Kybernetika

Jaroslav Morávek

A localization problem in geometry and complexity of discrete programming

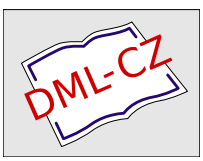
Kybernetika, Vol. 8 (1972), No. 6, (498)--516

Persistent URL: http://dml.cz/dmlcz/124660

Terms of use:

© Institute of Information Theory and Automation AS CR, 1972

Institute of Mathematics of the Academy of Sciences of the Czech Republic provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This paper has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://project.dml.cz

KYBERNETIKA — VOLUME 8 (1972), NUMBER 6

A Localization Problem in Geometry and Complexity of Discrete Programming*

JAROSLAV MORÁVEK

The algorithmical complexity of following problem is investigated: Given an arbitrary point of the Euclidean space, we want to determine whether it belongs to a given convex polyhedral set. An application to the theory of discrete programming is discussed.

1. INTRODUCTION

In this paper we shall use some basic concepts from the theory of convex polyhedral sets and the theory of directed graphs. The reader who is not familiar with these subjects may consult the many existing reference books, e.g. [1] or [2] (for convex polyhedral sets) and [3] or [4] (for graphs).

Let $\boldsymbol{E}^n$ denote the n-dimensional Euclidean space, and let C be a *convex polyhedral set* (which will be referred to in this paper as CPS) in $\boldsymbol{E}^n$, i.e. a nonempty set which can be expressed as

$$C = P_1 \cap P_2 \cap \ldots \cap P_f , \tag{1}$$

where $P_1, P_2, \ldots, P_f$ are closed halfspaces of $\boldsymbol{E}^n$, and f is a nonnegative integer. (If $f = 0$ we put $C = \boldsymbol{E}^n$, thus $\boldsymbol{E}^n$ is also a CPS.) In addition we assume in this paper that any CPS has dimension n, and that the integer f occuring in (1) is chosen minimal. This minimal f (which depends only on C) will be denoted by $\mathfrak{f}(C)$ and called *number of faces* of C.

We shall deal with the algorithmical solution of the following problem: For an arbitrary point $\boldsymbol{x} \in \boldsymbol{E}^n$ we want to determine whether $\boldsymbol{x} \in C$ or $\boldsymbol{x} \notin C$. This problem will be called *problem of localization of points of* $\boldsymbol{E}^n$ *with respect to* C, briefly *C-localization problem.*

* This paper was presented at the Summer school "Combinatorial Structures and Graph Theory" organized in Zlatá Idka by the Association of Slovak Mathematicians, May 1971.

The class of algorithms which will be allowed for the solution of C-localization problem can be characterized intuitively by the requirement that each of their elementary steps (elementary operations) consists in determining the relative location of $\boldsymbol{x}$ with respect to a certain oriented hyperplane prescribed by the algorithm, i.e. answering the following question: Either $\boldsymbol{x}$ belongs to the positive open halfspace determined by the hyperplane, or $\boldsymbol{x}$ belongs to the corresponding non-oriented hyperplane, or $\boldsymbol{x}$ belongs to the corresponding negative open halfspace.

The described elementary step will be called *localization of $\boldsymbol{x}$ with respect to a (given) hyperplane*, briefly *hyperplane localization* (which will be referred to in this paper as HL). The algorithms will be called *localization algorithms for C*, briefly *localization algorithms* (which will be referred to in this paper as LA) if there will be no danger of confusion.

An LA for C performs a series of HL's where the choice of the next HL depends on the result of the previous HL's. After the algorithm has finished its work, the point $\boldsymbol{x}$ is "located" in a set S (i.e. $\boldsymbol{x} \in S$) which is the intersection of a finite system of open halfspaces and/or hyperplanes of $\boldsymbol{E}^n$. Now for S it must hold either $S \subset C$ or $S \subset \boldsymbol{E}^n - C$; in the first case the C-localization problem is answered by "yes" in the second by "no".

The number of HL's occuring in an LA depends on the choice of $\boldsymbol{x}$. As *complexity index* (measure of complexity) of LA we introduce the maximum of all these numbers over all $\boldsymbol{x} \in \boldsymbol{E}^n$. Our main result consists in the derivation of a certain lower bound for the complexity index of any LA for C.

In Section 2 a formal definition of LA is presented, the purpose of which is to substitute the intuitive concept of LA by an adequate but precise mathematical concept. The formal definition of complexity index is also presented, and the following problem is proposed: Given a CPS C, it is to determine an LA for C, having minimum value of its complexity index.

In Section 3 the localization problem for an arbitrary convex polygon in $\boldsymbol{E}^2$ is investigated. In this special case we can indeed determine an LA with the minimum complexity index. The results of Section 3 illustrate the general concepts introduced in Section 2, and are used to construct certain convex polyhedra requiring "almost minimum" number of HL's.

In Section 4 we obtain a lower bound for the number of HL's required for the solution of the C-localization problem. This lower bound depends on the number of extreme points of C.

In Section 5 a countable sequence of convex polyhedra is constructed, and it is shown that the lower bound of Section 4 is asymptotically the best possible.

An analogous lower bound for the number of HL's of Section 4, but depending on the number of extreme halflines of C, can also be obtained. The formulation of the corresponding results is presented in Section 6 without proofs.

In the concluding Section 7 an application of the localization problem to the

 theory of algorithmical complexity in the discrete programming is discussed by using as an illustration the well-known travelling-salesman problem.

2. LOCALIZATION ALGORITHMS

In Section 1 the concept of LA was introduced intuitively by an algorithm using as elementary operations hyperplane localizations. This consists in the determination of the location of the considered point $\boldsymbol{x} \in \boldsymbol{E}^n$ with respect to an oriented hyperplane. An *oriented hyperplane*, by definition, is a hyperplane together with a prescription, according to which one of the open halfspaces, faced by the hyperplane, is called negative and the opposite positive.

Given an oriented hyperplane H we denote by

H^+ – the corresponding positive open halfspace

H^- – the corresponding negative open halfspace

H^0 – the set of all points lying on the corresponding non-oriented hyperplane.

From the formal point of view we are going to identify an LA for C with a certain trichotomic tree. The root and inner vertices of which are assigned to certain oriented hyperplanes, and to each of its output vertices there is assigned either set C or $\boldsymbol{E}^n - C$. The tree corresponds intuitively to a flow-chart of an algorithm where:

1. The work of the algorithm starts from the root.

2. The calculation is running along a branch which corresponds to a realization of the algorithm and which is uniquely determined by the choice of $\boldsymbol{x}$.

3. After a finite number of HL's the calculation finishes in an output vertex. If C (respectively $\boldsymbol{E}^n - C$) is assigned to this output vertex then $\boldsymbol{x} \in C$ (respectively $\boldsymbol{x} \in \boldsymbol{E}^n - C$).

After these preparatory considerations we present a formal definition of LA for C. Our formal concept of LA is related to the concept of linear-separating algorithm discussed in [5] and [6]. Let us consider a finite directed rooted tree with the set of vertices V, the root denoted by v_0, and the set of directed edges E. Further we assume that the tree satisfies following conditions:

a) Either three or no directed edge start from each vertex $\mathsf{v} \in \mathsf{V}$.

b) Exactly one directed edge leads into each vertex $\mathsf{v} \in \mathsf{V} - \{\mathsf{v}_0\}$, and there is no directed edge leading into v_0.

The finite, directed, rooted tree with properties a) and b) will be called a *trichotomic tree (with the root* v_0) and denoted by (V, E). A vertex $\mathsf{v} \in \mathsf{V}$ will be called an *output vertex* of (V, E) iff there is no directed edge starting from it. A vertex $\mathsf{v} \in \mathsf{V}$ will be called an *inner vertex* iff $\mathsf{v} \neq \mathsf{v}_0$ and v is not output vertex. Let us put V_i (resp. V_e) for the set of all inner (resp. output) vertices of V. Thus we have $\mathsf{V} = \{\mathsf{v}_0\} \cup \mathsf{V}_i \cup \mathsf{V}_e$. (Observe that $(\{\mathsf{v}_0\} \cup \mathsf{V}_e) \cap \mathsf{V}_i = \emptyset$ but $\{\mathsf{v}_0\} \cap \mathsf{V}_e$ may be nonempty; in this case, however, $\mathsf{V}_i = \emptyset$ and $\mathsf{V}_e = \{\mathsf{v}_0\}$.) An edge starting from $\mathsf{u} \in \mathsf{V}$ and leading into $\mathsf{v} \in \mathsf{V}$ will be simply denoted by (u, v).

Now, it is well known that to each $v \in V_e$ there is a unique (directed) path starting from v_0 and ending in v; this path is uniquely determined by the order of all its vertices $v_0, v_1, \ldots, v_{r-1}, v_r = v$. The path will be called a *branch* (*from* v_0 *to* v), and since there is a one-to-one correspondence between branches and sequences $v_0, v_1, \ldots, v_{r-1}, v_r = v$, we shall speak briefly about a branch $v_0, v_1, \ldots, v_{r-1}, v_r = v$.

Further, let us consider a mapping $\mathscr{H}$ of E into the set of all subsets of $\boldsymbol{E}^n$ where $\mathscr{H}$ has following properties:

1. For any $(u, v) \in E$ the set $\mathscr{H}(u, v)$, assigned to (u, v) by $\mathscr{H}$, is either a non-oriented hyperplane or an open halfspace of $\boldsymbol{E}^n$.

2. For any $u \in V - V_e$ there is an oriented hyperplane H of $\boldsymbol{E}^n$ such that the sets $\mathscr{H}(u, v_1)$, $\mathscr{H}(u, v_2)$, $\mathscr{H}(u, v_3)$, assigned respectively to the edges (u, v_1), (u, v_2), (u, v_3) starting from u (see condition a) of the definition of (V, E)), coincide after an appropriate permutation with the sets H^+, H^0, H^-, i.e.

$$\{\mathscr{H}(u, v_1), \mathscr{H}(u, v_2), \mathscr{H}(u, v_3)\} = \{H^+, H^0, H^-\} .$$

Now, by using mapping $\mathscr{H}$, we introduce a mapping G of V into the set of all subsets of $\boldsymbol{E}^n$ as follows: For $v = v_0$ we put $G(v_0) = \boldsymbol{E}^n$; for $v \in V - \{v_0\}$ we put $G(v) = \mathscr{H}(v_0, v_1) \cap \mathscr{H}(v_1, v_2) \cap \ldots \cap \mathscr{H}(v_{r-1}, v_r)$ where $v_0, v_1, \ldots, v_{r-1}, v_r = v$ is the uniquely determined path from v_0 to v.

At last, we shall consider a mapping $\psi : V^e \to \{C, \boldsymbol{E}^n - C\}$, and we denote the image of $v \in V_e$ in the mapping Ψ by $\Psi(v)$.

Definition. The trichotomic rooted tree (V, E) satisfying conditions a) and b), together with the mapping $\mathscr{H}$ satisfying c), and with the mapping ψ will be called a *localization algorithm for* C if the following condition is satisfied:

d) For any $v \in V_e$

$$\text{if } \Psi(v) = C \text{ then } G(v) \subset C ,$$

and

$$\text{if } \Psi(v) = \boldsymbol{E}^n - C \text{ then } G(v) \subset \boldsymbol{E}^n - C .$$

According to our definition, an LA for C is uniquely determined by the ordered quadruple $(V, E, \mathscr{H}, \psi)$. Instead of this complicated notation, we shall denote localisation algorithms by single capitals, e.g. $\mathscr{A}$, $\mathscr{A}_r$, etc. It follows from the definition of an LA for C that the family $\{G(v) \mid v \in V_e\}$ is a disjoint decomposition of $\boldsymbol{E}^n$, i.e.

$$G(v') \cap G(v'') = \emptyset \text{ if } v' \neq v'' ,$$

and

$$\bigcup_{v \in V_e} G(v) = \boldsymbol{E}^n .$$

Further it follows that the family $\{G(v) \mid v \in V_e\}$ is a refinement of $\{C, \boldsymbol{E}^n - C\}$, i.e. for every $v \in V_e$ either $G(v) \subset C$ or $G(v) \subset \boldsymbol{E}^n - C$.

We shall introduce notations for concepts concerning the LA for C. Vertices, edges, paths and branches of the corresponding tree (V, E) will be called resp. vertices, edges, paths and branches of the algorithm. The number of edges lying on a branch (i.e. number of vertices minus one) will be called *length* of the branch. The set $G(\mathsf{v})$ will be called a *set induced* (or *generated*) *by the algorithm in the vertex* v. The family $\{G(\mathsf{v}) \mid \mathsf{v} \in \mathsf{V}_e\}$ will be called a *decomposition induced* (*generated*) *by the algorithm*.

Now we are going to introduce the complexity index for localization algorithms. Let $\mathscr{A}$ be an LA for C. Put $L(\mathscr{A})$ for the maximum integer k such that there is a branch $\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_{k-1}, \mathsf{v}_k = \mathsf{v}$ having length k and the property $G(\mathsf{v}) \neq \emptyset$. The number $L(\mathscr{A})$ will be called the *complexity index* of $\mathscr{A}$. From the intuitive point of view $L(\mathscr{A})$ can be interpreted as follows: The number of HL's required by $\mathscr{A}$ depends on $\boldsymbol{x}$, and $L(\mathscr{A})$ equals the maximum number of HL's occuring in $\mathscr{A}$.

Proposition 2.1. *For every* CPS C *there is an* LA $\mathscr{A}$ *for* C *such that*

$$L(\mathscr{A}) = \mathfrak{f}(C) .$$

Proof. The set C can be written as*

$$C = \langle H_1^- \rangle \cap \langle H_2^- \rangle \cap \ldots \cap \langle H_f^- \rangle ,$$

where $f = \mathfrak{f}(C)$, and where $H_1, H_2, \ldots, H_f$ are certain oriented hyperplanes. We shall first describe the algorithm intuitively. In this algorithm, HL's are performed with respect to H_j successively for $j = 1, 2, \ldots, f$, and the condition $\boldsymbol{x} \in \langle H_j^- \rangle$ is checked. If $\boldsymbol{x} \in \langle H_j^- \rangle$ for all $j = 1, 2, \ldots, f$ then $\boldsymbol{x} \in C$; if $\boldsymbol{x} \in H_j^+$ for some $j \in \{1, 2, \ldots, f\}$ then $\boldsymbol{x} \notin C$. This procedure can be easily converted into a formal LA for C. Let us denote it by $\mathscr{A}_0$ and observe that $L(\mathscr{A}_0) = \mathfrak{f}(C)$ what completes the proof. □

The set of all LA's for C will be denoted by $\mathfrak{A}(C)$. It follows from the Proposition 2.1 that $\mathfrak{A}(C) \neq \emptyset$ for any CPS $C \subset \boldsymbol{E}^n$, and hence we introduce the following notation. Put

$$\mathscr{L}(C) = \min \{L(\mathscr{A}) \mid \mathscr{A} \in \mathfrak{A}(C)\} .$$

$\mathscr{L}(C)$ will be called a complexity index of C. Intuitively $\mathscr{L}(C)$ is the complexity index of the "optimal" LA for C.

Now the following problem arises: For any CPS C we want to determine $\mathscr{L}(C)$ and to find $\mathscr{A} \in \mathfrak{A}(C)$ such that $L(\mathscr{A}) = \mathscr{L}(C)$ (the "optimal" LA for C). The problem formulated in this general way seems to be rather difficult, and it follows from the discussion of Section 7 that any progress in its solving would probably lead to some

* The symbol $\langle X \rangle$, where $X \subset \boldsymbol{E}^n$, denotes here and in what follows the closure of X with respect to the natural topology of $\boldsymbol{E}^n$.

non-trivial results concerning the algorithmical complexity of certain difficult problems in discrete programming, as e.g. the travelling-salesman problem.

For this reason, our aim is more modest. We shall obtain a lower bound for $\mathscr{L}(C)$ which depends on the number of extreme points of C. Furthermore, we show that this lower bound is, in some sense, the best possible. Let us notice that our problem is trivial in $\mathbf{E}^1$. In the next Section we shall deal with the first non-trivial case where $n = 2$ and where C is an arbitrary convex polygon.

3. LOCALIZATION PROBLEM FOR POLYGON

Let $\hat{P}_r$ be a convex r-gon in $\mathbf{E}^2$ where $r \geqq 3$. In this Section let us choose a fixed orientation of $\mathbf{E}^2$, and with respect to this orientation choose a numeration $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_r$ of vertices of $\hat{P}_r$ such that.

1. $\boldsymbol{a}_\varrho$ and $\boldsymbol{a}_{\varrho+1}$ $(\varrho = 1, 2, \ldots, r-1)$ are neighbouring vertices of $\hat{P}_r$,
2. $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_r$ are numbered anti-clockwise.

Since $\hat{P}_r$ is a CPS in $\mathbf{E}^2$ it is meaningful to talk about the $\hat{P}_r$-localization problem. Oriented hyperplanes in $\mathbf{E}^2$ are oriented lines where the orientation of a line in $\mathbf{E}^2$ is induced by the orientation of $\mathbf{E}^2$ in an obvious way. Thus we shall speak about line localizations. An oriented line D divides $\mathbf{E}^2$ into three disjoint subsets:

D^+ – the corresponding positive open halfplane.
D^0 – the corresponding non-oriented line.
D^- – the corresponding negative open halfplane.

Our aim is to determine $\mathscr{L}(\hat{P}_r)$. In view of Proposition 2.1 there exists an LA for $\hat{P}_r$ having complexity index r. We shall construct a "better" LA for $\hat{P}_r$, requiring at most $1 + \rceil \log_2 r \lfloor$ localizations*. The algorithm will first be described in an intuitive fashion and it will be clear that it can be extended to a formal definition.

For the description of the algorithm we shall use in addition following notations: For any pair of distinct points $\boldsymbol{x}$, $\boldsymbol{y}$ of $\mathbf{E}^2$ let us consider the line passing through $\boldsymbol{x}$ and $\boldsymbol{y}$, and choose the orientation of the line such that if a point moves on the line from $\boldsymbol{x}$ to $\boldsymbol{y}$ then the negative open halfplane lies on the left-hand side. This oriented line will be denoted by $D(\boldsymbol{x}, \boldsymbol{y})$; notations $D(\boldsymbol{x}, \boldsymbol{y})^+$, $D(\boldsymbol{x}, \boldsymbol{y})^0$, $D(\boldsymbol{x}, \boldsymbol{y})^-$ have obvious meaning. Further, let us choose an arbitrary fixed interior point $\boldsymbol{j}$ of the segment with terminal points $\boldsymbol{a}_1$, $\boldsymbol{a}_{\rceil (1+r)/2 \lfloor}$, and at last put $\boldsymbol{a}_{r+1} = \boldsymbol{a}_1$.

Algorithm. The algorithm consists of two parts denoted by I and II.

Part I. A sequence of ordered pairs of integers

$$(\alpha_1, \beta_1), \ldots, (\alpha_\nu, \beta_\nu), \ldots, (\alpha_\mu, \beta_\mu) \tag{2}$$

* $\rceil \xi \lfloor ::=$ minimum integer a such that $a \geqq \xi$.

is constructed where

$$1 \leqq \alpha_1 \leqq \ldots \leqq \alpha_\nu \leqq \ldots \leqq \alpha_\mu < \beta_\mu \leqq \ldots \leqq \beta_\nu \leqq \ldots \leqq \beta_1 \leqq r + 1 ,$$

and

$$\beta_\mu = \alpha_\mu + 1 .$$

The sequence (2) is constructued in accordance with following recursive rules:

(i) Perform localization with respect to $D(\boldsymbol{a}_1, \boldsymbol{a}_{](1+r)/2[})$ and distinguish the following cases:

a)* $\qquad \boldsymbol{x} \in \langle D(\boldsymbol{a}_1, \boldsymbol{a}_{](1+r)/2[})^- \rangle ,$

b) $\qquad \boldsymbol{x} \in D(\boldsymbol{a}_1, \boldsymbol{a}_{](1+r)/2[})^+ .$

In case a) put $(\alpha_1, \beta_1) = (\,](1 + r)/2\,[, r + 1)$, in case b) put $(\alpha_1, \beta_1) = (1, \,](1 + r)/2\,[)$.Pass to the following step (ii).

(ii) Let us assume that the pair (α_ν, β_ν) has been already constructed. If $\beta_\nu = \alpha_\nu + 1$ put $\mu = \nu$; in this case, the construction of sequence (2) is finished, and we pass to the IInd part of the procedure. If $\beta_\nu > \alpha_\nu + 1$ we perform the localization with respect to $D(\boldsymbol{j}, \boldsymbol{a}_{](\alpha_\nu+\beta_\nu)/2[})$ and distinguish the following cases:

a) $\qquad \boldsymbol{x} \in \langle D(\boldsymbol{j}, \boldsymbol{a}_{](\alpha_\nu+\beta_\nu)/2[})^- \rangle ,$

b) $\qquad \boldsymbol{x} \in D(\boldsymbol{j}, \boldsymbol{a}_{](\alpha_\nu+\beta_\nu)/2[})^+ .$

Put

$$(\alpha_{\nu+1}, \beta_{\nu+1}) = \begin{cases} \left(\left]\dfrac{\alpha_\nu + \beta_\nu}{2}\right[, \beta_\nu\right) \text{ in case a)} \\ \left(\alpha_\nu, \left]\dfrac{\alpha_\nu + \beta_\nu}{2}\right[\right) \text{ in case b)}, \end{cases}$$

and with the pair $(\alpha_{\nu+1}, \beta_{\nu+1})$ return to the beginning of the step (ii).

Part II. Let us assume that a pair (α_μ, β_μ) with the property $\beta_\mu = \alpha_\mu + 1$ has been constructed. Then perform the localization with respect to $D(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})$, and distinguish following cases:

a) $\qquad \boldsymbol{x} \in \langle D(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})^- \rangle ,$

b) $\qquad \boldsymbol{x} \in D(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})^+ .$

The work of the algorithm is finished.

The described algorithm solves the $\hat{P}_r$-localization problem in the sense of the following proposition the easy proof of which is ommitted.

* $\langle D(\ldots) \rangle$ denotes the closure with respect to the natural topology of $\boldsymbol{E}^2$.

Proposition 3.1. *If the described algorithm stops in* IIa) (*respectively in* IIb)) *then* $x \in \hat{P}_r$ (*resp.* $x \notin \hat{P}_r$).

In the next proposition an upper bound for the number of line localizations required by the described algorithm is determined.

Proposition 3.2. *The maximum number of line localizations required by the algorithm does not exceed* $\rceil \log_2 r \lceil + 1$.

Proof. Since in the IInd part of our algorithm exactly one localization is required, it is sufficient to prove that the maximum number of localizations occuring in its Ist part does not exceed $\rceil \log_2 r \lceil$. But in any realization of our algorithm the number of localizations occuring in its Ist part is μ (i.e. it is equivalent the length of sequence (2)). It follows, however, from the description of the algorithm that μ equals the minimum value of the subscript v such that $\beta_v - \alpha_v = 1$. On the other hand we have

$$\beta_1 - \alpha_1 \leqq \max\left(r + 1 - \left]\frac{r+1}{2}\right[, \left]\frac{r+1}{2}\right[- 1\right) = \left]\frac{r}{2}\right[$$

and

$$\beta_{v+1} - \alpha_{v+1} \leqq \max\left(\beta_v - \left]\frac{\beta_v + \alpha_v}{2}\right[, \left]\frac{\beta_v + \alpha_v}{2}\right[- \alpha_v\right) = \left]\frac{\beta_v - {}_v\alpha}{2}\right[$$
$$(v = 1, 2, \ldots)$$

which by the induction yields

$$\beta_v - \alpha_v \leqq \left]\frac{r}{2^v}\right[\quad (v = 1, 2, \ldots).$$

Putting $v = \rceil \log_2 r \lceil$ in the last inequality we obtain $\beta_v - \alpha_v \leqq 1$. Thus $\mu \leqq \leqq \rceil \log_2 r \lceil$ which completes the proof. □

The algorithm for $\hat{P}_r$ just described in an intuitive fashion can be easily converted into a formal LA for $\hat{P}_r$; let us denote it by $\mathscr{A}_r$. Hence we obtain from propositions 3.1 and 3.2.

Proposition 3.3. *It holds* $\mathscr{A}_r \in \mathfrak{U}(\hat{P}_r)$ *and* $\mathscr{L}(\hat{P}_r) \leqq L(\mathscr{A}_r) \leqq \rceil \log_2 r \lceil + 1$ *for* $r = 3, 4, 5, \ldots$.

Further we are going to show that $\mathscr{A}_r$ is "optimal".

Theorem 1. *For* $r = 3, 4, 5, \ldots$ *it holds that*

$$\mathscr{L}(\hat{P}_r) = L(\mathscr{A}_r) = \rceil \log_2 r \lceil + 1.$$

In order to prove this theorem we shall use following lemma.

Lemma 3.1. *Let* $\mathscr{A} \in \mathfrak{A}(\hat{P}_r)$ *and* $L(\mathscr{A}) \leqq \rceil \log_2 r \lceil$. *Then there is an output vertex* u *in* $\mathscr{A}$ *such that the set* G*(u) *is open** and has a nonempty intersection with at least one side of* $\hat{P}_r$.

Proof. It is sufficient to construct a branch $\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_l$ having the following property: $G(\mathsf{v}_j)$ in open and has nonempty intersection with at least $\rceil r \cdot 2^{-j} \lceil$ sides of $\hat{P}_r$ $(j = 0, 1, \ldots, l)$. The sequence $\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_l$ will be constructed by the induction with respect to $j = 0, 1, \ldots, l$, and at the same time the required property of $G(\mathsf{v}_j)$ will be verified:

(i) The set $G(\mathsf{v}_0) = \boldsymbol{E}^2$ is open and it has nonempty intersection with all $r = \rceil r \cdot 2^{-0} \lceil$ sides of $\hat{P}_r$.

(ii) Let us assume that v_j has been constructed and distinguish following cases: If v_j is an output vertex put $l = j$ and stop the construction. If v_j is no output vertex then there are in $\mathscr{A}$ two distinct vertices $\mathsf{v}^{(1)}$ and $\mathsf{v}^{(2)}$ determined uniquely by the conditions

$$\mathscr{H}(\mathsf{v}_j, \mathsf{v}^{(1)}) = H^- \quad \text{and} \quad \mathscr{H}(\mathsf{v}_j, \mathsf{v}^{(2)}) = H^+$$

(where $\mathscr{H}$ denotes the mapping from the definition of LA, and H the line corresponding to v_j). Now we have clearly $G(\mathsf{v}^{(1)}) = G(\mathsf{v}_j) \cap H^-$ and $G(\mathsf{v}^{(2)}) = G(\mathsf{v}_j) \cap H^+$ and therefore $G(\mathsf{v}^{(1)})$ and $G(\mathsf{v}^{(2)})$ are open according to the induction hypothesis. Furthermore, according to this hypothesis $G(\mathsf{v}_j)$ has nonempty intersection with at least $\rceil r \cdot 2^{-j} \lceil$ distinct sides of $\hat{P}_r$, and since v_j is no output vertex we have $j \leqq L(\mathscr{A}) - 1 \leqq$ $\leqq \rceil \log_2 r \lceil - 1 < \log_2 r$, hence $\rceil r \cdot 2^{-j} \lceil \geqq 2$. Thus there exists an $s \in \{1, 2\}$ such that $G(\mathsf{v}^{(s)})$ has nonempty intersection with at least $\rceil \rceil r \cdot 2^{-j} \lceil . 2^{-1} \lceil = \rceil r \cdot 2^{-j-1} \lceil$ distinct sides of $\hat{P}_r$. Hence setting v_{j+1} equal to the corresponding $\mathsf{v}^{(s)}$ we accomplish the inductive construction of the sequence $\mathsf{v}_0, \mathsf{v}_1, \mathsf{v}_2, \ldots$, q.e.d. □

Proof of Theorem 1. In view of proposition 3.3 it is sufficient to prove that $L(\mathscr{A}) \geqq \rceil \log_2 r \lceil + 1$ for any $\mathscr{A} \in \mathfrak{A}(\hat{P}_r)$. Assuming on the contrary that $L(\mathscr{A}) \leqq$ $\leqq \rceil \log_2 r \lceil$ for some $\mathscr{A} \in \mathfrak{A}(\hat{P}_r)$ we obtain from lemma 3.1 that there is an output vertex v in $\mathscr{A}$ such that $G(\mathsf{v})$ is open and it has nonempty intersection with a side of $\hat{P}_r$. Hence $G(\mathsf{v}) \cap \hat{P}_r \neq \emptyset$ and $G(\mathsf{v}) \cap (\boldsymbol{E}^2 - \hat{P}_r) \neq \emptyset$ which is a contradiction with the fact that the family $\{G(\mathsf{v}) \mid \mathsf{v}$ is output vertex of A$\}$ must be a refinement of $\{\hat{P}_r, \boldsymbol{E}^2 - \hat{P}_r\}$. The obtained contradiction proves the theorem . □

4. LOWER BOUND FOR $\mathscr{L}(C)$

In this Section we are going to derive a lower bound for $\mathscr{L}(C)$ where C is an arbitrary CPS in $\boldsymbol{E}^n$. This lower bound depends on the number of extreme points of C.

* Induced in u by $\mathscr{A}$.

** In the sense of natural topology of $\boldsymbol{E}^2$.

A point $x_0 \in C$ is called an *extreme point* (which will be referred to in this paper as EP) if x_0 cannot be expressed as the center of a pair of distinct points $x^{(1)}$ and $x^{(2)}$ of C. Let us write

$$C = \bigcap_J (H_j^0 \cup H_j^-) = \bigcap_J \langle H_j^- \rangle ,$$

where H_j $(j = 1, 2, \ldots, \mathfrak{f}(C))$ are certain oriented hyperplanes. The following characterization of EP's is well-known.

Lemma 4.1. *Let be $x_0 \in C$. Then x_0 is an* EP *of C iff there is an n-tuple of hyperplanes $H_{j_1}, H_{j_2}, \ldots, H_{j_n}$ $(1 \leqq j_1 < j_2 < \ldots < j_n \leqq \mathfrak{f}(C))$ such that*

$$\{x_0\} = H_{j_1}^{(0)} \cap H_{j_2}^{(0)} \cap \ldots \cap H_{j_n}^{(0)} . \quad \square$$

It follows from lemma 4.1 that there is an injection of the set of all EP's of C into the set of all n-element subsets of $\{1, 2, \ldots, \mathfrak{f}(C)\}$, hence we have:

Lemma 4.2. *Every* CPS C *contains at most* $\binom{\mathfrak{f}(C)}{n}$ EP's. $\square$

Let us denote the set of all EP's of C by $V(C)$ and put $\mathfrak{v}(C) = \operatorname{card}(V(C))$, thus $0 \leqq \mathfrak{v}(C) \leqq \binom{\mathfrak{f}(C)}{n}$. A lower bound for the number of HL's required for the solution of the C-localization problem can be derived from following lemma.

Lemma 4.3. *Let $C_1, C_2, \ldots, C_s$ and C be* CPS's *of E^n, and let*

$$C_1 \cup C_2 \cup \ldots \cup C_s = C . \tag{3}$$

Then

$$\mathfrak{v}(C) \leqq s \binom{f^*}{n},$$

where

$$f^* = \max \{\mathfrak{f}(C_j) \mid j = 1, 2, \ldots, s\} .$$

Proof. It follows from (3) that

$$V(C) \subset \bigcup_{j=1}^{s} V(C_j) .$$

Combining this inclusion with lemma 4.2, we obtain our assertion. $\square$

Theorem 2. *For arbitrary* CPS C *in E^n it holds that*

$$2^{\mathscr{L}(C)} \cdot \binom{\mathscr{L}(C)}{n} \geqq \mathfrak{v}(C) .$$

Proof. Let $\mathscr{A} \in \mathfrak{A}(C)$ and $L(\mathscr{A}) = \mathscr{L}(C)$, and let us consider the decomposition $\mathfrak{S} = \{G(\mathsf{v}) \mid \mathsf{v} \in \mathsf{V}_e\}$ induced by $\mathscr{A}$. Put

$$\mathfrak{S}_0 = \{\langle G(\mathsf{v})\rangle \mid \mathsf{v} \in \mathsf{V}_e \wedge \dim(G(\mathsf{v})) = n \wedge G(\mathsf{v}) \subset C\}.$$

According to the definition of LA, the family $\mathfrak{S}$ is a refinement of $\{C, \mathbf{E}^n - C\}$, hence

$$\bigcup \{\langle G(\mathsf{v})\rangle \mid \langle G(\mathsf{v})\rangle \in \mathfrak{S}_0\} = C. \tag{4}$$

Further, sets of $\mathfrak{S}_0$ are CPS's, and the number of faces of each of them does not exceed the length of the longest branch of $\mathscr{A}$, i.e.

$$\mathfrak{f}(\langle G(\mathsf{v})\rangle) \leqq L(\mathscr{A}) \quad \text{if} \quad \langle G(\mathsf{v})\rangle \in \mathfrak{S}_0. \tag{5}$$

At last, it holds that

$$\operatorname{card}(\mathfrak{S}_0) \leqq 2^{\mathbf{L}(\mathscr{A})} \tag{6}$$

since from each vertex $\mathsf{u} \in \mathsf{V} - \mathsf{V}_e$ exactly two edges (u, v) start such that

$$\dim(\mathscr{H}(\mathsf{u}, \mathsf{v})) = n.$$

Now we apply lemma 4.3 to the finite family $\mathfrak{S}_0$ and obtain from (4), (5), (6) and lemma 4,2

$$\mathfrak{v}(C) \leqq \operatorname{card}(\mathfrak{S}_0) \cdot \left(\max\left\{\binom{\mathfrak{f}(\langle G(\mathsf{v})\rangle)}{n} \mid \langle G(\mathsf{v})\rangle \in \mathfrak{S}_0\right\}\right) \leqq 2^{L(\mathscr{A})} \cdot \binom{L(\mathscr{A})}{n}$$

which completes the proof. □

The meaning of the proved theorem is more simply observable from following corollary. This corollary will be formulated by using asymptotical inequalities:

Let $\{a_j\}_{j=1}^{\infty}$ and $\{b_j\}_{j=1}^{\infty}$ be arbitrary countable sequences having almost all members positive. We shall write

$$a_j \gtrsim b_j \quad (j \to \infty) \quad \text{iff} \quad \operatorname*{limes\,inferior}_{j\to\infty} \frac{a_j}{b_j} \geqq 1,$$

$$a_j \lesssim b_j \quad (j \to \infty) \quad \text{iff} \quad b_j \gtrsim a_j \quad (j \to \infty),$$

and

$$a_j \sim b_j \quad (j \to \infty) \quad \text{iff} \quad a_j \gtrsim b_j \quad \text{and} \quad b_j \gtrsim a_j \quad (j \to \infty).$$

The relations $\gtrsim$ and $\lesssim$ are called asymptotical inequality relations, and $\sim$ is called asymptotical equality.

Corollary. *Let* $\{C_j\}_{j=1}^{\infty}$ *be a countable sequence of* CPS's *in* $\mathbf{E}^n$ *and let*

$$\lim_{j\to\infty} \mathfrak{v}(C_j) = \infty.$$

Then

$$\mathscr{L}(C_j) \gtrsim \log_2 v(C_j) \quad (j \to \infty) .$$

5. LOWER BOUND IS ASYMPTOTICALLY EXACT

Now, we are going to show that the lower bound obtained in the preceding Section is asymptotically exact. We shall construct for any integer $n \geqq 3$ a countable sequence

$$\{C(v)\}_{v=n+1,n+2,\ldots}$$

of convex polyhedra* in $\boldsymbol{E}^n$ such that $C(v)$ contains exactly v EP's, and $\mathscr{L}(C(v)) \sim$ $\sim \log_2 v \, (v \to \infty)$.

Remark. Such a construction cannot be performed in $\boldsymbol{E}^1$ since each CPS in $\boldsymbol{E}^1$ contains at most two EP's. On the other hand there is no lost on generality when ommitting this case since the localization problem is trivial in $\boldsymbol{E}^1$. Furthermore, we ommit also the case of $\boldsymbol{E}^2$ since it is covered by theorem 2 of Section 3.

Construction of $C(v)$. Let us choose in $\boldsymbol{E}^n$ a two-dimensional linear variety Π and an arbitrary convex r-gon $\hat{P}_r$ in Π where $r = v - n + 2$, $v \geqq n + 1$ (hence $r \geqq 3$). Further, choose a fixed orientation in Π and let $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_r$ be the sequence of vertices of $\hat{P}_r$ such that

1. $\boldsymbol{a}_\varrho$ and $\boldsymbol{a}_{\varrho+1}$ are neighbouring vertices

$$(\varrho = 1, 2, \ldots, r - 1)$$

2. $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_r$ are numbered anti-clockwise.

At last let us choose arbitrary $n - 2$ points $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{n-2}$ in $\boldsymbol{E}^n - \Pi$ such that the affine hull of $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-2}\}$ is equivalent to $\boldsymbol{E}^n$, and put $C(v)$ for the convex hull of

$$\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-2}\} .$$

The set $C(v)$ is a convex polyhedron in $\boldsymbol{E}^n$, and we shall describe an LA for $C(v)$ which will be based on the similar idea as that from Section 3 used in the case of the $\hat{P}_r$-localization problem.

First, let us introduce some notations: Let $(\boldsymbol{x}, \boldsymbol{y})$ be an ordered pair of distinct points of Π. Put $D(\boldsymbol{x}, \boldsymbol{y})$ for the oriented line passing through $\boldsymbol{x}$ and $\boldsymbol{y}$ where the orientation is chosen such that if a point moves on the line from $\boldsymbol{x}$ to $\boldsymbol{y}$ then the negative open halfplane in Π lies on the left-hand side. The symbols $D(\boldsymbol{x}, \boldsymbol{y})^+$, $D(\boldsymbol{x}, \boldsymbol{y})^-$ and $D(\boldsymbol{x}, \boldsymbol{y})^0$ will denote resp. the corresponding positive open halfplane in Π, the corresponding negative open halfplane and the corresponding non-oriented

* Convex polyhedron ::= bounded CPS.

line. Further, it follows from the definition of $C(v)$ that the points $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-2}$ are linearly independent, hence there is a unique non-oriented hyperplane containing these points. Let us denote by $H(\boldsymbol{x}, \boldsymbol{y})$ the corresponding oriented hyperplane with the orientation chosen such that

$$D(\boldsymbol{x}, \boldsymbol{y})^- \subset H(\boldsymbol{x}, \boldsymbol{y})^- .$$

Further, there is a unique non-oriented hyperplane containing the set $\Pi \cup \{\boldsymbol{b}_1, \ldots, \ldots, \boldsymbol{b}_{k-1}, \boldsymbol{b}_{k+1}, \ldots, \boldsymbol{b}_{n-2}\}$ where $k = 1, 2, \ldots, n-2$. Let us put G_k for the corresponding oriented hyperplane with the orientation chosen such that $\boldsymbol{b}_k \in G_k^-$. At last choose a fixed interior point $\boldsymbol{j}$ of the segment with the terminal points $\boldsymbol{a}_1$ and $\boldsymbol{a}_{](1+r)/2[}$, and put $\boldsymbol{a}_{r+1} = \boldsymbol{a}_1$.

The localization algorithm for $C(v)$ will be described first in an intuitive fashion.

Algorithm for $C(v)$. The algorithm consists of three parts denoted by I, II and III.

Part I. A sequence of pairs of integers

$$(\alpha_1, \beta_1), \ldots, (\alpha_\nu, \beta_\nu), \ldots, (\alpha_\mu, \beta_\mu) \tag{7}$$

is constructed where

$$1 \leqq \alpha_1 \leqq \ldots \leqq \alpha_\nu \leqq \ldots \leqq \alpha_\mu < \beta_\mu \leqq \ldots \leqq \beta_\nu \leqq \ldots \leqq \beta_1 \leqq r + 1 ,$$

and $\beta_\mu = \alpha_\mu + 1$. The construction is performed by the induction:

(i) Perform an HL with respect to $H(\boldsymbol{a}_1, \boldsymbol{a}_{](1+r)/2[})$ and distinguish two following cases:

a) $$\boldsymbol{x} \in \langle H(\boldsymbol{a}_1, \boldsymbol{a}_{](1+r)/2[})^- \rangle ,$$

b) $$\boldsymbol{x} \in H(\boldsymbol{a}_1, \boldsymbol{a}_{](1+r)/2[})^+ .$$

If a) takes place put $(\alpha_1, \beta_1) = \left(\left]\frac{1+r}{2}\right[, r+1\right)$, and

if b) takes place put $(\alpha_1, \beta_1) = \left(1, \left]\frac{1+r}{2}\right[\right)$. In both cases pass to the next step (ii).

(ii) Let us assume that a pair (α_ν, β_ν) has been constructed where ν is an integer. Check the relation $\beta_\nu - \alpha_\nu = 1$; If $\beta_\nu - \alpha_\nu = 1$ then put $\mu = \nu$, stop the construction of (7), and pass over to the IInd part of the algorithm; if $\beta_\nu - \alpha_\nu > 1$ perform an HL with respect to $H(\boldsymbol{j}, \boldsymbol{a}_{](\alpha_\nu+\beta_\nu)/2[})$ and distinguish following cases:

a) $$\boldsymbol{x} \in \langle H(\boldsymbol{j}, \boldsymbol{a}_{](\alpha_\nu+\beta_\nu)/2[})^- \rangle ,$$

b) $$\boldsymbol{x} \in H(\boldsymbol{j}, \boldsymbol{a}_{](\alpha_\nu+\beta_\nu)/2[})^+ .$$

In case a) put $(\alpha_{v+1}, \beta_{v+1}) = \left(\left]\frac{\alpha_v + \beta_v}{2}\right[, \beta_v\right)$; in case b) put $(\alpha_{v+1}, \beta_{v+1}) = \left(\alpha_v, \left]\frac{\alpha_v + \beta_v}{2}\right[\right)$. With the obtained pair $(\alpha_{v+1}, \beta_{v+1})$ pass to the beginning of step (ii).

Part II. Let us assume that a pair (α_μ, β_μ) with the property $\beta_\mu - \alpha_\mu = 1$ has been obtained. Then we perform an HL with respect to $H(\boldsymbol{a}_{\alpha_\nu}, \boldsymbol{a}_{\beta_\nu})$ and distinguish following cases:

a) $$\boldsymbol{x} \in \langle H(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})^- \rangle ,$$

b) $$\boldsymbol{x} \in H(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})^+ .$$

In case a) pass over to the IIIrd part of the algorithm. In case b) stop the procedure.

Part III. Let us perform a series of HL's with respect to hyperplanes G_k, successively for $k = 1, 2, \ldots, n-2$, and stop the procedure as soon as $\boldsymbol{x} \in G_k^+$. Following cases can occur:

a) $$\boldsymbol{x} \in G_{k_0}^+ \quad \text{for some} \quad k_0 \in \{1, 2, \ldots, n-2\}$$

b) $$\boldsymbol{x} \in \bigcap_{k=1}^{n-2} \langle G_k^- \rangle .$$

In both cases the procedure is finished.

The described algorithm solves the $C(v)$-localization problem in the sense of following proposition.

Proposition 5.1. 1. *If the algorithm has stopped in* IIIb) *then* $\boldsymbol{x} \in C(v)$.
2. *If the algorithm has stopped in* IIb) *or* IIIa) *then* $\boldsymbol{x} \notin C(v)$.

Proof. 1. If the algorithm has stopped in IIIb) then according to its construction it holds that $\boldsymbol{x} \in C_0$ where

$$C_0 = \langle H(\boldsymbol{j}, \boldsymbol{a}_{\alpha_\mu})^- \rangle \cap \langle H(\boldsymbol{j}, \boldsymbol{a}_{\beta_\mu})^+ \rangle \cap \langle H(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})^- \rangle \cap \left(\bigcap_{k=1}^{n-2} \langle G_k^- \rangle\right) .$$

C_0 is a simplex in $\boldsymbol{E}^n$ with the vertices $\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu}, \boldsymbol{j}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-2}$. Since

$$\{\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu}, \boldsymbol{j}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-2}\} \subset C(v)$$

we obtain at last $\boldsymbol{x} \in C_0 \subset C(v)$, q.e.d.

2. If the algorithm has stopped in IIb) or IIIa) then there is an oriented hyperplane H such that $\boldsymbol{x} \in H^+$ and $C(v) \subset \langle H^- \rangle$ (i.e. H strictly separates $\boldsymbol{x}$ from $C(v)$). Indeed, we can put $H = H(\boldsymbol{a}_{\alpha_\mu}, \boldsymbol{a}_{\beta_\mu})$ in case IIb) and $H = G_{k_0}$ in case IIIa). Hence, $\boldsymbol{x} \notin C(v)$ what completes the proof. □

Now, let us estimate the number of HL's in the algorithm. Similarly as in the Section 3 it can be proved that the total number of HL's occuring in parts I and II does not exceed $1 +]\log_2 r[$. Further the number of HL's occuring in the IIIrd part does not exceed $n - 2$. Thus we have the following .

Proposition 5.2. *The maximum number of* HL*'s occuring in the described algorithm for* $C(v)$ *is at most*

$$]\log_2 r[+ n - 1 =]\log_2 (v - n + 2)[+ n - 1 . \square$$

Remark. It can be proved by a more detailed investigation of our algorithm that the maximum number of required HL's equals in fact $]\log_2 r[+ n - 1$, see the analogous result of theorem 1 Section 3.

The obtained results can be summarized as follows:

Theorem 3. *The sequence* $\{C(v)\}_{v=n+1,n+2,\ldots}$ *has following properties*:

1. $C(v)$ *is a convex polyhedron in* $\mathbf{E}^n$ *and* $\mathfrak{v}(C(v)) = v$.
2. $\mathscr{L}(C(v)) \sim \log_2 v \,(v \to \infty)$.

Proof. The property 1 obviously follows from the construction of $C(v)$. Now, according to theorem 2, it holds that $\mathscr{L}(C(v)) \gtrsim \log_2 v \,(v \to \infty)$. The described algorithm for the $C(v)$-localization problem can be described formally as a formal LA for $C(v)$. Let us denote it by $\mathscr{A}(v)$, hence $\mathscr{H}(v) \in \mathfrak{A}(C(v))$ and proposition 5.2 yields

$$\mathscr{L}(C(v)) \leqq L(\mathscr{A}(v)) =]\log_2 (v - n + 2)[+ n - 1 .$$

Therefore $\mathscr{L}(C(v)) \lesssim \log_2 v \,(v \to \infty)$ what completes the proof. $\square$

The results of Sections 3, 4 and 5 can be now summarized in the following compact form:

Theorem 4. *Let us put*

$$\lambda(v) = \min \{\mathscr{L}(C) \mid C \text{ is a CPS } in\ \mathbf{E}^n \text{ and } \mathfrak{v}(C) = v\} .$$

Then $\lambda(v) \sim \log_2 v \,(v \to \infty)$.

6. LOWER BOUND WITH RESPECT TO EXTREME HALFLINES

In this Section we state without a proof a result analogous to the Theorem 4 where instead of extreme points extreme halflines are considered. A lower bound which depends on the number of extreme halflines can be useful e.g. in the case of a convex polyhedral cone since each convex polyhedral cone contains at most one EP (the vertex of the cone), hence the Theorem 2 is not efficient in this case.

Let C be a CPS in $\boldsymbol{E}^n$. A halfline $D \subset C$ will be called an *extreme halfline* (which will be referred to as EHL) of C iff there is no pair of distinct halflines D' and D'' contained in C and such that

1. $D' \neq D \neq D''$
2. D is contained in the convex hull of $D' \cup D''$.

It is well-known that each CPS C contains at most $\binom{\mathfrak{f}(C)}{n-1}$ EHL's; let us denote this number by $\mathfrak{v}_0(C)$, hence $0 \leqq \mathfrak{v}_0(C) \leqq \binom{\mathfrak{f}(C)}{n-1}$.

The following theorem can be proved analogously as theorem 4:

Theorem 4'. *Let $n \geqq 3$ be an integer, and let us put*

$$\hat{\lambda}(v) = \min \{\mathscr{L}(C) \mid C \text{ is a CPS in } \boldsymbol{E}^n \text{ and } \mathfrak{v}_0(C) = v\},$$
$$\tilde{\lambda}(v) = \min \{\mathscr{L}(C) \mid C \text{ is a CPS in } \boldsymbol{E}^n \text{ and } \mathfrak{v}(C) + \mathfrak{v}_0(C) = v\}.$$

Then $\hat{\lambda}(v) \sim \tilde{\lambda}(v) \sim \log_2 v \; (v \to \infty)$.

7. CONCLUSION — AN APPLICATION IN DISCRETE PROGRAMMING

We are going to discuss an application of our approach to the study of algorithmical complexity of certain discrete programming problems. This approach is related to that originating in [5] and [6].

For the sake of simplicity, we shall discuss this for the case of a special problem in discrete programming, the co called travelling-salesman problem (see e.g. [7] or [8]) which will be referred to as TSP. One of the most common versions of this problem can be essentially formulated as follows:

Let m be a given positive integer, and let us consider the set $\mathfrak{D}$ of all square matrices

$$\mathbf{D} = (d_{i,j})_{\substack{i=1,\ldots,m \text{ (rows)} \\ j=1,\ldots,m \text{ (columns)}}}$$

the entries $d_{i,j}$ of which are real numbers and where $d_{i,i} = 0 \; (i = 1, 2, \ldots, m)$.

Further, let $\mathfrak{F}$ denote the family consisting of all permutations $(i_1, i_2, \ldots, i_m)$ of the set $\{1, 2, \ldots, m\}$. Two permutations $(i_1, i_2, \ldots, i_m) \in \mathfrak{F}$ and $(j_1, j_2, \ldots, j_m) \in \mathfrak{F}$ will be said *equivalent* iff one of them can be obtained from the other by means of a cyclic permutation, i.e. there is an $r \in \{1, 2, \ldots, m\}$ such that

$$i_k = \begin{cases} j_{k+r} & \text{if } k + r \leqq m, \\ j_{k+r-m} & \text{if } k + r > m \end{cases}$$

$(k = 1, 2, \ldots, m)$. The introduced relation is indeed an equivalence relation. Let us denote the corresponding family of all equivalence classes by $\mathfrak{F}^*$. Elements of $\mathfrak{F}^*$

will be called cyclic sequences of integers 1, 2, ..., m, and they will be referred to as CS's. Hence each CS can be represented by any of its elements $(i_1, i_2, \ldots, i_m)$; the CS containing the permutation $(i_1, i_2, \ldots, i_m)$ will be denoted by

$$\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow .$$

(Thus $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow = \rightarrow i_m \rightarrow i_1 \rightarrow \ldots \rightarrow i_{m-1} \rightarrow = \ldots$ etc.)

Further, we assign to each $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow \in \mathfrak{F}^*$ and to each $\mathbf{D} \in \mathfrak{D}$ the value

$$\mathbf{D}(i_1, i_2, \ldots, i_m) := \sum_{k=1}^{m-1} d_{i_k, i_{k+1}} + d_{i_m, i_1} .$$

(Let us observe that our definition is correct since the assigned value does not depend on the choice of the representative $(i_1, i_2, \ldots, i_m)$ of the CS.)

Now the TSP consists in the determination of an HC $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow$ such that $\mathbf{D}(i_1, i_2, \ldots, i_m)$ is minimum. The number $\mathbf{D}(i_1, i_2, \ldots, i_m)$ is called a **D**-*length* of $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow$, and hence we shall say that our problem consists in finding the **D**-*minimum* CS.

The most efficient algorithms for the solution of the TSP, known to the author, are not satisfactory enough since the maximum number of required additive operations (i.e. additions and/or subtractions in them equals asymptotically $m^2 \cdot 2^m$, and the same asymptotical estimate is true for the maximum number of required comparisons.*

Our present unability to solve the TSP "more quickly" is very probably connected with the fact that we do not have any "sufficiently efficient" algorithmical criterium for answering the question whether a given CS is **D**-minimum or not. (Let us compare this situation e.g. with the general linear programming – continuous case – or with the ordinary transportation problem where such criteria are known and where we have at our disposal "efficient" algorithms based on them.) Thus we are going to discuss following auxiliary problem:

Given an arbitrary matrix

$$\mathbf{D} = (d_{i,j})_{i=1,\ldots,m}^{j=1,\ldots,m} \in \mathfrak{D}$$

and a CS $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow \in \mathfrak{F}$, we have to determine whether $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots$ $\ldots \rightarrow i_m \rightarrow$ is **D**-minimal or not.

A trivial necessary and sufficient condition for $\rightarrow i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m \rightarrow$ to be

* Observe that Edmonds has introduced in [9] a useful working terminology for the classification of discrete programming algorithms. According to Edmonds an algorithm is called "good" iff the number of elementary operations in it increases at most as a polynomial of the "dimension" of the problem, i.e. as δ^c where δ means the "dimension", and c is a positive constant. In our case, the dimension is essentially m, and hence the known algorithms for the TSP are not "good" in the sense of Edmonds.

D-minimal is that

$$\mathbf{D}(i_1, i_2, \ldots, i_m) \leqq \mathbf{D}(j_1, j_2, \ldots, j_m) \tag{8}$$

holds for all $\to j_1 \to j_2 \to \cdots \to j_m \to \in \mathfrak{F}^*$. System (8) determines a convex polyhedral cone in $\mathfrak{D}$ considered as an Euclidean space, i.e. $\mathbf{D} = (d_{i,j})$ are considered as vectors with coordinates $d_{i,j}$. (The dimension of $\mathfrak{D}$ is $m \,.\, (m-1)$.) Thus the solution of our auxiliary problem consists in determining whether a given point $\mathbf{D} \in \mathfrak{D}$ belongs to the cone (8) or not, and hence we come essentially to the localization problem for this cone.

On the other hand, we want to show that it is "reasonable" to solve our localization problem in the class of LA's. Indeed, the majority of algorithms used in discrete programming (especially algorithms for the solution of TSP) can be built solely from the elementary operations additions, subtractions and comparisons. (Such algorithms do not use multiplications and divisions.) In the algorithms of this class, the values of certain linear forms of $\mathbf{D} = (d_{i,j})$, computed in each step using a finite number of additions and subtractions,* are mutually compared or, in other words, the values of certain linear forms are compared with zero. In other words, the considered algorithms perform certain HL's in the Euclidean space $\mathfrak{D}$, and hence they can be interpreted as certain LA's. Thus in any realization of such an algorithm the number of comparisons equals the number of HL's where the algorithm is interpreted as an LA.

In conclusion then if we had an "efficient" LA for the cone (8) then the existence of an „efficient" algorithm for TSP would be plausible. But on the other hand (and this is even more important from the theoretical point of view) if we have proved that no "efficient" LA for the cone (8) exists then there should be no "efficient" "additive" algorithm for TSP at all.

The described approach can be extended to many discrete programming problems, and we hope that it can be of some use in the search, in some cases, for „more efficient" discrete programming algorithms, and in the search, in other cases, for the proofs of the non-existence of "efficient" algorithms. Certain non-trivial lower bounds for the number of comparisons required by various discrete programming problems have been obtained by the author in [5] and [6].

(Received November 19, 1971.)

REFERENCES

[1] Kuhn, H.; Tucker, W. A.: Linear Inequalities and Related Systems. Princeton Univ. Press, Princeton, N. J. 1956.

[2] Grünbaum, B.: Convex Polytopes. Interscience Publishers, London—New York—Sydney 1967.

[3] Berge, C.: Théorie des graphes et ses applications. Dunod, Paris 1958.

* Thus the coefficients of these linear forms are integers.

[4] Ore, O.: Theory of Graphs. American Mathematical Society, Colloquium Publications 38 (1962).
[5] Morávek, J.: On the Complexity of Discrete Programming Problems. Aplikace matematiky *14* (1969), 6, 442—474.
[6] Morávek, J.: A Note Upon Minimal Path Problem. Journal of Mathematical Analysis and Appl. *30* (June 1970), 3, 702—717.
[7] Bellman, R.: Dynamic Programming Treatment of the Travelling Salesman Problem. J. Assoc. Comput. Mach. *9* (1962), 1, 61—63.
[8] Held M.; Karp R. M.: A Dynamic Programming Approach to Sequencing Problems. J. Soc. Industr. and Appl. Math. *10* (1962), 1, 196—210.
[9] Edmonds, J.: Paths, Trees and Flowers. Canad. J. Math. *17* (1965), 449—467.

VÝTAH

Lokalizační problém v geometrii a složitost diskrétního programování

JAROSLAV MORÁVEK

Pro danou konvexní polyedrickou množinu C Eukleidova prostoru se zkoumá algoritmické řešení tohoto problému: Pro libovolně zvolený bod prostoru máme určit, zda leží v C, nebo v komplementu k C. Tento problém nazýváme lokalizačním problémem a řešíme jej ve třídě algoritmů, jejichž každý elementární krok záleží v určení relativní polohy bodu vzhledem k nějaké orientované nadrovině předepsané algoritmem. Popsané elementární kroky nazýváme nadrovinovými lokalizacemi. Je získán jistý dolní odhad počtu nadrovinových lokalizací, nutných pro řešení lokalizačního problému vzhledem k C. Dále je ukázáno, že získaný dolní odhad je asymptoticky přesný a na příkladu úlohy o obchodním cestujícím se ukazuje souvislost studované problematiky s problematikou určování algoritmické složitosti problémů diskrétního programování.

RNDr. Jaroslav Morávek, CSc.; Matematický ústav ČSAV (Mathematical Institute — Czechoslovak Academy of Sciences), Žitná 25, Praha 1.