3 1975

prace

Instytutu Maszyn Matematycznych

rok XVII

Rok XVII

3 1975

prace Instytutu Maszyn Matematycznych

Zjednoczenie Przemysłu Automatyki i Aparatury Pomiarowej "MERA"

INSTYTUT MASZYN MATEMATYCZNYCH

Adres Redakcji: Instytut Maszyn Matematycznych
Branżowy Ośrodek INTE
Warszawa, ul. Krzywickiego 34
tel. 28-37-29

Druk IMM z. 106/75 n. 430 egz. pap. piśm. kl.III g. 70. GP-II-1435/68.

OD REDAKCJI

W związku z zaprzestaniem wydawania przez Instytut Maszyn Matematycznych czasopisma ALGORYTMY, redakcja "Prac IMM" postanowiła, za zgodą autorów, opublikować w niniejszym zeszycie trzy artykuły pracowników spoza Instytutu złożone do opublikowania w czasopiśmie ALGORYTMY.

SPIS TREŚCI ZESZYTU 3
СОДЕРЖАНИЕ
CONTENTS

Prace IMM
Zeszyt 3

681.322.06:65.012.122:
336.115

ADAPTACYJNY SYSTEM OPTYMALIZACJI NIELINIOWEJ

Jan GOLIŃSKI
Ośrodek Badawczo-Rozwojowy
Informatyki

Pracę złożono 10.11.1973

Przedstawiono założenia systemu optymalizacyjnego dla komputerów średniej wielkości, wykorzystywanego do rozwiązywania zadań technicznych i ekonomicznych o nieliniowej funkcji celu i nieliniowych ograniczeniach. System zawiera 5 algorytmów deterministycznych i stochastycznych. Algorytm nadrzędny oparty na pracy Busha i Mostellera losowo wybiera jeden z pięciu, który pracuje przez określony czas, a następnie sprawdza się efektywność jego działania. Wynik tej efektywności jest w pewien sposób oceniany i na podstawie oceny następuje modyfikacja wektora prawdopodobieństwa kolejnego wyboru algorytmu.

Na przykładach przedstawiono wyniki eksperymentów przeprowadzonych różnymi algorytmami.

1. WSTĘP

Mnogość istniejących i użytkowanych algorytmów, bogate o nich piśmiennictwo i dostęp do różnych strategii w postaci gotowych programów bibliotecznych postawiło użytkownika przed trudnym zadaniem wyboru strategii dla poszukiwania rozwiązań optymalnych. Trudności te wywołały całą serię badań porównawczych, których celem było podanie w miarę ogólnej recepty na właściwy wybór algorytmu.

Programy programowania liniowego znajdują się w każdej bibliotece. Obszerny ich wykaz podano w [1]. Programowanie nieliniowe również znajduje sobie coraz bardziej poczesne miejsce

w standardowej formie w różnych bibliotekach programów [2, 3]. Z czasem, przede wszystkim dlatego, że nie umiano sklasyfikować odpowiednio różne algorytmy programowania nieliniowego zaczęto opracowywać systemy zawierające ich zbiory. Jednym z takich systemów jest AID (Automated Improvement of Design). System ten opracowano w GENERAL ELECTRIC [4] i obejmuje on metody programowania nieliniowego, który można zastosować do rozwiązywania zadań technicznych i ekonomicznych. Zadania te wymagają następującego opisu: poszukuje się optimum funkcji $f_o(x)$

$$X = x_1, x_2, \ldots, x_n$$

przy warunkach

$$f_i(X) \geqslant 0 \quad i = 1, 2, \ldots, k$$

System pozwala na rozwiązywanie zadań spełniających następujące założenia: funkcje $f_i(x)$ są ciągłe, dopuszczalna liczba poszukiwanych zmiennych n = 15, a liczba maksymalnie uwzględnionych warunków k = 24. System zawiera podprogramy analizujące, o charakterze symulacyjnym, umożliwiające rozwiązywanie równań różniczkowych - zwykłych i cząstkowych.

W 1968 r. opracowano w SOETO system na małą maszynę cyfrową (ZAM 2) składający się z wielu niezależnych programów, z których każdy służy do szukania maksimum funkcji celu. Każdy program posiadał instrukcję korzystania z systemu. System przewidziany był na maszynę z dwoma bębnami i nagranym w sposób standardowy translatorem SAKO. Korzystanie z systemu polegało na przetłumaczeniu ciągu rozdziałów napisanych przez użytkownika w autokodzie, łącznie z dołączonym na końcu rozdziałem standardowym, podłożeniu taśmy z danymi i uruchomieniu liczenia.

Przykładem rozwiązania programowego z automatycznym wybieraniem algorytmu może być system opisany w [5], który w toku procesu obliczeniowego wybiera z grupy algorytmów optymalizacji taki, który według określonego prawdopodobieństwa pozwala na najszybsze znalezienie ekstremum. W systemie tym istnieją cztery algorytmy robocze: dwa gradientowe i dwa losowe. System ten został zrealizowany na maszynie IBM 7090/7094.

2. ADAPTACYJNY SYSTEM OPTYMALIZACJI NIELINIOWEJ NA MASZYNĘ CYFROWĄ ŚREDNIEJ WIELKOŚCI

2.1. Sformułowanie zadań systemu

System zawiera 5 algorytmów optymalizacji. Rozwiązywanie zadań optymalizacyjnych za pomocą tego systemu polega na dostarczaniu w standaryzowanej formie informacji o treści zadania oraz informacji pozwalających ocenić efektywność działania użytych algorytmów. Każde zadanie jest rozwiązywane na ogół przez wiele algorytmów, z których każdy poświęca temu zadaniu pewien określony nakład pracy. Po zakończeniu działania jednego z algorytmów następuje ocena jego działania i wybór następnego algorytmu. Postępowanie to kończy się z chwilą spełnienia warunków terminacji, którymi mogą być: maksymalny zadany czas lub określona wartość funkcji celu.

2.2. Model uczenia się

Wielu badaczy zajmowało się matematycznymi aspektami procesu adaptacji. Na ogół próbowali oni dostosować opis matematyczny do eksperymentu, który przeprowadzali. Nieliczne prace jedynie ujmują ten proces w sposób ogólny. Do takich należy liniowy model Busha i Mostellera [6].

Wprowadzili także takie pojęcia jak: wybór, ocena oraz parę złożoną z wyboru i oceny, którą zdefiniowali jako zdarzenie. Podstawowa różnica występująca w porównaniu z pojęciami statystycznej teorii decyzji polega u nich na tym, że zdarzenia zależą od alternatywnych wyborów. Cechą odróżniającą model Busha i Mostellera od innych jest jego sekwencyjny charakter. Tendencje próby charakteryzowane są przez wektor prawdopodobieństwa $p(l)$ (gdzie l jest kolejnym eksperymentem) opisujący te alternatywne wybory. Autorzy opisują drogę wektora zmieniającego się z próby na próbę. Oparto się przy tym na następujących założeniach:

- zmiana jest podana w formie operatorowej $T_{jk}p(l) = p(l+1)$, gdzie stochastyczny operator T_{jk} zależy od zdarzenia, które wystąpi w próbie l,
- każdy operator T_{jk} jest liniowy.

Przyjmuje się, że system zawiera r algorytmów $A_1, A_2 \ldots A_r$. Każdy z algorytmów rozwiązuje zadania optymalizacyjne. Przy rozwiązywaniu zadania system, wg określonego rozkładu, losowo wybiera algorytmy kolejno rozwiązujące zadania, przy czym:

a) każdy algorytm korzysta z wyników uzyskanych przez poprzednio wylosowane algorytmy,

b) wyboru algorytmu system dokonuje na podstawie wektora prawdopodobieństwa

$$p(l) = \begin{vmatrix} p_1(l) \\ p_2(l) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ p_r(l) \end{vmatrix} \qquad \sum_{j=1}^{r} p_j(l) = 1, \qquad (1)$$

gdzie l oznacza numer losowania liczony od początku pracy systemu. Elementy $p_j(l)$ wektora $p(l)$ są prawdopodobieństwem wyboru odpowiednich algorytmów w losowaniu j. I tak, można np. przyjąć, że wektor $p(o)$ ma równe elementy, tzn.

$$p_j(o) = \frac{1}{r} \quad (j = 1, 2, \ldots, r)$$

c) wylosowany algorytm rozwiązuje dane zadania w czasie Δt. Wielkość Δt określa system w zależności od liczb zmiennych decyzyjnych n,

d) po zakończeniu pracy wylosowanego algorytmu, system ocenia efekt działania tego algorytmu,

e) na podstawie wystawionej algorytmowi oceny system wyznacza nowy wektor prawdopodobieństwa $p(l+1)$. Ogólnie rzecz

biorąc ocena pozytywna dla danego algorytmu zwiększa prawdopodobieństwo jego wyboru w następnym losowaniu, zaś ocena negatywna zmniejsza, zwiększając jednocześnie prawdopodobieństwo wyboru pozostałych algorytmów. Algorytm o największej liczbie dobrych ocen będzie miał największe prawdopodobieństwo kolejnego użycia.

System w czasie rozwiązywania zadań uczy się więc je rozwiązywać, w sensie doboru najbardziej efektywnego z posiadanych algorytmów. Po rozwiązaniu szeregu zadań, system będzie preferował dla kolejnego zadania ten algorytm, który dotychczas okazał się być najbardziej skuteczny. System dopuszcza również rozwiązywanie zadania tylko jednym, określonym przez użytkownika algorytmem.

2.3. Ocena algorytmów

System przechowuje 21 uporządkowanych (od największej do najmniejszej lub odwrotnie) najlepszych z dotychczas wyznaczonych wartości funkcji celu oraz odpowiadające tym wartościom współrzędne punktów.

Każdy algorytm systemu po wyznaczeniu kolejnego punktu z obszaru G oblicza w tym punkcie wartość funkcji f(X), a następnie sprawdza, czy jest ona lepsza od wartości zawartych w bloku F. W przypadku znalezienia wartości lepszej od wartości znajdujących się w bloku F, algorytm umieszcza ją w odpowiednim miejscu tego bloku, jak również współrzędne punktu, kasując jednocześnie najgorszą wartość funkcji celu i współrzędne punktu, które tej funkcji celu odpowiadają.

W systemie przyjęto trójstopniową skalę ocen:
O_1 = 1 - ocena zła,
O_2 = 2 - ocena obojętna,
O_3 = 3 - ocena dobra.

Przy ocenie działania wylosowanego algorytmu wyróżnia się dwa przypadki:

1) wylosowany algorytm jest pierwszym algorytmem rozwiązującym dane zadanie,

2) wylosowany algorytm jest kolejnym algorytmem dla danego zadania.

W pierwszym przypadku kryterium oceny jest następujące:

- algorytm otrzymuje ocenę dobrą, jeśli wyznaczył w czasie $\triangle t$ więcej niż 14 wartości funkcji celu,
- algorytm otrzymuje ocenę obojętną, jeśli w czasie $\triangle t$ wyznaczył więcej niż 7, a nie więcej niż 14 wartości funkcji celu,
- algorytm otrzymuje ocenę złą, jeśli w czasie $\triangle t$ wyznaczył nie więcej niż 7 wartości funkcji celu.

W tym przypadku system traktuje następny wylosowany algorytm również jako pierwszy.

W drugim przypadku oznacza się przez μ_7 średnią arytmetyczną 7 pierwszych, przez μ_{14} średnią arytmetyczną 14 pierwszych, a przez μ średnią arytmetyczną wszystkich elementów przechowywanego bloku. Średnie μ_7 i μ_{14} system wylicza przed wywołaniem ocenianego (nie pierwszego dla danego zadania) algorytmu, a średnią μ po zakończeniu pracy przez ten algorytm.

W tym przypadku kryterium oceny (przy założeniu poszukiwania maksimum) jest następujące:

- algorytm otrzymuje ocenę dobrą, jeśli

$$\mu_7 < \mu$$

- algorytm otrzymuje ocenę obojętną, jeśli

$$\mu_{14} \leqslant \mu < \mu_7$$

- algorytm otrzymuje ocenę złą, jeśli

$$\mu < \mu_{14}$$

Po wystawieniu oceny pracującemu algorytmowi system wylicza nowy wektor prawdopodobieństwa $p(\ell + 1)$, według którego zostanie wylosowany następny algorytm. Wektor $p(\ell + 1)$ zależy od dotychczasowego wektora prawdopodobieństwa $p(\ell)$, i od zaistniałej sytuacji tzn. od tego, który algorytm był oceniany i jaką otrzymał on ocenę. W ten sposób realizowany jest proces uczenia się systemu. Zatem

$$p(\ell + 1) = T_{jk} \cdot p(\ell)$$

$$j = 1, 2..., r$$

$$k = 1, 2..., s$$

gdzie r - liczba algorytmów systemu, s - liczba ocen, a T_{jk} - operatory przyporządkowane zdarzeniu polegającemu na wylosowaniu algorytmu A_j, który po zadziałaniu otrzymał ocenę O_k. Przyjęto za [6], że operatory T_{jk} mają postać:

$$T_{jk} = \alpha I + (1 - \alpha)\Lambda_{jk}$$

gdzie α - współczynnik określający szybkość uczenia się systemu,

Λ_{jk} - macierze wymiaru r x r postaci:

$$\Lambda_{jk} = \begin{bmatrix} \lambda_{jk,1} & \cdots & \lambda_{jk,1} \\ \vdots & & \vdots \\ \lambda_{jk,r} & \cdots & \lambda_{jk,r} \end{bmatrix}$$

której elementy $\lambda_{jk,\nu}$ opisują stan adaptacji

I - macierz jednostkowa

Na parametry α oraz $\lambda_{jk,\nu}$ nałożone są ograniczenia:

$$0 \leqslant \lambda_{jk,\nu} \leqslant 1$$

$$\sum_{\nu=1}^{r} \lambda_{jk,\nu} = 1.$$

Z rozważań przeprowadzonych w [6] wynika, że wyżej określony proces uczenia jest zbieżny jeśli $|\alpha| \leqslant 1$. W systemie przyjęto $\alpha = \frac{1}{2}$.

Definiuje się kryterium uczenia się systemu, którym jest zadanie maksymalizacji średniej oceny granicznej wyrażonej równaniem:

$$O_\infty = \sum_{k=1}^{s} O_k \sum_{j=1}^{r} \pi_{jk} V_{\infty,j}$$

gdzie π_{jk} jest prawdopodobieństwem zdarzenia polegającego na tym, że wylosowany algorytm A_j otrzyma po działaniu ocenę O_k. Macierz π_{jk} wyraża się:

$$\pi_{jk} = \begin{bmatrix} \pi_{11} & \cdots & \pi_{1s} \\ \vdots & & \vdots \\ \pi_{r1} & & \pi_{rs} \end{bmatrix}$$

Elementy macierzy π_{jk} spełniają warunki:

$$0 \leqslant \pi_{jk} \leqslant 1$$

$$\sum_{k=1}^{s} \pi_{jk} = 1$$

Macierz π wyznacza się na podstawie wielu eksperymentów, polegających na rozwiązywaniu różnych zadań za pomocą wszystkich algorytmów zawartych w systemie. Ponadto system prowadzi histogramy otrzymywanych przez algorytmy systemu ocen w czasie eksploatacji, aby można było okresowo weryfikować przyjęte elementy macierzy. Występujące we wzorze na kryterium uczenia się systemu wielkości $V_{\infty,j}$ stanowią współrzędne wektora

$$V_{\infty} = \begin{bmatrix} V_{\infty,j} \\ \cdot \\ \cdot \\ \cdot \\ V_{\infty,r} \end{bmatrix}$$

który jest graniczną uśrednioną wartością wektora prawdopodobieństwa $p(\ell)$. Przez odpowiednią zmianę wektora Λ_{jk} można uzyskiwać coraz lepsze wartości O_{∞}. Zwiększanie się tej wartości zapewnia właściwy kierunek uczenia się systemu.

System w czasie eksploatacji dostarcza coraz obszerniejszego materiału statystycznego zawartego w T_{jk}. Istnieje sposób podany w [6], który umożliwia obliczenie wartości λ, maksymalizujące wyrażenie na O_{∞}. Do przeprowadzenia tych obliczeń potrzeba jednak obszernego materiału statystycznego o tym jakie oceny otrzymywały algorytmy rozwiązujące zadania. Po to jednak, aby system mógł zacząć pracować przyjęto arbitralnie pewne wartości λ, które umożliwiają zmienianie wektora prawdopodobieństwa wyboru algorytmów. Dla tego wyboru przyjęto, że graniczne wartości będą takie same dla każdego algorytmu i odpowiednio równe $\left[0, \frac{1}{2}, 1\right]$. Tzn. λ graniczne dla przypadku, kiedy dany algorytm otrzymywać będzie ciągle ocenę niedostateczną wynosi $\lambda_{gr} = 0$, przy ocenie dostatecznej $\lambda_{gr} = \frac{1}{2}$, a przy dobrej $\lambda_{gr} = 1$.

Przyjęta macierz λ do wstępnej eksploatacji opisana jest następująco:

$$\lambda_{jkv} = \begin{Vmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ 1 & 0 & 0 & 0 & 0 \end{Vmatrix}$$

Składowe każdego z tensorów T_{jk} można teraz zbudować z tak określonej macierzy λ_{jkv}.

Wstępnie tak określone λ pozwalają obliczyć wartość O_{∞}.

Dla przeprowadzenia tego rachunku oblicza się V_{∞} ze wzoru:

$$\left(\overline{\Lambda} - I\right) V_{\infty} = 0 \qquad (2)$$

$$\sum_{j=1}^{r} V_{\infty,j} = 1 \qquad (3)$$

gdzie:

$$\overline{\Lambda} = \begin{bmatrix} \bar{\lambda}_{11} & \cdots & \bar{\lambda}_{1r} \\ \vdots & & \vdots \\ \bar{\lambda}_{r1} & \cdots & \bar{\lambda}_{rr} \end{bmatrix}$$

$$\bar{\lambda}_{\nu} = \sum_{k=1}^{s} \pi_{jk} - \lambda_{jk,\nu} \qquad (4)$$

Mając wyznaczone $V_{\infty,j}$ z podanych zależności, oraz zebrane z eksploatacji π_{jk} można obliczyć średnią ocenę graniczną O_{∞} opisującą aktualny stan systemu. W czasie dalszej eksploatacji znowu zmienia się macierz π. Dla nowej zaktualizowanej macierzy $\pi\lambda_{jk\nu}$ należy metodą kolejnych przybliżeń przyjąć nowe wartości $\lambda_{jk,\nu}$, które poprawią uprzednio obliczoną wartość O_{∞}. Procedura ta powtarza się po określonym czasie eksploatacji, zależnym od tego, jak aktualne histogramy π_{jk} zmieniły się od ostatniej modyfikacji współczynników $\lambda_{jk,\nu}$.

2.4. Działanie systemu

Na podstawie przedstawionego modelu system stale "uczy się". Rozwiązanie każdego zadania optymalizacyjnego rozpoczyna się od wylosowania jednego z algorytmów na podstawie aktualnego wektora prawdopodobieństwa poszczególnych algorytmów. Algorytm ten liczy zadanie w określonym czasie. Po zakończeniu pracy system wystawia ocenę algorytmowi, modyfikuje wektor prawdopodobieństwa za pomocą odpowiedniego operatora T, losuje następny algorytm, pracuje nim, wystawia ocenę itd. Procedura ta może być

przerywana w sytuacji, gdy osiągnięta wartość funkcji celu zadawala stawiającego zadanie lub gdy czas przeznaczony na pracę maszyny wyczerpał się.

Zakłada się, że system optymalizuje funkcje do 20 zmiennych.

Użycie 5 różnych algorytmów optymalizacyjnych spowodowało konieczność jednorazowego wprowadzenia do maszyny pełnego kompletu danych zawierającego wszystkie parametry liczbowe wymagane przez poszczególne metody.

System po wylosowaniu jednego z algorytmów będzie wybierał w odpowiedni sposób z pola danych tylko te wartości, które potrzebne są do realizacji tego algorytmu. Komplet danych składa się z:

a) parametrów wspólnych dla wszystkich algorytmów,
b) parametrów indywidualnych dla każdego algorytmu.

Wspólnymi parametrami są:

- liczba zmiennych,
- liczba ograniczeń wyznaczających obszar dopuszczalny,
- ograniczenia funkcyjne dla zmiennych,
- punkt startu (może nie być podawany - w tym przypadku następuje wylosowanie tego punktu).

System ten został zrealizowany na maszynie ZAM-41. Użytkownik musi utworzyć rozdział napisany w autokodzie SAKO, zawierającym zaprogramowane wyrażenia obliczające wartości funkcji celu i sprawdzające warunki ograniczające obszar decyzji dopuszczalnych. Każde zadanie optymalizacyjne wymaga więc napisania nowego rozdziału użytkownika właściwego dla problemu.

Rozdział taki składa się z dwóch części:

stałej - jednakowej dla każdego zadania zawierającej pewne deklaracje formalne, z punktu widzenia użytkownika nieistotne, warunkujące prawidłowe działanie rozdziału jako części składowej systemu optymalizacyjnego,

wymiennej - pisanej przez użytkownika dla konkretnego zadania, sprawdzającej warunki ograniczające dla punktu bieżącego i obliczającej w tym punkcie wartości funkcji celu

3. ALGORYTMY DZIAŁAJĄCE W SYSTEMIE

3.1. Algorytm błądzący [7]

Oparty na metodzie Monte Carlo, umożliwia:

a) dowolnie dokładne przebadanie całego obszaru, zależnie od liczby przeprowadzonych losowań, przy czym punkty, przez które przechodzi proces są gęsto ułożone w sąsiedztwie brzegu, a rzadko we wnętrzu obszaru. W ten sposób osiąga się szczegółowe zbadanie okolic brzegu, w którym często znajduje się rozwiązanie optymalne;

b) skoncentrowanie się (po pewnej liczbie losowań) na poszukiwaniach w okolicy punktu najlepszego uzyskanego w procesie błądzenia.

Proces błądzenia rozpoczyna się od wylosowania punktu w obszarze dopuszczalnym, obliczenia wartości funkcji celu dla tego punktu, otoczenia go komórką o wymiarze k oraz wylosowania następnego punktu w tej komórce. Dla tego punktu oblicza się wartość funkcji celu i zapamiętuje ją, jeśli była lepsza od poprzedniej. Po każdym "dobrym kroku" tzn. znalezieniu punktu w obszarze dopuszczalnym, następuje zmiana środka komórki bez względu na to, czy krok ten poprawił osiągniętą wartość, czy też nie. Jeżeli natomiast kolejny punkt okaże się niedobry - wówczas nie następuje zmiana środka komórki. Po określonej liczbie prób dobrych powiększa się komórkę dookoła kolejnego punktu. W tak rozszerzonej komórce prowadzi się dalsze losowanie. Po kolejnym otrzymaniu określonej liczby punktów dobrych - następuje ponowne zwiększenie komórki itd. Jeżeli natomiast osiągnie się założoną sekwencję prób złych, to każdorazowo zmniejsza się komórkę. Podkreślić tu należy, że punkty, przez które przechodzi proces, są gęsto ułożone w sąsiedztwie brzegu, a rzadko wewnątrz obszaru. Wnętrze jest jednak zbadane w wystarczający

sposób, aby wykryć wyjątkową (w zadaniach technicznych) sytuację, gdy optimum leży wewnątrz obszaru R.

Po osiągnięciu przyjętej dla zadania liczby prób, poszukiwania ogranicza się do okolicy punktu najlepszego, uzyskanego w procesie błądzenia, przy czym środek komórki zmienia się tylko w przypadku, gdy losując natrafi się na punkt, dla którego wartość funkcji celu jest lepsza od poprzedniej.

3.2. Algorytm poszukujący ekstremum wzdłuż grani funkcji celu [8]

Poszukiwanie ekstremum opiera się na założeniu, że układ zmiennych dający poprawę wartości funkcji celu określa kierunek, wzdłuż którego można oczekiwać dalszej poprawy wyników. Strategia ta jest celowa zwłaszcza wówczas, gdy grzbiet funkcji celu wytycza w przybliżeniu linię prostą.

Postępowanie rozpoczyna się małym krokiem z arbitralnie przyjętego punktu startu. Ulega on zwiększeniu, gdy kolejne kroki pozwalają na poprawę wartości funkcji celu, w przeciwnym wypadku ulega on zmniejszeniu.

3.3. Algorytm losowo-gradientowy*)

Proces optymalizacji przebiega tutaj etapami. Każdy z etapów polega na:

1) losowaniu punktów z kostki aż do wylosowania punktu w obszarze,
2) przechodzeniu metodą gradientową z przyjętym krokiem opierając się na gradiencie liczonym jako różnice skończone z przyrostem do brzegu obszaru, lub do maksimum określonego innym warunkiem,
3) powtórzeniu procedury od 1 aż do osiągnięcia przyjętej liczby rezultatów (z zachowaniem najlepszych z poprzedniego etapu),

*) Koncepcję tego algorytmu zaproponował prof. dr J. Oderfeld

4) uporządkowaniu tych rezultatów metodą histogramową wg wartości funkcji celu,

5) zachowaniu określonej liczby najlepszych rezultatów do następnego etapu i zmianie wymiarów kostki losowania tak, aby obejmowała ona te punkty,

6) wypisaniu wyników,

7) w następnych etapach realizuje się ta sama procedura z pominięciem punktu 2.

3.4. Algorytm "simplex zmodyfikowany" [9]

Zakłada się, że znany jest punkt X, spełniający wszystkie ograniczenia, z którego rozpoczyna się procedurę. Procedura wykorzystuje k punktów, z których jeden jest punktem początkowym. Pozostałe k - 1 punktów stanowią układ punktów, który tworzy się korzystając z generatora liczb pseudolosowych. Losowo wybrany punkt musi spełniać ograniczenia jawne, lecz niekoniecznie pozostałe dane w postaci funkcyjnej. Jeżeli te ostatnie nie są spełnione przez punkt, to punkt przesuwamy o połowę odległości między wylosowanym punktem nie spełniającym warunku, a środkiem ciężkości układu utworzonego przez pozostałe punkty z obszaru dopuszczalnego. Opisane postępowanie powtarzamy aż do trafienia na punkt, który leży w obszarze.

Funkcję celu liczy się w każdym wierzchołku figury geometrycznej, opisanej przez wybrane punkty. Punkt, w którym wartość funkcji celu jest najmniejsza, zastępuje się przez inny, leżący na prostej przechodzącej przez ten najgorszy punkt i środek ciężkości figury utworzonej przez pozostałe punkty. Nowy, tak znaleziony punkt jest lustrzanym odbiciem starego względem środka ciężkości.

Jeżeli tak znaleziony punkt nie spełnia funkcyjnych warunków ograniczających, to dzieli się odcinek między tym punktem a środkiem ciężkości na połowę, sprawdza ponownie warunki, aż do uzyskania punktu dobrego. Jeżeli 5 kolejnych osiągniętych war-

tości funkcji celu są z określoną dokładnością bliskie sobie - to obliczenia zostają przerwane. Aby uniknąć znalezienia ekstremum lokalnego, powtarza się postępowanie zaczynając z różnych punktów startu.

3.5. Metoda Rosenbrocka [10]

Jest modyfikacją metody Gaussa-Seidela. Modyfikacja polega na tym, że po każdym cyklu iteracyjnym, aby przyspieszyć proces szukania punktu optymalnego dokonuje się obrotu układu współrzędnych.

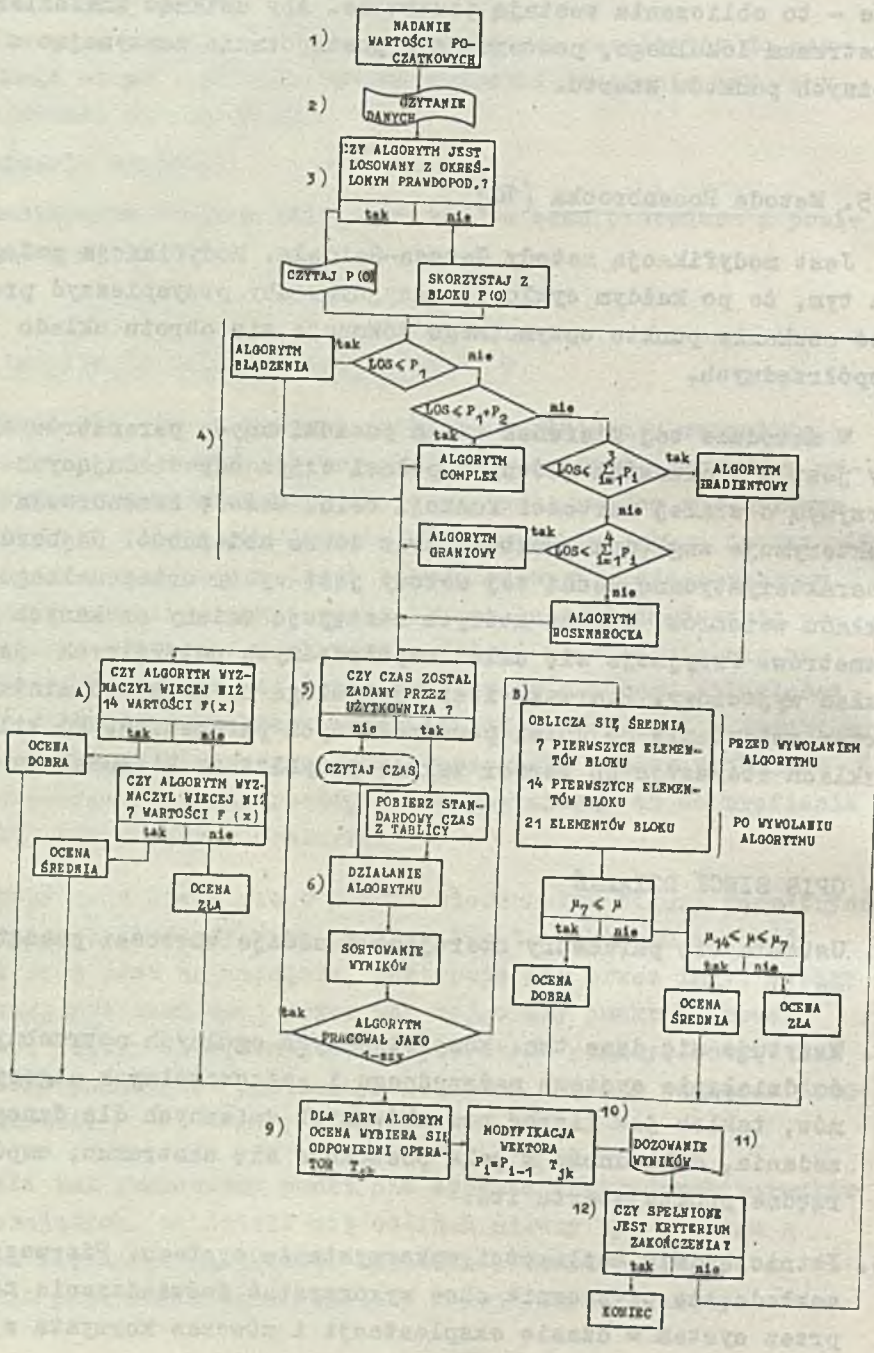
W metodzie tej kierunek zmian poszukiwanych parametrów zgodny jest z kierunkiem głównych półosi elips odpowiadających krzywym o stałej wartości funkcji celu. Metodę Rosenbrocka charakteryzuje względna prostota oraz dobra zbieżność. Najbardziej charakterystyczną cechą tej metody jest wybór ortogonalnego układu wersorów, wzdłuż których następują zmiany szukanych parametrów. Przyjmuje się układ współrzędnych naturalnych jako układ wyjściowy. Pierwsza iteracja polega na szukaniu minimum przez zmienianie po kolei poszczególnych parametrów. Po kilku cyklach iteracyjnych wersor wskaże przybliżony kierunek zwrotu.

4. OPIS SIECI DZIAŁAŃ

1. Ustawia się parametry sterujące i nadaje wartości początkowe.

2. Wczytuje się dane tzn. komplet danych ogólnych potrzebnych do działania systemu nadrzędnego i poszczególnych algorytmów, takich jak liczba poszukiwanych zmiennych dla danego zadania, dokładność z jaką poszukuje się ekstremum, współrzędne punktu startu itd.

3. Istnieją dwie możliwości wykorzystania systemu. Pierwsza zakłada, że użytkownik chce wykorzystać doświadczenie nabyte przez system w czasie eksploatacji i wówczas korzysta z prze-


1)
NADANIE WARTOŚCI POCZĄTKOWYCH
2)
ODCZYTANIE DANYCH
3)
CZY ALGORYTM JEST LOSOWANY Z OKREŚLONYM PRAWDOPOD.?
tak
nie
CZYTAJ P(0)
SKORZYSTAJ Z BLOKU P(0)
4)
ALGORYTM BŁĄDZENIA
LOS ≤ P1
LOS ≤ P1+P2
ALGORYTM COMPLEX
ALGORYTM GRADIENTOWY
ALGORYTM GRANIOWY
ALGORYTM ROSENBROCKA
A)
CZY ALGORYTM WYZNACZYŁ WIĘCEJ NIŻ 14 WARTOŚCI F(x)
OCENA DOBRA
CZY ALGORYTM WYZNACZYŁ WIĘCEJ NIŻ 7 WARTOŚCI F(x)
OCENA ŚREDNIA
OCENA ZŁA
5)
CZY CZAS ZOSTAŁ ZADANY PRZEZ UŻYTKOWNIKA ?
CZYTAJ CZAS
POBIERZ STANDARDOWY CZAS Z TABLICY
6)
DZIAŁANIE ALGORYTMU
SORTOWANIE WYNIKÓW
ALGORYTM PRACOWAŁ JAKO
B)
OBLICZA SIĘ ŚREDNIĄ
7 PIERWSZYCH ELEMENTÓW BLOKU
14 PIERWSZYCH ELEMENTÓW BLOKU
21 ELEMENTÓW BLOKU
PRZED WYWOŁANIEM ALGORYTMU
PO WYWOŁANIU ALGORYTMU
OCENA DOBRA
OCENA ŚREDNIA
OCENA ZŁA
9)
DLA PARY ALGORYM OCENA WYBIERA SIĘ ODPOWIEDNI OPERATOR
10)
MODYFIKACJA WEKTORA
11)
DOZOWANIE WYNIKÓW
12)
CZY SPEŁNIONE JEST KRYTERIUM ZAKOŃCZENIA ?
KONIEC

Rys. 1

chowywanego wektora prawdopodobieństw początkowych. Druga pozwala wprowadzić wektor początkowy dla danego zadania, wtedy kiedy użytkownik z góry może coś powiedzieć o efektywności algorytmów dla zadania. W skrajnym przypadku można np. liczyć tylko jednym z algorytmów.

4. Wybieranie algorytmu odbywa się wg kolejnych oczek sieci działań. W każdym przypadku algorytm wybiera z ogólnego pola danych, te które mu są potrzebne do realizacji strategii.

5. Sprawdza się czy czas dla liczenia pojedynczych algorytmów jest zadany przez użytkownika, czy też wybierany ze standardowej tablicy zależnie od liczby zmiennych parametrów opisujących zadanie.

6, 7. Następnie działa jeden z algorytmów.

Algorytm stanowi jednorozdziałowy program napisany w autokodzie. W każdym algorytmie mieści się standardowy podprogram sortowania lub sekwencja sortująca (dla algorytmów nie mających podprogramu sortowania). W wyniku działania podprogramu sortującego lub sekwencji sortującej otrzymujemy uporządkowane w kolejności od najlepszego do najgorszego tyle punktów ile algorytm zdołał w przydzielonym mu czasie wyznaczyć, nie więcej jednak niż 21 najlepszych. Punkty te są przechowywane w specjalnym bloku. Wartości tego bloku mogą ulegać zmianie tylko podczas sortowania.

8. W zależności od tego czy algorytm pracował jako pierwszy czy jako kolejny - wystawienie oceny odbywa się w różny sposób (A lub B).

9. Dla pary: algorytm, ocena $(A_j O_k)$, definiowanej jako zdarzenie, wybiera się z pamięci odpowiedni operator T_{jk}.

10. Za pomocą wybranego operatora T_{jk} modyfikuje się wektor prawdopodobieństwa.

11, 12. Drukowanie wyników i sprawdzanie kryterium zakończenia, które kończy proces lub powoduje powrót do 4.

Wobec różnorodności zadań optymalizacyjnych trudno ustalić jednolite kryterium kończenia pracy systemu. Zdecydowano się więc na wprowadzenie kilku różnych kryteriów, które będą wprowadzane za pomocą nadania odpowiedniej wartości zmiennej.

5. KORZYSTANIE Z SYSTEMU ORAZ PRZEPROWADZONE EKSPERYMENTY

Przykład 1

Przykład ten nie jest modelem rzeczywistego zjawiska fizycznego. Jednak ze względu na jego dwuwymiarowość ma prostą interpretację graficzną (Rys. 2).

Funkcja celu: $x \cdot y = \min$
Ograniczenia:

$$x^4 - 2x^3 + 3x^2 - 8x + 4 - y \geqslant 0$$

$$\sin(2x) + 12 - y \geqslant 0$$

$$y - \cos(4x) + 6 \geqslant 0$$

$$0 \leqslant x \leqslant 10$$

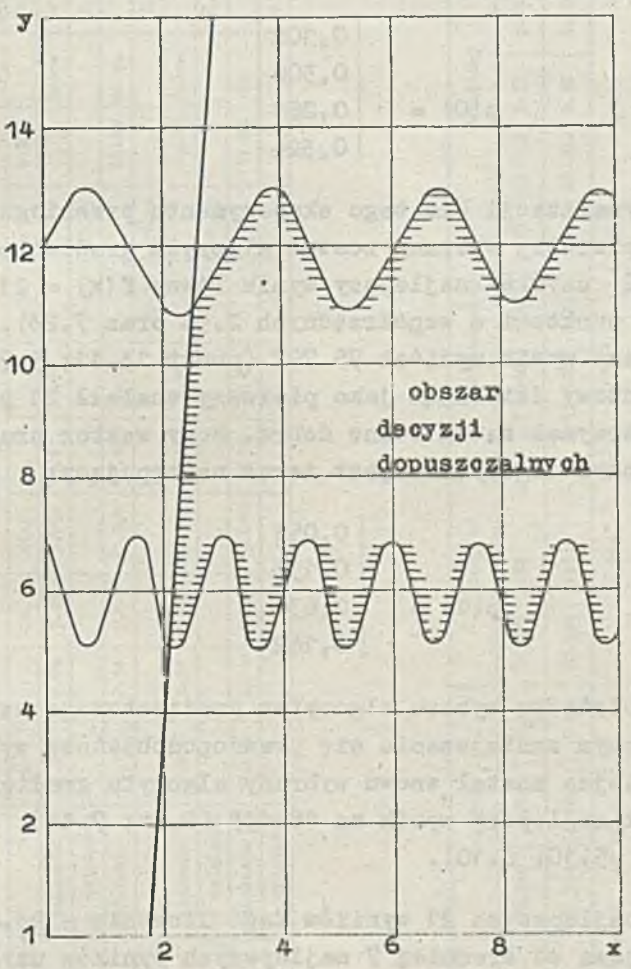
$$0 \leqslant y \leqslant 16$$

Zadanie to zostało wszechstronnie przebadane różnymi algorytmami niezależnymi. Przygotowano je do liczenia systemem SOPT. Sprowadza się to do napisania rozdziału użytkownika. Początek i koniec rozdziału jest standardowy i użytkownik nie wnikając w znaczenie musi je w rozdziale przez siebie pisanym powtórzyć. Reszta wg zasad pisania w autokodzie SAKO należy już do niego. Formuły obejmują bardzo prosty zapis funkcji celu i ograniczeń.

Praktyczne operowanie systemem SOPT na maszynie ZAM-41 jest ułatwione przez drukowanie tekstów na monitorze, przypominając użytkownikowi o poszczególnych czynnościach.

W czasie 2-minutowego liczenia algorytmami niezależnymi uzyskano następujące wyniki:

algorytm błądzący: $f(X) = 19.549;\ x_1 = 2.57;\ x_2 = 7.58$
algorytm complex: $f(X) = 14.377;\ x_1 = 2.58;\ x_2 = 5.56$
algorytm grad. los.: $f(X) = 11.546;\ x_1 = 2.30;\ x_2 = 5.03$
algorytm graniowy: $f(X) = 13.715;\ x_1 = 2.33;\ x_2 = 5.88$



Rys. 2

Najbardziej efektywnym dla zadania okazał się algorytm gradientowo losowy ($f(X) = 11.546$), a najgorszym algorytm błądzący ($f(X) = 19.549$).

Pierwsze obliczenie przeprowadzone systemem SOPT podano w tabeli 1. Czas przydzielony algorytmowi wyniósł 12'. Wektor prawdopodobieństw początkowych (w systemie działały 4 algorytmy) był tutaj następujący:

$$p(0) = \begin{bmatrix} 0,102 \\ 0,306 \\ 0,268 \\ 0,324 \end{bmatrix}$$

Proces optymalizacji dla tego eksperymentu przebiegał następująco: jako pierwszy wybrany został algorytm gradientowy, który w czasie 12' uzyskał najlepszy wynik równy $f(X) = 21.297$ (odpowiada to punktowi o współrzędnych 2.94 oraz 7.23). Dwudziesty pierwszy wynik wyniósł 75.792 (punkt 13.11; 5.78). Algorytm gradientowy działając jako pierwszy znalazł 21 punktów w obszarze, otrzymał zatem ocenę dobrą. Nowy wektor prawdopodobieństwa wyboru algorytmów jest teraz następujący:

$$p(1) = \begin{bmatrix} 0,051 \\ 0,153 \\ 0,634 \\ 0,162 \end{bmatrix}$$

Prawdopodobieństwo wyboru algorytmu gradientowego wzrosło, przy jednoczesnym zmniejszeniu się prawdopodobieństw wyboru pozostałych. Kolejno został znowu wybrany algorytm gradientowy, który poprawił najlepszy wynik na 18.135 (2.54; 7.14), a 21 wynik na 31.097 (5.10; 6.10).

Średnia z najlepszych 21 wyników tego liczenia - 23.84. Była zatem mniejsza od średniej 7 najlepszych wyników uzyskanych przednio (24.52). I tym razem algorytm otrzymał ocenę dobrą.

Wektor prawdopodobieństwa po tym liczeniu zmienił się następująco:

Tabela 1

Czas działania poszczególnego algorytmu 12		Najlepszy wynik po zadziałaniu algorytmu			21 kolejny najlepszy wynik po zadziałaniu algorytmu			Średnia			Ocena	Prawdopodobieństwa wyboru algorytmu			
												Błądz.	Complex	Grad.	Graniowy
Algorytm		f(x)	x_1	x_2	f(x)	x_1	x_2	z 7	z 14	z 21		0.102	0.306	0.266	0.324
Gradientowy	działanie	21.297	2.54	7.23	75.792	13.11	5.78				dobra				
	po działaniu							24.52	33.15			0.051	0.153	0.634	0.162
Gradientowy	działanie	18.135	2.54	7.14	31.097	5.10	6.10			23.84	dobra				
	po działaniu							20.16	21.50			0.026	0.077	0.818	0.031
Błądzący	działanie	11.669	2.31	5.05	13.309	2.38	5.59			12.05	dobra				
	po działaniu							11.72	11.79			0.513	0.036	0.409	0.041
Graniowy	działanie	11.669	2.31	5.05	13.010	2.34	5.57			11.98	zła				
	po działaniu											0.520	0.047	0.416	0.021

Tabela 2

Algorytm	Błądzący			Complex			Grad-Losowy			Graniowy			SCPT		
Najlepszy wynik	19.549	2.57	7.58	14.377	2.58	5.56	11.546	2.29	5.03	13.715	2.33	5.88	11.669	2.30	5.05
7 kolejny wynik	26.964	2.66	10.09	20.998	2.79	7.52	11.546	2.29	5.03	45.592	2.36	6.60	11.764	2.32	5.06
Czas działania	2 min.			2 min.			2 min.			2 min.			48 s		

$$p\ (2) = \begin{bmatrix} 0.026 \\ 0.077 \\ 0.818 \\ 0.081 \end{bmatrix}$$

W kolejnym losowaniu został wybrany algorytm błądzący, który poprawił najlepszy wynik z 18.135 na 11.669, a średnia 21 punktów po jego działaniu wynosiła 12.05. Zatem znowu lepiej od średniej 7 najlepszych z poprzedniego losowania. Tym razem ocena dobra dla algorytmu błądzącego spowodowała zwiększenie prawdopodobieństwa jego wyboru, a zmniejszenie szans wyboru dla pozostałych. W następnym losowaniu został wybrany algorytm graniowy, który nie poprawił już najlepszego wyniku a i średnia 21 wyników po jego działaniu była gorsza od 14 najlepszych wyników z poprzedniego działania. Algorytm graniowy otrzymał ocenę złą, co spowodowało odpowiednie zmiany w składowych wektora prawdopodobieństwa wyboru algorytmów.

Łączny czas liczenia czterech algorytmów w systemie wyniósł 48', a uzyskany wynik (11.669) jest bardzo bliski globalnemu ekstremum dla tego zadania. Zwraca tutaj uwagę fakt,że algorytm błądzący, który był bardzo mało efektywny przy niezależnym liczeniu, zadziałał rewelacyjnie po uprzednim przygotowaniu dokonanym przez algorytm gradientowy. Zestawienie wyników podaje tabela 2.

Z zadaniem tym przeprowadzono również inne eksperymenty. Pierwszy podaje (tab. 3), gdzie działał ciągle ten sam algorytm gradientowy, który stopniowo poprawiał wyniki aż do 14.629 (2.46; 5.95). Trzykrotnie algorytm gradientowy otrzymywał ocenę dobrą, a wektor prawdopodobieństwa zmieniał się następująco:

$$\begin{bmatrix} 0.102 \\ 0.306 \\ 0.268 \\ 0.324 \\ 0.000 \end{bmatrix} \begin{bmatrix} 0.051 \\ 0.153 \\ 0.634 \\ 0.162 \\ 0.000 \end{bmatrix} \begin{bmatrix} 0.048 \\ 0.060 \\ 0.829 \\ 0.062 \\ 0.000 \end{bmatrix} \begin{bmatrix} 0.030 \\ 0.030 \\ 0.889 \\ 0.034 \\ 0.007 \end{bmatrix} \begin{bmatrix} 0.023 \\ 0.023 \\ 0.903 \\ 0.032 \\ 0.019 \end{bmatrix}$$

Widać, jak zwiększa się prawdopodobieństwo wyboru algorytmu gradientowego tak długo jak długo otrzymywał on oceny dobre i dostateczne.

Tabela 3

Czas działania poszczególnego algorytmu 12		Najlepszy wynik po zadziałaniu algorytmu			21 kolejny najlepszy wynik po zadziałaniu algorytmu			Średnia			Ocena	Prawdopodobieństwa wyboru algorytmu				
												Błądz.	Complex	Grad.	Gran.	Rosen.
Algorytm		f(x)	x_1	x_2	f(x)	x_1	x_2	z 7	z 14	z 21		0.102	0.306	0.268	0.324	0.000
Gradient	działanie	17.423	2.76	6.33	37.050	5.63	6.58				dobra					
	po działaniu							20.233	21.933			0.057	0.153	0.634	0.162	0.000
Gradient	działanie	15.982	2.38	6.72	20.443	2.65	7.78			18.324	dobra					
	po działaniu							16.975	17.592			0.048	0.060	0.829	0.062	0.000
Gradient	działanie	15.982	2.38	6.72	19.841	2.84	6.69			16.674	dobra					
	po działaniu							16.560	17.180			0.030	0.030	0.889	0.034	0.007
Gradient	działanie	15.982	2.38	6.72	18.135	2.54	7.14			17.180	dost.					
	po działaniu							16.060	16.560			0.023	0.023	0.903	0.032	0.019
Gradient	działanie	14.629	2.46	5.95	17.420	2.75	6.33			17.010	ndst.					
	po działaniu															

- 27 -

Tabela 4

Czas działania poszczególnego algorytmu - 15		Najlepszy wynik po zadziałaniu algorytmu			21 najlepszy wynik po zadziałaniu algorytmu			Średnia			Ocena	Prawdopodobieństwa wyboru algorytmu				
												błądzący	complex	gradientowy	granicowy	Rosenbrocka
Algorytm		f(x)	x_1	x_2	f(x)	x_1	x_2	z 7	z 14	z 21		0.102	0.306	0.268	0.324	0.000
Complex	działanie	18.330	2.41	7.60	61.550	5.34	5.76				dost.					
	po działaniu							29.155				0.114	0.403	0.197	0.225	0.063
Błądzący	działanie	17.338	2.50	6.94	34.580	3.92	8.82			27.153	dobra					
	po działaniu							21.143	24.630			0.557	0.202	0.098	0.112	0.031
Complex	działanie	12.553	2.45	5.12	34.743	3.39	9.24			24.495	dost.					
	po działaniu							18.027	21.724			0.341	0.351	0.112	0.119	0.076
Błądzenia	działanie	12.553	2.45	5.12	27.017	4.27	6.32			22.16	nie-dost.					
	po działaniu							17.722	20.097			0.170	0.300	0.181	0,184	0.164
Rosen-brocka	działanie	11.639	2.31	5.03	26.793	3.75	7.14									
	po działaniu															

Z przykładem tym przeprowadzono również dwa inne eksperymenty, w których "uczestniczyło" już po pięć algorytmów: błądzący, complex, gradientowy, graniowy oraz Rosenbrock. Pierwszy eksperyment był przeprowadzony dla czasu działania poszczególnego algorytmu = 12'. Drugi natomiast charakteryzował się czasem działania równym 15'. W eksperymencie tym działały kolejno algorytmy: complex, błądzący, complex, błądzenia i Rosenbrocka. Zwrócić tutaj należy uwagę, że algorytm complex pracując jako pierwszy znalazł 9 punktów (tab. 4) w obszarze dopuszczalnym. Otrzymał więc ocenę dostateczną. Już pracujący jako drugi algorytm błądzący uzupełnił blok wyników do 21. Kolejne działanie complexu bardzo istotnie poprawiło wynik na 12.553, lecz z tego powodu, że średnia 21 wyników nie była imponująca, otrzymał on ocenę dostateczną. Ustalonym kryterium przerwania w tym eksperymencie było zadziałanie 5 kolejnych algorytmów. Należałoby tutaj podkreślić oczywistą efektywność systemu. Wskazuje to tab. 2, gdzie system uzyskał wynik zbliżony do najlepiej działającego algorytmu gradientowo-losowego, który działał 2 minuty, przy czym system SOPT pracował tylko 48 s.

Przykład 2 - reduktor planetarny

Zastosowanie systemu optymalizacyjnego pokazano na przykładzie reduktora silnika lotniczego (Rys. 3). Dane reduktora są następujące:

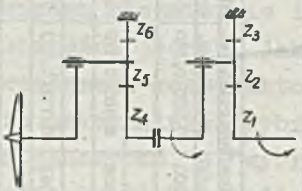
Moc wolnej turbiny	1200 kM
wymagana prędkość obrotowa śmigła	1500 obr/min
prędkość obrotowa wirnika turbiny napędzającej śmigło	18000 obr/min

Projekt reduktora obejmuje obliczenia wytrzymałościowe wszystkich kół zębatych, tzn. obliczenia sprawdzające zazębień najbardziej obciążonych: naprężenia od zginania, naprężenia styczne, powierzchniowego w punktach styku oraz temperaturę w punkcie początku współpracy zębów. Poszukiwanymi parametrami przekładni zębatej są:

$b = x_1$ szerokość wieńca koła zębatego,

$m = x_2$ moduł normalny koła zębatego,

$z = x_3$ liczba zębów koła słonecznego,
$k = x_4$ liczba satelitów przekładni zębatej.



Rys. 3

Ograniczenia nałożone na konstrukcję opisują się następująco:

1. na zginanie	$\varphi_1(x) = 1.16\ m^2 zbk - \sqrt{mz} - 2.2 > 0$
2. na docisk	$\varphi_2(x) = 0.016\ km^2 z^2 b - \sqrt{mz} - 2.2 \geqslant 0$
3. na grzanie	$\varphi_3(x) = 0.0035\ bm^2 z^2 - 1 \geqslant 0$
4 na graniczną liczbę zębów	$\varphi_4(x) = z - 28 \geqslant 0$
5. na względną szerokość wieńca zębatego	$\varphi_5(x) = b - 3m \geqslant 0$
6. na względną szerokość wieńca zębatego	$\varphi_6(x) = 15\ m - b \geqslant 0$
7. na minimalną liczbę satelitów	$\varphi_7(x) = k - 2 \geqslant 0$
8. sąsiedztwa	$\varphi_8(x) = \frac{z}{2}\left(3\sin\frac{\pi}{k} - 2\right) - 2 \geqslant 0$
9 montażowy	$\varphi_9(x) = MOD(z, 2) = 0$
10. gabarytowy	$\varphi_{10}(x) = 37 - 2zm \geqslant 0$
11. szerokość wieńca	$\varphi_{11}(x) = b \geqslant 0$
12. moduł zęba	$\varphi_{12}(x) = m > 0$

funkcja celu

$$f(x) = \frac{\pi}{4}\ b\ m^2 z^2 \left(\frac{k}{2} + 1\right)$$

Tabela 5

Czas działania poszczególnego algorytmu 1 min		Najlepszy wynik po zadziałaniu algorytmu					21 kolejny wynik po zadziałaniu algorytmu					Średnia			Ocena	Prawdopodobieństwa wyboru algor.				
Algorytm		f(x)	x_1	x_2	x_3	x_4	f(x)	x_1	x_2	x_3	x_4	z 7	z 14	z 21		błądz. 0.200	comp. 0.200	grad. 0.200	gran. 0.200	Ros. 0.200
Błądzący	działanie	702.50	1.72	0.4	36	3	1102.00	1.73	0.5	36	3				dobra					
	po działaniu											702.67	702.74			0.600	0.100	0.100	0.100	0.100
Gradientowy	działanie	571.54	2.41	0.25	44	3	705.34	1.73	0.4	36	3			687.74	dobra					
	po działaniu											657.34	680.02			0.300	0.090	0.550	0.050	0.050
Rosenbrock	działanie	562.30	1.55	0.4	34	3	702.72	1.73	0.4	36	3			643.88	dobra					
	po działaniu															0.150	0.025	0.275	0.025	0.525

Tabela 6

Czas działania poszczególnego algorytmu 1 min		Najlepszy wynik po zadziałaniu algorytmu					21 kolejny wynik po zadziałaniu algorytmu					Średnia			Ocena	Prawdopodobieństwa wyboru algor.				
Algorytm		f(x)	x_1	x_2	x_3	x_4	f(x)	x_1	x_2	x_3	x_4	z 7	z 14	z 21		błądz. 0.200	comp. 0.200	grad. 0.200	gran. 0.200	Ros. 0.200
Błądzący	działanie	716.33	2.81	0.3	38	3	1942.36	2.74	0.5	38	3.				dobra					
	po działaniu											716.33	752.74			0.600	0.100	0.100	0.100	0.100
Rosenbrock	działanie	561.27	1.32	0.35	42	3	970.99	2.30	0.35	38	3			720.30	dst.					
	po działaniu											624.45	670.39			0.363	0.113	0.113	0.113	0.300
Gradientowy	działanie	561.27	1.32	0.35	42	3	716.33	2.81	0.30	38	3			551.42	dst.					
	po działaniu											577.74	618.96			0.244	0.119	0.308	0.119	0.213
Graniczny	działanie	561.06	2.45	0.30	36	3	642.50	2.81	0.30	36	3			581.86	dst.					
	po działaniu											561.87	563.23			0.184	0.122	0.216	0.309	0.169
Błądzący	działanie	561.06	2.45	0.30	36	3	642.50	2.81	0.30	36	3			581.86	ndst.					
	po działaniu											561.87	563.23			0.092	0.186	0.233	0.280	0.209
Rosenbrock	działanie	561.06	2.45	0.30	36	3	565.93	1.65	0.30	44	3			563.32	ndst.					
	po działaniu											561.43	562.36			0.171	0.218	0.242	0.265	0.105
Gradientowy	działanie	561.06	2.45	0.30	36	3	565.33	1.25	0.40	38	3			563.09	ndst.					
	po działaniu															0.211	0.234	0.121	0.257	0.177

Funkcja ta opisuje objętość reduktora przy założeniu, że koła są pełne. Kryterium takie wydaje się słuszne, bowiem w urządzeniach lotniczych sprawą niezmiernie istotną jest ciężar urządzenia. W rzeczywistości jednak, jak wiadomo koła przekładni nie są pełne. Doświadczony konstruktor lotniczy potrafi jednak z pełnego gabarytu maksymalnie "wybrać" zbędny materiał. Ponadto między pełnym gabarytem, a "wybranym" zachodzi z dobrym przybliżeniem proporcjonalność (grubość piast, użebrowanie, głębokości wybrań są na ogół funkcjami szerokości wieńca lub modułu).

W zadaniu tym określono, że zmienna k - opisująca liczbę satelitów może być jedynie liczbą całkowitą, podobnie liczba zębów koła słonecznego. Wartości modułów wybierane są zgodnie z polską normą: 2, 2.5, 3, 3.5... mm.

Zadanie sformalizowane jak wyżej przez konstruktora zostaje napisane w autokodzie. Po wprowadzeniu rozdziału użytkownika do maszyny poddano zadanie obróbce przez system optymalizacyjny. Wyniki tego działania ujęto w tabelach 5 i 6. Tabela 5 pokazuje przebieg, w którym zadziałały 3 algorytmy każdy po 1 minucie. Założono jako początkowe prawdopodobieństwa wyboru algorytmów po 0.2, tzn., że każdy z pięciu miał na początku tę samą szansę wyboru. Pierwszy zadziałał błądzący, który uzyskując 21 punktów w obszarze dopuszczalnym otrzymał ocenę dobrą. Prawdopodobieństwo zmieniło się zatem dość istotnie. Jako drugi został wybrany algorytm gradientowy, który zgodnie z opisaną zasadą oceniania, również otrzymał dobrą ocenę, podobnie jak działający ostatni algorytm Rosenbrocka.

Ostatni wynik otrzymany po 3 minutach to $f(x) = 562.30$ cm^3, a punkt znaleziony ma współrzędne (1.55, 0.4, 34, 3). Dwudziesty pierwszy punkt (1.73, 0.4, 36, 3) ma funkcję celu 702.72 cm^3.

Eksperyment z tym przykładem powtórzono, przyjmując powtórnie wektor początkowy $p(0) = (0.2; 0.2; 0.2; 0.2; 0.2)$. Działały tutaj kolejno (tab. 6) algorytmy: błądzący, Rosenbrock, gradientowy, graniowy, błądzący, Rosenbrock i gradientowy.

Cała próba trwała 7 minut, a najlepszy osiągnięty wynik to 561.06 cm^3 odpowiadający punktowi (2.15; 0.3; 36; 3). Tutaj wynik 21 zbiegł się blisko z optymalnym i wyniósł 565.33 cm^3, co odpowiada punktowi (1.25; 0.4; 38; 3).
Punkt

$$\begin{bmatrix} 2.45 \\ 0.3 \\ 36 \\ 3 \end{bmatrix}$$

dla którego funkcja celu = 561.06 stanowi poszukiwane ekstremum. Leży ono na brzegu opisanym przez warunek na grzanie.

6. ZAKOŃCZENIE

Pozytywne wyniki doświadczeń spowodowały zainteresowanie innych użytkowników maszyn cyfrowych, w wyniku czego opracowano dwa inne systemy, różniące się nieco algorytmem liczenia. Są to systemy na EMC ODRA 1304 oraz minikomputer. W fazie opracowania znajduje się system na maszyny jednolitego systemu serii RIAD.

Literatura

[1] GASS S.I.: Programowanie liniowe. Metody i zastosowania, PWN, Warszawa 1963

[2] GARTH P., Mc CORMICK W. Ch., MYLANDER III, A. FIACCO: IBM Catalog of Programs, Program No 70-40-H2-3189 Sumtrac (Sumpt.)

[3] MUGELE R.A.: A Nonlinear Digital Optimizing Program for Process Control Systems, Proceedings for the Spring Joint Computer Conference, San Francisco, 1962, 1

[4] CASEY J.K., RUSTAY R.C.: AID - A General Purpose Computer Programs for Optimization, in Recent Advances in Optimization Techniques, John Wiley and Sons 1966

[5] FLOOD M.M., LEON A.: Adaptive Code for Optimization GROPE, In Recent Advances in Optimization Techniques. John Wiley and Sons, 1966

[6] THRALL R.M., COOMBS Ch., DAVIS R.L.: Decision Processes, John Wiley and Sons. Inc. 1954

[7] GOLIŃSKI J.: O badaniu pewnego procesu błądzenia zastosowanego do optymalnej syntezy maszyn, Archiwum Budowy Maszyn, 1968, t. XV, z. 2

[8] WILDE D.J.: Optimum Seeking Methods, Prentice Hall Inc., Englewood Cliffs, 1964

[9] BOX M.J.: A New Method of Constrained Optimization and a Comparison with Other Methods, The Computer Journal, 1964, t. 7, nr 3

[10] ROSENBROCK H.H.: An Automatic Method for Finding the Greatest or the Least Value of a Function, The Computer Journal 1960, t3, nr 2.

АДАПТАЦИОННАЯ СИСТЕМА НЕЛИНЕЙНОЙ ОПТИМАЛИЗАЦИИ

Резюме

В работе указаны исходные данные оптималиэационной системы для средней величины вычислительных цифровых машин. Система может быть использована для решения технических и экономических задач с нелинейной функцией цели и с нелинейными ограничениями. Система заключает пять детерминистических и стохастических алгоритмов. Главный алгоритм, основанный на работах Буша и Мостеллера случайно выбирает один из пяти алгоритмов, который работает в течении определённого времени, а потом проверяется эффективность его действия.

Результат этой эффективности оценивается и на этой основе происходит модификация вектора вероятности последовательного выбора алгоритма.

Примеры указывают результаты экспериментов проведённых разными алгоритмами причиняющие затруднение по отношению к выбору алгоритма для решения, и доказывают правильность дальнейших работ по системам оптималиэации нелинейного программирования.

AN ADAPTIVE NONLINEAR OPTIMIZATION SYSTEM

Summary

The paper presents foundations of an optimization system for medium size computers. The system may be used to solve technical or economic problems with nonlinear objective functions and constraints. The system contains five algorithms of deterministic and stochastic category. The master algorithm, based on ideas of Bush and Mosteller, choses on random one of these five. It performs computations for a certain time then its effectiveness is being checked. The results are estimated and on this basis follows the modification of the probability vector of the succeeding choice of algorithm.

Examples solved by different algorithms are presented. The results needed for the definition of the initial vector illustrate difficulties the user meets while chosing the proper algorithm. These difficulties, often occuring in practice, prove the demand for further studies on optimization systems.

Prace IMM
Zeszyt 3
© 1975.10

681.325.65.001.13

ZASTOSOWANIE DEKOMPOZYCJI DO SYNTEZY UKŁADÓW KOMBINACYJNYCH WOLNYCH OD RYZYKA DLA ZMIAN PRZYLEGŁYCH

Krzysztof WALCZAK

Instytut Maszyn Matematycznych
Politechniki Warszawskiej

Pracę złożono 23.04.1974

Rozważono problem zastosowania dekompozycji do syntezy układów kombinacyjnych wolnych od ryzyka. Podano twierdzenia umożliwiające zmniejszenie obszaru przeszukiwanych funkcji oraz sformułowano warunki rozstrzygające kiedy dekompozycja iloczynowa zawiera ryzyko. Zamieszczono algorytmy syntezy układów kombinacyjnych.

1. WSTĘP

Zastosowanie dekompozycji do syntezy układów kombinacyjnych umożliwia zrealizowanie funkcji dekomponowanej za pomocą znacznie mniejszej liczby elementów niż funkcji niedekomponowalnej takiej samej liczby zmiennych [6].

W pracy rozpatrywane będą dwa rodzaje dekompozycji: dekompozycja prosta oraz dekompozycja iloczynowa.

Funkcja $F(x_1, \ldots, x_n) = F(X)$ posiada dekompozycję prostą wtedy i tylko wtedy, gdy istnieją funkcje G i H takie, że

$$F(X) = G(H(A, B), B, C)$$

gdzie A, B, C są zbiorami zmiennych takimi, że $A \cup B \cup C = X$, $A \cap B = \emptyset$, $B \cap C = \emptyset$, $A \cap C = \emptyset$ oraz funkcja H(A, B) zależy od

zmiennych ze zbiorów A i B, a funkcja G(H, B, C) zależy od zmiennej H i zmiennych ze zbiorów B i C.

Funkcja F(X) posiada dekompozycję iloczynową wtedy i tylko wtedy, gdy istnieją funkcje G i H takie, że

$$F(X) = G(A, B) \cdot H(B, C)$$

gdzie zbiory A, B, C spełniają zależności: $A \cup B \cup C = X$, $A \cap B = \emptyset$, $B \cap C = \emptyset$, $A \cap C = \emptyset$.

Dekomponowanie danej funkcji boolowskiej, przeprowadzane będzie zgodnie z warunkami podanymi w [2], które przytoczono poniżej.

Warunkiem koniecznym istnienia dekompozycji ogólnej jest spełnienie dla dowolnych zbiorów zmiennych A_1 i A_2 takich, że $A_1 \cup A_2 = A$, $A_1 \cap A_2 = \emptyset$ następującej zależności między pochodnymi funkcji F:

$$F_{A_1} \cdot F_{A_2 C} = F_{A_2} \cdot F_{A_1 C}$$

gdzie F_{A_1}, F_{A_2}, $F_{A_1 C}$, $F_{A_2 C}$ oznaczają pochodną [1] funkcji F ze względu na wszystkie zmienne odpowiednio ze zbiorów A_1, A_2, $A_1 \cup C$, $A_2 \cup C$. (Np. $F_{A_1 C} = \frac{\partial F}{\partial a_1 \dots \partial a_k \partial c_1 \dots \partial c_m}$, $\{a_1, \dots a_k\} = A_1$, $\{c_1, \dots, c_m\} = C$). Jeśli natomiast prawdziwy jest związek $F_A \cdot F_C = F \cdot F_{AC}$, wówczas spełniony jest warunek konieczny istnienia dekompozycji iloczynowej. Powyższe warunki dla dekompozycji elementarnych tzn. postaci $F = G(H(a_1, a_2, B), B, c)$ lub $F = G(a, B) \cdot H(B, c)$ są dostateczne. W [2] ponadto udowodniono, że jeżeli dla dowolnej dekompozycji $F = G(H(A,B), B, C)$ istnieją dekompozycje podzbiorowe $F = G(H,(A^*, B^*), B^*, C^*)$, $B \subset B^*$, $A^* \subseteq A$, $C^* \subseteq C$, wówczas wyżej podane warunki konieczne są również dostateczne. Stwierdzenie to pozwala proces dekompozycji rozpoczynać od dekompozycji elementarnych, dla których warunki konieczne są jednocześnie dostateczne, a następnie poprzez sprawdzanie dekompozycji podzbiorowych i warunku koniecznego zmniejszać w miarę możliwości zbiór B. Jeżeli na przykład istnieją dekompozycje podzbiorowe $F = G(H(a_1, a_2, B_1), B_1, C_1)$ oraz $F = G(H(a_1, a_2, B_2), B_2, c_2)$ i jest spełniony warunek

$F_{a_1} \cdot F_{a_2 c_1 c_2} = F_{a_2} \cdot F_{a_1 c_1 c_2}$, to istnieje dekompozycja $F = G(H(a_1, a_2, B_3), B_3, c_1, c_2)$, gdzie $B_3 = B_1 \cap B_2$.

Jednym z podstawowych zadań syntezy układów kombinacyjnych jest uwolnienie sieci od ryzyka. Przedmiotem niniejszych rozważań będzie ryzyko strukturalne [3, 5, 7]. Jest ono zależne jedynie od struktury układu, a nie od jego funkcji i może być wyeliminowane przez odpowiednią realizację układu [5]. Przyjęto, że opóźnienia szkodliwe są ograniczone, co jest zgodne z modelem układu kombinacyjnego podanym w pracach [3, 5, 7].

W pracy założono, że rozważane są jedynie zmiany przyległe, tzn. takie, że w czasie przejścia zmienia się wartość jednej zmiennej wejściowej. Ograniczenie to jest uzasadnione, ponieważ układy kombinacyjne są częściami składowymi układów sekwencyjnych, w których ze względu na zjawiska szkodliwe dopuszcza się najczęściej tylko zmiany przyległe na wejściu. Przy powyższym założeniu oraz założeniach podanych w następnej części pracy układ może zawierać jedynie statyczne ryzyko strukturalne, które dla zmian przyległych było zdefiniowane w pracy [5] jako ryzyko statyczne, jednak w niniejszej pracy dla zachowania jednolitości stosowane będą definicje ryzyk podane w [3, 7] dla zmian dowolnych.

Wykorzystując twierdzenie podane w pracy [5], można zrealizować dowolną funkcję boolowską za pomocą układu kombinacyjnego wolnego od ryzyka strukturalnego, który jednak nie zawsze jest minimalny.

Celem niniejszej pracy jest podanie metody syntezy układów kombinacyjnych wolnych od ryzyka strukturalnego w sposób zbliżony do minimalnego, poprzez zdekomponowanie danej funkcji boolowskiej.

2. METODA SYNTEZY FUNKCJI BOOLOWSKICH ZA POMOCĄ DEKOMPOZYCJI

Na wstępie przyjmiemy, że realizacje funkcji G(H, B, C), H(A, B) dla dekompozycji prostej i G(A, B), H(B, C) dla dekompozycji iloczynowej są wolne od ryzyk strukturalnych statycz-

nych i dynamicznych oraz, że rozważamy tylko zmiany przyległe, co powoduje, że dla rozpatrywanych par następstwa funkcja F jest wolna od ryzyk funkcyjnych[3, 7].

2.1. Dekompozycja prosta

Przy powyższych założeniach realizacja funkcji F(X) nie zawsze musi być wolna od ryzyka strukturalnego. Wynika to z faktu, że funkcja G zmiennych H, B, C może zawierać ryzyko funkcyjne. Może to być jedynie ryzyko statyczne, ponieważ możliwa jest zmiana najwyżej dwóch zmiennych wejściowych funkcji G. Ryzyko to może występować tylko dla tych par następstwa, dla których zmienna aktywna, tzn. zmieniająca się w czasie przejścia, należy do zbioru B. Wtedy bowiem może zmieniać swą wartość funkcja H, a zatem funkcja G ma dwie zmienne aktywne (H oraz zmienną $b \in B$). Przy powyższych założeniach funkcja G nie może zawierać dynamicznego ryzyka funkcyjnego, co powoduje, że realizacja funkcji F jest wolna od dynamicznego ryzyka strukturalnego.

Tabela 1

c_1c_2 \ Hb	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	0	1	0
10	1	1	1	0

Przykład

Niech $F(a_1, a_2, b, c_1, c_2) = G(H(a_1, a_2, b), b, c_1, c_2)$, gdzie $G = c_1'b'H \vee c_1'c_2b' \vee c_1'c_2H \vee c_1c_2'b \vee c_2'b\,H' \vee c_1\,bH \vee c_1c_2'H'$ oraz $H = a_1'b' \vee a_1'a_2 \vee a_1a_2'$. Dla przejścia [00000, 00100] wartość funkcji H zmienia się z 1 na 0, a funkcja G zawiera statyczne ryzyko funkcyjne. Mamy bowiem:

G (1,0,0,0) = 1

G (1,1,0,0) = 0

$G(0,0,0,0) = 0$
$G(0,1,0,0) = 1.$

Ilustruje to tabela 1, będąca tabelą prawdy funkcji $G(H, b, c_1, c_2)$.

W powyższym przypadku funkcja G zawiera statyczne ryzyko funkcyjne, a zatem układ kombinacyjny realizujący funkcję F zawiera ryzyko strukturalne. Zjawisko to wynika z faktu, że istnieją dwie różne drogi sygnału b, przy czym każda z nich należy do realizacji innej funkcji.

Z powyższych rozważań wynika, że przy dekompozycyjnej metodzie syntezy zachodzi ścisły związek pomiędzy realizacją układu a jego funkcjami składowymi G i H. Istnienie ryzyka strukturalnego w układzie kombinacyjnym zależy od tego, czy G zawiera ryzyko funkcyjne. Ponieważ realizacje funkcji G i H są wolne od ryzyk strukturalnych, realizacja funkcji F wolna od ryzyka strukturalnego istnieje wtedy i tylko wtedy, gdy istnieje funkcja G wolna od ryzyka funkcyjnego.

Obecnie pokażemy, że przy powyższych założeniach dla danej pary następstwa zawsze można podać realizację funkcji $F = G(H(A, B), B, C)$ wolną od ryzyka strukturalnego. W tym celu utwórzmy macierz M będącą przekształconą tabelą prawdy funkcji F, której kolumny opisane są przez wartości zmiennych ze zbiorów A i B, a wiersze przez wartości zmiennych ze zbiorów B i C. W macierzy tej wyróżnia się bloki odpowiadające jednakowym wartościom zmiennych $b \in B$, które są tabelami prawdy dla funkcji $F(A, B^W, C)$, gdzie symbol B^W oznacza wektor konkretnych wartości zmiennych należących do zbioru B. Pozostałe miejsca w macierzy M są nieokreślone.

Przykład

Macierz opisująca funkcję F z poprzedniego przykładu przyjmuje postać podaną w tabeli 2.

Tabela 2

bc_1c_2 \ ba_1a_2	000	001	011	010	100	101	111	110	
000	1	1	0	1	–	–	–	–	α_0
001	1	1	1	1	–	–	–	–	
011	0	0	0	0	–	–	–	–	
010	0	0	1	0	–	–	–	–	α_1
100	–	–	–	–	1	0	1	0	β_0
101	–	–	–	–	0	1	0	1	β_1
111	–	–	–	–	0	1	0	1	
110	–	–	–	–	1	1	1	1	

W [4] udowodniono następujące twierdzenie, które będzie wykorzystane w dalszej części pracy.

Twierdzenie 1

Dekompozycja prosta funkcji F istnieje wtedy i tylko wtedy, gdy w macierzy M krotność kolumn w każdym z bloków jest nie większa niż 2.

Obecnie udowodnimy następujące twierdzenie.

Twierdzenie 2

Jeżeli realizacje funkcji G i H są wolne od ryzyk strukturalnych i rozważane są jedynie zmiany przyległe, to dla danej pary następstwa zawsze można podać realizację funkcji
F = G(H(A, B), B, C) wolną od ryzyka strukturalnego.

Dowód

Rozważamy tylko te pary następstwa, dla których zmienia swą wartość zmienna $b \in B$, albowiem przy wszystkich innych zmianach układ jest wolny od ryzyka na mocy założeń. Dla uproszczenia przyjmujemy, że zbiór B jest jednoelementowy. Nie zmniejszy to ogólności rozważań, ponieważ dla danej pary następstwa istotne są tylko bloki odpowiadające aktywnej zmiennej b. Zatem macierz M zawierać będzie dwa bloki: pierwszy określony dla

b = 0, a drugi dla b = 1. Zmiana wartości wejścia b powoduje więc przejście z jednego bloku do drugiego.

Wiersze bloku pierwszego mogą opisywać funkcje 1, 0, α_0, α_1, gdzie α_0 jest dowolną funkcją zmiennych należących do zbioru A, a $\alpha_1(A) = \alpha_0'(A)$. Analogicznie wiersze bloku drugiego mogą opisywać funkcje 1, 0, β_0, β_1, gdzie $\beta_0(A) = \beta_1'(A)$.

Funkcję H można utworzyć czterema różnymi sposobami:

$H = \alpha_0 + b(\alpha_0 + \beta_0)$
$H = \alpha_0 + b(\alpha_0 + \beta_1)$
$H = \alpha_1 + b(\alpha_1 + \beta_0)$
$H = \alpha_1 + b(\alpha_1 + \beta_1)$,

gdzie znak + oznacza sumowanie modulo 2.

Rozważaną parę następstwa oznaczamy przez $[(A^W, 0, C^W), (A^W, 1, C^W)]$, gdzie wektory A^W i C^W oznaczają ustalone wartości zmiennych ze zbiorów A i C.

Dowód wymaga rozpatrzenia następujących możliwości:

1. $F(A, 0, C^W) = \alpha_i(A)$
   $F(A, 1, C^W) = \beta_j(A)$, gdzie i = 0, 1; j = 0, 1.

W tym przypadku przyjmujemy $H = \alpha_i + b(\alpha_i + \beta_j)$. Przy tak określonej funkcji H ryzyko nie istnieje, bo funkcja H nie zmienia swojej wartości w czasie przejścia. Mamy bowiem $\alpha_i(A^W) = \beta_j(A^W)$ i $H(0, A^W) = \alpha_i(A)$, $H(1, A^W) = \beta_j(A^W)$, czyli $H(0, A^W) = H(1, A^W)$.

2. W przypadku przejścia następującego z wiersza złożonego z samych zer lub jedynek do wiersza nietrywialnego lub odwrotnie, metodę konstrukcji funkcji H pokażemy, gdy:

   $F(A, 0, C^W) = 1$
   i $F(A, 1, C^W) = \beta_1(A)$
   oraz $F(A^W, 0, C^W) = F(A^W, 1, C^W) = \beta_1(A^W) = 1$.

Za funkcję H przyjmuje się $H = \alpha_0 + b(\alpha_0 + \beta_1)$, gdy $\alpha_0(A^W) = 1$, a jeżeli $\alpha_0(A^W) = 0$, to $H = \alpha_1 + b(\alpha_1 + \beta_1)$. Niech

$\alpha_0(A^W) = 0$ i $H = \alpha_1 + b(\alpha_1 + \beta_1)$, wtedy mamy $G(H, 1, C^W) = H$. Zjawisko szkodliwe może powstać tylko wtedy, gdy zmienna wejściowa b układu G zmieni się szybciej niż wartość funkcji H. Jednak wtedy $G(H, b, C^W)\big|_{b=1} = (\alpha_1 + b(\alpha_1 + \beta_1))\big|_{b=0} = \alpha_1(A^W) = 1$, co oznacza, że ryzyko nie istnieje.

Przykład

Rozważmy funkcję $F(a_1, a_2, b, c_1, c_2) = G(H(a_1, a_2, b), c_1, c_2)$ z poprzedniego przykładu. Dla pary następstwa [00000, 00100] mamy F(00000) = F (00100) = 1. Przyjmując $H = \alpha_0 + b(\alpha_0 + \beta_0)$, otrzymujemy dla rozpatrywanej pary następstwa realizację wolną od ryzyka strukturalnego. Natomiast przy powyższej realizacji ryzyko istnieje dla przejścia [01001, 01101].

Zatem dla określonego zbioru par następstwa nie zawsze istnieje realizacja funkcji F = G(H (A, B), B, C) wolna od ryzyka strukturalnego. W procesie syntezy należałoby każdorazowo badać czy dla określonego zbioru par następstwa i danego zbioru B istnieje realizacja wolna od ryzyka poprzez rozważenie wszystkich możliwych realizacji dekompozycji, co jest bardzo pracochłonne. Poniższe twierdzenie zmniejsza znacznie liczbę sprawdzeń, które należy wykonać.

Twierdzenie 3

a) Dla określonego zbioru par następstwa istnieje wolna od ryzyka strukturalnego realizacja funkcji F = G(H (A, B), B, C) wtedy, gdy istnieje taka realizacja dla funkcji $F = G(H(A^*, B^*), B^*, C^*)$ $B^* \subseteq B$, $C^* \subseteq C$, $A \subseteq A^*$.

b) Twierdzenie odwrotne do podanego w punkcie a) jest prawdziwe, gdy w każdym bloku macierzy M istnieje co najmniej jeden wiersz nietrywialny.

Dowód

Udowodnimy najpierw, że jeżeli istnieje realizacja

$$F = G(H(A^*, B^*), B^*, C^*) \qquad (1)$$

wolna od ryzyka strukturalnego, to istnieje również realizacja wolna od ryzyka dla

$$F = G(H(A, B), B, C) \qquad (2)$$

Wynika to z faktu, że dekompozycja (1) zawiera w sobie wszystkie dekompozycje podzbiorowe (2), tzn. dekompozycję podzbiorową można otrzymać wprost z danej dekompozycji przez dodanie do funkcji G lub H zmiennych będących różnicą między zbiorem B i B^*, a od których funkcje te faktycznie nie zależą.

Obecnie pokażemy, że jeżeli dla określonego zbioru par następstwa i dla danego zbioru B istnieje wolna od ryzyka strukturalnego realizacja funkcji F = G(H(A, B), B, C), to taka realizacja istnieje również dla podzbioru zbioru B.

Oznaczmy zbiór par następstwa przez N. Ryzyko może istnieć tylko dla tych par następstwa, dla których zmienna aktywna należy do zbioru B. Niech N_B oznacza zbiór tych par następstwa, $N_B \subset N$. Przyjmijmy, że zbiór B ma s elementów. Wówczas macierz M jest zbudowana z 2^s bloków, a funkcja H może być zrealizowana na 2^{2^s} różnych sposobów, ponieważ dla każdego z bloków za funkcję H można przyjąć jeden z dwóch wierszy różnych od wektora 1 lub 0. Jeżeli dla zbioru B istnieje realizacja funkcji F = G(H(A, B), B, C) wolna od ryzyka, wtedy funkcja H jest wyznaczona w ten sposób, że pary następstwa należące do zbioru N_B w każdym z bloków definiują jako H jeden z dwóch wierszy różnych od wektora 1 lub 0 tak, że dla każdej pary następstwa układ jest wolny od ryzyka. Jeżeli teraz weźmiemy podzbiór B^* zbioru B, gdzie card B^* = r, wówczas macierz M będzie zawierać 2^r bloków. Oczywiście $N_{B^*} \subset N_B$ oraz każdy z nowo powstałych 2^r bloków będzie zawierał jakiś blok z poprzednich. Wówczas w nowo powstałych blokach za funkcję H przyjmuje się taki wiersz, który zawiera wiersz przyjęty za funkcję H w bloku poprzednim. Dla każdej pary następstwa należącej do zbio-

ru N_B istotne jest tylko zdefiniowanie odpowiedniego wiersza jako H' lub H, co pozostaje bez zmiany. Zatem w ten sposób utworzona funkcja F = G(H (A*, B*), B*, C*) posiada realizację wolną od ryzyka strukturalnego.

Z powyższego twierdzenia wynika, że jeżeli dla dekompozycji elementarnej określonej przez zbiór B istnieje realizacja wolna od ryzyka, to taka realizacja istnieje również dla podzbioru B i odwrotnie, gdy dla dekompozycji elementarnej nie istnieje realizacja wolna od ryzyka strukturalnego, wtedy realizacja taka nie będzie również istnieć dla podzbioru zbioru B.

2.2. Dekompozycja iloczynowa

Aby istniała dekompozycja iloczynowa, poszczególne bloki macierzy M muszą spełniać następujący warunek. W bloku może istnieć jedynie jeden rodzaj wierszy nietrywialnych oraz wiersze zerowe (wiersz złożony z samych jedynek traktowany jest jako wiersz nietrywialny). Zatem w przeciwieństwie do dekompozycji prostej, gdzie różnym nietrywialnym wierszom bloków można dowolnie przyporządkować funkcję H lub H', przy dekompozycji iloczynowej funkcja H jest przyporządkowana istniejącemu wierszowi nietrywialnemu. W przypadku, gdy pewien blok zawiera wyłącznie wiersze zerowe, za funkcję H przyjmuje się jeden z wierszy zerowych. Założenie to powoduje, że dla dekompozycji iloczynowej funkcje G i H wyznaczone są w sposób jednoznaczny, co pozwoli na uzależnienie warunku istnienia ryzyka jedynie od funkcji F. Powyższe założenie jest uzasadnione, ponieważ wszystkie inne realizacje dekompozycji, dla których w bloku złożonym z samych zer H przyjęto w sposób dowolny, mogą wprowadzać dodatkowe ryzyko, a zatem są rozwiązaniami gorszymi. Wynika to z faktu, że jeżeli w bloku złożonym z samych zer przyjęto funkcję H równą zeru, wówczas zarówno G jak i H są zerowe, co powoduje, że ryzyko istnieć nie może. Jeżeli funkcja H nie jest tożsamościowo równa zeru, wtedy dla danego przejścia może mieć miejsce G(X) = 0, H(X) = 1, co może prowadzić do pojawienia się ryzyka.

Przy przyjętych uprzednio założeniach układ realizujący funkcję $F = G(A, B) \cdot H(B, C)$ jest wolny od dynamicznego ryzyka strukturalnego i może zawierać jedynie statyczne ryzyko strukturalne.

Z własności funktora iloczynu wynika, że układ może posiadać statyczne ryzyko strukturalne dla par następstwa $[X^1, X^2]$ takich, że $F(X^1) = F(X^2) = 0$. Poniżej rozważane są tylko te pary następstwa, dla których zmienna aktywna należy do zbioru B. Wtedy bowiem może się jednocześnie zmieniać wartość funkcji G i H, co w konsekwencji może spowodować nieprawidłowe działanie układu.

Do badania stałości funkcji w podsześcianie wykorzystywany będzie Δ-operator [1]. Δ-operator definiuje się następująco: $\Delta_{X_A} = \bigvee_{e \in A} \frac{\partial F}{\partial e}$. W [1] pokazano, że funkcja otrzymana po zastosowaniu Δ_{X_A}-operatora do funkcji F nie zależy od zmiennych ze zbioru A. Można zatem wprowadzić następujące oznaczenie $\Delta_{X_A} F(A, B, C) = \varphi_A(B, C)$.

Parę następstwa oznaczamy przez $[(A^w, B^0, C^w), (A^w, B^1, C^w)]$, gdzie B^0 i B^1 oznaczają początkowe i końcowe wartości zmiennych należących do zbioru B.

Przy powyższych założeniach zachodzi następujące twierdzenie.

Twierdzenie 4

Układ realizujący funkcję $F(A, B, C) = G(A, B) \cdot H(B, C)$ jest wolny od ryzyka strukturalnego dla danej pary następstwa, w której zmienna aktywna należy do zbioru B, wtedy i tylko wtedy, gdy:

1) $F(X^1) \vee F(X^2) = 1$ lub

2) $\varphi_C(A^w, B^0) \cdot \varphi_A(B^1, C^w) = 0$ i $\varphi_C(A^w, B^1) \cdot \varphi_A(B^0, C^w) = 0$.

Dowód

Układ posiada ryzyko dla $F(A^w, B^0, C^w) = F(A^w, B^1, C^w) = 0$ wtedy i tylko wtedy, gdy wartość funkcji G zmienia się z 0 na 1, a wartość funkcji H z 1 na 0 lub odwrotnie.

Zatem układ ma ryzyko dla pary następstwa $[(A^w, B^0, C^w), (A^w, B^1, C^w)]$ wtedy i tylko wtedy, gdy

1) $G(A^w, B^0) = 1$, $G(A^w, B^1) = 0$, $H(B^0, C^w) = 0$, $H(B^1, C^w) = 1$
lub
2) $G(A^w, B^0) = 0$, $G(A^w, B^1) = 1$, $H(B^0, C^w) = 1$, $H(B^1, C^w) = 0$

Rozważmy najpierw punkt pierwszy.

Funkcja $H(B^1, C^w)$ przyjmuje wartość 1 wtedy i tylko wtedy, gdy istnieją takie A^z wartości zmiennych ze zbioru A, że $F(A^z, B^1, C^w) = 1$. Jeżeli bowiem istnieją wartości A^z takie, że $F(A^z, B^1, C^w) = 1$, to $F(A^z, B^1, C^w) = G(A^z, B^1) \cdot H(B^1, C^w) = 1$, skąd wynika, że $H(B^1, C^w) = 1$. Odwrotnie, jeżeli nie istnieją takie wartości A^z, to w bloku zdefiniowanym przez wartości B^1 wiersz określony przez wartości C^w składa się z samych zer, co oznacza, że $H(B^1, C^w) = 0$. Ponieważ $F(A^w, B^1, C^w) = 0$, więc istnienie takich wartości A^z, że $F(A^z, B^1, C^w) = 1$ można zbadać za pomocą operatora Δ. Jeżeli $\varphi_A(B^1, C^w) = 1$, wówczas funkcja F nie jest stała w podsześcianie B^1, C^w i istnieją takie wartości A^z, że $F(A^z, B^1, C^w) = 1$. Gdy $H(B^1, C^w) = 1$, to $G(A^w, B^1) = 0$, ponieważ $F(A^w, B^1, C^w) = G(A^w, B^1) \cdot H(B^1, C^w) = 0$.

Analogicznie $G(A^w, B^0) = 1$ wtedy i tylko wtedy, gdy $\varphi_C(A^w, B^0) = 1$. Warunek ten zapewnia też, że $H(B^0, C^w) = 0$.

Zatem $G(A^w, B^0) = 1$, $G(A^w, B^1) = 0$, $H(B^0, C^w) = 0$, $H(B^1, C^w) = 1$ wtedy i tylko wtedy, gdy $\varphi_C(A^w, B^0) \cdot \varphi_A(B^1, C^w) = 1$.

W sposób zupełnie analogiczny pokazuje się, że $G(A^w, B^0) = 0$, $G(A^w, B^1) = 1$, $H(B^0, C^w) = 1$, $H(B^1, C^w) = 0$ wtedy i tylko wtedy, gdy $\varphi_C(A^w, B^1) \cdot \varphi_A(B^0, C^w) = 1$.

Zaprzeczenie powyższych warunków daje tezę twierdzenia, co kończy dowód.

Obecnie pokażemy, że na podstawie warunków podanych w twierdzeniu 4 można otrzymać funkcję, która wyznacza wszystkie wartości zmiennych X, dla których przy zmieniającej się zmiennej $b \in B$ nie istnieje ryzyko. W tym celu zauważymy, że

$$\varphi_A(B^1, C^w) = \varphi_A(B^0, C^w) + \left.\frac{\partial \varphi_A}{\partial b}\right|_{B^0 C^w},$$

$$\varphi_C(A^w, B^1) = \varphi_C(A^w, B^0) + \left.\frac{\partial \varphi_C}{\partial b}\right|_{A^w B^0} \quad \text{oraz}$$

$$F(X^2) = F(X^1) + \left.\frac{\partial F}{\partial b}\right| X^1.$$

Na tej podstawie otrzymujemy następującą funkcję:

$$\psi = F \vee \left(F + \frac{\partial F}{\partial b}\right) \vee \left(\varphi_C \left(\varphi_A + \frac{\partial \varphi_A}{\partial b}\right)\right)' \cdot \left(\varphi_A \left(\varphi_C + \frac{\partial \varphi_C}{\partial b}\right)\right)'$$

Po prostych przekształceniach mamy:

$$\psi = F \vee \frac{\partial F}{\partial b} \vee \varphi_A' \cdot \varphi_C' \vee \varphi_C' \left(\frac{\partial \varphi_C}{\partial b}\right)' \vee \varphi_A' \left(\frac{\partial \varphi_A}{\partial b}\right)' \vee \varphi_A \frac{\partial \varphi_A}{\partial b} \varphi_C \frac{\partial \varphi_C}{\partial b}$$

Funkcja ψ jest niezależna od zmiennej b i jest równa jedności dla wszystkich wartości zmiennych X, dla których przy zmiennej aktywnej $b \in B$ układ jest wolny od ryzyka strukturalnego.

Proces syntezy polega na znalezieniu takiego minimalnego zbioru B, dla którego wszystkie pary następstwa o zmiennej aktywnej należącej do B spełniają warunki twierdzenia 4. Poniższe twierdzenie pozwala na zmniejszenie liczby sprawdzań, które należy wykonać.

Twierdzenie 5

a) Realizacja funkcji $F = G(A, B) \cdot H(B, C)$ jest wolna od ryzyka wtedy, gdy wolna jest od ryzyka strukturalnego realizacja dekompozycji $F = G(A^*, B^*) \cdot H(B^*, C^*)$ dla $B^* \subset B$, $A \subseteq A^*$, $C \subseteq C^*$.

b) Twierdzenie odwrotne do twierdzenia podanego w punkcie a) jest prawdziwe, gdy jest spełniony następujący warunek: $\varphi_{AC}(B) \vee (\varphi_{AC}(B))' \cdot F(X) \equiv 1$

Dowód

Dowód punktu a) twierdzenia wynika wprost z ogólnych własności dekompozycji, ponieważ dana dekompozycja zawiera w sobie wszystkie dekompozycje elementarne.

Dowód punktu a) można przeprowadzić również na podstawie warunków twierdzenia 4. Dla $B^* \subset B$ otrzymuje się nowe zbiory $A \subseteq A^*$, $C \subseteq C^*$. Operatory $\Delta_{X_A} F$ i $\Delta_{X_C} F$ badają stałość funkcji w podsześcianach mniejszych niż operatory $\Delta_{X_{A^*}} F$, $\Delta_{X_{C^*}} F$. Funkcja stała w podsześcianie jest również stała w podsześcianie podsześcianu, zatem na przykład, jeżeli $\Delta_{X_{A^*}} F = 0$ to i $\Delta_{X_A} F = 0$, co kończy dowód.

Udowodnimy teraz punkt b) twierdzenia. Zakładamy, że realizacja funkcji $F = G(A, B) \cdot H(B, C)$ jest wolna od ryzyka i pokażemy, że realizacja funkcji $F = G(A^*, B^*) \cdot H(B^*, C^*)$ dla $B^* \subset B$, $A \subseteq A^*$, $C \subseteq C^*$ też jest wolna dla danej pary następstwa od ryzyka strukturalnego.

Warunek podany w punkcie b) oznacza, że żaden z bloków nie może zawierać wszystkich wartości jednakowych lub, jeżeli zawiera wszystkie wartości jednakowe, to są one jedynkami. Zatem założenie to powoduje, że w macierzy M nie może istnieć blok złożony z samych zer.

Gdy realizacja funkcji $F = G(A, B) \cdot H(B, C)$ jest wolna od ryzyka dla pary następstwa $[(A^w, B^0, C^w), (A^w, B^1, C^w)]$ takiej, że $F(A^w, B^0, C^w) = F(A^w, B^1, C^w) = 0$, to są spełnione następujące warunki:

1) $\varphi_C(A^w, B^0) \cdot \varphi_A(B^1, C^w) = 0$

2) $\varphi_C(A^w, B^1) \cdot \varphi_A(B^0, C^w) = 0$

Załóżmy, że na przykład $\varphi_C(A^w, B^0) = 0$ oraz $\varphi_A(B^0, C^w) = 0$. Warunek $\varphi_C(A^w, B^0) = 0$ oznacza, że w bloku określonym przez wartości B^0 wektor zdefiniowany przez wartości A^w składa się z jednakowych wartości, a więc z samych zer, ponieważ $F(A^w, B^0, C^w) = 0$.

Analogicznie warunek $\varphi_A(B^0, C^w) = 0$ oznacza, że w bloku określonym przez wartości B^0 wiersz zdefiniowany przez wartości C^w składa się z samych zer.

Ponieważ nie ma bloków złożonych z samych zer, więc jeżeli dekompozycja $F = G(A^*, B^*) \cdot H(B^*, C^*)$ istnieje, wówczas wiersz lub wektor nowo utworzonego bloku zawierający zerowy wiersz lub wektor bloku dla dekompozycji $F = G(A, B) \cdot H(B, C)$ musi też być wierszem lub wektorem złożonym z samych zer, w przeciwnym bowiem razie nie byłyby spełnione warunki istnienia dekompozycji iloczynowej. A więc jeżeli $\varphi_C = 0$ i $\varphi_A = 0$, to $\varphi_{C^*} = 0$ i $\varphi_{A^*} = 0$, co oznacza, że są spełnione warunki twierdzenia 4 i realizacja funkcji $F = G(A^*, B^*) \cdot H(B^*, C^*)$ jest wolna od ryzyka.

Powyższe twierdzenie pozwala na odrzucenie w procesie syntezy tych dekompozycji elementarnych, dla których istnieje ryzyko, ponieważ zmniejszanie zbioru B nie może tego ryzyka wyeliminować. Natomiast sprawdzanie warunków istnienia ryzyka podczas zmniejszania zbioru B można zaprzestać dopiero wtedy, gdy jest spełniony warunek z punktu b) twierdzenia.

Przykład

Niech $F(x_1, x_2, x_3) = G(A, B) \cdot H(B, C) = G(x_1, x_2) \cdot H(x_2, x_3)$, gdzie $F(x_1, x_2, x_3) = x_1' x_2' x_3 \vee x_1 x_2$, $G(x_1, x_2) = x_1' x_2' \vee x_1 x_2$, $H(x_2, x_3) = x_2 \vee x_3$. Sprawdzimy, czy dla pary następstwa [000, 010] realizacja funkcji F zawiera ryzyko strukturalne.

Mamy $F(0,0,0) = F(0,1,0) = 0$, $\Delta_{x_A} F = \dfrac{\partial F}{\partial x_1} = x_2 \vee x_3 \Big|_{(x_2=1, x_3=0)} = 1.$

$$\Delta_{x_C} F = \frac{\partial F}{\partial x_3} = x_1' x_2' \Big|_{(x_1=0,\ x_2=0)} = 1,$$

czyli $\varphi_C(A^w, B^0) \cdot \varphi_A(B^1, C^w) = 1$. Na mocy twierdzenia 4 dla pary następstwa [0 0 0, 0 1 0] realizacja funkcji F zawiera ryzyko strukturalne. Istotnie $G(0, 0) = 1$, $G(0,1) = 0$,

H (0,0) = 0, H (1, 0) = 1 i zmiana ta może spowodować nieprawidłowe działanie układu.

3. ALGORYTMY SYNTEZY UKŁADU KOMBINACYJNEGO

Na podstawie wyżej podanych twierdzeń można podać następujące algorytmy syntezy układów kombinacyjnych wolnych od ryzyka strukturalnego wykorzystujące dekompozycję prostą i iloczynową.

Algorytm dla dekompozycji prostej F = G(H (A, B), B, C) jest następujący:

K r o k 1. Wygenerować wszystkie dekompozycje elementarne postaci F = G(H (a_1, a_2, B), B, c).

K r o k 2. Dla każdej dekompozycji elementarnej sprawdzić czy dla danego zbioru par następstwa istnieje realizacja wolna od ryzyka strukturalnego. Rozważać tylko te pary następstwa $[X^1, X^2]$, dla których F (X^1) = F (X^2) oraz zmienna aktywna należy do zbioru B.

K r o k 3. Dla dekompozycji elementarnych spełniających warunek podany w kroku 2 oraz warunek b) twierdzenia 3 zmniejszyć maksymalnie zbiór B, wykorzystując warunki istnienia dekompozycji podane w [2], a dla dekompozycji nie spełniających warunku z punktu b) twierdzenia 3 przy zmniejszeniu zbioru B sprawdzać każdorazowo, czy istnieje realizacja wolna od ryzyka, tak długo, aż będzie spełniony warunek z punktu b) twierdzenia 3.

K r o k 4. Wybrać spośród dekompozycji otrzymanych w punkcie 3 dekompozycję mającą najmniej liczny zbiór B.

K r o k 5. Zrealizować funkcje G i H tak, aby realizacje te nie zawierały ryzyka strukturalnego.

Na mocy podanych wyżej twierdzeń realizacja funkcji F będzie wolna od ryzyka strukturalnego.

Algorytm dla dekompozycji iloczynowej przedstawia się następująco:

K r o k 1. Wygenerować wszystkie dekompozycje elementarne postaci $F = G(a, b) \cdot H(B, c)$.

K r o k 2. Dla każdej dekompozycji elementarnej sprawdzić, czy dla wszystkich par następstwa zawierających zmienną aktywną należącą do zbioru B spełniony jest warunek twierdzenia 4.

K r o k 3. Dla dekompozycji elementarnych spełniających warunek podany w kroku 2 oraz warunek podany w punkcie b) twierdzenia 5 zmniejszyć maksymalnie zbiór B, wykorzystując warunki istnienia dekompozycji [2], a dla dekompozycji nie spełniających warunku z punktu b) twierdzenia 5 przy zmniejszaniu zbioru B sprawdzać każdorazowo warunki twierdzenia 4 tak długo, aż będzie spełniony warunek z punktu b) twierdzenia 5.

Dalsze kroki postępowania przebiegają analogicznie jak w algorytmie dla dekompozycji prostej.

Na zakończenie zauważmy, że wszystkie wnioski dotyczące dekompozycji iloczynowej przenoszą się na dekompozycję $F(A, B, C) = G(A, B) \vee H(B, C)$, ponieważ dekompozycję powyższą można otrzymać z dekompozycji funkcji $F'(A, B, C) = G'(A, B) \cdot H'(B, C)$.

Opisane powyżej algorytmy pozwalają na podanie realizacji danej funkcji boolowskiej wolnej od ryzyk strukturalnych dla zmian przyległych. Ponieważ funkcja jest realizowana w postaci zdekomponowanej o możliwie najmniejszym zbiorze B, realizacja ta jest zbliżona do minimalnej.

4. PODSUMOWANIE

W pracy rozważono problem zastosowania dekompozycji do syntezy układów kombinacyjnych wolnych od ryzyka strukturalnego. Zdekomponowanie funkcji boolowskiej prowadzi do realizacji układu kombinacyjnego quasi minimalnego, który nie zawsze jest wolny od ryzyka. Udowodniono, że dla zmian przyległych istnienie realizacji dekompozycji wolnej od ryzyka zależne jest od istnienia takiej realizacji dla dekompozycji elementarnych. Podano warunki rozstrzygające fakt czy dla danej pary następstwa dekompo-

zycja iloczynowa zawiera ryzyko. Na tej podstawie sformułowano algorytmy, za pomocą których dla zmian przyległych można otrzymać wolną od ryzyka strukturalnego realizację funkcji boolowskiej.

Literatura

[1] AKERS S.B.: On a Theory of Boolean Functions, J. SIAM, 1959, t. 7

[2] BAŃKOWSKI J.: Difference Theorems on Decomposition of Boolean Function, w przygotowaniu do druku

[3] BREDESON J., HULINA P.: Elimination of Static and Dynamic Hazards for Multiple Input Changes in Combinational Switching Circuits, Information and Control, 1972, nr 20, s. 114-124

[4] CURTIS H.A.: A New Approach to the Design of Switching Circuits, Van Nostrand, Princeton, 1962

[5] EICHELBERGER E.B.: Hazard Detection in Combinational and Sequential Switching Circuits, IBM Journal of Research and Development, 1965, t. 9, nr 2

[6] HARRISON M.A.: Wstęp do teorii sieci przełączających i teorii automatów, PWN, 1973

[7] SAPIECHA K.: Analiza sieci kombinacyjnych z ograniczonymi opóźnieniami szkodliwymi, Archiwum automatyki i telemechaniki, 1972, t. XVII, nr 3

[8] SHEN Y.S., McKELLER A.C., WEINER P.: A Fast Algorithm for the Disjunctive Decomposition of Switching Function, IEEE Transactions on Computers, 1971, t. C-20, nr 3

[9] THAYSE A., DAVIO M.: Boolean Differential Calculus and its Application to Switching Theory, IEEE Transactions on Computers, 1973, t. C-22, nr 4

ПРИМЕНЕНИЕ ДЕКОМПОЗИЦИИ К СИНТЕЗУ КОМБИНАЦИОННЫХ СХЕМ СВОБОДНЫХ ОТ РИСКА ПО ОТНОШЕНИЮ К СОСЕДНИМ ИЗМЕНЕНИЯМ

Резюме

В работе рассмотрена проблема применения декомпозиции к синтезу комбинационных логических схем свободных от риска. Представлены теоремы делающие возможным уменьшение пространства разыскиваных функции и сформулированы условия решающие когда произведенная декомпозиция содержает риск. Замещены алгоритмы синтеза комбинационных логических схем.

DECOMPOSITION APPLICATION TO THE SYNTHESIS OF HAZARD-FREE CIRCUITS FOR ADJACENT CHANGES

Summary

The paper considers the problem of decomposition application to the synthesis of hazard-free switching circuits. Given the theorems permitting a reduction of searched function space. Conditions deciding when the conjunction decomposition contains hazard is formulated. Algorithms of switching circuits are presented.

Prace IMM
Zeszyt 3

517.11

O DEKOMPOZYCJACH FUNKCJI SYMETRYCZNYCH

Wojciech ZAWADZKI
Politechnika Warszawska
Pracę złożono 20.09.1974

W pracy badane są własności dekompozycyjne symetrycznych funkcji boolowskich. Wykazano, że należą one do funkcji trudno dekomponowalnych. Podano przykład wskazujący na możliwość zastosowania rozkładów niewłaściwych w syntezie sieci symetrycznych.

1. WSTĘP

Niech n będzie dowolną liczbą naturalną, a V^n oznacza n-krotny iloczyn kartezjański V; $V = \{0,1\}$. V^n jest zbiorem wszystkich wektorów $X = (x_1, x_2, \ldots, x_n)$; $x_i \in V$, $i = 1, 2, \ldots n$. Funkcja n zmiennych boolowskich F jest funkcją od V^n do V; $F: V^n \longrightarrow V$.

Oznaczmy przez d(X) liczbę współrzędnych wektora X przyjmujących wartość 1. W zbiorze V^n określimy relację równoważności $\overset{\equiv}{d}$ w następujący sposób

$$X_p \overset{\equiv}{d} X_q \Longleftrightarrow d(X_p) = d(X_q) \quad (X_p, X_q \in V^n)$$

Definicja

Funkcją symetryczną nazywamy funkcję F(X), która spełnia następujący warunek:

$$\bigwedge_{X_p, X_q \in V^n} X_p \overset{\equiv}{d} X_q \Longrightarrow F(X_p) = F(X_q)$$

Algebra S_n funkcji symetrycznych n zmiennych jest podalgebrą algebry wszystkich funkcji n zmiennych, a jej generatorami są elementarne funkcje symetryczne, zdefiniowane w następujący sposób:

$$S_0(X) = \bar{x}_1\bar{x}_2 \dots \bar{x}_n,$$
$$S_1(X) = \bar{x}_1\bar{x}_2 \dots \bar{x}_n \vee \bar{x}_1 x_2 \dots \bar{x}_n \vee \dots \vee \bar{x}_1\bar{x}_2 \dots x_n,$$
$$\vdots$$
$$S_n(X) = x_1 x_2 \dots x_n \quad [1].$$

Twierdzenie 1 [1]

Funkcja symetryczna jest jednoznacznie wyznaczona przez zbiór liczb naturalnych $D = \{d_0 \dots d_k\}$, gdzie $0 \leqslant d_j \leqslant n$, oraz $k = 0, 1, \dots, n$, taki, że $F = 1$ wtedy i tylko wtedy, gdy d_j zmiennych przyjmuje wartość 1 ($j = 0, 1, \dots, k$).

Zbiór D będziemy nazywać zbiorem charakterystycznym funkcji symetrycznej. Zbiór pusty określa funkcję $F \equiv 0$, natomiast zbiór $D = \{1, 2, \dots, n\}$ funkcję $F \equiv 1$.

W niniejszej pracy badane są własności dekompozycyjne symetrycznych funkcji boolowskich.

2. DEKOMPOZYCJA WEWNĘTRZNA TYPU H (f (A, B), B, C)

Niech A, B, C będą podzbiorami właściwymi zbioru $\{x_1, x_2, \dots, x_n\}$. Dowolna funkcja boolowska F (A, B, C) może być przedstawiona w postaci [4]

$$F(A, B, C) = \bigcup_{i=1}^{2^{\bar{B}}-1} p_i(B)\, F_i(A, C) \quad *) \qquad (1)$$

gdzie:

$\bar{B}$ oznacza moc zbioru B,

*) Jeśli F(A, B, C) jest symetryczna, to funkcje $F_i(A, C)$ są również symetryczne [5].

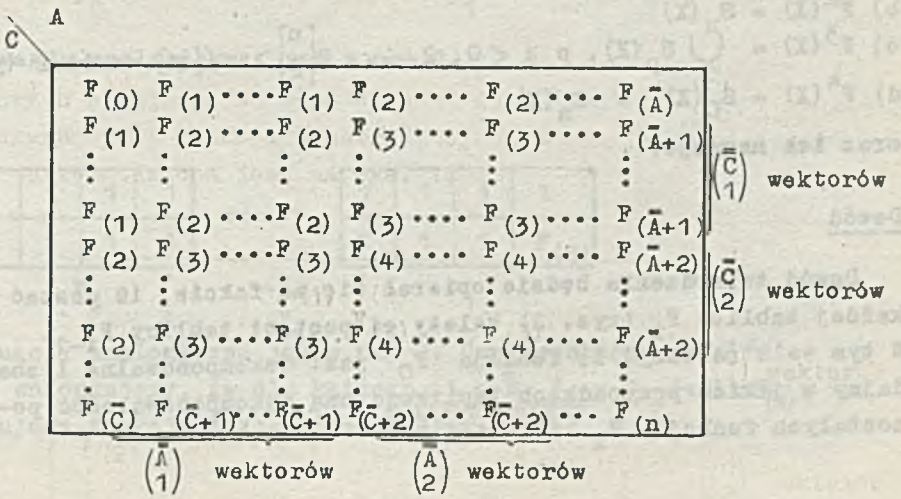
$$p_0(B) = \bar{x}_{i_1}\,\bar{x}_{i_2} \dots \bar{x}_{i_k}$$
$$p_1(B) = \bar{x}_{i_1}\,\bar{x}_{i_2} \dots x_{i_k}$$
$$\vdots$$
$$p_{H-1}(B) = x_{i_1}\,x_{i_2} \dots \bar{x}_{i_k}$$
$$p_H(B) = x_{i_1}\,x_{i_2} \dots x_{i_k},\ \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\} = B,\ H = 2^{\bar{B}} - 1.$$

F_i powstaje z funkcji F przez podstawienie takiej kombinacji wartości zmiennych, należących do zbioru B, że $p_i = 1$.

Curtis [4] wykazał prawdziwość następującego twierdzenia: funkcja F(A, B, C) posiada dekompozycję H (f(A, B), B, C) wtedy i tylko wtedy, gdy każda z funkcji $F_i(A, C)$ występujących w rozwinięciu (1) posiada dekompozycję H_i $(f_i(A), C)$.

Twierdzenia dotyczące dekompozycji rozłącznej (B = Ø) oparte są na własnościach tablic dekompozycji [2, 4]. Są to macierze $2^{\bar{C}} \times 2^{\bar{A}}$, których wiersze opisane są przez zmienne należące do C, kolumny - przez zmienne należące do A, a ich elementami są wartości funkcji. Liczba różnych kolumn (wierszy) tablicy dekompozycji zwana jest współczynnikiem kolumnowym (wierszowym) tablicy.

Niech $F_{(j)}$ oznacza wartość funkcji symetrycznej F, gdy j zmiennych przyjmuje wartość 1, n-j zmiennych wartość 0. Tablica dekompozycji funkcji symetrycznej odpowiadająca podziałowi A | C przedstawiona jest na rys. 1.



Rys. 1

Jak wykazał Ashenhurst [2], warunkiem koniecznym i dostatecznym istnienia dekompozycji $F(A, C) = H(f(A), C)$ jest, aby współczynnik kolumnowy ν tablicy $A \mid C$ spełniał warunek $\nu \leqslant 2$.

Rozpatrzmy najprostszą dekompozycję nietrywialną $F(x_1, x_2, x_3, B) = H(f(x_1, x_2, B), B, x_3)$. W pracy [3] wykazano, że poszukiwanie dekompozycji rozłącznych należy rozpocząć od badania tego typu rozkładów. Tablicę odpowiadającą podziałowi $x_1, x_2 \mid x_3$ dla funkcji $F_i(x_1, x_2, x_3)(1)$ przedstawiono na rys. 2.

x_3 \ x_1, x_2	00	01	10	11
0	$F_{(i)}$	$F_{(i+1)}$	$F_{(i+1)}$	$F_{(i+2)}$
1	$F_{(i+1)}$	$F_{(i+2)}$	$F_{(i+2)}$	$F_{(i+3)}$

F_i

Rys. 2

Twierdzenie 2

Jedynymi funkcjami symetrycznymi posiadającymi dekompozycję $H(f(x_1, x_2, B), B, x_3)$ dla dowolnych x_1, x_2, x_3 są funkcje:

a) $F^1(X) = S_0(X)$

b) $F^2(X) = S_n(X)$

c) $F^3(X) = \bigcup_p S_p(X)$, $p \in <0, 2, \ldots, 2\left[\frac{n}{2}\right]>$ ($[y]$ -entier (y))

d) $F^4(X) = S_0(X) \vee S_n(X)$

oraz ich negacje.

Dowód

Dowód twierdzenia będzie opierał się na fakcie, iż postać każdej tablicy F_i (rys. 2) zależy od postaci tablicy F_{i-1}. W tym celu załóżmy, że funkcja F_0 jest dekomponowalna i zbadajmy w jakich przypadkach implikuje ona dekomponowalność pozostałych funkcji F_i $(0 < i \leqslant \bar{B})$.

1)

0	0	0	0
0	0	0	1

F_0

$\Longrightarrow$

0	0	0	1
0	1	1	$F_{(4)}$

F_1

Taka postać funkcji F_0 musi być wykluczona, gdyż implikuje niedekomponowalną funkcję F_1

2)

0	0	0	0
0	0	0	0

F_0

$\Longrightarrow$

0	0	0	0
0	0	0	$F_{(4)}$

F_1

Jeśli $F_{(4)} = 1$, to zgodnie z pkt 1, F_2 jest niedekomponowalna. Zatem $F_{(4)} = 0$. Postępowanie analogiczne z każdą funkcją F_i prowadzi do wniosku, że $F_i = 0$ dla każdego $i < \overline{B}$. Jeśli $F_{(n)} = 1$, to otrzymujemy funkcję dekomponowalną $F = F^2(X)$.

3)

1	1	1	1
1	1	1	0

F_0

$\Longrightarrow$

1	1	1	0
1	0	0	$F_{(4)}$

F_1

Sytuacja analogiczna, jak w pkt 1.

4)

1	1	1	1
1	1	1	1

F_0

$\Longrightarrow$

1	1	1	1
1	1	1	$F_{(4)}$

F_1

Sytuacja analogiczna jak w pkt 2. Uwzględniając rezultat z pkt 3 wnioskujemy, że dla każdego $i < \overline{B}$ $F_i = 1$. Jeśli $F_{(n)} = 0$, otrzymujemy funkcję dekomponowalną $F = F^2(X)$.

5)

0	1	1	1
1	1	1	1

F_0 ⟹

1	1	1	1
1	1	1	$F_{(4)}$

F_1

Sytuacja analogiczna jak w pkt 4. Jeśli $F_{(n)} = 0$, otrzymujemy funkcję dekomponowalną $F^1(X)$, jeśli $F_{(n)} = 1$ - funkcję $F^4(X)$.

6)

1	0	0	0
0	0	0	0

F_0 ⟹

0	0	0	0
0	0	0	$F_{(4)}$

F_1

Sytuacja analogiczna jak w pkt 2. Jeśli $F_{(n)} = 0$, otrzymujemy funkcję dekomponowalną $F^1(X)$, jeśli $F_{(n)} = 1$ - funkcję $F^4(X)$.

7)

0	1	1	0
1	0	0	1

F_0 ⟹

1	0	0	1
0	1	1	0

F_1 ⟹

0	1	1	0
1	0	0	1

F_2

Sytuacja powyższa jest równoważna warunkowi $F_i = F_{i+2}$ dla każdego $i \in [0, \overline{B-2}]$. Otrzymujemy funkcję dekomponowalną $F^3(X)$.

8)

1	0	0	1
0	1	1	0

F_0 ⟹

0	1	1	0
1	0	0	1

F_1 ⟹

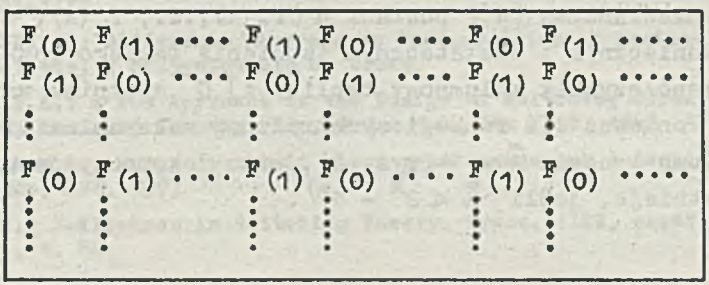
1	0	0	1
0	1	1	0

F_2

Sytuacja analogiczna jak w pkt 7: otrzymujemy funkcję $F^3(X)$.
W ten sposób dowód twierdzenia został zakończony*).

*) Łatwo zauważyć, że pozostałe (8) funkcje symetryczne F_0 nie są dekomponowalne

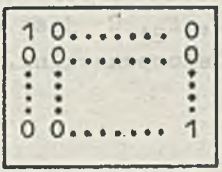
Funkcje F^1, F^2, F^3 oraz ich negacje posiadają także dekompozycje rozłączne H(f(A), C), przy czym zbiory A i C mogą być wybrane dowolnie. Dla funkcji F^3 warunek $F_i = F_{i+2}$ jest równoważny warunkowi $F_{(i)} = F_{(i+2)}$ (pkt 7, 8 dowodu), zatem współczynnik kolumnowy tablicy A | C (rys. 3) równy jest 2. Analiza macierzy funkcji F^4 (rys. 4) dowodzi, że nie posiada ona dekompozycji rozłącznej, natomiast może być przedstawiona w postaci H (f (A, x), x , C), gdzie zarówno x, jak i A i C mogą być wybrane dowolnie.



Rys. 3

x = 0 $x \in B$ x = 1

Tablice dekompozycji H(f(A, x), x, C) funkcji $F^4(X)$



$\nu = 3$

Tablica dekompozycji rozłącznej funkcji $F^4(X)$

Rys. 4

3. DEKOMPOZYCJA TYPU H(f (A, B), g(B, C))

Wśród tych dekompozycji główną rolę odgrywają: liniowa - f + g, oraz nieliniowa - f . g. Funkcja $F^3(X)$ może być przedstawiona w postaci $F^3(X) = 1 + x_1 + x_2 + ... + x_n$, natomiast $F^4(X)$ w postaci $F^4(X) = (x_1 + x_2) \cdot (x_2 + x_3)..... (x_{n-1} + x_n)$.

Z dotychczasowych rozważań wynika, że funkcje symetryczne należą w ogólnym przypadku do funkcji niedekomponowalnych. Natomiast ciekawe rezultaty można osiągnąć wykorzystując tzw. dekompozycje niewłaściwe([4]) postaci $H(f_1(A),..., f_k(A), C)$. Warunkiem koniecznym i dostatecznym istnienia takiego rozkładu jest, aby współczynnik kolumnowy tablicy $A|C$ spełniał warunek $\nu \leqslant 2^k$. Ponieważ dla funkcji symetrycznej maksymalna liczba różnych kolumn wynosi $\bar{\bar{A}} + 1$ (rys. 1), więc dekompozycja niewłaściwa istnieje, jeśli $\bar{\bar{A}} \leqslant 2^k - 1$*).

Przykład

Niech $F(x_1,..., x_5)$ będzie określona przez zbiór $D = \{0,1,4\}$. Poszukujemy rozkładu $H(f_1(x_1, x_2, x_3), f_2(x_1, x_2, x_3), x_4, x_5)$. Stosując metodę podaną w [4] otrzymujemy:

$$H(f_1, f_2, x_4, x_5) = f_1 f_2 x_5 \vee f_1 \bar{f}_2 x_4 x_5 \vee \bar{f}_1 f_2 \bar{x}_4 \bar{x}_5 \vee \bar{f}_1 \bar{f}_2 x_4 \vee \bar{f}_1 \bar{f}_2 \bar{x}_5,$$

$$f_1(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 x_3,$$

$$f_2(x_1, x_2, x_3) = x_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_2 x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3.$$

Powyższy przykład dowodzi, że dekompozycje niewłaściwe mogą mieć duże znaczenie przy syntezie funkcji symetrycznych, zwłaszcza, że funkcja H od zmiennych f_1, f_2, C jest w ogólnym przypadku niesymetryczna i może być dekomponowalna.

3. ZAKOŃCZENIE

W pracy omówiono własności dekompozycyjne symetrycznych funkcji boolowskich. Udowodniono, że wśród $2^{n+1} - 2$ nietrywialnych

*) Nie jest to warunek konieczny

funkcji n zmiennych jedynie 8 posiada dekompozycje właściwe. Na przykładzie pokazano możliwość zastosowania rozkładów niewłaściwych do syntezy sieci symetrycznych.

Literatura

[1] ARNOLD R.F., HARRISON M.A.: Algebraic Properties of Symmetric and Partially Symmetric Boolean Functions, IEEE Trans. 1963, t. EC-12, nr 3

[2] ASHENHURST R.: The Decomposition of Switching Functions, Harvard Computation Laboratory, Bell Laboratories Report, 1952, nr BL-1/II

[3] BAŃKOWSKI J.: Difference Theorems on Decomposition of Boolean Functions, w przygotowaniu do druku

[4] CURTIS H.A.: A New Approach to the Design of Switching Circuits, D. van Nostrand Company, Inc., Princeton, New York, 1962

[5] HARRISON M.A.: Wstęp do teorii sieci przełączających i teorii automatów, PWN, 1973

[6] SEMON W.: E-Algebras in Switching Theory, Trans. AIEE, część I, 1961 , t. 80

О РАЗЛОЖЕНИИ СИММЕТРИЧЕСКИХ ФУНКЦИИ

Резюме

В работе исследованы свойства разложения симметрических булевых функций. Показано, что они принадлежат к функциям, которые трудно разложить. Указан пример возможного применения несоответственных разложений в синтезе симметрических сетей.

ON SYMMETRICAL FUNCTION DECOMPOSITIONS

Summary

The paper considers decompositional properties of symmetrical Boolean functions. It has been demonstrated that they belong to hardly decomposable functions. An example is given indicating the possibility of improper decomposition application to the synthesis of symmetrical networks.

Prace IMM
Zeszyt 3

621.317.799:681.327.64:534.781

UKŁAD DO OBSERWACJI PRZEBIEGÓW CZASOWYCH O CZĘSTOTLIWOŚCI AKUSTYCZNEJ

Ryszard PATRYN
Instytut Maszyn Matematycznych
Pracę złożono 11.06.1975

W pracy omówiono schemat ideowy oraz zasadę działania laboratoryjnego urządzenia, umożliwiającego obserwację oscyloskopową sygnałów nieokresowych ze szczególnym zastosowaniem w odniesieniu do sygnałów mowy.

1. WSTĘP

Podczas badań sygnału mowy, związanych z pracami nad fonicznym układem wyjściowym maszyny cyfrowej prowadzonych w IMM, pojawił się problem obserwacji przebiegów czasowych.

Obserwację przebiegów czasowych można prowadzić wieloma metodami, np.:

- za pomocą oscyloskopu o długiej poświacie lub oscyloskopu pamiętającego obraz
- zapisu za pomocą pisaka na taśmie papierowej
- zapisu fotooptycznego na taśmie filmowej
- odczytywania taśmy magnetofonowej w zamkniętej pętli
- odczytywania taśmy magnetofonowej za pomocą wirującej głowicy
- cyklicznego odczytywania sygnałów z pamięci cyfrowej

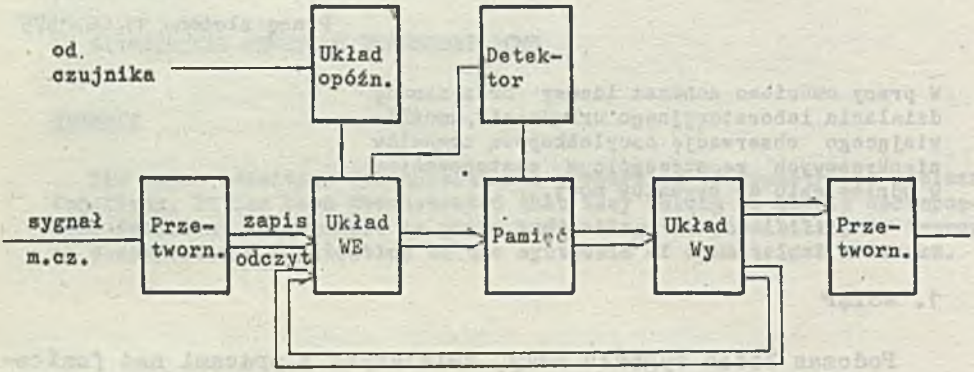
W opisywanym układzie wykorzystano ostatnią z wymienionych metod, uznając ją za najwygodniejszą. Układ został zaprojektowany z myślą zastosowania do badania sygnału mowy, ale może być również przydatny do badania innego rodzaju sygnałów o częstotliwości akustycznej.

2. OGÓLNY OPIS UKŁADU

W skład układu wchodzą:

- przetwornik analogowo-cyfrowy
- pamięć cyfrowa
- zespół opóźniający
- przetwornik cyfrowo-analogowy

Schemat blokowy układu przedstawia rys. 1.



Rys. 1. Schemat blokowy układu

Układ jest przystosowany głównie do obserwacji przebiegów zapisanych uprzednio na taśmie magnetofonowej. Ogólna zasada działania jest następująca. Na taśmie magnetofonowej, na której zapisano badany przebieg, umieszcza się znacznik w postaci przewodzącej folii w miejscu położonym blisko początku badanego fragmentu. Do wejścia układu (wejście przetwornika analogowo-cyfrowego - rys. 1) doprowadza się sygnał z wyjścia magnetofonu. Układy wejściowe i wyjściowe zostają ustawione na zapis, tzn. zostaje przerwane zwrotne podawanie sygnału z układów wyjściowych do wejściowych. W momencie pojawienia się znacznika na taśmie, specjalny czujnik przesyła impuls do układu opóźniającego, który po określonym czasie powoduje rozpoczęcie wpisywania do pamięci kolejnych próbek sygnału. Po załadowaniu całej pamięci, wykrywanym przez specjalny detektor, automatycznie blokowany jest dopływ sygnałów wejściowych. Następnie przełą-

cza się układ na odczyt; w tym stanie tworzy się zamkniętą pętlę, w której sygnały krążą cyklicznie i mogą być pobierane z układów wyjściowych, a po przetworzeniu cyfrowo-analogowym umożliwiają obserwację na ekranie oscyloskopu.

Do pierwszego słowa w pamięci wpisuje się same jedynki lub same zera, w ten sposób po przetworzeniu cyfrowo-analogowym powstanie impuls, który oprócz ułatwienia synchronizacji obrazu na ekranie oscyloskopu umożliwia dokładne umiejscowienie interesującego fragmentu i określenie właściwego czasu opóźnienia. Po powtórnym zapisie interesujący fragment przebiegu powinien znaleźć się w małej odległości (na ekranie oscyloskopu) od impulsu synchronizującego, co umożliwia takie rozciągnięcie obrazu, które pozwala na dokładniejszą analizę czasową przebiegu.

Oprócz opisanego wyżej sposobu wykorzystania układu, przewidziano też możliwość pracy bez zwrotnego podawania sygnału, który polega na tym, że sygnały nie krążą w pamięci, natomiast następuje ciągły przepływ impulsów przez układ, a na wyjściu pojawiają się sygnały po przetworzeniu cyfrowo-analogowym. Umożliwia to na przykład badanie samej części przetwornikowej.

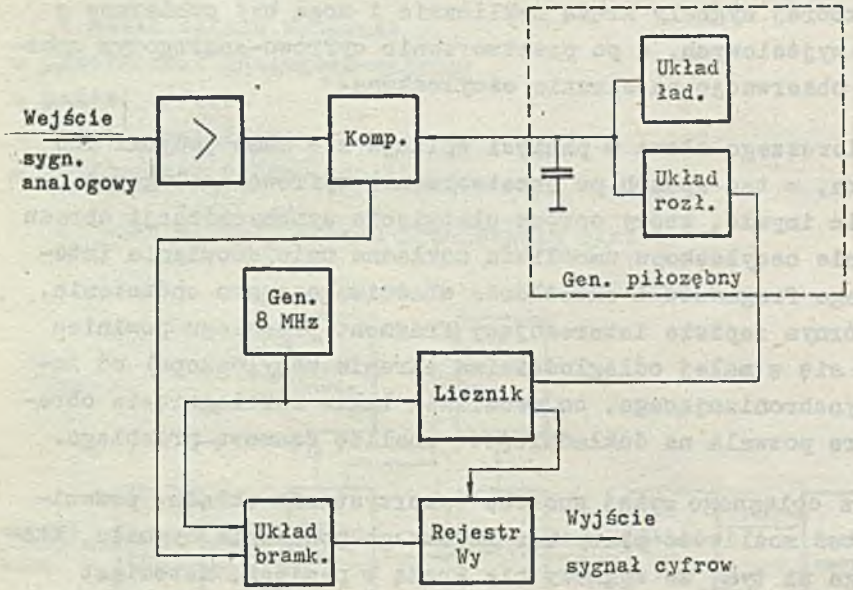
3. SZCZEGÓŁOWY OPIS UKŁADU

3.1. Przetwornik analogowo-cyfrowy

Przetwornik analogowo-cyfrowy (rys. 2) składa się ze wzmacniacza, komparatora, generatora piłozębnego, rejestru wyjściowego i licznika sterowanego z generatora fali prostokątnej. W działaniu przetwornika wykorzystano liniową zmianę napięcia w czasie w przebiegu generatora piłozębnego. Wyznaczony za pomocą licznika czas, po którym następuje zrównanie poziomu sygnału wejściowego i generatora piłozębnego w danym cyklu próbkowania wyznacza wartość cyfrową poziomu sygnału.

Zasadę próbkowania przedstawiono na rys. 3.

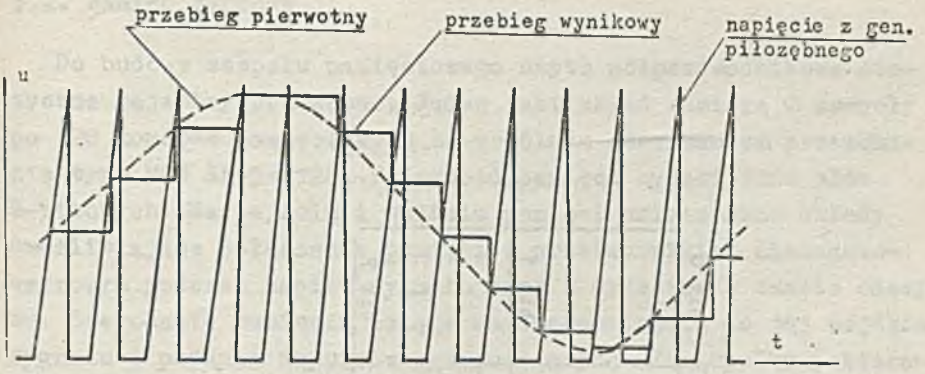
Generator fali prostokątnej wysterowuje licznik binarny o dziewięciu ogniwach, przy czym sygnał z ostatniego ogniwa



Rys. 2. Schemat blokowy przetwornika analogowo-cyfrowego

wyzwala każdorazowo przebieg piłozębny. Sygnał generatora piłozębnego jest porównywany z sygnałem akustycznym w komparatorze. W momencie zrównania się chwilowych wartości obu sygnałów z komparatora wysyłany jest impuls, który powoduje wpisanie do rejestru wartości licznika. W rezultacie takiego działania uzyskuje się próbkowanie przebiegu co jeden cykl licznika o dziewięciu ogniwach. Przy częstotliwości generatora ca 8 MHz częstotliwość próbkowania wynosi $8.10^6/2^9$ = 15,5 kHz. Przy przyjętym układzie licznika uzyskuje się 2^8 = 256 poziomów kwantyzacji, co odpowiada dynamice ca 41 dB.

Rejestr wartości cyfrowych utworzono z licznika o ośmiu ogniwach, który jest sterowany równolegle z licznikiem wyzwalającym generator piłozębny. Sygnał z komparatora powoduje przerwanie liczenia tego licznika i w ten sposób utrzymuje się wartość wyjściową, która jest stała aż do momentu zerowania, następującego po pojawieniu się impulsu z ostatniego ogniwa licznika sterującego generator piłozębny.



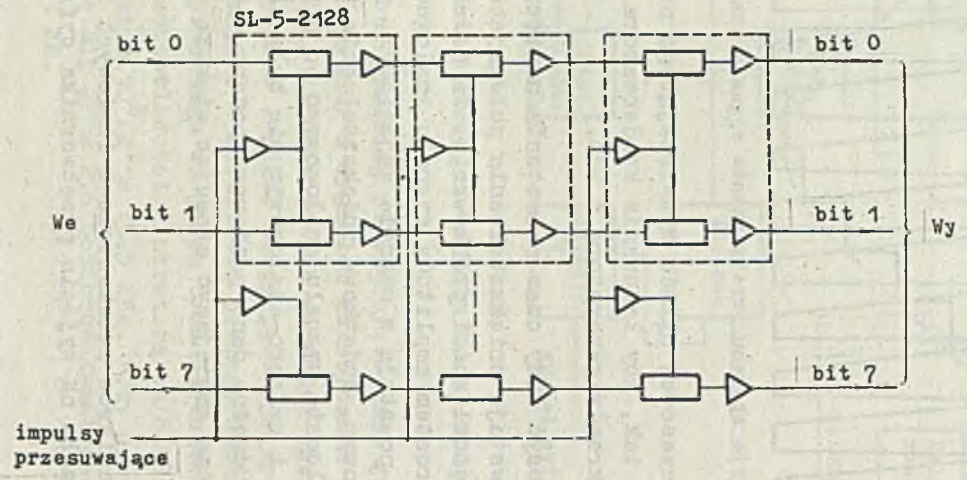
Rys. 3. Ilustracja sposobu przetwarzania sygnału analogowego

Wartość stałej czasowej układu wytwarzającego przebieg piłozębny jest dobrana tak, aby ładowanie kondensatora trwało przez mniej więcej 256 okresów generatora 8 MHz.

Ze względu na dosyć duży czas narastania napięcia w generatorze piłozębnym, powstają zniekształcenia polegające na pewnej modulacji fazy częstotliwości próbkowania. Ta modulacja zwiększa się wraz ze wzrostem amplitudy sygnału analogowego. W wyniku tego zjawiska powstają w sygnale wyjściowym, po przetworzeniu cyfrowo-analogowym, dodatkowe zniekształcenia nieliniowe. Wynikowe zniekształcenia sygnału analogowego po przekształceniu analogowo-cyfrowym i cyfrowo-analogowym nie były zbyt duże, jak dla sygnału mowy. Odnośne dane pomiarowe podano w tabeli 1. Zniekształcenia można zmniejszyć stosując większą częstotliwość próbkowania.

Tabela 1. Wyniki pomiarów zniekształceń liniowych i harmonicznych sygnału na wyjściu przetwornika cyfrowo-analogowego

f kHz	poziom dB	zniekształcenia harmoniczne %
0,5	+ 0,3	1,8
1	0	2,3
2	- 1	3,9
4	- 4,5	8,8
7	- 9	nie mierzono



Rys. 4. Schemat układu pamięciowego

3.2. Pamięć cyfrowa

Do budowy zespołu pamięciowego użyto półprzewodnikowe statyczne rejestry przesuwne. Jeden taki układ zawiera 2 zespoły po 128 komórek pamięciowych ze wspólnym sterowaniem przesuwania typu MOS SL-5-2128. Pojemność pamięci wynosi 1280 słów 8-bitowych. Na wejściu i wyjściu pamięci umieszczono układy umożliwiające połączenie pamięci z przetwornikiem analogowo-cyfrowym podczas zapisu sygnału oraz z wyjściem w czasie odczytu. Sterowanie pamięcią polega na doprowadzeniu do jej wejścia sygnału i podania impulsów przesuwających. Te impulsy pobierane są z licznika generatora piłozębnego. Uproszczony schemat pamięci pokazano na rys. 4.

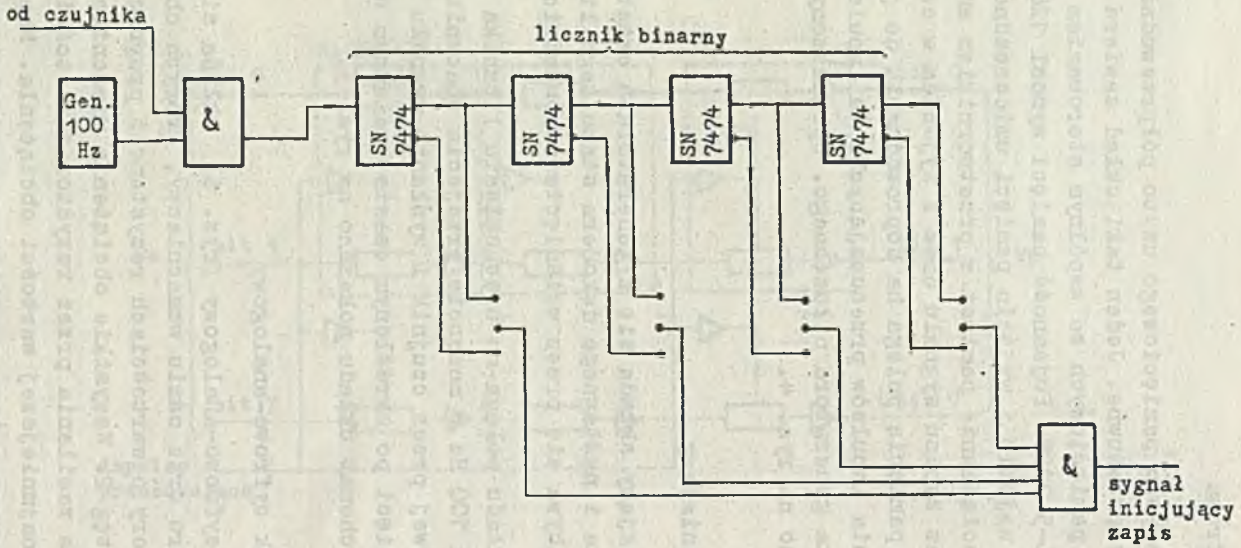
3.3. Układ opóźniający

Układ opóźniający składa się z generatora o częstotliwości 100 Hz, licznika i nastawnego dekodera stanu tego licznika (nastawianie odbywa się przez wciśnięcie odpowiednich klawiszy).

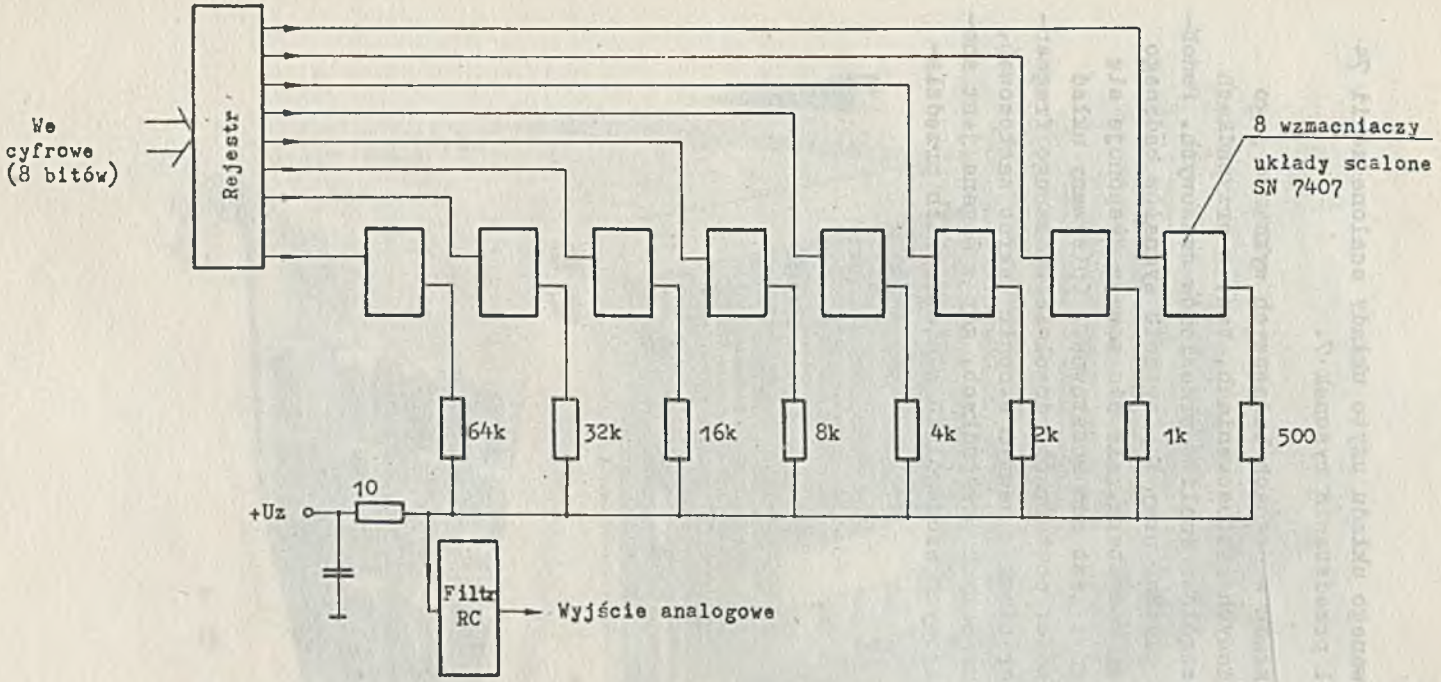
Działanie układu polega na uruchomieniu licznika sterowanego z generatora 100 Hz w momencie przejścia znacznika na taśmie magnetofonowej przez czujnik i wysłaniu sygnału inicjującego zapis do pamięci po określonym czasie, zależnym od ustawienia klawiszy. Schemat układu pokazano na rys. 5.

3.4. Przetwornik cyfrowo-analogowy

Przetwornik cyfrowo-analogowy (rys. 6) składa się z ośmiobitowego rejestru oraz ośmiu wzmacniaczy, których obciążenia stanowią rezystory o wartościach rezystancji proporcjonalnych do kolejnych potęg 2. Wszystkie obciążenia wzmacniaczy są połączone ze źródłem zasilania przez rezystor o wartości mniejszej o dwa rzędy od najmniejszej wartości obciążenia. Napięcie wyjściowe jest pobierane z tego rezystora poprzez trójczłonowy filtr dolnoprzepustowy RC. Stałe czasowe tego filtru dobrane w taki sposób, że tłumienie dla częstotliwości próbkowania wynosi 24 dB, a dla częstotliwości 7 kHz - 9 dB.



Rys. 5. Schemat układu opóźniającego

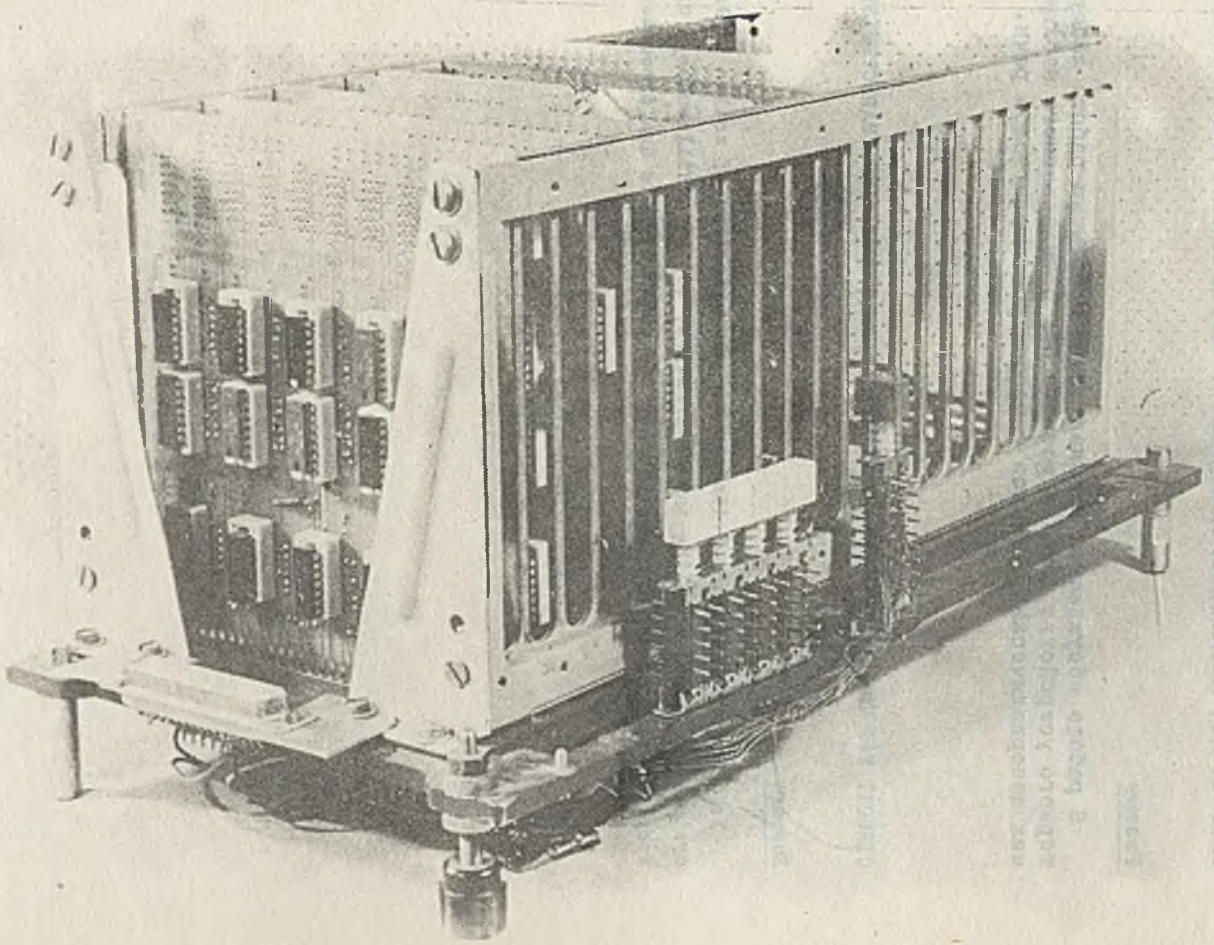


Rys. 6. Schemat przetwornika cyfrowo-analogowego

4. WNIOSKI KOŃCOWE

Do budowy opisywanego układu użyto układy scalone serii 74 TTL. Wykonany model przedstawia rysunek 7.

Wykorzystanie układu w pracach badawczych wykazało jego przydatność w założonych zastosowaniach, tzn. w przypadkach gdy chodziło o szczegółową analizę przebiegów czasowych. Podobne rezultaty można uzyskać przy zapisywaniu sygnałów badanego przebiegu do pamięci komputera, ale nie zawsze dysponuje się takim urządzeniem gdyż jest ono kosztowne. Opisywany układ nie stwarzał możliwości powtarzania dowolnie wybranego fragmentu przebiegu, co ogranicza w pewnym stopniu zakres zastosowań, ale może być pomocny w tych przypadkach, gdy wymagana jest analiza przebiegów czasowych szczególnie w niewielkich przedziałach czasu.



Rys. 7 Model układu

СИСТЕМА ДЛЯ НАБЛЮДЕНИЯ ВРЕМЕННЫХ ПРОБЕГОВ АКУСТИЧЕСКОЙ ЧАСТОТЫ

Резюме

В работе обсуждается идейная схема и принцип работы лабораторного устройства дающего возможность осцилоскопного наблюдения непериодических сигналов с особым учётом сигналов речи.

CIRCUIT FOR THE OBSERVATION OF ACUSTICAL FREQUENCY TIME CHARACTERISTICS

Summary

In the work a pictorial diagram and the principle of action of laboratory equipment enabling the observation, on the oscilloscope, of aperiodical signals with special use to speech signals are discussed.