tanulmányok

152/1983

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA

SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATO INTÉZETE

OPERATIONS RESEARCH SOFTWARE DESCRIPTIONS

(Vol. 2.)

Edited by

A. Prékopa and *G. Kéri*

Tanulmányok 152/1983

A kiadásért felelős:

DR VÁMOS TIBOR

Főosztályvezető:

Prékopa András

ISBN 963 311 169 2

ISSN 0324-2951

Hozott anyagról sokszorosítva

8314317 MTA KESZ Sokszorosító, Budapest. F. v.: dr. Héczey Lászlóné

C O N T E N T S

P R E F A C E

This Volume 2 contains 10 papers prepared within the framework of the activity of Working Group No.7 KNVVT (Komissija Naučnye Voprosy Vyčislitelnoĭ Tehniki, in English: Committee of the Scientific Problems of Computer Science). Together with Volume 1 (MTA SZTAKI Tanulmányok 140/1983) 19 papers have been collected, the subjects of which cover almost the whole field of optimization. Program packages and single programs for linear, nonlinear and discrete programming, transportation problems, network flows, problems of optimal control, optimization with multiple objective functions, inventory control problems, nonlinear and discrete approximation are described in these papers.

A wide range of practical applications are also covered by the papers given in these two volumes such as

- economic applications,
- applications in sociology, biology and medicine,
- problems of taxonomy,
- problems of "training by tutor",
- problems in industrial quality control,
- forecasting and prediction problems,
- classification and typology,
- production location of homogeneous products,
- statics of planar bar structures,
- safety stock planning,
- electricity production scheduling,
- highway engineering scheduling,
- other engineering problems (e.g. optimization of chemical reactor systems),
- computer aided design of engineering systems,
- equipment mounting in computing centres etc.

In the first paper of the present volume *H. Bernau, E. Halmos* and *Zs. Soós* deal with the optimal (i.e. minimum weight) design of planar bar structures. Different methods of nonlinear programming are used to solve problems of this kind, coming from the *IKARUS Bus Factory*. In the next paper *M. Černý* and *D. Glückaufová* deal with decision making problems in the presence of multiple (usually conflicting) criteria, and with their computer programs for a small desk computer Wang 2200. The third paper by *G. Christov, T. Encheva, M. Ivanchev, N. Janev, J. Jotcv* and *R. Kaltinska* give the description of their program package LSSP for solving sparse linear programming problems in general and also those of special structures. In the paper by *I. Deák, J. Hoffer, J. Mayer, A. Németh, B. Potecz, A. Prékopa* and *B. Strazicky* a case study is presented concerning optimal daily scheduling of electricity production in Hungary. The model is a large-scale, structured, mixed variable linear programming problem. In the paper by *L. Gömböcz, P. Kelle* and *A. Sebő* a multi-purpose inventory control program package is described. This package was applied for the inventory control problems of the *Danubian Iron Works in Dunaujváros, Hungary*. *E. Jasińska* and *E. Wojtych* tell us in their paper how they solved the problem of sugar-beet distribution by mixed variable linear programming techniques.

One of the two short papers given by *L. Lukšan* contains a description of a new class of methods for linearly constrained discrete nonlinear minimax approximation. The other one contains a short description of the software package SPONA for optimization and nonlinear approximation. It is designed for solving highly nonlinear technical problems with a relatively small number of variables. The main purpose of the paper by *B. Vizvári* is to discuss how to combine exact and heuristic elements to achieve an efficient method for discrete programming. This method can be very useful for the solution of large-scale integer programming problems that cannot be solved by exact methods within reasonable computer time. In the last paper of this volume

S. Walukiewicz describes the ellipsoid algorithm for the solution of linear programming problems. Some new aspects and modifications of the method are considered, such as deep cuts, surrogate constraints, range ellipsoids, different choices of the initial ball. The question that in which cases can the ellipsoid algorithm compete with the simplex method is discussed too. Computational experiences are also given.

Finally we express our thanks to every contributor for their cooperation in the publication of this collection.

The editors

A PROGRAM PACKAGE DETERMINING MINIMUM WEIGHT PLANAR STRUCTURES

H.Bernau (Budapest), E.Halmos (Győr)
Zs.Soós (Budapest)

(Hungary)

1. INTRODUCTION

With the optimal design of planar bar structures the dependence between the forces and the cross-sectional areas cannot be explicitly given. Two approaches evolved to complete tasks for the determination of optimal structures. A detailed survey of the development of these two trends is given in the paper of Venkayya [12]. The main difficulty in both attempt is that the relations between the design variables (cross-sectional areas, surfaces of plates ...) and those describing the behaviour of the structure (tension, stresses, displacements ...) generally cannot be given in an explicit way. This makes the direct application of programming methods very expensive as the calculations of the appropriate functions or their gradients require a complete analysis of the structure. In the models based on optimality conditions, these relations are used in an approximated form, and by the resulting inaccuracies the convergence is hard to be ensured. This remark shows, that the goodness of the approximations for the above mentioned implicit relations is of decisive importance with respect to the efficiency of solution methods [10], [12].

In this paper we will describe two models, which are the basis of the program package. Then we will give a short survey of the purpose, the usage and the moduls of the package.

2. STATEMENT OF THE PROBLEM AND EXPLANATION OF THE MODELS

Assume, that planar frames with minimum weight are to be investigated. It is assumed, that the material of the structure, and the layout of the bars in the frame are fixed in advance. The such cross-sectional areas and moments of inertia have to be determined that for a given external static load in the structure the bar-stresses should not exceed the limit values of stresses characterized by the material used and the total weight of the frame is minimal. About the frame the following will be assumed:

a) the bars are prismatic,
b) the static external loads work only in the nodal points,
c) the displacements in the nodal points are differentially small and their influence on the tension equilibrium can be neglected,
d) stress restrictions ensure the ideal elasticity of the bars,
e) the frame is statically indeterminate.

Under these conditions the behaviour of the loaded frame can be described by the following equation system [1], [11]:

$$\begin{aligned} R(t)y - A^T x &= 0 \\ Ay &= q \end{aligned} \qquad (2.1)$$

where

$R(t)$: the elasticity matrix of the bars dependent upon the cross-sectional areas and moments of inertia $t=(t_1, t_2, \ldots, t_n)$;

A : the matrix of geometric constants;

A^T: the transpose of A;

y : the vector of forces and bending moments;

x : the vector of nodal points displacements;

q : the vector of static external load fixed in advance.

This system yields for every vector t a linear equation system in x and y, i.e. in order to find out for a given vector t, the vector y of forces and moments the equation system (2.1) has to be solved. As the total weight of the frame has to be minimum the following optimization problem emerges

$$\min_{(t_1 \ldots t_n)} \rho \sum_{i=1}^{N} l_i t_i \tag{2.2}$$

$$\sigma_i(y,t) \leq \overline{\sigma}_i \qquad i=1,2,\ldots,N \tag{2.3}$$

$$t_i \geq \overline{t_i} \qquad i=1,2,\ldots,N \tag{2.4}$$

where l_i is the length of the i-th bar and t_i is its cross-sectional area, ρ is the specific weight of the material and $\sigma_i(y,t)$ is the stress in the i-th bar. The values $\overline{\sigma_i}, \overline{t_i}$ are limiting stresses fixed in advance for the bars as well as the lower limits for the cross-sectional areas, the layout of the frame is fixed.

It is easy to see that in this optimization problem the stresses σ_i cannot be given as explicit functions of the vector t, as the vector y results at any time from solution of the basic equation system (2.1). Taking into account, that for every vector $t>0$ the elasticity matrix $R(t)$ is regular, it follows from (2.1)

$$y(t) = R^{-1}(t)A^T x \tag{2.5}$$

from which

$$Ay(t) = AR^{-1}(t)A^T x = q \tag{2.6}$$

follows. As the frame has been assumed statically indeterminate, the matrix A has the full row rank and the matrix

$$C(t) = AR^{-1}(t)\ A^T$$

is thereby also regular for every $t>0$.
Then it results from (2.6), that

$$x(t)=C^{-1}(t)q,$$

and substituting this into (2.5) one gets for the dependence of the vector y upon the cross-sectional areas and moments of inertia the well-known basic relation of the displacement method [1]:

$$y(t)=R^{-1}\ t\ A^{T}C^{-1}(t)q. \tag{2.8}$$

So the optimization problem (2.2), (2.3), (2.4) takes the form:

$$\begin{aligned} &\min_{(t_1,\ldots,t_N)} \sum_{i=1}^{N} l_i t_i \\ &\sigma_i(y(t),t) \leq \bar{\sigma}_i \qquad i=1,2,\ldots,N \\ &t_i \geq \bar{t}_i \qquad i=1,2,\ldots,N \end{aligned} \tag{2.9}$$

where $y(t)$ results from the relation (2.8). As the vector t appears in the inverse of the matrix $C(t)$, $y(t)$ cannot be given as an explicit function. (It should be noted here, that the matrix $R^{-1}(t)$ can be given, on the basis of the diagonal block structure of the matrix $R(t)$ explicitly as a function of t.)

The first model is a developed version of an earlier model elaborated by E.Halmos and T.Rapcsák [6]. In their model the relation (2.8) is approximated by an explicit function of the vector t. The basis of their approximation is the following decomposition of the structure: every moving nodal point of the original structure gets a substructure assigned composed of this nodal point and the bars entering it. At the same time the bar-terminals not belonging to the nodal point will be fixed. For these substructures there are relations analog to

the relation (2.8).

$$y_\gamma(t)=R_\gamma^{-1}(t)A_\gamma^T C_\gamma^{-1}(t)q_\gamma \qquad \gamma=1,2,\ldots,M \tag{2.10}$$

where M is the number of moving nodal points of the original structure. In the cases of planar structures the matrices $C_\gamma(t)$ are of the order 2 or 3, and the inverse of these matrices can easily given from the elements of the matrices $C_\gamma(t)$. Thereby the relations (2.10) yield explicit relations for the dependence of the forces upon the vector t, if the loads q_γ are given. The vectors q_γ $\gamma=1,2,\ldots,M$ are the static loads of the nodal points in the substructures. In the model these loads will be so determined, that for an initial vector $\overset{o}{t}$ the displacements of the nodal points in the substructure coincide with the displacements of the nodal points in the original structure. This coincidence is ensured if the nodal point loads are fixed in the form

$$q_\gamma = C_\gamma(\overset{o}{t})\,[x]_\gamma \qquad \gamma=1,2,\ldots,M \tag{2.11}$$

where $[x]_\gamma$ is the displacement of the nodal point in the original structure for $t=\overset{o}{t}$. (These displacements are part of the solution of the system (2.1) for $t=\overset{o}{t}$). These nodal point loads will be considered as constants and the bar forces approximated in the following way

$$\bar{y}^i(t) = \begin{cases} y^i_{\gamma_1}(t) & \text{if the } i\text{-th bar appears only in one substructure} \\ y^i_{\gamma_1}(t)+y^i_{\gamma_2}(t) & \text{if the } i\text{-th bar appear in two substructures } \gamma_1 \text{ and } \gamma_2. \end{cases}$$

For the so defined forces $\bar{y}^i(t)$ it can be proved that for $t=\overset{o}{t}$ these coincide with the forces evolving actually in the original structure.

Two disadvantageous properties could be observed during the application of this approximation:

a) The nodal point loads q_γ from (2.11) chosen as constant in the model depend strongly upon the choice of the initial vector $\overset{o}{t}$. This requires the loads q_γ relative often to be determined anew in order to ensure the feasibility with respect to the stress constraints in (2.9).

b) At the bars connecting two moving nodal points and contained accordingly in two substructures. Both components $y_{\gamma_1}(t)$ and $y_{\gamma_2}(t)$ from (2.12) have for $t=\overset{o}{t}$ often an opposite sign and are absolutely considered relatively large compared with the actual forces in the original structure. For this reason the mechanical properties of the substructure differs strongly also for $t=\overset{o}{t}$ from those of the corresponding bar group in the original structure.

In order to ensure a closer connection between the substructures and the original structure an attempt was made to determine the nodal point loads such that for $t=\overset{o}{t}$ the forces in the substructures coincide with the forces in the original one. It was found out [4] that this coincidence can only be achieved if in the substructures a kinetic load f_γ is introduced for the originally fixed bar terminals. This kinetic load is chosen such that the resulting displacement of those terminals agree with the displacements of the corresponding nodal points in the original structure for $t=\overset{o}{t}$. Furthermore if one requires the coincidence, one gets for the nodal point loads

$$q_\gamma = C_\gamma(\overset{o}{t})\,[x]_\gamma + A_\gamma R_\gamma^{-1}(\overset{o}{t})\,f_\gamma \qquad \gamma=1,2,\ldots,M \tag{2.13}$$

and the forces in the substructures present themselves in the following form

$$y_\gamma(t)=R_\gamma^{-1}(t)A_\gamma^T\left[C_\gamma^{-1}(t)q_\gamma-C_\gamma^{-1}(t)A_\gamma R_\gamma^{-1}(t)f_\gamma\right]+R_\gamma^{-1}(t)f_\gamma \quad (2.14)$$

It can be shown [3] that for $t=\overset{o}{t}$ these forces coincide with the forces in the original structure. Two remarkable properties of this approximation [3]:

1. The nodal point loads defined in (2.13) are in all free moving nodal points independent of the choice of the initial vector $\overset{o}{t}$ and agree with external loads of the nodal points in the original structure. If a nodal point is fixed with respect to certain directions, then the corresponding components of q_γ contain the reaction forces of the original structure on the fixation for $t=\overset{o}{t}$.

2. For $t=\overset{o}{t}$ the composition of the loaded substructures yields the original structure in the loaded state.

After having discussed the first model, let us turn to the so called exact model. A further possibility for the solution of the problem (2.2), (2.3), (2.4) will be given to which the relation (2.8) or an approximation of this will not be required. To this we turn back to the relation (2.5). Considering in this relation besides the vector t also the displacement vector x as variable vector, the stress constraints can be set in the form:

$$\sigma_i(y,t)=\sigma_i(R^{-1}(t)A^Tx,t)\leq\bar{\sigma}_i \qquad i=1,2,\ldots,N.$$

As the matrix $R^{-1}(t)$ can be explicitly given as the function of t, an explicit function of t and x is obtained to determine the bar stresses. But in this case the second equation of basic system

$$Ay = AR^{-1}(t)A^Tx = q$$

has to taken into account in order to ensure the equilibrium of forces in the nodal points and so the optimal design problem

gets the following form

$$\min_{(t_1,\ldots,t_N)} \rho \sum_{i=1}^{N} l_i t_i$$

$$\sigma_i(R^{-1}(t)A^T x, t) \leq \sigma_i \qquad i = 1,2,\ldots,N \qquad (2.15)$$

$$AR^{-1}(t)A^T x = q$$

$$t_i \geq \bar{t}_i \qquad i = 1,2,\ldots,N$$

3. TEST RESULTS OF THE MODELS AND SOLUTION METHODS

Within a contract work with the IKARUS Bus Factory, Budapest, the presented models have been tested for designing various structures. The following optimization methods have been used to solve the corresponding optimization problems:

1) the linearized centrum method [7]:
2) a penalty algorithm SUMT with logarithmic penalty function [8]:
3) a modified Lagrange method [9].

With respect to the models and the applied solution methods the following experiences have been gained:

a) At the "exact model" the direct formulation has the advantage of not requiring any approximation, its drawback is that the number of variables and constraints increases in relation to the problem (2.9). Moreover the addition of equation constraints in problem (2.15) makes the treatment of this problem more difficult than the solution of the problem (2.9).

b) The application of the relation (2.14) yielded, compared with the relation (2.12) for $t \neq \overset{o}{t}$ substantially more accurate approximation for the forces. This caused the need of the updating of the nodal point loads to become more infrequent. The required computation time was for both models about the same.

c) To the solution of the problem (2.15) only the 2nd and 3rd can be applied because of the equation restrictions. At this problem it proved to be advantageous to fix in the particular unconstrained optimization phases the values of the vector t and x alternately, i.e. if in a phase the optimum of the penalty function with respect to the vector t has been found, then after updating the correspondent penalty parameters the values of t have been chosen as constant in the next phase, and the optimization is done with respect to the vector x, and vice versa.

d) With problems of smaller size the required time to solve tasks of type (2.15) amounted to about as much as for tasks of type (2.9). With problems of greater dimension (number of variables 50 and number of constraints in the same order of magnitude) the required computation time for the tasks (2.15) exceeded by far that for the tasks (2.9), yet the tasks (2.15) yielded generally better solution.

e) In all methods numerical gradients have been used. To the inaccuracies resulting therefrom the penalty method seemed to be less sensitive than the other methods.

f) Another advantage of this method is that if the starting point is feasible with respect to inequality constraints, it is ensured that the inequalities in all iteration points will be fulfilled. This guarantees that the vector t remains always positive. In the program for the modified Lagrange method [9] this is not the case and in consequence difficulties arise, as the matrix $R^{-1}(t)$ is not defined if components of t are zero or negative.

On these experiences the following strategy can be suggested to solve the design problem. Starting from an initial vector $\overset{o}{t}$, the use of the approximation (2.14) in the problem (2.9) yields an approximate solution. (Solving the problem (2.9) it can be necessary to update the nodal point loads q_γ and kinetic loads f_γ sequentially.) Thus the obtained solution t^1 as well as the vector x^1 resulting from the relation

$$x_\gamma^1 = C_\gamma^{-1}(t^1) q_\gamma \qquad \gamma = 1, 2, \ldots, M$$

can be used as the starting point in the problem (2.15) to find more accurate solution.

4. THE PROGRAM PACKAGE TO SOLVE THE OPTIMUM DESIGN PROBLEM

The developed and implemented system RUDMER has the following tasks:

1. To create the basic file containing the parameters of the planar structure.
2. To make the necessary modification on the basic file, if the user wants.
3. To solve the optimization problem using (2.9) model.
4. To solve the optimization problem using (2.15) form of the design problem.
5. To solve the (2.1) equation system for a fixed $\overset{o}{t}$ vector.

The function of the package is the following:

1. The program PRODUCE makes the first task. The user have to give the following parameters of the structure: the number of the bars and nodal points; the x and y coordinates of the nodal points; the indices of the connected nodal points for all bars; a two dimensional sign vector, whether the nodal point can move away into the directions x and y; and the external loads. The program creates the necessary basic file

from these informations, and sets up the matrices $R(t)$, $R^{-1}(t)$, A, and vector q.

2. Sometimes modification is needed and the modul CHANGE makes this task.

The possible modifications are:

a) new bar addition,

b) new nodal point addition (and some new bars, of course),

c) delete of a bar,

d) delete of a nodal point,

e) change of a nodal point fixing.

3. The program APROPT solves the (2.9) model using the (2.14) approximation.

In the first part creates the $R_\gamma^{-1}(t)$, A_γ matrices and q_γ, f_γ vectors for all nodal point, and sets up the constraints of the problem. Then solves the (2.9) nonlinear programming problem. The run of the ALAP to solve the (2.1) equation system is needed before the start of the APROPT.

4. The program PONTOPT solves the (2.15) nonlinear programming problem (second model).

5. The program ALAP solves the (2.1) equation system for $\overset{o}{t}$ fixed in advance, gives the bar stresses and prepares the necessary informations and datas for the running of the program APROPT.

The method to solve the nonlinear programming problem is the SUMT (Sequential Unconstrained Minimization Techniques) elaborated by Fiacco and McCormick. The package was implemented on a CDC 3300 computer in FORTRAN IV language.

REFERENCES

[1] J.H.Argyris, Die Matrizentheories der Statik; Ingeneur. Archiv. XXV (1967), pp. 177-192.

[2] H.Bernau, E.Halmos, Dimensioning of statically indeterminate lightweight structure of complex stress on the basis of minimumweight conditions; Working Paper MO/21, Comp. and Aut. Inst. H.A.S. (1980)

[3] H.Bernau, E.Halmos, Ein Modell zur Bestimmung optimaler Stabwerke; ZAMM 61 (1981) T329-T330.

[4] H.Bernau, E.Halmos, Zs.Soós, Ein Zerlegungsprinzip zur Bestimmung optimaler Stabwerke; Working Paper MO/23, Comp. and Aut.Inst. H.A.S. (1981)

[5] C.Fleury, M.Geradin, Optimality Criteria and Mathematical Programming in Structural Weight Optimization; Comp. and Struct. 8 (1978) p.. 7-17.

[6] E.Halmos, T.Rapcsák, Minimum Weight Design of the Statically Indeterminate Trusses; Math. Progr. Study 9 (1978) 109-119.

[7] P.Huard, Programation Mathematique Convexe; Bulletin de la Direction des Etudes et Recherches EDF Serie 1, (1968) 61-74

[8] F.A.Lootsma, The Subroutine MINI for Solving Nonlinear Optimization Problems; Philips International Inst. for Technological Studies, Eindhoven (1974)

[9] D.A.Pierre, M.J.Lowe, Mathematical Programming via Angmented Lagrangians: An Introduction with Computer Programs; Addison-Wesley Publ.Co., Inc. Reading Mass (1975)

[10] G.Sander, C. Felury, A Mixed Method in Structural Optimization; Int. Journal for Num. Methods in Eng. 13 (1978) pp. 385-404.

[11] J.Szabó, B.Roller, Rudszerkezetek elmélete és számitása; (Theory and Computation of Structures) Müszaki Könyvkiadó, Budapest, (1971).

[12] V.B. Venkayya, Structural Optimization, A Review and Some Recommendations; Int. Journal for Num. Methods in Eng. 13, (1978) pp. 203-228.

PROCEDURES FOR MULTICRITERIUM DECISION PROBLEMS ON A PROGRAMABLE CALCULATOR

M.Černý, D.Glückaufová

(Praha, Czechoslovakia)

1. INTRODUCTORY REMARKS

The problems of Multiple criteria decision making (MCDM) refer to making decisions in the presence of multiple usually conflicting criteria. Problems involving multiple criteria decision making are of common occurrence in everyday life. For example, in a personal context, the job one chooses may depend upon its prestige location salary, advancement opportunities, working conditions and so on. In a public context, the water resources development plan for a community should be evaluated in terms of cost, probability of water shortage, energy, recreation, flood protection, land and forest use, water quality etc.

One may state that there exist two different sets of MCDM problems due to the problem setting: one set contains problems involving finite number of elements (alternatives) and the other consistsof problems with infinite number of potential alternatives. The problems of Multiple criteria decision making (MCDM) can be therefore broadly classified into two categories in this respect:

- Complex evaluations of alternatives.
- Vector optimization.

The distinguishing feature of the problems belonging to the first group is that there is usually a limited (and rather small) number of predetermined alternatives. The alternatives have associated with them a level of the achievement of the attributes (characteristics), which may not necessarily be

quantifiable. The final selection of the "best" alternative is made with the help of inter and intra attribute comparisons.

Vector optimization problems are not associated with the problems where the alternatives are predetermined. The common features of vector optimization problems are that they possess:
- a set of quantifiable objectives,
- a set of well defined constraints.

2. PROGRAM SUPPORT FOR MCDM PROBLEMS

The nature of the MCDM problems requires the possibility of the flexible interactions among decision maker, analyst and computer in the whole process of solving the problem. Recently there became available various computing systems which make such interactions possible. Screen terminals and graphical displays connected to the computer are weel-known examples of such devices. Even better contact with the user give small computers of the desk type, which are now well-spread. In our institute we have at our disposal computer Wang 2200 VP, which has proved very useful for solving small and medium sized problems of MCDM.

The procedures for solving MCDM problems which we have developped on our computer form three different groups according to the nature of the problems solved:
- procedures supporting vector optimization problems,
- procedures for complex evaluation of alternatives,
- procedures used for the formalized analysis of the set of criteria.

Procedures for vector optimization problems must be based on a reliably working program for solving corresponding one criterial problems. That is why we have limited ourselves for the time being to multiobjective linear programming problems and to the problems of choice from a finite set. From the existing methods of the vector optimization we have chosen a modification of a so called STEM method. This method does not

require from the decision maker the explicit formulation on his local or global preference structure; the only necessary information needed concerns the maximum relaxations of the values of some objective functions in order to improve the values of other ones. The method used will be described in more detail in the next section of this paper.

The programming support of a modified STEM method has a form of a system of program modules, making it possible to solve the problems with 10 objective functions at most. Apart from this the multiple objective LP problems solved by this system can have up to 60 variables and 30 constraints; in the problems of choice from the finite set this set can have in the present program version maximally 120 elements.

The common feature of the majority of methods for complex evaluation of alternatives is the existence of subjective factors. One of the possible ways how to objectivize the results is the simultaneous application of several methods. Therefore it is convenient to build the programming support for those methods in the form of the system of procedures operating on a common data base.

The set of programs for the complex evaluation of alternatives consists of the procedures realizing:
- the method of basic alternative
- method AGREPREF
- method of approximating the fuzzy preference relation
- the Electra III method.

These programs make it possible to evaluate up to 40 alternatives according to 20 criteria. The programs communicate with user by asking for new or improved values of the weights of criteria and thresholds of sensitivity. The methods realized in the set are described in more detail in the section 4 of this paper.

The methods for the formalized analysis of the set of criteria are based on the assumption that the values of criteria on a finite set of alternatives are given.

Consequently their programming support is built in much the same way as the procedures for the complex evaluation of alternatives, i.e. on the common data basis. The programs make it possible to compute the coefficients of similarity or distance, the Kendall's and Spearman's rank correlation coefficients and the coefficients of consistency. The set contains also the program for determining the weights of criteria by the Saaty's method.

Apart from the simple approaches mentioned above the formalized analysis can be performed with the help of more complicated methods like GUHA method or cluster analysis method. These approaches however require more capacity and time and moreover their programming support exists on large size computers. Therefore we have not included them into our program system.

3. VECTOR OPTIMIZATION PROBLEM: MODIFIED STEM METHOD

The problem of vector optimization solved by the modified STEM method can be formulated as follows:

$$f_k(x) \rightarrow max \qquad (k=1,\ldots,r)$$
$$f_k(x) \rightarrow min \qquad (k=r+1,\ldots,m) \qquad 0 \leq r \leq m$$

subject to $x \in X$ (a feasible set).
Present state of programs makes it possible to solve two special cases of such problems:

1. Multiobjective linear programming problem, where

$$X = \{x \mid Ax = b, \quad x \geq 0\}$$
$$f_k(x) = c_k' x + d_k$$

2. Selection from a finite set of alternatives, where

$X=\{x_1, x_2, \ldots, x_n\}$

$f_k(x_i)$ - *given values.*

Let us note that this formulation of vector optimization problem is somewhat more complicated than the commonly used form. It would be of course possible to omit the constant terms in the objective functions and to assume (e.g.) that all functions are maximized. Such a transformation is of course made in the computation phases of the algorithm, but in the process of interaction with the DM it is better to stick to original expression of the functions, so that the DM is not forced to express himself in transformed values which may represent an unnecessary simplification to him. The process begins (as it is usual in STEM-type methods) by constructing a so called payoff matrix consisting of the elements

$z_{ik}=f_k(x_i^*)$ $(i,k=1,2,\ldots,m)$ where x_i^* solves the problem

$$f_i(x) \to max \ (min) \qquad (1)$$

subject to $x \in X$.

The diagonal elements $z_{kk}=z_k^*=f_k(x_k^*)$ represent the so called ideal values of objective functions.

The provisional or compromise solution computed by the analyst at each iteration step is obtained by solving the following problem (q denotes the number of iteration step):

$d \to min$

subject to

$$
\begin{aligned}
&x \in X, \\
&f_k(x) + v_k d \underline{\geq} z_k^* \quad (k \in K_1^{(q)}, k \underline{\leq} r) \\
&f_k(x) - v_k d \underline{\leq} z_k^* \quad (k \in K_1^{(q)}, k \underline{\geq} r) \qquad (2) \\
&f_k(x) \underline{\geq} h_k^{(q)} \quad (k \in K_2^{(q)}, \; k \underline{\leq} r) \\
&f_k(x) \underline{\leq} h_k^{(q)} \quad (k \in K_2^{(q)}, \; k > r).
\end{aligned}
$$

It is obviously the problem of minimizing the maximum deviation of an objective function from its ideal value. Here $K_1^{(q)}$ is the set of indices of those objective functions, the values of which are not yet marked by the decision maker (DM) as satisfactory, $K_2^{(q)}$ is the set of other objective functions. The limit values $h_k^{(q)}$ are determined as follows:

$$
\begin{aligned}
h_k^{(q)} &= h_k^{(q-1)} \underline{\pm} \Delta_k \quad (k \in K_2^{(q)} \quad K_2^{(q-1)}) \\
h_k^{(q)} &= f_k(x^{(q-1)}) \underline{\pm} \Delta_k \quad (h \in K_2^{q} - K_2^{(q-1)})
\end{aligned}
$$

where Δ_k is the amount of relaxation given by the DM.

The weights v_k are given by the DM who can choose one of the three possibilities:

1) $v_k = 1$ for all k,

2) $v_k = z_k^*$ for all k,

3) v_k = arbitrary ($\neq 0$).

The system of programs consists of five modules: the general program and special programs handling the data input and calculation steps solving the problems (1) and (2) for both above mentioned type of problems.

4. COMPLEX EVALUATION OF ALTERNATIVES

The complex evaluation of alternatives problems can be mathematically formulated in the following way:

Let $R_1,\ldots,R_m$ denote preference relations defined on a finite set of alternatives X. The preference relations R_i correspond either to different members of decision making collective, or to different viewpoints, from which the alternatives are evaluated. Our task is to find an aggregated relation R expressing the resulting preference. As the resulting preference should serve to order the set of alternatives, it is natural to require that the relation should be transitive, at least in some weaker sense. Ideally the resulting preference relation should be a complete ordering of the set of alternatives X. However it appears, that in modelling resulting preference relation it is sufficient to derive a relation which has somewhat weaker properties. There exist some very simple methods for aggregating individual criteria, the relative importance of which is expressed by means of numerical weights (e.g. the method of basic alternative, see Černý, Glückaufová, Toms 1980).

More sophisticated methods based on the concept of threshold of sensitivity make use of a fuzzy preference relation. Fuzzy preference relations S on a given set X of alternatives is defined as a fuzzy subset of the Cartesian product XxX. The membership function of a fuzzy relation S can be written as $u_S(x,y)$ where $x\epsilon X$, $y\epsilon X$ are interpreted as a degree of validity of the relation S for the pair (x,y). In modelling preferences a notation S_{xy} is frequently used instead of $u_S(x,y)$. It is usually assumed, that for any pair of alternatives (x,y) it holds:

$$S_{xy} + S_{yx} \leq 1.$$

The number $S_{x\sim y}=1-S_{xy}-S_{yx}$ can be interpreted as a degree of

indifference between x and y. The numbers $S_{x\sim y}$ define another fuzzy relation of indifference. This relation will be denoted by $\{S_{x\sim y}\}$; to distinguish we shall write $\{S_{xy}\}$ for the original fuzzy preference relation S. If $\{S_{x\sim y}\}$ is an empty relation, i.e. if $S_{xy}+S_{yx}=1$ for all (x,y), then S_{xy} is called a strict preference relation.

To each fuzzy preference relation $\{S_{xy}\}$ a strict preference relation $\{S^*_{xy}\}$ can be defined as follows:

$$S^*_{xy} = S_{xy} + \frac{1}{2} S_{x\sim y}.$$

Therefore we shall limit our attention from now on to strict fuzzy preference relations. The concept of transitivity, which plays a fundamental role in the theory of preference, can be extended to fuzzy preference relation in several ways. We shall give here the following definition:
A strict fuzzy preference relation is called transitive if for any triple of alternatives (x,y,z) it holds:

$$S_{xy} \geq \frac{1}{2}, \quad S_{yz} \geq \frac{1}{2} \rightarrow S_{yz} \geq max\ (S_{xy}, S_{yz}).$$

The fuzzy preference relation is in a sense the best tool to picture the real preferences in a formal way. However, to solve decision problems, it is often necessary to replace the fuzzy relation by a nonfuzzy one. This nonfuzzy relation can be assigned to the fuzzy relation in different ways. The simplest way would be to define the nonfuzzy relation $R=(P,I)$ as follows:

$$xPy \iff S_{xy} > S_{yx}$$

$$xIy \iff S_{xy} = S_{yx} = \frac{1}{2}.$$

As a generalization of the definition given above a whole class of nonfuzzy relations $R=(P_\alpha, I_\alpha)$ depending on a so called threshold of sensitivity α can be defined. Let α be a real number from the interval $\langle \frac{1}{2}, 1 \rangle$; then we shall define:

$$xP_\alpha y \iff S_{xy} > \alpha$$

$$xI_\alpha y \iff 1 - \alpha \leq S_{xy} \leq \alpha.$$

The nonfuzzy preference relation obtained in this way will usually serve to order the set of alternatives in some way. Therefore it is natural to require that the relation should be transitive at least in some weaker sense. Ideally, the nonfuzzy preference relation obtained should be a complete ordering of the set of alternatives X. However, it appears that in modelling preferences it is sufficient to derive a relation which has the properties of semiorder (see e.g. Luce, 1956).

Roberts (see Roberts, F.S., 1971) has proved the following theorem:

If a strict preference relation $\{S_{xy}\}$ is transitive in a sense defined above, then for any $\alpha \in \langle \frac{1}{2}, 1 \rangle$ the relation $R_\alpha = (P_\alpha, I_\alpha)$ is a semiorder.

The most of fuzzy preference relations obtained by directly aggregating individual preferences do not satisfy the requirement of transitivity which is fundamental in the above theorem. This fact leads to the construction of the so called method based on the approximation of fuzzy relation included in our system of methods for complex evaluation of alternatives. The main purpose of this method is to find the closest transitive fuzzy relation to the obtained one. According to the Robert's theorem it follows, that the corresponding nonfuzzy relation has the properties of a semiorder.

The other possibility how to handle the problem is to find to a nonfuzzy relation R (obtained from a fuzzy relation in the way described above) a relation $\bar{R}$ which has the proper-

ties allowing the ordering of alternatives and which is in some way the closest to the obtained relation R. The closeness of the relation $(R,\bar{R})$ can be measured for example by a distance function

$$d(R,\bar{R}) = \Sigma |R_{xy} - \bar{R}_{xy}| .$$

The problem of finding the relation $\bar{R}$ which minimizes $d(R,\bar{R})$ can be formulated as a bivalent programming problem the constraints of which depend on the requirements imposed on the relation $\bar{R}$. As such a problem is extremely complex, some approximation algorithms based on other approaches were suggested where the degree of closeness of resulting $\bar{R}$ to the relation R is measured by the so called coefficient of approximation

$$k = \frac{p + i}{n(n-1)/2},$$

where n is a number of alternatives, p is number of pairs of alternatives x,y for which xPy as well as $x\bar{P}y$ is valid; i is the number of pairs of alternatives (x,y) for which xIy as well as $x\bar{I}y$ is valid.

The best known algorithms of this class are AGREPREF (see Lagreze, 1974) which gives a semiorder as $\bar{R}$ and the whole group of so called Electra methods (see e.g. Roy 1968) which results in a pair of quasiorderings. In all methods mentioned above the fuzzy preference relation is arrived at by aggregating the family of preference into a single preference.

Unlike most applications of fuzzy sets, the values of membership function $u_S(x,y)$ in this case can be found in a natural "objective" way:

$$u_S(x,y) = S_{xy} = \sum_{i \in I_{xy}} p_i ,$$

where I_{xy} is a subset of indices $I=\{1,\ldots,m\}$ containing all indices i, such that xP_iy and p_i is a weight assigned to i-th subject (characteristic). The fuzzy relation obtained in this way can be replaced by a nonfuzzy relation R or a class of nonfuzzy relations R_α and approximated by the relation having desired properties in a way mentioned above.

A weak point of the class of methods just discussed is the use of thresholds. Their values are rather arbitrary, although their impact on the final solution may be significant. For example if we take the threshold values rather ambitious (for complete dominance) then it may be difficult to eliminate any of the alternatives; by relaxing the thresholds values we can reduce the number of nondominated solutions to the single one. The fact that the sensitivity thresholds are exogeneously determined by DM brings certain subjective factor into the algorithm.

A recently proposed method Electra III on the other hand does not require the threshold to be given exogeneously; the values of the sensitivity thresholds are generated in an iterative way by the algorithm itself.

5. THE FORMALIZED METHODS FOR THE ANALYSIS OF THE SET OF CRITERIA

Most papers dealing with multiple criteria problems regard the criteria as given and confine the analysis to a decision of how to achieve the solution.

When dealing with multiple criteria problems the decision maker usually says he considers many criteria, although often only few of them are essential.

Many authors (see e.g. Fishburn 1964) have found that normally at most 4 to 7 criteria are important, as working with few measures of effectiveness from the beginning increases the chance of success, there being less spread in necessary data. Other reasons for using few criteria include lower cost

and greater ease in estimating and using the multiple criteria model.

The problem of handling a large number of criteria has not been very much treated in multiple criteria literature. Fishburn (1964) discusses three approaches to this problem.

The first is to select a subset (about 6 to 10) of criteria most important to the decision maker and to use only these in the analysis.

Another method is first to select a subset of the more important criteria and to analyse the alternatives in terms of these criteria. Then another subset of criteria is added to those initially used and the alternatives are analysed with respect to this larger subset of criteria. If the results of this second analysis agree with those of the first one, the decision maker may be satisfied with the analysis. If the outcomes of the analysis differ, the decision maker adds a new subset of criteria and repeats the analysis.

The third approach for reducing a great number of criteria suggested by Fishburn is to select any subset of criteria and analyse the alternatives with respect to these criteria. Next a new subset of criteria is selected and processed and this process is repeated several times.

If the results of the analysis of the alternatives are not dependent on the various sets of criteria, the decision maker may use any of the subsets in his analysis.

All these methods for reducing the number of criteria exclude in various ways a subset from the analysis. The methods do not give any information on what effect this reduction will have on the decision. For all this reason some detailed analysis of the set of criteria is needed.

To study the relation between just two criteria some simple approaches based on the representation of criteria by incidence matrices are used.

Another possibility how to test the mutual relation between just two criteria is the use of rank correlation methods, especially the use of the disarray coefficient.

While the use of the coefficients mentioned above requires the transformation of the criteria into binary matrix (corresponding to the pairwise evaluation of the alternatives) the use of disarray coefficient requires to represent the criteria as rankings of alternatives.

The analysis of more than two criteria is usually performed for the following reasons:

- to reduce the number of criteria,
- to find out which criteria or which subsets of criteria influence to the greatest extent the solution,
- to test the set of criteria for consistency,
- to examine mutual relations between subsets of criteria.

To solve the majority of the problems mentioned the method of automatic generating of hypotheses (GUHA) can be used.

This method is applicable to all problems in which it is required to obtain unknown laws, relations or causal connections. Its usefulness consists in the combinations of the formal apparatus of mathematical logic, the operational capabilities of computers and of methodology of scientific research. The means of mathematical logic make it possible to find a suitable class of formalized statements to which the investigation of the model can be confined. The means of computer technique make possible to generate and verify all these formulas authomatically in a suitable ordering. As the output there we will appear all hypotheses true or almost true.

If the problem is just to divide the whole set of criteria into groups, elements of which are in some sense close to each other, the method of cluster analysis can be applied.

Cluster analysis is concerned with very general problem of grouping the entities of a given set into homogenous and well separated subsets, called clusters. To define a particular

cluster analysis problem it is necessary to precise the concepts of homogenity and separation. There exist different ways to construct dissimilarities from measurement of characteristics (i.e. from the values of criteria on the set of alternatives). One of the possibilities is to calculate simply the intercorrelations of the criteria; another possibility is to find for any pair of criteria disarray coefficients (see Kendall (1955)).

Another possibility how to use the rank correlation approach is to test the consistency of the set of criteria using Kendall's coefficient of concordance as a measure of agreement of m rankings:

$$W = \frac{12S}{m^2(n^3-n)}$$

All the techniques mentioned above have been tested on real life multiple-criteria problems.

REFERENCES

[1] Benayoun, R. and others: Linear Programming with Multiple Objective Functions: Step Method (STEM), Mathematical Programming 1, 1971, p.366-375.

[2] Černý, M., Glückaufová, D., Toms.M.: Metody komplexního vyhodnocování variant, Academia Praha, 1980.

[3] Černý, M., Glückaufová,D.: Aplikace metod vícekriteriálního vyhodnocování, SNTL Praha, 1982.

[4] Fishburn, P.C.: Decision and Value Theory, Wiley, New York, 1964.

[5] Glückaufová,D.: Některé nové směry vývoje metod vyhodnocování (ELECTRA III), EMO 1, 1982.

[6] Kendall,M.C.: Rank Correlation Methods, Charles Griffin, London, 1955.

[7] Lagreeze,E.J.: How we can use the notion of semiorders to build outranking relation in multicriteria decision making. Metra, Vol XIII, No 1, 1974.

[8] Luce, R.D.: Semiorders and a theory of utility discrimination, Econometrica, Vol. 24, 1956.

[9] Metoda GUHA, Skriptum vydané ke stejnojmennému kursu. Československá kybernetická společnost při ČSAV. Dům techniky ČVTS České Budějovice, 1976.

[10] Roberts, F.S.: Homogeneous Families of Semiorders and the Theory of probabilistic Consistency, Journ. of Math. Psych., 8, 1971.

[11] Roy, B.: Classement et choix en présence de points de vue multiples, RIRO No 8, VI. 1968.

PROGRAM PACKAGE LSSP FOR LINEAR PROBLEMS WITH SPARSE OR STRUCTURED MATRIX

G.Christov, T.Encheva, M.Ivanchev,
N.Janev, J.Jotov, R.Kaltinska

(Sofia, Bulgaria)

I. DESTINATION

The program-package LSSP is a product of Operation Research Department at the Centre of Mathematics and Mechanics of the Bulgarian Academy of Science that provides the ability to solve on ES computing system the following optimization problems:

- linear programming problems with a sparse matrix;
- plant-location problem;
- distribution problem of special type.

II. PROBLEM DESCRIPTION

1. Linear programming problem with a sparse matrix minimize or maximize $L(x) = \sum_{i=1}^{n} c_i x_i$ subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \lessgtr b_i \qquad i=1,2,\ldots,m$$

$$d_j \leq x_j \leq u_j \qquad j=1,2,\ldots,n$$

Lower bounds d_j are 0 or $-\infty$ (omitted) and upper bounds u_j are arbitrary real numbers. It is assumed that the relative number of nonzero a_{ij} is small, i.e. the matrix $||a_{ij}||$ is sparse. The program for solving the problem given, named SPARSE, is a realisation of the revised dual simplex method with bounded variables given in [1].

For presentation of the matrix $||a_{ij}||$ as for basis inverse in core only the nonzero elements are used. For limiting the computational errors, after a prescribed number of iterations the reinversion of the basis is done. The calculations are performed without any use of auxiliary memory, which is used only for recording and for correcting the input data.

2. Plant-location problem.

The mathematical model of this well-known linear integer optimization problem is:

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} d_i y_i$$

subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j=1,2,\ldots,n$$

$$0 \leq x_{ij} \leq y_i, \qquad i=1,2,\ldots,m; \quad j=1,2,\ldots,n$$

$$y_i \in \{0,1\}, \qquad i=1,2,\ldots,m$$

This problem could be solved by the program, called PLANLOC, which is a realisation of the s.c. integer simplex algorithm given in [2]. Geometrically, the algorithm starts from an extreme point of a polyhedron (the convex hull of the feasible solution of relaxed problem) and moves towards the optimal point on a path of edges, connecting some feasible solutions of the original problem. The existence of such a path is asserted from the theory. Relatively large problems ($n,m=100$) could be solved entirely in core (256K) because of a special representation of the basis inverse, only small portion of which (nxn) is maintained.

3. Distribution problem of special type.

This problem arises when a capacitated location problem is relaxed in order to be solved by branch and bound techniques. The mathematical model is

minimize $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$

subject to: $\sum_{i=1}^{m} x_{ij} = b_i \qquad j=1,2,\ldots,n$

$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij} \leq B$

$x_{ij} \geq 0$

There are also real problems (medical, production area, etc.) which could be presented in terms of this model.

The program for solving the problem is called TECHNO and is based on an algorithm described in [3]. The algorithm is one of simplex type but is specially designed to exploit the structure of the problem. Thus the program is highly efficient and has small time and storage requirements not only for problems of moderate size but also for large problems.

III. ORGANIZATION OF THE PACKAGE

Each of the problems listed could be stored as a phase in core-image library and runned with minimal user's effort. When the subroutines are stored in relocatable library, they could be invoked from the user's written codes. This could be done because of the unification of the structure and the functions of the programs forming the package.

Each program fulfils the following functions:

1) Data loading from arbitrary input device (card reader, magnetic tape, disk) with syntactical control. If errors are found the error message is printed and the program run is canceled.
2) Dynamic storage allocation depending on the size of the problem.
3) Solving of the problem.

4) Printout of the input data and solution. The volume and the type of this information is controled by the users by means of parameters.

Each program is written in FORTRAN except for the ASSEMBLER modules which control the dynamic storage allocation. The program structure is:

- main program: reads problem's dimensions, calls dynamic storage allocation routine and transfers the control to the main subroutine;
- main subroutine: reads and checks the input data, solves the problem, prints the input-output information;
- auxiliary subroutines: provide service for the main subroutine.

IV. IMPLEMENTATION

The package is implemented in the computing center of the University of Sofia and is used in education of the students in mathematics. It could be used without any restrictions in all areas where the above mentioned problems arise.

LITERATURE

[1] A.Tucker, Linear Inequalities and Systems, Annals of Mathematics Study, No.38, Princeton University Press.

[2] Н.И. Янев, Журнал вычислительной математики и математической физики, №3, 1981, 626-634.

[3] Г.Х. Иванов, Доклади на XI пролетна конференция на СМБ, Слънчев бряг, 1982.

OPTIMAL DAILY SCHEDULING OF ELECTRICITY PRODUCTION IN HUNGARY

I. Deák, J. Hoffer, J. Mayer, A. Németh
B.Potecz, A. Prékopa, and B. Strazicky

(Budapest, Hungary)

1. INTRODUCTION

At the Operations Research Department of the Computer and Automation Institute of the Hungarian Academy of Sciences there has been for several years a work in progress together with the experts of the Hungarian Electricity Boards Trust to apply operations research in the electricity power industry. In the course of this work the model and computer program system to be described in this paper (which can be considered as a case study) has been completed. Starting from the verbal statement of the problem we have arrived, through a large number of steps at the solution of the real problem with real data. These steps are: clarification of every detail of the physical problem, adequate mathematical modelling of the problem, building up the data system required for the mathematical model, preparation of a program system, using the permanent data base,suitable for producing the numerical data of the actual problem to be solved. In the course of the modelling, a kind of problem formulation, describing the reality well enough had to be found, enabling at the same time the problem to be handled computationally. The completed model leads to a large-scale mixed variable linear programming problem where the integer variables are of 0-1 type. A method had to be worked out on the CDC 3300 computer that gives a nearly optimal solution to the problem in an acceptable time. The computer program

system was required to present the results in the form prescribed by the user.

Characteristic for the entire work has been the constant co-operation among the experts of the two intsitutes resulting in a permanent corrective activity in the subsequent stages.

2. FORMULATION OF PHYSICAL PROBLEM

2.1. The overall electric power demand of the country as considered for each day separately as a function of the time is illustrated on Fig.1. where the shape of the curve is characteristic. The time corresponding to the initial point of the curve is the so-called evening peak load time. This is followed by a time interval with decreasing load, thereafter by some hours when the value of the demand differs from the minimum value to a little extent only, thereafter a stage with increasing load - and the whole is repeated once more. The shape of the curve is in every case of this type, but the length of the intervals as well as the demand values change daily.

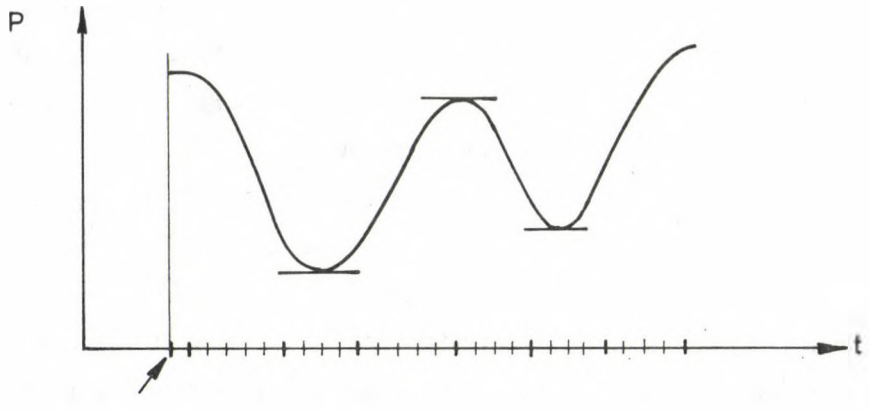

Fig.1.

A typical daily electric power demand function

The electric power demand of each day can be forecasted in advance with an accuracy of 1-2% on the basis of the data available on the day before. We investigate always the 25 hours period following the evening peak, this is subdivided into 23 one hour and 4 half-hour periods in which periods the demand can be assumed constant. The demand contains the estimated values of the power plant's own consumption and of the network losses.

2.2. The electric power demand is satisfied by the electric power generated in the country's power plants and from the neighbouring countries imported power. In our country there are about 20 such power plants that are considered in the model. The electric power imported from abroad in international co-operation is considered as one power plant with constant production.

In the power plants the power is generated by the combined operation of various aggregates in different modes of operation. Each mode of operation involves the combined work of certain aggregates. The applicable modes of operation and the physical quantities characterizing them are given for each power plant.

The given mode of operation of a power plant can run within given power limits and the production cost, as a function of the power level, is a function illustrated on Fig.2. This can fairly well be approximated by a piecewise linear function (Fig.3) where for the slopes the relations

$$c_1 < c_2 \ldots < c_k$$

always hold.

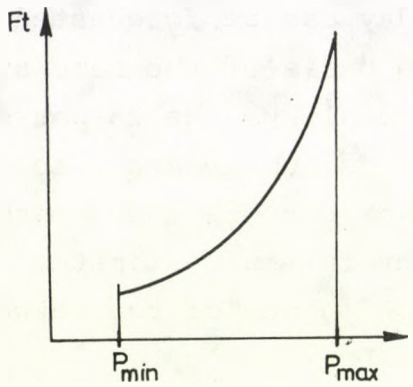

Fig.2.

Production cost function

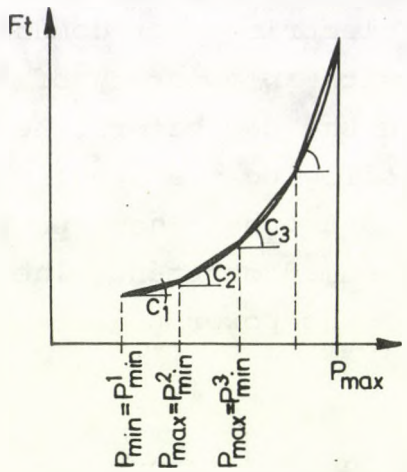

Fig.3.

Piecewise linear approximation of the production cost function

The change-over among modes of operation - start or shut off at least one of the generators - causes the turn of a mode of operation. Thus the change-over is not allowed among all possible modes of operation of a power plant, viz. not among those working with entirely different devices. An accidental failure or maintenance of the equipment can result in the daily change of the modes of operation in the power plant. Fig.4. shows an example of the modes of operation, and in Fig.5. we can see the function of still stand cost.

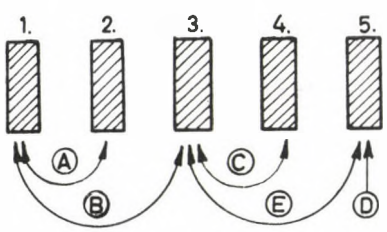

Fig.4.

An example for the definition of the modes of operation

1-2-3-4-5 denote generators, A-B-C-D-E are possible modes of operations, where the arrows indicate the generators that work in the given mode of operation. A direct change for example between the modes C and D is not allowed, but from C to E (it is a start of generator 5.) and from E to D a direct change is possible (shut off of generator 3.).

The electric network of the country is a set of nodes and branches. Its nodes are either power plants or points in which the power demands occur, and its branches power transmission lines and tranformers with given physical characteristics. Some of the network's nodes can be connected to power stations and from almost all the consumer's demands are supplied. Also the electrical network can (and does) daily change on account of maintenance, failure etc. Change means here that certain branches or nodes do not belong to the system on a given day, or the value of their physical characteristics differ from those in case of normal operation.

2.3. With this knowledge our task is to determine for each period of the following 25 hour duration the modes of operation to be applied in the different power plants and their production levels so that the power demand should be satisfied in each period, the physical restrictions on the actual network hold, moreover the so-called fuel contraints be satisfied with a minimum power production cost. The fuel constraints require that in some power plants the value of the daily overall production - directly connected with fuel consumption - should differ from a given value only to the extent of a given very small percentage. The reason of this restriction can be that we cannot consume more than the existing amount of fuel or that certain amount of fuel is expected to arrive on the next day and the storage capacity is limited.

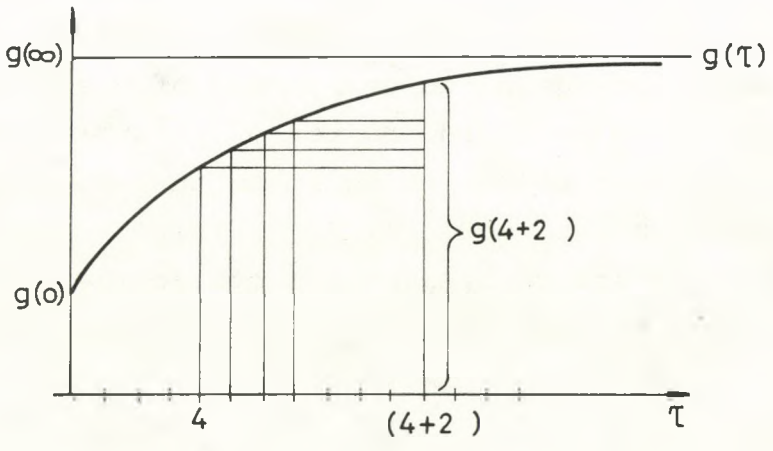

Fig. 5.
Stillstand cost function

The power production cost contains the actual production cost, the change-over cost resulting from the switching of modes of operation resp. standstill and restart of the machines, as well as the cost of loss of power in the network.

3. ASSUMPTIONS

Because of the sophisticated nature of the whole power system to be optimized we had to make some assumptions (simplifications) in order to obtain a model that can be handled.

3.1 By knowing the shape of the demand function we agree that in the first periods when the value of the demand does not increase we allow only such a change of the mode of operation which can be realized by shutting off a generator or generator groups. These periods together are called stop or shut off phases. No change in the mode of operation is allowed in the altogether 4 periods around the period with minimum demand (phase of stagnation); only the production level of the given mode of operation can be changed. In periods of increasing demand only such change of mode of operation is allowed where at least one of the generators is turned on (start periods). The investigated phases are therefore: stop, stagnation, start and once more stop, stagnation and start phases.

In connection with this we agree that at every plant we assign subscripts (integers) to every mode of operation starting from 1 and going up to the number of possible modes of operation at the given plant. We do it in such a way that whenever the transition from mode $j \to k$ $(j<k)$ is possible then from mode j to mode k we arrive by shutting off at least one generator. Note that a transition $j \to k$ is not always possible.

3.2. As a result of physical considerations we have agreed to prescribe the requirements limiting the physical state of the electric network only in the three periods with extreme demands (the first period, the first period of the first stagnation phase and the last period of the first start phase; these will be referred to as voltage check periods). That is, we assume that if in these periods the physical restrictions of the network are satisfied, then in periods of "intermediate" demand with the application of "intermediate" modes of operation (cf. assumption 3.1) the physical restrictions are also satisfied.

3.3. In order to determine the cost of power production the following simplification will be made.

a) The cost functions of the particular modes of operation will be approximated by piecewise linear functions.

b) Symmetric restarting will be assumed for the calculation of the still stand cost arising from the change of modes of operation. This means that if we shut off a generator at ℓ periods before the first period of the stagnation phase, then the restart takes place at ℓ periods after the last period of the stagnation phase, that is the still stand lasts $4+2\ell$ periods. The difference between the actual still stand cost and the approximate value of it will be neglected. The total cost in the $4+2\ell$ periods is subdivided into $4+2\ell$ parts and are assigned to these periods.

c) The cost arising from the network loss will be calculated from the difference between the loss value taken already into account in the demand function and the calculated value of the actual loss depending on the network.

4. MATHEMATICAL MODEL

4.1. The variables of the model. Denote by E the number of power plants and let $m(i)$ be the number of the modes of operation applicable in the i-th power plant $i=1,2,\ldots,E$. Hereinafter superscript t will always refer to the period, $t=1,2,\ldots,27$.

4.1.1. Mode of operation variable. Let x_{ij}^t be 0-1 variable defined as follows, where $i=1,2,\ldots,E$, $j=1,2,\ldots,m(i)-1$:

$$x_{ij}^t = \begin{cases} 0 & \text{if in power plant } i, \text{ in the period } t \text{ the } j\text{-th mode of operation or one with a subscript less than } j \text{ works,} \\ 1 & \text{if in power plant } i \text{ in period } t \text{ a mode of operation with a subscript greater than } j \text{ works} \end{cases}$$

In the sequel we shall use the notations x_{io}^t and $x_{im(i)}^t$ too and define them so that $x_{io}^t=1$ and $x_{im(i)}^t=0$. Note that

1. $x_{i,j-1}^t - x_{ij}^t = 1$ if and only if in power plant i in period t just jth mode of operation works $(j=1,2,\ldots,m(i))$, else $x_{i,j-1}^t - x_{ij}^t = 0$.

2. According to the above definition the variables belonging to the modes of operation of a fixed power plant can take in one period only the values $(\ldots1,1,1,0,0\ldots)$ where the 0 standing in the $1,0$ value exchange is in the jth place if just the jth mode of operation works. Among different periods the right-hand shift of the value exchange $1,0$ corresponds to a

mode of operation exchange reached by a shut off while the left-hand shift of the same corresponds to a start.

3. In the periods belonging to the stagnation phase we have $x_{ij}^{t_o} = x_{ij}^{t_o+1} = x_{ij}^{t_o+2} = x_{ij}^{t_o+3}$, where t_o is the first period of the stagnation phase, therefore it is sufficient to have only $x_{ij}^{t_o}$ among the variables of the model.

We well use, however, the symbols $x_{ij}^{t_o+1}, \ldots, x_{ij}^{t_o+3}$ formally in some relations where the simplicity of the expressions requires them.

4.1.2 Production-level variable. Denote $r(i,j)$ the number of the approximating lines in the approximation of the cost function belonging to the jth mode of operation of power plant i, and P_{ijmin} and P_{ijmax} the minimum and maximum production level of the mode of operation respectively. Denote p_{ijmin}^k, p_{ijmax}^k the power levels belonging to the terminal points of the kth approximating line of the cost function, where $p_{ijmin}^k = p_{ijmax}^k$, $k=1,\ldots,r(i,j)-1$, and $P_{ijmin}^1 = P_{ijmin}$, $P_{ijmax}^{r(i,j)} = P_{ijmax}$ hold. Denote P_{ij}^t the operation level in period t of the jth mode of operation of power plant i. In order to determine it let us introduce the variables

$$P_{ij}^{tk}, \quad i=1,2,\ldots,E, \quad j=1,2,\ldots m(i), \quad k=1,2,\ldots,r(i,j)$$

so that

4.1.2.1. $$P_{ij}^{tk} \geq 0,$$

4.1.2.2. $$P_{ij}^{tk} \leq P_{ijmax}^k - P_{ijmin}^k$$

4.1.2.3. $P_{ij}^{tk} > 0$, only if $P_{ij}^{t\ell} = P_{ijmax}^{\ell} - P_{ijmin}^{\ell}$

for all $\ell<k$, and $x_{ij-1}^{t} - x_{ij}^{t} = 1$.

i.e. if plant i works on the jth mode of operation in period t.

By using these variables the above mentioned level is given by the following sum:

4.1.2.4. $$P_{ij}^{t} = (x_{i,j-1}^{t} - x_{ij}^{t}) P_{ijmin} + \sum_{k=1}^{r(i,j)} P_{ij}^{tk}.$$

The production of power plant i in the period t is equal to

4.1.2.5. $$P_{i}^{t} = \sum_{j=1}^{m(i)} P_{ij}^{t} = \sum_{j=1}^{m(i)} \{(x_{i,j-1}^{t} - x_{ij}^{t}) . P_{ijmin} + \sum_{k=1}^{r(i,j)} P_{ij}^{tk}\}.$$

The daily production equals

4.1.2.6. $$P_{i} = \sum_{t=1}^{27} a_{t} . P_{i}^{t} = \sum_{t=1}^{27} a_{t} \cdot \left\{ \sum_{j=1}^{m(i)} \left((x_{i,j-1}^{t} - x_{i,j}^{t}) P_{ijmin} + \sum_{k=1}^{r(i,j)} P_{ij}^{tk} \right) \right\},$$

where $a=0,5$ or $1,0$ depending on the duration of period t.

4.1.3 Voltage variable. Denote s the number of the nodes of the network with adjustable voltage and v_i^1, v_i^2, v_i^3, $i=1,2,\ldots,s$ the voltage levels of these nodes in the three periods with extreme demands (voltage check periods).

4.2. Constraints of the model

4.2.1. Supply conditions. Denote P^t_{dem} the value of the power demand in period t. We require that the power demand be satisfied in each period, i.e.

$$\sum_{i=1}^{E} P_i^t = \sum_{i=1}^{E} \{ \sum_{j=1}^{m(i)} \left((x^t_{ij-1} - x^t_{ij}) P_{ijmin} + \sum_{k=1}^{r(i,j)} P^{tk}_{ij} \right) \} = P^t_{dem},$$

$t = 1, 2, \ldots, 27.$

4.2.2. Bounds on the power levels

$$0 \leq P^{tk}_{ij} \leq P^k_{ijmax} - P^k_{ijmin}, \qquad i=1,2,\ldots,E; \quad j=1,2,\ldots,m(i); \quad k=1,2,\ldots,r(i,j); \quad t=1,2,\ldots,27.$$

4.2.3 The variable coupling conditions require that the power level in period t of the jth mode of operation of power plant i should be between the bounds P_{ijmin} and P_{imjax}, i.e.

$$P_{ijmin} \cdot \left(x^t_{ij-1} - x^t_{ij} \right) \leq P^t_{ij} \leq P_{ijmax} \left(x^t_{ij-1} - x^t_{ij} \right).$$

Taking into account 4.1.2.4. we get the conditions:

$$\left(x^t_{ij-1} - x^t_{ij} \right) \cdot \left(P_{ijmax} - P_{ijmin} \right) - \sum_{k=1}^{r(i,j)} P^{tk}_{ij} \geq 0,$$

$i=1,2,\ldots,E; \quad j=1,2,\ldots,m(i); \quad t=1,2,\ldots,27.$

4.2.4. Start and stop conditions. These conditions ensure the implication $x^t_{ij}=1 \rightarrow x^{t+1}_{ij}=1$ in the shut off periods and the implication $x^t_{ij}=0 \rightarrow x^{t+1}_{ij}=0$ in the start periods.

Denote t_1 the last period preceding the examined day, $x^{t_1}_{ij}$ the realized value of the mode of operation in the above period, t_2 the serial number of the beginning of the second

shut down phase, t_3 and t_4 the serial numbers of the beginning of the first and second starting phase resp., ℓ_1, ℓ_2, ℓ_3, ℓ_4 the lengths of the corresponding phases (in periods) in the previous sequence.

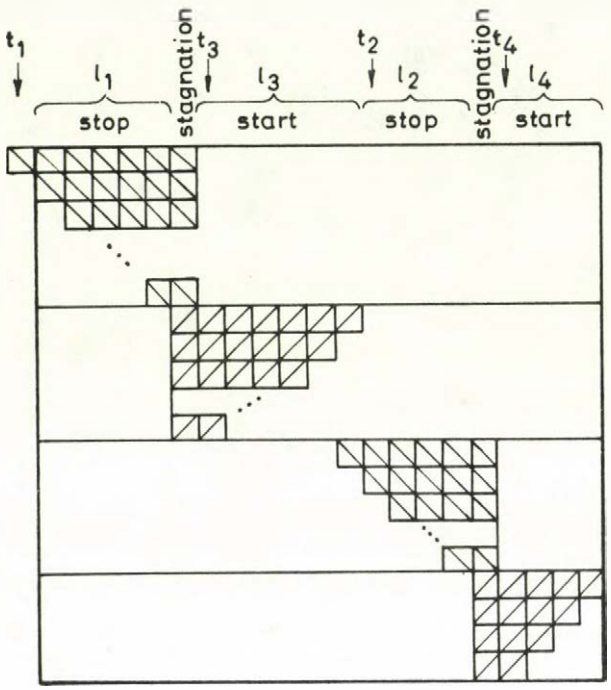

Fig.6.
Structure of the stop and start conditions

The shut off conditions are:

4.2.4.1. $$-(\ell_1+1)x_{ij}^{t_1} + \sum_{k=1}^{\ell_1+1} x_{ij}^{k} \geq 0, \qquad i = 1,2,\ldots,E; \quad j = 1,2,\ldots,m(i)-1.$$

4.2.4.2. $$\{-(\ell_1+1)+t\}.x_{ij}^{t} + \sum_{k=1+t}^{\ell_1+1} x_{ij}^{k} \geq 0,$$

$$i=1,2,\ldots,E; \quad j=1,2,\ldots,m(i)-1; \quad t=1,2,\ldots,\ell_1.$$

4.2.4.3. $$(-\ell_2+t)x_{ij}^{t_2+t} + \sum_{k=t_2+t+1}^{t_2+\ell_2} x_{ij}^{k} \geq 0,$$

$$i=1,2,\ldots,E;\ j=1,2,\ldots,m(i)-1;\ t=-1,0,1,\ldots,\ell_2-1.$$

The start conditions are the following:

4.2.4.4. $$x_{ij}^{t_3-4} + \sum_{k=t_3}^{t_3+t-1} x_{ij}^{k} - (t+1).x_{ij}^{t_3+t} \geq 0$$

$$i=1,2,\ldots,E;\ \ j=1,2,\ldots,m(i)-1;\ t=\ell_3-1,\ell_3-2,\ldots,1.$$

$$x_{ij}^{t_3-4} - x_{ij}^{t_3} \geq 0,\ i=1,2,\ldots E;\ \ j=1,2,\ldots,m(i)-1.$$

4.2.4.5. $$x_{ij}^{t_4-4} + \sum_{k=t_4}^{t_4+t-1} x_{ij}^{k} - (t+1)\, x_{ij}^{t_4+t} \geq 0,$$

$$i=1,2,\ldots,E;\ j=1,2,\ldots,m(i)-1;\ t=\ell_4-1,\ell_4-2,\ldots,1.$$

$$x_{ij}^{t_4-4} - x_{ij}^{t_4} \geq 0,\ \ i=1,2,\ldots,E.\ j=1,2,\ldots,m(i)-1.$$

Fig. 6. shows the structure of the matrix of these conditions.

4.2.5. Fuel constraints. These are constraints with lower and upper bounds, prescribed for the daily production of some power plants. Using 4.1.2.6. we can write them as follows:

$$E_{imin} \leq \sum_{t=1}^{27} a_t.\{\sum_{j=1}^{m(i)} \left(x_{ij-1}^{t} - x_{ij}^{t}\right) P_{ijmin} + \sum_{k=1}^{r(ij)} P_{ij}^{tk}\} \leq E_{imax}$$

where E_{imin}, E_{imax} are the given bounds, the i's are the subscripts of the power plants with fuel constraints.

4.2.6. Network conditions

According to the agreement in 3.2., the restrictions resulting from the electrical properties of the network will be taken into account in the three voltage check periods of the day. These conditions are the branch-load, the voltage and the reactive power source conditions. We describe only the content and form of these, the coefficients in the conditions depend on the network (which can be different during the three investigated periods) and a particular program system was designed for their determination.

The branch-load conditions ensure that the power transmission lines, cables and transformers forming the meshed system which transmits the power from the power plants to the consumers should not be over loaded. These conditions define the load caused by the effective power, viz. with the help of linear approximation of the exact quadratic expressions which yield a very good approximation in the solution domain characterizing the stable operation of the power systems. The form of the condition system is

$$4.2.6.1. \qquad -C_T \leq A.\begin{pmatrix} P \\ X \end{pmatrix} \leq C_T,$$

where A is the matrix of the coefficients. The number of its rows is equal to that of the branches, the number of its columns equals that of the sum of the power and mode of operation variables taken into account in the relevant period. C_T contains the loadability of the lines.

The number of these constraints is very large. We may, however, delete many of them and keep only a few that correspond to critical branches.

The voltage conditions ensure the voltage staying within prescribed limits at the nodes of the network. These involve also quadratic formulas where again linear approximation is used resulting in a properly accurate solution in the domain of operation.

The form of these conditions is:

4.2.6.2. $$V_{min} \leq B.V \leq V_{max}$$

where B is the matrix of the derived coefficients having as many rows as the number of the nodes of the network, while the number of its columns equals that of the voltage variables. B contains a unit matrix, V_{min} and V_{max} are the allowed minimal and maximal voltage thresholds of the nodes respectively. Actually the system of constraints contains all conditions corresponding to nodes with adjustable voltage, however for the remaining nodes it is sufficient to take into account only a few critical constraints.

Reactive source conditions ensure the reactive power of the reactive sources (performing the voltage control) not exceeding the allowed leading lagging power maxima, respectively. The reactive powers of the reactive sources are expressed by the voltages of the relevant nodes that we linearize around a given basepoint. This condition has the form

4.2.6.3. $$Q_{min} + \Delta Q_{min} . \underline{x} \leq C.V + Q_{const} \leq Q_{max} + \Delta Q_{max} . \underline{x}$$

where Q_{min}, Q_{max} limit the allowed leading and lagging power, respectively in s nodes, ΔQ_{min}, ΔQ_{max} contain the reactive power threshold changes resulting from the mode of operation change, C is the sxs matrix defining the change of the reactive supplies, Q_{const} is a constant vector with s elements, these elements being the reactive power supplies of the sources defined by the initial state of the vector.

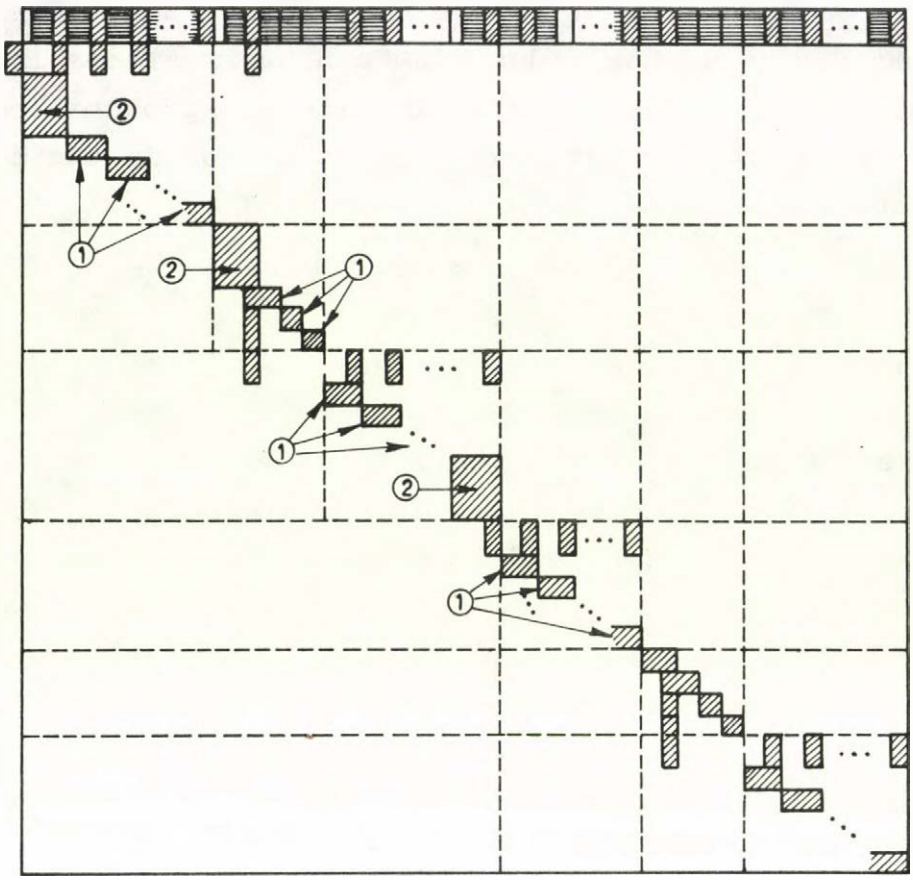

Fig.7.
Structure of the coefficient matrix of the whole model, where (1) and (2) have the structures given if Fig.8. and Fig.9.

4.3 Definition of the objective function. The objective function to be minimized consists of three parts:

$$K = K_1 + K_2 + K_3$$

where K_1 is the cost of power production, K_2 the cost of still-stand and K_3 the cost entailed by the network loss.

4.3.1. Definition of K_1. Denote C^k_{ij} the slope of the kth linear section of the function approximating the one-hour production cost curve of the jth mode of operation of power plant i, and C^o_{ij} the production cost of the level P_{ijmin}.

With these notations the cost of production on the level P^t_{ij} amounts to

4.3.1.1. $$K_{ij}(P^t_{ij}) = C^o_{ij} + \sum_{k=1}^{r(i,j)} C^k_{ij} \, P^{tk}_{ij}$$

if in the i-th power plant just the j-th mode of operation works. Thus

4.3.1.2. $$K_1 = \sum_{t=1}^{27} a_t \cdot \sum_{i=1}^{E} \sum_{j=1}^{m(i)} C^o_{ij} \cdot (x^t_{ij-1} - x^t_{ij}) + \sum_{k=1}^{r(i,j)} C^k_{ij} \cdot P^{tk}_{ij}$$

Note that $C^1_{ij} < C^2_{ij} \ldots < C^{r(ij)}_{ij}$ always holds, from which the fulfilment of the requirement 4.1.2.3. follows for such a solution which satisfies the coupling condition 4.2.3. and for which K_1 is minimal.

4.3.2. Definition of K_2. Fig.5. shows the cost function of the still-stand (or restarting) of the j-th mode of operation of power plant i as the function of the duration of the still stand. The function can be described by the formula

4.3.2.1. $$g_{ij}(\tau) = g_{ij}(0) + (g_{ij}(\infty) - g_{ij}(0)) \cdot (1 - e^{-C_{ij}\tau}),$$

where $g_{ij}(0)$, $g_{ij}(\infty)$ and C_{ij} are the constants characterizing the power plant and the mode of operation, $g_{ij}(0)$ denotes the

cost of starting without still-stand, and $g_{ij}(\infty)$ the cost of the so-called cold starting.

In accordance with the assumption 3.3.b, if a mode of operation is stopped with ℓ periods before the beginning of the stagnation phase, then its effect in the cost function will be taken into account with the value $g(4+2\ell)$. The corresponding value will be constructed with the help of properly chosen coefficients as a sum consisting of terms corresponding to the duration of the still-stand, - and the complete still--stand cost will take the form

$$4.3.2.2. \qquad K_2 = \sum_{t=1}^{27} \sum_{i=1}^{E} \sum_{j=1}^{m(i)-1} d_{ij}^{t} . x_{ij}^{t}$$

where d_{ij}^{t} is the properly chosen coefficient defined by the utilization of the function $g(\tau)$.

4.3.3. Definition of K_3

$$4.3.3.1. \qquad K_3 = \sum_{t} K_3^t$$

where t runs through the indices of the three voltage check periods.

The determination of the components of K_3^t - i.e. of the coefficients participating in its definition, - is a part of the procedure serving for the determination of the network conditions. We disregard its description, and give only the formulas:

$$4.3.3.2. \qquad K_3^t = \sum_{i=1}^{E} \sum_{j=1}^{m(i)} \left(a_{ij}^{t} \, x_{ij}^{t} + \sum_{k=1}^{r(i,j)} b_{ij}^{t} \, P_{ij}^{tk} \right) +$$

$$+ \sum_{\ell=1}^{s} h_{\ell}^{t} \, v_{\ell}^{t} + C_1^t + C_2^t .$$

5. A SURVEY OF THE MODEL STRUCTURE

Fig.7. is the schematical representation of the above described model. In its survey we point out that the conditions of the model have the following properties:

1. The fuel constraints contain besides the voltage variables all variables belonging to the given power plants and so practically they connect the variables of all the 27 periods.

2. The start-stop conditions contain the mode of operation variables of the corresponding phase, - these conditions connect the periods belonging to the given phases.

3. The connection among the particular phases is realized by the mode of operation variables belonging to the stagnation phase, these at the same time connect the periods belonging to the stagnation phase.

4. Further conditions of the model contain variables belonging to single periods only, the structures of these conditions are shown in Figs.8. and 9., respectively, - depending on the corresponding period being one without network conditions.

The size of the model - in choosing everywhere $r(ij)=1$ for the approximation of the cost function and taking the real size of the power system into account - is at most as follows:

the number of variables: 35 power variables for each period, 21 mode of operation variables and in the voltage check periods maximum 30 voltage variables, i.e. the number of continuous variables is 945 + 90 and the number of 0-1 variables is 441. The number of constraints amounts to about 1700, from these 420 conditions are start-stop conditions containing only 0-1 variables.

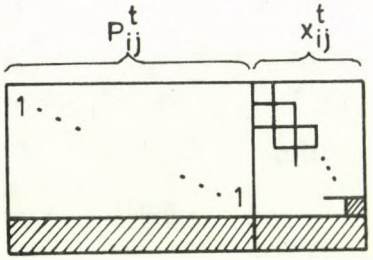

Fig.8.

Structure of the conditions in a "normal" period

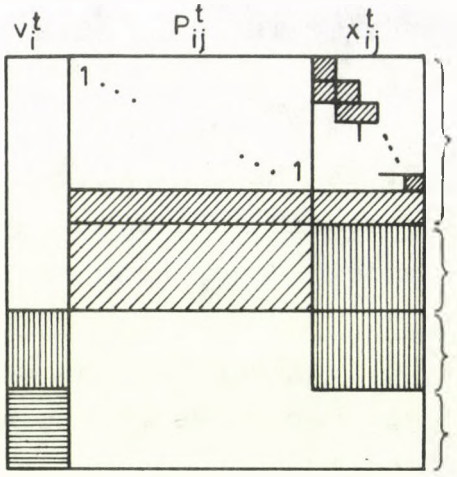

Fig.9.

Structure of the conditions in a special period

6. HOW TO SOLVE THE MODEL?

In order to complete our work we had to write a computer program for the CDC 3300 computer of the Hungarian Academy of Sciences for the solution of the problem. From among the possible ways we had the idea to apply the Benders decomposition method to solve the whole problem. This was rejected because, on one hand, it can happen that we will obtain a feasible solution only in the last step, so that if on account of computer time limitation the run had to be interrupted, the results till then would not contain the necessary information. On the other hand, there is a large number of variables of the pure 0-1 problems to be solved in the iterations of the decomposition and their constraints do not have favourable special structure. We thought of a version of the branch-and-bound algorithm in which the relevant linear programming problem could have been solved by the Dantzig-Wolfe decomposition, but because of the large number of the 0-1 variables we have rejected this idea, too.

Finally we have accepted the following algorithm:

1) We disregard the fuel constraints.

2) We solve the remaining large-scale mixed integer programming problem - in which the connections among the periods are ensured by the start-stop conditions and the mode of operation variables of the stagnation phases - the following way (Fig. 10.):

We solve successively the three mixed integer programming problems corresponding to the voltage check periods. We allow in the solution of the first problem every mode of operation applicable on the given day. In the solution of the second problem we allow only that modes of operations which are realizable from the modes of operations in the solution of the first problem by shut off. For the third problem we allow that modes of operations, realizable from the solution of the second problem by starting.

Thereafter we solve the intermediate problems and the problems corresponding to the following periods successively, by taking always the variables of the modes of operation of the neighbouring, already solved problems and the connections of the periods to the start-stop phases into account.

In every case the Benders decomposition method will be applied for the solution of the problem corresponding to one period.

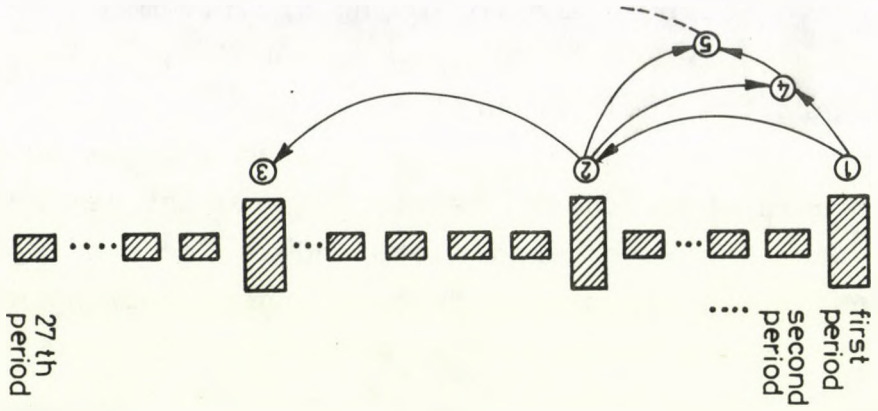

Fig.10.

The successive mode of solving the mixed-integer programming problem without fuel-conditions, where the numbers in circle indicate the order of the executions of computations

3) We check whether the fuel constraints are satisfied for the obtained solution. If yes, then the algorithm ends, else the following iterative procedure will be applied.

4) If in a power plant the daily power production is less than what is prescribed then the production cost coefficients of the given power plant will be multiplied by a multiplier less than 1, and if the daily power production is greater than what is prescribed, then they will be multiplied by a multiplier greater than 1. The values of the mode of operation variables will be fixed and the corresponding linear programming problem will be solved. If in the course of the solution the fuel constraints are satisfied by the new outputs obtained, the iterations ends.

Otherwise there are two cases: i) if in the course of the iteration processes we have already found solutions indicating underproduction and overproduction, too, then we will proceed according to paragraph 5; ii) else we will modify again the cost coefficients and repeat the solution of the linear programming problem.

5) The mode of operation values of the solution accepted as optimum are the fixed modes of operation and the production level will be defined by such a linear combination of any particular solution indicating the underproduction and overproduction which satisfies the fuel constraint.

Remark: The physical background and the preliminary survey of the data ensures that the described algorithm works well, i.e. it cannot occur that a mixed problem corresponding to a period has no feasible solution or that we obtain only such solutions in the 4-th step which violate this constraint only in the same direction.

7. CONCLUDING REMARKS

This paper gives only a short survey of the most important features of the model, without any claim to completeness. A brief sketch of the whole computer program system is shown on Fig. 11, and a study covering also details not discussed in this paper (e.g. computation of loss, determination of the network conditions etc.) is under preparation.

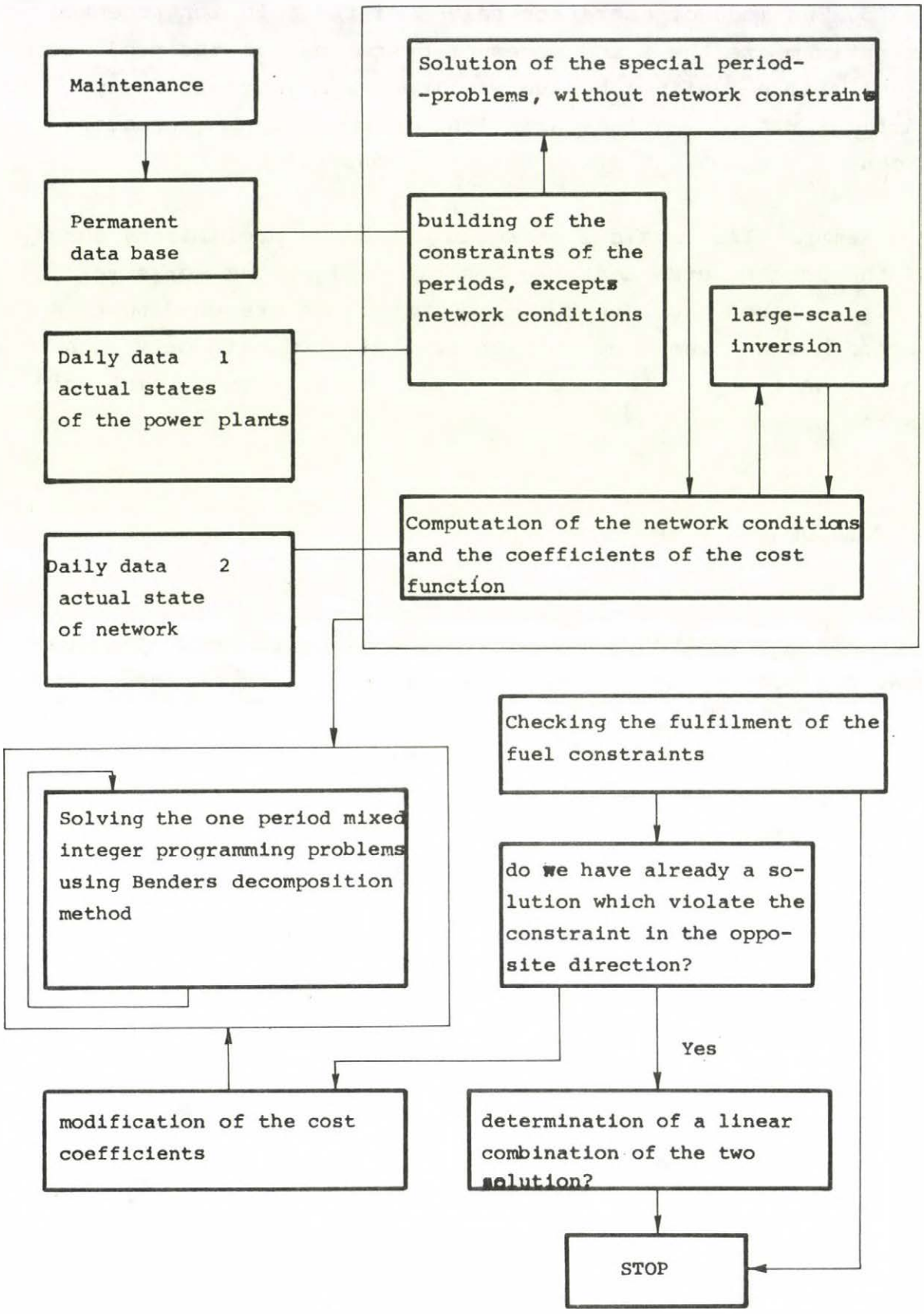
Maintenance
Permanent data base
Daily data 1 actual states of the power plants
Daily data 2 actual state of network
Solution of the special period--problems, without network constraints
building of the constraints of the periods, excepts network conditions
large-scale inversion
Computation of the network conditions and the coefficients of the cost function
Solving the one period mixed integer programming problems using Benders decomposition method
Checking the fulfilment of the fuel constraints
do we have already a solution which violate the constraint in the opposite direction?
Yes
modification of the cost coefficients
determination of a linear combination of the two solution?
STOP

Fig. 11.

REFERENCES

[1] I.Hano, Y.Tamura, S.Narita, K.Matsumote: Real time Control of System voltage and reactive power IEEE Trans PAS-88, No-10, 1969, pp. 1544-1559.

[2] C.M.Shen, M.A.Langhton: Power System load scheduling with security constraints using dual linear programming, Proc. IEE. Vol. 117, No.-11, November 1970, pp.2117-2127.

[3] A.de Perna, E.Mariani: Programmazione giornaliera delle centrali idroelettriche a bacine e a serbatoio in un sistema di produzione misto. L'Energia Elettrica-No.7, 1971, pp. 437-448.

[4] B. Stott: Review of Load Flow Calculation Methods. Proc. of IEEE Vol. 62, No.-7, 1974. pp. 916-929.

[5] O.I. Elgerd: Electrical Energy Systems Theory: An Introduction. McGraw Hill Book. Co., 1971.

[6] Benders, J.F.: Partitioning procedures for solving mixed variable programming problems. Numerische Mathematik, 1/238-252, 1962.

RELIABILITY TYPE INVENTORY CONTROL PROGRAM PACKAGE

L.Gömböcz, P.Kelle, A.Sebő

(Budapest, Hungary)

1. GENERAL CONCEPT

A multi-purpose inventory control program package has been developed with a modular program structure which has three main tasks

- demand forecasting,
- safety stock planning and
- order recommendation.

The main characteristic of our program system is that - considering the various demand and supply conditions occurring in practice - many different models are built in together with a system of automatic model choice based on the analysis of past periods. Beside the models well-known in literature new inventory models have been constructed for the program system which became especially suitable for socialist enterprises in production, commerce and supply.

The forecasting is based on extended versions of the exponential smoothing method for trend and seasonal influence combined with statistical tests and a simulation method for choosing the best model and its parameter values.

The safety stock is planned by using reliability-type inventory models because of the difficulties in evaluating the cost factors. Here the minimal level of safety stock is determined which ensures the continuous supply on a prescribed

probability (service) level. At the time of decision making both demand and delivery are connected with a lot of random factors. There is a typical case in which the delivery of an order occurs not an one occasion but at random moments of a period maybe in random parts. The new models which handle the above case under different random fluctuations of the delivery process are described later.

The enterprise considered has a continuous production and a periodic review inventory system with a fixed length of a period which may be different for the different items (one month, a quarter of a year etc.). The fix cost of ordering is relatively low and the order period is long so that at every review point an order is usually given.

The order recommendation is given on the basis of the material requirement plan or forecasted demand and on the basis of the safety stock plan made for a period ahead.

The program system contains many program modules to complete the above functions under the different conditions of supply and consumption for the different items. The modules are formally similar blocks - communicating through a common data structure - which can be changed among each other. The program system can be easily extended and adapted for many different enterprises in consequence of its structure and the many different models built in.

The program system was developed in PL/I for IBM 3031 under CMS and adapted for R40 under the DOS system.

2. INPUT DATA, ASSUMPTIONS ON SUPPLY AND DEMAND VARIATIONS

The inventory control program system is based on a stock management system. It consists of four files stored on tapes, containing the following data (among many others)

i) item master file with ABC grouping, length of the order period, order restrictions etc.,

ii) stock actions file with delivery and consumption dates and amounts,

iii) order file with purchase order dates and amounts,

iv) demand file with actual demands, material requirement plan or forecasted demand and forecast error.

The instants and amounts of deliveries and consumptions registeredand stored are the basis for choosing and fitting an appropriate model for the delivery and demand process of each item. The structure of both processes are similar: in the order period which will be denoted by $[0,T]$ at certain instants certain amounts appear in the store as input or as desired output. At the time of decision making (ordering) both delivery and demand are often connected with a lot of random factors. The typical supply and demand variations will be listed together, which contain all the important practical cases.

a) the delivery of an order (or the demand in an order period) occurs at once at a known or at a random instant of $[0,T]$,

b) it doesn't occur at once but at fixed instants of the interval $[0,T]$ in fixed lot sizes,

c) the instants and/or the lot sizes of the deliveries (demand) are random, they may have a time-homogeneous character or an inhomogeneous character,

d) there is a continuous delivery (demand) with a known or with a random (at the time of decision making not sufficiently known) demand rate,

e) the rate of the delivery (demand) fluctuates around its mean value with a random character.

For all the above cases an exact or an approximate solution was given for the calculation of the necessary safety stock which ensures the required service level. Here we put forward only some models and solutions for the cases c) and e) in section 4.

3. STRUCTURE OF THE PROGRAM SYSTEM

The main input and output files, procedures and their connections are summarized in Figure 1.

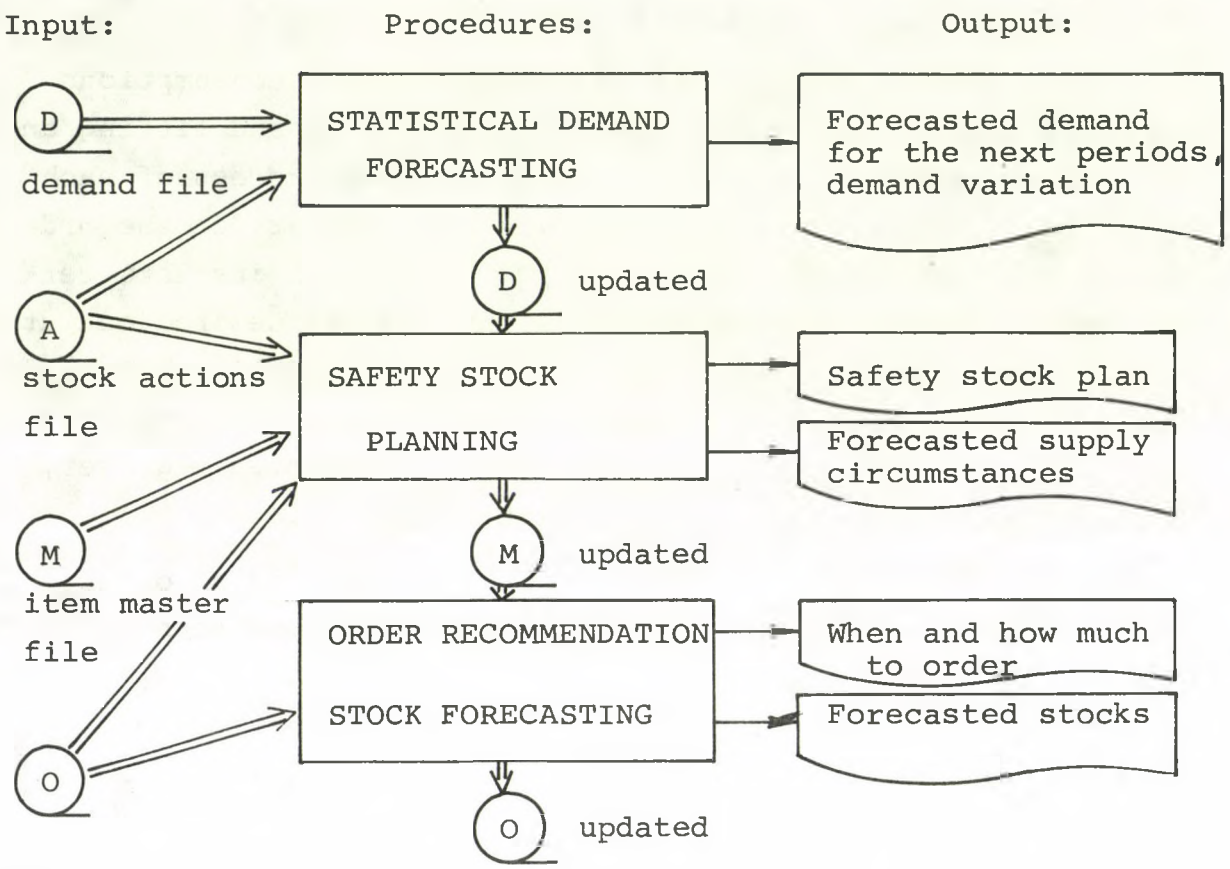

Figure 1.

Main blocks and their connections

The procedures (solution of the models, statistical preparations etc.) communicate with each other and with the main program through data structures defined by pointer variables. It has the following advantages

- the procedures at any level of the structure may communicate using minimal administration,

- new procedures with new parameters can fit the system with few efforts since the data structure can be completed and only the main program has to be translated again,

- parameter systems can be exchanged with the change of a single pointer,

- the procedures are formally equivalent, they are interchangeable and form a modular system.

The solution procedures of the inventory models and that of the statistical forecasting methods are isolated from the main program and from the data management system. Thus the local characteristics of the enterprise are separated from the models, the system can easily be adapted to other circumstances.

Each parameter has a standard value built in the system and has to be changed in exceptional cases only. The most important parameters of the forecasting method are initialized by each item using statistical and simulation methods.

There is a sequential run item by item since the data of the huge amount of items (30000 to 100000) are stored on tapes. The initialization of the forecasting parameters is done when the forecasting error exceeds the tolerance. The choice of the forecasting model (horisontal, seasonal, trend, trend-seasonal) and the choice of the safety stock planning model is automatic, based on the realization of the previous periods, however the user may have the outside decision, too.

The most important step for the preparation of the decision is when to order and how much. Since there is usually a periodic order possibility (probably with different length of period for different items) the choice refers to the amount ordered that may be zero, too. The procedure of the order recommendation is a simple calculation based on the results of demand forecasting and safety stock planning which have to be prepared a period ahead. In Figure 2 we outline this procedure.

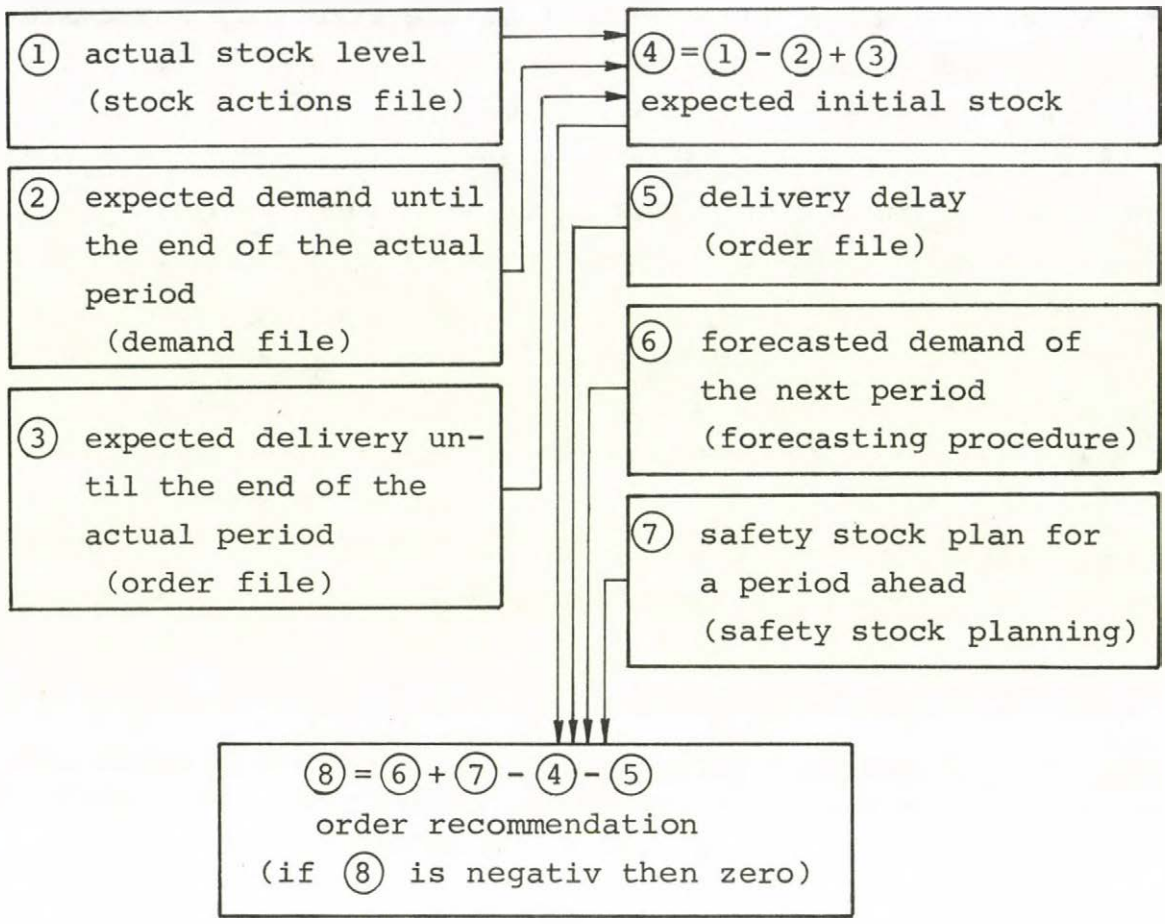

Figure 2

Procedure of the order recommendation

4. NEW MODELS AND SOLUTION METHODS

The enterprise plans an initial stock for each item which serves as safety stock for protection against the time delays and random disturbances of delivery and demand in the next order period. The reliability-type inventory models determine the minimal level of the safety stock M which ensures the continuous supply in the whole order period $[0,T]$ on a prescribed probability level $1-\varepsilon$. These models play an important role in the case of the random delivery process which is typical with many items.

For the case when the deliveries occur on random instants of the interval [0,T] in random lot sizes a general model was formulated in Kelle [2] for the time-homogeneous case. It means we assume that a delivery may occur at any moment of the time interval [0,T] with the same probability. Thus the subsequent deliveries happen on instants which are the elements of an ordered sample taken from the uniform distribution in [0,T]. For the amounts delivered at one occasion we allow any kind of distributions. It can be approximated with the help of statistical data of earlier observations available in practical situations. If the demand has a known rate or random rate with known distribution we can calculate the exact value of the necessary initial stock for a given service level. It means a fast iterative solution of an equation detailed in Kelle [2].

In many cases there is a minimal amount known in advance which arrives with certainty when a delivery occurs. The rest amount of delivery is at random subdivided among the lots delivered by a uniform distribution. This is the model of Prékopa [3] which is a special case of our above model. For the initial stock a very simple approximate formula can be given when the demand rate c is known (see Prékopa [3]):

$$M = cT\sqrt{\frac{1}{2n}\ln\frac{1}{\varepsilon}}$$

where n is the number of deliveries in the time period [0,T] and $1-\varepsilon$ is the probability of the continuous supply in [0,T] which is the prescribed service level. This formula yields a good approximation when n is large ($n>10$). Otherwise a numerical correction has been applied on the basis of the exact solution which can be achieved by specifying the results of Kelle [2].

The above formula has been extended to the case of a random demand rate with normal distribution and a standard deviation s

$$M = cT\sqrt{\frac{1}{2n(1-ns^2)}\ln\frac{1}{\varepsilon}}$$

described by Kelle [2].

When the random deliveries cannot be assumed to occur at each point of the interval [0,T] with the same probability (inhomogeneous case), more sophisticated models have to be constructed. Such model and its solution method using simulation technique was published by Prékopa - Kelle [4]. It is, however, time-consuming to use it as a standard routine in a program package.

For many items, especially for the basic materials delivery occurs almost every day so it can be considered as a continuous process which has an average rate r. The fluctuation around this intensity rate is often random at the time of decision making. The measure of uncertainty increases as the time passes. Assuming that the random influence as the sum of many effects is normally distributed, the delivery process can be approximated by a Wiener process $\xi(t)$. The mean rate r and the standard deviation s are the parameters which have to be fitted for the process on the basis of the statistical data of earlier observations. The whole amount delivered until time $t(0 \leq t \leq T)$ has the distribution $F(x) = \Phi\left(\frac{x-rt}{s \cdot \sqrt{t}}\right)$ where Φ denotes the standard normal distribution function.

Having a constant demand rate c the necessary safety stock is the minimal M which guarantees the continuous supply in [0,T] with a prescribed probability $1-\varepsilon$. This is the solution of the equation

$$P\left(\sup_{0 \leq t \leq T} \{ct - \xi(t)\} \leq M\right) = 1-\varepsilon \ .$$

The left-hand side can be expressed in the form

$$\Phi\left(\frac{M-(c-r)T}{s\sqrt{T}}\right) - e^{\frac{2M(c-r)}{s^2}} \Phi\left(\frac{-M-(c-r)T}{s\sqrt{T}}\right)$$

by using the theorem of Baxter-Donsker [1]. A fast iterative method has been given for the numerical solution. It has been

extended also to the case when the demand rate c is random with known distribution and to the case when demand and delivery have the same model based on the Wiener process with different parameter values.

5. FIELD OF APPLICATION

Due to the many kind of models built into the program package many different circumstances appearing in the practice of demand and supply can be controlled by the system. The flexible modular stucture ensures an easy adaptation to different stock management systems. The minimal amount of the necessary data is contained in the item master file and in the stock actions file. Both files are built up in the first step with any computerized stock management system. Our program package can be easily adapted to such an existing data system.

The program package can be applied to the inventory control of materials, spare parts, unfinished and finished goods. For different items different models are available, promoting the applicationswith enterprises in production, commerce and supply.

The program system works since half a year at the Danubian Iron Works in Dunaujváros, and proved that under a sufficient high service level a lower total inventory can be reached by the appropriate choice of the safety stocks.

The demand forecasting was prepared by using the data of the consumption for the previous three years that have been stored on tapes. The real demand data are not available since the data management of stockout situations is not satisfactory. Thus a heuristic correction system has to be applied. A considerable part of the items has a trend and seasonal fluctuations. Except the structural brakes and slow moving items, a satisfactory forecast could be given by using the simulation method to

select the best model and its parameters. For the most important items a material requirement plan is made outside of our system.

For the safety stock planning the supply conditions play an important role. On the basis of the delivery dates and amounts of the last two years the lead time or - at multiple deliveries of an order - the instants and amounts of the lots were estimated together with the reliability of the estimation.

An automatic procedure has been developed for choosing the best model of demand and delivery from those built into the program system and for fitting the parameters of the model. For the model choice an outside decision is also possible.

The experiences proved the necessity of the numerous different models for demand and delivery variations built into the system. There are many suppliers who guarantee the delivery until the end of the order period but cannot be obliged to deliver with a prescribed lead time. For the times and lots of delivery there is not enough deterministic-type information available at the time of ordering, this fact stresses the importance of the new models of the system.

REFERENCES

[1] Baxter, G. and Donsker, M.D., On the supremum functional for processes with stationary independent increments, Transactions of the American Mathematical Society. 1977. pp. 73-87.

[2] Kelle, P., Reliability type inventory models for random delivery process, in Chikán (ed.) Proceedings of the First International Symposium on Inventories. (Akadémia Kiadó - - Elsevier 1980) pp. 385-395.

[3] Prékopa, A., Reliability equation for an inventory problem and its asymptotic solutions, in Coll. on Application of Mathematics to Economics. (Akadémiai Kiadó 1965) pp.313-327.

[4] Prékopa, A. and Kelle, P., Reliability type inventory models based on stochastic programming, Mathematical Programming Study 9 (1978) pp. 43-58.

LOCATION OF DEPOTS FOR SUGAR-BEET DISTRIBUTION SYSTEM

E.Jasińska and E.Wojtych

(Warsaw, Poland)

1. PROBLEM STATEMENT

In the Polish climate conditions the extension of sugar-beet collection time over about 50 days i.e. about half of each sugar production season leads to the substantial increase of the losses of both sugar-beet volume and its sugar percentage. Thus the organization of sugar-beet distribution system affects directly the smoothness and efficiency of the whole sugar production process. For a certain enterprise the current system is based on the following assumptions.

The enterprise operating on the predetermined land area is the union of several sugar-mills. They are supplied with sugar-beet by any fixed set of individual farms either directly or via depots. The direct farms-sugar-mills deliveries are recommended in order to minimize the losses caused by reloading and long-term storing of the sugar-beet.

On the other hand the limited and impossible to increase storing capacities of the sugar-mills force the part of the total annual crop to be distributed via depots where it is stored usually up to the end of the collection time and then dispatched to the sugar-mills according to their demands. The relatively low unit costs for transportation of big quantities from depots to sugar-mills provide the economic motivation for shipping sugar-beet via depots.

In view of the forthcoming growth of sugar-beet volume distributed via depots the urgent modernization of their equipment is required. By some technological and economic considerations the lower as well as the upper limits on the

depot storing capacities are estimated. In consequence so far existing dense network of small depots should be substituted by the adequate number of fully automatized ones within given throughput limits.

The problem is to determine the number, locations and sizes of the depots to be selected from the given candidate set and to find the appropriate amounts of sugar-beet flows from farms to sugar-mills directly or via depots so as to minimize the total transportation and depot investment and operation costs.

The future growth of sugar-beet fertility will affect neither depot locations nor flow directions. The changes of depot size are admitted corresponding to the amount of sugar--beet to be stored and the need for addition of some new depots to the existing set may be considered.

The above described problem is of location-transportation type. The standard approach to the problems of this class is their mixed-integer formulation (refer to the book of Garfinkel and Nemhauser [7]). Then the optimum solution can be obtained by application of the algorithms being the modified versions of the branch and bound rule. Some of them are developed and verified by Akinc and Khumawala [1], Davies and Ray [4], Efroymson and Ray [5] and Sà [10].

The successful application of Benders decomposition principle is presented by Geoffrion and Graves in [8].

For the large scale complex problems the heuristic procedures leading to acceptable results must be developed as it was demonstrated by Feldman, Lehrer and Ray [6], Kuhn and Hamburger [9] and Wojtych [12].

In the paper the mixed-integer linear programming formulation is presented for the above depot location and sugar--beet transportation problem. The results of application of IBM/370 MPSX&MIP codes are discussed together with some suggestions on branching rule. As the substantial problem size reduction seemed to be necessary the following two approaches are considered:

- mixed-integer linear programming problem formulation for the aggregated sugar-beet suppliers
- looking for suboptimum solutions under some real-life assumptions

They are both evaluated on the basis of computational results from the point of view of the decision-maker requirements.

2. MIXED-INTEGER PROGRAMMING PROBLEM FORMULATION

In order to formulate the mixed-integer linear programming (MILP) problem the analysis of the depot investment and operation costs is necessary. The values of these costs are calculated for certain depot throughput values. The depot cost function is approximated by the convex piecewise linear cost function with one jump discontinuity (see Fig. 1).

The coefficients of both linear segments are estimated by application of the adequate version of the least squares procedure. The marginal cost for the throughput between $\underline{L}$ and L tons is smaller than that for the throughput exceeding L tons. This rather unexpected relation opposite to the economies of large scale rule results from the need to hire for the biggest depots the reloading machine time at the prices higher than the unit operating cost of their own tools. Nevertheless the small number of the depots above this uneconomic level is admitted by the sugar industry managers with hope that the transportation cost reduction resulting from their suitable location will balance this cost increase. The MILP model for typical case is slightly more complicated than the one investigated here.

The piecewise linear representation of the cost function enables treating any of the potential depots as the two separate depots: the basic one of the throughput between $\underline{L}$ and L tons and the additional one of the throughput to $(\bar{L}-L)$ tons i.e. equal to the excess over the basic depot upper throughput limit. Their cost functions are represented by the adequate

segments of the piecewise linear function. The additional depot can be opened if and only if the basic one at the same site reached its upper throughput limit. The binary decision variables associated with the linear segments provide the information whether the adequate depot is to be opened or otherwise.

The following notation is introduced:

The parameters:

i, k, j – indices to farms, potential depots and sugar-mills accordingly ($i \in J$, $k \in K$, $j \in J$)

a_i – the total supply of the i-th farm

g_j – the total collection capacity of the j-th sugar-mill for the whole sugar production season, i.e. the sum of the storing capacity and the total production capacity during the collection time

b_j – the total production capacity of the j-th sugar-mill for the whole sugar production season

h_{ik}, p_{ij}, l_{kj} – unit transportation costs between the points marked by indices

c_k^1, c_k^2 – unit operation costs of basic and additional depots respectively at k-th location ($c_k^1 < c_k^2$)

$$e_{kj} = l_{kj} + c_k^1, \qquad f_{kj} = l_{kj} + c_k^2$$

d_k^1, d_k^2 – fixed charges for basic and additional depots at k-th location which are incurred if the adequate depot works at any level

The decision variables:

x_{ik}, y_{ij} - the amounts of sugar-beet to be distributed between the points marked by indices

z_{kj}, s_{kj} - the amounts of sugar-beet to be delivered from the basic or additional depot at k-th location to j-th sugar-mill

$$v_k^1 = \begin{cases} 1 \text{ if } \underline{L} \leq \sum_j z_{kj} \leq L \\ 0 \text{ otherwise} \end{cases}$$

i.e. the basic depot is opened at k-th location $(k \in K)$

$$v_k^2 = \begin{cases} 1 \text{ if } 0 < \sum_j s_{kj} \leq \bar{L} - L \\ 0 \text{ otherwise} \end{cases}$$

i.e. the additional depot is opened at k-th location $(k \in K)$

The MILP model takes the form:

Minimize

$$\sum_i \sum_k h_{ik} x_{ik} + \sum_i \sum_j p_{ij} y_{ij} + \sum_k \sum_j e_{kj} z_{kj} + \sum_k \sum_j f_{kj} s_{kj} + \sum_k (d_k^1 v_k^1 + d_k^2 v_k^2) \quad (1)$$

subject to:

$$\sum_k x_{ik} + \sum_j y_{ij} = a_i \quad (i \in J) \quad (2)$$

$$\sum_i y_{ij} \leq g_j \quad (j \in J) \quad (3)$$

$$\sum_i y_{ij} + \sum_k z_{kj} + \sum_k s_{kj} \leq b_j \quad (j \in J) \quad (4)$$

$$\sum_i x_{ik} - \sum_j z_{kj} - \sum_j s_{kj} = 0 \quad (k \in K) \quad (5)$$

$$\sum_j z_{kj} - \underline{L}\, v_k^1 \geq 0 \qquad (6)$$

$$\sum_j z_{kj} - L\, v_k^1 \leq 0 \qquad (7)$$

$$\sum_j s_{kj} - (\bar{L}-L)\, v_k^2 \leq 0 \qquad (8)$$

$(k \in K)$

$$v_k^1 - v_k^2 \geq 0 \qquad (9)$$

$$v_k^1,\ v_k^2 \in \langle 0,1 \rangle \qquad (10)$$

$$v_k^1,\ v_k^2 \ \textit{integer} \qquad (11)$$

$k \in K$

$$\textit{all variables nonnegative} \qquad (12)$$

The objective function (1) represents the sum of the total transportation costs and the total depot investment and operation costs.

The constraints (2) ensure that the whole crop is taken from farms. The constraints (3) state that the amount of direct deliveries cannot exceed sugar-mill collection capacities. Respectively, constraints (4) keep the total deliveries to the sugar-mills below their technological production capacities. The constraints (5) correspond to the assumption that there are no losses of sugar-beet volume at the depots. The constraints (6)-(8) prevent the depot throughput limits to be violated simultaneously enforcing the correct logical relationship between linear and integer variables.

Namely both the binary variables take the values zero if and only if the adequate linear variables are all equal zero. This means that the zero shipments correspond to the depots with zero throughput which in fact are not to be opened. For the values one of the binary variables the constraints (6)-(8) became ordinary depot throughput limit constraints. In case when only the basic depot at site k $(k \in K)$ is to be opened

$v_k^1=1$ and $v_k^2=0$ if $\sum_j z_{kj} > 0$ and $\sum_j s_{kj} = 0$. In fact for $v_k^1=1$ the constraints (6)-(7) are the basic depot throughput limit constraints. For $v_k^2=0$ the corresponding constraint (8) takes the form $\sum_j s_{kj} \leq 0$ what together with nonnegativity constraints involves all the linear variables s_{kj} equal zero. In turn for any positive z_{kj} constraints (6)-(8) are satisfied only for $v_k^1=1$ but $\sum_j s_{kj}=0$ doesn't affect the value of v_k^2. However in the optimum solution $v_k^2=0$ minimizing the objective function value will be chosen.

3. APPLICATION OF THE IBM MPSX&MIP/370 SYSTEMS

For the selected enterprise the problem with 1588 farms, 12 sugar-mills, 49 possible depot locations was described as the MILP problem with 9047 continuous variables, 98 binary variables and 1809 constraints.

The implementation of MPSX&MIP/370 systems for solving MILP problem at hand was considered. The sizes of problems successfully treatable by these systems quoted in IBM publications strongly exceed the above mentioned ones (see Benichou and the others [3]). The additional features such as the coefficient matrix density (0.15) and the ratio of integer variables (0.01) were also promising. The calculations were performed on an IBM 370/145 computer. The specialized FORTRAN IV/G program for data set generation enabled input card deck reduction. Another program in FORTRAN controled the output layout and size.

The linear optimum solution was found in 120 minutes of the CPU time with optimum objective function value 144141024 and 39 binary variables taking noninteger values.

They are then sorted into decreasing sequence and indicate the priority order for branching in subsequent runs. The priority was also given to the depots with the biggest throughputs in the linear optimum solution. The depot layout given by

the decision-makers was also tested as the indication for branching strategy. The efforts to find any feasible integer solution using standard as well as user branching strategies have failed within reasonable calculation time.

These results indicate the need for the problem size reduction and/or looking for the algorithms leading to the acceptable approximate solutions after relatively short computer calculation time.

4. TWO SUBOPTIMAL APPROACHES

4.1 Problem size reduction by aggregation

What causes the enormous size of the problem (1)-(12) is the substantial number of farms. The constraints (2) contain separate conditions for each farm. Thus the aggregation of farms into supply zones leads to the model of the same form but with far less constraints and continuous variables.

It seemed reasonable to aggregate farms into bigger supply zones according to the following heuristic criteria:

- location in the neighbourhood of the same depots or sugar-mills
- location along the same routes
- short distances between farms in the same zone
- similar area and the shape of the zones.

As a result 1588 farms were aggregated into 128 zones. The central point representing each zone in the model was chosen. The distances between the central point and the depots and sugar-mills were calculated as the mean values of the distances from all farms in the zone. The supply of the central point was taken equal to the sum of sugar-beet amounts of the represented farms.

The aggregated MILP problem had 398 constraints and 1135 continuous and 98 binary variables. The application of MPSX&MIP systems led to four equivalent feasible integer solutions in 128 minutes (including 25 minutes of CPU time to complete the

linear phase of calculations) of the total IBM 370/145 computer time. As the solutions obtained were no more than 0,6% far from the estimated optimum and the objective function value reduces slowly the decision to terminate the calculations seemed to be justified. The objective function values of the integer solutions varied from each other of about 0,06%. Therefore these 4 solutions could be treated as the alternative configurations of the depot locations.

The return to the model (1)-(12) was necessary to find the real values of sugar-beet deliveries from individual farms. Fixing the values of binary variables according to the integer feasible solutions of the aggregated problem gave LP problem of about 3240 variables and 1644 constraints for each of the four possible depot configurations. They were solved subsequently applying MPSX/370 system each in about 23 minutes of the total computer time.

4.2 Two-stage heuristic procedure

Two-stage heuristic procedure is developed based on the following reasonable assumptions:

- the direct farms-sugar-mills deliveries are established up to the upper limit of the sugar-mill collection capacities
- the depots are to be located at the sugar-beet supply centers.

The calculations are performed according to the following scheme.

At the initial stage the direct flow pattern is established iteratively.

The n-th iteration result in finding

$$y^{n}_{ij(i)} = \begin{cases} a_i \quad if \quad \min\limits_{k,j} (h_{ik}+l_{kj}) \geq t^{n}_{j(i)} \quad \min\limits_{j} p_{ij} = \\ \qquad\qquad\qquad\qquad = t^{n}_{j(i)} p_{ij(i)} \\ 0 \quad otherwise \end{cases} \qquad (i \in I)$$

where t_j^n is the current value of so called excess ratio. Note that putting $t_j^1=1$ we start the iterative procedure with fixing the least cost flow pattern for direct deliveries where they are cheaper comparatively to adequate costs of shipments via depots.

The delivery scheme which satisfies

$$\sum_{i\in I} y_{ij}^n \leq g_j \qquad (j\in J)$$

is optimal.

Otherwise the value of excess ratio is increased by any positive value of step Δ_j^n and the iterative procedure continues. This permits after N iterations to decrease the total amount of direct flows to particular sugar-mills up to the level below collection capacities with the least possible cost increase.

The farms which supply is not yet disposed ($i\in\bar{I}$ where $\bar{I}=\{i: y_{ij}^N = 0 \quad j\in J\}$) and the sugar-mills with some spare production capacity (i.e. $j\in\bar{J}$ where $\bar{J}=\{j: b_j - \sum_{i\in I-\bar{I}} y_{ij}^N > 0\}$) are left for further consideration.

At the advanced stage the combination of Baumol and Wolfe approach to the warehouse location problem presented in [2] and Feldman, Lehrer and Ray drop routine discussed in [6] seemed to be promising.

The initial number of depots in the candidate set is taken equal to the number of medium size depots necessary to collect the total sugar-beet supply not yet disposed.

The initial candidate set of depots is established by selecting those with the biggest throughputs resulted from the assignment of each farm to any sugar-mill via the cheapest depot.

The depot layout is improved and the adequate delivery scheme is found by solving the series of the classical transportation problem (TP) farms-sugar-mills where farm supplies are equal a $a_i (i\in\bar{I})$ and sugar-mill demands take the values

$\beta_j = b_j - \sum_{i \in I - \bar{I}} y_{ij}^N \quad (j \in \bar{J})$. The elements of the cost matrix for m-th iteration are calculated as follows:

$$r^m_{ik(i)j} = \min_{k \in K^m} (h_{ik} + e_{kj}) + \delta^m_k \quad (i \in I,\ j \in J)$$

where

K^m - is the current depot set

δ^m_k - is the unit investment cost of the k-th depot depending on the throughput

Denoting by $\bar{u}^m_{ik(i)j}$ the optimum delivery scheme obtained in m-th iteration we get the depot throughputs according to the formula

$$w_k^m = \sum_{(i,j) \in I_k^m} \bar{u}^m_{ik(i)j} \qquad (k \in K)$$

where

$$J_k^m = \{(i,j):\ \bar{u}_{ik(i)j} > 0\}$$

The depot of the lowest throughput below lower limit is removed from the candidate set.

The procedure terminates when the set of depots within given throughput limits is obtained.

The convergence of the procedure was proved by the series of computation runs. At the initial stage the direct deliveries from about 1200 farms are found. Several sugar-mills are also eliminated from further considerations as their production capacities don't exceed their collection capacities. Thus the TP so solve at the advanced stage has about 400 supply points and no more than 12 demand points. The choice of the initial depot number and candidate set affects strongly the number of necessary iterations. The drop as well as add routine for depot layout improvement were verified. The depot throughputs never reached their upper limits.

The solution of no more than 10% from the optimum is obtained after 15 minutes of the IBM 370/145 computer CPU time what meets sugar enterprise requirements.

Both of the presented approaches are accepted by the decision-makers as the useful and efficient tools in planning the sugar-beet distribution system. It is up to the enterprise to select the one to be applied of the suitable accuracy and computation costs.

REFERENCES

[1] U.Akinc and B.M. Khumawala, An efficient branch and bound algorithm for the capacitated warehouse location problem, Management Sci.6 (1977) 584-594.

[2] W.L. Baumol and P. Wolfe, A warehouse location problem, Operations Res. (1958) 252-263

[3] M.Benichou, J.M.Gauthier, G.Hentges and G. Ribiere, The efficient solution of large scale linear programming problems, Math. Programming (1977) 280-322

[4] P.S. Davies and T.L.Ray, A branch and bound algorithm for the capacitated facilities location problem, Naval Res. Log. Quart. (1969) 331-344

[5] M.A. Efroymson and T.L. Ray, A branch and bound algorithm for plant locations, Operations Res. 3 (1966) 361-368

[6] E.Feldman, F.A. Lehrer and T.L.Ray, Warehouse location under continuous economies of scale, Management Sci. 9 (1966) 670-684.

[7] R.S.Garfinkel and G.L. Nemhauser, Integer Programming (New York, 1972)

[8] M.A. Geoffrion and G.W.Graves, Multicommodity distribution system design by Benders decomposition, Management Sci. 5 (1974) 822-844

[9] A.M.Kuhn and M.J.Hamburger, A heuristic program for locating warehouses, Management Sci. 9 (1963) 643-666

[10] G.Sà, Branch and bound and approximate solutions to capacitated plant location problem, Operations Res. 6 (1969) 1005-1016

[11] H.M.Wagner, Principles of operations research, (New Jersey, 1969)

[12] E.Wojtych, Heuristic methods for certain location problems, Proc. of the Polish-Danish MP Seminar-part I. Zaborów (1978)

A NEW ALGORITHM FOR LINEARLY CONSTRAINED DISCRETE NONLINEAR MINIMAX APPROXIMATION

L.Lukšan

(Praha, Czechoslovakia)

1. INTRODUCTION

We are concerned with the problem *(P)* of minimizing a non-differentiable function $F(x)$ of the form

$$F(x) = \max_{1 \leq i \leq m} f_i(x)$$

on the convex polytope

$$C = \{x \in R_n : a_j^T x \geq b_j, \quad 1 \leq j \leq k\}$$

where $f_i(x)$, $1 \leq i \leq m$ are real-valued functions defined on the n-dimensional vector space R_n and have continuous second order derivatives. Let

$$I(x) = \{i : f_i(x) = F(x)\}$$
$$J(x) = \{j : a_j^T(x) = b_j\}$$

be sets of indices of active functions and active constraints respectively and suppose that the vectors $[g_i^T(x), 1]^T$, $i \in I(x)$, $[a_j^T, 0]^T$, $j \in J(x)$ are linearly independent at the point $x \in R_n$. Then necessary conditions for the solution of problem *(P)* have the form

$$\left.\begin{aligned} &\sum_{i\in I(x)} u_i g_i(x) = \sum_{j\in J(x)} v_j a_j \\ &\sum_{i\in I(x)} u_i = 1 \\ &u_i \geq 0,\ i\in I(x) \\ &v_j \geq 0,\ j\in J(x) \end{aligned}\right\} \quad (N)$$

where u_i, $i\in I(x)$ and v_j, $j\in J(x)$ are nonnegative Lagrange multipliers and $g_i(x)$, $i\in I(x)$ are gradients of the functions $f_i(x)$, $i\in I(x)$.

If $C=R_n$ (unconstrained case) conditions (N) can be written in the form

$$Nr(g_i(x),\ i\in I(x)) = 0$$

where $Nr(g_i(x),\ i\in I(x))$ is a minimum Length vector from the convex hull of gradients $g_i(x)$, $i\in I(x)$. When conditions (N) are not satisfied the nonzero vector $s=-Nr(g_i(x),\ i\in I(x))$ exists. Demyanov and Malozemov [1] have shown that the vector s is the steepest descent direction for the function $F(x)$ and they have proposed an iterative method with iterations

$$x^+ = x + \alpha s \qquad (1)$$

where α is a steplength, which is taken so that $F(x^+)<F(x)$ (we use the notation $x^+=x+\alpha s$ instead of the standard notation $x_{k+1}=x_k+\alpha_k s_k$, $k=1,2,\ldots$).

We are going to propose a new class of iterative methods for the solution of problem (P) with iterations (1), where

$$\left.\begin{aligned} s &= S\,\tilde{s} \\ \tilde{s} &= -\tilde{g} \\ \tilde{g} &= Nr(\tilde{g}_i(x),\ i\in I(x)) \\ \tilde{g}_i(x) &= S^T g_i(x),\ i\in I(x) \end{aligned}\right\} \qquad (2)$$

and where the matrix S has a full rank. The matrix S will be taken so that its columns define a basis in the orthogonal complement of the subspace spanned by normals a_j, $j\in J(x)$, and it will be updated by the product form of the variable metric methods. Note that for $S=I$ (I is the unit matrix of the order n) we obtain the original method of Demyanov and Melozemov.

2. THE VARIABLE METRIC ALGORITHM

The algorithm described in this section is based on a feasible direction method with active set strategy. It works with three matrices A, S and R. The matrix A has columns a_j, $j\in J(x)$ which are linearly independent normals of the active constraints. The matrix S has columns which define a basis in the orthogonal complement of the subspace spanned by columns of the matrix A i.e. $[A, S]$ is a nonsingular square matrix of the order n and $A^TS=0$ and it is updated by the product form of the variable metric methods. The matrix R is upper triangular and $R^TR = A^TA$ holds.

STEP 1: Determine an initial feasible point $x\in C$. (It can be determined as a solution of linear programming problem as in [2]). Determine the set $J(x)$ of the indices of active constraints and matrices A, S and R defined above. Compute values $f_i(x)$, $1\leq i\leq m$ and gradients $g_i(x)$, $1\leq i\leq m$ of functions in the problem (P). Compute the value of the objective function $F(x)$ Determine set $I(x)$ of the indices of active functions.

STEP 2: Compute the feasible direction s by (2). (Reduced gradient $\tilde{g}=Nr(\tilde{g}_i(x),\ i\in I(x))$ can be determined by algorithm

described in [10]. Note that $\tilde{g}=\sum_{i\in I(x)} u_i\tilde{g}_i(x)$ where $\sum_{i\in I(x)} u_i=1$ and $u_i\geq 0$, $i\in I(x)$.) If $s=0$ go to the STEP 3, else go to the STEP 5.

STEP 3: Determine the Lagrange multiplier vector v as the solution of vector equation $R^TRv = A^Tg$ where

$$g = \sum_{i\in I(x)} u_i g_i(x)$$

and where u_i, $i\in I(x)$ are multipliers obtained in the STEP 2. If $v\geq 0$ then STOP else go to the STEP 4.

STEP 4: (Delete an active constraint) Let v_j be the most negative Lagrange multiplier. Delete index j from the set $J(x)$ and determine new matrices A, S and R. (They can be determined as in |4|). Go to the STEP 2.

STEP 5: Compute the maximum stepsize α_{max} that will keep the new point in (1) feasible (see |9|). Determine stepsize α, $0<\alpha\leq\alpha_{max}$ so that $F(x^+)<F(x)$ where $x^+=x+\alpha s$. Compute values $f_i(x^+)$, $1\leq i\leq m$ and gradients $g_i(x^+)$, $1\leq j\leq m$ of the functions in the problem (P). Compute the value of the objective function $F(x^+)$. (These values are computed clearly when stepsize determinations proceeds.)

STEP 6: (Variable metric update) Compute $\tilde{d}=\alpha\tilde{s}$ and $\tilde{y}=S^T(g^+-g)$ where

$$g^+ = \sum_{i\in I(x)} u_i g_i(x^+)$$

and where u_i, $i\in I(x)$ are multipliers obtained in the STEP 2. (g is defined in the STEP 3.) Compute $\tau=\tilde{y}^T\tilde{y}$, $\sigma=\tilde{y}^T\tilde{d}$ and $\varepsilon=\tilde{d}^T\tilde{d}$. If $\tau\leq 10^{-4}\ \tilde{g}^T\tilde{g}$ or $\sigma\leq 10^{-4}\tau$ then set CASE=2 and go to the STEP 7 else set CASE=1 and update the matrix S using one of following rules

$$S^+ = S + \frac{1}{\tau} s \left(\sqrt{\frac{\tau}{\sigma}}\,\tilde{d}-\tilde{y}\right)\tilde{y}^T \qquad (a)$$

$$S^+ = S + \frac{1}{\sigma} s\tilde{d} \left(\sqrt{\frac{\sigma}{\varepsilon}}\,\tilde{d}-\tilde{y}\right)^T \qquad (b)$$

$$S^+ = S + \frac{\sqrt{\frac{\varepsilon-\sigma}{\sigma-\tau}}-1}{\varepsilon-2\sigma+\tau} s(\tilde{d}-\tilde{y})(\tilde{d}-\tilde{y})^T \qquad (c)$$

(Note that (a) corresponds to DFP update, (b) corresponds to BFGS update and (c) corresponds to rank one update.)

STEP 7: (Add an active contraint) If $\alpha<\alpha_{max}$ go to the STEP 8. If $\alpha=\alpha_{max}$ determine the index j of the active constraint $a_j^T x^+=b_j$ which has limited the stepsize α. Set $J(x^+)=J(x)\cup\{j\}$ and $A^+=[A, a_j]$. Determine the new matrix S^+ as in [8] and the new matrix R^+ as in [3]. Set $A=A^+$, $S=S^+$ and $R=R^+$.

STEP 8: Set $x=x^+$, $f_i(x)=f_i(x^+)$, $1\leq i\leq m$, $g_i(x)=g_i(x^+)$, $1\leq i\leq m$, $F(x)=F(x^+)$, $J(x)=J(x^+)$ and go to the STEP 2.

3. THE COMBINED LP AND VM ALGORITHM

This section contains a brief description of and algorithm which is a combination of linear programming (LP) algorithm described in [7] and variable metric (VM) algorithm described in the previous section. The combined LP and VM algorithm consists of two stages and two switches. The first is LP stage which is completely described in [7]. The VM stage begins only if SWITCH1 is satisfied at the end of LP stage. If VM stage fails (SWITCH2) then LP stage in automatically performed again.

SWITCH1: The LP stage is left if following three conditions are satisfied: $I(x)$ has not been changed in three immediately subsequent iterations, the stepsize has been limited and $\|g\|$ has been small (see [5] for details).

SWITCH2: The VM stage is left if CASE=2 occurs in the STEP 6 fo the algorithm described in the previous section.

The combined LP and VM algorithm has good properties of both LP and VM algorithms i.e. possible quadratic rate of convergence of the LP algorithm and superlinear rate of convergence of the VM algorithm.

4. NUMERICAL EXPERIMENTS

Efficiency of three algorithms was tested by means of examples 1, 2, 4, 5 and 7 published previously in [7]. Results of the tests are arranged in the table 1.

	Ex.1	Ex.2	Ex.4	Ex.5	Ex.7
LP	NI= 7 NF= 9 NG= 9	NI=56 NF=76 NG=76	NI= 9 NF=11 NG=11	NI= 7 NF= 9 NG= 9	NI= 7 NF=11 NG=11
VM	NI=11 NF=24 NG=13	NI= 4 NF=11 NG= 6	NI=11 NF=24 NG=13	NI= 6 NF=14 NG= 8	NI=118 NF=241 NG=120
LP + VM	NI= 7 NF= 9 NG= 9	NI= 6 NF=10 NG= 8	NI= 9 NF=11 NG=11	NI=10 NF=17 NG=14	NI= 7 NF=11 NG=11

Table 1.

Each column of the table 1 corresponds to one example (numbers agree with [7]). The table 1 has 3 rows which correspond to three algorithms LP, VM and combined LP+VM. Three numbers are given for each algorithm and each example. They are number of iterations, NI, number of function evaluations NF and number of gradient evaluations NG. The same accuracy of results was reached by each algorithm.

The numerical experiments show the high efficiency of the combined LP and VM algorithm which has good properties of both

LP and VM algorithms. When the quadratic rate of convergence of the LP algorithm does not occur as in the example 2 then superlinear rate of convergence of the VM algorithm is yet kept. Only when both LP and VM algorithms have approximately the same efficiency, as in the example 5, the combined algorithm can be less effective but comparable with both of them.

5. CONCLUSION

We propose a new efficient algorithm for linearly constrained discrete nonlinear minimax approximation which can be used for filter design. This algorithm was implemented as FORTRAN subroutine in software package for optimization and nonlinear approximation SPONA (see [6]). All results in the previous section have been attained by this subroutine.

REFERENCES

[1] Demyanov, V.F., Malozemov, V.M.: Introduction to minimax. Wiley, New York, 1974

[2] Fletcher, R.: The calculation of feasible points for linearly constrained optimization problems. A.E.R.E.Harwell, Rept.No.R-6354, 1970

[3] Gill, P.E., Murray, W.: A numerically stable form of the simplex algorithm. Linear Algebra Appl. 7., 1973, 99-138.

[4] Gill, P.E., Murray, W.: Newton-type methods for unconstrained and linearly constrained optimization. Math. Programming 7, 1974, 311-350.

[5] Hald, J., Madsen, K.: Combined LP and Quasi-Newton methods for minimax optimization. Math. Programming 20, 1981. 49-62.

[6] Lukšan, L.: Software package for optimization and nonlinear aproximation.Proc. of the 2nd IFAC/IFIP Symposium on software for computer control, Prague, 1979

[7] Madsen, K, Schjaer-Jacobsen, H.: Linearly constrained minimax optimization. Math. Programming 14, 1978, 208-223

[8] Ritter, K.: A variable metric method for linearly constrained minimization problems. In: "Nonlinear programming 3" /O.L. Mangasarian, R.R. Meyer, S.M. Robinson eds./ Academic Press, London, 1978.

[9] Rosen, J.B.: The gradient projection method for non-linear programming. Part I:Linear constraints. SIAM J. Appl. Math.8, 1960, 181-217

[10] Wolfe,P.: Finding the nearest point in a polytope. Math. Programming 11, 1976, 128-149.

SPONA: SOFTWARE PACKAGE FOR OPTIMIZATION AND NONLINEAR APPROXIMATION

L.Lukšan

(Praha, Czechoslovakia)

1. INTRODUCTION

The software package for optimization and nonlinear approximation described below is a modification and extension of the former version described in [1]. The SPONA package has a modular structure. Its FORTRAN modules are stored in the external storage of the computer. The SPONA package is controlled by a simple input language. The statements of the input language. The statements of the input language are the input data for the compiler, which generates the control program. The compiler is written in FORTRAN and its output is again a set of FORTRAN statements, stored in the external storage. The control program contains a calling sequence of all optimization subroutines. It is linked together with all necessary modules and with external subroutines supplied by user in the next step. The created block of modules solves a problem which is defined by external subroutines and by the input data (see Fig. 1).

2. OPTIMIZATION SUBROUTINES AND PROBLEMS

The SPONA package contains more than 100 optimization subroutines based on various optimization methods. These subroutines are designed for solving of optimization problems which can be divided into 7 classes:

(1) Unconstrained minimization (44 subroutines). These subroutines can find a minimum of an objective function $F(x)$ where $x \in R_n$ (n-dimensional vector space). Almost all of them find an arbitrary local minimum but two subroutines can find a global minimum. Subroutines are based on direct methods which do not use derivatives, on gradient methods which use first

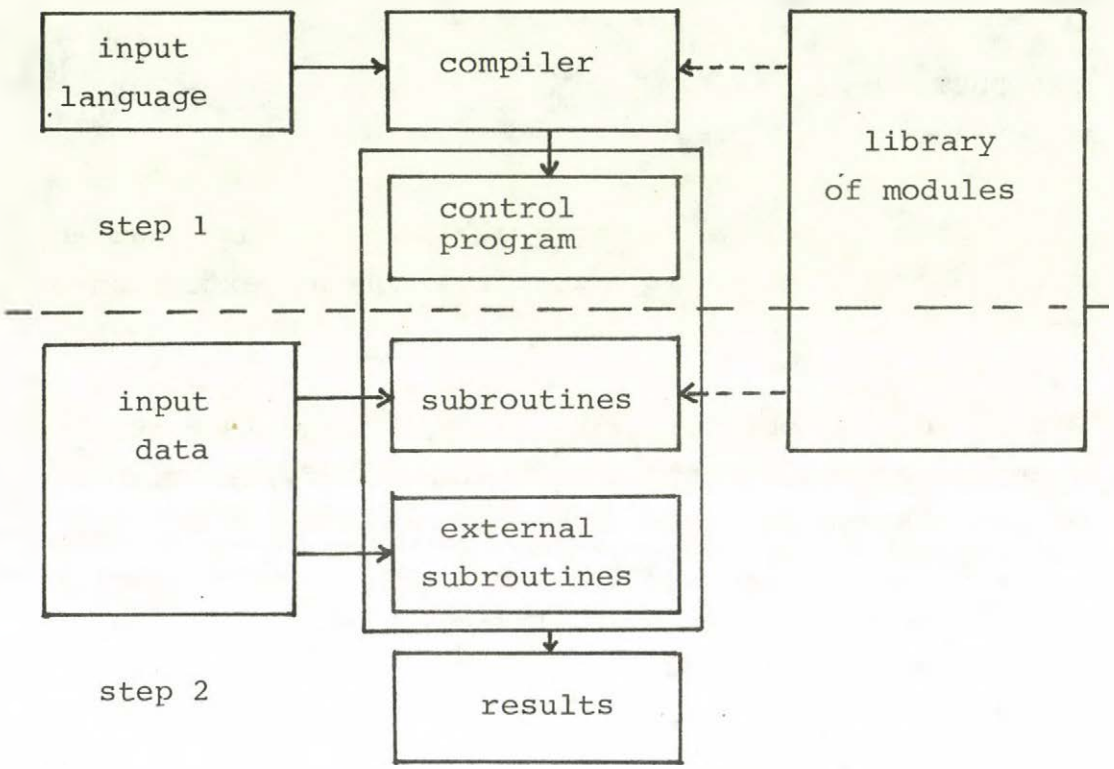

Fig. 1.

derivatives computed either analytically or numerically, or on second derivative methods. Standard problems can have at most 30 variables and large scale problems, which are solved by special subroutines, can have at most 1000 variables.

(2) Linearly constrained minimization (36 subroutines). These subroutines can find a local minimum of an objective function $F(x)$, $x \in R_n$ in the convex polytope given by inequalities $l_i \leq x_i \leq u_i$, $1 \leq i \leq n$ and by additional linear constraints

$$a_j^T x = b_j, \quad 1 \leq j \leq k$$

$$a_j^T x \geq b_j, \quad k < j \leq m.$$

Some subroutines solve special problems as linear or quadratic programming problems. Problems of this class can have at most 30 variables and 100 additional constraints.

(3) Least squares approximation (15 subroutines). These subroutines can find a local minimum of the objective function

$$F(x) = \sum_{i=1}^{m} (w_i | f_i(x) - y_i |)^2 \qquad (a)$$

with weights w_i, $1 \leq i \leq m$ and observations y_i, $1 \leq i \leq m$. Some subroutines can solve problems with linear constraints defined above. Problems of this class can have at most 30 variables, 100 additional constraints and 300 observations.

(4) Minimax approximation (7 subroutines). These subroutines can find a local minimum of the objective function

$$F(x) = \max_{1 \leq i \leq m} (w_i | f_i(x) - y_i |) \qquad (b)$$

with weights w_i, $1 \leq i \leq m$ and observations y_i, $1 \leq i \leq m$. Some subroutines can solve problems with linear constraints defined above. Problems of this class can have at most 30 variables, 100 additional constraints and 300 observations.

(5) Nonlinear programming (4 subroutines). These subroutines can find a local minimum of an objective function $F(x)$, $x \in R_n$ in the feasible region given by inequalities $l_i \leq x_i \leq u_i$, $1 \leq i \leq n$ and by additional nonlinear constraints

$$f_j(x) = 0, \quad 1 \leq j \leq k$$

$$f_j(x) \geq 0, \quad k < j \leq m.$$

Problems of this class can have at most 30 variables and 100 additional constraints.

(6) Solving of system of nonlinear functional equations (3 subroutines, at most 30 variables and 30 equations).

(7) Solving of systems of nonlinear differential equations (3 subroutines, at most 30 variables and 30 equations).

Note that an arbitrary objective function $F(x)$, $x \in R_n$ in classes (1), (2) and (5) can be replaced by special functions (a) and (b). Also further special functions can be used. For example, the function

$$F(x) = \int_{t_o}^{t_m} \varphi(t, y(t), x)\, dt \qquad (c)$$

where

$$\frac{dy(t)}{dt} = f(t, y(t), x), \quad y(t_o) = y_o$$

serves for estimation of parameters of the differential systems. Another special function can be used for centering in optimal tolerance design. These special functions are programmed in the SPONA package and are chosen automatically simultaneously with optimization subroutines by means of statements of input language.

3. ADDITIONAL POSSIBILITIES OF THE PACKAGE

The SPONA package has some additional possibilities which are advantageous for users. A sequence of up to five optimization subroutines can be defined for solving an optimization problem. In the case when some optimization subroutine fails the next one is started automatically. The external subroutines have a simple and unified form an can be tested by means of facilities contained in the SPONA package. All problems can be

scaled and some variables can be blocked automatically so that the optimization problem is simplified. The SPONA package contains many input and output subroutines. Furthermore, it contains standard test examples and facilities for testing of optimization subroutines.

4. CONCLUSION

The SPONA package has been used for solution of many technical problems most complicated of which was the design of an electromechanical filter produced by TESLA establishment. This package has also been used for testing the optimization methods. After many tests it was observed that there is no universal method which would be the best for all cases. This is the reason for building the extensive packages of optimization programs. The experience with the SPONA package is satisfactory and further development is assumed.

REFERENCES

[1] Lukšan, L.: Software package for optimization and nonlinear approximation. Proc. of the 2nd IFAC/IFIP Symposium on software for computer control, Prague 1979.

THE HEURISTIC METHODS OF DISCRETE PROGRAMMING - I

The method of neighbourhood

B.Vizvári

(Budapest, Hungary)

The aim of the paper is to discuss a heuristic method. In Section 1 we give a general description of it. In the next section we show some examples where the algorithm is not heuristic but exact method. The second aim of the same section is to demonstrate how wide-spread the method is. Therefore it cannot be regarded as somebody's discovery (in contradiction with papers [5],[6]). An application in integer programming is given in the last two sections.

1. THE GENERAL FRAME OF THE METHOD

This paper is devoted to a very efficient method. It gives exact algorithms in some cases and is only a heuristic method for other problems.

Let us assume that we have a finite set P and we are looking for a special element of P. Let a function $g(x)$ be given on P. It measures how far the properties of element x are from the desired properties. That means: if we have x_1, $x_2 \in P$ and

$$g(x_1) < g(x_2)$$

then we can state that x_2 is better than x_1. Therefore we can turn from the investigation of x_1 to that of x_2.

We need only one more notion to the method. For every element x of P let a subset $S(x)$ of P which has much less element than P, be given. We call $S(x)$ the neighbourhood of element x.

Now we can give the general frame of our method.

STEP 1. Let $k=0$. Choose $x_0 \in P$.

STEP 2. Let $G=\{x: x \in S(x_k);\ g(x_k) \underset{(=)}{<} g(x)\}$

STEP 3. a) if $G=\emptyset$ then the method is finished
b) else let $x_{k+1} \in G$ and $k=k+1$ go to STEP 2.

What is the idea of this method? If $S(x)$ contains much less elements than P, then in every iteration we have a simpler problem than the original one (STEP 2 and STEP 3). Furthermore we choose the neighbourhoods in such a way that for every two elements x, y of P the intersection

$$S(x) \cap S(y)$$

is small. That means: from a small subset of P we can reach a large subset, e.g. if

$$Q \subset P \quad \text{and} \quad |Q| << |P|$$

then

$$|Q| << |\{y;\ \exists\, x \in Q,\quad y \in S(x)\}|$$

Therefore the number of the necessary iterations will be small.

What does the method give? In the k-th iteration the procedure chooses a point from the neighbours of x_k which is better than x_k or at least as good. Finally we get a local optimum of the problem

$$\max_{x \in P} g(x)$$

Actually when the algorithm is finished then there is no point in the neighbourhood of x_k which is better than x_k in the sense defined by function $g(x)$. In the case of some problem classes x_k has always the desired properties therefore the method is exact. In other cases it is not true so we have only a heuristic algorithm.

Apparently there are some points in the general frame of our method which can be performed in very different ways for the different problems. These points are the following:

a) How to choose the function $g(x)$.
b) How to choose the neighbourhood $S(x)$.
c) The choice of the starting point x_o.
d) In the definition of the set G we can use either the constraint

$$g(x) < g(x_k)$$

or the constraint

$$g(x) \leq g(x_k).$$

The advantage of the first one is that we avoid cycling. However in the case of the second one we can reach more points, therefore our local optimum point can be better.

e) The choice of x_{k+1} from the set G.

There are a lot of possible strategies here. An extreme one is the following. We solve the problem

(1)
$$\max_{x \in G} g(x)$$

and x_{k+1} is the optimal solution. Since $S(x)$ contains only "few" elements, Problem (1) is not as hard as the original problem. A second extreme strategy is that we enumerate the elements of $S(x_k)$ and we choose the first one which is better than x_k. We can combine the two strategies, e.g. we change the point x_k only after a certain improvement.

2. SOME FAMOUS PROBLEMS

2.1 The shortest route problem

Let a graph $F=(V,E)$ be given without loops and multiple edges, where V denotes the set of vertices and E is the set of the edges of F. The length of every edge is known. The problem is to find the shortest route in the graph between the vertices A and B.

One of the well-known methods to solve this problem is based on dynamic programming (Bellman [1]). If we have a shortest route between A and B and this route goes through the vertices C and D, then the subroute with the endpoints C and D is a shortest route between C and D. We determine the shortest route from A to every vertex. Thus we shall have the desired route as well. Notice that in this way it is sufficient to give to every vertex W the point from which we go to W. From this information every route can be reconstructed.

Assume that the graph has n vertices, denoted by the positive integers from 1 to n. The starting point is vertex 1. Now the system of the routes can be given by an n dimensional vector $\underline{v}$, where v_j is the vertex in the route containing j just before j. If $v_j=0$ then we still have no route yet from 1 to j. Obviously the number of all possible vectors $\underline{v}$ is exp (n).

Let i and j be two vertices. If the (i,j) edge is in the graph then let c_{ij} be the length of the edge. Let m be a sufficiently large number, e.g.

$$m \geq \sum_{(i,j)\in E} c_{ij}+1$$

For a given vector $\underline{v}$ let $\underline{k}$ be the vector of costs of the routes. If $v_j=0$ then $k_j=m$. If $(i,j)\notin E$ then let $c_{ij}=m$. The cost k_j can be obtained trivially from the recursive formula:

$$k_j = c_{v_j j} + k_{v_j}$$

Now we have to give the set $S(\underline{v})$ and the function $g(\underline{v})$. Let $S(\underline{v})$ be the set of such vectors which differs from $\underline{v}$ in only one component i.e.

$$S(\underline{v}) = \left\{ \underline{w} : \exists\, \ell,\ 1<\ell<n;\quad w_\ell \neq v_\ell;\quad w_j = v_j,\ j\neq\ell \right\}$$

The number of the neighbours of an arbitrary vector $\underline{v}$ is $(n-1)^2$.

We choose function $g(\underline{v})$ as the total cost of the route, i.e.

$$g(\underline{v}) = \sum_{j=2}^{n} k_j.$$

In the definition of the set G we shall insist on the strict inequality

(2) $$g(w) < g(\underline{v}_k)$$

as here we minimize the costs, therefore we minimize the function $g(\underline{v})$, too. An arbitrary better solution will be choosen from G. The starting point (i.e. STEP 1 of the method) is the following:

STEP 1: if $(1,j) \in E$ then $v_j = 1$, otherwise $v_j = 0$

$j \geq 2$,

$k_j = c_{1j}$

Remember that if $(1,j) \notin E$ then $c_{1j} = m$ and so

$$k_j = m,$$

too.

Condition (2) is used in the definition of the set G. It is equivalent to the following constraints:

(3) $$\exists (i,j) \in E$$

(4) $$k_i + c_{ij} < k_j$$

Formally we do not need Constraint (3).
Namely if $(i,j) \notin E$, then $c_{ij} = m$ and hence

$$k_i + c_{ij} = k_i + m \geq k_j.$$

If (4) is true then

$$v_j \neq i.$$

Otherwise

$$k_i + c_{ij} = k_{v_j} + c_{v_j j} = k_j$$

The equivalence of Condition (2) and Constraints (3) and (4) are based on the well-known fact, that the numbers k_j give a decreasing sequence. Constraints (3) and (4) give the usual way of the definition of set G.

In this treatment of the method an administrative substep is necessary in STEP 3. Namely if we have an improvement in

k_j for a certain j then it can give improvement at other vertices, too. But the substep can be avoided by the careful organization of the algorithm.

2.2 The flowshop problem

There are given n jobs and two machines. Each job has two operations: a first one on Machine 1 and a second one on Machine 2. We process the jobs in the same order on the two machines. The processing time of job i on machine j is t_{ij} $i=1,\ldots,n$; $j=1,2$. The problem is to find the optimal order for the jobs where the total processing time is minimal.

Johnson's method is the following:

Suppose, that we have already optimal starting and finishing subsequences, but there are jobs for which the optimal order is not determined. Denote by N the set of such jobs. If $i \in N$ then we say that the processing times t_{i1} and t_{i2} belong to N. Let t be the shortest processing time belonging to N. Suppose that it occurs at job i. There are two cases: $t=t_{i1}$ or $t=t_{i2}$. In the first case i will be the last element of the starting subsequences and in the second case i will be the first element of the finishing subsequence. Jonhson proved the optimality of the procedure in [3].

From the proof it is obvious that in an arbitrary order we can exchange the order of two jobs which are after each other and their order relatively to each other is bad according to Johnson's method. With the repeated application of this fact the position of certain jobs can be exchanged to many places. This gives the idea for the following treatment of the method. The points are the different orders of the n jobs. This set is denoted by P. The neighbourhood of a given order

$$0 = (i_1, \ldots, i_n)$$

$$S(0) = \{R: R \in P;\ \exists (k,m) \quad \text{indices}$$

$$R = (i_1, \ldots, i_k, i_m, i_{k+1}, \ldots, i_{m-1}, i_{m+1}, \ldots i_n)\}$$

Of course in the definition of $S(0)$ we regard both of the cases $k<m$ and $k>m$. The number of neighbours of an arbitrary order 0 is $n(n-1)$. The function $g(0)$ is the total processing time. We want to minimize it, therefore we use in the definition of the set G the inequality

$$g(R) \leq g(0_k)$$

The starting point is an arbitrary order: namely the artifical order of the indices. Johnson's method gives a special rule for the choice of 0_{k+1} from G. But according to what we have said previously we need not know Johnson's result, the local optimum is a global optimum.

2.3 The greedy algorithm

Let N be a set of n elements and M a system of the subsets of N with the following property: for any $A \in M$ if $B \subset A$ then $B \in M$.

Every element p of N has a positive value: $v(p)$. The value of a set A is the sum of the value of its elements:

$$v(A) = \sum_{p \in A} v(p).$$

The problem is to find the set in M with the greatest value.

To solve the problem we have the following heuristic method. Suppose that the elements of N are in such an order that

$$v(p_1) \geq v(p_2) \geq \ldots \geq v(p_n)$$

Then the greedy method is as follows:

STEP 1: $A=\emptyset$ $i=1$

STEP 2: *if* $A \cup \{p_i\} \in M$ *then*

$A := A \cup \{p_i\}$

STEP 3: $i = i+1$

if $i \leq n$ *then go to STEP 2 else the algorithm is finished.*

The method is called greedy because in every step we choose the most valuable element of N that we are allowed to choose.

From the properties of M it follows that if p was not good then we cannot choose it at all (keeping the previous elements of N in the set A). Therefore the algorithms has at most n steps.

It is very easy to see that the method is only a heuristic algorithm. For example the well-known knapsack problem belongs to this problem class. But it is well-known that if M is a matroid then the greedy method gives always an optimal solution.

On the other hand the method is based on the neighbourhood. The starting point is the empty set. The neighbours of the set A are the sets which contain A and have one element more than A. Finally the function g is defined by the function v in the following way:

$$g(A) = \begin{cases} v(A) & \text{if} \quad A \in M \\ -\infty & \text{if} \quad A \notin M \end{cases}$$

3. APPLIYING THE METHOD FOR ZERO-ONE PROGRAMMING

The zero-one programming problem is to minimize a linear function of binary variables under the condition of linear inequalities:

$$\text{(IP)} \qquad \begin{aligned} &\min \ \underline{c}'\underline{x} \\ &A\underline{x} \geq \underline{b} \\ &\underline{x} \in \{0,1\}^n \end{aligned}$$

where A is an $m \times n$ matrix, and $\underline{c}$, $\underline{x}$, $\underline{b}$ are vectors of appropriate dimensions. Denote by F the set of the feasible solutions of (IP).

It is well-known that (IP) is a very difficult problem. Theoretically it is NP-complete. On the other hand the solution needs great computational efforts, too.

In general even to give a feasible solution is as difficult as the original problem. But the knowledge of a feasible solution accelerates the practical methods. Therefore it was a quite natural idea to generate a feasible solution by the method based on the neighbourhood.

There are two usual concepts of neighbourhood by this problem:

a) for any binary vector $\underline{x}$

$$S(\underline{x}) = \{\underline{y}: \ \underline{y} \text{ binary vector; } \underline{y} \text{ differs from } \underline{x} \text{ exactly in one component}\}$$

b) for any binary vector $\underline{x}$

$$S(\underline{x}) = \{\underline{y}: \ y \text{ binary vector; } \underline{y} \text{ differs from } \underline{x} \text{ exactly in two components}\}$$

Suppose, point $\underline{x}$ is a local optimum in the sense of the first neighbourhood. Then for $\underline{x}$ it is not necessary to be a local optimum in the sense of the second neighbourhood. That means $\underline{x}$ can be a starting point when we apply the method with te neighbours of second type. Thus we can further improve our point.

First of all we want to get a feasible solution. But after having it we want to get a better point, i.e. we want to improve our first feasible solution. These two activities can be separated by the choice of the function $g(\underline{x})$. Of course the separation is not necessary, the method is applicable in different ways.

Denote by $\underline{a}_i$ the i-th line of matrix A and by b_i the i-th component of $\underline{b}$. Point $\underline{x}$ satisfies the i-th constraint if

$$0 \geq b_i - \sum_{j=1}^{n} a_{ij}x_j$$

If w is a real number then let

$$|w|_+ = max\ \{w, o\}$$

Now we can measure the violation of a constraint by

$$|b_i - \sum_{j=1}^{n} a_{ij}x_j|_+$$

Then the violation of all the conditions is:

$$\sum_{i=1}^{m} |b_i - \sum_{j=1}^{n} a_{ij}x_j|_+$$

The sum is zero if and only if $\underline{x}$ is a feasible solution. Suppose that in the objective function vector we have nonpositive components. Then $g(x)$ separates the above mentioned two activities if

$$g(\underline{x}) = \begin{cases} \underline{c}'\underline{x} & \underline{x} \in F \\ \sum_{i=1}^{m} \left| b_i - \sum_{j=1}^{n} a_{ij} x_j \right|_+ & \underline{x} \notin F \end{cases}$$

4. COMPUTATIONAL RESULTS

The method was applied to zero-one programming. The algorithm was specified as follows:

a) The function $g(\underline{x})$ was choosen as it was suggested at the end of the last section.

b) The neighbours of a point $\underline{x}$ were the binary vectors which differ from $\underline{x}$ exactly in one component.

c) The method is sensitive for the choice of the starting point. Denote by F the set of the feasible solutions of a given problem. Let L and U be defined by the following equations:

$$L = \min_{\underline{x} \in F} \sum_{j=1}^{n} x_j$$

and

$$U = \max_{\underline{x} \in F} \sum_{j=1}^{n} x_j .$$

L is a lower bound for the number of such components of a feasible solution which are equal to one. U is an upper bound for the same quantity. The efficiency of the method depends on the choice of the starting point $\underline{x}_o$ in this respect. We have got the best results if the inequalities

$$L \leq \sum_{j=1}^{n} x_{oj} \leq u$$

were hold. (See Table I). We have chosen the starting point $\underline{x}_o$ for our problems in such a way that the frequency of ones was 30 per cent, e.g. the equation

$$\sum_{j=1}^{n} x_{oj} = 0.3 \times n$$

was satisfied.

d) At the k-th iteration we accepted a point as $\underline{x}_{k+1}$ only if it was better than $\underline{x}_k$.

e) We enumerated the neighbours of the point $\underline{x}_k$ and chose the first one that was better than $\underline{x}_k$. The order of the enumeration was determined on the following way. Before the start of the method every variable got a value. The desired order was the increasing value order. That means: the first neighbour of $\underline{x}_k$ differed in the component which had the less value, etc. We used four different evaluation of the variable x_j:

$$v_j^1 = c_j$$

$$v_j^2 = -c_j$$

$$v_j^3 = \sum_{i=1}^{m} a_{ij}$$

$$v_j^4 = \sum_{i=1}^{m} a_{ij} - d.m$$

where d is the density of the matrix. It was 0,5 in our problems.

We solved randomly generated problems. Ten problems belong to the first group. They have 10 constraints, 50 variables and are named R150 - R159. The second group consists of ten problems, too. They have 100 variables, 10 constraints. We refer them as R110-R119. Finally the third group has six problems with 30 constraints and 100 variables. Their names are R310-R315.

The method gave feasible solutions for all of these problems. The necessary CPU time was 3-10 seconds in a CDC 3300 computer. The efficiency of the method is shown by the following. We tried to solve Problem R310 with IBM's MPSX on an IBM 30/31 computer. After several attempt we have got nothing at all, though the longest running was 15 minutes.

Conclusions. The method of neighbourhood is a very efficient method in zero-one programming to generate feassible solutions. But further investigations of its behaviour are necessary at large structured and real-life problems.

REFERENCES

[1] R. Bellman, "On a Routing Problem", Quart. Appl. Math. 16 (1968)

[2] E.Boros, "The Heuristic Methods of Discrete Programming II: The Linear Method", (to appear)

[3] S.M. Johnson, "Optimal Two and Three Stage Production Schedules with Set-Up Times Included", Naval Res. Logist. Quart Vol.1. (1954) No.1. 61-68.

[4] B. Korte, "Approximative Algorithms for Discrete Optimization Problems", Institut fur Ökonometrie und Operations Research, Universität Bonn, Rep. No. 7762-OR, 1977.

[5] M.J. Krone and K. Steiglitz, "Heuristic - Programming Solution of a Flowshop-Scheduling Problem", Oprs. Res. Vol. 22. (1974) 629-638.

[6] S. Reiter and G. Sherman, "Discrete Optimizing", J. of SIAM Vol. 13. (1965) 864-889.

TABLE I

THE IMPORTANCE OF THE STARTING POINT
(EXPERIENCES WITH PROBLEM R310)

DENSITY OF THE STARTING POINT	THE VALUE OF THE OBJECTIVE FUNCTION V1	V2	V3
0%	-48	NF	NF
10%	-65	NF	NF
20%	-20	-89	-110
30%	-105	NF	-85
40%	-85	-51	-82
50%	-81	NF	-101

NF = THERE WAS FOUND NO FEASIBLE SOLUTION

IN THE K-TH VARIANT OF THE METHOD THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF THE VALUES VK(J) (J=1,...,N) WHERE

V1(J)= C(J)
V2(J)=-C(J)
V3(J)=SUMMA (I=1 TO M) A(I,J) - M*D*C(J)

TABLE II

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL F,SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION CPU TIME
R150	13	301	1	2,43	12	251	1,75;	0,73	12	251	1,75;	0,73
R151	16	411	4	3,54	12	269	2,13;	1,09	15	361	2,77;	1,62
R152	21	395	5	5,27	16	244	3,13;	1,82	20	345	4,21;	2,90
R153	28	538	4	4,63	24	353	2,73;	1,64	27	488	3,76;	2,57
R154	17	367	2	3,73	15	272	2,21;	1,16	16	317	2,82;	1,77
R155	17	469	4	3,47	13	342	2,12;	1,07	16	419	2,83;	1,78
R156	19	389	5	3,21	14	258	1,74;	0,75	18	339	2,45;	1,46
R157	28	619	5	4,07	23	471	2,57;	1,39	27	569	3,32;	2,15
R158	19	357	3	3,06	16	236	1,85;	0,99	18	307	2,36;	1,50
R159	28	513	5	4,17	23	342	2,58;	1,44	27	463	3,40;	2,26

ITER = ITERATIONS
F,SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)=C(J)

TABLE III

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME		LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME	
R150	35	474	9	4,27	26	336	2,63	1,66	34	424	3,47	2,51
R151	36	677	10	5,56	26	442	3,07	1,85	35	627	4,60	3,38
R152	18	453	3	2,96	15	354	2,02	1,09	17	403	2,32	1,39
R153	36	619	8	4,83	28	434	2,72	1,71	35	569	3,96	2,95
R154	37	597	5	4,41	32	430	2,96	1,64	36	547	3,71	2,39
R155	20	460	2	2,84	18	365	2,01	1,11	19	410	2,23	1,33
R156	28	504	3	3,43	25	433	2,21	1,23	27	454	2,81	1,83
R157	24	405	2	3,68	22	320	2,46	1,37	23	355	2,97	1,60
R158	32	593	7	3,92	25	416	2,22	1,28	31	543	3,28	1,34
R159	24	417	3	3,99	21	338	2,75	1,59	23	367	3,14	1,98

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(I)=C(J)

TABLE IV

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION CPU TIME
R150	19	455	6	3,40	13	350	1,90	0,83	18	405	2,70	1,68
R151	13	371	3	3,00	10	269	1,66	0,68	12	321	2,31	1,63
R152	17	426	2	2,93	15	307	2,01	1,09	16	376	2,22	1,30
R153	25	751	3	3,41	22	667	2,14	1,24	24	701	2,80	1,90
R154	16	467	2	2,74	14	382	1,63	0,85	15	417	2,12	1,27
R155	20	553	5	3,35	15	406	1,98	1,01	19	503	2,70	1,74
R156	20	565	6	3,21	14	382	1,66	0,81	19	515	2,58	1,74
R157	23	677	3	4,67	20	578	2,95	1,58	22	627	3,64	2,27
R158	17	517	2	3,02	15	436	1,94	0,96	16	467	2,33	1,44
R159	21	445	3	3,37	18	342	2,24	1,26	20	395	2,68	1,71

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE

$$V(J) = \text{SUMMA } (I=1 \text{ TO } M)\ A(I,J)$$

TABLE V

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME		LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME	
R150	20	523	5	4,77	15	439	2,51;	1,23	19	473	3,93;	2,65
R151	13	263	2	3,40	11	206	2,07;	0,95	12	213	2,53;	1,59
R152	16	473	1	4,07	15	423	3,00;	1,78	15	423	3,00;	1,78
R153	31	782	6	5,76	25	616	3,23;	1,70	30	732	4,76;	3,23
R154	40	834	12	4,66	28	545	2,81;	1,79	39	784	3,72;	2,70
R155	22	514	7	3,75	15	368	1,90;	0,87	21	464	3,15;	2,12
R156	20	539	5	3,44	15	409	1,93;	0,93	19	489	2,73;	1,73
R157	32	832	8	4,23	24	625	2,33;	1,30	31	782	3,54;	2,51
R158	19	623	4	3,83	15	465	1,85;	0,90	18	573	3,10;	2,15
R159	23	600	2	3,91	21	542	2,68;	1,40	22	550	3,05;	1,77

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)= SUMMA (I=1 TO M) A(I,J)*M*D*C(J)

TABLE VI

PROBLEM	LP	V1			V2			V3			V4		
NAME	OPTIMUM	N.S.	N.F.	ZB	N.S.	N.F.	ZB	N.S.	N.F.	ZB	N.S.	N.F.	ZB
R150	-7	301	1	20	474	9	22	455	6	11	523	5	21
R151	-6	411	4	12	677	10	19	371	3	16	263	2	17
R152	-15	395	5	2	453	3	-4	426	2	7	473	1	6
R153	-14	538	4	0	619	8	3	751	3	2	782	6	0
R154	14	367	2	45	597	5	24	467	2	40	834	12	37
R155	-25	469	4	-7	460	2	-12	553	5	-12	514	7	-10
R156	-16	389	5	1	504	3	1	565	6	-3	539	5	0
R157	-8	619	5	10	405	2	10	677	3	6	832	8	12
R158	-3	357	3	21	593	7	10	517	2	23	623	4	20
R159	-13	513	5	5	417	3	-1	445	3	3	600	2	0

N.S. =THE NUMBER OF STEPS
N.F. =THE NUMBER OF FEASIBLE SOLUTIONS
ZB =THE VALUE OF THE OBJECTIVE FUNCTION AT THE BEST FEASIBLE SOLUTION FOUND BY THE HEURISTIC METHOD

IN THE K-TH VARIANT OF THE METHOD THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES VK(J) (J=1,...,N) WHERE
V1(J)=C(J); V2(J)=-C(J); V3(J)=SUMMA (I=1 TO M) A(I,J);
V4=SUMMA (I=1 TO M) A(I,J)-M*D*C(J)

TABLE VII

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME
R110	36	992	25	14,80	11	292	7,02; 1,85	35	892	12,51; 7,34
R111	23	741	17	11,87	6	241	6,20; 1,27	22	641	9,92; 4,98
R112	26	1171	21	15,29	5	367	5,23; 0,86	25	1071	12,94; 8,57
R113	26	884	20	14,05	6	293	6,63; 1,67	25	784	11,07; 7,11
R114	21	754	16	12,72	5	116	6,52; 1,23	20	654	10,71; 5,42
R115	27	1110	15	13,41	12	425	7,50; 2,37	26	1010	11,08; 6,94
R116	34	1238	15	13,03	19	697	7,81; 1,79	33	1138	10,97; 5,95
R117	25	1114	11	9,50	14	500	5,40; 1,54	24	1014	7,69; 4,83
R118	43	1906	22	16,72	21	785	5,92; 2,08	42	1806	14,05; 10,21
R119	17	822	3	10,54	14	695	6,59; 2,35	16	722	8,53; 4,28

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)=C(J)

TABLE VIII

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME
R110	29	981	18	12,92	11	425	5,27; 1,39	28	881	12,92; 9,14
R111	22	842	16	15,02	6	271	6,56; 1,76	21	742	12,82; 7,98
R112	30	1425	24	15,49	6	296	7,07; 1,34	29	1325	13,44; 7,72
R113	35	1044	24	16,41	11	271	7,25; 1,94	34	944	14,01; 8,70
R114	21	743	12	11,04	9	355	5,56; 1,40	20	643	8,69; 4,54
R115	23	905	17	10,45	6	256	4,54; 0,73	22	805	8,45; 4,64
R116	23	995	5	13,72	18	682	7,66; 2,51	22	895	11,53; 6,37
R117	31	1122	14	14,76	17	581	8,02; 2,41	30	1022	12,61; 7,00
R118	25	851	10	12,72	15	511	7,56; 2,79	24	751	10,29; 5,52
R119	22	1022	10	12,31	12	586	6,71; 1,64	21	922	10,18; 5,11

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)=C(J)

TABLE IX

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME		LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME	
R110	27	1350	17	13,44	10	463	6,48	1,47	26	1250	11,12	6,11
R111	24	1178	18	12,78	6	275	6,89	1,22	25	1078	10,75	4,88
R112	25	1567	18	11,01	7	445	5,23	1,23	24	1467	8,80	4,81
R113	22	1012	18	12,87	4	91	6,69	0,90	21	912	10,72	4,93
R114	21	938	17	13,25	4	237	6,07	0,97	20	838	11,06	5,96
R115	25	1068	15	14,54	10	317	7,37	1,95	24	968	12,36	6,94
R116	25	1276	13	11,73	12	643	5,68	1,46	24	1176	9,14	4,22
R117	25	1041	12	12,56	13	621	5,77	1,93	24	941	9,46	6,61
R118	30	1260	12	12,62	18	698	7,36	2,14	29	1160	10,53	5,31
R119	25	1341	11	14,91	14	552	7,97	2,86	24	1241	12,72	7,60

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)= SUMMA (I=1 TO M) A(I,J)

TABLE X

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME		LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME	
R110	27	1343	17	10,07	10	543	5,00;	1,36	26	1242	8,17;	4,55
R111	23	1062	17	11,59	6	199	4,77;	1,04	22	962	9,67;	5,94
R112	21	1006	12	13,53	9	371	5,65;	1,57	20	906	11,11;	7,03
R113	28	1318	18	14,73	10	395	6,41;	1,92	27	1218	12,23;	7,64
R114	26	1238	19	13,12	7	326	5,71;	1,31	25	1138	11,06;	6,36
R115	24	1101	15	10,71	9	303	5,25;	1,31	23	1001	8,65;	4,71
R116	20	890	6	10,14	14	404	5,18;	1,49	19	790	10,14;	6,44
R117	28	1537	12	14,37	16	721	7,87;	2,51	27	1437	11,66;	6,30
R118	27	1302	10	14,38	17	842	8,31;	2,84	26	1202	11,61;	6,14
R119	24	1317	10	11,03	14	787	6,25;	2,05	23	1217	8,93;	4,74

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)= SUMMA (I=1 TO M) A(I,J)=M*D*C(J)

TABLE XI

PROBLEM	LP	V1		V2		V3		V4	
NAME	OPTIMUM	N.S.	N.F. ZB	N.S.	N.F. ZB	N.S.	N.F. ZB	N.S.	N.F. ZB
R110	-241	992	25 -239	981	18 -221	1350	17 -223	1343	17 -211
R111	-228	741	17 -191	842	16 -205	1178	18 -212	1062	17 -200
R112	-289	1171	21 -276	1425	24 -278	1567	18 -277	1006	12 -249
R113	-242	884	20 -224	1044	24 -233	1012	18 -231	1318	18 -226
R114	-237	754	16 -220	743	12 -219	938	17 -223	1238	19 -223
R115	-222	1110	15 -185	905	17 -203	1068	15 -206	1101	15 -199
R116	-220	1238	15 -174	995	5 -145	1276	13 -183	890	6 -154
R117	-195	1114	11 -154	1122	14 -145	1041	12 -157	1537	12 -168
R118	-235	1906	22 -202	851	10 -183	1260	12 -181	1302	10 -193
R119	-208	822	3 -170	1022	10 -163	1341	11 -189	1317	10 -171

N.S. =THE NUMBER OF STEPS
N.F. =THE NUMBER OF FEASIBLE SOLUTIONS
ZB =THE VALUE OF THE OBJECTIVE FUNCTION AT THE BEST FEASIBLE SOLUTION FOUND BY THE HEURISTIC METHOD

IN THE K-TH VARIANT OF THE METHOD THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES VK(J) (J=1,...,N) WHERE
V1(J)=C(J); V2(J)=-C(J); V3(J)=SUMMA (I=1 TO M) A(I,J);
V4=SUMMA (I=1 TO M) A(I,J)-M*D*C(J)

TABLE XII

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME
R310	37	1347	6	16,73	31	1004	10,17; 6,16	36	1247	14,22; 10,20
R311	40	1611	1	16,02	39	1511	13,25; 8,87	39	1511	13,25; 8,87
R312	47	1752	11	19,11	36	1290	9,95; 4,99	46	1652	15,76; 10,77
R313	45	1823	10	21,87	35	1248	9,41; 5,05	44	1723	18,90; 14,55
R314	46	1647	11	19,67	35	1131	7,90; 3,93	45	1547	16,81; 12,83
R315	53	2067	16	23,83	37	1121	10,41; 5,79	52	1967	20,95; 16,33

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)=C(J)

TABLE XIII

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME
R310	44	1754	4	20,32	40	1504	12,87; 8,91	43	1654	16,76; 12,80
R311	48	1848	9	20,27	39	1376	10,37; 6,04	47	1748	17,30; 12,98
R312	51	1600	12	14,59	39	1111	7,70; 3,95	50	1500	12,18; 6,42
R313	43	1298	8	12,63	35	893	7,98; 4,25	42	1198	9,65; 5,93
R314	39	1824	5	11,26	34	1576	7,03; 3,53	38	1724	8,89; 5,39
R315	31	1222	7	14,41	24	694	7,04; 3,29	30	1122	11,73; 7,98

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(J)=-C(I)

REMARK: IN PROBLEM R310 AND R314 THE DENSITY OF THE STARTING POINTS WAS 20 PERCENT.

TABLE XIV

PROBLEM NAME	TOTAL NUMBER OF ITER	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F.SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITER	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITER	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME
R310	35	1911	5	17.71	30	1623	11.26; 6.66	34	1811	14.92; 8.42
R311	38	1626	1	12.92	37	1526	10.18; 5.75	37	1526	10.18; 5.75
R312	44	1996	5	18.52	39	1768	11.06; 7.26	43	1896	15.88; 12.08
R313	32	1536	4	16.16	28	1319	9.51; 4.47	31	1436	13.30; 8.26
R314	40	1947	8	14.82	32	1554	7.35; 3.64	39	1847	12.28; 8.56
R315	50	2393	12	18.75	38	1845	9.92; 6.79	49	2293	16.28; 12.65

ITER = ITERATIONS
F.SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE
V(I)= SUMA (I=1 TO M) A(I,J)

REMARK: IN PROBLEM R312 THE DENSITY OF THE STARTING POINT WAS
40 PERCENT

TABLE XV

PROBLEM NAME	TOTAL NUMBER OF ITERATIONS	TOTAL NUMBER OF STEPS	TOTAL NUMBER OF F,SOL	TOTAL CPU TIME	FIRST FEASIBLE SOLUTION NUMBER OF ITERATIONS	FIRST FEASIBLE SOLUTION NUMBER OF STEPS	FIRST FEASIBLE SOLUTION CPU TIME	LAST FEASIBLE SOLUTION NUMBER OF ITERATIONS	LAST FEASIBLE SOLUTION NUMBER OF STEPS	LAST FEASIBLE SOLUTION CPU TIME
R310	44	2214	7	14,39	37	1698	9,01; 5,61	43	2114	11,93; 8,53
R311	59	3126	15	20,70	44	2245	10,76; 6,96	58	3026	18,16; 14,37
R312	49	2487	14	13,83	35	1704	7,58; 4,21	48	2387	11,59; 8,22
R313	34	1781	2	11,92	32	1587	8,48; 5,09	33	1681	9,47; 6,08
R314	40	1980	4	12,26	36	1852	7,68; 4,19	39	1880	9,93; 6,45
R315	38	2049	10	15,88	28	1357	9,23; 4,89	37	1949	13,01; 8,67

ITER = ITERATIONS
F,SOL= FEASIBLE SOLUTIONS

THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES

V(J) (J=1,...,N) WHERE

$$V(J) = \text{SUMMA } (I=1 \text{ TO } M)\ A(I,J)*M*D*C(J)$$

TABLE XVI

PROBLEM	V1			V2			V3			V4		
NAME	N.S.	N.F.	ZB	N.S.	N.F.	ZB	N.S.	N.F.	ZB	N.S.	N.F.	ZB
R310	1347	6	-105	1754	4	-70	1911	5	-98	2214	7	-85
R311	1611	1	-63	1848	9	-89	1626	1	-68	3126	15	-91
R312	1752	11	-129	1600	12	-111	1996	5	-132	2487	14	-159
R313	1823	10	-87	1298	8	-113	1536	4	-98	1781	2	-94
R314	1647	11	-100	1824	5	-59	1947	8	-98	1980	4	-63
R315	2067	16	-167	1222	7	-168	2393	12	-139	2049	10	-180

N.S. =THE NUMBER OF STEPS
N.F. =THE NUMBER OF FEASIBLE SOLUTIONS
ZB =THE VALUE OF THE OBJECTIVE FUNCTION AT THE BEST FEASIBLE SOLUTION FOUND BY THE HEURISTIC METHOD

IN THE K-TH VARIANT OF THE METHOD THE ENUMERATION ORDER OF THE NEIGHBOURS WAS BASED ON THE INCREASING ORDER OF VALUES VK(J) (J=1,...,N) WHERE
V1(J)=C(J); V2(J)=-C(J); V3(J)=SUMMA (I=1 TO M) A(I,J);
V4=SUMMA (I=1 TO M) A(I,J)-M*D*C(J)

THE ELLIPSOID ALGORITHM FOR LINEAR PROGRAMMING

S. Walukiewicz

(Warsaw, Poland)

The problem concerning the efficiency of the simplex algorithm has been stated at the very beginning of linear programming. In 1969 Gale wrote in [6] that this problem "has stood as a challenge to workers in the field for twenty years and now remains, in my opinion, the principal open question in the theory of linear computation". In 1971 Klee and Minty [11] show that it is possible for the simplex algorithm to require an exponential number of iterations to reach an optimal solution (see Clausen [5] for a tutorial presentation of their result and also Charnes et al. [4]) . Therefore the simplex algorithm is not polynomial and as a consequence one can ask: Does there exist a polynomial algorithm for linear programming? The positive answer to this question was given first by Khachiyan in the beginning of 1979 who basing on the results of Shor [17] presented a polynomial algorithm for solving a linear programming problem on Turing machine [10].

The resolution of this major theoretical question concerning linear programming has resulted in a big flow of working papers and articles in popular press (see Wolfe [20] for bibliography). Almost all of these papers in more or less direct way try to answer the question: What is a practical importance of Khachiyan's algorithm? As a result many modifications have been proposed and the algorithm has been called the ellipsoid algorithm because it constructs a sequence of ellipsoids.

The aim of this paper is to present a selfcontained description of the ellipsoid algorithm and point out its theoretical and practical importance.

In Section 1 we present an idea of the algorithm. From methodological point of view it is better to show how the ellipsoid algorithm solves a given system of linear inequalities while in Section 5 we show how a given linear programming problem can be transformed into a system or systems of linear inequalities. Section 2 contains a description of the algorithm and in the next section we present some modifications of it which have proved to be useful in practice.

A sketch of the proof of polynomiality of the algorithm is given in Section 4. Section 6 presents results obtained so far in computational experiments with the ellipsoid algorithm. In concluding remarks (Section 7) we point out advantages and drawbacks of the ellipsoid algorithm in comparison with the simplex method.

1. AN IDEA OF THE ALGORITHM

Consider a system of m linear inequalities in R^n, which we will denote as

$$(P) \qquad a_i^T x \leq \beta_i, \quad i=1,\ldots,m.$$

Through this chapter we will assume that (P) satisfies three assumptions:

A1: $n \geq 2$.

A2: $a_i \neq 0$, $i=1,\ldots,m$.

A3: All data in (P) are integer.

Assumptions A1 and A2 are made to rule out some pathological cases. Surely, any system of linear inequalities with only one unknown can be solved in polynomial time. Next, if $a_i=0$ and $\beta_i \geq 0$, then the i-th inequality is redundant. On the other hand, if $a_i=0$ but $\beta_i<0$ then the i-th inequality is inconsistent and therefore (P) is inconsistent too. Assumption A3 is necessary only in the proof of polynomiality of the algorithm (Section 4).

By $F(P)$ we denote the set of all solutions to (P). We assume that all vectors are columns and a^T denotes the transposition of a.

Now we present an idea of the ellipsoid algorithm solving graphically a system of two inequalities in R^2 labeled in Fig. 1. as (1) and (2).

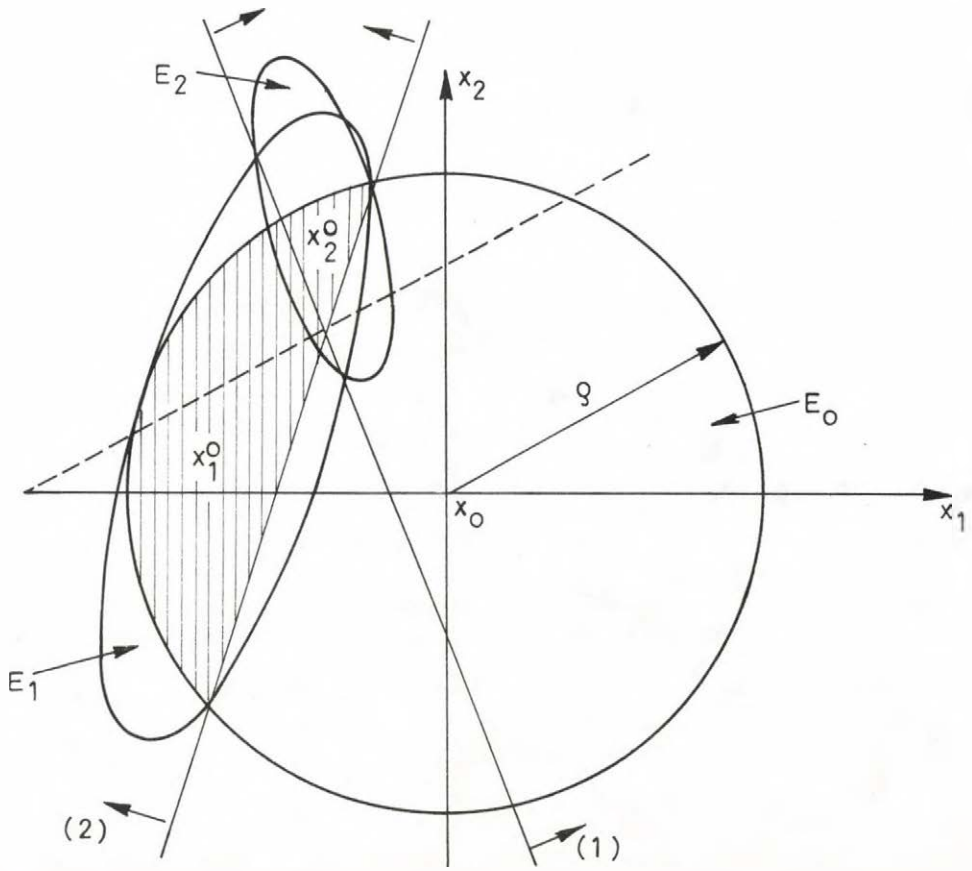

Fig.1.

At the initial iteration we construct an ellipsoid (a ball) E_o with the centre x_o at the coordinates origin and the radius ρ in such a way that it contains at least one solution to (P) if such one exists. Since $x_o \notin F(P)$, we construct the next ellipsoid E_1 in such a way that it includes the set of all candidates for the solutions to (P) contained in E_o. With reference

to violated at x_o inequality (1) the set of candidates represents a shaded part of E_o. We will call such an inequality a cut as it cuts off some part of E_o. We will show later that the algorithm converges to a solution independently of the choice of the cut (reference constraint) (in our example we may choose (2) as a cut as well), but the speed of convergence depends heavily on this choise. From Fig.1 we can see that the centre x_1 of E_1 does not satisfy (2), although it satisfies (1), so in the next iteration we construct E_2 and check that $x_2 \in F(P)$.

Let H_i be a half - space defined by the i-th inequality

$$H_i = \{x \in R^n \mid a_i^T x \leq \beta_i\}, \quad i=1,\ldots,m.$$

The ellipsoid E_k, $k=0,1,\ldots$ may be represented in the form

$$E_k = \{x \in R^n \mid (x-x_k)^T J_k^{-1} (x-x_k) \leq 1\}, \qquad (1)$$

where x_k is its centre and J_k is an $n \times n$ positive definite matrix of reals, i.e. a matrix for which $x^T J_k x \geq 0$ for any $x \in R^n$ and $x^T J_k x = 0$ if and only if $x=0$.

Using the above notation we can say that the ellipsoid algorithm constructs a sequence of ellipsoids $E_o, E_1, \ldots, E_k, \ldots$ according to two principles:

i) Inclusion Principle

$$E_k \cap H_i \subseteq E_{k+1}, \quad k=0,1,\ldots,$$

where H_i corresponds to any inequality $a_i^T x \leq \beta_i$ of (P) violated at x_k.

ii) Convergence Principle

$$\frac{Vol\ E_{k+1}}{Vol\ E_k} = r_k < 1, \quad k=0,1,\ldots,$$

where $Vol\ E_k$ is the volume of E_k in R^n.

At this point of presentation of the algorithm the above sequence of ellipsoids may be infinite, but in Section 4 we will show that under the assumption A3 the ellipsoid algorithm finds a solution to *(P)* or determines that $F(P)=\emptyset$ after finite number of iterations.

It is interesting to note that basing on i) and ii) one can construct a sequence of balls, polytopes or, in general, any convex bodies which converge to a solution of *(P)* if such one exists. In Section 7 we show that the sequence of ellipsoids has some advantages in comparison with so far known algorithms of constructing sequences of balls or polytopes.

2. A DESCRIPTION OF THE ALGORITHM

In this section we show how having an ellipsoid $E_k=(J_k,x_k)$, $k=0,1,\ldots$ one can construct the next ellipsoid $E_{k+1}=(J_{k+1},x_{k+1})$ according to the above principles. To describe this updating process in a fairly general way we transform E_k into a unit ball described in a coordinate system $z_1,\ldots,z_n$ obtained form $x_1,\ldots,x_n$ by a suitable linear transformation.

Any positive definite matrix J_k can be represented as

$$J_k = Q_k^T Q_k \tag{2}$$

where Q_k is a matrix of reals and $\det Q_k \neq 0$. Given any nonzero vector $a \in R^n$, we can transform E_k into a unit ball as follows. Let the vector $Q_k a$ be normalized to have unit Euclidean length, and then choose an orthonormal matrix R_k, i.e., $R_k^T R_k = I$, that reduces it to the first column of the identity matrix I:

$$q_k = \frac{Q_k a}{\|Q_k a\|} \quad \text{and} \quad R_k q_k = e_1 \tag{3}$$

where $\|a\| = \sqrt{a^T a}$ is Euclidean norm of a.

Now we define the linear transformation

$$x - x_k = Q_k^T R_k^T z \tag{4}$$

Substituting (4) into (1) by (2) we obtain

$$E_k = \{z \in R^n \mid z^T z \leq 1\},$$

i.e., E_k is a unit ball with its centre at the point $z_k = 0$, i.e., the origin of the new system of coordinates. The transformations (3) and (4) have a geometrical interpretation as follows. First we rotate by matrix R_k the unit ball E_k in such a way that the first unit vector becomes equal to q_k and next Q_k transforms this unit ball into E_k shifted to the origin.

We define $d_i(x_k)$ as the <u>algebraic distance</u> from x_k to the hyperplane bounding the half - space H_i in the metric corresponding to the matrix J_k

$$d_i(x_k) = \frac{a_i^T x_k - \beta_i}{\| Q_k a_i \|} = \frac{a_i^T x_k - \beta_i}{\sqrt{a_i^T J_k a_i}} \tag{5}$$

<u>Theorem 1.</u>

i) If $d_i(x_k) \leq -1$, then $H_i \cap E_k = E_k$ and the i-th inequality is redundant (inessential) in (P).

ii) If $-1 < d_i(x_k) \leq 0$, then $H_i \cap E_k \neq \emptyset$ and x_k satisfies the i-th inequality.

iii) If $0 < d_i(x_k) \leq 1$, then $H_i \cap E_k \neq \emptyset$ and x_k violates the i-th inequality.

iv) If $d_i(x_k) > 1$, then $H_i \cap E_k = \emptyset$, i.e., (P) is inconsistent.

<u>Proof.</u> Under the transformation (4) $a_i^T x \leq \beta_i$ becomes $-z \geq d_i(x_k) e_1$ or $-z_i \geq d_i(x_k)$. So $-1 < d_i(x_k) \leq 1$, if and only if $H_i \cap E_k \neq \emptyset$ and $E_k \cap H_i \neq E_k$, i.e., we have the case ii) and iii). The remaining two possibilities we have in the case i) and iv).

Geometrical interpretation of Theorem 1 is given Fig.2.

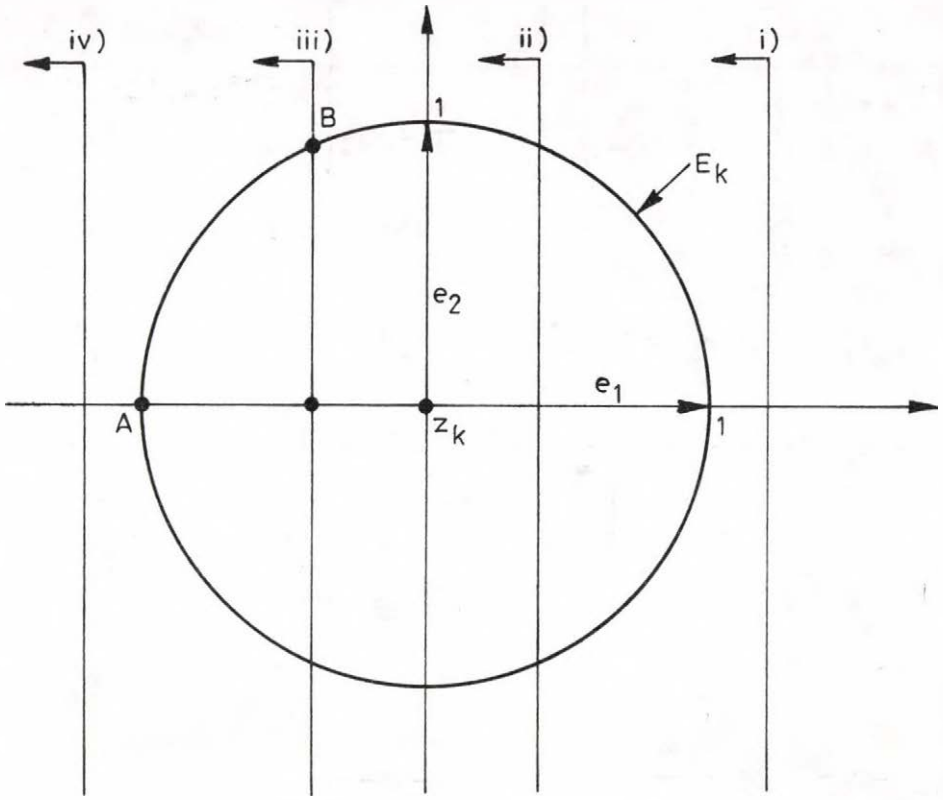

Fig. 2.

It follows from Theorem 1 that only in the case iii) we construct a new ellipsoid $E_{k+1}=(J_{k+1}, x_{k+1})$. This ellipsoid with the centre at the point

$$z_{k+1} = -\gamma e_1, \quad \gamma \geq 0 \tag{6}$$

has all major axes equal except the first one and must take the form

$$\frac{1}{\omega^2}(z_1 - z_{k+1,1})^2 + \frac{1}{\sigma^2}\sum_{j=2}^{n} z_j^2 \leq 1 \tag{7}$$

where quantities $\omega, \sigma > 0$ and $\gamma \geq 0$ have yet to be specified. In matrix notation (7) is

$$(z-z_{k+1})^T \begin{bmatrix} \frac{1}{\omega} & \\ & \frac{1}{\sigma} I_{n-1} \end{bmatrix} \begin{bmatrix} \frac{1}{\omega} & \\ & \frac{1}{\sigma} I_{n-1} \end{bmatrix} (z-z_{k+1}) \leq 1 \qquad (8)$$

From (4) we have

$$z=(Q_k^T R_k^T)^{-1}(x-x_k)=R_k(Q_k^{-1})^T(x-x_k) \qquad (9)$$

Substituting (9) and (6) in (8) we obtain

$$\frac{1}{\sigma^2}(x-x_{k+1})^T Q_k^{-1} R_k^T \begin{bmatrix} \frac{\sigma^2}{\omega^2} & \\ & I_{n-1} \end{bmatrix} R_k(Q_k^T)^{-1}(x-x_{k+1}) \leq 1 \qquad (10)$$

where

$$x_{k+1}=x_k - \gamma Q_k^T \frac{Q_k a}{\|Q_k a\|} = x_k - \gamma \frac{J_k a}{\sqrt{a^T J_k a}} \qquad (11)$$

We wish to write (10) in a form analogous to (1), namely

$$(x-x_{k+1})^T J_{k+1}^{-1}(x-x_{k+1}) \leq 1$$

where

$$J_{k+1}=Q_{k+1}^T Q_{k+1}$$

Since for a given nonsingular matrix J_k its inverse J_k^{-1} is uniquely defined and $J_k J_k^{-1} = I$, the reader can check that

$$J_{k+1} = Q_{k+1}^T Q_{k+1}=\sigma^2 Q_k^T R_k^T (I-(1-\frac{\omega^2}{\sigma^2}) e_1 e_1^T) R_k Q_k =$$
$$=\sigma Q_k^T (I-\delta q_k q_k^T)\sigma Q_k =$$
$$=\sigma Q_k^T (I-\xi q_k q_k^T)^2 \sigma Q_k$$

Therefore by (2) and (3)

$$J_{k+1}=\sigma^2\left(J_k-\delta\frac{J_k a(J_k a)^T}{a^T J_k a}\right) \tag{12}$$

$$Q_{k+1}=\sigma\left(I-\xi\frac{Q_k a(Q_k a)^T}{\|Q_k a\|^2}\right) Q_k\,. \tag{13}$$

where

$$\delta=1-\left(\frac{\omega}{\sigma}\right)^2, \qquad \xi=1-\frac{\omega}{\sigma}$$

To complete the updating process we have to specify the quantities σ, ω and γ. Let

$$a^T x \leq \beta \tag{14}$$

be an inequality of *(P)* violated by x_k and the algebraic distance from x_k to the hyperplane $a^T x=\beta$ be equal to α, i.e.

$$\alpha = \frac{a^T x_k-\beta}{\|Q_k a\|} = \frac{a^T x_k-\beta}{\sqrt{a^T J_k a}} \tag{15}$$

If we choose (14) as a cut, then the boundary of the ellipsoid of minimal volume E_{k+1} must contain point A (see Fig.2) and therefore by (7)

$$\frac{1}{\omega^2}\left(-1+\gamma\right)^2=1, \tag{16}$$

and all points like *B*, i.e., points of the form $\left(-\alpha, \pm\frac{1}{n-1}\sqrt{1-\alpha^2}, \ldots, \pm\frac{1}{n-1}\sqrt{1-\alpha^2}\right)$, so by (7).

$$\frac{1}{\omega^2}(-\alpha+\gamma)^2+\frac{1}{\sigma^2}(1-\alpha^2)=1 \tag{17}$$

From (16) and (17) we have

$$\omega=1-\gamma \quad \text{and} \quad \sigma=\sqrt{1+\alpha}\,\frac{1-\gamma}{\sqrt{1+\alpha-2\gamma}} \tag{18}$$

The volume of E_{k+1} is

$$Vol\ E_{k+1}=c\omega\sigma^{n-1}=c(1+\alpha)^{\frac{n-1}{2}}(1-\gamma)^{n}(1+\alpha-2\gamma)^{-\frac{n-1}{2}}=f(\gamma)$$

where c is some constant coefficient. The value of γ that mimimizes $f(\gamma)$ is found by setting the derivative of $f(\gamma)$ to zero

$$\gamma=\frac{1+n\alpha}{1+n} \tag{19}$$

Substituting (19) in (18) we obtain

$$\omega=\frac{n}{n+1}(1-\alpha) \quad \text{and} \quad \sigma=\sqrt{\frac{n^2}{n^2-1}(1-\alpha^2)}. \tag{20}$$

As E_k and E_{k+1} are obtained from E'_k and E'_{k+1} by the same linear transformation and the volume of $E'_k=1$, then

$$r_k(\alpha)=\frac{Vol\ E_{k+1}}{Vol\ E_k}=\left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}}(1-\alpha^2)^{\frac{n-1}{2}}\frac{n}{n+1}(1-\alpha)<1 \tag{21}$$

Therefore we prove

Theorem 2.

For a given ellipsoid $E_k=(J_k,x_k)$, $k=0,1,\ldots$ the new ellipsoid $E_{k+1}=(J_{k+1},x_{k+1})$ of the smallest volume, which satisfies Inclusion and Convergence Principles is given by (11) and (12) where the coefficients γ,ω and σ are given by (19) and (20).

It is interesting to note that Theorem 2 states more general result as for $-\frac{1}{n}<\alpha\leq 1$, $r(\alpha)<1$ and also for $-1<\alpha<-\frac{1}{n}$ we have $r(\alpha)<1$, but then the smallest ellipsoid containing $H=\{x\,|\,a^Tx\leq\beta\}\cap E_k$ is E_k, so then the Convergence Principle is violated. For $\alpha=-\frac{1}{n}$ we have by (21) $Vol\ E_{k+1}=Vol\ E_k$. Khachiyan

in [10] constructed E_{k+1} for $\alpha=0$ while from (21) one can see that to have $r_k(\alpha)$ as small as possible and therefore at least locally the highest speed of convergence it is much better to choose

$$\alpha = \max_i d_i(x_k) \qquad (22)$$

and then we always have $\alpha>0$. The rule (22) has proven to be useful in computational practice and the ellipsoids obtained under such a rule are called deep-cut ellipsoids. Such a modification was proposed by many authors e.g. [15] and [19]. The other modifications we will discuss in the next section.

3. MODIFICATIONS

The volume of the new ellipsoid E_{k+1} can be made smaller than the volume of the deep-cut ellipsoid described in previous section if we consider so called "range ellipsoid" shown in Fig.3.

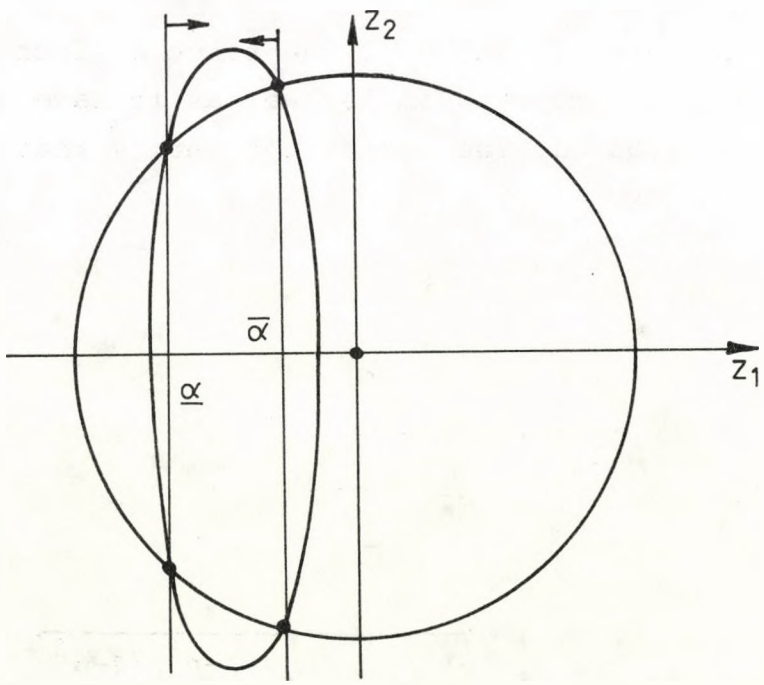

Fig.3.

If in *(P)* we have an equality

$$a^T x = \beta, \tag{23}$$

then one of possible way to find a numerical solution to *(P)* is to relax (23) to a range inequality

$$\underline{\beta} \leq a^T x \leq \overline{\beta} \tag{24}$$

where $\underline{\beta} = \beta - \varepsilon$, $\overline{\beta} = \beta + \varepsilon$ and $\varepsilon > 0$.
The algebraic distances from x_k to the bounding hyperplanes of (24) are

$$\overline{\alpha} = \frac{a^T x_k - \overline{\beta}}{\sqrt{a^T J_k a}} \quad \text{and} \quad \underline{\alpha} = \frac{a^T x_k - \underline{\beta}}{\sqrt{a^T J_k a}}$$

From Theorems 2 and 1 we have that $-\frac{1}{n} < \overline{\alpha} \leq 1$ and $-1 < \underline{\alpha} < \frac{1}{n}$ and also we can require that if $\underline{\alpha} \leq -1$ then we can arbitrarily set $\underline{\alpha} = -1$. For simplicity of notation we put $\alpha = \overline{\alpha}$ and $\eta = -\underline{\alpha}$. Then we may assume $0 < \alpha < \eta \leq 1$.

The parameters γ, ω and σ of the range ellipsoid E_{k+1} can be derived in the complete similar way as it have been done for the deep-cut ellipsoid. The reader can verify that for the range ellipsoid

$$\gamma = \alpha + \omega\sqrt{1 - \frac{1-\alpha^2}{\sigma^2}} \tag{25}$$

where

$$\omega = (\eta - \alpha) / \left(\sqrt{1 - \frac{1-\eta^2}{\sigma^2}} + \sqrt{1 - \frac{1-\alpha^2}{\sigma^2}}\right) \tag{26}$$

and

$$\sigma^2 = \frac{n^2}{n^2 - 1}\left\{1 - \frac{\eta^2 + \alpha^2}{2} + \sqrt{\left(\frac{\eta^2 - \alpha^2}{2}\right)^2 + \frac{(1-\eta^2)(1-\alpha^2)}{n^2}}\right. \tag{27}$$

For $\eta=1$ we obtain (19) and (20). Konig and Pallaschke in [12] describe the other method for computing η under the assumption that $F(P)$ is bounded.

The next modification of the ellipsoid algorithm consists in taking some nonnegative combination of violated inequalities which is called a surrogate cut. Examples show that in some cases a surrogate cut is deeper than a cut obtained by (22). For instance, (1) and (2) are violated inequalities at x_o and if the surrogate cut is a dotted line in Fig. 1, then the centre of the next ellipsoid is a solution of (P) and the algorithm makes one iteration less.

In general let $\bar{A}$ be a matrix whose columns are some subset of the a_i's for which $0<\alpha_i\leq 1$ and let $\bar{b}$ be the corresponding vector of β_i's. Then the inequality

$$u^T\bar{A}^T x \leq u^T\bar{b} \tag{28}$$

holds for any $x \subseteq F(P)$ and $u\geq 0$.
If we denote $a=\bar{A}u$ and $\beta=u^T\bar{b}^T$, then we want to choose $u\geq 0$ so that

$$\alpha = \frac{a^T x_k-\beta}{\sqrt{a^T J_k a}} = \frac{u^T(\bar{A}^T x_k-\bar{b})}{\sqrt{u^T\bar{A}^T J_k \bar{A}u}}$$

is maximized. This is a quadratic programming problem. If A has a full rank, then α is maximized over all u not just nonnegative u's by

$$u^* = (\bar{A}^T J_k \bar{A})^{-1}(\bar{A}^T x_k-\bar{b}). \tag{29}$$

see Goldfarb and Todd [9].

If $u^*\geq 0$ then for $u=u^*$ (28) is deepest surrogate cut. Solving a quadratic programming or cumputing u^* by (29) may be computationally too expensive. Therefore in practice one satisfies with nearly deepest surrogate cut by choosing $u=1$, or $u_i=a_i^T x_k-\beta_i>0$.

In [3] surrogate cuts are generated from all inequalities by an iterative procedure which combines two inequalities at a time.

4. POLYNOMIALITY

Having all data in *(P)* integer we can compute so called length L of the input, i.e., the number of symbols +, -, 0, 1 required to encode the data of *(P)*

$$L = \sum_{\substack{1\le i\le m\\ 1\le j\le n\\ a_{ij}\neq 0}} \lfloor \log|a_{ij}| \rfloor + \sum_{\substack{1\le i\le m\\ \beta_i\neq 0}} \lfloor \log|\beta_i| \rfloor + \lfloor \log m \rfloor + \lfloor \log n \rfloor +$$

$$+ 2mn + 2m + 4 \qquad (30)$$

where $\lfloor x \rfloor$ means the greatest integer less or equal x and all logarithms are on the base 2. In (30) we assume that all integers m, n and the entries of A and b are encoded in some specified order, separated by sign bits. In this section we show that the algorithm requires at most $6n(n+1)L$ iteration to find a solution to *(P)* or to show that $F(P)=\emptyset$

We will assume that exact arithmetic is used. In [10] Khachiyan has shown that since $k\le 6n(n+1)L$ then finite - precision arithmetics with $23L$ bits before the point and $38nL$ after suffice. We make this assumption for simplicity of presentation. We show that the following inclusions are true

$$S(a^*,\tau) \subseteq F(P) \subseteq S(0,\rho), \qquad (31)$$

where $S(a^*,\tau)$ denotes the ball of radius τ centered at a^*.

From (21) we have for $k=0,1,\ldots$

$$\frac{Vol\ E_{k+1}}{Vol\ E_k} < \frac{n}{n+1}\left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} < e^{-1/2(n+1)} \qquad (32)$$

Suppose $k>2n(n+1)\log(\rho/\tau)$. Then, assuming the algorithm continues this far, the volume of E_k will have shrank to less than $(\tau/\rho)^n$ time the volume of $S(0,\rho)$ and therefore cannot contain the ball of radius τ. But from (31) we have that the sequence of ellipsoids satisfies

$$S(a^*,\tau) \subseteq F(P) \subseteq E_k \qquad \text{for any } k$$

This contradiction proves that the algorithm must terminate with $x_k \in F(P)$ for some $k \leq 2n(n+1)\log(\rho/\tau)$. Hence if ρ and τ are regarded as the part of the input of (P), or $\log(\rho/\tau)$ is polynomial in L, the number of iterations required is polynomial.

First we whow that if $F(P)\neq\emptyset$, then

$$F(P) \cap S(0,2^L)\neq\emptyset$$

and therefore $\|x\| \leq 2^L$, where x is a solution of (P). In matrix notations we can write (P) as $A^T x \leq b$ and then by Cramer's rule

$$|x_j| = \left|\frac{\det A_i}{\det A}\right| \leq \frac{|\det A_i|}{1}$$

By Hadamard's inequality $|\det A_i|$ is less or equal the product of norms of all columns of A_i and next by (30).

$$|x_j| \leq \frac{1}{mn}\, 2^L \leq \frac{1}{n}\, 2^L$$

So $\|x\| \leq 2^L$. Clearly, however, $F(P)$ need not contain a ball of any positive radius. Thus Khachiyan [10] perturbed (P) to obtain a system

$$(P') \qquad 2^L a_i^T x < 2^L \beta_i + 1, \qquad i=1,\ldots,m$$

and then proved that $F(P)\neq\emptyset$ if and only if $F(P')\neq\emptyset$. Obviously $F(P) \subseteq F(P')$. He also showed in [10] how a solution to (P) can be obtained in polynomial time from a solution to (P').

The reason for considering (P') is that if (P) has a feasible solution, say $\hat{x}$, then $S(\hat{x}, 1/\max\|2^L a_i\|)$ is contained in $F(P')$. Since $\|a_i\| \leq 2^L$, we obtain

$$S(\hat{x}, 2^{-2L}) \subseteq F(P')$$

if $\hat{x} \in F(P)$. Therefore if $F(P) \neq \emptyset$, then

$$S(\hat{x}, 2^{-2L}) \subseteq F(P') \cap S(0, 2^L) \subseteq S(0, 2^L) \tag{33}$$

To write (P') we need at most $(m(n+1)+1)L$ bits and hence the number of bits is polynomial in L. If we apply the algorithm to (P') then by (33) $\tau = 2^{-2L}$ and $\rho = 2^L$.
Since

$$k \leq 2n(n+1)\log\rho/\tau = 6n(n+1)L \tag{34}$$

therefore if the algorithm fails to terminate after $6n(n+1)L$ iterations, we can conclude that $F(P) = \emptyset$. We state this result in the from of

Theorem 3.

The ellipsoid algorithm requires at most $6n(n+1)L$ iterations to find a solution to (P) or to determine that $F(P) = \emptyset$.

5. SOLVING LP PROBLEMS

In this section we describe three methods of transforming a given linear programming problem

$$(LP) \quad v(LP) = \max\{c^T x \mid A^T x \leq b, \quad x \geq 0\}$$

into a system or systems of linear inequalities. Next we show how the radius ρ of the initial ellipsoid can be estimated for LP problems since, as a rule, for LP problems taken from prac-

tice we can provide much better estimation of ρ than 2^L given in the previous section. We will assume that A is an $\bar{m}x\bar{n}$ matrix.

5.1 Primal - dual Formulation

The dual problem to *(LP)* is

$$(D) \quad v(D)=\min\{b^T y \mid Ay \geq c,\ y \geq 0\}$$

By duality theorem solving *(LP)* is equivalent to solving a system *(P)* of $m=2(\bar{m}+\bar{n})+1$ inequalities in R^n, where $n=\bar{m}+\bar{n}$

$$\begin{aligned}(P) \quad & A^T x \leq b \\ & -x \leq 0 \\ & -Ay \leq -c \\ & -y \leq 0 \\ & -c^T x + b^T y \leq 0\end{aligned}$$

So solving *(P)* we have an optimal primal and dual solutions.

From a practical viewpoint, there are several disadvantages to this approach. First, the high dimensionality slows convergence. Second, as the all solutions to *(P)* lie in the hyperplane $c^T x=b^T y$, the volume of the solution set in R^n equals to zero and therefore some perturbation of the right hand sides in *(P)* is necessary. Third, if $F(P)=\emptyset$, then it is not clear whether *(LP)* is infeasible or unbounded.

Some of these difficulties can be mitigated if we note that, except for its final constraint, *(P)* separates into two subsystems. Thus if no cut is based on the last constraint, the matrix J_k defining the ellipsoid E_k separates into two blocks and only one block is updated in each iteration. It seems reasonable to base cuts only on primal constraints until the primal feasibility is reached, and then only on dual constraint until the dual feasibility is reached.

The two remaining approaches are based upon systems of linear inequalities of the form

$$(P)\quad \begin{aligned} A^T x &\le b \\ -x &\le 0 \\ -c^T x &\le -\zeta \end{aligned} \qquad (35)$$

for various values of parameter ζ. These methods do not produce optimal dual solutions.

5.2 Bisection Method

In this method first we find by the ellipsoid algorithm a primal feasible solution, i.e., solution to the system $A^T x \le b$, $-x \le 0$. If x_k is such a solution, then $\underline{\zeta}=c^T x_k$ is a lower bound on $v(PL)$. If there is no solution we terminate $F(PL)=\emptyset$. If $F(PL)$ is bounded, and E_o was constructed in such a way that $F(PL) \subseteq E_o$ (see the end of this section), the upper bound on $v(PL)$ can be computed as

$$\bar{\zeta} = c^T x_k + \sqrt{c^T J_k c}$$

Unboundedness of $F(PL)$ can be detected by solving the system of dual constraints $-Ay \le -c$, $-y \le 0$. Having $\underline{\zeta}$ and $\bar{\zeta}$ we put $\zeta=1/2(\underline{\zeta}+\bar{\zeta})$ in (35) and solve it. There are two possibilities:

i) The system (35) is feasible, i.e., x_{k+p} solves (35), $p \ge 1$, then we put $\underline{\zeta}=c^T x_{k+p}$, compute new ζ and again solve (35) by the ellipsoid method.

ii) If the system (35) is inconsistent then we backtrack to ellipsoid E_k and put $\bar{\zeta}=\zeta$ compute new ζ and solve again (35) for this new value of ζ. Computations are terminated when $\underline{\zeta}$ and $\bar{\zeta}$ are sufficiently close.

It is easy to see that the backtrackings is the main disadvantage of such an approach. Avoiding such backtrackings leads to the final method that we shall consider.

5.3 Sliding Objective Function Method

As before we assume without loss of generality that $x_k \in F(PL)$ and that $F(PL)$ is bounded. Then $\zeta = c^T x_k$ in (35) and

$$-c^T x \leq -c^T x_k$$

is a valid cut with $\alpha = 0$ for the construction of E_{k+1}. A possible improvement of this method consists in finding as large as possible $\delta > 0$ such that $x = x_k + \delta s$ is a feasible point of $F(PL)$, where s is an ascent direction e.g. $s = c$ or $s = J_k c$. So we have

$$a_i^T x \leq \beta_i \qquad i = 1, \ldots, m$$

and

$$x = x_k + \delta s \geq 0$$

A solution to the above system is given by

$$\bar{\delta} = \min_i \frac{\beta_i - a_i^T x_k}{a_i^T s}, \quad \bar{\delta} \geq 0, \quad x_k - \bar{\delta} s \geq 0$$

This method is probably the most efficient for practical implementation. It always considers feasible systems (35) and never backtracks. Computations can be terminated when for given $\varepsilon > 0$

$$c^T \left(x_k + \frac{J_k c}{\sqrt{c^T J_k c}} \right) - c^T (x_k + \delta s) = \sqrt{c^T J_k c} - \delta c^T s \leq \varepsilon$$

5.4 An Estimation of ρ

In many *LP* problems taken from practice the bounds $0 \leq x \leq d$ are given or can be easily derived from $Ax \leq b$. Therefore we can take $x_o = \frac{1}{2} d$ as a good starting point for the ellipsoid algorithm

and $\rho = d\sqrt{n}/2$. Another possibility is if $a_{ij} \geq 0$ for all j then from $a_i^T x \leq \beta_i$ we have

$$\sum_{j=1}^{n} a_{ij}^2 x_j^2 \leq \beta_i^2$$

and

$$\rho \leq \sqrt{\sum_{j=1}^{n} (\beta_i / a_{ij})^2}$$

In many practical *LP* problems we know that $F(PL) \neq \emptyset$ and therefore $F(P) \neq \emptyset$. Then we can choose ρ in such a way that the biggest algebraic distance $\alpha < 1$, e.g. $\alpha = \frac{1}{2}$. In this approach if for some iteration number k we obtain $\alpha > 1$ then we cannot say according to Theorem 1 that $F(P) = \emptyset$. This only means that our choice of ρ is not good for all iterations and we have to increase E_k to have again $\alpha \leq 1$. Computational experiments show that such an approach is probably the most efficient (see Waluk and Walukiewicz [18]).

6. COMPUTATIONAL RESULTS

The main question for computational experiments with the ellipsoid algorithm can be formulated in the form: Could the ellipsoid algorithm compete with the simplex method? These experiments can be divided into two groups: experiments with "typical" large linear programming problems taken from practice (e.g. energy models) and experiments with pathological in some sense linear programming problems.

Probably the most representative work for the first group is the paper [7] by Gill et al. They consider some known linear programming problems stated in the form

$$(P) \qquad c^T x \leq t+\delta t$$

$$b_l-\delta b_l \leq Ax \leq b_u+\delta b_u$$

$$l-\delta l \leq x \leq u+\delta u$$

where all perturbations are positive vector and are rather substantial (about 5%). For all such problems $F(P)\neq\emptyset$. They point out that a vital difference between the simplex method and the ellipsoid algorithm consists in the workspace required. Namely in large *LP* problems a matrix A is sparse and such a sparsity can be kept in each iteration without any loss of accuracy of the simplex method. Unfortunately, in the ellipsoid method, even when J_o is a diagonal matrix, J_k or Q_k becomes rapidly dense.

So far this drawback can be ruled out only partially. For simplicity of notation we write $\bar{J}_k=\sigma^2 J_k$ and $Q_k=\sigma Q_k$. Then by (12) we can write $\bar{J}_{k+1}$ in so called summation form

$$\bar{J}_{k+1}=\bar{J}_o-\sigma_1 p_1 p_1^T - \sigma_2 p_2 p_2^T - \ldots - \sigma_k p_k p_k^T \qquad (36)$$

where $p_k=Q_k^T q_k$, $\quad k=1,2,\ldots$

Similarly $\bar{Q}_{k+1}$ can be expressed in so called product form (see (13)).

$$\bar{Q}_{k+1} = (I-\zeta_1 q_1 q_1^T)(I-\zeta_2 q_2 q_2^T)\ldots(I-\zeta_k q_k q_k^T)\bar{Q}_o \qquad (37)$$

So in each iteration only one new η-vector and one scalar are updated and this approach is practical only if $k<n$. Usually the number of iterations required to solve the system;

$$(P) \quad a_i^T x \leq \beta_i+\varepsilon, \qquad i=1,\ldots,m, \qquad (38)$$

where ε is a given tolerance, is substantialy greater than n. Therefore using (36) or (37) one has to restart procedure after every l iterations (e.g. in [7], $l=20$). The restart con-

sists in substituting E_l by the ball B_l with the center x_l containing E_l. Such a substitution violates Convergence Principle since, in general, $Vol\ B_l > Vol\ E_{l-1}$. One of possible approaches is to shrink B_l in such a way that Convergence Principle is satisfied. But then Inclusion Principle is violated and the ellipsoid algorithm is not an exact method in such a case. Also so called cycling strategy [7], i.e., a strategy when only the last l vectors p_k or q_k are kept in memory cannot, in general, guarantee the ellipsoid algorithm to be exact.

Bednarczuk in [1] constructed and solved some number of ill-conditioned linear programming problems stated in the form $v(LP)=max\{c^T x \mid Ax \leq b,\ x \geq 0\}$ where A is an $n \times n$ matrix and

$$c^T = \left(\frac{2}{1+1} + \sum_{i=2}^{n} \frac{1}{i+1},\ \frac{2}{2+1} + \sum_{i=2}^{n} \frac{1}{i+2},\ \ldots,\ \frac{2}{n+1} + \sum_{i=2}^{n} \frac{1}{i+n} \right)$$

$$A = \begin{bmatrix} 1/2 & 1/3 & \ldots & 1/(n+1) \\ 1/3 & 1/4 & \ldots & 1/(n+2) \\ \ldots & \ldots & \ldots & \ldots \\ 1/(n+1) & 1/(n+2) & \ldots & 1/(n+n) \end{bmatrix}$$

$$b^T = \left(\sum_{i=1}^{n} \frac{1}{i+1},\ \ldots,\ \sum_{i=1}^{n} \frac{1}{i+n} \right).$$

The optimal solution to this problem is $(x^*)^T=(1,1,\ldots,1)$ and the optimal solution to its dual is $(y^*)^T=(2,1,\ldots,1)$.

It is easy to see that the constrained hyperplanes are almost parallel and, as a result one may expect numerical troubles when solving such problems by the simplex method. Even for $n=5$ the differences between the theoretical optimal solutions and ones obtained by the simplex method (MPSX on the IBM 360/50 computer) can be substantial as it is shown in Table 1.

Table 1.

	Primal solution		*Dual solution*		
x^*	*Simplex method*	*Ellipsoid method*	y^*	*Simplex method*	*Ellipsoid method*
1	0.9732	1.0000	2	1.9731	1.9983
1	1.3002	0.9905	1	1.3005	1.0130
1	0.0000	1.0517	1	0.0000	0.9678
1	2.2850	0.9154	1	2.2842	1.0306
1	0.4381	1.0437	1	0.4387	0.9901

It should be noted that all computations in both methods have been done with the same precision. If we arbitrarily allow 10% error than the simplex method gives the correct answer for one primal and dual variable, while the ellipsoid method gives correct values for all variables. It is interesting to note that for such pathological examples increasing the accuracy of computations in the simplex method does not reduce the above errors [1]. More results for such ill-conditioned problems are given in [18].

Clausen in [5] presented an example for exponentiality of the simplex method which is some modification of the Klee- - Minty example [11]. This example can be written in the form

$v(PL)=max\{c^T x \mid Ax \leq b,\ x \geq 0\}$ where

$$c^T=(h^{n-1},\ h^{n-2}\ \frac{1}{p}\ ,\ldots,\ \frac{1}{p^{n-1}})$$

$$A = \begin{bmatrix} 1 & & & & & \\ 2h & p & 1 & & & 0 \\ 2h^2 & p^2 & 2hp & 1 & \ldots & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 2h^{n-1} & p^{n-1} & \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

$$b^T = (1, p, p^2, \ldots, p^{n-1})$$

In this formulation h and p are given parameters; in our experiments we use $h=0.25$ and $p=5$. The optimal solution $(x^*)^T = (0,0,\ldots,0,p^{n-1})$ is obtained by the simplex method after 2^n iterations.

In our experiments we use the sliding objective function formulation of the above problem. Results presented in Table 2 (see also [18]) show that the ellipsoid algorithm finds the optimal solution within the smaller number of iterations and in the shorter computer time. All these experiments have been done on MERA 400 minicomputer

Table 2.

	Simplex method		*Ellipsoid algorithm*	
n	*Iteration number*	*CPU time in sec.*	*Iteration number*	*CPU time in sec.*
6	64	7	54	7
7	128	13	87	12
8	256	23	129	17
9	512	44	165	22
10	632	70	218	35
12	2528	240	243	55

In this experiment the tolerance $\varepsilon=10^{-5}$ see(38). For $n=10$ and $n=12$ the iteration number is not equal 2^n because of the cumulation of errors.

Charnes et al. in [4] show that the dual to Klee-Minty's problem can be solved in polynomial number of steps. They also show that Clausen's modification is free of this defect, but again it can be solved in polynomial number of iterations by so called interval programming [4]. But as they pointed out in [4], the above statements by no means invalidate the idea of Klee - Minty's example, namely, that the tilted unit hypercube has 2^n extreme points and if one must start "at the bottom" then the primal simplex algorithm can take $O(2^n)$ iterations to find the optimal solution.

7. CONCLUDING REMARKS

Now, in a summary, we give a general description of the ellipsoid algorithm for solving (38). A given *(LP)* problem can be transformed to (38) by one of methods described in Section 5.

1. (Initialize) Set $k=0$ and choose the initial ellipsoid as described is Section 5.4.

2. (Terminate?) If x_k satisfies (38) accept it as an optimal solution within a required tolerance ε and terminate

3. (Choose a cut). According to (22) choose a cut or construct a surrogate cut (Section 3) .

4. (Update) Compute x_{k+1}, J_{k+1} *(or* Q_{k+1}*)* by (11) (36) or (37). Set $k=k+1$ and return to Step 2.

When comparing the ellipsoid algorithm with the simplex method we have to keep in mind that we are comparing the general method with the specific one. In fact, the ellipsoid algorithm is an example of subgradient method see (Shor [17]) which generate a sequence of points $x_0, x_1, \ldots$ convergent to a solution of *(P)*. The only one requirement in this method is that *F(P)* should be convex, therefore the relations describing *F(P)* may be nonlinear and nondifferentiable. So solving linear programming problem *(LP)* in R^n by the ellipsoid algorithm we compute x_{k+1} by (11) ignoring the fact that the optimal solution to *(LP)* is, in general, in a vertex of *F(LP)*. This information is essential in the simplex method.

This is the main reason why, in general, the simplex method does much better than the ellipsoid algorithm. But it is at the same time the main reason why the simplex algorithm is inefficient in comparison with the ellipsoid algorithm in the case of ill - conditioned problems, problems with exponential number of extreme points [18] and in the case of degenerate problems [16].

Basing on Inclusion and Convergence Principles one can construct a sequence of balls in R^n in a completely similar way as it was done in Section 1 for ellipsoids case. Then instead of (11) we will have

$$x_{k+1} = x_k - \lambda \frac{a^T x_k - \beta}{a^T a} a \tag{39}$$

and this relation is valid for $0<\lambda<2$. For $\lambda=2$, (39) gives Motzkin and Schoenberg's relaxation method [14] for solving system of linear inequalities. It is easy to see that the convergence of the ball method is slower then the ellipsoid one and Goffin [8] demonstrates that an exponential in the data number of iterations may be required to solve (P) (see also Bland et al [2]).

For both ellipsoid and ball method we have in general

$$E_k \cap F(P) \subset E_{k+1} \quad but \quad E_k \cap F(P) \neq E_{k+1} \tag{40}$$

Levin in [13] proposed a method of minimization of a convex function f over a bounded polyhedron $P_o \subseteq R^n$ in which we have equality in (40). The method produces a sequence of points $\{x_k\}$ and polytopes $\{P_k\}$ by choosing x_k as the center of gravity of P_k and

$$P_{k+1} = \{x \in P_k \mid g_k^T x \leq g_k^T x_k\}$$

where g_k is a subgradient of f at x_k. Since f is convex, P_{k+1} contains all points x of P_k for which $f(x) \leq f(x_k)$. As x_k is the gravity center of P_k then $Vol\ P_{k+1} \leq (1-\frac{1}{e}) Vol\ P_k$, so Convergence Principle is satisfied and set of candidates for the minimum of f is exactly equal P_{k+1}.

The main drawback of Levin's method is a difficulty of calculating the gravity center of a polytope in R^n when $n \geq 3$. So the ellipsoid method can be considered as a certain step in the development of subgradient methods.

R E F E R E N C E S

[1] E. Bednarczuk, "O wynikach testów zadań programowania liniowego w oparciu o pakiety programów maszyn IBM i ROBOTRON", Prace IBS PAN, z. 2 (1977)(in Polish).

[2] R.G. Bland, D. Goldfarb and M.J. Todd "The Ellipsoid Method: A Survey", Technical Report No. 476, School of Operations Research and Industrial Engineering, Cornel University, Ithaca, N.Y. (1980).

[3] E. Boros and A. Sebő, "Approximative Solutions to Linear Programming Problems with the Modification of Khachiyan's Algorithm", Report MO/18, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest (1980).

[4] A. Charnes, W.W. Cooper, S. Duffuaa, and M.Kress, "Complexity and Computability of Solutions to Linear Programming Systems", International J. of Computer and Information Sciences 9 (1980) 483-506.

[5] J. Clausen, "A Tutorial Note on the Complexity of the Simplex - algorithm" in J. Krarup and S. Walukiewicz, Eds. Proceedings of DAPS - 79 Institute of Datalogy, University of Copenhagen (1980) 51-65.

[6] D.Gale, "How to solve linear inequalities" American Mathematicl Monthly 76 (1969) 589-599.

[7] P.E.Gill, W.Murray, M.A.Saunders, and M.H. Wright, "A Numerical Investigation of Ellipsoid Algorithm for Large-Scale Linear Programming", in Large-Scale Linear Programming, G.B. Dantzing, M.A.H. Dempster, and M.J.Kallio, Eds. IIASA Collaborative Proceedings Series CP-81-S1, Laxenburg Austria (1981) 487-509.

[8] J.L. Goffin, "On the Non-polynomiality of the Relaxation Method for Systems of Inequalities", Faculty of Management, Mc Gill University, Montreal (1979).

[9] D.Goldfarb and M.J. Todd, "Modifications and Implementation of the Shor-Khachiyan Algorithm for Linear Programming", Technical Report No 446, Cornell University, Ithaca, N.Y. (1980).

[10] L.G. Khachiyan, "A Polynomial Algorithm in Linear Programming", Doklady Akademii Nauk SSSR 244:5(1979), 1093-1096, translated in Soviet Mathematics Doklady 20:1 (1979) 191-194.

[11] V.Klee and G.J. Minty, "How Good is the Simplex Algorithm", in O. Shisha Ed. Inequalities III, Academic Press, New York, (1972) 159-175.

[12] M.Konig and D. Pallaschke, "On Khachiyan's Algorithm and Minimal Ellipsoids", Numer.Math. 36 (1981) 211-223.

[13] A.Yu. Levin, "An Algorithm for Minimization of Convex Functions", Doklady Akademii Nauk SSSR 160 (1965) 1244-1247 (in Russian).

[14] T. Motzkin and I.J.Schoenberg, "The Relaxation Method for Linear Inequalities", Canadian J. of Mathematics 6 (1954) 363-404.

[15] W.Murray, "Ellipsoidal Algorithm for Linear Programming", Working Paper 79-1, Systems Optimization Laboratory, Department of Operations Research, Stanford University (1979).

[16] D. Pallaschke, private communication.

[17] N.Z.Shor, "New Development Trends in Nondifferentiable Optimization", Kibernetika 13(1)(1977) 87-91 translated in Cybernetics 13(6)(1977) 881-886.

[18] B.Waluk and S. Walukiewicz, "Eksperyment obliczeniowy z algorytmem elipsoidalnym". Prace IBS PAN (1981)(in Polish).

[19] S.Walukiewicz, "Ellipsoidal Algorithm for Linear Programming", Technical Report LITH-MART-R-11, Department of Mathematics, Linkoping Institute of Technology (1980).

[20] P. Wolfe, "A Bibliography for the Ellipsoid Algorithm", IBM Research Center Report, (July 1980).

A TANULMÁNYSOROZATBAN 1982-BEN MEGJELENTEK

130/1982 Barabás Miklós - Tőkés Szabolcs: A lézer printer képalkotás hibái és optikai korrekciójuk

131/1982 RG-II/KNVVT "Szisztemü upravlenija bazani dannüh i informacionnüe szisztemü" Szbornik naucsno-iszszledovatel'szkih rabot rabocsej gruppü RG-II KNVVT, Bp. 1979. T o m I.

132/1982 RG-II/KNVVT T o m II.

133/1982 RG-II KNVVT T o m III.

134/1982 Knuth Előd - Rónyai Lajos: Az SDLA/SLT adatbázis lekérdező nyelv alapjai /orosz nyelven/

135/1982 Néhány feladat a tervezés-automatizálás területéről. Őrmény-magyar közös cikkgyüjtemény

136/1982 Somló János: Forgácsoló megmunkálások folyamatainak optimálási és irányitási problémái

137/1982 KGST I-15.1. Szakbizottság 1979. és 80. évi előadásai

138/1982 Kovács László: Számitógép-hálózati protokollok formális specifikálása és verifikálása

139/1982 Operációs rendszerek elmélete 7. visegrádi téli iskola

A TANULMÁNYSOROZATBAN 1983-BAN MEGJELENTEK

140/1983 Operation Research Software Descriptions (Vol.1.) Szerkesztette: Prékopa András és Kéri Gerzson

141/1983 Ngo The Khanh: Prefix-mentes nyelvek és egyszerü determinisztikus gépek

142/1983 Pikler Gyula: Dialógussal vezérelt interaktiv gépészeti CAD rendszerek elméleti és gyakorlati megfogalmazása

143/1983 Márkusz Zsuzsanna: Modellelméleti és univerzális algebrai eszközök a természetes és formális nyelvek szemantikaelméletében

144/1983 Publikációk '81 /Szerkesztette: Petróczy Judit/

145/1983 Telcs András: Belsõ állapotu bolyongások

146/1983: Varga Gyula: Numerical Methods for Computation of the Generalized Inverse of Rectangular Matrices

147/1983 Proceedings of the joint Bulgarian-Hungarian workshop on "Mathematical Cybernetics and data Processing" /Szerkesztette: Uhrin Béla/

148/1983 Sebestyén Béla: Fejezetek a részecskefizikai elektronikus kisérleteinek adatgyüjtõ, -feldolgozó rendszerei köréből

149/1983 L. Keviczky, J. Hethéssy: A general approach for deterministic adaptive regulators based on explicit identification

150/1983 IFIP TC.2 WORKING CONFERENCE "System Description Methodologies" May 22-27. 1983. Kecskemét. /Szerkesztette: Knuth Elõd/