Aplikace matematiky

Jaroslav Morávek

On the complexity of discrete programming problems

Aplikace matematiky, Vol. 14 (1969), No. 6, 442–474

Persistent URL: http://dml.cz/dmlcz/103254

ON THE COMPLEXITY OF DISCRETE PROGRAMMING PROBLEMS*)

JAROSLAV MORÁVEK

(Received April 18, 1968)

I. INTRODUCTION

In this paper we shall be concerned with a general discrete (integer) programming problem, as follows:

To maximize the given function of a discrete variable x

$$f(x, \alpha_1, \alpha_2, \ldots, \alpha_n) \tag{1}$$

subject to the constraints

$$x \in \boldsymbol{X} \tag{2}$$

and

$$g_i(x, \alpha_1, \alpha_2, \ldots, \alpha_n) \leqq 0 \quad (1 \leqq i \leqq n), \tag{3}$$

where the following assumptions are made:

A) $\boldsymbol{X}$ is a finite, nonempty set (the range of the discrete variable x),

B) $\alpha_1, \alpha_2, \ldots, \alpha_n$ are parameters of the problem (1), (2), (3), where $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \in \boldsymbol{A}$, $\boldsymbol{A}$ being a given non-empty set of R^n. (The symbol R^n denotes the set of all ordered n-tuples of real numbers, which will be called n-space in the sequel), and

C) f and g_i are real-valued functions defined on the set $\boldsymbol{X} \times \boldsymbol{A}$.

Remark. Usually we shall have $\boldsymbol{A} = R^n$ or $\boldsymbol{A} = R^{n+}$, where R^{n+} denotes the non-negative cone in R^n.

*) The results of this paper were presented on the 6th International Symposium on Mathematical Programming (August 1967, Princeton University), on the 10th Scientific Colloquy (September 1967, Technische Hochschule Ilmenau, Germany), and on the Conference on Applications of Mathematics to Economics (December 1967, Bucharest University and the Romanian Academy of Sciences, Bucharest).

Example. Let us consider the following discrete programming problem: To maximize the value of a linear form

$$c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n$$

subject to the condition that $x_1, \ldots, x_n$ are integers satisfying the following system of inequalities

$$a_{i1}x_1 + \ldots + a_{in}x_n \leqq b_i \quad (1 \leqq i \leqq n)$$

and

$$0 \leqq x_j \leqq 1 \quad (1 \leqq j \leqq n).$$

This problem can be presented in the form (1), (2), (3) by putting

$$x = (x_1, x_2, \ldots, x_n),$$
$$\boldsymbol{X} = \{(x_1, \ldots, x_n) \mid x_j \in \{0, 1\}\ (1 \leqq j \leqq n)\} = \{0, 1\}^n$$

and by considering real parameters a_{ij}, b_i, and c_j as parameters α_j of problem (1), (2), (3). Functions f and g_i can be introduced in an obvious way.

In the example mentioned above functions f and g_i are linear forms of the parameters (where x is fixed), and at the same time the coefficients of these forms are integers. In fact this is true for any discrete programming problem known to the author. For this reason the following assumption concerning problem (1), (2), (3) will be added to assumptions A), B), C):

D) Functions $f(x, \alpha_1, \ldots, \alpha_n)$ and $g_i(x, \alpha_1, \ldots, \alpha_n)$ can be expressed as follows:

$$f(x, \alpha_1, \alpha_2, \ldots, \alpha_n) = \sum_{j=1}^{n} c_j(x) \cdot \alpha_j, \quad g_i(x, \alpha_1, \ldots, \alpha_n) = \sum_{j=1}^{n} c_j^{(i)}(x) \cdot \alpha_j$$

where coefficients $c_j(x)$ and $c_j^{(i)}(x)$ $(x \in \boldsymbol{X})$ are integers.

A concrete numerical problem can be obtained from problem (1), (2), (3) by substituting numerical values for parameters α_j.

Discrete programming problems are very often discussed in the literature because they occur in many problems of mathematics, operations research, engineering, etc. For the solution of the discrete programming problems many algorithms have been described. Then the essential problem arises to compare different algorithms from the point of view of their efficiency (i.e. processing time and storage requirements, etc.), and that of finding an "optimum" algorithm. Presenting the last problem in this way, we at once face serious "philosophical" difficulties caused by the fact that no general and mathematically rigorous concept of a discrete programming algorithm is available and therefore it is not clear enough how to "measure" the efficiency of such an algorithm.

If we want therefore to study the problem of optimization of discrete programming algorithms by using theoretical methods, we have to find first of all a particular answer to the main question: "What is an algorithm of discrete programming in general?" – In other words it is necessary to introduce a certain class of algorithms.

In this paper a class of algorithms is introduced for the solution of the problem (1), (2), (3), and certain complexity indices of algorithms are defined. At the same time we try that these definitions fulfil the following requirements:

1) The formal definition of an algorithm must be general and natural enough to describe naturally a large number of algorithms practically interesting and efficient (described in the literature and so far imaginable).

2) The definition of complexity (i.e. the measure of efficiency of an algorithm) must be sufficiently related to the required processing time, when realizing an algorithm by means of a computer.

According to the author's opinion the last requirement can be reached by introducing a complexity index in terms of the number of required elementary operations.

Thus in this paper, instead of describing or investigating a concrete algorithm, a very large class of algorithms is investigated. In this sense there exists a relation to the paper [1], where a certain class of algorithms for solving a 0, 1-linear programming problem is investigated. An elementary step of an algorithm in [1] consists in checking whether a given 0, 1-vector is a feasible solution or not. After a set of 0, 1-vectors has been checked, the algorithm finds an optimum vector. The complexity index in [1] is defined as the number of checked vectors, and bounds on the complexity index are derived.

Our approach differs from that in [1] mainly in the concept of complexity. In this paper we make use of the fact that most of discrete programming algorithms require only additive operations addition and subtraction and predicates of comparison of real numbers. (According to the opinion of E. BALAS [2] multiplications and divisions seem not to be natural in discrete programming algorithms, particularly because they cause round-off errors.) Using certain algebraic construction, based on the graph theory, a class of *linear separating algorithms* is defined, where the algorithms use additions, subtractions and comparisons as elementary acts.

A main part of the paper consists of deriving certain lower (pessimistic) bounds on the number of comparisons required by linear separating algorithms for solving certain special discrete programming problems.

In section II problem (1), (2), (3) is generalized in an adequate way, and then the concept of a linear separating algorithm is introduced, and simple properties of this concept are shown. The linear separating algorithm is defined as a trichotomic finite rooted tree with labelled nodes and edges. At the end of section II a theorem is given about the existence of a linear separating algorithm.

In section III estimations of the number of required comparisons concerning following discrete programming problems are derived:

1) Linear programming problems with 0, 1-variables.

2) Polynomial programming problem with 0 and 1 variables.

3) Shortest route problem in certain acyclic networks with labelled directed edges.

4) At the end of section III the complexity of a branching algorithm for finding a 0, 1-solution of a given linear equation is discussed. The discussed problem is related to the so called knap-sack problem.

The most definitive result is obtained in the case of the shortest route problem. It is proved that the Bellman's dynamic programming method yields an algorithm which is optimum in the sense of the number of required comparisons.

In the concluding section IV the obtained results are briefly discussed and possible ways of further development of the ideas are shown.

In the sequel use of several lemmas is made. Proofs of these lemmas are presented in Appendix (V.).

II. GENERAL CONCEPTS

1. Generalization of Problem (1), (2), (3). First of all we are going to show that a certain finite system of subsets of $\boldsymbol{A}$ corresponds to problem (1), (2), (3). Let us put

$$\boldsymbol{A}(x) = \left\{(\alpha_1, \ldots, \alpha_m) \in \boldsymbol{A} \,\middle|\, \begin{array}{l}\text{Problem (1), (2), (3) with } \alpha_1, \ldots, \alpha_n \text{ as values} \\ \text{of the parameters has optimum solution } x\end{array}\right\} =$$

$$= \{(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A} \mid g_i(x, \alpha_1, \ldots, \alpha_n) \leqq 0 \,(1 \leqq i \leqq m)\} \cap$$

$$\cap \bigcap_{x' \in X - \{x\}} \left[\bigcup_{i=1}^{m} \{(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A} \mid g_i(x', \alpha_1, \ldots, \alpha_n) > 0\} \cup\right.$$

$$\left. \cup \{(\alpha_1, \ldots, \alpha_n) \in A \mid f(x', \alpha_1, \ldots, \alpha_n) \leqq f(x, \alpha_1, \ldots, \alpha_n)\}\right]$$

for each $x \in \boldsymbol{X}$, and

$$\boldsymbol{A}(\emptyset) = \left\{(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A} \,\middle|\, \begin{array}{l}\text{Problem (1), (2), (3) with } \alpha_1, \ldots, \alpha_n \text{ as values} \\ \text{of the parameters has no feasible solution}\end{array}\right\} =$$

$$= \bigcap_{x \in X} \bigcup_{i=1}^{m} \{(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A} \mid g_i(x, \alpha_1, \ldots, \alpha_n) > 0\}\,.$$

Making use of assumption D) concerning functions f and g_i we can see that each of the sets $\boldsymbol{A}(x)$ and $\boldsymbol{A}(\emptyset)$ can be expressed as the intersection of $\boldsymbol{A}$ with a union of a finite system of certain convex polyhedral cones (the cones under consideration may be defined both by inequalities $\leqq$ and $<$). The system $\{\boldsymbol{A}(\emptyset)\} \cup \{\boldsymbol{A}(x) \mid x \in \boldsymbol{X}\}$ is covering of $\boldsymbol{A}$, i.e.

$$\boldsymbol{A}(\emptyset) \cup \bigcup_{x \in \boldsymbol{X}} \boldsymbol{A}(x) = \boldsymbol{A}\,.$$

Each algorithm for solving problem (1), (2), (3) must in fact realize in a constructive way the following mapping: To each vector $(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A}$ the algorithm must assign a set of the system $\{\boldsymbol{A}(\emptyset)\} \cup \{\boldsymbol{A}(x) \mid x \in \boldsymbol{X}\}$, containing the given vector $(\alpha_1, \ldots, \alpha_n)$.

Remark. Requiring an algorithm to determine the set of all the optimum solutions of problem (1), (2), (3), we introduce the system of sets $\{\tilde{\boldsymbol{A}}(\boldsymbol{Y}) \mid \boldsymbol{Y} \subset \boldsymbol{X}\}$, where

$$\tilde{\boldsymbol{A}}(\boldsymbol{Y}) = \left\{(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A} \;\middle|\; \begin{array}{l}\boldsymbol{Y} = \{x\} \text{ is the set of all the optimum solutions of problem} \\ (1), (2), (3) \text{ with } \alpha_1, \ldots, \alpha_n \text{ as values of the parameters}\end{array}\right\},$$

instead of the system $\{\boldsymbol{A}(\emptyset)\} \cup \{\boldsymbol{A}(x) \mid x \in \boldsymbol{X}\}$. Then it results immediately $\tilde{\boldsymbol{A}}(\emptyset) = \boldsymbol{A}(\emptyset)$, and the conclusion that the system $\{\tilde{\boldsymbol{A}}(\boldsymbol{Y}) \mid \boldsymbol{Y} \subset \boldsymbol{X}\}$ is disjoint decomposition of $\boldsymbol{A}$.

Now, it is immediately seen that problem (1), (2), (3) is a special case of a more general problem (called Basic Problem):

(BP) Let $\boldsymbol{A}$ be a nonempty set, $\boldsymbol{A} \subset R^n$, and let $\{\boldsymbol{A}_\iota \mid \iota \in I\}$ be a finite system of subsets of $\boldsymbol{A}$, where $\{\boldsymbol{A}_\iota \mid \iota \in I\}$ is a covering of $\boldsymbol{A}$, i.e.

$$\bigcup_{\iota \in I} \boldsymbol{A}_\iota = \boldsymbol{A}.$$

Problem (BP) consists in finding an algorithm, which has to determine such a set $\boldsymbol{A}_\iota$ to each given vector $(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A}$, that $(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A}_\iota$.

As functions f, and g_i are linear forms according to the assumption (D), we make the additional adequate assumption concerning the problem (BP):

Each of the sets $\boldsymbol{A}_\iota$ $(\iota \in I)$ may be expressed as

$$\boldsymbol{A}_\iota = \boldsymbol{A} \cap (\boldsymbol{A}_\iota^{(1)} \cup \ldots \cup \boldsymbol{A}_\iota^{(\varrho)} \cup \ldots \cup \boldsymbol{A}_\iota^{(r_\iota)}),$$

where each set $\boldsymbol{A}_\iota^{(\varrho)}$ can be expressed as

$$\boldsymbol{A}_\iota^{(\varrho)} = \left\{(\alpha_1, \ldots, \alpha_n) \in \boldsymbol{A} \;\middle|\; \begin{array}{l}\sum_{j=1}^{n} b_{\varkappa,j}^{(\iota,\varrho)} \cdot \alpha_j \leqq 0 \quad (\varkappa = 1, 2, \ldots, k_{\iota,\varrho}) \\ \sum_{j=1}^{n} c_{\lambda,j}^{(\iota,\varrho)} \cdot \alpha_j < 0 \quad (\lambda = 1, 2, \ldots, l_{\iota,\varrho})\end{array}\right\},$$

where $b_{\varkappa,j}^{(\iota,\varrho)}$ and $c_{\lambda,j}^{(\iota,\varrho)}$ are integers (i.e. $\boldsymbol{A}_\iota^{(\varrho)}$ are certain convex polyhedral cones, which are in general neither closed nor open, and at the same time the coefficients of the linear homogeneous inequalities determining $\boldsymbol{A}_\iota^{(\varrho)}$ are integers).

The problem (BP) is very general, and it is not restricted to the area of the discrete programming. It covers a very large class of combinatorial problems.

2. Class of Linear Separating Algorithms. The purpose of this paragraph is to describe formally the preparatory considerations of the preceding paragraphs. We

present a formal definition of the Linear Separating algorithm (LS-algorithm), which is to formalize the intuitive concept of an arbitrary algorithm built up only from additions, subtractions, and comparisons, where a comparison is a predicate defined in the set of all ordered pairs of real numbers (x, y), taking on three values:

x is larger than y, x is equal to y, or x is less than y.

Definition. *An LS-algorithm for solving problem* (BP) *is a finite trichotomic rooted tree* $\mathbf{T}$, *having labeled nodes and edges. At the same time, the tree* $\mathbf{T}$ *has the following properties* a–d:

a) *The root* v_0 *has degree* (*i.e. number of incident edges*) 3, *and each node differing from* v_0 *has degree either* 1 *or* 4.

Let us denote sets of nodes having degree 1 or 4 as V_1 or V_4 respectively. The set of all nodes of $\mathbf{T}$ will be denoted by V, thus $V = \{v_0\} \cup V_1 \cup V_4$. It is known that to each node $v_r \in V_1$ there exists a unique sequence of different nodes $\{v_0, v_1, \ldots, v_r\}$ such that the nodes v_{j-1} and v_j are connected by an edge for $j = 1, 2, \ldots, r$. The sequence $\{v_0, v_1, \ldots, v_r\}$ will be called a *branch*, and denoted by $B(v_r)$, i.e.

$$B(v_r) = \{v_0, v_1, \ldots, v_r\} .$$

Further, it is known that a unique partial ordering $\prec$ exists such that $u' \prec u''$ is valid if and only if such a branch $B(v_r) = \{v_0, v_1, \ldots, v_r\}$ exists that $u' = v_\varrho$, and $u'' = v_\sigma$ for a pair of indices ϱ and σ, where $\varrho < \sigma$. Now, each edge of $\mathbf{T}$ can be oriented as follows:

A directed edge *) (u, v) starts from node u, and enters node v if and only if nodes u and v are neighbours, and $u \prec v$ holds. Thus it is clear that from each node $v \in \in V_4 \cup \{v_0\} = V - V_1$ exactly 3 lines start.

b) *With each of the said three lines one of the numbers either* -1, *or* 0, *or* $+1$ *is associated, where each of the numbers* -1, *or* 0, *or* $+1$ *is associated with the considered triple of edges only once.* Thus, each edge (u, v) of $\mathbf{T}$ has been labelled by a number, which will be denoted by sign (u, v).

c) *With each node* $v \in \{v_0\} \cup V_4$ *a linear form* $\mathscr{L}_v(\alpha_1, \alpha_2, \ldots, \alpha_n)$ *of parameters* $\alpha_1, \alpha_2, \ldots$ *and* α_n *is associated, where* $\mathscr{L}_v(\alpha_1, \alpha_2, \ldots, \alpha_n) = \sum_{j=1}^{n} c_j^{(v)} . \alpha_j$, *and where coefficients* $c_1^{(v)}, c_2^{(v)}, \ldots$ *and* $c_n^{(v)}$ *are integers, and where* $(c_1^{(v)}, c_2^{(v)}, \ldots, c_n^{(v)}) \neq (0, 0, \ldots, 0)$.

d) *Each node* $v \in V_1$ *is labelled by a certain index* $\iota \in I$, *such that the following condition is fulfilled:*

If sign $(\mathscr{L}_{v_j}(\alpha_1, \alpha_2, \ldots, \alpha_n)) = \text{sign}\,(v_j, v_{j+1})$ $(0 \leqq j \leqq r - 1)$, where $B(v_r) = \{v_0,$

*) In the sequel we say simply edge.

$v_1, \ldots, v_r\}$, then $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \boldsymbol{A}_\iota$. The definition of the *LS*-algorithm is accomplished.

3. Interpretation of LS-algorithm. The LS-algorithm can be interpreted in this natural way:

i) *First Step of LS-algorithm.* The procedure starts from the root v_0.

ii) *General Step of LS-algorithm.* Let us assume that a node $u \in \{v_0\} \cup V_4$ has been reached in the algorithm. By an application of a finite number of operations $+$ and $-$ the value $\mathscr{L}_v(\alpha_1, \ldots, \alpha_n)$ is determined. As the result of the comparison of $\mathscr{L}_v(\alpha_1, \ldots, \alpha_n)$ and 0 a node v is determined such that v neighbours u, $u \prec v$, and sign $(u, v) = \text{sign } \mathscr{L}_u(\alpha_1, \ldots, \alpha_n)$.

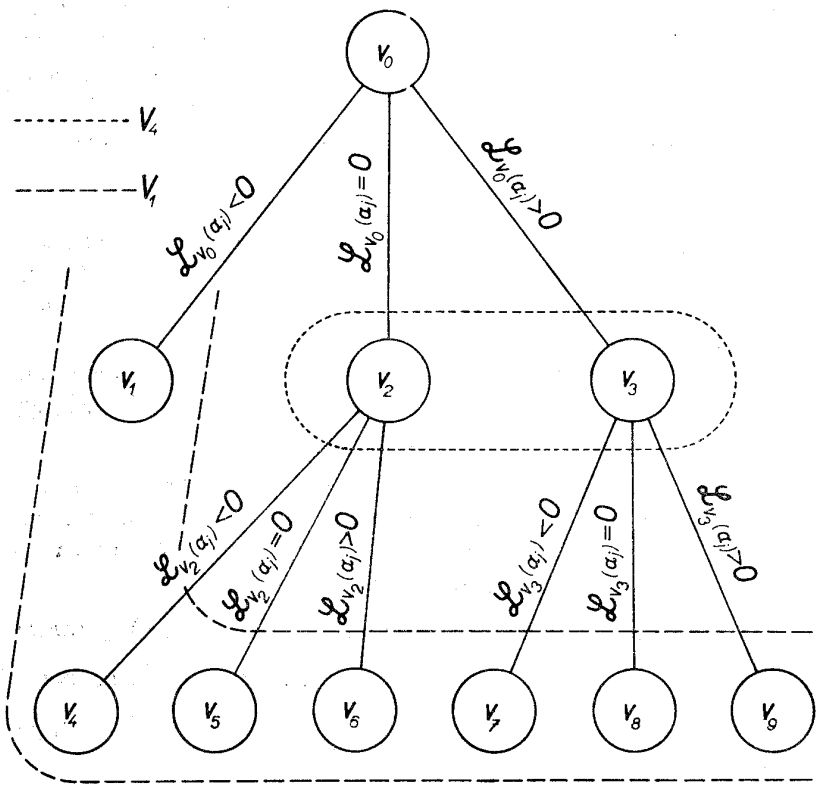

Fig. 1.

iii) *Final Step of LS-algorithm.* At certain stage of the procedure an end node $v \in V_1$ has been reached. The subscript $\iota \in I$ assigned to v according to the definition of LS-algorithm determines the set $\boldsymbol{A}_\iota$ such that $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \boldsymbol{A}_\iota$.

An LS-algorithm can be schematically shown as in figure 1. In our approach we are going to discuss bounds on number of comparisons required by an LS-algorithm. If we wanted, however, to estimate also the number of additive operations, it should be necessary to introduce an alternative tree-algorithm, the tree containing also nodes corresponding to additions and subtractions.

This paragraph is concluded by a remark concerning the used terminology. In an LS-algorithm n-space is successively partitioned (separated) by hyperplanes $\mathscr{L}_u(\alpha_1, \ldots, \alpha_n) = 0$. A similar term was used in threshold logic (see e.g. [3]).

4. Complexity of LS-algorithm. In the preceding paragraphs it was shown that in an LS-algorithm a comparison is associated with each node. Therefore, if the process runs along a branch $B(v_r) = \{v_0, v_1, \ldots, v_r\}$, then the number of required comparisons equals the *length* of the branch, i.e. r. Now, the following definition will be introduced: A node $v \in V_1$ (resp. the corresponding branch $B(v)$) is called *proper*, if a vector of parameters $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in A$ exists such that the algorithm terminates in the node v, when starting from the vector $(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

For the purpose of the following paragraphs we introduce two complexity indices $C_1(\mathbf{T})$ and $C_2(\mathbf{T})$:

1) $C_1(\mathbf{T})$ denotes the length of the longest branch of $\mathbf{T}$.

2) $C_2(\mathbf{T})$ denotes the length of the shortest proper branch of $\mathbf{T}$.

From the latter definition it follows immediately $C_1(\mathbf{T}) \geqq C_2(\mathbf{T})$ for each LS-algorithm $\mathbf{T}$.

We have already noticed that the general idea of LS-algorithm does not exclude the possibility to discuss the complexity of discrete programming in terms of additive operations. In this paper, however, we investigate only comparisons requirements, and we remark that the problem concering additive operations seems to be more difficult. On the other hand, the adequacy of our approach consists, in our opinion, in the following facts:

1) Comparisons usually occur as elementary operations in a computer;

2) Comparisions describe naturally the logical structure (branching) of an algorithm (resp. computational process).

5. Existence of LS-algorithm. A close connection between problem (BP) of paragraph II.1., and an LS-algorithm is shown in following theorem.

Theorem 1. *There exists an LS-algorithm* $\mathbf{T}$ *for solving problem* (BP) *to each problem* (BP). *Especially, to each discrete programming problem* (1), (2), (3) *there exists such an LS-algorithm* $\mathbf{T}_0$ *for solving problem* (1), (2), (3) *that*

$$C_1(\mathbf{T}) \leqq (m + 1) \,.\, \mathrm{card}\,(\boldsymbol{X}) - 1 \,.$$

Proof. First let us remember that each of the sets $\mathbf{A}_\iota$ occurring in problem (BP) can be expressed as

$$\mathbf{A}_\iota = \mathbf{A} \cap \left(\mathbf{A}_\iota^{(1)} \cup \ldots \cup \mathbf{A}_\iota^{(\varrho)} \cup \ldots \cup \mathbf{A}_\iota^{(r_\iota)}\right),$$

where $\mathbf{A}_\iota^{(\varrho)}$ is certain convex polyhedral cone (see II.1.). Let us denote by $\mathfrak{A}$ the system of all hyperplanes facing at least one of the cones $A_\iota^{(\varrho)}$. Let the equations of the hyperplanes of $\mathfrak{A}$ be

$$k_1^{(\sigma)} . \alpha_1 + k_2^{(\sigma)} . \alpha_2 + \ldots + k_n^{(\sigma)} . \alpha_m = 0,$$

where $\sigma = 1, 2, \ldots, R$. Let us notice that coefficients $k_j^{(\sigma)}$ can be assumed to be integers (see the end of paragraph II. 1.)

Now, let $(\delta_1, \delta_2, \ldots, \delta_R)$ denote arbitrary vector, with the coordinates taking on either $+1$, or -1, or 0, and let us put

$$H(\delta_1, \delta_2, \ldots, \delta_R) =$$
$$= \{(\alpha_1, \ldots, \alpha_n) \in \mathbf{A} \mid \operatorname{sign}\left(k_1^{(\sigma)} . \alpha_1 + \ldots + k_n^{(\sigma)} . \alpha_n\right) = \delta_\sigma \quad (1 \leqq \sigma \leqq R)\}.$$

Further, let us denote by $\mathfrak{B}$ the system of all sets $H(\delta_1, \ldots, \delta_R)$, where $\delta_\sigma \in \{-1, 0, 1\}$ for $1 \leqq \sigma \leqq R$. $\mathfrak{B}$ is a disjoint decomposition of $\mathbf{A}$, and at the same time $\mathfrak{B}$ is a refinement of system $\{\mathbf{A}_\iota \mid \iota \in I\}$ in the following sense:

$$\forall \iota \in I \; \forall (\alpha_1, \ldots, \alpha_n) \in \mathbf{A}_\iota \; \exists H(\delta_1, \ldots, \delta_R) \in \mathfrak{B}$$
$$((\alpha_1, \ldots, \alpha_n) \in H(\delta_1, \ldots, \delta_R) \subset \mathbf{A}_\iota).$$

Now we are going to construct an LS-algorithm $\mathbf{T}$ for the solution of the problem (BP) as follows: Let us put

$$V = \{v(\delta_1, \ldots, \delta_\sigma) \mid \delta_1 \in \{0, -1, 1\}, \ldots, \delta_\sigma \in \{0, -1, 1\}; \sigma = 0, 1, \ldots, R\},$$

$v_0 = v(\emptyset)$ for the root, and

$$V_1 = \{v(\delta_1, \ldots, \delta_R) \mid \delta_1 \in \{-1, 0, 1\}, \ldots, \delta_R \in \{-1, 0, 1\}\}.$$

The set of the edges will consist of the ordered pairs

$$(v(\delta_1, \ldots, \delta_\sigma), v(\delta_1, \ldots, \delta_\sigma, \delta_{\sigma+1})),$$

where $\delta_1 \in \{-1, 0, 1\}, \ldots, \delta_{\sigma+1} \in \{-1, 0, 1\}$ $(0 \leqq \sigma \leqq R - 1)$. The node $v(\delta_1, \ldots, \delta_\sigma) \in$ $\in V - V_1$ will be labelled by $k_1^{(\sigma+1)} . \alpha_1 + \ldots + k_n^{(\sigma+1)} . \alpha_n$ $(\sigma = 0, 1, \ldots, R - 1)$, and the edge $(v(\delta_1, \ldots, \delta_\sigma), v(\delta_1, \ldots, \delta_{\sigma+1}))$ by $\operatorname{sign}(v(\delta_1, \ldots, \delta_\sigma), v(\delta_1, \ldots, \delta_{\sigma+1})) =$ $= \operatorname{sign} \delta_{\sigma+1}$, where $\sigma = 0, 1, \ldots, R - 1$. Each end node $v(\delta_1, \ldots, \delta_R) \in V_1$ will be labeled by such an index $\iota \in I$, that

$$H(\delta_1, \ldots, \delta_R) \subset \mathbf{A}_\iota.$$

It is clear that $\mathbf{T}$ is an LS-algorithm for solving problem (BP), and the first half of the theorem is proved.

An LS-algorithm for the last half of the Theorem will be described in an intuitive way. In the said algorithm elements $x \in \boldsymbol{X}$ are successively examined in some chosen order and feasibility conditions (3) are checked. In this part of the algorithm $m \,.$. card $(\boldsymbol{X})$ comparisons are required. Thus a set $\tilde{\boldsymbol{X}}$ of all feasible solutions is generated $(\tilde{\boldsymbol{X}} \subset \boldsymbol{X})$. Comparing values $f(x)$ $(x \in \tilde{\boldsymbol{X}})$, we determine optimum solution x, that is to say one of the optimum solutions. In that final part of the algorithm, required number of comparisons is not larger than card $(\boldsymbol{X}) - 1$. The described trivial algorithm can be obviously represented as an LS-algorithm for solving problem (1), (2), (3), and at the same time the number of required comparisons does not exceed $(m + 1)$. card $(\boldsymbol{X}) - 1$, q.e.d.

III. BOUNDS OF COMPLEXITY OF SPECIAL PROBLEMS

1. Linear Programming Problem with 0 and 1 Variables. 1.1. Auxiliary Problem. The linear programming problem with 0 and 1 variables has been described in the Example of the Introduction.

It is to maximize a function

$$c_1 \,.\, x_1 + c_2 \,.\, x_2 + \ldots + c_n \,.\, x_n \tag{4}$$

of the variables $x_1, \ldots, x_n$, subject to

$$x_j \in \{0, 1\} \tag{5}$$

and

$$\sum_{j=1}^{n} a_{ij} \,.\, x_j \leqq b_i \quad (1 \leqq i \leqq m) \,. \tag{6}$$

As shown in the Example, problem (4), (5), (6) is a special case of problem (1), (2), (3), and therefore an LS-algorithm for solving that problem exists. The purpose of the section III.1, is to derive a lower bound for $C_1(\mathbf{T})$, where $\mathbf{T}$ denotes arbitrary algorithm for solving problem (4), (5), (6).

First, let us introduce an auxiliary problem (P): Real values $a_1, a_2, \ldots, a_n$, and a are given. The problem consists in deciding whether linear equation

$$a_1 \,.\, x_1 + a_2 \,.\, x_2 + \ldots + a_n \,.\, x_n = a \tag{7}$$

has a solution $(x_1, \ldots, x_n)$ satisfying conditions (5).

Problem (P) can be obviously considered as a special case of problem (BP), if putting $I = \{0, 1\}$, and

$$\mathbf{A}_0 = \{(a_1, \ldots, a_n, a) \in R^{n+1} \mid \text{equation (7) has no solution (5)}\},$$

and

$$\mathbf{A}_1 = \{(a_1, \ldots, a_n, a) \in R^{n+1} \mid \text{equation (7) has solution (5)}\}.$$

Problem (4), (5), (6) is in some respect "more difficult" than problem (P), as shown in the following lemma.

Lemma 1. *For each LS-algorithm* $\mathbf{T}$ *for solving* (4), (5), (6) *with* $m \geqq 2$, *such an LS-algorithm* $\tilde{\mathbf{T}}$ *for solving* (P) *exists, that*

$$C_j(\tilde{\mathbf{T}}) \leqq C_j(\mathbf{T}) \quad (j = 1, 2).$$

Proof. Beginning with $\mathbf{T}$, we shall construct an LS-algorithm $\tilde{\mathbf{T}}$ for solving (P) as follows:

1) Put $a_{1j} = a_j$, and $a_{2j} = -a_j$, and $c_j = 0$ for $j = 1, 2, \ldots, n$ in $\mathbf{T}$.

2) Put $b_1 = a$, and $b_2 = -a$, and $a_{ij} = b_i = 0$ for $i = 3, 4, \ldots, m, j = 1, 2, \ldots, n$ in $\mathbf{T}$.

3) If node $v \in V_1$ has been labelled by vector $(x_1, x_2, \ldots, x_n)$, then it is relabelled by 1.

4) If node $v \in V_1$ has been labelled by $\emptyset$ it is relabelled by 0.

5) Performing operations 1)–4), we derive a labeled tree $\mathbf{T}^{(1)}$ from $\mathbf{T}$. Let us assume that the sequence $\mathbf{T}^{(1)}, \mathbf{T}^{(2)}, \ldots, \mathbf{T}^{(s)}$ $(s \geqq 1)$ of trees has been constructed. Following two cases can occur:

a) None of nodes of $\mathbf{T}^{(s)}$ is labeled by a zero linear form. Then $\tilde{\mathbf{T}} = \mathbf{T}^{(s)}$ is the required LS-algorithm for solving (P).

b) There exists a node v in $\mathbf{T}^{(s)}$, which is labeled by a zero form. In this case the following reduction of $\mathbf{T}^{(s)}$ is applied: Let symbols v_{-1}, v_1, u, and w denote nodes of $\mathbf{T}^{(s)}$, which are uniquely determined by the following properties:

α) v_{-1}, v_1, u, and w are adjacent with node v,

β) $v_{-1} \prec v$, $v \prec v_1$, $v \prec u$, and $v \prec w$,

γ) $\operatorname{sign}(v, v_1) = 0$, $\operatorname{sign}(v, u) = 1$, and $\operatorname{sign}(v, w) = -1$.

The said reduction consists in detaining node v together with the incident lines (v_{-1}, v), (v, u), (v, v_1), and (v, w), and in detaining oriented subtrees defined by u, and w as roots. Thereafter nodes v_{-1} and v_1 are connected by a new line (v_{-1}, v_1), which is to be labelled by $\operatorname{sign}(v_{-1}, v_1) = \operatorname{sign}(v_{-1}, v)$. A new tree resulting by the said reduction is denoted by $\mathbf{T}^{(s+1)}$.

Now, it is clear that after a finite number of operations b) case a) takes place, and thus proof of the lemma is accomplished.

1.2. Lower Bound of Complexity of (P).

Theorem 2. *Let* $\mathbf{T}$ *be an LS-algorithm for solving* (P). *Then*

$$C_1(\mathbf{T}) \geqq \frac{(n-1)^2}{2}.$$

Proof. Put

$$B_n = \{(x_1, \ldots, x_n) \mid x_j \in \{0, 1\}\ (j = 1, 2, \ldots, n)\},$$

and

$$R(B) = \left\{(a_1, \ldots, a_n, a) \in R^{n+1} \;\middle|\; \begin{array}{l} \sum_{j=1}^{n} a_j \cdot x_j > a \text{ if } (x_1, \ldots, x_n) \in B \\ \sum_{j=1}^{n} a_j \cdot x_j < a \text{ if } (x_1, \ldots, x_n) \notin B \end{array}\right\}$$

for $B \subset B^n$. Let us denote by $\mathfrak{R}$ the system of all nonempty sets $R(B)$. The following lemma is valid:

Lemma 2.

$$\log_2 (\operatorname{card} (\mathfrak{R})) \geqq \frac{(n-1)^2}{2}.$$

(Proof of this lemma is presented in Appendix V.1.)

Now, let $\mathbf{T}$ be an LS-algorithm for solving problem (P). Let us put

$$C(v) = \left\{(a_1, \ldots, a_n, a) \in R^{n+1} \;\middle|\; \begin{array}{c} \operatorname{sign} \mathscr{L}_{v_j}(a_1, \ldots, a_n, a) = \operatorname{sign} (v_j, v_{j+1}) \\ (0 \leqq j \leqq r-1) \end{array}\right\}$$

where $v = v_r \in V_1$, and where $B(v_r) = \{v_0, v_1, \ldots, v_r\}$ denotes the corresponding branch. Further, let us put

$$\mathfrak{C} = \{C(v) \mid C(v) \neq \emptyset, \operatorname{sign} (v_j, v_{j+1}) \neq 0 \quad (j = 0, 1, \ldots, r-1)\}.$$

(The condition $C(v) \neq 0$ says that v must be a proper node.) The system $\mathfrak{C}$ contains no more than $2^{C_1(\mathbf{T})}$ sets. Now, let us observe that systems $\mathfrak{R}$, and $\mathfrak{C}$ have the following properties:

a) $\dim R(B) = n + 1$ if $R(B) \in \mathfrak{R}$,

b) If $\operatorname{sign} (v_j, v_{j+1}) = 0$ is valid at least for one edge lying on $B(v)$, then $\dim C(v) \leqq \leqq n$,

c) If $R(B) \cap C(v) \neq \emptyset$, where $R(B) \in \mathfrak{R}$, and $C(v) \in \mathfrak{C}$, then $C(v) \subset R(B)$.

Properties a) and b) are obvious, and property c) can be shown as follows: Sup-

posing on the contrary that c) is not true, a vector $(x_1, \ldots, x_n) \in B^n$ must exist such that

$$C(v) \cap \{(a_1, \ldots, a_n, a) \mid a_i x_1 + \ldots + a_n . x_n \eqsim a\} \neq \emptyset ,$$

which is contradiction.

Using properties a), b), and c), we obtain the following property:

d) $\forall R(B) \in \mathfrak{R} \; \exists C(v) \in \mathfrak{C}(C(v) \subset R(B))$.

Making use of property d) and of lemma 2, we obtain

$$2^{C_1(\mathbf{T})} \geqq \text{card}\,(\mathfrak{C}) \geqq \text{card}\,(\mathfrak{R}) \geqq 2^{(n-1)^2/2} ,$$

which completes the proof.

1.3. Bound of Complexity of Problem (4), (5), (6). Following theorem follows immediately by combining Lemma 1, and Theorem 2.

Theorem 3. *If* $\mathbf{T}$ *is arbitrary LS-algorithm for solving problem* (4), (5), (6), *then*

$$C_1(\mathbf{T}) \geqq \frac{(n-1)^2}{2} ,$$

i.e. a_{ij}, b_i, *and* c_j *can be chosen so that algorithm* $\mathbf{T}$ *requires at least* $(n-1)^2/2$ *comparisons to solve the corresponding problem.*

Remark. To prove theorems 2 and 3 a simple cardinality (entropy) method was used. The main idea of the proof consists in using a bound for card $(\mathfrak{R})$ (see lemma 2). It can be simply shown that card $(\mathfrak{R})$ equals number of all threshold functions depending at most on n boolean variables (see e.g. [4]), and it is was shown ([4], [5]) that

$$\log_2 (\text{card}\,(\mathfrak{R})) \lesssim n^2 (n \to \infty) .$$

Thus it is clear that the method of the proof of theorems 2 and 3 cannot yield any sufficiently better bound. To obtain a more definitive bound (depending also on m) it will be probably necessary to use some more sophiticated ideas of combinatorial geometry.

1.4. Case of p-Value Variables. The results of theorems 2 and 3 can be generalized to a more general case, where discrete variables x_j take on p different values $(p \geqq 2)$, i.e.

$$x_j \in \{0, 1, \ldots, p-1\} \quad (j = 1, 2, \ldots, n) . \tag{8}$$

In this more general case following theorems hold:

Theorem 2_p. *If $\mathbf{T}_n$ is an LS-algorithm for solving problem (P_p), where (P_p) is introduced in the same way as problem (P), but each variable is assumed to fulfil (8), then*

$$C_1(\mathbf{T}_n) \gtrsim \tfrac{1}{2} \cdot \log_2 p \cdot n^2 \quad (n \to \infty).$$

Theorem 3_p. *If $\mathbf{T}_n$ is an LS-algorithm for solving problem (4), (6), (8), then*

$$C_1(\mathbf{T}_n) \gtrsim \tfrac{1}{2} \cdot \log_2 p \cdot n^2 \quad (n \to \infty).$$

Theorems 2_p and 3_p can be proved analogously as theorems 2 and 3 but a lower bound of number of p-value threshold functions is to be used [3].

2. Complexity of Integer Polynomial Programming. In this paragraph we show that an analogous cardinality method can be used in the case where the functions f and g_i occurring in (1), (2), (3) are polynomials of x_j. Let us consider following integer programming problem:

To maximize function

$$\sum_{\substack{1 \geqq j_1, \ldots, j_n \geqq 0 \\ j_1 + \ldots + j_n \leqq k}} c(j_1, \ldots, j_n) \cdot x_1^{j_1} \ldots x_n^{j_n}, \tag{9}$$

subject to

$$x_j \in \{0, 1\} \quad (j = 1, 2, \ldots, n) \tag{5}$$

and

$$\sum_{\substack{1 \geqq j_1, \ldots, j_n \geqq 0 \\ j_1 + \ldots + j_n \leqq k}} a^{(i)}(j_1, \ldots, j_n) \cdot x_1^{j_1}, \ldots, x_n^{j_n} \leqq b^{(i)} \tag{10}$$

$(i = 1, 2, \ldots, m)$, where $a^{(i)}(j_1, \ldots, j_n)$, $b^{(i)}$, and $c(j_1, \ldots, j_n)$ are parameters of the problem, which can take on arbitrary real values. To derive a lower bound of the complexity we introduce an auxiliary problem in an analogous way as in paragraph III. 1.1.

We have to decide, whether equation

$$\sum_{\substack{1 \geqq j_1, \ldots, j_n \geqq 0 \\ j_1 + \ldots + j_n \leqq k}} a(j_1, \ldots, j_n) \cdot x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} = a \tag{11}$$

has or has not a solution $(x_1, \ldots, x_n)$, where

$$x_j \in \{0, 1\} \tag{5}$$

and where $a(j_1, \ldots, j_n)$ and a denote given real parameters.

The following theorem holds:

Theorem 4. *If* $\mathbf{T}(n)$ *is an LS-algorithm for solving problem* (9), (5), *and* (10), *where* $m \geqq 2$, *then*

$$C_1(\mathbf{T}(n)) \gtrsim n^{k+1} \cdot \frac{k^k}{(k+1)^{k+1} \cdot k!} \quad \text{if} \quad n \to \infty \text{ and } k = k(n) = o(\sqrt{n}) .$$

Theorem 4 can be proved analogously as theorem 3 of III.1.3., but the following lemma 3 is to be used instead of lemma 2.

Lemma 3. *The system of hyperplanes*

$$\{(a(j_1, \ldots, j_n), a) \mid \sum_{\substack{1 \geqq j_1, \ldots, j_n \geqq 0 \\ j_1 + \ldots + j_n \leqq k}} a(j_1, \ldots, j_n) \cdot x_1^{j_1}, \ldots \; x_n^{j_n} = a\} ,$$

where $x_j \in \{0, 1\}$ $(j = 1, \ldots, n)$ *divides the space of the points with coordinates* $a(j_1, \ldots, j_n)$ *and* a (*the dimension of this space equals* $\binom{k+n}{n} + 1$) *into a system* $\mathfrak{S}$ *of nonempty open polyhedral cones. Let us put* $M(n) = \operatorname{card}(\mathfrak{S})$. *Then*

$$\log_2 M(n) \gtrsim \frac{n^{k+1} \cdot k^k}{(k+1)^{k+1} \cdot k!} , \quad \text{if} \quad 1 \leqq k = k(n) = o(\sqrt{n}), \text{ and } n \to \infty .$$

Remark. The asymptotic inequality $\gtrsim$ is used only to simplify corresponding expressions in Theorem 4 and Lemma 3.

3. Shortest Route Problem. 3.1. Formulation of Shortest Route Problem. Now, we shall be concerned with the problem of determining a shortest route in an oriented acyclic graph with labelled edges, $\boldsymbol{G} = (\mathfrak{N}, \boldsymbol{E}, \lambda)$, where $\mathfrak{N}$ denotes the set of nodes, $\boldsymbol{E}$ the set of edges, and λ denotes the labelling of edges, and where the graph has the following special structure:

1) The set of nodes $\mathfrak{N}$ is partitioned into disjoint sets $\mathfrak{N}_1, \ldots, \mathfrak{N}_r$, i.e.

$$\mathfrak{N} = \mathfrak{N}_1 \cup \mathfrak{N}_2 \cup \ldots . \cup \mathfrak{N}_n ,$$

and

$$\mathfrak{N}_i \cap \mathfrak{N}_j = \emptyset \quad \text{if} \quad i \neq j ,$$

where

$$\mathfrak{N}_j = \{N(j, 1), N(j, 2), \ldots, N(j, a_j)\}$$

$(j = 1, 2, \ldots, n)$, where $n, a_1, a_2, \ldots, a_n$ are positive integers, and $n \geqq 2$.

2) The set of edges is

$$\boldsymbol{E} = \left\{(N(j, k), N(j+1, l)) \;\middle|\; \begin{matrix} k = 1, 2, \ldots, a_j; \; l = 1, 2, \ldots, a_{j+1}; \\ j = 1, 2, \ldots, n-1 \end{matrix}\right\} .$$

3) λ is the labeling of the edges, i.e. a real-valued function defined on $\boldsymbol{E}$. The label of edge $(N(j, k), N(j+1, l))$ is denoted by $\lambda(j, k, l)$ (we make use of an obvious fact that edge $(N(j, k), N(j+1, l))$ is uniquely determined by a triple of indices (j, k, l)). Graph $\boldsymbol{G}$, where $n = 3$, $a_1 = 2$, $a_2 = 3$, and $a_3 = 2$ is shown in the figure 2.

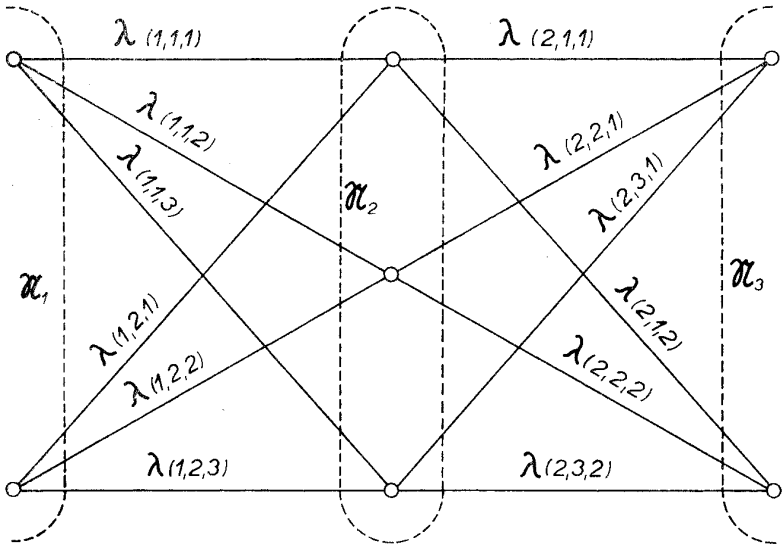

Fig. 2.

Following the main idea of this paper, the graph as a combinatorial structure is assumed to be fixed, but labels $\lambda(j, k, l)$ as parameters of the problem vary arbitrarily $(-\infty < \lambda(j, k, l) < +\infty)$.

Let us consider the set of all routes in the graph $\boldsymbol{G}$, which start from a node of $\mathfrak{N}_1$. Each of these routes is in a one-to-one correspondence to a sequence $\{k_j\}_{j=1}^{r}$, where $2 \leqq r \leqq n$, and where $1 \leqq k_j \leqq a_j$ $(j = 1, 2, \ldots, r)$. Let us denote the route determined by the sequence $\{k_j\}_{j=1}^{r}$ as $R(k_1, \ldots, k_r)$. With each route $R(k_1, \ldots, k_r)$ we associate the number

$$L(k_1, \ldots, k_r) \stackrel{\text{df.}}{=} \sum_{j=1}^{r-1} \lambda(j, k_j, k_{j+1}),$$

called *length* of the route. In the shortest route problem we have to determine a route $R(k_1, \ldots, k_n)$ starting from a node of $\mathfrak{N}_1$ and entering a node of $\mathfrak{N}_n$ such that $L(k_1, \ldots, k_n)$ reaches a minimum value.

3.2. Dynamic Programming Algorithm. To solve the shortest route problem the well-known dynamic programming method can be used. By this method a shortest

route is determined, starting from a node of $\mathfrak{N}_1$ and entering a node $N(j, l)$, for each given node $N(j, l) \in \mathfrak{N} - \mathfrak{N}_1$. The algorithm proceeds recursively as follows:

A) To each node $N(2, l) \in \mathfrak{N}_2$ $(l = 1, 2, \ldots, a_2)$ a node $N(1, k^{(l)}) \in \mathfrak{N}_2$ $(k^{(l)} = 1, 2, \ldots, a_1)$ is determined such that

$$\lambda(1, k^{(l)}, l) \leqq \lambda(1, k, l)$$

for $k = 1, 2, \ldots, a_1$. Then $R(k^{(l)}, l)$ is a shortest route entering $N(2, l)$.

B) Let us suppose that in the algorithm a shortest route entering node $N(j, l)$ has been determined for each $N(j, l) \in \mathfrak{N}_j$, where j is fixed and $1 \leqq l \leqq a_j$. Let us denote this route by $R(k_1^{(j,l)}, \ldots, k_{j-1}^{(j,l)}, l)$. Now, we determine a shortest route starting from $\mathfrak{N}_1$ and entering $N(j + 1, m)$. The corresponding shortest route $R(k_1^{(j,l^{(m)})}, \ldots, k_{j-1}^{(j,l^{(m)})}, l^{(m)}, m)$ is determined by choosing such an index $l^{(m)}$ $(1 \leqq l^{(m)} \leqq a_j)$ that

$$L(k_1^{(j,l^{(m)})}, \ldots, k_{j-1}^{(j,l^{(m)})}, l^{(m)}, m) \leqq L(k_1^{(j,l)}, \ldots, k_{j-1}^{(j,l)}, l, m),$$

where $l = 1, 2, \ldots, a_j$.

C) Having determined a shortest route $R(k_1^{(l)}, \ldots, k_{n-1}^{(l)}, l)$ entering $N(n, l)$ for each node $N(n, l) \in \mathfrak{N}_n$ $(l = 1, 2, \ldots, a_n)$, we determine an index $\hat{l}$ $(1 \leqq \hat{l} \leqq a_n)$ such that

$$L(k_1^{(\hat{l})}, \ldots, k_{n-1}^{(\hat{l})}, \hat{l}) \leqq L(k_1^{(l)}, \ldots, k_{n-1}^{(l)}, l)$$

for $l = 1, 2, \ldots, a_n$. Route $R(k_1^{(\hat{l})}, \ldots, k_n^{(\hat{l})}, \hat{l})$ is a solution of the shortest route problem.

3.3. Complexity of Dynamic Programming Algorithm. First let us notice that the shortest route problem can be treated as a special case of general problem (1), (2), (3), if we put

$$\boldsymbol{A} = \{(\lambda(j, k, l)) \mid -\infty < \lambda(j, k, l) < +\infty\}$$

($\boldsymbol{A}$ is the space of dimension $\sum_{j=1}^{n-1} a_j \cdot a_{j+1}$),

$$\boldsymbol{X} = \{R(k_1, k_2, \ldots, k_n) \mid 1 \leqq k_j \leqq a_j \quad (j = 1, 2, \ldots, n)\}$$

(thus $\boldsymbol{X}$ denotes the set of all routes of $\boldsymbol{G}$ starting from $\mathfrak{N}_1$ and entering $\mathfrak{N}_n$)

$$x = R(k_1, k_2, \ldots, k_n), \quad \text{and} \quad f(x) = L(k_1, k_2, \ldots, k_n).$$

Restrictions $g_i(x) \leqq 0$ do not occur in this problem, i.e. $m = 0$ in the terms of problem (1), (2), (3).

The dynamic programming algorithm can be obviously represented as an LS-algorithm for solving the shortest route problem. We are going to show that the said algorithm requires

$$\sum_{j=1}^{n-1} a_j \,.\, a_{j+1} - \sum_{j=2}^{n-1} a_j - 1 \tag{12}$$

comparisons for each choice of parameters λ. In fact, at stage (A) of the algorithm it is necessary to use $(a_1 - 1)\, a_1$ comparisons, and $(a_j - 1)\, a_{j+1}$ comparisons $(2 \leqq j \leqq n-1)$ at each stage (B). Thus the number of comparisons required at stages (A) and (B) equals $\sum_{j=1}^{n-1} (a_j - 1) \,.\, a_{j+1}$. Adding $a_n - 1$ comparisons required at stage (C) to the last expression, we obtain expression (12).

3.4. Optimality of Dynamic Programming Algorithm.

It is shown in the following theorem that the dynamic programming algorithm is optimum with respect to the class of all LS-algorithms for solving the shortest route problem as to the number of required comparisons.

Theorem 5. *If* $\mathbf{T}$ *is an LS-algorithm for solving the shortest route problem, then*

$$C_2(\mathbf{T}) \geqq \sum_{j=1}^{n-1} a_j \,.\, a_{j+1} - \sum_{j=2}^{n-1} a_j - 1 \,. \tag{13}$$

Proof. Let $v = v_r \in V_1$ be a proper node and $B(v) = \{v_0, v_1, \ldots, v_r\}$ the corresponding branch, and let node v be labelled by a route $R(k_1^{(0)}, k_2^{(0)}, \ldots, k_n^{(0)})$ (according to the definition of an LS-algorithm). Let us introduce sets

$$K_1 = \{(\lambda(j, k, l)) \mid \operatorname{sign} \mathscr{L}_{v_\varrho}(\lambda(j, k, l)) = \operatorname{sign}(v_\varrho, v_{\varrho+1}) \quad (\varrho = 0, 1, \ldots, r-1)\} \,,$$

where

$$\mathscr{L}_v(\lambda(j, k, l)) = \sum_{j=1}^{n-1} \sum_{k=1}^{a_j} \sum_{l=1}^{a_{j+1}} c^{(v)}(j, k, l) \,.\, \lambda(j, k, l)$$

(thus K_1 is the set of all edge labellings for which algorithm $\mathbf{T}$ finishes in node v), and

$$K_2 = \{(\lambda(j, k, l)) \mid L(k_1^{(0)}, \ldots, k_n^{(0)}) \leqq L(k_1, \ldots, k_n) \quad \text{for each route } R(k_1, \ldots, k_n)\} =$$

$$= \{(\lambda(j, k, l)) \mid \sum_{j=1}^{n-1} \lambda(j, k_j^{(0)}, k_{j+1}^{(0)}) \leqq \sum_{j=1}^{n-1} \lambda(j, k_j, k_{j+1}) \text{ for each route } R(k_1, \ldots, k_n)\} \,.$$

(Thus K_2 is the set of all labellings such that $R(k_1^{(0)}, \ldots, k_n^{(0)})$ is a shortest route in $\mathbf{G}$).

Sets K_1 and K_2 are convex polyhedral cones (K_1 does not contain in general its faces) in the $\sum_{j=1}^{n-1} a_j \,.\, a_{j+1}$-dimensional space, and $K_1 \subset K_2$. From the last relation we obtain

$$K_1^* \supset K_2^*, \tag{14}$$

where K_j^* denotes a polar cone associated with K_j $(j = 1, 2)$ (see e.g. [6]). To continue the proof we are going to use following two lemmas concerning the form of K_1^* and K_2^*. (The proofs of these lemmas are presented in Appendix 3.)

Lemma 4. *Set K_1^* consists of all points $(\mu(j, k, l))$ the coordinates of which can be expressed as*:

$$\mu(j, k, l) = \sum_{\varrho=0}^{r-1} c^{(v_\varrho)}(j, k, l) \,.\, [\mu_\varrho(1 - \operatorname{sign}^2(v_\varrho, v_{\varrho+1})) - \mu_\varrho^2 \,.\, \operatorname{sign}(v_\varrho, v_{\varrho+1})]$$
$$(k = 1, 2, \ldots, a_j,\ l = 1, 2, \ldots, a_{j+1},\ j = 1, 2, \ldots, n-1),$$

where $-\infty < \mu_\varrho < +\infty$ (r denotes the length of branch $B(v)$).

Lemma 5. *Set K_2^* consists of all points $(\mu(j, k, l))$, the coordinates of which can be expressed as follows*:

$$\mu(j, k, l) = -\sum_{k_1=1}^{a_1} \sum_{k_2=1}^{a_2} \ldots \sum_{k_{j-1}=1}^{a_{j-1}} \sum_{k_{j+2}=1}^{a_{j+2}} \ldots \sum_{k_n=1}^{a_n} \tau(k_1, k_2, \ldots, k_{j-1}, k, l, k_{j+2}, \ldots, k_n)$$

if

$$(k, l) \neq (k_j^{(0)}, k_{j+1}^{(0)})$$

and

$$\mu(j, k, l) = \sum_{k_1=1}^{a_1} \ldots \sum_{k_{j-1}=1}^{a_{j-1}} \sum_{\substack{1 \leqq k_j \leqq a_j \\ 1 \leqq k_{j+1} \leqq a_{j+1} \\ (k_j, k_{j+1}) \neq (k_j^{(0)}, k_{j+1}^{(0)})}} \sum_{k_{j+2}=1}^{a_{j+2}} \ldots \sum_{k_n=1}^{a_n}$$
$$\tau(k_1, \ldots, k_{j-1}, k_j, k_{j+1}, k_{j+2}, \ldots, k_n)$$

if

$$(k, l) = (k_j^{(0)}, k_{j+1}^{(0)}),$$

where

$$\tau(k_1, k_2, \ldots, k_n) \geqq 0.$$

Now let us continue the proof. It follows from lemma 4

$$\dim K_1^* \leqq r \tag{15}$$

and

$$\dim K_2^* \leqq \dim K_1^* \tag{16}$$

follows from (14). To accomplish the proof it is sufficient to prove inequality

$$\dim K_2^* \geqq \sum_{j=1}^{n-1} a_j \cdot a_{j+1} - \sum_{j=2}^{n-1} a_j - 1 . \tag{17}$$

(In fact, inequality (13) follows by combining (15), (16), (17), and by using relation $C_2(\mathbf{T}) = \min \{r \mid v_r \in V_1, v_r \text{ is proper node}\}$.) To prove inequality (17) we notice that points of K_2^* are linear combinations (with nonnegative coefficients) of rows of certain matrix, as shown in lemma 5. Thus it is sufficient to prove that the rank of the matrix under consideration is not less than

$$\sum_{j=1}^{n-1} a_j \cdot a_{j+1} - \sum_{j=2}^{n-1} a_j - 1 .$$

The columns of the matrix do correspond to triples (j, k, l), where $k = 1, 2, \ldots, a_j$, $l = 1, \ldots, a_{j+1}, j = 1, 2, \ldots, n-1$. Now let us cancel the columns which do correspond to the triples $(j, k_j^{(0)}, k_{j+1})$ $(k_{j+1} = 1, 2, \ldots, a_{j+1}, j = 1, 2, \ldots, n-2)$, and that corresponding to triple $(n-1, k_{n-1}^{(0)}, k_n^{(0)})$. In this way the matrix has been reduced to a new matrix, having

$$\sum_{j=1}^{n-1} a_j \cdot a_{j+1} - \sum_{j=2}^{n-1} a_j - 1$$

columns. But the columns of the reduced matrix are linearly independent, as shown in the following lemma:

Lemma 6. *System of linear homogeneous equations*

$$\sum_{j=1}^{n-1} \sigma(j, k_j^{(0)}, k_{j+1}^{(0)}) - \sum_{j=1}^{n-1} \sigma(j, k_j, k_{j+1}) = 0 \tag{18}$$

$(k_j = 1, 2, \ldots, a_j; j = 1, 2, \ldots, n)$, *and*

$$\sigma(j, k_j^{(0)}, k_{j+1}) = 0 \tag{19}$$

$(k_{j+1} = 1, 2, \ldots, a_{j+1}; j = 1, 2, \ldots, n-2)$, *and*

$$\sigma(n-1, k_{n-1}^{(0)}, k_n^{(0)}) = 0 \tag{20}$$

has only trivial solution $\sigma(j, k, l) \equiv 0$.

Remark. Equations (18) correspond to linear combinations of the original matrix, and equations (19) and (20) correspond to the cancelled columns.

The proof of lemma 6 is presented in Appendix 4. Lemma 6 has accomplished the proof of Theorem 5.

Remark on the proof of theorem 5. From the fact that the lower bound obtained in theorem 5 is exact (it can be realized by the dynamic programming algorithm) we obtain

$$\dim K_2^* = \sum_{j=1}^{n-1} a_j \,.\, a_{j+1} - \sum_{j=2}^{n-1} a_j - 1 \,.$$

Remark. Putting $n = 2$, $a_1 = 1$, and $a_2 = m$ in Theorem 5, we obtain after appropriate changes of notation the following statement: The number of comparisons required in arbitrary LS-algorithm for determining a minimum element in a given sequence of m real numbers is not less than $m - 1$.

4. Complexity of Special Algorithms for Solving Problem (P). In the foregoing paragraphs bounds of the complexity of certain special discrete programming problems were derived. Lower bounds obtained in paragraphs III.1. and III.2. are rather low yet. On the other hand, their improvement seems to be very difficult. The difficulty of the problem is obviously caused by the fact that a very general class of algorithms is considered. In order to derive better lower bounds, it is possible to simplify the original problem as follows: We restrict appropriately the class of all LS-algorithms, at the same time we require that the restricted class contain some of the well-known, resp. interesting algorithms. As an example of the mentioned approach we shall examine a class of certain special LS-algorithms for solving problem (P) (see paragraph III.1.1.). First, a branching algorithm for solving problem (P) is described.

Let us put $a_i^+ = \sum_{j=1}^{i} \max(a_j, 0)$, and $a_i^- = \sum_{j=1}^{i} \min(a_j, 0)$ for $i = 1, 2, \ldots, n$. In the algorithm a sequence of sets $E_n, E_{n-1}, \ldots, E_j, \ldots$ is processed, where E_j contains equations of the form

$$a_1 \,.\, x_1 + \ldots + a_j \,.\, x_j = a - a_{j+1} \,.\, \sigma_{j+1} - \ldots - a_n \,.\, \sigma_n \,,$$

where $x_1, \ldots, x_j$ denote the unknowns, and $\sigma_{j+1}, \ldots, \sigma_n$ parameters, and where $x_\iota \in \in \{0, 1\}$, $\sigma_\varkappa \in \{0, 1\}$. At the same time vector $(\sigma_{j+1}, \ldots, \sigma_n)$ is uniquely determined by an equation of E_j, but in general this is no mapping onto the set of all vectors $(\sigma_{j+1}, \ldots, \sigma_n)$, where $\sigma_\varkappa \in \{0, 1\}$. The sequence of sets $E_n, E_{n-1}, \ldots$ is generated in the algorithm until the evidence is obtained, whether the equation

$$a_1 \,.\, x_1 + \ldots + a_n \,.\, x_n = a \tag{7}$$

has a solution $(x_1, \ldots, x_n)$ or not, where $x_j \in \{0, 1\}$ for $j = 1, 2, \ldots, n$.

The algorithm proceeds as follows:

a) a_i^+ and a_i^- are computed for $i = 1, 2, \ldots, n$. a_n^+ and a_n^- are compared with a. We distinguish three possible cases:

a1) It holds either $a_n^+ < a$ or $a_n^- > a$. Then equation (7) has no solution (5) and the procedure stops.

a2) It holds either $a_n^+ = a$ or $a_n^- = a$. Then equation (7) has a solution (5), and the procedure stops.

a3) It holds $a_n^- < a < a_n^+$. Putting $E_n = \{a_1 \,.\, x_1 + \ldots + a_n \,.\, x_n = a\}$, we pass to the next step b).

b) Let us suppose a sequence $E_n, E_{n-1}, \ldots, E_i$ has been constructed. Let us put F_i for the system of all equations

$$a_1 \,.\, x_1 + \ldots + a_{i-1} \,.\, x_{i-1} = a - a_i \,.\, \sigma_i - \ldots - a_n \,.\, \sigma_n$$

(unknowns $x_1, \ldots, x_{i-1}$), such that equation

$$a_1 \,.\, x_1 + \ldots + a_i \,.\, x_i = a - a_{i+1} \,.\, \sigma_{i+1} - \ldots - a_n \,.\, \sigma_n$$

belongs to set E_i, and $\sigma_i \in \{0, 1\}$. Now, values a_{i-1}^- and a_{i-1}^+ are compared with $a - a_{i+1} \,.\, \sigma_{i+1} - \ldots - a_n \,.\, \sigma_n$ and three possible cases are distinguished:

b1) It holds either $a_{i-1}^- > a - a_i \,.\, \sigma_i - \ldots - a_n \,.\, \sigma_n$ or $a_{i-1}^+ < a - a_i \,.\, \sigma_i - \ldots - a_n \,.\, \sigma_n$ for all equations of F_i. Then equation (7) has no solution (5) and the procedure is over.

b2) There exists at least one equation of F_i such that

$$a_{i-1}^- = a - a_i \,.\, \sigma_i - \ldots - a_n \,.\, \sigma_n$$

or

$$a_{i-1}^+ = a - a_i \,.\, \sigma_i - \ldots - a_n \,.\, \sigma_n \,.$$

Then equation (7) has a solution (5) and thus the procedure stops.

b3) It holds neither (b1) nor (b2). In such a case there exist equations of F_i such that

$$a_{i-1}^- < a - a_i \,.\, \sigma_i - \ldots - a_n \,.\, \sigma_n < a_{i-1}^+ \,. \tag{21}$$

Let us put E_{i-1} for the set of all equations of F_i which satisfy relations (21), and pass at the beginning of stage (b).

Thus the description of the algorithm is accomplished.

5. Complexity of Branching Algorithm. It is easy to verify that the algorithm of the paragraph above can be represented as an LS-algorithm for solving problem (P). Let us notice that the only comparisons occurring in the algorithm are
either

$$a_i \gtreqless 0 \tag{22}$$

or

$$\sum_{j=1}^{n} a_j \cdot x_j - a \gtreqless 0 \quad (x_j \in \{0, 1\}) . \tag{23}$$

In the following theorem there is derived a lower bound of the complexity of each algorithm which requires only comparisons (22) or (23).

Theorem 6. *Let* $\mathbf{T}$ *be an LS-algorithm for solving problem* (P), *where the only comparisons occurring in* $\mathbf{T}$ *are either* (22) *or* (23). *Then*

$$C_1(\mathbf{T}) \geqq \left(\left[\frac{n}{2}\right]\right) + \left(\left[\frac{n}{2}\right] + 1\right).$$

Proof. Let us put

$$B_0 = \left\{(x_1, \ldots, x_n) \in B^n \;\middle|\; \sum_{j=1}^{n} x_j > \left[\frac{n}{2}\right]\right\},$$

and

$$R(B_0) = \left\{(a_1, \ldots, a_n, a) \in R^{n+1} \;\middle|\; \sum_{j=1}^{n} a_j \cdot x_j \begin{cases} > a & \text{if } (x_1, \ldots, x_n) \in B_0 \\ < a & \text{if } (x_1, \ldots, x_n) \notin B_0 \end{cases}\right\}$$

(see notation of paragraph III.1.2.). It is easy to verify that $R(B_0) \neq \emptyset$, and that a node $v \in V_1$ exists in $\mathbf{T}$ such that (notation $C(v)$ was introduced in III. 1.2.)

1) $C(v) \subset R(B_0)$; 2) $\dim C(v) = n + 1$,

and 3) There exists a point $(a_1, \ldots, a_n, a)$ in $C(v)$ such that $a_j > 0$ for $j = 1, 2, \ldots, n$, and $a > 0$.

As in algorithm $\mathbf{T}$ only comparisons (22) or (23) occur, set $C(v)$ consists of all points $(a_1, \ldots, a_n, a)$, which satisfy system of inequalities

$$a_j > 0 \quad \text{if} \quad j \in J ,$$

$$a_1 \cdot x_1 + \ldots + a_n \cdot x_n > a \quad \text{if} \quad (x_1, \ldots, x_n) \in K ,$$

and

$$a_1 \cdot x_1 + \ldots + a_n \cdot x_n < a \quad \text{if} \quad (x_1, \ldots, x_n) \in L ,$$

where $J \subset \{1, 2, \ldots, n\}$, $K \subset B^n$, $L \subset B^n$ and where $K \cap L = \emptyset$. Furthermore sets K and L must satisfy relations

$$K \subset B_0 \quad \text{and} \quad L \cap B_0 = \emptyset . \tag{24}$$

To accomplish the proof it is sufficient to establish inclusions

$$K \supset \left\{(x_1, \ldots, x_n) \in B^n \;\middle|\; \sum_{j=1}^{n} x_j = \left[\frac{n}{2}\right] + 1\right\}, \tag{25}$$

$$L \supset \left\{(x_1, \ldots, x_n) \in B^n \;\middle|\; \sum_{j=1}^{n} x_j = \left[\frac{n}{2}\right]\right\}. \tag{26}$$

We are going to prove only inclusion (26) because the proof of (25) is quite analogous. Thus, suppose $(x_1^{(0)}, \ldots, x_n^{(0)}) \notin L$ and $\sum_{j=1}^{n} x_j^{(0)} = [n/2]$ for a 0, 1-vector $(x_1^{(0)}, \ldots, x_n^{(0)})$. Put

$$a_j^{(0)} = \begin{cases} 1 + \dfrac{1}{n} & \text{if} \quad x_j^{(0)} = 1 \\ 1 & \text{if} \quad x_j^{(0)} = 0 \end{cases}$$

for $j = 1, 2, \ldots, n$, and

$$a^{(0)} = \left(1 + \frac{1}{n}\right)\left[\frac{n}{2}\right].$$

We are going to show that

$$(a_1^{(0)}, \ldots, a_n^{(0)}, a^{(0)}) \in C(v) - R(B_0), \tag{27}$$

which is a contradiction. First notice that $\sum_{j=1}^{n} a_j^{(0)} \cdot x_j^{(0)} = a^{(0)}$, so that $(a_1^{(0)}, \ldots, a_n^{(0)}, a^{(0)}) \notin R(B_0)$. If $(x_1, \ldots, x_n) \in K$, then $(x_1, \ldots, x_n) \in B_0$ because of (24), thus $\sum_{j=1}^{n} x_j \geqq [n/2] + 1$. It follows from the last inequality

$$\sum_{j=1}^{n} a_j^{(0)} \cdot x_j \geqq \left[\frac{n}{2}\right] + 1 + \frac{1}{n} > a^{(0)}.$$

On the other hand, if $(x_1, \ldots, x_n) \in L$ then $(x_1, \ldots, x_n) \neq (x_1^{(0)}, \ldots, x_n^{(0)})$, and $(x_1, \ldots, x_n) \notin B_0$ because of (24), so that

$$\sum_{j=1}^{n} x_j \leqq \left[\frac{n}{2}\right].$$

It follows from the last inequality

$$\sum_{j=1}^{n} a_j^{(0)} \cdot x_j \leqq \left[\frac{n}{2}\right]\left(1 + \frac{1}{n}\right) - \frac{1}{n} < a^{(0)}.$$

At last, it holds $a_j^{(0)} > 0$ $(j = 1, \ldots, n)$, thus relation (27) has been established. The proof of Theorem 6 is accomplished.

Remark. A similar result has been established in [1] in a different way.

IV. CONCLUDING REMARKS

In this paper an attempt was made to present a general theory of the complexity in the discrete programming. It should be noticed that the described approach is not the only possible, and that in this approach a number of open problems exists, e.g.

1) to obtain bounds of the number of required additions and subtractions,

2) to investigate the class of algorithms which include also the multiplication and the division.

As to the concept of the complexity, let us notice that it would be of a considerable interest to obtain nontrivial bounds for a new complexity index $C_E(\mathbf{T})$, where $C_E(\mathbf{T})$ denotes the average value of the length of a branch in the sense of a probability measure defined for subsets of $\boldsymbol{A}$.

The mathematical methods for obtaining lower bounds can be summarized as follows:

a) cardinality (entropy) technique for lower bounds in theorems 2, 3, and 4.
b) use of the concept of dimension of convex polyhedral cones in theorem 5, and
c) simple geometrical considerations in theorem 6.

In order to develop further efficient proof techniques it should be necessary to investigate a series of special discrete programming problems. e.g. the travelling-salesman problem, network flows problems, assignment problems, etc. At the same time we think that it would be preferable to start our discussion with some special classes of LS-algorithms which could be chosen so as to include e.g. algorithms of [2], [7]–[10] respectively.

V. APPENDIX

1. Proof of Lemma 2. We are going to construct a system $\mathfrak{R}_n$ containing not less than $2^{(n-1)^2/2}$ nonempty and mutually different cones $R(B)$. Each of the cones of $\mathfrak{R}_n$ will be represented by a vector which belongs to this cone. Thus system $\mathfrak{R}_n$ will be determined by a set $\mathfrak{W}_n$ containing $N(n)$ $(n+1)$-dimensional vectors, i.e.

$$\mathfrak{W}_n = \{(a_1(i, n), \ldots, a_n(i, n), 1) \mid 1 \leqq i \leqq N(n)\}$$

($N(n)$ is determined in the proof.)

System of sets $\{\mathfrak{W}_n\}_{n=1}^{+\infty}$ is constructed by means of an induction as follows:

(i) Put $a_1(1, 1) = -1$, and $a_1(2, 1) = 2$ (the corresponding sets are $R(\emptyset)$ and $R(\{1\})$.).

(ii) Induction step. Let us suppose that set $\mathfrak{W}_n$ has been constructed. We can suppose that following condition is fulfilled for each vector of $\mathfrak{W}_n$:

If $(x_1, \ldots, x_n) \in B^n$ and $(x'_1, \ldots, x'_n) \in B^n$ and $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$, then

$$a_1(i, n) \cdot x_1 + \ldots + a_n(i, n) \cdot x_n \neq a_1(i, n) \cdot x'_1 + \ldots + a_n(i, n) \cdot x'_n . \tag{28}$$

(Relation (28) holds for $n = 1$, and in general it is easy to guarantee it because cones $R(B)$ are open.)

Now, let us put

$$r(i, n, x_1, \ldots, x_n) = 1 - (a_1(i, n) \cdot x_1 + \ldots + a_n(i, n) \cdot x_n) .$$

It follows from (28) that 2^n numbers $r(i, n, x_1, \ldots, x_n)$ (where i and n are fixed) may be ordered with respect to their magnitude. Let their order be

$$r_0(i, n) < r_1(i, n) < \ldots < r_{2^n}(i, n) < r_{2^n+1}(i, n) ,$$

where a more simple notation has been applied, and $r_0(i, n) = -\infty$ or $r_0(i, n) = +\infty$ have been joined to the beginning or to the end of the above sequence respectively. Now, let us choose $2^n + 1$ numbers $p(i, n, l)$ according to conditions

$$r_{l-1}(i, n) < p(i, n, l) < r_l(i, n) .$$

Set $\mathfrak{W}_{n+1}$ will be constructed as follows: $\mathfrak{W}_{n+1}$ contains all $(n + 2)$-dimensional vectors

$$(a_1(i, n), a_2(i, n), \ldots, a_n(i, n), p(i, n, l), 1) ,$$

where $i = 1, 2, \ldots, N(n)$, $l = 1, 2, \ldots, 2^n + 1$.

Thus set $\mathfrak{W}_{n+1}$ contains $N(n + 1) = (2^n + 1) \cdot N(n)$ vectors. The last recurrent formula together with $N(1) = 2$ accomplishes the proof.

Remark. It can be simply shown that there exists a one-to-one correspondence between cones of $\mathfrak{R}_n$ and threshold functions of n variables (see e.g. [4]). This lower bound has been established in terms of the number of threshold functions in [12], [14] and [3].

2. Proof of Lemma 3. Similarly as in the proof of lemma 2 we are going to construct a subsystem $\mathfrak{S}_0$ of $\mathfrak{S}$, where each cone of $\mathfrak{S}$ is represented by a vector

$$(\hat{a}(j_1, \ldots, j_n), \hat{a}) , \tag{29}$$

contained in it. Vectors (29) are constructed as follows:

a) We put $\hat{a} = \frac{1}{4}$ for each vector (29). Variables $x_1, \ldots, x_n$ are divided into two disjoint groups, the first group containing $x_1, \ldots, x_{[n/(k+1)]}$, and the second $x_{[n/(k+1)]+1}, \ldots, x_n$.

The remaining coordinates of vectors (29) are chosen as follows:

b) First let us put

$$\hat{a}(1, 0, \ldots, 0) = 2^0 ,$$

$$\hat{a}(0, \ldots, 0, \overset{j}{1}, 0, \ldots, 0) = 2^{j-1} ,$$

and

$$\hat{a}(0, \ldots, 0, \overset{[n/(k+1)]}{1}, 0, \ldots, 0) = 2^{[n/(k+1)]-1}$$

for each vector (29).

c) Coordinates $\hat{a}(j_1, \ldots, j_n)$ with $j_1 = \ldots = j_{[n/(k+1)]} = 0$ and $0 \leqq j_r \leqq 1$ $(r = [n/(k+1)] + 1, \ldots, n)$, and $j_{[n/(k+1)]-1} + \ldots + j_n = k$ are chosen mutually independent of each other from the set of numbers

$$\{\tfrac{1}{2}, -1 + \tfrac{1}{2}, -2 + \tfrac{1}{2}, \ldots, -2^{[n/(k+1)]} - 2 + \tfrac{1}{2}, -2^{[n/(k+1)]} - 1 + \tfrac{1}{2}\} .$$

d) At last the remaining coordinates $\hat{a}(j_1, \ldots, j_n)$ of vectors (29) are to be put zeroes.

It can be seen easily that each vector (29) satisfying a)–d) is contained in a cone of $\mathfrak{S}$, and different vectors belong to different cones. The number of all vectors (29) equals the number of all ways in which coordinates c) can be chosen. Each of coordinates c) ranges over the set containing $2^{[n/(k+1)]}$ values, and the number of coordinates c) equals

$$\binom{n - \left[\frac{n}{k+1}\right]}{k} .$$

Thus the number of vectors (29) equals

$$2^{[n/(k+1)] \cdot \binom{n-[n/(k+1)]}{k}} .$$

Now, making use of assumption $1 \leqq k = o(\sqrt{n})$ $(n \to \infty)$, we obtain

$$\log_2 M(n) \geqq \left[\frac{n}{k+1}\right] \binom{n - \left[\frac{n}{k+1}\right]}{k} \sim \frac{n}{k+1} \cdot \frac{\left(\frac{kn}{k+1}\right)^k}{k!} = n^{k+1} \cdot \frac{k^k}{(k+1)^{k+1} \cdot k!} .$$ *)

The proof of lemma 3 has been accomplished.

*) It holds

$$\frac{\left(\frac{kn}{k+1}\right)^k}{k!} = \frac{\left(n - \frac{n}{k+1}\right)^k}{k!} \geqq \binom{n - \left[\frac{n}{k+1}\right]}{k} \geqq \frac{\left(n - \frac{n}{k+1} - k\right)^k}{k!} \geqq$$

3. Polyhedral Convex Cones. Polar Cones. In this appendix we aply the usual matrix and vector notation. Let $\mathfrak{A}$ be a set of column vectors in R^n, and let us put

$$\mathfrak{A}^* = \{\mathbf{y} \in R^n \mid \mathbf{y}^T . \mathbf{x} \leqq 0 \quad \text{if} \quad \mathbf{x} \in \mathfrak{A}\} .$$

It can be simply shown that:

1) $\mathfrak{A}^*$ is convex cone in R^n (the last fact enables to call $\mathfrak{A}^*$ by the term *polar cone*),

2) $\mathfrak{A}^*$ is closed set in R^n,

3) It holds $\overline{\mathfrak{A}}^* = \mathfrak{A}^*$ ($\overline{\mathfrak{A}}$ denotes the closure of $\mathfrak{A}$),

4) $\mathfrak{A}_1 \subset \mathfrak{A}_2 \subset R^n \Rightarrow \mathfrak{A}_1^* \supset \mathfrak{A}_2^*$.

Now, let $\mathfrak{A}$ be a set of all solutions of a system of linear inequalities, i.e.

$$\mathfrak{A} = \{\mathbf{x} \in R^n \mid \mathbf{A} . \mathbf{x} \leqq \mathbf{0}\} . \tag{30}$$

In this case the following theorem on the representation of $\mathfrak{A}^*$ holds (see e. g. [6]):

Theorem (Farkas). *If* $\mathfrak{A}$ *is defined by* (30), *then*

$$\mathfrak{A}^* = \{\mathbf{y} \in R^n \mid \textit{where } \mathbf{y}^T = \boldsymbol{\lambda}^T . \mathbf{A} \textit{ for } \boldsymbol{\lambda} \geqq \mathbf{0}\} .$$

The Farkas's theorem is related to the case when $\mathfrak{A}$ is a set of all solutions of a system of linear inequalities with signs $\leqq$. In the other case, if some of signs $\leqq$ are replaced by sharp inequalities $<$, the following modified theorem holds:

Proposition 1. *If* $\mathfrak{A} = \{\mathbf{x} \in R^n \mid \mathbf{A}_1 . \mathbf{x} \leqq \mathbf{0}, \mathbf{A}_2 . \mathbf{x} < \mathbf{0}\} \neq \emptyset$ *then*

$$\mathfrak{A}^* = \{\mathbf{y} \in R^n \mid \mathbf{y} = \boldsymbol{\lambda}_1^T . \mathbf{A}_1 + \boldsymbol{\lambda}_2^T . \mathbf{A}_2 \textit{ where } \boldsymbol{\lambda}_j \geqq \mathbf{0} \quad (j = 1, 2,)\} .$$

Proof. The statement follows from property 3) and from relation $\overline{\mathfrak{A}} = \mathfrak{A}_0$, where

$$\mathfrak{A}_0 = \{\mathbf{x} \in R^n \mid \mathbf{A}_j . \mathbf{x} \leqq \mathbf{0} \quad (j = 1, 2)\} .$$

Relation $\overline{\mathfrak{A}} = \mathfrak{A}_0$ can be proved as follows:

a) It holds $\mathfrak{A} \subset \mathfrak{A}_0$, and $\overline{\mathfrak{A}}_0 = \mathfrak{A}_0$, thus $\overline{\mathfrak{A}} \subset \mathfrak{A}_0$.

$$\geqq \frac{\left(\frac{kn}{k+1}\right)^k \left(1 - \frac{k+1}{n}\right)^k}{k!} = \frac{\left(\frac{kn}{k+1}\right)^k}{k!} \left[\left(1 - \frac{1}{\frac{n}{k+1}}\right)^{n/(k+1)}\right]^{k(k+1)/n} \sim \frac{\left(\frac{kn}{k+1}\right)^k}{k!} ,$$

if $k = o(\sqrt{n})$, $n \to \infty$.

b) Let $\mathbf{x}_0 \in \mathfrak{A}_0$, i.e. $\mathbf{A}_j \,.\, \mathbf{x}_0 \leqq \mathbf{0}$ $(j = 1, 2)$. According to the assumption of the proposition, there exists $\mathbf{y}_0 \in \mathfrak{A}$, and thus it holds $\mathbf{A}_1 \,.\, \mathbf{y}_0 \leqq \mathbf{0}$ and $\mathbf{A}_2 \,.\, \mathbf{y}_0 < \mathbf{0}$ for $\mathbf{y}_0$. Put $\mathbf{x}_n = \mathbf{x}_0 + (1/n) \,.\, \mathbf{y}_0$ $(n = 1, 2, \ldots)$. It holds $\mathbf{x}_n \in \mathfrak{A}$ $(n = 1, 2, \ldots)$, and $\mathbf{x}_n \to \mathbf{x}_0$ if $n \to +\infty$. Thus $\mathbf{x}_0 \in \overline{\mathfrak{A}}$, and the proof is accomplished.

In following proposition a case is investigated when the set $\mathfrak{A}$ is defined by a system of linear constraints, where the set of constraints may be partitioned into two disjoint groups, the first group containing linear equations and the second linear inequalities having form $<$.

Proposition 2. *Let*

$$\mathfrak{A} = \{\mathbf{x} \in R^n \mid \mathbf{A}_1 \,.\, \mathbf{x} = \mathbf{0}, \mathbf{A}_2 \,.\, \mathbf{x} < \mathbf{0}\} \neq 0$$

Then

$$\mathfrak{A}^* = \{\mathbf{y} \in R^n \mid \mathbf{y} = \boldsymbol{\lambda}_1^{\mathsf{T}} \,.\, \mathbf{A}_1 + \boldsymbol{\lambda}_2^{\mathsf{T}} \,.\, \mathbf{A}_2 \text{ where } (\textit{only!}) \; \boldsymbol{\lambda}_2 \geqq \mathbf{0}\} \,.$$

Proof. Set $\mathfrak{A}^*$ can be expressed as

$$\mathfrak{A}^* = \left\{\mathbf{x} \in R^n \;\middle|\; \begin{pmatrix} \mathbf{A}_1 \\ -\mathbf{A}_1 \end{pmatrix} \,.\, \mathbf{x} \leqq \mathbf{0}, \mathbf{A}_2 \,.\, \mathbf{x} < \mathbf{0}\right\},$$

and proposition 1 is to be applied.

Proposition 2 can be formulated in an equivalent way:

Proposition 3. *Let $\mathfrak{A} \neq \emptyset$ be the set of all vectors $\mathbf{x} \in R^n$ satisfying system of conditions* $\operatorname{sign}(\mathbf{a}_j^{\mathsf{T}} \,.\, \mathbf{x}) = \delta_j$ $(j = 1, 2, \ldots, n)$, *where δ_j are given integers* $(\delta_j \in \in \{-1, 0, 1\})$. *Then*

$$\mathfrak{A}^* = \left\{\mathbf{y} \in R^n \;\middle|\; \mathbf{y} = \sum_{j=1}^{n} \lambda_j \,.\, \mathbf{a}_j, \text{ where } \lambda_j \begin{cases} \geqq 0 & \text{if } \delta_j = -1 \\ \leqq 0 & \text{if } \delta_j = 1 \\ \text{arbitrary,} & \text{if } \delta_j = 0 \end{cases}\right\},$$

i.e.

$$\mathfrak{A}^* = \{\mathbf{y} \in R^n \mid \mathbf{y} = \sum_{j=1}^{n} [-\delta_j \,.\, \mu_j^2 + \mu_j \,.\, (1 - \delta_j^2)] \,.\, \mathbf{a}_j, -\infty < \mu_j < +\infty\} \,.$$

Lemma 4 follows immediately from Proposition 3, where subscript ϱ denotes a row and triple (j, k, l) a column of the matrix under consideration.

Lemma 5 follows immediately from the Farkas's theorem. It is only necessary to notice that a row of the matrix under consideration is labelled by n-tuple $(k_1, \ldots, k_n)$ (i.e. it corresponds uniquely to a route of graph $\mathbf{G}$), and a column is labelled by triple (j, k, l) (i.e. it corresponds uniquely to an edge of graph $\mathbf{G}$).

4. Proof of Lemma 6. We are going to prove lemma 6 by an induction method:

(i) First, let us show $\sigma(n-1, k, l) = 0$ if $1 \leqq k \leqq a_{n-1}$, and $1 \leqq l \leqq a_n$. According to equation (20) it is sufficient to investigate triples $(n-1, k, l)$ with $(k, l) \neq (k_{n-1}, k_n)$. Let be $(n-1, k, l)$ such a triple and let us choose an equation of system (18) which corresponds to

$$k_1 = k_1^{(0)}, \ldots, k_{n-2} = k_{n-2}^{(0)}, k_{n-1} = k, \quad k_n = l\,.$$

The said equation is

$$\sigma(n-2, k_{n-2}^{(0)}, k_{n-1}^{(0)}) + \sigma(n-1, k_{n-1}^{(0)}, k_n^{(0)}) - \sigma(n-2, k_{n-2}^{(0)}, k) - \\ - \sigma(n-1, k, l) = 0\,.$$

But from the last equation and from (19) and (20) it follows $\sigma(n-1, k, l) = 0$.

(ii) Let us assume that

$$\sigma(i, \varkappa_i, \lambda_i) = 0 \quad (1 \leqq \varkappa_i \leqq a_i, 1 \leqq \lambda_i \leqq a_{i+1}, i = j+1, j+2, \ldots, n-1) \tag{31}$$

has been established for some j $(1 \leqq j \leqq n-2)$. We are going to show

$$\sigma(j, k, l) = 0 \quad (1 \leqq k \leqq a_j, 1 \leqq l \leqq a_{j+1})\,.$$

According to equations (19) it is sufficient to establish the last fact only for triples (j, k, l) with $k \neq k_j^{(0)}$. Let be (j, k, l) such a triple and let us choose an equation of system (18) such that

$$k_1 = k_1^{(0)}, \ldots, k_{j-1} = k_{j-1}^{(0)}, k_j = k, k_{j+1} = l\,,$$

(Indices $k_{j+2}, \ldots, k_n$ can be chosen arbitrarily.) Making use of relations (31) we obtain equation

$$\sigma(j-1, k_{j-1}^{(0)}, k_j^{(0)}) + \sigma(j, k_j^{(0)}, k_{j+1}^{(0)}) - \sigma(j-1, k_{j-1}^{(0)}, k) - \sigma(j, k, l) = 0$$

if $j \geqq 2$, and equation

$$\sigma(1, k_1^{(0)}, k_2^{(0)}) - \sigma(1, k, l) = 0$$

if $j = 1$. In both cases $\sigma(j, k, l) = 0$ follows from (19). The proof of lemma 6 is accomplished.

References

[1] *Коробков, В. К.:* О некоторых целочисленных задачах линейного программирования, Проблемы кибернетики, том 14, Москва 1965.
[2] *Balas, E.:* An Additive Algorithm for Solving Linear Programs with 0—1 Variables, Opns. Res. 13, 517—549, 1965.
[3] *Bloch, M.* and *Morávek, J.:* Bounds of the number of threshold functions, Information Processing Machines, Praha 1967.
[4] *Winder, R. O.:* Bounds of threshold gate realizability, TRNS IEEE EC — 12, Oct. 63.
[5] *Нечипорук, Е. И.:* О синтезе схем из пороговых элементов. Проблемы кибернетики, том 11, Москва 1964.
[6] *Goldman, A. J., Tucker, A. W.:* Polyhedral Convex Cones, Linear Inequalities and Related Systems, Princeton 1956.
[7] *Ford, L. R., Fulkerson, D. R.:* Flows in Networks, Princeton 1962.
[8] *Bellman, R. E.:* Dynamic Programming Treatment of the Travelling Salesman Problem, J. Assoc. for Comp. Mach. 9, 61—63 (1962).
[9] *Little, J. D. C., Murty, K. G., Sweeney, D. W.,* and *Karel, C.:* An Algorithm for the Travelling Salesman Problem, Opns. Res. 11, 972—989, 1963.
[10] *Vlach M.:* Řešení dopravního problému metodou větvení, Ekonomicko-matematický obzor, 2, N. 4, 1966.
[11] *Yajima, S., Ibaraki, T.:* A lower Bound of the Number of Threshold Functions, IEEE, EC Dec. 1965, Vol. 14, N. 6.
[12] *Morávek, J.:* О некоторых оценках для пороговых функций. Term paper, Leningrad University, 1963.

Souhrn

O SLOŽITOSTI PROBLÉMŮ DISKRÉTNÍHO PROGRAMOVÁNÍ

JAROSLAV MORÁVEK

Tato práce je příspěvkem k obecné teorii problémů diskrétního (celočíselného) programování, speciálně se v ní zkoumá teoretickými prostředky složitost takových problémů.

V úvodní části práce je zformulován jistý velmi obecný typ problému diskrétního programování:

Maximalizovat funkci

(1) $$f(x, \alpha_1, \alpha_2, \ldots, \alpha_n)$$

diskrétní proměnné x, na množině určené omezeními

(2) $$x \in \boldsymbol{X}$$

a

(3) $$g_i(x, \alpha_1, \alpha_2, \ldots, \alpha_n) \leqq 0 \quad (i = 1, 2, \ldots, m).$$

Přitom dále předpokládáme, že

A) $\boldsymbol{X}$ je daná neprázdná konečná množina (obor diskrétní proměnné x),

B) $\alpha_1, \alpha_2, \ldots, \alpha_n$ označují parametry problému (1), (2), (3), přičemž $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \in \boldsymbol{A}$, kde $\boldsymbol{A}$ je daná neprázdná množina prostoru R^n (R^n označuje množinu všech uspořádaných n-tic reálných čísel)

C) Funkce $f(x, \alpha_1, \ldots, \alpha_n)$ a $g_i(x, \alpha_1, \ldots, \alpha_n)$ se dají vyjádřit ve tvaru

$$f(x, \alpha_1, \ldots, \alpha_n) = \sum_{j=1}^{n} c_j(x) \,.\, \alpha_j$$

a

$$g_i(x, \alpha_1, \ldots, \alpha_n) = \sum_{j=1}^{n} c_j^{(i)}(x) \,.\, \alpha_j \,,$$

kde $c_j(x)$ a $c_j^{(i)}(x)$ jsou celá čísla pro $x \in \boldsymbol{X}$.

Konkrétní (t.j. numerický) problém ze třídy problémů (1), (2), (3) je určen dosazením číselných hodnot na místo parametrů $\alpha_1, \alpha_2, \ldots, \alpha_n$.

V obecné části práce II definujeme s použitím jisté algebraické konstrukce spočívající na teorii grafů třídu tzv. lineárně separujících algoritmů (LS-algoritmus) pro řešení problému (1), (2), (3), resp. pro řešení jistého ještě obecnějšího problému identifikace jistých kuželů v R^n. Pojem LS-algoritmu formalizuje intuitivní pojem algoritmu diskrétního programování, který používá jakožto elementárních aktů pouze operací sčítání, odčítání a predikátu srovnání dvou reálných čísel.

V práci jsou zavedeny dva různé indexy složitosti: První je definován jako maximální a druhý jako minimální počet srovnání, která se mohou vyskytnout při použití algoritmu.

Obecná část práce je zakončena větou o existenci LS-algoritmu (věta 1). Jádrem práce je část III, ve které je uvedeno několik vět o počtu srovnání nutných k řešení následujících problémů diskrétního programování:

1) Úloha celočíselného lineárního programování, speciálně úloha lineárního programování s proměnnými 0 a 1.

2) Úloha polynomiálního programování s proměnnými 0 a 1 (účelová funkce a funkce vystupující v omezeních jsou polynomy diskrétních proměnných).

3) Úloha o nejkratší cestě v jistém hranově ohodnoceném grafu.

4) Nakonec se zkoumá složitost jisté třídy algoritmů (obsahující jeden známý algoritmus typu větvení) pro rozhodnutí, zdali daná lineární rovnice pro n dvouhodnotových neznámých má řešení. Poslední úloha souvisí s tzv. „knap sack" problémem.

Zmíníme se o metodách získání odhadů pro jednotlivé problémy diskrétního programování. V případech 1) a 2) bylo použito mohutnostní (entropijní) metody blízké např. k úvahám, pomocí nichž se provádí odhad minimálního počtu vážení nutných k nalezení falešné mince. V případě 1) je dolní odhad získán zkoumáním

počtu konvexních polyedrických kuželů, na které rozdělí n-rozměrný prostor systém všech nadrovin určených lineárně nezávislými n-ticemi bodů n-rozměrné krychle. Poznamenejme, že problém enumerace systému těchto kuželů je isomorfní s problémem určení počtu všech prahových funkcí algebry logiky, závisejících na n proměnných.

Při získání odhadu v případě úlohy o nejkratší cestě se vychází z následujícího intuitivně zřejmého faktu: Jestliže jeden konvexní polyedrický kužel je částí druhého konvexního polyedrického konvexního kuželu, přičemž počet stěn posledního kuželu není větší než dimense prostoru, potom počet stěn prvního kuželu není menší než počet stěn druhého kuželu. Tato geometricky intuitivní představa je přitom upřesněna v terminologii polárních kuželů a používá se Farkasova lemmatu o vyjádření polárního kuželu.

K odhadu složitosti problému 4) bylo nutné určit počet stěn jistého konvexního polyedrického kuželu.

Nejúplnější výsledek se podařilo získat v případě úlohy o nejkratší cestě v grafu. Je dokázáno, že aplikací metody dynamického programování lze získat algoritmus pro řešení této úlohy, který je optimální ve smyslu minimálního počtu použitých srovnání.

Na závěr lze poznamenat, že popsaný přístup umožňuje z jednotného hlediska diskutovat značně širokou třídu problémů nejen diskrétního programování, ale vůbec kombinatorické a diskrétní matematiky.

Author's address: RNDr. *Jaroslav Morávek*, CSc., MÚ ČSAV, Praha 1, Žitná 25.