ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XV, 4 (1977)

A. ADAMSKI, A. KORYTOWSKI and W. MITKOWSKI (Kraków)

A CONCEPTION OF OPTIMALITY FOR ALGORITHMS AND ITS APPLICATION TO THE OPTIMAL SEARCH FOR A MINIMUM

1. Introduction. Many problems in the systems theory are solved by the search for a minimum of an appropriate function. Nowadays, a great number of minimization algorithms are known — the problem is generally treated in the optimization theory. We expect that application of the optimization theory methods to the optimization algorithms will allow to systematize the field. Suitable performance criteria will make possible to compare algorithms, to indicate where each of them can be applied most effectively, and may be helpful in producing new rapidly convergent algorithms.

In this paper we present a new definition of the optimal algorithm which satisfies Bellman's optimality principle and its application to the problem of the optimal search for a minimum. The definition is based on a minimax criterion — the optimal algorithm is the "best" one for the "worst" function minimized. The traditional definition of optimality for algorithms leads to results which do not agree with our intuition of optimality. It does not determine an optimal algorithm in a unique way and allows to call optimal algorithms which do not utilize the entire information available from previous steps of computations. Our main purpose is to demonstrate that application of Bellman's principle eliminates these difficulties. For example, it is well known that the Kiefer algorithm is optimal for the search for a minimum of a unimodal function. According to the traditional definition, it remains optimal under an additional convexity assumption on the function, whereas it is obvious that then there exist "better" algorithms. The new definition of optimality requires that an optimal algorithm does not neglect any information that may improve final results of computations, therefore it yields a different optimal algorithm for convex functions.

To make sure that the optimal algorithm exists, some constraints are assumed in relation to the distance between the points at which the function values are calculated. Next, two problems of the optimal search for a minimum of a function of one variable are considered and the corre-

sponding optimal algorithms are presented. In the first case we assume that the function considered is unimodal and in the second — that it is convex. The optimal algorithms yield, in a given number of steps, the best possible estimation of the abscissa of the minimum. Finally, some numerical results are discussed which prove that the introduction of the new optimality definition is advantageous. The algorithm, optimal according to our definition, yields better results than the Kiefer algorithm which is optimal according to the traditional definition.

The problems of algorithms optimality have been studied in [1]-[5]. Kiefer ([4] and [5]) considered the optimal search for the minimum of a unimodal function of one variable. A review may be found in [1] and [3]. Most of the investigations are concerned with functions that satisfy Lipschitz conditions together with their derivatives and that are unimodal ([2] and [3]).

2. Optimal algorithm. First we define an algorithm. Let F be a set of functions with the domain $X = [a, b]$ and the range Y. Denote by S^k the set of all k-element sequences s^k in $X \times Y$ such that

$$\forall s^k \in S^k \ \exists f \in F \ \forall i \in (1:k)$$
$$f(x_i) = y_i \text{ and } |x_i - x_j| \geqslant \varepsilon \text{ for } i \neq j,\ i, j = 1, \ldots, k,$$

where $s_i^k = (x_i, y_i)$ is the i-th element of the sequence s^k, and $0 < \varepsilon < b - a$. We say that the sequence s^k is situated on the function f and f runs over s^k. Denote by $F(s^k)$ a subset of F containing all functions that run over s^k. Obviously, $F(s^0) = F$.

Let n be a positive integer, called the *maximal number of steps of an algorithm*. An *algorithm* A is a sequence of $n-1$ computable functions

$$p_i\colon (X \times Y)^{i-1} \to X \qquad (i = 2, \ldots, n)$$

and an element $x_1 \in X$. The actual number of steps n_1 is determined by means of a sequence of logical functions:

$$l_i\colon (X \times Y)^i \to \{\textbf{true}, \textbf{false}\} \qquad (i = 1, \ldots, n),$$
$$l_n(s^n) = \textbf{true} \text{ for every } s^n, \qquad n_1 = \min\big(i, l_i(s^i) = \textbf{true}\big).$$

For every $f \in F$ the algorithm A yields a sequence s^{n_1} as follows:

$$(1) \qquad s_1^{n_1} = \big(x_1, f(x_1)\big),$$
$$x_i = p_i(s_1^{n_1}, \ldots, s_{i-1}^{n_1}), \qquad s_i^{n_1} = \big(x_i, f(x_i)\big) \qquad (i = 2, \ldots, n_1).$$

Assume $\sigma^k \in S^k$ to be the set of all points over which the function runs, known before starting the computations. Denote by $\mathscr{A}_n$ the set of all algorithms A such that

$$\sigma^k \cup s^{n_1} \in S^{n_1+k}$$

for all possible values of $n_1 \leqslant n$, where s^{n_1} is given by (1). Define a *criterial function* d by

$$d\colon \mathscr{A}_n \times F \to R.$$

In virtue of (1) the function d determines a function

$$d_{n_1}\colon S^{n_1} \to R$$

for every possible value of n_1. For simplicity, we shall omit the index n_1 and write $d(s^k)$.

Now we introduce the basic concept of the *optimal algorithm* $\hat{A} \in \mathscr{A}_n$, $\hat{A} = (\hat{x}_1, \hat{p}_1, \ldots, \hat{p}_n)$. We begin with determining the function $\hat{p}_n$. We require that at $\hat{p}_n$ the criterial function d takes the value

$$\inf_{p_n} \sup_{f \in F(s^{n-1})} d(s^{n-1}, s_n)$$

for every s^{n-1} such that $s^{n-1} \cup \sigma^k \in S^{k+n-1}$. Here

$$s_n = \big(x_n, f(x_n)\big), \qquad x_n = p_n(s^{n-1}).$$

Next we determine successively $\hat{p}_{n-1}, \hat{p}_{n-2}, \ldots, \hat{p}_2$ and $\hat{x}_1$. We require that at $\hat{p}_{n-i+1}$ the function d takes the value

$$\inf_{p_{n-i+1}} \sup_{f \in F(s^{n-i})} d(s^{n-i}, s_{n-i+1}, \hat{s}_{n-i+2}, \ldots, \hat{s}_n)$$

for every s^{n-i} such that $s^{n-i} \cup \sigma^k \in S^{k+n-i}$ $(i = 1, \ldots, n-1)$. Here

$$s_{n-i+1} = \big(x_{n-i+1}, f(x_{n-i+1})\big), \qquad x_{n-i+1} = p_{n-i+1}(s^{n-i}),$$

$$\hat{s}_j = \big(\hat{x}_j, f(\hat{x}_j)\big), \qquad \hat{x}_j = \hat{p}_j(s^{n-i}, s_{n-i+1}, \hat{s}_{n-i+2}, \ldots, \hat{s}_{j-1})$$

$$(j = n-i+2, \ldots, n).$$

We determine $\hat{x}_1$ similarly. At $\hat{x}_1$ the function d should take the value

$$\inf_{x_1} \sup_{f \in F} d(s_1, \hat{s}_2, \ldots, \hat{s}_n),$$

where $s_1 = \big(x_1, f(x_1)\big)$, $\hat{s}_2, \ldots, \hat{s}_n$ are determined as above.

It is easy to see that the optimal algorithm satisfies Bellman's optimality principle: *every subsequence of k last functions $\hat{p}_i$ is an optimal algorithm from $\mathscr{A}_k$, determined on the set $F(s^{n-k})$*. According to the traditional definition of optimality, there is only one requirement for an algorithm $\hat{A}$ to be optimal [3]:

$$\sup_{f \in F} d(\hat{A}, f) = \inf_{A \in \mathscr{A}_n} \sup_{f \in F} d(A, f). \tag{2}$$

An algorithm which is optimal according to our definition is also optimal according to the latter definition, but the inverse assertion is not true. Introduction of a more complicated definition of optimality

is profitable from the computational point of view. Algorithms that satisfy only condition (2) do not often utilize the entire information about the function f, available from previous steps of computations. Therefore, condition (2) is too weak. For instance, let us consider the problem of search for a minimum of a unimodal convex function on an interval. According to (2) the Kiefer algorithm is optimal, though a large part of information is neglected during the computations.

3. Generalized Kiefer algorithm. Consider the problem of the optimal search for a minimum of a unimodal function on an interval $X = [a, b]$. Optimality is understood in the sense of the best estimation of the abscissa of the minimum. If before starting the computations the function value is known at no point of the interval, the Kiefer algorithm [4] is optimal. We study the problem under more general assumption: at the beginning the function values are known at an arbitrary number of points. The solution of this problem will be useful in later investigations concerning the optimal search for a minimum of a convex function. We use the Fibonacci sequence

$$L_{-1} = 0, \quad L_0 = L_1 = 1, \quad L_{i+1} = L_i + L_{i-1} \ (i = 1, \ldots).$$

A function f determined in the interval $[a, b]$ is *unimodal* if there exists a point $x_0 \in [a, b]$ such that f is strictly decreasing in $[a, x_0]$ and strictly increasing in $[x_0, b]$.

From now on we use the following notation. For an arbitrary index r and $s_r \in X \times Y$, x_r will denote the first element of the pair s_r, and y_r (or $f(x_r)$) will denote the second one.

Let σ^k be a sequence of k points situated on a certain unimodal function determined in $[a, b]$. In our considerations, F is the set of all unimodal functions determined in $[a, b]$ which run over σ^k. Let s^i be an arbitrary sequence whose points are different from the points of σ^k. Assume that $s^i \cup \sigma^k \in S^{i+k}$. Of course, $\sigma^0 = s^0 = \emptyset$. Three cases are possible:

(a) $\sigma^k \cup s^i = \emptyset$. In this case we substitute $x_p, x_m := a$ and $x_z := b$.

(b) The sequence $\sigma^k \cup s^i$ includes two minimal points (points with minimal ordinates) s_α and s_β, $x_\alpha < x_\beta$. Then substitute $x_p, x_m := x_\alpha$ and $x_z := x_\beta$.

(c) The sequence $\sigma^k \cup s^i$ includes a unique minimal point s_m. Denote by x_m the abscissa of s_m, by x_p — the abscissa of the nearest point of the sequence on the left-hand side of x_m, and by x_z — the abscissa of the nearest point on the right-hand side of x_m. If on the left-hand side of x_m (on the right-hand side of x_m) there are no points of the sequence $\sigma^k \cup s^i$, substitute $x_p := a$ $(x_z := b)$.

We define the *criterial function* $d(s^{n_1})$ by

$$d(s^{n_1}) = x_z - x_p. \tag{3}$$

The value $d(s^{n1})$ is the best possible estimation of the actual position of the minimum of the function f, namely

$$d(s^n) = \inf_{g,h}(h-g), \quad \bar{x}' \in [g, h] \text{ for every } f' \in F(s^n),$$

where $\bar{x}'$ denotes the abscissa of the minimum of the function f'.

We describe the optimal algorithm that minimizes the criterial function (3).

Let us fix our attention on the i-th step of the algorithm ($i = 1, \ldots, n$). Denoting by s^{i-1} the sequence of points obtained in previous steps, we can assert that the sequence $\sigma^k \cup s^{i-1}$ has been known. After determining the quantities x_p, x_m, and x_z, we obtain the abscissa x_i of the i-th point in the following way. Write $c = x_z$, $e = 1$ if $x_z - x_m > x_m - x_p$, and $c = x_p$, $e = -1$ otherwise. Let ε be a sufficiently small positive number and let

$$\chi = \begin{cases} \varepsilon & \text{if } (c, f(c)) \notin \sigma^k \cup s^{i-1}, \\ 2\varepsilon & \text{if } (c, f(c)) \in \sigma^k \cup s^{i-1}, \end{cases}$$

$$\psi = e(c - x_m) L_{n-i-1}/L_{n-i+1} + (-1)^{n-i}\varepsilon/L_{n-i+1}.$$

If $\sigma^k \cup s^{i-1} = \emptyset$, substitute $x_i := x_m + \psi$.

If $\sigma^k \cup s^{i-1} \neq \emptyset$ and $e(c - x_m) < \chi$, finish the computations.

If $\sigma^k \cup s^{i-1} \neq \emptyset$ and $e(c - x_m) \geqslant \chi$, substitute $x_i := x_m + e\max(\psi, \varepsilon)$.

Next compute the value $f(x_i)$, determine new values x_p, x_m, x_z and go to the $(i+1)$-st step.

We prove optimality of this algorithm, omitting the trivial cases $e(c - x_m) < \chi$ and $\psi < \varepsilon$.

If the number of minimal points is zero or two, our algorithm becomes the well-known Kiefer algorithm. It is evident from any standard proof of Kiefer's algorithm optimality that it is also optimal according to our definition of optimality (see [4] and [5]). Now let us assume that at the i-th step there is a unique minimal point with the abscissa x_m. In this case, optimality appears easily by the following fact. Let us take into consideration the larger of the intervals $[x_p, x_m]$ and $[x_m, x_z]$ and apply the Kiefer algorithm to it. In the worst case the estimation of the abscissa of the minimum is the same as obtained by means of our algorithm, which completes the proof.

4. Optimal search for a minimum of a convex function. We describe the optimal algorithm of the search for a minimum of a convex function on an interval $[a, b]$. We assume that the functions in question are unimodal; the case of convex and non-unimodal functions is trivial and will be neglected. Let function values be known at k points of the interval before the beginning of computations. Assume that σ^k is a sequence com-

posed of these points. Let F be the set of all unimodal convex functions which run over σ^k.

A function $f\colon [a, b] \to R$ is *convex* if

$$f(\lambda x_1 + (1-\lambda)x_2) \leqslant \lambda f(x_1) + (1-\lambda) f(x_2)$$

for all $\lambda \in [0, 1]$ and $x_1, x_2 \in [a, b]$.

Let s^i be an arbitrary sequence whose points are different from those of σ^k and let $\sigma^k \cup s^i \in S^{k+i}$. As before we distinguish three cases:

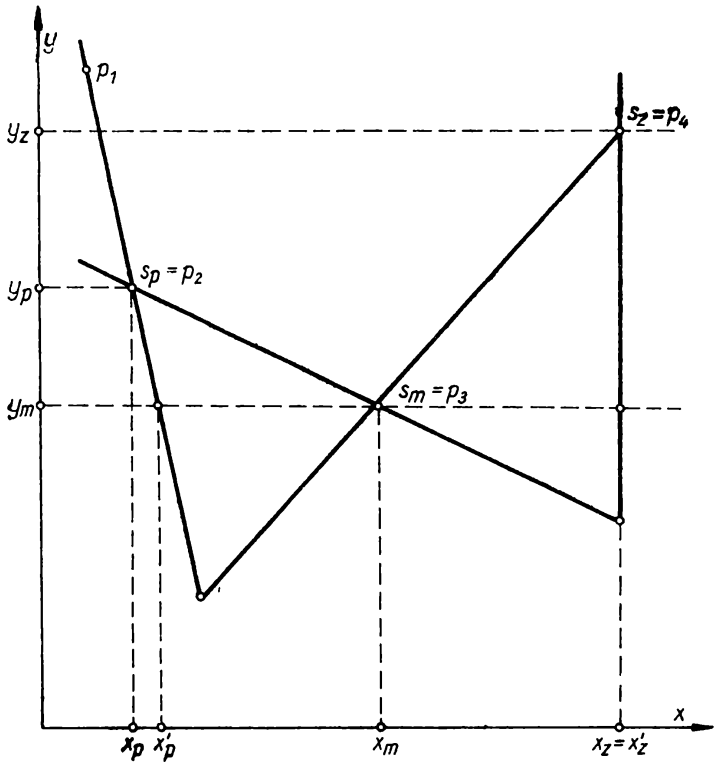

Fig. 1

(a) $\sigma^k \cup s^i = \emptyset$. Substitute $x_p, x'_p := a$, $x_z, x'_z := b$, and $x_m := a$.

(b) The sequence $\sigma^k \cup s^i$ includes two minimal points s_α and s_β, $x_\alpha < x_\beta$. Substitute $x'_p := x_\alpha$, $x'_z := x_\beta$, $s_p, s_m := s_\alpha$, and $s_z := s_\beta$.

(c) The sequence $\sigma^k \cup s^i$ includes a unique minimal point s_m. If there are no points of $\sigma^k \cup s^i$ in the interval $(x_m, b]$, substitute $x_z, x'_z := b$. If in the interval $(x_m, b]$ there is a unique point of the sequence, say s_α, substitute $x'_z := x_\alpha$, and $s_z := s_\alpha$. If in the interval $(x_m, b]$ there are more points of the sequence, denote by s_z the point with the smallest abscissa, and by s_β the point with the smallest abscissa in $(x_z, b]$. Substitute

$$x'_z := x_z - (x_\beta - x_z)(y_z - y_m)/(y_\beta - y_z).$$

The quantities x_p and x'_p are determined in a similar way. If in $[a, x_m)$ there are no points of the sequence, substitute $x_p, x'_p := a$. If in $[a, x_m)$

there is a unique point s_a of the sequence, substitute x_p, $x'_p := x_a$. If there are more points of the sequence in $[a, x_m)$, denote by s_p the point of the sequence with the largest abscissa, and by s_β the point with the largest abscissa in $[a, x_p)$. Substitute

$$x'_p := x_p - (x_\beta - x_p)(y_p - y_m)/(y_\beta - y_p).$$

The criterial function is determined by

$$d(s^{n_1}) = x'_z - x'_p, \tag{4}$$

where x'_z and x'_p are the proper values at the last step. The value $d(s^{n_1})$ is the best possible estimation of the abscissa of the minimum.

After $i-1$ steps of the algorithm, values of the function are known at $k+i-1$ points. The points compose a sequence $\sigma^k \cup s^{i-1} \in S^{k+i-1}$.

We describe the optimal algorithm.

The abscissa x_i of the i-th point is determined in the following way. Let $e = 1$, $c = x_z$, $c' = x'_z$ if $x_z - x_m > x_m - x_p$, and $e = -1$, $c = x_p$, $c' = x'_p$ otherwise. Let also ($\varepsilon > 0$ being sufficiently small)

$$\psi = e(c' - x_m) L_{n-i-1}/L_{n-i+1} + (-1)^{n-i}\varepsilon/L_{n-i+1}.$$

If $\sigma^k \cup s^{i-1} = \emptyset$, substitute $x_i := x_m + \psi$.

Now let us consider the case $\sigma^k \cup s^{i-1} \neq \emptyset$. If

$$e(c' - x_m) < \varepsilon \quad \text{or} \quad e(c - x_m) < 2\varepsilon \wedge \big(c, f(c)\big) \in \sigma^k \cup s^{i-1},$$

the computations should be finished. If neither of these conditions is satisfied, substitute

$$x_i := x_m + e \max(\psi, \varepsilon).$$

Next compute the value $f(x_i)$, determine new values x_p, x'_p, x_m, x'_z, x^s and go to the $(i+1)$-st step.

We prove optimality of this algorithm.

Let us consider the situation after the i-th step. The sequence of points $\sigma^k \cup s^i$ over which the function runs and, of course, the quantities x'_p, x'_z have been known. In the proof we omit trivial cases $e(c' - x_m) < \varepsilon$ and $e(c - x_m) < 2\varepsilon$. We introduce the following notation:

$\xi_p = x'_p$, and $\xi_z = x'_z$ at the i-th step,

F_1 — the set of all unimodal functions running over $\sigma^k \cup s^i$ with their minima in the interval (ξ_p, ξ_z),

F_2 — the set of all convex unimodal functions running over $\sigma^k \cup s^i$.

Let us consider algorithms in which the r-th abscissa x_r depends merely on the values x'_p, x_m, x'_z actual for the r-th step. Such a property of an algorithm will be denoted by W. It is evident from the definition of the criterial function that the optimal algorithm should have the property W. We introduce the following notation:

A^2 — an arbitrary algorithm from $\mathscr{A}_{n-i}$ with the property W, determined on F_2;

A^1 — an algorithm from $\mathscr{A}_{n-i}$ determined on F_1, obtained from A^2 by replacing x'_p, x'_z by x_p, x_z in every case except $x'_p = \xi_p$ and $x'_z = \xi_z$;

d_2 — the criterial function $\mathscr{A}_{n-i} \times F_2 \to R$ defined by (4);

d_1 — a criterial function obtained from d_2 by replacing x'_p, x'_z by x_p, x_z in every case except $x'_p = \xi_p$ and $x'_z = \xi_z$.

First we prove the following:

$$\forall f \in F_1 \; \exists \{f^j \mid f^j \in F_2,\ j = 1, \ldots\} \; d_1(A^1, f) = \lim_{j\to\infty} d_2(A^2, f^j).$$

We point out a suitable sequence $f^j \in F_2$ for an arbitrary function $f \in F_1$. The algorithm A^1 applied to the function f yields a sequence of points (x_r, y_r) for $r = 1, \ldots, n-i$. The algorithm A^2 applied to f^j yields a sequence (x_r^j, y_r^j) for $r = 1, \ldots, n-i$. Denote by L_g a broken line that connects the points, already known, over which the function f runs. Denote by L_d a broken line that is identical with L_g outside the interval (x_p, x_z), and inside (x_p, x_m) it is the lower limit of the area in which the minimum of the function f may occur. The functions f^j satisfy the following conditions:

(1) If $x_r^j \notin (x'_p, x'_z)$, the point (x_r^j, y_r^j) should be situated on L_d.

(2) If $x_r^j \in (x'_p, x'_z)$, it is required that

$$\operatorname{sgn}(y_r^j - y_m^j) = \operatorname{sgn}(y_r - y_m), \qquad |y_r^j - y_m^j| = j^{-j}\,\Delta y,$$

where

$$\Delta y = \begin{cases} L_g(x_r^j) - y_m^j & \text{if } y_r > y_m, \\ y_m^j - L_d(x_r^j) & \text{if } y_m > y_r, \\ 0 & \text{if } y_r = y_m. \end{cases}$$

In these formulas, x'_p, x'_z, y_m^j, y_m are the actual values at the r-th step. y_m is the ordinate of the minimal point s_m for the function f, and so is y_m^j for the function f^j. The sequence f^j is what we have sought for. In this way it is proved that for an arbitrary algorithm A^2 there exists an algorithm A^1 which does not utilize convexity of the function whose minimum is sought for, and such that

$$d_1(A^1, f) = \lim_{j\to\infty} d_2(A^2, f^j),$$

where f^j are a sequence of appropriate convex unimodal functions. Hence it appears that such an algorithm A^2 is optimal for which the corresponding algorithm A^1 is the generalized Kiefer algorithm. It is easy to verify that the described algorithm has this property, which completes the proof.

5. Numerical examples. A few examples will illustrate the performance of the algorithm A^1 described in Section 4. For comparison, we

present results obtained by means of the Kiefer algorithm for the same problems. The latter algorithm is denoted by A^k. It is assumed that $a = -1$, $b = 1$, and $k = 0$, i.e. no values of the function determined on $[-1, 1]$ are known before starting the computations. Computations have been made for various number of steps n, $4 \leqslant n \leqslant 25$. Always $n_1 = n$. The parameter ε is assumed negligibly small. For every function considered the algorithms are compared in two respects: what minimal values of the functions are obtained at every fixed n and how fast these minimal values converge to the actual minimum. This way of comparing algorithms seems most suitable for practical purposes.

If $n \leqslant 5$, the algorithms A^1 and A^k yield the same results. If $n = 6$, the difference is $\sim\varepsilon$. The algorithms have been tested with "angular" unctions

$$f(x) = \begin{cases} a_1|x - b_0| & \text{if } x \leqslant b_0, \\ a_2|x - b_0| & \text{if } x \geqslant b_0. \end{cases}$$

In case $a_1 = a_2$, $b_0 \approx 0.5(a+b) = 0$ and $n \geqslant 7$, the algorithm A_1 yields a result that differs from the exact one by no more than ε. The algorithms performance in this case is illustrated in Fig. 2 where results for the function $f_1(x) = |x - 0.1|$ are presented.

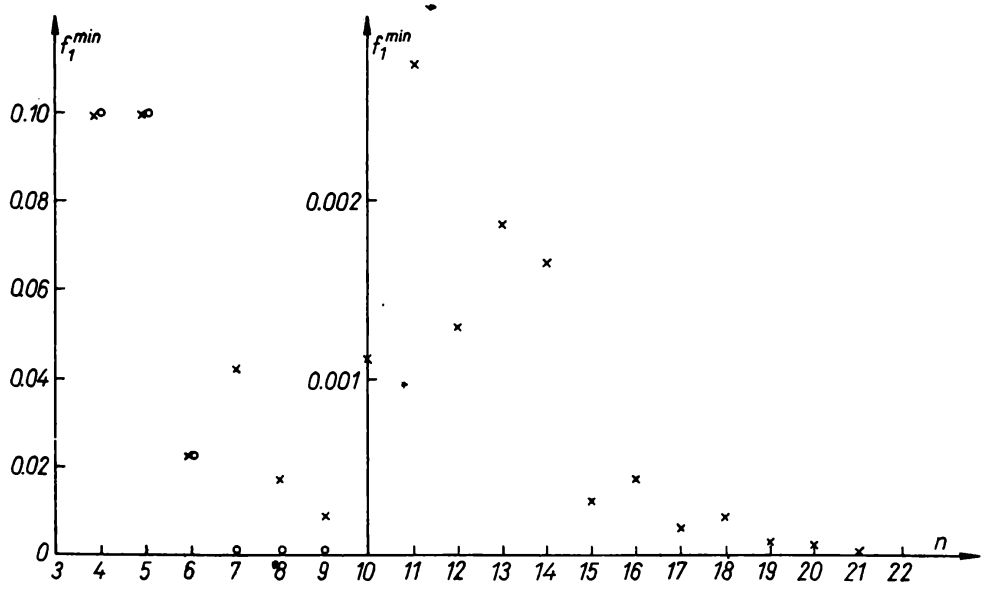

Fig. 2
○ – algorithm A^1, × – algorithm A^k

If a_1 is far removed from a_2 or b_0 is near ± 1, the supremacy of A over A^k vanishes. This is shown in Fig. 3 for the function

$$f_2(x) = \begin{cases} -x + 0.1 & \text{if } x \leqslant 0.1, \\ 100(x - 0.1) & \text{if } x \geqslant 0.1. \end{cases}$$

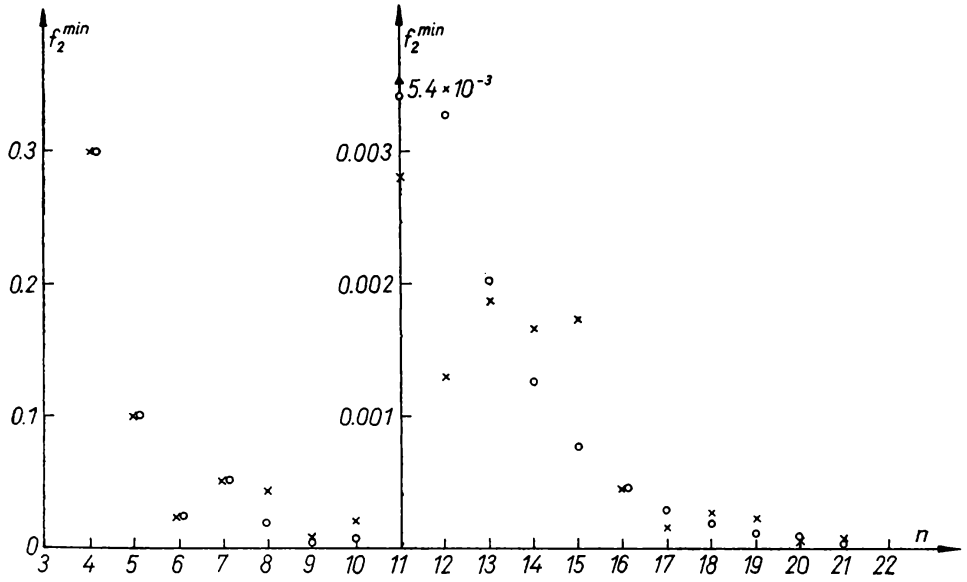

Fig. 3
○ – algorithm A^1, × – algorithm A^k

The results for

$$f_3(x) = (10x-1)^2 \quad \text{and} \quad f_4(x) = \exp[(10x-1)^2]$$

are presented in Figs. 4 and 5, respectively. The prevalence of A^1 can clearly be seen for every n except $n = 9$ and $n = 10$.

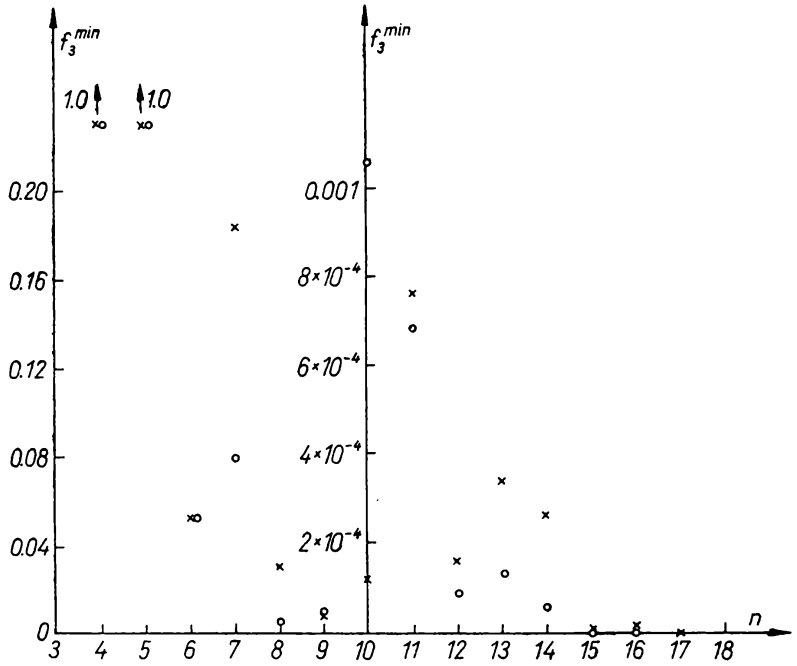

Fig. 4
○ – algorithm A^1, × – algorithm A^k

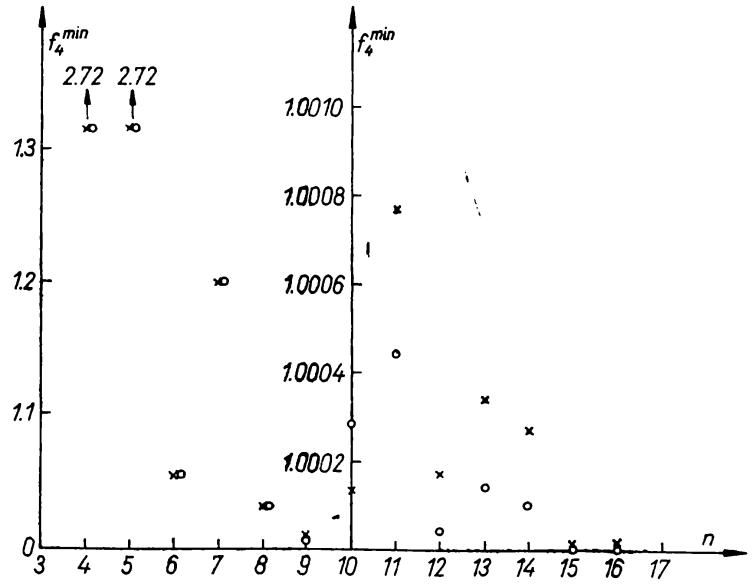

Fig. 5
○ – algorithm A^1, × – algorithm A^k

6. Conclusions. The construction of algorithm optimality definition in such a way that the optimal algorithm satisfies Bellman's optimality principle makes possible to compose optimal algorithms which utilize better the information obtained during computations. It can be seen from the numerical results presented in Section 5. Generally speaking, the algorithm A^1 yields better convergence to minimum. Only in case of functions really asymmetrical in the neighbourhood of their minimum, or if the minimum is situated near an end of the interval, the prevalence of A^1 over A^k vanishes. According to the traditional definition of optimality, both of the algorithms are optimal, according to ours – A^1 only.

References

[1] N. S. Bakhvalov, *On optimal methods of solving problems*, Aplikace Matematiky 13 (1968), p. 27-37.

[2] F. L. Chernoushko, *On optimal search for extreme of unimodal functions*, ZVM i MF 10 (1970), p. 922-934.

[3] V. V. Ivanov, *On optimal algorithms of control*, IFAC, Paris 1972.

[4] J. Kiefer, *Sequential minimax search for a maximum*, Proc. Amer. Math. Soc. 4 (1953), p. 502-510.

[5] D. J. Wilde, *Optimum seeking methods*, Prentice-Hall, Englewood Cliffs, N. J., 1964.

Received on 14. 10. 1975

A. ADAMSKI, A. KORYTOWSKI i W. MITKOWSKI (Kraków)

POJĘCIE OPTYMALNOŚCI ALGORYTMÓW I JEGO ZASTOSOWANIE DO POSZUKIWANIA MINIMUM

STRESZCZENIE

W pierwszej części pracy podano nową definicję optymalności algorytmów która pozwala nazwać *optymalnymi* tylko takie algorytmy, które w pełni wykorzystują informację dostępną w każdym kroku obliczeń. Spełniona jest zasada optymalności Bellmana. Przyjęto minimaksowe kryterium optymalności: optymalny algorytm jest „najlepszy" dla „najgorszego" przypadku. Następnie skonstruowano algorytmy optymalne dla dwóch zadań poszukiwania minimum funkcji jednej zmiennej w przedziale ograniczonym. Pierwszy algorytm jest uogólnieniem znanego algorytmu Kiefera na przypadek, w którym przed rozpoczęciem obliczeń znane są wartości funkcji badanej w pewnych punktach przedziału. Zakładana jest unimodalność. W drugim zadaniu poszukuje się minimum funkcji wypukłej. Przedstawiono wyniki numeryczne, uzyskane dla kilku typowych funkcji, i porównano je z wynikami otrzymanymi za pomocą algorytmu Kiefera.