Walenty Ostasiewicz

ZASTOSOWANIE KOMBINATORYKI DO ROZWIAZYWANIA ZAGADNIEN EKSTREMALNYCH

Praca doktorska

Promotor
Prof.dr Zdzisław Hellwig

Wyższa Szkoła Ekonomiczna
Wrocław 1973

WPROWADZENIE

Praca niniejsza poświęcona jest wybranym zastosowaniom metod kombinatorycznych w rachunku ekonomicznym, ekonometrii i statystyce matematycznej.

Wśród wszystkich pojęć kombinatorycznych centralne miejsce zajmuje pojęcie k o m b i n a c j i, które w kombinatoryce posiada zastrzeżone znaczenie /por. [86,99] /.

Idea kombinacji / od łac. combinatio - łączenie, połączenie/ od najdawniejszych czasów wykorzystywana jest przez człowieka we wszelkiej jego działalności. Jest to jedna z najbardziej ogólnych i prostych idei jakich człowiek nie tworzy dowolnie, lecz które narzuca mu sama natura [15] .

Szerokie możliwości zastosowań ogólnych idei kombinatorycznych zarówno w działalności praktycznej jak i w nauce, spowodowały konieczność opracowania podstaw teoretycznych rachunku kombinatorycznego.

Pierwsze próby stworzenia ogólnej teorii takiego rachunku podjęli niezależnie od siebie W.Leibniz oraz J.Bernoulli. J.Bernoulli rozwija ten rachunek jako podstawowe narzędzie tworzonego przez siebie rachunku prawdopodobieństwa. Natomiast W.Leibniz rozwija kombinatorykę jako podstawową metodę projektowanej Characteristica Universalis [79] . Uzyskane wyniki przedstawił w swej pracy "Dissertatio de Arte Combinatoria", dając początek nowej gałęzi matematycznej zwanej dziś kombinatoryką lub analizą kombinatoryczną [86,99] .

Dokładne określenie przedmiotu badań współczesnej kombinatoryki jest niezwykle trudne. Dlatego też kombinatoryką żartobliwie nazywa się to czym zajmują się kombinatorycy [85], lub zalicza się do niej wszystkie ciekawe problemy o zbiorach skończonych [23], albo też traktuje się ją jako teorię zbiorów skończonych [2].

Do niedawna kombinatoryka zajmowała się głównie dwoma podstawowymi problemami.

Jeden z nich zwany problemem i s t n i e n i a polega na udowodnieniu istnienia przynajmniej jednego odpowiednio zdefiniowanego obiektu kombinatorycznego. Drugi zaś, zwany problemem p r z e l i c z a n i a polega na opracowaniu metod /wzorów/ pozwalających określić liczbę wszystkich obiektów kombinatorycznych o zadanych własnościach.

Oba wymienione problemy są szeroko opracowane zarówno w literaturze klasycznej jak i współczesnej, ważniejsze pozycje podane są w bibliografii /por. [23,30,86,87] /.

Sytuacja w kombinatoryce ulega radykalnej zmianie z chwilą pojawienia się maszyn matematycznych. Pojawienie się tych maszyn spowodowało rozwój szeregu nowych dyscyplin naukowych, w szczególności "nowej" matematyki. Matematykę tę nazywa się bardzo różnie: matematyką cybernetyczną, maszynową, dyskretną, informacyjną itp. /por. [41,60,92] /. Cechą najbardziej charakterystyczną nowej matematyki jest jej charakter kombinatoryczny; operuje ona /podobnie jak matematyka pre-newtonowska/ zbiorami skończonymi. W matematyce tej obok dowodu matematycznego pojawia się nowy rodzaj konstrukcji formalno-logicznych, a mianowicie algorytmy.

Tak więc w kombinatoryce pojawia się trzeci problem, który można nazwać problemem a l g o r y t m i z a c j i. Problem ten polega na opracowaniu algorytmów generowania /otrzymywania/ dowolnych obiektów kombinatorycznych oraz algorytmów manipulowania tymi obiektami: porządkowania, wybierania, selekcji, przekształcania itp.

Maszyny matematyczne nie tylko spowodowały rozwój nowych dyscyplin naukowych, ale ułatwiają też rozwiązanie "kapitalnego problemu naszych czasów" /por. [21] str. 12/, tzn. umiejętności podejmowania trafnych decyzji.

W przypadku gdy zbiór wszystkich możliwych decyzji jest zbiorem wypukłym, a trafność decyzji ocenia się za pomocą pewnej funkcji celu określonej na tym zbiorze, to wyboru trafnej decyzji często można dokonać korzystając ze sposobów dostarczanych przez dziedzinę wiedzy zwaną badaniami operacyjnymi lub rachunkiem ekonomicznym [1, 18, 21, 89] .

Natomiast w przypadku gdy zbiór wszystkich decyzji jest zbiorem skończonym lub dyskretnym, a więc z natury swej kombinatoryczny, a funkcja celu określana jest za pomocą wyrażeń logiczno-kombinatorycznych, to jedynymi metodami, które zawsze mogą ułatwić podjęcie trafnej decyzji, są metody kombinatoryczne.

Do niedawna jednak metody kombinatoryczne nie znajdowały szerszego zastosowania praktycznego. Wydaje się, że można tu wymienić trzy podstawowe przyczyny takiego stanu rzeczy.

Pierwszą z nich było przekonanie o konieczności zbadania wszystkich możliwych wariantów danego problemu. Niesłuszność tego przekonania dla szerokiej klasy zagadnień dotyczą-

cych asymptotycznego zachowania się funkcji kombinatorycznych wykazana jest w pracach S.Ulama [40,59,96] .

Poza tym z chwilą pojawienia się maszyn matematycznych zaczęto rozwijać w kombinatoryce specjalne metody pozwalające znaleźć najlepszy wariant bez konieczności przeliczania wszystkich możliwości. Dano tym samym początek nowej dyscyplinie zwanej p r o g r a m o w a n i e m k o m b i n a t o r y c z n y m [44,87] . Do ważniejszych z opracowanych aktualnie metod tego działu matematyki zalicza się iteracyjne metody typu gradientowego /deterministyczne i losowe/ [8,13,84] , metody osaczania /głównie statystycznego/ [60,89] oraz metody heurystyczno-kombinatoryczne [58] .

Drugą przyczyną ograniczonego stosowania metod kombinatorycznych był rozpowszechniony pogląd o ich rzekomo mniejszej elegancji niż np. metod analizy matematycznej i o tym, że metody te nie dają pełnego rozwiązania problemu. Ostatnio przekonano się jednak /por. [36,53] /, że metody kombinatoryczne dają o w i e l e lepsze rozwiązania skomplikowanych zagadnień aniżeli metody "eleganckie", które często polegają na "aproksymowaniu powierzchni Everestu za pomocą płaszczyzny" [36] .

Trzecią i najważniejszą przyczyną ograniczonego stosowania metod kombinatorycznych jest to, że efektywnie stosować je można głównie przy użyciu współczesnej techniki obliczeniowej, a do tego niezbędne są odpowiednie algorytmy maszynowe.

Problem a l g o r y t m i z a c j i zagadnień kombinatorycznych jest głównym tematem niniejszej pracy.

Większość rozpatrywanych algorytmów dotyczy takich obiektów kombinatorycznych, które definiowane są w postaci skończonych ciągów o określonych własnościach.

Obiektami takimi są np. kombinacje, permutacje, partycje, punkty /wektory/ zbiorów wypukłych itp., które mają zastosowanie m.in. w zagadnieniach optymalizacyjnych. Obiekty te w pracy nazwane są s ł o w a m i k o m b i n a t o r y c z n y m i. Ich definicja podana jest w rozdziale 1, W rozdziale tym podane są też inne podstawowe określenia oraz interpretacja słów kombinatorycznych.

Rozdział 2 poświęcony jest opisowi algorytmów generowania wybranych słów kombinatorycznych. Przy czym, starano się podać metody generowania takich słów, które znajdują bezpośrednie zastosowanie praktyczne, a dla których brak jest odpowiednich algorytmów maszynowych.

Opis każdego algorytmu poprzedzony jest definicją tych słów, które on generuje. Poza tym wskazywane są możliwości zastosowań tych algorytmów. Rozdział ten zakończony jest krótką analizą efektywności podanych algorytmów.

Nawet tak proste obiekty jakimi są słowa kombinatoryczne sprawiają kłopoty przy ich przechowywaniu w pamięci maszyny /zajmują dużo miejsca/. Poza tym manipulowanie ciągami jest mniej wygodne niż operowanie liczbami. Pożądanym byłoby więc odpowiednio ponumerować słowa kombinatoryczne i operować tylko numerami. Aby jednak podejście takie można było

zrealizować potrzebne są odpowiednie algorytmy wzajemnie jednoznacznego numerowania słów oraz identyfikowania ich według numerów.

Algorytmy takie prezentowane są w rozdziale 3.

Na początku tego rozdziału podany jest sposób numerowania elementów zbioru słów kombinatorycznych według dowolnej relacji porządku liniowego określonego w tym zbiorze. Ponieważ do dowolnego numerowania wystarczy umieć numerować dany zbiór oraz zbiór permutacji według relacji porządku leksykograficznego, to w dalszej części rozdziału dla wybranych zbiorów określone są tylko funkcje numerujące według tego porządku. Dla każdej z tych funkcji określona jest funkcja odwrotna, która pozwala identyfikować słowa kombinatoryczne według przyporządkowanych im numerów.

Rozdział zakończony jest wskazaniem możliwości wykorzystania tzw. funkcji numerujących do generowania pewnych ogólniejszych podzbiorów kombinatorycznych.

Rozdział 4 zawiera wybrane zastosowania algorytmów generowania oraz numerowania słów kombinatorycznych rozpatrywanych w poprzednich rozdziałach.

Dziedzinę zastosowań rachunku kombinatorycznego w jego współczesnej postaci umownie podzielono na dwie klasy zagadnień: ekstremalnych i nieekstremalnych.

Na przykładzie zagadnień ekstremalnych pokazane są zalety metod kombinatoryczno-heurystycznych, natomiast w części poświęconej zagadnieniom nieekstremalnym pokazano m.in. że algorytmy numerowania słów kombinatorycznych umożliwiają

maszynowe operowanie tablicami wielowymiarowymi. Tablice takie występują przy badaniu zależności wielowymiarowych zmiennych losowych, w analizie wariancji i w wielu innych zagadnieniach statystycznych, ekonometrycznych itp.

W ostatnim, tzn.5 rozdziale naszkicowane są perspektywy rozwoju metod kombinatorycznych oraz możliwości ich zastosowań.

Formalnie cała praca dzieli się na rozdziały, które dzielą się na paragrafy, a te z kolei na punkty. Numeracja wzorów dokonywana jest w sposób ciągły, odrębnie dla każdego rozdziału. Pozycje literatury zaś umieszczone w nawiasach kwadratowych, posiadają numerację ciągłą dla całej pracy. Do pracy załączony jest wykaz ważniejszych programów, które są odpowiednikiem bądź opisywanych w pracy algorytmów, bądź też są to programy wykorzystujące te algorytmy. Programy te zostały zapisane w języku MOST-1 lub ALGOL-1204 i wypróbowane odpowiednio na maszynach ODRA-1003 i ODRA-1204. Tabulogramy programów /lub procedur/ wraz z danymi testowymi i opisem sposobu korzystania znajdują się w bibliotece Laboratorium Obliczeniowego przy Instytucie Metod Rachunku Ekonomicznego.

W przypadku powoływania się w tekście pracy na daną pozycję tego wykazu, jej numer poprzedzony jest literą P.

1. POJĘCIA PODSTAWOWE

1.1. S ł o w a k o m b i n a t o r y c z n e

Niech dany będzie pewien skończony zbiór A kolejnych liczb całkowitych, który nazywany będzie a l f a b e t e m, a jego elementy l i t e r a m i.
Jeśli zbiór A jest zbiorem kolejnych n liczb naturalnych to oznaczany on będzie symbolem N_n. Liczebność dowolnego zbioru X oznaczana będzie symbolem $|X|$.

Każdy skończony ciąg elementów zbioru A nazwiemy s ł o w e m k o m b i n a t o r y c z n y m nad alfabetem A.

Zbiór wszystkich możliwych m-literowych słów nad n elementowym alfabetem oznaczany będzie symbolem K. Dowolny element tego zbioru zapisywany będzie w postaci wektora:

$$k = (k_1, k_2, \ldots, k_m) \qquad /1/$$

Najczęściej słowa oznaczane będą symbolami k, x, y, p; przy tym jeśli nie będą czynione żadne inne założenia, to symbole te zawsze oznaczać będą słowa nad alfabetem będącym zbiorem liczb całkowitych. Stąd też zapis $a \leqslant x_i \leqslant b$ oznacza, że wartością x_i może być dowolna liczba całkowita z przedziału $[a,b]$ przy czym zadawane liczby a i b będą zawsze całkowitymi.

Dwa dowolne słowa można łączyć w jedno słowo za pomocą tzw. operacji k o n k a t e n a c j i [29] zwanej też iloczynem prostym[1]. Niech $x = (x_1, x_2, \ldots, x_s)$ i

---

[1] Operacja ta analogicznie jak w języku PL/1 oznaczana będzie symbolem $\|$.

$y = (y_1, y_2, \ldots, y_t)$ są pewnymi słowami. W wyniku konkatenacji słów x i y otrzymujemy słowo $k = x \| y = (x_1, x_2, \ldots, x_s) \| (y_1, y_2, \ldots, y_t) = (x_1, x_2, \ldots, x_s, y_1, y_2, \ldots, y_t)$.

W przypadku gdy słowo x jest jednoliterowe i nie będzie to prowadziło do nieporozumień, to zamiast zapisu $x = (x_1)$ używany będzie zapis x_1 lub x.

W zbiorze K tzn. zbiorze wszystkich m literowych słów nad n elementowym alfabetem wyprowadzimy relację porządku liniowego oznaczaną symbolem $\preccurlyeq$ a zwaną porządkiem l e k s y k o g r a f i c z n y m, którą definiuje się następująco [65,83] :

$$k \preccurlyeq k' \Leftrightarrow k = k' \vee \exists_i \left(k_i < k'_i \wedge \forall_j \; j < i \Rightarrow k_j = k'_j \right)$$

Oprócz tej relacji potrzebna będzie relacja $\prec$, której definicja jest następująca [65] :

$$k \prec k' \Leftrightarrow k \preccurlyeq k' \wedge k \neq k' \qquad \text{dla dowolnych } k, k' \in K.$$

Element $k' \in K$ nazywa się n a j m n i e j s z y m lub pierwszym jeśli $k' \prec k$ dla każdego $k \neq k' \in K$. Analogicznie słowo $k' \in K$ nazywa się n a j w i ę k s z y m lub ostatnim jeśli

$k \prec k'$ dla każdego $k \in K$ różnego od k'.

1.2. Interpretacja słów kombinatorycznych

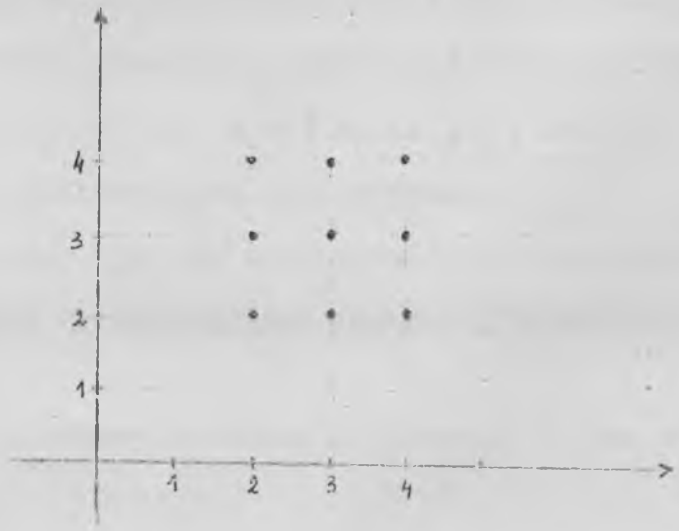
Przy algorytmizacji zagadnień kombinatorycznych wykorzystywana będzie dwojaka interpretacja elementów zbioru K:

- geometryczna,
- liczbowa.

Pomijamy tu statystyczną interpretację, którą można znaleźć w [31].

Załóżmy, że dany jest n elementowy alfabet A tzn. zbiór kolejnych liczb całkowitych $A = \{a, a+1, \ldots, b\}$, gdzie $b-a+1 = n$. Zbiór wszystkich możliwych ciągów m elementowych oznaczamy zgodnie z umową symbolem K.

W interpretacji geometrycznej dowolny element zbioru K traktować będziemy jako pewien punkt w m wymiarowej przestrzeni liczb całkowitych. Zbiór K jest wówczas kostką w tej przestrzeni. Dla przykładu weźmy zbiór $A = \{2,3,4\}$ oraz przyjmijmy, że m=2, wówczas $K = \{(2,2), (2,3), (2,4), (3,2), (3,3), (3,4), (4,2), (4,3), (4,4)\}$ tzw. zbiorem K jest kostka dwuwymiarowa przedstawiona na poniższym rysunku.



W interpretacji liczbowej, każdy element zbioru K traktujemy jako zapis pewnej liczby m cyfrowej, w której k_i oznacza cyfrę i-tej pozycji spełniającą warunek:

$$a \leqslant k_i \leqslant b \qquad /2/$$

Zbiór K jest więc zbiorem wszystkich liczb m cyfrowych o podstawie[1] b-a+1. Cyfra o najmniejszej wartości oznaczona jest symbolem a, zaś cyfra o największej wartości symbolem b.

W dalszych rozważaniach wygodniej będzie zamiast liczb o stałej podstawie rozpatrywać liczby o zmiennej podstawie tzn. takich, w których każda cyfra posiada swój zakres zmienności. Ten ogólniejszy przypadek otrzymamy wówczas, gdy warunek /2/ zamienimy następującym warunkiem:

$$d_i \leqslant k_i \leqslant g_i \qquad i=1,2,\ldots,m \qquad /3/$$

gdzie d_i oraz g_i są to ustalone pewne liczby całkowite.

Jeśli przyjmiemy, że $d_i = 0$ oraz $g_i = 9$ dla $i=1,2,\ldots,m$, to zbiór K będzie zbiorem m-cyfrowych nieujemnych całkowitych liczb dziesiętnych.

Korzystając z podanych interpretacji elementy zbioru K nazywać będziemy liczbami, wektorami lub punktami. Natomiast litery słowa $x = (x_1,\ldots,x_m)$ nazywać będziemy składowymi, współrzędnymi lub cyframi.
Przyjmujemy przy tym, że symbol $\omega(i,x)$ oznacza i-tą współrzędną wektora x natomiast symbol $\nu(c,x)$ oznaczać zawsze

---

[1] Definicję podstawy systemu liczbowego można znaleźć w [51].

będzie numer /indeks/ tej składowej wektora x, która jest równa liczbie c. Dla porządku możemy przyjąć, że w przypadku nieokreśloności, wartością funkcji ω oraz ν jest nieskończoność.

2. GENEROWANIE

2.1. W s t ę p

Mając dany alfabet $A = \{a, a+1, \ldots, b\}$, można utworzyć nad nim $(b-a+1)^m$ różnych m literowych słów kombinatorycznych. Wśród tych słów można wyodrębnić pewne podzbiory słów o określonych własnościach.

Algorytmy generowania wybranych podzbiorów będą rozpatrywane w niniejszym rozdziale. Będą to punkty prostopadłościanu, punkty zbioru wypukłego, kombinacje oraz tzw. podziały /partycje/.

Rozpatrywane tu algorytmy znajdują zastosowanie[1] głównie w programowaniu kombinatorycznym, w statystyce wielowymiarowej oraz w programowaniu maszyn cyfrowych. Konkretne przykłady zastosowań będą sygnalizowane przy rozpatrywaniu odpowiednich algorytmów, a niektóre z nich są rozpatrzone w rozdz. 4.

2.2. P r o s t o p a d ł o ś c i a n

Niech będą dane dwa wektory

$$d = (d_1, d_2, \ldots, d_m) \quad \text{oraz} \quad g = (g_1, g_2, \ldots, g_m),$$

takie, że ich współrzędne są nieujemnymi literami całkowitymi spełniającymi warunek:

---

[1] por. rozdział 4 i 5.

$$d_i \leqslant g_i \quad \text{dla} \quad i = 1,2,\ldots,m.$$

P r o s t o p a d ł o ś c i a n e m m wymiarowym oznaczonym symbolem $P_m[d:g]$, nazywany będzie zbiór tych punktów

$$p = (p_1,p_2,\ldots,p_m),$$

współrzędne których spełniają warunek:

$$d_i \leqslant p_i \leqslant g_i \quad i = 1,2,\ldots,m. \qquad /1/$$

Liczba $l_i = g_i - d_i + 1$ nazywana będzie długością i-tej krawędzi. Gdy $l_1 = l_2 = \ldots = l_m$, to prostopadłościan nazywa się k o s t k ą, jeśli zaś $d_i = 1$ oraz $g_i = n$, to prostopadłościan nazywa się zbiorem w a r i a c j i z p o w t ó r z e n i a m i z n po m.

Przy konstrukcji algorytmu generowania punktów prostopadłościanu wykorzystana będzie interpretacja liczbowa słów kombinatorycznych /por. par. 1.3/.

Nietrudno zauważyć, że najmniejszą liczbą spełniającą warunek /1/ jest liczba $d = (d_1,d_2,\ldots,d_m)$, natomiast największą - liczba $g = (g_1,g_2,\ldots,g_m)$.

Aby otrzymać wszystkie punkty prostopadłościanu należy więc do najmniejszej liczby d, dodawać jedynkę tak długo, aż otrzymamy liczbę największą g. Dodawanie jedynki należy wykonywać według następującego algorytmu.

W danej liczbie $k = (k_1,k_2,\ldots,k_m)$ szukamy pierwszej /licząc od strony prawej/ cyfry k_s mniejszej od swej maksymalnej wartości g_s. Do cyfry k_s dodajemy jedynkę,

natomiast jako wartości cyfr $k_{s+1}, k_{s+2}, \ldots, k_m$ przyjmujemy ich niminalne wartości $d_{s+1}, d_{s+2}, \ldots, d_m$.

Algorytm ten wygodnie jest przedstawić w postaci następującego schematu:

$$\begin{array}{l} k = (k_1, k_2, \ldots, k_{s-1}, k_s, g_{s+1}, \ldots, g_m) \\ + \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 1 \\ \hline k+1 = (k_1, k_2, \ldots, k_{s-1}, k_s+1, d_{s+1}, \ldots, d_m) \end{array} \qquad /2/$$

Zapis $k+1$ zawsze oznaczać będzie operację dodawania jedynki do danego słowa k, lub też wynik tej operacji tzn. takie słowo, które w uporządkowaniu leksykograficznym występuje bezpośrednio po słowie k.

Opisany algorytm generowania punktów prostopadłościanu występuje w szeregu innych algorytmach / por. następne paragrafy tego rozdziału oraz rozdz. 5/, poza tym okazał się szczególnie przydatny przy operowaniu tablicami wielowymiarowymi /por. rozdz. 4/.

2.3. Z b i ó r w y p u k ł y

Z b i o r e m w y p u k ł y m oznaczonym symbolem ZW(n) nazywany będzie zbiór tych punktów

$$x = (x_1, x_2, \ldots, x_n)$$

o współrzędnych całkowitych nieujemnych, które spełniają warunki

$$\sum_{j=1}^{n} w_{ij} x_j^2 \leq z_i \qquad i = 1,2,\ldots,m \qquad /3/$$

gdzie w_{ij} oraz z_i są nieujemnymi liczbami całkowitymi, poza tym przyjmuje się, że wykładnik potęgi q jest jednakowy dla całego układu i wynosi 1 lub 2.

O t o c z k ą w y p u k ł ą nazywany będzie następujący zbiór

$$OW = \left\{ x \in ZW(n) : x \oplus 1 \notin ZW(n) \ \forall \ x \ominus 1 \notin ZW(n) \right\},$$

gdzie symbolem $x \oplus 1$ oznaczono dodanie jedynki do jakiejkolwiek współrzędnej wektora x, oraz symbolem $x \ominus 1$ odjęcie jedynki. Innymi słowy punkt x ze zbioru wypukłego $ZW(n)$ należy do otoczki wypukłej wówczas, jeśli po dodaniu lub odjęciu jedynki od jakiejkolwiek współrzędnej tego punktu, przestaje on należeć do zbioru wypukłego.

B r z e g i e m zbioru wypukłego nazywany będzie zbiór tych punktów otoczki wypukłej, które spełniają przynajmniej jedną równość w układzie /3/.

W szczególnym przypadku, gdy $m=1$, punktami brzegowymi będą wszystkie nieujemne całkowitoliczbowe rozwiązania równania

$$w_1 x_2^q + w_2 x_2^q + \ldots + w_n x_n^q = z.$$

Rozpatrzymy teraz algorytmy generowania zdefiniowanych wyżej zbiorów. Algorytmy te można zastosować między innymi przy optymalizowaniu "brzydkich", skomplikowanych funkcji celu oraz przy dyskryminacji zbiorów skończonych /por. [12,37] oraz rozdz. 4/.

Załóżmy na chwilę, że umiemy generować wszystkie punkty zbioru $ZW(n-1)$. Wówczas aby wygenerować punkty zbioru $ZW(n)$ należy dla każdego z punktów $x=(x_1, x_2, \ldots, x_{n-1}) \in ZW(n-1)$

jako wartość n-tej współrzędnej dopisać kolejno wartości:

$$0,1,2,\ldots,s(n) \qquad /4/$$

gdzie $s(n) = \text{entier}\left(\sqrt[q]{\min_{1 \le i \le m} r_i} \right)$

$$r_i = \frac{z_i - \sum_{j=1}^{n-1} w_{ij} x_j^q}{w_{in}}, \quad i = 1,2,\ldots,m.$$

Nietrudno zauważyć, że każdy z tak i tylko tak otrzymanych punktów $x = (x_1, x_2, \ldots, x_n)$ będzie spełniał układ /3/ a tym samym należał do zbioru $ZW(n)$.

Łatwy tego dowód można przeprowadzić korzystając z prostej interpretacji geometrycznej przedstawionego wyżej algorytmu: Przez każdy z wygenerowanych punktów $x = (x_1, x_2, \ldots, x_{n-1})$ z przestrzeni $n-1$ wymiarowej prowadzona jest prosta równoległa do osi X_n.

Jako wartość n-tej współrzędnej brane są wszystkie punkty leżące na tej prostej od 0 aż do momentu pierwszego przebicia pewnej płaszczyzny układu /3/.

Aby taki algorytm generowania punktów zbioru $ZW(n)$ można było zrealizować należy umieć generować punkty zbioru $ZW(n-1)$, ale w tym celu wystarczy w przeprowadzonym rozumowaniu zamienić n na n-1 natomiast n-1 na n-2 itd. aż do momentu, gdy $n=1$.

Przyjmując, że punktami zbioru $ZW(1)$ są kolejne liczby:

$$0,1,2,\ldots,s(1) = \text{entier}\left(\sqrt[q]{\min_i \frac{z_i}{w_{i1}}} \right)$$

otrzymujemy rekurencyjną procedurę generowania punktów zbioru $ZW(n)$.

Podaną procedurę łatwo można rozszerzyć na ogólniejszy przypadek, gdy zbiór wypukły określony jest w dodatniej ćwiartce układu współrzędnych za pomocą dowolnych powierzchni rzędu drugiego o nieujemnych i całkowitych współczynnikach.

Procedurę tę łatwo można też przystosować do generowania punktów otoczki wypukłej (por. [P12]) . W tym celu do każdego z wygenerowanych punktów $x = (x_1, x_2, \ldots, x_{n-1}) \in ZW(n-1)$ należy, jako wartość n-tej współrzędnej przyjąć liczbę 0 oraz liczbę $s(n)$. W przypadku gdy punkt $x = (x_1, x_2, \ldots, x_{n-1})$ należy do otoczki wypukłej n-1 wymiarowej, wówczas jako wartość n-tej współrzędnej należy przyjmować kolejno wszystkie wartości:

$$0, 1, 2, \ldots, s(n).$$

Zauważmy, że jeśli do punktu $x = (x_1, \ldots, x_{n-1})$ jako n-tą współrzędną dopiszemy liczbę $s(n)$ określoną we wzorze /4/ i przy tym taką, że dla pewnego $1 \leq i_o \leq m$ zachodzi równość:

$$s(n) = \sqrt[q]{r_{i_o}}, \qquad /5/$$

to tak otrzymany punkt $x = (x_1, x_2, \ldots, x_n)$ będzie punktem brzegowym, gdyż punkt ten spełnia równość:

$$\sum_{j=1}^{n} w_{i_o j} x_j^q = z_{i_o}.$$

W ten sposób otrzymujemy algorytm generowania punktów brzegowych: dla każdego z punktów $x = (x_1, x_2, \ldots, x_{n-1}) \in ZW(n-1)$

sprawdzamy warunek /5/, jeśli jest on spełniony, to punkt brzegowy otrzymujemy jako konkatenację $(x_1,x_2,\ldots,x_{n-1}) \| s(n)$, jeśli zaś warunek /5/ nie jest spełniony, to badamy go dla następnego punktu $x \in ZW(n-1)$ itd. aż przebadamy wszystkie $x \in ZW(n-1)$.

Dla niektórych szczególnych przypadków opisanych wyżej algorytmów można skonstruować znacznie efektywniejsze procedury. Możliwość taka została wykorzystana w opracowanych programach na maszynę Odra 1204 (por. [P11,P13,P19]) . Przykładowo niżej wymienione są 3 ważniejsze przypadki.

Przypadek 1.

Gdy $m=1$ oraz $q=1$, to punktami brzegowymi są rozwiązania następującego równania[1]:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = z$$

Zamiast w_{1j} użyto symbolu a_j.

Algorytm generowania takich punktów znajduje zastosowanie w programowaniu całkowitoliczbowym np. przy optymalizowaniu tzw. funkcji plecakowych (por. [44]) . Efektywność stosowania tego algorytmu w porównaniu z programowaniem dynamicznym została przeanalizowana w [73] .

Przypadek 2.

Gdy $n-1$ oraz $q=2$, to punktami zbioru wypukłego są punkty elipsoidy:

$$a_1x_1^2 + a_2x_2^2 + \ldots + a_nx_n^2 \leq z$$

[1] W praktyce często występuje potrzeba rozwiązywania takiego równania przy dodatkowym warunku, że x_i mają przyjmować wartości 0 lub 1. Ciekawy i efektywny algorytm typu branch-and-bound rozwiązania takiego zadania sformułowany został przez W.Bukietyńskiego (por. [11] oraz [P20]) .

Algorytm generowania takich punktów znajduje zastosowanie przy przeprowadzaniu doświadczeń ekstremalnych (por.[66,68])

Przypadek 3.

Gdy m=1, q=1 oraz $a_1 = a_2 = \ldots = a_n = 1$, to punktami brzegowymi są rozwiązania równania

$$x_1 + x_2 + \ldots + x_n = z,$$

tzn. partycje uporządkowane (por. [30,86]) liczby naturalnej z. Algorytm ten również znajduje zastosowanie w planowaniu doświadczeń ekstremalnych /służy do wyznaczania miejsca na sympleksie n-1 wymiarowym, gdzie mają być przeprowadzone doświadczenia [68] /. Poza tym algorytm generowania partycji znajduje zastosowanie przy wielomianowej dyskryminacji zbiorów skończonych (por.[90]).

2.4. K o m b i n a c j e

2.4.1. Kombinacje z powtórzeniami[1]

K o m b i n a c j ą z p o w t ó r z e n i a m i z n po m nazywa się taki punkt

$$k = (k_1, k_2, \ldots, k_n),$$

który spełnia warunki:

$$1 \leq k_i \leq n$$
$$i < j \Rightarrow k_i \leq k_j \qquad /6/$$

[1] Algorytm generowania takich kombinacji znajduje zastosowanie m.in. przy operowaniu wielowymiarowymi tablicami trójkątnymi [71] .

Algorytm generowania jest następujący.

Bierzemy najmniejszą liczbę k spełniającą warunki /6/ tzn. $k = (1,1,\ldots,1)$ i dodajemy do niej jedynkę tak długo, aż otrzymamy największą liczbę tzn. $k = (n,n,\ldots,n)$. Zauważmy, że z definicji relacji porządku leksykograficznego wynika, że dwie kombinacje z powtórzeniami k i k' zajmujące kolejne miejsca w tym uporządkowaniu posiadają następujące własności:

1. $k'_s = k_s + 1$ dla pewnego $1 \leq s \leq m$
2. $k'_i = k_i$ dla $i = 1,2,\ldots,s-1$
3. $k'_i = k'_s$ dla $i = s+1,\ldots,m$
4. $k_i = m$ dla $i = s+1,\ldots,m$

Stąd też mając kombinację k generowanie kombinacji $k+1$ można zapisać w postaci następującego schematu:

$$k = (k_1, k_2, \ldots, k_{s-1}, k_s, n, n, \ldots, n)$$

$$+ \qquad\qquad\qquad\qquad\qquad\qquad 1$$

$$k+1 = (k_1, k_2, \ldots, k_{s-1}, k_s+1, k_s+1, \ldots, k_s+1)$$

To znaczy jeśli w danej kombinacji k mamy taką sytuację, że $k_s < n$ oraz $k_i = n$ dla $i > s$, to następną w porządku leksykograficznym kombinację otrzymamy w ten sposób, że do współrzędnej k_s dodajemy jedynkę i jako wartości wszystkich pozostałych k_i dla $i > s$ współrzędnych przyjmujemy wartość współrzędnej k_s.

2.4.2. Kombinacje bez powtórzeń[1]

K o m b i n a c j ą b e z p o w t ó r z e ń z n po m nazywamy punkt

$$k = (k_1, k_2, \ldots, k_m),$$

który spełnia warunki:

$$1 \leq k_i \leq n$$

$$i < j \Rightarrow k_i < k_j$$

Zauważmy, że najmniejszą liczbą spełniającą te warunki jest liczba $k = (1,2,\ldots,m)$, natomiast największą

$$k = (n - m+1,\ n - m+2,\ \ldots,\ n).$$

Innymi słowy kombinacją bez powtórzeń jest taka liczba (lub punkt prostopadłościanu), $k = (k_1, k_2, \ldots, k_m)$, że

$$d_i \leq k_i \leq g_i$$

gdzie $d_i = i$

$g_i = n - m + i$ dla $i = 1,2,\ldots,m$.

Załóżmy, że dana jest kombinacja $k = (k_1, \ldots, k_s, \ldots, k_m)$ taka, że $k_s < g_s$ oraz $k_i = g_i$ dla $i > s$. Wówczas następną w porządku leksykograficznym kombinację k otrzymamy jeśli do kombinacji k dodamy jedynkę według następującego schematu:

---

[1] Rozpatrywany tu algorytm zastosowano m.in. w programie optymalnego wyboru predyktant (por. [33] oraz [P15]), a także przy obliczaniu współczynnika zależności (por. [32] oraz [P25]).

$$k = (k_1, k_2, \ldots, k_{s-1}, k_s, \ldots, k_m)$$

$$+ \qquad\qquad\qquad\qquad\qquad\qquad 1$$

$$\acute{k} = k+1 = (k_1, k_2, \ldots, k_{s-1}, k_s+1, k_s+2, \ldots, k_s+m\ s+1)$$

Jeśli zaś dla każdego i zachodzi równość $k_i = g_i$, to jedynki dodać nie można, co świadczy o wygenerowaniu wszystkich kombinacji.

Zauważmy, że w zależności od tego w jaki sposób określimy wektory $d = (d_1, d_2, \ldots, d_m)$ oraz $g = (g_1, g_2, \ldots, g_m)$, to według podanego algorytmu możemy otrzymać kombinacje o różnych właściwościach.

Kilka przykładów podajemy niżej. Przyjmijmy, że ze zbioru $N_n = \{1,2,\ldots,n\}$ wyodrębniony został podzbiór $N_s = \{1,2,\ldots,s\}$ taki, że $s \leq n-m+1$.

Przykład 1.

Przyjmując $d_i = s + 1$

$$g_i = n - m + 1 \qquad \text{dla } i = 1,2,\ldots,m$$

otrzymamy takie kombinacje z n po m, z których żadna nie zawiera ani jednego elementu zbioru N_s.
Jest to oczywiste z tego względu, że najmniejsza kombinacja ma wówczas postać $k = (s+1, s+2, \ldots, s+m)$ tzn. zawiera składowe większe od s, a każda następna kombinacja tym bardziej będzie zawierała składowe większe od s.

Przykład 2.

Przyjmując $d_1 = 1$, $g_1 = s$,

$$d_i = s + i - 1 \qquad \text{dla } i = 2,3,\ldots,m$$

$$g_i = n - m + 1 \qquad \text{dla } i = 2,3,\ldots,m$$

wygenerujemy takie kombinacje z n po m, że każda z nich zawiera dokładnie jeden element zbioru N_s.
Jest to równie oczywiste jak w przykładzie 1.

Przykład 3.

Przyjmując $d_i = i$ dla $i = 1,2,\ldots,m$

$g_i = i$ dla $i = 1,2,\ldots,s$

oraz $g_j = n-m+j$ dla $j = s+1, s+2,\ldots,m$

otrzymamy tylko takie kombinacje z n po m, że każda z nich zawiera wszystkie elementy zbioru N_s.

Przykład 4.

Przyjmując $d_i = i$ dla $i=1,2,\ldots,m$

oraz $g_i = s$, $g_j = n-m+j$ dla $j=2,3,\ldots,m$

wygenerujemy tylko takie kombinacje, że każda z nich zawiera pewien niepusty podzbiór zbioru N_s.

Przykład 5.

Jeśli przyjmiemy $d_i \neq i$ dla $i=1,2,\ldots,m$

$g_i = i$ dla $i=1,2,\ldots,s$

$g_{s+1} = c$

oraz $g_j = n-m+j$ dla $j=s+2, s+3,\ldots,m$

przy założeniu, że $s+1 \leqslant c \leqslant n-m+s-1$, to każda z nich zawiera dokładnie wszystkie elementy zbioru N_s oraz jedną spośród liczb $s+1, s+2, \ldots, c$.

2.5. P o d z i a ł y

2.5.1. Podział zbioru

Załóżmy, że dany jest n elementarny zbiór liczb całkowitych X.

P o d z i a ł e m (por. [23,82]) m-elementowym zbioru X nazywamy ciąg podzbiorów tego zbioru:

$$X_1; X_2; \ldots; X_m \qquad /7/$$

taki, że

1. $\bigcup_{i=1}^{m} X_i = X$
2. $i \neq j \Rightarrow X_i \cap X_j = \emptyset$
3. $X_i \neq \emptyset$
4. liczność każdego X_i jest ustalona i wynosi n_i przy czym $\sum_{i=1}^{m} n_i = n$ gdzie $n = |X|$.

Algorytm generowania jest prosty: składowymi każdego X_i są możliwe podzbiory n_i - elementowe zbioru $X \setminus \bigcup_{j=1}^{i-1} X_j$

dla $i = 2,3,\ldots,m$.

Natomiast X_1 stanowią wszystkie n_1-elementowe podzbiory zbioru X. Aby otrzymać pewien n_i elementowy podzbiór zbioru X, generujemy kombinację z n po n_i, która określa numery tych elementów zbioru X, które mają wejść do n_i-elementowego podzbioru.

Zrezygnujmy teraz z warunku 4.

Algorytm generowania takich podziałów jest następujący.

Jako składowe X_i dla $i = 1,2,\ldots,m-1$ przyjmujemy kolejno wszystkie podzbiory $1,2,\ldots,w_i$-elementowe zbioru

$$X \setminus \bigcup_{j=1}^{i-1} X_j$$

gdzie $w_i = n - \sum_{j=1}^{i-1} n_j - m + i$.

Natomiast jako składową X_m przyjmujemy podzbiór $X \setminus \bigcup_{j=1}^{m-1} X_j$

tzn. pozostałą część zbioru X.

Algorytm ten wynika z następujących niżej rozważań.

Zauważmy, że aby był spełniony warunek 3, to dowolne X_i może być podzbiorem co najwyżej $n - m + 1$ elementowym. Wówczas wszystkie pozostałe podzbiory ciągu /7/ będą jednoelementowe, tzn. najmniejsze jakie mogą być.

Załóżmy, że utworzone są podzbiory $X_1, X_2, \ldots, X_{i-1}$ tak, aby spełnione były warunki 2 i 3. W sumie podzbiory te dają zbiór $\bigcup_{j=1}^{i-1} X_j$ złożony z $\sum_{j=1}^{i-1} n_j$ elementów.

Aby był spełniony warunek 2, to podzbiór X_i może zawierać elementy tylko ze zbioru $X \setminus \bigcup_{j=1}^{i-1} X_j$, liczność którego wynosi $n - \sum_{j=1}^{i-1} n_j$.

Aby pozostałe $X_{i+1}, X_{i+2}, \ldots, X_m$ podzbiory nie były puste, to podzbiór X_i może zawierać co najwyżej w_i elementów, gdzie $w_i = n - \sum_{j=1}^{i-1} n_j - m + i$.

Tak więc podzbiór X_i może być podzbiorem $1,2,\ldots,w_i$ elementowym zbioru $X \setminus \bigcup_{j=1}^{i-1} X_j$.

Jeśli wygenerowane są podzbiory $X_1, X_2, \ldots, X_{m-1}$ tak, że spełniają żądane warunki, to warunek 1 wymaga, aby jako podzbiór X_m przyjąć różnicę $X \setminus \bigcup_{j=1}^{n-1} X_j$.

Pomijamy tu nieco trudniejszy algorytm generowania takich podziałów zbioru X gdy kolejność składowych $X_1, X_2, \ldots, X_m$ nie odgrywa żadnej roli. Ilość takich podziałów określa się za pomocą tzw. liczb Stirlinga drugiego rodzaju (por. [86]).

2.5.2. Podział liczby naturalnej

Podziały liczby naturalnej nazywa się zwykle partycjami [86], rozróżnia się przy tym partycje uporządkowane zwane też kompozycjami [86] oraz partycje nieuporządkowane, zwane krótko partycjami. Jeśli nie będzie to prowadziło do nieporozumień, to zarówno jedne jak i drugie nazywać będziemy partycjami.

P a r t y c j ą u p o r z ą d k o w a n ą liczby n nazywamy (por. [30]) taki punkt

$x = (x_1, x_2, \ldots, x_m)$, że

$$x_1 + x_2 + \ldots + x_m = n$$

oraz każde x_i jest nieujemną liczbą całkowitą.
Algorytm generowania opisany był w paragrafie 2.3.
Algorytm ten służył do wygenerowania wszystkich partycji, niżej podajemy inny algorytm, za pomocą którego można wygenerować kolejną partycję na podstawie danej[1].
Weźmy dwie kolejne partycje w porządku leksykograficznym, wcześniejszą z nich oznaczmy symbolem $x = (x_1, \ldots, x_m)$ natomiast późniejszą symbolem $x' = (x'_1, x'_2, \ldots, x'_m)$.
Partycje te posiadają następujące własności:

1. $x'_s = x_s + 1$ dla pewnego $1 \leqslant s \leqslant m$
2. $x'_i = x_i$ dla $i = 1,2,\ldots,s-1$
3. $x_{s+1} = n - \sum_{j=1}^{s} x_j$
4. $x_j = 0$ dla $j = s+2, s+3, \ldots, m$
5. $x'_j = 0$ dla $j = s+1, s+2, \ldots, m-1$
6. $x'_m = x_{s+1} - 1$

Pierwsze dwie własności wynikają bezpośrednio z definicji porządku leksykograficznego i z założenia, że $x' = x+1$.

---

[1] Algorytm taki jest bardzo wygodny przy stosowaniu go jako odrębnej procedury w innych algorytmach.

Ponieważ wszystkie składowe muszą w sumie dawać liczbę n, to największą wartością składowej x_i może być liczba

$$n - \sum_{j=1}^{i-1} x_j$$

wówczas pozostałe składowe $x_{i+1}, x_{i+2}, \ldots, x$ są równe zeru.

Ponieważ $x'_s = x_s + 1$, to właśnie składowa x_{s+1} musi posiadać najwyższą wartość, w przeciwnym bowiem razie partycję następną po x można by było otrzymać dodając jedynkę do x_{s+1} a nie do x_s. Jeśli składowa x_{s+1} posiada największą wartość, to wszystkie następne muszą więc być równe zeru /własność 4/. Aby między partycją x i x' nie można było zmieścić żadnej innej, to składowe $x'_{s+1}, x'_{s+2}, \ldots, x'_{m-1}$ muszą posiadać najmniejsze wartości tzn. zera, natomiast składowa x'_m musi posiadać taką wartość aby wszystkie w sumie dawały liczbę n, a ta wartość równa jest liczbie $x_{s+1} - 1$. W ten sposób otrzymujemy następujący schemat dodawania jedynki do danej partycji x:

$$x = (x_1, x_2, \ldots, x_{s-1}, x_s, x_{s+1}, 0, 0, \ldots, 0)$$

$$+ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 1$$

$$x' = x+1 = (x_1, x_2, \ldots, x_{s-1}, x_s+1, 0, 0, 0, \ldots, x_{s+1} - 1)$$

Jeśli zażądamy aby składowe partycji były liczbami naturalnymi, to ilość ich wyraża się wówczas wzorem

$$C_{n-1}^{m-1}$$

Do wygenerowania ich można posłużyć się opisanym wyżej algorytmem lub opisanym w paragrafie 2.3. W tym celu należy generować partycje liczby $n-m$ i do każdej składowej dodawać jedynkę. Dla generowania kompozycji liczby n o składowych dodatnich istnieje też algorytm 72a-procedury o nazwie comp.1 i comp. 2 (por. [3] str.43).

Natomiast dla wygenerowania partycji nieuporządkowanych [30] istnieje algorytm o numerze 114a w postaci procedury cp generator (por. [4] str. 22).

2.6. P r z e k s z t a ł c e n i a

Wiadomo [83], że dla dwu równolicznych zbiorów istnieje przekształcenie wzajemnie jednoznaczne jednego zbioru na drugi. Fakt ten można wykorzystać do generowania sytuacji kombinatorycznych. Jeśli bowiem umiemy generować elementy zbioru X i znamy postać odwzorowania $f: X \rightarrow Y$, to tym samym potrafimy wygenerować elementy zbioru Y.

W paragrafie tym rozpatrzone będą dwa przykłady wykorzystania takiej możliwości (por. [P20, P21]). Pokażemy mianowicie jak można otrzymać permutację z powtórzeniami na podstawie danego podziału zbioru oraz odwrotnie, w jaki sposób otrzymać podział z permutacji. W drugim zaś przykładzie podana będą wzajemnie jednoznaczne odwzorowania zbiorów kombinacji z powtórzeniami i partycji.

Przykład 1.

Zbiór podziałów zdefiniowany w paragrafie 2.5 składa się jak wiadomo (por. [23,82]) z $\frac{n!}{n_1!\, n_2! \ldots n_m!}$ elementów, z tylu też elementów składa się zbiór permutacji z powtórzeniami /definicję którego można znaleźć w [99] /.

Algorytm generowania podziałów zbioru opisany był w par. 2.5, teraz podamy sposób jak z danego podziału

$$X_1, X_2, \ldots, X_m \qquad /8/$$

otrzymać permutację

$$p = (p_1, p_2, \ldots, p_n). \qquad /9/$$

Zapiszmy podział /8/ w postaci wektora o n składowych:

$$X = X_{11}, X_{12}, \ldots, X_{1n_1}, X_{21}, \ldots, X_{2n_2}, \ldots, X_{m1}, X_{m2}, \ldots, X_{m\,n_m} \qquad /10/$$

gdzie $X_{i1}, X_{i2}, \ldots, X_{in_i}$ są to zapisane w kolejności rosnącej elementy zbioru X_i $(i=1,2,\ldots,m)$.

Aby na podstawie podziału /10/ otrzymać permutację /9/ należy dla każdego $j=1,2,\ldots,m$ wykonać następujące podstawienia:

$$p_{i_j} = j \quad \text{dla} \quad i_j = X_{j1}, X_{j2}, \ldots, X_{jn_j}.$$

Mniej formalnie znaczy to, że daną liczbę $1 \leq j \leq m$ należy wstawić w tych miejscach wektora p, które posiadają numery:

$$X_{j1}, X_{j2}, \ldots, X_{jn_j}.$$

Algorytm taki wynika bezpośrednio z interpretacji podziałów

i permutacji / por. [99] str.93 lub [31] str. 16/.

Korzystając z tej interpretacji można łatwo skonstruować odwzorowanie odwrotne, które permutacji /9/ przyporządkowuje odpowiednio podział /10/.

Dla otrzymania podziału /10/ z permutacji /9/ należy jako wartości składowych $X_{j1}, X_{j2}, \ldots, X_{jn_j}$ wektora /10/ przyjąć kolejno numery tych składowych wektora p które są równe liczbie j.

Symbolicznie przekształcenie to zapiszemy w postaci:

$$X_{ij} = \sqrt{}(j,p) \qquad \begin{matrix} i=1,2,\ldots,m \\ j=1,2,\ldots,n_j \end{matrix}$$

funkcja $\sqrt{}$ określona była w par. 1.3.

Przykład 2.

Niech $k = (k_1, k_2, \ldots, k_m)$ oznacza kombinację z powtórzeniami z n po m /por. par. 2.5/ oraz

$$x = (x_1, x_2, \ldots, x_n)$$

oznacza partycję n elementową liczby m.

Poza tym niech dwu argumentowe operacją $\odot$ określona będzie następująco[1]:

$$a \odot b = \begin{cases} 1 & \text{jeśli } a = b \\ 0 & \text{jeśli } a \neq b \end{cases}$$

---

[1] odpowiada jej następujące wyrażenie w języku ALGOL-60 if a = b then 1 else 0.

Wówczas na podstawie kombinacji k i-tą składową partycji x otrzymamy z zależności:

$$x_i = \sum_{j=1}^{m} i \odot k_j \qquad i=1,2,\ldots,n \qquad /11/$$

Mniej formalnie oznacza to, że składowa x_i równa jest ilości tych składowych kombinacji k, które są równe indeksowi i.

Algorytm ten wynika również z interpretacji obu rozpatrywanych zbiorów /por. [99] /, która pozwala też określić odwzorowania odwrotne do /11/. Odwzorowanie to przedstawimy jako następującą konkatenację:

$$k = w_1(x_1) \| w_2(x_2) \| \ldots \| w_n(x_n) , \qquad /12/$$

gdzie symbolem $w_i(n)$ oznaczono wektor o n składowych taki, że każda jego składowa równa jest liczbie i tzn.

$$w_i(n) = (\underbrace{i,i,\ldots,i}_{n})$$

gdy $n=0$, to $w_i(n) = \emptyset$.

Nietrudno pokazać, że odwzorowanie określone zależnością /11/ posiada taką własność, że jeśli weźmiemy dwie takie kombinacje k i k', że $k \preccurlyeq k'$, to otrzymamy odpowiednio dwie partycje x i x' takie, że $x' \preccurlyeq x$.

2.7. Efektywność algorytmów generowania

Problem efektywności algorytmów jest jednym z podstawowych problemów w teorii i praktyce programowania maszyn cyfrowych, i jak dotychczas nie posiada wyczerpującego opracowania. Główną tego przyczyną jest trudność w ustaleniu kryterium efektywności. W większości wypadków jako takie kryterium przyjmuje się czas realizacji danego algorytmu na maszynie cyfrowej. Aby uniezależnić się od konkretnych typów maszyn przyjmuje się zwykle ilość operacji /np. dodawań/ niezbędnych do realizacji danego algorytmu. Ponieważ ilość wszystkich operacji niezbędnych do realizacji opisanych wcześniej algorytmów jest proporcjonalna do ilości operacji dodawań, to w paragrafie tym przykładowo przeanalizowany będzie algorytm generowania kombinacji bez powtórzeń /ze względu na ilość niezbędnych dodawań/.

Zauważmy, że idealny algorytm generowania N elementowego zbioru słów kombinatorycznych powinien potrzebować N operacji. W przypadku kombinacji bez powtórzeń zbiór składa się z C_n^k elementów, pokażemy, że do wygenerowania ich za pomocą algorytmu opisanego w par. 2.4 potrzeba $C_{n+1}^k - k - 1$ operacji dodawań.

Ilość dodawań niezbędnych do wygenerowania wszystkich kombinacji z n po k oznaczymy symbolem G(n,k). Przy wyprowadzaniu wzoru

$$G(n,k) = C_{n+1}^k - k - 1 \qquad /13/$$

korzystać będziemy z tzw. liczb simpleksowych /por. [74] /. Liczbę simpleksową, którą oznaczamy symbolem S_n^p określa się następująco

$$S_n^p = \sum_{i=1}^{n} S_i^{p-1}$$

przyjmując, że $S_n^1 = 1 + 2 + \ldots + n$, przy czym zamiast symbolu S_n^1 będzie stosowany też symbol S_n.

Zauważmy, że w analizowanym algorytmie /por. punkt 2.4.2/ do każdej współrzędnej x_i punktu[1] $x = (x_1, x_2, \ldots, x_k)$ niezależnie od pozostałych, należy dodać n-k jedynek. Poza tym dodanie jedynki do jakiejkolwiek współrzędnej x_i powoduje automatyczne dodawanie jedynek do x_j dla $j > i$. Ilość dodawań jedynek do x_1 wyraża się wzorem

$$a_1 = n - k.$$

Każde dodanie jedynki do x_1 powoduje dodanie jedynki do x_2. Przy czym pierwsze dodanie jedynki do x_1 powoduje dodanie $n - k$ jedynek do x_2 /gdyż tyle potrzeba aby $x_2 = g_2 = n - k + 2$/.

Następne dodanie jedynki do x_1 spowoduje dodanie n-k-1 jedynek do x_2 itd. W sumie dodanie n-k jedynek do x_1 spowoduje, że ilość jedynek automatycznie dodawanych do x_2 wyraża się wzorem:

$$(n-k) + (n-k-1) + \ldots + 1 + 0 = (n-k) + S_{n-k-1}.$$

---

[1] Zamiast zapisu $k = (k_1, k_2, \ldots, k_m)$ obecnie stosowany jest zapis $x = (x_1, \ldots, x_k)$.

Ponieważ niezależnie od dodawania jedynek do x_1, do x_2 dodaje się swoje n-k jedynek, to ogólna ilość dodawań do x_2 wyraża się wzorem:

$$a_2 = (n-k) + (n-k) + (n-k-1) + \ldots + 1 + 0 = 2(n-k) + S_{n-k-1}$$

Dodanie tych jedynek do x_2 spowoduje automatyczne dodawanie do x_3. Przy czym dodanie pierwszych n-k jedynek spowoduje dodanie

$$(n-k) + (n-k-1) + \ldots + 1 = S_{n-k} \text{ jedynek.}$$

Dodanie następnych n-k jedynek do x_2 spowoduje dalszych S_{n-k} dodawań do x_3. Dodanie n-k-1 jedynek do x_2 powoduje automatyczne dodanie kolejnych

$$(n-k-1) + (n-k-2) + \ldots + 1 + 0 = S_{n-k-1}$$

jedynek do x_3 itd.
Ogólnie wykonanie a_2 dodawań do x_2 spowoduje, że ilość dodawań do x_3 wyrazi się wzorem:

$$S_{n-k} + S_{n-k} + S_{n-k-1} + S_{m-k-2} + \ldots + S_1 = 2 \cdot S_{n-k} + S^2_{n-k-1}$$

Uwzględniając jeszcze n-k dodawań jakie należy wykonać do x_3 niezależnie od pozostałych dodawań otrzymujemy:

$$a_3 = n-k+2\, S^1_{n-k} + S^2_{n-k-1} = 3(n-k) + 2 \cdot S^1_{n-k-1} + 1 \cdot S^2_{n-k-1}.$$

Analogicznie rozumując dochodzimy do wniosku, że ilość dodawań jakie należy wykonać do x_i wyraża się wzorem:

$$a_i = i\,(n-k) + \sum_{j=1}^{i-1} (i-j) \cdot S^j_{n-k-1} .$$

Sumując po wszystkich i, ostatecznie otrzymujemy, że

$$G(n,k) = \sum_{i=1}^{k} i(n-k) + \sum_{i=1}^{k} \sum_{j=1}^{i-1} (i-j) \cdot S^j_{n-k-1} \qquad /14/$$

Ponieważ $S^p_n = C^{n-1}_{n+p}$ (por. [74]), to prawą stronę równości /14/ przekształcimy następująco:

$$\sum_{i=1}^{k} \sum_{j=1}^{i-1} (i-j) \cdot S^j_{n-k-1} = \sum_{i=1}^{k} \sum_{j=1}^{i-1} (i-j) \cdot C^{n-k-2}_{n-k-1+j} =$$

$$= \sum_{i=1}^{k} \left((i-1)\, C^{n-k-2}_{n-k-1+1} + \ldots + 1 \cdot C^{n-k-2}_{n-k-1+(i-1)} \right) =$$

$$= \sum_{i=1}^{k} \left((i-1) \cdot C^{n-k-2}_{n-k} + (i-2) \cdot C^{n-k-1}_{n-k+1} + \ldots + 1 \cdot C^{n-k-2}_{n-k+(i-2)} \right) =$$

$$= \sum_{i=1}^{k} \left(C^1_{i-1} \cdot C^{n-k-2}_{n-k} + C^1_{i-2} \cdot C^{n-k-2}_{n-k+1} + \ldots + C^1_1 \cdot C^{n-k-2}_{n-k+(i-2)} \right) =$$

$$= \sum_{i=1}^{k} \left(\sum_{j=0}^{i} C^1_{j+1} \cdot C^{n-k-2}_{n-k+(i-2)-j} - \left(C^1_{i+1} \cdot C^{n-k-2}_{n-k-2} + C^1_i \cdot C^{n-k-2}_{n-k-1} \right) \right) =$$

$$= \sum_{i=1}^{k} \left(\sum_{j=0}^{i} C^1_{j+1} \cdot C^{n-k-2}_{n-k+(i-2)-j} - \left((i+1) \cdot 1 + i\,(n-k-1) \right) \right) =$$

$$= \sum_{i=1}^{k} \sum_{j=0}^{i} C_{j+1}^{1} \cdot C_{n-k+(i-2)-j}^{n-k-2} - \sum_{i=1}^{k} i(n-k) - \sum_{i=1}^{k} 1$$

Ostatecznie otrzymujemy:

/15/

$$\sum_{i=1}^{k} \sum_{j=1}^{i-1} (i-j) \cdot S_{n-k-1}^{j} = \sum_{i=1}^{k} \sum_{j=0}^{i} C_{j+1}^{1} \cdot C_{n-k+(i-2)-j}^{n-k-2} - \sum_{i=1}^{k} i(n-k)-1$$

Ponieważ następująca tożsamość:

$$\sum_{j=0}^{m-q} C_{p+j}^{p} \cdot C_{m-p-j}^{q-p} = C_{m+1}^{q+1} \qquad /16/$$

jest prawdziwa (por. [99] str.57), to wzór /14/ zapiszemy w postaci:

$$G(n,k) = \sum_{i=1}^{k} i(n-k) + \sum_{i=1}^{k} C_{n-k+i}^{n-k} - \sum_{i=1}^{k} i(n-k) - k \qquad /17/$$

gdyż korzystając z /16/ mamy:

$$\sum_{j=0}^{i} C_{j+1}^{1} \cdot C_{n-k+(i-2)-j}^{n-k-2} = C_{n-k+i}^{n-k} \qquad /18/$$

Prawdziwość wzoru /18/ jest oczywista, gdyż wystarczy w równości /16/ przyjąć, że p=1, q = n-k-1 oraz m=n-k+i-1 i otrzymamy wzór /18/. Po uproszczeniu wzoru /17/ otrzymujemy:

$$G(n,k) = \sum_{i=1}^{k} C_{n-k+i}^{n-k} - k =$$

$$= C_{n-k+1}^{n-k} + C_{n-k+2}^{n-k} + \ldots + C_{n}^{n-k} - k =$$

$$= C_{n-k}^{n-k} + C_{n-k+1}^{n-k} + C_{n-k+2}^{n-k} + \ldots + C_{n}^{n-k} - C_{n-k}^{n-k} - k$$

Ostatecznie

$$G(n,k) = \sum_{i=0}^{k} C_{n-k+i}^{n-k} - C_{n-k}^{n-k} - k \qquad /19/$$

Ponieważ następująca tożsamość:

$$\sum_{i=0}^{m-1} C_{n+i}^{n} = C_{n+m}^{n+1}$$

jest prawdziwa (por. [99] str.54), to wzór /19/ zapiszemy w postaci

$$G(n,k) = C_{n+1}^{k} - 1 - k ,$$

która jest analogiczna ze wzorem /13/.

3. ZAGADNIENIA IDENTYFIKACJI

3.1. W s t ę p

3.1.1. Definicje i oznaczenia

Załóżmy, że dany jest pewien zbiór $X \subset K$. Elementami tego zbioru są m-elementowe słowa kombinatoryczne $x = (x_1, x_2, \ldots, x_m)$ nad n-elementowym alfabetem liczb całkowitych. Na zbiorze X definiować będziemy pewne funkcje o wartościach ze zbioru liczb naturalnych, które oznaczać będziemy małymi literami alfabetu greckiego. Funkcje określone na zbiorze punktów prostopadłościanu oznaczać będziemy symbolem Ψ, w szczególnym przypadku, gdy punktami są wariacje stosować będziemy symbol α. Symbolem β oznaczone będą funkcje na zbiorze kombinacji bez powtórzeń oraz symbol γ zarezerwowany jest dla funkcji określonych na zbiorze permutacji bez powtórzeń. Jeśli nie będzie zachodziła potrzeba precyzowania zbioru słów, to funkcje na nim określone oznaczać będziemy symbolem φ. Wartość pewnej funkcji $\varphi : X \to N_w$ dla określonego elementu $x \in X$ oznaczać będziemy symbolem $\varphi(x)$.

Jeżeli konieczne będzie podkreślenie, że x należy do zbioru m-elementowych słów nad n-elementowym alfabetem, to wartość funkcji φ w punkcie $x \in X$ oznaczać będziemy symbolem $\varphi(x,n,m)$.

W przypadku permutacji bez powtórzeń musi zachodzić wówczas $n=m$, stąd też zamiast $\gamma(x,n,m) = \gamma(x,n,n)$ pisać będziemy krócej $\gamma(x,n)$.

Powiemy, że funkcja różnowartościowa $\varphi : X \to N_w$ przekształcająca zbiór X na równoliczny mu zbiór N_w określa na zbiorze X relację porządku liniowego R jeśli dla dowolnych $x, x' \in X$ zachodzi warunek:

$$xRx' \Leftrightarrow \varphi(x) \leq \varphi(x')$$

Wiadomo [2,65] że w zbiorze złożonym z w elementów można określić w! różnych relacji porządku liniowego. Relacje te oznaczymy symbolami:

$$R_1, R_2, \ldots, R_{w!} ,$$

natomiast funkcje określające te porządki oznaczymy symbolami:

$$\varphi_1, \varphi_2, \ldots, \varphi_{w!} .$$

Powiemy, że funkcja φ_i n u m e r u j e elementy zbioru X według relacji R_i, zaś wartość $\varphi_i(x)$ nazwiemy n u m e r e m elementu $x \in X$.

W przypadku funkcji odwrotnej $\varphi_i^{-1} : N_w \to X$ powiemy, że i d e n t y f i k u j e albo denumeruje elementy zbioru X według ich numerów.

Wartość funkcji φ_i^{-1} w pewnym punkcie $1 \leq l \leq w!$ oznaczać będziemy $\varphi_i^{-1}(l)$. Zgodnie z definicją funkcji ω zapis

$$\omega\left(j, \varphi_i^{-1}(l)\right)$$

oznacza składową x_j wektora $x = (x_1, x_2, \ldots, x_n)$ będącego

wartością funkcji φ_i^{-1} w punkcie 1. Zamiast powyższego zapisu stosowany też będzie równoważny mu zapis

$$\varphi_{ij}^{-1}(1).$$

3.1.2. Problem

Zadanie, które tu stawiamy polega na tym, aby:

- podać sposób numerowania elementów danego zbioru $X \subset K$ według dowolnej relacji R_i oraz
- podać sposób identyfikacji elementów tego zbioru według zadanych numerów.

Symbolicznie zapiszemy to w postaci dwóch pytań:

1. $\varphi_i(x) = ?$ dla dowolnego $x \in X$
2. $\varphi_i^{-1}(j) = ?$ dla dowolnego $j \in N_w$

gdzie i jest zadaną liczbą $1 \leqslant i \leqslant w!$ oraz $w = |X|$.

3.1.3. Rozwiązanie

Przyjmijmy na chwilę, że dla dowolnego zbioru $X \subset K$ znana jest funkcja określająca porządek leksykograficzny zdefiniowany w rozdziale 1. Funkcje numerujące według tego porządku oznaczać będziemy symbolem φ_1, tzn. przypisujemy im indeks równy jeden.

Przy tych założeniach formułujemy odpowiedzi na dwa postawione wyżej pytania:

1. $\varphi_i(x) = \gamma(\varphi_1(x), \gamma_1^{-1}(i,w))$ /1/

2. $\varphi_i^{-1}(j) = \varphi_1^{-1}(\omega(j, \gamma_1^{-1}(i,w)))$ /2/

3.1.4. Uzasadnienie

Aby podane rozwiązanie /1/ i /2/ było poprawne należy pokazać, że:

- wszystkie funkcje $\varphi_1, \varphi_2, \ldots, \varphi_{w!}$ są różne, oraz
- każda z tych funkcji rzeczywiście numeruje elementy zbioru X tzn., że każda φ_i jest różnowartościowa i przekształca X na N_w.

Pokażemy najpierw, że wszystkie funkcje dla $i=1,2,\ldots,w!$ są różne tzn., że

$$i \neq j \Rightarrow \varphi_i(x) \neq \varphi_j(x) \qquad \text{dla } x \in X.$$

Niech $\gamma_i^{-1}(i,w)$ będzie permutacją

$$p = (p_1, p_2, \ldots, p_w),$$

natomiast $\gamma_i^{-1}(j,w)$ permutacją

$$p' = (p'_1, p'_2, \ldots, p'_w).$$

Z założenia różnowartościowości funkcji γ_i^{-1} wynika więc, że dla $i \neq j$ prawdziwa jest nierówność:

$$p \neq p' ,$$

oznacza to, że istnieje takie $1 \leq s \leq w$ oraz takie $1 \leq s' \leq w$, że $s \neq s'$, dla których zachodzi równość:

$$p_s = p'_s , \quad .$$

Poza tym funkcja φ_1 z założenia spełnia nierówność

$$1 \leq \varphi_1(x) \leq w \qquad \text{dla } x \in X$$

oznacza to, że istnieje takie $x \in X$, że

$$\varphi_1(x) = p_s$$

Dla $x \in X$ spełniającego tę równość mamy

$$\varphi_i(x) = \gamma(\varphi_1(x), \gamma_1^{-1}(i,w)) = s$$

oraz

$$\varphi_j(x) = \gamma(\varphi_1(x), \gamma_1^{-1}(j,w)) = s' .$$

A więc jeśli $i \neq j$, to w zbiorze X istnieje takie $x \in X$, że

$$\varphi_i(x) \neq \varphi_j(x)$$

tzn., że wszystkie funkcje φ_i są różne /różnią się przynajmniej w jednym punkcie $x \in X$/.

Pokażemy teraz, że funkcje określane wzorem /1/ są różnowartościowe tzn., że

$$x \neq x' \Rightarrow \varphi_i(x) \neq \varphi_i(x') .$$

Różnowartościowość ta wynika z założonej różnowartościowości funkcji φ_1, mamy bowiem

$$x \neq x' \Rightarrow \varphi_1(x) \neq \varphi_1(x') \Rightarrow \vee(\varphi_1(x), \gamma_1^{-1}(i,w)) \neq \vee(\varphi_1(x'), \gamma_1^{-1}(i,w)$$

Nietrudno zauważyć, że

$$\min \varphi_i(x) = 1$$

gdyż $\gamma_1^{-1}(i,w)$ jest pewną permutacją w-elementową

$$p = (p_1, p_2, \ldots, p_w)$$

taką, że $1 \leq p_s \leq w$,

natomiast φ_1 z założenia jest taką funkcją, że $1 \leq \varphi_1(x) \leq w$

tzn. w permutacji p zawsze znajdzie się taka składowa p_s, która jest równa liczbie $\varphi_1(x)$, a indeks tej składowej jest wartością funkcji φ_i w punkcie $x \in X$. Jeśli przy tym dane $x \in X$ jest takie, że $\varphi_1(x) = p_1$, to $\varphi_i(x) = 1$, jeśli zaś $\varphi_1(x) = p_w$, to $\varphi_i(x) = w$, a więc

$$\max \varphi_i(x) = w$$

Tak więc funkcja φ_i / i = 1,2,...,w!/ określona wzorem /1/ jest funkcją różnowartościową przekształcającą X na N_w

tzn., że funkcja ta numeruje elementy zbioru X zgodnie z relacją R_i.

Pokażemy teraz, że funkcja odwrotna do φ_i wyraża się wzorem /2/, tzn., że dla każdego $x \in X$ zachodzi

$$\varphi_i^{-1}(\varphi_i(x)) = x$$

Zauważmy, że jeśli $\gamma(c,x) < \infty$, to

$$\omega(\gamma(c,x), x) = c$$

Stąd też dla każdego $x \in X$ otrzymujemy:

$$\varphi_i^{-1}(\varphi_i(x)) = \varphi_1^{-1}(\omega(\varphi_i(x), \gamma_1^{-1}(i,w))) =$$

$$= \varphi_1^{-1}(\omega(\gamma(\varphi_1(x), \gamma_1^{-1}(i,w)), \gamma_1^{-1}(i,w))) =$$

$$= \varphi_1^{-1}(\varphi_1(x)) = x.$$

W ten sposób pokazaliśmy, że φ_i numeruje elementy zbioru X a funkcja φ_i^{-1} identyfikuje je.

3.1.5. Wnioski

Pokazaliśmy, że jeśli znana jest funkcja określająca porządek leksykograficzny, to na podstawie wzorów /1/ i /2/ można otrzymać wszystkie pozostałe funkcje.
Dlatego też w dalszej części zajmiemy się wyłącznie definiowaniem funkcji określających porządek leksykograficzny. Funkcje takie podamy dla tych podzbiorów zbioru K, które najczęściej występują w zastosowaniach, a więc dla wariacji, permutacji, kombinacji itp. Przestrzegać przy tym będziemy następujących zasad.

Ponieważ dla wybranych zbiorów rozpatrywać będziemy tylko funkcje określające porządek leksykograficzny, tzn. takie, którym przypisany jest indeks równy jedności, to będziemy go opuszczali.

Tak więc zamiast φ_1, φ_1^{-1}, φ_{1i}^{-1} będziemy odpowiednio pisali φ, φ^{-1} oraz φ_i^{-1}.

Poza tym w dalszej części niniejszego rozdziału przestrzegane będą następujące zasady.

Dla każdego spośród wybranych zbiorów kombinatorycznych $X \subset K$ określana będzie funkcja $\varphi: X \to \{1, 2, \ldots, |X|\}$ oraz podana do niej odwrotna.

Po określeniu tych dwóch funkcji pokażemy, że φ numeruje elementy X zgodnie z porządkiem leksykograficznym, natomiast funkcja φ^{-1} identyfikuje te elementy.
O funkcji φ wykażemy kolejno, że

1. $x \prec x' \Rightarrow \varphi(x) < \varphi(x')$,

2. $\min \varphi(x) = 1$,

3. $\max \varphi(x) = w$, gdzie w - liczebność zbioru X.

W ten sposób zostanie pokazane, że funkcja φ jest

- różnowartościowa,
- przekształca X na równoliczny z nim zbiór $N_w = \{1, 2, \ldots, w\}$,
- spełnia warunek (por. [83]):

$$x \preccurlyeq x' \Leftrightarrow \varphi(x) \leqslant \varphi(x') \qquad \text{dla } x, x' \in X,$$

a więc zostanie pokazane, że funkcja φ określa w zbiorze X porządek leksykograficzny.

Wymienione wyżej trzy własności oznaczane będą dalej symbolem WI.

Funkcję odwrotną φ^{-1} będziemy konstruować bezpośrednio z funkcji φ albo wykażemy o niej, że dla dowolnego k ze zbioru określoności funkcji φ zachodzi:

$$\varphi^{-1}(\varphi(k)) = k .$$

Ponieważ często pojawiać się będą sumy postaci $\sum_{i=a}^{b} x_i$ takie, że $a > b$, to aby nie powtarzać założeń za każdym razem, przyjmujemy tu generalnie, że sumy takie są równe zeru. Poza tym jeśli nie będą czynione żadne inne założenia, to symbol $[x]$ zawsze oznaczać będzie największą liczbę całkowitą nie większą niż x.

3.2. P r o s t o p a d ł o ś c i a n

Załóżmy, że dany jest pewien prostopadłościan P tzn. zbiór takich punktów $k = (k_1, k_2, \ldots, k_m)$, które dla danych wektorów całkowitoliczbowych $d = (d_1, d_2, \ldots, d_m)$ i $g = (g_1, g_2, \ldots, g_m)$ spełniają warunek: $d_i \leqslant k_i \leqslant g_i$.

Niech $l_i = g_i - d_i + 1$ oraz $w = |P| = \prod_{i=1}^{m} l_i$.

Funkcję $\Psi : P \rightarrow N_w$ numerującą zgodnie z porządkiem leksykograficznym punkty prostopadłościanu P wyrazimy w postaci następującego wzoru:

$$l = \Psi(k) = 1 + \sum_{i=1}^{m} (k_i - d_i)\, z_i \qquad /3/$$

gdzie:

$$z_m = 1 \qquad /3a/$$

oraz

$$z_i = \prod_{j=i+1}^{m} l_j \qquad \text{dla } i=0,1,\ldots,m-1$$

lub rekurencyjnie

$$z_i = l_{i+1} \cdot z_{i+1} \qquad \text{dla } i=0,1,\ldots,m-1 \qquad /3b/$$

Funkcję identyfikującą punkty prostopadłościanu P tzn. funkcję odwrotną do Ψ określimy w postaci dwóch różnych wzorów.

Wzór 1.

Za pomocą tego wzoru dla dowolnego numeru $1 \leqslant j \leqslant w$ składowe odpowiadającego mu punktu $k = (k_1, k_2, \ldots, k_m)$ wy-

znacza się począwszy od składowej k_m a skończywszy na składowej k_1. Wzór jest następujący:

$$k_{m-i+1} = \text{rest}\left(C_{m-i+1}/l_{m-i+1}\right) + d_{m-i+1} \quad \text{dla } i=1,2,\ldots,m \qquad /4/$$

gdzie

$$C_i = \left[\frac{C_{i+1}}{l_{i+1}} \right] \qquad \text{dla } i=1,2,\ldots,m-1$$

$$C_m = j-1$$

symbolem $\text{rest}(a/b)$ oznaczono resztę z dzielenia a przez b

Wzór 2.

Zależność, która służy do wyznaczenia kolejnych składowych punktu $k = (k_1, k_2, \ldots, k_m)$ począwszy od k_1 a kończywszy na k_m ma następującą postać:

$$k_i = d_i + \left[\frac{j - 1 - s_{i-1}}{z_i} \right] \qquad \text{dla } i=1,2,\ldots,m \qquad /5/$$

gdzie $s_i = s_{i-1} + \left(k_i - d_i \right) \cdot z_i$

$s_o = 0$

z_i określone są wzorem /3a/

j jest danym numerem $1 \leqslant j \leqslant w$

O funkcji Ψ wykażemy, że ma własności WL.

Weźmy dwa dowolne punkty $k, k' \in P$ takie, aby $k \prec k'$. Pokażemy, że prawdziwa jest wówczas następująca nierówność:

$$\Psi(k) < \Psi(k')$$

Korzystając z definicji /3/ nierówność tę zapiszemy w postaci:

$$1 + \sum_{i=1}^{m} (k_i - d_i) z_i < 1 + \sum_{i=1}^{m} (k'_i - d_i) z_i \qquad /6/$$

Niech dla pewnego $1 \leqslant s \leqslant m$ zachodzi

$$k_s < k'_s \quad \text{oraz} \quad k_j = k'_j \quad \text{dla} \quad j < s$$

(jest to tylko sprecyzowanie założenia, że $k \prec k'$).
Nierówność /6/ zapiszemy wówczas w postaci:

$$\sum_{i=s}^{m} (k_i - d_i) z_i < \sum_{i=s}^{m} (k'_i - d_i) z_i ,$$

jest ona równoważna nierówności:

$$\sum_{i=s}^{m} (k_i - k'_i) z_i < \sum_{i=s}^{m} (d_i - d_i) z_i = 0 ,$$

która z kolei jest równoważna nierówności:

$$\sum_{i=s+1}^{m} (k_i - k'_i) z_i < (k'_s - k_s) z_s \qquad /7/$$

Jeśli przyjmiemy, że składowe $k'_{s+1}, k'_{s+2}, \ldots, k'_m$ są równe swoim minimalnym wartościom tzn. $k'_i = d_i$ dla $i > s$, oraz jeśli przyjmiemy, że $k_s = k_s + 1$, to nierówność możemy tylko pogorszyć. Jeszcze bardziej ją pogorszymy jeśli przyjmiemy, że składowe $k_s, k_{s+1}, \ldots, k_m$ są równe swoim maksymalnym wartościom tzn., że $k_i = g_i = d_i + l_i - 1$.

Powyższe założenia znaczą tyle, że przyjmujemy takie dwa punkty k i k', że k' znajduje się bezpośrednio po k w uporządkowaniu leksykograficznym, tzn. k' = k + 1.
Przy takich założeniach nierówność /7/ przyjmie postać:

$$\sum_{i=s+1}^{m} \left(d_i + l_i - 1 - d_i\right) z_i < \left(k_s + 1 - k_s\right) z_s$$

po uproszczeniach otrzymujemy

$$\sum_{i=s+1}^{m} \left(l_i \cdot z_i - z_i\right) < z_s$$

Korzystając z /3b/ iloczyn $l_i \cdot z_i$ występujący w tej nierówności możemy zamienić liczbą z_{i-1}, wówczas lewą stronę tej nierówności przekształcimy następująco:

$$\sum_{i=s+1}^{m} \left(l_i \cdot z_i - z_i\right) = \sum_{i=s+1}^{m} \left(z_{i-1} - z_i\right) =$$

$$= \left(z_s - z_{s+1}\right) + \left(z_{s+1} - z_{s+2}\right) + \ldots + \left(z_{m-1} - z_{m-i}\right) + \left(z_{m-1} - z_m\right) =$$

$$= z_s - z_m = z_s - 1 \qquad /8/$$

Otrzymujemy więc oczywistą nierówność

$$z_s - 1 < z_s$$

W ten sposób pokazaliśmy, że funkcja określana wzorem /3/ spełnia warunek

$$k \prec k' \Rightarrow \Psi(k) < \Psi(k')$$

tzn. jest ściśle rosnąca.

Pokażemy teraz, że $\min \Psi(k) = 1$ oraz $\max \Psi(k) = w$. Zauważmy, że minimalną wartość funkcja Ψ przyjmuje wówczas, gdy punkt k jest najmniejszy tzn., gdy $k_i = d_i$ wówczas otrzymujemy $\min \Psi(k) = 1 + \sum_{i=1}^{m} (k_i - d_i) z_i =$

$$= 1 + \sum_{i=1}^{m} (d_i - d_i) z_i = 1.$$

Natomiast maksymalną wartość przyjmuje gdy k jest punktem największym tzn., gdy $k_i = g_i = d_i + l_i - 1$, wówczas otrzymujemy:

$$\max \Psi(k) = 1 + \sum_{i=1}^{m} (d_i + l_i - 1 - d_i) z_i =$$

$$= 1 + \sum_{i=1}^{m} (l_i \cdot z_i - z_i) = 1 + z_o - 1 = z_o = w$$

gdyż podobnie jak w /8/ $\sum_{i=1}^{m} (l_i \cdot z_i - z_i) = z_o - 1$

oraz zgodnie z definicją /3b/ mamy $z_o = l_1 \cdot l_2 \cdot \ldots \cdot l_m = w$

Pokazaliśmy więc, że funkcja określona wzorem /3/ spełnia wszystkie warunki funkcji ustalającej porządek leksykograficzny.

Przed wyprowadzeniem funkcji odwrotnej do Ψ, wykażemy prawdziwość następujących dwóch nierówności:

$$0 \leqslant \frac{k_i - d_i}{l_i} < 1. \qquad /9/$$

$$0 \leqslant \frac{\sum_{i=s}^{m} (k_i - d_i) z_i}{z_{s-1}} < 1 \qquad /10/$$

Ponieważ $l_i > 0$ oraz $t_i \geqslant d_i \Rightarrow t_i - d_i \geqslant 0$, to nierówność

$$0 \leqslant \frac{k_i - d_i}{l_i}$$ jest prawdziwa.

Ponieważ $l_i > 0$, to nierówność

$$\frac{k_i - d_i}{l_i} < 1$$

można zapisać w postaci $k_i - d_i < l_i$.

Podstawiając za l_i jego definicję tzn. $l_i = g_i - d_i + 1$ otrzymujemy

$$k_i - d_i < g_i - d_i + 1$$

Upraszczając przez d_i otrzymujemy oczywistą nierówność $k_i < g_i + 1$, gdyż maksymalną wartością jaką może przyjąć k_i jest g_i.

W ten sposób pokazaliśmy prawdziwość nierówności /9/.

Przejdźmy teraz do nierówności /10/.

Ponieważ $z_{s-1} > 0$ oraz $\sum_{i=s}^{m} (k_i - d_i) z_i \geqslant 0$, to ułamek

występujący w wyrażeniu /10/ jest nieujemny.

Pokażemy, że jest on również mniejszy od 1.

Ponieważ $z_{s-1} > 0$, to do pokazania mamy nierówność:

$$\sum_{i=s}^{m} (k_i - d_i) z_i < z_{s-1} \qquad /11/$$

Jeśli przyjmiemy, że $k_i = g_i = d_i + l_i - 1$, to nierówność /11/ możemy tylko pogorszyć. Przyjmując to założenie oraz uwzględniając równość /8/ lewą stronę nierówności /11/ przekształcamy następująco:

$$\sum_{i=s}^{m} (k_i - d_i) z_i = \sum_{i=s}^{m} (l_i - 1) z_i = \sum_{i=s}^{m} (l_i \cdot z_i - z_i) = z_{s-1} - 1$$

otrzymujemy więc oczywistą nierówność

$$z_{s-1} - 1 < z_{s-1}$$

Tym samym pokazaliśmy prawdziwość /10/.

Wyprowadźmy teraz wzór /4/ określający funkcję identyfikującą punkty prostopadłościanu według ich numerów. Wartość funkcji $\Psi(k)$ zapiszemy w postaci:

$$j = \Psi(k) = 1 + \sum_{i=1}^{m} (k_i - d_i) z_i =$$

$$= 1 + (k_1 - d_1) l_2 \cdot l_3 \cdots l_m + (k_2 - d_2) l_3 l_4 \cdots l_m + \cdots +$$

$$+ (k_{m-1} - d_{m-1}) l_m + (k_m - d_m)$$

Odejmijmy stronami 1 oraz podzielmy przez l_m:

$$\frac{j-1}{l_m} = (k_1 - d_1) l_2 l_3 \cdots l_{m-1} + (k_2 - d_2) l_2 l_4 \cdots l_{m-1} + \cdots +$$

$$+ (k_{m-1} - d_{m-1}) + \frac{k_m - d_m}{l_m}$$

Otrzymaliśmy więc część całkowitą:

$$C_{m-1} = (k_1 - d_1) l_2 l_3 \cdots l_{m-1} + \cdots + (k_{m-1} - d_{m-1})$$

oraz resztę:

$$r_m = k_m - d_m$$

ponieważ zgodnie z /9/ zachodzi nierówność

$$0 \leqslant \frac{k_m - d_m}{l_m} < 1.$$

Dodając do otrzymanej reszty r_m liczbę d_m w sposób jednoznaczny otrzymujemy składową k_m tzn.

$$k_m = r_m + d_m = \text{rest}\left(\frac{j-1}{l_m}\right) + d_m$$

Jeżeli dalej podzielimy C_{m-1} przez l_{m-1}, to otrzymamy część całkowitą:

$$C_{m-2} = (k_1 - d_1) l_2 l_3 \cdots l_{m-2} + \cdots + (k_{m-2} - d_{m-2})$$

oraz resztę

$$r_{m-1} = k_{m-1} - d_{m-1}$$

Stąd

$$k_{m-1} = \text{rest}\left(\frac{C_{m-2}}{l_{m-1}}\right) + d_{m-1}$$

Powtarzając postępowanie takie n razy, wyznaczymy kolejno wszystkie składowe $k_m, k_{m-1}, \ldots, k_1$.

Przyjmijmy, że $C_m = j - 1$, wówczas postępowanie powyższe zapisujemy w postaci zależności /4/.

Wyprowadźmy teraz inną postać funkcji odwrotnej tzn. tę, która wyrażona jest wzorem /5/.

Wartość funkcji Ψ w punkcie $k = (k_1, k_2, \ldots, k_m)$ zapiszemy teraz w postaci:

$$j = 1 + (k_1 - d_1)z_1 + (k_2 - d_2)z_2 + \ldots + (k_m - d_m)z_m \qquad /12/$$

Odejmując stronami 1 oraz dzieląc przez z_1 otrzymujemy:

$$\frac{j-1}{z_1} = k_1 - d_1 + \frac{(k_2 - d_2)z_1 + \ldots + (k_m - d_m)z_m}{z_1}$$

tzn. otrzymujemy część całkowitą

$$C_1 = k_1 - d_1$$

oraz zgodnie z wzorem /10/ ułamek właściwy.

Stąd też składową k_1 jednoznacznie obliczamy jako:

$$k_1 = d_1 + C_1 = d_1 + \left[\frac{j - 1}{z_1} \right]$$

Zapiszmy teraz równość /12/ w postaci:

$$j - 1 - (k_1 - d_1)z_1 = (k_2 - d_2)z_2 + \ldots + (k_m - d_m)z_m$$

dzieląc ją stronami przez z_2 otrzymujemy część całkowitą:

$$C_2 = k_2 - d_2$$

oraz ułamek właściwy:

$$\frac{(k_3 - d_3)z_3 + \ldots + (k_m - d_m)z_m}{z_2}$$

Tak więc składową k_2 obliczamy ze wzoru:

$$k_2 = d_2 + c_2 = d_2 + \left[\frac{j - 1 - (k_1 - d_1)z_1}{z_2} \right]$$

Analogiczne rozumowanie prowadzi do wzoru:

$$k_i = d_i + \left[\frac{j - 1 - \sum_{s=1}^{i-1} (k_s - d_s)z_s}{z_i} \right] \qquad /13/$$

Sumę występującą pod znakiem funkcji entier można obliczać rekorencyjnie:

$$S_0 = 0$$

$$S_i = S_{i-1} + (k_i - d_i)z_i \qquad \text{dla } i=1,2,\ldots,m$$

wówczas wzór /13/ przyjmuje postać wzoru /5/.

Zauważmy, że $k_m = d_m + \left[\frac{j - 1 - S_{m-1}}{z_m} \right] = d_m + \left[\frac{j - 1 - S_{m-1}}{1} \right] =$

$= d_m + j - 1 - S_{m-1}$ co można wykorzystać w celu przyspieszenia obliczeń na maszynie.

Uwaga:

W przypadku, gdy $d_i = 1$ oraz $g_i = n$ dla $i=1,2,\ldots,m$ punkty prostopadłościanu są wariacjami z powtórzeniami, wówczas wzory /3/, /4/ i /5/ odpowiednio się uproszczą. Funkcje dla tego szczególnego przypadku zgodnie z założeniem paragrafu 3.1, oznaczać będziemy symbolem α .

3.3. K o m b i n a c j e b e z p o w t ó r z e ń

Niech $K_b = \left\{(k_1,k_2,\ldots,k_m) \;\forall\; i < j \Rightarrow k_i < k_j,\quad 1 \leqslant k_i \leqslant n\right\}$

oraz $w = |K_b|$.

Funkcję $\beta : K_b \rightarrow N_w$ numerującą kombinacje bez powtórzeń z n po m według porządku leksykograficznego wyrazimy w postaci wzoru:

$$l = \beta(k) = 1 + \sum_{i=p}^{m} \left(C_{n-k_{i-1}}^{m-i+1} - C_{n-k_i+1}^{m-i+1} \right) \qquad /14/$$

przyjmując $k_j = j$ dla $j < p$

gdzie $1 \leqslant p \leqslant m$

Jeżeli p=1, to funkcja /14/ numeruje wszystkie kombinacje z n po m, jeśli zaś $p > 1$ wówczas numeruje tylko takie kombinacje, w których pierwsze p-1 składowych jest ustalone: $k_1 = 1,\ k_2 = 2,\ldots,\ k_{p-1} = p-1$.

Funkcja odwrotna do β, która każdemu numerowi $1 \leqslant l \leqslant C_n^m$ jednoznacznie przyporządkowuje odpowiadającą mu kombinację $k = (k_1,k_2,\ldots,k_m)$ określona jest wzorem:

$$k_i = \beta_i^{-1}(l) = k_{i-1} + \vartheta_i \qquad /15/$$

przyjmując, że $k_0 = 0$,

oraz

$$\vartheta_i = (\mu z)\left[\sum_{j=1}^{z} C_{n-k_{i-1}-j}^{m-i} \geqslant l - \sum_{j=1}^{i-1} \sum_{s=1}^{\vartheta_j-1} C_{n-k_{j-1}-s}^{m-j}\right] \qquad /15a/$$

Dla funkcji β^{-1} przyjęto p=1.

Symbolem $(\mu z)\left[R(z)\right]$ oznaczona jest operacja minimum [29], która oznacza najmniejszą liczbę całkowitą z, dla której spełniony jest warunek R(z).

Wykażemy, że funkcja β ma własności WL.

Zauważmy na wstępie, że ilość takich kombinacji, gdy ustalone są składowe $k_1 = 1, k_2 = 2, \ldots, k_{p-1} = p-1$ wyraża się wzorem:

$$C_{n-p+1}^{m-p+1}$$

ponieważ p-1 składowych jest ustalona.

Jeśli p=1 to ilość kombinacji jest równa C_n^m.

Weźmy dwie kombinacje k i k', takie aby spełniony był warunek $k \prec k'$. Niech to będą takie kombinacje, że $k' = k+1$ tzn.

$$k'_s = k_s + 1 \qquad \text{dla pewnego } p \leqslant s \leqslant m \qquad /16a/$$

$$k_i = n - m + i \qquad \text{dla } i = s+1, s+2, \ldots, m \qquad /16b/$$

$$k'_i = i \qquad \text{dla } i = s+1, s+2, \ldots, m \qquad /16c/$$

$$k_i = k'_i \qquad \text{dla } i = 1, 2, \ldots, s-1 \qquad /16d/$$

Pokażemy, że zachodzi wówczas nierówność

$$1 + \sum_{i=p}^{m} \left(C_{n-k_{i-1}}^{m-i+1} - C_{n+1-k_i}^{m-i+1} \right) < 1 + \sum_{i=p}^{m} \left(C_{n-k'_{i-1}}^{m-i+1} - C_{n+1-k'_i}^{m-i+1} \right)$$

Korzystając z /16d/ możemy odrzucić składniki dla i=p, p+1, ..., s-1 z obu stron podanej nierówności. Poza tym korzystając z /16b/ i /16c/ zamiast k_i podstawimy n-m+i

oraz zamiast k_i^1 podstawimy i dla $i=s+1, s+2, \ldots, m$,
co powoduje wyzerowanie się składników sumy o tych indeksach, dlatego też powyższa nierówność jest równoważna nierówności:

$$C_{n-k_{s-1}}^{m-s+1} - C_{n+1-k_s}^{m-s+1} < C_{n-k_{s-1}^2}^{m-s+1} - C_{n+1-k_s^2}^{m-s+1}$$

Zapiszemy ją w postaci:

$$C_{n-k_{s-1}}^{m-s+1} - C_{n-k_{s-1}^2}^{m-s+1} < C_{n+1-k_s}^{m-s+1} - C_{n+1-(k_s+1)}^{m-s+1}$$

Skorzystaliśmy tu też z /16a/ zamieniając k_s^2 na k_s+1.
Zgodnie z /16d/ lewa strona tej nierówności jest równa zeru, natomiast prawa strona jest równa $C_{n-k_s}^{m-s}$, wynika to z tożsamości [99]:

$$C_{b-1}^{a-1} = C_b^a - C_{b-1}^a \qquad /17/$$

Wystarczy bowiem przyjąć $a = m - s + 1$ oraz $b = n+1-k_s$.
Tak więc ostatnia nierówność przyjmuje postać

$$0 < C_{n-k_s}^{m-s} \qquad /18/$$

Zauważmy, że $n-k_s \geqslant m-s$ jest prawdziwa, gdyż podstawiając za k_s jego maksymalną wartość otrzymujemy $n-(n-m+s) \geqslant m-s$
Poza tym ponieważ $s \leqslant m$ oraz $k_s \leqslant n$, nierówność /18/ jest prawdziwa. Pokazaliśmy więc, że funkcja określona wzorem /14/ jest ściśle rosnąca.
Stąd też wynika, że najmniejszą wartość osiąga wówczas, gdy jej argumentem jest kombinacja najmniejsza tzn. gdy

$$k_i = i \qquad \text{dla } i=1,2,\ldots,m$$

Otrzymujemy wówczas:

$$\min \beta(k) = 1 + \sum_{i=p}^{m} \left(C_{n-(i-1)}^{m-i+1} - C_{n-i+1}^{m-i+1} \right) = 1 + 0 = 1$$

Natomiast wartość maksymalną osiąga dla kombinacji największej tzn. gdy

$$k_i = n - m + i \qquad \text{dla } i=p,p+1,\ldots,m$$

Podstawiając te wartości do wyrażenia /14/ i wyłączając pierwszy składnik poza znak sumy otrzymujemy:

$$\max \beta(k) = 1 + C_{n-k_{p-1}}^{m-p+1} - C_{n+1-(n-m+p)}^{m-p+1} +$$

$$+ \sum_{i=p+1}^{m} \left(C_{n-(n-m+i-1)}^{m-i+1} - C_{n-(n-m+i-0)+1}^{m-i+1} \right) =$$

$$= 1 + C_{n-p+1}^{m-p+1} - C_{m-p+1}^{m-p+1} + 0 =$$

$$= 1 + C_{n-p+1}^{m-p+1} - 1 + 0 = C_{n-p+1}^{m-p+1} \qquad /19/$$

Pokazaliśmy więc, że funkcja określona wzorem /14/ przyjmuje kolejno wartości $1,2,\ldots,C_{n-p+1}^{m-p+1}$.

Przejdźmy teraz do funkcji odwrotnej.

Mamy pokazać, że zachodzi równość:

$$\beta^{-1}(\beta(k)) = k$$

tzn., że

$$\beta_i^{-1}(1) = k_i \qquad \text{dla } i=1,2,\ldots,m$$

Ponieważ $\beta_i^{-1}(1) = k_{i-1} + \vartheta_i$ - to musimy pokazać, że jeżeli zamiast 1 wstawimy prawą stronę wzoru /14/ to wynikiem operacji minimum /15a/ określającej ϑ_i będzie $k_i - k_{i-1}$. Wówczas otrzymamy:

$$k_i = \beta_i^{-1}(1) = k_{i-1} + k_i - k_{i-1} = k_i.$$

Zauważmy, że liczba całkowita t będąda wynikiem operacji minimum:

$$(\mu z)\left[L(z) \geqslant P\right]$$

musi spełniać jednocześnie dwie nierówności:

$$L(t) \geqslant P$$

oraz $$L(t-1) < P$$

Dlatego też aby pokazać, że $k_i - k_{i-1}$ jest wynikiem operacji minimum /15a/ musimy wykazać prawdziwość następujących nierówności.

$$\sum_{j=1}^{w_i} C_{n-j-k_{i-1}}^{m-i} \geqslant 1 - \sum_{j=1}^{i-1} \sum_{s=1}^{w_j-1} C_{n-s-k_{j-1}}^{m-j} \qquad /20/$$

$$\sum_{j=1}^{w_i-1} C_{n-j-k_{i-1}}^{m-1} < 1 - \sum_{j=1}^{i-1} \sum_{s=1}^{w_j-1} C_{n-s-k_{j-1}}^{m-j} \qquad /21/$$

gdzie 1 określone jest wzorem /14/,

oraz $w_i = k_i - k_{i-1}$.

W celu uproszczenia dowodu nierówności /20/ i /21/ pokażemy, że

$$\sum_{i=a}^{b} C_{n-i}^{m} = C_{n-a+1}^{m+1} - C_{n-b}^{m+1} \qquad /22/$$

przyjmując, że $C_n^m = 0$ gdy $n < m$.

Korzystając z tożsamości /17/ każdy składnik sumy równości /22/ zapiszemy w postaci różnicy dwóch składowych. Wówczas otrzymamy:

$$\sum_{i=a}^{b} C_{n-i}^{m} = \sum_{i=a}^{b} \left(C_{n-i+1}^{m+1} - C_{n-i}^{m+1} \right) =$$

$$= C_{n-a+1}^{m+1} - C_{n-a}^{m+1} + C_{n-a+1-1}^{m+1} - C_{n-a-1}^{m+1} + \ldots +$$

$$= C_{n-b+2}^{m+1} - C_{n-b+1}^{m+1} + C_{n-b+1}^{m+1} - C_{n-b}^{m+1}$$

Po uproszczeniach otrzymujemy wzór /22/

Korzystając z tego wzoru oraz podstawiając za 1 prawą stronę /14/, nierówność /20/ zapiszemy w postaci:

$$C_{n-k_{i-1}}^{m-i+1} - C_{n-k_{i-1}-(k_i-k_{i-1})}^{m-i+1} \geqslant 1 + \sum_{j=1}^{m} \left(C_{n-k_j-1}^{m-j+1} - C_{n-k_j+1}^{m-j+1} \right) -$$

$$- \sum_{j=1}^{i-1} \left(C_{n-k_{j-1}}^{m-j+1} - C_{n-k_{j-1}-(k_j-k_{j-1})+1}^{m-j+1} \right)$$

Po uproszczeniach otrzymujemy:

$$C_{n-k_{i-1}}^{m-i+1} - C_{n-k_i}^{m-i+1} \geqslant 1 + \sum_{j=i}^{m} \left(C_{m-k_{j-1}}^{m-j+1} - C_{n-k_{j+1}}^{n-j+1} \right)$$

Przenosząc pierwszy składnik sumy z prawej strony na lewą i dokonując uproszczenia otrzymujemy:

$$C_{n-k_i+1}^{m-i+1} - C_{n-k_i}^{m-i+1} \geqslant 1 + \sum_{j=i+1}^{m} \left(C_{n-k_{j-1}}^{m-j+1} - C_{n-k_{j+1}}^{m-j+1} \right) \qquad /23/$$

Nierówność /23/ będzie prawdziwa jeśli wykażemy, że zachodzi ona w przypadku, gdy jej lewa strona przyjmie wartość minimalną.

Minimalną wartość przyjmuje ona wówczas, gdy k_i przyjmuje swoją maksymalną wartość, tzn. $k_i = n-m+i$. Warunek $i < j \Rightarrow k_i < k_j$ występujący w definicji kombinacji pociąga wówczas $k_j = n-m+j$ dla $j = i+1, i+2, \ldots, m$ tzn. składowe k_j dla $j > i$ również przyjmują maksymalne wartości. Jeśli w nierówności /23/ zamiast k_i przyjmiemy $n-m+i$ oraz zamiast k_j przyjmiemy $n-m+j$ dla $j=i+1, \ldots, m$ to otrzymamy oczywistą nierówność: $1 \geqslant 1$.

W ten sposób dowiedliśmy prawdziwości /20/.

Aby wykazać prawdziwość /21/ podobnie jak przy nierówności /20/, stosujemy wzór /22/, za 1 podstawiamy jego definicję, dokonujemy niezbędnych uproszczeń i otrzymujemy:

$$C_{n-k_i+1}^{n-i+1} - C_{n-k_i+1}^{n-i+1} < 1 + \sum_{j=i+1}^{m} \left(C_{n-k_{j-1}}^{m-j+1} - C_{n-k_j+1}^{m-j+1} \right)$$

Ponieważ suma prawej strony tej nierówności zawsze jest

nieujemna, to nierówność ta jest spełniona, gdyż w najgorszym przypadku, tzn. gdy $k_i = k_{i-1}+1$ otrzymamy oczywistą nierówność

$$0 < 1 + 0$$

Obie nierówności /20/ i /21/ są więc prawdziwe, a to dowodzi równości:

$$\beta^{-1}(\beta(k)) = k.$$

3.4. K o m b i n a c j e z p o w t ó r z e n i a m i

Niech $K_p = \{(k_1,k_2,\ldots,k_m) : i < j \Rightarrow k_i \leq k_j,\ 1 \leq k_i \leq n\}$

oraz $w = |K_p| = C^m_{n+m-1}$.

Przyjmijmy też, że

$$r_i = n+m-i+1-k_{i-1} \qquad \text{dla } i = 1,2,\ldots,m$$

$$u_i = k_i - k_{i-1} \qquad \text{dla } i = 1,2,\ldots,m$$

$$k_0 = 1.$$

Funkcję $\varphi : K_p \rightarrow N_w$ numerującą kombinacje z powtórzeniami określamy wzorem:

$$l = \varphi(k) = 1 + \sum_{i=1}^{m} \sum_{j=i}^{u_i} C^{m-i}_{r_i-j} \qquad /24/$$

Natomiast funkcja do niej odwrotna określona jest następująco:

$$k_i = \varphi_i^{-1}(l) = k_{i-1} + \vartheta_i - 1 \quad \text{dla } i=1,2,\ldots,m \qquad /25/$$

gdzie

$$\psi_i = \mu(z)\left[\sum_{j=1}^{z} c_{r_i-j}^{m-i} \geqslant l - \sum_{j=1}^{i-1}\sum_{s=1}^{u_j} c_{r_j-s}^{m-j}\right]$$

Sprawdzimy, że funkcja φ ma własności WL.
Korzystając z tożsamości /22/, funkcję /24/ możemy zapisać w postaci:

$$1 = 1 + \sum_{i=1}^{m}\left(c_{r_i}^{m-i+1} - c_{r_i-u_i}^{m-i+1}\right).$$

Weźmy dwie takie kombinacje $k, k' \in K_p$ że $k \prec k'$ pokażemy, że wówczas zachodzi nierówność:

$$1 + \sum_{i=1}^{m}\left(c_{r_i}^{m-i+1} - c_{r_i-u_i}^{m-i+1}\right) < 1 + \sum_{i=1}^{m}\left(c_{r_i'}^{m-i+1} - c_{r_i'-u_i'}^{m-i+1}\right) \quad /26/$$

Przyjmijmy, że kombinacje k i k' zajmują kolejne miejsca w porządku leksykograficznym tzn., że $k' = k+1$.
Kombinacje takie mają własności:

1. $k_s' = k_s + 1$ dla pewnego $1 \leqslant s \leqslant m$
2. $k_i' = k_i$ dla $i = 1,2,\ldots,s-1$
3. $k_i' = k_s'$ dla $i = s+1,s+2,\ldots,m$
4. $k_i = n$ dla $i=s+1,s+2,\ldots,m$

Korzystając z własności 2, z obu stron nierówności /26/ możemy odrzucić wszystkie składniki sum dla $i=1,2,\ldots,s-1$.
Zauważmy, że dla $i=s+2,s+3,\ldots,m$ mamy:

$$u_i = k_i - k_{i-1} = n - n = 0 \qquad /4/$$

$$u'_i = k'_i - k'_{i-1} = k'_s - k'_s = 0 \qquad /3/$$

$$r_i = n + m - i + 1 - k_{i-1} = m-i+1 \qquad /4/$$

$$r'_i = n+m-i+1-k'_{i-1} = n+m-i+1-k'_s \qquad /3/$$

W nawiasach obok umieszczone są numery podanych wyżej własności, z których skorzystaliśmy.

Podstawiając te wartości do nierówności /26/ otrzymujemy, że składniki sum dla $i=s+2, s+3, \ldots, m$ są równe zeru. Dlatego też do wykazania pozostała nierówność:

$$\sum_{i=s}^{s+1} \left(C_{r_i}^{m-i+1} - C_{r_i-u_i}^{m-i+1} \right) < \sum_{i=s}^{s+1} \left(C_{r'_i}^{m-i+1} - C_{r'_i-u'_i}^{m-i+1} \right) \qquad /27/$$

Zauważmy, że

$$u'_{s+1} = k'_{s+1} - k'_s = k'_s - k'_s = 0 \quad /\text{z własności 3/}$$

$$r'_s = n + m - s + 1 - k'_{s-1} = r_s \quad /\text{ z własności 2/}$$

$$r_{s+1} - u_{s+1} = n+m-s-i+1-k_s-k_{s+1}+k_s = n+m-s-n=m-s \text{/z własności 4/}$$

$$u'_s = u_s+1 \qquad /\text{z własności 1 i 2/}$$

Podstawmy te wartości do /27/, dokonajmy uproszczeń i otrzymamy

$$- C_{r_s-u_s}^{m-s+1} + C_{r_{s+1}}^{m-s} - C_{m-s}^{m-s} < - C_{r_s-u_s-1}^{m-s+1}$$

Przekształcając otrzymujemy:

$$C_{r_{s+1}}^{m-s} - 1 < C_{r_s-u_s}^{m-s+1} - C_{r_s-u_s-1}^{m-s+1}$$

Stosując do prawej strony tej nierówności tożsamość /17/ otrzymujemy

$$C^{m-s}_{r_{s+1}} - 1 < C^{m-s}_{r_s - u_s - 1}$$

Prawa strona tej nierówności jest równa $C^{m-s}_{r_{s+1}}$, gdyż

$$r_s - u_s - 1 = n+m-s+1-k_{s-1}-k_s+k_{s-1}-1=n+m-s-k_s=r_{s+1}.$$

Cała nierówność jest więc równoważna oczywistej nierówności

$$-1 < 0$$

a to dowodzi prawdziwości /26/. Funkcja /24/ jest więc ściśle rosnąca.

Minimalną wartość osiąga dla kombinacji najmniejszej tzn. takiej, że $k_i = i$ dla $i=1,2,\ldots,m$ i jest równa:

$$\min \varphi(k) = 1 + \sum_{i=1}^{m} \sum_{j=1}^{0} C^{m-i}_{r_i - j} = 1$$

gdyż $u_i = k_i - k_{i-1} = 0$ dla $i=1,2,\ldots,m$

Natomiast wartość maksymalną osiąga dla kombinacji takiej, w której $k_i = n$ dla $i=1,2,\ldots,m$ wówczas otrzymujemy:

$$\max \varphi(k) = 1 + C^{m-1+1}_{r_1} - C^{m-1+1}_{r_1-u_1} + \sum_{i=s}^{m} \left(C^{m-i+1}_{r_i} - C^{m-i+1}_{r_i-u_i} \right) =$$

$$= 1 + C^{m}_{n+m-1} - C^{m}_{m} + 0 = C^{m}_{n+m-1}.$$

Przejdźmy teraz do dowodu równości:

$$\varphi^{-1}\left(\varphi(k)\right) = k$$

tzn. musimy pokazać, że jeśli we wzorze /25/ określającym funkcję φ^{-1} podstawimy zamiast 1 jego definicję /24/, to otrzymamy tożsamość:

$$k_i = k_i \qquad \text{dla } i=1,2,\ldots,m.$$

Podobnie jak w przypadku kombinacji bez powtórzeń musimy więc pokazać, że jeśli przyjmujemy $z = k_i - k_{i-1} + 1$, to będzie prawdziwa następująca nierówność:

$$\sum_{j=1}^{z} C_{r_i-j}^{m-i} \geqslant l - \sum_{j=1}^{i-1} \sum_{s=1}^{u_j} C_{r_j-s}^{m-j} \qquad /28/$$

gdzie l określone jest wzorem /24/.

Jeśli zaś weźmiemy $z = k_i - k_{i-1}$, to nierówność powyższa nie będzie spełniona.

W ten sposób pokażemy, że wynikiem operacji minimum określającej ϑ_i we wzorze /25/ jest liczba $k_i - k_{i-1} + 1$.

Korzystając z tożsamości /22/ i dokonując niezbędnych uproszczeń nierówność /28/ zapiszemy w postaci:

$$C_{r_i}^{m-i+1} - C_{r_i-z}^{m-i+1} \geqslant 1 + \sum_{j=i}^{m} \left(C_{r_j}^{m-j+1} - C_{r_j-u_j}^{m-j+1} \right)$$

Wyłączmy pierwszy składnik spod znaku sumy, przenieśmy go na lewą stronę nierówności, po dokonaniu uproszczenia zastosujmy do lewej strony tożsamość /17/ i w wyniku otrzymamy:

$$C^{m-i}_{r_i-u_i} \geqslant 1 + \sum_{j=i+1}^{m} \left(C^{m-j+1}_{r_j} - C^{m-j+1}_{r_j-u_j} \right).$$

Ponieważ funkcja φ jest rosnąca, to suma prawej strony tej nierówności osiąga wartość maksymalną, gdy $k_j = n$ dla $j=i+1,\ldots,m$ i jest równa

$$C^{m-i}_{r_{i+1}} - C^{m-i}_{r_{i+1}-u_{i+1}} + \sum_{i=i+2}^{m} \left(C^{m-j+1}_{r_j} - C^{m-j+1}_{r_j-u_j} \right) = C^{m-i}_{r_{i+1}} - C^{m-i}_{r_{i+1}-u_{i+1}}$$

gdyż $u_j = k_j - k_{j-1} = n - n = 0$ dla $j=i+2,\ldots,m$ i równa zeru.

Podstawiając maksymalną wartość do prawej strony otrzymujemy

$$C^{m-i}_{r_i-u_i} \geqslant 1 + C^{m-i}_{r_{i+1}} - C^{m-i}_{r_{i+1}-u_{i+1}}$$

Ponieważ $r_i - u_i = r_{i+1} + 1$, to lewa strona jest równa

$$C^{m-i}_{r_{i+1}} + 1.$$

Przenosząc na lewą stronę składnik $C^{m-i}_{r_{i+1}}$ i stosując tożsamość /17/ nierówność przyjmie postać:

$$C^{m-i-1}_{r_{i+1}} \geqslant 1 - C^{m-i}_{r_{i+1}-u_{i+1}}$$

Nierówność ta jest prawdziwa, gdyż w najgorszym przypadku tzn. jeśli przyjmiemy $k_i = k_{i+1} = n$, to lewą stronę maksymalnie zmniejszymy zaś prawą zwiększymy, ale

wówczas otrzymujemy:

$$C_{m-i}^{m-i-1} \geqslant 1 - C_{m-i}^{m-i} = 0$$

W ten sposób dowiedliśmy prawdziwości /28/.

Pokażemy teraz, że w przypadku gdy $z = k_i - k_{i-1}$ tzn. gdy $z = u_i$, nierówność /28/ nie będzie spełniona.

Mamy więc pokazać, że prawdziwa jest nierówność:

$$\sum_{j=1}^{u_i} C_{r_i-j}^{m-i} < 1 + \sum_{j=1}^{m} \sum_{s=1}^{u_j} C_{r_j-s}^{m-j} - \sum_{j=1}^{i-1} \sum_{s=1}^{u_j} C_{r_j-s}^{m-j}$$

zamiast 1 wstawiliśmy tu jego określenie /24/ oraz zamiast z przyjęliśmy u_i.

Nierówność powyższą możemy też zapisać w postaci:

$$0 < 1 + \sum_{j=i+1}^{m} \sum_{s=1}^{u_j} C_{r_j-s}^{m-j}$$

Ponieważ tu podwójna suma jest zawsze nieujemna, to nierówność jest prawdziwa.

W ten sposób pokazaliśmy, że wyrażenie /25/ określa funkcję odwrotną do φ.

3.5. P a r t y c j e

Niech $X = \left\{(x_1, x_2, \ldots, x_m) : x_1+x_2+\ldots+x_m = n, 0 \leq x_i \leq n\right\}$

oraz $w = |X| = C^{m-1}_{n+m-1}$.

Funkcję $\varphi : X \to N_w$ numerującą partycje $x \in X$ według porządku leksykograficznego zapiszemy w postaci:

$$l = \varphi(x) = 1 + \sum_{i=1}^{m-1} \left(C^{m-i}_{u_i} - C^{m-i}_{u_i - x_i} \right) \qquad /29/$$

gdzie $u_i = n + m - i - \sum_{j=1}^{i} x_{j-1}$,

$x_0 = 0$.

Funkcją do niej odwrotną jest:

$$x_i = \varphi^{-1}(l) = (\mu z)\left[l - A_i \leq \sum_{j=0}^{z} C^{m-i-1}_{u_i-i-j} \right] \qquad /30/$$

gdzie

$$A_i = \sum_{r=1}^{i-1} \sum_{j=0}^{x-1} C^{m-r-i}_{u_r-i-j} \qquad /30a/$$

Funkcja φ ma własność WL.

Weźmy dwie takie partycje x i x' aby $x' = x+1$ pokażemy, że

$$1 + \sum_{i=1}^{m-1} \left(C^{m-i}_{u_i} - C^{m-i}_{u_i-x_i} \right) < 1 + \sum_{i=1}^{m-1} \left(C^{m-i}_{u'_i} - C^{m-i}_{u'_i-x'_i} \right)$$

Przyjmijmy, że $x'_s = x_s+1$ wówczas z obu stron tej nierówności możemy odrzucić jednakowe składniki dla i=1,2,...,s-1. Ponieważ partycje x i x' zajmują kolejne miejsca w uporządkowaniu leksykograficznym /założenie, że $x' = x+1$/, to $x_i = x'_i = 0$ dla i=s+2,s+3,...,m-1, dlatego też i te składniki sum możemy odrzucić. Ostatecznie do pokazania otrzymujemy następującą nierówność:

$$C^{m-s}_{u_s} - C^{m-s}_{u_s-x_s} + C^{m-s-1}_{u_{s+1}} - C^{m-s-1}_{u_{s+1}-x_{s+1}} < C^{m-s}_{u'_s} - C^{m-s}_{u'_s-x'_s} +$$

$$+ C^{m-s-1}_{u'_{s+1}} - C^{m-s-1}_{u'_{s+1}-x'_{s+1}} \qquad /31/$$

Zauważmy, że

1. $u_s = u'_s$ gdyż $x_j = x'_j$ dla j=1,2,...,s-1
2. $x'_{s+1} = 0$ /założenie/
3. $x'_s = x_s + 1$ /założenie/
4. $u_{s+1} = u_s - x_s - 1$ / z określenia u_i/

Uwzględniając 1-4. i dokonując niezbędnych uproszczeń, po odpowiednim przegrupowaniu składników nierówność /31/ przyjmie postać

$$C^{m-s-1}_{u_s-x_s-1} - C^{m-s-1}_{u_{s+1}-x_{s+1}} < C^{m-s}_{u_s-x_s} - C^{m-s}_{u_s-x_s-1}$$

Stosując do prawej strony tej nierówności tożsamość /17/ i upraszczając otrzymujemy nierówność:

$$- C^{m-s-1}_{u_{s+1}-x_{s+1}} < 0$$

Ponieważ z założenia, że $x' = x+1$ oraz $x'_s = x_s+1$ mamy $x_i = 0$ dla $i \geqslant s+2$, to $x_1+x_2+\ldots+x_{s+1} = n$, stąd

$$u_{s+1} - x_{s+1} = n + m - s - 1 - x_1-x_2-\ldots-x_s-x_{s+1}=m-s-1$$

otrzymujemy więc oczywistą nierówność

$$- 1 < 0$$

W ten sposób wykazaliśmy, że funkcja określona wzorem /29/ jest ściśle rosnąca.

Funkcja φ osiąga najmniejszą wartość wówczas, gdy partycja jest najmniejsza tzn. $x_1 = 0$, $x_2 = 0$, ..., $x_{m-1} = 0$, $x_m = n$ wówczas otrzymujemy

$$\min \varphi(x) = 1 + \sum_{i=1}^{m-1} \left(C^{m-i}_{u_i} - C^{m-i}_{u_i-0} \right) = 1$$

Wartość maksymalną osiąga zaś wówczas, gdy

$x_1 = n$, $x_2 = 0$, $x_3 = 0$, ..., $x_m = 0$ i wynosi ona:

$$\max \varphi(x) = 1 + C^{m-1}_{u_1} - C^{m-1}_{u_1-n} + \sum_{i=2}^{m-1} \left(C^{m-i}_{u_i} - C^{m-i}_{u_i-0} \right) =$$

$$= 1 + C^{m-1}_{n+m-1} - C^{m-1}_{n+m-1-n} = 1 + C^{m-1}_{n+m-1} - 1 = C^{m-1}_{n+m-1}$$

Pokażemy teraz prawdziwość równości:

$$\varphi^{-1}(\varphi(x)) = x$$

tzn. mamy pokazać, że jeśli we wzorze /30/ zamiast 1 wstawimy jego definicję /29/, to otrzymamy tożsamość

$$x_i = x_i$$

W tym celu, podobnie jak w przypadku kombinacji wykażemy prawdziwość dwóch nierówności:

$$1 - A_i \leqslant \sum_{j=0}^{x_i} C_{u_i-1-j}^{m-i-1} \qquad /32/$$

$$1 - A_i > \sum_{j=0}^{x_i-1} C_{u_i-1-j}^{m-i-1} \qquad /33/$$

gdzie 1 określona jest wzorem /29/.

Jeśli zamiast A_i w nierówności /32/ wstawimy jego określenie /30a/, natomiast z prawej strony tej nierówności spod znaku sumy wyłączymy ostatni składnik i połączymy jednakowe składniki występujące z obu stron nierówności, to otrzymamy:

$$1 - \sum_{r=1}^{i} \sum_{j=0}^{x_r-1} C_{u_r-1-j}^{m-r-1} \leqslant C_{u_i-1-x_i}^{m-i-1}$$

Po zastosowaniu tożsamości /22/ nierówność tę zapiszemy w postaci:

$$1 - \sum_{r=1}^{i} \left(C_{u_r}^{m-r} - C_{u_r-1-x_r-1}^{m-r} \right) \leqslant C_{u_i-1-x_i}^{m-i-1}$$

Przyjmując $x_r=0$ dla $r > i$ nierówność tę możemy tylko pogorszyć, ale wówczas otrzymujemy:

$$1 \leqslant C^{m-i-1}_{u_i-1-x_i}$$

Nierówność ta jest prawdziwa, gdyż $u_i-1-x_i=n+m-i-x_1-x_2-\ldots-x_{i-1}-1-x_i \geqslant m-i-1$ /suma $x_1+x_2+\ldots+x_i$ może być co najwyżej równa n/, a to dowodzi prawdziwości /32/.

Pokażemy teraz, że nierówność /33/ jest również prawdziwa. Korzystając z tożsamości /22/, wzór /29/ możemy zapisać w postaci:

$$1 = 1 + \sum_{r=1}^{m-1} \sum_{j=0}^{x_r-1} C^{m-r-1}_{u_r-1-j}$$

Jeśli wyrażenie to podstawimy do nierówności /33/ oraz zamiast A_i też wstawimy jego określenie, to po dokonaniu uproszczeń nierówność /33/ przyjmie postać

$$1 + \sum_{r=i+1}^{m-1} \sum_{j=0}^{x_r-1} C^{m-r-1}_{u_r-1-j} > 0$$

Ponieważ podwójna suma jest nieujemna, to nierówność ta jest prawdziwa.

Nieujemność tej sumy wynika z nierówności:

$$u_r - 1 - j > m - r - 1$$

Rozpisując ją dokładniej otrzymujemy:

$$n + m - r - \sum_{s=1}^{r} x_{s-1} - 1 - j > m - r - 1$$

nawet gdy $j = x_r - 1$ otrzymujemy:

$$n + m - r - x_0 - x_1 - \ldots - x_{r-1} - 1 - x_r + 1 > m - r - 1$$

a więc

$$n - x_1 - x_2 - \ldots - x_r > -1 ,$$

suma $x_1+x_2+\ldots+x_r$ nie przekracza n / z definicji partycji/. W ten sposób udowodniliśmy prawdziwość obu nierówności /31/ i /32/.

3.6. P e r m u t a c j e

Niech $P = \left\{(p_1,p_2,\ldots,p_n) : i \neq j \Rightarrow p_i \neq p_j,\ 1 \leq p_i \leq n\right\}$ oraz $w = |P| = n!$

Funkcję $\gamma : P \to N_w$ określimy w postaci:

$$1 = \gamma(p) = 1 + \sum_{i=1}^{n} a_i\,(n-1)! \qquad /34/$$

gdzie

$a_i = |N(i)|$ tzn. liczebność zbioru $N(i)$, który określa się następująco:

$$N(i) = \left\{j \in N_n : j < p_i,\ j \neq p_k \quad \text{dla } k = 1,2,\ldots,i-1\right\}.$$

Funkcję odwrotną do γ określa wzór:

$$p_i = \gamma^{-1}(1) = (a_1 + 1)\uparrow\left\{ j \in N_n : j \neq p_k \quad \text{dla } k=1,2,\ldots,i-1 \right\}$$

gdzie

$$a_i = \left\lfloor \frac{1 - i - \sum_{j=1}^{i-1} a_j (n-j)!}{(n-i)!} \right\rfloor$$

przy czym symbolem $(r)\uparrow Z$ oznaczono r-ty co do wielkości element zbioru Z. Pełne uzasadnienie tych wzorów podane zostało w [72].

3.7. O p t y m a l i z a c j a a l g o r y t m ó w

Obliczanie wartości funkcji numerujących oraz identyfikujących wymaga wielokrotnego obliczania wartości C_n^m. Z tego też względu czas obliczeń może być dość duży, nie mówiąc już o tym, że może wystąpić tzw. nadmiar - wartość silni może nie zmieścić się w jednej komórce pamięci maszyny. Można jednak zaproponować szereg sposobów omijających te trudności. Niżej pokażemy jak można efektywnie obliczać wartości funkcji numerującej kombinacje bez powtórzeń.

Zgodnie z definicją kombinacji, wartością składowej k_i może być dowolna liczba następującego ciągu:

$$i, i+1, \ldots, n-m+1 \qquad /35/$$

Ciąg ten składa się z $n-m+1$ liczb, ponumerujmy je kolejnymi liczbami $1,2,\ldots,n-m+1$. Symbolem w_{ij} oznaczmy tę liczbę ciągu /35/, która posiada numer j, oczywiście zachodzi następująca równość:

$$w_{ij} = i + j - 1 \qquad \text{dla } i=1,2,\ldots,m \quad j=1,2,\ldots,n-m+1$$

Niech symbol a_{ij} oznacza ilość takich kombinacji $k = (k_1,\ldots,k_m)$, w których składowe $k_1,k_2,\ldots,k_{i-1}$ są ustalone oraz $k_i = w_{ij}$.

Zauważmy, że zachodzi następująca równość:

$$a_{ij} = C^{m-i}_{n-w_{ij}} = C^{m-i}_{n-i-j+1} \qquad /36/$$

gdyż jeśli $k_i = w_{ij}$, to zgodnie z definicją kombinacji, wartościami pozostałych $m-i$ składowych k_j dla $j > i$ mogą być liczby

$$w_{ij} + 1,\ w_{ij} + 2,\ldots,n$$

liczb tych jest $n-w_{ij}$. Stąd też liczba a_{ij} równa jest ilości kombinacji z $n-w_{ij}$ po $m-i$, co określa powyższy wzór.

Liczby a_{ij} posiadają ciekawe własności, a mianowicie:

1. $a_{mj} = 1$ dla $j=1,2,\ldots,n-m+1$
2. $a_{i,n-m+1} = 1$ dla $i=1,2,\ldots,m$
3. $a_{ij} = a_{i+1,j} + a_{i,j+1}$ dla $i,j > 1$

Pierwsze dwie własności łatwo sprawdzamy korzystając bezpośrednio z definicji /34/, natomiast trzecia własność wynika z tożsamości /17/.

Z liczb a_{ij} utwórzmy tablicę o m wierszach i $n-m+1$ kolumnach:

$$a=\begin{bmatrix} a_{11}, & a_{12}, & \cdots, & a_{1,n-m+1} \\ a_{21}, & a_{22}, & \cdots, & a_{2,m-m+1} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{m1}, & a_{m2}, & \cdots, & a_{m,n-m+1} \end{bmatrix} \qquad /37/$$

Elementy tej tablicy można obliczać rekurencyjnie korzystając z podanych trzech własności, stosując przy tym tylko operację dodawania /szybszą od mnożenia/ bez konieczności liczenia silni.

Na podstawie tej tablicy znowu tylko poprzez dodawanie odpowiednich elementów możemy obliczać wartość funkcji /14/ oraz funkcji do niej odwrotnej.

Uwzględniając fakt, że

$$\sum_{i=1}^{b} x_i = \sum_{i=a}^{a+b-1} x_{i-a+1}$$

oraz korzystając z tożsamości /22/ możemy dokonać następującego przekształcenia wzoru /14/:

$$1 = \beta(k) = 1 + \sum_{i=1}^{m} \left(C_{n-k_{i-1}}^{m-i+1} - C_{n-k_i+1}^{m-i+1} \right) =$$

$$= 1 + \sum_{i=1}^{m} \sum_{j=1}^{k_i-k_{i-1}-1} C_{n-k_{i-1}-j}^{m-i} = 1 + \sum_{i=1}^{m} \sum_{j=t_i}^{k_i-1} C_{n-i-j+1}^{m-i} =$$

$$= 1 + \sum_{i=1}^{m} \sum_{j=t_i}^{k_i-i} a_{ij}$$

gdzie $t_i = k_{i-1} - i+2$.

Tak więc wartość funkcji /14/ równa jest sumie odpowiednich elementów tablicy /37/.

Przekształcenia wzoru /15/ dokonuje się identycznie.

W przypadku kombinacji z powtórzeniami można stosować tę samą metodę. Zupełnie analogiczne rozważania prowadzą do takiej samej tablicy z tym, że w danym przypadku będzie to tablica o m wierszach i n kolumnach.

Rozpatrzmy teraz numerowanie permutacji.
Obliczenie wartości funkcji określonej wzorem /34/, jak łatwo zauważyć, wymaga dokonania $\frac{n+1}{2}$. n mnożeń.

Jeśli jednak sumowanie będziemy dokonywali w odwrotnym porządku niż to jest zapisane wzorem /34/ oraz wartość silni będziemy liczyć rekurencyjnie, to do obliczenia wartości funkcji /34/ potrzeba będzie 2n mnożeń.
Przy obliczaniu wartości funkcji odwrotnej, poważną trudność może sprawić obliczenie liczb a_i.
Zauważmy [72], że $0 \leqslant a_i \leqslant n-i$, tzn. liczby a_i są punktami $a = (a_1, a_2, \ldots, a_n)$ prostopadłościanu n wymiarowego takiego, że $0 \leqslant a_i \leqslant n-i$. Ponieważ $a_n = 0$, to możemy rozpatrywać prostopadłościan n-1 wymiarowy.
Tak więc każda permutacja n-elementowa odpowiada pewnemu punktowi prostopadłościanu n-1 wymiarowego. Na podstawie tego spostrzeżenia w łatwy sposób możemy nie tylko obliczać wartość funkcji /25/ lecz także i generować wszystkie permutacje lub też tylko następną do danej. Gdyż w tym celu wykorzystujemy efektywne algorytmy generowania punktów prostopadłościanu, numerowania i identyfikowania tych punktów.

Zauważmy, że możemy dokonać następującego przekształcenia funkcji /3/ - numerującej punkty prostopadłościanu:

$$j = 1 + \sum_{i=1}^{m} (k_i - d_i) z_i = 1 + (k_1 - d_1) l_2 l_3 \ldots l_m + \ldots +$$

$$+ (k_{m-1} - d_{m-1}) l_m + k_m - d_m =$$

$$= 1 + \left(\ldots\left((k_1 - d_1) l_2 + k_2 - d_2\right) l_3 + \ldots + k_{m-1} - d_{m-1}\right) l_m + k_m - d_m$$

To znaczy, że wartość funkcji /3/ w punkcie k możemy obliczać według efektywnego schematu Hornera [19].

3.8. Uwagi końcowe i uogólnienia

Rozpatrzymy teraz kilka zagadnień nieco ogólniejszych od tych, które rozpatrywane były w poprzednich paragrafach tego rozdziału.

Niech dany będzie $N_n = \{1, 2, \ldots, n\}$, na podstawie tego zbioru tworzymy wszystkie możliwe słowa m-elementowe tzn. wariancje z powtórzeniami z n po m. Zbiór ten oznaczmy symbolem K, zaś jego liczebność symbolem a, tzn. $a = n^m$. Zbiór wszystkich podzbiorów zbioru K oznaczymy symbolem X, tzn. $X = 2^K$. Przyjmujemy, że każdy podzbiór zbioru K przedstawiać będziemy w standardowej postaci jako ciąg słów uporządkowanych leksykograficznie.

Poza tym uporządkujemy wszystkie podzbiory zbioru K w ten sposób, że mniej liczny jest wcześniejszy od bardziej licznego, natomiast podzbiory równoliczne uporządkujemy według relacji porządku leksykograficznego /traktując je jako ciągi elementów zbioru N_n /.

W ten sposób otrzymamy następujący ciąg podzbiorów:

$$X_1, X_2, \ldots, X_w \qquad /38/$$

gdzie $w = |X| = 2^{|K|} = 2^a$

Symbolem w_i oznaczymy liczebność podzbioru

$$X_i = \left\{ x_1^i, x_2^i, \ldots, x_{w_i}^i \right\}.$$

Rozpatrzmy teraz kilka zadań.

Zadanie 1.

Dany jest pewien podzbiór $\left\{x_1^i, x_2^i, \ldots, x_{w_i}^i\right\} \subset X$ należy określić jego miejsce w ciągu /38/, tzn. obliczyć numer i.

Odpowiedź na to zadanie daje następujący wzór:

$$i = 1 + \sum_{j=1}^{w_i - 1} C_a^j + \beta(k, a, w_i) \qquad /39/$$

gdzie

$$k = \left(\alpha\left(x_1^i, n, m\right), \alpha\left(x_2^i, n, m\right), \ldots, \alpha\left(x_{w_i}^i, n, m\right)\right) \qquad /39a/$$

Wzór ten wynika z prostych rozważań:

zauważmy, że dany zbiór X_i o nieznanym indeksie i, składa się z różnych słów zbioru K. Stąd wektor numerów jakie te słowa posiadają w leksykograficznym uporządkowaniu zbioru K stanowi pewną kombinację z a po w_i. Zauważmy też, że j-ta składowa tej kombinacji jest równa

$$\alpha\left(x_j^i, n, m\right)$$

jest to numer słowa $x_j^i \in X_i$ w zbiorze K.

Stąd też cała kombinacja wyraża się wzorem /39a/. Numer tej kombinacji w zbiorze wszystkich kombinacji z a po w_i wynosi $\beta(k, a, w_i)$.

Jeśli uwzględnimy, że dany w_i-elementowy zbiór X_i, w ciągu /38/ poprzedzają wszystkie j-elementowe zbiory /j=0,1,...,w_i-1/ to otrzymamy wzór /39/.

Zadanie 2.

Dany jest pewien numer $1 \leqslant i \leqslant w$ należy podać wszystkie elementy zbioru X_i znajdującego się na i-tym miejscu w ciągu /38/.

Aby taki zbiór określić należy przede wszystkim ustalić ilu elementowy jest ten zbiór, tzn. określić liczbę w_i. W tym celu, jak wynika to z /39/, należy znaleźć taką całkowitą liczbę w_i aby spełniała układ nierówności:

$$\begin{cases} 1 + C_a^1 + C_a^2 + \ldots + C_a^{w_i-1} < i \\ 1 + C_a^1 + C_a^2 + \ldots + C_a^{w_i} \geqslant i \end{cases}$$

a więc

$$w_i = (\mu z)\left[\sum_{j=1}^{z} C_a^j \geqslant i-1\right] \qquad /40/$$

Jeśli już wiemy, że szukany zbiór znajduje się w grupie podzbiorów w_i-elementowych, to musimy określić jego miejsce w tej grupie.

Wyraża się ono oczywistym wzorem:

$$b = i - \left(1 + C_a^1 + C_a^2 + \ldots + C_a^{w_i - 1}\right) \qquad /41/$$

Mając numer b, za pomocą funkcji β^{-1} możemy otrzymać odpowiadającą mu kombinację z a po w_i, a mianowicie jest to kombinacja $\beta^{-1}(b, a, w_i)$. Kombinacja ta określa numery słów jakie wchodzą do zbioru X_i.
Zauważmy, że j-tą składową kombinacji $\beta^{-1}(b, a, w_i)$ otrzymujemy za pomocą funkcji ω w sposób następujący:

$$\omega\left(j, \beta^{-1}(b, a, w_i)\right)$$

składowa, ta określa numer pewnego słowa ze zbioru K. Słowo to identyfikujemy za pomocą funkcji α^{-1}. Stąd też dla zadanego $1 \leqslant i \leqslant w$ odpowiadający mu zbiór X_i określimy następująco:

$$X_i = \left\{\alpha^{-1}\left(\omega\left(j, \beta^{-1}(b-a-w_i)\right), n, m\right)\right\}_{j=1,2,\ldots,w_i} \qquad /42/$$

Zadanie 3.

Dane jest pewne słowo x_j^i, o którym wiadomo, że należy do zbioru X_i /i jest znane/ należy określić numer j tego słowa jaki posiada ono w leksykograficznym uporządkowaniu zbioru X_i.

Aby to zadanie rozwiązać nyleży, podobnie jak w zadaniu poprzednim, określić ilu elementowy jest zbiór o numerze i,

następnie należy określić jego miejsce wśród podzbiorów w_i elementowych.

Zauważmy, że dane słowo x_j^i w uporządkowaniu leksykograficznym zbioru K zajmuje miejsce o numerze $\alpha(x_j^i, n, m)$.

Stąd też po wyznaczeniu liczby w_i według wzoru /40/ oraz liczby b według wzoru /41/, nieznany numer j określimy ze wzoru:

$$j = \gamma\left(\alpha(x_j^i, n, m),\ \beta^{-1}(b, a, w_i)\right) \qquad /43/$$

Zadanie 4.

Dany jest pewien numer $1 \leqslant j \leqslant w_i$ oraz numer $1 \leqslant i \leqslant w$ należy określić element x_j^i tzn. element, który w zbiorze X_i zajmuje j-te miejsce.

Zupełnie analogiczne rozważania jak przy zadaniu 2 prowadzą do następującego wzoru:

$$x_j^i = \alpha^{-1}\left(\omega(j,\ \beta^{-1}(b, a, w_i)),\ n, m\right) \qquad /44/$$

Jeśli uzmiennimy j od 1 co 1 aż do w_i, to otrzymamy cały zbiór X_i.

Niech n=3, m=2, chcemy określić X_{77}.

Ponieważ $a = 3^2 = 9$, to stwierdzamy, że liczba $w_{77} = 3$ jest wynikiem operacji /40/, stąd $b = 77 - (1 + C_9^1 + C_9^2) = 77 - (1 + 9 + 36) = 31$.

Zgodnie z wzorem /42/ otrzymujemy:

$$X_{77} = \left\{\alpha^{-1}(\omega(1,\ \beta^{-1}(31,9,3)),3,2),\ \alpha^{-1}(\omega(2,\ \beta^{-1}(31,9,3)),3,2),\right.$$

$$\bar{\alpha}^{1}\left(\omega\left(3,\ \bar{\beta}^{1}\left(31,9,3\right)\right),\ 3,2\right)\Big\} =$$

$$= \Big\{ \bar{\alpha}^{1}\left(\omega\left(1,\left(2,3,6\right),3,2\right),\ \bar{\alpha}^{1}\left(\omega\left(2,\left(2,3,6\right),3,2\right),\ \bar{\alpha}^{1}\left(\omega\left(3,\left(2,3,6\right),\ 3,2\right)\right)\Big\}$$

$$= \Big\{ \bar{\alpha}^{1}\left(2,3,2\right),\ \bar{\alpha}^{1}\left(3,3,2\right),\ \bar{\alpha}^{1}\left(6,3,2\right)\Big\} =$$

$$= \Big\{ \left(1,2\right),\ \left(1,3\right),\ \left(2,3\right)\Big\}$$

a więc zbiór X_{77} stanowią kombinacje bez powtórzeń z 3 po 2. W podobny sposób można sprawdzić, że przy powyższych założeniach zbiór X_{21} jest zbiorem permutacji 2 elementowych, zbiór X_{349} jest zbiorem kombinacji z powtórzeniami z 3 po 2.

4. WYBRANE ZASTOSOWANIA METOD KOMBINATORYCZNYCH

4.1. W s t ę p

Jednym z ważniejszych problemów nauki i praktyki jest tzw. problem optymalności. Określa się go zwykle następująco: dany jest pewien zbiór D rozwiązań dopuszczalnych, w którym należy znaleźć taki element $x_0 \in D$, który maksymalizuje lub minimalizuje pewną funkcję celu $f(x)$.

Wśród wszystkich zagadnień ekstremalnych wyodrębnia się klasę tzw. zagadnień r e g u l a r n y c h [44]. Charakteryzują się one między innymi tym, że

- funkcja celu posiada tylko jedno ekstremum,
- łatwo sprawdza się warunki konieczne i dostateczne istnienia ekstremum lokalnego,
- zbiór rozwiązań dopuszczalnych jest spójny i zwykle wypukły.

Do rozwiązania zagadnień regularnych opracowano szereg efektywnych algorytmów, które doczekały się licznych opracowań podręcznikowych i monograficznych /por. [8,18,47,89] /.

Zagadnienia regularne stanowią raczej niewielką klasę wśród wszystkich zagadnień optymalizacyjnych. W większości przypadków są to zagadnienia nieregularne charakteryzujące się m.in. tym, że:

- funkcja celu posiada wiele ekstremów,
- nie jest znana postać analityczna funkcji celu,
- zbiór rozwiązań dopuszczalnych jest zbiorem skończonym określanym zwykle za pomocą skomplikowanych wyrażeń logicznych i warunków kombinatorycznych itp.

Do rozwiązywania zagadnień nieregularnych istniejące klasyczne metody programowania matematycznego nie są przydatne, poza nielicznymi przypadkami r e g u l a r y z a c j i zagadnień nieregularnych[1].

Do rozwiązywania podobnych zagadnień stosuje się prawie wyłącznie metody kombinatoryczne. Metody te polegają na częściowym przeliczaniu /partial enumeration/ możliwych wariantów rozwiązań. W szczególnym przypadku może to być zbadanie /przeliczenie/ wszystkich możliwości.

W większości przypadków do eliminacji niekorzystnych wariantów stosowane są różne reguły h e u r y s t y c z n e stąd też metody te nazywane są metodami kombinatoryczno-heurystycznymi [58] .

Wśród dotychczas opracowanych metod kombinatorycznych największe znaczenie posiadają metody o s a c z a n i a[2].

Główna idea tych metod polega na tym, że buduje się taki ciąg $D_1, D_2, \ldots, D_n$ podzbiorów zbioru rozwiązań dopuszczalnych, że każdy z nich zawiera rozwiązanie optymalne oraz każdy następny podzbiór jest mniej liczny od poprzedniego. Postępowanie kończy się wówczas, gdy ostatni podzbiór jest jednoelementowy /szukane rozwiązanie/ lub jest dostatecznie mały aby można było jako rozwiązanie optymalne przyjąć dowolny punkt z tego zbioru lub znaleźć je poprzez zbadanie wszystkich możliwości.

---

[1] O metodach regularyzacji szeroko traktuje monografia [44] .

[2] Termin zapożyczony z pracy [34] .

W przypadku optymalizacji funkcji jednej zmiennej, najprostszą metodą osaczania jest metoda dychotomii [98], lub metoda złotego podziału [98]. Dowodzi się [100], że najbardziej efektywną metodą znajdowania punktu ekstremalizującego unimodalną funkcję jednej zmiennej jest metoda Fibonacciego[1].

Czynione są też próby uogólnienia jej na przypadek wielowymiarowy [49].

Do znajdowania punktów ekstremalizujących funkcje określone na zbiorach słów kombinatorycznych opracowano cały szereg efektywnych metod obejmowanych wspólną nazwą branch and bound. Zalicza się do nich metodę Land i Dpig [44], metodę Little'a i in. [54], addytywny algorytm Balasa [7,26], algorytm leksykograficzny [45] itp.

W następnym paragrafie danego rozdziału dokonano przeglądu niektórych typowych zagadnień ekstremalnych rozwiązywanych za pomocą metod kombinatorycznych.

Doboru tych zagadnień dokonano celowo, chodziło bowiem o to aby przedstawić uniwersalny charakter podejścia kombinatorycznego przy rozwiązywaniu skomplikowanych zagadnień ekstremalnych.

I tak w rozpatrywanym punkcie 4.2.1 problemie optymalnego wyboru predyktant rozważa się zadanie wyznaczania ekstremum funkcji określonej na zbiorze kombinacji. Do jego rozwiązania przedstawiono dwie metody kombinatoryczne /dokładną i przybliżoną/.

---

[1] Metoda ta zaprogramowania została na maszynę ODRA-1204 i wykorzystana jest przy rozwiązaniu zagadnienia Steinera /por. punkt 4.2.5 niniejszego rozdziału/.

W literaturze można spotkać wiele zagadnień tego typu gdy obszarem określoności pewnej funkcji jest zbiór wszystkich lub niektórych podzbiorów danego zbioru. Zagadnieniem takim jest np. problem wyboru optymalnego wariantu inwestycyjnego rozpatrywany w punkcie 4.2.2.
Cechą wspólną takich zagadnień jest to, że są one nieregularne i nie mają efektywnych algorytmów rozwiązania.

Oprócz zagadnień polegających na wyznaczeniu ekstremum funkcji określonej na zbiorze kombinacji, w literaturze znanych jest wiele takich zagadnień, gdzie funkcje celu określone są na zbiorze permutacji.

Jednym z najbardziej znanych zagadnień tego typu jest tzw. problem komiwojażera. Jedyne metody jakie stosuje się do jego rozwiązania są metodami kombinatorycznymi, najbardziej znaną z nich jest metoda branch and bound [54] . W paragrafie 4.2.4. przedstawiono metodę przybliżonego rozwiązania tego zadania, ciekawe rozwiązanie tego problemu w tzw. metryce rzymskiej podane jest w [37] .

Tego samego typu jest też szeroko znane w literaturze zagadnienie optymalnych sekwencji operacji. Zagadnienie to, nie doczekało się dotychczas żadnego efektywnego rozwiązania. Ostatnio w Instytucie Metod Rachunku Ekonomicznego przygotowana została praca L.Warężaka [97] , w której przedstawione są niektóre metody kombinatoryczne rozwiązania tego zadania. Praca ta dostarcza licznych przykładów i dowodów znaczenia stosowanego obecnie podejścia kombinatorycznego do zagadnień programowania dyskretnego. W punkcie 4.2.6 rozpatrzono rozwiązanie

tego zadania w przypadku gdy liczba operacji nie jest duża /nie większa niż 9/.

Nieco odmiennym zagadnieniem jest problem najkrótszych dendrytów. W szczególnym przypadku gdy najkrótszy dendryt może zawierać tylko punkty danego zbioru, to do znalezienia go stosowane są efektywne algorytmy o charakterze kombinatorycznym /por. [24,80,94] /. Natomiast w przypadku gdy najkrótszy dendryt może zawierać dodatkowe punkty spoza danego zbioru, to do znalezienia go dotychczas nie opracowano żadnego efektywnego algorytmu. Przybliżoną metodę kombinatoryczną przedstawiono w punkcie 4.2.5.

W paragrafie 4.3 pokazano, że metody kombinatoryczne /algorytmy rozpatrywane w rozdziałach 2 i 3/ umożliwiają efektywne stosowanie środków współczesnej techniki obliczeniowej do rozwiązywania szeregu nieekstremalnych zagadnień wielowymiarowej statystyki matematycznej oraz rachunku ekonomicznego.

4.2. Zagadnienia ekstremalne

4.2.1. Optymalny wybór predyktant

Załóżmy, że pewna zmienna losowa Y może być objaśniona /scharakteryzowana/ za pomocą zmiennych $X_1, X_2, \ldots, X_n$ zwanych predyktantami.

Jeśli predyktanty są między sobą skorelowane /zależne/, to zmienną Y można opisać za pomocą mniejszej niż n liczby predyktant.

Zadanie polega na tym, aby liczbę tę określić oraz wskazać odpowiednie predyktanty.

Jedna z podstawowych trudności tego zadania polega na tym, aby skonstruować taką unormowaną funkcję określoną na zbiorze wszystkich kombinacji predyktant, która osiąga wartość maksymalną dla kombinacji optymalnej. Kombinacja predyktant jest optymalna jeśli zawiera predyktanty możliwie najmniej skorelowane między sobą, natomiast silnie ze zmienną Y.

Funkcję taką, po raz pierwszy zaproponował Z.Hellwig [33] i jest ona następująca:

$$H(m,k) = \sum_{j=1}^{m} \frac{r_{k_j}^2}{\sum_{i=1}^{m} |r_{k_i k_j}|}$$

gdzie

$k = (k_1, k_2, \ldots, k_{1n})$ - kombinacja numerów predyktant,

$r_{k_i k_j}$ - współczynnik korelacji między zmiennymi X_{k_i} oraz X_{k_j}

r_{k_j} - współczynnik korelacji między zmienną Y oraz X_{k_j}.

Do chwili obecnej nie znaleziono efektywnego algorytmu znajdowania ekstremum tej funkcji dla dowolnych wartości n. Gdy $n \leq 10$ ekstremum można znaleźć poprzez zbadanie wszystkich kombinacji. Program taki został opracowany na maszynę Odra 1204 /por.[P15]/, program ten przewiduje też możliwość znalezienia kilku najlepszych kombinacji.

Gdy n jest duże /n > 10/, można wstępnie / na podstawie macierzy korelacji/ wyodrębnić m najlepszych predyktant i dopiero spośród nich szukać kombinacji optymalnej. Możliwość takiej redukcji /według niepublikowanej metody Z.Hellwiga/ została również przewidziana w tym programie. Dla dużych n można też w następujący sposób szukać przybliżonego rozwiązania.

Szukamy najlepszej kombinacji jednoelementowej, następnie uzupełniamy ją do najlepszej dwuelementowej tzn. szukamy najlepszej dwuelementowej ale tylko spośród tych, które zawierają znalezioną jednoelementową. Dwuelementową rozszerzamy dalej do najlepszej trójelementowej itd. Postępowanie powtarzamy tak długo, aż dojdziemy do kombinacji n-elementowej lub też, gdy się okaże, że żadna kombinacja m+1 elementowa nie jest lepsza od znalezionej najlepszej m-elementowej. Postępowanie takie również zostało zaprogramowane[1] na maszynę ODRA 1204 /por. [P16] /. Przeprowadzone eksperymenty dały pozytywne wyniki, tzn. rozwiązania przybliżone często były identyczne z dokładnymi lub też różniły się nieznacznie /w sensie liczby predyktant w kombinacji/.

4.2.2. Zagadnienia wyboru optymalnego wariantu inwestycyjnego

Spośród zagadnień tego typu, sformułowanych i rozwiązywanych przez W.Bukietyńskiego [10,11] wybierzemy tylko niewielki fragment charakteryzujący stopień złożoności problemu, jego

---

1 W opracowanych programach wykorzystane są algorytmy generowania, numerowania oraz identyfikowania kombinacji bez powtórzeń.

charakter kombinatoryczny, a jednocześnie będący przykładem zastosowań omawianych w tej pracy algorytmów.

Pomijając merytoryczną stronę, podajemy tu formalne sformułowanie problemu.

Załóżmy, że dany jest zbiór liczb rzeczywistych

$$R = \{r_1, r_2, \ldots, r_n\}$$

oraz pewna liczba G.

Należy znaleźć taki podzbiór $r = \{r_{i_1}, r_{i_2}, \ldots, r_{i_m}\} \in 2^R$, który spełnia warunek

$$\sum_{r_i \in r} r_i \geqslant G \qquad /1/$$

oraz maksymalizuje pewną funkcję $B(m, r)$.

Postać jej pomijamy tutaj, koncentrując uwagę na zbiorze rozwiązań dopuszczalnych. Zbiór ten możemy zapisać następująco:

$$Z_B = \left\{ r \subset R : \sum_{r_i \in r} r_i \geqslant G \right\}$$

wówczas zadanie można sformułować następująco: znaleźć takie m^* oraz takie $r^* = (r_{i_1}^*, r_{i_2}^*, \ldots, r_{i_m}^*) \in Z_B$ aby

$$B(m^*, r^*) = \max B(m, r).$$

Do skonstruowania zbioru Z_B wykorzystujemy algorytm generowania kombinacji bez powtórzeń, gdyż składowe takiej kombinacji można traktować jako numery tych elementów zbioru R, które wchodzą do odpowiedniego podzbioru.

Dlatego też używać będziemy skrótu: kombinacja spełnia warunek /1/, rozumiejąc przez to, że odpowiadający jej podzbiór spełnia ten warunek.

Podamy teraz dwie reguły, które pozwolą wygenerować zbiór Z_B.

Załóżmy, że elementy zbioru R są uporządkowane rosnąco. Jeśli znajdziemy pewną kombinację m-elementową spełniającą warunek /1/, to możemy wówczas wygenerować wszystkie następne / w uporządkowaniu leksykograficznym/ kombinacje m-elementowe ponieważ będą również spełniały warunek /1/. Jeśli kombinacja m-elementowa zajmuje pierwsze miejsce w grupie kombinacji m-elementowych i spełnia warunek /1/, to bez sprawdzenia tego warunku generujemy wszystkie kombinacje m,m+1,...,n-elementowe.

Korzystając z tych reguł opracowany został program /per. [P22] /generowania wszystkich podzbiorów zbioru R stanowiących zbiór Z_B.

4.2.3. Zagadnienia plecakowe

Załóżmy, że dana jest pewna funkcja $g(x_1,x_2,\ldots,x_n)$, taka, że x_i może przyjmować wartości ze skończonego zbioru X_i /i = 1,2,...,n/. Zadanie polega na tym, aby znaleźć taki wektor

$$x = (x_1,x_2,\ldots,x_n)$$

spełniający ograniczenie

$$\sum_{i=1}^{n} x_i = m,$$

dla którego funkcja $g(x)$ osiąga ekstremum. Zagadnienia tego typu nazywane są zagadnieniami p l e c a k o w y m i /por. [44,50] /.

Jedną z konkretyzacji podobnego zagadnienia jest problem optymalnego przydziału samochodów: danych jest n zakładów transportowych, którym należy przydzielić m samochodów.

Efektywność wykorzystania samochodów przez i-ty zakład zależy od ilości przydzielonych mu samochodów i określona jest za pomocą pewnej funkcji $f_i(x)$ /przeważnie w postaci stablicowanej/.

Należy tak rozdzielić samochody, aby dawały największą łączną efektywność, tzn. należy znaleźć wektor

$$x = (x_1, x_2, \ldots, x_n)$$

o współrzędnych całkowitych nieujemnych spełniających warunek:

$$x_1 + x_2 + \ldots + x_n = m$$

i maksymalizujący funkcję:

$$g(x) = \sum_{i=1}^{n} f_i(x_i)$$

Do rozwiązywania zagadnień plecakowych zaproponowano szereg algorytmów [44,50] . W szczególności można stosować metodę programowania dynamicznego [8,89] , lub też metodę kombinatoryczną polegającą na zbadaniu wszystkich możliwości. Na maszynie cyfrowej Odra 1003 przeprowadzono szereg eksperymenrów w celu ustalenia efektywności obu metod. Okazało się, że programowanie dynamiczne daje rozwiązania w czasie o wiele krótszym niż metoda bezpośredniego badania.

Z drugiej jednak strony programowanie dynamiczne potrzebuje ogromnych objętości pamięci /ilość komórek jest proporcjonalna do wielkości m/ i na maszynie o małej pamięci może mieć ograniczone zastosowanie. Metoda bezpośrednia natomiast, polegająca na kolejnym generowaniu i sprawdzeniu rozwiązań równania: $x_1 + x_2 + \ldots + x_n = m$ mimo, że jest wolniejsza ale nie potrzebuje pamięci i przy tym ze względu na swą prostotę jest łatwa w rozpowszechnianiu.

4.2.4. Problem komiwojażera

Zadanie polega na tym, aby dla danych n punktów na płaszczyźnie znaleźć najkrótszą łamaną zamkniętą przechodzącą przez wszystkie punkty /por. [31] /.

Do rozwiązania tego zadania proponowano wiele różnych algorytmów [93] . Największe uznanie wśród nich zdobył algorytm tzw. metody podziału i granic /branch and bound/, opisany w wielu pracach /por. [54,87] /. Jedną z cech charakterystycznych maszynowej realizacji tego algorytmu jest stosunkowo duże zapotrzebowanie na objętość pamięci. Liczba niezbędnych komórek pamięci roboczej zależy od konkretnego zadania i z góry określić jej nie można.

Dla przykładu program rundreise [43] zaadaptowany na maszynę Odra 1204 po przekładzie zajmuje 1628 komórek. Do rezerwacji dynamicznej pozostaje 9834 komórek, z czego na rozbudowę drzewa rozgałęzień /istota metody [34] /można wykorzystać $4967 - n^2 - n - (n+1)^2$ komórek.

Na przykład przy szukaniu najkrótszej trasy dla 25 punktów liczba ta /wynosząca 3591/ po wykonaniu 1 iteracji /ok. 1 godz./ okazała się niewystarczająca.
Natomiast najkrótszą trasę dla 17 punktów znaleziono po 15 minutach.

Do przybliżonego rozwiązania tego zadania można wykorzystać zaproponowaną przez Z.Hellwiga ideę agregacji.

W myśl tej idei dla znalezienia najkrótszej trasy należy dany zbiór punktów podzielić na rozłączne podzbiory zwane agregatami. Po wyodrębnieniu agregatów należy znaleźć dla nich najkrótszą trasę zamkniętą. Następnie należy określić najkrótsze połączenie w każdym agregacie. Jeżeli liczba wyodrębnionych agregatów jest zbyt duża, to można je dalej agregować aż się otrzyma tyle agregatów, że najlepsze ich połączenie można znaleźć poprzez zbadanie wszystkich możliwości.

Załóżmy, że dany jest następujący zbiór punktów

$$X = \left\{ (x_1 y_1), (x_2, y_2), \ldots, (x_n, y_n) \right\}$$

Rozbijemy go na podzbiory /agregaty/ zawierające nie więcej niż 7 punktów.

W tym celu obliczamy

$a = \max x_i - \min x_i$ /rozpiętość wg osi OX/

$b = \max y_i - \min y_i$ / rozpiętość wg osi OY/.

Jeśli $a > b$ to zbiór X rozbijamy na dwa podzbiory X_1 i X_2, takie że

$$X_1 = \left\{(x_i, y_i) \in X : x_i \leq \left[\frac{a}{2}\right]\right\} \quad \text{oraz} \quad X_2 = X \setminus X_1$$

Jeśli zaś $a \leqslant b$, to

$$X_1 = \left\{(x_i, y_i) \in X : y_i \leq \left[\frac{b}{2}\right]\right\} \quad \text{oraz} \quad X_2 = X \setminus X_1$$

to znaczy zbiór X rozcinamy prostą równoległą do osi OX przechodzącą przez punkt $\left(0, \left[\frac{b}{2}\right]\right)$.

Jeśli zbiór X_i zawiera więcej niż 7 punktów, to postępujemy z nim podobnie jak ze zbiorem X, rozbijając go na dalsze dwa podzbiory.

Postępowanie powtarzamy tak długo, aż otrzymamy podzbiory zawierające co najwyżej 7 punktów.

Utworzenie agregatów kończy pierwszy etap metody agregacji. Następnym etapem jest cykliczne połączenie utworzonych agregatów tak, aby suma długości odcinków łączących była najmniejsza oraz aby żadne dwa różne odcinki nie posiadały wspólnego punktu.

Po połączeniu agregatów mamy taką sytuację, że w każdym z nich dwa różne punkty należą do różnych odcinków łączących, nazwijmy te punkty związanymi.

Ostatni etap w metodzie agregacji polega na tym, aby w każdym podzbiorze znaleźć najkrótszą drogę łączącą dwa punkty związane i przechodzącą przez wszystkie pozostałe. Ponieważ każdy podzbiór zawiera co najwyżej 7 punktów, to najkrótszej drogi szukamy metodą pełnego przeliczania możliwości.

Postępowanie to ilustrują poniższe rysunki.

Dane punkty

Podział na agregaty

Połączenie agregatów

Rozwiązanie

Metodę powyższą sprawdzono dla czterech zestawów miast polskich. Współrzędne miast wzięto z mapki Polski w skali 1 : 5.000.000. Podane są one w cm, przy czym część całkowita od dziesiętnej oddzielona jest kropką dziesiętną[1], zaś pary współrzędnych oddzielone są od siebie średnikiem.
Wyniki obliczeń przedstawione są w tabeli 1, przy czym długość połączenia podana jest w km /po uwzględnieniu skali/.
Obliczenia wykonano dla następujących zestawień:

---

[1] Zastosowano więc notację J.Nepera

Zestaw I /10 miast/:

6.8,6; 16.5,7.5; 13.4, 12.9; 11.7, 21.1; 20,0 04.4;
16.1,18.9; 4.1,11.7; 6.0,20.8; 14.9,1.2; 21.5,20.4;

Zestaw II /20 miast/:

6.8, 6; 7.6, 8.6; 13.7,11.0; 2.2,16.7; 11.7, 21.1;
21.0,9.1; 4.8,20.8; 9.4,14,0; 16.1,18.9; 4.1,11.7;
14.9,4.3; 12.9,19.9; 4.2,7.3; 14.8,11.0; 17.3,12.9;
2.2,17.9; 14.1,14.1; 14.9,1.2; 3.3,15.0; 21.5,20.4;

Zestaw III /25 miast/:

/wzięto z [46] /

-6.7, 2.3; -4.4, 3.6;-1.5, 4.4; -1.3, 4.0; -0.2, 3.5;
1.2, 2.6; -2.2, 2.8; -1.4, 1.1; 4.7, 1.2; -3.7,-0.3;
-5.6,-1.2; 1.8,-0.8; -0.3,-4.8;-4.8,-2.8; -3.6,-3.1;
2.1,-2.6; 4.1,-2.9; -4.7,-3.8;-0.8,-4.0; 1.3,-3.8;
-1.4,-5.1; 0.4,-5.6; 1.9,-5.7; 3.2,-5.6; -0.9,-6.1.

Zestaw IV /30 miast/:

5.8, 7.2; 7.6, 8.6; 16.5, 7.5; 13.7, 11.0; 22.2, 16.7;
13.4,12.9; 11.7,21.1; 21.0, 9.1; 12.6, 14.5; 9.5, 23.0;
20.0, 4.4; 9.4,14.0; 23.6, 8.9; 7.6, 13.7; 4.1, 11.7;
14.9, 4.3; 12.9,19.9; 4.2, 7.3; 7.2, 18.7; 17.3, 12.9;
10.5, 8.0; 20.0, 6.5; 2.2,17.9; 6.0, 20.8; 14.1, 14.1;
14.9, 1.2; 3.3,15.0; 21.5,20.4; 20.0, 1.7; 10.2, 16.5;

Jak wynika z zamieszczonej tabeli, wyniki otrzymane metodą agregacji różnią się bardzo mało od wyników otrzymanych metodą branch-and-bound. Czas uzyskiwania wyników przybliżonych

był jednak nieporównywalnie krótszy: metoda agregacji realizowana była na maszynie ODRA 1003, zaś metoda branch and bound na maszynie ODRA 1204 i czasy realizacji były porównywalne /natomiast szybkość pierwszej maszyny jest wielokrotnie mniejsza od szybkości drugiej/.

Tabela 1

Zestaw / Treść	I	II	III	IV
Komiwojażer /agregacja/	3455	4854	2744	5924
Komiwojażer /branch,bound/	3400	4350	2450	5200
Dendryt /Steinera/	2660	3550	2107	4269

4.2.5. Zagadnienie najkrótszych dendrytów

Załóżmy, że na płaszczyźnie danych jest n punktów. Punkty te należy połączyć odcinkami o najmniejszej łącznej długości. Zadanie to pochodzące jeszcze od A.Cayley'a /1821-1895/, po raz pierwszy rozwiązał Boruvka w 1926 roku /por. [70] /.

W literaturze [25] wymienia się zwykle algorytmy Kruskala i Prima pochodzące odpowiednio z 1956 i 1957 roku. Metodą

taksonomii wrocławskiej [24] zostało ono rozwiązane w 1951 roku.

Inne zadanie o najkrótszym dendrycie, posiadające duże znaczenie praktyczne, które prawdopodobnie nie posiada jeszcze efektywnego rozwiązania, polega na tym, że danych n punktów należy połączyć łamaną o najmniejszej długości. Jest to uogólnienie tzw. zagadnienia Steinera /por. [14] / dlatego też szukaną łamaną nazwiemy dendrytem Steinera[1].

Znany jest następujący warunek konieczny rozwiązania optymalnego [14] : jeśli danych jest n punktów, to może być co najwyżej n-2 punktów dodatkowych i w każdym z nich schodzą się trzy odcinki pod kątami 120°.

Jednak ilość wszystkich rozwiązań spełniających ten warunek rośnie wraz ze wzrostem n według prawa silni.

Prawdopodobnie jedyną znaną metodą jest potraktowanie tego zadania jako zmodyfikowanego zagadnienia izoperymetrycznego i rozwiązania metodami rachunku wariancyjnego [14] . Sposób taki został zaprogramowany na maszynę BESM-2 i jak podano w [38] program dla 30 punktów liczy 2000 rozkazów i czas liczenia przekracia 1 godz.

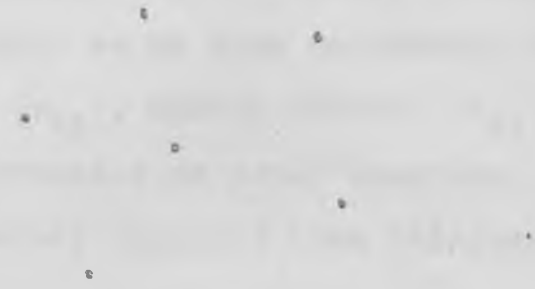
Niżej podamy przybliżony sposób rozwiązania tego zadania, który został zaprogramowany na maszynie ODRA 1204 /por. [P26]/.

Spośród danych n punktów wybieramy takie trzy, aby dendryt Steinera na tych punktach był minimalny. Następnie spośród niepołączonych jeszcze punktów wybieramy takie dwa aby przy-

---

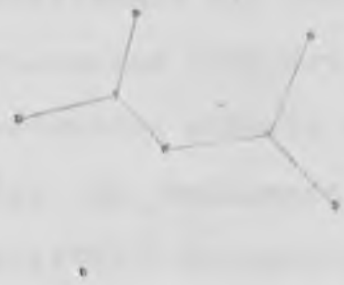
[1] Inne uogólnienie znane jest jako zagadnienie środka między /por. [37] oraz [P14] /.

łączenie ich do zbudowanego już dendrytu dało dendryt minimalny na pięciu punktach itd., aż połączymy wszystkie punkty. Postępowanie takie ilustrują poniższe rysunki.



Dane punkty

Najkrótszy dendryt Steinera na trzech punktach



Dołączono najlepsze dwa punkty

Pozostały punkt połączono z najbliższym

Opisany algorytm zastosowano do znajdowania najkrótszych połączeń dla wymienionych czterech zestawów miast polskich. Wyniki umieszczono również w tabeli 1.

4.2.6. Optymalne sekwencje operacji

Zadanie to formułuje się w ten sposób: danych jest m maszyn oraz n detali /czynności/, które obrabiane /wykonywane/ są na tych maszynach. Dana jest też pewna macierz $[c_{ij}]$, której element c_{ij} określa czas wykonywania i-tej czynności na j-tej maszynie.

Należy określić taką kolejność wykonywania czynności, która daje minimalny czas.

Tak postawione zadanie nie posiada jak dotychczas dokładnego rozwiązania dla dowolnych n i m, ciekawsze z opracowanych metod rozpatrzone są w pracy [9] lub [97].

Podtawione zadanie można rozwiązać poprzez zbadanie wszystkich n! możliwości /zakładając jednakową kolejność na wszystkich maszynach/.

Załóżmy, że oprócz ustalenia optymalnej kolejności chcemy też otrzymać dokładny harmonogram obróbki tzn. chcemy uzyskać macierz kolejności $[k_{ij}]$, której element k_{ij} określa czas rozpoczęcia wykonywania i-tej czynności na j-tej maszynie. Na podstawie takiej macierzy w łatwy sposób można sporządzić tzw. wykres Gantta, na którym widoczne są wszystkie przestoje maszyn[1].

Przyjmijmy, że:

1/ pierwsza czynność zaczyna być wykonywana na pierwszej maszynie w chwili zerowej,

---

[1] Opracowany program /por. [P17] /przewiduje możliwość automatycznego sporządzania takiego wykresu /w całkowitych jednostkach czasu/.

2/ kolejność wykonywania czynności określana jest za pomocą permutacji

$$(i_1, i_2, \ldots, i_n)$$

gdzie i_p - numer czynności zajmującej p-te miejsce w kolejności,

3/ element k_{ij} macierzy kolejności oznaczać będziemy symbolem $K[i,j]$ analogicznie elementy C_{ij} symbolem $C[i,j]$,

wówczas dla danej permutacji $(i_1, i_2, \ldots, i_n)$ macierz kolejności określimy według następujących zależności

$$K[i_1, 1] = 0$$

$$K[i_p, 1] = K[i_{p-1}, 1] + C[i_{p-1}, 1] \quad \text{dla } p=2,3,\ldots,n \qquad /2/$$

$$K[i_1, j] = K[i_1, j-1] + C[i_1, j-1] \quad \text{dla } j=2,3,\ldots,m \qquad /3/$$

oraz

$$K[i_p, j] = \max\left\{K[i_{p-1}, j] + C[i_{p-1}, j];\ K[i_p, j-1] + C[i_p, j-1]\right\} \qquad /4/$$

$$\text{dla } j = 2,3,\ldots,m$$

$$p = 2,3,\ldots,n.$$

Zależność /2/ określa momenty rozpoczęcia każdej czynności na pierwszej maszynie, zależność /3/ - momenty rozpoczęcia pierwszej czynności, kolejno na wszystkich maszynach. Długość całego cyklu obróbki wszystkich n czynności na m maszynach dla danej permutacji $(i_1, i_2, \ldots, i_n)$ wynosi oczywiście $K[i_n, m] + C[i_n, m]$.

Stosując podane zależności /2/ - /4/, przeprowadzono szereg eksperymentów na maszynie ODRA 1003. Okazało się, że podobną metodę można stosować dla małych $n(n < 9)$, np. dla $n=3$ czas szukania najlepszej kolejności /łącznie z wyprowadzeniem wszystkich ciągów/ wyniósł 3 min., wyprowadzenie wykresów Gantta zajęło dodatkowo 4 min.

Stosując ideę agregacji /zarówno maszyn jak i czynności/ podane algorytmy z powodzeniem można wykorzystać do rozwiązywania zagadnień o dużych wartościach n.

4.3. Zagadnienia nieekstremalne

4.3.1. Wielowymiarowe tablice korelacyjne

Przy programowaniu /na maszynie cyfrowej/ zagadnień ekonomicznych głównie o charakterze statystycznym często zachodzi potrzeba operowania tablicami wielowymiarowymi. Istniejące języki programowania posiadają aparat pozwalający operować tylko tablicami prostokątnymi i tylko przy ustalonym wymiarze.

Stosując funkcje numerujące kombinacje /por. rozdz. 3/, w łatwy sposób można operować zarówno tablicami prostokątnymi jak i trójkątnymi bez konieczności ustalenia wymiaru. Odpowiednie algorytmy podane są w [71], niżej zaś rozpatrzony będzie algorytm budowy tablic korelacyjnych oraz algorytm otrzymywania podtablic danej tablicy.

Oba algorytmy można stosować w wielu zagadnieniach statystyki wielowymiarowej /analiza wariancji, analiza regresji, badanie współzależności lub niezależności n wymiarowych zmiennych losowych/.

Rozpatrzmy więc algorytm budowy tablic korelacyjnych zwanych też tablicami współzależności [102] .

Załóżmy, że dane są realizacje n-wymiarowej zmiennej losowej $X = (X_1, X_2, \ldots, X_n)$. Na podstawie tych danych należy zbudować n-wymiarową tablicę korelacyjną.

Przyjmijmy, że dla zmiennej X_s dany jest wektor $g^s = (g_1^s, g_2^s, \ldots, g_{n_s}^s)$, który określa szereg rozdzielczy o $n_s + 1$ przedziałach klasowych:

$$(-\infty, g_1^s], (g_1^s, g_2^s], \ldots, (g_{n_s}^s, \infty) ,$$

Przedziałem klasowym zmiennej X nazywany będzie iloczyn kartezjański

$$(-\infty, g_1^s] \times (g_1^s, g_2^s] \times \ldots \times (g_{n_s}^s, \infty) \qquad /5/$$

Przedziały takie zwane też kostkami tworzą pewną tablicę n-wymiarową. Współrzędne przedziału /5/ w tej tablicy określone są za pomocą wektora

$$i = (i_1, i_2, \ldots, i_n),$$

gdzie $$1 \leq i_j \leq n_j+1 \quad \text{dla } j=1,2,\ldots,n .$$

Zbudowanie tablicy korelacyjnej polega więc na tym, aby każdą obserwację zmiennej losowej X zaliczyć do odpowiedniej kostki.

W tym celu dla danej obserwacji $(x_1,x_2,\ldots,x_n)$ należy określić współrzędne $i = (i_1,i_2,\ldots,i_n)$ odpowiadającej jej kostki. Współrzędne te określamy ze wzoru:

$$i_j = (\mu z)\left[x_j < g_z^j\right] \qquad \text{dla } j=1,2,\ldots,n$$

gdzie symbolem (μz) oznaczono operację minimum /por. rozdz. 3/. Ponieważ elementy tablicy korelacyjnej/jak i każdej innej/ w pamięci maszyny zapamiętywane są w postaci ciągu, to mając dany wektor wskaźników $i = (i_1,i_2,\ldots,i_n)$ należy na jego podstawie obliczyć miejsce jakie zajmuje w tym ciągu odpowiadająca mu kostka.

Miejsce to równe jest liczbie:

$$\Psi(i,d,g,n)$$

gdzie

Ψ - funkcja numerująca wariacje z powtórzeniami,

i - dany wektor wskaźników,

natomiast wektory d oraz g określają odpowiednio dolne i górne granice zmienności współrzędnych wektora $i = (i_1,i_2,\ldots,i_n)$ tzn. $d_j=1$ oraz $g_j = n_j + 1$ dla $j=1,2,\ldots,n$.

Przy rozpatrywaniu tablic korelacyjnych prawie zawsze zachodzi potrzeba obliczenia wektorów częstości lub liczebności brzegowych.

Ponieważ wektory takie można traktować jako jednowymiarowe podtablice w odpowiedni sposób otrzymane z tablicy n-wymiaro-

wej , to niżej rozpatrzymy ogólny algorytm otrzymywania dowolnych podtablic m-wymiarowych. Zajmiemy się przy tym tylko tym przypadkiem gdy $m < n$, podtablice tego samego wymiaru co dana tablica rozpatrzone zostały w [71] .

Aby otrzymać podtablicę m wymiarową z tablicy n wymiarowej, należy przede wszystkim wybrać te m spośród n współrzędnych, które będą określały daną podtablicę. Z pozostałymi n-m współrzędnymi można postąpić dwojako. Pierwszy sposób polega na tym, że ustala się pewne wartości tych współrzędnych tzn. tablicę n wymiarową tnie się w tych miejscach otrzymując w wyniku pewien płat lub plaster /m wymiarowy/. Stąd też podtablice tego rodzaju nazywa się p l a s t r a m i[1] /por. [95] /.

Drugi sposób polega na tym, że podtablicę m wymiarową otrzymuje się z tablicy n wymiarowej w wyniku sumowania po wszystkich wartościach pozostałych n-m wskaźników. Otrzymana w ten sposób podtablica nazywana będzie k o n d e n s a t e m.

Jeśli np. w danej macierzy przeprowadzone będzie sumowanie po wszystkich wartościach wskaźnika i, to otrzymany kondensat będzie wektorem, j-ty element którego w statystyce oznacza się zwykle symbolem $X_{\cdot j}$. Kropka na miejscu wskaźnika i oznacza, że po wskaźniku tym dokonano sumowania tzn.

$$X_{\cdot j} = \sum_{i} X_{ij}$$

---

[1] Wyższe języki programowania: PL/1,ALGOL-68,ALGEK /por. [95]/ posiadają odpowiedni aparat operowania takimi podtablicami, dlatego też sposób ich otrzymywania jest tu pominięty.

Gdy liczba wskaźników jest duża, stosowanie tej symboliki jest uciążliwe, a w ogóle niemożliwe gdy nie jest ona z góry ustalona. Dlatego też w dalszej części zamiast $X_{\cdot j}$ stosowany będzie zapis X^{1}_{ij}, jedynka u góry oznacza, że wykropkowany jest pierwszy wskaźnik[1].

Ogólnie zapis:

$$X^{k_1,k_2,\ldots,k_m}_{i_1,i_2,\ldots,i_n}$$

oznacza, że wykropkowane są te współrzędne wektora $i=(i_1,i_2,\ldots,i_n)$, których numery tworzą wektor $k=(k_1,k_2,\ldots,k_m)$, tzn. wykropkowane są współrzędne $i_{k_1},i_{k_2},\ldots,i_{k_m}$.

Tak więc

$$X^{k_1,k_2,\ldots,k_m}_{i_1,i_2,\ldots,i_n} = \sum_{i_{k_1}} \sum_{i_{k_2}} \cdots \sum_{i_{k_m}} X_{i_1,i_2,\ldots,i_n} \qquad /6/$$

W celu uproszczenia zapisów, wektor $(i_{k_1},i_{k_2},\ldots,i_{k_m})$ oznaczony będzie symbolem $i(k)$, zaś suma postaci $\sum_{i_1}\sum_{i_2}\cdots\sum_{i_n} X_{i_1 i_2 \ldots i_n}$ zapisywana będzie krócej jako $\sum_{i} X_i$. Zgodnie z tą umową wyrażenie /6/ przyjmie postać

$$X^{k}_{i} = \sum_{i(k)} X_i$$

---

[1] Notacja taka jest zbliżona do notacji Einsteina /por. [98] str. 240 /.

Jeśli stosuje się tzw. sumowanie uśrednione tzn. jeśli elementy X_i^k dzielone są przez ilość wszystkich dodanych elementów, to stosowany będzie zapis $\bar{X}_i^k$.
Jeśli $k = (1,2,\ldots,n)$ tzn. sumowanie dokonywane jest po wszystkich wskaźnikach $i_1,i_2,\ldots,i_n$ to zamiast zapisu $\bar{X}_i^k$ stosowany będzie zapis $\bar{X}$.

Rozpatrzmy teraz sposób otrzymywania sum postaci /6/. Załóżmy, że dana jest pewna n wymiarowa tablica A, elementy której identyfikowane są za pomocą wektora wskaźników $j = (j_1,j_2,\ldots,j_n) \in J$ takiego, że $d_s \leq j_s \leq g_s$. Elementy tablicy A zapisywać będziemy w postaci

$$A_{j_1,j_2,\ldots,j_n} \quad \text{lub krócej} \quad A_j$$

Dana jest też pewna kombinacja $k = (k_1,k_2,\ldots,k_m)$ z n po m określająca numery tych współrzędnych tablicy A, po których należy dokonać sumowania.
Numery pozostałych współrzędnych tworzą kombinację będącą tzw. uzupełnieniem /dopełnieniem/ kombinacji k, kombinacja ta oznaczona będzie symbolem $k' = (k'_1, k'_2, \ldots, k'_{n-m})$.

Po dokonaniu sumowania po wskazanych współrzędnych otrzymamy podtablicę n-m wymiarową, którą oznaczymy symbolem B.
Tablicę tę określa następujący zbiór wskaźników:

$$I = \left\{(i_1,i_2,\ldots,i_{n-m}): d_{k'_s} \leq i_s \leq g_{k'_s}, s=1,2,\ldots,n-m\right\} \quad /7/$$

Element tablicy B oznaczymy symbolem

$B_{i_1,i_2,\ldots,i_{n-m}}$ lub krócej B_i.

Aby dla danego $i = (i_1,i_2,\ldots,i_{n-m}) \in I$ obliczyć element B_i należy wykonać następujące czynności:

1/ wygenerować zbiór W takich wektorów $w=(w_1,w_2,\ldots,w_m)$ że $d_{k_s} \leq w_s \leq g_{k_s}$,

2/ na podstawie zbioru W utworzyć zbiór J takich wektorów $j = (j_1,j_2,\ldots,j_n)$, że $j_s = w_s$ dla $s = k_1,k_2,\ldots,k_m$ oraz wartości pozostałych n-m składowych są równe wartościom kolejnych składowych wektora $i = (i_1,i_2,\ldots,i_{n-m})$,

3/ dodać do siebie wszystkie elementy tablicy A identyfikowane za pomocą wektorów wskaźników ze zbioru J, otrzymaną sumę traktować jako element $B_{i_1,i_2,\ldots,i_{n-m}}$.

Jeśli czynności 1/-3/ wykonamy dla każdego $i \in I$, to otrzymamy całą tablicę B.

Korzystając z podanego algorytmu łatwo można obliczyć wektor liczebności /częstości/ brzegowych.

Załóżmy, że dla N realizacji zmiennej losowej $X = (X_1,X_2,\ldots,X_n)$ zbudowana została n wymiarowa tablica liczebności T, elementy której identyfikowane są za pomocą wektora wskaźników $j = (j_1,j_2,\ldots,j_n)$ takiego, że $1 \leq j_s \leq n_s$.

Niech $b_r = (b_{r1},b_{r2},\ldots,b_{rn_r})$ oznacza liczebności brzegowe zmiennej X_r. Zgodnie z definicją i-tą składową wektora b_r oblicza się ze wzoru:

$$b_{ri}=\sum_{j_1}\sum_{j_2}\cdots\sum_{j_{r-i}}\sum_{j_{r+i}}\cdots\sum_{j_n} T_{j_1,j_2,\ldots,j_{r-1},i,j_{r+1},\ldots,j_n}$$

Aby otrzymać wektor b_r należy więc dla każdego $i=1,2,\ldots,n$ wykonać czynności 1/-3/ opisanego wyżej algorytmu otrzymywania tablic-kondensatów przyjmując, że $k = (i,2,\ldots,r-i,r+1,\ldots,n)$.
Ponieważ w danym przypadku $k' = r$, to zbiór I określony wzorem /7/ jest następujący: $I = \{1,2,\ldots,n_r\}$.
Niżej rozpatrzone są dwa przykłady zastosowania podanych algorytmów.

4.3.2. Współczynnik zależności

Załóżmy, że na podstawie realizacji zmiennej losowej $X = (X_1,X_2,\ldots,X_n)$, tworzymy tablicę częstości T /według opisanego wyżej algorytmu/. Przyjmijmy dalej, że elementy tej tablicy identyfikowane są za pomocą wektora wskaźników $i = (i_1,i_2,\ldots,i_n)$ takiego, że $1 \leqslant i_s \leqslant g_s$, przy czym zamiast zapisu $T_{i_1,i_2,\ldots,i_n}$ będzie też stosowany zapis T_i.
Dla każdej zmiennej X_s /s=1,2,...,n/ obliczymy częstości brzegowe $p_s = (p_{s,1},p_{s,2},p_{s,3},\ldots,p_{s,g_s})$ gdzie

$$p_{s,j}=\sum_{i_1}\cdots\sum_{i_{s-i}}\sum_{i_{s+1}}\cdots\sum_{i_n} T_{i_1,i_2,\ldots,i_{s-1},j,i_{s+1},\ldots,i_n}$$

dla $j = 1,2,\ldots,g_s$

Na podstawie tablicy T i wektorów częstości brzegowych można obliczyć współczynnik zależności stochastycznej między zmiennymi $X_1, X_2, \ldots, X_n$.

Przyjmijmy następujące uogólnienie współczynnika Hellwiga [32] na przypadek wielowymiarowy:

$$\delta^2 = \frac{1 - \sum_i \min\left(T_i, B_i\right)}{1 - r^{1-n}}$$

gdzie

$$B_i = \prod_{s=1}^{n} p_{s,i_s}$$

$$r = \min\left\{g_1, g_2, \ldots, g_n\right\}$$

Można udowodnić[1], że $0 \leq \delta^2 \leq 1$.

Do obliczenia wartości tego współczynnika wykorzystuje się więc algorytm budowy tablic korelacyjnych oraz algorytm otrzymywania podtablic-kondensatów /wektory częstości brzegowych/. Czas obliczenia zależy głównie od czasu niezbędnego na utworzenie tablicy częstości, a ten z kolei zależy od wielkości liczb $g_1, g_2, \ldots, g_n$. Dla przykładu gdy $n=6$ i $g_s=3$ $/s=1,2,\ldots,n/$ obliczanie wartości δ^2 trwało na maszynie ODRA 1204 ok. 1 godz., gdy zaś $n=6$ oraz $g_s=2$ zaledwie 6 min.

W przypadku gdy chcemy obliczyć współczynnik zależności między zmiennymi losowymi, których numery tworzą kombinację

---

[1] Dowód podany jest w przygotowywanej przez autora specjalnej pracy poświęconej zagadnieniom zależności stochastycznej zmiennych losowych wielowymiarowych.

$k = (k_1, k_2, \ldots, k_m)$, to najpierw na podstawie tablicy T należy obliczyć m-wymiarową podtablicę-kondensat i z nią dalej postępować jak z tablicą T.

4.3.3. Analiza wariancji

Załóżmy, że wyniki obserwacji zależą od n czynników $A_1, A_2, \ldots, A_n$. Jeśli czynnik A_i posiada m_i poziomów /wartości/ to wszystkie obserwacje można przedstawić w postaci tablicy n wymiarowej, oznaczymy ją symbolem X. Elementy tej tablicy oznaczać będziemy symbolem $X_{i_1, i_2, \ldots, i_n}$ lub krócej X_i.

Zadanie analizy wariancji polega między innymi na określeniu efektu działania każdego z czynników oraz wzajemnego ich oddziaływania. Ponieważ czynników jest n, to do przeanalizowania mamy $w = 2^n - 1$ różnych zestawów czynników, odpowiadające im kombinacje oznaczymy symbolami $k_1, k_2, \ldots, k_r, \ldots,$
Analiza wariancji polega na tym, że ogólną zmienność S, tzn. sumę kwadratów odchyleń poszczególnych obserwacji od ogólnej średniej $\bar{X}$ rozbija się na sumę niezależnych składników $S_1, S_2, \ldots, S_w$.

Przy czym składniki $S_1, S_2, \ldots, S_n$ określają odpowiednio wpływ czynników $A_1, A_2, \ldots, A_n$, natomiast składniki S_{n+1}, $S_{n+2}, \ldots, S_w$ określają wzajemne oddziaływanie odpowiednich kombinacji czynników.

Sumę S oblicza się ze wzoru:

$$S = \sum_{i} \left(X_i - \bar{X}\right)^2 \qquad /8/$$

natomiast składnik S_r dla $1 \leqslant r \leqslant w$ oblicza się według wzoru:

$$S_r = D \sum_{i(k_r)} \left(\bar{X}_i^{k'_r} + (-1)^j \sum_{j=1}^{|k_r|} \sum_{k_{r,j}} \bar{X}_i^{k'_r \| k_{r,j}} \right)^2 \qquad /9/$$

gdzie:

k_r - kombinacja o numerze r,

k'_r - uzupełnienie kombinacji k_r tzn. składowymi k'_r są te elementy, które nie weszły do kombinacji k_r ,

$|k_r|$ -długość kombinacji k_r tzn. ilość składowych,

$k_{r,j}$- j-elementowa podkombinacja kombinacji k_r,

$i(k_r)$- te składowe wektora $i = (i_1, i_2, \ldots, i_n)$, których numery tworzą kombinację k_r,

D - ilość zsumowanych elementóe tablicy X, jeśli $i(k_r) = (i_{j_1}, i_{j_2}, \ldots, i_{j_s})$, to

$D = m_{i_{j_1}} \cdot m_{i_{j_2}} \cdot \ldots \cdot m_{i_{j_s}}$,

przy czym symbol $\sum_{k_{r,j}}$ oznacza sumowanie po wszystkich podkombinacjach $k_{r,j}$ oraz symbol $\|$ oznacza operację konkatenacji.

W literaturze poświęconej analizie wariancji rozpatruje się zwykle przypadki co najwyżej 3 czynników. Przykładowo dla n=3 wzory /8/ i /9/ w postaci rozwiniętej podane są w [35], niezbyt formalny zapis tych wzorów dla $n=5$ podany jest w [20].

Postać wzoru /9/ jest bardzo wygodna do zaprogramowania na maszynę cyfrową. Łatwo zauważyć, że przy programowaniu tego wzoru wykorzystuje się algorytmy generowania kombinacji i punktów prostopadłościanu a także algorytmy numerujące te słowa /por. [P31] /.

4.3.4. Planowanie eksperymentów

Przy badaniu skomplikowanych zjawisk /procesów/ fizycznych, chemicznych, gospodarczych itp. często zachodzi potrzeba przeprowadzania eksperymentów. Dokonuje się ich albo w celu wykrycia mechanizmu badanego zjawiska tzn. znalezienia jego modelu matematycznego, albo też w celu określenia warunków przy których dany proces będzie przebiegał optymalnie.

Od niedawna rozwija się specjalna gałąź wiedzy zwana matematyczną teorią planowania eksperymentów optymalnych. Opracowuje się w niej takie metody przeprowadzania eksperymentów, które byłyby jednocześnie i tanie i dostarczały możliwie najwięcej informacji o badanym zjawisku.

Załóżmy, że badane zjawisko można opisać pewnym wielomianem $w(x_1, x_2, \ldots, x_n)$. W celu znalezienia nieznanych współczynników tego wielomianu przeprowadza się eksperymenty.

Przy czym przez eksperyment rozumie się wówczas obliczenie wartości tego wielomianu w pewnym punkcie $x = (x_1, x_2, \ldots, x_n)$. Zbiór wszystkich punktów, w których przeprowadza się eksperymenty nazywa się planem eksperymentu [22,68].

Wiadomo [68], że przy określaniu nieznanych współczynników wielomianu stopnia n, opisującego własności mieszanki o q składowych, optymalnym planem eksperymentu jest regularna siatka na simpleksie /por. [103] /.

Do wygenerowania współrzędnych $x_1, x_2, \ldots, x_q$ węzłów tej siatki wykorzystamy algorytm generowania partycji /por. par. 2.5.2./.

Załóżmy, że mamy następującą partycję liczby n:

$$P = (p_1, p_2, \ldots, p_q),$$

wówczas współrzędne x_i punktu węzłowego regularnej siatki na q-1 wymiarowym simpleksie określić można z zależności /por. [103] /:

$$x_i = \frac{p_i}{n} \qquad i = 1,2,\ldots,q$$

W celu wygenerowania tzw. centroidalnych [68,103] siatek na simpleksie, postąpimy następująco: generujemy kolejno wszystkie kombinacje bez powtórzeń z q po s /s=1,2,...,q/, kombinacji takich jest 2^q-1, tyle też jest punktów węzłowych tzw. siatki centroidalnej.

Mając daną kombinację

$$k = (k_1, k_2, \ldots, k_s)$$

współrzędne punktu $x = (x_1, x_2, \ldots, x_q)$ żądanej siatki otrzymamy z zależności /por. [103] /:

$$x_i = \begin{cases} \frac{1}{s} & \text{dla } i = k_1, k_2, \ldots, k_s \\ 0 & \text{dla pozostałych } i \end{cases}$$

Obie metody zostały zaprogramowane na maszynę ODRA 1204.

4.3.5. Wyszukiwanie informacji

Dany jest pewien skończony zbiór Z. Każdy element tego zbioru scharakteryzowany jest za pomocą wektora cech

$$c = (c_1, c_2, \ldots, c_n)$$

przyjmijmy, że wartością i-tej cechy c_i mogą być liczby $1, 2, \ldots, b_i$.

Zadanie polega na tym, aby zorganizować taką maszynową ewidencję elementów zbioru Z aby:

- wymagała minimalnej ilości komórek pamięci,
- pozwalała na szybkie usuwanie lub dodawanie elementów oraz
- pozwalała na szybki wybór wszystkich elementów posiadających zadane cechy.

Najbardziej praktycznym sposobem przechowywania elementów zbioru Z są tzw. łańcuchy /por. [42,95] /. Łańcuch tworzy się w ten sposób, że do pierwszej połowy komórki przeznaczonej na zapamiętanie elementu $z \in Z$ zapisuje się kod tego elementu np. jego nazwę, zaś do drugiej połowy zapisuje się adres następnej komórki zawierającej element o tych samych cechach. Jeśli takiego elementu brak, to do drugiej połowy

komórki wpisuje się standardowy symbol np. zero.
Usuwania czy też wstawienia nowych elementów dokonuje się wówczas za pomocą tzw. operacji łańcuchowych /por. [42,87] / należy w tym celu ustalić tylko numer łańcucha.
Łańcuchy ponumerujemy tak, aby zachowany był porządek leksykograficzny odpowiadających tym łańcuchom wektorów cech.
W ten sposób otrzymamy następujący ciąg łańcuchów:

$$ł_1, ł_2, \ldots, ł_w$$

gdzie w - ilość wszystkich możliwych realizacji wektora cech tzn.

$$w = \prod_{i=1}^{n} b_i.$$

Numer łańcucha traktować będziemy jako względny adres tej komórki, gdzie rozpoczyna się dany łańcuch.
Jeśli dany jest pewien wektor wartości cech $c=(c_1, c_2, \ldots, c_n)$, to odpowiadający mu numer łańcucha oblicza się za pomocą funkcji α. Po znalezieniu numeru łańcucha łatwo odczytuje się wszystkie jego elementy.

Zadanie nieco się komplikuje, gdy zadane są tylko niektóre spośród n cech. Niech np. dane będą wartości następujących cech:

$$c_{i_1}, c_{i_2}, \ldots, c_{i_m} \qquad /10/$$

Na podstawie tych wartości nie możemy obliczyć jednego numeru łańcucha, gdyż ciągowi /10/ odpowiada już zbiór łańcuchów.

Aby określić numery łańcuchów odpowiadających ciągowi /10/ postępujemy w ten sposób, że generujemy takie wektory $c = (c_1, c_2, \ldots, c_n)$, w których składowe o numerach $i_1, i_2, \ldots, i_m$ są równe zadanym liczbom /10/. Następnie na podstawie tych wektorów za pomocą funkcji α obliczamy numery łańcuchów. Do wygenerowania wektorów o podanej własności korzystamy z algorytmu generowania punktów prostopadłościanu. W tym celu przyjmujemy

$$g_j = d_j = c_j \qquad \text{dla } j = i_1, i_2, \ldots, i_m$$

$$d_j = 1 \qquad \text{dla pozostałych } j \text{ oraz}$$

$$g_j = b_j \qquad \text{dla pozostałych } j.$$

Postępowanie takie znacznie się upraszcza, gdy zadane są początkowe wartości wektora cech, np. gdy są dane

$$c_1, c_2, \ldots, c_m \qquad /11/$$

W takim przypadku należy wygenerować kolejne punkty począwszy od

$$d = (c_1, c_2, \ldots, c_m, 1, 1, \ldots, 1)$$

a skończywszy na

$$g = (c_1, c_2, \ldots, c_m, b_{m+1}, b_{m+2}, \ldots, b_n)$$

Punkty te w porządku leksykograficznym zajmują kolejne miejsc dlatego wystarczy obliczyć numer pierwszego i ostatniego z nich. Niech to będą odpowiednio numery n_1 i n_2, wówczas numery łańcuchów odpowiadające zadanemu ciągowi /11/ będą następujące:

$$n_1, n_1+1, \ldots, n_2$$

Do otrzymania tych wartości wystarczy więc wygenerować dwa punkty i dwa razy obliczyć wartości funkcji α.

5. PERSPEKTYWY METOD KOMBINATORYCZNYCH

5.1. U w a g i w s t ę p n e

Wszystkie metody kombinatoryczne /niezależnie od innych podziałów/ podzielić można na dokładne i przybliżone. Do metod dokładnych zalicza się zwykle takie metody, które w skończonej liczbie kroków dają rozwiązania dokładne, wszystkie pozostałe metody nazwiemy przybliżonymi.

Brak metod dokładnych lub ograniczone możliwości ich stosowania wymuszają poszukiwanie metod przybliżonych. Najważniejsze z nich są te, które gwarantują monotoniczne polepszanie wyniku wraz ze wzrostem liczby kroków. Dla zapewnienia tej własności zwykle żąda się aby rozwiązywane zagadnienie spełniało pewne warunki. W praktyce często jednak warunki takie albo nie są spełnione, albo też jest bardzo trudno sprawdzić czy one zachodzą. Stąd też ostatnio obserwuje się coraz większe zainteresowanie przybliżonymi metodami heurystycznymi. Podstawę tych metod stanowi tzw. wnioskowanie prawdopodobne [72] bazujące na analogii oraz indukcji /por. [58,77,78]/ i korzystające ze starej zasady prób i błędów /por. [51] /. Próby /eksperymenty/ dokonywane są albo według pewnych reguł heurystycznych, albo też losowo. Okazało się przy tym, że zastosowanie idei randomizacji jest bardzo efektywne /por. [36,61,98] /.

Metody heurystyczne, po raz pierwszy zastosowano na początku lat sześćdziesiątych do rozwiązywania takich problemów jak gra w szachy, dowodzenie twierdzeń itp./por [101] /.

Ostatnio metody te w połączeniu z metodami kombinatorycznymi w postaci tzw. systemów heurystycznej samoorganizacji[1] stosowane są do rozwiązywania zagadnień z praktyki gospodarczej, między innymi wykorzystywane są do sterowania tzw. systemami wielkimi [27,52], a w szczególności systemami ekonomiczno-społecznymi.

Jedną z cech charakterystycznych większości metod heurystyczno-kombinatorycznych przeznaczonych do rozwiązywania zagadnień ekstremalnych jest to, że nie gwarantują one monotonicznego polepszania szukanego rozwiązania. Stąd też powstaje podstawowy problem kiedy obliczenia należy przerwać. Spośród różnych dotychczas opracowanych r e g u ł s t o p u [17,104] głównie o charakterze statystycznym lub teoriogrowym [79] warto zwrócić uwagę na heurystyczną metodę o nazwie Las Vegas [1] , która krótko scharakteryzowana jest w następnym paragrafie danego rozdziału.

5.2. M e t o d a L a s V e g a s

Załóżmy, że chcemy znaleźć taki element $x_0 \in X$, dla którego pewna funkcja $f(x)$ osiąga wartość maksymalną. Dla osiągnięcia tego celu można postąpić następująco: ze zbioru X wybieramy ciąg punktów:

$$x_1, x_2, \ldots, x_n$$

dla których obliczamy wartości:

$$f(x_1),\ f(x_2), \ldots,\ f(x_n)$$

---

[1] Por. [36] oraz paragraf 5.4 danego rozdziału.

i jako x_0 przyjmujemy taki punkt x_{i_0}, że $f(x_{i_0})=\max_i f(x_i)$. Wiadomo, że im więcej punktów badamy, tym większe jest prawdopodobieństwo tego, że znajdziemy punkt optymalny x_0. Dlatego też nasuwa się pytanie: czy nie warto jeszcze zbadać wartości funkcji w punkcie x_{n+1}? Odpowiedzi udziela tzw. metoda Las Vegas podana w [1] .Idea tej metody polega na tym, że otrzymany zbiór par $(x_i, f(x_i))$ $i=1,2,\ldots,n$ przedstawiamy na płaskim wykresie otrzymując pewien zbiór /chmurę/ punktów na płaszczyźnie. Chmurę tę ograniczamy od góry możliwie gładką funkcją $g(x)$ /np. ręcznie/, a następnie funkcję tę ekstrapolujemy do punktu x_{n+1}. Jeśli okaże się, że przyrost funkcji $g(x)$ w punkcie x_{n+1} jest mniejszy od zadanej liczby K, to obliczenia należy przerwać. Oznacza to, że koszt zbadania wartości funkcji $f(x)$ w kolejnym n+1-szym punkcie jest już większy od spodziewanego polepszenia wyniku.

Nietrudno zauważyć, że w metodzie tej czas niezbędny dla osiągnięcia takiego punktu, w którym dalsze polepszenie rozwiązania jest już nieopłacalne, silnie zależy od sposobu generowania punktów, w których oblicza się wartość optymalizowanej funkcji.

Do chwili obecnej nie opracowano jeszcze żadnych efektywnych metod dotyczących kolejności generowania tych punktów. W sytuacjach takich proponuje się zwykle zbadać kilka różnych wariantów. Podstawowym problemem jest wówczas zagwarantowanie "różności" wariantów. W przypadku gdy zbiór X jest pewnym zbiorem słów kombinatorycznych, to do spełnienia

tego warunku można wykorzystać algorytmy rozpatrzone w rozdziale 2 oraz 3, które pozwalają generować słowa w dowolnej kolejności.

5.3. R a n d o m i z a c j a

Ideę wykorzystania randomizacji do przybliżonego rozwiązywania zagadnień matematycznych po raz pierwszy / według [39] / zastosował A.Hall w 1873 roku do oszacowania wartości liczby π.

Następnie w 1949 roku sformułowano ją / N. Metropolis i S. Ulam / w postaci tzw. metody Monte Carlo również do rozwiązywania zagadnień polegających na szacowaniu pewnych wielkości matematycznych / głównie całek oznaczonych /.

W latach pięćdziesiątych idea randomizacji zastosowana została do rozwiązywania zagadnień ekstremalnych, przy czym takich zagadnień, dla których nie jest znana postać analityczna funkcji celu. Wartość tej funkcji można jedynie mierzyć w dowolnym punkcie dla przeważnie z pewnym błędem. Do rozwiązania podobnych zagadnień stosuje się metody opracowane w teorii doświadczeń ekstremalnych [22,67] oraz teorii aproksymacji stochastycznej [56,98].
Pierwszą grupę metod zapoczątkowali w 1951 roku Box i Wilson, natomiast drugą - Robbins i Monro, również w 1951 roku.
W chwili obecnej istnieje już bogata literatura dotycząca obu grup metod / por.[61,67,84,98] /.

Nieco odmienną grupę metod stanowią.tzw. sekwencyjne metody będące stochastycznym odpowiednikiem metod gradientowych posiadające również bogatą bibliografię /por. [76,81,84, 104] /.

Wydaje się, że warto zwrócić uwagę na stochastyczny odpowiednik metody branch-and-bound. Metoda ta /według [64] / zaproponowana została w 1965 roku przez matematyka radzieckiego A.A.Feldbauma.

Idea tej metody polega na tym, że zbiór rozwiązań dopuszczalnych D dzieli się na dwa rozłączne podzbiory D_0 i D_1. Następnie jeden z tych podzbiorów odrzuca się jako gorszy. Natomiast drugi z nich, gdzie "osacza" się rozwiązanie optymalne, dalej dzieli się na dwa podzbiory itd., aż otrzymamy podzbiór dostatecznie mały. W podzbiorze tym rozwiązanie optymalne może być odszukane poprzez zbadanie wszystkich możliwości lub np. stosując jakikolwiek algorytm regularny lub wykorzystujący metodę prób i błędów.

Jako kryterium wyboru jednego spośród zbioru D_0 i D_1 przyjmuje się wartość oczekiwaną M_i najmniejszej wartości z N elementowej próbki, przy czym próbkę stanowi N wartości funkcji celu f(x) przy losowo wybranym argumencie $x \in D_i$,

Aby wartość można było oszacować konieczna jest znajomość rozkładu wartości funkcji. Jeśli argument jest wybrany losowo z rozkładem jednostajnym, to dla większości zagadnień optymalizacyjnych wartość funkcji celu ma rozkład logarytmiczno-normalny /por. [64] /.

Po oszacowaniu wartości M_0 i M_1 wybiera się ten podzbiór, dla którego ta wartość jest mniejsza / w przypadku szukania minimum/.

Cechą charakterystyczną wszystkich wymienionych tu metod statystycznych jest to, że są one bardzo proste w realizacji maszynowej i słabo reagują na zakłócenia losowe. Poza tym nie zależą one od wymiaru zagadnienia, co jest jednym z podstawowych problemów metod deterministycznych. Godnym uwagi jest również ten fakt, że metody statystyczne pozwalają w łatwy sposób zastosować proces samouczenia się i adaptacji /por. [16,36,105] /.

5.4. S y s t e m y h e u r y s t y c z n e j s a m o - o r g a n i z a c j i

Metody kombinatoryczne /polegające na przeliczeniu określonych wariantów/ stanowią podstawę rozwijanych ostatnio różnych procedur naśladujących zdolności adaptacyjne i ewolucyjne żywych organizmów. Procedury takie realizowane są albo w postaci specjalnych urządzeń technicznych, albo też w postaci programów maszynowych.

Pierwszym urządzeniem tego typu jest homeostat Ashby'ego [6,18], który prócz pewnych walorów teoretycznych, większego znaczenia praktycznego nie posiada. Znaczenie takie posiadają tzw. systemy heurystycznej samoorganizacji [36] . Są to albo urządzenia techniczne, albo programy maszynowe służące do rozwiązywania takich zagadnień cybernetycznych jak rozpoznawanie obrazów, prognozowanie, identyfikacja charakterystyk obiektów sterowania itp.

Zagadnienia tego typu sprowadzają się do pewnego zagadnienia interpolacyjnego polegającego na tym, że na podstawie niewielkiej ilości informacji należy odtworzyć pewną nieznaną funkcję $f(x)$. Funkcję tę, na podstawie wartości w pewnych punktach, aproksymuje się zwykle tzw. wielomianem Kołmogorowa-Gabora [36] .

Pierwszym systemem heurystycznej samoorganizacji jest perceptron Rosenblatta [63] , oprócz niego opracowano szereg innych systemów /por. [36,48,101] /, wymienimy tu tylko system Iwachnienki o nazwie MGUA /metoda grupowego uwzględnienia argumentu/. Procedury tego systemu są na wskroś kombinatoryczne, a poza tym bardzo jaskrawo wykorzystywana jest w nim idea agregacji /co zresztą widoczne jest w nazwie systemu/, dlatego też niżej krótko przedstawimy koncepcję tego systemu /por. [36] /.

Załóżmy, że dany jest pewien ciąg punktów /węzłów interpolacji/:

$$x^1, x^2, \ldots, x^N, x^{N+1}, \ldots, x^{N+m}, \qquad /1/$$

gdzie $x^i = (x_1^i, x_2^i, \ldots, x_n^i)$.

Pierwszych N punktów ciągu /1/ nazywa się ciągiem uczącym lub treningowym, zaś następnych m punktów stanowi tzw. ciąg testujący.

W każdym z punktów ciągu /1/ znana jest wartość pewnej nieznanej funkcji $y = f(x_1, x_2, \ldots, x_n)$, tzn. dany jest też ciąg

$$y_1 = f(x^1), \ldots, y_{N+m} = f(x^{N+m}) \qquad /2/$$

Zadanie polega na tym, aby na podstawie tych danych odtworzyć funkcję $f(X)$.

Przyjmijmy, że nieznana funkcja posiada następującą postać:

$$y = f(x_1,x_2,\ldots,x_n) = A_0 + A_1x_1 + \ldots + A_n.x_n \qquad /3/$$

Nietrudno zauważyć, że każdy wielomian stopnia p o q zmiennych

$$z = \sum a_{i_1,i_2,\ldots,i_t} x_1^{i_1} x_2^{i_2} \ldots x_q^{i_t} \qquad /i_1+i_2+\ldots+i_t \leq p/$$

można sprowadzić do postaci /3/.

Wprowadzając mianowicie oznaczenia $A_k = a_{i_1 i_2,\ldots,i_t}$ oraz $x'_k = x_1^{i_1} x_2^{i_2} \ldots x_q^{i_t}$ otrzymujemy formę

$$z' = \sum_{k=1}^{M} A_k \cdot x'_k \quad \text{gdzie} \quad M = C^p_{p+q} \quad /\text{por. } [36] /.$$

Aby według algorytmu metody MGUA oszacować współczynniki A_i we wzorze /3/ postępujemy następująco:

Na podstawie funkcji /3/ tworzymy następujący ciąg funkcji "częściowych":

$$z_1 = f(x_1,x_2),\ z_2 = f(x_1,x_3),\ldots,\ z_r = f(x_{n-1},x_n) \qquad /4/$$

gdzie $r = C^2_n$.

Każda z funkcji ciągu /4/ posiada taką samą postać jak funkcja /3/. Współczynniki tych funkcji szacujemy zwykłymi metodami regresji liniowej [56] . Następnie dla każdej

funkcji z ciągu /4/ obliczamy wielkość błędu d_k^2 na podstawie wzoru:

$$d_n^2 = \frac{1}{m} \sum_{i=1}^{m} \left(y_{i+N} - z_{k,i+N} \right)^2, \quad k = 1,2,\ldots,C_n^2, \qquad /5/$$

gdzie $z_{k,N+i}$ jest to warzość funkcji z_k w i-tym punkcie ciągu testującego.

Wielkości obliczonych błędów służą do wyboru kilku[1] najlepszych funkcji z_i do dalszych obliczeń. Załóż.y, że wybrane pięć kolejnych funkcji z_1, z_2, z_3, z_4, z_5, na podstawie tych zmiennych tworzymy następny ciąg funkcji częściowych:

$$v_1 = f(z_1, z_2),\ v_1 = f(z_1, z_3), \ldots,\ v_{10} = f(z_4, z_5) \qquad /6/$$

Spośród tego ciągu dalej wybieramy kilka najlepszych i procedurę powtarzamy do chwili, gdy otrzymywane błędy d_k^2 .przestają wyraźnie się zmniejszać. Załóżmy, że w ostatnim kroku spośród pewnego ciągu:

$$w_1 = f(u_1, u_2),\ w_2 = f(u_1, u_3), \ldots,\ w_r = f(u_{t-1}, u_t) \qquad /7/$$

wybrano jedną najlepszą funkcję, niech to będzie funkcja w_1, której postać jest następująca:

$$w_1 = D_0 + D_1 \cdot u_1 + D_2 \cdot u_2$$

Stanowi ona pierwsze równanie w układzie, z którego wyznacza się współczynniki równania /3/. Następne dwa równania są to kombinacje liniowe dla u_1 oraz u_2. Równania określające u_1 oraz u_2 wyznaczają dalsze równania, te z kolei jeszcze

---

[1] Do określenia liczby wybieranych funkcji w pracy [36] zaproponowano kilka metod polegających na badaniu różnych wariantów.

wcześniejsze, aż dojdziemy do równań, w których występują zmienne $x_1, x_2, \ldots, x_n$.

Eliminując z otrzymanego w ten sposób układu równań zmienne przejściowe $z, r, \ldots, u, w$ otrzymujemy pewną formę:

$$w_1 = B_0 + B_1 \cdot x_1 + \ldots + B_n x_n ;$$

będącą analogonem formy /3/.

Z podanych w pracy [36] przykładów wynika, że opisana wyżej procedura jest efektywnie stosowana do rozwiązywania szerokiej klasy zagadnień interpolacyjnych.

Warto też zauważyć, że procedury heurystycznej samoorganizacji znajdują coraz szersze zastosowanie przy sterowaniu systemami socjalno-ekonomicznymi, w szczególności przy planowaniu optymalnym.

Wynika to z tego, że stosowane dotychczas klasyczne metody matematyczne okazały się niewystarczające chociażby z tego względu, że metody ilościowe nie są w stanie w pełni odzwierciedlić aktywnego i twórczego charakteru planowania ekonomicznego /por. [101] str.9/. Poza tym probelmu optymalnego planowania nie da się w żaden sposób sprowadzić do jakiegoś zagadnienia ekstremalnego, gdyż ustalenie jednego kryterium optymalności jest niemożliwe [21] . Stąd też wydaje się, że proces planowania optymalnego powinien przebiegać według ogólnej zasady systemów heurystycznej samoorganizacji tzn. powinien polegać na wielokrotnym generowaniu różnych kombinacji wariantów oraz wyborze planów w określonym sensie najlepszych, w ostatnim kroku dokonuje się wyboru planu "średnio-optymalnego" /por. [21] str.100/.

Weźmy dla przykładu prosty model macierzowy przepływów międzygałęziowych. Podstawowe równanie tego modelu jest następujące

$$X = A \cdot Y$$

gdzie

X - wektor produkcji globalnej,

Y - wektor produktu końcowego,

A - zadana macierz współczynników taka, że $A = (I-a)^{-1}$, przy czym a - macierz współczynników nakładów bezpośrednich.

Przyjmijmy, też taki wariant planowania, gdy za punkt wyjścia przyjmuje się wielkość i strukturę produktu globalnego, a oblicza się wektor produktu końcowego.
W sposób eksperymentalny można ustalić dolne i górne granice dla każdej składowej wektora $X = (x_1, x_2, \ldots, x_n)$ takie, że dla każdego wektora X spełniającego warunek:

$$d_i \leq x_i \leq g_i \qquad i=1,2,\ldots,n \qquad /8/$$

otrzymamy taki wektor $Y = (y_1, y_2, \ldots, y_n)$, który zapewnia racjonalną strukturę dochodu narodowego.

Mając ustalone wektory D i G można przystąpić do generowania różnych planów. W tym celu wystarczy odpowiednio skwantować przedziały $[d_i, g_i]$ i generować wszystkie wektory

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

spełniające warunek /8/.

Dla każdego z tych wektorów obliczamy wektor $Y=(y_1 y_2,\ldots,y_n)$, otrzymując w ten sposób zbiór różnych planów.

Każdy wektor Y można interpretować jako pewien punkt w przestrzeni n-wymiarowej. Po otrzymaniu wszystkich punktów Y, można przeprowadzić dla nich analizę taksonomiczną otrzymując w ten sposób pewne klasy planów podobnych /równoważnych/.

Bez konieczności ustalenia żadnego kryterium ilościowego można więc wybrać taki plan, który zapewnia najbardziej pożądaną strukturę dochodu narodowego[1].

---

[1] Podejście powyższe zaproponował Z.Hellwig na jednym ze swoich seminariów.

ZAKONCZENIE

Formalne metody rozwiązywania różnych zagadnień nazywane dziś algorytmami, poszukiwane były już w starożytności. Samo pojęcie algorytm pochodzące od nazwiska matematyka arabskiego Muhammeda ibn al-Khwarizmi żyjącego w IX wieku, w literaturze pojawiło się dopiero w XII wieku /z łacińskiego przekładu dzieł tego matematyka/.

Od samego początku swego rozwoju, algorytmy zapisywane były w bardzo różnorodny sposób. Z chwilą pojawienia się maszyn matematycznych powstała konieczność opracowania jednolitego sposobu ich zapisu.

Historycznie pierwszym sformułowanym i ujednoliconym sposobem zapisu algorytmów były tzw. schematy blokowe [63] , następnie pojawiły się języki algorytmiczne [35] , ostatnio stosowane są również tzw. tablice decyzyjne [58] .

W pracy niniejszej nie skorzystano jednak z żadnej z tych form zapisu algorytmów, gdyż zapis w języku algorytmicznym mimo swej ścisłości jest mało czytelny, natomiast schematy blokowe czy też tablice decyzyjne w przypadku rozpatrywanych w pracy algorytmów zawierających dużo predykatów są bardzo szczegółowe i również mało czytelne.
Poza tym przy rozpatrywaniu algorytmów starano się podać nie tylko ich opis lecz również i sposób ich konstrukcji pozwalający na tworzenie nowych algorytmów.

Niezależnie jednak od werbalnego zapisu stosowanego w pracy, wszystkie algorytmy, jak już wspomniano w wprowadzeniu, zapisane są w określonym języku algorytmicznym i popraw-

ność ich została sprawdzona na maszynie ODRA-1204 "maszyny matematyczne okazały się bowiem bezwzględnym filtrem dla wszelkich błędnych pomysłów" [75] .

Należy jednak zauważyć, że sam fakt możliwości realizacji danego algorytmu na maszynie, nie świadczy jeszcze o jego przydatności praktycznej /jeśli nic nie wiadomo o czasie jego realizacji/.

W chwili obecnej bowiem, zagadnienie teoretycznej rozstrzygalności problemu jest coraz bardziej wypierane problemem praktycznej realizowalności danego algorytmu określonymi środkami i w określonym czasie /por. [45] str.228 /.

Większość zagadnień ekstremalnych o charakterze kombinatorycznym /dyskretnym/ teoretycznie jest rozstrzygalna, mimo to dla wielu z nich albo brak jest efektywnych algorytmów praktycznego rozwiązania, albo też istniejące klasyczne metody programowania matematycznego nie mogą być w rozsądnym czasie zrealizowane nawet przy użyciu najbardziej nowoczesnych środków techniki obliczeniowej.

Stąd też obserwuje się coraz większe zainteresowanie nieklasycznymi metodami głównie o charakterze kombinatorycznym. O wielkiej przydatności praktycznej takich metod świadczą wyniki wielu opublikowanych prac /por. np. [36,76] / jak również wyniki prac prowadzonych w Instytucie Metod Rachunku Ekonomicznego.

Wydaje się, że przedstawione wyniki w rozdziale 4 świadczą również o dużej praktycznej przydatności metod kombinatorycznych do rozwiązywania skomplikowanych zagadnień rachunku ekonomicznego, statystyki matematycznej i ekonometrii.

W danej pracy ograniczono się jednak tylko do rozpatrzenia zagadnień dotyczących stosunkowo wąskiej klasy obiektów kombinatorycznych,a mianowicie słów kombinatorycznych.

Oprócz nich istnieje dużo szersza klasa obiektów zwanych systemami incydencji /por. [130,88]/ posiadającymi nie tylko znaczenie teoretyczne lecz i duże możliwości zastosowań praktycznych: planowanie doświadczeń, kodowanie informacji itp. Dlatego też wydaje się, że należy zwrócić dużą uwagę na zagadnienie algorytmizacji tych obiektów /macierze Hadamarda, kwadraty łacińskie, bloki planowania itp./ jak również na zagadnienie rozwoju i zastosowań metod kombinatorycznych o charakterze probabilistycznym.

Tak więc zagadnienia przedstawione w niniejszej pracy stanowią tylko pewien fragment szerokiej problematyki, którą można określić mianem kombinatoryczne problemy wyznaczania ekstremum funkcji wielu zmiennych.
Problematyka ta znajduje się obecnie w centrum uwagi zarówno osób zajmujących się informatyką, jak i programowaniem matematycznym.

B i b l i o g r a f i a

1. Ackoff R., M.Susieni - Osnowy issledowanija opieracyj, Moskwa 1971

2. Aleksandrow P.S. - Lekcji po analiticeskoj geometrii, Moskwa 1968

3. Algorytmy /51-100/, Moskwa 1966

4. Algorytmy /101-150/, Moskwa 1967

5. Algorytmy i programy słuczajnogo poiska /pr.zbiorowa/, Ryga 1969

6. Ashby R. - Konstrukcja mozga, Moskwa 1964

7. Balas E. - Additiwnyj algorytm dla reszenija zadacz liniejnogo programirowanija z peremiennymi prinimajuszczymi znaczenija o ili 1, Kiberneticzeskij sbornik Nr 6, 1969

8. Bojarinow A.T., Kafarow W.W. - Metody optimizacji w chimiczeskoj technologii, Moskwa 1969

9. Budzyński S., Chorobiński A. - Matematyczne metody optymalizacyjne w projektowaniu zakładów przemysłowych zagadnienie obciążenia maszyn i urządzeń, Warszawa 1968

10. Bukietyński W. - O pewnym zagadnieniu optymalizacyjnym, Prace Naukowe WSE 35/1972

11. Bukietyński W. - Problemy optymalizacji dyskretnej /monografia przygotowywana do druku/.

12. Bukietyński W. Hellwig Z. Królik U. Smoluk A. - Uwagi o dyskryminacji zbiorów skończonych, Prace Naukowe WSE 21/1969

13. Burkow W.N. Łowiecki S.E. - Metody reszenija ekstremalnych kombinatorynch zadacz, Izw.AN SSSR Technicz.Kibernetika 1968,Nr 4

14. Courant E.
Robbins H. - Co to jest matematyka, Warszawa 1967

15. Cournot A. - Osnowy teorii szansow i wierojatnostej, Moskwa 1970

16. Cypkin J.Z. - Osnowy teorii obuczejuszczychsja sistem, Moskwa 1970

17. Cypkin J.Z. - Adaptacja i obuczenije w awtomaticzeskich sistemach, Moskwa 1968

18. Czerwiński Z. - Matematyka na usługach ekonomii, Warszawa 1969

19. Demidowicz B.P.
Moron S.A. - Osnowy wyczislitelnoj matematiki, Moskwa 1966

20. Dugue D. - Teoreticzeskaja i prikładnaja statistika, Moskwa 1972

21. Elementy rachunku ekonomicznego /pod red.Z.Hellwiga/, Wrocław 1970

22. Fedorow W.W. - Teorija optymalnego eksperimenta, Moskwa 1971

23. Fachsmeyer I. - Kombinatorik, Berlin 1969

24. Florek K.,
Łukaszewicz J.
Perkal J.
Steinhaus H.
Zubrzycki S. - Taksonomia wrocławska, Przegląd Antropologiczny 27/1951

25. Ford L.
Fulkerson D. - Przepływy w sieciach, Warszawa 1969

26. Freemann R.I. - Computational experience with a Balasian integer programming algorithm, O.R. 5, 1966

27. Głuszkow W. - Wstęp do cybernetyki, Warszawa 1967

28. Gołowina L. - Liniejnaja algebra i niekotoryje prilożenija, Moskwa 1971

29. Grzegorczyk A. - Zarys logiki matematycznej, Warszawa 1961

30. Hall M. - Kombinatoryka, Moskwa 1970

31. Hellwig Z. - Elementy rachunku prawdopodobieństwa i statystyki matematycznej, Warszawa 1970

32. Hellwig Z. - On the measurement of stochastical dependence. Zastosowania Matematyki X. 1969

33. Hellwig Z. - Problem optymalnego wyboru predyktant, Przegląd Statystyczny, 3/4, XVI,1969

34. Hellwig Z.
Królik U.
Smoluk A. - Niektóre numeryczne zagadnienia e-penetracji złoża, Prace Naukowe WSE 12/1968

35. Huitson A. - Dispersionnyj analiz, Moskwa 1971

36. Iwachnienko A.G. - Sistemy ewristiczeskoj samoorganizacji w techniczeskoj kibernetike,Kijów 1971

37. Jasiński R. - Metoda wyznaczania środka miedzi, Prace Naukowe WSE 34/1972

38. Jakutawiczene B.I. - Algorytmy dla reszenija zadaczi o kratczojszem sojedinitwni toczek na płostasti w pracy zbiorowej: Samonastraiwejuszczijesja sistemy, Moskwa 1965

39. Jermakow S.M. - Metod Monte Carlo i smieżnyje woprosy, moskwa 1971

40. Kac N.
Ulam S. - Matematika i logika, Moskwa 1971

41. Kemeny J.
Snell J.
Thompson G. - Wwidienije w konieczнuju matematiku, Moskwa 1965

42. Kitov A.E. - Programmirowanije informacionno-logiczeskich zadacz, Moskwa 1967

43. Knödel W. - Graphentheoretische Methoden and ihre Anwendungen,Springer-Verlag 1969

44. Korbut A.A.
Finkelstein J.J. - Diskretnoje programmirowanije, Moskwa 1969

45. Korte B.
Krelle W.
Oberhofer W. - Ein lexikografischer Suchalgorithmus zur Lösung allgemeiner ganzahliger. Programmiernngsanfgeben Ufo Nr 2, 1969

46. Królik U.
Smoluk A. - Algorytm wyznaczania najkrótszej drogi, Prace Naukowe WSE 21/1969

47. Kryński H.E. - Matematyka dla ekonomistów, Warszawa 1966

48. Kulikowski J. - Cybernetyczne układy rozpoznające, Warszawa 1972

49. Kuzowkin A.I. - Obobszczenije poiska Fibonacci na mnogomernyj słuczaj, Ekonomika i matematiczeskije metody, 6/1968

50. Lawler E.L.
Bell M.D. - A method for solving discrete optimization problems, O.R.Nr 6,14 1966

51. Ledly R.S. - Programmirowanije i ispolzowanije cyfrowych wyczislitelnych maszyn, Moskwa 1966

52. Lerner A.J. - Zarys cybernetyki, Warszawa 1971

53. Lichtenstein W.E. - Modeli diskretnego programmirowanija, Moskwa 1971

54. Little J.D.C.
Murty K.G.
Sweeney D.W.
Karel C. - An algorithm for the traveling salesman problem, O.R. 11, 1963

55. Lawrow S.S.
Gonczarowa L.I. - Awtomaticzeskaja obrabotka dannych, Moskwa 1971

56. Mańczak K. - Metody identyfikacji wielowymiarowych obiektów sterowania, Warszawa 1971

57. Matematiczeskije metody reszenija ekonomiczeskich zadacz, /pod red.Bagrinowskiego A.K./, Nowosybirsk 1971

58. Matematika i kibernetika w ekonomikie /słownik-poradnik/ Moskwa 1971

59. Matematyka w świecie współczesnym, Warszawa 1966

60. Mazurkiewicz A. - Matematyka w przetwarzaniu informacji, Materiały sympozjum, Zakopane 1968

61. Metody statisticzeskoj optymizacji, Ryga 1968

62. Mietody optymizacji awtomaticzeskich sistiem /pod red.J.Z. Cypkina/, Moskwa 1972

63. Minsky M. Papert S. - Perceptrony, Moskwa 1971

64. Mockus - Woprosy optimizacji w niekotorych mnogoekstremalnych zadaczach swjezanych s wyborom optimalnogo warianta złożnych sistem. Samonastraiwajuszczujesja awtomaticzeskije sistemy,Moskwa 1965

65. Mostowski A.W. Pawlak Z. - Logika dla inżynierów,Warszawa 1970

66. Nalimow W.W. - Teorija eksperimenta, Moskwa 1971

67. Nalimow W.W. Czernowa N.A. - Statystyczne metody planowania doświadczeń ekstremalnych, Warszawa 1967

68. Nowyje idei w planirowanii eksperimenta, /pod red. Nalimowa W.W./, Moskwa 1969

69. Neumann J. - Teorija samowosproizwodiaszczychsja awtomatow, Moskwa 1971

70. Ore C. - Teorija grafow, Moskwa 1968

71. Ostasiewicz W. - Operowanie tablicami wielowymiarowymi. Informatyka i zarządzanie w przedsiębiorstwie przemysłowym /pod red. Z.Hellwiga/, Wrocław 1972

72. Ostasiewicz W. - Numerowanie permutacji, Prace Naukowe WSE Wrocław 33/1972

73. Ostasiewicz W. - Efektywność programowania dynamicznego, Prace Naukowe WSE Wrocław 33/1972

74. Ostasiewicz W. - Liczby S_n^p i ich zastosowanie w planowaniu doświadczeń ekstremalnych, Prace Naukowe WSE /w druku/

75. Pawlak Z. - Gramatyka i matematyka, Warszawa 1965

76. Perwozwański A.A. - Poisk, Moskwa 1970

77. Polya G. - Jak to rozwiązać, Warszawa 1964

78. Polya G. - Matematika i prawdopodobnyje rassużdżenija, Moskwa 1957

79. Prikładnaja kombinatornaja matematika, /pod red.Beckenbacha E./, Moskwa 1968

80. Prim R. - Kratczajszyje swjazywajuszczije sieti i niekotoryje obobszczenija, Kiberneticzeskij sbornik Nr 2, 1961

81. Problemy statisticzeskoj optimizacji, Ryga 1968

82. Rademacher H. Toeplitz O. - O liczbach i figurach, Warszawa 1956

83. Rasiowa H. - Wstęp do matematyki współczesnej, Warszawa 1968

84. Rastrigin A.A. - Statisticzeskije metody poiska, Moskwa 1968

85. Recent Progress in Combinatorics /ed. by W.T.Tutte/, Academic Press, 1969

86. Riordan J. - An intruduction to combinatorial analysis, J.Wiley 1958

87. Rybnikow K.A. - Wwiedenije w kombinatornyj analiz, Moskwa 1972

88. Ryser H. - Kombinatornaja matematika, Moskwa 1966

89. Sadowski W. - Teoria podejmowania decyzji, Warszawa 1964

90. Sebestyen G.S. - Decision-making processes in pattern recognition, New York 1962

91. Sobol I.M. - Metod Monte Carlo, Moskwa 1972

92. Tiepłow L. - Czto sczitat', Moskwa 1970

93. Trybuś G. - Zastosowanie zmiennej losowej dystansowej do przybliżonego rozwiązania problemu komiwojażera, Prace Naukowe WSE 35/1972

94. Trybuś G. - Algorytmy porządkowania dendrytowego, Prace Naukowe WSE 12/1968

95. Turski W. - Podstawy użytkowania maszyn cyfrowych, Warszawa 1968

96. Ulam S. - Nierieszennyje matematiczeskije zadaczi, Moskwa 1964

97. Warężak L. - Optymalne ciągi operacji, Prace Naukowe WSE /w druku/

98. Wilde D. - Metody poiska ekstremuma, Moskwa 1967

99. Wilenkin N.J. - Kombinatoryka, Moskwa 1969

100. Worobjew N.N. - Czisła Fibonacci, Moskwa 1969

101. Wyczislitelnyje maszyny i myszlenije /pod red. E.Feigenbauma i J.Feldmana/, Moskwa 1967

102. Zając K. - Zarys metod statystycznych, Warszawa 1971

103. Zetginidze I.G. - Matematiczeskoje płanirowanije eksperimenta dla issledowanija i optimizacji swojestw smiesiej, Tbilisi, 1971

104. Zieliński R. - Metody Monte Carlo, Warszawa 1970

105. Zujew A.K. - Samonastrojka w technike i żywoj prirode, Ryga 1964

Wykaz procedur i programów

1. genkom - procedura generowania kombinacji
2. numkon - procedura obliczania numeru kombinacji
3. konwenum - procedura generowania kombinacji według jej numeru
4. punkt - procedura generowania punktów prostopadłościanu
5. numpunkt - procedura obliczania numeru punktu
6. punwenum - procedura generowania punktu według numeru
7. permutacja - procedura generowania permutacji według jej numeru
8. nrp - procedura obliczania numeru permutacji
9. Fibonacci - procedura minimalizacji funkcji jednomodalnej
10. simlat - procedura generowania siatek na simpleksie n-wymiarowym
11. elipsoida - procedura generowania punktów elipsoidy n-wymiarowej
12. gepol - program generowania punktów zbioru wypukłego, brzegu tego zbioru oraz otoczki wypukłej /por.2.3/
13. równanie - program generowania rozwiązań równania

$$a_1x_1 + \ldots + a_nx_n = Z$$

14. Steiner - procedura znajdowania takiego $x \in R^n$, że suma odległości tego punktu od danych punktów jest minimalna
15. Predo - program optymalnego wyboru predyktant
16. predu - program przybliżonego znajdowania optymalnej kombinacji predyktant /por. 4.2.1/
17. kolejność - program generujący wszystkie warianty kolejności obróbki n detali na m maszynach

18.* dyna - program dyskretnego programowania dynamicznego

19. partycje - program generujący partycje liczby naturalnej

20. par-kom - procedura przekształcania partycji na kombinację

21. kom-por - procedura przekształcania kombinacji na partycję

22.* bukza - program generujący zbiór Bukietyńskiego /por.4.2.2/

23. parset - program generowania partycji zbioru

24. dopunkt - procedura znajdowania punktu dopuszczalnego w zbiorze określonym warunkami $g_j(x) \geq 0$ gdzie $g_j(x)$ - funkcje wypukłe

25. Mesdeh - program obliczenia współczynnika zależności

26. Dendryt Steinera - program przybliżonego znajdowania najkrótszego dendrytu Steinera

27. butako - program budowy tablic korelacyjnych

28. kondensat - procedura budowy podtablic-kondensatów /por. 4.3.1/

29. inorebuk - program rozwiązywania równości i nierówności pseudobulowskich kombinatoryczną metodą Bukietyńskiego /por. 2.3 oraz [11] /,

30. punkty - rekurencyjna procedura generowania punktów prostopadłościanu

31. analiza wariancji - program obliczania sum S_r /por. 4.3.3/

Uwaga: gwiazdka przy numerze pozycji oznacza, że dany program zapisany jest w języku MOST-1, pozostałe zaś zapisane są w języku ALGOL-1204.

Spis treści

str.