
<td></td>
</tr>
<tr>
<td>SPS</td>
<td>Arithm. matrix</td>
<td>N,DK</td>
<td>( i \in SPS_j \text{ if } i \in DABL_j )</td>
<td>SP5 is an alternative representation of DABL since DABL is often stored in secondary storage. The components in SP5 are left justified, and DK must be larger than the maximum number of day conflicts in which an activity can be included.</td>
</tr>
<tr>
<td>DDF</td>
<td>Logical matrix</td>
<td>UD,N</td>
<td>( DDF_{ij} = \begin{cases} 1, &amp; \text{if activity } j \text{ can be allocated to day } i \ 0, &amp; \text{otherwise} \end{cases} )</td>
<td>Equivalent with LEDIG for hour conflicts.</td>
</tr>
<tr>
<td>DDFV</td>
<td>Arithm. matrix</td>
<td>DMA,2</td>
<td>( DDFV_{i1} = \text{TT}(DABL_i) ) ( DDFV_{i2} =</td>
<td>DDF &amp; DABL_i</td>
</tr>
<tr>
<td>Notation</td>
<td>Data type</td>
<td>Dimension</td>
<td>Definition</td>
<td>Comment</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>W7</td>
<td>Logical vector</td>
<td>DMA</td>
<td>$W_{7i} = \begin{cases} 1, &amp; \text{if }</td>
<td>DABL_{ij}</td>
</tr>
<tr>
<td>DTAP</td>
<td>Logical matrix</td>
<td>UD,N</td>
<td>$DTAP_j = \bigcup_{k = 1}^{m} DDF_k \subseteq DABL_j$</td>
<td>Analogous to TAPT for hour conflicts; i.e. DTAP is the day conflict picture.</td>
</tr>
<tr>
<td>W11</td>
<td>Logical matrix</td>
<td>UD,N</td>
<td>$W_{11ij} = \begin{cases} 1, &amp; \text{if a period of activity } j \text{ has got forced allocation to day } i. \ 0, &amp; \text{otherwise} \end{cases}$</td>
<td>W11 represents, among other things, the consequences of (4.21). W11 can also be modified by means of the day conflicts alone: If $SP_j \geq</td>
</tr>
<tr>
<td>W8</td>
<td>Logical vector</td>
<td>N</td>
<td>$W_{8i} = \bigvee_{j=1}^{n} W_{11ij}$</td>
<td>W8 is a joint representation of the activities with forced allocation to certain days.</td>
</tr>
<tr>
<td>W12</td>
<td>Logical vector</td>
<td>N</td>
<td>$W_{12j} = \begin{cases} 1, &amp; \text{if activity } j \text{ is included in at least one TDTC (BDTC)} \ 0, &amp; \text{otherwise} \end{cases}$</td>
<td>It would have been natural to define W12 analogously to W8; i.e. that the vector indicates directly the tight (broken) day conflicts, but since DABL is stored in secondary storage, the chosen representation is more suitable.</td>
</tr>
<tr>
<td>DTV</td>
<td>Logical matrix</td>
<td>UD,N</td>
<td>$W_{11s+1} = W_{11s} \cup DTVs$ (or $DTV_s := W_{11s} + 1 + W_{11s}$)</td>
<td>DTV defines new activities with forced allocation to certain days due to the current allocation.</td>
</tr>
<tr>
<td>SP2</td>
<td>Arith. matrix</td>
<td>N, UD</td>
<td>$SP_{2jd} =</td>
<td>LEDPER_{2jd}</td>
</tr>
<tr>
<td>SP12</td>
<td>Arith. matrix</td>
<td>N, UD</td>
<td>$SP_{12jd} = TD(KOMB_j \cup W_{11d})$</td>
<td>SP12 defines the number of time units for the TCs which have got forced allocation on the various days.</td>
</tr>
</tbody>
</table>
Several of these data areas are analogous to data areas for direct resource conflicts, e.g. DABL, DDF, DDFV, and DTAP. Note that no particular day conflict matrix is defined, and that DABL is usually stored in secondary storage. This implies a number of special definitions of some data areas, e.g. W1, SP5, Wl2. To consider forced allocations and the division of the time frame, data areas like W8, W11, DTV, and SP12 are defined. Note that forced allocations are connected to days and (usually) not to certain time units. This is sufficient, since interactions between time units on the same day are investigated for an extensive set of conditions.

5.4.2 Necessary conditions

Some necessary conditions for considering the following factors will be defined:
1. Interactions between day conflicts and other conflict types.
2. Conditions valid exclusively for day conflicts.
3. Interaction between forced allocations due to previous allocations, and new day conflicts due to the current allocation.

LEDIGd is found by means of the conditions in the previous sections (5.58) and (5.83). By considering day conflicts the freedom picture is modified:

\[(5.89) \quad \text{LEDIG}^d := \text{LEDIG}^d + \text{DTAP}_d \]

(5.20) must still be satisfied. For day conflicts only (5.20) may be transformed to:

\[(5.90) \quad \text{DDFV}_{i2} \geq \text{DDFV}_{i1} \quad \text{for} \quad i \in W7 \]

By means of (5.89) and (5.20) day conflicts are connected with direct resource conflicts, similar to other conflict types, and (5.90) considers day conflicts only. The interactions between the conflicts arising for day d and other days are listed below:

1. If \( W8_k = 1 \), then \( W11_kd = 1 \); i.e., forced allocations of an activity must be done before other allocations of it. (The rules for the allocation sequence consider this.)

2. Activities which after current allocation still can use day d, are called q1, and new activities being day-blocked are called q2:
\[ q_1 := \bigvee_{j=1}^{\text{LEDIG}_{d,s+1}} \]

\[ q_2 := \text{DDF}_{d}^{\text{gL}} + q_1 \]

(one may disregard activity \( k \), which is allocated to day \( d \))

\[ q_2^k := 0 \]

\( q_2 \) must be consistent with forced allocations on day \( d \); i.e.:

\[ W_{1d} \land q_2 = 0 \]

3. TDTCs (BDTC) with one or several new blockings on day \( d \) are called \( q_3 \).

\[ q_3 := \text{DABL} \land (W_{12} \land q_2) \]

These day-TCs must still be able to use day \( d \), and a necessary condition is:

\[ (5.92) \quad q_3 \in (\text{DABL} \land q_1) \]

\[ (5.92) \] is a special case of \((5.90)\), and in practice SP5 is used to find whether this condition is satisfied.

4. New forced allocations are represented by DTV, and a joint representation of these activities is:

\[ q^4 := \bigvee_{j=1}^{\text{DTV}_j} \]

They must not get forced allocation to more days than the number of periods for the activities, i.e.:

\[ (5.93) \quad |(W_{11} \lor \text{DTV}^{\text{gL}})_j| \leq \text{SP}_j \quad \text{for } j \in q^4 \]

5. \( q_2 \) may get forced allocations to other days if:

\[ |\text{DDF}_j^{\text{gL}}| = \text{SP}_j + 1 \quad \text{for } j \in q^2 \]

(This is identical with: \( |\text{DDF}_j^{\text{gL}} + 1| = \text{SP}_j \))

Activities satisfying this condition are called \( q_5 \), and new activities for TC \( \text{KOMB}_h \) forced to day \( h \) are given by:

\[ q^6_h := (\text{KOMB}_h \land \text{DDF}_h^{\text{gL}} \land q_5) + W_{1h} \]

In section 5.3.2 it was defined that SP3 represents the utility of TCs on the various days, and a new necessary condition is:
(5.94) \[ \text{SP}^{3}_{i,h} \geq \text{SP}^{2}_{i,h} + \text{TD}(q^{6}_{h}) \]
for \(i=1,2,\ldots,KM\)
and \(h=1,2,\ldots,UD\)

((5.94) is an important special case of (5.20) and (5.21)).

(5.89) connects the day conflicts with the other conflict types, whereas (5.90) to (5.94) are somewhat simplified conditions for interactions between days and forced allocations. New forced allocations due to vectors \(q_2\) and \(q_4\) may lead to new blockings for other days; this is, however, unlikely.

Assuming that further consequences of \(q_2\) and \(q_4\) can be ignored, it will be sufficient to investigate (5.20) for the freedom picture of the actual day of an allocation, whereas (5.90) to (5.94) sufficiently consider the interaction between the days.

This is a vital aspect for the chosen strategy, and for practical structures it has proved valid. Regardless of the school structure, computing time will set a practical limit for which possible consequences can be examined. It may become necessary to expand and modify the discussed conditions, but the principles for this will be analogous.

The day conflicts have hitherto been formulated as "direct" day conflicts, but "complex" day conflicts may also occur (i.e., the number of periods of a given set of activities which can be allocated to the same day must not exceed a given value). This requirements usually appears as a quality criterion, and rarely as an absolute requirement.

5.5 The form of the conflict picture

The consequences of the various conflict types will be summed up qualitatively. From the relations discussed in this chapter is found:

1. New conflicts due to direct and complex resource conflicts usually arise only for the time units where an allocation is done.
2. The variable period length requirement spreads the conflicts around the time units where an allocation is done, and conflicts
from this requirement are limited to the actual day of the allocation.

3. The day conflicts spread to other days, and the various conflicts may be superposed.

In figure 5.24 is shown a principle outline of how the number of new blockings relate to the time units for an allocation. (Some of the conflict types discussed later are also indicated.) By examining practical problems one discovers the accordance with figure 5.24, or rather, one usually finds that the actual conflict pictures are only a subset of the conflicts which are theoretically possible from the relations examined. This verifies that the strategy evaluates a sufficient number of possible consequences.

The form of the conflict picture is conceptually important, since it is a guide for the acceptable approximations. It seems natural to assume that the "conflict-distribution" indicated in figure 5.24 is typical for all problems of resource allocation, and any approximate method for solution of such problems must consider characteristics of the possible conflict pictures. A good measure for the quality of a method is that the conflict pictures actually formed are only subsets of the conflicts theoretically possible from the method.
1. Conflicts for current time intervals.
(Direct and complex resource conflicts, forced allocation, incidence conditions, etc.)

2. Conflicts for current day.
(Variable period length, continuity conditions, intermission conditions, quality criteria, etc.)

3. Conflicts for "neighbouring days".
(Quality criteria, precedence- and succeedence relations, etc.)

4. Conflicts for all days.
(Day conflicts, forced allocations, etc.)

Qualitative survey of the number of conflicts per time unit by allocation of an activity to time units 3 and 4 on Wednesday after an arbitrary number of steps in the allocation process.
6. THE ALLOCATION SEQUENCE

Summary

The degree of freedom for an activity is defined, and under otherwise equal conditions the activity with the smallest degree of freedom is allocated. This main rule is modified from for example, the prehistory or empirical rules. Principles for finding alternative solutions are discussed.

An exact method is characterized by: One can show that a solution exists for all steps of the allocation, and for each component in LEDPER equal to 1 there exists at least one solution where this allocation is possible. The allocation sequence is in principle uninteresting under such ideal conditions, and is then only important as regards necessary computing time. For practical structures the conditions are not so favourable, and the rules for the allocation sequence are vital to a successful result.

Manual scheduling methods have shown that apparently impossible schedules can be simple if one starts at the right end of the problem. The explanation is that even though one cannot keep track of all possible consequences, one can implicitly provide for the most important conditions being satisfied by a suitable allocation sequence. Complex interactions between global conditions are difficult to evaluate from only a local viewpoint. It is therefore sensible as soon as possible to allocate the activities which are known from experience to create problems. There is most freedom in the beginning, and of course the rest of the activities must adjust to the critical activities.

In other words, it seems natural always to allocate the "most critical" activity. This concept is not unambiguous and self-evident. The most critical activity may for example be:

1. The activity with the smallest number of available periods.
2. The activity with the largest period length.
3. The activity with most potential conflict possibilities.
4. The activity included in most TCs.
5. One of the activities included in the largest TC.
6. One of the activities included in forced assignments.
7. The activity which is most complicated to adjust manually, etc.

Practical experience shows that the combination of several such factors must be considered. Some methods are mentioned where for example activities with large period length are allocated first, or activities which include the most used resources are allocated first. The idea behind such principles is sound, but too stereotype; so that the effect of other critical conflict possibilities may be neglected.

The main requirement of the rules to be defined is that they are nuanced and flexible; i.e., several factors must be considered, while simultaneously the rules can be modified by simple means.

The rules for the allocation sequence can be divided into three kinds:

1. The main rule, when all other conditions are equal for the activities.
2. Regard for the prehistory (i.e., forced allocations).

The degree of freedom for an activity is defined as:

(6.1) From the set of available periods for an activity can be found subsets of periods to which at least one period of the activity must be allocated. The smallest of these sets is called the degree of freedom for the activity.

The most important and simplest division into such subsets is found from the condition that only one period of an activity can be allocated to one day.

Activity \( j \) consists of \( \text{SP}_j \) periods, and the number of available days is \( \text{DL}_j := |\text{DDF}_j^T| \). A vector \( d_j \) satisfies:

(6.2) \[
\begin{align*}
&d_j \in \text{DDF}_j^T \\
&|d_j| = \text{DL}_j - \text{SP}_j + 1 \quad (>0)
\end{align*}
\]
Available periods for the activities on the various days are given by SP2, and one will determine:

\[ d^j_i = \min_{i \in SP2} \{ i \in d^j_i \} \]

where \( d^j_i \) satisfies (6.2).

The vector satisfying both (6.2) and (6.3) is called \( d^j_{\text{min}} \) and the corresponding \( DF_j \) may be the degree of freedom for activity \( j \), since at least one period must be allocated to days \( d^j_{\text{min}} \). Available periods on these days are given by \( p^j \):

\[ p^j = \text{LEDPER}^T_j \Lambda q^j \]

where \( q^j = \begin{cases} 1, \text{if } \text{MOD}(i-1,\text{UD}) = 0 & d^j_{\text{min}} \\ 0, \text{otherwise} \end{cases} \)

The main rule for the allocation is:

(6.5) Under otherwise equal conditions the activity \( j \) with lowest \( DF_j \) from (6.3) is allocated, actual periods are given by (6.4).

If several activities have the same minimal \( DF_j \), the one with maximum value for \( |KOLMA|_j \) is allocated.

The justification of the latter modification is that as soon as possible one wants to allocate activities with the highest number of conflict possibilities, in order to consider these while there is still some freedom in the system.

The main rule is favorable for computing time, but it is more important that several necessary conditions are implicitly satisfied.

Assume for example:

\[ KOLM_{ij} = 1 \]
\[ DF_j = 1 \text{ and } DF_i = 2 \]
\[ p^1 \in p^i \]

If activity \( i \) is allocated first, it must be secured that activity \( j \) still has one available period, whereas an opposite sequence gives that such a condition is implicitly satisfied. By expanding this example to more complex situations, one finds analogous conditions; the explanation of which is founded on the rule that apparently unfortunate conflicts can be accepted more readily because the
allocated activity has the smallest number of possibilities, and that one may evade a critical situation later in the allocation. This principle appears to be so obvious that further examples should be unnecessary.

If activity \( j \) is included in day conflicts, \( p^j \) can be further limited. Assume:
\[
j \in \text{DABL}_i
\]
One knows:
\[
\begin{align*}
\text{DDVF}_{i1} & > \ SP_j \\
\text{DDVF}_{i2} & \geq \ DL_j
\end{align*}
\]
The days available for \( \text{DABL}_i \), but not for activity \( j \), are called \( q_1 \):
\[
q_1 := \sum_{k \in \text{DABL}_i} V_k \text{DDF}_k^T + \text{DDF}_j^T
\]
Per definition is:
\[
|q_1| = \text{DDVF}_{i2} - \ DL_j
\]
Utility \( G(q_1 \setminus \text{DDF}, \text{DABL}_i) \) can be evaluated as shown previously, but this utility is assumed to be approximately maximal (due to the slack structure of the day conflicts this is acceptable):
\[
G(q_1 \setminus \text{DDF}, \text{DABL}_i) = \text{MIN}(\{q_1, \text{DDVF}_{i1} - \ SP_j\})
\]
\[
= \text{MIN}(\text{DDVF}_{i2} - \ DL_j, \text{DDVF}_{i1} - \ SP_j)
\]
Of the days which can be used by activity \( j \), a number may possibly have to be used by other activities in \( \text{DABL}_i \). The minimal number of days which must be used in this way is called \( D_M_i(j) \).
\[
D_M_i(j) := \text{DDVF}_{i1} - \ SP_j - G(q_1 \setminus \text{DDF}, \text{DABL}_i)
\]
\[
= \text{MAX}(0, (\text{DDVF}_{i1} - \text{DDVF}_{i2}) - (\ SP_j - \ DL_j))
\]
\( D_M_i(j) \) is interpreted as the freedom loss of \( \text{DDF}_j^T \) due to \( \text{DABL}_i \). One of the day conflicts where \( j \) is included, maximizes the freedom loss, which is symbolized \( D_M(j) \):
\[
(6.6) \quad D_M(j) = \max_{i \in \text{DABL}_j} \text{MAX}(0, (\text{DDVF}_{i1} - \text{DDVF}_{i2}) - (\ SP_j - \ DL_j))
\]
It is now demanded that:

\[(6.7) \quad \begin{cases} d_j^3 \in \text{DF}_j^T \\ |d_j^3| = DL_j - SP_j + 1 - DM(j) \end{cases} \]

(If \(|d_j^3| \leq 0\) according to (6.7), no solution exists due to day conflicts.)

If \(W_{l_j} \equiv 1\), \(DF_j\) is determined by (6.7) and (6.3). It cannot be absolutely concluded that \(j\) must be allocated to one of the corresponding periods \(p_j^i\). Therefore \(DF_j\) is employed to determine \(j\) shall be allocated, while possible periods \(p_j^i\) are still determined by (6.2) and (6.4). This is a simple and satisfactory way of considering consequences of day conflicts at an early stage.

If \(W_{S_j} \equiv 1\), then activity \(j\) has got forced allocation to one or several days. \(DF_j\) is then determined by:

\[(6.8) \quad DF_j := \text{MIN} (SP_{j,i}^2) \quad \text{if} \quad W_{S_j} \equiv 1 \quad i \in W_{l_j}^T \]

The corresponding \(p_j^i\) are the periods on the day satisfying (6.8).

Activity \(j\) may also have forced allocation to certain time units, e.g. due to (5.21). Several conditions may lead to such forced allocations. This is represented by vector \(W_2\):

\[(6.9) \quad W_2^j := \begin{cases} 1, \text{ if activity } j \text{ has forced allocation to a subset of the available time units.} \\ 0, \text{ otherwise.} \end{cases} \]

Note that it is not always explicitly represented to which time units activity \(j\) has got forced allocation. The reasons for that are:

1. One cannot always find an unambiguous representation of those time units where an activity has got forced allocation (e.g., if \(r_j^0\) in the relation (5.21)).

2. If more complete conflict pictures are made due to forced allocations to certain time units, and the original problem definition is self-contradictory, one runs the risk of getting a partial solution which cannot be adjusted manually. (This problem disappears when the school structure is simple.)
These factors are considered in the allocation sequence:

(6.10) When \( j \in W_2 \), \( DF_j \) is determined by the mentioned relations, but the activities in \( W_2 \) are allocated before other activities.

The minimum degree of freedom is then found from a subset of the activities (the search process is selectively done). The result of this is that forced allocations are usually allocated immediately, which in many ways is equivalent to making complete conflict pictures. The advantages of the chosen principle are reduction of computing time, better possibilities for finding a partial solution which can be adjusted manually, and uniform treatment of forced allocations. The disadvantage is of course that incomplete conflict pictures increase the probability for erroneous allocations.

Which sets of activities ought to get allocation-priority can be defined externally. The justification for this is as follows:

1. experience as to which sets of activities most frequently give insolvable conflicts for the actual school structure,

2. the consideration for manual adjustment of self-contradictory specifications, and

3. the regard for particular quality criteria.

Such elements are represented by vector \( W_0 \):

(6.11) \[
W_0_j := \begin{cases} 
1, & \text{if activity } j \text{ has allocation-priority} \\
0, & \text{otherwise}
\end{cases}
\]

This gives a two stage selective search process: Activities in \( W_2 \) are allocated first, and the activities in \( W_0 \) are allocated before the rest of the activities in \( W_3 \).

Several external directives can be defined: \( W_0 \) is given an initial value. After allocation of these sets of activities, \( W_0 \) is redefined, etc. When all external directives have been executed, \( W_3 \) is assigned to \( W_0 \).

For tight (and self-contradictory) systems another priority is more suitable:: For the first directives by means of \( W_0 \), these activities have priority to activities in \( W_2 \). For later directives the
activities in $W_0$ and $W_2$ are treated equally, and only when the most complicated activities have been allocated, is $W_2$ given priority compared to other activities. Such organization facilitates a possible manual adjustment.

The discussed principles may be expanded by using still more "levels" to control the allocation. The danger is then that it becomes too detached from the main rule, which in practice becomes confusing and unstable.

The rules for the allocation sequence may be summarized in short:

The degree of freedom for an activity is normally determined by (6.2) and (6.3). In special cases (6.7) and (6.3), or possibly (6.8), are used. The activity with the lowest degree of freedom (and most conflict possibilities) will be allocated. This main rule is modified by vectors $W_2$ and $W_0$.

When one solution exists for a system, usually a large number of other solutions exist too. For the actual method there are the following possibilities for generation of several solutions:

1. changing pre-assignments or blockings,
2. defining new quality criteria, and/or
3. guiding the allocation sequence.

The first possibility defines explicitly which new requirements are demanded from a solution, while the other possibilities do this indirectly. From a practical point of view the latter is best, since new conditions are not then too strictly formulated. (Some methods use stochastic processes to find alternative solutions. This is a dubious technique, since one has little control of the solution alternatives, and error situations may just as well be due to the method as to the problem definition.) With the mentioned possibilities and a thorough knowledge of the program, one can predict characteristics of various solution alternatives with a high degree of certainty.

Changing of the initial conditions is in other words the only possible way to find several solutions, and in principle an unlimited number of solutions may be made in this way. Insignificant changes of the initial conditions have few consequences for the solution,
which indicates that the method is stable. By means of the initial conditions one may sometimes escape from hopeless conflict situations. Assume that a schedule is made from standard initial conditions and that no solution is found. An analysis of the intermediate results may show very particular conflict situations, and other initial conditions may offer an acceptable partial solution. This proves of course that the method is imperfect; among other things because very particular interactions between the conflicts are not considered. However, changing the initial conditions more than a couple of times is unnecessary. When this is done, it can definitely be proved that an impossible schedule is due to a self-contradictory specification.

The initial conditions are particularly important when regarding possible manual adjustment. The activities containing many resources (especially if this is many student groups) are almost impossible to adjust manually, while simple activities can be moved more easily. Vectors W17 and W18 represent two levels for the difficulty of allocating an activity manually:

$$W17_j = \begin{cases} 1, & \text{if activity } j \text{ consists of more than one student group and one teacher} \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (6.12)$$

$$W18_j = \begin{cases} 1, & \text{if the sum of student groups and teachers is larger than 4 for activity } j \\ 0, & \text{otherwise} \end{cases}$$

For impossible schedules the activities in W17 and W18 are given priority to activities in W2; i.e., error situations are accepted if they can be corrected manually. This is an important reason for not making too complete conflict pictures due to forced allocations, as one runs the risk that it becomes impossible to allocate the activities in W17 and W18.

Standard initial conditions may change for the various school structures. To discuss these more empirical evaluations would lead too far, but if one has no particular knowledge of a school structure, the following initial conditions are recommended:
Directive 1: \( W_0 \) initially consists of the activities in \( W_{18} \) with period length 2 (or more) plus all other activities with period length larger than 2. (\( W_0 := (W_{18} \cup W_5) \cup W_6 \))

Directive 2: \( W_0 \) is redefined to:
   a. The rest of the activities in \( W_{18} \).
   b. Activities with period length 2.
   c. Activities using overloaded special rooms.

Directive 3: The rest of the activities in \( W_{17} \).

The reason for giving priority to the mentioned sets of activities is: After a number of steps in the allocation, chain reactions may occur as a result of forced allocations, and practical experience has shown that the probability for this resulting in error situations is smaller if the mentioned activities are allocated early.

For the relations discussed in this chapter an important data area, \( S\Pi_1 \), will be defined. It is an arithmetic matrix, called the description of the activities. \( S\Pi_1 \) partly contains the definition of the different activities (e.g., the number of periods, and period length). Further, \( S\Pi_1 \) contains important parameters for calculation of the degrees of freedom and some qualitative parameters to be discussed later. One (or several) row vector(s) in \( S\Pi_1 \) describes an activity, and the current program system assumes a row vector to consist of 10 components. The dimension of \( S\Pi_1 \) is \([n,10]\), and \( S\Pi_{1j} \) is defined as follows:

\[ S\Pi_{1j1} := S\Pi_{1j}, \text{ i.e., the number of periods for activity } j. \]
\[ S\Pi_{1j2} := \text{The number of periods without forced allocation to certain days, i.e.:} \]
\[ S\Pi_{1j2} := S\Pi_{1j1} - |W_{11_j}^T| \]
\[ S\Pi_{1j3} := \text{PL}_j, \text{ i.e. the period length for activity } j. \text{ (If an externally defined activity has variable period length, this is represented by several row vectors in } S\Pi_1). \]
\[ S\Pi_{1j4} := DF_j, \text{ i.e., the degree of freedom for activity } j. \]
\[ S\Pi_{1j5} := \text{The number of days where activity } j \text{ has available "desired" periods. This parameter is connected with qualitative criteria to be defined later.} \]
SP1_{j6} := |W11_j^T|, i.e., the number of periods for activity $j$ with forced allocation to certain days.

Note: SP1_{j1} = SP1_{j2} + SP1_{j6}

SP1_{j7} := The type of activity $j$. SP1_{j7} defines which time units are desired or acceptable for activity $j$, and this parameter is discussed in chapter 9.

SP1_{j8} := Special quality criteria for activity $j$. SP1_{j8} is defined in chapter 9.

SP1_{j9} := DM(j), i.e., the freedom loss for DDF_j^T due to day-blockings.

SP1_{j10} := |DDF_j^T| - |W11_j^T|, i.e., the number of available days for activity $j$ where it does not have forced allocation.

SP1 is not an unchanging data structure like e.g. KOLMA. The definition of SP1 can be modified due to the design of the algorithms, regard for the computing time, or special properties of certain school structures. In connection with SP1 two auxiliary vectors are defined:

EXTERN: The vector gives an external symbol for the activities used for intermediate printout etc. The symbol usually consists of the name of the first student group which has the activity plus a code number.

RANK: A ranking of potential conflict possibilities:
If \(|KOLMA_j| \geq |KOLMA_i|\) then RANK_j < RANK_i
(i.e., lowest value in RANK has priority under otherwise equal conditions.)
7. CRITERIA FOR ALLOCATION OF AN ACTIVITY

Summary

The most important global condition is (5.20). By considering the form of typical conflict pictures, (5.20) is transformed to relations which are satisfactory with regard to computing time, while simultaneously from a practical point of view they limit the solution space sufficiently.

By means of local criteria one tries to allocate the activities in such a way as to maximize the probability of (5.20) being satisfied for future allocations. The best principle for this is found from an analogy with global criteria.

Pre-assignments (and blockings) determine the first part of the allocation. The data structure is modified to account for this, and the rules for the allocation sequence find the next activity to be allocated. It is assumed:

Activity k will be allocated to one of the periods $p^k$ on days $d^k$ after s steps in the allocation. $d^k$ and $p^k$ are given by the relations in chapter 6. (p is a reference to one of the periods in $P^k$, and $p$ belongs to day $d=\text{MOD}(p-1,\text{UD})+1$).

1. For each possible period $p$ one wants to know whether the transformation

(7.1) $\text{LEDIG}_s \rightarrow \text{LEDIG}_{s+1}$

still maintains the (assumed) existence of a solution.

2. For those periods giving acceptable transformations of the form (7.1), one will find the period(s) which probably is the best when considering that there must be at least one acceptable transformation (7.1) for all later stages in the allocation.

7.1 Global criteria

In chapter 5 limitations of the solution space due to global conditions were discussed. (These conditions must of course be modified when other conflict types are introduced.) Due to the form of the conflict picture certain simplifications may be done in order to decide whether period $p$ gives an acceptable transformation (7.1):
(7.1) is assumed to provide the existence of a solution if:

a. The transformation

\[(7.2) \quad \text{LEDIG}^{d,s} \rightarrow \text{LEDIG}^{d,s+1}\]

still maintains the existence of a solution.

b. The conditions for interactions between days (section 5.4) are satisfied.

The most important condition to be satisfied by (7.2) is (5.20); i.e.

\[(7.3) \quad G(\text{LEDIG}^{s+1}, \text{KOMB}_i) \geq \text{TT}(\text{KOMB}_i)^{s+1}\]

for \(i=1,2,\ldots, KM\)

One knows:

\[\text{TT}(\text{KOMB}_i)^{s+1} = \text{TT}(\text{KOMB}_i)^{s} - \delta_{ik} \cdot PL_k\]

where \(\delta_{ik} = \begin{cases} 
1, & \text{if } k \in \text{KOMB}_i \\
0, & \text{otherwise}
\end{cases}\)

Assume that \(\text{KOMB}_i\) does not get any new freedom loss for other days than day \(d\); i.e.:

\[G(\text{LEDIG}^{s+1}, \text{KOMB}_i) = G(\text{LEDIG}^{s}, \text{KOMB}_i) - G(\text{LEDIG}^{d}, s, \text{KOMB}_i) + G(\text{LEDIG}^{d}, s+1, \text{KOMB}_i)\]

(7.3) may be written as:

\[(7.4) \quad G(\text{LEDIG}^{s}, \text{KOMB}_i) - \text{TT}(\text{KOMB}_i)^{s} \geq G(\text{LEDIG}^{d}, s, \text{KOMB}_i) - G(\text{LEDIG}^{d}, s+1, \text{KOMB}_i) - \delta_{ik} \cdot PL_k\]

for \(i=1,2,\ldots, KM\)

(7.4) covers an important special case not apparent from (5.20):
Assume that a TC is broken. (5.20) will then not be satisfied for a number of the possible periods. This may be due to the pre-history and not the actual allocation. By comparing utilities before and after an allocation like in (7.4), an unnecessary limitation of solution space due to prehistory is avoided. With the defined data areas, (7.4) may be written:

\[(7.5) \quad \text{TTR}_{i2} - \text{TTR}_{i3} \geq SP_{i3} - G(\text{LEDIG}^{d}, s+1, \text{KOMB}_i) - \delta_{ik} \cdot SP_{k3}\]

for \(i=1,2,\ldots, KM\)

If \(i \in W9\) (i.e., \(\text{KOMB}_i\) is a TTC or BTC), (7.5) is reduced to:

\[(7.6) \quad 2 \geq SP_{i3} - G(\text{LEDIG}^{d}, s+1, \text{KOMB}_i) - \delta_{ik} \cdot SP_{k3}\]

for \(i \in W9\)
Computing is reduced to finding $G(\text{LEDIG}^{d,s+1}, \text{KOMB}_i)$. This is done in two ways:

1. KOMB$_i$ is an STC

In this case the probability of (7.5) being satisfied is high, and a rough approximation for evaluation of $G$ may be used. It is simplest to assume maximum utility (relation (5.29)). This approximation is too rough because variable period length is not considered, and relation (5.85) is used to determine $G$. One knows that most of the new conflicts arise for those time units where an allocation is made, and this is used to improve (5.85):

It is particularly investigated whether freedom loss for KOMB$_i$ occurs for the time units where an allocation is attempted. This is not discussed in detail, but the consequence is that one can find a more correct expression for $G$ than (5.85) when $k$ has period length 2 or more.

2. KOMB$_i$ is a TTC (BTC)

Again a modified version of (5.85) is employed to consider variable period length; however, (7.6) is often not satisfied due to other conditions, and in addition $G(\text{LEDIG}^{d,s+1}, \text{KOMB}_i)$ must be evaluated from the more complete method mentioned in section 5.3. Since the current time units are on the same day, other conflict types may also be considered, e.g.:

If forced allocations occur for certain time units, these activities must be blocked for other time units of the actual day. (No activity can have more than one period on one day.) If an activity included in day-conflicts gets forced allocation, the activities with which it is day-conflicting must be blocked for LEDIG$^{d,s+1}$.

A method combining the method of section 5.3 and elements of the kind mentioned above, has been tried in practice. Unfortunately, it used much computing time, but provided that a problem is very strict and resource conflicts completely dominating, the method is suitable. Even for tight realistic structures it has fortunately proved possible to make considerable simplifications to calculate the utility.
(In fact, it has been difficult to find realistic situations where a more complete method gives another value of \( G \) than these simplifications. The error situations which arise are usually due to other factors than an approximate evaluation of \( G \).) The simplifications are based on the following properties of practical structures:

a. Usually a small number of activities is responsible for freedom loss for a TC, and this happens when only a few activities can be allocated to the same time units.

b. For those time units where the number of hours for activities in \( \text{KOMB}_i \) available, is larger than \(|\text{LEDIG}_{d,s} \text{ A KOMB}_i|\), the probability of a freedom loss occurring is small.

c. For the rest of the time units on day \( d \) maximum utility may be assumed.

d. The rules for the allocation sequence act more or less as a "feed back", and increase the probability of the above being correct.

These elements will be transformed into an algorithm. One sets:

\[
\begin{align*}
q^i &= \text{LEDIG}^{d,s} \text{ A KOMB}_i \\
T_i &= |q^i| \quad (T_i \text{ is available time units for KOMB}_i \text{ on day } d) \\
r_i &= 0 \quad (r_i \text{ is initial freedom loss for KOMB}_i) \\
C^i &= \text{LEDIG}^{d,s+1} \text{ A KOMB}_i
\end{align*}
\]

1. If \( TD(C^i_t) \geq |q^i| \), then time unit \( t \) can get no freedom loss, and it is eliminated from \( q^i \):

\[
\begin{align*}
q^i_t &= 0 \\
|q^i| &= |q^i| - 1
\end{align*}
\]

2. \(|C^i_t| = 1\), i.e., forced allocation for time unit \( t \), and \( t \) is eliminated from \( q^i \):

\[
\begin{align*}
q^i_t &= 0 \\
|q^i| &= |q^i| - 1
\end{align*}
\]

\[\text{LEDIG}^d_t = \text{LEDIG}^d_t + \text{KOLMA}_j \quad \text{where } C^i_{tj} \equiv 1\]

\[\text{LEDIG}^d_p = \begin{cases} 1, & \text{if } p = t \\ 0, & \text{otherwise} \end{cases}\]
If activity $j$ has period length unequal to 1, then activity $j$ must not be blocked for neighbouring time units of $t$. (This modification is not listed.)

3. $|C_{j}^{i}| = 0$, i.e., $\text{KOMB}_{i}$ has got a freedom loss for time unit $t$:

$$q_{t}^{i} = 0$$

$$|q^{i}| = |q^{i}| - 1$$

$$r_{i}^{i} = r_{i}^{i} + 1$$

4. Points 1, 2 and 3 define an iteration. When this terminates, the remaining time units in $q^{i}$ are assumed to have maximum utility; i.e.,

$$(7.7) \quad G(\text{LEDIG}^{d_{i}, s_{i}^{+}, \text{KOMB}_{i}}) = t_{i}^{p} - r_{i}^{p} - \text{MAX}(0, |q_{i}^{i}| - \text{TD}(q_{i}^{i}))$$

where $q_{i}^{i} = \sum_{p \in q^{i}} C_{i}^{p}$

(7.7) is an approximate expression for (5.47). Computing time for (7.7) is modest, and this relation has proved satisfactory. The resulting $q^{i}$ from the iteration may lead to forced assignments if $|q^{i}| \geq \text{TD}(q_{i}^{i})$. These consequences are evaluated by using (5.21) and (5.22) iteratively. Accordingly, limitations of the solution space due to the general condition (5.20) are solved in practice as follows:

It is assumed that TCs can only get freedom loss for the actual day of an allocation, and all TCs for the system must satisfy (7.5) where $G$ is evaluated from a modified version of (5.85). In addition, the TTCs for the system must satisfy (7.6), where $G$ is evaluated from (7.7). (For very tight structures (5.47) is used instead of (7.7).)

### 7.2 Local criteria

Assume that $p^{k}$ is reduced such that the remaining periods satisfy the global conditions discussed in the previous section (and other global conditions mentioned in chapter 5). One knows that this is not a sufficient limitation of the solution space, and one wants to find the period $p \in P^{k}$ maximizing the probability of a solution existing after execution of the actual allocation. This will be done from
more local criteria, and possibly a wrong allocation may be done. Apparently this is vital to the chosen strategy, but experience has shown that it is not as important as one might think, because:

On terminating global limitations of \( P^k \), it is often found:

a. For simple structures the choice of period may be done from qualitative criteria alone, since the limitation of \( P^k \) is sufficient.

b. For tight structures characteristic chain reactions occur due to the prehistory, so that possible periods are reduced to a small number (frequently 1).

c. In those cases where it is difficult to distinguish between possibilities, the various choices ordinarily only lead to "symmetrical" interchanges in the solution.

This is only approximately correct, but one important point is indicated from this:

The strategy in question results in limitations of the solution space that are so complete, that the global conditions become much more important than local conditions.

A number of local "optimization functions" have been investigated. They can be divided into two main principles:

**Principle 1**

The most obvious is to attempt to minimize the number of new conflicts arising. This is based on the following idea:

Assume that \( p1 \) and \( p2 \) are the only possible periods for activity \( k \). The corresponding conflict pictures are \( \text{TAP}^{P1}_k \) and \( \text{TAP}^{P2}_k \). Through a suitable coordinate transformation the same row vectors of the two conflict pictures represent the time units where an allocation may be done. If \( \text{TAP}^{P1}_k \leq \text{TAP}^{P2}_k \) and a solution exists, it will still exist if activity \( k \) is allocated to period \( p1 \). This is valid provided that the system only consists of direct resource conflicts.

It is simple to show that this is correct, but the idea has other important weaknesses:

a. Nothing can be concluded as to the existence of a solution by allocation to period \( p2 \).

b. The probability is small for the various conflict pictures being
connected by means of subset relations. By considering "partial" conflict pictures where all blocked activities belong to the same TC, subset relations are frequent; however, this does not indicate whether these are also fulfilled for the total conflict pictures.

c. Other factors than subset relations are dominating for other conflict types.

These objections are neglected at present, and some alternative functions based on the idea of minimizing new conflicts will be mentioned. DFTAP_p is a collective term for functions of the sum of new conflicts resulting from possibility p. The simplest form of DFTAP_p is:

\[ DFTAP_p := \sum_{i=1}^{TMAX} \sum_{j=1}^{n} TAP_{ij}^p \]  

(7.8)

(7.8) disregards that each activity may represent several periods and that the period length is variable. For these elements to be considered, one may say:

\[ DFTAP_p := \sum_{i=1}^{TMAX} TT(TAP_{i1}^p) \]  

(7.9)

Or:

\[ DFTAP_p = \sum_{i=1}^{TMAX} TD(TAP_{i1}^p) \]  

(7.10)

(7.10) is better than (7.9), since (7.10) also takes into account that not more than one period of an activity can be allocated to the same day.

When estimating G in chapter 5 it was pointed out that a number of new blockings were acceptable, while others may lead to error situations. It is therefore natural to assume that new conflicts could be given a weighting. For instance, it is obvious that the difference between available time units and the number of unallocated periods may be a weight for new conflicts for an activity. FH^S symbolizes a collective term for the data structure representing the prehistory, and the general form of DFTAP_p when using weights will be:
\[ T_{\text{MAX}} \]

\[ \text{DFTAP}_P := \sum_{i=1}^{T_{\text{MAX}}} f(P_i, T_{\text{AP}})_i \]

Several alternative designs of (7.11) have been examined, and the result will be summed up:

The chosen function for the weight of a conflict has little influence on the quality of the complete strategy. The simplest forms of (7.11) are in reality the best (e.g. (7.10) with some modifications). One may safely claim that regardless of the complexity of (7.11), it will be simple to find realistic cases where it leads to error allocation. The best that may be said of (7.11) is that it is acceptable as a rule of thumb for want of anything better.

The actual strategy was for some time successfully based on some form of (7.11). It is now known that this success resulted from limitations due to global conditions rather than to (7.11). Characteristic weaknesses of such functions were found:

a. The activities allocated first have a tendency to "lump" on the same time units. A certain lumping is correct, but is overestimated by (7.11). It turns out that activities allocated later belonging to tight TCs are often unable to utilize those time units where lumpings occurred in the beginning.

b. (7.11) is unstable (i.e., gives variable results) due to insignificant changes of the initial conditions.

c. Manual scheduling shows that when a schedule has much freedom, a possible period may be chosen almost arbitrarily, because different choices only lead to symmetrical interchanges in the schedule. However, (7.11) has the property that only a small number of periods is possible although the freedom may be large. This is often due to insignificant blockings, and (7.11) limits the possibilities too much. This has unfavourable results for the qualitative criteria which can be taken into account.

In short, relations like (7.11) make it difficult to distinguish between significant and insignificant conflicts.
Principle 2

The above principle evaluated the conflicts for each activity. In the following, conflicts for sets of activities will be regarded. This is done from the TCs in which the activities are included. Global limitations are determined by the TTCs of a system. It seems natural to define local criteria from the STCs of the system such that they become identical with global conditions ultimately; i.e., as soon as a TC becomes tight, global and local conditions become identical.

Parameter \( RT_i^S := \text{MAX}(0, G_i^S - \text{TT(KOMB}_i)) \) indicates whether KOMB\(_i\) is a TTC (BTC) or an STC. (When \( RT_i^S \) is 0, KOMB\(_i\) is tight or broken.) Limitation of the solution space is a result of new freedom losses arising from a possible allocation. For DFTP\(_P\) one may say that this is equivalent to the corresponding sets of conflicts having a weight \( \omega = \). In other words, instead of adding weights to each new conflict, new freedom losses for the TCs will be summed with corresponding weights. The form of the weights will be \( 1/RT_i^{S+1} \) (or alternatively \( f(1/RT_i^{S+1}) \)):

\[
(7.12) \quad \text{DFTP}_P := \sum_{i \in S} \frac{G_i^S - G_i^{S+1} - \delta_{ik} \cdot PL_k}{RT_i^{S+1}}
\]

where \( \delta_{ik} := \begin{cases} 
1, & \text{if } k \in \text{KOMB}_i \\
0, & \text{otherwise}
\end{cases} \)

By doing the same simplifications as in section 7.1 and utilizing the defined data structure, one may set:

\[
r_i^d := SP3_{id} - G(\text{LEDIG}_d^{s+1}, \text{KOMB}_i) - \delta_{ik} \cdot SP1_{k3}
\]

and:

\[
(7.13)^* \quad \text{DFTP}_P := \sum_{i \in S} \frac{r_i^d}{\text{MAX}(0, TTR_{12} - TTR_{11} - i_d)}
\]

The utility is in this case calculated from (5.85), with the modification mentioned in section 7.1. Ordinarily, \( k \) will be allocated to one of the periods minimizing DFTP\(_P\) from relation (7.13).

\(^*\) (7.13) might naturally lead to a global limitation of the solution space due to an STC if the denominator becomes zero for some \( i \in S \).
The above is equivalent to minimizing the probability that new global conditions may arise. If \( \omega \) STCs is far from being tight (\( TTR_{12} > TTR_{11} \)), a freedom loss is given a small weight, whereas near tight STCs are given a large weight. When there is a small probability of new TTCs arising, the probability of the prehistory at any time being consistent with the conditions for the system increases.

An activity is included in several TCs, and in that way new conflicts are also given a weight from potential conflict possibilities. It has been pointed out that the most characteristic difference between (7.11) and (7.12) is that (7.11) evaluates each conflict by itself, whereas (7.12) evaluates sets of conflicts. (7.12) offers a much higher possibility of neglecting the effect of illusory freedom in LEDIG, which will be given a particular mention:

Assume: \( PL_k \geq 1 \) and that new conflicts only arise for the time unit where an allocation is possible. Activity \( k \) has two possible periods \( p_1 \) and \( p_2 \), and \( KOMB_1 \) is the only STC in which the new conflicts are included.

Assume further:

\[
TAPT_{p_1} \in TAPT_{p_2} \\
0 < TT(LEDIG^S_{p_1} \land KOMB_1) < TT(LEDIG^S_{p_2} \land KOMB_1)
\]

Relations like (7.11) result in period \( p_1 \) being chosen, whereas (7.12) most likely will result in \( p_1 \) and \( p_2 \) being equally suitable, since no new freedom loss arises for any of the possibilities. More blockings for period \( p_2 \) arise, but this is illusory freedom because the activities are mutually conflicting. The number of new conflicts are therefore secondary compared to the number of new freedom losses. This explains why (7.11) is unstable, and why the activities lump too much. Under otherwise equal conditions it is of course advantageous to choose period \( p_1 \), but if this point is stressed too much, the same weaknesses as for (7.11) will arise.

(7.12) may be modified to consider possible manual adjustment and for the empirical fact that certain sets of activities easily lead to error situations due to complex interactions between conditions. The weight for \( KOMB_1 \) will then get the form \( \omega_1 = E_1 / RT^S_1 \), where \( E_1 \) is
an empirical parameter determined from analogous considerations as the rules for the allocation sequence. This possibility has not been tried, but it seems interesting since local criteria may be guided by external directives, which may be a way of treating qualitative criteria.

The operative consequences of (7.12) will be mentioned: When the freedom of the system is large, no important freedom loss occurs, and the choice of period is determined by qualitative criteria. For "critical" allocations one gets a distinct differentiation of the possibilities. For the last part of the allocation the interactions between the TCs are usually evaluated, and choice of period is again determined from qualitative criteria.

Experience from manual scheduling indicates that this is more in accordance with the actual circumstances than the results of (7.11). (Nevertheless, this relation gives often acceptable results, since for several critical allocations one gets the same results as for (7.12). In those cases where different choices appear, (7.12) is definitely the best one.)

In chapter 9 local optimization criteria and qualitative ones will be combined. One chooses the period $p \in P^k$ minimizing (7.12) (i.e. (7.13)) and simultaneously considering qualitative criteria "in the best way". This is of course an heuristic principle. The form of (7.12) is esthetically attractive, since both tight and slack TCs can be given uniform treatment. Thus, it may be tempting to say that this verifies the correctness in principle of the idea behind (7.12), but obviously the actual form of (7.13) may be modified on the basis of more experience.

Administrative processes for choice of period will be indicated: DFTAP for a possible period $p$ is stored in an arithmetic vector CRIT. The corresponding LEDIG$^d$, $p$, $s^1$ is stored in secondary storage. It is useful to store additional temporary information:

\[ DT_{W30}^P := \mathbf{V} \ TAP_{i1}^P, \text{ i.e., } W30 \text{ are the activities which will get new } i=1 \text{ conflicts if period } p \text{ is chosen.} \]

\[ W31^p := \text{Activities with forced allocation to day } d \text{ if period } p \text{ is chosen.} \]
$W_{32}^P = \text{KOMB } A W_{30}^P$, i.e., $W_{32}$ are the TCs which will be affected if period $p$ is chosen.

By means of CRIT the period to which $k$ will be allocated is found, and corresponding data areas are transferred to primary storage.
8. MODIFICATION OF DATA STRUCTURE DUE TO CHOSEN ALLOCATION

Summary

The data structure representing the prehistory has been given the collective term $Fh^d$, and the transformation $Fh^d + Fh^{d+1}$ will be done. This implies a number of administrative "cleaning up" processes, which are consequences of the definitions in the previous chapters. In principle this shows nothing new, and it will only be given a summary mention. Unfortunately, due to the extensive data structure, the various relations will probably seem confusing.

Further, one will compensate for the previous approximations by examining the chosen allocation for a more comprehensive set of global conditions, and transform a number of implicit conditions into explicit ones. Simplified, this is called: Generation of new conditions.

From (7.13) (and qualitative criteria) is found:

Activity $k$ with period length $PL_k$ is allocated to period $p$ on day $d$ after $s$ steps in the allocation.

8.1 Modification of the prehistory

a. Allocation of activity $k$

Vector $A$ represents physical time units for period $p$ by operation

$$KONV(PL_k, 1, p, A):$$

$$\left\{ \begin{array}{l}
FFT^S_{k1} := 1 \\
TA^S_{i1} := L(FA^S_{SPu_k}, TA^S_{i1})
\end{array} \right. \text{ for } i \in A$$

b. Special modifications for activity $k$

$$SP^S_{k1} := SP^S_{k1} - 1$$

If $SP^S_{k1} = 0$:

$$\left\{ \begin{array}{l}
W_0 := W_3 := (W_3 + PL) \\
W_2 := W_17 := W_{18} := 0 \\
LEDG^T_k := LEDPER^T_k := DDF^T_k := 0
\end{array} \right. \text{ if } (W_3 + PL_k) \text{ symbolizes one of the vectors } W_4, W_5, \text{ or } W_6,$$

depending on the period length.
If $W_{S_k^0}^S = 1$:

\[
\begin{align*}
SP_{k6}^{S+1} &= SP_{k6}^S - 1 \\
SP_{k7}^{S+1} &= SP_{k7}^S - PL_k \\
W_{l_d}^{S+1} &= 0 \\
W_{S_k^0}^{S+1} &= 0 \quad \text{if } SP_{k6}^S = 0
\end{align*}
\]

(8.4)

If $W_{S_k^0}^S = 0$:

\[
\begin{align*}
SP_{k2}^{S+1} &= SP_{k2}^S - 1 \\
SP_{k10}^{S+1} &= SP_{k10}^S - 1 \\
DDF_{dk}^{S+1} &= 0
\end{align*}
\]

(8.5)

If $(W_{S_k^0}^S = 0) \land (W_{l_d}^S = 1)$:

\[
\begin{align*}
DDF_{S_{11}}^{V_{11}^S} &= DDF_{S_{11}}^{V_{11}^S} - 1 \\
DDF_{S_{12}}^{V_{12}^S} &= DDF_{S_{12}}^{V_{12}^S} - 1 \\
DABL_{k1}^{S+1} &= 0 \quad \text{if } SP_{k1}^{S+1} = 0 \\
W_{l_1}^{S+1} &= 0 \quad \text{if } " - " \quad \text{and} \quad |DABL_{l_1}^S| \leq 1 \\
W_{l_2}^{S+1} &= 0 \quad \text{if } " - " \quad \text{or} \quad W_{l_1}^{S+1} \land DABL_{k2}^S = 0 \\
SP_{k9}^{S+1} &= 0
\end{align*}
\]

(8.6)

(8.2) is unconditional. (8.3) is a "cleaning up" if all periods of $k$ are allocated. (8.4) or (8.5) are executed depending on whether the actual allocation is forced to day $d$. (8.6) is only executed when $k$ is included in day conflicts, and the actual allocation is not forced to day $d$. (It will be shown later that all periods with forced allocation to a day are eliminated from day conflicts.)
(8.7) \[ \text{LEDIG}^{s+1}_i := \text{LEDIG}^{d,s+1}_j \quad \text{for } j = 1, 2, \ldots, DT \]
\[ \quad \text{and } i = (j-1).UD^d \]

Calculation of \( \text{LEDIG}^{d,s+1}_i \) has been shown previously. The consequences of (8.7) for LEDPER and SP2 will be shown. The modifications of LEDPER and SP2 are most easily found by treating the various period lengths individually. Activities with period length \( pl \) blocked in period \( j \) on day \( d \) are given by:

\[
\begin{cases}
\text{KONV}(p1, 1, j, B) \\
q_j^i := (W3+pl) A & (\forall \text{TAPT}^S_i) \\
\text{where } \text{TAPT}^S_i := \text{LEDIG}^S_1^{} \cdot \text{LEDIG}^{S+1}_i \\
\text{LEDPER}^{S+1}_j := \text{LEDPER}^{S}_j + q_j^i \\
\text{SP2}_{id} := \text{SP2}_{id-1} \quad \text{for } i \in q_j^i
\end{cases}
\]

(8.8) is done for \( pl = 1, 2, 3 \), and the periods belonging to day \( d \), since (8.7) does not change LEDIG for other time units.

d. Modification of day conflicts

The activities with new blockings are called \( W30^p \), and those among them completely blocked on day \( d \) are called \( q2 \):

\[ q2 := W30 + \text{DT} \sum_{i=1}^{\text{LEDIG}^{S+1}_1} \]
\[ q2_k := 0 \quad \text{(because activity } k \text{ is treated specially in section 8.1.b)} \]

One sets:

\[
\begin{cases}
\text{DDF}^{S+1}_d := \text{DDF}^{S}_d + q2 \\
\text{SP1}^{S+1}_{i10} := \text{SP1}^{S}_{i10} - 1 \quad \text{for } i \in q2
\end{cases}
\]

DTV will represent the activities which have got new forced assignments to certain days. Initially DTV = 0. When evaluating the utilities for the TCs, it was found that activities W31 were forced to day \( d \); thus:

\*Superscript \( p \) is unnecessary after choice of allocation.
(8.10) \[
\begin{align*}
\text{DTV}_d &= W31 \\
\text{DTV}_t &= \text{DDF}^T_i \quad \text{if} \quad \text{SP}_{i10}^{t+1} \geq \text{SP}_{i10}^{t+1} \quad \text{and} \quad i \in q^2
\end{align*}
\]

(8.9) influences the utility of the DTCs of the system. The DTCs not including activity $k$ which may get modified are denoted $q^3$:

\[
q^3 := W7A \left( \text{V} \quad \text{DABL}^T_i \right) \ast \text{DABL}^T_k
\]  

For $i \in q^3$ one sets:

\[
\begin{align*}
\text{DDF}_{i2}^{S+1} &= \text{DDF}_{i2}^{S-1} \quad \text{if} \quad |\text{DABL}_i \ast \text{DDF}^S_d| = 0 \\
\text{W}12_i^{S+1} &= \text{W}12_i^{S} \ast \text{DABL}_i \quad \text{if} \quad \text{DDF}^S_{i1} = \text{DDF}^S_{i2} \\
\text{DTV}_d &= \text{DTV}_d \ast (\text{DABL}_i \ast \text{DDF}_d) \\
& \quad \text{if} \quad |\text{DABL}_i \ast \text{DDF}_d| = 1 \\
& \quad \text{and} \quad \text{DABL}_i \in \text{W}12_i^{S+1}
\end{align*}
\]

It has previously been assumed that a DTC always has maximum utility. The consequence is that when $|\text{DABL}_i \ast \text{DDF}_d| \geq 2$ and $i \in q^3$, no modifications like (8.11) are of interest.

The relations discussed in this section only show in principle the necessary modifications of FH. When designing a program several relations must be transformed in order to reduce computing time. This is done by introducing auxiliary areas and using an extensive "masking technique".

8.2 Generation of new conditions

All conditions for a solution are of course implicitly given by the problem definition, and it is desirable to transform implicit conditions to explicit ones. This is equivalent to examining the allocation in question for a more complete set of global conditions than the ones used to distinguish between possible periods. Accordingly, it is assumed that the prehistory is consistent with the new conflicts and conditions. This compromise is done out of consideration for computing time, and the mentioned assumption is valid to a surprisingly high degree. This verifies that the previous limitations of the solution space are sufficient in practice. The new conditions
will act as a "stabilizing" factor for the rest of the allocation, and they partly compensate for simplifications made when choosing the period. The most important of those simplifications were:

a. The utility, G, has been evaluated only for a limited set of the TCs of the system, and it was evaluated from simplified expressions.

b. All consequences of forced allocations were not transferred to the conflict picture.

c. It was assumed that possible freedom loss for the TCs only occurred for the actual day of an allocation, and interactions between the days due to forced assignments and new day blockings were neglected.

The justification for these simplifications has been discussed. Obvious principles for considering these factors are:

a. Find sets of activities with few real allocation possibilities due to interactions between different conditions, and find SSCs which are likely to become tight due to the current allocation.

b. Modify the rules for the allocation sequence.

c. Introduce new blockings due to the activities forced to certain days.

If the set of TCs must be limited, it will not be the same set of TCs most frequently limiting the solution space for the complete allocation. It is very time consuming to redefine the set of TCs describing the system in the "best" way possible. One is frequently content with using the set of direct resource conflicts as the TCs of the system; but particularly for the first part of the allocation TCs of a higher order might be important. Later these are uninteresting, since the remaining activities are completely described by direct resource conflicts. Out of consideration for computing time the set of TCs must be determined initially. Compensation for all TCs not being examined is done in the following way:

One knows that certain SCs are highly likely to become tight. This is investigated after each allocation, and when it has been satisfied, a TSC is treated as if it were a TTC, which may in turn influence the allocation sequence.
The consequences will be:

Interactions between resource conflicts leading to TCs of an higher order will be evaluated.

By considering a TSC like a TTC, G is determined separately for the sets of activities which easily may lead to error situations. This leads to limitations of the solution space which may be difficult to find from approximate expressions for (5.20). (This implies that the performed simplifications give more correct values for G.)

By giving priority to certain TSCs when allocating, one takes into consideration interactions between other SCs which later in the allocation may lead to error situations, and an unnecessary lumping of the earliest allocated activities is avoided.

Assume that q is an SSC. The probability for it becoming tight is approximately given by:

$$P = \frac{TT(q)}{|LEDIG \setminus q|} \cdot \frac{k_0 \cdot |q|}{|LEDIG \setminus q|}$$

Both numerator and denominator grow as $|q|$ increases, but it is difficult to find a general expression for the maximum value of $P$.

Qualitatively one may say: SC q will be found such that $|q|$ is as large as possible, while simultaneously vectors $LEDIG_i^T$ for $i \in q$ are as equal as possible. This last property is satisfied for sets of activities with an approximately equal descriptor; i.e.:

Activities defined by a small number of resources and with at least two common resources are most likely to form a TSC. For streamed school structures there will in addition arise TSCs for activities having one common resource and including student groups from the same stream. (The latter is due to student groups within the stream having very similar freedom vectors owing to parallel activities.)

Accordingly, the most important SCs are:

1. Activities consisting of common class and common teacher.
2. Activities consisting of common class and common type of room.
3. Activities with common teacher and attached student groups belonging to the same stream.
4. Activities with common type of room and attached student groups belonging to the same stream.

5. Activities consisting of common teacher and common type of room.

Activities consisting of several common student groups are not included among the important SCs. Such activities are ordinarily allocated early. Further, the freedom vector \((LEDIG_1^T)\) differs widely for such activities, or the number of hours is low. The mentioned sets of activities are a practical simplification of the extensive search process necessary to investigate all SCs. (In practice points 1, 3, and 4 above are the most important.)

A data structure simplifying the search for TSCs will be defined. Figure 8.1 outlines how the TCs are represented initially in KOMB.

KOMB consists of the two matrices KLF and LLF, and is a collective term for them (it is technically simple to define such a data structure). KLF and LLF are in their turn constructed of several parts:

**KLF**

1. There is external reference to CL classes (student groups). The CL first vectors of KLF are the activity sets for the various classes.

2. For each stream of a streamed structure all activities in which at least one student group from the stream is included, are represented in a vector.

3. For each room type a vector is defined so that all activities using the room type are included. The number of "stream vectors" and "room type vectors" is CL1-Cl, and these are ordinarily not TCs.

4. KLF has a given upper limit CLM. Vectors \(KLF_{CL1+1}^{CLM}\) may be used for storing new TSCs.

**LLF**

5. A school consists of AL teachers, and the AL first vectors of LLF are the activity sets for the various teachers.

6. Direct resource conflicts due to room type requirements are stored immediately thereafter, and the number of such TCs is AL1-AL.
<table>
<thead>
<tr>
<th>KOMB</th>
<th>N</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLF</td>
<td>1</td>
<td>Direct resource conflicts due to common class (student group)</td>
</tr>
<tr>
<td>CL</td>
<td>2</td>
<td>Activities belonging to same stream</td>
</tr>
<tr>
<td>CL1</td>
<td>3</td>
<td>Activities using the same room types</td>
</tr>
<tr>
<td>CLM</td>
<td>4</td>
<td>Auxiliary area</td>
</tr>
<tr>
<td>LLF</td>
<td>5</td>
<td>Direct resource conflicts due to common teacher</td>
</tr>
<tr>
<td>CLM+AL</td>
<td>6</td>
<td>Direct resource conflicts due to common room type</td>
</tr>
<tr>
<td>CLM+ALL</td>
<td>7</td>
<td>TCs of higher order</td>
</tr>
<tr>
<td>CLM+AL2</td>
<td>8</td>
<td>Auxiliary area</td>
</tr>
</tbody>
</table>

Figure 8.1
7. TCs of a higher order are generated from a selection of the total set of activities so that the activities with a large number of resources are included. These TCs are stored following the room type TCs, and the number is AL2-AL1. (For many school structures it is unnecessary to define TCs of a higher order.)

8. LLF has a given upper limit ALM. Vectors $LLF_{AL2+1}$ - $LLF_{ALM}$ are used for storing new TSCs. The vertical dimension of $KOMB$ is $C = CLM + ALM$.

The physical representation of $KOMB$ differs from the previously idealized matrix. This is done in order to reduce storage space, and because definite limits are used for representation of data. A number of row vectors in $KOMB$ are not TCs, but these will simplify the algorithms in order to find new TSCs. Vector $W10$ defines which vectors are not (current) TCs:

$$(8.12) \quad W10_i := \begin{cases} 1, \text{if } TTR_{i1} \leq 1, \text{ or if } KOMB_i \text{ is no TC (or TSC)} \\ 0, \text{otherwise} \end{cases}$$

From figure 8.1 is found that the initial value of $W10$ is:

$$(8.12) \quad W10^0_i := \begin{cases} \hat{0}, \text{if } (1 \leq CL) \vee (CLM < i \leq CLM + AL2) \\ 1, \text{otherwise} \end{cases}$$

In the previous chapter $W32$ defines those vectors in $KOMB$ which get new blockings, and one may say:

$$(8.13) \quad \begin{cases} W33 := (W32 + W10) \wedge W9 \\ W34 := (W32 \wedge W10) + W33 \end{cases}$$

$W33$ will be the TTCs getting new blockings, and $W34$ the STCs getting new blockings.

Arithmetic matrix $KL$ represents the hour numbers for important SCs:

$$(8.14) \quad KL_{ij} := TT(KLF_i \wedge LLF_j) = TT(KOMB_i \wedge KOMB_j + CLM)$$

Even if $KLF_i$ is no TC, $KLF_i \wedge LLF_j$ is of course an SC. $KL$ is outlined in figure 8.2.
<table>
<thead>
<tr>
<th>KL</th>
<th>Number of hours for activities with one common class and one common teacher</th>
<th>Number of hours for activities with common class and common room type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL</td>
<td>Number of hours for activities with common stream and common teacher</td>
<td>Number of hours for activities with common stream and common room type</td>
</tr>
<tr>
<td>CL1</td>
<td>Number of hours for activities with common room type and common teacher</td>
<td>Uninteresting, since ( LLF_j \notin KL_j )</td>
</tr>
</tbody>
</table>

Figure 8.2

KL is a consequence of the experiences made concerning which SCs will most probably become tight. KL is an expansion of the data structure known in other literature as a requirement-matrix. An extensive calculation is necessary to evaluate G for those SCs belonging to the components in KL and which are getting new conflicts. A logical matrix KLTK limits this calculation:

\[
(8.15) \quad KLTK_{ij}^{CLM} = \begin{cases} 
1, & \text{if the SC belonging to } KL_{ij} \text{ can give a } \\
& \text{TSC leading to limitation of solution space} \\
0, & \text{otherwise}
\end{cases}
\]

(For practical reasons, it is simpler to connect \( KL_{ij} \) with \( KLTK_{ij}^{CLM} \) than with \( KLTK_{ij}' \).)

Initially the following components in \( KLTK_{ij} \) have the value 0:

\[
(8.16) \begin{cases} 
\text{a. If } j \leq CLM \\
\text{b. If } |KOMB_1 \land KOMB_2| \leq 1 \\
\text{c. If } (KOMB_1 \in KOMB_2) \lor (KOMB_2 \in KOMB_1) = 1
\end{cases}
\]
8.2.1 Modification of data structure

a. Activity sets where activity_k is included

\[
\begin{align*}
TTR^{S+1}_{i1} &= TTR^{S}_{i1} - PL_k \\
K_{i j-CLM}^{S+1} &= K_{i j-CLM}^S - PL_k \\
W_{i j} &= 0 \text{ if } TTR^{S+1}_{i1} = 0 \\
KLT_{i j}^{S+1} &= 0 \text{ if (8.16.b) or (8.16.c) is satisfied}
\end{align*}
\]

for \( i \in KOMB^T_k \) and \( j \in KOMB^T_k \)

b. Parameters for the TCs of the system

\[
\begin{align*}
TTR^{S+1}_{i2} &= TTR^{S}_{i2} - SP^{S+1}_{id}(LEDIG^d, KOMB_i) \\
SP^{S+1}_{id} &= G(LEDIG^d, KOMB_i) \\
SP10^{S+1}_{id} &= |LEDIG^d \wedge KOMB_i| \\
W_{i} &= 1 \text{ if } TTR^{S+1}_{i1} \geq TTR^{S+1}_{i2}
\end{align*}
\]

for \( i \in (W32+W10) \)

If \( W3^S = 0 \) and \( SP1^S + SP2^S > 1 \), then \( G(LEDIG, KOMB_i) \) must be re-calculated for \( i \in KOMB^T_k \), since these TCs have got new blockings for other days than day \( d \). (I.e., (8.18.b) and (8.18.c) are done for \( d=1,2,\ldots\) UD and \( TTR^{S+1}_{i2} = |SP3^{S+1}1| \).)

New activities may be forced to day \( d \) if:

\[
(8.19) \quad SP3^{S+1}_{id} \geq SP12^{S+1}_{id} + TD(DDF_d \wedge KOMB_i) \quad \text{where } i \in W33
\]

If (8.19) is satisfied due to allocation \( s \), forced day assignments are represented by DTV:

\[
(8.20) \quad DTV_d := DTV_d V (DDF_d \wedge KOMB_i)
\]

If the TC getting forced day assignments due to (8.19) has no freedom loss for day \( d \) (i.e., \( SP3_{id} = SP10_{id} \)), these activities will be given priority for the following allocations:

\[
(8.21) \quad W2 := W2 V ((W1_{id} V DDF_d) \wedge KOMB_i)
\]
C. Generating of new TSCs

G is evaluated for each allocation for the corresponding SCs of 
Klij-CLM satisfying:

\[
\begin{align*}
\text{a. } & \ KLT_{ij} = 1 \\
\text{b. } & \ W32_i \land W32_j \geq 1 \\
\text{c. } & \ (KOMB_i \land KOMB_j) \land W30 \geq 1
\end{align*}
\]

(8.22)

G is calculated from (5.87), and if \( G(\text{LEDIG}, KOMB_i \land KOMB_j) \geq Klij-CLM \) 
this SC is tight or broken, and the data structure is modified:

The first value in W10 higher than CLM+AL2 with value 1 
is called \( t \)

\[
\begin{align*}
\text{a. } & \ KLT_{ij} := 0 \\
\text{b. } & \ W9_t := 1 \\
\text{c. } & \ W10_t := 0
\end{align*}
\]

(8.23)

\[
\begin{align*}
\text{d. } & \ KOMB_t := KOMB_i \land KOMB_j \\
\text{e. Evaluation of } G \text{ from (5.87) gives as intermediate results} \\
\text{the initial value of } SP3_t \text{ and } SP10_t \\
\text{f. } & \ SP12 \text{ for } d=1,2,---UD
\end{align*}
\]

(8.19) may be satisfied for KOMB_t and the data structure further modified due to (8.20) and (8.21). For the future allocations KOMB_t will be considered as an independent TTC.

If (8.22) is satisfied for a large number of values, the calculation of (5.87) becomes time-consuming. This calculation may be reduced if, for each step of the allocation, one modifies and stores the utility for each day of the SCs defined by KL and KLT. Instead of (5.87) a simpler expression analogous to (8.18.a) could then be used. This requires much storage space, and has not been done in the present program.

d. An important special case for new TSCs

Most of the new blockings arise for those time units A, where an 
allocation is being done. Those TTCs where activity k is not inclu-
ded and which have got new blockings must still be available for
these time units. This set of TTCs is called q1, and the activities still available for time units A are called q2.

\[
q1 := W9 \land (W32 \leftrightarrow \text{KOMB}_k^T)
\]
\[
q2 := \bigvee_{j \in A} (\text{LEDIG}_j^{S+1})
\]

It may be assumed for TC, \(i \in q1\):

The SC most likely to become tight due to the actual allocation is:

\[
q^3_i := \text{KOMB}_i \neq q2,
\]

because it is the largest SC of KOMB\(_i\) which is definitely known to have got reduced utility.

This is important to practical structures: Assume KOMB\(_i\) are the activities of a class. This class is not included in activity k, but since KOMB\(_i\) has new blockings, probably one or several of the teachers teaching the class are included in activity k. Most school structures are based on an approximate class-teacher principle; i.e., for pedagogical reasons each class is taught by a small number of teachers. A subset of these teachers is now completely blocked for the time units A. The probability is therefore increased for the freedom vector being equal for the activities where class i is included together with the mentioned subset of teachers.

It is not always useful to represent q^3\(_i\) as a new TSC. That is only done in those cases where q^3\(_i\) may give limitations not apparent from KOMB\(_i\). One sets:

\[
q^4_i := \text{LEDIG A (KOMB}_i + q^3_i)
\]
\[
q^5_i := \text{LEDIG A q^3}_i
\]

Assume:

\[
q^4_i \land q^5_i \neq 1
\]

\[
|q^4_i + q^5_i| \geq T(T(KOMB_i + q^3_i))
\]

Accordingly, it may be concluded:

Activities KOMB\(_i\) + q^3\(_i\) cannot be allocated to interval q^4\(_i\) \land q^5\(_i\), since KOMB\(_i\) is a TTC. (The activities have illusory freedom for this interval.) q^3\(_i\) will be a TSC.
This may be represented in two ways:

1. \( \text{KOMB}_i + q^3 \) gets forced assignment to \( q^4 + q^5 \). The allocation sequence is modified in order to give priority to these activities; i.e., \( W_2 := W_2 V (\text{KOMB}_i + q^3) \).

2. \( q^3 \) is represented as a new TSC by means of (8.23), (8.19), (8.20), and (8.21).

In practice the representation is chosen which has consequences for the smallest number of activities; i.e., 1 is chosen if \( \text{TT}(\text{KOMB}_i + q^3) \leq (\text{TT}(q^3)) \) and 2 otherwise.

The principle discussed above is an approximate way of securing existence of a schedule not only from the point of view of each individual resource, but also from the point of view of each individual time unit.

**E. Forced Day Assignments**

Activities forced to certain time units on the day of an actual allocation are represented by \( W_{31} \). These activities will be given priority for the future allocation:

\[
W_2 := W_2 V W_{31}
\]

Activities forced to certain days due to current allocation are represented by \( DTV \). This is a temporary matrix, and its consequences will be transferred to \( W_8, W_1, DDF, SP_1, \) and \( SP_2 \). New forced day assignments to day \( d \) are called \( q^d \) (where \( d = 1, 2, \ldots, UD \)).

\[
q^d := DTV_d + W_{11}^S
\]

The modification of the data structure will be:

\[
\begin{align*}
\text{a.} & \quad W_8^{S+1} := W_8^S V q^d \\
\text{b.} & \quad W_1^{S+1} := W_1^S V q^d \\
\text{c.} & \quad DDF_d^{S+1} := DDF_d^S + q^d \\
\text{d.} & \quad SP_{12}^{S+1} := SP_{12}^S - 1 \\
\text{e.} & \quad SP_{16}^{S+1} := SP_{16}^S + 1 \quad \text{for} \quad i \in q^d \\
\text{f.} & \quad SP_{110}^{S+1} := SP_{110}^S - 1 \\
\text{g.} & \quad SP_{12}^{S+1} := SP_{12}^S + TD(\text{KOMB}_j A q^d) \text{ where } j \in V, \text{KOMB}^T \quad \text{for} \quad q^T_d \\
\end{align*}
\]
If \( q_1^d \neq Wl=1 \), some activities which are included in day conflicts have got forced assignments, and a period of these activities is eliminated from corresponding day-conflicts simultaneously with other activities in these day conflicts being blocked on day \( d \):

\[
q_2^d := \bigvee_{j \in \text{WL} \cap q_1^d} DABL_j^T
\]
i.e., \( q_2^d \) are the day conflicts where some activities must be blocked on day \( d \).

\[
\begin{cases}
\text{LEDIG}_{j}^{S+1} := \text{LEDIG}_{j}^{S} + (\bigvee_{i \in q_2^d} DABL_i) \\
\text{for all } j \text{ where MOD}((j-1), Ud) + 1 = d
\end{cases}
\]

\((8.25)^{k}\)

\[
\text{DDFV}_{i1}^{S+1} := \text{DDFV}_{i1}^{S} - 1
\]

\[
\text{DDFV}_{i2}^{S+1} := \text{DDFV}_{i2}^{S} - 1
\]

\((8.25){k}\) modifies the freedom picture, and the previous relations in chapter 8 must be done iteratively. Technically, considerable simplifications can be done compared to a complete iteration. This is not discussed in detail. \((8.25){k}\) leads to the conflict picture due to the chosen allocation becoming more complete, and experience has shown that these new conflicts only exceptionally have unfortunate consequences for the global conditions.

\( ^{k}(8.25) \) assumes that \( q_1^d \) is consistent with regard to day conflicts; i.e., \(|\text{DABL}_j \cap q_1^d|=1 \) for \( j \in q_2 \). If this is not the case, special administrative cleaning up-processes must be done.
9. DESIRABLE REQUIREMENTS

Summary

Desirable requirements must necessarily be treated in a more heuristic way than the absolute requirements. The best principle has proved to be a representation of desirable requirements as patterns (or masks) with a relative ranking. These patterns are then employed to limit the solution space more than was possible from global conditions. The most important desirable requirements have consequences for the allocation sequence. The actual program design for treating desirable requirements depends on the school structure, but the main principle is common to all school structures.

In chapter 7 it was mentioned that local "optimization criteria" are combined with desirable requirements to determine the choice of period, and this will be discussed more closely in the following. Global conditions only offer a necessary limitation of the solution space; the consequences thereof, are that a method treating qualitative criteria must:

a. Preserve the existence of a solution due to absolute requirements.

b. Satisfy the desirable requirements in the best way possible in the final schedule.

Two different principles seem natural for fulfilment of this:

1. One attempts to quantify the qualitative criteria, and a weight is assigned to the different values. This renders a basis for defining an optimization function, from which a period may be chosen.

2. The desirable requirements are employed to limit the solution space more than was possible from global conditions; i.e., one finds the subset of periods satisfying the largest number of desirable requirements, and uses the local optimization mentioned in chapter 7 to distinguish between these.

The most obvious principle is probably 1, and such an optimization function is mentioned in other literature as well. However, there are a number of serious objections to it, and in connection with the
strategy discussed here, this principle was early abandoned. Some of the most important objections are noted below:

a. It is very difficult to quantify the desirable requirements, and the weights for the various criteria will vary considerably from one school structure to another. (In fact, they will vary considerably from year to year for the same school.)

b. It will be difficult to distinguish between the local optimization, which is done to be able to preserve the existence of a solution, and the regard for the desirable requirements. On the whole, the same instabilities as for functions of the type (7.11) will occur.

c. It will be difficult to prescribe certain properties to be satisfied in the final schedule.

d. The subjective evaluation of which schedule is the best, makes it still more difficult to undertake a sensible quantification.

The list of the arguments above may be completed, and principle 2 is considered as the best one. This will be justified by pointing out characteristic properties of desirable requirements.

A common feature of most desirable requirements is that one cannot beforehand give an unambiguous definition of which periods are most desirable for the individual activity, since a desirable requirement as a rule is a condition connected with the relative time sequence for sets of activities. Such a partial schedule is called a pattern, and from the set of possible patterns for more closely defined sets of activities one can describe qualitatively which patterns are more desirable than others. The sets of activities which ought to satisfy desirable requirements are ordinarily characterized by one or several resources being common for all activities in the set. It may also happen that the activities of a set represent the same type of education; anyhow, by means of simple references it is easy to define the actual sets of activities. Depending on the actual kind of desirable requirement, one may describe qualitatively the properties which the patterns of the corresponding sets of activities should have. This may be one or several of the following properties:

1. It is undesirable (or desirable) that a set of activities is allocated to certain time units.
2. A set of activities should be allocated as sequentially as possible.

3. Certain sets of activities should be allocated as parallel as possible (the various sets of activities must then not be conflicting).

4. A set of activities should be allocated such that a given number of days are not used. (Alternatively it may be desirable not to use a number of half-days.)

5. A set of activities should be allocated such that the activities are evenly distributed on the various days.

6. A set of activities should be allocated such that the periods for the individual activity are evenly distributed to good and bad education hours.

The individual qualitative statement may be given a ranking; i.e., state how strong a requirement is. Assume for instance that certain time units are undesirable for a set of activities. This may be formulated as an absolute requirement (e.g. if, as an initial condition, it is defined that a resource cannot be used for certain time units), or one might say that some time units are highly undesirable, undesirable, etc. Furthermore, the different qualitative statements may mutually be given a ranking. This ranking is based on the practical motivation for the various requirements, and the ranking will vary from school structure to school structure. Although there are a large number of practical reasons for the various desires, these may be transformed to a small number of qualitative statements. One could, of course, attempt to quantify these; which would, however, imply an artificial formalism, and for the problem area in question where the absolute requirements are dominating, this is meaningless. One limits the task to finding a suitable representation of the different desirable patterns and their relative ranking.

It is assumed that the various desires may be ranked in X types depending on the strength of the desire. A logical matrix SF with dimension [X, n] is defined:

\[
(9.1) \quad SF_{xj} = \begin{cases} 
1, & \text{if activity } j \text{ is included in a desired pattern of type } x \\
0, & \text{otherwise}
\end{cases}
\]
For each of the X types is defined a data structure $WISH^X$ with dimension $[N,TMAX]$:

\[
WISH^X_{ij} = \begin{cases} 
1, & \text{if time unit } i \text{ is desired for activity } j, \text{ and} \\
& \text{the ranking of this desire is } x \\
0, & \text{otherwise}
\end{cases}
\]

(The matrices $WISH^X$ are relatively large, and are stored in a secondary storage (drum).)

The desirable requirements are classified as implicit or explicit requirements, as follows:

**Implicit desirable requirement**: These are desires common to the schools belonging to the school structure in question. The allocation strategy considers such desires, although they are not mentioned especially in the problem specification (the desire for an even workload for students is an example of such a requirement).

**Explicit desirable requirement**: These are desires specially relevant for the actual school, and they must be stated in the problem definition (e.g. certain hours are undesired by some teachers).

In [1] is discussed the possibilities a school has for specifying desirable requirements. It is simple to allow still more specification possibilities, but experience has shown that schools tend to list their desires without realistic proportions. Accordingly, one should strive to use mainly implicit desires. This is particularly valid for desires in connection with patterns. If, however, the desires are connected with certain time units, it is simpler to let the schools define them.

The problem definition and the implicit desirable requirements initialize SP and $WISH^X$ (for $x=1,2,\ldots,X$). For each allocation these data structures are modified, as the number of possible desirable patterns for the individual set of activities is reduced. A desirable requirement is, as mentioned above, ordinarily connected to sets of activities and not to the individual activity, and it is perfectly possible that this is being included in several desirable requirements. One may find that the corresponding patterns are not in accordance with each other. For the moment possible weaknesses of the mentioned data structures are disregarded, and it will be shown how they are employed to limit the solution space.
Assume: Activity \( k \) will be allocated to one of the periods \( q_1 \) belonging to days \( q_2 \) and represented by the physical time units \( q_3 \). Desirable time units in \( \text{WISH}^1 \) have higher priority than desirable time units in \( \text{WISH}^2 \), etc.

The subset of time units from \( q_3 \) satisfying the largest number of desirable patterns is given by:

\[
q^4 := q_3 \land ( \land_{x \in SF^T_k} \text{WISH}^x_k )
\]

There is no guarantee that \( q^4 \) is unequal to the 0-vector. One will provide for this being satisfied, which is done by disregarding the desirable patterns with lowest priority. A vector \( q_5 \) is defined as follows:

\[
q_5^x := \begin{cases} 
1, & \text{if } SF^T_{xk} \equiv 1 \text{ and } q^3 \land ( \land_{i \in q_5 \text{ and } i \leq x} \text{WISH}^i_k ) \equiv 1 \\
0, & \text{otherwise}
\end{cases}
\]

(9.3)

Note that the components of \( q_5 \) are iteratively defined. Any new component in \( q_5 \) depends on \( q_3 \) and the form of desirable patterns with higher priority. This ensures that the most important desirable requirements are considered first. One sets:

\[
q^4 := q_3 \land ( \land_{x \in q_5^T_k} \text{WISH}^x_k )
\]

(9.4)

\( q^4 \) will be the time units which correspond best to the patterns in which activity \( k \) should be included. If the period length of activity \( k \) is 1, then \( q^4 \) is identical with the most desirable periods for \( k \); otherwise one finds the subset of \( q_1 \) whose corresponding physical time units are identical with \( q^4 \). (It will usually suffice to require that the physical time units for each desirable period must have coincidence with \( q^4 \).)

In other words, the various desirable requirements are represented as patterns (or masks), which limit the allocation possibilities \( q_1 \) by means of (9.3) and (9.4). \( q_1 \) is divided into two vectors \( q_1' \) and \( q_1'' \), so that \( q_1' \) is (approximately) represented by \( q^4 \) and \( q_1'' \) by \( q^3 \land q^4 \). One attempts now to allocate activity \( k \) to the period in \( q_1' \) which is consistent with the global conditions and minimizes (7.13). If none of the \( q_1' \) periods are in accordance with the global conditions, \( q_3 \) is replaced by \( q^3 \land q^4 \) in relations (9.3) and (9.4), and
one finds the subset of qi" satisfying the desirable patterns in the best way, etc. In other words, relations (9.3) and (9.4) are used iteratively. In this way one will eventually find a period satisfying the global conditions, and activity k is then allocated to this period. (If one cannot find a period consistent with the global conditions, one will in some cases prohibit allocation of a period from activity k, or choose a period for activity k which later will create an error situation. This depends on which global conditions are not fulfilled.)

It is in the nature of the problem that the principles employed for the desirable requirements are necessarily the most intuitively (or heuristically) defined part of the entire strategy. Furthermore, experience has shown that the importance and design of the desirable requirements may vary for the different school structures (i.e., the importance of the desirable requirements depends on the freedom present in the system due to the absolute requirements. However, it is not evident from this that it is easier to find an acceptable schedule for a simple school structure, since in such a case one will tighten the problem by defining a large set of desirable requirements. Further, simple school structures (unfortunately) offer a larger opportunity for subjective evaluations, and in many ways the problem is simpler if there is only the question of finding a schedule at all, than if the question is finding the "best" solution from a considerable number of possibilities.).

In principle there are several objections to the mentioned method, and below a summary evaluation is listed:

1. No unambiguously defined optimization function for the desirable requirements is used. This is considered as an advantage, since the concept "optimal schedule" is regarded as a fiction. Instead one tries to make the final schedule so that certain partial schedules are included in qualitatively defined patterns. This is equivalent to the evaluation used in practice, where each section of the schedule is separately evaluated. A small number of qualitative parameters are easily quantified; e.g., utilization of heavily used resources, intermediate hours for the teachers, the number of activities which are allocated to undesirable hours, etc. Such requirements are often formulated as absolute requirements,
and those sets of desirable requirements which are given highest priority are approximately quantified, since they are satisfied provided that it is at all possible. (There is of course a practical limit to how many desirable requirements may be given highest priority.)

2. Representation of the desirable requirements is simple, and so is the calculation to find the most desirable time units. For each step in the allocation is found firstly which of the possible periods corresponds best to the desired patterns, and secondly the accordance with global conditions for a subset of the possible periods is evaluated. Such an organization drastically reduces necessary computing time (in other words, one attempts to avoid calculation of the consequences of global conditions for undesirable periods).

3. The outlined data structure is very flexible, and one may easily add new desirable requirements or neglect the effect of certain desirable requirements. It has been pointed out that the desirable requirements may be formulated very similarly, although the practical motivation for the different requirements vary greatly.

4. The limitations of the solution space done by means of (9.3) and (9.4) are denoted masking. This limitation is obviously artificial from the point of view of absolute conflicts alone, and the minimal value of (7.13) in only found for a subset of the possible periods. The probability is thereby increased for error allocation due to the absolute requirements. This weakness will be common for any method which considers desirable requirements. An advantage of the outlined method is that absolute and desirable requirements are clearly differentiated. Should it turn out that the total strategy gives unsatisfactory results for certain school structures, it will thus be simple to find which conditions are treated unsatisfactorily.

A comment on the danger of breaking absolute requirements: When regarding tight systems one must partly neglect the consideration for desirable requirements, whereas they play an important part for simple structures; i.e., one is in this case completely aware of the danger of breaking absolute requirements. The practical consequence is, accordingly, that a few activities might not be
allocated, but the actual partial schedule sufficiently considers desirable requirements. As a rule it is simple to modify the schedule of a simple school structure manually; the resulting manual work is usually modest. Alternatively one may find a schedule satisfying the absolute requirements, but not satisfactory for the desirable ones. The necessary manual adjustment work may easily become so extensive that such a schedule is of almost no value.

5. Previously it has not been discussed how the different desirable patterns are represented in WISH. It should be apparent that it is difficult to define unambiguously which time units are, from a global point of view, most desirable for sets of activities. A pattern partly depends on the prehistory of the current situation, and partly on unknown factors; i.e., future allocations. The most important weakness of the outlined principle is that the most desirable periods from a local evaluation are not necessarily the most desirable ones from a global evaluation. This will be shown by a simple example: Assume that it is desirable for activities i, j, and k to be allocated as sequentially as possible. Activity i is allocated to time unit 3 on day d. Activity k will now be allocated to either time unit 2 or 4 on day d. Locally evaluated, these possibilities are equal. It is assumed that independent of how activity k is allocated, activity j will later in the allocation be forced to time unit 5 on day d. Accordingly, time unit 4 is far better than time unit 2 for activity k, which is almost impossible to deduce from a local evaluation.

Innumerable examples analogous to the above may be constructed, which may be a serious objection. It must be admitted that from an operative point of view the regard for desirable requirements is an aspect where the current program system may function unsatisfactorily in some cases; this is particularly the case for simple school structures. This weakness can hardly be avoided, which shows the necessity of a certain manual adjustment. It must be added that desirable requirements are treated equally well, if not better, in computer made schedules than by a manual technique. However, from an esthetic viewpoint it is unsatisfactory that desirable requirements are not always dealt with correctly. Till now, one has mainly worked with complicated school structures, and
the mentioned masking technique functions amazingly well for such structures. The desirable requirements play a subordinate role here, and a local evaluation of desirable requirements is preferable to a more or less random allocation.

For simple structures one has till now used a considerable amount of explicit requirements, often connected to definite time units, to direct the strategy. This is in some ways an artificial limitation of possible solutions, but it is justifiable for simple structures. Furthermore, the optimization (7.13) may be disregarded; i.e., existence of solution is assumed for any possibility consistent with global conditions due to absolute requirements. For these possibilities desirable requirements are used alone to find the "best" possibility. One has been working with defining some "global" conditions, in order to be able to consider the desirable requirements in the best way possible. This work has not been completed, and will not be discussed here. The question is, however, how useful is this when regarding computing time? (As mentioned before, for tight structures one cannot stress desirable requirements too much, and for simple structures one may easily make the necessary adjustments manually.)

Based on the initial conditions and explicit desirable requirements one may partially prescribe the frame of the various desirable patterns. The activities consisting of a large number of resources are probably included in a large number of desirable patterns. Such activities contain numerous conflict possibilities, and one risks breaking one of the absolute requirements if the solution space is limited too much due to desirable requirements. Accordingly, one often considers only the high priority-desires for such activities, and the activities are allocated early. This leads to the frame of the desirable patterns adjusting to the chosen assignment of multi-conflict activities.

6. As stated previously, it is simple to define desirable requirements, but there is obviously a practical limit as to how many requirements of this type can be taken into account. If desirable requirements are not defined critically, one may risk that they no longer correspond. For instance, from one point of view one might
want a set of activities to be as sequential as possible, whereas from another viewpoint one might want the same set to be evenly distributed throughout the week. If in addition these desires have equal priority, the result may be a very arbitrary allocation.

Another danger of having a large number of desires is that the most essential ones may be neglected, since the internal ranking becomes diffuse, and the concept "acceptable schedule" gets a vague definition. Therefore a selection is made from the possible desirable requirements; however, this may in its turn cause too stereotype an evaluation of the various possibilities. On the whole, manual technique is more flexible if the number of desirable requirements is large.

9.1 Examples of representation of desirable requirements
The original motivation for representing desirable requirements as a mask was due to one particular condition:

It has previously been assumed that the activities must be allocated within a definite time frame. This is, however, only approximately correct for practical structures. One does want all activities to be allocated within a definite time frame, but if absolutely necessary, other hours are acceptable for certain activities if no other solution can be found. (The student schedules must still satisfy the continuity requirement.)

This is a very strong desirable requirement, and parameter $SP_{k7}$ defines which time units are desirable for activity $k$, and this parameter is denoted the activity type. The consequences of this parameter depend of course on the actual time frame of a school structure. A few examples will be given:

A tight streamed school structure may have a 5-day week with six hours per day (i.e., each student should have 36 hours per week). The structure is assumed to be tight, and if absolutely necessary one has to accept one hour before and one hour after ordinary education time. A streamed school structure cannot tolerate too narrowly defined activity types, since that could result in a limitation of the solution space which easily conflicts with the absolute requirements.
A reasonable set of activity types for such a school structure is outlined in figure 9.1:

<table>
<thead>
<tr>
<th>( S_{1k7} )</th>
<th>Desired time units</th>
<th>Expansion possibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All; i.e., boundary hours are completely acceptable:</td>
<td>None.</td>
</tr>
<tr>
<td>1</td>
<td>Ordinary time frame; i.e., the six intermediate hours of each day.</td>
<td>0, i.e., boundary hours are accepted if absolutely necessary.</td>
</tr>
<tr>
<td>2</td>
<td>Ordinary time frame.</td>
<td>None.</td>
</tr>
<tr>
<td>3</td>
<td>Ordinary time frame. In addition is required that no double or triple hours are assigned over the lunch break. These activities are considered heavy, so that no student group may have more than one double or triple period of type 3 on any day.</td>
<td>None.</td>
</tr>
<tr>
<td>4</td>
<td>Ordinary time frame, except Monday.</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 9.1

A few comments regarding figure 9.1:
The left column indicates the possible values for \( S_{1k7} \), and the corresponding desirable time units are defined. If due to global conditions it should prove impossible to assign all activities within the desired time frame, it is expanded; i.e., the activity type is changed for the activities which cannot be allocated. These possibilities are shown in figure 9.1. If an activity type has no expansion possibilities, the result is that the actual activity will not be allocated when expansion is necessary. A useful classification of activities based on figure 9.1 is:

\[
S_{1k7} = \begin{cases} 
0 & \text{Initially no activities.} \\
1 & \text{All activities representing ability subjects like crafts, gymnastics, etc.} \\
2 & \text{Oral subjects.} \\
3 & \text{Written subjects. Double- and triple hours in such subjects for a student group must be day-blocked (represented by DABL), and LEDPER for these activities must be blocked for the periods lasting over the lunch break.} \\
4 & \text{Home-work subjects taught only once a week.} 
\end{cases}
\]
It is given beforehand which time units are within the desired pattern (time frame), and these are represented by means of WISH$^1$. Note that none of the activity types from figure 9.1 imply any considerable limitation of the solution space.

The university structure usually requires well differentiated activity types, which is justifiable if the structure is simple. Assume that such a structure consists of 11 hours per day, and figure 9.2 is an example of a set of activity types.

<table>
<thead>
<tr>
<th>$k_7$</th>
<th>Desired time units</th>
<th>Expansion possibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All hours</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>The 9 first hours each day</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>&quot; 7 &quot; &quot; &quot; &quot; &quot; &quot;</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>&quot; 4 &quot; &quot; &quot; &quot; &quot; &quot;</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>&quot; 2 &quot; &quot; &quot; &quot; &quot; &quot;</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Hours 3-7 incl. &quot; &quot;</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>The 7 last hours  &quot; &quot;</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>&quot; 4 &quot; &quot; &quot; &quot; &quot; &quot;</td>
<td>6</td>
</tr>
</tbody>
</table>

*Figure 9.2*

The activity type is gradually modified; e.g. assume activity $k$ being of type 4 (from figure 9.2). If this cannot be allocated to the two first hours, the type is transformed to 3, and if activity $k$ cannot be allocated to the 4 first hours of any day either, the type is transformed to 2, etc. The classification of activities in activity types is determined partly empirically, and if desirable time units for the activities are too narrowly defined, the final result will be unsatisfactory.

The part of the program treating consequences of the activity type (expansion rules etc.) may be special for each school structure. This is however a simple procedure.

Possible time units for an activity are given by LEDIG, and the combination of WISH$^1$ and LEDIG gives the most desirable time units for an activity. These time units may be represented in a data structure LEDIG$^1$: 
Similarly, new data structures LEDPER' and DDF' for respectively desirable periods and desirable days may be defined. In the current program only DDF' is represented as a data structure of its own. The important parameters for describing the most desirable time units are however given by SPL and SP2:

In chapter 5.3. SP2 defines the number of possible periods for the activities on the various days. It is usually more convenient to define SP2 as the number of most desirable periods on the different days from the freedom picture defined by (9.5). Correspondingly, SPL^T_k defines the degrees of freedom for the activities so that only the most desirable periods are evaluated. SPL^T_k defines the number of available desirable days for the activities. SPL^T_k defines in which way the degrees of freedom for the activities will be determined:

\[
\text{SPL}^T_k = \begin{cases} 
0, & \text{if degree of freedom for activity } k \text{ is determined from possible periods.} \\
1, & \text{if degree of freedom for activity } k \text{ is determined from the most desirable periods}
\end{cases}
\]

The consequences of the modified data structure will be:

The most desirable periods guide the rules for the allocation sequence; i.e., when an activity will be allocated, whereas the possible time units are used for finding which periods are acceptable with regard to the global conditions.

This rather heuristic rule has proved valuable for consideration of desirable requirements, but it must of course be used with moderation. The use of the rule has been based on the following assumptions:

The desired time units from WISH_k naturally represent important desirable requirements. It is assumed that the number of these is small, and that these desirable requirements are connected with certain time units (and that they do not limit the solution space too much). Forced allocations to certain time units are also represented by WISH_k. For simple school structures WISH_k will be allowed to make a considerable limitation of the solution space.

A large number of school structures will use the activity types defined by figure 9.1. For these structures it has proved most
suitable to determine possible time units within the ordinary time frame; i.e., boundary hours are eliminated when examining global conditions, and they are only used as a kind of "buffer" for impossible situations.

Desirable requirements given as desired patterns will be discussed. The dominating influence of the absolute requirements necessitates a simplified treatment of such requirements. The set of desirable requirements and the part of the program representing them, will vary for the different school structures. These are, however, rather simple procedures, which will not be specially discussed here. In figure 9.3 is given a verbal description of some desirable requirements and how they modify WISH\textsuperscript{1} --- WISH\textsuperscript{X} for each allocation. For these examples it is assumed that the desirable requirements are ranked on 4 levels, and that activity k is allocated to period p on day d after s steps in the allocation.

<table>
<thead>
<tr>
<th>Examples of implicit desirable requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Desirable requirement</strong></td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>1. The periods for each activity ought to be evenly distributed throughout the week</td>
</tr>
<tr>
<td>2. No activity should get only bad hours</td>
</tr>
<tr>
<td>3. Activities should not be allocated to time units where it becomes necessary to use unwanted room alternatives</td>
</tr>
<tr>
<td>Desirable requirements</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>4. Easy subjects for a student group should preferably not be concentrated</td>
</tr>
<tr>
<td>5. The teachers should not have an uneven work load. (For student groups this requirement is absolute.)</td>
</tr>
</tbody>
</table>

Figure 9.3

The explicit desirable requirements connected with certain time units are initially defined, and their ranking is given (this is discussed in [1]). The explicit desirable requirements defined as patterns may be transformed to one of the more general qualitative statements mentioned on pp. 185 and 186. (It would lead too far to discuss practical motivations for such patterns.) If activity \( k \) is included in desirable patterns, the matrices' WISH are modified. This is done analogously to figure 9.3. In other words, the time units to be included in desired patterns are modified step by step.

Practical experience with the mentioned method will be mentioned briefly: For structures with a reasonable degree of difficulty the program manages to sufficiently consider desires connected with certain time units, but desires connected with patterns might be treated rather randomly if they have low priority. For simple school structures it is acceptable that the sets of activities which are included in patterns are allocated early, thereby getting better results. It should be self-evident that the strategy may be improved to increase the consideration for desirable patterns. The future will show to which degree this is useful (or possible).

One desirable requirement is treated in a somewhat special way: The requirements for the individual teacher schedule may vary, but as a
rule the number of intermediate hours should be minimized. (The analogous condition for the student schedules is usually absolute by means of a continuity requirement.) It is assumed that the limitations of the solution space done by means of (9.3) and (9.4) imply that the remaining periods are equal with regard to desirable requirements, except for the wish to minimize the number of times teachers are available. This desire is easily quantified, and is used for a further differentiation of the current periods q1':

Assume that mt_j^S are intermediate hours for teacher j prior to allocation s, and that mt_j^{s+1} are intermediate hours for teacher j after allocation s; assume further that q6 are the teachers included in the activity to be allocated. MT symbolises the square sum of new intermediate hours arising because of allocation s:

\[
MT := \sum_{j \in q6} (mt_j^{s+1} - mt_j^S)^2 \cdot \delta_j
\]

where \( \delta_j := \begin{cases} 
1, & \text{if } mt_j^{s+1} - mt_j^S \geq 0 \\
-1, & \text{otherwise}
\end{cases} \)

Under otherwise equal conditions the period is chosen which minimizes MT (one may safely assume that the "discontent" increases at least in square proportion to the number of new intermediate hours). This further differentiation of the possible periods may have considerable consequences for computing time. The various possibilities are examined sequentially according to their fulfilment of desirable requirements. The first period consistent with the global conditions and with a freedom loss of 0 (from (7.13)), is chosen without examination of other alternatives, thus causing a considerable reduction of computing time. (For simple school structures the first period consistent with the global conditions may be accepted, or a period is accepted if the freedom loss is less than a given value.)

The treatment of the desirable requirements shows that only to a small degree is evaluated how well one may satisfy desires for activities not yet allocated. Due to the absolute requirements this should obviously not be stressed too much.
10. SPECIAL CONFLICT TYPES

Summary

A summary mention is given of some conflict types representing a selection of the more particular conditions found in practice. The essential point is to show existing conflict types, and how they fit into the general strategy by means of the following principles:

a. generation of more complete conflict pictures,
b. expansion of the set of necessary global conditions, and
c. modification of the rules for the allocation sequence.

The treatment of a conflict type depends on how "critical" it is. If the condition has consequences for a small number of activities, it is simplest to do a preassignment so that the prehistory of the actual problem satisfies special conditions. This possibility is used to a certain degree, but much preassignment may become time consuming and lead to an unnecessary limitation of the solution space. A number of conditions have consequences for all activities so that a preassignment becomes meaningless.

Too many special conditions increase the probability that interaction between the conflicts results in the relations in chapter 5 limiting the solution space insufficiently. A large number of absolute requirements may further imply a self-contradictory problem specification. Accordingly, there is a practical limit as to how many conditions can be treated simultaneously, a limit which partly must be determined empirically.

It is assumed that the conflicts arising due to special conditions are few compared to the central conditions. Usually this is valid, and the result thereof is that special conditions may be adapted to the frame mentioned in chapter 5, and that interactions between new conditions may partly be neglected.

10.1 Special complex conflicts

A complex resource conflict is a statement of how many activities from a given set of activities can be allocated to the same time unit. Naturally complex conflicts may be formulated for sets of time units; e.g.:
1. **Complex day conflict.** For certain subsets of activities in which a certain resource (ordinarily a class) is included, only a certain number of activities can be allocated to the same day. The motive for such a requirement is to secure an even work load.

2. **Intermission requirement.** Within certain time intervals it is required that each student (and teacher) has at least one hour available for lunch break, and that the number of students (i.e. classes) lunching simultaneously must be smaller than a given figure to avoid crowding of the cafeteria.

If these kinds of requirements are formulated as desirable, then administrative routines are defined, keeping track of which periods might become undesirable for the various activities, and this is gradually represented by means of the data structure \( WISH^1, \ldots, WISH^X \). If, however, the requirements are absolute, then the consequences must be represented by the conflict pictures for each step.

Assume that the TCs defined by vector \( q_1 \) are included in one complex day conflict, and that \( q_2 \) are the activities included in such conflicts. The figure \( d_{k_1} \) denotes how many activities from \( q_2 \) may be allocated to the same day due to TC, \( i \). When allocating activity \( k \) to day \( d \) and \( k \in q_2 \), and \( W_{k, 0} = 0 \), a new necessary condition is:

\[
\text{DT} \quad (\bigvee_{j=1}^{PT} (\text{PFT}_j^d) \lor \text{W1l}_d) \land (\text{KOMB}_k^1 \land q_2) \leq d_{k_1} - 1
\]

where \( i \in (\text{KOMB}_k^T \land q_1) \)

In each case where the equality of (10.1) is satisfied, the corresponding activities included in \( q_2 \) must be blocked for day \( d \). This completing of the conflict picture for each step secures that the final schedule satisfies the complex day conflicts. (It is simple to expand (10.1) in order to cover the possibility of a TC being included in several complex day conflicts.)

The intermission requirement may be completely eliminated in the following way: For each student group (and possibly teacher) a "fictitious" activity representing the lunch break is defined. These activities consist of the same number of periods as there are days in a week, and initially they get forced allocation to the time frame for the lunch break. The cafeteria is considered as an independent room
type consisting of the same number of "rooms" as the maximum number of student groups able to lunch simultaneously. Particular criteria may be defined for the assumptions for the assignment of a lunch break. The relations of chapter 5 will then cover the intermission requirement.

The mentioned principle is rather laborious, and may easily become time consuming. An alternative and simpler method of treating the intermission requirement, analogous to (10.1), will be discussed.

LEDIG_{b,d}^i is defined as the selection of LEDIG^d containing the time units where a lunch break may take place on day d. The number of time units on one day where a lunch break can take place is denoted bt, and the maximum number of students that can lunch simultaneously is denoted bm. Logical vector q3 defines the student groups (and teachers) which shall have lunch break, and arithmetic matrix CLB[bt,UD] keeps track of how many student groups have been assigned for lunch on the various time units.

Assume that activity k is allocated to day d. To simplify matters one may also assume that this allocation only has consequences for the intermission requirement of the resources included in activity k, and then only on the assumption that at least one of the physical time units for the period in question is also used as lunch break. A necessary condition for securing lunch break for all student groups will be:

\[(10.2) \quad |(\text{LEDIG}_{b,d}^i \land Q) \land \text{KOMB}_i^T) \geq 1\]

where \(Q_j := \begin{cases} 0\text{-vector if } \text{CLB}_{jd} \geq bm \\ 1\text{-vector, otherwise} \end{cases}\)

and \(i \in (q3 \land \text{KOMB}_k^T)\)

The possibilities not satisfying (10.2) are rejected, and if the equality of (10.2) is satisfied the conflict picture must be modified: \(q4\) denotes the set of TCs satisfying the equality of (10.2) and one states:

\[q5 := (\bigvee_{j=1}^{bt} \text{LEDIG}_{b,d}^j) \land (\bigvee_{i \in q4} \text{KOMB}_i)\]
The set of activities $q_5$ must be blocked completely for the time units used for lunch break on day $d$, and these conflicts spread to other time units of day $d$ by means of (5.83). This completing of the conflict picture implies that for future allocations it is unnecessary to examine (10.2) on day $d$ for those TCs where equality (10.2) is satisfied for the current allocation. (The final FFT will, due to the mentioned blocking, automatically provide for the intermission requirement on day $d$ being satisfied for these TCs.) For each allocation, matrix CLB is modified so that new student groups that have got their intermission requirement satisfied are added to the corresponding component in CLB. (10.2) is not as exact as the introduction of fictitious activities, but from a practical point of view it is just as satisfactory and considerably less complicated. One may even disregard the restrictions resulting from CLB; i.e., assume that the lunch breaks automatically are distributed evenly throughout the possible time units.

So far it has been presumed that direct conflicts for an activity are unambiguously given by the resources included in the activity, but for some school structures the following is valid:

One or several resources for certain activities are not utilized in all weeks.

The reason for this may be that certain education is given only bi-weekly, or that it is based on concentrated reading; i.e., during parts of the school year one subject is taught, while for the same time interval another subject is taught for the rest of the year. Often different resources are used for these subjects. Such activities are called X-activities, and the resources not utilized each week are called X-resources for the activity. X-activities do not have the same cycle as ordinary activities, and an alternative is expanding the time frame to encompass a two- or several week schedule. This is rather awkward, and may easily give errors since X-activities are implicitly included in an incidence relation:

Ordinary activities have cyclic education, and for X-activities one will require (desire) corresponding X-resources to be included as X-resources of other activities for the same time units.
This requirement is often a consequence of the necessity to secure maximum utility of the resources of the system; thus, the schedule is mainly the same for all weeks. (For administrative reasons it is also difficult to use schedules which differ substantially from one week to another.) For a schedule containing X-resources the same resource will apparently be included in several activities simultaneously; however, none of these occur in the same week. This is analogous to the requirements defined by means of a room type. The resources included in an activity as X-resources may be defined by direct resource requirements (student groups, teachers) or complex resource requirements (room types). Below is an example of the representation of conflicts due to X-activities:

Assume that an X-resource may be included in 2 activities on the same time unit. The X-activities of the system are given by vector XA. The conflict matrix, KOLMA, is still representing only direct resource conflicts, and the modification of KOLMA due to resource conflict KOMB_i is:

\[
KOLMA_j := \begin{cases} 
KOLMA_j \lor KOMB_i & \text{for } j \in (KOMB_i \lor XA) \\
KOLMA_j \lor (KOMB_i \lor XA) & \text{for } j \in (KOMB_i \land XA) \\
1 & \text{otherwise}
\end{cases}
\]

(10.3)

Example: KOMB_i: \[\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{array}\]
gives KOLMA: \[\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{array}\]

Assume that activity k is an X-activity and that q^k are the resources included in activity k as X-resources. Activity k will be allocated to time unit j, and the current conflict picture is given by for instance (5.58). TAPT^S_j must be modified in order to take X-resources into consideration:

q^6 is now symbolising the resources for activity k already allocated to time unit j as X-resources:

\[q^6 := (KOMB \land FPT^S_j) \land q^k\]

The resources q^6 are now completely blocked for time unit j; i.e.:

\[
TAPT^S_j := TAPT^S_j \lor ((\lor_{i \in q^6} (KOMB_i) \land LEDIG^S_j))
\]

(10.4)
An X-resource for activity k defined by a complex resource requirement (i.e., a room type) is treated analogously. This requires a particular data structure not discussed here; however, the main principle is:

1. The first time a room type is used as an X-resource for a time unit, the consequences are equal to those of an ordinary room type requirement, with the exception that the activities using the room type as an X-resource are not blocked.

2. The second time a room type is used as an X-resource for a time unit no rooms are blocked due to this requirement; however, activities containing the room type as an X-resource lose access to one room of the room type.

X-activities lead to a somewhat particular problem, since the number of hours for the TCs is ambiguously defined if X-activities are included. For the relations based on (5.20) the number of hours for a TC is defined as the minimal number of time units necessary to allocate the current TC; i.e.:

\[
(10.5) \quad TT(KOMB_i) = \sum_{j \in KOMB_i} SP_j \cdot FL_j - \text{MAX}(0, (X1-X2)/2)
\]

where:

- \(X1\) = Number of hours for X-activities of \(KOMB_i\) not yet allocated.
- \(X2\) = The number of time units for \(KOMB_i\) still available, and to which one of the activities \(KOMB_i\) \& XA is allocated.

(10.5) is valid provided that an X-resource may be used twice for the same time unit. Of course it is possible to design cases where X-activities result in certain necessary conditions becoming rather dubious. But a system containing X-resources is ordinarily specified in such a way that the number of hours for the student groups is unambiguously given initially, which makes it easy to treat X-activities. (The examined systems show that the X-activities are very well "packed" by the program.)

10.2 The continuity requirement

For school structures where the student schedules are not tight, this requirement might be just as dominating as the central conflict
types in chapter 5. An alternative way of dealing with the continuity requirement is to eliminate it in a similar way as the intermission requirement. This implies introducing a number of fictitious activities for all student groups, thus making those TCs tight. Furthermore, one sees to it that these fictitious activities are allocated to boundary hours every day. This idea is, however, rather unsatisfactory, since the fictitious activities necessarily must be given priority in the allocation to ensure their assignment to boundary hours. In this way the solution space is limited on the basis of deficient information, and in reality the same result is achieved by using a suitable pre-blocking as the initial condition. This is definitely an unnecessary limitation of the solution space, and the continuity requirement must be treated much more thoroughly.

Logical vector \( W41 \) defines which TCs shall satisfy a continuity requirement, and matrices \( DAYF[KM,UD] \) and \( DAYL[KM,UD] \) are defined as follows:

\[
(10.6) \quad \begin{cases}
DAYF_{id}^{:} = \text{First time unit on day } d \text{ which } KOMB_i \text{ must use} \\
DAYL_{id}^{:} = \text{Last time unit on day } d \text{ which } KOMB_i \text{ must use}
\end{cases}
\]

In the final schedule \( KOMB_i \) must coincide with \( FFT \) for all time units between the limits \( DAYF_{id} \) and \( DAYL_{id} \) for \( d=1,2,---,UD \). These matrices may partly be initialized so that TCs defined by \( W41 \) are forced to certain parts of the day. Otherwise \( DAYF \) and \( DAYL \) are modified for each step for the classes included in the activity being allocated. In the following it is assumed that activity \( k \) will be allocated to period \( ti \) on day \( d \):

\[
(10.7) \quad \begin{cases}
DAYF_{id}^{s+1} = \text{MIN}(DAYF_{id}^{s}, (ti-1)/UD+1) \\
DAYL_{id}^{s+1} = \text{MAX}(DAYL_{id}^{s}, (ti-1)/UD+PL_k)
\end{cases} \quad \text{for } i \in (W41 \land KOMB_k^T)
\]

In connection with these matrices, logical matrices \( PKT^d \) are defined for \( d=1,2,---,UD \):

\[
(10.8) \quad PKT_{ij}^{d,s} = \begin{cases}
1, \text{ if } KOMB_i \land LEDO_j^{d,s} \geq 0 \\
0, \text{ otherwise}
\end{cases}
\]

In other words, \( PKT \) defines the time units which are completely blocked for the various TCs due to the prehistory.
Assume that \( LEDIG_{d,s+1} \) is found for all conditions other than the continuity requirement. Modification of the conflict picture and new global conditions due to this requirement will be shown:

Assume that time unit \( j \) satisfies:

\[
\begin{align*}
& \{ \text{LEDIG}_{d,s+1} \} \times \text{KOMB}_i \equiv 0 \\
& \{ \text{FTI}_{d,s+1} \} \times \text{KOMB}_i \equiv 0 \\
& \text{for } i \in W_{41} \text{ and } PKT_{i,j}^{d,s} = 0
\end{align*}
\]

(10.9) shows that \( \text{KOMB}_i \) is completely blocked for time unit \( j \) due to the current allocation.

**a. Modification of the conflict picture:**

\[
\begin{align*}
& \text{LEDIG}_{d,s+1}^{p} = \text{LEDIG}_{d,s+1}^{p} \times \text{KOMB}_i \\
& \text{for } p = 1, 2, \ldots (j-1) \text{ if } j < \text{DAYF}_{s+1}^{id} \\
& \text{or for } p = (j+1) \ldots DT \text{ if } j > \text{DAYF}_{s+1}^{id}
\end{align*}
\]

(10.10) reduces the freedom picture in such a way that future allocations of activities belonging to \( \text{KOMB}_i \) cannot lead to a continuity break due to a complete blocking of time unit \( j \). (Note however that (10.10) may lead to a considerable freedom loss for \( \text{KOMB}_i \), thus other conditions, for instance (7.4) and (7.13), may possibly eliminate those possibilities where the conflicts due to (10.10) are unacceptable.)

**b. Necessary global conditions:**

b1. If \( \text{DAYF}_{s+1}^{id} \leq j \leq \text{DAYF}_{s+1}^{id} \), then a continuity break for \( \text{KOMB}_i \) will occur, and the possibility ti will have to be rejected. (In the case where all possibilities lead to a continuity break one attempts to assign the break close to the boundary of the day in order to simplify manual adjustment.)

b2. In chapter 7 a TTC and an STC were treated differently. An STC with a continuity requirement has a kind of intermediate position: \( LEDIG^{d,s} \) now denotes a selection of \( LEDIG^{S} \) so that the former represents time units in \( LEDIG^{S} \) where \( \text{KOMB}_i \) must coincide
with the final FFT due to the continuity requirement. LEDIG_{i,s} is found by means of DAYF_{i} and DAYL_{i}. q_{i,s} is a logical vector representing the time units where KOMB_{i} has got forced assignment at step s. A necessary condition for i \in W4l for each allocation will be:

\[(10.11)\quad G(\text{LEDIG}_{i,s+1}, \text{KOMB}_{i}) = G(\text{LEDIG}_{i,s}, \text{KOMB}) + |q_{i,s+1} + q_{i,s}| - \delta_{ik} \cdot PL_{k}\]

where \(\delta_{ik} = \begin{cases} 1, & \text{if } i \in \text{KOMB}_{k}^T \\ 0, & \text{otherwise} \end{cases}\)

It can be shown that (10.11) is identical with (7.3). If KOMB_{i} is a TTC, then \(|q_{i,s+1} + q_{i,s}| = 0\) per definition, and if KOMB_{i} is an STC, then (10.11) states that the freedom loss must be 0 for those time units where KOMB_{i} shall satisfy a continuity requirement (i.e., an STC is considered as a TTC for a subset of the available time units). (10.11) is transformed in the same way as (7.3) to a condition analogous to (7.4), which in its turn results in a simplified algorithm analogous to (7.7).

If KOMB_{i} is an STC and \(|q_{i,s+1}| < TT^S (\text{KOMB}_{i}) - \delta_{ik} \cdot PL_{k}\), then one will try to preserve as much freedom as possible for the future allocations of activities in KOMB_{i}, and the STC is still included in the optimizing condition (7.13). An STC on which a continuity requirement is imposed is included in the global conditions (7.3) as well as the local conditions (7.13). This is the most important principle for the consideration of the continuity requirement.

b3. If i \in (W4l \land \text{KOMB}_{k}^T), then allocation of activity k might lead to \(|q_{i,s+1}| > |q_{i,s}|\), and a new necessary condition for each step will be:

\[(10.12)\quad TT(\text{KOMB}_{i}) \geq |\text{LEDIG}_{i,s+1} \land \text{KOMB}_{i}| + PL_{k}\]

for i \in (W4l \land \text{KOMB}_{k}^T)

(10.12) states that the number of unallocated KOMB_{i} time units must be larger than or equal to the number of LEDIG_{i,s+1} time units where an activity is not yet allocated.
c. Modification of the allocation sequence:

For each step matrices DAYF, DAYL, and PKT are modified. This may lead to a number of forced allocations if one of the following conditions is satisfied:

\[(10.13)\quad TD(\text{LEDIG}^{i,d,s+1} \land \text{KOMB}_1) \leq |q^{i,d,s+1} + PKT^{d,s+1}_1|\]

for \(i \in (W32 \land W41)\)

and \(d = 1,2,\ldots,\text{UD}\)

\[(10.14)\quad TT(\text{LEDIG}^{i,s+1} \land \text{KOMB}_1) \leq |q^{i,s+1} + PKT^{s+1}_1|\]

for \(i \in (W32 \land W41)\)

(10.13) states that if the number of hours for the activities which can be allocated to the time units that must be used on day \(d\) is less than or equal to the corresponding number of time units not yet utilized, then the activities in \(\text{KOMB}_1\) get forced assignment to these time units. This is represented by means of \(W11\) and \(W8\) as mentioned in chapter 8, and desired time units on day \(d\) are represented by \(\text{WISH}^1\). (10.14) is an assertion analogous to (10.13) for all time units of the week which must be used by \(\text{KOMB}_1\). In this case the forced allocations cannot be represented by \(W8\) and \(W11\); however, the most desirable time units for \(\text{KOMB}_1\) may be represented by \(\text{WISH}^1\), and the degree of freedom for the activities is determined by the freedom picture given by (9.5).

If (10.13) or (10.14) are satisfied, then the degrees of freedom for the corresponding activities will be reduced, which automatically influences the allocation sequence. (10.11) is a reformulation of (5.20) to include consideration of the continuity requirement. Obviously interaction between this requirement and other conditions easily leads to the limitations due to (5.20) becoming insufficient from a practical point of view. This is compensated for because each time (10.13) or (10.14) is satisfied the corresponding activities will be represented in \(W2\) in order to give them priority for future allocations. (Although an STC with a continuity requirement is a less strict condition than a TTC, it will be more complicated to handle, and one will attempt to avoid it by means of the rules for the allocation sequence.)
New blockings may lead to fulfilment of (10.9) for time units belonging to days other than that of the actual allocation, and the freedom picture is modified by means of (10.10). (Alternatively the time units not desired as a result of (10.9) may be represented by means of $WISH^3$.)

10.3 The split requirement

This requirement is a consequence of the continuity requirement, and its practical motivation will be mentioned: The students of one class generally partake in the same lessons, but for a number of subjects they are regrouped. These new groups must be such that all student TCs can satisfy the continuity requirement. If all student TCs are tight, particular conditions for this split requirement are superfluous, since it is avoided by means of suitable activity definitions when simultaneous education can be unambiguously prescribed, or when specifying the problem, a class can be split into a number of independent student groups and the conditions of chapter 5 ensure satisfaction of the split requirement. A modification of the conflict vector particularly important for the split requirement is (5.55).

If the student schedules are not tight, regroupings of the students in a class will often be done such that for certain activities a number of students in a class have to be exempt, as there are no suitable subjects to offer simultaneously for these students. Such activities are denoted S-activities and are represented by $W40$:

$$W40_i := \begin{cases} 1, & \text{if the resources included in activity } i \text{ are defined in such a way that a number of students from the actual classes must be exempt for activity } i. \\ 0, & \text{otherwise} \end{cases}$$

If the student-TCs are approximately tight, the treatment of S-activities is simple, since it is initially required that they must be allocated to the first or last hour of the day. For some structures a number of student-TCs are far from tight, and S-activities may then occur anywhere. Modification of the conflict picture as a result of S-activities will be shown.

For $k \in (W40 \land KOMB_1)$ is required:
1. None of the possible periods for activity $k$ must be represented by the physical time units $q_{i,d,s+1}^{i,d,s+1}$ for $d=1,2,...,D$, unless the first or last time unit in $q_{i,d,s+1}^{i,d,s+1}$ is simultaneously included in the period. The conflict picture is modified for each step due to this condition.

2. The above mentioned condition ensures that possible periods for $S$-activities at any time correspond to the continuity requirement. If the activity to be allocated is an $S$-activity, then the following must be particularly required: The activities of the classes included in the $S$-activity are blocked such that the $S$-activity becomes the first or last activity of the day. (DAYF and DAYL determine which blockings are necessary in each case.)

The above rules are nothing but administrative routines for ensuring that the continuity requirement for all students is satisfied in the final schedule. It is interesting to compare this modification of the conflict picture and the form of (7.13), since that condition attempts to assign $S$-activities to boundary hours under otherwise equal conditions. There is, however, the danger of exaggerated "boundary-assignment" of $S$-activities, which may render it difficult to fill the central education time or result in students getting an uneven work load on the various days. This is taken into consideration through (7.13) being "counteracted" by (10.12), and a highly desirable requirement of an even work load. Note that an $S$-activity can never get forced assignment due to a continuity requirement. A more correct form of (10.12) is:

$$\text{TT}(\text{KOMB}_{i,w40}) \geq |\text{LEDIG}_{i,s+1}^{i,s+1} \text{ KOMB}_{i} \rangle + \text{PL}_k \cdot \delta_k$$

where $\delta_k = \begin{cases} 0, & \text{if } k \in W40 \\ 1, & \text{otherwise} \end{cases}$

10.4 Other particular conflict types

The previous sections of this chapter describe how new conflict types are fitted to the general strategy, and the examples indicate the flexibility of the strategy. It would lead too far to present conditions for all existing conflict types, but a brief description of some important requirements will be given:
1. Initial conditions

These are very important when specifying the problem; they may, however, easily be adjusted to the strategy:

a. Preassignment. This is treated like an ordinary allocation with the exception that no choice between various possibilities will be made. The conflict picture will be generated for one possibility only, and this is accepted even if it does not correspond with the global conditions.

b. Blockings and forced allocation of resources. These requirements are represented by blocking certain time units in LEDIG, or if room types are blocked, they are transferred to TA. Following registration of all such requirements the rest of the data structure is modified so that it corresponds with LEDIG and TA.

When all initial conditions are represented, one finds whether they are mutually consistent, and whether the initial freedom picture is consistent with the global conditions. This is often not the case, and one is notified about the self-contradictions. On this basis is determined whether the problem must be redefined or whether the program will be allowed to attempt scheduling after all. Of course there is no guarantee that a solution exists if the initial situation is acceptable, but in a large number of cases one may stop a scheduling attempt which is bound to be a failure. This is important when regarding the total run time for production of many schedules.

2. Sequence requirements

The most common sequence requirement is linked activities where two or several activities are to be allocated sequentially on the same day. This requirement may have rather a large influence on the previously discussed conditions, but fortunately most structures consist of a small number of linked activities, and in practice a simplified treatment of these has been justified. All linked activities are allocated as early as possible. This may be done quite simply by demanding preassignment of all linked activities. A certain limitation of the solution space will certainly occur, but provided that there is large freedom in the system due to other initial conditions and that common sense is
used when preassigning, the result is entirely satisfactory. Many systems have quite a large number of initial conditions as well as linked activities. In that case preassignment of the linked activities may become laborious, while at the same time the risk of a wrong allocation is large. Preassignment of linked activities is therefore left to the program; i.e., the activities included in linked activities are allocated first, but instead of generating the conflict pictures for each possible assignment of the individual activity, a common generation is done for the set making up a linked activity. Naturally, only possibilities consistent with a sequence requirement are examined. This is a certain improvement compared with entirely manual education, since global as well as local conditions attempt to secure a more sensible preassignment and simultaneously to avoid obvious self-contradictions.

Another kind of sequence requirement is the case where two activities are to be allocated to neighbouring days. A global condition may be defined which ensures the satisfaction of this requirement for the activities in question, and when one of these activities is allocated, the other gets forced allocation to one of the neighbouring days.

The above is a rather superficial way of considering the sequence requirement, and it is obviously possible to find more extensive conditions. This has, however, not been necessary hitherto.
11. ADDITIONAL PROCEDURES TO THE TIME ALLOCATION

Summary

The previous chapters have discussed the principles of the time allocation of the activities. This chapter will add a few important factors to the strategy:

1. Treating error situations and manual adjustment of schedules.
2. The principles for room allocation.

Although these factors represent very different aspects of the total strategy, they have in common that they are difficult to formalize. The reason is that these factors are closely connected with the school structure in question. For example, manual adjustment is closely connected with the requirements for an acceptable schedule for a school structure. This requires an extensive practical knowledge of the most frequently occurring problems, and which compromises to use to get out of hopeless situations. This experience is most easily acquired by working with practical problems. Eventually one learns numerous "tricks"; which makes it evident that personnel who are responsible for the run of the program must be very familiar with scheduling.

The room allocation can be partly formalized but the principles may vary, among other things, depending on the geographical shape of the school.

The following does not offer any detailed discussion of the actual program, but rather a more verbal description.

11.1 Error situations and manual adjustment

When a strategy is based only on necessary conditions, error situations are apt to occur; i.e., it may be impossible to allocate a period of an activity. This is due to two factors:

1. The given problem may be self-contradictory, and one cannot prove that a solution does not exist.
2. The strategy may be incomplete, and wrong allocations are made so that a solution no longer exists.
A common way of compensating an incomplete strategy is by means of "back-tracking"-mechanisms; i.e., at the instant it is definitely known that no solution exists, it is assumed that one of the earlier assignments is wrong, and one returns to a previous situation for another attempt at finding a solution. Related to a scheduling program such a technique has its severe limitations:

1. The majority of problems are the result of a specification which is self-contradictory, and regardless of which "way" is chosen no solution exists.
2. Back-tracking is administratively complicated, it requires much storage space and is very time-consuming.

Back-tracking is a method for how to generate new possibilities; however, a far more interesting objective is to find out why a strategy has lead to a wrong allocation. Finding criteria for which possibilities may lead to solution is far more important than a purely mechanical examination of all possibilities.

During the development of the strategy in question errors have naturally been discovered. The error situations which have occurred, have been the primary aid in improving the strategy. An analysis of these has lead to major generalizations of the necessary conditions for finding a solution. Another important result of such an analysis are practical rules which should be followed when specifying a problem. (In this paper is given a generalization of the necessary conditions. By applying them to a particular school structure one may anticipate the consequences of various specifications, which are a valuable aid to the individual school principal.) Hitherto it has not proved useful to let the program alone make compromises when no solution can be found, a manual technique is used in combination with the program. How this is done will be summarily discussed.

An evident assumption for using a program is a formally (syntactically) correct specification of the problem. This is done on standard forms (see [1]), and when filling in the forms certain rules must be followed to secure an unambiguous specification of the problem; some extra information is also required for control of data. The first phase of the program tests whether the problem is formally correct. Correcting data errors is easy, but frequently time-consuming, routine work.
When all formal errors have been corrected it is investigated whether the initial situation is in accordance with the necessary conditions. Numerous errors are found in that way; typical examples are:

1. TCs of higher order are formed so that the allowed time frame is exceeded.

2. Streamed structures often imply that certain SCs (or TCs) cannot be assigned within the allowed time frame as a number of time units will be completely blocked for these activities.

3. Some resources (rooms) are overloaded. Schools assume rather too easily that the room-load is justifiable if a room is in use for fewer hours than the number of hours of a week. Streamed structures often necessitate that some overloaded rooms cannot be used at all for a number of hours.

4. The prescribed period division for certain sets of activities imply that they cannot be allocated within a given time frame.

5. Pre-blocking of resources may easily lead to DTCs being blocked for too many days.

   etc.

It is relatively simple to find errors of this kind and possibly consult the school to ensure that the data become correct.

Making a schedule from a given specification is much simpler than proving that a solution exists. During the allocation, situations may arise where a period of an activity cannot be allocated unless one of the absolute requirements is broken. This period is eliminated from the system by a set of cleaning up processes (analogous to section 8.1).

From a practical viewpoint it must be pointed out that one of the most intricate technical problems has been to design the program such that error situations are satisfactorily tackled so that they have no consequences for further allocation. Assume for instance that a number of forced assignments will be done, and that allocation of one of these is impossible. Forced assignments have at an earlier stage limited the solution space, and now one of the activities causing this limitation is eliminated from the system. Accordingly,
one will return the actual freedom present if one activity can be disregarded. This may be an extensive administrative problem, and when designing the program much time was spent on handling the peculiarities which might arise due to an error situation.

It is assumed that the eliminated period can be manually adjusted afterwards. The factors to be taken into consideration are:

1. How simple is it to assign an activity manually?

2. Which requirements do the school in question make to an acceptable schedule?

The difficulties of assigning an activity manually are a function of the resources included in it. It is simple to make a manual adjustment if the number of resources is small (e.g. one class, one teacher, and one room), whereas a large number of resources renders it almost impossible. The periods eliminated from the system are classified in types according to the number of resources included. (This is done by means of W17 and W18.)

If the number of unallocated periods for one of these types exceeds a corresponding limit, the allocation is terminated. The result of a run could thus be:

1. The allocation is terminated after a number of steps because the error situations which have arisen, are too complicated for achieving a feasible solution by manual adjustment.

2. The allocation is finished with possibly a small number of remaining unallocated activities.

In both cases the partial schedule is evaluated from the requirements to an acceptable schedule; e.g., the possibilities for transforming or simplifying the original problem. In other words, as a compromise "absolute" requirements are redefined. The most common way of doing it is:

a. Switch teachers in some subjects.

b. Break the parallel bound between certain subjects.

c. Accept poor room alternatives for certain activities.

d. Change the period division for some activities and possibly remove certain day blockings.

e. Attempt to remove pre-blockings of certain activities.
What should be done in the individual case depends of course on which compromises are most readily accepted by the school and which errors have occurred.

The rules for the allocation sequence provide for the most critical activities being allocated first, which among other things implies that complicated interactions between the conflict types are dealt with early, and that possible error situations arise quickly. The major errors are usually discovered before 10% of the activities are allocated. Should it be necessary to terminate an allocation, then two kinds of information are available:

1. A number of printouts stating which global conditions have been broken hitherto in the allocation.

2. A limited partial schedule containing an unacceptable number of error situations.

On that basis it is possible to prove the reason why certain activities have not been allocated. The current program system is so complete that possible error situations are nearly always due to the problem specification. (Exceptionally local and qualitative criteria may lead to an error allocation. This is due to rather peculiar interactions between the conflict types, and regard for computing time forbids evaluation of these. The problem is avoided by preassigning.)

When self-contradictions in the problem have been definitely proved, a number of alternatives may be found for avoiding them and a compromise is done. (Possibly after consulting with the school.) An example: Assume that the error situations are due to two streams A and B. It may be apparent that a number of theoretical activities for stream A must concur with optional activities for stream B. If this is not possible, certain TCs will exceed the time frame, or room utilization becomes so uneven that no solution exists. (Theoretical activities require many classrooms while optional activities usually use special rooms.) An analysis of a partial schedule shows:

1. The period division and the day conflicts for the two sets of activities are defined in such a way that it is completely impossible to assign the necessary number of periods to the same time units. (Theoretical activities should be distributed
evenly throughout the week while optional activities should be
taught sequentially.}

2. Independent of the period division it becomes apparent that the
two sets of activities have so many resources (teachers) in
common that a sufficient number of periods cannot be assigned to
the same time units.

Assume further that the actual error situation is typical of the
school structure in question, and when terminating a schedule one
gets a printout stating which resources in pairs are common for the
activities of the two sets. An attempt is made at changing the
teachers for the different activities such that activities with a
small number of common resources become conflict-free. An attempt is
made to utilize the resulting new freedom within the given period-di-
vision (which possibly has to be changed). Obviously the mentioned mo-
difications may have considerable consequences for the schedule, and
the school itself must determine the necessary modifications. The ad-
vantage of the program in this case is that self-contradictions
found manually when the schedule is "nearly" completed, are quickly
pointed out. Most error situations are usually due to a small num-
ber of conditions; furthermore, the same errors seem to occur repe-
atedly.

Modifications of the problem result in additional runs. It is
hardly ever necessary to make more than 2 runs; i.e., the first run
results in a partial schedule, while the second time, an approximately
complete solution is found. The remaining unallocated activities
are assigned manually by modifying the absolute requirements (e.g.
switch teachers, use poor room alternatives, change the period-
division, etc.). Simultaneously it is attempted to adjust the
schedule manually to make it better, in accordance with desirable
requirements.

These finishing touches on the schedule are a question of experience.
Preferably the program should do the mentioned adjustment, but the
advantage of manual evaluation is that one has complete control over
the compromises being done. They are often done from rather special
circumstances at the individual school. One may readily imagine a
gradual development where a program takes over the final adjustment
or possibly modifies the problem; this must, however, come as a result of more experience.

The objections to the manual adjustment are that a partial schedule may be of no value if the adjustment becomes too extensive. Hitherto manual work to achieve a complete schedule has very seldom exceeded 8 hours (including time for analysis of self-contradictory data and final adjustment). For the simplest structures a complete schedule is usually found at the first try, whereas for complicated structures 1-3% of the periods may not be allocated. (These figures must be seen in the light of the examined structures.) Experience from manual adjustment may be summed up:

In order to assign manually the unallocated activities, an absolute requirement must (nearly) always be modified. This strongly verifies that error situations which have arisen are mainly due to the problem definition and not to the strategy. However, it has often proved possible to consider desirable requirements in a better way without breaking absolute requirements, and the strategy may be improved in this field. On the other hand, experience has shown that if desirable requirements are given too much priority, absolute requirements may be broken. It is difficult to determine beforehand which priority should be given to desirable requirements. The finishing touch of a schedule is a question of conscience, and the schools themselves rather than the program should decide what must be corrected.

11.2 Room allocation

Room allocation may be divided into three phases:

1. Room utilization: The various room resources are attempted to be utilized such that all subjects are being taught in the most suitable rooms, and the schedule must be made such that for any time unit there are sufficient rooms of any room type. These factors are taken into consideration by means of the principles discussed in section 5.2. In chapter 9 it was pointed out that the requirement for suitable rooms has consequences for the allocation sequence. The room utilization is treated in parallel with the time allocation, and other phases of the room allocation may be evaluated globally.
An important measure of the quality of a schedule is the room utilization; a parameter which is easily quantified. Experience has shown that this is one of the factors treated far better by the program than by manual technique. (This does not prove that the chosen method is correct, but rather that by manual scheduling one easily loses track of the possibilities and accordingly neglects the room utilization.) Many schools have difficulties with getting a sufficient number of rooms, and the importance of a maximal room utilization is easily realized.

2. Room distribution: Each activity must get the appropriate (alternatively acceptable) type of room. Contiguous education in one subject must have the same room. Some special rooms will be overloaded, and the use of these should be divided as evenly as possible between the student groups.

3. Room balancing: Within each type of room it is ordinarily desirable for the same class to stay in the same room as much as possible. One wishes to avoid unnecessary moving of classes, and when a class has to change rooms, it is desired that it should use a minimum number of different rooms. The room balancing is here done out of regard for the students. Some schools prefer that the teacher has his particular room, or that a room should be used for education in one subject only (the subject-room principle).)

The room distribution is a set of rules stating how the student groups should be distributed to the room types, whereas the room balancing is a set of rules for distributing rooms within one type of room. The desire for even utilization of the special rooms implies some moving on part of the classes. In the following is assumed that all use of special rooms will be distributed as evenly as possible, while the classes will preferably stay in one classroom for ordinary education.

11.2.1 Principles for room distribution

The types of rooms may be classified in the following groups:

Group 1: Room types without alternatives.

Group 2: Room types with alternatives. None of the alternatives are classrooms or alternative classrooms.

Group 3: Room types with alternatives which can also be used as classrooms.
Group 4: Classrooms.

The room types are treated in that order, and each type is treated individually. To secure contiguous education in the same room, rooms are distributed first for triple hours, secondly for double hours, and finally for single hours.

Allocating rooms for room types of group 1 is trivial, and group 4 will be treated by the room balancing. For room types of groups 2 or 3, rooms are distributed to the activities consecutively and one always chooses the most desirable of the unoccupied rooms. If the room types are overloaded (i.e., one has to use undesired alternatives) the use of the best rooms must be distributed between the classes. Usually a class will have few hours in a particular special room, and simple rules for exchange of rooms can be used. Good room alternatives are denoted primary rooms and undesired room alternatives are denoted secondary rooms. Each activity gets a weight \( w \):

\[
\begin{align*}
  s & = \frac{P}{(S+P)} \\
  w \ &= \frac{s}{(S+P)}
\end{align*}
\]

where \( P \) = the number of time units in which the activity is allocated to primary rooms.

\( S \) = the number of time units in which the activity is allocated to secondary rooms.

One first attempts to exchange rooms for activities with a high weight.

Assume that activities a and b are allocated to the same time unit and that both use the same type of room. The purely mechanical room distribution has assigned activity a to a primary room and b to a secondary room. The conditions for a possible room exchange are:

1. Activity a has more hours in primary than in secondary rooms.
2. Activity b has more hours in secondary than in primary rooms.
3. A possible exchange of rooms must not imply that activity a gets the primary room during the same (or less) number of hours than activity b did before the exchange of rooms (i.e., no exchange is done if activity a consists of only one period).
4. Exchange of rooms must be in accordance with the requirement of the same room for double- and triple hours.
The corresponding weights are modified after each exchange of rooms. (Assume for instance that one room type consists of one primary and one secondary room, and that all activities using the room type consist of two single hours. If a solution is found without manual adjustment, each activity can always get the primary room at least for one hour.)

The rules for exchanges of rooms may be formulated in a more general way, but the above is usually sufficient. In addition to these rules, conditions are defined for giving priority to certain classes (for instance senior classes) when exchanging rooms.

11.2.2 Principles for room balancing

The rooms within a room type will be balanced, and to simplify matters it is assumed that no alternatives exist. The problem is:

Given R rooms and CL classes. CL is larger than R, but there are always enough rooms for any hour. Assign rooms, using a minimal number of room-changes for the individual classes.

A simple room balancing for 5 classes, 3 rooms, and 6 hours is outlined in figure 11.1.

![Figure 11.1](image)

Matrices A and B are defined as:

\[
A_{ij} := \begin{cases} 
1, & \text{if class } i \text{ uses the room type in question for hour } j. \\
0, & \text{otherwise}
\end{cases}
\]

(11.1)

\[
B_{ij} := \text{The class that has been assigned to room } i \text{ for time unit } j.
\]

Given:

(11.2) \( R \geq |A_j| \) for all \( j \)
From matrix B is seen that regardless of how the rooms are distributed to the classes at least one class will have to change rooms. One distinguishes between two kinds of room-changes:

**Unnatural room-change:** A class uses two different rooms of the same room type for contiguous hours.

**Natural room-change:** A class uses two different rooms of the same room type, but not for contiguous hours.

(In figure 11.1 class 3 has a natural room-change, as it has education in other room types during hours 2-5.)

The room balancing should be conducted in such a way that no class gets unnatural room-changes.

This can always be satisfied for a room type without alternatives. (Assume that until time unit j the classes have been assigned rooms without any unnatural room-changes. One may always provide for the classes using the room type for time units j and j+1 to get the same room.) Vector \( a^i \) is defined as:

\[
\begin{align*}
\text{a. } & a_i^i \in A_i \\
\text{b. If } a_j^i & \neq 1, \text{ then } (A_i \ast a^i)_{j+\delta} = 0, \\
\text{where } \delta & = 1
\end{align*}
\]

(11.3)

(a\(^i\) is a selection of \( A_i \) such that class i may have different rooms for time units \( a^i \) and \( (A_i \ast a^i) \) without an unnatural room-change.)

p vectors \( a_1^i, a_2^i, \ldots, a_p^i \) satisfying the following will be found:

\[
\begin{align*}
\text{a. } |a_i^i \wedge a_j^j| & \equiv 0 \\
\text{for } i & = i_1, i_2, \ldots, i_p \\
\text{and } i & \neq j, \\
\text{b. } | \bigvee_{i = i_1}^{i_p} a_i^i | & = \text{MAX} \\
\text{c. As a secondary condition is wanted } p = \text{MIN.}
\end{align*}
\]

(11.4)
The room balancing will be done for DT hours. The maximum vector-sum is DT (minimum p=1, and maximum p=DT). When a set of vectors satisfying (11.4) has been found, the data structure is modified:

\[
\begin{align*}
q &= q + 1 \\
A_i^j &= A_i^j + a^i_j \\
B_{qj} &= i \text{ if } a^i_j \neq 1
\end{align*}
\]

for \(i = i_1, i_2, \ldots, i_p\).

q is a stage variable, initially 0. (11.4) and (11.5) define an iteration which stops when \(|A| \equiv 0\). The final q must be less than R due to (11.2).

The search process of finding a set of vectors satisfying (11.4) may in principle be rather extensive. As a curiosity may be mentioned that this is done by splitting each \(A_i^j\) into the maximum number of subvectors satisfying (11.3). Each subvector is considered as a descriptor (see section 5.1). For the set of descriptors for all \(A_i^j\), a conflict matrix is formed, and TCs are generated as shown in chapter 5. From these TCs it is possible to find the set of vectors satisfying (11.4). In practice much simpler rules are sufficient:

1. \(a^i\) is set equal to \(A_i\) for all \(i\). For \(p = 1, 2, 3, \ldots, DT\) one tries to find a vector-sum equal to DT.

2. The row vectors of \(A\) are combined in pairs in order to find a vector-sum for two subvectors \(a^{i1}\) and \(a^{i2}\) equal to DT. (This gives at least one natural room-change.)

3. \(a^{i1}\) is the subvector with the highest number of contiguous l-components starting in time unit 1. \(a^{i2}\) is the subvector which satisfies (11.3) and (11.4), and which has the highest number of contiguous l-components starting at (possibly after) time unit \(|a^{i1}| + 1\), etc. The subvectors will be formed from a minimum number of row vectors of \(A\). This will be shown by an example:

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 \\
3 & 1 & 1 & 1 & 1 & 1 \\
4 & 1 & 1 & 1 & 1 & 1 \\
5 & 1 & 1 & 1 & 1 & 1 \\
6 & 1 & 1 & 1 & 1 & 1 \\
7 & 1 & 1 & 1 & 1 & 1 \\
8 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Figure 11.2

CL=9 and R=6

A
By applying the mentioned rules is found:

Fig. 11.3

B is a room balancing of A consisting entirely of natural room exchanges. Room assignments are as packed as possible. If the room balancing aims at minimizing usage of certain rooms, then B as given in figure 11.3 is an alternative. Note that room 6 has a small utilization. This availability will often be used to reduce natural room-changes, which is done by a simplified principle:

By regarding different combinations of two or three rooms in B, trials are made to swap classes for some time units so that a natural room-change is removed without another one arising.

In figure 11.3 classes 1, 2, 3, and 5 have natural room-changes. Figure 11.4 shows how these room-changes are gradually eliminated:

Figure 11.4

(One succeeded in removing all room-changes in figure 11.3. This is, however, not always possible.)
The mentioned method guarantees that no class has unnatural room-changes, and that simplified rules reduce natural room-changes. An additional condition is that each class during the week should use as few different rooms as possible.

Each class has a "natural room-change" from one day to another, and room balancing is done day by day. (It is time-consuming and unnecessarily complicated to do the balancing for a week as a whole.) Following room balancing for all days, it must be determined which physical room a row vector of the various B-matrices shall represent. This is done by gradually constructing the week-schedule such that the classes are assigned to the same room as much as possible, and the maximum number of different rooms for a single class will be minimized. Figure 11.5 indicates how the room balancing for Wednesday is distributed to the physical rooms.

<table>
<thead>
<tr>
<th>MONDAY</th>
<th>TUESDAY</th>
<th>WEDNESDAY</th>
<th>B</th>
</tr>
</thead>
</table>
| 1 2 3 4 5 6 | 1 2 3 4 5 6 | 1 2 3 4 5 6 | \[
| 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 8 | 8 |
| 2 | 2 | 2 | 4 | 4 | 4 | 4 | 4 | 2 | 2 |
| 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 2 | 2 |
| 4 | 3 | 3 | 7 | 7 | 7 | 7 | 7 | 2 | 2 |
| 5 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 2 | 2 |

Figure 11.5

Note that room 1 is preferred for both classes 1 and 5. Class 1 gets the room, since class 8, which is also using it, has already got two different rooms.

Theoretically, the room balancing might be improved, but the method is satisfactory and considerably better than rather arbitrary manual methods for room balancing. Characteristic properties of the method are:

1. Unnatural room-changes are avoided. (The requirement of double- and triple periods in the same room is thus automatically satisfied.)

2. All classes have some room-change, but totally the number of room-changes is "minimized". (Typical "ambulating" classes are avoided; i.e., one does not use the principle of a number of classes getting a room of their own, while the rest of the classes more or less arbitrarily get an available room.)
3. The classes which change rooms use a small number of rooms during the week.

The described model for room balancing is simplified, and in addition the program must consider the following factors:

1. A room type may have alternatives. This is equivalent to some of the activities getting prescribed physical rooms. One cannot without restrictions do the mentioned room-changes or assign the classes to rooms only considering that unnatural room-changes should be avoided. Room type alternatives may lead to unnatural room-changes. This can be taken into account in two ways:
   a. Those activities which are going to have prescribed rooms are assigned initially. This partly determines the B-matrix, and room-changes for these activities must not occur. The remaining activities are balanced from the beforementioned principles, and one should see to that double- and triple periods get the same room.
   b. The room balancing is done only for the rooms primarily belonging to the room type, and the alternatives are used as buffers for hours where there are not enough rooms. This is the simplest way, and it is usually entirely satisfactory.

2. Some classes have preference rooms; i.e., it is undesirable for these classes to change rooms. A row vector in the B-matrix ought not to contain classes with different preference rooms. This implies that the number of room-changes increases for other classes.

3. One may prescribe which classes should use a certain group of rooms; i.e., one prescribes which classes should be included in the same row vectors in the B-matrix.

4. Certain teachers or student groups should have the same room for all periods of certain activities.

5. Room-changes must be done by taking into account the geographical distance between the rooms. (This factor should possibly be evaluated during the time allocation.)
12. SUMMARY AND CONCLUSION

From the achieved results it should be evident that using a program for scheduling is both possible and useful, and that a general and simultaneously realistic scheduling strategy has been developed.

The central concept for formalizing the problem is an activity. The activity concept is defined from simultaneous requirements for the resources of the system, which makes possible an abstraction of the various practical motivations for the requirements of a schedule. Conditions other than pure resource requirements are formulated as relations to the relative time sequence between the activities. (The formalization of the problem also shows the limitation of the strategy. For the problem in question the most important conditions are found from the conflict matrix, which is not necessarily the case for analogous problems.)

One of the most important results is that the specification possibilities allow a precise and covering description of the problems existing in practice. It has been very encouraging to know that the schools greatly appreciate being able to properly formulate a problem hitherto regarded as rather vague. The rules for data specification are in many ways just as valuable for manual scheduling, but they are especially important for computer scheduling due to the vital communication problems. The data specification implies a definite distinction between the central decisions to achieve an acceptable schedule, and the more mechanical and time consuming allocation process. This necessitates more thorough advance work than usual for manual methods. The advantage is that the problem is in reality solved before the actual scheduling takes place. During manual scheduling one must often make awkward compromises, whereas a program offers the possibility of choosing more acceptable compromises when defining the problem.

An assumption for adopting a program is its economic competency. In this connection it is vital that the data structure and the corresponding operators (procedure sets) for logical variables are efficient in space- and time usage. A suitable set of operators is also very important for the program development itself. Several relations are complicated, and are defined by a large number of
ABBREVIATIONS FOR VARIABLES, OPERATORS AND CONCEPTS.

A lot of symbols have been necessary to define the various relations. The following survey contains the most frequently used symbols, but temporary and auxiliary symbols which have not the same meaning in all contexts, have been omitted.

The symbols have been given a classification and the following abbreviations are used:

a. **Variables:**
   - am: arithmetic matrix
   - as: arithmetic scalar
   - av: arithmetic vector
   - lm: logical matrix
   - ls: logical scalar
   - lv: logical vector

   The dimension of vectors and matrices is listed. (The vertical dimension is listed first for matrices).

b. **Operators** are denoted 0, and the logical operators which got a particular mention in chapter 4, are not listed.

c. **Concepts** are denoted C, and the meaning of a concept in this survey is usually a set of activities, either separately or as a unit, having a particular property. A concept might also be a characteristic parameter for the individual activity, thus the difference between a variable and a concept may become vague; e.g. the symbol SP denotes a vector defining the number of unallocated periods for the activities and this is physically represented in the first column of data area SP1.

The following convention is used: Any symbol representing a data area in the current program is classified as a variable and any symbol only used for the relations in this book is classified as a concept.

There is given a page reference showing where the symbols are defined or mentioned.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Classification</th>
<th>Dimension</th>
<th>Reference</th>
<th>Short explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>as</td>
<td></td>
<td>175</td>
<td>The number of teachers.</td>
</tr>
<tr>
<td>ALM</td>
<td>as</td>
<td></td>
<td>176</td>
<td>The vertical dimension of data area LBF.</td>
</tr>
<tr>
<td>AMAX</td>
<td>as</td>
<td></td>
<td>115</td>
<td>The number of fictitious rooms.</td>
</tr>
<tr>
<td>AR</td>
<td>lm</td>
<td>ATY, RTMAX</td>
<td>112</td>
<td>The room type hierarchy.</td>
</tr>
<tr>
<td>ATY</td>
<td>as</td>
<td></td>
<td>108</td>
<td>The number of externally defined room types.</td>
</tr>
<tr>
<td>BPTC</td>
<td>C</td>
<td></td>
<td>190</td>
<td>Broken day-terminal combination.</td>
</tr>
<tr>
<td>BSC</td>
<td>C</td>
<td></td>
<td>73</td>
<td>Broken sub-combination.</td>
</tr>
<tr>
<td>BTC</td>
<td>C</td>
<td></td>
<td>73</td>
<td>Broken terminal combination.</td>
</tr>
<tr>
<td>CL (cl)</td>
<td>as</td>
<td></td>
<td>176</td>
<td>The number of classes.</td>
</tr>
<tr>
<td>CLB</td>
<td>am</td>
<td>bt, UD</td>
<td>202</td>
<td>bt is the number of hours pr. day where a lunch-break may occur, and CLB keeps track of how many classes have been assigned lunch-break on the various time-units.</td>
</tr>
<tr>
<td>CLM</td>
<td>as</td>
<td></td>
<td>176</td>
<td>The vertical dimension of data area KLF.</td>
</tr>
<tr>
<td>CRIT</td>
<td>av</td>
<td>TMAX</td>
<td>167</td>
<td>Temporary storage of the freedom loss from (7.13) for the various possibilities.</td>
</tr>
<tr>
<td>DABL</td>
<td>lm</td>
<td>DMA, N</td>
<td>140</td>
<td>Defines the day terminal combinations.</td>
</tr>
<tr>
<td>DAYF</td>
<td>am</td>
<td>KM, UD</td>
<td>206</td>
<td>Defines the first time unit on the various days which the TCs must use due to the continuity requirement.</td>
</tr>
<tr>
<td>DAYL</td>
<td>am</td>
<td>KM, UD</td>
<td>206</td>
<td>Defines the last time unit on the various days which the TCs must use due to the continuity requirement.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Classification</td>
<td>Dimension</td>
<td>Reference</td>
<td>Short explanation</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>KOLMA</td>
<td>1m</td>
<td>N, N</td>
<td>70</td>
<td>The conflict matrix.</td>
</tr>
<tr>
<td>KOMB</td>
<td>1m</td>
<td>KM, N</td>
<td>69</td>
<td>Defines the terminal combinations. KOMB is denoted resource_matrix_or_PC_matrix.</td>
</tr>
<tr>
<td>KONV</td>
<td>0</td>
<td></td>
<td>128</td>
<td>Performs a coordinate transformation for periods of a given length to the &quot;corresponding&quot; set of periods with another length.</td>
</tr>
<tr>
<td>KV</td>
<td>C</td>
<td></td>
<td>103</td>
<td>The set of activities getting new conflicts due to direct resource conflicts, which is not found from the conflict matrix alone, but as a result of the most obvious consequences of relation (5.20).</td>
</tr>
<tr>
<td>LEDIG</td>
<td>1m</td>
<td>TMAX, N</td>
<td>32</td>
<td>The freedom picture defining available time units for the activities. There is made several selections from this data area e.g. LEDIG^d defining available time_units_on_day_d, etc.</td>
</tr>
<tr>
<td>LEDPER</td>
<td>1m</td>
<td>TMAX, N</td>
<td>129</td>
<td>Defines the available periods for the activities.</td>
</tr>
<tr>
<td>LJ</td>
<td>0</td>
<td></td>
<td>117</td>
<td>The function of this operation is to represent in TA new room blockings due to the current allocation. This is done by left-justifying the l-components of the room requirement (defined by FA) according to certain rules.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Classification</td>
<td>Dimension</td>
<td>Reference</td>
<td>Short explanation</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>LLF</td>
<td>lm</td>
<td>ALM, N</td>
<td>176</td>
<td>A selection of the TC-matrix KOMB consisting among other things of the activity sets for teachers and direct resource conflicts due to rooms.</td>
</tr>
<tr>
<td>N(p1)</td>
<td>as</td>
<td></td>
<td>32</td>
<td>The number of activities.</td>
</tr>
<tr>
<td>PKT</td>
<td>lm</td>
<td>KM, TMAX</td>
<td>206</td>
<td>Due to the prehistory some time units cannot be used by any activity of a TC. PKT keeps track of this.</td>
</tr>
<tr>
<td>PL(p1)</td>
<td>C</td>
<td></td>
<td>125</td>
<td>The period length of the activities.</td>
</tr>
<tr>
<td>q1,q2</td>
<td>lv</td>
<td>(N)</td>
<td></td>
<td>The letter q followed by an integer denotes always an auxiliary logical vector which may be used in a lot of different contexts.</td>
</tr>
<tr>
<td>R</td>
<td>lm</td>
<td>RTMAX, RMAX</td>
<td>108</td>
<td>The room type matrix defining which physical rooms belong to the various room types.</td>
</tr>
<tr>
<td>r</td>
<td>C</td>
<td></td>
<td>74</td>
<td>Denotes the freedom loss for a TC with respect to a time interval.</td>
</tr>
<tr>
<td>RANK</td>
<td>av</td>
<td>N</td>
<td>156</td>
<td>Defines the allocation-priorities of the activities when only considering potential conflicts.</td>
</tr>
<tr>
<td>RJ</td>
<td>O</td>
<td></td>
<td>116</td>
<td>This operator is used to represent in FA the room requirements of the activities. This is done by means of right-justifying room requirements within subvectors of FAi.</td>
</tr>
<tr>
<td>RMAX</td>
<td>as</td>
<td></td>
<td>108</td>
<td>The number of physical rooms.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Classification</td>
<td>Dimension</td>
<td>Reference</td>
<td>Short explanation</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ROM</td>
<td>av</td>
<td>ATY</td>
<td>115</td>
<td>Defines the number of primary rooms for the room types.</td>
</tr>
<tr>
<td>ROTY</td>
<td>av</td>
<td>RTMAX</td>
<td>116</td>
<td>A row vector of FA or TA consists of several subvectors. ROTY defines the first and last component of the subvectors.</td>
</tr>
<tr>
<td>RR</td>
<td>C</td>
<td></td>
<td>121</td>
<td>The set of activities getting new blockings due to room conflicts only.</td>
</tr>
<tr>
<td>RT</td>
<td>O</td>
<td></td>
<td>165</td>
<td>A weight describing the &quot;tightness&quot; of a TC.</td>
</tr>
<tr>
<td>RTMAX</td>
<td>as</td>
<td></td>
<td>109</td>
<td>The total number of room types.</td>
</tr>
<tr>
<td>SC</td>
<td>C</td>
<td></td>
<td>72</td>
<td>Sub-combination.</td>
</tr>
<tr>
<td>SDTC</td>
<td>C</td>
<td></td>
<td>140</td>
<td>Slack day terminal combination.</td>
</tr>
<tr>
<td>SF</td>
<td>1m</td>
<td>X, N</td>
<td>186</td>
<td>X is the number of levels of desirable requirements, and SF defines which activities are included in desirable requirements on the various levels.</td>
</tr>
<tr>
<td>SP</td>
<td>C</td>
<td></td>
<td>73</td>
<td>The number of unallocated periods for the activities.</td>
</tr>
<tr>
<td>SP1</td>
<td>am</td>
<td>N, 10</td>
<td>155</td>
<td>SP1 is denoted the description of the activities and defines important parameters for an activity: e.g. number of unallocated periods, period length, degree of freedom, available days etc.</td>
</tr>
<tr>
<td>SP2</td>
<td>am</td>
<td>N, UD</td>
<td>141</td>
<td>SP2 defines the number of available (and desirable) periods for the activities on the various days.</td>
</tr>
<tr>
<td>SP3</td>
<td>am</td>
<td>KM, UD</td>
<td>136</td>
<td>SP3 defines the utility of the various days for the TCs.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Classification</td>
<td>Dimension</td>
<td>Reference</td>
<td>Short explanation</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>SP4</td>
<td>av</td>
<td>N</td>
<td>119</td>
<td>SP4 defines which room requirement in FA is valid for the individual activity.</td>
</tr>
<tr>
<td>SP5</td>
<td>am</td>
<td>N, DK</td>
<td>140</td>
<td>SP5 defines in which day-conflicts (in DABL) the individual activity is included.</td>
</tr>
<tr>
<td>SP10</td>
<td>am</td>
<td>KM, UD</td>
<td>136</td>
<td>SP10 defines the number of available time units on the various days for the TCs.</td>
</tr>
<tr>
<td>SP12</td>
<td>am</td>
<td>KM, UD</td>
<td>141</td>
<td>SP12 defines the number of time units for the TCs which are forced to the various days, and which are not yet allocated.</td>
</tr>
<tr>
<td>SSC</td>
<td>C</td>
<td></td>
<td>73</td>
<td>Slack sub-combination.</td>
</tr>
<tr>
<td>STC</td>
<td>C</td>
<td></td>
<td>73</td>
<td>Slack terminal combination.</td>
</tr>
<tr>
<td>TA</td>
<td>lm</td>
<td>TMAX, AMAX</td>
<td>120</td>
<td>TA defines the room blockings for the various time units.</td>
</tr>
<tr>
<td>TAPT</td>
<td>lm</td>
<td>TMAX, N</td>
<td>33</td>
<td>TAPT is the conflict picture, which is a superposition of all new blockings arising from the possibility which is being examined (usually this investigation may be restricted to the day in question, and the dimension of TAPT will then be [DT,N]).</td>
</tr>
<tr>
<td>TC</td>
<td>C</td>
<td></td>
<td>72</td>
<td>Terminal combination.</td>
</tr>
<tr>
<td>TD</td>
<td>0</td>
<td></td>
<td>133</td>
<td>TD defines the maximum number of time units of an activity set which may be allocated to a day interval when considering day conflicts.</td>
</tr>
<tr>
<td>TDTC</td>
<td>C</td>
<td></td>
<td>140</td>
<td>Tight day terminal combination.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Classification</td>
<td>Dimension</td>
<td>Reference</td>
<td>Short explanation</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>W31</td>
<td>lv</td>
<td>N</td>
<td>167</td>
<td>Defines the activities getting forced assignment to certain time units of the day where the current allocation is done.</td>
</tr>
<tr>
<td>W32</td>
<td>lv</td>
<td>XM</td>
<td>168</td>
<td>Defines which row vector of KOMB where at least one activity is setting new blockings.</td>
</tr>
<tr>
<td>W33</td>
<td>lv</td>
<td>XM</td>
<td>177</td>
<td>Defines the TTCs included in W32.</td>
</tr>
<tr>
<td>W34</td>
<td>lv</td>
<td>N</td>
<td>177</td>
<td>Defines the STCs included in W32.</td>
</tr>
<tr>
<td>W40</td>
<td>lv</td>
<td>N</td>
<td>210</td>
<td>Defines the activities which must be allocated to a &quot;boundary&quot; period due to a continuity requirement.</td>
</tr>
<tr>
<td>W41</td>
<td>lv</td>
<td>XM</td>
<td>206</td>
<td>Defines the TCs which shall fulfill a continuity requirement.</td>
</tr>
<tr>
<td>XA</td>
<td>lv</td>
<td>N</td>
<td>204</td>
<td>Defines the activities not being taught every week.</td>
</tr>
</tbody>
</table>
APPENDIX 2.

LITERATURE REFERENCES.

   Tapir forlag, NTH (second version, 69).

The above book forms a unit together with the current one since it defines the "external" formalization of the master schedule. For the moment there exists no complete English translation, but there is made a preliminary abstract with the name: "The Nor-Data scheduling system".

The following is far from being a complete bibliography, but extensive references are made to show the different approaches.

   The Computer Journal, Jan. 66.


   The Computer Journal, Jan. 61.

   Revue roumaine de mathematiques pures et appliquees. No 5, 65.


   Ny Teknik, 69.
8| Berghuis, J., Heiden, A.J., Bakker, R. : The preparation of school time tables by electronic computer. BIT, No 2, 64.

9| Blakesley, J.F. : A data processing system for scheduling university students to classes. Thesis, Purdue University, 60.


| 18 | Foxley, E.,  
    Lockyer, K. | : The construction of examination timetables.  
                   The Computer Journal, 68. |
                   IFIP Congress, 62. |
| 20 | Gotlieb, C.C.,  
    Chima J. | : Tests on a computer method for constructing school timetables.  
                   Communications of the ACM, Mar. 64. |
                   University of Toronto, 64. |
                   Elektronische Datenverarbeitung, 9/68. |
| 23 | Greko, B. | : School scheduling through capacitated network flow analysis.  
                   Statskontoret, 65. |
| 24 | Grimes, J.E. | : Scheduling to reduce conflict in meetings.  
                   Communications of the ACM, Jun. 70. |
| 25 | Grung, B. | : Et programsystem for generering av timeplaner.  
                   University of Bergen, 66. |
| 26 | Gunzenhäuser, R.,  
                   Matematik-Technik-Wirtsch., 64. |
| 27 | - " - | : Ein Program zur elektronischen Berechnung von Stundenplänen.  
                   Elektron. Datenverarbeitung, 64. |
| 28 | Hall, A.H.,  
    Acton, F.S. | : Scheduling university course examinations by computer.  
                   Communications of the ACM, Apr. 67. |
| 30 | Holzman, A.G. Turkes, W.R. Et.al. | Optimal scheduling in educational Institutions. University of Pittsburgh, 64. (This book consists of several different approaches to the problem). |
| 31 | Johnston, H.L. Woltenden, K. | Computer aided construction of school timetables. IFIP Congress, 68. |
| 32 | Lawrie, N.L. | Linear programming as an aid to school timetabling. British Joint Comp. Conf., 66. |
| 36 | - " - | Bemerkungen zum kollisionsindexverfahren als Permutationsprinzip Teil 1, 2. Elektron. Datenverarbeitung, 67. |


[41] - " - : A generalization of a method for the construction of class/teacher timetables. IFIP Congress, 68.


Williams, C.W. : Scheduling students by computer. Data process for education. May 64.

