CONTINUOUS DYNAMIC LINEAR PROGRAMMING

*Viljem RUPNIK**

1. INTRODUCTION

In recent times we can see that in management science a very strong emphasis is being given to control theory and systems analysis. Thus, many classical optimization methods must be adapted to serve in problem areas concerned with dynamic systems, their stability and similar questions.

In this paper we explore the possibility of using linear programming techniques in control theory. A dynamic version of the usual linear programming formulation has been proposed, and thus we arrive at a class of continuous dynamic linear programming problems which appears to be worthwile considering in the field of management science.

2. STATEMENT OF THE PROBLEM

In the theory of linear programming we face the following problem:

$$\begin{aligned} &\text{opt } (c, x) \\ &A x = b \qquad (1) \\ &x \geqslant o \end{aligned}$$

where: $c = (c_1, \ldots c_n)$ is an n-dimensional vector, defined on some real-valued domain of an n-dimensional vector space; $x = (x_1, \ldots x_n)'$ is defined as an n-dimensional column vector, belonging to an n-dimensional real-valued vector space; $b = (b_1, \ldots b_m)'$ is an m-dimensional vector, being an element of an n-dimensional realvalued vector space; and finally, A being an m x n matrix of real constant coefficients.

Problem (1) is called a linear programming problem in standard form. A solution to problem (1) — if it exists — appears as a vector x

*) Professor of Mathematics, University of Ljubljana.

that satisfies non-homogeneous systems of linear equations in (1), having all components nonnegative.

Starting from (1) we define a new problem by introducing the following functions. Let us define: time variable t, which is continuously defined on the closed interval $[o, T]$; $c_j(t)$, $j = 1, \ldots n$ as real-valued scalar functions, defined for each $t \in [o, T]$ and being at least piecewise continuous and uniform, but arbitrary otherwise; $x_j(t)$, $j = 1, \ldots n$ as real-valued scalar functions, defined on $[o, T]$ as uniform, nonnegative and continuous or at least piecewise continuous; $b_i(t)$, $i = 1, \ldots m$, as real-valued scalar functions defined on $[o, T]$ as uniform, nonnegative and continuous or at least piecewise continuous and piecewise differentiable.

A matrix A remains to be defined as in problem (1). We want to find a vector $x(t) = [x_1(t), \ldots x_n(t)]'$ that satisfies a system of equations $Ax(t) = b(t)$ and brings the optimum value to $[c(t), x(t)]$ for every $t \in [o, T]$. Thus, we arrive at the following problem:

$$\left.\begin{array}{l} opt\,[c(t),\ \ x(t)] \\ Ax(t) = b(t) \\ x(t) \geqslant o \\ o \leqslant t \leqslant T \end{array}\right\} \qquad (2)$$

In the problem above, we allow both $c(t)$ and $b(t)$ to vary over time and for that reason we shall call it a continuous dynamic linear programming problem. We shall try to work out the computational algorithm for problem (2).

It is convenient to tackle this problem first by assuming:

$$c_j(t) = \text{const}_j\ , \quad \left\{\begin{array}{l} o \leqslant t \leqslant T \\ j = 1, \ldots n \end{array}\right. \qquad (3)$$

This will help us to analyze the fundamental structure of a class of continuous dynamic linear programming problems like (2). The objective function is linear in $x(t)$ and the constraints are also linear; we have therefore a kind of linear programming problem.

On the basis of assumption (3) we have the following subclass of problems:

$$\left.\begin{array}{l} opt\ [c,\ \ x(t)] \\ A\,x(t) = b(t) \\ x(t) \geqslant o \\ o \leqslant t \leqslant T \end{array}\right\} \qquad (4)$$

which can be called *a continuous b-dynamic linear programming problem*. The approach we shall use will show that parametric linear

programming problems form a subclass of problems (4), because we assume here a general nature of vector $b(t)$.

Similarly, by assuming:

$$b_i(t) = const_i,\ o \leqslant t \leqslant T,\ i = 1, \ldots m \tag{5}$$

we arrive at the following subclass of problems

$$\left.\begin{array}{c} opt\,[c(t),\ x(t)] \\ A\,x(t) = b\ (= const) \\ x(t) \geqslant o \\ o \leqslant t \leqslant T \end{array}\right\} \tag{6}$$

which can be called *a continuous c-dynamic linear programming problem*. We shall not discuss (6) at any greater length, because the relevant approach is a use of sensitivity analysis in linear programming problems, aimed at criterion coefficients. We shall rather show a solution procedure for (2) as a kind of synthesis of solution procedures for problems (4) and (6). Thus, we shall first discuss b-dynamic subproblems in detail. It will be clear how to deal with a *c*-dynamic subproblem, as well as how to solve the original problem (2).

At the end we shall show an application in management science.

3. STUDY OF A B-DYNAMIC SUBPROBLEM — A FEASIBILITY ALGORITHM

a) General Outline

Let us give an outline of the procedure we would like to establish. We shall try to solve problem (4) at $t = o$ by the simplex algorithm. When using the simplex algorithm, the choice of an admissible basis depends on coefficients a_{ij}, coefficients of objective function and the values of the right-band sides of the successive computational tableaus. When a vector which is going to be introduced into a new basis is determined by the smallest negative coefficient c_j of the objective function, let us say c_s, a vector we eliminate is indicated by that row index r for which the ratio between the components of the right-hand sides and coefficients a_{is} reaches minimum value. As the components of the right-hand sides change, i. e. when the bounding functions are changing with time t, so the index r will change, indicating the smallest ratio. We shall have to find that value of t, say $t = t'$, where index r moves from its present position to some other one. From t' on there will be the same basis again, used up to some other point $t = t''$, at which we have to switch to another basis.

In this way the whole interval $[o,\ T]$ will be partitioned into subintervals, where on each of them we shall have some basis improv-

ing the value of criterion function. We shall have a sequence of transformations for the second iteration covering the whole region $[o, T]$. At the same time we shall have to record the analytical expressions for the whole set of bounding functions, corresponding to the given interation. We shall, for the moment, assume they are nonnegative everywhere on $[o, T]$. Let us now start to work out the solution procedure.

b) The First Iteration

Let problem (4) have a canonical form so that, using Dantzig's simplex algorithm, the first feasible solution can easily be read off at $t = 0$. Let us denote by

$$b^0(t) = \begin{pmatrix} b_1^0(t) \\ \cdot \\ \cdot \\ \cdot \\ b_m^0(t) \end{pmatrix}$$

the initial bound $b(t)$ to emphasize its belongingness to the initial tableau which is the starting tableau for the first iteration. In general, let

$$b^{(k)}(t) = \begin{pmatrix} b_1^{(k)}(t) \\ \cdot \\ \cdot \\ \cdot \\ b_m^{(k)}(t) \end{pmatrix} \tag{7}$$

mean a bound belonging to k-th iteration at time t. A subinterval on on which expressions like (4) are defined is not greater than the initial subinterval. Let in general

$$A^{(k)} = \left(a_{ij}^{(k)}\right) \tag{8}$$

be a matrix of coefficients of the constraints in problem (4) obtained after the k-th iteration for a particular t on a segment of $[o, T]$. The initial tableau therefore is, for $t = 0$,

$$A^{(0)} x_{(0)}^{(0)} = b^{(0)}(0) \tag{9}$$

where $A^{(0)} = \left(a_{ij}^{(0)}\right)$ and

$$x^{(0)}(0) = \begin{pmatrix} x^r(0) \\ \cdot \\ \cdot \\ \cdot \\ x_n(0) \end{pmatrix}$$

Let $min\ c_j = c_s < o$ be determined and form the ratio $b_i^{(0)}(0)/a_{is}$, $i = 1, \ldots m$. The row index r of a vector which will be introduced into a basis is now determined by

$$0 < \frac{b_r^{(0)}(0)}{a_{rs}^{(0)}} = \min_i \frac{b_{(i)}^{(0)}(0)}{a_{is}^{(0)}} \geqslant 0$$

Using the element $a_{rs}^{(0)}$ as a pivot element we transform the initial tableau at $t = 0$ by a matrix

$$T_{rs} = \begin{vmatrix} 1, & 0, \ \ldots & -\frac{a_{is}^{(0)}}{a_{rs}^{(0)}}, & \ldots & 0 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & & \\ 0, & 0, \ \ldots & \frac{1}{a_{rs}^{(0)}}, & \ldots & 0 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & & \\ 0, & 0, \ \ldots & -\frac{a_{ms}^{(0)}}{a_{rs}^{(0)}}, & \ldots\ldots & 1 \end{vmatrix}$$

So, after the first interation, we have the constraints of problem (4) in the following form

$$T_{rs} = A^{(0)} x^{(0)} = T_{rs} b^{(0)}(0) \tag{9}$$

Let us write[1])

$$T_{rs} \ A^{(0)} = A_{rs}^{(1)} \quad \text{and} \quad T_{rs} \ b^{(0)}(0) = b_{rs}^{(1)}(0)$$

Before continuing a simplex algorithm at $t = 0$ (this optimization could be called *a point optimization)* we shall try to determine all the values of $t > o$ such that:

$$\min_i \frac{b_i^{(0)}(t)}{a_{is}^{(0)}} = \frac{b_r^{(0)}(t)}{a_{rs}^{(0)}} \geqslant 0$$

which means that the same transformation T_{rs} may be applied in the first iteration for all $o \leqslant t \leqslant t'$, where point t' must yet be determined.[2]) We can do this as follows.

Let us first assume that we have no pair of identical bounding functions in any iteration on any of the subintervals. For the first iteration on the first subinterval those functions are $b_i^{(0)}(t)$, $i = 1, \ldots m$. But according to the simplex algorithm we shall have to consider the modified function $b_i^{(0)}(t)/a_{is}^{(0)}$, $i = 1, \ldots m$. Consider the function $b_r^{(0)}(t)/a_{rs}^{(0)}$ defined by (9). In order to obtain the region or subinterval on which this function is minimal we have to check all the possible common roots of odd order of a set of equations

---

[1]) In further text we shall simplify the notaion by writing $T(r, s)$ instead of Trs, and $b^{(1)}(r, s)$ instead of $b_{rs}^{(1)}$. All simillar expressions will be put in this form.

[2]) We have assumed all bounding functions to be nonnegative for all interations. A somewhat broader case will be discussed later (see: An Infeasibility Algorithm).

$$\frac{b_r^{(0)}(t)}{a_{rs}^{(0)}} = \frac{b_i^{(0)}(t)}{a_{is}^{(0)}}, \quad i \neq r, \; i = 1, \cdots, \quad m \tag{10}$$

If there is no such root inside $[o, T]$, the function $b_r^0(t)/a_{rs}^{(0)}$ remains minimal throughout $[o, T]$. If there is a point $t = t'$ which is a root of some odd order of (10) for some i, let us say p, then the r-th upper bound, i. e. function $b_r^{(0)}(t)/a_{rs}^{(0)}$ is minimal on $[o, t']$. Then we investigate the function $b_p^{(0)}(t)/a_{ps}^{(0)}$ and try to find out whether this function is the minimal function or not on $[t' \; T]$ by checking all the possible roots of and odd order of

$$\frac{b_p^{(0)}(t)}{a_{ps}^{(0)}} = \frac{b_i^{(0)}(t)}{a_{ps}^{(0)}}, \quad i \neq p, \quad i = 1, \quad \cdots m \tag{10'}$$

If the smallest such root, say $t = t'' < T$, belongs to a function $b_q^{(0)}(t)/a_{qs}^{(0)}$

the p-th modified upper bound $b_p^{(0)}(t)$, i. e. $b_p^{(0)}(t)/a_{ps}^{(0)}$ is minimal on $[t', t'']$. We further investigate the minimality of the function $b_q^{(0)}(t)/a_{qs}^{(0)}$ in some region right from t'', by seeking some roots of an odd order, etc. In such a way we obtain subintervals $[o, t']$, $[t', t'']; \ldots [t^{(n-1)}, t^{(n)} \geqslant T]$. Let us now introduce a uniform and systematic notation. Let $r_\tau^{(1)}$ denote that row index for which we eliminate the corresponding vector from the current basis of the initial tableau (in carrying out the first iteration) on the subinterval $[t_{\tau-1}, t_r]$ of $[o, T]$, and let $s_\tau^{(1)}$ have a similar meaning, denoting the index of the vector introduced into a (new) basis.

We must therefore write (9)

$$T\left(r_1^{(1)}, s_1^{(1)}\right) A^{(0)} x^{(0)} = T\left(r_1^{(1)}, s_1^{(1)}\right) b^{(0)}(0) \tag{9'}$$

and let it be, in short

$$T\left(r_1^{(1)}, s_1^{(1)}\right) A^{(0)} = A\left(r^{(1)}, s_1^{(1)}\right), \; T\left(r_1^{(1)}, s_1^{(1)}\right) b^{(0)}(0) = b^{(1)}(r_1, s_1; 0)$$

Here $r_1^{(1)}$ is defined by

$$\frac{b^{(0)}\left(r_1^{(1)}, t\right)}{a^{(0)}\left(r_1^{(1)}, s_1^{(1)}\right)} = \min_i \frac{b_i^{(0)}(t)}{a\left(i, s_1^{(1)}\right)}$$

for the first iteration on the first subinterval. In a similar way, expressions (10) and (10'(have to be changed. As the subintervals obtained by the procedure above belong to the first iteration, we shall write them as being defined by the points

$$0 < t_1^{(1)} < t_2^{(1)} < \cdots < t_{l_1}^{(1)} \geqslant T.$$

At the same time we have found a sequence of transformations

$T_{(1)} = \left\{ T\left(r_1^{(1)}, s_1^{(1)}\right), \cdots T\left(r_{l_1}^{(1)}, s_{l_1}^{(1)}\right) \right\}$, where $T\left(r_j^{(1)}, s_j^{(1)}\right)$, $j=1, \cdots l_1$ is defined as $T(r, s)$.

Thus, we have the first computational tableau on the subinterval $\left(t_{\tau-1}^{(1)}, t_\tau^{(1)}\right]$ as follows:

$$T\left(r_\tau^{(1)}, s_\tau^{(1)}\right) A^{(0)} x(t) = T\left(r_\tau^{(1)}, s_\tau^{(1)}\right) b^{(0)}(t)$$

or

$$A\left(r_\tau^{(1)}, s_\tau^{(1)}\right) x(t) = b^{(1)}(r_\tau, s_\tau; t),$$

and the corresponding recursive formula

$$T\left(r_\rho^{(k+1)}, s_\rho^{(k+1)}\right) T\left(r_\tau^{(k)}, s_\tau^{(k)}\right) \cdot A^{(0)} = A\left(r_\rho^{(k+1)}, s_\rho^{(k+1)}\right)$$

for a region $(t_{\rho-1}, t_\rho] \subset (t_{\tau-1} t_\tau]$ we have yet to determine) can be used.

c) The Second and Further Iterations

Let us now start the second iteration at $t = 0$ and let an improving basis be introduced by the transformation $t\left(r_1^{(2)}, s_1^{(2)}\right.$ where the indices $r^{(2)}$ and $s^{(2)}$, have an analogous role to those appearing in the first iteration. After the second iteration at $t = 0$ is completed we try to extend the region of t so that he basis just obtained will continue improving the value of the objective function (this operation represents a part of what we shall further call *a local optimization)*. We do this as in the first iteration. We find a set of minimal functions on $[o, T]$, step by step, starting with the function

$$\frac{b^{(1)}\left(r_1 s_1, r_1^{(2)} t\right)}{a^{(0)}\left(r_1^{(1)} s^{(1)}\right)}$$

which is minimal at least at a point $t = 0$. In such a way we determine a corresponding sequence of points

$$0 < t_1^{(2)} < t_2^{(2)} < \cdots < t_{l_2}^{(2)} \geqslant T.$$

Here, for example, $t_1^{(2)}$ is determined as follows. We check the region of minimality of the function being minimal at $t = 0$. If this region is greater or equal to $[o, t_1^{(1)}]$, then $t_1^{(2)} = t_1^{(1)}$. Let us now assume that there are one or more minimal functions, different from the initial

one on $[o, t_1^{(1)}]$. We record the increasing sequence of the roots of an odd order, i. e. we take down the points of intersections of a set of minimal curves on the region $[o, t_1^{(1)}]$. The first element of this increasing sequence is the point $t_1^{(2)}$, the second is $t_2^{(2)}$, etc., until we hit point $t_1^{(1)}$, which we identify with $t_j^{(2)}$. Then we repeat the same procedure on $[t_1^{(1)}\ t_2^{(1)}]$, starting with the minimal function at $t_1^{(1)} = t_j^{(2)}$ and determined by the transformation $T\left(r_2^{(1)}, s_2^{(1)}\right)$. We continue in this way until we reach T. In general, for the i-th interation we can construct the sequence

$$0 < t_1^{(1)} < t_2^{(1)} < \cdots \qquad < t_{l_i}^{(i)} \geq T, \qquad i = 1, \cdots K.$$

It may, of course, happen that the number of iterations varies by t; let us say that we have K as a maximal number of iterations throughout $[o, T]$ and thus we admit some transformations to be identical. As a whole, we obtain the following transformation matrices

$$T\left(r_j^{(i)},\ s_j^{(i)}\right) \qquad i = 1, \cdots K; \qquad j = 1, \cdots l_i$$

which have to be used when optimizing the given objective function at the given constraints on $[o. T]$.

However, when switching at point $t = t_1^{(1)}$, from $T\left(r_1^{(1)}, s_1^{(1)}\right)$ for example, being used on $[o, t_1^{(1)}]$ to transformation $T\left(r_2^{(1)}, s_2^{(1)}\right)$ which is going to be used on $\left(t_1^{(1)}, t_2^{(1)}\right]$, we have[3])

$$\frac{b^0\left(r_1; t_1^{(1)}\right)}{a^{(0)}\left(r_1^{(1)}, s_1^{(1)}\right)} = \frac{b^{(0)}\left(r_2; t_1^{(1)}\right)}{a^{(0)}\left(r_2^{(1)}, s_2^{(1)}\right)}$$

i. e. minimum occurs for two different indices, $r_1^{(1)}$ and $r_2^{(1)}$. Degeneracy will occur in the next step. So, at all the switching points $t_j^{(i)}$ belonging to the i-th iteration we have to apply the degeneracy procedure starting with $(i + 1)$-th iteration.

We have assumed so far that there is no pair of identical (coinciding) modified functions of any right-hand side. In the opposite situation, if at a point $t = t_{\tau-1}^{(i)}$ a degeneracy occurs, we have a sequence of transformations $T\left(r_\tau^{(j)}, s_\tau^{(j)}\right)$, j = 1, ... K, which has to be used on $\left(t_{\tau-1}^{(i)}, t_\tau^{(i)}\right]$ where $t_\tau^{(i)}$ is a point of switching to the next minimal bounding function.

d) The Complete Representation

If we assume all the bounding functions in any of the iterations to be nonnegative, we can summarize the procedure as follows. The

[3]) Here, we simply write $b^{(0)}(r_1; t)$ instead of $b^{(0)}_{r_1}(t)$.

linear programming problem (4) can be solved by a sequence of point and local optimizations.

First, we have to find all switching points of the bounding functions beloinging to the initial computational tableau. These points define subintervals with a uniquely determined modified minimal function and corresponding transformation matrix. This transformation is used at the left-end point of each subinterval (a point optimization) as well as for any t inside that subinterval, including the right-end point (a local optimization). As a set of new bounding funtions does not have to have one and the same minimal bounding function as in the first iteration, we start determining all the possible switching points belonging to the modified bounds by the transformation in the first iteration, starting with the subinterval $[o, \ t_1^{(1)}]$, then $(t_1^{(1)}, t_2^{(1)}]$ etc. This means therefore that when reaching $t = T$ in the way mentioned above for the first iteration, we start the second iteration at $t = O$ first (a point optimization). The number of switching points does not decrease, with the number of iterations. In the given interation we have some (if any) newly obtained switching points from the current iteration plus the switching points from the previous iteration.

After each point optimization we have to optimize locally; that means we have to determine the region on which the same feasible basis can be used as at the initial point of that region. In order to continue the local optimization over the next region, we have to have analytical expressions for the right-hand sides in each iteration and each subinterval into which $[o, T]$ is going to be split.

As the transformation used in the i-th iteration on the subinterval $(t_{\tau-1}^{(i)}, t_\tau^{(i)}]$ is

$$b^{(i)}(r_\tau s_\tau, j; t) = b^{(i-1)}(r_\tau s_\tau, j; t) - a_{js}^{(i-1)} \frac{b^{(i-1)}(r_\tau s_\tau, r_\tau^{(i)}; t)}{a^{(i-1)}(r_\tau^{(i)}, s_\tau^{(i)})} \qquad (11)$$

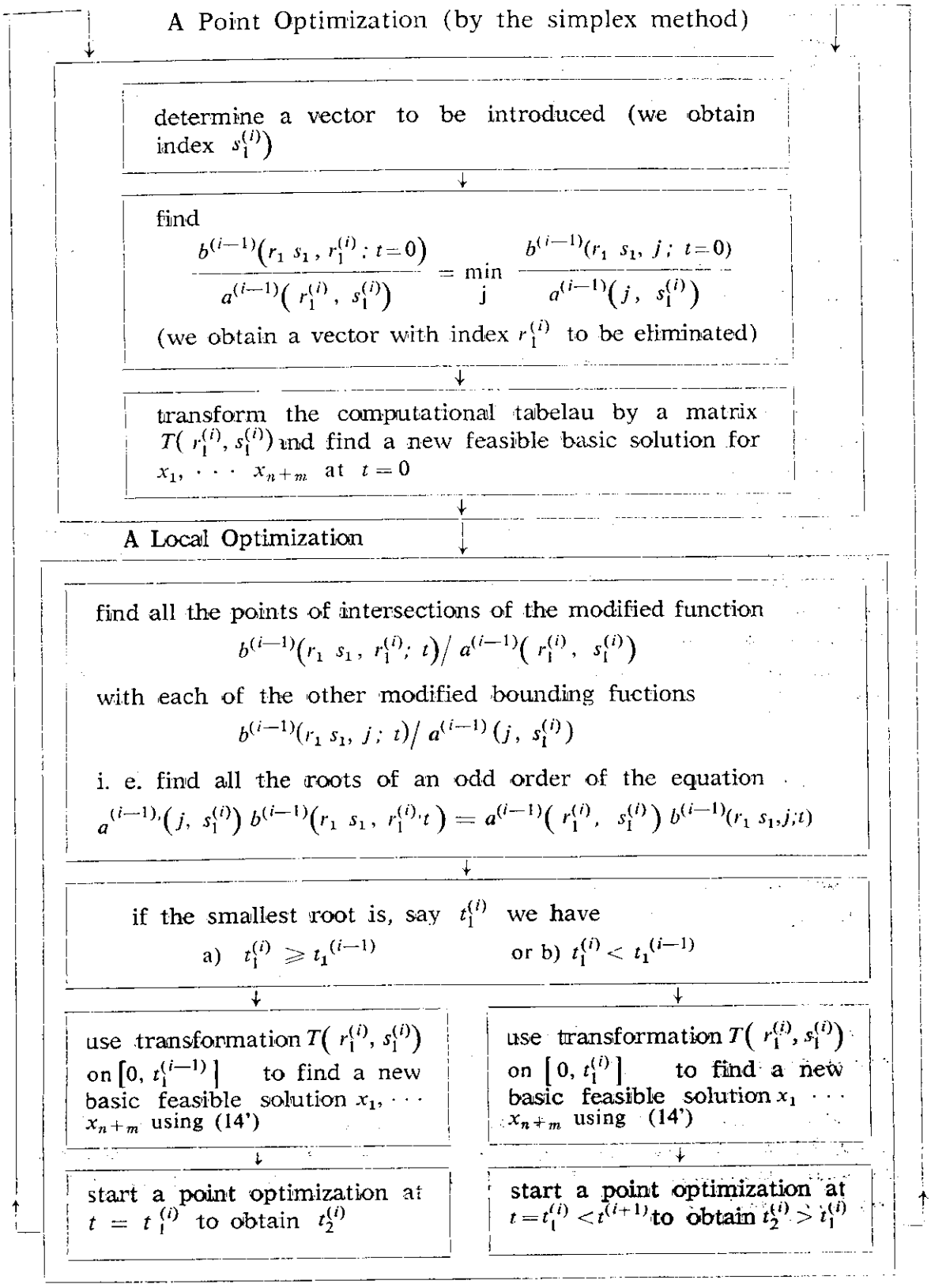
we have, after the k-th iteration, using (11) as a recursive on a subinterval, say $(t_{\sigma-1}, t_\sigma] \subset (t_{\tau-1}, t_\tau]$

$$b^{(k)}(r_\sigma s_\sigma, j; t) = L_{\sigma j}^{(k)} \{b^{(0)}(r_\sigma s_\sigma, j; t)\} \qquad (11')$$

i.e. the j-th bounding function obtained by the k-th iteration as a linear form of initial bounding functions $b^{(0)}(r_\sigma s_\sigma, j; t) \equiv b_j(t)$

We assumed all the bounding functions to be feasible every-where on $[o, T]$ and for any iteration. Let us consequently call the solution procedure discussed up to now *a feasibility algorithm* and demonstrate it by flow chart no. 1.

Flow Chart No. 1: A Feasibility Algorithm



4. Study of a b-dynamic Subproblem — An Infeasibility Algorithm

a) General Outline

So far we have assumed the nonnegativity of all the bounding functions $b^{(i)}(r_\tau\, s_\tau, j; t)$ for all iterations and all subintervals $(t_{\tau-1}^{(i)}, t_\tau^{(i)}]$. However, this is not, in general, the case, except for the initial tableau.

The optimality of a given basis is invariable so long as it gives a feasible solution for $x_j(t)$, $j = 1, \ldots n$. An infeasibility occurs when for $i \in \{1, \ldots m\}$ we have $b^{(i)}\, r_\tau\, s_\tau, j; t) < o$. We shall discuss what additional algorithm will have to be added to the computational procedure described above if infeasibilities occur.

In order to assure the feasibility of the solutions, we have to investigate the bounding functions (11) with respect to their zeros and gradients. It will be seen that two important cases arise here: the smallest zeros of an odd order of the obtained bounding functions (11) arrived 'at in the i-th iteration where for some value of t some infeasibilities occur coincide (Case B) or not (Case A).

Let us consider the subinterval $(t_{\tau-1}^{(i)}, t_\tau^{(i)}]$ and let the function $b^{(i)}(r_\tau\, s_\tau, j_1; t)$ become zero at $t = \bar{t}_\tau^{(i)} < t_\tau^{(i)}$ and be negative from there on, so that we have

$$b^{(i)} r_\tau\, s_\tau, j; t) \geqslant 0 \quad \text{for} \quad t_{\tau-1}^{(i)} < t \leqslant t_\tau^{(i)},\ j = 1, \cdots m;\ j \neq j_1$$

$$b^{(i)} r_\tau\, s_\tau, j_1; t) \geqslant 0 \quad \text{for} \quad t_{\tau-1}^{(i)} < t \leqslant \bar{t}_\tau^{(i)}$$

$$b^{(i)} r_\tau\, s_\tau, j_1; t) < 0 \quad \text{for} \quad \bar{t}_\tau^{(i)} < t \leqslant t_\tau^{(i)}.$$

A feasible basis obtained at point $t = t_{\tau-1}^{(i)}$ in the i-th iteration will therefore remain feasible on $(t_{\tau-1}^{(i)}, \bar{t}_\tau^{(i)}]$. At point $\bar{t}_\tau^{(i)}$ we have to find another basis which will be feasible again. That means we shall have to find a new set of feasible bounding functions. This operation may be called *an infeasibility algorithm*. Despite determining the subinterval $(t_{\tau-1}^{(i)}, \bar{t}_\tau^{(i)}]$ on which only one and the same bounding function is minimal, i.e. throughout which one and the same transformation $T(r_\tau^{(i)}\ s_\tau^{(i)})$ could be used, we have to check the feasibility of all bounding functions (11) on the subinterval $(\bar{t}_{\tau-1}^{(i)}, t_\tau^{(i)}]$.

An infeasibility algorithm will formally consist of a pivoting operation. We shall replace variable x_p and introduce another one, say x_q, of course, if this is possible at all.

b) Detailed Discussion

The constraints of problem (4) have at $t = \bar{t}_\tau^{(i)}$ the following form (for the sake of simplicity we have not denoted a particular basis):

$$\sum_{j=1}^{u+m} a_{1j}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau,\ 1;\ \bar{t}_\tau^{(i)}\right)$$

.....

$$\sum_{j=1}^{n+m} a_{pj}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau,\ p;\ \bar{t}_\tau^{(i)}\right)$$

.....

$$\sum_{j=1}^{n+m} a_{j_1 j}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau,\ j_1;\ \bar{t}_\tau^{(i)}\right) = 0$$

.....

$$\sum_{j=1}^{n+m} a_{mj}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau,\ m,\ \bar{t}_\tau^{(i)}\right)$$

.....

After the transformation we have

$$\sum_{j=1}^{n+m} \bar{a}_{1j}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau,\ 1;\ \bar{t}_{1q}^{(i)}\right) - a_{1q}^{(i)}\,\frac{b^{(i)}\left(r_\tau\, s_\tau,\ p;\ \bar{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}}$$

.....

$$\sum_{j=1}^{n+m} \bar{a}_{pj}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = \frac{b^{(i)}\left(r_\tau\, s_\tau,\ p,\ \bar{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}}$$

.....

$$\sum_{j=1}^{n+m} \bar{a}_{j_1 j}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau\, j_1;\ \bar{t}_\tau^{(i)}\right) - a_{j_1 q}^{(i)}\,\frac{b^{(i)}\left(r_\tau\, s_\tau,\ p;\ \bar{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}} \equiv$$

$$\equiv -\,a_{j_1 q}^{(i)}\,\frac{b^{(i)}\left(r_\tau\, s_\tau,\ p;\ \bar{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}}$$

.....

$$\sum_{j=1}^{n+m} \bar{a}_{mj}^{(i)}\, x_j\left(\bar{t}_\tau^{(i)}\right) = b^{(i)}\left(r_\tau\, s_\tau,\ m;\ \bar{t}_\tau^{(i)}\right) - a_{mq}^{(i)}\,\frac{b^{(i)}\left(r_\tau\, s_\tau,\ m;\ \bar{t}_\tau^{(i)}\right.}{a_{pq}^{(i)}}$$

where

$$\overline{A}^{(i)} = T\left(p_\tau^{(i)}, q_\tau^{(i)}\right) \quad A^{(i)} = \left(\overline{a}_{kj}^{(i)}\right)$$

First of all, it must be

$$a_{j_1q}^{(i)} \Big/ a_{pq}^{(i)} > 0$$

From this it follows

$$\text{if} \qquad a_{j_1q}^{(i)} \lesseqgtr 0 \qquad \text{then} \qquad a_{pq}^{(i)} \lesseqgtr 0 \quad .$$

In order to ensure a new set of feasible bounding functions, we have to admit a case

$$a_{j_1q}^{(i)} < 0 \qquad \text{and} \qquad a_{pq}^{(i)} > 0 \tag{12}$$

only, because all

$$b^{(i)}\left(r_\tau s_\tau, j; \ \overline{t}_\tau^{(i)}\right) > 0, \qquad j = 1, \cdots m; \quad j \neq i \ .$$

Case A-I: Let us assume that there exists at least one negative $a_{j_1q}^{(i)}$; if there are more of them, we choose

$$a_{j_1q}^{(i)} = \max_{j} \quad a_{j_1j}^{(i)} < 0 \tag{13}$$

We determined here a vector with index q to be introduced. Next, we shall choose p so as to minimize a new bound, i.e. we shall take only strictly positive bounds into account when determining a vector to be replaced, first of all, on the basis of the following criterion:

$$\frac{b^{(i)}\left(r_\tau s_\tau, \ p, \ \overline{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}} = \min_{j} \ \frac{b^{(i)}\left(r_\tau s_\tau, j; \ \overline{t}_\tau^{(i)}\right)}{a_{jq}^{(i)}} \tag{13'}$$

If there are all $b^{(i)}(r_\tau s_\tau, j; \ \overline{t}_\tau^{(i)}) = o, \ j = 1, \ldots m; \ j \neq j_1$, no feasibility can be assured, and the problem (4) is solvable only up to the i-th iteration on $[o, \overline{t}_\tau^{(i)}]$. We shall discuss this question in detail later.

Let us assume that there is at least one positive upper bound at $t = \overline{t}_\tau^{(i)}$ different from $b^{(i)}\left(r_\tau s_\tau, j_1; \ \overline{t}_\tau^{(i)}\right)$.

By (13') we found a vector to be replaced from the basis; let us assume all (13') be monotonically differentiable. The corresponding function defined by (13') could be used on some region $\left(\overline{t}_\tau^{(i)}, \ \overline{\overline{t}}_\tau^{(i)}\right]$, where $\overline{t}_\tau^{(i)} < \overline{\overline{t}}_\tau^{(i)} \leqslant t_\tau^{(i)}$ and $\overline{\overline{t}}_\tau^{(i)}$ will be determined as a point of intersection between this function and some other one starting to replace it at this point as a new minimal function.

In order to determine this point let us define two nondecreasing sequences: a nondecreasing sequence F

$$F = \left\{ b^{(i)} \left(r_\tau s_\tau, j; \overline{t}_\tau^{(i)}\right) / a_{js}^{(i)} \right\} \qquad j = 1, \cdots m$$

and a nondecreasing sequence G

$$G = \left\{ \frac{d}{dt} \left[b^{(i)} \left(r_\tau s_\tau, j; \overline{t}\right) / a_{js}^{(i)} \right] t = \overline{t}_\tau^{(i)} \right\} \qquad j = 1, \cdots m$$

In a case of ties, when setting up sequence F, we arrange its elements (in a tie) in the same order as they are ordered in G. The same is true when there are ties in G. When there are ties for the same subset of elements in both F and G, we arrange them arbitrarily.

If the minimal element in F coincides with the minimal one in G, i.e. when one and the same function has the smallest value and the smallest gradient at $t = \overline{t}_\tau^{(i)}$, then the function given by (13') is used for the construction of a new feasible bounding function which will be used (instead of $b^{(i)}(r_\tau s_\tau, j_1; t)$ right from $t = \overline{t}_\tau^{(i)}$ until a zero of an odd order occurs (either of the same function we have just introduced or of some other one).

However, it is quite a special case. Let us now assume that the two minimal elements in F and G belong to different functions. Let these two functions be

$$\frac{b^{(i)}(r_\tau s_\tau, p; t)}{a_{pq}^{(i)}} \in F \qquad \frac{b^{(i)}(r_\tau s_\tau, \overline{p}; t)}{a_{\overline{p}q}^{(i)}} \in G$$

A set of bounding functions obtained by introducing the minimal function (13') can be used up to a point $\overline{\overline{t}}_\tau^{(i)}$ which is a root of an odd order of the equation

$$a_{\overline{p}q}^{(i)} \; b^{(i)}(r_\tau s_\tau, p; t) = a_{pq}^{(i)} \; b^{(i)}(r_\tau s_\tau, \overline{p}; t) \qquad (13'')$$

Now we choose

$$\left.\begin{array}{l} b^{(i)}\left(r_\tau s_\tau, p; t\right) \text{ on } \left(\overline{t}_\tau^{(i)}, \overline{\overline{t}}_\tau^{(i)}\right] \\ b^{(i)}\left(r_\tau s_\tau, \overline{p}; t\right) \text{ on } \left(\overline{\overline{t}}_\tau^{(i)}, t_\tau^{(i)}\right] \end{array}\right\} \text{ if } \frac{b^{(i)}\left(r_\tau s_\tau, \overline{p}; \overline{t}_\tau^{(i)}\right)}{a_{\overline{p}q}^{(i)}} > \frac{b^{(i)}\left(r_\tau s_\tau, p; \overline{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}} \qquad (14)$$

$$b^{(i)}\left(r_\tau s_\tau, \overline{p}; t\right) \text{ on } \left(\overline{t}_\tau^{(i)}, t_\tau^{(i)}\right] \text{ if } \frac{b^{(i)}\left(r_\tau s_\tau, \overline{p}; \overline{t}_\tau^{(i)}\right)}{a_{\overline{p}q}^{(i)}} = \frac{b^{(i)}\left(r_\tau s_\tau, p; \overline{t}_\tau^{(i)}\right)}{a_{pq}^{(i)}}$$

$$\left.\begin{array}{l} b^{(i)}\left(r_\tau s_\tau, p; t\right) \text{ on } \left(\overline{t}_\tau^{(i)}, t_\tau^{(i)}\right] \\ t_\tau^{(i)} \text{ being zero of an odd} \\ \text{order of the function } b^{(i)}(r_\tau s_\tau, p; t) \end{array}\right\} \text{ if no } \overline{\overline{t}}_\tau^{(i)} < t_\tau^{(i)} \text{ exists}$$

In such a way we determine a sequence of sets of minimal feasible bounding functions from $\overline{t}_\tau^{(i)}$ on up to $t_\tau^{(i)}$. The next point of infeasibility is therefore either $\overline{\overline{t}}_\tau^{(i)} = t_\tau$ where we repeat the feasibility algorithm as used at $t = \overline{t}_\tau^{(i)}$, or $\overline{\overline{t}}_\tau^{(i)} < t_\tau^{(i)}$, where we change the basis according to rule (14). We apply the feasibility algorithm again at $\overline{\overline{t}}_\tau^{(i)}$. When all the bounding functions are of quite a general nature, we may have more than one point $\overline{\overline{t}}_\tau^{(i)} \neq t_\tau^{(i)}$. In such a case we have to change the set of bounding functions at each such point as above.

Case A—II: Let us now assume that after using the feasible set of bounding functions up to a point $\overline{t}_\tau^{(i)}$ we arrived at the coefficients matrix with the i-th row consisting of nonnegative coefficients. It can easily be seen that no positive new upper bound exists for any small region right from $\overline{t}_\tau^{(i)}$

If we multiply the j-th row by —1 and introduce an artificial variable, then by every pivot element $a_{pq}^{(i)}$for which $p \neq j$, $q \in \{1, \ldots n+m\}$ both determined so as to minimize the new right-hand side, a new basis can be obtained which contains an artificial vector corresponding to the artificial variable. When optimizing in Phase I of the simplex method we can not get rid of the artificial vairable, which means that the original problem (i.e. without an artificial variable) has no feasible solution .If the pivot element is nogative, we obtain the same solution at $t = \overline{t}_\tau^{(i)}$ as before. No improvement has been achieved. This basis cannot be used further, because it gives one of the optimal variables the value of a negative bounding function. This means that if all $a_{j_1 j}^{(i)} > o$, $j = 1, \ldots n + m$, for that row index j_1 for which an infeasibility occurs at $t = \overline{t}_\tau^{(i)}$, the optimizing procedure may terminate without reaching the optimum on the subinterval starting at point $t_{\tau-1}^{(i)}$.

Case B: So far we have been assuming that all the smallest zeros of an odd order of the minimal and feasible modified bounding functions were noncoinciding. However, if this is not the case, we have to ensure, let us say, λ feasible and minimal bounding functions

$$b^{(i)}\ (r_\tau\ s_\tau,\ 1_\lambda;\ t),\ \cdots\ b^{(i)}\ (r_\tau\ s_\tau,\ j_\lambda;\ t) \tag{15}$$

at a point $t = \overline{t}_\tau^{(i)}$ which is a common root of an odd order of these functions. According to the procedure described in A-I there would be a unique way of minimizing a new set of feasible bounding functions, i.e. we could use the same criterion for determining index p. But, when determining index s, we cannot expect, in general, all $a_{jp\,q}^{(i)} < o$, $p = 1, \ldots \lambda$ to occur in one and the same column. If this is the case, no new feasible bound can be found. The original problem is therefore feasible in the i-th iteration on $(t_{\tau-1}^{(i)}, \overline{t}_\tau^{(i)}]$ only. The detailed discusion of the situation we just described will be given in Ch. 4.

c) A Complete Representation

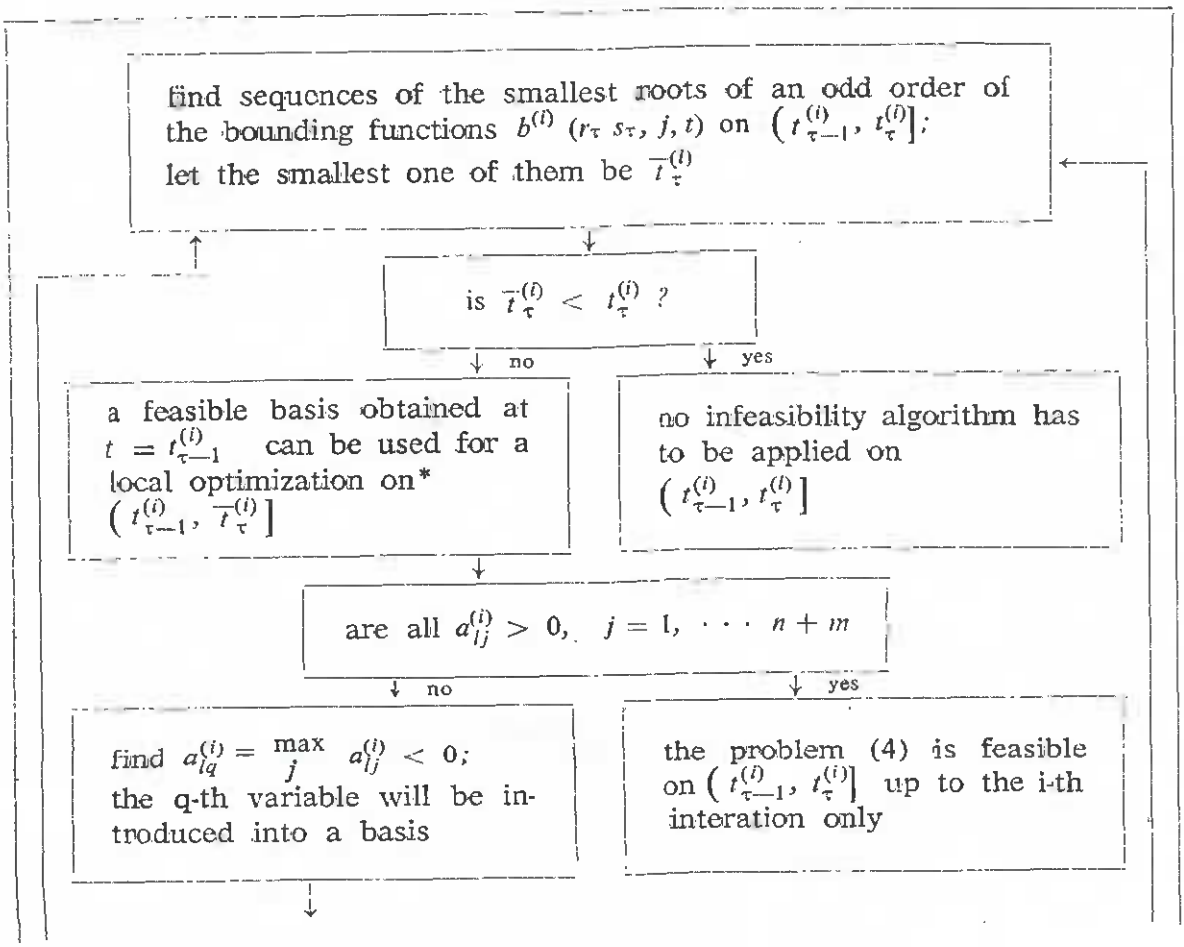
Let us now make a summary of the results obtained. We first assumed that no infeasibility occurs as to the bounding function any-

where on [o, T] and in any iteration. In this case the procedure is demonstrated by flow chart no. 1. Second, assuming some infeasibilities may occur on the part of bounding functions, we established an additional procedure, which enables us to eliminate infeasibilities if the following conditions hold:

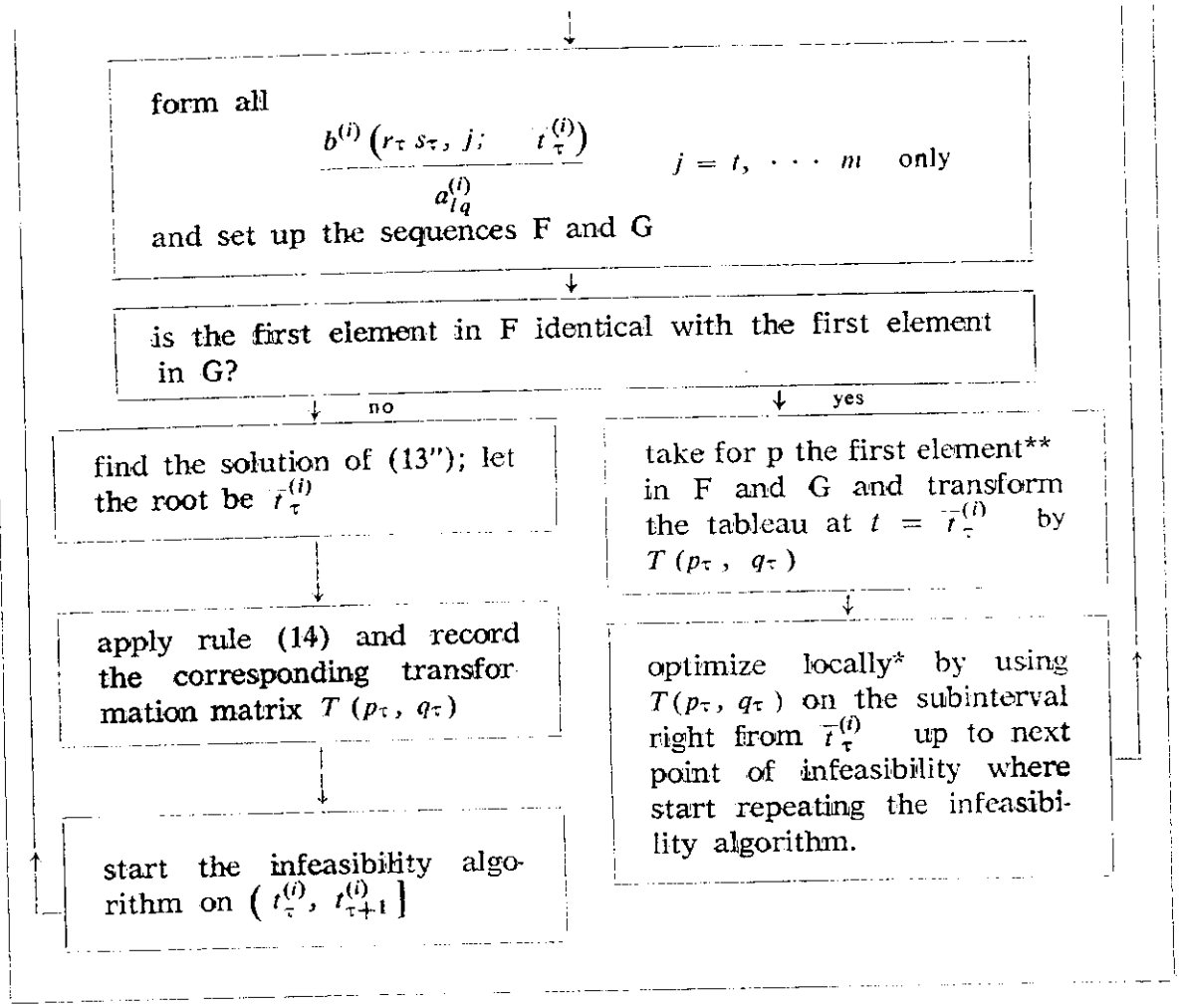
1) all the smallest roots of an odd order of the bounding functions (15) do not coincide at any iteration ($i = 1, \ldots K$) on any subinterval $(t_{\tau-1}^{(i)}, t_{\tau}^{(i)}] \subset [0, T]$ if $\max_j a_{j_p q}^{(i)} = a_{i_p q}^{(i)}$ does not occur at the same q for all $p = 1, \ldots \lambda$.

2) at each iteration we obtain a matrix of coefficients $a_{ij}^{(i)}$ where we have at least one $a_{qj}^{(i)} < o$ for that q for which an infeasibility occurs. Let us now demonstrate an infeasibility algorithm by flow chart no. 2.

Flow Chart No. 2: An Infeasibility Algorithm



* When optimizing locally, we record the corresponding set of new feasible bounding functions.



5. Study of a b-dynamic Subproblem — A Complete Algorithm

We studied problem (4) by first assuming that there was no infeasibility of any bounding functions. Next we have admitted infeasibilities. Let us now set up *a complete algorithm.*

We first apply the feasibility algorithm in order to obtain the first iteration. This gives us a sequence $T_{(1)}$ of transformations in 2.—b). There is no need to apply the infeasibility algorithm in the initial tableau because we assumed all initial bounding functions $b_j^{0)}(t)$ to be nonnegative. By a system of transformations we transform the initial tableau and obtain the first tableau to be used for the second iteration. The bounding functions of the first tableau may become nonnegative at some point inside $[o, T]$. We therefore start the second iteration by application of the feasibility algorithm on subinterval $[o, t^{(1)}]$. This operation may give us a partitioning of $[o, t_1^{(1)}]$ into subintervals

$$[0, t_1^{(2)}], \quad (t_1^{(2)}, t_2^{(2)}], \quad \cdots \quad (t_{p_1^1}^{(2)}, t_1^{(1)}] \tag{16}$$

** i.e. we take for p that index for which the corresponding function is in the first place in F or G respectively.

so that a corersponding sequence of transformations

$$T\left(r_1^{(2)},\ s_1^{(2)}\right),\quad \cdots\quad T\left(r_{p_1^1}^{(2)},\ s_{p_1^1}^{(2)}\right) \tag{17}$$

is generated. Before we continue the second iteration on $(t_1^{(2)},\ t_2^{(2)})$ in the same way as on $(t_1^{(1)},\ t_2^{(1)}]$, we have to check some potential infeasibilities which may occur when (using the feasibility algorithm) optimizing locally over subinterval (16). We therefore apply the infeasibility algorithm we discussed in section 3.

This algorithm may provide us with sets of feasible bounding functions over the subintervals (16). If on some subinterval, say $(t_{\tau-1},\ t_\tau^{(2)}]$ an infeasibility cannot be eliminated, the optimizing procedure terminates with the first iteration, not because we have reached the optimum value of criterion function, but because there is no feasible solution from $t_{\tau-1}^{(2)}$ on up to $t_\tau^{(2)}$.

If, however, all the infeasibilities are eliminated by the infeasibility algorithm, we proceed by applying the feasibility algorithm in the computational tableaus obtained by transformation of all original tableaus (obtained by transformations (17) on each respective subinterval) with the transformations.

$$T\left(r_j^{(2)},\ s_j^{(2)}\right),\quad T\left(p_{1j}^{(2)},\ q_{1j}^{(2)}\right),\quad \cdots\quad T\left(p_{\mu j}^{(2)},\ q_{\mu j}^{(2)}\right)$$
$$j = 1,\quad \cdots\quad p_1^1 \tag{18}$$

which (except for the first one) have been obtained by using an infeasibility algorithm on the subinterval $(t_{j-1}^{(2)},\ t_j^{(2)}]$. It is evident that a set of transformations (18) is the final one because a set of bounding functions feasible at a point remains feasible over some neighbourhood right from $t_{j-1}^{(2)}$ over which $T(r_j^{(2)},\ s_j^{(2)})$ therefore remains applicable. After this region infeasibilities may start occuring and therefore the rest of the transformations of (18) to be applied.

After the application of the infeasibility algorithm over all the subinterval (16) is over, we continue in the same way the examination of potential infeasibilities on all the subintervals, corresponding to $(t_1^{(1)},\ t_2^{(1)}]$. So, we continue until $t_{l_1}^{(1)} \geq T$. Of course, the second »iteration« we obtained does not necessarily mean an improvement of the value of the objective function over all the subintervals belonging to this step. If no infeasibility occurs, we certainly improve the value of the objective function. If there are some points of infeasibility, we have to replace the original transformation $T\left(r_j^{(2)},\ s_j^{(2)}\right)$ after some initial subregion, with a set of transformations (18), which do not necessarily bring us an improvement in the value of the objective function. We shall try to do that in subsequent iterations.

In general, we shall have two types of transformations $T(r_j^{(1)}, s_j^{(i)})$ and $T\ (p_j^{(i)}, q_j^{(i)})$. If we arrive at a set of transformations »covering« the whole $[o, T]$, for the whole set of iterations necessary to optimize the objective function in (4), we shall say that we obtained *a complete* ***set of transformations.***

Let us now for the sake of simplicity denote by

$$T_j^{(i)} \qquad \begin{matrix} i = 1; \ \cdots \ K \\ j = 1, \ \cdots \ \omega. \end{matrix} \tag{19}$$

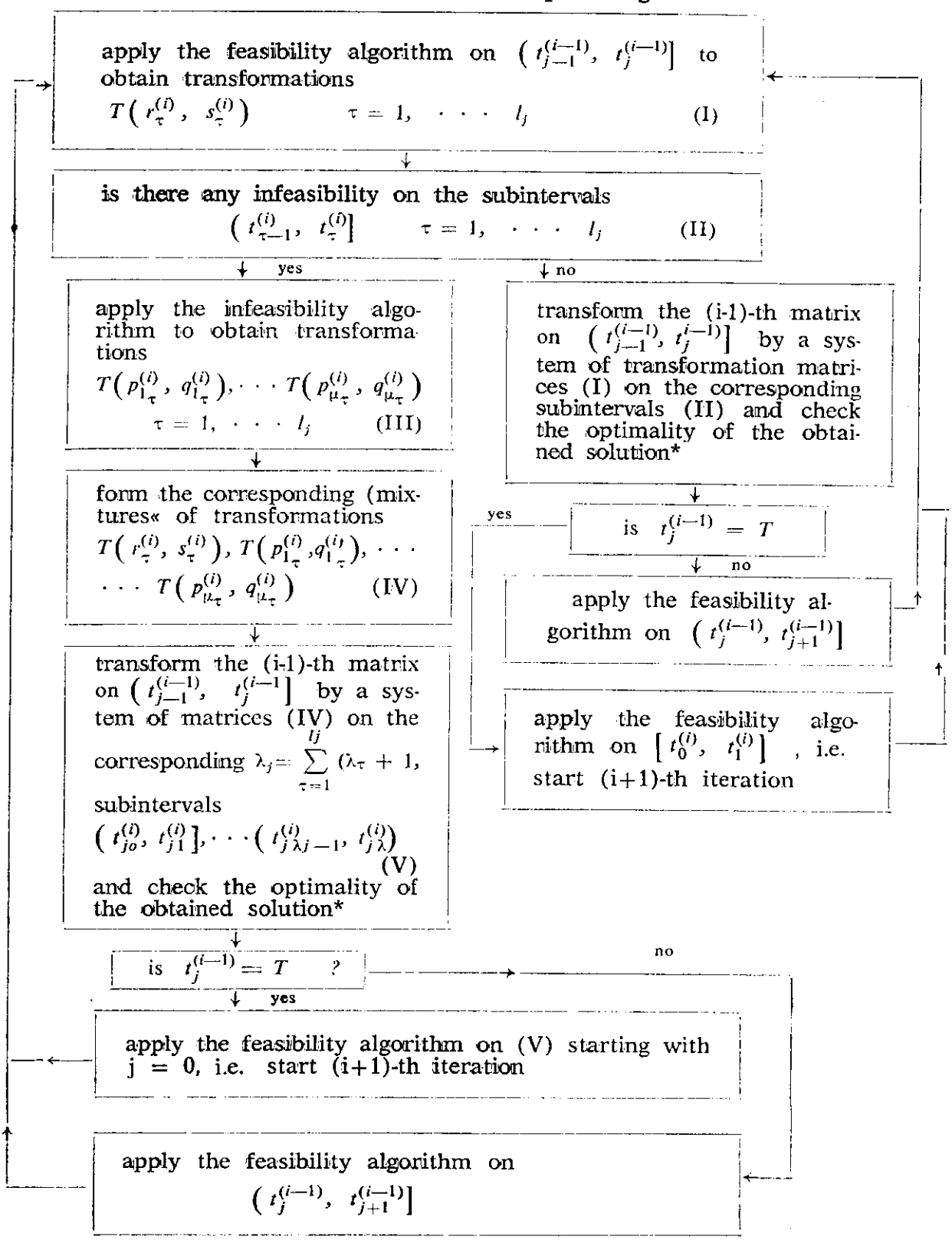
the j-th transformation used in the i-th iteration, i.e. on the j-th subinterval when solving problem (4) by interchangeable use of feasibility and infeasibility algorithms, and let a set of transformations (19) represent a complete set of transformations. Here, ω is the maximum number of subintervals obtained on $[o, T]$ over the complete set of iterations.

Earlier we postponed a discusion of a case when infeasibility cannot be eliminated. Let us now see what that means. If from some point, say $t_\tau^{*(i)}$ on some particular bounding function remains negative, we cannot ensure a new set of feasible bounding functions. This is a result of an unsuccesful application of the infeasibility algorithm on subinterval $(t_{\tau-1}^{(i)}, t_\tau^{(i)}]$, where such a point $t_\tau^{*(i)}$ lies inside that subinterval. It follows from this that no feasible solution exists on $(t_{\tau-1}^{*(i)}, t_\tau^{(i)})$. In another words, we have succeeded in optimizing the objective function up to $(i-1)$ — th iteration only on $(t_{\tau-1}^{*(i)}, t_\tau^{(i)}]$, i.e. on $(t_{\tau-1}^{*(i-1)}, t_\tau^{(i-1)}]$ with $t_{\tau-1}^{*(i)} \equiv t_{\tau-1}^{*(i-1)}$ and $t_\tau^{(i)} \equiv t_\tau^{(i-1)}$. Thus, a failure of the infeasibility algorithm on $(t_{\tau-1}^{*(i)}, t_\tau^{(i)}]$ compells us to replace the belonging (expected) solution with that obtained in previous iteration on the same subinterval.

In a case of succesful application of infeasibility algorithms only a matrix $T_j^{(i)}$ is being used in the i-th iteration on a subinterval $(t_{j-1}^{(i)}, t_j^{(i)}]$ and may be a »feasibility« matrix, i.e. a matrix of the type $T\ (r_j^{(i)}, s_j^{(i)})$ or »infeasibility« matrix of the type $T\ (p_j^{(i)}, q_j^{(i)})$. Using a complete algorithm under the above assumption we arrive at feasibility and infeasibility matrices, but if an identity transformation matrix follows a feasibility matrix it means that the optimum has been obtained in the i-th iteration on $(t_{j-1}^{(i)}, t_j^{(i)}]$, or the optimizing procedure for that interval terminates at the i-th iteration. If an identity transformation matrix follows an infeasibility matrix, it means that the infeasibility algorithm stops the complete algorithm in the i-th iteration on $(t_{j-1}^{(i)}, t_j^{(i)}]$. As to the termination of the whole optimizing procedure, we have identity transformations belonging to all further iterations on $(t_{j-1}^{(i)}, t_j^{(i)}]$. Thus, a solution may be optimal (when the last non-identity transformation is of the type $T\ (r_j^{(i)}, s_j^{(i)})$ or suboptimal (in either case).

We shall now demonstrate the whole solution procedure for problem (4) by flow chart no. 3., which shows a complete algorithm.

Flow Chart No. 3.: A Complete Algorithm



*) In this chart we have to check the optimality of the obtained feasible solution. If it is optimal we record the corresponding subinterval and iteration in order to avoid further iterations on a given subinterval. As we have seen all the subsequent matrices are identical matrices.

6. STUDY OF A PROBLEM — A SIMULTANEOUS APPROACH

So far we have developed the complete algorithm for a continuous b-dynamic linear programming problem. Let us return to original problem (2) and suppose we found a solution of the corresponding problem (4). The partitioning of $[o, T]$ has been obtained through the use of the complete algorithm belonging to (4). How can we include the changing vector $c(t)$ into the solution obtained in this way?

We can carry out a point optimization at $t = o$. After that we move to the right from $t = o$ and observe whether the solution obtained can be used for any $t > o$. As we have seen, what prevented us from using the solution obtained was the »behaviour« of $b(t)$. Now, we have two possible causes for stopping the use of the solution obtained: either the behaviour of $b(t)$ or the behaviour of $c(t)$ (or both). Thus, after a point optimization at $t = o$ we proceed from $t = o$ to the right, but we watch both:

1) $b(t)$ according to the complete algorithm for a continuous b-dynamic linear programming problem

2) $c(t)$ according to the stability test procedures amounting to criterion coefficients.

If the $c(t)$ — stability test fails at any t, say $t' > o$, there a point optimization should take place (because of $c(t)$ — stability test failure). If the $c(t)$ — stability test is passed, then a point optimization should take place at a point t indicated by the complete algorithm for a continuous b-dynamic linear programming problem.

If we tried to develop a complete algorithm for a continuous c-dynamic linear programming problem, we would find a procedure similar to the one for the b-dynamic subproblem, although a bit simpler. Therefore we shall discuss it in detail.

Let us now refer to a $b(t)$ — *algorithm* which we have discussed in detail. Similarly, a $c(t)$ — *algorithm* could stand for an algorithm which gives us a solution of (6). Now, from what we said above, we can say roughly, that a simultaneous approach could be used for problem (2) by applying the approach of $b(t)$ — and $c(t)$ — algorithm at the same time. When changing t, the points optimization take place at »critical« points indicated either by the $b(t)$ — or the $c(t)$ — algorithm. Thus, we can look for a solution of (2).

7. APPLICATION

a) Optimal product line problems

In operations research we often have to deal with optimal product line problems. Suppose we have an entrepreneur who wants to maximize his total profit accruing from the optimal selection of his products from a (technologically) feasible set of products.

Let P_j, $j = 1, \ldots n$ be the products entering into the selection procedure. Let c_j, $j = 1, \ldots n$ be the corresponding profit—coefficents,

and matrix $A = (a_{ij})_{mxn}$ should represent the technology, enabled by a resource vector b. Thus, the standard formulation of an optimal product line problem is

$$\max (c, x)$$
$$Ax \leqslant b$$
$$x \geqslant o$$

where c is a profit vector $c = (c_1, \ldots c_n)$ and $x = (x_1 \ldots x_n)$. A problem like the above is usually called a production — planning problem, where, of course, this term is very restrictive indeed. A planning period in business is one of practical size, say a year or half a year, or even a month. Let us suppose that business conditions vary in time substantially. This may force the entrepreneur to act »optimally« more often than usually. In this case, we can formally set $c=c(t)$, $x=x(t)$ and $b = b(t)$.

Through such a »dynamization« we arrive at problem (2), which is a continuous version of a discrete set of optimal product line problems. There is a need of reappraisal of his product policy, at time t, when new levels of stocks of materials, labour power, etc. are available, i.e. an existing value of $b(t)$ should be taken into account. Similarly, profit coefficients $c_j(t)$ may have assumed different levels in comparison with a previous production program. Planning periods get shorter and thus planning tends to control and management procedures; this is just what we meant in the introduction to this paper.

To see the importance of such a control aspect, let us allow that the upper limits to selling quantities are imposed on an optimal product line problem. In the case of an oligopoly, profits $c_j(t)$, $j = 1, \ldots n$ may vary substantially, owing to the competitors' actions, although demand remains unchanged. In such a case, we have to replace $Ax \leqslant b$ with:

$$\begin{pmatrix} A \\ E \end{pmatrix} x(t) \leqslant \begin{pmatrix} b(t) \\ m(t) \end{pmatrix}$$

where E is a unit matrix and vector $m(t)$ represents the levels of existing demand. Such a »control-oriented« optimal product line problem is extremely helpful in the case of a service oriented enterprise, where the dynamics in its queues is important so as to include penalty or damage coefficients and lost profit coefficients into the functional.

b) Systems approach to economic consolidation and cooperation

Aggregation and disaggregation problems are another case of the useful application of continuous dynamic linear programming. We could conceive that two partners A and B (e.g. enterprises) realised that they could successfully combine their resources b_A and b_B con-

fronting their different technologies, say M_A and M_B (these are matrices) and selecting among their potential production programs x_A and x_B It may well happen that their profits differ, even for the same production program; therefore, it is reasonable to assume some c_A and c_B On the basis of the diagnostic theory of economic consolidation and cooperation, [1], [2], they are able to decide upon a merger or against it, at $t = t_o$. But such a desision might be withdrawn at some $t > t_o$. We therefore switch to the following problem, e.g. for a maximum profit

$$\max \left\{ [c_A(t), \quad x_A(t)] + [c_B(t), \quad x_B(t)] \right\} = F_o$$

subject to

$$M_A x_A(t) + M_B x_B(t) \leqslant b_A(t) + b_B(t)$$

$$\underline{x}_A(t) \leqslant x_A(t) \leqslant \bar{x}_A(t)$$

$$\underline{x}_B(t) \leqslant x_B(t) \leqslant \bar{x}_B(t)$$

$$x_A(t) \geqslant 0, \quad x_B(t) \geqslant 0$$

$$0 \leqslant t \leqslant T$$

where we assumed 1) both resources $b_A(t)$ and $b_B(t)$ are additive (a similar approach can be used if they are not), 2) the two partners are »coupled« horizontally, 3) $\underline{x}_A(t)$ resp. $\bar{x}_A(t)$ are given the lower resp. upper limit (estimated or forecast) of demand, amounting to a set of products being supplied by partner A; $\underline{x}_B(t)$ resp. $\bar{x}_B(t)$ has a similar meaning.

Suppose that the two partners A and B are willing to be »coupled« as long as $F_o > F_o(A) + F_o(B)$, where $F_o(A)$ resp. $F_o(B)$ are optimal values of functional belonging to A resp. B only with the corresponding constraints $M_A x_A(t) \leqslant b_A(t)$ resp. $M_B x_B(t) \leqslant b_B(t)$ and $\underline{x}_A(t), \leqslant x_A \leqslant_A \bar{x}(t)$, $x_A(t) \geqslant o$ resp. $\underline{x}_B(t) \leqslant x_B(t) \leqslant \bar{x}_B(t)$, $x_B(t) \geqslant o$ with $o \leqslant t \leqslant T$ in both cases. Thus, the concept of a continuous dynamic linear programming turns out to be useful where we want to find $F_o(A)$ and $F_o(B)$.

(Rad primljen avgusta 1973.)

REFERENCES

1) Viljem Rupnik, *Metodologija za ugotavljanje ekonomskih efektov horizontalne integracije in kooperacije*, RCEF—ISOR, 1969.

2) Viljem Rupnik, *Metodologija za ugotavljanje ekonomskih efektov vertikalne integracije in kooperacije*, RCEF—ISOR, 1970.

ZVEZNO DINAMIČNO LINEARNO PROGRAMIRANJE

Viljem RUPNIK

Resume

V razpravi obravnavamo problem, kako rešiti takšen linearni program, kjer so spremenljivke v funkcionalu, koeficienti funkcionala in desne strani omejitev linearnega tipa odsekoma zvezne funkcije. Za rešljivost tega problema se zahteva, da so vrednosti desnih strani takšnega problema tudi enkrat odsekoma zvezno odvedljive.

Celotno nalogo je mogoče rešiti tako, da se najprej lotimo izdelave algoritma za rešitev takšnega problema, kjer se spreminja samo vektor na desni strani omejitev (s tem pa seveda tudi rešitveni vektor). Ta postopek sloni na stabilitetnih lastnostih linearnega programa in so posebej razčlenjeni pogoji, ki morajo biti izpolnjeni za rešitev.

Če omejeni algoritem kombiniramo s stabilitetno analizo linearnega programa glede na kriterialne koeficiente, lahko rešimo prvotni problem po principu izmenične uporabe točkovne in segmentne optimizacije.