ROCZNIKI POLSKIEGO TOWARZYSTWA MATEMATYCZNEGO
Seria III: MATEMATYKA STOSOWANA XI (1977)

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

Optymalne algorytmy poszukiwania minimum

(*Praca przyjęta do druku 21.8.1975*)

1. Wstęp. Wiele zagadnień teorii systemów i sterowania sprowadza się do poszukiwania minimum odpowiednio wybranej funkcji. Obecnie znanych jest bardzo wiele algorytmów minimalizacji funkcji — ogólnie problemem tym zajmuje się teoria optymalizacji. Wydaje się, że zastosowanie metod optymalizacji do samej tej teorii, konkretnie do algorytmów poszukiwania minimum, pozwoliłoby do pewnego stopnia usystematyzować tę dziedzinę. Mamy tu przede wszystkim na myśli wprowadzenie kryteriów jakości algorytmów, co umożliwiłoby porównywanie algorytmów pomiędzy sobą, wskazanie najwłaściwszego dla każdego z nich zakresu zastosowań, a nawet uzyskanie nowych efektywnych algorytmów.

W pracy podamy definicję algorytmu optymalnego spełniającego zasadę optymalności Bellmana i przedyskutujemy jej związek z definicją najczęściej spotykaną w literaturze. Wprowadzimy również pojęcie algorytmu optymalnego w każdym kroku. Definicje te oparte są na kryteriach minimaksowych — optymalny algorytm jest „najlepszy" dla „najgorszej" funkcji minimalizowanej w danym zbiorze. Przy takich kryteriach algorytm optymalny w wielu wypadkach nie istnieje. Rozwiązaniem problemu jest wtedy ciąg algorytmów, nazywany minimalizującym. W zadaniach minimalizacji często spotyka się funkcje wypukłe. Tymczasem w literaturze nie rozważano dotychczas optymalnych algorytmów poszukiwania minimum funkcji wypukłej nawet jednej zmiennej na odcinku. Temu zagadnieniu poświęcona jest nasza praca. Skonstruowano w niej pomocniczy algorytm optymalny (ściślej — ciąg minimalizujący), który w zadanej liczbie kroków daje najlepsze oszacowanie odciętej minimum funkcji przy założeniu jej unimodalności, a następnie rozwiązano zagadnienie optymalnego poszukiwania minimum funkcji wypukłej. Podany algorytm daje w zadanej liczbie kroków optymalne oszacowanie odciętej minimum. Z kolei przedstawiono optymalny w każdym kroku algorytm oszacowywania minimalnej wartości funkcji wypukłej. W pracy zamieszczono wyniki numeryczne uzyskane za pomocą zaproponowanych algorytmów dla kilku typowych przypadków. Przeprowadzono porównanie z algorytmem Kiefera (metoda ciągu Fibonacciego) i z algorytmem opartym na aproksymacji kwadratowej.

Ze znanych wyników, bliskich tematyce artykułu, należy wymienić metodę Kiefera [4] minimalizacji funkcji unimodalnych jednej zmiennej. Przegląd uogólnień i pokrewnych rezultatów można znaleźć w [1], [3]. Rozważano głównie optymalne poszukiwanie minimum funkcji, które wraz z pochodnymi spełniają warunek Lipschitza [2], [3].

2. Algorytmy optymalne. Jakkolwiek będziemy się zajmować algorytmami poszukiwania minimum, podane niżej definicje są znacznie bardziej ogólne. Najpierw wprowadzimy pojęcie algorytmu. Niech F będzie zbiorem funkcji określonych na zbiorze X i przyjmujących wartości w zbiorze Y. Oznaczmy przez S^k zbiór wszystkich k-wyrazowych ciągów s^k w $X \times Y$, o następujących własnościach:

$$\forall\, s^k \in S^k, \quad \exists\, f \in F, \quad \forall\, i \in (1:k),$$

$$f(x_i) = y_i,$$

gdzie $s_i^k = (x_i, y_i)$ jest i-tym wyrazem ciągu s^k. O ciągu s^k będziemy mówić, że *leży na funkcji* f, zaś o funkcji f — że *przechodzi przez ciąg punktów* s^k. Przez $F(s^k)$ oznaczymy podzbiór F, utworzony przez wszystkie funkcje leżące na ciągu s^k.

Algorytmem n-krokowym A określonym na zbiorze funkcji F nazywamy ciąg funkcji wyliczalnych

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

oraz element $x_1 \in X$. Zbiór wszystkich takich algorytmów oznaczymy przez $\mathscr{A}_n$. Dla każdej funkcji $f \in F$ algorytm A generuje ciąg $s^n \in S^n$ według schematu

$$s_1^n = \big(x_1, f(x_1)\big),$$

$$x_i = p_i(s_1^n, \ldots, s_{i-1}^n), \quad s_i^n = \big(x_i, f(x_i)\big), \quad i = 2, \ldots, n. \tag{1}$$

Na zbiorze S^n definiujemy funkcję kryterialną

$$d\colon S^n \to R.$$

Poprzez zależności (1) funkcja d określa jednoznacznie funkcję

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

którą również będziemy nazywać kryterialną.

Zdefiniujemy algorytm optymalny $\hat{A} \in \mathscr{A}_n$, $\hat{A} = (\hat{x}_1, \ldots, \hat{p}_n)$. Zaczniemy od określenia funkcji $\hat{p}_n$. Żądamy, aby na $\hat{p}_n$ osiągane było dla każdego $s^{n-1} \in S^{n-1}$

$$\inf_{p_n} \sup_{f \in F(s^{n-1})} d(s^{n-1}, s_n), \tag{2.1}$$

gdzie $s_n = \big(x_n, f(x_n)\big)$, $x_n = p_n(s^{n-1})$. Dalej kolejno określamy $\hat{p}_{n-1}, \hat{p}_{n-2}, \ldots, \hat{p}_2$ i $\hat{x}_1$. Żądamy, żeby na $\hat{p}_{n-i+1}$ osiągane było dla każdego $s^{n-i} \in S^{n-i}$ $(i = 1, \ldots, n-1)$

$$\inf_{p_{n-i+1}} \sup_{f \in F(s^{n-i})} d(s^{n-i}, s_{n-i+1}, \hat{s}_{n-i+2}, \ldots, \hat{s}_n), \tag{2.2}$$

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

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

Analogicznie określamy $\hat{x}_1$. Ma na nim być osiągnięte

$$\inf_{x_1} \sup_{f \in F} d(s_1, \ldots, \hat{s}_n), \tag{2.3}$$

gdzie $s_1 = (x_1, f(x_1))$, $\hat{s}_2, \ldots, \hat{s}_n$ określone jak wyżej.

Dla większości interesujących problemów nie istnieją algorytmy, na których osiągane są kresy (2). Zadanie optymalizacji algorytmów postawimy wtedy inaczej. Będziemy poszukiwać ciągu algorytmów $\{A(\varepsilon) \in \mathscr{A}_n, \varepsilon > 0\}$ określonego następująco: Niech s_k oznacza parę $(x_k, f(x_k))$, $x_k = p_k(s^{k-1})$, a $s_k(\varepsilon) = (x_k(\varepsilon), f(x_k(\varepsilon)))$, $x_k(\varepsilon) = p_k(\varepsilon)(s^{k-1})$. Żądamy, aby ciąg $p_n(\varepsilon)$ miał następującą własność:

$$\lim_{\varepsilon \to 0} \sup_{f \in F(s^{n-1})} d(s^{n-1}, s_n(\varepsilon)) = \inf_{p_n} \sup_{f \in F(s^{n-1})} d(s^{n-1}, s_n) \tag{3.1}$$

dla każdego $s^{n-1} \in S^{n-1}$.

Ogólnie, ciąg $p_{n-i+1}(\varepsilon)$ (dla $i = 1, \ldots, n-1$) ma spełniać dla każdego $s^{n-i} \in S^{n-i}$

$$\lim_{\varepsilon \to 0} \sup_{f \in F(s^{n-i})} d(s^{n-i}, s_{n-i+1}(\varepsilon), \ldots, s_n(\varepsilon)) = \tag{3.2}$$

$$= \inf_{p_{n-i+1}} \lim_{\varepsilon \to 0} \sup_{f \in F(s^{n-i})} d(s^{n-i}, s_{n-i+1}, s_{n-i+2}(\varepsilon), \ldots, s_n(\varepsilon));$$

$x_1(\varepsilon)$ określamy podobnie. Ma być

$$\lim_{\varepsilon \to 0} \sup_{f \in F} d(s_1(\varepsilon), \ldots, s_n(\varepsilon)) = \inf_{x_1} \lim_{\varepsilon \to 0} \sup_{f \in F} d(s_1, s_2(\varepsilon), \ldots, s_n(\varepsilon)). \tag{3.3}$$

Ciąg $A(\varepsilon)$ o tych własnościach będziemy nazywać *minimalizującym*. Ciąg taki istnieje zawsze, nawet gdy algorytm optymalny nie istnieje. Tam gdzie to nie spowoduje niejasności, będziemy używać nazwy *algorytm optymalny* zamiast *ciąg minimalizujący* i opuszczać wskaźnik ε, przy czym sformułowania będą się odnosić do własności granicznych ciągu minimalizującego dla $\varepsilon \to 0$.

Łatwo dostrzec, że algorytm optymalny spełnia zasadę optymalności Bellmana — każde k ostatnich funkcji $\hat{p}_i$ (ewentualnie $p_i(\varepsilon)$) tworzy optymalny algorytm ze zbioru $\mathscr{A}_k$ na zbiorze $F(s_1, \ldots, s_{n-k})$. W pracach [1], [3] od algorytmu optymalnego wymaga się tylko, aby osiągane na nim było

$$\inf_{A \in \mathscr{A}_n} \sup_{f \in F} d(s^n), \tag{4}$$

przy czym ciąg s^n otrzymuje się według schematu (1). Algorytm optymalny według naszej definicji jest oczywiście optymalny w tym drugim sensie, ale niekoniecznie na odwrót. Przyjęcie bardziej skomplikowanej definicji optymalności jest uzasadnione. Warunek (4) pozwala nazwać optymalnymi takie algorytmy, które nie wyzyskują w pełni informacji o funkcji f uzyskanej w poprzednich krokach, jest więc za słaby. Na przykład, w zadaniu poszukiwania minimum funkcji unimodalnej wypukłej na odcinku ograniczonym algorytm Kiefera jest przy warunku (4) optymalny, tymczasem jest oczywiste, że nie wykorzystuje on całej informacji możliwej do uzyskania.

Znalezienie w jawnej postaci algorytmu optymalnego jest zwykle trudne, dlatego warto rozważać algorytmy suboptymalne. Jedną z możliwych definicji podamy niżej. Niech s_k, x_k i p_k oraz $\bar{s}_k$, $\bar{x}_k$ i $\bar{p}_k$ będą ze sobą powiązane jak w definicji ciągu minimalizującego. Niech n nadal oznacza liczbę kroków algorytmu. Wprowadzimy n funkcji kryterialnych d_i: $(X \times Y)^i \to R$, $i = 1, \ldots, n$. Algorytmem optymalnym w każdym kroku nazywamy algorytm $\bar{A} = (\bar{x}_1, \bar{p}_2, \ldots, \bar{p}_n)$, w którym na $\bar{x}_1$ osiągane jest

$$\inf_{x_1 \in X} \sup_{f \in F} d_1(s_1), \tag{5.1}$$

a na funkcjach $\bar{p}_i$, $i = 2, \ldots, n$, osiąga się

$$\inf_{p_i} \sup_{f \in F(\bar{s}^{i-1})} d_i(\bar{s}^{i-1}, s_i). \tag{5.2}$$

Tutaj $\bar{s}^k = (\bar{s}_1, \ldots, \bar{s}_k)$. W przypadku, gdy taki algorytm nie istnieje, wprowadzamy ciąg minimalizujący w każdym kroku

$$A(\varepsilon) = \big(x_1(\varepsilon), p_2(\varepsilon), \ldots, p_n(\varepsilon)\big).$$

Zachodzi

$$\lim_{\varepsilon \to 0} \sup_{f \in F} d_1\big(s_1(\varepsilon)\big) = \inf_{x_1} \sup_{f \in F} d_1(s_1), \tag{6.1}$$

$$\lim_{\varepsilon \to 0} \sup_{f \in F(s^{i-1}(\varepsilon))} d_i\big(s^{i-1}(\varepsilon), s_i(\varepsilon)\big) = \inf_{p_i} \lim_{\varepsilon \to 0} \sup_{f \in F(s^{i-1}(\varepsilon))} d_i\big(s^{i-1}(\varepsilon), s_i\big), \tag{6.2}$$

gdzie $s^i(\varepsilon)$, $s_i(\varepsilon)$, s_i określone jak w definicji ciągu minimalizującego.

3. Uogólniony algorytm Kiefera. Rozważymy zadanie optymalnego poszukiwania minimum funkcji unimodalnej na odcinku ograniczonym $[a, b]$. Jeżeli nie jest znana wartość funkcji w żadnym punkcie przedziału, to przy zadanej liczbie kroków n optymalny (w sensie minimalizacji oszacowania odciętej minimum) jest algorytm Kiefera [4]. Niżej podamy rozwiązanie dla przypadku bardziej ogólnego, gdy na początku znane są wartości funkcji w dowolnej ilości punktów przedziału $[a, b]$. Będzie ono przydatne w późniejszych rozważaniach dotyczących funkcji wypukłych. Wynik ten ma postać ciągu minimalizującego, zależnego od parametru ε. Posłużymy się ciągiem Fibonacciego określonym następująco:

$$L_{-1} = \varepsilon, \quad L_0 = L_1 = 1, \quad L_{i+1} = L_i + L_{i-1}, \quad i = 1, \ldots \tag{7}$$

Funkcję f określoną w przedziale $[a, b]$ nazywamy *unimodalną*, jeżeli istnieje punkt $x_0 \in [a, b]$ taki, że funkcja f jest ściśle malejąca w przedziale $[a, x_0]$ i ściśle rosnąca w przedziale $[x_0, b]$.

Będziemy stosować następującą umowę. Dla dowolnego wskaźnika r i elementu $s_r \in X \times Y$, x_r będzie oznaczać pierwszy składnik pary, y_r (ewentualnie $f(x_r)$) — drugi.

Niech σ^k będzie ciągiem k punktów leżącym na pewnej funkcji unimodalnej f określonej w $[a, b]$. W naszym zadaniu F będzie zbiorem wszystkich funkcji unimo-

dalnych określonych w $[a, b]$, przechodzących przez punkty ciągu σ^k. Niech s^i będzie dowolnym ciągiem z S^i, którego punkty nie pokrywają się z punktami σ^k. Zakładamy $\sigma^0 = s^0 = \emptyset$. Możliwe są trzy przypadki:

a. $\sigma^k \cup s^i = \emptyset$. Podstawiamy wówczas $x_p, x_m := a, x_z := b$.

b. Ciąg $\sigma^k \cup s^i$ ma dwa punkty minimalne (o najmniejszej rzędnej) s_α i s_β, $x_\alpha < x_\beta$. Podstawiamy $x_p, x_m := x_\alpha, x_z := x_\beta$.

c. Ciąg $\sigma^k \cup s^i$ ma dokładnie jeden punkt minimalny. Oznaczmy przez x_m odciętą tego punktu, przez x_p odciętą najbliższego punktu ciągu po lewej stronie, a przez x_z — odciętą najbliższego punktu ciągu po prawej stronie. Jeśli po lewej (po prawej) stronie x_m nie ma punktów ciągu $\sigma^k \cup s^i$, podstawiamy $x_p := a$ ($x_z := b$).

Wprowadzamy funkcję kryterialną d:

$$d(s^n) = x_z - x_p. \tag{8}$$

Zauważmy, że wartość funkcji kryterialnej $d(s^n)$ jest najlepszym możliwym oszacowaniem położenia minimum funkcji $f \in F(s^n)$. Zachodzi mianowicie

$$d(s^n) = \inf_{g,h} (h-g), \quad \bar{x}' \in [g, h] \quad \text{dla wszystkich } f' \in F(s^n), \tag{9}$$

gdzie $\bar{x}'$ oznacza odciętą minimum funkcji f'.

Podamy opis optymalnego algorytmu minimalizacji funkcji kryterialnej (8). Przed wykonaniem i-tego kroku algorytmu, $i = 1, \ldots, n$, znany jest ciąg $\sigma^k \cup s^{i-1}$, gdzie s^{i-1} jest ciągiem otrzymanym w poprzednich $i-1$ krokach algorytmu. Po określeniu wielkości x_p, x_m, x_z odciętą i-tego punktu x_i otrzymujemy ze wzoru

$$x_i = x_m + (x_c - x_m) L_{n-i-1} / L_{n-i+1}, \tag{10}$$

gdzie

$$x_c = \begin{cases} x_p, & \text{gdy} \quad x_m - x_p > x_z - x_m, \\ x_z, & \text{gdy} \quad x_z - x_m \geqslant x_m - x_p. \end{cases}$$

Następnie obliczamy wartość funkcji $f(x_i)$, określamy nowe wartości x_p, x_m i x_z i przechodzimy do kolejnego, $(i+1)$-szego kroku. Udowodnimy optymalność tego algorytmu. W sytuacji, kiedy ilość punktów minimalnych wynosi zero lub dwa, opisany algorytm przechodzi w algorytm Kiefera. Ze standardowego dowodu optymalności [5] tego algorytmu wynika, że jest on również optymalny w sensie naszej definicji dla funkcji unimodalnych. Załóżmy teraz, że w i-tym kroku mamy dokładnie jeden punkt minimalny o odciętej x_m, wielkości x_p i x_z definiujemy jak wyżej. Optymalność w tym wypadku wynika natychmiast z faktu, że po n-tym kroku oszacowanie odciętej minimum jest nie większe niż $\max(x_m - x_p, x_z - x_m)/L_{n-i}$ (z dokładnością do ε). Takie samo oszacowanie daje w najgorszym przypadku algorytm Kiefera zastosowany do dłuższego z przedziałów $[x_p, x_m]$, $[x_m, x_z]$.

4. Optymalne poszukiwanie minimum funkcji wypukłej. Podamy algorytm optymalnego poszukiwania minimum funkcji wypukłej na odcinku ograniczonym $[a, b]$.

Założymy unimodalność rozważanych funkcji; przypadek funkcji wypukłych nieunimodalnych jest trywialny i zostanie pominięty. Założymy, że przed rozpoczęciem obliczeń znamy wartości funkcji w k punktach przedziału. Punkty te tworzą ciąg σ^k. F jest zbiorem wszystkich funkcji unimodalnych wypukłych, które przechodzą przez punkty ciągu σ^k. Funkcję f: $[a, b] \to R$ nazywamy *wypukłą*, jeżeli

$$f(\lambda x_1 + (1-\lambda)x_2) \leqslant \lambda f(x_1) + (1-\lambda) f(x_2), \quad \forall \lambda \in [0, 1], \forall x_1, x_2 \in [a, b].$$

Niech s^i będzie dowolnym ciągiem z S^i, którego punkty nie pokrywają się z σ^k, $s^0 = \sigma^0 = \emptyset$. Podobnie jak poprzednio, rozróżniamy trzy przypadki:

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

b. Ciąg $\sigma^k \cup s^i$ ma dwa punkty minimalne s_α i s_β, $x_\alpha < x_\beta$. Podstawiamy $x'_p := x_\alpha$, $x'_z := x_\beta$, $s_p, s_m := s_\alpha$, $s_z := s_\beta$.

c. Ciąg $\sigma^k \cup s^i$ ma dokładnie jeden punkt minimalny s_m. Jeżeli w przedziale $(x_m, b]$ nie ma punktów ciągu $\sigma^k \cup s^i$, podstawiamy $x_z, x'_z := b$. Jeżeli w $(x_m, b]$ jest dokładnie jeden punkt ciągu s_α, podstawiamy $x'_z := x_\alpha$, $s_z = s_\alpha$. Jeżeli w $(x_m, b]$ jest więcej niż jeden punkt ciągu, oznaczmy przez s_z punkt o najmniejszej odciętej, a przez s_β punkt ciągu o najmniejszej odciętej w $(x_z, b]$. Podstawiamy

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

Wartości x_p, x'_p wyznaczamy podobnie. Jeśli w $[a, x_m)$ nie ma punktów ciągu, $x_p, x'_p := a$. Jeśli w $[a, x_m)$ jest dokładnie jeden punkt s_α, podstawiamy $x_p, x'_p := x_\alpha$. Jeśli w $[a, x_m)$ jest więcej punktów ciągu, oznaczmy przez s_p punkt o największej odciętej, przez s_β punkt o największej odciętej w $[a, x_p)$. Podstawiamy

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

Funkcja kryterialna d określona jest równością

$$d(s^n) = x'_z - x'_p, \tag{11}$$

gdzie x'_z, x'_p są odpowiednimi wartościami w n-tym kroku. Wartość funkcji kryterialnej $d(s^n)$ jest najlepszym możliwym oszacowaniem położenia minimum funkcji $f \in F(s^n)$. Zachodzi mianowicie (9) z podstawieniem za $F(s^n)$ właściwego zbioru funkcji.

Po $i-1$ krokach algorytmu znamy wartości badanej funkcji w $k+i-1$ punktach, tworzących ciąg $\sigma^k \cup s^{i-1}$. Optymalny jest algorytm, w którym odciętą i-tego punktu oblicza się ze wzoru

$$x_i = x_m + (x_c - x_m) L_{n-i-1}/L_{n-i+1}, \tag{12}$$

gdzie

$$x_c = \begin{cases} x'_p, & \text{gdy} \quad x_m - x'_p > x'_z - x_m, \\ x'_z, & \text{gdy} \quad x'_z - x_m \geqslant x_m - x'_p. \end{cases}$$

Opiszemy dokładnie algorytm optymalny w przypadku, kiedy $k = 0$. Wskaźnik i oznacza numer kroku, $i = 1, \ldots, n$. W każdym kroku raz obliczamy wartość

funkcji $f(x_i)$. Na początku obliczeń podstawiamy x_m, x_p, $x'_p := a$, x_z, $x'_z := p$, $z := 0$, $i := 1$.

(*) Następnie podstawiamy

$$x_i := x'_p + (x'_z - x'_p) L_{n-i-1}/L_{n-i+1},$$

$$x'_z := \begin{cases} x'_z, & \text{gdy } z = 0, \\ x'_z + (f(x_i) - y_m)(x'_z - x_z)/(y_m - y_z), & \text{gdy } z = 1, \end{cases}$$

$$x'_p := \begin{cases} x'_p, & \text{gdy } p = 0, \\ x'_p + (f(x_i) - y_m)(x'_p - x_p)/(y_m - y_p), & \text{gdy } p = 1, \end{cases}$$

$$x_m := x_i, \quad y_m := f(x_m), \quad i := i+1.$$

(**) Rozróżniamy teraz dwa przypadki:

A. Jeżeli $x'_z - x_m > x_m - x'_p$, podstawiamy

$$x_i := x_m + (x'_z - x_m) L_{n-i-1}/L_{n-i+1}.$$

Wyróżniamy trzy podprzypadki:

a. $f(x_i) > y_m$. Podstawiamy

$$x'_z := \begin{cases} x_i, & \text{gdy } z = 0, \\ x_i + (f(x_i) - y_m)(x_i - x_m)/(y_z - f(x_i)), & \text{gdy } z = 1, \end{cases}$$

$$x_z := x_i, \quad y_z := f(x_i), \quad z := 1.$$

b. $f(x_i) < y_m$. Podstawiamy

$$x'_p := \begin{cases} x_m, & \text{gdy } p = 0, \\ x_m + (f(x_i) - y_m)(x_m - x_p)/(y_m - y_p), & \text{gdy } p = 1, \end{cases}$$

$$x'_z := \begin{cases} x'_z, & \text{gdy } z = 0, \\ x'_z + (f(x_i) - y_m)(x'_2 - x_z)/(y_m - y_z), & \text{gdy } z = 1, \end{cases}$$

$$x_p := x_m, \quad y_p := y_m,$$

$$x_m := x_i, \quad y_m := f(x_i).$$

W przypadkach a i b podstawiamy $i := i+1$ i przechodzimy do następnego kroku w miejscu (**).

c. $f(x_i) = y_m$. Podstawiamy $x'_p := x_m$, $x'_z := x_i$, $i := i+1$ i przechodzimy do następnego kroku w miejscu (*).

B. Jeśli $x'_z - x_m \leqslant x_m - x'_p$, postępujemy jak w punkcie A, zamieniając wszędzie (również we wskaźnikach) p na z i odwrotnie.

Udowodnimy optymalność podanego algorytmu w ogólnym przypadku, $k \geqslant 0$. Rozważmy sytuację po i-tym kroku. Znane są punkty $\sigma^k \cup s^i$, przez które przechodzi badana funkcja. Znane są również wielkości x'_p, x'_z. Wprowadzamy oznaczenia: $\xi_p = x'_p$, $\xi_z = x'_z$ w i-tym kroku, F_1 — zbiór wszystkich funkcji unimodalnych przechodzących przez punkty $\sigma^k \cup s^i$ i mających minimum w przedziale (ξ_p, ξ_z),

F_2 — zbiór wszystkich funkcji unimodalnych wypukłych przechodzących przez punkty $\sigma^k \cup s^i$.

Z określenia funkcji kryterialnej wynika, że algorytmu optymalnego wystarczy szukać wśród takich, w których r-ta odcięta x_r zależy wyłącznie od aktualnych wartości x'_p, x_m, x'_z. Taką własność nazwiemy własnością W. Oznaczmy: A^2 — dowolny $(n-i)$-krokowy algorytm o własności W, określony na F_2, A^1 — $(n-i)$-krokowy algorytm określony na F_1, powstały z A^2 przez zastąpienie w każdym kroku x'_p, x'_z przez x_p, x_z, o ile $x'_p \neq \xi_p$, $x'_z \neq \xi_z$, d_2 — funkcja kryterialna $\mathscr{A}_{n-i} \times F_2 \to R$, określona przez (11), d_1 — funkcja kryterialna powstała z d_2 przez zastąpienie x'_p, x'_z przez x_p, x_z, o ile $x'_p \neq \xi_p$, $x'_z \neq \xi_z$.

Pokażemy, że

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

Dowód przeprowadzimy przez wskazanie dla dowolnej funkcji $f \in F_1$ odpowiedniego ciągu $f^j \in F_2$. Algorytm A^1 generuje na f ciąg punktów (x_r, y_r), $r = 1, \ldots, n-i$. Algorytm A^2 generuje na f^j ciąg (x_r^j, y_r^j), $r = 1, \ldots, n-i$. Wprowadzimy łamaną górną L_g złożoną z odcinków łączących znane punkty, przez które przechodzi funkcja minimalizowana, oraz łamaną dolną L_d, która poza przedziałem (x_p, x_z) jest identyczna z L_g, a między x_p a x_z ogranicza od dołu obszar, do którego oszacowane jest położenie minimum. Na funkcje f^j nakładamy następujące warunki:

1. Jeżeli w kolejnym kroku $x_r^j \notin (x'_p, x'_z)$, punkt (x_r^j, y_r^j) ma leżeć na L_d.
2. Jeżeli w kolejnym kroku $x_r^j \in (x'_p, x'_z)$, żądamy, żeby

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

gdzie

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

W powyższych zależnościach x'_p, x'_z, y_m^j, y_m oznaczają wartości aktualne w r-tym kroku. y_m jest rzędną punktu minimalnego s_m określonego jak w opisie algorytmu dla funkcji f. Podobnie y_m^j dla funkcji f^j. Ciąg f^j jest poszukiwanym ciągiem funkcji. Udowodniliśmy w ten sposób, że dla dowolnego algorytmu A^2 istnieje algorytm A^1 nie wykorzystujący wypukłości funkcji, taki że wartość d_2 uzyskana za pomocą A^2 na odpowiednim ciągu funkcji jest w granicy równa wartości d_1 uzyskanej za pomocą A^1. Stąd wynika, że optymalny jest taki algorytm A^2, dla którego odpowiedni algorytm A^1 jest uogólnionym algorytmem Kiefera. Łatwo sprawdzić, że taką własność ma opisany algorytm, co kończy dowód.

5. Algorytm minimalizujący w każdym kroku oszacowanie minimalnej wartości funkcji wypukłej. Podamy algorytm optymalny w każdym kroku, który minimalizuje oszacowanie minimalnej wartości funkcji wypukłej określonej w przedziale $[a, b]$. Zbiorem F jest zbiór wszystkich funkcji unimodalnych wypukłych f: $[a, b] \to R$,

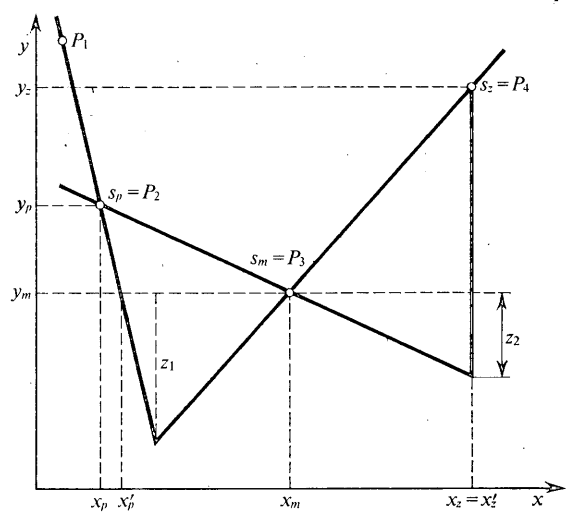
przechodzących przez punkty zadanego ciągu σ^k. Założymy, że $k \geqslant 3$ i że F nie jest zbiorem pustym. Wielkości x_p, x'_p, x_m, x_z, x'_z, s_p, s_m, s_z definiujemy jak poprzednio. Po wyznaczeniu za pomocą dowolnego algorytmu i nowych punktów s^i, przez które przechodzi funkcja, $i = 0, 1, \ldots, n$, możemy oszacować minimalną wartość badanej funkcji $f_{\min}$

$$f_{\min} \in [y_m - z, y_m], \qquad z = \max(z_1, z_2), \tag{14}$$

$$z_1 = \frac{(x_m - x'_p)(y_z - y_m)(y_p - y_m)}{(x_z - x_m)(y_p - y_m) + (x'_p - x_p)(y_z - y_m)},$$

$$z_2 = \frac{(x_m - x'_z)(y_p - y_m)(y_z - y_m)}{(x_p - x_m)(y_z - y_m) + (x'_z - x_z)(y_p - y_m)}.$$

W przypadku, gdy po prawej (po lewej) stronie punktu s_m nie ma żadnego punktu ciągu $\sigma^k \cup s^i$, wielkości y_z (odpowiednio y_p) są nieokreślone. We wzorach (14) kładziemy wtedy $y_z \to \infty$ (lub $y_p \to \infty$). Łatwo zauważyć, że oszacowanie (14) jest najlepsze z możliwych. Rozważaną sytuację ilustruje rys. 1, na którym punkty $P_1, \ldots, P_4$ tworzą ciąg $\sigma^k \cup s^i$. Z rysunku widać, że położenie punktu minimalnego funkcji jest ograniczone do dwu trójkątów, których wysokościami są z_1 i z_2.



Rys. 1. Oszacowanie minimum funkcji wypukłej

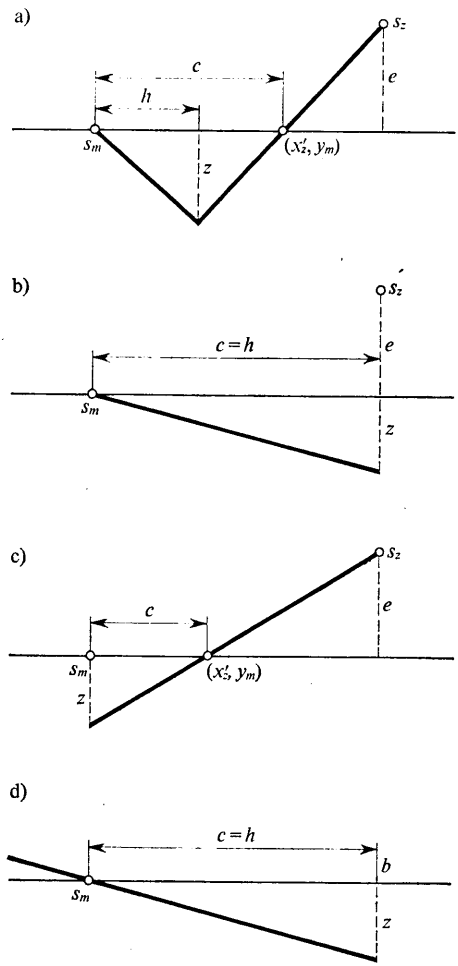
Zdefiniujemy funkcję kryterialną d, wspólną dla wszystkich kroków algorytmu. Rozważmy wielkość z, określoną przez (14). Po $i-1$ krokach algorytmu znamy ciąg punktów s^{i-1}, przez które przechodzi badana funkcja. W i-tym kroku z zależy wyłącznie od punktu s_i, otrzymywanego w tym kroku. Zaznaczymy to pisząc $z(s_i)$. Przyjmujemy zgodnie z oznaczeniami (5)

$$d(s_1) = z(s_1), \qquad d(s^{i-1}, s_i) = z(s_i), \qquad i = 2, \ldots, n. \tag{15}$$

Opiszemy algorytm optymalny w każdym kroku podając sposób wyboru odciętej x_i w i-tym kroku, $i = 1, \ldots, n$. Oczywiste są następujące implikacje:

$$(16) \qquad (z_1 \geqslant z_2) \Rightarrow (x_i \in [x'_p, x_m]), \quad (z_1 \leqslant z_2) \Rightarrow (x_i \in [x_m, x'_z]),$$

gdzie x'_p, x_m, x'_z, z_1, z_2 wyznaczone są w poprzednim, $(i-1)$-szym kroku. W ten sposób ograniczamy wybór do jednego z przedziałów (x'_p, x_m), (x_m, x'_z). Łatwo zauważyć, że w wielu wypadkach przyjęte kryterium nie wystarcza do jednoznacznego wyznaczenia x_i. Będziemy wówczas wybierać x_i według metody tej samej, co w przypadku jednoznaczności. Po wybraniu przedziału (x'_p, x_m) lub (x_m, x'_z) ilość możliwych sytuacji wynosi osiem, z których cztery powstają przez lustrzane odbicia pozostałych. Rozpatrzymy więc po kolei cztery przypadki, podając dla każdego z nich optymalną



Rys. 2. Wyróżnione sytuacje w algorytmie optymalnym w każdym kroku

odciętą i-tego punktu. Za każdym razem będziemy rozważać ten z trójkątów stanowiących łącznie obszar, do którego ograniczone jest położenie minimalnego punktu, którego wysokość jest większa. Długość podstawy tego trójkąta, równą $x_m - x'_p$ lub $x'_z - x_m$, oznaczymy przez c. Przez e oznaczymy odpowiednio $y_p - y_m$ lub $y_z - y_m$. Wielkość $x_m - x_h$ oznaczona zostanie przez h, przy czym x_h jest odciętą wierzchołka trójkąta. Przez k_1 oznaczymy $x_i - x_m$. Wyznaczenie k_1 wystarcza oczywiście do obliczenia x_i, zatem niżej podawać będziemy tylko k_1. Wprowadzamy oznaczenie

$$\alpha = 1 + e/z.$$

Dalej opuścimy wyprowadzenia, które są elementarne, ale żmudne.

P r z y p a d e k 1. Zakładamy $e \geqslant 0$, $c > h > 0$. Rozważaną sytuację przedstawia rys. 2a. W przypadku lustrzanego odbicia zmieniamy wskaźnik z na p. Wprowadzamy oznaczenia

$$\varrho = h/c, \quad b_1 = h\varrho, \quad b_2 = h(1+\alpha-\alpha\varrho), \quad b_3 = 0{,}25c(1+\varrho)^2,$$
$$b_4 = 0{,}5c[\alpha(1+\varrho) - \sqrt{\alpha^2(1+\varrho)^2 - 4\alpha\varrho}], \quad b_5 = h(1-0{,}25\alpha\varrho),$$
$$b_6 = c[\varrho\bar{c} + 2(1-2\varrho)\varphi - \sqrt{8\varrho(1-\varrho)\varphi(\bar{c}-\varphi)}], \quad \bar{c} = \alpha - (\alpha-1)\varrho,$$
$$\varphi = \sqrt{\alpha\varrho(1-\varrho)}, \quad b_{7,8} = 0{,}5c[\alpha \pm \sqrt{\alpha(\alpha - 4(\alpha-1)\varrho)}],$$
$$b_9 = c\beta,$$

gdzie β jest jedynym pierwiastkiem równania

$$\beta^2 - 2\beta\sqrt{\beta} - (\alpha-1)\varrho\beta + 2\alpha\sqrt{\beta} - \alpha = 0$$

leżącym w przedziale $[1/4, 1]$.

Schemat wyznaczania k_1 podajemy w postaci blokowej na rys. 3.

P r z y p a d e k 2. Załóżmy $e > 0$, $c = h > 0$. Rozważaną sytuację przedstawiono na rys. 2b. Oznaczamy

$$b_1 = c(1-0{,}25\alpha), \quad b_2 = c[\alpha - \sqrt{\alpha(\alpha-1)}].$$

Wielkość k_1 określona jest następująco

$$k_1 = \begin{cases} b_1, & \text{gdy} \quad \alpha < 4/3, \\ b_2, & \text{gdy} \quad \alpha \geqslant 4/3. \end{cases}$$

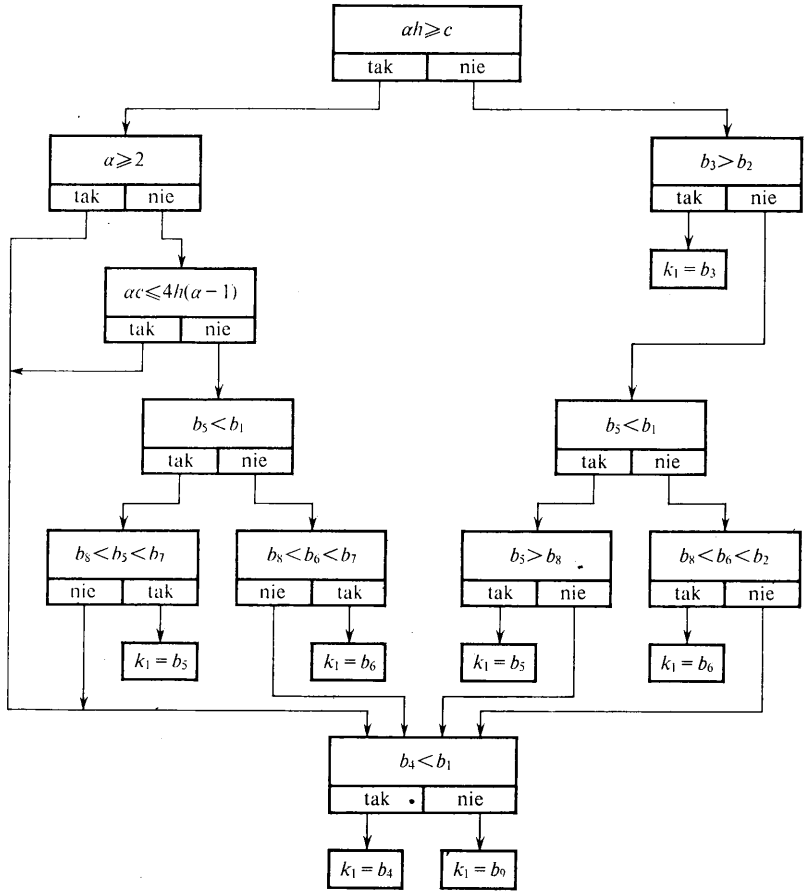
P r z y p a d e k 3. Niech $e \geqslant 0$, $c > h = 0$ (patrz rys. 2c). Wtedy $k_1 = 0{,}25c$.

P r z y p a d e k 4. Zakładamy $e \to \infty$, $c = h > 0$ (rys. 2d). W tym przypadku $k_1 = 0{,}5c$. W przypadkach przedstawionych na rys. 2, to znaczy gdy $x_i \in [x_m, x'_z]$, mamy

$$x_i = x_m + k_1.$$

W przypadku, gdy $x_i \in [x'_p, x_m]$ (lustrzane odbicia rys. 2a, b, c, d, z zamianą wskaźnika z na p), mamy

$$x_i = x_m - k_1.$$



Rys. 3. Schemat wyznaczania k_1

6. Wyniki numeryczne i porównanie algorytmów. Na kilku przykładach przeanalizujemy działanie algorytmu A_1, minimalizującego oszacowanie odciętej minimum — opisanego w punkcie 4 — i algorytmu A_2, minimalizującego w każdym kroku oszacowanie rzędnej (punkt 5). Dla porównania przedstawimy wyniki uzyskiwane dla tych samych zadań za pomocą algorytmu Kiefera A_k oraz algorytmu opartego na aproksymacji badanej funkcji parabolą drugiego stopnia A_p. Będziemy poszukiwać minimum funkcji unimodalnych wypukłych określonych w przedziale $[-1, 1]$. Przyjmiemy $k = 0$, tzn. przed rozpoczęciem obliczeń nie są znane wartości funkcji w żadnym punkcie przedziału. W algorytmach A_2 i A_p trzy pierwsze punkty wyznaczane będą według algorytmu Kiefera A_k. Obliczenia przeprowadzono dla różnej ilości kroków n, $4 \leqslant n \leqslant 25$. Dla każdej rozważanej funkcji porównujemy mini-

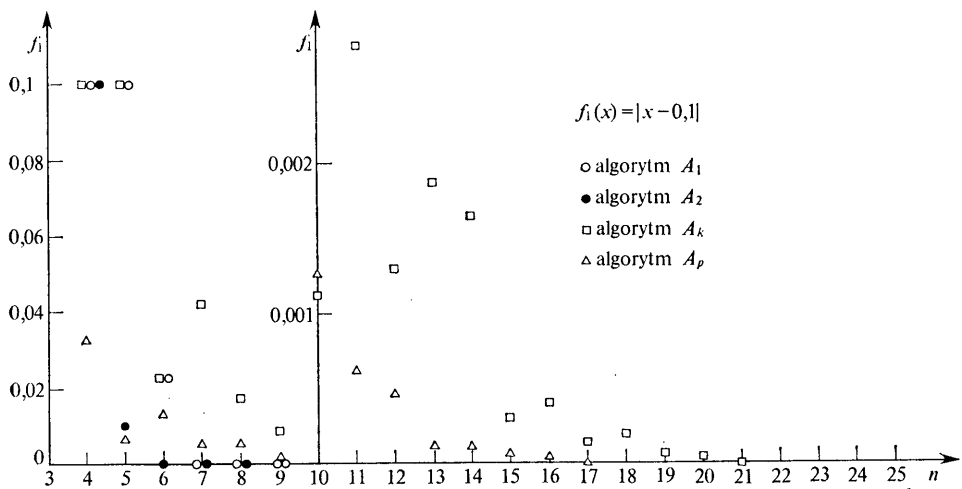
malne wartości uzyskane za pomocą każdego z algorytmów przy ustalonej ilości kroków oraz szybkość zbieżności do rzeczywistego minimum. Taki sposób porównywania algorytmów jest najbardziej uzasadniony ze względów praktycznych.

Zaczniemy od porównania algorytmu Kiefera A_k z algorytmem A_1. W obu algorytmach parametr ε przyjmujemy pomijalnie mały w stosunku do dokładności obliczeń. Dla ilości kroków $n \leqslant 5$ algorytmy A_k i A_1 pokrywają się. Dla $n = 6$ dają te same wyniki z dokładnością do ε. Algorytmy przetestowano na funkcjach kątowych

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

W przypadku $a_1 = a_2$, $b_0 \approx 0{,}5(a+b) = 0$, algorytm A_1 dla $n \geqslant 7$ daje wynik dokładny z błędem ε. Działanie algorytmów w tym wypadku ilustruje rys. 4, na którym przedstawiono wyniki dla funkcji

$$f_1(x) = |x-0{,}1|.$$



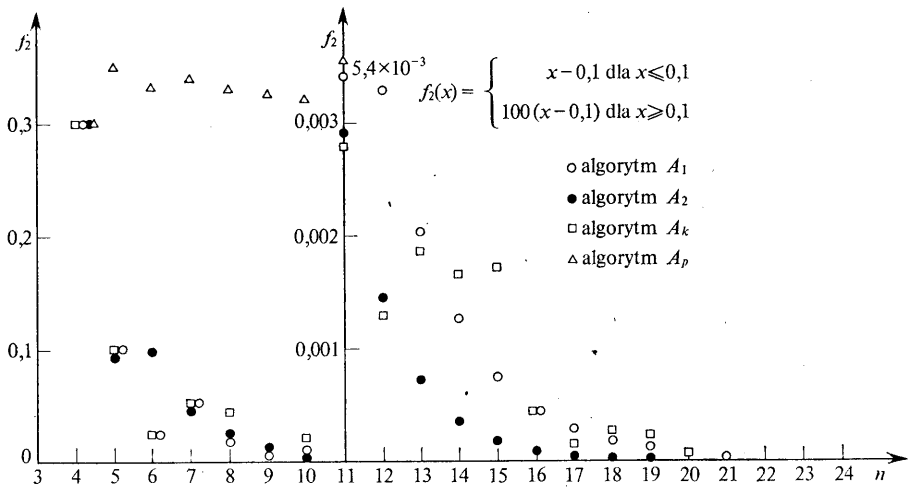
Rys. 4. Wyniki obliczeń dla funkcji $f_1(x) = |x-0{,}1|$

Gdy a_1 różni się znacznie od a_2 lub b_0 bliskie jest ± 1, różnica między A_1 a A_k zaciera się i dla różnych n lepsze wyniki może dawać jeden lub drugi. Rysunek 5 przedstawia wyniki uzyskane dla funkcji

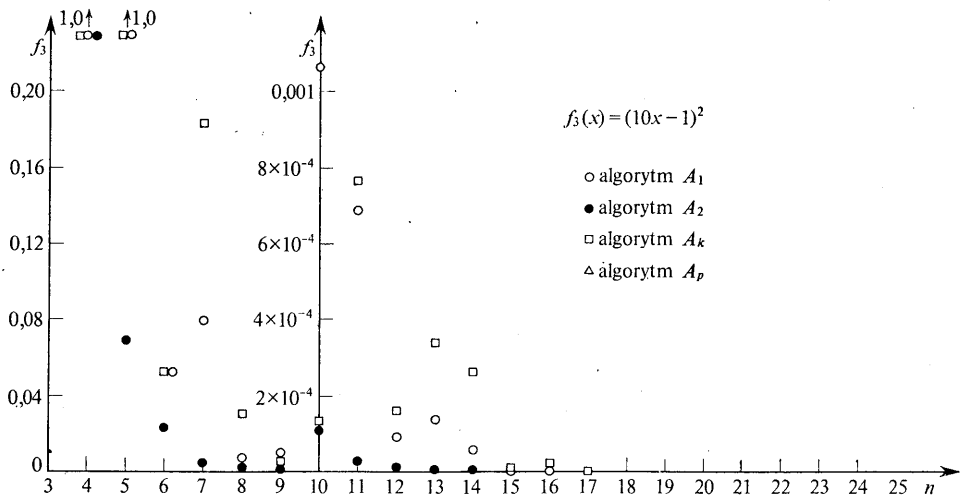
$$f_2(x) = \begin{cases} -x+0{,}1, & \text{gdy} \quad x \leqslant 0{,}1, \\ 100(x-0{,}1), & \text{gdy} \quad x \geqslant 0{,}1. \end{cases}$$

Na rys. 6 i 7 przedstawiono rezultaty dla funkcji

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



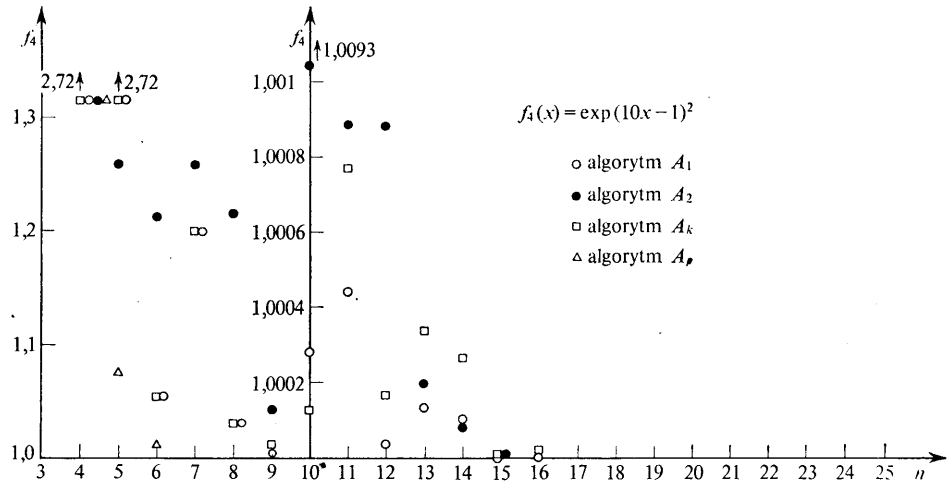
Rys. 5. Wyniki obliczeń dla funkcji $f_2(x)$



Rys. 6. Wyniki obliczeń dla funkcji $f_3(x) = (10x-1)^2$

Można tu zauważyć wyraźną przewagę algorytmu A_1 nad A_k. Jedynie dla $n = 9$ i $n = 10$ lepszy jest algorytm Kiefera. Ogólnie można stwierdzić, że algorytm A_1 daje zwykle szybszą niż A_k zbieżność do minimum. Tylko w przypadku funkcji silnie niesymetrycznych w otoczeniu minimum lub gdy minimum położone jest blisko brzegu przedziału, przewaga A_1 nad A_k zanika.

Na rysunkach pokazano również rezultaty testowania algorytmu A_2. W najczęściej spotykanych przypadkach algorytm ten daje lepszą zbieżność niż A_k i A_1.



Rys. 7. Wyniki obliczeń dla funkcji $f_4(x) = \exp(10x-1)^2$

Wskażemy dwa przypadki, w których algorytm A_2 jest wyraźnie gorszy niż A_1: w przypadku funkcji kątowych zbliżonych do f_1 (rys. 4) oraz w przypadku funkcji o gwałtownie malejącej stromości w miarę zbliżania się do minimum (funkcja f_4 na rys. 7).

Dla porównania pokazano również wyniki uzyskane za pomocą algorytmu A_p. Na ogół daje on rezultaty najlepsze — jednakże dość często może okazać się zawodny. Wszystkie algorytmy A_1, A_2, A_k są zawsze zbieżne do minimum funkcji. Algorytmy oparte wyłącznie na stosowaniu aproksymacji parabolicznej, jak A_p, tej własności nie mają (patrz rys. 5). Z tego powodu mogą być niebezpieczne, szczególnie w przypadku funkcji wyraźnie niesymetrycznych w otoczeniu minimum (jak funkcja f_2).

7. Podsumowanie. Skonstruowanie definicji optymalności algorytmu w ten sposób, żeby algorytm optymalny spełniał zasadę Bellmana, pozwala lepiej wykorzystać posiadaną informację o badanej funkcji. Widać to z porównania wyników obliczeń (punkt 6). Zgodnie z klasyczną definicją, algorytm Kiefera jest optymalny dla funkcji wypukłych, według naszej definicji — wyłącznie algorytm A_1 lub jego modyfikacje, polegające na symetrycznym przestawieniu punktów.

Z przytoczonych wyników obliczeń wynika, że szybciej zbieżne są algorytmy oparte na minimalizacji oszacowania minimalnej wartości funkcji. Algorytm A_2 w większości przypadków ma wyraźną przewagę nad A_1, mimo że jest optymalny tylko w każdym kroku, a nie globalnie.

W najczęściej spotykanych przypadkach algorytmy optymalne oparte na prostych kryteriach minimaksowych są wyraźnie gorsze niż standardowa metoda aproksymacji parabolą drugiego stopnia. Z drugiej strony — są bardziej niezawodne.

Przedstawione rezultaty mogą stanowić punkt wyjścia do dalszych badań. Interesujące byłoby uzyskanie i zbadanie algorytmu optymalnego opartego na minimalizacji oszacowania minimalnej rzędnej (analogicznego do A_2, ale optymalnego globalnie). Można rozwinąć uogólnienia na funkcje wypukłe wielu zmiennych. Przypuszczamy, że lepszą zbieżność miałyby algorytmy optymalne w sensie odpowiednio dobranego kryterium opartego nie na „najgorszym" możliwym przypadku, ale na „średnim". Ważne również byłoby ustalenie, w jakim sensie optymalne są standardowe algorytmy optymalizacji, dające zwykle dobrą zbieżność.

Zaznaczmy na koniec, że algorytm A_1 da się łatwo przerobić na algorytm złotego podziału, niezależny od ilości kroków n.

Prace cytowane

[1] N. S. Bachvalov, *Ob optimalnych metodach rešenija zadač*, Aplikace Matematiky 13, 1 (1968).

[2] F. L. Černous'ko, *Ob optimalnom poiske ekstremuma unimodalnych funkcij*, ŽVM i MF, 10, 4 (1970).

[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).

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