Akademia Ekonomiczna we Wrocławiu
Wydział Gospodarki Regionalnej i Turystyki w Jeleniej Górze

Andrzej Bąk

ZASTOSOWANIE TECHNIK SYMULACJI KOMPUTEROWEJ DO OPTYMALIZACJI WYBORU METOD WIELOWYMIAROWEJ ANALIZY PORÓWNAWCZEJ

(rozprawa doktorska)

Promotor: prof. dr hab. Tadeusz Grabiński

Jelenia Góra 1995

Spis treści

Wstęp

Inspiracją do podjęcia powyższego tematu rozprawy doktorskiej jest brak spójnych i jednoznacznych kryteriów wskazujących na to, które z wielu metod wielowymiarowej analizy porównawczej (WAP) są najbardziej przydatne i efektywne w określonych sytuacjach badawczych.

W ostatnich latach obserwujemy dynamiczny rozwój podstaw teoretycznych wielowymiarowej analizy porównawczej. Równocześnie wzrasta zainteresowanie przedstawicieli różnych dyscyplin naukowych (np. biologów, ekonometryków, ekonomistów, psychologów, socjologów, statystyków) w praktycznym wykorzystaniu metod WAP do realizacji własnych celów badawczych. W warunkach bogatej oferty metodologicznej pojawia się problem optymalizacji wyboru metody WAP pod kątem konkretnego zjawiska złożonego, będącego przedmiotem badań.

W literaturze przedmiotu spotkać można postulaty teoretyczne stawiane metodom WAP oraz pewne ogólne kierunki zastosowań określonych metod. Brak jest jednak przesłanek umożliwiających wybór optymalnej metody w odniesieniu do zbioru wejściowych danych statystycznych. Należy podkreślić, iż stajemy tutaj przed koniecznością wyboru nie tylko metody WAP, ale pewnej ścieżki postępowania. Procedura WAP składa się bowiem z wielu etapów i elementów, z których każdy może znacząco wpłynąć na ostateczny wynik analizy. Właściwa, z punktu widzenia celu i przedmiotu analizy oraz zbioru danych wejściowych, ścieżka badań powinna przynosić wyniki dobrze oddające relacje rzeczywiste. Może być jednak również tak, iż źle określona procedura postępowania, z punktu widzenia tych samych kryteriów, przyniesie wyniki niewłaściwie odzwierciedlające badany fragment rzeczywistości.

W proponowanej pracy doktorskiej podjęto próbę określenia odporności metod WAP na charakter i strukturę wejściowych danych statystycznych oraz wskazania w tym kontekście optymalnej procedury postępowania.

Pewnym dodatkowym argumentem zachęcającym do podjęcia niniejszego tematu była świadomość, iż w trakcie podjętych badań powstanie oprogramowanie komputerowe, być może użyteczne w innych analizach metodologicznych, empirycznych lub zajęciach dydaktycznych.

Celem podstawowym proponowanej pracy doktorskiej jest próba jakościowego uszeregowania oraz wskazania optymalnej konfiguracji metod WAP w świetle struktury i charakteru wejściowych danych statystycznych oraz zestawu kryteriów określających jakość poszczególnych metod, a także z punktu widzenia stopnia zgodności ostatecznego odwzorowania wielowymiarowej przestrzeni zmiennych pierwotnych w jednowymiarową przestrzeń zmiennej syntetycznej.

Poza właściwie określoną metodologią postępowania, niezbędnym środkiem osiągnięcia postawionego celu jest dysponowanie odpowiednim oprogramowaniem komputerowym. Na rynku informatycznym dostępne są atrakcyjne pakiety programów matematycznych, statystycznych, ekonometrycznych, symulacyjnych itp., o rozmaitych funkcjach i zastosowaniach. Niemniej jednak nie umożliwiają one w pełni realizacji postawionego celu pracy, przede wszystkim ze względu na przyjęty układ badań oraz zbiór analizowanych metod. W związku z tym podjęto próbę stworzenia oryginalnego oprogramowania do prowadzenia analiz metodami WAP z możliwością symulacyjnej optymalizacji wyboru metody.

Podstawową metodą badawczą w prezentowanej pracy doktorskiej jest metoda indukcyjna. Wynika to z faktu, iż pewne ogólne wnioski i oceny formułowane są w oparciu o dużą liczbę eksperymentów symulacyjnych. Narzędziem realizacji planowanych eksperymentów jest sprzęt komputerowy oraz specjalnie przygotowane oprogramowanie. Wykorzystano ponadto

techniki symulacji cyfrowej oraz programowe generatory liczb losowych do przygotowania zbiorów liczbowych danych statystycznych, będących podstawą badań.

Podstawowe tez niniejszej rozprawy doktorskiej są nastpujące:

1. Metody WAP odznaczają się pewną, możliwą do oszacowania, wrażliwością na charakter i strukturę przetwarzanych danych statystycznych. Materiał liczbowy będący przedmiotem przetwarzania metodami WAP podlega pewnym znanym rozkładom zmiennych losowych oraz charakteryzuje się określonymi parametrami strukturalnymi. Można zatem oczekiwać, iż znajomość stopnia stabilności poszczególnych metod WAP w odniesieniu do określonej struktury danych statystycznych pozwoli na wybór najlepszej metody w konkretnej sytuacji badawczej.

2. Poziom wrażliwości poszczególnych metod WAP na zmiany w strukturze wejściowych danych statystycznych jest różny. Oznacza to konieczność oszacowania stopnia odporności tych metod na pewne zaburzenia i modyfikacje w zbiorach danych statystycznych. Istotny wpływ na ostateczny wynik badań metodami WAP ma w tym kontekście ich konkretna konfiguracja. Wynika to z faktu, iż poszczególne metody powodują w pierwotnych danych statystycznych różnorodne zaburzenia, zniekształcenia oraz straty w informatywności o różnej sile i znaczeniu.

3. Zakłada się, iż jest możliwe określenie zależności pomiędzy charakterem danych wejściowych a sposobem reakcji poszczególnych metod WAP przy wykorzystaniu technik symulacji cyfrowej, odpowiedniego oprogramowania oraz wystarczająco sprawnego, w kontekście złożoności numerycznej obliczeń, sprzętu komputerowego.

Prezentowana rozprawa doktorska składa się z pięciu rozdziałów oraz aneksu.

W rozdziale pierwszym przedstawiono procedurę wielowymiarowej analizy porównawczej w układzie i zakresie umożliwiającym realizację celu pracy. W pierwszym rzędzie omówiono główne zagadnienia związane z gromadzeniem, przekształcaniem i komputerowym przetwarzaniem liczbowych wartości cech opisujących analizowane obiekty, a także scharakte-

ryzowano szczegółowo, w sposób formalny i opisowy, obiekty wielowymiarowe. Przedstawiono sposoby pomiaru wartości cech i miary podobieństwa oraz ich wpływ na dalszy przebieg przetwarzania. W analizie zjawisk złożonych istotny jest kierunek wpływu poszczególnych cech na zachowanie się badanego obiektu. Z tego względu omówiono sposoby identyfikacji charakteru zmiennych oraz przekształcania pierwotnych danych statystycznych. W dalszej części tego rozdziału przedstawiono argumenty uzasadniające traktowanie wartości cech jako realizacji zmiennych losowych o określonych rozkładach prawdopodobieństwa. Następnie scharakteryzowano sposoby wyznaczania współczynników wagowych zmiennych, sposoby normalizacji cech oraz metody wyznaczania zmiennych syntetycznych. Rozdział ten zamykają uwagi dotyczące problemu stabilności metod WAP i jego praktycznych implikacji oraz prezentacja mierników jakości wykorzystanych do ilościowego oszacowania efektywności poszczególnych metod WAP.

W rozdziale drugim omówiono zagadnienia dotyczące otrzymywania losowych danych statystycznych dla potrzeb eksperymentów symulacyjnych. W szczególności przedstawiono algorytmy generowania liczb losowych o zadanych rozkładach prawdopodobieństwa, będące podstawowym narzędziem tworzenia zbiorów wejściowych danych liczbowych. Z uwagi na ścisłą zależność pomiędzy jakością programowych generatorów liczb losowych a ostatecznymi wynikami analizy zamieszczono uwagi dotyczące statystycznej oceny ich jakości. Istotnym elementem tego rozdziału jest wskazanie sposobów generowania wartości losowych tworzących wektor, macierz oraz kostkę danych. W tym rozdziale przedstawiono również parametry statystyczne stosowane do opisu struktur danych liczbowych.

W rozdziale trzecim przedstawiono, w oparciu o literaturę przedmiotu, ogólną metodologię modelowania symulacyjnego oraz uzasadnienie jej wykorzystania w prezentowanej rozprawie. Następnie zaprezentowano układ eksperymentu symulacyjnego, zmierzającego do udowodnienia podstawowych tez pracy. W dalszej części rozdziału trzeciego zaprezentowano

reguły generowania i przetwarzania zbiorów danych oraz zasady oceny wyników eksperymentów symulacyjnych.

W rozdziale czwartym zaprezentowano rezultaty eksperymentów symulacyjnych oraz przedstawiono ich interpretację merytoryczną. W szczególności omówiono przekroje analizy wyników oraz uzyskane uporządkowania uwzględnionych w badaniach konfiguracji WAP. Scharakteryzowano ponadto tendencję zbieżności wyników i jej uzasadnienie oraz omówiono efekty przeprowadzonej analizy wrażliwości.

W rozdziale piątym zamieszczono zwięzłą charakterystykę oprogramowania stworzonego dla realizacji podstawowego celu pracy. Na początku omówiono wykorzystane narzędzia i techniki programowania. W dalszej części tego rozdziału przedstawiono funkcjonalną strukturę programu symulacyjnego i podano krótkie charakterystyki podstawowych modułów i funkcji składowych poszczególnych klas. W końcowej części tego rozdziału przedstawiono struktury wykorzystywanych w programie zbiorów danych.

W rozdziałach pierwszym, drugim i trzecim zamieszczono ponadto ważniejsze algorytmy zaimplementowane w programie symulacyjnym, stosując jednolitą notację ułatwiającą ich analizę. Nazwy klas i funkcji składowych używane w prezentowanych algorytmach dotyczą konstrukcji programowych opisanych w rozdziale piątym oraz przedstawionych w aneksie do pracy. Numeracja tablic, rysunków, algorytmów oraz wzorów jest ciągła w ramach każdego rozdziału, natomiast numeracja przypisów jest ciągła w ramach całej pracy.

W aneksie do pracy zamieszczono większość obszernych tablic zawierających wyniki sesji symulacyjnych, kody źródłowe podstawowych klas programu komputerowego stworzonego i wykorzystanego do przeprowadzenia badań, a także szczegółowe struktury plików wejściowych i wyjściowych.

Niektóre fragmenty niniejszej rozprawy były prezentowane i dyskutowane na konferencjach naukowych poświęconych zastosowaniom metod taksonomicznych i analizie danych, a także publikowane w zeszytach naukowych.

Do przygotowania programu symulacyjnego oraz redakcyjnego złożenia pracy wykorzystano następujące oprogramowanie komputerowe:

- pakiet Borland C/C++ v. 4.0 firmy Borland International, Inc.,
- pakiet CodeBase++ v. 5.1 firmy Sequiter Software, Inc.,
- pakiet Clipper v. 5.0 firmy Computer® Associates International, Inc.,
- arkusz kalkulacyjny Excel v. 5.0 firmy Microsoft Corporation,
- edytor tekstu Word for Windows v. 6.0 firmy Microsoft Corporation.

Pragnę złożyć wyrazy podziękowania Panu Profesorowi Tadeuszowi Grabińskiemu, który zgodził się wziąć na siebie ciężar promowania niniejszej rozprawy, dzieląc się z autorem swoją wiedzą i doświadczeniem oraz udzielając wsparcia, którego w sposób wymierny nie da się oszacować, a które jest szczególnie ważne w trudnych momentach.

Rozdział 1

PROCEDURA WIELOWYMIAROWEJ ANALIZY PORÓWNAWCZEJ

1.1. Opis obiektów wielowymiarowych

Wielowymiarowa analiza porównawcza (WAP) jest dyscypliną naukową umożliwiającą analizę obiektów i zjawisk złożonych, tj. takich, na których stan i zachowanie wpływa jednocześnie wiele cech (zmiennych) i czynników. Zwięzła definicja podana przez Hellwiga mówi, iż „(...) metody i technika porównywania obiektów wielocechowych nazywają się wielowymiarową analizą porównawczą (...)" [87, s. 48]. Do podstawowych zadań WAP zalicza się najczęściej (por. [72]):

- hierarchizację (porządkowanie liniowe) obiektów wielocechowych na podstawie wartości pewnej miary syntetycznej (agregatowej), będącej wypadkową zmiennych diagnostycznych,
- klasyfikację (grupowanie) obiektów w przestrzeni zmiennych diagnostycznych w celu wyodrębnienia podzbiorów obiektów jednorodnych (homogenicznych),
- redukcję wstępnie zdefiniowanego zbioru zmiennych w celu wyodrębnienia podzbioru cech diagnostycznych, istotnie wpływających na zachowanie się badanych obiektów w przestrzeni i czasie,
- normalizację zmiennych i określanie kierunku ich wpływu na analizowane zjawisko,

- określanie współrzędnych wzorców rozwoju (punktów odniesienia), umożliwiających prognozowanie trajektorii przebiegu badanego zjawiska.

Należy zaznaczyć, iż badania zjawisk ekonomicznych metodami WAP można prowadzić zarówno w układzie statycznym (w oparciu o dane przekrojowe), jak i w układzie dynamicznym (w oparciu o szeregi przekrojowo-czasowe)[1].

Do podstawowych kategorii WAP zalicza się *obiekty* i *cechy*. Z uwagi na ich pierwotne znaczenie nie podlegają one definiowaniu (por. [28, s. 80], [87, s. 47]). W zależności od kontekstu badań podawane są jednakże różnorodne określenia obiektów i cech. W dalszej części pracy przez obiekty rozumieć się będzie jednostki badania, podlegające przetwarzaniu metodami WAP, zaś cechy (zmienne) rozumiane będą jako charakterystyki (właściwości), którymi te jednostki się odznaczają [90], [126]. Zarówno obiekty, jak i cechy posiadają swoje liczbowe reprezentacje (obrazy), co umożliwia ich wszechstronne analizowanie oraz przetwarzanie przy pomocy metod WAP. Czynności związane z liczbowym odwzorowaniem cechy za pomocą pewnej miary nazywane są pomiarem, natomiast zastosowana miara określana jest mianem skali pomiaru lub skali prezentacji wartości cechy (por. np. [92, s. 13], [130, s. 59], [150, s. 28]). Znane są cztery skale prezentacji wartości zmiennych (nominalna, porządkowa, przedziałowa oraz ilorazowa), jednakże zaleca się stosowanie w praktyce badawczej jednej, najmocniejszej skali pomiaru wspólnej dla wszystkich analizowanych zmiennych ze względu na fakt, iż zestaw dozwolonych przekształceń jest bogatszy na skalach mocniejszych. Można to osiągnąć na drodze transformacji skal pomiarowych ze słabszych na silniejsze (por. [22], [50], [172], [174]).

W ujęciu formalnym przedmiotem badań WAP jest zbiór obiektów Ω zawierający n elementów:

$$\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}, \tag{1.1}$$

[1] Przykłady zastosowań metod WAP zamieszczone są m.in. w pracach: [6], [15], [50], [63], [64], [72], [74], [80], [139], [141]. Prace [16], [19], [66], [72], [75], [76], [79], [83], [108] zawierają natomiast referencje dotyczące komputerowych implementacji algorytmów WAP.

oraz zbiór cech Φ zawierający m elementów:

$$\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}. \tag{1.2}$$

Bezpośredni proces badawczy prowadzony jest na liczbowych realizacjach (wartościach, obserwacjach) poszczególnych cech w przestrzeni obiektów lub okresów. Numeryczny obraz uwzględnionych w badaniu zmiennych uzyskuje się w wyniku przekształcenia:

$$X_j : \Omega \rightarrow R^n, \ (j = 1,2,\ldots,m), \tag{1.3}$$

gdzie: m - liczba cech,
n - liczba obiektów,
R - zbiór liczb rzeczywistych.

W rezultacie powyższego zabiegu otrzymuje się macierz obserwacji postaci:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}, \tag{1.4}$$

gdzie: x_{ij} - wartość cechy φ_j dla obiektu ω_i,
$i = 1,2, \ldots, n$,
$j = 1,2, \ldots, m$.

Wiersze macierzy **X**, tzn. wektory $\mathbf{X}_{i\cdot} = [x_{i1}, x_{i2}, \ldots, x_{im}]$ należy traktować jako liczbowe reprezentacje obiektów (każdy wektor opisuje jeden z n obiektów), natomiast kolumny macierzy **X**, czyli wektory $\mathbf{X}_{\cdot j} = [x_{1j}, x_{2j}, \ldots, x_{nj}]^T$ należy interpretować jako liczbowe obrazy cech (każda kolumna zawiera wartości jednej z m zmiennych).

Powyższe uwagi dotyczą WAP w ujęciu statycznym, co oznacza, że elementami macierzy $\mathbf{X} = [x_{ij}]$ są dane o charakterze przekrojowym zaobserwowane w określonym momencie t_0. Dla potrzeb dynamicznej wielowymiarowej analizy porównawczej (DWAP) należy uwzględnić dodatkowo czynnik czasu, a więc wprowadzić do macierzy **X** trzeci wymiar, reprezentujący kolejne momenty $t = 1,2, \ldots, v$. Macierz obserwacji przybierze wówczas następującą postać:

$$\mathbf{X}^t = \begin{bmatrix} x_{11}^t & x_{12}^t & \cdots & x_{1m}^t \\ x_{21}^t & x_{22}^t & \cdots & x_{2m}^t \\ \hdashline x_{n1}^t & x_{n2}^t & \cdots & x_{nm}^t \end{bmatrix}. \tag{1.5}$$

Elementy macierzy $\mathbf{X}^t = \left[x_{ij}^t\right]$ należy interpretować jako wartości cech opisujących badane obiekty w kolejnych analizowanych momentach. Przedmiotem badań w DWAP jest zatem następujący ciąg macierzy:

$$\mathbf{X}^t \Rightarrow \left\{\left[x_{ij}^1\right],\left[x_{ij}^2\right],\ldots,\left[x_{ij}^v\right]\right\} \quad (i = 1,2, \ldots, n;\ j = 1,2, \ldots, m) \tag{1.6}$$

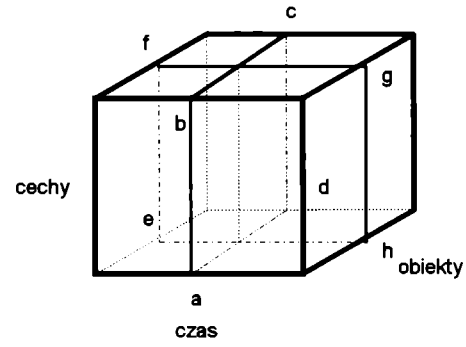
gdzie: n - liczba obiektów,
m - liczba cech,
v - liczba okresów (momentów).

W przypadku macierzy o postaci (1.5) należy zwrócić uwagę na możliwość analizy różnych jej przekrojów. Praktyczne znaczenie w WAP mają w szczególności dwa przekroje (zob. rys. 1.1), generujące następujące dwuwymiarowe struktury danych (por. [50, s. 33-37], [90, s. 22-23]):

1) wielowymiarowy szereg przekrojowy, w którym każdy z n obiektów opisany jest wartościami m zmiennych w danym okresie,

2) wielowymiarowy szereg czasowy, w którym dany obiekt opisany jest wartościami m zmiennych w kolejnych okresach.

Macierz (1.5) w postaci trójwymiarowej kostki danych, reprezentuje natomiast wielowymiarowy szereg przekrojowo-czasowy i może być podstawą wielowymiarowej analizy porównawczej w ujęciu dynamicznym.



(a,b,c,d) - przekrój obiekty x cechy
(e,f,g,h) - przekrój cechy x czas

Rysunek 1.1. Ilustracja podstawowych przekrojów trójwymiarowej macierzy obserwacji

W dalszej części pracy rozpatrywane będą następujące struktury danych:

- struktury dwuwymiarowe o postaci: $\mathbf{X} = \left[x_{ij}\right]_{n \times m}$ (np. obiekty x cechy),
- struktury trójwymiarowe o postaci: $\mathbf{X} = \left[x_{ijt}\right]_{n \times m \times v}$ (np. obiekty x cechy x okresy).

1.2. Miary podobieństwa

Miary podobieństwa są wykorzystywane w wielowymiarowej analizie porównawczej do określania siły związku (jednorodności) pomiędzy obiektami, cechami lub skupieniami (klasami, grupami). Szczególną rolę odgrywa tutaj miara nazywana odległością, zdefiniowana w postaci iloczynu kartezjańskiego na zbiorze liczb rzeczywistych:

$$d: R^m \times R^m \rightarrow R$$

i spełniająca warunki:

a) odległość między obiektami ω_i i ω_k wynosi 0, co oznacza, że nie jest możliwe ich rozróżnienie:

$$(\forall \omega_i \in \Omega)(\forall \omega_k \in \Omega)(d(\omega_i,\omega_k)=0) \Leftrightarrow (\omega_i = \omega_k), \tag{1.7}$$

b) zachodzi relacja symetryczności dla $d(\omega_i,\omega_k) > 0$:

$$(\forall \omega_i \in \Omega)(\forall \omega_k \in \Omega)(d(\omega_i,\omega_k) = d(\omega_k,\omega_i)). \tag{1.8}$$

Jeżeli ponadto spełniony jest warunek nierówności trójkąta:

$$(\forall \omega_i \in \Omega)(\forall \omega_k \in \Omega)(\forall \omega_l \in \Omega)(d(\omega_i,\omega_k) \le d(\omega_i,\omega_l)+d(\omega_l,\omega_k)), \tag{1.9}$$

to odległość d określa się mianem *metryki*, w przeciwnym zaś razie miara d jest *quasimetryką (semimetryką)*. W sytuacji, kiedy nie obowiązuje warunek (1.7) miara d nosi nazwę *pseudo-metryki*. Jeżeli natomiast zaostrzony zostanie warunek (1.8) do postaci:

$$d(\omega_i,\omega_k) \le \max(d(\omega_i,\omega_k), d(\omega_k,\omega_l)), \tag{1.10}$$

to uzyskana miara d określana będzie mianem *ultrametryki* (por. [35], [41], [130]).

Miarę bliskości uzyskać można w wyniku przekształcenia miary odległości. Jeżeli $d(\omega_i,\omega_k) \in [0,1]$ to odpowiednia formuła ma postać ([69], [130]):

$$h(\omega_i,\omega_k) = 1 - d(\omega_i,\omega_k). \tag{1.11}$$

W sposób ogólny podstawowa grupa mierników odległości określana jest za pomocą formuły Minkowskiego:

$$d_{ik} = d(\mathbf{z}_i,\mathbf{z}_k) = \left[\sum_{j=1}^{m}(z_{ij} - z_{kj})^p\right]^{1/p}, \tag{1.12}$$

gdzie: $d(\mathbf{z}_i,\mathbf{z}_k)$ - odległość między wektorami znormalizowanych obserwacji opisującymi i-ty oraz k-ty obiekt,
$i, k = 1,2,...,n$,
$j=1,2,...,m$,
n - liczba obiektów,
m - liczba cech.

Z formuły (1.12) wyprowadza się zależnie od wartości parametru p trzy następujące mierniki odległości:

1) odległość Hamminga (dla $p = 1$):

$$d_{ik} = \sum_{j=1}^{m} \left| z_{ij} - z_{kj} \right|, \tag{1.13}$$

2) odległość Euklidesa (dla $p = 2$):

$$d_{ik} = \left[\sum_{j=1}^{m} (z_{ij} - z_{kj})^2 \right]^{1/2}, \tag{1.14}$$

3) odległość Czebyszewa (dla $p \to \infty$)

$$d_{ik} = \max_{i} \left\{ \left| z_{ij} - z_{kj} \right| \right\}. \tag{1.15}$$

Jeżeli do wzorów na odległość Hamminga (1.13) i Euklidesa (1.14) wprowadzimy współczynniki wagowe[2] zmiennych to przyjmą one następującą postać:

- odległość Hamminga:

$$d_{ik} = \sum_{j=1}^{m} w_j \left| z_{ij} - z_{kj} \right|, \tag{1.16}$$

- odległość Euklidesa:

$$d_{ik} = \left[\sum_{j=1}^{m} w_j (z_{ij} - z_{kj})^2 \right]^{1/2} \tag{1.17}$$

Często wykorzystywany jest również kwadrat odległości Euklidesa oraz wartość przeciętna tego kwadratu, czyli odpowiednio:

- kwadrat odległości Euklidesa:

$$d_{ik}^2 = \sum_{j=1}^{m} (z_{ij} - z_{kj})^2, \tag{1.18}$$

- wartość przeciętna kwadratu odległości Euklidesa:

[2] Współczynniki wagowe zmiennych omówione są w punkcie 1.4 niniejszego rozdziału.

$$d_{ik}^2 = \frac{1}{m}\sum_{j=1}^{m}(z_{ij} - z_{kj})^2 . \tag{1.19}$$

Spośród wielu miar odległości do najpowszechniej stosowanych w badaniach aplikacyjnych zaliczyć należy następujące mierniki:

1) odległość Mahalanobisa wyrażoną wzorem:

$$d_{ik} = \left[(\mathbf{z}_i - \mathbf{z}_k)^T \Sigma^{-1}(\mathbf{z}_i - \mathbf{z}_k)\right]^{1/2}, \tag{1.20}$$

gdzie: Σ - macierz kowariancji,

lub w postaci uszczegółowionej wzorem:

$$d_{ik} = \left[\sum_{j=1}^{m}\sum_{l=1}^{m}(z_{ij} - z_{kj})(z_{il} - z_{kl})s_{jl}\right]^{1/2}, \tag{1.21}$$

gdzie: s_{jl} - element macierzy odwrotnej do macierzy kowariancji,

2) odległość kątowa wyrażona wzorem:

$$d_{ik} = \frac{\mathbf{z}_i^T \mathbf{z}_k}{\left[(\mathbf{z}_i^T \mathbf{z}_i)(\mathbf{z}_k^T \mathbf{z}_k)\right]^{1/2}}, \tag{1.22}$$

lub w postaci uszczegółowionej wzorem:

$$d_{ik} = \frac{\sum_{j=1}^{m} z_{ij} z_{kj}}{\left[\sum_{j=1}^{m} z_{ij}^2 \sum_{j=1}^{m} z_{kj}^2\right]^{1/2}}, \tag{1.23}$$

3) odległość Braya-Curtisa:

$$d_{ik} = \frac{\sum_{j=1}^{m}|z_{ij} - z_{kj}|}{\sum_{j=1}^{m}(z_{ij} + z_{kj})}, \tag{1.24}$$

4) odległość „Canberra":

$$d_{ik} = \frac{1}{m}\sum_{j=1}^{m}\frac{|z_{ij} - z_{kj}|}{(z_{ij} + z_{kj})}, \tag{1.25}$$

5) odległość Jeffreysa-Matusita:

$$d_{ik} = \frac{1}{m}\sum_{j=1}^{m}\left(\sqrt{z_{ij}} - \sqrt{z_{kj}}\right)^2, \tag{1.26}$$

6) odległość Clarka:

$$d_{ik} = \left[\frac{1}{m}\sum_{j=1}^{m}\left(\frac{z_{ij} - z_{kj}}{z_{ij} + z_{kj}}\right)^2\right]^{1/2}, \tag{1.27}$$

7) odległość łukową:

$$d_{ik} = \left[\frac{1 - \sum_{j=1}^{m} z_{ij} z_{kj}}{\sum_{j=1}^{m} z_{ij}^2 \sum_{j=1}^{m} z_{kj}^2}\right]^{1/2} \tag{1.28}$$

1.3. Identyfikacja charakteru zmiennych

W wielowymiarowej analizie porównawczej przedmiotem porządkowania liniowego są obiekty, których pozycję w określonym szeregu wyznaczają wartości pewnej cechy agregatowej. Z reguły wyższe wartości zmiennej agregatowej świadczą o wyższej pozycji danego obiektu. Jednak w zbiorze zmiennych diagnostycznych nie zawsze zachodzi tego typu prawidłowość. Zdarza się bowiem tak, iż rosnące wartości zmiennej wpływają negatywnie na ocenę danego obiektu lub stanu zjawiska. W związku z tym rozróżnia się w zbiorze cech trzy kategorie zmiennych o następujących właściwościach (por. [29], [86], [87], [88]):

1) zmienne stymulanty, których wzrost wartości wpływa korzystnie na ocenę obiektu:

$$\underset{x_i, x_k}{\wedge} \left(x_i \geq x_k\right) \Rightarrow \omega_i \succ \omega_k \tag{1.29}$$

2) zmienne destymulanty, których wzrost wartości wpływa niekorzystnie na ocenę obiektu:

$$\underset{x_i, x_k}{\wedge} \left(x_i \geq x_k\right) \Rightarrow \omega_i \prec \omega_k \tag{1.30}$$

3) zmienne nominanty, których wzrost wartości może wpływać zarówno korzystnie, jak i niekorzystnie na ocenę obiektu:

$$\underset{x_i, x_k \leq x_{nom}}{\wedge} \left(x_i > x_k\right) \Rightarrow \omega_i \succ \omega_k$$

oraz (1.31)

$$\underset{x_i, x_k \leq x_{nom}}{\wedge} \left(x_i > x_k\right) \Rightarrow \omega_i \prec \omega_k$$

gdzie: x_{nom} - wartość nominalna cechy.

W celu identyfikacji charakteru zmiennych diagnostycznych stosuje się metody analizy i oceny merytorycznej oraz metody statystyczne, bazujące na macierzy korelacji, omówione w pracach [68] i [72]. Do grupy formalnych (statystycznych) metod określania charakteru zmiennych zalicza się:

1) metodę analizy struktury macierzy korelacji,

2) metodę wykorzystującą procedurę analizy czynnikowej,

3) metodę analizy współczynników korelacji pomiędzy zmiennymi diagnostycznymi a zmienną syntetyczną.

W niniejszej pracy dla potrzeb formalnej identyfikacji charakteru cech wykorzystano metodę bazującą na procedurze analizy czynnikowej. Analiza czynnikowa zaliczana jest do grupy metod wielowymiarowej analizy statystycznej służących badaniu współzależności między zmiennymi. W sensie ogólnym analiza czynnikowa obejmuje klasyczną analizę czynnikową oraz metodę głównych składowych. Klasyczna analiza czynnikowa, której główne idee oraz podstawowe założenia metodologiczne sformułowali Spearman i Thurstone, wykorzy-

stywana jest przede wszystkim do badania wewnętrznych zależności między zmiennymi. Metoda głównych składowych natomiast, której podstawy teoretyczne stworzyli Pearson i Hotelling, znajduje zastosowanie zarówno w analizie współzależności zbioru zmiennych, jak i w analizie struktury zbioru obserwacji (por. np. [6], [100], [101]).

Punktem wyjścia w metodzie analizy czynnikowej jest macierz korelacji zmiennych diagnostycznych [6], [163], [180]:

$$\mathbf{R} = \left[r_{kj}\right]_{m \times m} \tag{1.32}$$

gdzie: $r_{kj} = \frac{1}{n}\mathbf{Z}^T\mathbf{Z} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} z_{ik} z_{ij} & \text{dla } k \neq j \\ 1 & \text{dla } k = j \end{cases}$

Na podstawie (1.32) wyznacza się elementy pierwszej głównej składowej. Są one interpretowane jako współczynniki korelacji pomiędzy poszczególnymi zmiennymi diagnostycznymi a pierwszym czynnikiem głównym, który jest nośnikiem największego zasobu informacji o analizowanym zbiorze cech.[3] Przyjmuje się, iż zmienne dodatnio skorelowane z pierwszym czynnikiem głównym mają charakter stymulant, zaś zmienne o korelacji ujemnej z tym czynnikiem są destymulantami. Algorytm 1.1 przedstawia procedurę identyfikacji charakteru zmiennych diagnostycznych opartą na metodzie analizy czynnikowej.

Algorytm 1.1

Identyfikacja charakteru zmiennych metodą analizy czynnikowej

K01: Dana jest macierz obserwacji: $\mathbf{X} = \left[x_{ij}\right]_{n \times m}$

K02: Obliczyć macierz korelacji zmiennych: $\mathbf{R} = \left[r_{ij}\right]_{m \times m}$ przy pomocy funkcji *corr*

K03: Wyznaczyć największą wartość własną macierzy korelacji: $\max \lambda_l$ przy pomocy funkcji *reyleigh*

K04: Wyznaczyć wektor własny macierzy korelacji odpowiadający największej wartości własnej: $\mathbf{a}_l$ przy pomocy funkcji *hotelling*

K05: Obliczyć ładunki czynnikowe pierwszej głównej składowej: $w_{jl} = \sqrt{\lambda_l}\,\mathbf{a}_l$ przy pomocy funkcji *loadings*

---

[3] W analizie czynnikowej za czynniki niosące wystarczający zasób informacji o badanym zjawisku, uznaje się na ogół te, które łącznie wyjaśniają założony procent wariancji (przyjmuje się najczęściej wartości między 75% a 90%), a jednocześnie żaden kolejny czynnik nie objaśnia więcej niż 5% (por. [163, s. 313], [184, s. 37],). Krytyczny przegląd propozycji dotyczących metod szacowania liczby akceptowanych czynników głównych znajduje się w pracy [184]. Charakterystykę wybranych technik postępowania w tym względzie zawiera również praca [100]. W monografii [120] omówiono natomiast wiele problemów dotyczących zarówno analizy czynnikowej, jak i metody głównych składowych.

W algorytmie 1.1 w celu wyznaczenia największej wartości własnej wykorzystano metodę Reyleigha [176, s. 96-97], która jest przydatna w przypadku macierzy symetrycznych dodatnio określonych (jak wiadomo, własności takie posiada macierz korelacji). Do wyznaczenia wszystkich wartości własnych oraz odpowiadających im wektorów własnych zastosowano metodę iteracyjnej redukcji macierzy korelacji zaproponowaną przez Hotellinga dla macierzy symetrycznych[4] [176, s. 97-99].

Najczęściej stosowanym zabiegiem, mającym na celu ujednolicenie kierunku oddziaływania zmiennych na analizowane zjawisko, jest przekształcenie cech destymulant w cechy stymulanty. Operacja ta polega na pomnożeniu wartości cech destymulant w znormalizowanym zbiorze obserwacji przez -1, tzn.:

$$z_{ij}^{'} = \begin{cases} z_{ij} & \text{dla cech stymulant} \\ -z_{ij} & \text{dla cech destymulant} \end{cases} . \tag{1.33}$$

Jeżeli żąda się ponadto, aby zbiór obserwacji nie zawierał wartości ujemnych, to można to osiągnąć za pomocą kolejnego przekształcenia [65, s. 118-124]:

$$z_{ij}^{''} = \begin{cases} z_{ij}^{'} & \text{gdy} \quad \min\limits_{i,j}\left\{z_{ij}^{'}\right\} > 0 \\ z_{ij}^{'} + \sigma & \text{gdy} \quad \min\limits_{i,j}\left\{z_{ij}^{'}\right\} \leq 0 \end{cases} \tag{1.34}$$

gdzie: $\sigma = -\min\limits_{i,j}\left\{z_{ij}^{'}\right\} + \frac{1}{5}\sigma_{z'}$,

$\sigma_{z'}$ - odchylenie standardowe zbioru obserwacji znormalizowanych (całej macierzy danych),

$\min\limits_{i,j}\left\{z_{ij}^{'}\right\}$ - wartość minimalna w zbiorze obserwacji znormalizowanych (całej macierzy danych).

[4] Przegląd numerycznie stabilnych algorytmów wyznaczania wartości własnych i wektorów własnych również dla innych typów macierzy zawierają m.in. następujące prace [11], [14], [21], [47], [55], [93], [104], [105], [114], [157].

1.4. Współczynniki wagowe zmiennych

W badaniach ekonomicznych znaczenie merytoryczne zmiennych, opisujących analizowane zjawisko, odgrywa zazwyczaj rolę podstawową. Z tego punktu widzenia nie zawsze ranga poszczególnych zmiennych uwzględnionych w badaniu jest jednakowa. W celu zróżnicowania znaczenia wyspecyfikowanych cech należy dokonać odpowiedniego merytorycznego lub statystycznego ważenia ich wartości liczbowych. Najczęściej współczynniki wagowe zmiennych wyznaczane są według formuł statystycznych. Jednakże w literaturze przedmiotu podkreśla się rolę merytorycznej analizy i ważenia zmiennych, np. na podstawie ocen niezależnych ekspertów.

Tablica 1.1

Formuły ważenia zmiennych

Lp.	Współczynnik wagowy
1	$w_j = \frac{1}{m}$
2	$w_j = \frac{V_j}{\sum_{j=1}^{m} V_j}$

gdzie: m - liczba cech,

$V_j = \frac{\sigma_j}{\bar{x}_j}$ - współczynnik zmienności obliczony dla pierwotnych wartości cech (przed normalizacją zmiennych).

Źródło: [65]

Dość powszechnie, mimo pewnych zastrzeżeń przedstawionych w pracach [1] i [139], w badaniach empirycznych stosowane są następujące sposoby statystycznego wyznaczania współczynników wagowych zmiennych:

1) przyjęcie wag stałych, co oznacza przypisanie jednakowego znaczenia wszystkim zmiennym,

2) ustalenie wartości współczynników wagowych na podstawie zasobu informacji zawartego w poszczególnych cechach (podstawą oceny są współczynniki zmienności).

Formuły ważenia zmiennych wykorzystane w badaniach przedstawia tablica 1.1 Współczynniki te spełniają własności: $w_j \geq 0$ oraz $\sum_{j=1}^{m} w_j = 1$.

1.5. Normalizacja zmiennych

Zgromadzone w formie macierzy danych (1.4) i (1.5), wartości liczbowe zmiennych posiadają zazwyczaj różne miana i różne obszary zmienności. Nie jest zatem możliwe bezpośrednio wykonywanie na tych wartościach operacji arytmetycznych, ani też ich porównywanie. Pewne elementarne przekształcenia na pierwotnych wartościach cech, nazywane normalizacją statystyczną zmiennych, umożliwiają pokonanie wyżej wymienionych ograniczeń formalnych i trudności interpretacyjnych.

Operacja normalizacji statystycznej polega na takiej transformacji pierwotnych wartości cech, aby otrzymane nowe realizacje zmiennych charakteryzowały się pożądanymi własnościami formalnymi (por. [22], [27], [29]).

Jako główne cele normalizacji statystycznej zmiennych wymienia się najczęściej następujące (por. [29], [66]):

1) uzyskanie podstaw formalnych do wykonywania podstawowych działań arytmetycznych w zbiorach wartości cech o różnych mianach (postulat addytywności),

Tablica 1.2

Formuły normalizacji zmiennych

Lp.	Struktury dwuwymiarowe $\mathbf{X}=\left[x_{ij}\right]_{n\times m}$	Struktury trójwymiarowe $\mathbf{X}=\left[x_{ijt}\right]_{n\times m\times v}$		Własności przekształcenia
		ujęcie statyczne	ujęcie dynamiczne	
Przekształcenie $z_{ij}=\ldots$				
Przekształcenia ilorazowe				
1	$\frac{x_{ij}}{\min\limits_{i} x_{ij}}$	$\frac{x_{ij}^{t}}{\min\limits_{i} x_{ij}^{t}}$	$\frac{x_{i,tj}}{\min\limits_{i,t} x_{i,tj}}$	$z_{ij}\geq 0$
2	$\frac{x_{ij}}{\max\limits_{i} x_{ij}}$	$\frac{x_{ij}^{t}}{\max\limits_{i} x_{ij}^{t}}$	$\frac{x_{ij}}{\max\limits_{i} x_{ij}}$	$z_{ij}\in\langle 0,1\rangle$
3	$\frac{x_{ij}}{\bar{x}_{j}}$	$\frac{x_{ij}^{t}}{\bar{x}_{j}^{t}}$	$\frac{x_{i,tj}}{\bar{x}_{j}}$	$\bar{z}_{j}=$ $z_{ij}\geq 0$
4	$\frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}}$	$\frac{x_{ij}^{t}}{\sum_{i=1}^{n} x_{ij}^{t}}$	$\frac{x_{i,tj}}{\sum_{i=1}^{n}\sum_{t=1}^{v} x_{i,tj}}$	$\sum_{i=1}^{n} z_{ij}=$ $z_{ij}\in\langle 0,1\rangle$
Standaryzacja				
5	$\frac{x_{ij}-\bar{x}_{j}}{\sigma_{j}}$	$\frac{x_{ij}^{t}-\bar{x}_{j}^{t}}{\sigma_{j}^{t}}$	$\frac{x_{i,tj}-\bar{x}_{j}^{\cdot}}{\sigma_{j}^{\cdot}}$	$\bar{z}_{j}=0$ $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_{ij}-\bar{z}_{j})^{2}}=1$
6	$\frac{x_{ij}}{\sigma_{j}}$	$\frac{x_{ij}^{t}}{\sigma_{j}^{t}}$	$\frac{x_{i,tj}}{\sigma_{j}^{\cdot}}$	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_{ij}-\bar{z}_{j})^{2}}=1$ $z_{ij}\geq 0$
Unitaryzacja				
7	$\frac{x_{ij}}{\max\limits_{i} x_{ij}-\min\limits_{i} x_{ij}}$	$\frac{x_{ij}^{t}}{\max\limits_{i} x_{ij}^{t}-\min\limits_{i} x_{ij}^{t}}$	$\frac{x_{i,tj}}{\max\limits_{i,t} x_{i,tj}-\min\limits_{i,t} x_{i,tj}}$	$\max\limits_{i} z_{ij}-\min\limits_{i} z_{ij}=1$ $z_{ij}\geq 0$
8	$\frac{x_{ij}-\bar{x}_{j}}{\max\limits_{i} x_{ij}-\min\limits_{i} x_{ij}}$	$\frac{x_{ij}^{t}-\bar{x}_{j}^{t}}{\max\limits_{i} x_{ij}^{t}-\min\limits_{i} x_{ij}^{t}}$	$\frac{x_{i,tj}-\bar{x}_{j}^{\cdot}}{\max\limits_{i,t} x_{i,tj}-\min\limits_{i,t} x_{i,tj}}$	$\bar{z}_{j}=0$ $\max z_{j}-\min z_{j}=$
9	$\frac{x_{ij}-\min\limits_{i} x_{ij}}{\max\limits_{i} x_{ij}-\min\limits_{i} x_{ij}}$	$\frac{x_{ij}^{t}-\min\limits_{i} x_{ij}^{t}}{\max\limits_{i} x_{ij}^{t}-\min\limits_{i} x_{ij}^{t}}$	$\frac{x_{i,tj}-\min\limits_{i,t} x_{i,tj}}{\max\limits_{i,t} x_{i,tj}-\min\limits_{i,t} x_{i,tj}}$	$z_{j}\in\langle 0,1\rangle$ $\min z_{j}=0 \quad \max z_{j}=1$ $z_{j}=\frac{\bar{x}_{j}-\min\limits_{i} x_{ij}}{\max\limits_{i} x_{ij}-\min\limits_{i} x_{ij}}$

gdzie: $\bar{z}_{j}=\frac{1}{n}\sum_{i=1}^{n} z_{ij}$, $\bar{x}_{j}=\frac{1}{n}\sum_{i=1}^{n} x_{ij}$, $\bar{x}_{j}^{t}=\frac{1}{n}\sum_{i=1}^{n} x_{ij}^{t}$, $\bar{x}_{j}^{\cdot}=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v} x_{i,tj}$,

$\sigma_{j}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_{j})^{2}}$, $\sigma_{j}^{t}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}^{t}-\bar{x}_{j}^{t})^{2}}$, $\sigma_{j}^{\cdot}=\sqrt{\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}(x_{i,tj}-\bar{x}_{j}^{\cdot})^{2}}$,

z_{ij} - znormalizowana wartość x_{ij}.

Źródło: [29], [66], [153].

2) określenie wspólnego kierunku preferencji zmiennych, zwykle poprzez przekształcenie cech o charakterze destymulant w cechy o właściwościach stymulant (postulat jednolitej preferencji).

Ponadto wymaga się czasami, aby w wyniku operacji normalizacji realizacje zmiennych uzyskiwały dwie dodatkowe własności, tzn.:

a) wszystkie wartości cech były nieujemne (postulat dodatniości),

b) wszystkie wartości cech mieściły się w określonym i znanym przedziale zmienności (postulat stałości rozstępu lub stałości wartości ekstremalnych).

Do najważniejszych metod statystycznej normalizacji zmiennych, uwzględnionych w prowadzonych badaniach, zalicza się:

- przekształcenia ilorazowe,
- standaryzację,
- unitaryzację.

Odpowiednie formuły obliczeniowe wykorzystane w pracy zawiera tablica 1.2.

W przypadku przekształceń ilorazowych punktem odniesienia poszczególnych realizacji cech są ich wartości minimalne, maksymalne, średnie lub suma wszystkich wartości danej zmiennej.

W przypadku standaryzacji punktem odniesienia jest odchylenie standardowe danej cechy. W rezultacie tego przekształcenia otrzymane zmienne mają średnie arytmetyczne równe zeru, a odchylenia standardowe równe jedności.

W przypadku unitaryzacji, jako punkt odniesienia wykorzystuje się rozstęp danej cechy, w rezultacie czego, nowe zmienne przyjmują wartości z przedziału [0,1].

1.6. Syntetyczne miary rozwoju

Syntetyczne miary rozwoju (SMR), określane też mianem taksonomicznych mierników rozwoju, zmiennych agregatowych lub zmiennych syntetycznych, umożliwiają określenie stanu badanego zjawiska złożonego za pomocą jednej agregatowej wielkości (liczby) (por. [126], [158]). Z uwagi na tę szczególną własność są one wykorzystywane przede wszystkim do liniowego porządkowania obiektów wielocechowych. Podstawowa idea konstrukcji SMR polega na zastąpieniu wartości wielu zmiennych diagnostycznych, opisujących dany obiekt, pewną wartością agregatową (skalarem), co sprawia, że możliwe jest określenie funkcji porządkującej rozpatrywane obiekty względem siebie.

W literaturze przedmiotu, która w dużej mierze jest dorobkiem polskiej myśli statystycznej, spotkać można wiele propozycji dotyczących tworzenia SMR. Jako przykłady wymienić można m.in.: miarę rozwoju gospodarczego Hellwiga [88][5], absolutny miernik rozwoju Cieślak [33], zmienną syntetyczną Bartosiewicz [13], [112], zmodyfikowaną miarę rozwoju gospodarczego Pluty [136], syntetyczną miarę rozwoju społeczno-gospodarczego Strahl [159], agregatową miarę rozwoju Borysa [30].

W schemacie konstrukcyjnym syntetycznych miar rozwoju wyróżnić można następujące istotne momenty decyzyjne:

- określenie jednolitego kierunku preferencji zmiennych,
- ustalenie współczynników wagowych zmiennych,
- określenie sposobu normalizacji wartości zmiennych,
- ustalenie współrzędnych punktu odniesienia,
- określenie postaci analitycznej funkcji syntetyzującej.

---

[5] Praca Hellwiga [63] zainicjowała badania dotyczące metod porządkowania liniowego, które zaowocowały licznymi propozycjami w tym zakresie, przedstawionymi m.in. w pracach: [2], [13], [14], [30], [33], [64], [65], [66], [68], [72], [112], [126], [134], [136], [159], [160], [181].

Tablica 1.3

Syntetyczne miary rozwoju

Lp.	SMR	Wartość SMR $q_i = \ldots$
	Formuły bezwzorcowe	
1	średnia arytmetyczna	$\sum_{j=1}^{m} w_j z_{ij}''$
2	średnia geometryczna	$\prod_{j=1}^{m} z_{ij}''^{w_j}$
3	średnia harmoniczna	$\frac{1}{\sum_{j=1}^{m} \frac{w_j}{z_{ij}''}}$
	Formuły wzorcowe	
4/11	odległość Hamminga	$\sum_{j=1}^{m} w_j \left\| z_{ij}'' - z_{0j}'' \right\|$
5/12	odległość Euklidesa	$\left[\sum_{j=1}^{m} w_j \left(z_{ij}'' - z_{0j}'' \right)^2 \right]^{1/2}$
6/13	odległość Jeffreysa-Matusita	$\sum_{j=1}^{m} w_j \left(\sqrt{z_{ij}''} - \sqrt{z_{0j}''} \right)^2$
7/14	odległość Braya-Curtisa	$\frac{\sum_{j=1}^{m} w_j \left\| z_{ij}'' - z_{0j}'' \right\|}{\sum_{j=1}^{m} w_j \left(z_{ij}'' + z_{0j}'' \right)}$
8/15	odległość Clarka	$\left[\sum_{j=1}^{m} w_j \left(\frac{z_{ij}'' - z_{0j}''}{z_{ij}'' + z_{0j}''} \right)^2 \right]^{1/2}$
9/16	odległość „Canberra"	$\sum_{j=1}^{m} w_j \frac{\left\| z_{ij}'' - z_{0j}'' \right\|}{\left(z_{ij}'' + z_{0j}'' \right)}$
10/17	odległość kątowa	$\frac{\sum_{j=1}^{m} w_j z_{ij}'' z_{0j}''}{\left[\sum_{j=1}^{m} w_j \left(z_{ij}'' \right)^2 \sum_{j=1}^{m} w_j \left(z_{0j}'' \right)^2 \right]^{1/2}}$

gdzie: z_{ij}'' - unormowana wartość j-tej cechy diagnostycznej,

$i = 1,2,\ldots,n$; n - liczba obiektów, m - liczba zmiennych,

w_j - współczynnik wagowy j-tej cechy diagnostycznej,

$$z_{0j}'' = \begin{cases} \min\limits_{i} \left\{ z_{ij}'' \right\} & \text{– dolny biegun zbioru} \\ \max\limits_{i} \left\{ z_{ij}'' \right\} & \text{– górny biegun zbioru} \end{cases}$$

Źródło: [65].

Sposoby identyfikacji charakteru zmiennych, wyznaczania współczynników wagowych oraz metody normalizacji zmiennych zostały scharakteryzowane w punktach 1.4 oraz 1.5. Pewnego usystematyzowania wymagają natomiast różnorodne propozycje dotyczące postaci funkcji agregującej oraz określania współrzędnych wzorca (antywzorca) rozwoju. Interesującą propozycję w tym zakresie przedstawił Grabiński [72], dzieląc SMR na bezwzorcowe i wzorcowe.

W przypadku bezworcowych SMR oblicza się wartości średnie zbiorów obserwacji opisujących poszczególne obiekty. Wykorzystuje się w tym celu formuły średniej arytmetycznej, geometrycznej oraz harmonicznej.

Wzorcowe SMR wykorzystują jako funkcję agregującą jeden z przedstawionych wcześniej mierników odległości, z tym, że wyznaczana jest odległość poszczególnych obiektów od pewnego hipotetycznego obiektu wzorcowego, za który przyjmuje się na ogół współrzędne dolnego lub górnego bieguna zbioru obserwacji.

Tablica 1.3 zawiera SMR wykorzystane w analizach symulacyjnych. Formuły oznaczone numerami 1, 2, 3 to bezwzorcowe mierniki syntetyczne, natomiast mierniki wzorcowe o numerach 4, 5, 6, 7, 8, 9, 10 obliczane były w oparciu o dolny biegun zbioru (antywzorzec), zaś mierniki oznaczone numerami 11, 12, 13, 14, 15, 16, 17 - w oparciu o górny biegun zbioru danych (wzorzec).

Uzyskane na podstawie tych formuł wartości zmiennych syntetycznych przekształca się następnie w agregatowe mierniki rozwoju (znormalizowane i umożliwiające prowadzenie analiz porównawczych) według wzoru [65, s. 91-92][6]:

$$q_i' = \frac{q_i}{\|\mathbf{Q}\|} \quad (i = 1,2,\ldots,n), \tag{1.35}$$

gdzie: q_i - wartość SMR dla i-tego obiektu,
$\mathbf{Q} = [q_i]$ - wektor zawierający SMR poszczególnych obiektów,
$\|\mathbf{Q}\|$ - norma wektora $\mathbf{Q}$ wyznaczona według wzoru: $\|\mathbf{Q}\| = \max_i\{q_i\} - \min_i\{q_i\}$.

[6] Skalowanie wartości zmiennej syntetycznej można przeprowadzić również w oparciu o alternatywne formuły przedstawione w pracach [64], [65].

1.7. Ocena jakości metod wielowymiarowej analizy porównawczej

Określenia o charakterze wartościującym, takie jak „efektywność", „jakość", „odporność", „poprawność", „stabilność" oraz „wrażliwość" niosą w sobie, zarówno w sensie semantycznym, jak i w intuicyjnym odbiorze podobne treści. Odnoszone do metod ekonometrycznych, statystycznych, numerycznych, czy też na gruncie teorii systemów, używane są w celu uwypuklenia pewnych specyficznych właściwości, które metody te powinny posiadać, aby mogły być uznane za wiarygodne i poprawne z punktu widzenia stawianych im zadań. Zjawiska i zdarzenia charakteryzowane przez te pojęcia posiadają zazwyczaj dość jednoznaczny kierunek preferencji (w przypadku „wrażliwości" kierunek ten jest przeciwny w stosunku do pozostałych terminów).

Na gruncie statystyki, ekonometrii i analizy numerycznej używa się tradycyjnie pojęć „odporność", „stabilność" lub „wrażliwość" (por. [85], [111], [119). Za stabilne uznaje się te metody numeryczne i modele ekonometryczne (w przypadku ekonometrii w szczególności dotyczy to metod estymacji parametrów oraz testowania hipotez), które odznaczają się wysokim poziomem niezmienności charakterystyk teoretycznych względem przekształceń modeli i danych [119].

W teorii i analizie systemów używane jest najczęściej pojęcie „stabilności". Nie oznacza ono jednak niezmienności charakterystyk systemu, ani też nie jest równoznaczne z brakiem jakiejkolwiek dynamiki jego elementów. Jak podkreśla Weinberg, stabilności, w odniesieniu do systemu, nie należy rozumieć jako sytuacji całkowitego braku zmian, ale raczej jako zmienność w ustalonych granicach (por. [128, s. 125-127]). Podkreśla się również, że stabilność systemu wyznaczana jest przez jego zdolność powrotu do stanu (punktu) równowagi (ang. *point attractor*). Roetzheim za system stabilny uważa taki, który zachowuje się w sposób powtarzalny i po wyjściu ze stanu równowagi wraca do właściwego położenia. Zwraca on ponadto

uwagę na to, iż pomiędzy dwoma skrajnymi położeniami jakimi są stabilność i chaos, występuje szereg stanów przejściowych, będących źródłem skomplikowanego i często nieprzewidywalnego zachowania systemu [146, s. 43-49].

W kontekście analizy metod taksonometrycznych częściej używane są pojęcia „jakość", „efektywność" oraz „poprawność". Potrzeba jakościowej oceny tych metod zrodziła się na skutek szybkiego postępu w dziedzinie taksonomii. Opracowano bardzo dużą liczbę algorytmów i procedur taksonomicznych, co spowodowało istotne trudności w wyborze najbardziej właściwych narzędzi w badaniach aplikacyjnych. W związku z tym pojawiły się prace analizujące formalną i praktyczną przydatność metod taksonometrycznych. Prowadzone były m.in. badania nad efektywnością procedur porządkowania liniowego (np. [65], [66], [72]) oraz jakością metod grupowania (np. [63], [65], [67], [69], [111], [140]). Poddawano w nich ocenie poszczególne elementy składowe algorytmów taksonometrycznych, takie jak systemy wag, metody normowania cech, formuły agregacji zmiennych czy metody analizy skupień (grupowania)[7].

Podstawę jakościowej oceny metod taksonometrycznych stanowią wyniki symulacyjnych eksperymentów numerycznych, analizowane pod kątem postulowanych własności oraz w aspekcie odporności zastosowanych procedur [65], [69]. Kierunek preferencji jest tutaj zgodny z intuicją, tzn. wyżej oceniane są metody przynoszące wyniki o własnościach najbardziej zbliżonych do postulowanych oraz najbardziej odporne na różnego rodzaju zaburzenia (zmiany) w wejściowych zbiorach danych.

Kierując się podstawowym celem pracy, zaprojektowano eksperyment symulacyjny, zmierzający do uzyskania wyników umożliwiających jakościową ocenę włączonych do badań ścieżek WAP. Każda taka ścieżka jest konfiguracją utworzoną z poszczególnych elementów

[7] W pracy [63] techniki symulacji cyfrowej stosowano również do oceny efektywności niektórych metod ustalania optymalnego wektora zmiennych objaśniających.

składowych procedury WAP, takich jak: współczynniki wagowe zmiennych, sposoby normalizacji zmiennych oraz formuły wyznaczania syntetycznych miar rozwoju (zob. rys. 1.2).

Każda z rozpatrywanych konfiguracji wartościowana jest na podstawie rezultatów przetwarzania generowanych losowo wejściowych zbiorów danych o różnych charakterystykach opisowych.

Oszacowanie jakości poszczególnych ścieżek WAP oraz ustalenie konfiguracji optymalnych dla pewnych rodzajów wejściowych zbiorów danych przebiegać będzie w oparciu o mierniki poprawności metod WAP.

Szeroki przegląd tego rodzaju mierników stosowanych do oceny efektywności metod taksonometrycznych (w szczególności dotyczących algorytmów porządkowania liniowego oraz metod grupowania) pod kątem ich własności i odporności zawierają prace Grabińskiego [65], [66], [67], [69], [72], Pociechy [140] i Luli [111].

Mierniki jakości wykorzystywane w pracy do szacowania efektywności badanych ścieżek WAP oraz optymalizacji wyboru zamieszczone są w tablicy 1.4. Dotyczą one w szczególności pomiaru i oceny takich własności procedur WAP, jak (por. [66], [72]):

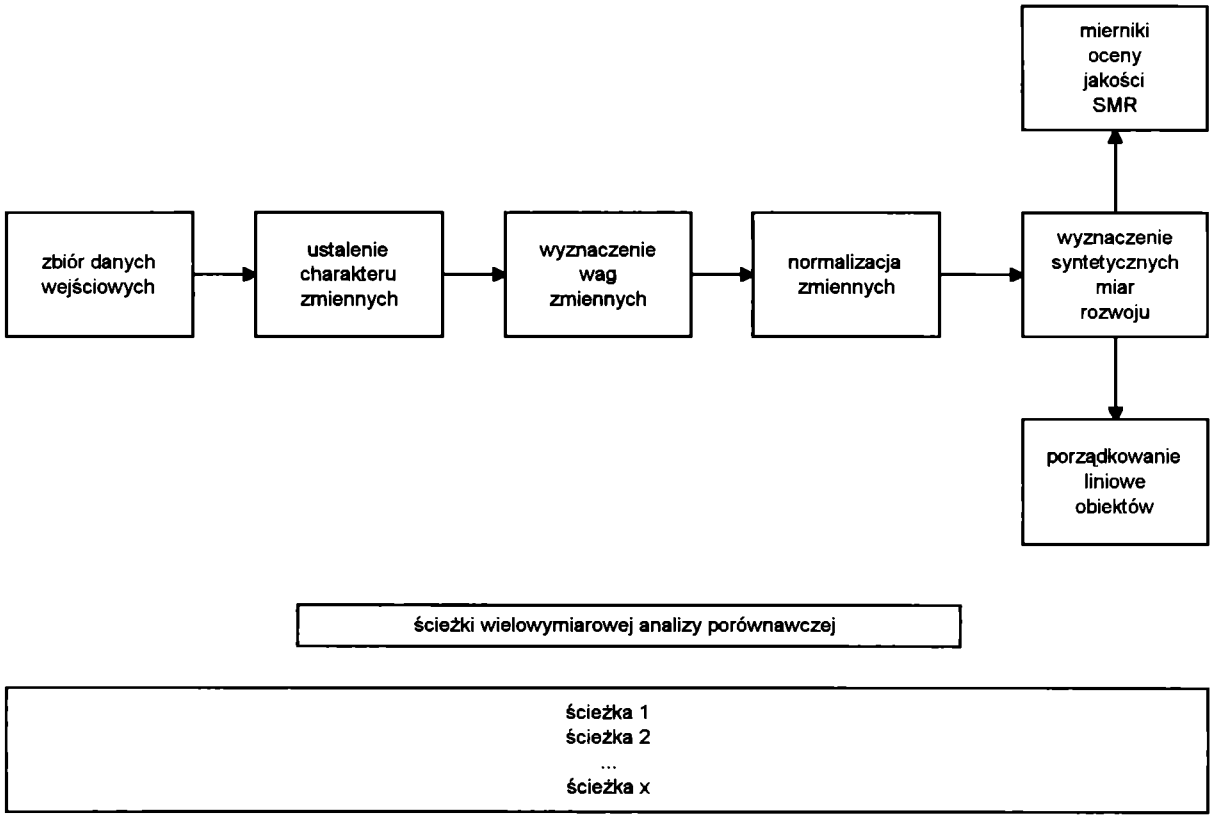
- zgodność odwzorowania, mierzona wskaźnikiem zróżnicowania odległości między obiektami w przestrzeni zmiennych diagnostycznych oraz w przestrzeni zmiennej syntetycznej (mierniki 1, 2, 3),
- korelacja liniowa pomiędzy zmienną syntetyczną a zmiennymi diagnostycznymi, mierzona przeciętnym współczynnikiem „nieokreśloności" (4) oraz współczynnikiem „jednoznaczności" zmiennej syntetycznej (5),
- korelacja rangowa zmiennej syntetycznej ze zmiennymi diagnostycznymi, mierzona współczynnikiem „nieokreśloności" (6), współczynnikiem „jednoznaczności" zmiennej syntetycznej (7) oraz uogólnionym rangowym współczynnikiem rozbieżności (8),

Tablica 1.4

Mierniki jakości metod wielowymiarowej analizy porównawczej

Lp.	Rodzaj miernika	Postać analityczna miernika	Oznaczenia
1	Zgodność odwzorowania	$\frac{\sum_{i=1}^{n-1}\sum_{j>i}^{n}\left(d_{ij}-\tilde{d}_{ij}\right)^2}{\sum_{i=1}^{n-1}\sum_{j>i}^{n}\tilde{d}_{ij}^2}$	$\tilde{d}_{ij}$ - odległość między i-tym oraz j-tym obiektem w m-wymiarowej przestrzeni cech diagnostycznych
2		$\frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j>i}^{n}\left(\frac{d_{ij}-\tilde{d}_{ij}}{\tilde{d}_{ij}}\right)^2$	d_{ij} - odległość między i-tym oraz j-tym obiektem w m-wymiarowej przestrzeni cech diagnostycznych
3		$\frac{\sum_{i=1}^{n-1}\sum_{j>i}^{n}\frac{\left(d_{ij}-\tilde{d}_{ij}\right)^2}{\tilde{d}_{ij}}}{\sum_{i=1}^{n-1}\sum_{j>i}^{n}\tilde{d}_{ij}}$	n - liczba obiektów
4	Korelacja liniowa zmiennej syntetycznej ze zmiennymi diagnostycznymi	$1-\frac{1}{m}\sum_{j=1}^{m}r_{qz_j}$	r_{qz_j} - współczynnik korelacji między zmienną syntetyczną oraz j-tą zmienną diagnostyczną m - liczba zmiennych diagnostycznych
5		$\frac{1}{m}\sum_{j=1}^{m}l_j$	$l_j = \begin{cases} 0 & \text{dla } r_{qz_j} \geq 0.5 \\ 1 & \text{dla } 0.0 \leq r_{qz_j} < 0.5 \\ 2 & \text{dla } -0.5 \leq r_{qz_j} < 0.0 \\ 3 & \text{dla } r_{qz_j} < -0.5 \end{cases}$
6	Korelacja rangowa zmiennej syntetycznej ze zmiennymi diagnostycznymi	$1-\frac{1}{m}\sum_{j=1}^{m}\rho_{qz_j}$	ρ_{qz_j} - współczynnik korelacji rang Spearmana między zmienną syntetyczną oraz j-tą zmienną diagnostyczną
7		$\frac{1}{m}\sum_{j=1}^{m}l_j$	l_j - ustalone dla współczynników ρ_{qz_j} jak we wzorze 5
8		$\frac{2}{ml}\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\dot{x}_{ij}-\dot{q}_i\right\|$	$\dot{x}_{ij}$ - ranga i-tego obiektu ze względu na j-tą zmienną pierwotną $\dot{q}_i$ - ranga i-tego obiektu ze względu na j-tą zmienną syntetyczną $l = \begin{cases} n^2 & \text{dla } n \text{ parzystego} \\ n^2-1 & \text{dla } n \text{ nieparzystego} \end{cases}$
9	Zmienność i koncentracja zmiennej syntetycznej	$\frac{s_q}{\bar{q}}$	$\bar{q}$, s_q - średnia i odchylenie standardowe zmiennej syntetycznej,
10		$\frac{s_\Delta}{\bar{\Delta}}$	$\bar{\Delta}$, s_Δ - średnia i odchylenie standardowe z wielkości: $\Delta_i = \tilde{q}_i - \tilde{q}_{i-1} \quad (i=1,\ldots,n-1)$, gdzie: $\tilde{q}_i$ - to uporządkowane niemalejąco realizacje zmiennej syntetycznej
11	Odległość taksonomiczna zmiennej syntetycznej od zmiennych diagnostycznych	$\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|z_{ij}^{'}-q_i^{'}\right\|$	$z_{ij}^{'}$ - standaryzowana wartość j-tej zmiennej pierwotnej dla i-tego obiektu
12		$\left[\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(z_{ij}^{'}-q_i^{'}\right)^2\right]^{1/2}$	$q_i^{'}$ - standaryzowana wartość zmiennej syntetycznej dla i-tego obiektu

Źródło: [63].



x - liczba możliwych ścieżek WAP, wynikająca z iloczynu liczby uwzględnionych współczynników wagowych, sposobów normalizacji zmiennych oraz syntetycznych miar rozwoju.

Rysunek 1.2. Procedura wielowymiarowej analizy porównawczej

- zmienność i koncentracja zmiennej syntetycznej, mierzone współczynnikiem obliczonym dla realizacji zmiennej syntetycznej (9) oraz dla pierwszych różnic uporządkowanych niemalejąco wartości zmiennej syntetycznej (10),
- przeciętna odległość taksonomiczna zmiennej syntetycznej od zmiennych diagnostycznych mierzona w oparciu o mierniki Hamminga (11) oraz Euklidesa (12).

Ze względu na bardzo dużą liczbę możliwych do uzyskania konfiguracji elementów składowych procedury WAP, koniecznością wydaje się skonstruowanie pewnego wskaźnika syntetycznego, ułatwiającego interpretację uzyskanych wyników. Poniżej przytoczono dwa przykłady wskaźników syntetycznych [148, s. 45-47]:

$$g^* = \sum_{i=1}^{q} \beta_i g_i \tag{1.36}$$

$$g^* = \sqrt{\sum_{i=1}^{q} \beta_i g_i^2} \tag{1.37}$$

gdzie: g^* - miernik agregatowy,
g_i - mierniki cząstkowe,
β_i - współczynniki wagowe mierników cząstkowych ($\beta_i \geq 0$ przy założeniu, że mierniki są zgodne i malejące wartości każdego z nich są korzystne; mierniki są zgodne, jeżeli zmniejszenie lub zwiększenie każdego z nich przynosi korzystne rezultaty),
q - liczba mierników cząstkowych.

W prowadzonych badaniach skorzystano z formuły (1.37), przy czym przyjęto jednakowe wagi dla poszczególnych mierników cząstkowych.

Rozdział 2

PRZYGOTOWANIE I CHARAKTERYSTYKA DANYCH WEJŚCIOWYCH

2.1. Generatory liczb losowych

Jednym ze sposobów badania właściwości metod WAP może być modelowanie symulacyjne, czyli próba określenia optymalnych warunków stosowania poszczególnych metod drogą eksperymentów symulacyjnych przeprowadzonych przy wykorzystaniu sprzętu komputerowego i odpowiedniego oprogramowania.

Podstawowym warunkiem prowadzenia eksperymentów symulacyjnych (w tym również modelowania symulacyjnego metod WAP) jest dysponowanie odpowiednimi zbiorami danych wejściowych lub narzędziami umożliwiającymi łatwe otrzymywanie takich zbiorów. Zakłada się przy tym, że dane te mają charakter losowy oraz spełniają inne nałożone postulaty (np. są zgodne z określonym rozkładem teoretycznym zmiennej losowej, posiadają żądane parametry opisowe, są elementem określonej przestrzeni liczbowej itp.). Wynika z powyższego, iż punktem wyjścia wszelkich badań symulacyjnych jest skonstruowanie odpowiedniego „urządzenia stochastycznego" umożliwiającego otrzymywanie liczb losowych, spełniających wyspecyfikowane własności. Przede wszystkim „urządzenie" takie powinno gwarantować, że każdy kolejny losowany element ma jednakowe szanse (prawdopodobieństwo) wyboru oraz, że pomiędzy poszczególnymi elementami losowanego ciągu nie istnieje żadna zależność statystyczna (kolejne elementy ciągu otrzymywane są w sposób niezależny, tzn. wartość elementu następne-

go w żadnym stopniu nie zależy od wartości elementu poprzedniego). Rao w pracy [145] napisał, że nie ma prostej definicji ciągu liczb losowych i podał własną, podkreślając jej nieprecyzyjność. Napisał mianowicie: „oczekuje się, że takie ciągi, zwane *ciągami liczb losowych*[1], będą przejawiać maksymalną nieokreśloność (chaos lub entropię) w tym sensie, że dany ciąg wylosowanych dotychczas cyfr nie daje żadnej wskazówki, aby odgadnąć wynik następnego ciągnienia" [145, s. 22-23].

Znane są różne źródła liczb losowych (np. ruletka, kostka do gry, urna z ponumerowanymi kulami, tablice liczb losowych, urządzenia fizyczne oparte na pomiarze siły promieniowania izotopów lub natężenia szumów elektronowych itp.), jednakże nie zawsze mogą być one w prosty sposób adaptowane do potrzeb modelowania symulacyjnego na maszynach cyfrowych. Z tych też powodów już w latach 1940-1950 rozpoczęto w Stanach Zjednoczonych prace zmierzające do szerokiego zastosowania rachunku prawdopodobieństwa i elektronicznych maszyn cyfrowych do rozwiązywania złożonych problemów praktycznych[2]. Jednym z nurtów tych badań były próby związane z poszukiwaniem analitycznych sposobów otrzymywania liczb losowych. Prekursorami tych badań byli matematycy amerykańscy[3]: Metropolis, von Neumann oraz Ulam. Ich wysiłki zaowocowały sformułowaniem podstawowych założeń metody Monte Carlo[4] oraz pierwszym analitycznym generatorem liczb losowych[5]. Analityczne metody produkowania liczb losowych szybko znalazły powszechne zastosowanie w symulacji komputerowej z uwagi na prostotę kodowania odpowiednich algorytmów oraz możliwość odtwarzania takich samym ciągów losowych bez konieczności ich przechowywania w pamięci zewnętrznej. Stało się tak pomimo tego, iż liczby losowe otrzymywane przy pomocy

---

[1] Wyróżnienie kursywą za cytowaną pracą [145].

[2] Z historycznego punktu widzenia przyjmuje się, że pierwszym zastosowaniem zjawisk losowych w procesach obliczeniowych była praca Halla pt. *On an experiment determination of* π, opublikowana w 1873 roku, traktująca o wyznaczaniu wartości liczby π za pomocą rzutów igły na poliniowaną równoległymi prostymi płaszczyznę papieru [94, s. 9].

[3] John von Neumann był z pochodzenia Węgrem, natomiast Stanisław Ulam wywodził się ze słynnej lwowskiej szkoły matematycznej.

[4] Terminu „metody Monte Carlo" użyli pierwszy raz Metropolis i Ulam w pracy pt. *The Monte Carlo methods*, wydanej w 1949 roku. Współtwórcą metody Monte Carlo był von Neumann.

[5] Generator kwadratowy przedstawiony przez von Neumanna w 1949 roku i zrealizowany na pierwszej w historii maszynie cyfrowej ENIAC [187].

programowych generatorów nie spełniają całkowicie wszystkich postulatów losowości. Generatory te wykorzystują bowiem odpowiednie formuły arytmetyczne do generowania ciągów liczbowych o takich własnościach, które pozwalają wprawdzie uznać otrzymane liczby za realizacje losowe, chociaż „przewidywanie" kolejnych wartości jest tylko kwestią znajomości formuły i wartości początkowych parametrów. Liczby otrzymywane przy pomocy generatorów programowych określa się w związku z tym jako liczby pseudolosowe (lub quasi-losowe) dla odróżnienia od liczb naprawdę losowych. Tak otrzymanym ciągom liczb losowych stawia się też różnorodne kryteria jakości, których spełnienie weryfikuje się przy pomocy rozmaitych testów statystycznych. Na podstawie wyników tych testów dany generator jest akceptowany jako „urządzenie" produkujące liczby wystarczająco losowe, bądź też odrzucany. Podstawowa trudność w konstruowaniu „dobrych" generatorów programowych polega na odpowiednim doborze stałych parametrów generatora.

Generatory liczb losowych podzielić można na dwie grupy (por. [40], [43], [53], [102], [169], [187] oraz zob. rys. 2.1):

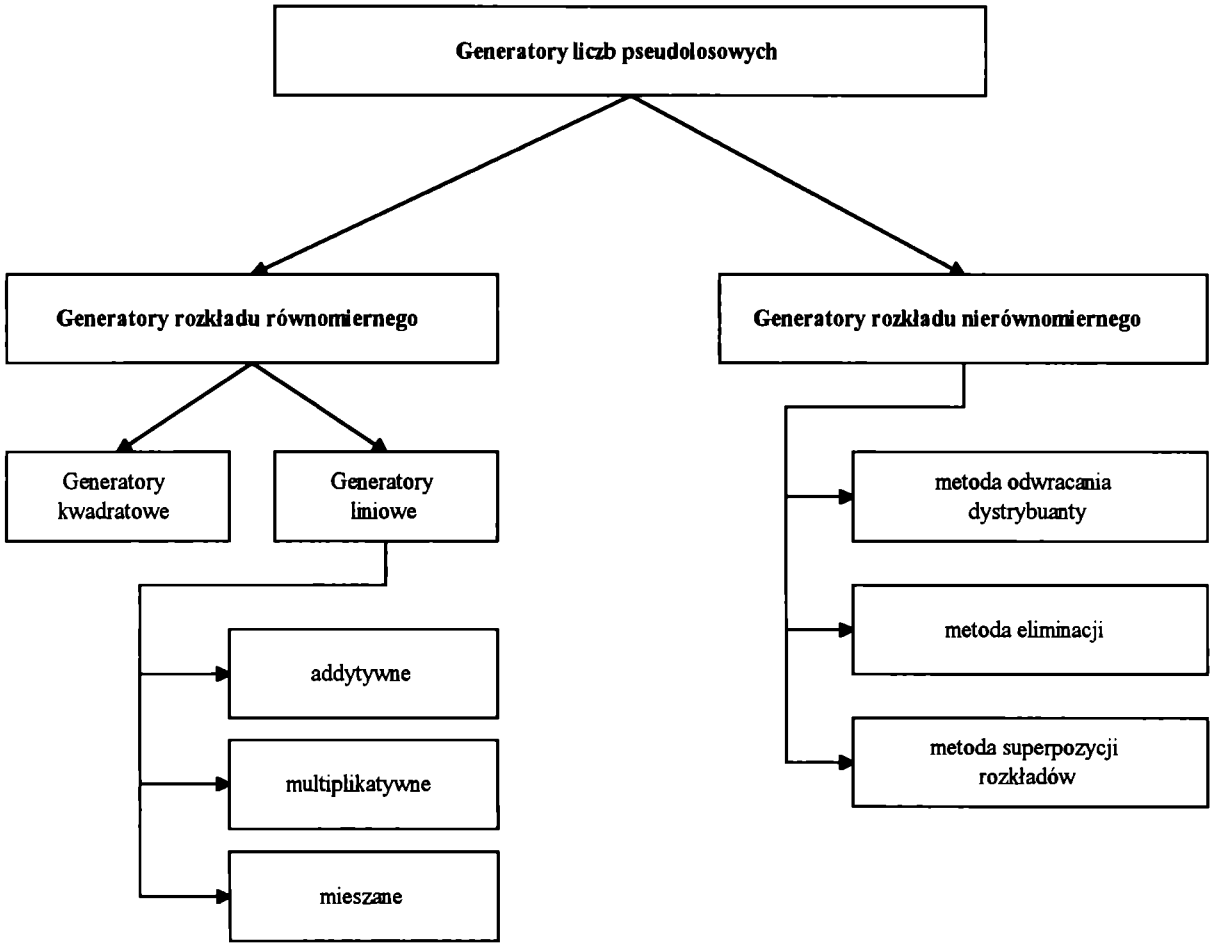
a) generatory liczb losowych o rozkładzie równomiernym,

b) generatory liczb losowych o rozkładzie dowolnym (nierównomiernym).

Generatory liczb losowych o rozkładzie równomiernym umożliwiają otrzymywanie liczb losowych takich, że jeżeli zmienna losowa przyjmuje każdą wartość z przedziału $[\alpha,\beta]$ z jednakowym prawdopodobieństwem, to funkcja gęstości tej zmiennej jest stała i równa:

$$F(x) = \begin{cases} \dfrac{1}{\alpha - \beta} & \text{dla } \alpha \le x \le \beta \\ 0 & \text{w pozostałych przypadkach} \end{cases} \qquad (2.1)$$

gdzie: $F(x)$ - funkcja gęstości prawdopodobieństwa,
α,β - granice przedziału.



Rysunek 2.1. Metody generowania liczb pseudolosowych

W szczególności generatory te wykorzystywane są do otrzymywania liczb losowych z przedziału [0,1] i wówczas mówimy o symulacji metodami Monte Carlo[6]. Ponadto generatory liczb losowych o rozkładzie jednostajnym służą do produkowania liczb losowych o innych rozkładach. Wśród generatorów o rozkładzie równomiernym wyróżnia się generatory kwadratowe oraz generatory liniowe.

Generator kwadratowy był historycznie pierwszym generatorem programowym (podał go von Neumann w 1949 roku). Generator ten oparty jest na następującej zależności rekurencyjnej:

$$x_{i+1} = \left\lfloor x_i^2 \times P^{-\mu/2} \right\rfloor - \left\lfloor x_i^2 \times P^{-3\mu/2} \right\rfloor \times P^{\mu} \qquad (2.2)$$

[6] Metody Monte Carlo znajdują zastosowanie w teorii masowej obsługi, teorii gier, ekonomii matematycznej, rozwiązywaniu równań różniczkowych, całkowaniu numerycznym, zadaniach optymalizacji, odwracaniu macierzy, obliczaniu wartości i wektorów własnych itp. [94], [188].

gdzie: μ - liczba cyfr wyrazu x_i,
P - podstawa systemu liczbowego,
$\lfloor x \rfloor$ - oznacza całkowitą część liczby x.

Generatory liniowe kongruencyjne[7] konstruowane są w oparciu o następującą ogólną formułę [94, s. 43]:

$$x_{i+k} = \left(\sum_{j=0}^{k-1} \alpha_j x_{i+j} + \beta \right) \bmod \mu \qquad \left(\alpha_j \neq 0;\ k > 1 \right) \tag{2.3}$$

gdzie: x_i, α_j oraz β są liczbami całkowitymi z przedziału lewostronnie domkniętego $[0, \mu)$,
μ - określona, dodatnia liczba całkowita,
mod - operator dzielenia modulo przez moduł μ.

Ponieważ w wyniku dzielenia modulo otrzymuje się resztę z dzielenia przez moduł μ, więc liczba całkowita x_i otrzymana w oparciu o (2.3) należy do przedziału $[0, \mu - 1]$. Przekształcenie x_i / μ przeskalowuje liczbę x_i na przedział [0,1) [40, s. 19]. Okres kongruencyjnych generatorów liniowych wyprowadzanych na podstawie wzoru (2.3) definiuje się jako najmniejszą liczbę λ, dla której;

$$(x_0, \ldots, x_{k-1}) = (x_\lambda, \ldots, x_{\lambda+k-1}). \tag{2.4}$$

Do najbardziej znanych i najpowszechniej stosowanych w praktyce realizacji powyższej formuły ogólnej należą generatory addytywne, multiplikatywne i mieszane.

Generator addytywny określony następującą formułą:

$$x_{i+1} = (x_i + x_{i-1}) \bmod \mu \tag{2.5}$$

nazywany jest generatorem Fibonacciego.

Generator multiplikatywny wyrażony jest formułą następującej postaci:

$$x_{i+1} = (\alpha x_i) \bmod \mu \tag{2.6}$$

gdzie: α oznacza stały mnożnik.

Generator mieszany określony jest następującym wzorem:

---

[7] O dwóch liczbach całkowitych mówi się, że przystają według modułu μ (gdzie μ jest liczbą naturalną), jeżeli ich różnica jest podzielna przez μ. Określenie to, jak również wiele innych szczegółów dotyczących tego zagadnienia, zawiera praca Sierpińskiego [151].

$$x_{i+1} = (\alpha x_i + \beta) \bmod \mu \qquad (\beta \neq 0) \tag{2.7}$$

gdzie: α i β są odpowiednio dobranymi wartościami stałymi.

Liczby losowe o rozkładzie jednostajnym wykorzystywane są do generowania ciągów losowych o innych, dowolnych rozkładach. W celu otrzymania zadanego rozkładu należy najpierw generować liczby losowe o rozkładzie równomiernym, a następnie przekształcać je do postaci żądanego rozkładu przy pomocy odpowiednich metod. Do najczęściej stosownych w praktyce metod konwersji rozkładu równomiernego w rozkład dowolny należą: metoda odwracania dystrybuanty, metoda eliminacji oraz metoda superpozycji rozkładów.

W *metodzie odwracania dystrybuanty* wykorzystuje się generatory liczb losowych o rozkładzie równomiernym na przedziale [0,1), a otrzymane z nich ciągi liczb poddaje się przekształceniu tak, aby uzyskać nowy ciąg o żądanym rozkładzie. Przekształcenie to można zapisać wzorem:

$$X = F^{-1}(R) \tag{2.8}$$

gdzie: X - zmienna losowa o dystrybuancie $F(x)$ lewostronnie ciągłej i niemalejącej,
R - zmienna losowa o rozkładzie równomiernym na przedziale [0,1),
$F^{-1}(R)$ - funkcja odwrotna do dystrybuanty F.

W omawianej metodzie niekiedy trudną do pokonania barierą może okazać się określenie postaci funkcji $F^{-1}(R)$ lub znaczna złożoność numeryczna algorytmu i wówczas korzysta się z pewnych aproksymacji funkcji odwrotnej do dystrybuanty F.

Metoda eliminacji przedstawiona w 1951 r. przez von Neumanna umożliwia generowanie ciągu liczb losowych o zadanym rozkładzie prawdopodobieństwa w oparciu o pary niezależnych liczb losowych o rozkładzie równomiernym, które poddawane są odpowiednim przekształceniom. Jeżeli dana jest zmienna losowa X o funkcji gęstości prawdopodobieństwa $f(x)$ określonej na przedziale (a,b) oraz równej 0 poza tym przedziałem i dodatkowo $f(x)$ jest ograniczona pewną stałą c ($c>0$), to wówczas liczby losowe o gęstości rozkładu $f(x)$ generuje się według algorytmu 2.1 [187].

Algorytm 2.1

Generowanie liczb losowych o rozkładzie $f(x)$ metodą eliminacji

K01: Wylosować parę niezależnych liczb losowych: (r_1, r_2), przy czym r_1 jest realizacją zmiennej losowej o rozkładzie równomiernym na przedziale (a,b), zaś r_2 jest realizacją zmiennej losowej o rozkładzie równomiernym na przedziale $(0,c)$. Liczby r_1 i r_2 losuje się przy pomocy generatora liczb losowych o rozkładzie równomiernym na przedziale (0,1), a następnie lokalizuje w żądanych przedziałach (a,b) i $(0,c)$ korzystając - odpowiednio - z przekształceń $(b-a)\cdot r_1 + a$ oraz $c \cdot r_2$.

K02: Jeżeli $r_2 \leq f(r_1)$, to wówczas przyjmuje się $x_1 = r_1$, zaś w przeciwnym razie eliminuje się parę (r_1, r_2) i wraca się do punktu K01.

Metoda superpozycji (składania) rozkładów, zaproponowana w 1956 r. przez Butlera, opiera się na wykorzystaniu generatorów liczb losowych o znanych rozkładach prawdopodobieństwa do produkowania liczb losowych o żądanym rozkładzie na drodze pewnych przekształceń funkcji gęstości. Bezwarunkowy rozkład (prawdopodobieństwo całkowite) zmiennej losowej X dany jest wzorem:

$$f(x) = \int_{-\infty}^{\infty} g_y(x) h(y)\, dy \tag{2.9}$$

gdzie: X - zmienna losowa, której gęstość rozkładu $g_y(x)$ zależna jest od parametru y,
y - parametr, będący zmienną losową o gęstości rozkładu h.

Zmienną losową X o gęstości rozkładu wyrażonej wzorem (2.9) proponuje się w omawianej metodzie generować według następującego algorytmu 2.2

Algorytm 2.2

Generowanie liczb losowych o rozkładzie $f(x)$ metodą superpozycji rozkładów

K01: Wylosować liczbę y zgodnie z rozkładem o gęstości h,

K02: Wylosować liczbę x zgodnie z gęstości g_y, przy czym y jest tutaj parametrem wylosowanym w punkcie K01.

Do najczęściej stosowanych generatorów rozkładów różnych od równomiernego należą: generatory rozkładu normalnego, dwumianowego, beta, wykładniczego, hiperwykładniczego, gamma, potęgowego, Bernoulliego, Erlanga, Poissona, Weibulla. Charakterystyki generatorów tych i inych rozkładów zawierają prace [40], [53], [169].

W tablicach 2.1-2.4 zestawiono przykłady generatorów rozkładu równomiernego najczęściej wykorzystywanych w implementacjach komputerowych. Większość z nich została

opracowana w latach 1950-1970 dla takich komputerów, jak: IBM-7090, IBM 360, CDC-6600, UNIVAC-1108, ODRA-1304, ODRA-1305. Proste przeniesienie tych generatorów na mikrokomputery klasy IBM PC nie jest możliwe z uwagi na różnice w długości słowa maszynowego poszczególnych procesorów. Pojawia się w związku z tym konieczność odpowiedniej modyfikacji znanych generatorów lub poszukiwania nowych. Do procedur produkujących ciągi liczb losowych na podstawie liniowych generatorów kongruencyjnych, wprowadza się czasami pewne dodatkowe przekształcenia i modyfikacje, zmierzające do polepszenia własności statystycznych tychże ciągów. W szczególności, manipulacje na ciągach generowanych liczb losowych mają na celu uzyskanie takich realizacji, które czyniłyby zadość również pewnym elementarnym własnościom rozkładów wielowymiarowych. Niektóre z tych metod omówione są w [187, s. 51-53], natomiast odpowiednie algorytmy zamieszczone są w [40, s. 35-36]. Podstawowe przekształcenia tego rodzaju, polegają m.in. na korzystaniu z układu kilku generatorów liniowych, pewnych manipulacjach na bitach poszczególnych liczb oraz na odpowiednich przesunięciach elementów generowanych ciągów [40], [48], [187].

Tablica 2.1

Generatory multiplikatywne

Lp.	Zarodek x_0	Mnożnik α	Moduł μ
1	47594118	23	100000001
2	1	1220703125	549755813888
3	1	762939453125	4398046511104
4	1	100003	10000000000
5	1	131075	34359738368
6	1	262147	34359738368
7	1	5	34359738368
8	1	3125	67108846
9	1	65539	2147483648
10	1	65549	2147483648
11	1	10396840753	34359738368
12	1	262147	33554432
13	1	8355611	34359738368
14	1	13619301789	34359738368
15	1	8192	67101323
16	1	8192	67099547
17	1	32768	16775723
18	1	152587890625	549755813888
19	1220703125	1220703125	2147483648

Źródło: [71, s. 60-61]

Przykładami praktycznych realizacji takich przekształceń są generatory opublikowane przez Wichmanna i Hilla oraz Margsalię i Zamana, charakteryzujące się bardzo dobrymi własnościami statystycznymi oraz długimi okresami. W 1987 r. Wichmann i Hill zaproponowali złożony, kombinowany generator liniowy dla procesorów 16-bitowych o okresie rzędu $6{,}95 \cdot 10^{12}$ [48]. Podstawowe parametry tego generatora zawiera tablica 2.4. Również w 1987 r. Marsaglia i Zaman przedstawili generator złożony z układu równań liniowych, opartych na ciągu Fibonacciego[8]. Okres tego generatora wynosi 2^{144}.

Tablica 2.2

Generatory mieszane

Lp.	Zarodek x_0	Stała β	Mnożnik α	Moduł μ
1	0	1	129	34359738368
2	0	1	101	10000000000
3	314159265	1	262145	34359738368
4	0	2718281829	3141592653	34359738368
5	1000	457	625	100000
6	1000	457	1001	100000

Źródło: [71, s. 62]

Tablica 2.3

Generator addytywny (Fibonacciego)

Lp.	Wyraz x_0	Wyraz x_1	Moduł μ
1	0	1	17592186044416

Źródło: [31, s. 212]

Tablica 2.4

Generator liniowy kombinowany

Nr równania	Zarodek x_0	Stała β	Mnożnik α	Moduł μ
1	1	2	171	177
2	10000	35	172	176
3	3000	63	170	178

Źródło: [48, s. 34]

[8] Generator ten został opublikowany przez G. Marsaglię i A. Zamana w raporcie „*Toward a Universal Random Number Generator*", Florida State University Report: FSU-SCRI-87-50 (1987). Jest to generator, który przeszedł pozytywnie wszystkie testy stosowane do oceny jakości generatorów liczb losowych i jest uznawany za najlepszy z obecnie znanych.

2.2. Statystyczna ocena jakości generatorów liczb losowych

Otrzymane z generatorów programowych ciągi liczb (zbiory danych statystycznych) nie są w istocie losowe, ponieważ są one całkowicie zdeterminowane przez zadawane wartości początkowe parametrów. Tym niemniej jednak „dobre" generatory programowe produkują ciągi liczb pseudolosowych o takich właściwościach, które pozwalają uznać te ciągi za realizacje rzeczywiście lub wystarczająco losowe (jak gdyby były one otrzymane przy pomocy urządzenia całkowicie losowego). O „dobroci" danego generatora rozstrzyga się w oparciu o zbiór właściwości $W_1, W_2, \ldots$, które powinien on posiadać. Jeżeli którejkolwiek z tych właściwości nie spełnia wygenerowany ciąg $\{x_n\}$ z zadanym prawdopodobieństwem $1-\alpha$, to generator programowy, z którego ten ciąg otrzymano, jest dyskwalifikowany na poziomie istotności α. Nie ma oczywiście żadnych gwarancji, że generator posiadający żądane właściwości statystyczne spełni również wszystkie inne testy, które można byłoby skonstruować jako kryteria oceny jego jakości. Można jedynie sformułować ogólną zasadę, że im więcej różnych testów spełnia dany generator, tym większe można mieć do niego zaufanie.

Jakość programowych generatorów liczb losowych weryfikuje się przy pomocy rozmaitych testów statystycznych, które podzielić można na dwie podstawowe grupy (por. [45], [53], [71], [169], [187]):

1) testy zgodności rozkładu, takie jak np.:

- test średniej,
- test chi-kwadrat,
- test Kołmogorowa.

2) testy niezależności, takie jak np.:

- test serii,
- test autokorelacji,

- testy kombinatoryczne (np. test pokerowy, test kolekcjonera).

Test średniej polega na sprawdzeniu, czy otrzymana empirycznie z wygenerowanego ciągu $\{x_n\}$ średnia arytmetyczna $\bar{x}_n$ różni się istotnie od wartości oczekiwanej średniej arytmetycznej $E(\bar{x}_n)$. Dla rozkładu równomiernego wartość oczekiwana $E(\bar{x}_n)=\frac{1}{2}$, jeżeli wszystkie realizacje zmiennej losowej X_i są rzeczywiście niezależne.

Test zgodności χ^2 jest nieparametrycznym testem częstości pozwalającym sprawdzić, czy liczebności klas w wygenerowanym ciągu $\{x_n\}$ są zbliżone do liczebności teoretycznych. Kryterium weryfikacji stanowi statystyka postaci:

$$\chi^2=\sum_{i=1}^{k}\frac{(n_i-np_i)^2}{np_i} \tag{2.10}$$

gdzie: n - długość ciągu $\{x_n\}$,
k - liczba klas,
n_i - liczebność i-tej klasy,
p_i - prawdopodobieństwo, że zmienna losowa o danym rozkładzie przyjmie wartość z i-tej klasy.

Test zgodności λ *Kołmogorowa* jest nieparametrycznym testem częstości sprawdzającym, na ile wartości dystrybuanty wylosowanego ciągu $\{x_n\}$ są zbieżne z wartościami teoretycznymi dystrybuanty danego rozkładu. Należy obliczyć bezwzględne różnice między wartościami obu dystrybuant, aby znaleźć wartość statystyki będącej kryterium porównywania, tzn.

$$D=\max_x\left|F_n(x)-F(x)\right| \tag{2.11}$$

oraz

$$\lambda=D\sqrt{n} \tag{2.12}$$

gdzie: $F_n(x)$ - dystrybuanta empiryczna,
$F(x)$ - dystrybuanta teoretyczna.

Test serii jest nieparametrycznym testem istotności umożliwiającym sprawdzenie losowości otrzymanego ciągu liczb $\{x_n\}$. Należy wszystkie elementy ciągu $\{x_n\}$ podzielić na dwa rozłączne podzbiory A i B, a następnie utworzyć ciąg $\{y_n\}$ o elementach:

$$y_i = \begin{cases} a, & \text{jeżeli } x_i \in A \\ b, & \text{jeżeli } x_i \in B \end{cases} \tag{2.13}$$

W ciągu $\{y_n\}$ powstaną serie złożone z następujących kolejno po sobie liczb losowych tej samej klasy (w ciągu $\{y_n\}$ są one reprezentowane literałami a i b). Liczba serii stanowi statystykę, będącą sprawdzianem testu.

Test autokorelacji umożliwia weryfikację hipotezy o niezależności poszczególnych elementów branych parami wygenerowanego ciągu $\{x_n\}$. W przypadku rozkładu równomiernego estymatorem autokowariancji określającej poziom zależności między elementami ciągu $\{x_n\}$ jest następująca formuła:

$$\hat{R}_k = \frac{1}{n-k} \sum_{i=1}^{n-k} [(X_i - \overline{X})(X_{i+k} - \overline{X})] \tag{2.14}$$

gdzie: $\overline{X} = \frac{1}{2}$,

k - opóźnienie między dwoma realizacjami zmiennej losowej X.

Należy następnie sprawdzić, czy otrzymane wartości $\hat{R}_k$ różnią się istotnie od zera.

Testy kombinatoryczne są testami istotności operującymi na podciągach wygenerowanego ciągu $\{x_n\}$, otrzymanych za pomocą metod analizy kombinatorycznej. Ciąg $\{x_n\}$ przekształca się zgodnie z wybranym schematem do ciągu $\{y_n\}$, który jest następnie dzielony na k-elementowe rozłączne grupy. Weryfikacja polega na sprawdzeniu hipotezy, czy prawdopodobieństwa występowania poszczególnych rodzajów k-elementowych grup otrzymanych z ciągu $\{y_n\}$ są zgodne z prawdopodobieństwami teoretycznymi.

W niniejszej pracy nie prowadzono badań, dotyczących jakości i efektywności generatorów liczb losowych. W wyborze generatorów programowych stosowanych na etapie przygo-

towywania zbiorów danych, kierowano się ocenami i wynikami testów zamieszczonymi w literaturze przedmiotu. Wykorzystano m.in. generator biblioteczny kompilatora Borland C/C++ oraz generator rozkładu równomiernego oparty na ciągu Fibonacciego.

2.3. Generowanie prób losowych

Dane liczbowe (realizacje zmiennych opisujących analizowane obiekty z ewentualnym uwzględnieniem czynnika czasu), będące przedmiotem przetwarzania metodami WAP gromadzone są w strukturach (tablicach wielowymiarowych) o postaci:

a) macierzy obserwacji

$$\mathbf{X} = \left[x_{ij}\right] \qquad (i = 1,2,\ldots n;\ j = 1,2,\ldots,m) \tag{2.15}$$

gdzie: n - liczba obiektów,
m - liczba zmiennych.

b) kostki obserwacji

$$\mathbf{X} = \left[x_{ijt}\right] \quad (i = 1,2,\ldots,n;\ j = 1,2,\ldots,m;\ t = 1,2,\ldots,v) \tag{2.16}$$

gdzie: n i m - jw.,
v - liczba analizowanych lat (okresów lub momentów czasu).

Nie wszystkie przedstawione wyżej generatory liczb losowych (zmiennych losowych) mogą być bezpośrednio wykorzystane w praktyce do produkowania zbiorów danych w postaci tego typu struktur. Jest to, najogólniej rzecz biorąc, uwarunkowane okresem danego generatora, złożonością obliczeniową algorytmu mierzoną rozmiarem potrzebnej pamięci i czasem

działania, poziomem trudności kodowania odpowiednich algorytmów. Najczęściej w zastosowaniach wykorzystywane są następujące rozwiązania:

a) generowanie danych statystycznych z generatorów rozkładu równomiernego,

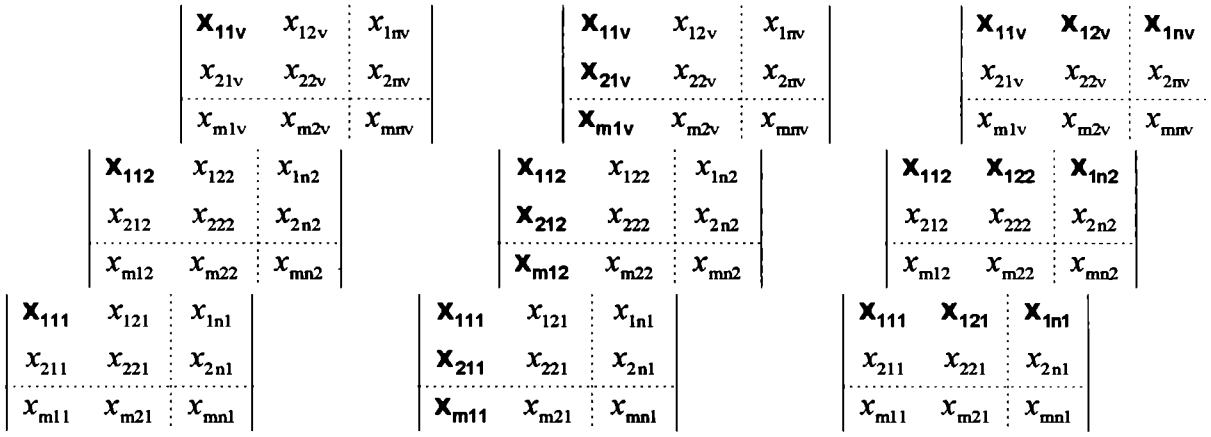
b) generowanie danych statystycznych z wielowymiarowego generatora rozkładu normalnego,

c) generowanie danych statystycznych z generatora procesów stochastycznych.

W przypadku *generowania danych statystycznych z generatorów rozkładu równomiernego* sposób i kolejność produkowania poszczególnych elementów struktur macierzowych nie są istotne jeżeli dysponujemy „urządzeniem stochastycznym" produkującym liczby rzeczywiście losowe. W sytuacji, w której do otrzymywania zbiorów danych w postaci powyższych struktur wykorzystujemy programowe generatory liczb losowych o określonym okresie i zgodności właściwiej jest generować i umieszczać kolejne otrzymywane liczby w tych strukturach zgodnie z algorytmami, które minimalizują ryzyko nierównomierności rozkładu i przekroczenia okresu danego generatora. W konstrukcji takich algorytmów szczególnie przydatne mogą być procedury wykorzystywane do przetwarzania tablic rozproszonych (np. Niektóre funkcje mieszające) oraz pewne algorytmy teorii grafów (zob. [42], [89]). W dalszym ciągu przedstawione zostaną propozycje rozwiązania omawianego problemu. Ponieważ macierz obserwacji potraktować można jako uproszczoną postać kostki obserwacji, proponowane algorytmy dotyczyć będą tej drugiej struktury.

Pierwsza propozycja oparta jest na idei zamieszczonej w pracy [187] i polega na jednoczesnym wykorzystaniu wielu generatorów tak, aby poszczególne elementy kostki w trzech podstawowych przekrojach otrzymywane były z różnych generatorów. Ilustruje to rys. 2.2.

Druga propozycja oparta jest na algorytmie przeszukiwania losowego tablic rozproszonych (zob. [89]) i polega na jednoczesnym wykorzystaniu czterech generatorów, z których jeden służy do losowania wartości elementu, natomiast trzy pozostałe do losowego wyznaczania adresu otrzymanej liczby w kostce danych.



a) elementy losowane *warstwami* zgodnie z formułą:
$x_{ijt} = (\alpha_{ij} x_{ijt-1}) \bmod \mu_i$

b) elementy losowane *kolumnami* zgodnie z formułą:
$x_{ijr_t} = (\alpha_{ij} x_{ijr-1_t}) \bmod \mu_i$

c) elementy losowane *wierszami* zgodnie z formułą:
$x_{i^s jt} = (\alpha_{ij} x_{i^{s-1} jt}) \bmod \mu_i$

Rysunek 2.2. Metody generowania elementów kostki danych

Idea tej propozycji przedstawiona jest na rysunku 2.3. Należy podkreślić, iż w tym przypadku szczególnie istotna jest jakość generatorów obliczających adresy, wskazujące miejsce elementu w kostce danych. Idealnym rozwiązaniem byłyby generatory dostarczające liczb z przedziałów 1,...,*m;* 1,...,*n;* 1,...,*v* w taki sposób, aby wybrana została każda wartość dokładnie jeden raz. W przeciwnym przypadku powstaje niebezpieczeństwo, iż pozostaną w kostce danych elementy o nieokreślonych wartościach. Niemniej jednak z sytuacją taką spotykamy się również w praktyce, kiedy np. dane empiryczne są niekompletne lub niepełne i zachodzi konieczność interpolacji bądź ekstrapolacji brakujących wartości zmiennych. W związku z tym algorytm ten może być interesujący również z tego punktu widzenia. Ogólną procedurę postępowania przedstawia algorytm 2.3.

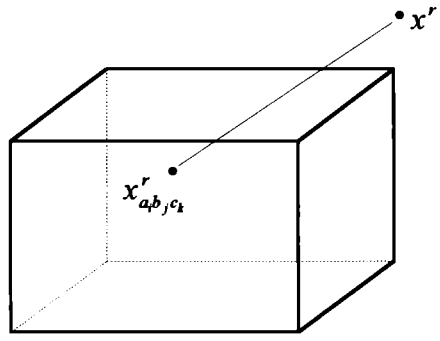
Algorytm 2.3

Generowanie elementów kostki danych

K01: Wyznaczyć liczbę losową według formuły: $x_i = (\alpha \cdot x_{i-1}) \bmod$,

K02: wyznaczyć największą liczbę całkowitą z iloczynu: $l_i = \lfloor x_i \cdot h \rfloor$, gdzie h oznacza jeden z wymiarów kostkidanych, zaś $\lfloor x \rfloor$ największą część całkowitą liczby x,

K03: Wyznaczyć adres w kostce danych: $q_i = (q_0 + l_i) \bmod h$, gdzie q_0 oznacza adres początkowy wylosowany lub przyjęty arbitralnie z przedziału 1,...,h



$$x^r = (\alpha_1 \cdot x^{r-1}) \bmod \mu,$$
$$a_i = (\alpha_2 \cdot a_{i-1}) \bmod \ ,$$
$$b_j = (\alpha_3 \cdot b_{j-1}) \bmod 1,$$
$$c_k = (\alpha_4 \cdot c_{k-1}) \bmod 1.$$

Rysunek 2.3. Generowanie wartości i adresów elementów kostki danych

Przedstawione propozycje mają charakter wstępnych założeń, które wymagają uszczegółowienia i weryfikacji przynajmniej z dwóch powodów. Po pierwsze należy poddać rzetelnym testom statystycznym wszystkie stosowane w badaniach symulacyjnych generatory liczb losowych, ponieważ sprawdzonych i zadowalających generatorów nie ma w literaturze przedmiotu zbyt wiele. Po drugie zaś, należy poddać eksperymentom symulacyjnym proponowane schematy konstruowania kostki danych pod kątem ich efektywności obliczeniowej.

Bardzo często w różnego rodzaju badaniach empirycznych, dotyczących zbiorowości statystycznych, wykorzystywane są próby w postaci wektorów losowych o wielowymiarowym rozkładzie normalnym. Wektory te otrzymywane są przy pomocy specjalnie skonstruowanego generatora *danych statystycznych o wielowymiarowym rozkładzie normalnym dla zadanej macierzy wariancji-kowariancji*. Wynika to z faktu, iż zazwyczaj w badaniach o charakterze ekonomicznym zakłada się określone skorelowanie między poszczególnymi zmiennymi losowymi. Wielowymiarowe wektory losowe można wygenerować na mocy twierdzenia głoszącego, że jeżeli wektor losowy $\mathbf{Z}' = [Z_1, Z_2, ..., Z_m]$ ma wielowymiarowy rozkład normalny o parametrach $N(0,1)$, to wektor losowy $\mathbf{X}' = [X_1, X_2, ..., X_m]$ można przedstawić w postaci:

$$\mathbf{X} = \mathbf{HZ} + \boldsymbol{\mu} \tag{2.17}$$

gdzie: **H** - macierz trójkątna dolna spełniająca zależność: $\mathbf{HH}' = \Sigma$,
Σ - macierz wariancji-kowariancji,

μ - wektor średnich.

Elementy macierzy $\mathbf{H} = \left[h_{ij}\right]_{m\times m}$ wyznacza się, korzystając z macierzy wariancji-kowariancji $\Sigma = \left[\sigma_{ij}\right]_{m\times m}$, metodą pierwiastków kwadratowych na podstawie następujących wzorów (por. [43, s. 565-566], [53, s. 232-233], [63, s. 266-267]):

$$h_{i1} = \frac{\sigma_{i1}}{\sqrt{\sigma_{11}}} \qquad \text{dla } 1 \le i \le m$$

$$h_{ii} = \sqrt{\sigma_{ii} - \sum_{j=1}^{i-1} h_{ij}^2} \qquad \text{dla } 1 < i \le m \qquad (2.18)$$

$$h_{ij} = \begin{cases} \dfrac{\sigma_{ij} - \sum_{k=1}^{j-1} h_{ik} h_{jk}}{h_{jj}} & \text{dla } 1 < j < i \le m \\ 0 & \text{dla } i < j \le m \end{cases}$$

gdzie: $\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \vdots & \sigma_{1m} \\ \sigma_{21} & \sigma_{22} & \vdots & \sigma_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{m1} & \sigma_{m2} & \vdots & \sigma_{mm} \end{bmatrix}$,

$\sigma_{ij} = E\{[X_i - E(X_i)][X_j - E(X_j)]\}$ dla $i \neq j$,

$\sigma_{ii} = E\left\{\left[X_i - E(X_i)\right]^2\right\}$ dla $i = j$.

Algorytm 2.4, przedstawiający procedurę otrzymywania wektorów losowych według powyższej metody, wykorzystano do generowania zbiorów danych, będących przedmiotem przetwarzania metodami WAP w trakcie analizy symulacyjnej.

Algorytm 2.4

Generowanie *n* wektorów losowych o rozkładzie normalnym

K01: Dany jest wektor średnich: $\mu = \left[\overline{\mu}_j\right]_m$ oraz macierz wariancji-kowariancji $\Sigma = \left[\sigma_{ij}\right]_{m\times m}$

K02: Obliczyć macierz **H**, taką że: $\mathbf{HH}' = \Sigma$

K03: Wygenerować *m*-wymiarowy wektor losowy: $\mathbf{Z} = \left[Z_j\right]_m$ o rozkładzie normalnym $N(0,1)$

K04: Obliczyć: $\mathbf{X} = \mathbf{HZ} + \mu$

K05: Powtarzać kroki K03-K04 *n* razy.

Ostatnie rozpatrywane rozwiązanie dotyczy *generowania danych statystycznych z generatora procesów stochastycznych.* W modelowaniu zjawisk ekonomicznych dość powszechnie występuje parametr interpretowany jako czas, okres lub moment, który w sposób niezdeterminowany wpływa na stan i rozwój rozpatrywanego zjawiska. Modele matematyczne takich zjawisk określa się mianem procesów stochastycznych (losowych). Istnieją analityczne metody generowania takich procesów, polegające na tym, że losuje się albo kolejne wartości funkcji czasu, albo też kolejne wartości stanu procesu (zjawiska) w określonych momentach. Mamy wówczas do czynienia odpowiednio z *procesami sygnałowymi* (np. proces Poissona) oraz z *procesami błądzenia przypadkowego* (np. proces Wienera) [23].

W procesie sygnałowym historia stanów danego układu (systemu) związana jest ściśle z kolejnymi momentami $t_0, t_1, t_2, \ldots, t_n$. Stan procesu ulega zmianie o wartość Δ (najczęściej stałą) na skutek pojawienia się kolejnego impulsu (sygnału). Proces sygnałowy jest więc wyznaczony przez dwie zmienne losowe:

1) czas $\xi_i = t_i - t_{i-1}$ upływający pomiędzy kolejnymi stanami procesu,

2) przyrost wartości stanu procesu $\Delta_i = X(t_i) - X(t_{i-1})$.

Generowanie trajektorii procesu sygnałowego może być realizowane zgodnie z algorytmem 2.5.

Algorytm 2.5

Generowanie trajektorii procesu sygnałowego

K01: Wygenerować liczbę losową ξ_i zgodnie z rozkładem wykładniczym,
K02: Wygenerować liczbę losową Δ_i zgodnie z zadanym rozkładem,
K03: Obliczyć $t_i = t_{i-1} + \xi_i$ oraz $x + \Delta_i$ i przejść do punktu K01.

Zasadniczą różnicą pomiędzy procesem Poissona a procesem Wienera jest to, iż trajektorie tego ostatniego są ciągłe z prawdopodobieństwem równym 1. Stan procesu *X(t)* w momentach *t* może więc przyjmować wartości ze zbioru liczb rzeczywistych. Ponieważ nie jest możliwe wygenerowanie wszystkich stanów procesu we wszystkich kolejnych momentach,

losuje się jedynie wartości stanów w momentach wybranych. Zadanie sprowadza się do generowania liczb losowych o rozkładzie normalnym w kolejnych zadanych momentach $t_0 < t_1 < t_2 < ... < t_n$ i dodawaniu ich do wartości stanu poprzedniego $X(t_{i-1})$. Można zatem sformułować procedurę postępowania, zilustrowaną algorytmem 2.6 (por. [23]).

Algorytm 2.6

Generowanie trajektorii procesu błądzenia przypadkowego

K01: Przyjąć w momencie t_0 stan procesu $X(t_0) = 0$,
K02: Wylosować liczbę losową U_i zgodnie z rozkładem normalnym,
K03: Obliczyć $X(t_i) = X(t_{i-1}) + U_i$ i przejść do punktu K01.

2.4. Analiza danych statystycznych

Wiarygodność modelu symulacyjnego oceniana jest zarówno w fazie jego tworzenia (np. ocena jakości użytych generatorów liczb losowych), jak i w trakcie prowadzonych przy jego pomocy badań. Weryfikacji poddawane są wyniki otrzymywane podczas eksperymentów prowadzonych na przygotowanym modelu symulacyjnym. Poprawność tych rezultatów jest szacowana na podstawie obserwacji przebiegu procesu rzeczywistego i danych historycznych. Do najczęściej wykorzystywanych narzędzi oceny jakości modelu symulacyjnego należą testy statystyczne, parametry rozkładu zmiennych losowych, charakterystyki opisowe populacji statystycznej. W przypadku modeli stochastycznych otrzymane wartości parametrów opisowych są oczywiście estymatorami szacowanych wielkości rzeczywistych. W tablicach 2.5-2.8

zestawiono najczęściej stosowane opisowe parametry statystyczne, które będą wykorzystane na etapie generowania i przetwarzania zbiorów danych liczbowych.

Zbiory danych przetwarzane metodami WAP mogą być liczbowymi realizacjami wielowymiarowych zmiennych losowych (w przypadku podejścia stochastycznego) lub też wielowymiarowymi obserwacjami opisującymi w sposób wyczerpujący analizowane zjawisko złożone (w przypadku podejścia opisowego). Podejście stochastyczne umożliwia wnioskowanie o populacji na podstawie pobranej próby losowej, zaś w podejściu opisowym wnioskowanie dotyczyć może skończonego zbioru danych (por. [90], [92]). W obu przypadkach zbiory wielowymiarowych obserwacji opisywane są w sposób syntetyczny za pomocą parametrów (charakterystyk) położenia, rozproszenia, asymetrii i koncentracji oraz współzależności.

Do najważniejszych parametrów położenia należą:

- wartość oczekiwana,
- mediana.

Tablica 2.5

Parametry położenia

Parametr	Struktura danych	
	dwuwymiarowa $\mathbf{X}=\left[x_{ij}\right]_{n\times m}$	trójwymiarowa $\mathbf{X}=\left[x_{ijt}\right]_{n\times m\times v}$
wartość oczekiwana	$\overline{\mathbf{x}}=\left[\overline{x}_j\right]=\frac{1}{n}\sum_{i=1}^{n}x_{ij}$	$\overline{\mathbf{x}}=\left[\overline{x}_j\right]=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}x_{ijt}$
mediana	$\mathbf{a}=\left[a_j\right]=\begin{cases}\frac{x_{m+1}}{2}\\ \frac{\frac{x_{m+1}}{2}+\frac{x_{m+2}}{2}}{2}\end{cases}$	$\mathbf{a}=\left[a_j\right]=\begin{cases}\frac{x_{nv+1}}{2}\\ \frac{\frac{x_{nv+1}}{2}+\frac{x_{nv+2}}{2}}{2}\end{cases}$

gdzie: $\overline{\mathbf{x}}$ - wektor średnich; $\mathbf{a}$ - wektor median.
Źródło: opracowanie własne na podstawie: [107], [183].

Do najważniejszych parametrów rozproszenia zalicza się:

- macierz kowariancji,
- odchylenie standardowe,
- współczynnik zmienności,

- odchylenie przeciętne.

Spośród parametrów asymetrii i koncentracji wymienić należy przede wszystkim:

- współczynnik koncentracji,
- współczynnik asymetrii.

Jako miarę współzależności stosuje się najczęściej macierz korelacji (w prowadzonych badaniach obliczano ponadto wyznacznik macierzy korelacji).

Tablica 2.6

Parametry rozproszenia

Parametr	Struktura danych	
	dwuwymiarowa $\mathbf{X}=[x_{ij}]_{n\times m}$	trójwymiarowa $\mathbf{X}=[x_{ijt}]_{n\times m\times v}$
macierz kowariancji	$\Sigma=[\sigma_{jl}]=\begin{cases}\sigma_{jl} & \text{dla } j\neq l\\ \sigma_j^2 & \text{dla } j=l\end{cases}$ gdzie: $\sigma_{jl}=\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)(x_{il}-\bar{x}_l)$ $\sigma_j^2=\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2$ $\bar{x}_j=\frac{1}{n}\sum_{i=1}^{n}x_{ij}$ $\bar{x}_l=\frac{1}{n}\sum_{i=1}^{n}x_{il}$	$\Sigma=[\sigma_{jl}]=\begin{cases}\sigma_{jl} & \text{dla } j\neq l\\ \sigma_j^2 & \text{dla } j=l\end{cases}$ gdzie: $\sigma_{jl}=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}(x_{ijt}-\bar{x}_j)(x_{ilt}-\bar{x}_l)$ $\sigma_j^2=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}(x_{ijt}-\bar{x}_j)^2$ $\bar{x}_j=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}x_{ijt}$ $\bar{x}_l=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}x_{ilt}$
odchylenie standardowe	$\mathbf{S}=[\sigma_j]=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}$	$\mathbf{S}=[\sigma_j]=\sqrt{\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}(x_{ijt}-\bar{x}_j)^2}$
współczynnik zmienności	$\mathbf{V}=[v_j]=\frac{\sigma_j}{\bar{x}_j}$ gdzie: $\sigma_j=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}$	$\mathbf{V}=[v_j]=\frac{\sigma_j}{\bar{x}_j}$ gdzie: $\sigma_j=\sqrt{\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}(x_{ijt}-\bar{x}_j)^2}$
odchylenie przeciętne	$\mathbf{D}=[d_j]=\frac{1}{n}\sum_{i=1}^{n}\lvert x_{ij}-\bar{x}_j\rvert$	$\mathbf{D}=[d_j]=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\lvert x_{ijt}-\bar{x}_j\rvert$

Źródło: opracowanie własne na podstawie: [11], [107], [133], [183].

Wymienione wyżej statystyczne charakterystyki opisowe wykorzystano do oceny właściwości statystycznych generowanych zbiorów danych liczbowych. Generowane zbiory danych różnicowane były w szczególności ze względu na wektory średnich oraz macierze wariancji-kowariancji.

Tablica 2.7

Parametry asymetrii i koncentracji

Parametr	Struktura danych	
	dwuwymiarowe $\mathbf{X}=\left[x_{ij}\right]_{n\times m}$	trójwymiarowe $\mathbf{X}=\left[x_{ijt}\right]_{n\times m\times v}$
współczynnik asymetrii $-1\le a_j\le 1$	$\mathbf{A}=\left[a_j\right]=\dfrac{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij}-\bar{x}_j\right)^3}{\left\lvert\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij}-\bar{x}_j\right)^3\right\rvert+\sigma^3}$	$\mathbf{A}=\left[a_j\right]=\dfrac{\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\left(x_{ijt}-\bar{x}_j\right)^3}{\left\lvert\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\left(x_{ijt}-\bar{x}_j\right)^3\right\rvert+\sigma^3}$
współczynnik koncentracji $0\le b_j\le 1$	$\mathbf{B}=\left[b_j\right]=\dfrac{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij}-\bar{x}_j\right)^4-\sigma_j^4}{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij}-\bar{x}_j\right)^4}$	$\mathbf{B}=\left[b_j\right]=\dfrac{\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\left(x_{ijt}-\bar{x}_j\right)^4-\sigma_j^4}{\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\left(x_{ijt}-\bar{x}_j\right)^4}$

Źródło: opracowanie własne na podstawie: [107], [183].

Tablica 2.8

Parametry współzależności

Parametr	Struktura danych	
	dwuwymiarowa $\mathbf{X}=\left[x_{ij}\right]_{n\times m}$	trójwymiarowa $\mathbf{X}=\left[x_{ijt}\right]_{n\times m\times v}$
macierz korelacji	$\mathbf{R}=\left[r_{jl}\right]=\begin{cases}\dfrac{\sigma_{jl}}{\sqrt{\sigma_j^2\sigma_l^2}} & \text{dla } j\ne l\\ 1 & \text{dla } j=l\end{cases}$ gdzie: $\sigma_{jl}=\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij}-\bar{x}_j\right)\left(x_{il}-\bar{x}_l\right)$ $\sigma_j^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij}-\bar{x}_j\right)^2$ $\bar{x}_j=\frac{1}{n}\sum_{i=1}^{n}x_{ij}$ $\bar{x}_l=\frac{1}{n}\sum_{i=1}^{n}x_{il}$	$\mathbf{R}=\left[r_{jl}\right]=\begin{cases}\dfrac{\sigma_{jl}}{\sqrt{\sigma_j^2\sigma_l^2}} & \text{dla } j\ne l\\ 1 & \text{dla } j=l\end{cases}$ gdzie: $\sigma_{jl}=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\left(x_{ijt}-\bar{x}_j\right)\left(x_{ilt}-\bar{x}_l\right)$ $\sigma_j^2=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}\left(x_{ijt}-\bar{x}_j\right)^2$ $\bar{x}_j=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}x_{ijt}$ $\bar{x}_l=\frac{1}{nv}\sum_{i=1}^{n}\sum_{t=1}^{v}x_{ilt}$
wyznacznik macierzy korelacji[9]	$\lvert\mathbf{R}\rvert$	$\lvert\mathbf{R}\rvert$

Źródło: opracowanie własne na podstawie: [107], [183].

[9] Wyznacznik macierzy korelacji obliczano za pomocą funkcji *det*, której algorytm oparto na rozkładzie trójkątnym macierzy symetrycznej, wyznaczanym metodą Gaussa-Banachiewicza opisaną np. w pracy [99]. Wyznacznik macierzy korelacji otrzymać można również w oparciu o wektor wartości własnych, obliczany dla potrzeb analizy czynnikowej (iloczyn wartości własnych macierzy **R** jest równy wyznacznikowi tej macierzy).

Rozdział 3

MODELOWANIE SYMULACYJNE METOD WIELOWYMIAROWEJ ANALIZY PORÓWNAWCZEJ

3.1. Metodologia modelowania symulacyjnego

Do kluczowych pojęć z zakresu metod symulacyjnych należą: „system", „model" oraz „modelowanie symulacyjne" i są one ściśle związane z teorią badań systemowych. Naturalnym uzasadnieniem tego faktu jest nieustanne dążenie człowieka do poznawania otaczającej go rzeczywistości reprezentowanej w badaniach naukowych w postaci różnorodnych systemów i modeli. System i model są pojęciami powszechnie stosowanymi w wielu dziedzinach wiedzy, a także bardzo często w znaczeniu potocznym. W różnych kontekstach podawane są rozmaite znaczenia tych terminów i, jak dotychczas, nie ma ścisłych, ogólnie przyjętych i akceptowanych ich definicji. Poniżej przytoczono, z konieczności tylko niektóre, interpretacje tych terminów, w celu zilustrowania stanowisk zajmowanych przez różnych autorów z zakresu teorii badań systemowych, cybernetyki oraz teorii modelowania i symulacji.

Von Bertalanffy napisał, że „system można zdefiniować jako zbiór elementów wzajemnie powiązanych ze sobą i z otoczeniem" [128, s. 37].

Beneš w monografii [20] wprowadził pojęcie „kompleksu", definiując je jako „system, który tworzy wielka liczba elementów (...). Elementami kompleksu mogą być np. cząsteczki, podzespoły czy też całe urządzenia techniczne, mogą być nimi jednak również organizmy

żywe, ludzie czy też grupy ludzi" [20, s. 9]. W definicji kompleksu istotna jest jego skala zarówno w sensie złożoności (jest to system złożony z dużej liczby elementów), jak również w sensie rozproszenia (jest to system rozległy w znaczeniu przestrzennym).

Flakiewicz i Oleński przyjmują, że pojęcie „system" oznacza „(...) określoną całość złożoną z części o określonych właściwościach i z relacji między tymi częściami i (lub) między ich właściwościami" [54, s. 18-19].

Na gruncie cybernetyki system określa się często jako „układ" stanowiący „pewną całość składającą się z połączonych części" [59, s. 17]. Jak podkreśla Gościński, w cybernetyce nie rozróżnia się układów i systemów, uznając obydwa terminy za tożsame.

Gordon na użytek badań symulacyjnych definiuje system jako „zbiór obiektów powiązanych określonymi wzajemnymi zależnościami lub oddziaływaniami" [58, s. 17]. Zaznacza przy tym, że definicja ta dotyczy systemów statycznych. W przypadku systemów dynamicznych należy uwzględnić jeszcze zmiany w czasie, zachodzące pod wpływem interakcji między elementami systemu, jak również na skutek oddziaływania otoczenia.

Istota zależności między elementami systemu jest w sposób szczególny podkreślona w definicji Mynarskiego, który pisze, iż „układy są mniej lub bardziej zwartymi całościami, różniącymi się od swego otoczenia stopniem powiązań ich elementów" [122, s. 7].

Hall stwierdza, iż „system jest to zbiór obiektów wraz z relacjami istniejącymi pomiędzy tymi obiektami oraz pomiędzy ich własnościami" [81, s. 93]. Obiektami w systemie są elementy, między którymi zachodzą określone relacje (interakcje). Obiekty te opisane są pewnymi charakterystykami (atrybutami), które przyjmują w danych momentach określone wartości. Zakłada się, że w systemie występują tylko takie obiekty, między którymi istnieją wzajemne oddziaływania tworzące zbiór relacji wiążących poszczególne elementy.

Pojęcie modelu jest również precyzowane i definiowane w aspekcie konkretnych badań. Barton stwierdza, że model jest „(...) konkretnym, interpretacyjnym wyrazem albo jednej lub

kilku hipotez (...)", przy czym „każda teoria może prowadzić do niemal nieskończonej liczby modeli i zastosowań" [12, s. 35].

W cytowanej już pracy Gordona czytamy, iż „model definiuje się jako zbiór informacji o systemie, zebranych w celu jego zbadania" [58, s. 22].

Sieniawski przez model rozumie „abstrakcyjne lub materialne odwzorowanie danego obiektu (systemu) lub jego fragmentu, skonstruowane tak, że badanie go dostarcza informacji o tym obiekcie. Podobieństwo modelu do obiektu może być np. przestrzenne, fizyczne i/lub matematyczne [149, s. 143].

Według Lewandowskiego natomiast „model - to zbiór składowych - abstrakcyjnych wyobrażeń składowych systemu i specyfikacja oddziaływań między nimi" [5, s. 402].

Z kolei w pracy [102] czytamy, że „model systemu jest ilościową i jakościową reprezentacją systemu i jego zachowania się, które pokazuje wpływ czynników istotnych, ze względu na zamierzony cel badań, które mają być przeprowadzone na modelach" [102, s. 7].

Terminy takie jak „badania symulacyjne", „modelowanie symulacyjne", „symulacja cyfrowa", „symulacja komputerowa", „modelowanie cyfrowe", „modelowanie statystyczne", „metoda Monte Carlo" i podobne, odnoszone są z reguły do pewnych ściśle zdefiniowanych systemów hipotetycznych, będących modelami systemów rzeczywistych. Metody symulacyjne są wykorzystywane do przeprowadzania eksperymentów na tych modelach w celu poznania ich właściwości, sposobów reakcji na różnorodne bodźce zewnętrzne, zasad funkcjonowania itp. W obszernej bibliografii dotyczącej zagadnień modelowania i symulacji znaleźć można również różnorodne definicje i interpretacje tych pojęć.

Czerwiński, zwracając uwagę na wieloznaczność terminu „symulacja", definiuje ją jako „wykonywanie pewnych zabiegów mających dać odpowiedź na pytanie, *jak zachowałby się w pewnych okolicznościach obiekt*[1] zobrazowany danym modelem". Autor czyni przy tym założenie, że model „wystarczająco dobrze" obrazuje zachowanie obiektu [39, s. 183].

---

[1] Wyróżnienie kursywą za cytowaną pracą [39].

Koleśnik, Huzar i Fryźlewicz stwierdzają, iż „metoda symulacji cyfrowej jest techniką wykonywania badań eksperymentalnych na modelu rzeczywistego systemu, realizowanych w określonym celu z wykorzystaniem komputerów" [102, s. 4].

Naylor określa symulację cyfrową jako „technikę numeryczną służącą do dokonywania eksperymentów na pewnych rodzajach modeli matematycznych, które opisują przy pomocy maszyny cyfrowej zachowanie się złożonego systemu w ciągu długiego okresu czasu" [123, s. 21].

Evans, Wallace i Sutherland definiują symulację jako „(...) zastosowanie modelu w celu chronologicznego wygenerowania historii stanów tego modelu, która jest uważana za historię stanów modelowanego systemu" [52, s. 19]. W tejże pracy czytamy dalej, że „badania symulacyjne - to jedno lub więcej zastosowań symulacji w badaniach systemu lub zbioru systemów" [52, s. 19].

Z kolei Tyszer pisze, iż „symulacja cyfrowa jest algorytmiczną metodą prowadzenia (za pomocą maszyny cyfrowej) eksperymentów na modelach dynamicznych istniejących lub projektowanych systemów" [169, s. 14].

Następna z przytoczonych definicji pochodzi od Kondratowicza, który terminem „modelowanie i symulacja" określa „czynności wyróżniania i formalizacji cech systemu oraz ustalania relacji zachodzących między nimi w czasie, w celu znalezienia ich odpowiedników w innych systemach" [106, s. 16].

Syntetyczną definicję podał Zeigler, pisząc iż „modelowanie i symulacja" oznacza „zestaw działań związanych z konstruowaniem modeli systemów rzeczywistych i symulowania ich na komputerze" [186, s. 23]. W dalszym ciągu Zeigler wyróżnia w swojej definicji trzy podstawowe elementy - system rzeczywisty, model oraz komputer i precyzuje, iż modelowanie dotyczy zależności pomiędzy systemami rzeczywistymi i ich modelami, natomiast symulacja odnosi się przede wszystkim do relacji zachodzących między komputerami a modelami.

W świetle przytoczonych wyżej definicji pojęć dotyczących cybernetyki, teorii systemów oraz teorii modelowania i symulacji można przyjąć, że modelowanie symulacyjne jest metodą analizy systemów rzeczywistych za pomocą przetwarzanych komputerowo modeli symulacyjnych. Należy w tym miejscu zaznaczyć, że systemy odwzorowywane z pomocą modelu mogą faktycznie istnieć, albo też mogą być systemami sztucznymi lub projektowanymi. Model systemu rzeczywistego to z reguły opis formalny (wyrażone w postaci układów równań matematycznych, za pomocą instrukcji języka programowania lub przedstawione z wykorzystaniem innego sformalizowanego alfabetu) właściwości oraz wzajemnych relacji zachodzących pomiędzy elementami tego systemu. Badanie systemu rzeczywistego za pomocą modelu symulacyjnego wymaga wyodrębnienia zbioru tych informacji, które w stopniu decydującym wpływają na jego funkcjonowanie. Obserwowalne organizmy (układy) ekonomiczne są zazwyczaj bardzo złożone, co sprawia, że nie jest możliwe (ani też celowe i konieczne) budowanie symulacyjnego modelu komputerowego uwzględniającego wszystkie występujące w rzeczywistości obiekty, charakterystyki i relacje. Złożoność układu rzeczywistego przejawia się przede wszystkim w dużej liczbie atrybutów (cech) związanych z każdym jego elementem oraz relacji, które mogą zachodzić pomiędzy tymi atrybutami we wszystkich możliwych kombinacjach. W praktyce za wystarczająco wiarygodny model symulacyjny uznaje się taki, który uwzględnia wszystkie istotne charakterystyki systemu rzeczywistego oraz algorytmy zmienności ich realizacji w czasie. W badaniach empirycznych zakłada się ponadto, że analizowany rzeczywisty układ ekonomiczny jest systemem dynamicznym i posiada zarejestrowaną lub możliwą do odtworzenia historię stanów, która obrazuje jego zachowanie w przeszłości. Na podstawie tych danych można prognozować trajektorię stanów systemu w przyszłości oraz weryfikować poprawność i wiarygodność modelu symulacyjnego.

Formalizując zagadnienie można przyjąć, że system tworzy uporządkowana para postaci (por. [3], [128], [150]):

$$S = (X, R) \tag{3.1}$$

gdzie:

$$X = (X_1, X_2, \ldots, X_n) \tag{3.2}$$

stanowi zbiór n elementów (obiektów) systemu, oraz

$$R \subseteq X_1 \times X_2 \times \ldots \times X_n \tag{3.3}$$

jest zbiorem uporządkowanych relacji, wiążących te elementy.

Każdy element systemu opisany jest wektorem atrybutów:

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{im_i}) \tag{3.4}$$

gdzie: x_{ij} - j-ty atrybut opisujący i-ty element systemu S,
m_i - liczba atrybutów związanych z i-tym obiektem,
$i = 1,2,\ldots,n$,
$j = 1,2,\ldots,m_i$.

Historię stanów i-tego elementu systemu, czyli wartości poszczególnych atrybutów zaobserwowane w kolejnych momentach czasu t przedstawia macierz:

$$\mathbf{X}^i = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m_i} \\ x_{21} & x_{tj} & \ldots & x_{2m_i} \\ \ldots & \ldots & \ldots & \ldots \\ x_{v1} & x_{v2} & \ldots & x_{vm_i} \end{bmatrix} \tag{3.5}$$

gdzie: $t = 1,2,\ldots,v$,
v - liczba stanów (momentów czasu).

Elementy tak przedstawionego systemu oraz zachodzące pomiędzy nimi interakcje są reprezentowane w modelu symulacyjnym za pomocą układów równań (nierówności) algebraicznych (najczęściej nieliniowych) o dużej z reguły liczbie niewiadomych.

W sposób ogólny model systemu statycznego przedstawia się w postaci równania wyjścia (por. np. [5], [132]):

$$y = f(u) \tag{3.6}$$

gdzie: y - wyjście systemu,

u - wejście systemu,
f - funkcja zależności.

Model systemu dynamicznego, uwzględniającego czynnik czasu, przedstawia się w postaci równania stanu:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), u(t)) \tag{3.7}$$

oraz równania wyjścia:

$$y(t) = g(x(t), u(t)) \tag{3.8}$$

gdzie: x - stan systemu,
g - funkcja określająca wyjście systemu,
f - funkcja określająca stan systemu,
y, u - jw.

Ściśle zdefiniowane zależności funkcyjne umożliwiają rejestrację historii stanów systemu w kolejnych fazach modelowania symulacyjnego. Ze względu na losową naturę modelowanych zdarzeń i zachowań użytecznym narzędziem badań mogą być przede wszystkim symulacyjne modele probabilistyczne o charakterze dynamicznym. Na etapie konstruowania modeli tego rodzaju należy opracować trzy podstawowe algorytmy (por. np. [53], [102], [169], [178]):

1. Algorytm aktualizacji czasu systemowego (symulacyjnego). Wykorzystuje się w tym celu jedną z dwóch technik:

a) technikę stałego kroku,

b) technikę kolejnych zdarzeń.

2. Algorytm zmiany stanu systemu (sterowania przebiegiem symulacji). Wykorzystuje się tutaj jedną z trzech następujących metod:

a) metodę planowania zdarzeń,

b) metodę przeglądania działań,

c) metodę interakcji procesów.

3. Algorytm generowania zmiennych losowych. W tym celu korzysta się, zależnie od przeznaczenia modelu, z następujących metod[2]:

a) metody generowania zmiennych losowych o rozkładzie równomiernym,

b) metody generowania zmiennych losowych o rozkładzie dowolnym.

Proces przygotowania i realizacji badań symulacyjnych jest przedsięwzięciem złożonym, co stwarza konieczność dzielenia go na pewne logicznie uszeregowane etapy. W literaturze przedmiotu spotkać można różne, często bardzo odmienne propozycje w tym zakresie (por. [123], [169], [186]). Wydaje się, iż to zróżnicowanie stanowisk co do struktury i przebiegu procedury symulacyjnej jest uzasadnione zarówno brakiem formalnych reguł tworzenia modeli symulacyjnych (o czym była już mowa wyżej), jak i metodologicznymi trudnościami w liniowym, sekwencyjnym uporządkowaniu poszczególnych etapów procesu modelowania cyfrowego.

Ogólna procedura modelowania symulacyjnego powinna składać się z następujących kluczowych etapów[3]:

a) sformułowanie problemu,

b) sformułowanie modelu (formalizacja problemu),

c) przygotowanie danych (generowanie zmiennych losowych),

d) opracowanie programu komputerowego (faza projektowania i kodowania),

e) sprawdzenie poprawności modelu,

f) przeprowadzenie eksperymentów symulacyjnych,

g) interpretacja uzyskanych wyników.

---

[2] Metody generowania liczb losowych o różnych rozkładach prawdopodobieństwa oraz zagadnienia poprawności statystycznej tych metod, a także niektóre algorytmy komputerowe generowania zmiennych losowych dla potrzeb eksperymentów symulacyjnych przedstawione są m.in. w pracach: [31], [40], [43], [53], [58], [94], [95], [102], [115], [123], [132], [169], [187], [188]. Przegląd tych metod dla potrzeb niniejszej pracy zamieszczono w rozdziale 2.

[3] Przegląd różnych propozycji w tym zakresie znajduje się w pracy [102].

Na rzecz podkreślenia podstawowych zalet metod symulacyjnych przytacza się w literaturze przedmiotu najczęściej następujące argumenty: (por. m.in. [53, s. 29-33], [102, s. 4-6], [123, s. 31-32], [169, s. 14-16], [149, s. 145]):

1) symulację komputerową można stosować w przypadkach, kiedy nie jest możliwe ze względów metodologicznych, technicznych lub ekonomicznych zastosowanie innych metod i technik badawczych,

2) metody symulacyjne umożliwiają rozpatrywanie i analizowanie wielu alternatywnych rozwiązań, wariantów i hipotez, co w istotny sposób ułatwia podjęcie optymalnej decyzji,

3) symulacja komputerowa może być podstawą analizy określonego systemu w sytuacjach, o których brak jest wyczerpujących informacji rzeczywistych, a które mogą być wygenerowane przez model symulacyjny,

4) metody symulacyjne umożliwiają analizowanie systemu w czasie nierzeczywistym, tzn. w warunkach komprymacji lub wydłużenia rzeczywistego czasu „życia" systemu,

5) eksperyment symulacyjny można wielokrotnie powtarzać lub odtwarzać w tych samych warunkach, co z reguły nie jest możliwe w przypadku innego rodzaju doświadczeń (np. laboratoryjnych),

6) metody symulacyjne dają możliwość kontroli wszystkich warunków i parametrów eksperymentu oraz śledzenia pośrednich wyników całego procesu symulacyjnego.

Podobnie jak i każda inna metoda badawcza, symulacja komputerowa nie jest pozbawiona pewnych wad i niedogodności. Do najczęściej stawianych w literaturze przedmiotu zastrzeżeń pod adresem modelowania symulacyjnego należą:

1) brak precyzyjnych, formalnych reguł konstruowania modeli symulacyjnych, co wymaga od badacza rekompensaty w postaci dużego doświadczenia i intuicji,

2) brak ścisłych i dokładnych metod szacowania wiarygodności modelu symulacyjnego i jego adekwatności w stosunku do systemu rzeczywistego,

3) duży nakład pracy i czasu potrzebny na opracowanie eksperymentu, przygotowanie i uruchomienie komputerowego programu symulacyjnego oraz na przeprowadzenie doświadczeń i właściwą ocenę wyników.

Pomimo tych niedoskonałości i braków metody symulacji komputerowej rozwijają się niezwykle dynamicznie, począwszy od schyłku lat pięćdziesiątych, tzn. od momentu wprowadzenia do powszechnego użytkowania techniki komputerowej. Znajdują zastosowanie we wszystkich niemal dziedzinach badań naukowych i są w praktyce wykorzystywane w najróżniejszych sferach życia. Można tu przytoczyć na przykład liczne zastosowania wojskowe, astronautyczne i edukacyjne, czy też wykorzystywanie metod i technik symulacyjnych w modelowaniu i analizie systemów ekonomicznych, biologicznych, socjologicznych, demograficznych meteorologicznych, telekomunikacyjnych, ekologicznych, transportowych itp. Na gruncie ekonomii modelowanie symulacyjne znajduje zastosowanie w tych rozmaitych, mniej lub bardziej szczegółowych dziedzinach, w których kluczową rolę odgrywa proces podejmowania decyzji, zwłaszcza w warunkach niepewności, ryzyka, niepełnej informacji czy wielowariantowości rozwiązań. Z tych też względów metody symulacyjne są bardzo użytecznym i często wykorzystywanym narzędziem do tworzenia modeli symulacyjnych firm, rynku, zarządzania, marketingu, produkcji, zapasów, kolejek itp., a także dla potrzeb prognozowania. Generalnie można stwierdzić, że modelowanie symulacyjne może być użyteczne w różnych sferach aktywności ekonomicznej przedsiębiorstw.[4]

[4] Przykłady zastosowań metod symulacyjnych zawierają m.in. prace [109], [117], [123], [131], [177], [178].

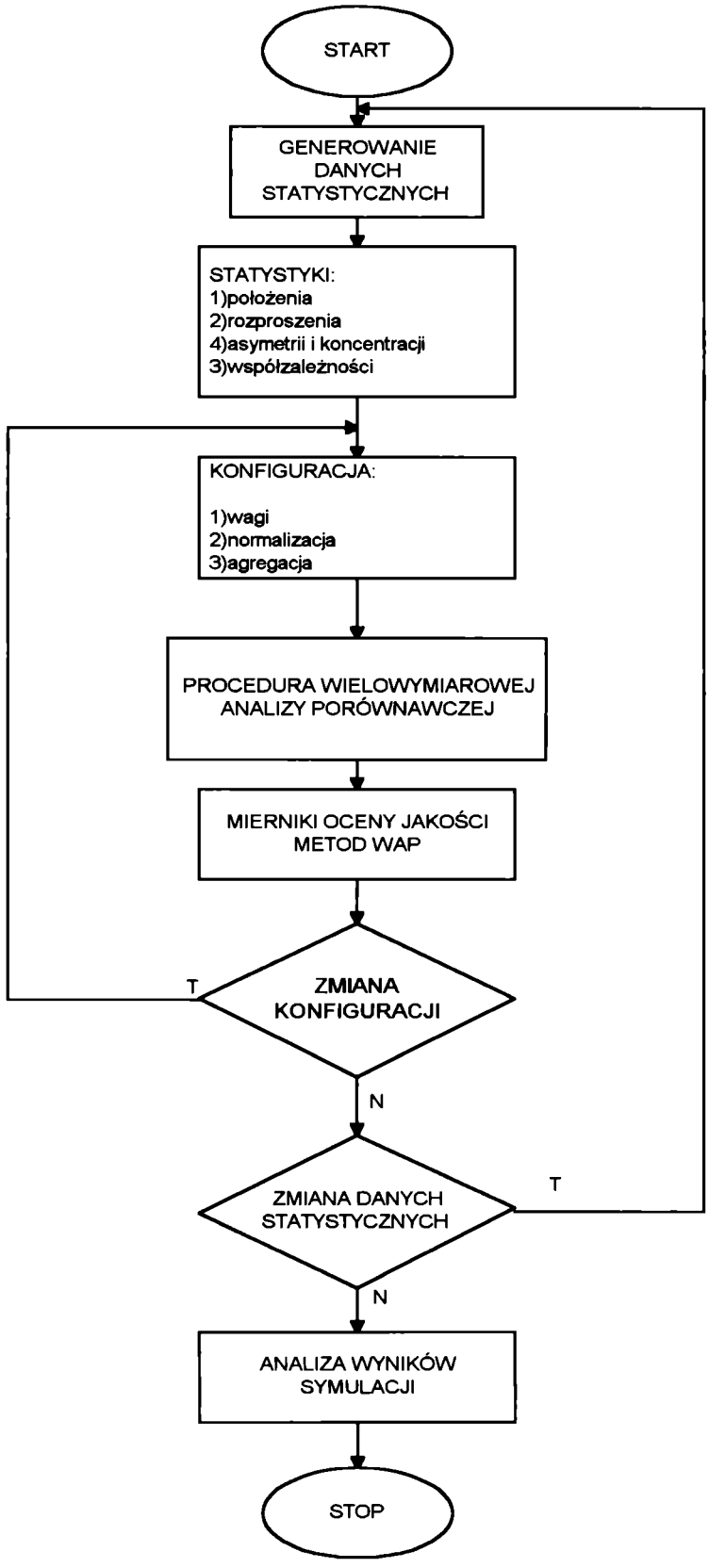
3.2. Układ eksperymentu symulacyjnego

Podstawowym celem niniejszej pracy jest zbadanie odporności poszczególnych konfiguracji metod wielowymiarowej analizy porównawczej na charakter oraz strukturę przetwarzanych zbiorów danych liczbowych. Zgodnie z założeniami do analizy przyjęto 2 formuły ważenia zmiennych, 9 formuł normalizacji oraz 17 formuł agregacji cech. W rezultacie otrzymano 306 ścieżek przetwarzania danych, których jakość i statystycznie rozumianą przydatność poddano ocenie metodą modelowania symulacyjnego. W tym celu generowano zbiory danych zróżnicowane pod względem liczby cech, liczby obiektów, liczby okresów oraz o różnych wektorach wartości oczekiwanych i macierzach wariancji-kowariancji. Do generowania tych zbiorów wykorzystano generatory rozkładu równomiernego i normalnego oraz generator wektorów losowych o wielowymiarowym rozkładzie normalnym, które scharakteryzowane zostały w rozdziale 2 (punkty 2.1 i 2.3 oraz algorytm 2.4). Otrzymane zbiory danych przetwarzane były zgodnie z konfiguracją każdej z 306 analizowanych ścieżek wielowymiarowej analizy porównawczej, będących wynikiem złożenia poszczególnych elementów procedury WAP. Wyniki tego procesu poddano ocenie i rangowaniu. Ogólny układ eksperymentu symulacyjnego ilustruje rysunek 3.1, natomiast szczegółowy schemat postępowania przedstawia algorytm 3.1.

Algorytm 3.1

Symulacyjna ocena jakości metod WAP

K01: Dany jest plik: PART.DBF zawierający parametry generatora wektorów losowych o wielowymiarowym rozkładzie normalnym
K02: Wygenerować według algorytmu 2.4 zadaną liczbę zbiorów danych, obliczając ich charakterystyki statystyczne: położenia, rozproszenia, asymetrii, koncentracji, współzależności
K03: Wczytać z pliku TD*.DBF wygenerowany zbiór danych
K04: Wczytać z pliku PATHS.DBF ścieżkę WAP
K05: Wyznaczyć zmienną syntetyczną zgodnie z daną ścieżką
K06: Obliczyć mierniki jakości zmiennej syntetycznej
K07: Jeśli nie jest to ostatnia ścieżka, to przejść do punktu K04
K08: Jeśli nie jest to ostatni zbiór danych, to przejść do punktu K03
K09: Ustalić kolejność ścieżek WAP na podstawie zagregowanych wartości mierników jakości



Rysunek 3.1. Schemat blokowy eksperymentu symulacyjnego

3.3. Reguły generowania zbiorów danych

Do generowania zbiorów danych liczbowych zastosowano następujące generatory, będące funkcjami składowymi klasy *nrand*[5]:

- *rget* - produkuje liczbę losową o rozkładzie równomiernym na podstawie *r* metodą kongruencji liniowej,
- *rm_random* - generator o rozkładzie równomiernym[6], oparty na ciągu Fibonacciego,
- *dlots* - generuje liczbę losową typu *double* z przedziału [0, 1] przy wykorzystaniu generatora *rget* o zakresie [0, LONG_MAX], przy czym wartość stałej LONG_MAX zależy od platformy sprzętowej stosowanej do prowadzonych obliczeń,
- *dablots* - generuje liczbę losową typu *double* z przedziału [a, b],
- *dxrnd* - generuje liczby pseudolosowe typu double o rozkładzie normalnym dla danej wartości oczekiwanej *ex* i danego odchylenia standardowego *stdx*,
- *ex_rnd* - generuje wektor średnich,
- *vc_rnd* - oblicza macierz wariancji-kowariancji,
- *mvnd* - generuje wektory losowe o wielowymiarowym rozkładzie normalnym przy zadanym wektorze wartości oczekiwanych i danej macierzy wariancji-kowariancji[7],

W prowadzonych badaniach przyjęto założenie, że generowane dane liczbowe do testowania poszczególnych ścieżek WAP różnić się będą następującymi własnościami:

1) charakterem struktur danych, tzn. przetwarzane będą struktury dwuwymiarowe o postaci $\mathbf{X} = [x_{ij}]_{n \times m}$ oraz struktury trójwymiarowe o postaci $\mathbf{X} = [x_{ijt}]_{n \times m \times v}$,

---

[5] Większość generatorów klasy *nrand* przygotowano na podstawie prac: [40], [46], [53], [82], [116], [123], [143], [144], [187], [188].

[6] Generator opracowany przez Marsaglię i Zamana. Autorami tłumaczenia z języka FORTRAN na język C są Lintell i Hickling z Management Consultants Ltd. Na podstawie tego tłumaczenia opracowano wersję obiektową generatora w języku C++.

[7] Generator wielowymiarowego rozkładu normalnego opracowano na podstawie programu w języku FORTRAN autorstwa Sassera z Harvard Business School, zamieszczonego w [123, s. 525-527] oraz w oparciu o pracę [43].

2) liczbą zmiennych,

3) liczbą obiektów,

4) liczbą okresów,

5) wektorem wartości oczekiwanych,

6) macierzą wariancji-kowariancji.

Wektory wartości oczekiwanych oraz macierze wariancji-kowariancji uzyskiwano w dwojaki sposób:

1) przyjmowano zadaną wartość średnią *Ex* jednakową dla wszystkich *m* zmiennych oraz konstruowano diagonalną macierz wariancji-kowariancji z jednakową wartością *Sx* na głównej przekątnej (pozostałe elementy równe 0),

2) generowano wektor średnich oraz macierz wariancji-kowariancji o wartościach z ustalonego przedziału $[a,b]$, stosując opisane wyżej generatory programowe, przy czym algorytm opracowano w taki sposób, aby otrzymana macierz była półokreślona dodatnio. Jest to warunek konieczny, aby wygenerowaną macierz $\Sigma = [\sigma_{ij}]_{m \times m}$ uznać za macierz wariancji-kowariancji. Postulat ten zrealizowano w oparciu o nierówność Schwarza postaci [133]:

$$\sigma_{ij}^2 \leq \sigma_{ii}\sigma_{jj}. \tag{3.9}$$

W tablicach 3.1-3.3 zamieszczono rozmiary generowanych zbiorów danych (dwu- i trójwymiarowych macierzy), granice przedziałów liczbowych, z których generowano wartości poszczególnych zmiennych oraz granice przedziałów liczbowych, z których losowano elementy wektorów średnich i elementy macierzy wariancji-kowariancji.

Szczegółowe parametry generowanych zbiorów zawierają tablice A.1-A.3 zamieszczone w aneksie do pracy, natomiast procedury generowania zbiorów danych w obu powyższych układach przedstawiają algorytmy 3.2 i 3.3. W trzech przebiegach dla każdego z dwóch zestawów parametrów wygenerowano po 75 zbiorów danych zgodnie z założeniami (1) i (2),

które zapisano w plikach dyskowych standardu DBF o nazwach TD*.DBF (łącznie 450 zbiorów danych).

Algorytm 3.2

Generowanie zbiorów danych - układ nr 1

K01: Dany jest plik parametrów generatora: PART.DBF
K02: Wczytać wymiary zbioru *m, n, v* oraz parametry *Ex* i *Sx*
K03: Utworzyć wektor *vEX(m)* i macierz *mVC(m, m)* przy pomocy funkcji składowych *rep_v* i *diag* klasy *mva*
K04: Wygenerować zbiór *mDAT* z generatora *mvnd* z klasy *nrand*
K05: Obliczyć charakterystyki statystyczne zbioru *mDAT* przy pomocy funkcji składowych *mean_x*, *varcov*, *stdev*, *med_x*, *var_coef*, *asym_coef*, *conc_coef* oraz *mdet* klasy *mva*
K06: Zapisać zbiór *mDAT* w pliku TD*.DAT oraz charakterystyki statystyczne w pliku PS*.DBF
K07: Jeśli nie ostatni zestaw parametrów, to przejść do punktu K02

Algorytm 3.3

Generowanie zbiorów danych - układ nr 2

K01: Dany jest plik parametrów generatora: PART.DBF
K02: Wczytać wymiary zbioru *m, n, v* oraz parametry *a* i *b*
K03: Utworzyć wektor *vEX(m)* i macierz *mVC(m, m)* przy pomocy funkcji składowych klasy *nrand* odpowiednio *ex_rnd* i *vc_rnd*
K04: Wygenerować zbiór *mDAT* z generatora *mvnd* z klasy *nrand*
K05: Obliczyć charakterystyki statystyczne zbioru *mDAT* przy pomocy funkcji składowych *mean_x*, *varcov*, *stdev*, *med_x*, *var_coef*, *asym_coef*, *conc_coef* oraz *mdet* klasy *mva*
K06: Zapisać zbiór *mDAT* w pliku TD*.DAT oraz charakterystyki statystyczne w pliku PS*.DBF
K07: Jeśli nie ostatni zestaw parametrów, to przejść do punktu K02

Tablica 3.1

Wymiary generowanych zbiorów danych

m	n	v
5	10	1
10	15	2
15	20	3
25	25	4
-	50	-

Źródło: opracowanie własne.

Tablica 3.2

Granice przedziałów generowanych wartości zmiennych

a	b
10	200
50	250
100	300
150	400
300	500
350	750
500	1000
550	1750
750	2750
-	3750

Źródło: opracowanie własne.

Tablica 3.3

Granice przedziałów generowanych wartości średnich i odchyleń standardowych

Ex	Sx
10	10
50	50
100	55
125	100
500	200
525	255
725	355
1000	500
1525	1255
2575	1755

Źródło: opracowanie własne.

3.4. Reguły przetwarzania zbiorów danych

Dla potrzeb eksperymentu symulacyjnego wygenerowano 450 zbiorów danych wejściowych zróżnicowanych pod względem wartości parametrów omówionych w punkcie 3.3. Liczbę oraz rozmiary zbiorów ustalono w taki sposób, aby zapotrzebowanie na zasoby sprzętu komputerowego (takie jak zużycie czasu procesora, rozmiar potrzebnej pamięci operacyjnej oraz wymagana przestrzeń pamięci dyskowej) nie przekraczało parametrów technicznych typowego obecnie zestawu komputerowego klasy IBM PC/80486 DX.

Każdy z wygenerowanych zbiorów został poddany procedurze przetwarzania metodami wielowymiarowej analizy porównawczej w układzie każdej z 306 ścieżek różniących się konfiguracją formuł ważenia zmiennych, formuł normalizacji oraz formuł wyznaczania wartości zmiennych syntetycznych. Zastosowano następującą procedurę postępowania:

1) określenie charakteru zmiennych (identyfikacja zmiennych o charakterze stymulant i zmiennych o własnościach destymulant),

2) ustalenie wartości współczynników wagowych poszczególnych zmiennych (postępowanie dotyczy pierwotnych realizacji zmiennych diagnostycznych),

3) normalizacja zmiennych diagnostycznych (postępowanie dotyczy pierwotnych realizacji zmiennych diagnostycznych),

4) przekształcenie zmiennych o charakterze destymulant do postaci zmiennych o własnościach stymulant (postępowanie dotyczy znormalizowanych wartości zmiennych diagnostycznych),

5) eliminacja wartości zerowych i ujemnych (postępowanie dotyczy znormalizowanych wartości zmiennych diagnostycznych),

6) wyznaczenie dolnego i górnego bieguna zbioru znormalizowanych wartości zmiennych diagnostycznych,

7) obliczenie realizacji zmiennej syntetycznej,

8) obliczenie mierników jakości zmiennej syntetycznej.

Szczegółową procedurę postępowania przedstawia algorytm 3.4.

Algorytm 3.4

Przetwarzanie zbiorów danych metodami WAP

K01: Utworzyć unikalny identyfikator sesji ID*
K02: Wczytać zbiór danych z pliku TD*.DBF
K03: Określić charakter zmiennych metodą analizy czynnikowej, korzystając z funkcji składowych *hotelling*, *loadings*, *characters* klasy *mva*
K04: Wczytać układ ścieżki WAP z pliku PATHS.DBF
K05: Wyznaczyć wagi zmiennych przy pomocy funkcji składowej *weight* z klasy *mva*
K06: Znormalizować zmienne przy pomocy funkcji składowej *normal* z klasy *mva*
K07: Przekształcić zmienne destymulanty do postaci zmiennych stymulant przy pomocy funkcji składowej *dest2stym* z klasy *mva*
K08: Wyskalować elementy macierzy danych $\mathbf{X} = \left[x_{ijt}\right]_{mxmxv}$ w taki sposób, aby spełniony był warunek $x_{ijt} > 0$, korzystając z funkcji składowej *neg2pos* z klasy *mva*
K09: 1. Jeżeli agregacja bezwzorcowa, to obliczyć wektor *vAGR* zawierający realizacje zmiennej syntetycznej, korzystając z funkcji składowej *aggreg* z klasy *mva*
2. Jeżeli agregacja wzorcowa według dolnego bieguna zbioru danych, to wyznaczyć wektor zawierający antywzorzec przy pomocy funkcji *lowerpole* i to obliczyć wektor *vAGR* zawierający realizacje zmiennej syntetycznej, korzystając z funkcji składowej *aggreg* z klasy *mva*
3. Jeżeli agregacja wzorcowa według górnego bieguna zbioru danych, to wyznaczyć wektor zawierający wzorzec przy pomocy funkcji *upperpole* i to obliczyć wektor *vAGR* zawierający realizacje zmienne syntetycznej, korzystając z funkcji składowej *aggreg* z klasy *mva*

K10: Wyznaczyć mierniki jakości zmiennej syntetycznej, korzystając z funkcji składowych *gauge01* ... *gauge12* z klasy *mva*
K11: Wyznaczyć agregatowy miernik jakości
K12: Zapisać identyfikator sesji i mierniki jakości w pliku GAUGES.DBF
K13: Jeżeli nie jest to ostatnia ścieżka, to przejść do punktu K04
K14: Jeżeli nie jest to ostatni zbiór danych, to przejść do punktu K01

3.5. Reguły oceny wyników symulacji

Uzyskane w wyniku procedury przetwarzania zbiorów danych (algorytm 3.4) mierniki jakości posłużyły za podstawowe kryterium oceny wrażliwości poszczególnych ścieżek WAP.

Dla każdego ze zbiorów otrzymano 306 wektorów zmiennych syntetycznych, dla których obliczono po 12 mierników jakości. Z uwagi na jednolity kierunek preferencji wszystkich mierników jakości zmiennych syntetycznych wyznaczono mierniki agregatowe, ułatwiające interpretację i ocenę uzyskanych wyników, w oparciu o formułę (1.36) zamieszczoną w rozdziale 1.

W wyniku agregacji cząstkowych mierników jakości zmiennych syntetycznych otrzymano 137700 agregatowych mierników jakości, które posłużyły następnie za podstawę do porządkowania ścieżek WAP według kryterium ich odporności na charakter oraz strukturę przetwarzanych zbiorów danych. Powyższe przekształcenia pozwoliły ostatecznie na uszeregowanie rozpatrywanych 306 konfiguracji według ich malejącej odporności na charakter i strukturę zbiorów danych liczbowych. Oznacza to, że jest możliwe wskazanie pewnych konfiguracji metod WAP, które „pracują" poprawnie niezależnie (lub umiarkowanie zależnie) od charakteru przetwarzanych danych, określonego zestawem charakterystyk statystycznych.

Rangowanie poszczególnych ścieżek przeprowadzono w kilku przekrojach. Na początku uwzględniono wszystkie uzyskane wyniki, przypisując poszczególnym konfiguracjom rangi z przedziału [1, 306], przy czym 306 punktów uzyskiwała ścieżka najlepsza, zaś 1 punkt - ścieżka najgorsza. Następnie przeprowadzono rangowanie uwzględniając p najlepszych ścieżek z każdego przebiegu symulacyjnego, przy czym wartość p przybierała kolejno wartości 10, 5, 3 i 1, zaś punkty przypisywano odpowiednio z przedziału $[1, p]$. Sposób rangowania ścieżek ilustruje algorytm 3.5.

Algorytm 3.5

Rangowanie ścieżek WAP

K01: Uporządkować rekordy pliku GAUGES.DBF według identyfikatorów przetwarzanych zbiorów
K02: Utworzyć listę ścieżek *vP*
K03: Wczytać agregatowe mierniki jakości dotyczące jednego zbioru do tablicy *vA*
K04: Posortować niemalejąco elementy tablicy *vA*, korzystając z funkcji składowej *qsort_up* z klasy *mva*
K05: Kolejnym pozycjom na liście ścieżek *vP* przypisać punkty z przedziału <1, *p*> zgodnie z miejscem danej ścieżki w uporządkowanej tablicy *vA*
K06: Jeżeli nie jest to ostatni zbiór danych, to przejść do punktu K03
K07: Uporządkować listę ścieżek *vP* według niemalejącej sumy uzyskanych punktów

Ponadto przeprowadzono analizę zbieżności wyników odnośnie do uzyskanego uporządkowania poszczególnych ścieżek WAP. Celem tejże analizy było uzyskanie odpowiedzi na pytanie, czy przetwarzania kolejnych zbiorów danych wpływa znacząco na uszeregowanie poszczególnych konfiguracji? Analizę tę prowadzono w oparciu zarówno o przetwarzane zbiory danych w trzech przedstawionych wyżej układach, jak również na podstawie dodatkowo wygenerowanych 150 zbiorów. Analiza zbieżności może być poza tym prowadzona na bieżąco w trakcie użytkowania programu komputerowego poprzez wbudowany mechanizm „samouczenia", zilustrowany algorytmem 3.6.

Algorytm 3.6

Analiza zbieżności wyników szeregowania ścieżek WAP

K01: Otworzyć plik PX.DBF zawierający dane o uporządkowaniu ścieżek WAP
K02: Utworzyć unikalny identyfikator sesji ID*
K03: Wczytać zbiór danych z pliku dyskowego *.DBF
K04: Określić charakter zmiennych metodą analizy czynnikowej, korzystając z funkcji składowych *hotelling*, *loadings*, *characters* klasy *mva*
K05: Wczytać układ ścieżki WAP z pliku PATHS.DBF
K06: Wyznaczyć wagi zmiennych przy pomocy funkcji składowej *weight* z klasy *mva*
K07: Znormalizować zmienne przy pomocy funkcji składowej *normal* z klasy *mva*

K08: Przekształcić zmienne destymulanty do postaci zmiennych stymulant przy pomocy funkcji składowej *dest2stym* z klasy *mva*

K09: Wyskalować elementy macierzy danych $\mathbf{X} = \left[x_{ijt} \right]_{n \times m \times v}$ w taki sposób, aby spełniony był warunek $x_{ijt} > 0$, korzystając z funkcji składowej *neg2pos* z klasy *mva*

K11: 1. Jeżeli agregacja bezwzorcowa, to obliczyć wektor *vAGR* zawierający realizacje zmiennej syntetycznej, korzystając z funkcji składowej *aggreg* z klasy *mva*

2. Jeżeli agregacja wzorcowa według dolnego bieguna zbioru danych, to wyznaczyć wektor zawierający antywzorzec przy pomocy funkcji *lowerpole* i to obliczyć wektor *vAGR* zawierający realizacje zmienne syntetycznej, korzystając z funkcji składowej *aggreg* z klasy *mva*

3. Jeżeli agregacja wzorcowa według górnego bieguna zbioru danych, to wyznaczyć wektor zawierający wzorzec przy pomocy funkcji *upperpole* i to obliczyć wektor *vAGR* zawierający realizacje zmienne syntetycznej, korzystając z funkcji składowej *aggreg* z klasy *mva*

K12: Wyznaczyć mierniki jakości zmiennej syntetycznej, korzystając z funkcji składowych *gauge01* ... *gauge12* z klasy *mva*

K13: Wyznaczyć agregatowy miernik jakości

K14: Zapisać identyfikator sesji i mierniki jakości w pliku GAUGES.DBF

K15: Zapisać agregatowy mierniki dotyczący *l*-tej ścieżki w tablicy *vAG*

K16: Zapisać numer *l*-tej ścieżki w tablicy *vP*

K17: Posortować niemalejąco elementy tablicy *vAG*, korzystając z funkcji składowej *qsort_up* z klasy *mva*

K18: Kolejnym pozycjom na liście ścieżek *vP* przypisać narastająco punkty z przedziału [1, *p*] zgodnie z miejscem danej ścieżki w uporządkowanej tablicy *vAG*

K19: Zaktualizować zawartość pliku PX.DBF

K20: Jeżeli nie jest to ostatnia ścieżka, to przejść do punktu K05

K21: Jeżeli nie jest to ostatni zbiór danych, to przejść do punktu K02

Rezultaty eksperymentów symulacyjnych oceniano także w świetle analizy wrażliwości, której podstawowe założenia przedstawiono w rozdziale 4 (punkt 4.4).

Rozdział 4

INTERPRETACJA UZYSKANYCH WYNIKÓW

4.1. Przekroje analizy wyników

Zgodnie z regułami przedstawionymi w punkcie 3.5, przeprowadzono rangowanie analizowanych ścieżek WAP kierując się kryterium wartości agregatowego miernika jakości. W celu eliminowania ścieżek „najgorszych" stopniowo zmniejszano wartość parametru *p*, uzyskując kolejno rezultaty dla *p* równego 306, 10, 5, 3 i 1. Analiza uzyskanych wyników umożliwiła wyodrębnienie podzbiorów ścieżek charakteryzujących się największą i najmniejszą odpornością i stabilnością w aspekcie przetwarzania zbiorów danych o rozmaitych właściwościach i strukturze.

Dodatkowo przeprowadzono rangowanie elementów składowych procedury WAP, tzn. systemów wag, sposobów normalizacji zmiennych i formuł wyznaczania cech agregatowych w celu uzyskania informacji o stabilności poszczególnych formuł.

Rezultaty eksperymentów symulacyjnych prezentowane są w przekroju trzech zestawów danych, zróżnicowanych pod względem parametrów omówionych w punkcie 3.3, a także w przekroju kolejnych wartości parametru *p*.

Obliczone rangi poszczególnych konfiguracji metod WAP prezentowane są zarówno w postaci wartości bezwzględnych, jak i w formie wskaźników dynamiki, ilustrujących wzajemne

różnice między kolejnymi ścieżkami. Uwagi te dotyczą również ocen związanych z elementami składowymi ścieżek WAP (formułami ważenie, normalizacji i agregacji zmiennych).

Ze względu na dużą liczbę i znaczne rozmiary tablic, w tekście pracy zaprezentowano jedynie wyniki najbardziej zwarte i syntetyczne, natomiast rezultaty szczegółowe, ale istotne z punktu widzenia przeprowadzonych badań, zamieszczono w aneksie do pracy.

Oznaczenia w postaci trójek liczb stosowane w tablicach prezentowanych w tekście pracy wynikają z iloczynu kartezjańskiego zbiorów W (formuły ważenia: 1-2), N (formuły normalizacji: 1-9) i A (formuły agregacji: 1-17), tzn.:

$$W \times N \times A = \{(w,n,a):(w \in W) \wedge (n \in N) \wedge (a \in A)\}$$

gdzie: w=1,2;
n=1,2,...,9;
a=1,2,...,17.

Odpowiednie wzory natomiast zamieszczone są w tablicach 1.1, 1.2 i 1.3 w rozdziale 1.

4.2. Uzyskane uporządkowania ścieżek

W rezultacie przeprowadzonych badań symulacyjnych otrzymano klasyfikacje analizowanych 306 ścieżek WAP w przekrojach omówionych w punkcie 4.1.

W tablicach 4.1-4.10 oraz A.4-A.19 (tablice zawarte w aneksie) przedstawiono wyniki, ilustrujące uporządkowanie ścieżek WAP w przekroju zestawów danych oraz w zależności od wartości parametru p, przy czym dane te dotyczą wskaźników dynamiki (tablice dla p=306 z

uwagi na rozmiary zamieszczono w aneksie). Ujęcie to umożliwia prześledzenie zmian jakościowych, ocenianych zestawem mierników, kolejnych procedur WAP. W aneksie do pracy załączono natomiast tablice zawierające rangi poszczególnych ścieżek w liczbach bezwzględnych.

Jak wynika z danych zamieszczonych w tablicach 4.1-4.10, uzyskane wyniki dla trzech zestawów zbiorów są bardzo zbliżone. Przyjmowane do oceny poszczególnych ścieżek różne wartości parametru p również nie wpływają na istotne zróżnicowanie ich uszeregowania. W przypadku, kiedy uwzględniono rangi poszczególnych konfiguracji w przekroju trzech zestawów danych dla p=1, najlepsze rezultaty uzyskały ścieżki oznaczone trójkami: {1,4,7}, {2,4,7}, {2,4,1}, {1,6,2}, {2,2,2}, {1,7,2}, {2,7,2}, {2,4,9}, {1,2,2}, {2,1,2}. W sytuacji, kiedy uwzględniono rangi poszczególnych konfiguracji w przekroju trzech zestawów danych oraz wartości dla p=1, p=3, p=5 i p=10, najlepsze okazały się ścieżki oznaczone trójkami: {1,4,7}, {2,4,7}, {2,2,2}, {1,6,2}, {1,7,2}, {1,2,2}, {2,4,2}, {2,7,2}, {2,1,2}, {2,3,2}. Za wyjątkiem rangowania wyczerpującego (dla p=306), we wszystkich innych analizowanych przypadkach (również dla wartości p równej 100, 50 i 25, których to wyników nie zamieszczono zarówno ze względu na ich podobieństwo, jak i z powodu obszerności tablic) czołowe pozycje zajmują ścieżki o numerach {1,4,7} i {2,4,7}. Dla p=1, a więc w sytuacji wyboru jednej, najlepszej ścieżki dla każdego z 450 przetwarzanych zbiorów danych, konfiguracje {1,4,7} i {2,4,7} uzyskały razem 171 punktów na 450 możliwych, czyli okazały się najlepsze w około 53 %. Dla p=3, a więc w sytuacji uwzględniania trzech najlepszych ścieżek dla każdego z 450 przetwarzanych zbiorów, konfiguracje {1,4,7} i {2,4,7} uzyskały łącznie 1122 punkty na 2700 możliwych, czyli były najlepsze w około 42 % przypadków. Konfiguracje, które uzyskały najwięcej punktów w przeprowadzonych eksperymentach złożone są z następujących elementów procedury WAP (12 „najlepszych" ścieżek w porządku chronologicznym):

- {1,2,2}: wagi stałe, normalizacja według przekształcenia ilorazowego z maksimum w mianowniku, agregacja bezwzorcowa według średniej geometrycznej,
- {1,4,7}: wagi stałe, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja wzorcowa według odległości Braya-Curtisa z dolnym biegunem zbioru,
- {1,6,2}: wagi stałe, standaryzacja (wartość przez odchylenie standardowe), agregacja bezwzorcowa według średniej geometrycznej,
- {1,7,2}: wagi stałe, unitaryzacja (wartość przez rozstęp), agregacja bezwzorcowa według średniej geometrycznej,
- {2,1,2}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z minimum w mianowniku, agregacja bezwzorcowa według średniej geometrycznej,
- {2,2,2}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z maksimum w mianowniku, agregacja bezwzorcowa według średniej geometrycznej,
- {2,3,2}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego ze średnią arytmetyczną w mianowniku, agregacja bezwzorcowa według średniej geometrycznej,
- {2,4,1}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja bezwzorcowa według średniej arytmetycznej,
- {2,4,2}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja bezwzorcowa według średniej geometrycznej,

- {2,4,7}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja wzorcowa według odległości Braya-Curtisa z dolnym biegunem zbioru,
- {2,4,9}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja wzorcowa według odległości „Canberra" z dolnym biegunem zbioru,
- {2,7,2}: wagi obliczone na podstawie współczynnika zmienności, unitaryzacja (wartość przez rozstęp), agregacja bezwzorcowa według średniej geometrycznej.

We wszystkich analizowanych przypadkach i przekrojach zdecydowanie najgorsze rezultaty uzyskiwały konfiguracje {2,4,3} i {1,4,3}, złożone z następujących elementów:

- {2,4,3}: wagi obliczone na podstawie współczynnika zmienności, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja bezwzorcowa według średniej harmonicznej,
- {1,4,3}: wagi stałe, normalizacja według przekształcenia ilorazowego z sumą obserwacji w mianowniku (udziały), agregacja bezwzorcowa według średniej harmonicznej.

Tablica 4.1

Uporządkowanie ścieżek WAP dla p=10 - zestaw danych nr 1

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	2,4,4	0,969	91	1,6,8	1,000	136	2,2,11	1,000
2	2,4,7	0,840	47	1,5,4	0,968	92	1,7,4	1,000	137	2,3,9	1,000
3	1,7,2	0,399	48	1,6,4	0,933	93	1,8,11	1,000	138	2,3,11	1,000
4	2,2,2	0,997	49	2,3,4	0,964	94	2,3,13	1,000	pozostałe		0,000
5	1,6,2	0,973	50	1,8,1	0,963	95	2,5,9	1,000			
6	2,7,2	0,914	51	2,4,17	1,000	96	2,7,3	1,000			
7	1,2,2	0,976	52	1,2,1	0,962	97	2,8,5	1,000			
8	1,4,9	0,990	53	1,1,1	0,920	98	1,2,9	0,889			
9	2,6,2	0,826	54	1,4,10	0,957	99	1,2,17	1,000			
10	2,3,2	0,886	55	1,6,5	1,000	100	1,3,3	1,000			
11	2,4,2	0,981	56	2,2,4	0,909	101	1,6,3	1,000			
12	2,1,2	0,985	57	1,3,10	0,950	102	2,5,17	1,000			
13	2,4,9	0,975	58	2,3,8	1,000	103	1,3,15	0,875			
14	1,3,2	0,879	59	1,7,9	0,947	104	1,8,13	1,000			
15	1,4,2	0,971	60	2,6,17	1,000	105	1,9,15	1,000			
16	1,8,2	0,982	61	2,2,8	0,944	106	2,1,17	1,000			
17	2,9,2	0,789	62	2,3,17	0,941	107	2,5,14	0,857			
18	1,1,2	0,939	63	2,8,4	1,000	108	1,1,8	0,833			
19	1,5,2	0,951	64	2,4,10	0,938	109	1,3,11	1,000			
20	1,9,2	0,897	65	2,7,4	1,000	110	1,4,11	1,000			
21	2,8,2	0,971	66	2,7,17	1,000	111	1,7,17	1,000			
22	2,7,1	0,961	67	1,2,8	0,933	112	2,3,15	1,000			
23	1,4,8	0,949	68	1,5,5	1,000	113	2,5,6	1,000			
24	2,5,2	0,968	69	2,9,4	1,000	114	2,6,10	1,000			
25	1,6,1	0,944	70	2,1,10	0,929	115	1,1,4	0,800			
26	2,3,1	0,941	71	2,9,5	1,000	116	1,3,7	1,000			
27	2,4,8	0,988	72	1,3,1	0,923	117	1,7,11	1,000			
28	2,4,1	0,987	73	1,9,4	1,000	118	2,8,14	1,000			
29	2,1,5	0,987	74	2,1,8	1,000	119	1,1,10	0,750			
30	1,7,1	0,883	75	2,7,5	1,000	120	1,1,17	1,000			
31	2,9,1	1,000	76	2,7,10	1,000	121	1,6,6	1,000			
32	2,1,1	0,941	77	1,3,5	0,917	122	1,9,13	1,000			
33	1,5,1	0,875	78	1,6,10	1,000	123	2,9,16	1,000			
34	2,2,1	1,000	79	1,6,17	1,000	124	1,3,9	0,667			
35	2,2,5	0,893	80	1,7,10	1,000	125	1,3,17	1,000			
36	1,1,5	0,960	81	1,3,8	0,909	126	1,5,11	1,000			
37	1,9,1	1,000	82	1,3,13	1,000	127	1,6,7	1,000			
38	2,3,5	1,000	83	1,5,15	1,000	128	1,6,16	1,000			
39	2,8,1	1,000	84	1,8,4	1,000	129	1,8,6	1,000			
40	2,4,5	0,979	85	1,8,14	1,000	130	2,2,17	1,000			
41	1,9,5	0,851	86	2,3,10	1,000	131	2,5,1	1,000			
42	2,6,1	0,950	87	1,2,5	0,900	132	2,7,11	1,000			
43	1,7,5	0,921	88	1,2,10	1,000	133	1,2,16	0,500			
44	1,8,5	0,971	89	1,4,1	1,000	134	1,7,7	1,000			
45	2,1,4	0,941	90	1,4,5	1,000	135	2,1,7	1,000			

Źródło: opracowanie własne.

Tablica 4.2

Uporządkowanie ścieżek WAP dla p=10 - zestaw danych nr 2

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	2,6,1	1,000	91	2,6,4	1,000
2	2,4,7	0,914	47	2,9,5	0,964	92	2,7,4	1,000
3	2,4,2	0,604	48	2,7,5	0,963	93	2,8,7	1,000
4	2,2,2	0,928	49	2,8,5	0,885	94	1,3,9	0,667
5	1,7,2	0,894	50	2,1,4	0,957	95	2,8,4	1,000
6	1,2,2	0,995	51	1,2,5	0,955	96	1,7,6	0,500
7	2,3,2	0,992	52	2,6,5	1,000	97	2,5,9	1,000
8	1,6,2	0,963	53	1,2,1	0,952	pozostałe		0,000
9	2,1,2	0,981	54	2,3,5	1,000			
10	1,4,2	0,810	55	2,5,5	1,000			
11	1,3,2	0,986	56	1,3,8	0,950			
12	1,1,2	0,989	57	1,6,4	0,947			
13	2,7,2	0,882	58	1,2,9	0,889			
14	2,6,2	0,988	59	1,7,4	1,000			
15	2,4,1	0,757	60	2,1,8	1,000			
16	1,9,2	0,625	61	2,7,1	1,000			
17	2,1,1	0,930	62	1,3,5	0,938			
18	1,4,9	0,972	63	1,4,5	1,000			
19	1,8,2	0,942	64	1,5,4	1,000			
20	2,9,2	0,847	65	2,3,4	0,933			
21	2,3,1	0,988	66	1,8,4	0,929			
22	2,3,16	0,976	67	2,2,9	0,846			
23	1,5,2	0,888	68	2,4,3	0,909			
24	2,3,17	0,986	69	2,5,1	1,000			
25	2,4,9	0,986	70	1,1,5	0,900			
26	2,8,2	1,000	71	1,1,8	1,000			
27	2,2,1	0,971	72	1,2,4	1,000			
28	1,7,1	0,985	73	1,9,4	0,889			
29	1,6,1	0,939	74	2,9,1	1,000			
30	2,2,8	1,000	75	1,1,1	0,875			
31	2,3,15	0,968	76	1,3,1	1,000			
32	2,5,2	0,983	77	2,2,4	1,000			
33	2,1,5	0,847	78	1,9,5	0,857			
34	2,4,6	1,000	79	2,4,17	1,000			
35	2,3,8	0,920	80	1,4,1	0,833			
36	2,4,4	0,978	81	1,5,6	1,000			
37	2,4,5	0,978	82	1,8,5	1,000			
38	1,9,1	0,977	83	2,4,8	1,000			
39	1,2,8	0,930	84	2,5,4	1,000			
40	1,6,5	1,000	85	2,8,1	1,000			
41	2,2,5	0,975	86	2,8,9	1,000			
42	1,5,5	0,974	87	1,7,5	0,800			
43	1,5,1	0,842	88	2,9,4	1,000			
44	1,8,1	0,938	89	1,4,17	0,750			
45	1,4,8	0,933	90	2,3,9	1,000			

Źródło: opracowanie własne.

Tablica 4.3

Uporządkowanie ścieżek WAP dla p=10 - zestaw danych nr 3

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	1,5,1	1,000	91	1,8,7	1,000
2	2,4,7	0,829	47	2,1,4	0,957	92	2,8,4	1,000
3	2,2,2	0,522	48	2,6,1	0,978	93	2,8,6	1,000
4	1,6,2	0,934	49	1,1,1	0,909	94	1,2,4	0,750
5	2,4,2	0,971	50	1,9,1	0,975	95	1,5,6	1,000
6	1,2,2	0,987	51	1,2,8	0,974	96	2,5,4	1,000
7	1,7,2	0,949	52	1,4,5	0,947	97	2,7,4	1,000
8	2,7,2	0,989	53	1,3,5	0,972	98	2,8,9	1,000
9	2,6,2	0,892	54	1,5,5	1,000	99	1,1,8	0,667
10	1,3,2	0,972	55	1,6,5	1,000	100	1,3,9	1,000
11	1,4,2	0,971	56	1,3,1	0,971	101	1,8,8	1,000
12	2,3,2	0,885	57	1,4,1	1,000	102	1,2,9	0,500
13	2,1,2	0,995	58	1,2,1	0,882	103	1,8,6	1,000
14	1,1,2	0,923	59	2,3,4	0,967	104	2,7,8	1,000
15	2,4,1	0,984	60	1,1,5	0,931	pozostałe		0,000
16	1,4,9	0,899	61	2,9,5	1,000			
17	1,8,2	0,899	62	1,5,4	0,963			
18	1,5,2	0,862	63	2,8,5	1,000			
19	1,9,2	0,954	64	1,8,1	0,962			
20	2,9,2	0,960	65	2,7,1	1,000			
21	2,1,1	0,983	66	1,6,4	0,960			
22	2,4,5	0,898	67	2,5,5	1,000			
23	2,5,2	0,953	68	2,6,5	1,000			
24	2,1,5	0,990	69	1,2,5	0,917			
25	2,4,9	0,990	70	2,7,5	1,000			
26	2,3,5	0,949	71	2,3,8	0,818			
27	2,8,2	0,989	72	2,2,4	0,889			
28	2,3,1	0,892	73	2,5,1	1,000			
29	1,6,1	0,976	74	1,3,8	0,938			
30	2,3,16	0,988	75	2,2,9	1,000			
31	2,3,17	0,875	76	2,9,1	1,000			
32	2,2,5	0,957	77	2,1,8	0,933			
33	1,7,1	0,940	78	1,7,4	0,857			
34	1,7,5	0,952	79	1,8,9	0,917			
35	2,3,15	1,000	80	2,4,3	0,909			
36	1,4,17	0,983	81	2,8,1	0,900			
37	2,4,4	0,983	82	1,9,4	0,889			
38	1,8,5	0,966	83	2,6,8	1,000			
39	1,9,5	0,929	84	1,8,4	0,875			
40	2,2,1	0,981	85	1,1,4	0,857			
41	2,2,8	1,000	86	2,9,4	1,000			
42	2,4,6	0,980	87	2,1,6	0,833			
43	2,4,8	0,980	88	2,6,4	1,000			
44	2,4,17	0,980	89	1,3,4	0,800			
45	1,4,8	0,979	90	1,4,4	1,000			

Źródło: opracowanie własne.

Tablica 4.4

Uporządkowanie ścieżek WAP dla p=5 - zestaw danych nr 1

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	2,8,1	1,000	91	1,7,10	1,000
2	2,4,7	0,822	47	1,3,13	0,833	92	2,3,4	1,000
3	1,4,9	0,294	48	1,5,15	1,000	93	2,5,14	1,000
4	1,6,2	0,907	49	1,6,17	1,000	94	2,7,4	1,000
5	1,7,2	1,000	50	1,8,14	1,000	95	2,7,5	1,000
6	2,4,9	0,794	51	2,1,10	1,000	pozostałe		0,000
7	1,2,2	0,935	52	2,4,10	1,000			
8	2,2,2	0,972	53	2,6,1	1,000			
9	2,7,2	0,957	54	2,7,17	1,000			
10	2,6,2	0,910	55	1,6,8	0,800			
11	1,8,2	0,803	56	1,8,11	1,000			
12	2,3,2	0,857	57	2,1,4	1,000			
13	2,1,2	0,929	58	2,2,5	1,000			
14	2,9,2	0,974	59	2,3,8	1,000			
15	2,4,2	0,947	60	2,3,13	1,000			
16	1,3,2	0,972	61	2,5,9	1,000			
17	1,4,2	1,000	62	2,7,3	1,000			
18	1,5,2	0,771	63	1,2,1	0,750			
19	1,4,8	0,963	64	1,2,17	1,000			
20	2,4,8	0,923	65	1,3,3	1,000			
21	2,7,1	0,958	66	1,6,3	1,000			
22	1,6,1	0,957	67	1,6,5	1,000			
23	2,8,2	1,000	68	1,7,5	1,000			
24	1,9,2	0,909	69	1,8,1	1,000			
25	1,1,2	0,900	70	2,2,4	1,000			
26	2,1,5	0,944	71	2,2,8	1,000			
27	2,1,1	0,941	72	2,3,10	1,000			
28	1,1,5	0,938	73	2,4,17	1,000			
29	2,5,2	1,000	74	2,5,17	1,000			
30	1,7,1	0,800	75	2,7,10	1,000			
31	2,9,1	0,917	76	1,3,5	0,667			
32	2,3,5	0,909	77	1,3,8	1,000			
33	1,5,1	0,900	78	1,3,15	1,000			
34	2,3,1	1,000	79	1,6,4	1,000			
35	2,4,5	1,000	80	1,6,10	1,000			
36	1,7,9	0,889	81	1,8,13	1,000			
37	2,4,1	1,000	82	1,9,15	1,000			
38	2,6,17	1,000	83	2,1,8	1,000			
39	1,4,10	0,875	84	2,1,17	1,000			
40	1,9,5	1,000	85	2,4,4	1,000			
41	1,3,10	0,857	86	2,9,5	1,000			
42	1,8,5	1,000	87	1,2,8	0,500			
43	1,9,1	1,000	88	1,2,10	1,000			
44	2,2,1	1,000	89	1,4,5	1,000			
45	2,3,17	1,000	90	1,5,4	1,000			

Źródło: opracowanie własne.

Tablica 4.5

Uporządkowanie ścieżek WAP dla p=5 - zestaw danych nr 2

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	1,2,9	0,750
2	2,4,7	0,854	47	2,1,4	1,000
3	1,6,2	0,371	48	2,1,8	1,000
4	1,7,2	0,991	49	2,5,5	1,000
5	2,4,2	0,991	50	2,6,1	1,000
6	2,2,2	0,939	51	2,6,5	1,000
7	1,2,2	0,963	52	2,7,5	1,000
8	2,1,2	0,827	53	1,2,5	0,667
9	2,3,2	0,907	54	1,3,5	1,000
10	2,7,2	0,910	55	1,5,4	1,000
11	2,4,1	0,930	56	1,7,4	1,000
12	1,3,2	0,864	57	2,4,4	1,000
13	1,1,2	0,982	58	2,4,5	1,000
14	1,4,2	0,964	59	2,7,1	1,000
15	2,6,2	0,963	60	2,9,5	1,000
16	1,4,9	0,712	61	1,1,8	0,500
17	1,9,2	0,973	62	1,4,5	1,000
18	2,3,16	0,833	63	1,6,4	1,000
19	2,4,9	0,767	64	1,8,4	1,000
20	1,8,2	0,913	65	2,3,4	1,000
21	2,1,1	0,952	66	2,3,5	1,000
22	2,3,17	1,000	67	2,8,5	1,000
23	2,2,8	0,950	68	2,9,1	1,000
24	1,6,1	0,947	pozostałe		0,000
25	2,9,2	1,000			
26	1,7,1	0,944			
27	1,5,2	0,941			
28	2,3,1	1,000			
29	2,1,5	0,938			
30	2,8,2	1,000			
31	2,3,8	0,800			
32	2,5,2	0,917			
33	1,2,8	0,909			
34	1,6,5	1,000			
35	1,9,1	1,000			
36	2,3,15	1,000			
37	2,2,5	0,900			
38	1,4,8	0,889			
39	1,5,5	1,000			
40	2,2,1	1,000			
41	1,2,1	0,750			
42	1,3,8	0,833			
43	1,8,1	1,000			
44	1,2,4	0,800			
45	1,5,1	1,000			

Źródło: opracowanie własne.

Tablica 4.6

Uporządkowanie ścieżek WAP dla p=5 - zestaw danych nr 3

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	1,2,8	1,000
2	2,4,7	0,740	47	1,3,5	1,000
3	1,6,2	0,378	48	1,4,1	0,889
4	2,2,2	0,878	49	1,5,1	1,000
5	1,7,2	0,965	50	1,9,1	1,000
6	2,4,2	0,988	51	1,1,1	0,875
7	1,2,2	0,866	52	1,3,1	1,000
8	1,4,9	0,972	53	2,9,5	1,000
9	2,7,2	0,986	54	2,1,4	0,857
10	2,4,1	0,985	55	2,2,1	1,000
11	1,3,2	0,836	56	1,2,5	0,833
12	2,1,2	0,946	57	1,8,1	1,000
13	2,6,2	1,000	58	2,6,5	1,000
14	1,4,2	0,981	59	1,2,1	0,800
15	1,1,2	0,769	60	1,3,8	1,000
16	1,9,2	1,000	61	2,5,5	1,000
17	2,3,2	1,000	62	2,8,5	1,000
18	1,8,2	0,950	63	1,5,5	0,750
19	2,1,5	0,974	64	1,6,5	1,000
20	2,4,9	0,973	65	2,3,8	1,000
21	1,5,2	0,917	66	2,6,8	1,000
22	2,3,16	0,909	67	2,7,1	1,000
23	2,9,2	0,933	68	1,5,4	0,667
24	2,1,1	0,857	69	1,6,4	1,000
25	2,8,2	1,000	70	2,1,8	1,000
26	2,3,5	0,875	71	2,4,4	1,000
27	2,4,5	1,000	72	2,5,1	1,000
28	2,5,2	1,000	73	2,7,5	1,000
29	1,6,1	0,952	74	2,9,1	1,000
30	2,3,17	1,000	75	1,7,4	0,500
31	1,4,17	0,900	76	2,2,4	1,000
32	2,4,17	0,944	77	2,3,4	1,000
33	1,7,5	0,882	78	2,8,1	1,000
34	2,3,1	1,000	pozostałe		0,000
35	2,2,5	0,933			
36	1,8,5	0,929			
37	2,2,8	1,000			
38	2,4,8	1,000			
39	1,4,8	0,923			
40	1,7,1	1,000			
41	1,4,5	0,833			
42	1,9,5	1,000			
43	2,3,15	1,000			
44	2,6,1	1,000			
45	1,1,5	0,900			

Źródło: opracowanie własne.

Tablica 4.7

Uporządkowanie ścieżek WAP dla p=3 - zestaw danych nr 1

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	2,3,13	1,000
2	2,4,7	0,810	47	2,3,17	1,000
3	1,4,9	0,224	48	2,4,10	1,000
4	1,6,2	0,907	49	2,5,9	1,000
5	1,7,2	0,949	50	2,7,3	1,000
6	2,4,9	0,865	51	2,8,1	1,000
7	1,2,2	0,781	52	1,2,1	0,500
8	2,6,2	1,000	53	1,2,17	1,000
9	2,2,2	0,760	54	1,3,3	1,000
10	1,8,2	0,947	55	1,6,3	1,000
11	2,7,2	1,000	56	1,9,1	1,000
12	2,9,2	0,778	57	2,2,1	1,000
13	2,1,2	0,929	58	2,3,8	1,000
14	2,3,2	0,923	59	2,3,10	1,000
15	1,4,2	0,917	60	2,4,1	1,000
16	1,5,2	0,909	61	2,4,5	1,000
17	1,4,8	0,900	62	2,5,17	1,000
18	2,4,2	1,000	63	2,7,10	1,000
19	2,4,8	1,000	64	2,7,17	1,000
20	1,6,1	0,889	65	2,9,1	1,000
21	2,1,5	1,000	pozostałe		0,000
22	2,8,2	1,000			
23	1,3,2	0,875			
24	2,1,1	0,714			
25	2,5,2	1,000			
26	2,7,1	1,000			
27	1,1,5	0,800			
28	1,7,1	1,000			
29	1,7,9	1,000			
30	2,6,17	1,000			
31	1,1,2	0,750			
32	1,3,10	1,000			
33	1,3,13	1,000			
34	1,4,10	1,000			
35	1,5,15	1,000			
36	1,6,17	1,000			
37	1,8,14	1,000			
38	1,9,2	1,000			
39	2,1,10	1,000			
40	2,3,5	1,000			
41	1,6,8	0,667			
42	1,8,5	1,000			
43	1,8,11	1,000			
44	1,9,5	1,000			
45	2,3,1	1,000			

Źródło: opracowanie własne.

Tablica 4.8

Uporządkowanie ścieżek WAP dla p=3 - zestaw danych nr 2

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	2,7,5	1,000
2	2,4,7	0,850	pozostałe		0,000
3	1,6,2	0,282			
4	1,7,2	0,958			
5	1,2,2	0,848			
6	2,4,2	0,974			
7	2,2,2	0,947			
8	2,4,1	1,000			
9	2,7,2	0,833			
10	2,1,2	0,900			
11	2,3,2	0,704			
12	1,1,2	0,947			
13	1,3,2	0,889			
14	1,4,9	1,000			
15	2,6,2	1,000			
16	1,4,2	0,938			
17	1,9,2	0,800			
18	2,3,16	0,833			
19	2,2,8	0,900			
20	1,7,1	0,889			
21	2,4,9	1,000			
22	2,1,1	0,875			
23	2,1,5	1,000			
24	1,5,2	0,857			
25	1,6,1	1,000			
26	1,8,2	1,000			
27	2,5,2	1,000			
28	2,9,2	1,000			
29	2,3,1	0,833			
30	2,3,8	1,000			
31	1,9,1	0,800			
32	1,2,1	0,750			
33	1,2,8	1,000			
34	2,8,2	1,000			
35	1,2,4	0,667			
36	1,3,8	1,000			
37	1,6,5	1,000			
38	2,2,5	1,000			
39	1,4,8	0,500			
40	1,5,1	1,000			
41	1,8,1	1,000			
42	2,1,4	1,000			
43	2,1,8	1,000			
44	2,2,1	1,000			
45	2,5,5	1,000			

Źródło: opracowanie własne.

Tablica 4.9

Uporządkowanie ścieżek WAP dla p=3 - zestaw danych nr 3

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	46	1,2,8	0,667
2	2,4,7	0,682	47	1,3,8	1,000
3	1,6,2	0,313	48	1,9,5	1,000
4	2,2,2	0,854	49	1,1,1	0,500
5	2,4,1	0,943	50	1,2,1	1,000
6	2,4,2	1,000	51	1,2,5	1,000
7	1,4,9	0,970	52	1,3,1	1,000
8	1,7,2	0,938	53	1,5,1	1,000
9	2,1,2	0,733	54	1,6,5	1,000
10	2,7,2	1,000	55	1,8,1	1,000
11	1,2,2	0,955	56	2,2,1	1,000
12	2,1,5	0,905	57	2,3,8	1,000
13	1,3,2	0,947	58	2,6,5	1,000
14	2,6,2	1,000	59	2,6,8	1,000
15	1,9,2	0,944	60	2,7,1	1,000
16	1,4,2	0,941	61	2,9,5	1,000
17	1,8,2	1,000	pozostałe		0,000
18	2,4,9	1,000			
19	1,1,2	0,875			
20	2,3,2	1,000			
21	1,5,2	0,786			
22	2,3,16	0,909			
23	2,9,2	0,900			
24	1,4,17	0,889			
25	2,1,1	1,000			
26	2,4,17	1,000			
27	2,4,5	0,875			
28	2,8,2	1,000			
29	1,6,1	0,857			
30	2,2,8	1,000			
31	2,3,5	1,000			
32	1,7,1	0,833			
33	1,7,5	1,000			
34	2,2,5	1,000			
35	2,5,2	1,000			
36	1,1,5	0,800			
37	1,4,5	1,000			
38	1,8,5	1,000			
39	2,3,1	1,000			
40	2,4,8	1,000			
41	1,3,5	0,750			
42	1,4,1	1,000			
43	1,4,8	1,000			
44	1,9,1	1,000			
45	2,6,1	1,000			

Źródło: opracowanie własne.

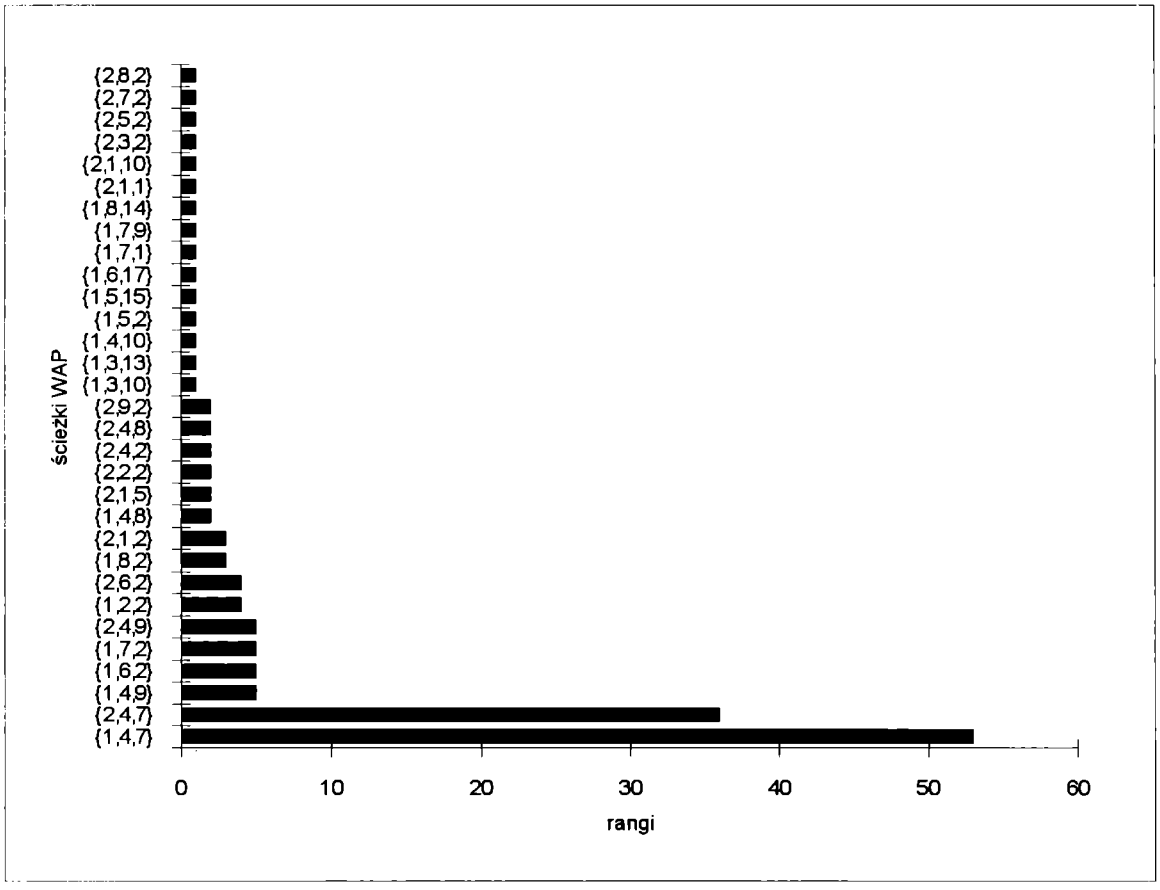
Tablica 4.10

Uporządkowanie ścieżek WAP dla p=1

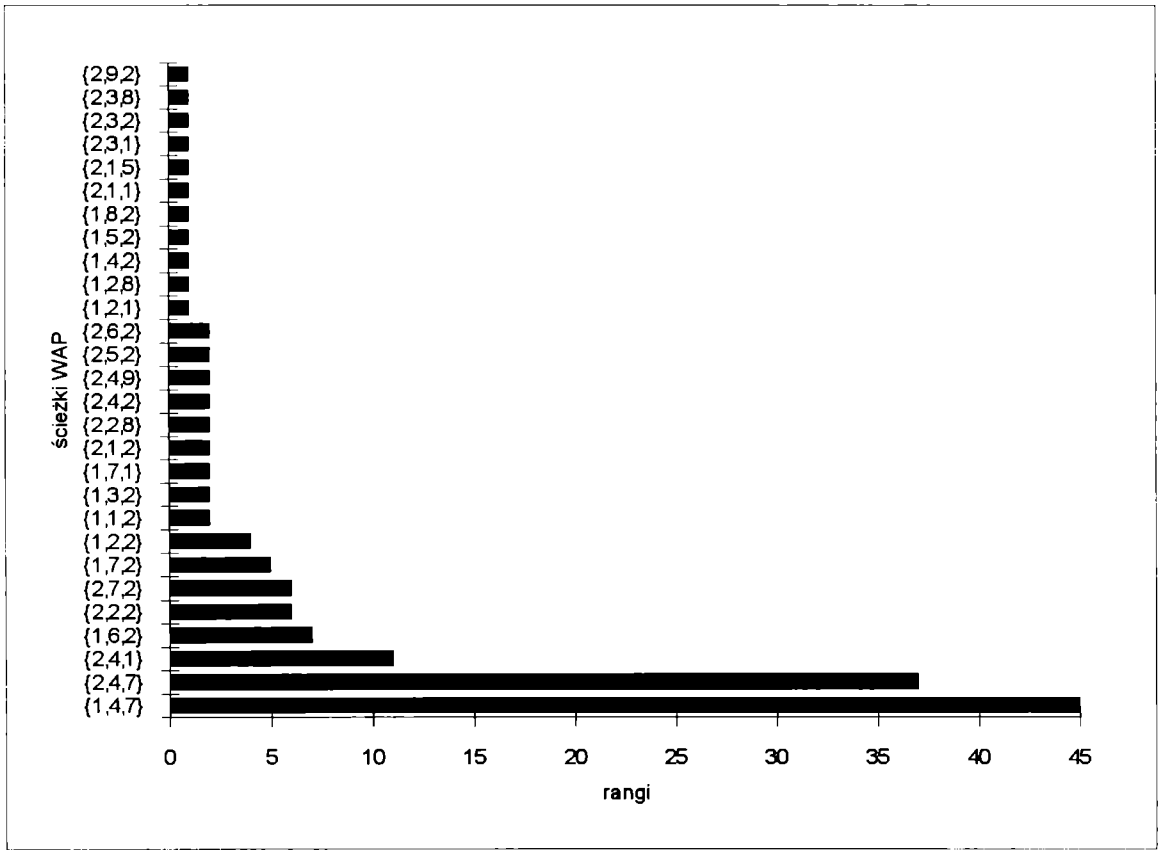
Zestaw danych nr 1			Zestaw danych nr 2			Zestaw danych nr 3		
Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	1,4,7	-	1	1,4,7	-	1	1,4,7	-
2	2,4,7	0,679	2	2,4,7	0,822	2	2,4,7	0,435
3	1,4,9	0,139	3	2,4,1	0,297	3	2,4,1	0,500
4	1,6,2	1,000	4	1,6,2	0,636	4	1,6,2	0,800
5	1,7,2	1,000	5	2,2,2	0,857	5	2,1,2	0,625
6	2,4,9	1,000	6	2,7,2	1,000	6	2,2,2	1,000
7	1,2,2	0,800	7	1,7,2	0,833	7	2,4,9	1,000
8	2,6,2	1,000	8	1,2,2	0,800	8	2,7,2	1,000
9	1,8,2	0,750	9	1,1,2	0,500	9	1,4,9	0,800
10	2,1,2	1,000	10	1,3,2	1,000	10	1,9,2	1,000
11	1,4,8	0,667	11	1,7,1	1,000	11	2,1,5	1,000
12	2,1,5	1,000	12	2,1,2	1,000	12	1,2,2	0,750
13	2,2,2	1,000	13	2,2,8	1,000	13	1,7,2	1,000
14	2,4,2	1,000	14	2,4,2	1,000	14	1,8,2	1,000
15	2,4,8	1,000	15	2,4,9	1,000	15	2,3,2	1,000
16	2,9,2	1,000	16	2,5,2	1,000	16	1,1,2	0,667
17	1,3,10	0,500	17	2,6,2	1,000	17	1,4,2	1,000
18	1,3,13	1,000	18	1,2,1	0,500	18	2,4,17	1,000
19	1,4,10	1,000	19	1,2,8	1,000	19	2,6,2	1,000
20	1,5,2	1,000	20	1,4,2	1,000	20	1,1,5	0,500
21	1,5,15	1,000	21	1,5,2	1,000	21	1,3,2	1,000
22	1,6,17	1,000	22	1,8,2	1,000	22	1,4,5	1,000
23	1,7,1	1,000	23	2,1,1	1,000	23	1,4,17	1,000
24	1,7,9	1,000	24	2,1,5	1,000	24	1,5,2	1,000
25	1,8,14	1,000	25	2,3,1	1,000	25	1,6,1	1,000
26	2,1,1	1,000	26	2,3,2	1,000	26	1,7,1	1,000
27	2,1,10	1,000	27	2,3,8	1,000	27	1,8,5	1,000
28	2,3,2	1,000	28	2,9,2	1,000	28	2,1,1	1,000
29	2,5,2	1,000	pozostałe		0,000	29	2,2,5	1,000
30	2,7,2	1,000				30	2,4,5	1,000
31	2,8,2	1,000				31	2,6,1	1,000
pozostałe		0,000				32	2,8,2	1,000
						33	2,9,2	1,000
						pozostałe		0,000

Źródło: opracowanie własne.

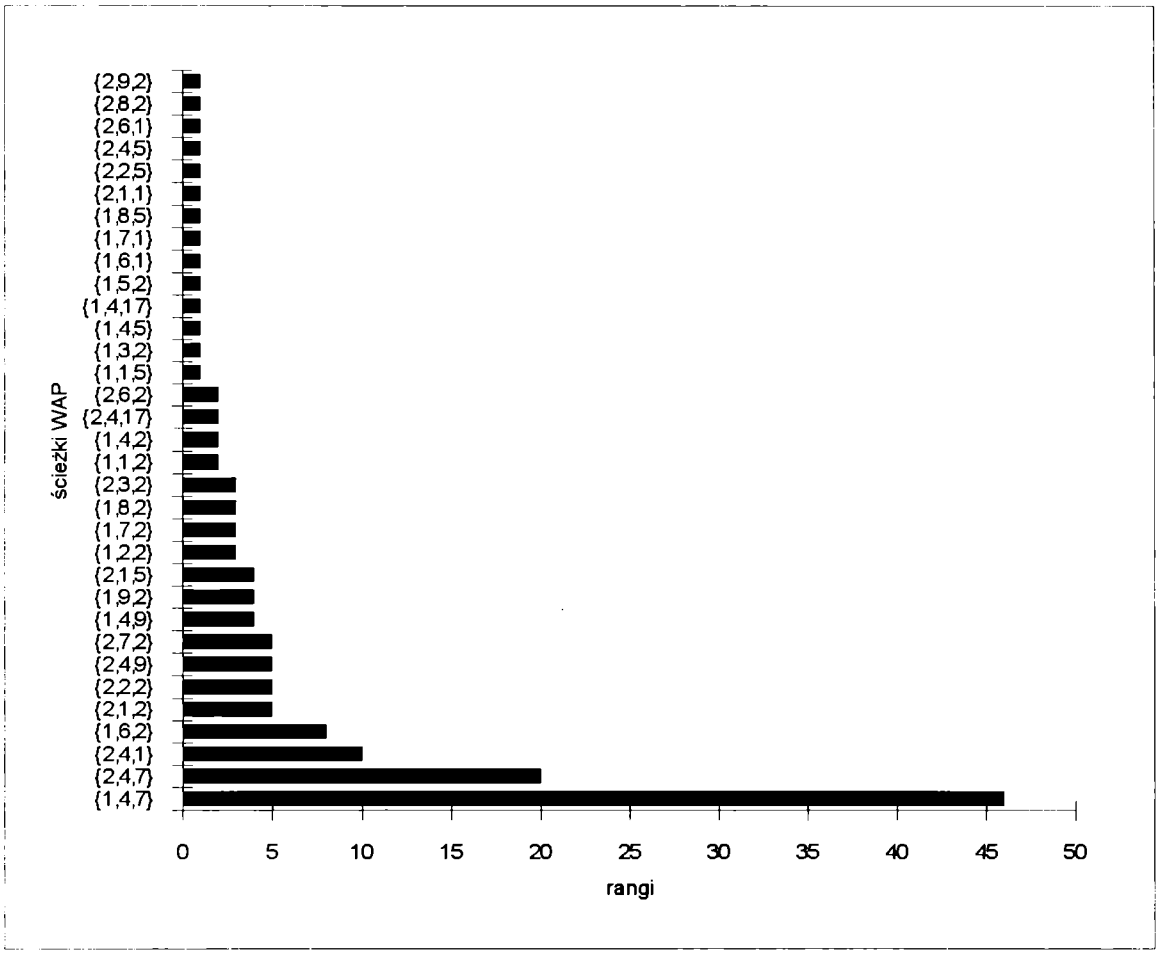
Na rysunkach 4.1-4.3 przedstawiono uporządkowania ścieżek WAP, które uzyskały więcej niż 1 punkt przy wartości parametru p=1, w poszczególnych trzech zestawach danych. Rysunek 4.4 przedstawia z kolei dziesięć najlepszych ścieżek w przekroju analizowanych łącznie trzech zestawów danych. Podstawą wykreślonych histogramów, ilustrujących skalę różnic pomiędzy poszczególnymi konfiguracjami, były rangi w wartościach bezwzględnych, zamieszczone w aneksie.



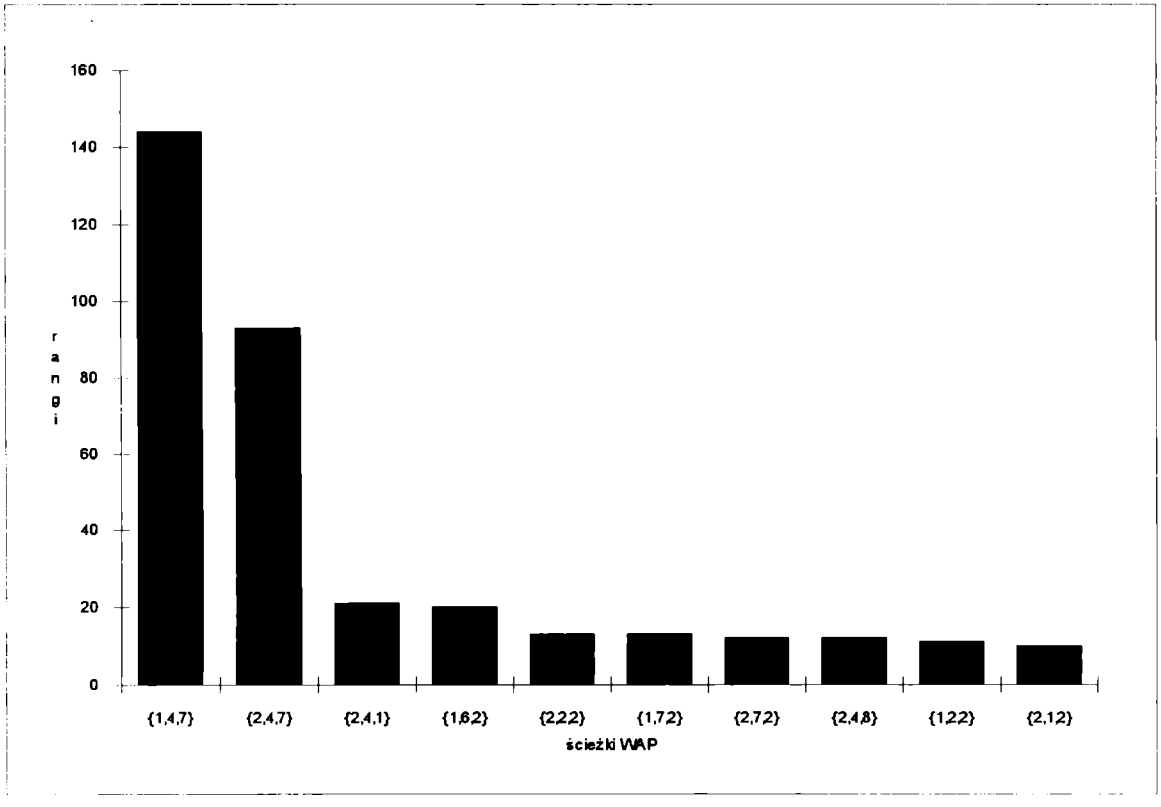
Rysunek 4.1. Uporządkowanie ścieżek dla p=1 - zestaw danych nr 1



Rysunek 4.2. Uporządkowanie ścieżek dla p=1 - zestaw danych nr 2



Rysunek 4.3. Uporządkowanie ścieżek dla p=1 - zestaw danych nr 3



Rysunek 4.4. Dziesięć pierwszych ścieżek w przekroju trzech zestawów danych

W tablicach 4.11-4.19 zestawiono wyniki dotyczące uporządkowania poszczególnych elementów składowych procedury WAP. Z przedstawionych danych można wyprowadzić następujące wnioski:

1. Sposoby ważenia zmiennych. W większości przypadków (za wyjątkiem p=306 w obu zestawach danych i p=10 w drugim zestawie danych) lepszy okazuje się system wag stałych, a więc jednakowe znaczenie przypisane wszystkim cechom. Generalnie można stwierdzić, że ocena punktowa w liczbach bezwzględnych obu analizowanych systemów wag nie różni się drastycznie. W literaturze przedmiotu podkreśla się, iż problem ważenia zmiennych nie jest ostatecznie i zadawalająco rozwiązany (por. [65], [139], [174]). Istnieją argumenty przemawiające zarówno na rzecz wag stałych (jednakowe znaczenie wszystkich cech), jak i na korzyść wag różnicujących znaczenie poszczególnych cech (por. np. [28]).[1]

2. Sposoby normalizacji zmiennych. W większości przypadków (za wyjątkiem p=306 w obu zestawach danych) najlepszy okazuje się normalizacja udziałowa (przekształcenie ilorazowe z sumą obserwacji w mianowniku). W dalszej kolejności znajduje się przekształcenie ilorazowe z maksimum w mianowniku, unitaryzacja (wartość przez rozstęp) oraz standaryzacja. Zwraca tutaj wagę nie najlepsza pozycja standaryzacji, bardzo często stosowanej w praktyce w różnego rodzaju analizach taksonomicznych. Krytyczne uwagi dotyczące standaryzacji w aspekcie badaniach podobieństwa struktury, poziomu i zależności przedstawiono w pracy [49, s. 118-122]. Szereg uwag merytorycznych odnośnie do sposobów normalizacji cech zawiera natomiast praca [29].

3. Sposoby agregacji zmiennych. W przypadku p=1 dla obu zestawów danych oraz p=3 dla pierwszego zestawu najlepsza okazała się formuła agregacji według odległości Braya-Curtisa z dolnym biegunem zbioru, natomiast we wszystkich pozostałych przypadkach na pierwszym miejscu znalazła się bezwzorcowa formuła agregacji według średniej geometrycz-

[1] Krytyczną dyskusję dotyczącą zagadnienia ważenia zmiennych zawiera praca [1]. Przedstawiono w niej również pewną propozycję kompromisową.

nej. Na dalszych miejscach znalazły się: formuła odległości „Canberra" z dolnym biegunem zbioru oraz bezwzorcowa agregacja według średniej arytmetycznej. Uzasadnieniem bardzo dobrego usytuowania bezwzorcowej agregacji według średniej geometrycznej może być jej własność „wygładzania" skrajnych i nietypowych wartości w zbiorze obserwacji [183].

Tablica 4.11

Uporządkowanie formuł ważenia zmiennych - zestaw danych nr 1

Parametr *p*	Pozycja	Waga	Ranga	r_i/r_{i-1}
306	1	2	3533042	-
	2	1	3512608	0,994
10	1	1	4130	-
	2	2	4120	0,998
5	1	1	1192	-
	2	2	1058	0,888
3	1	1	493	-
	2	2	407	0,826
1	1	1	86	-
	2	2	64	0,744

Źródło: opracowanie własne.

Tablica 4.12

Uporządkowanie formuł normalizacji zmiennych - zestaw danych nr 1

Parametr p	Pozycja	Normalizacja	Ranga	r_i/r_{i-1}
306	1	2	828424	-
	2	3	824374	0,995
	3	8	803233	0,974
	4	9	797786	0,993
	5	1	789947	0,990
	6	7	776245	0,983
	7	5	746375	0,962
	8	6	746264	1,000
	9	4	733002	0,982
10	1	4	3112	-
	2	7	946	0,304
	3	2	844	0,892
	4	6	804	0,953
	5	3	690	0,858
	6	1	618	0,896
	7	9	444	0,718
	8	8	443	0,998
	9	5	349	0,788
5	1	4	1147	-
	2	7	225	0,196
	3	6	212	0,942
	4	2	166	0,783
	5	3	134	0,807
	6	1	118	0,881
	7	8	97	0,822
	8	9	86	0,887
	9	5	65	0,756
3	1	4	549	-
	2	6	82	0,149
	3	7	72	0,878
	4	2	47	0,653
	5	3	37	0,787
	6	1	36	0,973
	7	8	35	0,972
	8	5	21	0,600
	9	9	21	1,000
1	1	4	106	-
	2	6	10	0,094
	3	7	8	0,800
	4	1	7	0,875
	5	2	6	0,857
	6	8	5	0,833
	7	3	3	0,600
	8	5	3	1,000
	9	9	2	0,667

Źródło: opracowanie własne.

Tablica 4.13

Uporządkowanie formuł agregacji zmiennych - zestaw danych nr 1

Parametr p	Pozycja	Agregacja	Ranga	r_i/r_{i-1}
306	1	2	710537	-
	2	1	693743	0,976
	3	4	674485	0,972
	4	5	664725	0,986
	5	7	554513	0,834
	6	9	525229	0,947
	7	6	512568	0,976
	8	8	498298	0,972
	9	17	452139	0,907
	10	10	353976	0,783
	11	12	239002	0,675
	12	15	236369	0,989
	13	11	226428	0,958
	14	3	207578	0,917
	15	16	193101	0,930
	16	14	153101	0,793
	17	13	149858	0,979
10	1	2	3614	-
	2	7	1846	0,511
	3	1	884	0,479
	4	9	523	0,592
	5	5	478	0,914
	6	8	258	0,540
	7	4	248	0,961
	8	10	130	0,524
	9	17	121	0,931
	10	11	29	0,240
	11	13	29	1,000
	12	15	29	1,000
	13	3	25	0,862
	14	14	20	0,800
	15	6	10	0,500
	16	16	6	0,600
	17	12	0	0,000
5	1	2	840	-
	2	7	807	0,961
	3	9	196	0,243
	4	1	139	0,709
	5	5	80	0,576
	6	8	66	0,825
	7	17	35	0,530
	8	10	33	0,943
	9	4	14	0,424
	10	13	11	0,786
	11	3	10	0,909
	12	15	9	0,900
	13	14	6	0,667
	14	11	4	0,667
	15	6	0	0,000
	16	12	0	0,000
	17	16	0	0,000

3	1	7	429	-
	2	2	276	0,643
	3	9	81	0,293
	4	1	31	0,383
	5	8	21	0,677
	6	5	20	0,952
	7	10	13	0,650
	8	17	12	0,923
	9	13	5	0,417
	10	3	4	0,800
	11	14	3	0,750
	12	15	3	1,000
	13	11	2	0,667
	14	4	0	0,000
	15	6	0	0,000
	16	12	0	0,000
	17	16	0	0,000
1	1	7	89	-
	2	2	35	0,393
	3	9	11	0,314
	4	8	4	0,364
	5	10	3	0,750
	6	1	2	0,667
	7	5	2	1,000
	8	13	1	0,500
	9	14	1	1,000
	10	15	1	1,000
	11	17	1	1,000
	12	3	0	0,000
	13	4	0	0,000
	14	6	0	0,000
	15	11	0	0,000
	16	12	0	0,000
	17	16	0	0,000

Źródło: opracowanie własne.

Tablica 4.14

Uporządkowanie formuł ważenia zmiennych - zestaw danych nr 2

Parametr p	Pozycja	Waga	Ranga	r_i/r_{i-1}
306	1	2	3661030	-
	2	1	3384620	0,924
10	1	2	4444	-
	2	1	3806	0,856
5	1	2	1147	-
	2	1	1103	0,962
3	1	1	455	-
	2	2	445	0,978
1	1	2	78	-
	2	1	72	0,923

Źródło: opracowanie własne.

Tablica 4.15

Uporządkowanie formuł normalizacji zmiennych - zestaw danych nr 2

Parametr p	Pozycja	Normalizacja	Ranga	r_i/r_{i-1}
306	1	3	850834	-
	2	2	842277	0,990
	3	1	835820	0,992
	4	8	797000	0,954
	5	9	789815	0,991
	6	7	748925	0,948
	7	5	745062	0,995
	8	6	735552	0,987
	9	4	700365	0,952
10	1	4	2896	-
	2	2	1093	0,377
	3	3	1074	0,983
	4	1	852	0,793
	5	6	775	0,910
	6	7	757	0,977
	7	9	294	0,388
	8	5	256	0,871
	9	8	253	0,988
5	1	4	992	-
	2	2	273	0,275
	3	3	232	0,850
	4	7	211	0,909
	5	6	204	0,967
	6	1	184	0,902
	7	9	67	0,364
	8	5	44	0,657
	9	8	43	0,977
3	1	4	484	-
	2	2	95	0,196
	3	7	85	0,895
	4	6	72	0,847
	5	1	61	0,847
	6	3	57	0,934
	7	9	22	0,386
	8	5	14	0,636
	9	8	10	0,714
1	1	4	98	-
	2	2	14	0,143
	3	7	13	0,929
	4	6	9	0,692
	5	1	6	0,667
	6	3	5	0,833
	7	5	3	0,600
	8	8	1	0,333
	9	9	1	1,000

Źródło: opracowanie własne.

Tablica 4.16

Uporządkowanie formuł agregacji zmiennych - zestaw danych nr 2

Parametr *p*	Pozycja	Agregacja	Ranga	r_i/r_{i-1}
306	1	2	722971	-
	2	1	691236	0,956
	3	4	669356	0,968
	4	5	655146	0,979
	5	7	553010	0,844
	6	9	522530	0,945
	7	8	504190	0,965
	8	6	501736	0,995
	9	17	459629	0,916
	10	10	323944	0,705
	11	15	245234	0,757
	12	12	237707	0,969
	13	11	227460	0,957
	14	3	214603	0,943
	15	16	197363	0,920
	16	13	166099	0,842
	17	14	153436	0,924
10	1	2	4555	-
	2	7	1588	0,349
	3	1	779	0,491
	4	5	423	0,543
	5	8	225	0,532
	6	9	211	0,938
	7	4	184	0,872
	8	16	80	0,435
	9	17	79	0,988
	10	15	60	0,759
	11	6	56	0,933
	12	3	10	0,179
	13	10	0	0,000
	14	11	0	0,000
	15	12	0	0,000
	16	13	0	0,000
	17	14	0	0,000
5	1	2	1131	-
	2	7	684	0,605
	3	1	176	0,257
	4	9	63	0,358
	5	5	62	0,984
	6	8	58	0,935
	7	16	30	0,517
	8	17	20	0,667
	9	4	16	0,800
	10	15	10	0,625
	11	3	0	0,000
	12	6	0	0,000
	13	10	0	0,000
	14	11	0	0,000
	15	12	0	0,000
	16	13	0	0,000
	17	14	0	0,000

3	1	2	387	-
	2	7	370	0,956
	3	1	72	0,195
	4	9	24	0,333
	5	8	21	0,875
	6	5	13	0,619
	7	16	10	0,769
	8	4	3	0,300
	9	3	0	0,000
	10	6	0	0,000
	11	10	0	0,000
	12	11	0	0,000
	13	12	0	0,000
	14	13	0	0,000
	15	14	0	0,000
	16	15	0	0,000
	17	17	0	0,000
1	1	7	82	-
	2	2	45	0,549
	3	1	16	0,356
	4	8	4	0,250
	5	9	2	0,500
	6	5	1	0,500
	7	3	0	0,000
	8	4	0	0,000
	9	6	0	0,000
	10	10	0	0,000
	11	11	0	0,000
	12	12	0	0,000
	13	13	0	0,000
	14	14	0	0,000
	15	15	0	0,000
	16	16	0	0,000
	17	17	0	0,000

Źródło: opracowanie własne.

Tablica 4.17

Uporządkowanie formuł ważenia zmiennych - zestaw danych nr 3

Parametr p	Pozycja	Waga	Ranga	r_i/r_{i-1}
306	1	2	3647140	-
	2	1	3398510	0,932
10	1	2	4319	-
	2	1	3931	0,910
5	1	1	1134	-
	2	2	1116	0,984
3	1	1	469	-
	2	2	431	0,919
1	1	1	83	-
	2	2	67	0,807

Źródło: opracowanie własne.

Tablica 4.18

Uporządkowanie formuł normalizacji zmiennych - zestaw danych nr 3

Parametr p	Pozycja	Normalizacja	Ranga	r_i/r_{i-1}
306	1	2	836992	-
	2	3	831329	0,993
	3	8	805490	0,969
	4	1	804729	0,999
	5	9	797182	0,991
	6	7	774955	0,972
	7	5	751795	0,970
	8	6	745262	0,991
	9	4	697916	0,936
10	1	4	2896	-
	2	3	974	0,336
	3	2	923	0,948
	4	6	780	0,845
	5	1	755	0,968
	6	7	747	0,989
	7	8	397	0,531
	8	9	392	0,987
	9	5	386	0,985
5	1	4	1016	-
	2	3	216	0,213
	3	2	209	0,968
	4	6	194	0,928
	5	7	184	0,948
	6	1	178	0,967
	7	9	95	0,534
	8	8	85	0,895
	9	5	73	0,859
3	1	4	490	-
	2	2	72	0,147
	3	6	71	0,986
	4	1	68	0,958
	5	7	63	0,926
	6	3	59	0,937
	7	9	32	0,542
	8	8	28	0,875
	9	5	17	0,607
1	1	4	92	-
	2	1	13	0,141
	3	6	12	0,923
	4	2	9	0,750
	5	7	9	1,000
	6	8	5	0,556
	7	9	5	1,000
	8	3	4	0,800
	9	5	1	0,250

Źródło: opracowanie własne.

Tablica 4.19

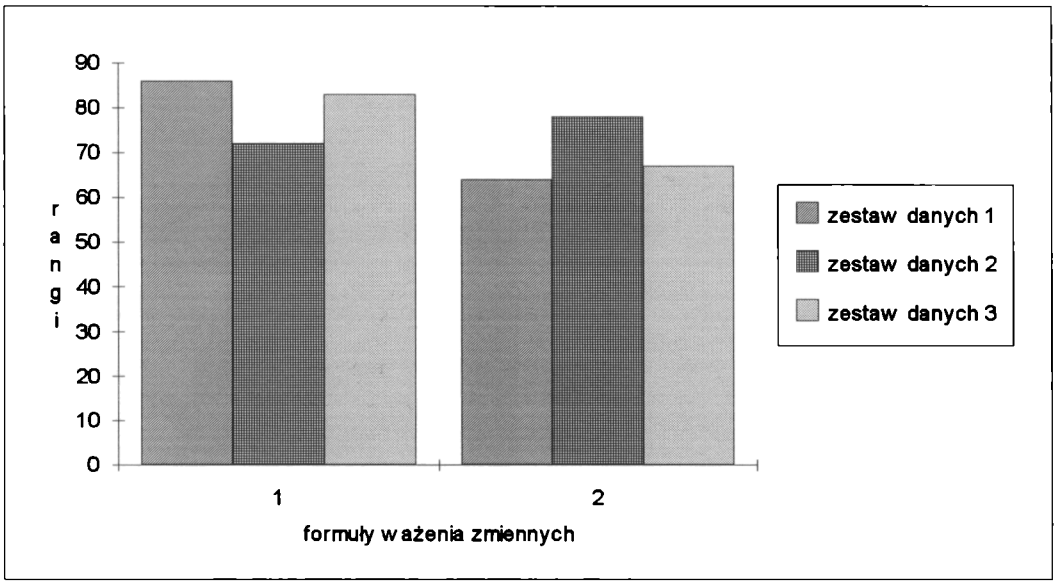
Uporządkowanie formuł agregacji zmiennych - zestaw danych nr 3

Parametr p	Pozycja	Agregacja	Ranga	r_i/r_{i-1}
306	1	2	702901	-
	2	1	687987	0,979
	3	4	672279	0,977
	4	5	657488	0,978
	5	7	543437	0,827
	6	6	522557	0,962
	7	9	511716	0,979
	8	8	494124	0,966
	9	17	459366	0,930
	10	10	338207	0,736
	11	12	250891	0,742
	12	15	241822	0,964
	13	11	236013	0,976
	14	3	202756	0,859
	15	16	195525	0,964
	16	13	168118	0,860
	17	14	160463	0,954
10	1	2	3855	-
	2	7	1407	0,365
	3	1	942	0,670
	4	5	848	0,900
	5	9	300	0,354
	6	4	263	0,877
	7	8	245	0,932
	8	17	177	0,722
	9	16	80	0,452
	10	6	63	0,788
	11	15	60	0,952
	12	3	10	0,167
	13	10	0	0,000
	14	11	0	0,000
	15	12	0	0,000
	16	13	0	0,000
	17	14	0	0,000
5	1	2	966	-
	2	7	609	0,630
	3	1	209	0,343
	4	5	192	0,919
	5	9	105	0,547
	6	8	59	0,562
	7	17	55	0,932
	8	16	30	0,545
	9	4	15	0,500
	10	15	10	0,667
	11	3	0	0,000
	12	6	0	0,000
	13	10	0	0,000
	14	11	0	0,000
	15	12	0	0,000
	16	13	0	0,000
	17	14	0	0,000

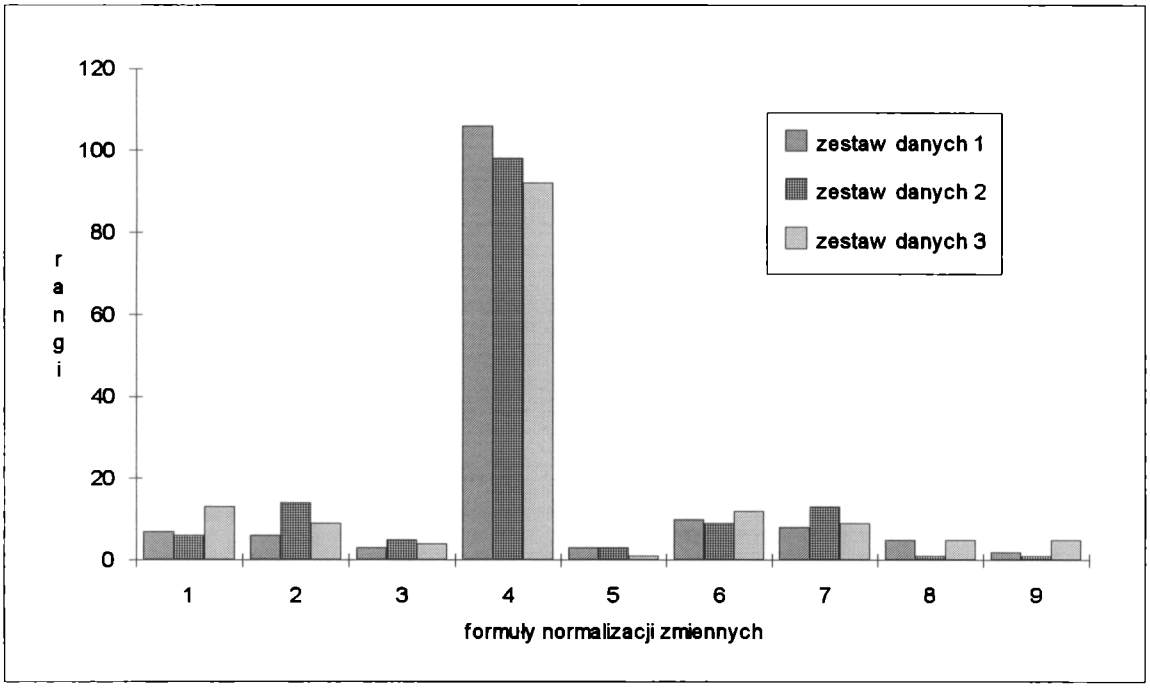
3	1	2	349	-
	2	7	323	0,926
	3	1	72	0,223
	4	5	63	0,875
	5	9	48	0,762
	6	8	19	0,396
	7	17	16	0,842
	8	16	10	0,625
	9	3	0	0,000
	10	4	0	0,000
	11	6	0	0,000
	12	10	0	0,000
	13	11	0	0,000
	14	12	0	0,000
	15	13	0	0,000
	16	14	0	0,000
	17	15	0	0,000
1	1	7	66	-
	2	2	49	0,742
	3	1	14	0,286
	4	5	9	0,643
	5	9	9	1,000
	6	17	3	0,333
	7	3	0	0,000
	8	4	0	0,000
	9	6	0	0,000
	10	8	0	0,000
	11	10	0	0,000
	12	11	0	0,000
	13	12	0	0,000
	14	13	0	0,000
	15	14	0	0,000
	16	15	0	0,000
	17	16	0	0,000

Źródło: opracowanie własne.

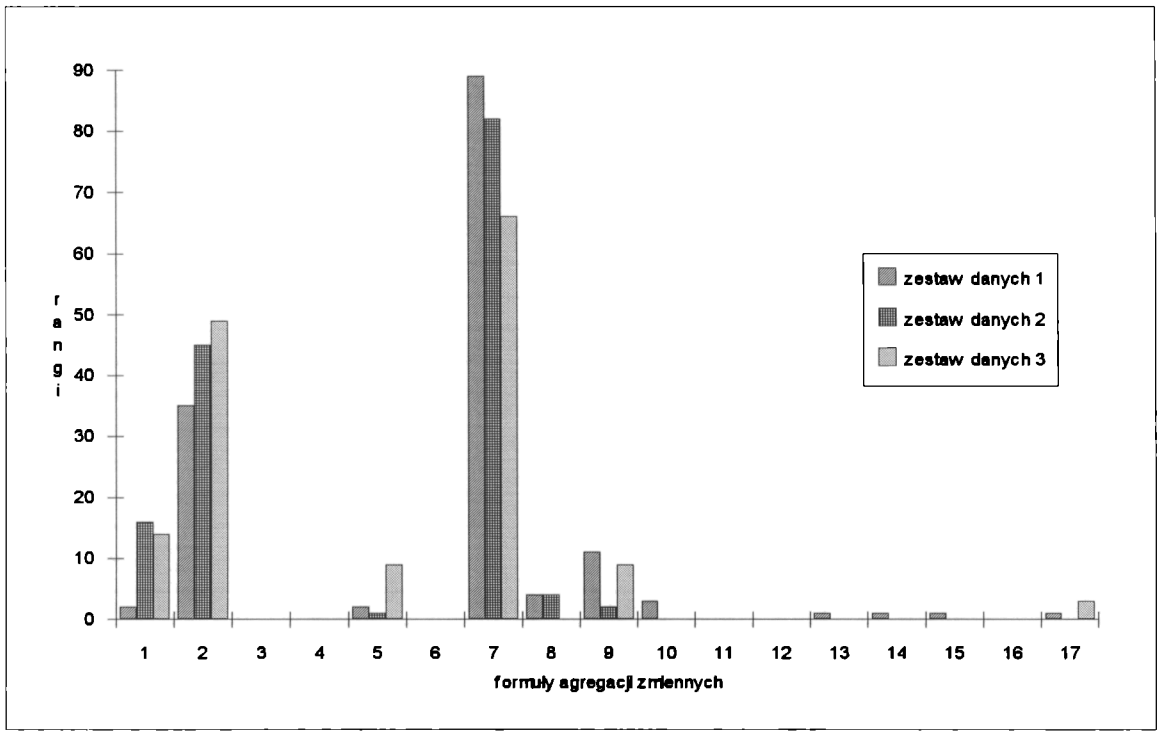
Na rysunkach 4.5-4.7 przedstawiono rangi poszczególnych elementów składowych procedury WAP w przekroju trzech zestawów danych i dla wartości parametru p=1. Histogramy te ilustrują zróżnicowanie rozpatrywanych formuł ważenia, normalizacji i agregacji zmiennych uzyskane na podstawie rang, jakie uzyskały one w przeprowadzonych eksperymentach symulacyjnych.



Rysunek 4.5. Rangi formuł ważenia zmiennych dla p=1



Rysunek 4.6. Rangi formuł normalizacji zmiennych dla p=1



Rysunek 4.7. Rangi formuł agregacji zmiennych dla p=1

4.3. Trend zbieżności wyników

W oparciu o dotychczas uzyskane i zaprezentowane wyżej wyniki przebiegów symulacyjnych można sformułować wniosek o ich widocznej zbieżności[2]. Upoważnia to postawienia tezy, iż „najlepsze" ścieżki procedury WAP są stabilne i odporne na charakter i strukturę przetwarzanych danych, natomiast konfiguracje „najgorsze" wykazują wyraźną wrażliwość i zależność od charakteru danych i z reguły przynoszą złe rezultaty w świetle stosowanych mierników jakości. Tablice A.20-A.22 zamieszczone w aneksie, zawierają uporządkowanie

[2] Pojęcie „zbieżności" jest tutaj używane w sensie podobieństwa ciągów uporządkowań uzyskiwanych dla różnych zestawów danych wejściowych i różnych wartości parametru p.

dziesięciu „najlepszych" konfiguracji po przetworzeniu każdego z 450 zbiorów danych. W większości przypadków układ ścieżek jest bardzo podobny.

Tablica 4.20

Układ ścieżek po przetworzeniu 150, 300 i 450 zbiorów

Pozycja	Liczba zbiorów					
	150		300		450	
	Nr ścieżki	Ranga	Nr ścieżki	Ranga	Nr ścieżki	Ranga
1	1,4,7	113	1,4,7	210	1,4,7	295
2	2,4,7	97	2,4,7	190	2,4,7	268
3	2,2,2	68	2,2,2	144	2,2,2	205
4	2,7,2	59	2,4,2	126	2,4,2	184
5	1,2,2	56	1,2,2	121	1,2,2	177
6	1,7,2	55	1,7,2	114	1,7,2	162
7	1,6,2	53	2,3,2	114	2,3,2	162
8	2,4,2	44	1,6,2	108	1,6,2	156
9	2,3,2	42	2,1,2	107	2,7,2	155
10	2,6,2	41	2,7,2	103	2,1,2	148
11	2,1,2	40	1,1,2	96	1,4,2	142
12	1,4,9	39	1,4,2	96	1,3,2	139
13	1,3,2	37	1,3,2	94	2,6,2	137
14	1,4,2	35	2,6,2	90	1,1,2	131
15	1,1,2	33	1,4,9	53	2,4,1	77
16	1,8,2	27	2,4,1	48	1,8,2	75
17	2,4,9	25	1,8,2	46	1,4,9	73
18	2,9,2	24	2,9,2	44	2,9,2	69
19	2,5,2	22	1,9,2	39	1,5,2	64
20	1,5,2	22	2,3,1	37	1,9,2	60
21	2,8,2	21	2,8,2	36	2,1,1	57
22	2,3,1	21	1,5,2	36	2,5,2	55
23	2,4,1	20	2,4,9	35	2,4,5	54
24	1,9,2	20	2,5,2	35	2,3,1	54
25	2,7,1	19	2,2,1	34	2,8,2	52
26	2,1,5	15	2,1,1	32	2,4,9	48
27	1,6,1	15	2,4,5	28	2,2,1	47
28	2,9,1	15	1,6,1	26	1,6,1	42
29	1,4,8	15	2,7,1	25	1,7,1	41
30	2,2,1	14	1,7,1	25	2,4,4	41
31	1,7,1	14	2,1,5	24	2,1,5	37
32	1,9,1	13	2,4,4	23	2,3,5	35
33	1,5,1	13	1,9,1	21	2,2,5	34
34	2,4,8	12	2,2,5	21	2,7,1	33
35	2,1,1	12	1,5,1	20	1,9,1	32
36	2,2,5	12	1,4,8	19	1,5,1	30
37	2,8,1	11	2,9,1	18	2,6,1	29
38	2,3,5	11	2,6,1	18	2,1,4	28
39	2,4,5	11	2,3,5	17	2,2,8	26
40	1,9,5	10	2,1,4	15	1,4,8	26
41	2,3,4	10	2,3,4	15	2,3,4	23
42	1,2,1	10	1,8,1	15	1,2,1	22
43	1,7,5	9	2,2,8	15	2,3,17	22
44	1,8,5	9	1,2,1	14	2,9,1	22
45	2,4,4	9	1,6,4	14	1,8,1	21
46	1,6,4	9	1,5,4	13	2,3,15	21
47	2,6,1	9	2,4,8	13	1,9,5	21

48	1,5,4	8	2,3,8	13	2,4,8	21
49	1,8,1	8	1,1,5	12	1,7,5	21
50	2,1,4	8	2,3,17	12	2,3,16	20
51	2,8,4	7	1,2,8	12	2,4,3	20
52	1,1,5	7	1,9,5	12	1,5,5	20
53	2,7,4	7	2,8,1	12	1,2,8	20
54	2,4,17	6	2,7,5	11	1,6,5	20
55	1,1,1	6	2,8,5	11	1,8,5	20
56	2,2,4	5	1,7,5	11	2,4,6	20
57	2,7,5	5	2,3,15	11	1,5,4	19
58	2,9,4	5	2,4,6	10	1,6,4	19
59	2,8,5	4	1,8,5	10	2,3,8	18
60	1,7,4	4	1,5,5	10	1,1,1	17
61	1,3,10	4	1,6,5	10	2,8,5	16
62	2,3,8	4	2,4,3	10	1,1,5	16
63	1,5,5	4	2,3,16	10	2,4,17	16
64	1,9,4	4	1,1,1	9	2,8,1	15
65	1,6,5	4	2,9,5	9	2,7,5	15
66	1,2,8	4	1,8,4	8	1,4,5	15
67	1,4,10	3	2,4,17	8	1,2,5	14
68	1,8,4	3	2,7,4	8	2,9,5	14
69	1,2,5	3	1,2,5	8	1,3,5	12
70	1,2,10	3	2,8,4	8	2,2,4	12
71	2,9,5	3	1,7,4	8	2,5,5	12
72	2,2,8	3	1,4,5	8	1,8,4	11
73	1,4,1	3	2,2,4	7	1,4,1	11
74	1,3,1	3	2,1,8	7	1,3,1	11
75	1,6,10	2	2,9,4	6	1,7,4	11
76	1,7,10	2	1,3,5	6	2,1,8	10
77	2,3,17	2	1,9,4	6	2,5,1	10
78	2,4,10	2	1,4,1	5	1,4,17	10
79	2,1,10	2	1,3,8	5	2,6,5	10
80	2,1,8	2	1,3,1	5	2,2,9	9
81	2,3,10	2	1,2,9	5	2,7,4	9
82	2,6,17	2	2,5,1	5	2,8,4	9
83	1,6,17	2	2,5,5	5	1,9,4	9
84	1,7,9	2	2,2,9	4	1,3,8	8
85	1,1,4	2	1,1,8	4	2,9,4	8
86	1,3,8	2	2,6,5	4	1,2,9	6
87	1,2,9	2	1,3,10	4	1,1,8	5
88	1,4,5	2	1,4,10	3	1,3,10	4
89	2,7,17	2	1,2,10	3	1,1,4	4
90	2,7,10	2	1,3,9	2	1,8,9	3
91	1,3,5	2	1,6,17	2	1,4,10	3
92	1,3,15	1	2,5,9	2	1,2,10	3
93	1,1,17	1	2,1,10	2	2,6,4	3
94	1,3,11	1	1,7,9	2	2,5,4	3
95	1,7,17	1	1,7,10	2	1,3,9	3
96	1,8,14	1	2,4,10	2	1,7,10	2
97	1,3,13	1	2,6,17	2	1,6,17	2
98	1,8,6	1	2,7,17	2	2,4,10	2
99	1,3,17	1	2,7,10	2	2,6,17	2
100	2,2,11	1	2,5,4	2	1,8,6	2
101	1,8,11	1	2,3,9	2	1,7,9	2
102	2,6,10	1	1,1,4	2	1,5,6	2
103	1,8,13	1	1,6,10	2	2,3,9	2
104	1,2,17	1	2,3,10	2	1,2,4	2
105	1,3,7	1	2,2,17	1	2,8,9	2

106	1,9,15	1	1,2,4	1	1,4,4	2
107	2,5,9	1	2,6,4	1	2,7,17	2
108	2,1,7	1	1,8,13	1	2,3,10	2
109	1,3,3	1	1,8,14	1	1,6,10	2
110	2,8,14	1	1,8,11	1	2,7,10	2
111	2,5,14	1	1,3,7	1	2,1,10	2
112	2,1,17	1	1,5,15	1	2,5,9	2
113	1,3,9	1	2,6,10	1	2,3,13	1
114	2,5,17	1	2,1,7	1	2,1,6	1
115	2,5,1	1	1,5,11	1	2,3,11	1
116	2,5,6	1	1,2,17	1	1,8,14	1
117	1,2,16	1	1,2,16	1	1,8,13	1
118	1,9,13	1	2,8,7	1	2,6,10	1
119	1,5,11	1	2,1,17	1	2,1,7	1
120	1,4,11	1	1,5,6	1	2,9,16	1
121	1,7,7	1	1,9,13	1	1,1,10	1
122	2,3,13	1	2,5,14	1	2,6,8	1
123	2,9,16	1	1,1,17	1	2,5,6	1
124	1,6,8	1	2,5,17	1	2,2,11	1
125	2,3,9	1	2,2,11	1	2,1,17	1
126	1,6,16	1	2,5,6	1	1,1,17	1
127	1,1,8	1	1,9,15	1	1,9,13	1
128	2,3,11	1	1,3,3	1	2,5,14	1
129	2,7,3	1	1,1,10	1	1,9,15	1
130	1,6,3	1	1,3,15	1	2,2,17	1
131	2,7,11	1	1,4,11	1	2,5,17	1
132	2,2,17	1	1,3,17	1	1,7,7	1
133	1,5,15	1	2,7,3	1	1,7,11	1
134	2,3,15	1	1,7,11	1	1,3,4	1
135	1,6,7	1	1,7,6	1	1,7,6	1
136	1,1,10	1	1,7,7	1	1,7,17	1
137	1,7,11	1	1,6,6	1	1,4,11	1
138	1,6,6	1	1,6,7	1	2,7,3	1
139	pozostałe	0	1,6,3	1	1,3,3	1
140			2,8,9	1	2,7,8	1
141			1,4,17	1	2,8,6	1
142			1,6,16	1	1,6,8	1
143			1,6,8	1	1,3,11	1
144			2,9,16	1	1,6,7	1
145			1,3,13	1	1,6,6	1
146			2,7,11	1	2,8,7	1
147			2,3,13	1	1,6,16	1
148			2,8,14	1	2,8,14	1
149			1,3,11	1	1,3,15	1
150			2,3,11	1	1,8,7	1
151			1,8,6	1	1,3,17	1
152			1,7,17	1	1,8,11	1
153			pozostałe	0	1,8,8	1
154					1,5,15	1
155					1,2,16	1
156					1,5,11	1
157					1,6,3	1
158					1,2,17	1
159					1,3,13	1
160					2,7,11	1
161					1,3,7	1
162					pozostałe	0

Źródło: opracowanie własne.

W tablicy 4.20 zestawiono ścieżki, które w wyniku rangowania narastającego przy p=10 uzyskały więcej niż 1 punkt. Analiza tych wyników wykazuje, że tendencja zbieżności ma charakter trwały i kolejne, przetwarzane zbiory nie powodują istotnych zmian w uporządkowaniu poszczególnych konfiguracji. Analizę zbieżności wyników prowadzono również w oparciu o mechanizm „samouczenia" przedstawiony za pomocą algorytmu 3.6, uzyskując zbliżone rezultaty. Z uwagi na obszerność tablic nie wszystkie wyniki zamieszczono w tekście niniejszej pracy.

4.4. Analiza wrażliwości

Analiza wrażliwości jest metodą badawczą dość powszechnie stosowaną w różnego rodzaju eksperymentach i badaniach empirycznych w takich dziedzinach wiedzy, jak np. analiza systemowa, programowanie liniowe, ekonometria. Pewne interpretacje wrażliwości i pojęć semantycznie podobnych przytoczono w rozdziale 1.

W analizie systemowej metoda ta stosowana jest w celu określenia poziomu stabilności danego systemu lub jego modelu matematycznego. Główna idea analizy wrażliwości polega tutaj na pomiarze stopnia zmian pewnych parametrów, opisujących wyjście systemu, pod wpływem modyfikacji innych parametrów, definiujących jego wejście (por. [128]). Stanem najbardziej pożądanym jest w tym przypadku określona stabilność układu (modelu), tzn. mała wrażliwość parametrów wyjścia na zmiany wprowadzane parametrami wejścia (chodzi przy tym zazwyczaj o niewielkie zaburzenia danych wejściowych).

Analiza wrażliwości znajduje również zastosowanie w programowaniu liniowym i innych badaniach optymalizacyjnych. Głównym przedmiotem badań są w tym przypadku zmiany obserwowane w rozwiązaniu optymalnym implikowane zaburzeniami wprowadzanymi do zbioru danych wejściowych (argumentów funkcji celu). Istotnym rezultatem badań jest określenie obszaru zmienności danych, dla których pewne rozwiązanie pozostaje ciągle optymalne (por. [56]).

Na gruncie ekonometrii analiza wrażliwości stosowana jest najczęściej w badaniach symulacyjnych, prowadzonych w oparciu o zdefiniowany model ekonometryczny. Postępowanie badawcze zmierza wówczas do oszacowania poziomu zmian określonych wielkości ekonomicznych na skutek zmian wartości parametrów modelu (por. [131]).

Metodę analizy wrażliwości wykorzystano w niniejszej pracy w celu zbadania stopnia zróżnicowania w ciągach uporządkowań obiektów na skutek zmian w wejściowych zbiorach danych statystycznych. Wygenerowany losowo dla danego wektora średnich i zadanej macierzy wariancji-kowariancji, zbiór danych poddano następującym modyfikacjom (por. [65]):

1) ze zbioru n obiektów opisanych realizacjami m zmiennych diagnostycznych usunięto wybrany losowo i-ty obiekt,

2) do zbioru n obiektów dodano obiekt n+1 o realizacjach losowo wybranego i-tego obiektu pierwotnego,

3) do zbioru m zmiennych diagnostycznych dodano zmienną m+1 o realizacjach, będących kombinacją liniową cech pierwotnych,

4) przeprowadzono ortogonalizację zmiennych diagnostycznych, wykorzystując w tym celu metodę Grama-Schmidta[3],

5) wprowadzono niewielkie, losowe zaburzenia w realizacjach zmiennych diagnostycznych.

[3] Algorytm ortogonalizacji macierzy metodą Grama-Schmidta opracowano na podstawie prac [99] i [114].

Pierwotny zbiór danych oraz każdy z pięciu zbiorów zmodyfikowanych poddano procedurze przetwarzania w układzie każdej z 306 ścieżek WAP. Uzyskane ciągi uporządkowań porównywano parami według schematu: uporządkowanie zbioru pierwotnego według ścieżki 1 i uporządkowanie zbioru zmodyfikowanego nr 1 według ścieżki 1, ..., uporządkowanie zbioru pierwotnego według ścieżki 306 i uporządkowanie zbioru zmodyfikowanego nr 1 według ścieżki 306, ..., uporządkowanie zbioru pierwotnego według ścieżki 1 i uporządkowanie zbioru zmodyfikowanego nr 5 według ścieżki 1, ..., uporządkowanie zbioru pierwotnego według ścieżki 306 i uporządkowanie zbioru zmodyfikowanego nr 5 według ścieżki 306. Poziom zgodności jednoimiennych i równolicznych ciągów uporządkowań mierzono współczynnikiem τ korelacji rang Kendalla danym wzorem [45]:

$$\tau = \sum_{i=1}^{n} \sum_{j=i+1}^{n-1} \xi\left(a_i, a_j, b_i, b_j\right) \tag{4.1}$$

gdzie: $\xi\left(a_i, a_j, b_i, b_j\right) = \begin{cases} 1 & \text{jeżeli } \left(a_i - a_j\right)\left(b_i - b_j\right) > 0 \\ 0 & \text{jeżeli } \left(a_i - a_j\right)\left(b_i - b_j\right) = 0 \\ -1 & \text{jeżeli } \left(a_i - a_j\right)\left(b_i - b_j\right) < 0 \end{cases}$,

a_i, a_j, b_i, b_j - elementy ciągów uporządkowań $\{a_i\}$ oraz $\{b_i\}$ $(1 \le i \le j \le n)$,
n - liczba elementów.

W przypadku modyfikacji (1) i (2) zmieniających liczebność zbiorów obiektów uwzględniano podciągi złożone z elementów występujących w obu uporządkowaniach danej pary.

Wszystkie otrzymane współczynniki korelacji rang Kendalla między ciągami uporządkowań obiektów zbioru pierwotnego oraz obiektów zbiorów zmodyfikowanych zawiera tablica A.23 zamieszczona w aneksie. W tablicy 4.21 przedstawiono natomiast współczynniki Kendalla dotyczące 10 „najlepszych" ścieżek WAP w świetle wyników zaprezentowanych w punkcie 4.2.

Z tablicy 4.21 wynika, że do najmniej wrażliwych na zburzenia danych ścieżek WAP z dziesiątki „najlepszych" w świetle mierników jakości, należą procedury {1,4,7} i {2,4,7}. Na

podstawie danych tejże tablicy można również sformułować wniosek, iż zwiększenie liczby obiektów oraz zwiększenie liczby zmiennych istotnie wpływa na uporządkowanie obiektów. Świadczą o tym wartości współczynników korelacji rang Kendalla dla zmodyfikowanych zbiorów nr 2 i 3. Natomiast zmniejszenie liczby obiektów nie wywołuje na ogół żadnych przesunięć lub powoduje przestawienie obiektów sąsiadujących ze sobą.

Tablica 4.21

Współczynniki korelacji rang Kendalla (10 „najlepszych" ścieżek WAP)

Ścieżka	Współczynniki korelacji rang Kendalla				
	Modyfikacja 1	Modyfikacja 2	Modyfikacja 3	Modyfikacja 4	Modyfikacja 5
1,2,2	1,000	-0,200	-0,200	0,422	0,733
1,4,7	1,000	0,600	0,600	0,422	0,956
1,4,9	1,000	-0,200	-0,200	0,156	0,733
1,6,2	1,000	-0,200	-0,200	0,867	0,778
1,7,2	1,000	-0,200	-0,200	0,867	0,733
2,2,2	1,000	-0,111	-0,111	0,244	0,778
2,4,2	1,000	-0,111	-0,111	0,244	0,778
2,4,7	0,944	0,644	0,644	0,378	0,956
2,6,2	1,000	-0,111	-0,111	0,333	0,778
2,7,2	1,000	-0,111	-0,111	0,333	0,778

Źródło: opracowanie własne.

Tablica 4.22

Najwyższe współczynniki korelacji rang Kendalla

Współczynniki korelacji rang Kendalla									
Modyfikacja 1		Modyfikacja 2		Modyfikacja 3		Modyfikacja 4		Modyfikacja 5	
Ścieżka	τ	Ścieżka	τ	Ścieżka	τ	Ścieżka	τ	Ścieżka	τ
2,5,7	1,000	1,4,5	0,867	1,4,5	0,867	1,6,8	1,000	1,5,13	1,000
2,2,17	1,000	1,9,5	0,867	1,9,5	0,867	1,7,8	0,956	1,3,12	1,000
2,6,13	1,000	1,1,5	0,867	1,1,5	0,867	1,6,15	0,956	1,5,2	1,000
1,1,8	1,000	2,7,5	0,867	2,7,5	0,867	1,6,3	0,956	1,4,12	1,000
2,4,16	1,000	2,9,5	0,867	2,9,5	0,867	1,7,3	0,956	1,2,12	1,000
1,6,8	1,000	1,7,5	0,867	1,7,5	0,867	1,6,6	0,911	2,8,5	1,000
1,7,6	1,000	1,8,5	0,867	1,8,5	0,867	1,7,15	0,911	1,5,12	1,000
1,2,3	1,000	1,3,5	0,867	1,3,5	0,867	1,6,13	0,911	1,1,12	1,000
2,4,8	1,000	2,8,5	0,867	2,8,5	0,867	1,7,2	0,867	1,6,12	1,000

Źródło: opracowanie własne.

W tablicy 4.22 zamieszczono najwyższe współczynniki korelacji rang Kendalla obliczone między ciągiem pierwotnym a ciągami uporządkowań poszczególnych zbiorów zmodyfikowanych. Informacje zawarte w tej tablicy wskazują na brak istotnych zależności między jakością poszczególnych ścieżek WAP (ocenianą zestawem mierników przedstawionych w tablicy 1.4)

a ich wrażliwością na zaburzenia i modyfikacje danych wejściowych. W kolejnych dziesiątkach najbardziej stabilnych, względem zaburzeń danych, procedur WAP tylko raz wystąpiła ścieżka (oznaczona trójką {1,7,2}) należąca do dziesiątki „najlepszych" procedur w ocenie przeprowadzonej przy pomocy mierników jakości.

Niemniej jednak należy podkreślić, że procedury WAP najwyżej ocenione w świetle kryteriów jakości nie wykazują drastycznej niestabilności w ocenie prowadzonej przy zastosowaniu analizy wrażliwości.

Rozdział 5

SYMULACYJNY PROGRAM KOMPUTEROWY

5.1. Narzędzia i techniki programowania

Do tworzenia programów komputerowych używa się różnego rodzaju narzędzi, metod i technik, kierując się w ich wyborze przede wszystkim charakterem zadania programistycznego. Środki te na przestrzeni ostatnich kilkudziesięciu lat ulegały dość istotnym przeobrażeniom. Język programowania używany jest w końcowej fazie opracowywania programu komputerowego i służy do zakodowania wcześniej opracowanego algorytmu. W celu kompleksowego opracowania złożonego zadania stosuje się odpowiednie metody konstruowania programów oraz techniki programowania. W literaturze dotyczącej metodologii programowania można doszukać się pewnej niekonsekwencji terminologicznej, wyrażającej się w nieprecyzyjnym używaniu pojęć „metody" i „techniki". Z etymologicznego punktu widzenia uzasadnione jest używanie terminu „metoda" w znaczeniu szerszym, dla określenia pewnych ogólnych zasad postępowania dla osiągnięcia postawionego celu, zaś pojęcia „technika" w znaczeniu węższym, dla określenia pewnych środków technicznych, którymi należy się posłużyć, aby zrealizować postawione zadanie. W tym kontekście za podstawowe metody konstruowania programów (algorytmów) należy uznać metodę analityczną (zstępującą) oraz metodę syntetyczną (wstępującą). Natomiast wśród znanych technik programowania wyróżnić można

programowanie sekwencyjne, programowanie strukturalne oraz programowanie zorientowane obiektowo ([36], [44], [121], [170]).

Do najważniejszych cech programowania sekwencyjnego zaliczyć należy:

- konieczność deklarowania zmiennych globalnych, a więc dostępnych w każdym miejscu programu,
- konieczność powielania tych samych bloków instrukcji w różnych miejscach programu,
- obecność instrukcji skoku, której nadużywanie może powodować nieczytelność i niezrozumiałość programu,
- konieczność zapisywania całego programu źródłowego w jednym pliku dyskowym.

Do najważniejszych cech programowania strukturalnego zalicza się:

- możliwość definiowania podprogramów (procedur i funkcji) i wywoływania ich w różnych miejscach programu głównego,
- możliwość deklarowania zmiennych lokalnych, dostępnych w określonych podprogramach (procedurach, funkcjach, modułach),
- możliwość zapisywania różnych modułów programu w osobnych plikach dyskowych,
- możliwość niezależnego testowania i kompilacji zwartych modułów programu głównego,
- możliwość dynamicznego zarządzania pamięcią operacyjną,
- możliwość pełnego korzystania z zestawu konstrukcji strukturalnych,
- możliwość pisania programów przejrzystych i zrozumiałych, takich, które można czytać „z góry na dół".

Do najważniejszych cech programowania obiektowego zaliczyć należy:

- ścisły związek programu z danymi (enkapsulacja),
- nowy standardowy typ danych zwany klasą, złożony z pól (określają one właściwości obiektu danej klasy) oraz metod (funkcji składowych), opisujących czynności wykonywane na obiektach danej klasy,

- możliwość definiowania klas potomnych (pochodnych), które mogą dziedziczyć określone pola i funkcje składowe klasy macierzystej (bazowej), przy czym dostępne jest dziedziczenie jedno- i wielobazowe,
- możliwość przesłaniania w klasach pochodnych pól i funkcji składowych klasy bazowej oraz mechanizm przeciążania operatorów,
- mechanizm polimorfizmu, umożliwiający umieszczanie w jednej klasie wielu definicji tej samej funkcji składowej lub redefiniowania w klasach potomnych funkcji deklarowanych w klasie bazowej (implementacja funkcji wirtualnych).

Przedstawione skrótowo powyżej elementarne atrybuty podstawowych technik programowania wskazują na istotne zmiany w sposobach kodowania algorytmów opisujących różne fragmenty rzeczywistości. Jest to istotne z punktu widzenia efektywności programowania, ocenianej przez pryzmat czasu potrzebnego na przygotowanie i testowanie programu, jego niezawodność, sprawność, łatwość konserwacji i rozbudowy itd.

Opracowując program symulacyjny, potrzebny do realizacji celu niniejszej pracy, kierowano się powyższymi przesłankami, mając w szczególności na uwadze takie zaprojektowanie programu, aby jego rozbudowa i uzupełnianie w nowe procedury WAP nie nastręczały istotnych trudności. W procesie projektowania algorytmów WAP korzystano zarówno z metody analitycznej, jak i syntetycznej, natomiast w fazie programowania wykorzystano technikę obiektową z uwagi na jej bezsporną przewagę nad pozostałymi.

Do przygotowania programów komputerowych wykorzystano dwa pakiety narzędziowe:

1) Borland C++ v. 4.0, zawierający najbardziej aktualną i pełną implementację standardu języków C i C++ amerykańskiej firmy Borland International Inc.,

2) CodeBase++ v. 5.1, zawierający najnowszą wersję obiektowej biblioteki do zarządzania bazami danych w języku C++ kanadyjskiej firmy Sequiter Software Inc.

Wybór języka C++ do kodowania algorytmów WAP podyktowany był przede wszystkim faktem, że zaimplementowano w nim wszystkie podstawowe mechanizmy i konstrukcje

umożliwiające efektywne programowanie obiektowe. Język C++ jest, jak wiadomo, nadzbiorem języka C, powszechnie uznanego i cenionego za niezwykłą zwartość, przejrzystość, logikę i efektywność. Dlatego też język C++ łączy w sobie w sposób naturalny siłę języka C z filozofią programowania obiektowego, oferującą nieznane dotychczas mechanizmy komputerowej reprezentacji obiektów świata rzeczywistego. Warto również wspomnieć i o tym, że geneza języka C++ wiąże się bezpośrednio zarówno z zaletami, jak i niedostatkami języka Simula 67, przeznaczonego w szczególności do programowania symulacyjnego (por. [10], [82], [129], [161]).

Bibliotekę CodeBase++ wykorzystano przede wszystkim do zarządzania plikami dyskowymi w standardzie XBase (dBase III Plus, dBase IV, FoxPro, Clipper) z poziomu języka C/C++ [37], [38]. Bazy danych typu XBase (pliki DBF - DataBase File) są wygodnym i efektywnym formatem gromadzenia i przechowywania dużych zbiorów danych, w tym również danych liczbowych. Są one poza tym powszechnie uznanym standardem, akceptowanym przez ogromną większość innych programów komercyjnych (takich jak np. bazy danych, arkusze kalkulacyjne, programy do obliczeń statystycznych, edytory tekstu), co jest bardzo ważne i atrakcyjne z punktu widzenia wieloetapowego przetwarzania danych.

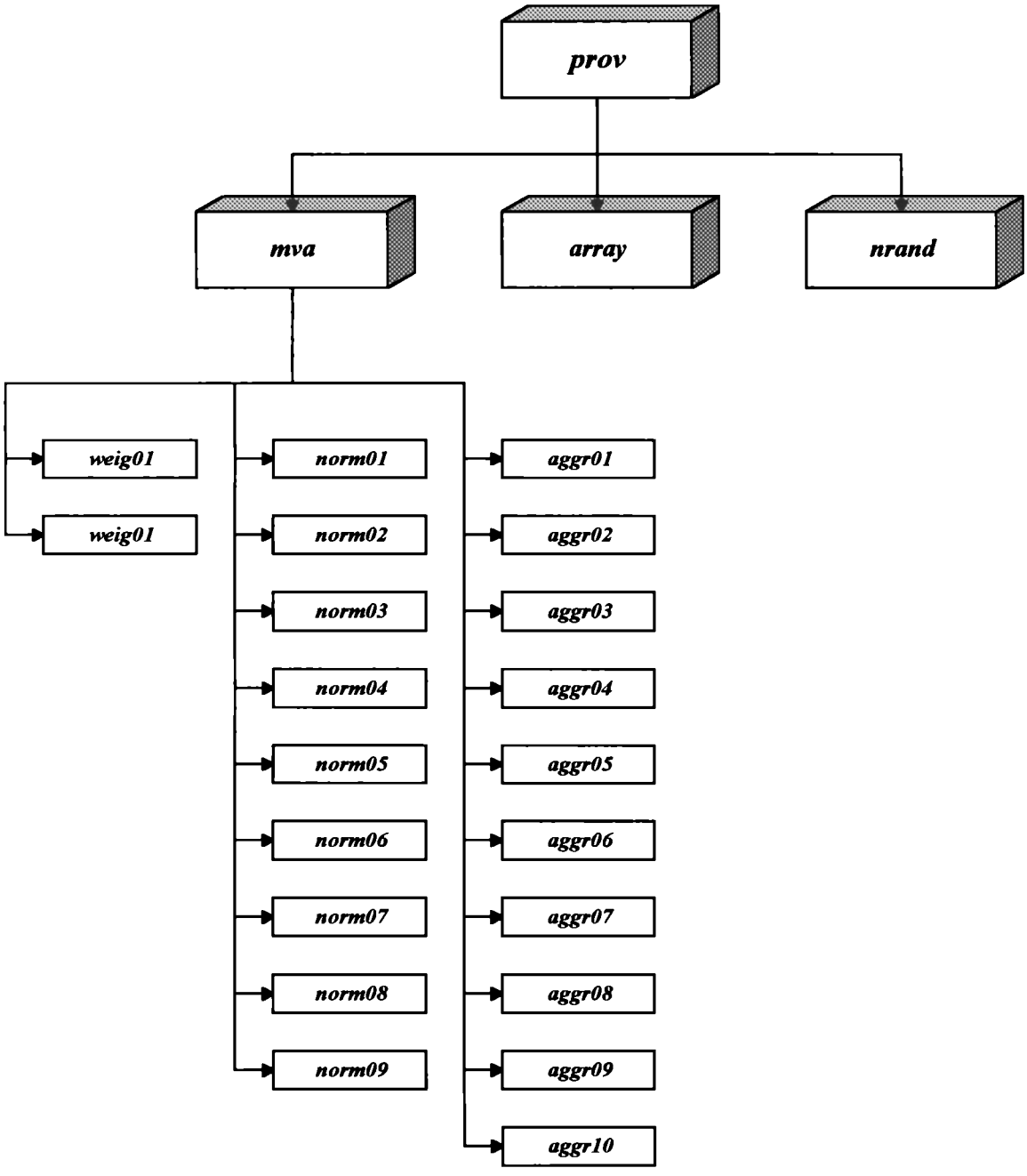
5.2. Struktura programu

Symulacyjny program komputerowy zaprojektowano i zaprogramowano w taki sposób, aby możliwe było przetwarzanie dużej liczby zbiorów danych liczbowych zgodnie z algorytmem przedstawionym na rysunku 3.1 w rozdziale 3.

Konstrukcję programu symulacyjnego oparto na hierarchii klas, co ilustruje rysunek 5.1. Postulat otwartej architektury programu, wyrażający się przede wszystkim w możliwości dołączania nowych procedur WAP, zrealizowano przy wykorzystaniu podstawowych mechanizmów programowania obiektowego, tzn. techniki dziedziczenia oraz zjawiska polimorfizmu (por. [8], [24], [25], [26], [110], [185]). W klasie *mva* zadeklarowano funkcje wirtualne *weight*, *normal* i *aggreg* używane do wyznaczania współczynników wagowych, normalizacji zmiennych oraz obliczania zmiennych syntetycznych według formuł przedstawionych w tablicach 1.1-1.3 w rozdziale 1. Funkcje te następnie redefiniowano w klasach pochodnych, uzyskując pożądane procedury i zachowując możliwość dodawania nowych. W programie głównym, w szczególności w jego części przetwarzającej ścieżki WAP, wykorzystano siłę wskaźników języków C/C++ oraz zjawisko dynamicznego (późnego) wiązania funkcji, przejawiające się w tym, iż właściwa funkcja składowa, aktywowana na rzecz obiektu, wybierana jest nie w fazie kompilacji programu, lecz podczas jego wykonywania. Dopasowanie funkcji odbywa się wówczas na podstawie aktualnego typu obiektu. W celu wybrania właściwego obiektu najwygodniej jest posłużyć się wskaźnikiem do klasy bazowej, ustawiając tak, aby wskazywał na określony obiekt klasy pochodnej[1]. Przykład programu realizującego przedstawiony schemat zamieszczono w aneksie. Wszystkie działania na tablicach typu *double* (jedno- i wielowymiarowych) wykonywane są przy użyciu wskaźników, w oparciu o funkcje

[1] Jak wiadomo, w języku C++, na skutek automatycznej konwersji typów, możliwe jest przypisanie wskaźnikowi do obiektu klasy bazowej adresu obiektu klasy potomnej [73], [110], [161].

składowe klasy *array*. Do operacji wskaźnikowych na tablicach wielowymiarowych wykorzystano wyrażenia indeksujące omówione szczegółowo w pracy [166].



Rysunek 5.1. Hierarchia klas programu symulacyjnego

Poniżej przedstawione zostaną krótkie charakterystyki poszczególnych klas i ważniejszych funkcji składowych. Szczegóły referencyjne i implementacyjne zawarte są natomiast w

odpowiednich plikach nagłówkowych i programowych, których treść zamieszczono w aneksie do pracy.

Przeznaczenie i podstawowe funkcje składowe poszczególnych klas.

1. Klasa *prov* jest bazową klasą wszystkich pozostałych. Wyposażona jest w standardowy konstruktor i wirtualny destruktor, redefiniowany w klasie *array*. Zawiera funkcje o charakterze globalnym, takie jak:

- *loc* - przekształca dwuwymiarowy adres elementu macierzy na jednowymiarowy adres elementu wektora [166],
- *loc2* - przekształca dwuwymiarowy adres indeksowy elementu macierzy na jednowymiarowy adres wskaźnikowy elementu wektora [166],
- *length* - oblicza długość łańcucha znakowego,
- *substr* - wycina fragment z łańcucha znaków,
- *id_str* - tworzy unikalny identyfikator w postaci łańcucha znaków,
- *round* - zaokrągla liczbę rzeczywistą typu *double* z zadaną precyzją.

2. Klasa *mva* jest klasą pochodną klasy *prov*. Wyposażona jest w dwa konstruktory (bezparametrowy i sparametryzowany) oraz standardowy destruktor. Zawiera funkcje składowe implementujące podstawowe algorytmy WAP, funkcje wykonujące operacje na tablicach, obsługujące pliki dyskowe itp. Do najważniejszych funkcji tej klasy należą:

- *xcreate* - tworzy pliki DBF,
- *screate* - tworzy pliki DBF według zadanej struktury,
- *tcreate* - tworzy pliki DBF oraz plik indeksowy,
- *ccreate* - tworzy pliki DBF dla trójwymiarowej tablicy danych,
- *use_dbf* - otwiera plik DBF,
- *use_idx* - otwiera plik indeksowy CDX,
- *repl_s* - zapisuje w polu rekordu bieżącego łańcuch znaków,
- *get_s* - pobiera zawartość pola i zwraca w postaci łańcucha znaków,

- *put_i* - zapisuje w polu rekordu bieżącego wartość numeryczną typu *int*,
- *get_i* - pobiera z pola rekordu bieżącego wartość numeryczną typu *int*,
- *put_d* - zapisuje w polu pliku wartość numeryczną typu *double*,
- *get_d* - pobiera z pola pliku wartość numeryczną typu *double*,
- *dbf2mat* - przepisuje zawartość pliku DBF do macierzy,
- *dbf2arr* - przepisuje zawartość pliku DBF do tablicy trójwymiarowej,
- *dbf2vec* - przepisuje zawartość dolnego trójkąta pliku DBF do wektora,
- *mat2dbf* - zapisuje zawartość macierzy w pliku DBF,
- *vec2dbf* - zapisuje zawartość wektora w bieżącym rekordzie pliku DBF,
- *tr2vec* - przepisuje dolny trójkąt macierzy symetrycznej do wektora,
- *qsort_up* - sortuje niemalejąco tablicę liczb typu *double* według algorytmu *quicksort*,
- *qsort_dn* - sortuje nierosnąco tablicę liczb typu *double* według algorytmu *quicksort*,
- *rep_v* - zapisuje skalar *d* na elementach wektora,
- *diag* - zapisuje skalar *d* na głównej przekątnej oraz liczbę *x* poza główną przekątną macierzy *S*,
- *vdiag* - zapisuje wektor *D* na głównej przekątnej oraz liczbę *x* poza główną przekątną macierzy *S*,
- *transp* - wykonuje transpozycję macierzy,
- *mult* - oblicza iloczyn macierzy *C=A*B*,
- *norm_cor* - oblicza normę macierzy korelacji,
- *norm_frob* - oblicza normę Frobeniusa macierzy *S* [55],
- *max_v* - znajduje największą wartość w wektorze,
- *max_m* - znajduje największą wartość w macierzy,
- *max_x* - znajduje największą wartość w każdej z kolumn macierzy *S*,
- *min_v* - znajduje najmniejszą wartość w wektorze,
- *min_m* - znajduje najmniejszą wartość w macierzy,

- *min_x* - znajduje najmniejszą wartość w każdej z kolumn macierzy *S*,
- *mean_v* - oblicza średnią arytmetyczną z elementów wektora,
- *mean_m* - oblicza średnią arytmetyczną z elementów macierzy,
- *mean_x* - oblicza wektor średnich arytmetycznych z macierzy *S*,
- *stdev_v* - oblicza odchylenie standardowe z elementów wektora,
- *stdev_m* - oblicza odchylenie standardowe z elementów macierzy,
- *stdev_x* - oblicza wektor odchyleń standardowych z macierzy *S*,
- *sum_v* - oblicza sumę elementów wektora,
- *sum_m* - oblicza sumę elementów macierzy,
- *sum_x* - oblicza wektor sum z macierzy *S* (sumy kolumn),
- *med_x* - oblicza wektor median z macierzy (mediany kolumn),
- *asym_coef* - oblicza wektor współczynników asymetrii z macierzy,
- *conc_coef* - oblicza wektor współczynników koncentracji z macierzy,
- *var_coef* - oblicza współczynniki zmienności kolumn macierzy oraz sumę tych współczynników,
- *corr* - oblicza macierz współczynników korelacji liniowej Pearsona,
- *corr_xq* - oblicza współczynniki korelacji liniowej pomiędzy zmienną syntetyczną oraz *j*-tą zmienną diagnostyczną,
- *spearman* - oblicza współczynniki korelacji rang Spearmana pomiędzy zmienną syntetyczną oraz *j*-tą zmienną diagnostyczną,
- *kendall* - oblicza współczynniki korelacji rang Kendalla (tau) pomiędzy wektorami uporządkowań,
- *varcov* - oblicza macierz wariancji-kowariancji,
- *reyleigh* - wyznacza wartość własną i wektor własny macierzy symetrycznej metodą Reylaigh'a [176],

- *hotelling* - wyznacza wszystkie wartości własne oraz wektory własne macierzy symetrycznej metodą Hotellinga [176],
- *loadings* - oblicza ładunki czynnikowe,
- *characters* - identyfikuje charakter zmiennych,
- *dest2stym* - przekształca cechy destymulanty na cechy stymulanty,
- *neg2pos* - skaluje znormalizowaną macierz danych,
- *upperpole* - wyznacza górny biegun macierzy danych znormalizowanych,
- *lowerpole* - wyznacza dolny biegun macierzy danych znormalizowanych,
- *det_eigen* - oblicza wyznacznik macierzy korelacji na podstawie wektora wartości własnych tej macierzy,
- *gauge01, gauge02, gauge03* - mierniki zgodności odwzorowania,
- *gauge04, gauge05* - mierniki oparte na współczynniku korelacji liniowej Pearsona między zmienną syntetyczną oraz zmiennymi diagnostycznymi,
- *gauge06, gauge07, gauge08* - mierniki oparte na współczynniku korelacji rang Spearmana między zmienną syntetyczną oraz zmiennymi diagnostycznymi,
- *gauge09*, *gauge10* - miernik zmienności zmiennej syntetycznej,
- *gauge11*, *gauge12* - miernik oparty na odległości taksonomicznej między standaryzowaną zmienną syntetyczną oraz standaryzowanymi zmiennymi diagnostycznymi,
- *dist_m01* - odległość Euklidesa,
- *dist_v01* - odległość Euklidesa między elementami wektora,
- *scale_q01*, *scale_q02*, *scale_q03*, *scale_q04* - skalowanie wektora zmiennej syntetycznej,
- *mlu_decomp* - rozkłada macierz S na czynniki trójkątne LU według algorytmu Gaussa-Banachiewicza z wyborem elementu głównego [99],
- *lu_decomp* - rozkłada macierz S na czynniki trójkątne LU według algorytmu Gaussa-Banachiewicza bez wyboru elementu głównego [99],

- *mdet* - oblicza wyznacznik macierzy S na podstawie rozkładu trójkątnego LU otrzymanego z wyborem elementu głównego [99],
- *det* - oblicza wyznacznik macierzy S na podstawie rozkładu trójkątnego LU otrzymanego bez wyboru elementu głównego [99],
- *minverse* - oblicza macierz odwrotną na podstawie rozkładu trójkątnego LU wyznaczonego z wyborem elementu głównego [99],
- *inverse* - oblicza macierz odwrotną na podstawie rozkładu trójkątnego LU wyznaczonego bez wyboru elementu głównego [99],

Klasa *mva* zawiera ponadto funkcje wirtualne, zdefiniowane w klasach pochodnych:

- *weight*,
- *normal*,
- *aggreg*.

3. Klasa *array* jest klasą pochodną klasy *prov*. Wyposażona jest w dwa konstruktory (bezparametrowy i sparametryzowany) oraz niestandardowy destruktor odzyskujący dynamicznie pamięć przydzielaną operatorem *new* dla tablic jedno-, dwu- i trójwymiarowych typu *double*. Zawiera funkcje składowe tworzące w zapasie pamięci tablice typu *double* oraz indeksujące elementy tablic dwu- i trójwymiarowych. Do najważniejszych funkcji tej klasy należą:

- *alloc* - przydziela pamięć operacyjną dla tworzonej tablicy,
- *loc3* - przekształca trójwymiarowy adres indeksowy elementu tablicy na jednowymiarowy adres wskaźnikowy elementu wektora [166],
- *cub2v* - wybiera z trójwymiarowej tablicy danych wszystkie obserwacje dla j-tej cechy,
- *cub2o* - wybiera z trójwymiarowej tablicy danych wszystkie obserwacje dla i-tego obiektu,
- *cub2t* - wybiera z trójwymiarowej tablicy danych wszystkie obserwacje dla k-tego okresu,

4. Klasa *nrand* jest klasą pochodną klasy *prov*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcje składowe dostarczające liczb losowych. Do najważniejszych funkcji tej klasy należą:

- *rget* - produkuje liczbę losową metodą kongruencji liniowej,
- *rseed* - inicjuje generator zarodkiem *s*,
- *ilot* - generuje liczbę losową typu *long int* z przedziału *[0, LONG_MAX]*,
- *dlots* - generuje liczby losowe typu *double* z przedziału *[0, 1]*,
- *flots* - generuje liczby losowe typu *float* z przedziału *[0,1]*,
- *ablots* - generuje liczby losowe typu *int* z przedziału *[a, b]*,
- *dablots* - generuje liczby losowe typu *double* z przedziału *[a, b]*,
- *nrnd* - generuje liczby losowe o rozkładzie normalnym z przedziału *[0,1]*,
- *trnd* - generuje liczby losowe o rozkładzie normalnym z przedziału *[0,1]* metodą Teichroewa [167],
- *dxrnd* - generuje liczby losowe typu *double* o rozkładzie normalnym dla danej wartości oczekiwanej *ex* i danego odchylenia standardowego *stdx*,
- *fxrnd* - generuje liczby losowe typu *float* o rozkładzie normalnym dla danej wartości oczekiwanej *ex* i danego odchylenia standardowego *stdx*,
- *rm_random* - generator rozkładu równomiernego,
- *rm_seed* - produkuje zarodek dla generatora *rm_random*,
- *rm_ini, rm_start* - funkcje inicjujące generator *rm_random*,
- *rm_rnd* - generuje jedną liczbę losową dla generatora *rm_random*,
- *mvnd* - generuje wektory losowe o wielowymiarowym rozkładzie normalnym przy zadanym wektorze wartości oczekiwanych i danej macierzy wariancji-kowariancji metodą pierwiastków kwadratowych [123],
- *ex_rnd* - generuje wektor średnich,
- *vc_rnd* - generuje macierz wariancji-kowariancji.

5. Klasa *weig01* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową obliczającą wagi zmiennych.

- *weight* - wyznaczenie współczynników wagowych według formuły nr 1 z tablicy 1.1.

6. Klasa *weig02* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową obliczającą wagi zmiennych.

- *weight* - wyznaczenie współczynników wagowych według formuły nr 2 z tablicy 1.1.

7. Klasa *normal01* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

- *normal* - normalizacja zmiennych według formuły nr 1 z tablicy 1.2.

8. Klasa *normal02* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

- *normal* - normalizacja zmiennych według formuły nr 2 z tablicy 1.2.

9. Klasa *normal03* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

- *normal* - normalizacja zmiennych według formuły nr 3 z tablicy 1.2.

10. Klasa *normal04* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

- *normal* - normalizacja zmiennych według formuły nr 4 z tablicy 1.2.

11. Klasa *normal05* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

- *normal* - normalizacja zmiennych według formuły nr 5 z tablicy 1.2.

12. Klasa *normal06* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

- *normal* - normalizacja zmiennych według formuły nr 6 z tablicy 1.2.

13. Klasa *normal07* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

• *normal* - normalizacja zmiennych według formuły nr 7 z tablicy 1.2.

14. Klasa *normal08* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

• *normal* - normalizacja zmiennych według formuły nr 8 z tablicy 1.2.

15. Klasa *normal09* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową normalizującą macierz danych.

• *normal* - normalizacja zmiennych według formuły nr 9 z tablicy 1.2.

16. Klasa *aggr01* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

• *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 1 z tablicy 1.3.

17. Klasa *aggr02* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

• *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 2 z tablicy 1.3.

18. Klasa *aggr03* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

• *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 3 z tablicy 1.3.

19. Klasa *aggr04* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

• *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 4/11 z tablicy 1.3.

20. Klasa *aggr05* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

• *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 5/12 z tablicy 1.3.

21. Klasa *aggr06* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

• *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 6/13 z tablicy 1.3.

22. Klasa *aggr07* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

- *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 7/14 z tablicy 1.3.

23. Klasa *aggr08* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

- *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 8/15 z tablicy 1.3.

24. Klasa *aggr09* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

- *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 9/16 z tablicy 1.3.

25. Klasa *aggr10* jest klasą pochodną klasy *mva*. Wyposażona jest w standardowy konstruktor oraz destruktor. Zawiera funkcję składową wyznaczającą zmienną syntetyczną.

- *aggreg* - wyznaczenie zmiennej syntetycznej według formuły nr 10/17 z tablicy 1.3.

5.3. Struktury plików danych

Przetwarzane przez program symulacyjny dane liczbowe oraz produkowane wyniki, a także pewne informacje pomocnicze są gromadzone i przechowywane w plikach dyskowych formatu DBF. Wykorzystywane przez program pliki DBF zestawione są w tablicy 5.1.

Plik `PART.DBF` posiada ustaloną strukturę i zawartość w postaci danych dotyczących parametrów generowanych zbiorów danych, tzn. takie wartości jak: liczba cech, liczba obiektów, liczba okresów, granice przedziałów liczbowych, z których generowane są dane liczbowe

oraz wartości średniej arytmetycznej i wariancji. Założenia co do zakresu tych parametrów omówiono w punkcie 3.3.

Tablica 5.1

Pliki DBF wykorzystywane przez program symulacyjny

Lp.	Nazwa pliku	Przeznaczenie pliku DBF
1	PART.DBF	plik zawierający parametry generatora wektorów losowych
2	RNDAT.DBF	plik zawierający parametry wygenerowanych zbiorów danych
3	PS*.DBF	plik charakterystyk statystycznych wygenerowanych zbiorów
4	TD*.DBF	plik zawierający wygenerowany zbiór danych o wymiarach *mxnxv*
5	VC*.DBF	plik zawierający macierz wariancji-kowariancji o wymiarach *mxm*
6	PATHS.DBF	plik zawierający identyfikatory ścieżek WAP
7	GAUGES.DBF	plik zawierający mierniki jakości zmiennych syntetycznych
8	RANGP.DBF	plik zawierający rangi ścieżek WAP
9	RANGW.DBF	plik zawierający rangi formuł ważenia zmiennych
10	RANGN.DBF	plik zawierający rangi formuł normalizacji zmiennych
11	RANGA.DBF	plik zawierający rangi formuł agregacji
12	PX.DBF	plik zawierający bieżące uporządkowanie ścieżek WAP

Źródło: opracowanie własne.

Plik RNDAT.DBF posiada również ustaloną strukturę i zawiera parametry wygenerowanych zbiorów danych wykorzystywane podczas realizacji procedury WAP. Charakter tych wartości jest taki sam, jak parametrów zawartych w pliku PART.DBF, omówionych wyżej.

Pliki PS*.DBF, TD*.DBF i VC*.DBF tworzone są dynamicznie przez program i ich struktura oraz rozmiary (liczba pól i liczba rekordów) zależą od liczby zmiennych, liczby obiektów oraz liczby okresów, czyli wymiarów kostki danych. Do nazw tych plików w miejsce blankietu '*' dołączane są sześcioznakowe identyfikatory poszczególnych zbiorów danych.

W pliku PATHS.DBF zapisane są identyfikatory analizowanych ścieżek WAP oraz poszczególnych formuł ważenia, normalizacji i agregacji zmiennych zgodnie z systematyką przyjętą w rozdziale 1 (tablice 1.1-1.3). Dla potrzeb prowadzonych badań plik ten zawierał 306 rekordów.

Plik GAUGES.DBF przeznaczony jest do przechowywania wartości obliczonych mierników jakości zmiennych syntetycznych.

Pozostałe pliki RANGP.DBF, RANGW.DBF, RANGN.DBF i RANGA.DBF przeznaczone są przechowywania rang odpowiednio poszczególnych ścieżek WAP, wag, formuł normalizacji oraz agregacji zmiennych.

Plik PX.DBF przeznaczony jest do przechowywania informacji dotyczących bieżącego uporządkowania 306 analizowanych i dostępnych ścieżek WAP. Jego zawartość może być aktualizowana każdorazowo po przetworzeniu danego zbioru danych, co oznacza, że uszeregowanie ścieżek może ulec zmianie.

Struktury wszystkich omówionych plików zawierają tablice A.24-A.35 zamieszczone w aneksie.

Zakończenie

W różnego rodzaju pracach naukowo-badawczych nieustannie, chociaż czasem mniej a czasem bardziej wyraźnie, pojawia się problem właściwych relacji pomiędzy rozwojem teorii, stymulowanym wolną i twórczą myślą człowieka, a postępem zastosowań osiągnięć tejże teorii. W świetle tego dylematu wydaje się, iż niniejsza praca jest próbą uporządkowania osiągnięć teoretycznych oraz wskazania warunków aplikacyjnych.

Metody WAP, a w szczególności metody porządkowania liniowego, które są przedmiotem badań prezentowanej rozprawy, stanowią doniosłe osiągnięcie polskiej myśli statystycznej. Z uwagi na zakres, obszerność oraz interdyscyplinarne implikacje tego dorobku, szczegółowe własności formalne i aplikacyjne poszczególnych metod i procedur nie są jeszcze w pełni i zadowalająco rozpoznane i zbadane. Jedną z przyczyn istotnie utrudniających prowadzenie efektywnych badań eksperymentalnych była do niedawna duża złożoność obliczeniowa niektórych algorytmów WAP. Jednakże dynamiczny postęp w elektronice, rozwój technologii komputerowej oraz nowoczesne oprogramowanie coraz skuteczniej usuwają tego rodzaju trudności i przeszkody. W związku z tym powstają sprzyjające warunki do szerszego stosowania metod i technik symulacyjnych, jako efektywnych narzędzi diagnozy własności numerycznych oraz jakości obliczeniowej metod WAP. W prezentowanej pracy przedstawiono próbę wykorzystania technik symulacji komputerowej oraz pewnych narzędzi informatycznych do oceny efektywności i optymalizacji wyboru ścieżek WAP.

Do podstawowych rezultatów przeprowadzonych badań i płynących stąd wniosków zaliczyć należy następujące:

1. Metody WAP uwzględnione w badaniach są zróżnicowane pod względem jakości wyników uzyskiwanych na skutek ich zastosowania. Podstawą estymacji jakości poszczególnych ścieżek jest transformacja wielowymiarowej przestrzeni zmiennych diagnostycznych w jednowymiarową przestrzeń zmiennej syntetycznej i wynikające stąd zniekształcenia.

2. Metody WAP uznawane za stabilne (odporne, mało wrażliwe) „pracują dobrze" niezależnie od wartości parametrów opisowych i struktury przetwarzanych zbiorów danych.

3. Wśród poszczególnych elementów składowych procedury WAP, charakteryzujących się różnymi własnościami formalnymi, wyróżnić można podzbiory bardziej i mniej wrażliwe.

4. Widoczny jest trend zbieżności wyników eksperymentów symulacyjnych, przejawiający się w nieznacznych zmianach w ciągach uporządkowań w miarę zwiększania liczby przebiegów symulacyjnych.

5. Analiza wrażliwości potwierdza jakość „najlepszych" ścieżek WAP, chociaż najmniej wrażliwe na zaburzenia danych okazały się w analizowanych przykładach konfiguracje spoza „najlepszej" dziesiątki.

6. Niektóre z uzyskanych wyników nie potwierdzają często stosowanych w badaniach empirycznych założeń. Chodzi mianowicie o dość powszechne przekonanie, że najlepszą formułą normalizacji cech jest standaryzacja, zaś najbardziej właściwą formuła agregacji odległość Euklidesa.

Dodatkowym rezultatem podjętej pracy jest symulacyjny program komputerowy, w którym zaimplementowano rozpatrywane metody WAP oraz wiele algorytmów numerycznych i procedur ogólnego przeznaczenia, które mogą być wykorzystane do realizacji podobnych zadań. Program zakodowano w stylu obiektowym, co stanowi o jego otwartej architekturze oraz ułatwia daleko idące modyfikacje i rozbudowę.

Na zakończenie należy jeszcze zwrócić uwagę na pewne ograniczenia, które przyjęto podejmując badania, a które były konieczne, aby praca mieściła się w zarysowanych tematem ramach i nie przekraczała objętościowo ogólnie przyjętych normach. Pierwszym poważnym

dylematem, który należało rozstrzygnąć, był zakres i dobór metod analizowanych w trakcie badań. Mając na uwadze wyżej wymienione ograniczenia i przesłanki, włączono ostatecznie do badań metody WAP najczęściej stosowane w praktyce. Należy jednak podkreślić, że wybór taki ma zawsze charakter arbitralny i jest bardzo trudnym problemem decyzyjnym. Drugie istotne rozstrzygnięcie dotyczyło zakresu analizy wrażliwości. Ostatecznie ma ona w pracy charakter fragmentaryczny z tego względu, iż przeprowadzono ją głównie w celu zbadania stabilności najlepiej ocenionych konfiguracji WAP. Ponadto należy zaznaczyć, że w pracy rozpatrywano zbiory danych o parametrach wielowymiarowego rozkładu normalnego. Uzyskanych wyników nie można zatem w prosty sposób uogólniać na zmienne losowe podlegające innym rozkładom.

Bardziej kompleksowe i wyczerpujące ujęcie tak obszernej tematyki wymaga dalszych badań, z czego autor przedstawionej pracy zdaje sobie sprawę.

Spis tablic

Spis rysunków

Spis algorytmów

Literatura

[1] Abrahamowicz M., Zając K.: *Metoda ważenia zmiennych w taksonomii numerycznej i procedurach porządkowania liniowego.* Wrocław: AE1986. W: Prace Naukowe AE we Wrocławiu nr 328.

[2] Abrahamowicz M.: *Konstrukcja syntetycznych mierników rozwoju w świetle twierdzenia Arrowa.* Wrocław: AE1985. W: Prace Naukowe AE we Wrocławiu nr 311.

[3] Adamkiewicz W.: *Cybernetyczne aspekty badania obiektów technicznych.* Wrocław: Ossolineum 1983.

[4] Ajwazjan S. A.: *Podstawowe kierunki zastosowań wielowymiarowej stosowanej analizy statystycznej w badaniach procesów społeczno-ekonomicznych.* W: *Metody i modele ekonomiczno-matematyczne w doskonaleniu zarządzania gospodarką socjalistyczną.* Red. W. Welfe. Warszawa: PWE 1981.

[5] *Analiza systemowa - podstawy i metodologia.* Red. W. Findeisen. Warszawa: PWN 1985.

[6] *Badania przestrzenne rynku i konsumpcji. Przewodnik metodyczny.* Red. S. Mynarski. Warszawa: PWN 1992.

[7] Baranek K.: *Uwagi o wyborze algorytmu grupowania hierarchicznego w taksonomii numerycznej.* Kraków: AE 1981. W: Zeszyty Naukowe AE w Krakowie nr 138.

[8] Barkakati N.: *Borland C++ 4. Przewodnik programisty.* Warszawa: Oficyna Wydawnicza READ ME 1995.

[9] Baron B.: *Metody numeryczne w Turbo Pascalu.* Gliwice: HELION 1994.

[10] Barteczko K.: *Programowanie obiektowe. Praktyczne wprowadzenie do programowania obiektowego w języku C++.* Warszawa: LUPUS 1993.

[11] Bartkowiak A.: *Podstawowe algorytmy statystyki matematycznej.* Warszawa: PWN 1979.

[12] Barton B. F.: *Wprowadzenie do symulacji i gier.* Warszawa: WNT 1974.

[13] Bartosiewicz S.: *Wariacje na temat wzorców rozwoju.* „Przegląd Statystyczny" 1976, nr 4.

[14] Bartosiewicz S.: *Zmienne syntetyczne w modelowaniu ekonometrycznym.* Wrocław: AE 1984. W: Prace Naukowe AE we Wrocławiu nr 262.

[15] Bąk A., Niewęgłowska J., Sobczak E.: *Taksonomiczna analiza towarowych struktur eksportu i importu wybranych państw świata.* Wrocław: AE 1991. W: Prace Naukowe AE we Wrocławiu nr 604.

[16] Bąk A.: *Generowanie danych losowych dla potrzeb symulacyjnej optymalizacji wyboru metod WAP.* W: *Zastosowanie metod taksonomicznych w gospodarce.* Taksonomia. Z. 1. Jelenia Góra-Wrocław-Kraków: Sekcja Klasyfikacji i Analizy Danych PTS 1994.

[17] Bąk A.: *Programowanie obliczeń numerycznych w języku Clipper.* Wrocław: AE 1992. W: Prace Naukowe AE we Wrocławiu nr 643.

[18] Bąk A.: *Redukcja zbioru atrybutów w modelowaniu symulacyjnym systemów ekonomicznych z historią stanów.* Wrocław: AE 1993. W: Prace Naukowe AE we Wrocławiu nr 658.

[19] Bąk A.: *Wybrane procedury wielowymiarowej analizy porównawczej struktur gospodarczych (pakiet WAPSG).* Wrocław: AE 1991. W: Prace Naukowe AE we Wrocławiu nr 602.

[20] Beneš J.: *Teoria systemów.* Warszawa: PWN 1979.

[21] Björck Å., Dahlquist G.: *Metody numeryczne.* Wyd. 2. Warszawa: PWN 1987.

[22] Bobowski Z., Walesiak M.: *Skale pomiaru cech (w ujęciu zwężonym) a zagadnienie normalizacji cech.* Wrocław: AE 1987. W: Prace Naukowe AE we Wrocławiu nr 395.

[23] Bobrowski D.: *Probabilistyka w zastosowaniach technicznych.* Warszawa: WNT 1986.

[24] *Borland C++ version 4.0. Library Reference.* Scotts Valley: Borland International: 1993.

[25] *Borland C++ version 4.0. Programmer's Guide.* Scotts Valley: Borland International: 1993.

[26] *Borland C++ version 4.0. User's Guide.* Scotts Valley: Borland International: 1993.

[27] Borys T.: *Elementy teorii jakości.* Warszawa: PWN 1980.

[28] Borys T.: *Kategoria jakości w statystycznej analizie porównawczej.* Wrocław: AE 1993. Prace Naukowe AE we Wrocławiu nr 284. Seria: Monografie i opracowania nr 23.

[29] Borys T.: *Metody normowania cech w statystycznych badaniach porównawczych.* „Przegląd Statystyczny" 1978, nr 2.

[30] Borys T.: *Propozycja agregatowej miary rozwoju obiektów.* „Przegląd Statystyczny" 1978, nr 3.

[31] Buslenko N. P., Golenko D. I., Sobol I. M., Sragowicz W. G., Szrejder J. A.: *Metoda Monte Carlo.* Warszawa: PWN 1967.

[32] Cieślak M.: *Dobór syndromu zmiennych do porządkowania liniowego obiektów wielowymiarowych.* Wrocław: AE1986. W: Prace Naukowe AE we Wrocławiu nr 328.

[33] Cieślak M.: *Taksonomiczna procedura programowania rozwoju gospodarczego i określania zapotrzebowania na kadry kwalifikowane.* „Przegląd Statystyczny" 1974, nr 1.

[34] Cieślak M.: *Zagadnienie „ruchomego celu" w wielowymiarowej analizie porównawczej.* „Przegląd Statystyczny" 1990, nr 1-2.

[35] Cigánik M.: *Systemy informacyjne w nauce, technice i ekonomice.* Warszawa: PWE 1984.

[36] Coad P., Nicola J.: *Programowanie obiektowe.* Warszawa: Oficyna Wydawnicza READ ME: 1993.

[37] *CodeBase++ version 5.1. Reference Guide.* Sequiter Software 1993.

[38] *CodeBase++ version 5.1. User's Guide.* Sequiter Software 1993.

[39] Czerwiński Z.: *Matematyczne modelowanie procesów ekonomicznych.* Warszawa: PWE 1982.

[40] Dagpunar J.: *Principles of Random Variate Generation.* Oxford: Clarendon Press 1988.

[41] Dąbrowski M., Laus-Mączyńska K.: *Metody wyszukiwania i klasyfikacji informacji.* Warszawa: WNT 1978.

[42] Deo N.: *Teoria grafów i jej zastosowania w technice i informatyce.* Warszawa: PWN 1980.

[43] Devroye L.: *Non-Uniform Random Variate Generation.* New York: Springer-Verlag 1986.

[44] Dijkstra.: *Umiejętność programowania.* Warszawa: WNT 1978.

[45] Domański C.: *Testy statystyczne.* Warszawa: PWE 1990.

[46] Dowdy S. M.: *Statistical Experiments Using BASIC.* Boston: PWS Publishers 1986.

[47] Dryja M., Jankowscy J. i M.: *Przegląd metod i algorytmów numerycznych.* Cz. II. Wyd. 2. Warszawa: WNT 1988.

[48] Dudek M.: *Liczby losowe i Monte Carlo.* Wrocław: „Mikroklan" 1992, nr 2.

[49] Dziechciarz J., Kowalewski G.: *O pewnych przekształceniach danych (rodzaje i problemy).* W: *Zastosowanie metod taksonomicznych w gospodarce.* Taksonomia. Z. 1. Jelenia Góra-Wrocław-Kraków: Sekcja Klasyfikacji i Analizy Danych PTS 1994.

[50] *Ekonometria przestrzenna.* Red. A. Zeliaś. Warszawa: PWE 1991.

[51] *Ekonometria wspomagana mikrokomuterem.* Red. S. Bartosiewicz. Wrocław: AE 1991.

[52] Evans G. W., Wallace G. F., Sutherland G. L.: *Symulacja na maszynach cyfrowych.* Warszawa: WNT 1973.

[53] Fishman G. S.: *Symulacja komputerowa. Pojęcia i metody.* Warszawa: PWE 1981.

[54] Flakiewicz W., Oleński J.: *Cybernetyka ekonomiczna.* Warszawa: PWE 1989.

[55] Fortuna Z., Macukow B., Wąsowski J.: *Metody numeryczne.* Wyd. 2,Warszawa: WNT 1993.

[56] Gass S. I.: *Programowanie liniowe.* Warszawa: PWN 1976.

[57] Gilbert G. G., Koehler D. O.: *Applied Finite Mathematics.* New York: McGraw-Hill 1984.

[58] Gordon G.: *Symulacja systemów.* Warszawa: WNT 1974.

[59] Gościński J.: *Elementy cybernetyki w zarządzaniu.* Warszawa: PWN 1968.

[60] Góral A., Ludwiczak B.: *Programy statystyczne w języku BASIC.* Kraków: AE 1988.

[61] Góralski A.: *Algorytmy i programy statystyki kwalitatywnej.* Warszawa: WNT 1979.

[62] Grabiński T., Malina A., Zeliaś A.: *Metody analizy danych empirycznych na podstawie szeregów przekrojowo-czasowych.* Kraków: AE 1990.

[63] Grabiński T., Wydymus S., Zeliaś A.: *Metody doboru zmiennych w modelach ekonometrycznych.* Warszawa: PWN 1982.

[64] Grabiński T., Wydymus S., Zeliaś A.: *Metody prognozowania rozwoju społeczno-gospodarczego.* Warszawa: PWE 1983.

[65] Grabiński T., Wydymus S., Zeliaś A.: *Metody taksonomii numerycznej w modelowaniu zjawisk społeczno-gospodarczych.* Warszawa: PWN 1989.

[66] Grabiński T.: *Koncepcja badań efektywności procedur porządkowania liniowego.* Kraków: AE 1984. W: Zeszyty Naukowe AE w Krakowie nr 181.

[67] Grabiński T.: *Metody analizy zbieżności wyników dyskryminacji zbiorów.* Kraków: AE 1980. W: Zeszyty Naukowe AE w Krakowie nr 127.

[68] Grabiński T.: *Metody określania charakteru zmiennych w wielowymiarowej analizie porównawczej.* Kraków: AE 1985. W: Zeszyty Naukowe AE w Krakowie nr 213.

[69] Grabiński T.: *Metody taksonometrii.* Kraków: AE 1992.

[70] Grabiński T.: *Program na testowanie poprawności generatorów liczb losowych o rozkładzie równomiernym.* Kraków: AE 1981. W: Zeszyty Naukowe AE w Krakowie nr 138.

[71] Grabiński T.: *Statystyczna analiza poprawności wybranych generatorów liczb losowych o rozkładzie równomiernym.* Kraków: AE 1980. Zeszyty Naukowe nr 131.

[72] Grabiński T.: *Wielowymiarowa analiza porównawcza w badaniach dynamiki zjawisk ekonomicznych.* Kraków: AE 1984. Zeszyty Naukowe AE w Krakowie. Seria specjalna: Monografie nr 61.

[73] Grębosz J.: *Symfonia C++. Programowanie w języku C++ orientowane obiektowo. T. I-III.* Kraków: Oficyna Kallimach 1993.

[74] Groch J.: *Badania diagnostyczne uzdrowisk polskich z zastosowaniem metod wielowymiarowej analizy porównawczej.* Kraków: UJ 1991. Rozprawy Habilitacyjne nr 220.

[75] Guzik B., Burzała M.: *Dyskryminacja zbioru obiektów metodą kul. Program obliczeniowy.* Poznań: AE 1991. Zeszyty Naukowe AE w Poznaniu nr 176.

[76] Guzik B., Burzała M.: *Pakiet programowy obliczania mierników syntetycznych i dendrytów przy dwuszczeblowym podziale zbioru cech.* Poznań: AE 1991. Zeszyty Naukowe AE w Poznaniu nr 184.

[77] Guzik B., Hadasik D.: *O metodach periodyzacji rozwoju.* „Przegląd Statystyczny" 1988, nr 1.

[78] Guzik B., Hadasik D.: *Podobieństwo dendrytu w wielowymiarowej analizie porównawczej.* „Przegląd Statystyczny" 1987, nr 1.

[79] Guzik B., Hadasik D.: *Wyznaczanie bezwzorcowego miernika syntetycznego oraz dendrytu wrocławskiego. Program obliczeniowy.* Poznań: AE 1991. Zeszyty Naukowe AE w Poznaniu nr 176.

[80] Hadasik D.: *Wielowymiarowa analiza porównawcza poziomu życia w wybranych krajach.* Poznań: AE 1991. Zeszyty Naukowe AE w Poznaniu nr 184.

[81] Hall A. D.: *Podstawy techniki systemów. Ogólne zasady projektowania.* Warszawa: PWN 1980.

[82] Hansen T. L.: *C++ - zadania i odpowiedzi.* Warszawa: WNT 1994.

[83] Hanusik K., Łaganowska U.: *Metody ilościowe w badaniach ekonomicznych.* Opole: WSP 1989.

[84] Hellwig Z.: *Elementy rachunku prawdopodobieństwa i statystyki matematycznej.* Warszawa: PWN 1980.

[85] Hellwig Z.: *Stabilność i odporność struktur taksonometrycznych.* W: *Klasyfikacja i analiza danych - problemy teoretyczne.* Taksonomia. Z. 2. Jelenia Góra-Wrocław-Kraków: Sekcja Klasyfikacji i Analizy Danych PTS 1995.

[86] Hellwig Z.: *Taksonometria w konstrukcjach i ocenach strategii gospodarczych.* W: *Zastosowanie metod taksonomicznych w gospodarce.* Wrocław: AE 1994.

[87] Hellwig Z.: *Wielowymiarowa analiza porównawcza i jej zastosowanie w badaniach wielocechowych obiektów gospodarczych.* W: *Metody i modele ekonomiczno-matematyczne w doskonaleniu zarządzania gospodarką socjalistyczną.* Red. W. Welfe. Warszawa: PWE 1981.

[88] Hellwig Z.: *Zastosowanie metody taksonomicznej do typologicznego podziału krajów ze względu na poziom ich rozwoju oraz zasoby i strukturę wykwalifikowanych kadr.* „Przegląd Statystyczny" 1968, nr 4.

[89] Jagielski R.: *Tablice rozproszone.* Warszawa: WNT 1982.

[90] Jajuga K.: *Statystyczna analiza wielowymiarowa.* Warszawa: PWN 1993.

[91] Jajuga K.: *Statystyczna teoria rozpoznawania obrazów.* Warszawa: PWN 1990.

[92] Jajuga K.: *Statystyka ekonomicznych zjawisk złożonych - wykrywanie i analiza niejednorodnych rozkładów wielowymiarowych.* Wrocław: AE 1987. Prace Naukowe AE we Wrocławiu nr 371. Seria: Monografie i opracowania nr 39.

[93] Jankowscy J. i M.: *Przegląd metod i algorytmów numerycznych.* Cz. I. Wyd. 2. Warszawa: WNT 1988.

[94] Jermakow S. M.: *Metoda Monte Carlo i zagadnienia pokrewne.* Warszawa: PWN 1976.

[95] Johnson M. E.: *Multivariate Statistical Simulation.* New York: John Wiley & Sons, Inc. 1987.

[96] Jones C. B.: *Konstrukcja oprogramowania metodą systematyczną.* Warszawa: WNT 1984.

[97] Jóźwiak J., Podgórski J.: *Statystyka od podstaw.* Warszawa: PWE 1992.

[98] Kendall M. G., Buckland W. R.: *Słownik terminów statystycznych.* Warszawa: PWE 1986.

[99] Kiełbasiński A., Schwetlick H.: *Numeryczna algebra liniowa.* Warszawa: WNT 1992.

[100] Kim J. O., Mueller C. W.: *Factor Analysis. Statistical Methods and Practical Issues.* Beverly Hills: Sage Publications 1978.

[101] Kim J. O., Mueller C. W.: *Introduction to Factor Analysis. What it is and How To Do It.* Beverly Hills: Sage Publications 1978.

[102] Koleśnik K., Huzar Z., Fryźlewicz Z.: *Symulacja komputerowa.* Wrocław: PW 1976.

[103] Kolonko J.: *Analiza dyskryminacyjna i jej zastosowania w ekonomii.* Warszawa: PWN 1980.

[104] Kolupa M., Witkowski J. M.: *Wybrane metody numeryczne algebry liniowej w ekonometrii.* Warszawa: PWN 1981.

[105] Kolupa M.: *Elementarny wykład algebry liniowej dla ekonomistów.* Warszawa: PWN 1976.

[106] Kondratowicz L.: *Modelowanie symulacyjne systemów.* Warszawa: WNT 1978.

[107] Krzysztofiak M., Urbanek D.: *Metody statystyczne.* Warszawa: PWN 1979.

[108] Kucharczyk J.: *Algorytmy analizy skupień w języku ALGOL 60.* Warszawa: PWN 1982.

[109] Lipiec-Zajchowska M.: *Metody symulacji komputerowej w prognozowaniu makroekonomicznym.* Warszawa: PWE 1988.

[110] Lippman S. B.: *Podstawy języka C++.* Warszawa: WNT 1994.

[111] Lula P.: *Badanie odporności taksonomicznych metod grupowania* (praca doktorska). Kraków: AE 1992 (maszynopis).

[112] Malarska A.: *Ocena przydatności miar rozwoju do zhierarchizowanej rejonizacji zjawisk złożonych.* „Przegląd Statystyczny" 1991, nr 2.

[113] *Mała encyklopedia statystyki.* Warszawa: PWE 1976.

[114] Marciniak A., Gregulec D., Kaczmarek J.: *Basic Numerical Procedures in Turbo Pascal for Your PC.* Poznań: Wydawnictwo NAKOM 1992.

[115] Martin F. F.: *Wstęp do modelowania cyfrowego.* Warszawa: PWN 1976.

[116] *Maszyny cyfrowe i ich zastosowanie.* Red. Z. Hellwig. Warszawa: PWE 1979.

[117] Mączka P.: *Komputerowa symulacja działania sieciowych systemów obsługi masowej.* Poznań: AE 1993. Zeszyty Naukowe AE w Poznaniu nr 213.

[118] *Metody statystyki międzynarodowej.* Red. A. Zeliaś. Warszawa: PWE 1988.

[119] Milo W.: *Odporność w ekonometrii.* Łódź: Wydawnictwo UŁ 1992.

[120] Morrison D. F.: *Wielowymiarowa analiza statystyczna.* Warszawa: PWN 1990.

[121] Myers G. J.: *Projektowanie niezawodnego oprogramowania.* Warszawa: WNT 1980.

[122] Mynarski S.: *Elementy teorii systemów i cybernetyki.* Warszawa: PWN 1981.

[123] Naylor T. H.: *Modelowanie cyfrowe systemów ekonomicznych.* Warszawa: PWN 1975.

[124] Nowak E. *Problemy doboru zmiennych do modelu ekonometrycznego.* Warszawa: PWN 1984.

[125] Nowak E.: *Ćwiczenia mikrokomputerowe z ekonometrii.* Wrocław: AE 1988.

[126] Nowak E.: *Metody taksonomiczne w klasyfikacji obiektów społeczno-gospodarczych.* Warszawa: PWE 1990.

[127] Nowak E.: *Problem informacji w modelowaniu ekonometrycznym.* Warszawa: PWN 1990.

[128] *Ogólna teoria systemów.* Red. G. J. Klir. Warszawa: WNT 1976.

[129] Oktaba H., Ratajczak W.: *Simula 67.* Warszawa: WNT 1980.

[130] Ostasiewicz W.: *Zastosowanie zbiorów rozmytych w ekonomii.* Warszawa: PWN 1986.

[131] Pawłowski Z.: *Elementy ekonometrii.* Warszawa: PWN 1981.

[132] Perkowski P.: *Technika symulacji cyfrowej.* Warszawa: WNT 1980.

[133] Plucińska A., Pluciński E.: *Elementy probabilistyki.* Warszawa: PWN 1979.

[134] Pluta W.: *Agregatowe zmienne diagnostyczne w badaniach regresyjnych.* „Przegląd Statystyczny" 1976, nr 1.

[135] Pluta W.: *Metody oceny wyników delimitacji.* Wrocław: AE 1984. W: Prace Naukowe AE we Wrocławiu nr 262.

[136] Pluta W.: *Taksonomiczna procedura prowadzenia syntetycznych badań porównawczych za pomocą zmodyfikowanej miary rozwoju gospodarczego.* „Przegląd Statystyczny" 1976, nr 4.

[137] Pluta W.: *Wielowymiarowa analiza porównawcza w badaniach ekonomicznych.* Warszawa: PWE 1977.

[138] Pluta W.: *Wielowymiarowa analiza porównawcza w modelowaniu ekonometrycznym* Warszawa: PWN 1986.

[139] Pociecha J., Podolec B., Sokołowski A., Zając K.: *Metody taksonomiczne w badaniach społeczno-ekonomicznych.* Warszawa: PWN 1988.

[140] Pociecha J.: *Kryteria oceny procedur taksonomicznych.* „Przegląd Statystyczny" 1982, nr 1-2.

[141] Podolec B., Zając K.: *Ekonometryczne metody ustalania rejonów konsumpcji.* Warszawa: PWE 1978.

[142] Pofelski L., Skomorowski M.: *Symulacja komputerowa.* Kraków: UJ 1983.

[143] Polańscy R. i Z.: *Algorytmy i programy dla inżynierów.* Warszawa: Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT 1991.

[144] Przygoda Z., Skrzypek J.: *Programy obliczeniowe w systemie CYBER -72 dla ekonometryków.* Kraków: AE 1985.

[145] Rao C. R.: *Statystyka i prawda.* Warszawa: PWN 1994.

[146] Roetzheim W. H.: *Laboratorium złożoności.* Warszawa: Intersoftland 1994.

[147] Rozin B. B.: *Teoria rozpoznawania obrazów w badaniach ekonomicznych.* Warszawa: PWN 1979.

[148] Seidler J., Badach A., Molisz W.: *Metody rozwiązywania zadań optymalizacji.* Warszawa: WNT 1980.

[149] Sieniawski L. A.: *Problemy i metody programowania.* Wrocław: PW 1980.

[150] Sienkiewicz P.: *Teoria efektywności systemów.* Wrocław: Ossolineum 1987.

[151] Sierpiński W.: *Wstęp do teorii liczb.* Wyd. 3. Warszawa: WSiP 1987.

[152] *Słownik matematyki i cybernetyki ekonomicznej.* Wyd. 2. Warszawa: PWE 1985.

[153] Sobczak E.: *Ocena przydatności formuł normalizacji do prezentacji skali i kształtu struktur gospodarczych.* Wrocław: AE 1991. W: Prace Naukowe AE we Wrocławiu nr 604.

[154] Sobczak E.: *Ocena przydatności miar odległości do analizy zróżnicowania kształtu i skali struktur ekonomicznych.* Wrocław: AE 1991. W: Prace Naukowe AE we Wrocławiu nr 600.

[155] Sobczak W., Malina W.: *Metody selekcji i redukcji informacji.* Warszawa: WNT 1985.

[156] Sokołowski: *Empiryczne testy istotności w taksonomii.* Kraków: AE 1992. Zeszyty Naukowe AE w Krakowie. Seria specjalna: Monografie nr 108.

[157] Stoer J., Bulirsch R.: *Wstęp do analizy numerycznej.* Warszawa: PWN 1987.

[158] Strahl D.: *Metody programowania rozwoju społeczno-gospodarczego.* Warszawa: PWE 1990.

[159] Strahl D.: *Propozycja konstrukcji miary syntetycznej.* „Przegląd Statystyczny" 1978, nr 2.

[160] Strahl D.: *Ścieżka selektywnego rozwoju.* Wrocław: AE 1987. W: Prace Naukowe AE we Wrocławiu nr 395.

[161] Stroustrup B.: *Język C++.* Warszawa: WNT 1994.

[162] Szulc S.: *Metody statystyczne.* Warszawa: PWE 1968.

[163] Tadeusiewicz R., Izworski A., Majewski J.: *Biometria.* Kraków: Wydawnictwa AGH 1993.

[164] Tatarkiewicz J., Witowski A.: *2^5 numerycznych programów w języku BASIC.* Warszawa: Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT 1987.

[165] Theil H.: *Zasady ekonometrii.* Warszawa: PWN 1979.

[166] Tremblay J-P., Sorenson P. G.: *Introduction to Data Structures with Applications.* New York: McGrow-Hill 1984.

[167] Trybuś G.: *Zmienna losowa dystansowa. Teoria i zastosowania.* Wrocław: AE 1981. Prace Naukowe AE we Wrocławiu nr 173. Seria: Monografie i Opracowania nr 1.

[168] Turski W. M.: *Metodologia programowania.* Warszawa: WNT 1980.

[169] Tyszer J.: *Symulacja cyfrowa.* Warszawa: WNT 1990.

[170] Van Tassel D.: *Praktyka programowania.* Warszawa: WNT 1978.

[171] Walesiak M.: *Kilka uwag o niektórych hierarchicznych metodach klasyfikacji.* Wrocław: AE 1986. W: Prace Naukowe AE we Wrocławiu nr 328.

[172] Walesiak M.: *Skale pomiaru cech (w ujęciu zwężonym) a zagadnienie wyboru postaci analitycznej syntetycznych mierników rozwoju.* Wrocław: AE 1988. W: Prace Naukowe AE we Wrocławiu nr 447.

[173] Walesiak M.: *Sposoby wyznaczania optymalnej liczby klas w zagadnieniu klasyfikacji hierarchicznej.* Wrocław: AE 1988. W: Prace Naukowe AE we Wrocławiu nr 449.

[174] Walesiak M.: *Statystyczna analiza wielowymiarowa w badaniach marketingowych.* Wrocław: AE 1993. Prace Naukowe AE we Wrocławiu nr 654. Seria: Monografie i opracowania nr 101.

[175] Walesiak M.: *Zagadnienie oceny podobieństwa zbioru obiektów w czasie w syntetycznych badaniach porównawczych.* „Przegląd Statystyczny" 1993, nr 1.

[176] Wanat K.: *Algorytmy numeryczne.* Gliwice: DIR 1993.

[177] Wilusz T., Wołoszyn J.: *Modelowanie cyfrowe systemów ekonomicznych w języku DELTA.* Kraków: AE 1980.

[178] Winkowski J.: *Programowanie symulacji procesów.* Warszawa: WNT 1974.

[179] Wirth N.: *Wstęp do programowania systematycznego.* Warszawa: WNT 1987.

[180] Wójcik A. R., Laudański Z.: *Planowanie i wnioskowanie statystyczne w doświadczalnictwie.* Warszawa: PWN 1989.

[181] Wydymus S.: *Taksonometryczna koncepcja badania dynamiki globalnego rozwoju społeczno-gospodarczego.* Wrocław: AE 1985. W: Prace Naukowe AE we Wrocławiu nr 311.

[182] Wydymus S.: *Taksonometryczne mierniki tempa rozwoju społeczno-gospodarczego.* Wrocław: AE 1984. W: Prace Naukowe AE we Wrocławiu nr 262.

[183] Zając K.: *Zarys metod statystycznych.* Warszawa: PWE 1982.

[184] Zakrzewska M.: *Zasób zmienności wspólnej czy liczba czynników wspólnych - teoria i praktyka.* W: *Z psychometrycznych problemów diagnostyki psychologicznej.* Red. J. Brzeziński i E. Hornowska. Poznań: Wydawnictwo Naukowe UAM 1993.

[185] Zalewski A.: *Programowanie w językach C i C++ z wykorzystaniem pakietu Borland C++.* Poznań: Wydawnictwo NAKOM 1994.

[186] Zeigler B. P.: *Teoria modelowania i symulacji.* Warszawa: PWN 1984.

[187] Zieliński R.: *Generatory liczb losowych.* Warszawa: WNT 1979.

[188] Zieliński R.: *Metody Monte Carlo.* Warszawa: WNT 1970.

Akademia Ekonomiczna we Wrocławiu
Wydział Gospodarki Regionalnej i Turystyki w Jeleniej Górze

Andrzej Bąk

ZASTOSOWANIE TECHNIK SYMULACJI KOMPUTEROWEJ DO OPTYMALIZACJI WYBORU METOD WIELOWYMIAROWEJ ANALIZY PORÓWNAWCZEJ

(aneks do rozprawy doktorskiej)

Promotor: prof. dr hab. Tadeusz Grabiński

Jelenia Góra 1995

Spis treści

Tablica A.1

Zestaw parametrów generatora nr 1

m	n	v	a	b	Ex	Sx
5	10	1	10.000	200.000	10.000	10.000
5	10	1	50.000	300.000	50.000	50.000
5	10	1	100.000	400.000	100.000	100.000
5	10	1	300.000	500.000	500.000	200.000
5	10	1	500.000	1000.000	1000.000	500.000
5	20	1	10.000	200.000	10.000	10.000
5	20	1	50.000	300.000	50.000	50.000
5	20	1	100.000	400.000	100.000	100.000
5	20	1	300.000	500.000	500.000	200.000
5	20	1	500.000	1000.000	1000.000	500.000
5	50	1	10.000	200.000	10.000	10.000
5	50	1	50.000	300.000	50.000	50.000
5	50	1	100.000	400.000	100.000	100.000
5	50	1	300.000	500.000	500.000	200.000
5	50	1	500.000	1000.000	1000.000	500.000
10	15	1	10.000	200.000	10.000	10.000
10	15	1	50.000	300.000	50.000	50.000
10	15	1	100.000	400.000	100.000	100.000
10	15	1	300.000	500.000	500.000	200.000
10	15	1	500.000	1000.000	1000.000	500.000
10	25	1	10.000	200.000	10.000	10.000
10	25	1	50.000	300.000	50.000	50.000
10	25	1	100.000	400.000	100.000	100.000
10	25	1	300.000	500.000	500.000	200.000
10	25	1	500.000	1000.000	1000.000	500.000
15	10	1	10.000	200.000	10.000	10.000
15	10	1	50.000	300.000	50.000	50.000
15	10	1	100.000	400.000	100.000	100.000
15	10	1	300.000	500.000	500.000	200.000
15	10	1	500.000	1000.000	1000.000	500.000
15	25	1	10.000	200.000	10.000	10.000
15	25	1	50.000	300.000	50.000	50.000
15	25	1	100.000	400.000	100.000	100.000
15	25	1	300.000	500.000	500.000	200.000
15	25	1	500.000	1000.000	1000.000	500.000
15	50	1	10.000	200.000	10.000	10.000
15	50	1	50.000	300.000	50.000	50.000
15	50	1	100.000	400.000	100.000	100.000
15	50	1	300.000	500.000	500.000	200.000
15	50	1	500.000	1000.000	1000.000	500.000
25	50	1	10.000	200.000	10.000	10.000
25	50	1	50.000	300.000	50.000	50.000
25	50	1	100.000	400.000	100.000	100.000
25	50	1	300.000	500.000	500.000	200.000
25	50	1	500.000	1000.000	1000.000	500.000
5	10	2	10.000	200.000	10.000	10.000
5	10	2	50.000	300.000	50.000	50.000
5	10	2	100.000	400.000	100.000	100.000
5	10	2	300.000	500.000	500.000	200.000
5	10	2	500.000	1000.000	1000.000	500.000
5	10	3	10.000	200.000	10.000	10.000
5	10	3	50.000	300.000	50.000	50.000
5	10	3	100.000	400.000	100.000	100.000
5	10	3	300.000	500.000	500.000	200.000
5	10	3	500.000	1000.000	1000.000	500.000
5	15	4	10.000	200.000	10.000	10.000
5	15	4	50.000	300.000	50.000	50.000
5	15	4	100.000	400.000	100.000	100.000
5	15	4	300.000	500.000	500.000	200.000
5	15	4	500.000	1000.000	1000.000	500.000
10	10	3	10.000	200.000	10.000	10.000

10	10	3	50.000	300.000	50.000	50.000
10	10	3	100.000	400.000	100.000	100.000
10	10	3	300.000	500.000	500.000	200.000
10	10	3	500.000	1000.000	1000.000	500.000
10	15	3	10.000	200.000	10.000	10.000
10	15	3	50.000	300.000	50.000	50.000
10	15	3	100.000	400.000	100.000	100.000
10	15	3	300.000	500.000	500.000	200.000
10	15	3	500.000	1000.000	1000.000	500.000
15	20	2	10.000	200.000	10.000	10.000
15	20	2	50.000	300.000	50.000	50.000
15	20	2	100.000	400.000	100.000	100.000
15	20	2	300.000	500.000	500.000	200.000
15	20	2	500.000	1000.000	1000.000	500.000

Źródło: opracowanie własne.

Tablica A.2

Zestaw parametrów generatora nr 2

m	n	v	a	b	Ex	Sx
5	10	1	50.000	250.000	125.000	55.000
5	10	1	150.000	750.000	525.000	255.000
5	10	1	350.000	1750.000	725.000	355.000
5	10	1	550.000	2750.000	1525.000	1255.000
5	10	1	750.000	3750.000	2575.000	1755.000
5	20	1	50.000	250.000	125.000	55.000
5	20	1	150.000	750.000	525.000	255.000
5	20	1	350.000	1750.000	725.000	355.000
5	20	1	550.000	2750.000	1525.000	1255.000
5	20	1	750.000	3750.000	2575.000	1755.000
5	50	1	50.000	250.000	125.000	55.000
5	50	1	150.000	750.000	525.000	255.000
5	50	1	350.000	1750.000	725.000	355.000
5	50	1	550.000	2750.000	1525.000	1255.000
5	50	1	750.000	3750.000	2575.000	1755.000
10	15	1	50.000	250.000	125.000	55.000
10	15	1	150.000	750.000	525.000	255.000
10	15	1	350.000	1750.000	725.000	355.000
10	15	1	550.000	2750.000	1525.000	1255.000
10	15	1	750.000	3750.000	2575.000	1755.000
10	25	1	50.000	250.000	125.000	55.000
10	25	1	150.000	750.000	525.000	255.000
10	25	1	350.000	1750.000	725.000	355.000
10	25	1	550.000	2750.000	1525.000	1255.000
10	25	1	750.000	3750.000	2575.000	1755.000
15	10	1	50.000	250.000	125.000	55.000
15	10	1	150.000	750.000	525.000	255.000
15	10	1	350.000	1750.000	725.000	355.000
15	10	1	550.000	2750.000	1525.000	1255.000
15	10	1	750.000	3750.000	2575.000	1755.000
15	25	1	50.000	250.000	125.000	55.000
15	25	1	150.000	750.000	525.000	255.000
15	25	1	350.000	1750.000	725.000	355.000
15	25	1	550.000	2750.000	1525.000	1255.000
15	25	1	750.000	3750.000	2575.000	1755.000
15	50	1	50.000	250.000	125.000	55.000
15	50	1	150.000	750.000	525.000	255.000
15	50	1	350.000	1750.000	725.000	355.000
15	50	1	550.000	2750.000	1525.000	1255.000
15	50	1	750.000	3750.000	2575.000	1755.000
25	50	1	50.000	250.000	125.000	55.000
25	50	1	150.000	750.000	525.000	255.000
25	50	1	350.000	1750.000	725.000	355.000
25	50	1	550.000	2750.000	1525.000	1255.000

25	50	1	750.000	3750.000	2575.000	1755.000
5	10	2	50.000	250.000	125.000	55.000
5	10	2	150.000	750.000	525.000	255.000
5	10	2	350.000	1750.000	725.000	355.000
5	10	2	550.000	2750.000	1525.000	1255.000
5	10	2	750.000	3750.000	2575.000	1755.000
5	10	3	50.000	250.000	125.000	55.000
5	10	3	150.000	750.000	525.000	255.000
5	10	3	350.000	1750.000	725.000	355.000
5	10	3	550.000	2750.000	1525.000	1255.000
5	10	3	750.000	3750.000	2575.000	1755.000
5	15	4	50.000	250.000	125.000	55.000
5	15	4	150.000	750.000	525.000	255.000
5	15	4	350.000	1750.000	725.000	355.000
5	15	4	550.000	2750.000	1525.000	1255.000
5	15	4	750.000	3750.000	2575.000	1755.000
10	10	3	50.000	250.000	125.000	55.000
10	10	3	150.000	750.000	525.000	255.000
10	10	3	350.000	1750.000	725.000	355.000
10	10	3	550.000	2750.000	1525.000	1255.000
10	10	3	750.000	3750.000	2575.000	1755.000
10	15	3	50.000	250.000	125.000	55.000
10	15	3	150.000	750.000	525.000	255.000
10	15	3	350.000	1750.000	725.000	355.000
10	15	3	550.000	2750.000	1525.000	1255.000
10	15	3	750.000	3750.000	2575.000	1755.000
15	20	2	50.000	250.000	125.000	55.000
15	20	2	150.000	750.000	525.000	255.000
15	20	2	350.000	1750.000	725.000	355.000
15	20	2	550.000	2750.000	1525.000	1255.000
15	20	2	750.000	3750.000	2575.000	1755.000

Źródło: opracowanie własne.

Tablica A.3

Zestaw parametrów generatora nr 3

m	n	v	a	b	Ex	Sx
5	10	1	10.000000	200.000000	10.000000	10.000000
5	10	1	50.000000	300.000000	50.000000	50.000000
5	10	1	100.000000	400.000000	100.000000	100.000000
5	10	1	300.000000	500.000000	500.000000	200.000000
5	10	1	500.000000	1000.000000	1000.000000	500.000000
5	20	1	10.000000	200.000000	10.000000	10.000000
5	20	1	50.000000	300.000000	50.000000	50.000000
5	20	1	100.000000	400.000000	100.000000	100.000000
5	20	1	300.000000	500.000000	500.000000	200.000000
5	20	1	500.000000	1000.000000	1000.000000	500.000000
5	50	1	10.000000	200.000000	10.000000	10.000000
5	50	1	50.000000	300.000000	50.000000	50.000000
5	50	1	100.000000	400.000000	100.000000	100.000000
5	50	1	300.000000	500.000000	500.000000	200.000000
5	50	1	500.000000	1000.000000	1000.000000	500.000000
10	15	1	10.000000	200.000000	10.000000	10.000000
10	15	1	50.000000	300.000000	50.000000	50.000000
10	15	1	100.000000	400.000000	100.000000	100.000000
10	15	1	300.000000	500.000000	500.000000	200.000000
10	15	1	500.000000	1000.000000	1000.000000	500.000000
10	25	1	10.000000	200.000000	10.000000	10.000000
10	25	1	50.000000	300.000000	50.000000	50.000000
10	25	1	100.000000	400.000000	100.000000	100.000000
10	25	1	300.000000	500.000000	500.000000	200.000000
10	25	1	500.000000	1000.000000	1000.000000	500.000000
15	10	1	10.000000	200.000000	10.000000	10.000000
15	10	1	50.000000	300.000000	50.000000	50.000000

15	10	1	100.000000	400.000000	100.000000	100.000000
15	10	1	300.000000	500.000000	500.000000	200.000000
15	10	1	500.000000	1000.000000	1000.000000	500.000000
15	25	1	10.000000	200.000000	10.000000	10.000000
15	25	1	50.000000	300.000000	50.000000	50.000000
15	25	1	100.000000	400.000000	100.000000	100.000000
15	25	1	300.000000	500.000000	500.000000	200.000000
15	25	1	500.000000	1000.000000	1000.000000	500.000000
15	50	1	10.000000	200.000000	10.000000	10.000000
15	50	1	50.000000	300.000000	50.000000	50.000000
15	50	1	100.000000	400.000000	100.000000	100.000000
15	50	1	300.000000	500.000000	500.000000	200.000000
15	50	1	500.000000	1000.000000	1000.000000	500.000000
25	50	1	10.000000	200.000000	10.000000	10.000000
25	50	1	50.000000	300.000000	50.000000	50.000000
25	50	1	100.000000	400.000000	100.000000	100.000000
25	50	1	300.000000	500.000000	500.000000	200.000000
25	50	1	500.000000	1000.000000	1000.000000	500.000000
5	10	2	10.000000	200.000000	10.000000	10.000000
5	10	2	50.000000	300.000000	50.000000	50.000000
5	10	2	100.000000	400.000000	100.000000	100.000000
5	10	2	300.000000	500.000000	500.000000	200.000000
5	10	2	500.000000	1000.000000	1000.000000	500.000000
5	10	3	10.000000	200.000000	10.000000	10.000000
5	10	3	50.000000	300.000000	50.000000	50.000000
5	10	3	100.000000	400.000000	100.000000	100.000000
5	10	3	300.000000	500.000000	500.000000	200.000000
5	10	3	500.000000	1000.000000	1000.000000	500.000000
5	15	4	10.000000	200.000000	10.000000	10.000000
5	15	4	50.000000	300.000000	50.000000	50.000000
5	15	4	100.000000	400.000000	100.000000	100.000000
5	15	4	300.000000	500.000000	500.000000	200.000000
5	15	4	500.000000	1000.000000	1000.000000	500.000000
10	10	3	10.000000	200.000000	10.000000	10.000000
10	10	3	50.000000	300.000000	50.000000	50.000000
10	10	3	100.000000	400.000000	100.000000	100.000000
10	10	3	300.000000	500.000000	500.000000	200.000000
10	10	3	500.000000	1000.000000	1000.000000	500.000000
10	15	3	10.000000	200.000000	10.000000	10.000000
10	15	3	50.000000	300.000000	50.000000	50.000000
10	15	3	100.000000	400.000000	100.000000	100.000000
10	15	3	300.000000	500.000000	500.000000	200.000000
10	15	3	500.000000	1000.000000	1000.000000	500.000000
15	20	2	10.000000	200.000000	10.000000	10.000000
15	20	2	50.000000	300.000000	50.000000	50.000000
15	20	2	100.000000	400.000000	100.000000	100.000000
15	20	2	300.000000	500.000000	500.000000	200.000000
15	20	2	500.000000	1000.000000	1000.000000	500.000000

Źródło: opracowanie własne.

Tablica A.4

Uporządkowanie ścieżek WAP dla p=306 - zestaw danych nr 1

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	2,2,2	41212	46	2,9,4	37729	91	1,8,8	31270	136	2,4,6	26813
2	1,2,2	40998	47	1,7,5	37699	92	1,2,7	31230	137	1,4,6	26653
3	2,7,2	40534	48	2,7,4	37640	93	2,9,7	31128	138	2,6,6	26504
4	2,3,2	40525	49	2,8,4	37635	94	1,9,6	30923	139	2,8,17	26245
5	2,4,2	40264	50	1,5,5	37567	95	2,3,7	30767	140	1,6,6	26195
6	1,3,2	40152	51	1,1,2	37543	96	1,8,7	30698	141	2,6,8	25999
7	1,6,2	40041	52	2,5,4	37542	97	1,3,7	30665	142	2,1,6	25733
8	1,7,2	39885	53	2,1,1	37440	98	2,8,7	30608	143	1,6,8	25705
9	1,4,7	39770	54	2,3,5	37334	99	2,9,6	30409	144	1,1,6	25657
10	2,3,1	39758	55	2,6,4	37296	100	2,9,9	30185	145	1,2,17	25331
11	1,4,2	39641	56	1,1,1	37100	101	1,9,9	30183	146	1,6,17	25151
12	2,6,2	39585	57	1,2,4	37086	102	2,8,8	30083	147	1,9,17	24985
13	2,7,1	39305	58	1,6,5	37082	103	1,1,9	29515	148	2,6,17	24972
14	2,2,1	39176	59	2,9,5	37031	104	1,9,8	29481	149	2,7,17	24962
15	1,9,2	39115	60	2,8,5	37026	105	1,7,7	29302	150	1,7,17	24909
16	1,7,1	39005	61	2,2,5	36961	106	1,1,8	29259	151	1,3,17	24808
17	1,9,1	38920	62	2,7,5	36957	107	1,1,7	29135	152	1,8,10	24807
18	2,6,1	38890	63	1,3,5	36948	108	2,7,9	29102	153	2,8,10	24651
19	2,9,2	38827	64	1,3,4	36918	109	2,1,8	29077	154	1,5,17	24615
20	1,6,1	38708	65	2,5,5	36879	110	2,1,7	28998	155	2,9,17	24567
21	1,8,2	38681	66	1,4,4	36720	111	2,9,8	28968	156	2,5,17	24279
22	1,2,1	38624	67	2,6,5	36640	112	2,1,9	28947	157	2,2,17	23970
23	2,9,1	38583	68	1,4,5	36541	113	2,7,7	28937	158	2,3,17	23966
24	1,5,1	38581	69	2,4,5	36407	114	1,5,9	28750	159	1,5,10	23712
25	2,4,1	38523	70	1,2,5	36399	115	1,7,9	28609	160	1,1,17	23677
26	1,3,1	38486	71	1,1,4	36233	116	1,6,7	28405	161	2,5,10	23366
27	1,5,2	38468	72	2,1,4	36043	117	1,5,7	28267	162	2,1,17	23061
28	2,8,2	38437	73	2,1,5	35858	118	1,4,17	28085	163	1,4,9	22744
29	2,1,2	38323	74	1,1,5	35755	119	2,5,7	28078	164	2,9,10	20497
30	2,5,2	38306	75	2,2,8	34125	120	1,5,8	28060	165	2,4,10	20190
31	1,8,1	38286	76	1,2,9	33782	121	2,6,7	27895	166	1,9,10	20144
32	1,5,4	38215	77	2,2,9	33710	122	2,5,9	27719	167	1,4,10	19618
33	2,5,1	38214	78	1,2,8	33665	123	2,7,8	27719	168	2,4,9	19508
34	1,4,1	38154	79	2,8,6	33058	124	2,4,17	27675	169	2,7,10	18790
35	1,9,4	38051	80	1,8,6	33022	125	2,7,6	27594	170	1,2,15	18100
36	2,3,4	37992	81	2,3,8	32559	126	1,2,6	27440	171	1,7,10	18077
37	2,8,1	37990	82	1,3,8	32373	127	2,3,6	27415	172	2,6,10	17971
38	1,8,4	37959	83	1,3,9	32344	128	2,6,9	27339	173	2,3,10	17902
39	2,2,4	37940	84	1,5,6	32274	129	1,7,8	27266	174	1,6,10	17891
40	2,4,7	37902	85	1,8,9	32245	130	2,5,8	27175	175	2,2,15	17787
41	1,7,4	37883	86	2,3,9	32095	131	1,3,6	27093	176	1,3,10	17558
42	1,9,5	37857	87	2,5,6	31914	132	1,6,9	27039	177	2,3,15	17511
43	2,4,4	37805	88	2,8,9	31413	133	2,2,6	27010	178	2,1,10	17485
44	1,6,4	37798	89	2,2,7	31403	134	1,8,17	26881	179	2,2,10	17447
45	1,8,5	37784	90	1,9,7	31325	135	1,7,6	26861	180	2,1,15	17184

Źródło: opracowanie własne.

Tablica A.4 (c.d.)

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
181	1,3,15	17030	226	1,3,11	12411	271	1,1,13	8844
182	1,2,10	17009	227	2,8,12	12384	272	1,7,13	8735
183	1,1,10	16861	228	2,5,11	12383	273	2,9,14	8591
184	2,1,12	16653	229	2,6,15	12363	274	1,2,13	8469
185	1,1,15	16491	230	2,9,12	12322	275	1,4,13	8327
186	1,3,3	16412	231	2,7,12	12228	276	2,4,16	8254
187	1,4,14	16343	232	2,6,11	12182	277	2,9,13	8238
188	2,4,14	16121	233	2,5,12	12147	278	2,6,16	8195
189	2,3,12	15886	234	1,4,11	12127	279	1,6,13	8099
190	1,1,12	15634	235	1,5,12	11937	280	1,1,14	8045
191	2,4,12	15557	236	2,6,12	11878	281	2,5,15	7986
192	2,3,3	15555	237	2,8,15	11774	282	2,7,13	7878
193	1,2,3	15253	238	1,9,12	11721	283	1,2,14	7710
194	1,7,15	15252	239	1,6,12	11690	284	1,8,13	7698
195	2,2,3	15198	240	1,2,11	11678	285	2,8,13	7686
196	1,9,3	15180	241	1,9,11	11664	286	1,9,13	7663
197	2,9,3	15173	242	1,7,12	11661	287	1,5,15	7642
198	1,1,3	15048	243	1,8,12	11628	288	1,3,14	7630
199	2,1,3	14950	244	2,4,8	11606	289	2,6,13	7419
200	2,2,16	14935	245	1,8,11	11563	290	2,5,13	7237
201	2,1,11	14846	246	2,6,3	11539	291	1,5,13	7209
202	1,7,3	14827	247	1,7,11	11518	292	1,9,14	7154
203	2,2,12	14697	248	1,8,15	11505	293	2,5,16	6606
204	2,1,16	14624	249	1,5,11	11029	294	2,5,14	6468
205	2,7,3	14489	250	1,6,11	10959	295	1,5,3	6230
206	2,3,11	14426	251	2,9,16	10662	296	2,7,14	6193
207	2,3,16	14369	252	1,7,16	10383	297	2,5,3	6045
208	2,7,15	14344	253	1,4,16	10139	298	1,5,16	5770
209	1,2,16	14266	254	2,8,14	10111	299	1,5,14	5705
210	1,3,12	14104	255	2,8,16	10053	300	1,4,15	5679
211	2,2,11	13990	256	2,1,13	9994	301	2,6,14	5640
212	2,4,11	13968	257	1,9,16	9897	302	1,7,14	5231
213	2,9,15	13943	258	2,7,16	9791	303	2,4,15	4821
214	1,4,8	13908	259	2,1,14	9580	304	1,6,14	4716
215	1,9,15	13884	260	1,8,16	9531	305	1,4,3	301
216	1,3,16	13801	261	2,2,14	9425	306	2,4,3	192
217	1,4,12	13683	262	2,3,14	9388			
218	1,1,11	13433	263	1,8,3	9367			
219	1,2,12	13192	264	2,3,13	9347			
220	1,6,15	13073	265	2,8,3	9318			
221	1,1,16	12921	266	2,2,13	9206			
222	2,8,11	12786	267	1,8,14	9050			
223	2,9,11	12786	268	2,4,13	8963			
224	2,7,11	12679	269	1,6,16	8904			
225	1,6,3	12501	270	1,3,13	8846			

Źródło: opracowanie własne.

Tablica A.5

Wskaźniki dynamiki rang ścieżek WAP dla p=306 - zestaw danych nr 1

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	2,2,2	-	46	2,9,4	0,999	91	1,8,8	0,998	136	2,4,6	0,998
2	1,2,2	0,995	47	1,7,5	0,999	92	1,2,7	0,999	137	1,4,6	0,994
3	2,7,2	0,989	48	2,7,4	0,998	93	2,9,7	0,997	138	2,6,6	0,994
4	2,3,2	1,000	49	2,8,4	1,000	94	1,9,6	0,993	139	2,8,17	0,990
5	2,4,2	0,994	50	1,5,5	0,998	95	2,3,7	0,995	140	1,6,6	0,998
6	1,3,2	0,997	51	1,1,2	0,999	96	1,8,7	0,998	141	2,6,8	0,993
7	1,6,2	0,997	52	2,5,4	1,000	97	1,3,7	0,999	142	2,1,6	0,990
8	1,7,2	0,996	53	2,1,1	0,997	98	2,8,7	0,998	143	1,6,8	0,999
9	1,4,7	0,997	54	2,3,5	0,997	99	2,9,6	0,993	144	1,1,6	0,998
10	2,3,1	1,000	55	2,6,4	0,999	100	2,9,9	0,993	145	1,2,17	0,987
11	1,4,2	0,997	56	1,1,1	0,995	101	1,9,9	1,000	146	1,6,17	0,993
12	2,6,2	0,999	57	1,2,4	1,000	102	2,8,8	0,997	147	1,9,17	0,993
13	2,7,1	0,993	58	1,6,5	1,000	103	1,1,9	0,981	148	2,6,17	0,999
14	2,2,1	0,997	59	2,9,5	0,999	104	1,9,8	0,999	149	2,7,17	1,000
15	1,9,2	0,998	60	2,8,5	1,000	105	1,7,7	0,994	150	1,7,17	0,998
16	1,7,1	0,997	61	2,2,5	0,998	106	1,1,8	0,999	151	1,3,17	0,996
17	1,9,1	0,998	62	2,7,5	1,000	107	1,1,7	0,996	152	1,8,10	1,000
18	2,6,1	0,999	63	1,3,5	1,000	108	2,7,9	0,999	153	2,8,10	0,994
19	2,9,2	0,998	64	1,3,4	0,999	109	2,1,8	0,999	154	1,5,17	0,999
20	1,6,1	0,997	65	2,5,5	0,999	110	2,1,7	0,997	155	2,9,17	0,998
21	1,8,2	0,999	66	1,4,4	0,996	111	2,9,8	0,999	156	2,5,17	0,988
22	1,2,1	0,999	67	2,6,5	0,998	112	2,1,9	0,999	157	2,2,17	0,987
23	2,9,1	0,999	68	1,4,5	0,997	113	2,7,7	1,000	158	2,3,17	1,000
24	1,5,1	1,000	69	2,4,5	0,996	114	1,5,9	0,994	159	1,5,10	0,989
25	2,4,1	0,998	70	1,2,5	1,000	115	1,7,9	0,995	160	1,1,17	0,999
26	1,3,1	0,999	71	1,1,4	0,995	116	1,6,7	0,993	161	2,5,10	0,987
27	1,5,2	1,000	72	2,1,4	0,995	117	1,5,7	0,995	162	2,1,17	0,987
28	2,8,2	0,999	73	2,1,5	0,995	118	1,4,17	0,994	163	1,4,9	0,986
29	2,1,2	0,997	74	1,1,5	0,997	119	2,5,7	1,000	164	2,9,10	0,901
30	2,5,2	1,000	75	2,2,8	0,954	120	1,5,8	0,999	165	2,4,10	0,985
31	1,8,1	0,999	76	1,2,9	0,990	121	2,6,7	0,994	166	1,9,10	0,998
32	1,5,4	0,998	77	2,2,9	0,998	122	2,5,9	0,994	167	1,4,10	0,974
33	2,5,1	1,000	78	1,2,8	0,999	123	2,7,8	1,000	168	2,4,9	0,994
34	1,4,1	0,998	79	2,8,6	0,982	124	2,4,17	0,998	169	2,7,10	0,963
35	1,9,4	0,997	80	1,8,6	0,999	125	2,7,6	0,997	170	1,2,15	0,963
36	2,3,4	0,998	81	2,3,8	0,986	126	1,2,6	0,994	171	1,7,10	0,999
37	2,8,1	1,000	82	1,3,8	0,994	127	2,3,6	0,999	172	2,6,10	0,994
38	1,8,4	0,999	83	1,3,9	0,999	128	2,6,9	0,997	173	2,3,10	0,996
39	2,2,4	0,999	84	1,5,6	0,998	129	1,7,8	0,997	174	1,6,10	0,999
40	2,4,7	0,999	85	1,8,9	0,999	130	2,5,8	0,997	175	2,2,15	0,994
41	1,7,4	0,999	86	2,3,9	0,995	131	1,3,6	0,997	176	1,3,10	0,987
42	1,9,5	0,999	87	2,5,6	0,994	132	1,6,9	0,998	177	2,3,15	0,997
43	2,4,4	0,999	88	2,8,9	0,984	133	2,2,6	0,999	178	2,1,10	0,999
44	1,6,4	1,000	89	2,2,7	1,000	134	1,8,17	0,995	179	2,2,10	0,998
45	1,8,5	1,000	90	1,9,7	0,998	135	1,7,6	0,999	180	2,1,15	0,985

Tablica A.5 (c.d.)

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
181	1,3,15	0,991	226	1,3,11	0,993	271	1,1,13	1,000
182	1,2,10	0,999	227	2,8,12	0,998	272	1,7,13	0,988
183	1,1,10	0,991	228	2,5,11	1,000	273	2,9,14	0,984
184	2,1,12	0,988	229	2,6,15	0,998	274	1,2,13	0,986
185	1,1,15	0,990	230	2,9,12	0,997	275	1,4,13	0,983
186	1,3,3	0,995	231	2,7,12	0,992	276	2,4,16	0,991
187	1,4,14	0,996	232	2,6,11	0,996	277	2,9,13	0,998
188	2,4,14	0,986	233	2,5,12	0,997	278	2,6,16	0,995
189	2,3,12	0,985	234	1,4,11	0,998	279	1,6,13	0,988
190	1,1,12	0,984	235	1,5,12	0,984	280	1,1,14	0,993
191	2,4,12	0,995	236	2,6,12	0,995	281	2,5,15	0,993
192	2,3,3	1,000	237	2,8,15	0,991	282	2,7,13	0,986
193	1,2,3	0,981	238	1,9,12	0,995	283	1,2,14	0,979
194	1,7,15	1,000	239	1,6,12	0,997	284	1,8,13	0,998
195	2,2,3	0,996	240	1,2,11	0,999	285	2,8,13	0,998
196	1,9,3	0,999	241	1,9,11	0,999	286	1,9,13	0,997
197	2,9,3	1,000	242	1,7,12	1,000	287	1,5,15	0,997
198	1,1,3	0,992	243	1,8,12	0,997	288	1,3,14	0,998
199	2,1,3	0,993	244	2,4,8	0,998	289	2,6,13	0,972
200	2,2,16	0,999	245	1,8,11	0,996	290	2,5,13	0,975
201	2,1,11	0,994	246	2,6,3	0,998	291	1,5,13	0,996
202	1,7,3	0,999	247	1,7,11	0,998	292	1,9,14	0,992
203	2,2,12	0,991	248	1,8,15	0,999	293	2,5,16	0,923
204	2,1,16	0,995	249	1,5,11	0,959	294	2,5,14	0,979
205	2,7,3	0,991	250	1,6,11	0,994	295	1,5,3	0,963
206	2,3,11	0,996	251	2,9,16	0,973	296	2,7,14	0,994
207	2,3,16	0,996	252	1,7,16	0,974	297	2,5,3	0,976
208	2,7,15	0,998	253	1,4,16	0,977	298	1,5,16	0,955
209	1,2,16	0,995	254	2,8,14	0,997	299	1,5,14	0,989
210	1,3,12	0,989	255	2,8,16	0,994	300	1,4,15	0,995
211	2,2,11	0,992	256	2,1,13	0,994	301	2,6,14	0,993
212	2,4,11	0,998	257	1,9,16	0,990	302	1,7,14	0,927
213	2,9,15	0,998	258	2,7,16	0,989	303	2,4,15	0,922
214	1,4,8	0,997	259	2,1,14	0,978	304	1,6,14	0,978
215	1,9,15	0,998	260	1,8,16	0,995	305	1,4,3	0,064
216	1,3,16	0,994	261	2,2,14	0,989	306	2,4,3	0,638
217	1,4,12	0,991	262	2,3,14	0,996			
218	1,1,11	0,982	263	1,8,3	0,998			
219	1,2,12	0,982	264	2,3,13	0,998			
220	1,6,15	0,991	265	2,8,3	0,997			
221	1,1,16	0,988	266	2,2,13	0,988			
222	2,8,11	0,990	267	1,8,14	0,983			
223	2,9,11	1,000	268	2,4,13	0,990			
224	2,7,11	0,992	269	1,6,16	0,993			
225	1,6,3	0,986	270	1,3,13	0,993			

Źródło: opracowanie własne.

Tablica A.6

Uporządkowanie ścieżek WAP dla p=10 - zestaw danych nr 1

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	999	46	2,4,4	31	91	1,6,8	9	136	2,2,11	1
2	2,4,7	839	47	1,5,4	30	92	1,7,4	9	137	2,3,9	1
3	1,7,2	335	48	1,6,4	28	93	1,8,11	9	138	2,3,11	1
4	2,2,2	334	49	2,3,4	27	94	2,3,13	9	pozostałe		0
5	1,6,2	325	50	1,8,1	26	95	2,5,9	9			
6	2,7,2	297	51	2,4,17	26	96	2,7,3	9			
7	1,2,2	290	52	1,2,1	25	97	2,8,5	9			
8	1,4,9	287	53	1,1,1	23	98	1,2,9	8			
9	2,6,2	237	54	1,4,10	22	99	1,2,17	8			
10	2,3,2	210	55	1,6,5	22	100	1,3,3	8			
11	2,4,2	206	56	2,2,4	20	101	1,6,3	8			
12	2,1,2	203	57	1,3,10	19	102	2,5,17	8			
13	2,4,9	198	58	2,3,8	19	103	1,3,15	7			
14	1,3,2	174	59	1,7,9	18	104	1,8,13	7			
15	1,4,2	169	60	2,6,17	18	105	1,9,15	7			
16	1,8,2	166	61	2,2,8	17	106	2,1,17	7			
17	2,9,2	131	62	2,3,17	16	107	2,5,14	6			
18	1,1,2	123	63	2,8,4	16	108	1,1,8	5			
19	1,5,2	117	64	2,4,10	15	109	1,3,11	5			
20	1,9,2	105	65	2,7,4	15	110	1,4,11	5			
21	2,8,2	102	66	2,7,17	15	111	1,7,17	5			
22	2,7,1	98	67	1,2,8	14	112	2,3,15	5			
23	1,4,8	93	68	1,5,5	14	113	2,5,6	5			
24	2,5,2	90	69	2,9,4	14	114	2,6,10	5			
25	1,6,1	85	70	2,1,10	13	115	1,1,4	4			
26	2,3,1	80	71	2,9,5	13	116	1,3,7	4			
27	2,4,8	79	72	1,3,1	12	117	1,7,11	4			
28	2,4,1	78	73	1,9,4	12	118	2,8,14	4			
29	2,1,5	77	74	2,1,8	12	119	1,1,10	3			
30	1,7,1	68	75	2,7,5	12	120	1,1,17	3			
31	2,9,1	68	76	2,7,10	12	121	1,6,6	3			
32	2,1,1	64	77	1,3,5	11	122	1,9,13	3			
33	1,5,1	56	78	1,6,10	11	123	2,9,16	3			
34	2,2,1	56	79	1,6,17	11	124	1,3,9	2			
35	2,2,5	50	80	1,7,10	11	125	1,3,17	2			
36	1,1,5	48	81	1,3,8	10	126	1,5,11	2			
37	1,9,1	48	82	1,3,13	10	127	1,6,7	2			
38	2,3,5	48	83	1,5,15	10	128	1,6,16	2			
39	2,8,1	48	84	1,8,4	10	129	1,8,6	2			
40	2,4,5	47	85	1,8,14	10	130	2,2,17	2			
41	1,9,5	40	86	2,3,10	10	131	2,5,1	2			
42	2,6,1	38	87	1,2,5	9	132	2,7,11	2			
43	1,7,5	35	88	1,2,10	9	133	1,2,16	1			
44	1,8,5	34	89	1,4,1	9	134	1,7,7	1			
45	2,1,4	32	90	1,4,5	9	135	2,1,7	1			

Źródło: opracowanie własne.

Tablica A.7

Uporządkowanie ścieżek WAP dla p=5 - zestaw danych nr 1

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	443	46	2,8,1	6	91	1,7,10	1
2	2,4,7	364	47	1,3,13	5	92	2,3,4	1
3	1,4,9	107	48	1,5,15	5	93	2,5,14	1
4	1,6,2	97	49	1,6,17	5	94	2,7,4	1
5	1,7,2	97	50	1,8,14	5	95	2,7,5	1
6	2,4,9	77	51	2,1,10	5	pozostałe		0
7	1,2,2	72	52	2,4,10	5			
8	2,2,2	70	53	2,6,1	5			
9	2,7,2	67	54	2,7,17	5			
10	2,6,2	61	55	1,6,8	4			
11	1,8,2	49	56	1,8,11	4			
12	2,3,2	42	57	2,1,4	4			
13	2,1,2	39	58	2,2,5	4			
14	2,9,2	38	59	2,3,8	4			
15	2,4,2	36	60	2,3,13	4			
16	1,3,2	35	61	2,5,9	4			
17	1,4,2	35	62	2,7,3	4			
18	1,5,2	27	63	1,2,1	3			
19	1,4,8	26	64	1,2,17	3			
20	2,4,8	24	65	1,3,3	3			
21	2,7,1	23	66	1,6,3	3			
22	1,6,1	22	67	1,6,5	3			
23	2,8,2	22	68	1,7,5	3			
24	1,9,2	20	69	1,8,1	3			
25	1,1,2	18	70	2,2,4	3			
26	2,1,5	17	71	2,2,8	3			
27	2,1,1	16	72	2,3,10	3			
28	1,1,5	15	73	2,4,17	3			
29	2,5,2	15	74	2,5,17	3			
30	1,7,1	12	75	2,7,10	3			
31	2,9,1	11	76	1,3,5	2			
32	2,3,5	10	77	1,3,8	2			
33	1,5,1	9	78	1,3,15	2			
34	2,3,1	9	79	1,6,4	2			
35	2,4,5	9	80	1,6,10	2			
36	1,7,9	8	81	1,8,13	2			
37	2,4,1	8	82	1,9,15	2			
38	2,6,17	8	83	2,1,8	2			
39	1,4,10	7	84	2,1,17	2			
40	1,9,5	7	85	2,4,4	2			
41	1,3,10	6	86	2,9,5	2			
42	1,8,5	6	87	1,2,8	1			
43	1,9,1	6	88	1,2,10	1			
44	2,2,1	6	89	1,4,5	1			
45	2,3,17	6	90	1,5,4	1			

Źródło: opracowanie własne.

Tablica A.8

Uporządkowanie ścieżek WAP dla p=3 - zestaw danych nr 1

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	237	46	2,3,13	2
2	2,4,7	192	47	2,3,17	2
3	1,4,9	43	48	2,4,10	2
4	1,6,2	39	49	2,5,9	2
5	1,7,2	37	50	2,7,3	2
6	2,4,9	32	51	2,8,1	2
7	1,2,2	25	52	1,2,1	1
8	2,6,2	25	53	1,2,17	1
9	2,2,2	19	54	1,3,3	1
10	1,8,2	18	55	1,6,3	1
11	2,7,2	18	56	1,9,1	1
12	2,9,2	14	57	2,2,1	1
13	2,1,2	13	58	2,3,8	1
14	2,3,2	12	59	2,3,10	1
15	1,4,2	11	60	2,4,1	1
16	1,5,2	10	61	2,4,5	1
17	1,4,8	9	62	2,5,17	1
18	2,4,2	9	63	2,7,10	1
19	2,4,8	9	64	2,7,17	1
20	1,6,1	8	65	2,9,1	1
21	2,1,5	8	pozostałe		0
22	2,8,2	8			
23	1,3,2	7			
24	2,1,1	5			
25	2,5,2	5			
26	2,7,1	5			
27	1,1,5	4			
28	1,7,1	4			
29	1,7,9	4			
30	2,6,17	4			
31	1,1,2	3			
32	1,3,10	3			
33	1,3,13	3			
34	1,4,10	3			
35	1,5,15	3			
36	1,6,17	3			
37	1,8,14	3			
38	1,9,2	3			
39	2,1,10	3			
40	2,3,5	3			
41	1,6,8	2			
42	1,8,5	2			
43	1,8,11	2			
44	1,9,5	2			
45	2,3,1	2			

Źródło: opracowanie własne.

Tablica A.9

Uporządkowanie ścieżek WAP dla p=306 - zestaw danych nr 2

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	2,4,2	42908	46	2,3,8	37532	91	2,8,9	31906	136	1,8,17	26193
2	2,2,2	42362	47	1,9,1	37430	92	2,9,7	31562	137	2,3,17	26164
3	2,3,2	42280	48	2,2,8	37372	93	1,8,6	31513	138	1,4,6	25937
4	2,1,2	42163	49	1,3,1	37354	94	1,5,6	31506	139	2,1,6	25931
5	2,6,2	41067	50	2,5,5	37317	95	2,8,7	31429	140	2,2,17	25908
6	2,7,2	40868	51	1,7,1	37253	96	1,8,9	31359	141	1,3,6	25881
7	1,4,2	40523	52	1,1,1	37224	97	2,9,9	30992	142	1,7,8	25768
8	1,2,2	40367	53	1,6,1	37198	98	2,1,7	30901	143	2,7,17	25551
9	2,4,1	40313	54	1,5,4	37095	99	2,8,8	30848	144	1,2,6	25520
10	1,1,2	40076	55	1,8,1	36921	100	2,9,6	30725	145	2,8,10	25500
11	2,2,1	39919	56	2,8,5	36797	101	1,3,7	30492	146	2,1,17	25486
12	1,3,2	39889	57	2,2,9	36709	102	1,2,7	30389	147	1,3,17	25006
13	2,3,1	39639	58	2,9,5	36587	103	1,9,7	29888	148	1,9,17	24937
14	2,1,1	39556	59	2,6,5	36443	104	1,8,7	29796	149	1,7,6	24919
15	1,6,2	39541	60	1,8,4	36419	105	1,9,9	29723	150	2,5,17	24902
16	2,5,2	39479	61	2,7,5	36376	106	1,8,8	29669	151	1,6,6	24789
17	1,4,7	39475	62	1,4,4	36293	107	2,9,8	29639	152	1,2,17	24730
18	2,5,1	39263	63	1,4,5	36237	108	1,1,7	29542	153	2,6,17	24651
19	1,7,2	39245	64	1,9,4	36206	109	1,9,6	29414	154	1,1,6	24518
20	2,9,1	39170	65	2,3,9	36203	110	2,4,17	29033	155	1,7,17	24492
21	2,9,2	39111	66	1,3,5	36167	111	2,5,7	28885	156	1,1,17	24425
22	2,6,1	39050	67	1,6,4	36119	112	2,5,9	28876	157	1,6,8	24408
23	2,8,2	39050	68	1,7,4	35971	113	2,7,9	28567	158	1,8,10	24158
24	2,7,1	39019	69	1,2,4	35896	114	2,5,8	28565	159	1,5,17	23893
25	2,4,4	38992	70	1,5,5	35778	115	2,7,7	28359	160	2,5,10	23761
26	2,4,7	38812	71	1,1,4	35776	116	1,9,8	28280	161	1,6,17	23727
27	2,8,1	38597	72	1,3,4	35737	117	2,6,9	28184	162	1,5,10	22518
28	2,5,4	38544	73	1,3,8	35646	118	1,5,9	28117	163	2,3,15	21063
29	2,2,4	38462	74	1,1,5	35626	119	2,6,7	28015	164	2,9,10	20932
30	2,3,4	38454	75	1,2,8	35623	120	1,5,7	27609	165	2,2,15	20809
31	2,4,5	38381	76	2,1,8	35524	121	2,7,8	27590	166	2,1,15	20546
32	2,1,4	38349	77	1,2,5	35459	122	2,6,6	27276	167	1,9,10	19109
33	2,3,5	38315	78	1,8,5	35237	123	2,4,6	27223	168	2,4,10	18754
34	1,5,2	38151	79	1,9,5	35020	124	2,3,6	27153	169	1,3,15	18735
35	2,8,4	38050	80	1,6,5	34942	125	1,4,17	27127	170	1,2,15	18453
36	1,4,1	38020	81	1,2,9	34860	126	1,5,8	27087	171	1,1,15	18352
37	1,9,2	37996	82	1,7,5	34817	127	1,7,7	27048	172	2,3,16	18256
38	1,8,2	37895	83	1,3,9	34435	128	2,8,17	26967	173	2,2,16	17895
39	2,1,5	37887	84	2,1,9	34410	129	2,7,6	26966	174	2,1,16	17538
40	2,9,4	37834	85	1,1,8	33839	130	1,6,7	26778	175	2,1,12	17365
41	2,2,5	37760	86	1,1,9	32966	131	1,7,9	26733	176	2,3,12	17199
42	1,5,1	37661	87	2,5,6	32892	132	2,2,6	26696	177	2,4,14	16963
43	1,2,1	37649	88	2,8,6	32877	133	2,9,17	26437	178	2,4,12	16735
44	2,7,4	37587	89	2,3,7	32030	134	1,6,9	26396	179	2,1,3	16525
45	2,6,4	37572	90	2,2,7	32000	135	2,6,8	26290	180	2,3,10	16455

Tablica A.9 (c.d.)

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
181	2,6,10	16452	226	2,5,11	12848	271	2,9,14	8662
182	2,7,10	16417	227	2,7,12	12677	272	1,8,16	8644
183	2,2,12	16354	228	2,8,15	12566	273	2,7,13	8624
184	2,1,10	16281	229	2,5,12	12487	274	2,6,13	8593
185	1,4,10	16231	230	1,6,15	12360	275	1,6,16	8576
186	2,9,3	15878	231	2,6,12	12263	276	2,5,3	8417
187	2,3,11	15874	232	1,1,11	11920	277	1,7,13	8312
188	2,2,10	15851	233	2,1,13	11918	278	1,8,14	8285
189	1,4,14	15810	234	2,3,13	11873	279	2,5,15	8278
190	2,1,11	15801	235	1,3,11	11857	280	1,6,13	8111
191	2,4,11	15774	236	1,4,11	11711	281	2,5,13	7832
192	1,1,3	15428	237	2,4,13	11661	282	1,3,14	7562
193	2,3,3	15364	238	2,2,13	11607	283	1,2,14	7261
194	2,2,11	15272	239	1,4,9	11597	284	1,1,14	7250
195	2,9,15	15248	240	1,2,11	11179	285	1,9,13	7234
196	1,3,16	15025	241	2,8,16	10925	286	1,8,13	7230
197	1,6,10	14858	242	1,8,15	10918	287	2,5,16	6975
198	1,2,16	14828	243	2,9,16	10913	288	1,5,3	6855
199	2,7,15	14798	244	1,8,11	10862	289	2,7,14	6740
200	2,2,3	14765	245	2,3,14	10838	290	1,5,15	6716
201	1,9,3	14596	246	1,8,12	10808	291	1,9,14	6604
202	1,1,16	14490	247	1,9,11	10745	292	2,5,14	6345
203	1,3,3	14476	248	2,2,14	10723	293	1,5,13	6184
204	1,7,10	14418	249	1,9,12	10690	294	2,6,14	5844
205	1,3,10	14340	250	2,8,3	10566	295	2,4,16	5782
206	2,7,3	14171	251	2,8,14	10533	296	2,4,8	5550
207	1,1,10	14159	252	2,4,9	10497	297	1,5,16	4995
208	1,1,12	13984	253	2,1,14	10412	298	1,4,16	4983
209	1,7,15	13904	254	1,6,11	10374	299	1,4,8	4960
210	1,2,10	13750	255	1,5,12	10326	300	1,7,14	4865
211	1,3,12	13746	256	1,7,11	10304	301	1,5,14	4604
212	2,6,3	13677	257	1,5,11	10301	302	1,6,14	4135
213	1,7,3	13550	258	1,7,12	10273	303	2,4,15	3784
214	2,8,11	13462	259	1,6,12	10110	304	2,4,3	3143
215	1,2,3	13419	260	2,7,16	9962	305	1,4,15	2590
216	2,9,11	13348	261	1,7,16	9865	306	1,4,3	1687
217	1,4,12	13301	262	2,9,13	9846			
218	2,8,12	13241	263	1,3,13	9794			
219	2,6,15	13153	264	1,1,13	9652			
220	2,9,12	13101	265	1,4,13	9578			
221	1,2,12	13047	266	1,2,13	9383			
222	1,6,3	12974	267	1,8,3	9112			
223	1,9,15	12961	268	1,9,16	8997			
224	2,7,11	12916	269	2,6,16	8714			
225	2,6,11	12912	270	2,8,13	8667			

Źródło: opracowanie własne.

Tablica A.10

Wskaźniki dynamiki rang ścieżek WAP dla p=306 - zestaw danych nr 2

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	2,4,2	-	46	2,3,8	0,999	91	2,8,9	0,997	136	1,8,17	0,996
2	2,2,2	0,987	47	1,9,1	0,997	92	2,9,7	0,989	137	2,3,17	0,999
3	2,3,2	0,998	48	2,2,8	0,998	93	1,8,6	0,998	138	1,4,6	0,991
4	2,1,2	0,997	49	1,3,1	1,000	94	1,5,6	1,000	139	2,1,6	1,000
5	2,6,2	0,974	50	2,5,5	0,999	95	2,8,7	0,998	140	2,2,17	0,999
6	2,7,2	0,995	51	1,7,1	0,998	96	1,8,9	0,998	141	1,3,6	0,999
7	1,4,2	0,992	52	1,1,1	0,999	97	2,9,9	0,988	142	1,7,8	0,996
8	1,2,2	0,996	53	1,6,1	0,999	98	2,1,7	0,997	143	2,7,17	0,992
9	2,4,1	0,999	54	1,5,4	0,997	99	2,8,8	0,998	144	1,2,6	0,999
10	1,1,2	0,994	55	1,8,1	0,995	100	2,9,6	0,996	145	2,8,10	0,999
11	2,2,1	0,996	56	2,8,5	0,997	101	1,3,7	0,992	146	2,1,17	0,999
12	1,3,2	0,999	57	2,2,9	0,998	102	1,2,7	0,997	147	1,3,17	0,981
13	2,3,1	0,994	58	2,9,5	0,997	103	1,9,7	0,984	148	1,9,17	0,997
14	2,1,1	0,998	59	2,6,5	0,996	104	1,8,7	0,997	149	1,7,6	0,999
15	1,6,2	1,000	60	1,8,4	0,999	105	1,9,9	0,998	150	2,5,17	0,999
16	2,5,2	0,998	61	2,7,5	0,999	106	1,8,8	0,998	151	1,6,6	0,995
17	1,4,7	1,000	62	1,4,4	0,998	107	2,9,8	0,999	152	1,2,17	0,998
18	2,5,1	0,995	63	1,4,5	0,998	108	1,1,7	0,997	153	2,6,17	0,997
19	1,7,2	1,000	64	1,9,4	0,999	109	1,9,6	0,996	154	1,1,6	0,995
20	2,9,1	0,998	65	2,3,9	1,000	110	2,4,17	0,987	155	1,7,17	0,999
21	2,9,2	0,998	66	1,3,5	0,999	111	2,5,7	0,995	156	1,1,17	0,997
22	2,6,1	0,998	67	1,6,4	0,999	112	2,5,9	1,000	157	1,6,8	0,999
23	2,8,2	1,000	68	1,7,4	0,996	113	2,7,9	0,989	158	1,8,10	0,990
24	2,7,1	0,999	69	1,2,4	0,998	114	2,5,8	1,000	159	1,5,17	0,989
25	2,4,4	0,999	70	1,5,5	0,997	115	2,7,7	0,993	160	2,5,10	0,994
26	2,4,7	0,995	71	1,1,4	1,000	116	1,9,8	0,997	161	1,6,17	0,999
27	2,8,1	0,994	72	1,3,4	0,999	117	2,6,9	0,997	162	1,5,10	0,949
28	2,5,4	0,999	73	1,3,8	0,997	118	1,5,9	0,998	163	2,3,15	0,935
29	2,2,4	0,998	74	1,1,5	0,999	119	2,6,7	0,996	164	2,9,10	0,994
30	2,3,4	1,000	75	1,2,8	1,000	120	1,5,7	0,986	165	2,2,15	0,994
31	2,4,5	0,998	76	2,1,8	0,997	121	2,7,8	0,999	166	2,1,15	0,987
32	2,1,4	0,999	77	1,2,5	0,998	122	2,6,6	0,989	167	1,9,10	0,930
33	2,3,5	0,999	78	1,8,5	0,994	123	2,4,6	0,998	168	2,4,10	0,981
34	1,5,2	0,996	79	1,9,5	0,994	124	2,3,6	0,997	169	1,3,15	0,999
35	2,8,4	0,997	80	1,6,5	0,998	125	1,4,17	0,999	170	1,2,15	0,985
36	1,4,1	0,999	81	1,2,9	0,998	126	1,5,8	0,999	171	1,1,15	0,995
37	1,9,2	0,999	82	1,7,5	0,999	127	1,7,7	0,999	172	2,3,16	0,995
38	1,8,2	0,997	83	1,3,9	0,989	128	2,8,17	0,997	173	2,2,16	0,980
39	2,1,5	1,000	84	2,1,9	0,999	129	2,7,6	1,000	174	2,1,16	0,980
40	2,9,4	0,999	85	1,1,8	0,983	130	1,6,7	0,993	175	2,1,12	0,990
41	2,2,5	0,998	86	1,1,9	0,974	131	1,7,9	0,998	176	2,3,12	0,990
42	1,5,1	0,997	87	2,5,6	0,998	132	2,2,6	0,999	177	2,4,14	0,986
43	1,2,1	1,000	88	2,8,6	1,000	133	2,9,17	0,990	178	2,4,12	0,987
44	2,7,4	0,998	89	2,3,7	0,974	134	1,6,9	0,998	179	2,1,3	0,987
45	2,6,4	1,000	90	2,2,7	0,999	135	2,6,8	0,996	180	2,3,10	0,996

Tablica A.10 (c.d.)

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
181	2,6,10	1,000	226	2,5,11	0,995	271	2,9,14	0,999
182	2,7,10	0,998	227	2,7,12	0,987	272	1,8,16	0,998
183	2,2,12	0,996	228	2,8,15	0,991	273	2,7,13	0,998
184	2,1,10	0,996	229	2,5,12	0,994	274	2,6,13	0,996
185	1,4,10	0,997	230	1,6,15	0,990	275	1,6,16	0,998
186	2,9,3	0,978	231	2,6,12	0,992	276	2,5,3	0,981
187	2,3,11	1,000	232	1,1,11	0,972	277	1,7,13	0,988
188	2,2,10	0,999	233	2,1,13	1,000	278	1,8,14	0,997
189	1,4,14	0,997	234	2,3,13	0,996	279	2,5,15	0,999
190	2,1,11	0,999	235	1,3,11	0,999	280	1,6,13	0,980
191	2,4,11	0,998	236	1,4,11	0,988	281	2,5,13	0,966
192	1,1,3	0,978	237	2,4,13	0,996	282	1,3,14	0,966
193	2,3,3	0,996	238	2,2,13	0,995	283	1,2,14	0,960
194	2,2,11	0,994	239	1,4,9	0,999	284	1,1,14	0,998
195	2,9,15	0,998	240	1,2,11	0,964	285	1,9,13	0,998
196	1,3,16	0,985	241	2,8,16	0,977	286	1,8,13	0,999
197	1,6,10	0,989	242	1,8,15	0,999	287	2,5,16	0,965
198	1,2,16	0,998	243	2,9,16	1,000	288	1,5,3	0,983
199	2,7,15	0,998	244	1,8,11	0,995	289	2,7,14	0,983
200	2,2,3	0,998	245	2,3,14	0,998	290	1,5,15	0,996
201	1,9,3	0,989	246	1,8,12	0,997	291	1,9,14	0,983
202	1,1,16	0,993	247	1,9,11	0,994	292	2,5,14	0,961
203	1,3,3	0,999	248	2,2,14	0,998	293	1,5,13	0,975
204	1,7,10	0,996	249	1,9,12	0,997	294	2,6,14	0,945
205	1,3,10	0,995	250	2,8,3	0,988	295	2,4,16	0,989
206	2,7,3	0,988	251	2,8,14	0,997	296	2,4,8	0,960
207	1,1,10	0,999	252	2,4,9	0,997	297	1,5,16	0,900
208	1,1,12	0,988	253	2,1,14	0,992	298	1,4,16	0,998
209	1,7,15	0,994	254	1,6,11	0,996	299	1,4,8	0,995
210	1,2,10	0,989	255	1,5,12	0,995	300	1,7,14	0,981
211	1,3,12	1,000	256	1,7,11	0,998	301	1,5,14	0,946
212	2,6,3	0,995	257	1,5,11	1,000	302	1,6,14	0,898
213	1,7,3	0,991	258	1,7,12	0,997	303	2,4,15	0,915
214	2,8,11	0,994	259	1,6,12	0,984	304	2,4,3	0,831
215	1,2,3	0,997	260	2,7,16	0,985	305	1,4,15	0,824
216	2,9,11	0,995	261	1,7,16	0,990	306	1,4,3	0,651
217	1,4,12	0,996	262	2,9,13	0,998			
218	2,8,12	0,995	263	1,3,13	0,995			
219	2,6,15	0,993	264	1,1,13	0,986			
220	2,9,12	0,996	265	1,4,13	0,992			
221	1,2,12	0,996	266	1,2,13	0,980			
222	1,6,3	0,994	267	1,8,3	0,971			
223	1,9,15	0,999	268	1,9,16	0,987			
224	2,7,11	0,997	269	2,6,16	0,969			
225	2,6,11	1,000	270	2,8,13	0,995			

Źródło: opracowanie własne.

Tablica A.11

Uporządkowanie ścieżek WAP dla p=10 - zestaw danych nr 2

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	828	46	2,6,1	28	91	2,6,4	3
2	2,4,7	757	47	2,9,5	27	92	2,7,4	3
3	2,4,2	457	48	2,7,5	26	93	2,8,7	3
4	2,2,2	424	49	2,8,5	23	94	1,3,9	2
5	1,7,2	379	50	2,1,4	22	95	2,8,4	2
6	1,2,2	377	51	1,2,5	21	96	1,7,6	1
7	2,3,2	374	52	2,6,5	21	97	2,5,9	1
8	1,6,2	360	53	1,2,1	20	pozostałe		0
9	2,1,2	353	54	2,3,5	20			
10	1,4,2	286	55	2,5,5	20			
11	1,3,2	282	56	1,3,8	19			
12	1,1,2	279	57	1,6,4	18			
13	2,7,2	246	58	1,2,9	16			
14	2,6,2	243	59	1,7,4	16			
15	2,4,1	184	60	2,1,8	16			
16	1,9,2	115	61	2,7,1	16			
17	2,1,1	107	62	1,3,5	15			
18	1,4,9	104	63	1,4,5	15			
19	1,8,2	98	64	1,5,4	15			
20	2,9,2	83	65	2,3,4	14			
21	2,3,1	82	66	1,8,4	13			
22	2,3,16	80	67	2,2,9	11			
23	1,5,2	71	68	2,4,3	10			
24	2,3,17	70	69	2,5,1	10			
25	2,4,9	69	70	1,1,5	9			
26	2,8,2	69	71	1,1,8	9			
27	2,2,1	67	72	1,2,4	9			
28	1,7,1	66	73	1,9,4	8			
29	1,6,1	62	74	2,9,1	8			
30	2,2,8	62	75	1,1,1	7			
31	2,3,15	60	76	1,3,1	7			
32	2,5,2	59	77	2,2,4	7			
33	2,1,5	50	78	1,9,5	6			
34	2,4,6	50	79	2,4,17	6			
35	2,3,8	46	80	1,4,1	5			
36	2,4,4	45	81	1,5,6	5			
37	2,4,5	44	82	1,8,5	5			
38	1,9,1	43	83	2,4,8	5			
39	1,2,8	40	84	2,5,4	5			
40	1,6,5	40	85	2,8,1	5			
41	2,2,5	39	86	2,8,9	5			
42	1,5,5	38	87	1,7,5	4			
43	1,5,1	32	88	2,9,4	4			
44	1,8,1	30	89	1,4,17	3			
45	1,4,8	28	90	2,3,9	3			

Źródło: opracowanie własne.

Tablica A.12

Uporządkowanie ścieżek WAP dla p=5 - zestaw danych nr 2

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	369	46	1,2,9	3
2	2,4,7	315	47	2,1,4	3
3	1,6,2	117	48	2,1,8	3
4	1,7,2	116	49	2,5,5	3
5	2,4,2	115	50	2,6,1	3
6	2,2,2	108	51	2,6,5	3
7	1,2,2	104	52	2,7,5	3
8	2,1,2	86	53	1,2,5	2
9	2,3,2	78	54	1,3,5	2
10	2,7,2	71	55	1,5,4	2
11	2,4,1	66	56	1,7,4	2
12	1,3,2	57	57	2,4,4	2
13	1,1,2	56	58	2,4,5	2
14	1,4,2	54	59	2,7,1	2
15	2,6,2	52	60	2,9,5	2
16	1,4,9	37	61	1,1,8	1
17	1,9,2	36	62	1,4,5	1
18	2,3,16	30	63	1,6,4	1
19	2,4,9	23	64	1,8,4	1
20	1,8,2	21	65	2,3,4	1
21	2,1,1	20	66	2,3,5	1
22	2,3,17	20	67	2,8,5	1
23	2,2,8	19	68	2,9,1	1
24	1,6,1	18	pozostałe		0
25	2,9,2	18			
26	1,7,1	17			
27	1,5,2	16			
28	2,3,1	16			
29	2,1,5	15			
30	2,8,2	15			
31	2,3,8	12			
32	2,5,2	11			
33	1,2,8	10			
34	1,6,5	10			
35	1,9,1	10			
36	2,3,15	10			
37	2,2,5	9			
38	1,4,8	8			
39	1,5,5	8			
40	2,2,1	8			
41	1,2,1	6			
42	1,3,8	5			
43	1,8,1	5			
44	1,2,4	4			
45	1,5,1	4			

Źródło: opracowanie własne.

Tablica A.13

Uporządkowanie ścieżek WAP dla p=3 - zestaw danych nr 2

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	200	46	2,7,5	1
2	2,4,7	170	pozostałe		0
3	1,6,2	48			
4	1,7,2	46			
5	1,2,2	39			
6	2,4,2	38			
7	2,2,2	36			
8	2,4,1	36			
9	2,7,2	30			
10	2,1,2	27			
11	2,3,2	19			
12	1,1,2	18			
13	1,3,2	16			
14	1,4,9	16			
15	2,6,2	16			
16	1,4,2	15			
17	1,9,2	12			
18	2,3,16	10			
19	2,2,8	9			
20	1,7,1	8			
21	2,4,9	8			
22	2,1,1	7			
23	2,1,5	7			
24	1,5,2	6			
25	1,6,1	6			
26	1,8,2	6			
27	2,5,2	6			
28	2,9,2	6			
29	2,3,1	5			
30	2,3,8	5			
31	1,9,1	4			
32	1,2,1	3			
33	1,2,8	3			
34	2,8,2	3			
35	1,2,4	2			
36	1,3,8	2			
37	1,6,5	2			
38	2,2,5	2			
39	1,4,8	1			
40	1,5,1	1			
41	1,8,1	1			
42	2,1,4	1			
43	2,1,8	1			
44	2,2,1	1			
45	2,5,5	1			

Źródło: opracowanie własne.

Tablica A.14

Uporządkowanie ścieżek WAP dla p=306 - zestaw danych nr 3

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	2,2,2	41122	46	1,7,1	37277	91	2,9,7	31459	136	1,6,9	27064
2	2,4,2	40779	47	2,8,5	37186	92	2,9,9	31265	137	1,2,6	26882
3	2,7,2	40291	48	1,8,1	37111	93	2,1,8	30796	138	1,5,8	26559
4	2,3,2	40159	49	1,9,2	37094	94	1,2,7	30792	139	1,6,6	26462
5	2,6,2	39806	50	2,9,5	36982	95	2,8,8	30680	140	2,8,17	26262
6	2,4,1	39776	51	1,1,2	36952	96	1,3,7	30669	141	1,7,6	26215
7	2,9,2	39605	52	2,1,5	36848	97	2,8,7	30648	142	2,9,17	26047
8	1,2,2	39411	53	1,4,4	36780	98	1,8,9	30540	143	2,7,17	26008
9	2,2,1	39379	54	1,6,4	36765	99	2,1,9	30509	144	1,1,6	25967
10	2,5,1	39319	55	2,7,5	36765	100	2,1,7	30407	145	2,8,10	25931
11	2,8,2	39234	56	1,8,4	36756	101	1,9,7	30297	146	2,2,17	25844
12	2,5,2	39216	57	1,1,1	36630	102	2,9,8	30188	147	1,6,8	25734
13	2,1,2	39157	58	1,9,4	36560	103	1,9,6	30034	148	2,3,17	25705
14	2,3,1	39116	59	2,6,5	36518	104	2,4,17	30025	149	1,2,17	25397
15	1,4,2	39041	60	1,2,4	36496	105	2,7,9	29985	150	1,8,17	25212
16	2,6,1	38830	61	1,7,4	36318	106	1,8,7	29808	151	1,3,17	25158
17	2,4,4	38775	62	1,4,5	36308	107	1,8,8	29548	152	1,7,17	25108
18	2,1,1	38721	63	1,3,5	36264	108	1,4,17	29381	153	1,9,17	25000
19	2,5,4	38711	64	1,5,5	36234	109	2,7,8	29367	154	2,6,17	24844
20	2,9,1	38710	65	1,3,4	36202	110	2,7,7	29274	155	1,8,10	24578
21	2,7,1	38506	66	1,1,4	35733	111	1,1,8	29243	156	2,5,17	24388
22	1,7,2	38505	67	1,8,5	35674	112	1,1,7	29183	157	1,6,17	24159
23	2,4,5	38458	68	1,2,5	35614	113	1,9,9	29092	158	2,1,17	24117
24	2,3,5	38406	69	1,9,5	35471	114	1,1,9	28936	159	2,5,10	23933
25	1,3,2	38397	70	1,6,5	35372	115	2,6,9	28819	160	1,5,17	23474
26	2,2,4	38329	71	2,2,9	35238	116	2,5,9	28804	161	1,1,17	23237
27	2,8,1	38320	72	1,7,5	35237	117	2,6,7	28793	162	1,5,10	22386
28	2,3,4	38231	73	2,2,8	35198	118	2,4,6	28786	163	2,9,10	21180
29	1,5,1	38226	74	1,1,5	34937	119	2,3,6	28714	164	2,4,10	20432
30	1,6,2	38090	75	2,3,8	33887	120	1,9,8	28665	165	1,9,10	19487
31	1,5,2	38024	76	1,2,8	33819	121	2,5,7	28575	166	2,2,15	19090
32	1,4,1	38021	77	2,3,9	33663	122	2,2,6	28311	167	2,3,15	18851
33	1,8,2	38018	78	2,8,6	33563	123	1,7,7	28211	168	1,4,10	18591
34	2,1,4	37957	79	1,2,9	33536	124	1,7,9	28139	169	2,1,12	18399
35	2,8,4	37940	80	2,5,6	33379	125	2,6,8	27991	170	2,1,15	18390
36	2,2,5	37828	81	1,4,7	33146	126	2,6,6	27981	171	2,7,10	17681
37	2,9,4	37751	82	1,3,8	32540	127	2,7,6	27914	172	2,3,10	17350
38	2,6,4	37747	83	1,3,9	32498	128	1,5,7	27912	173	2,1,10	17263
39	1,5,4	37745	84	1,5,6	32493	129	1,6,7	27806	174	1,2,15	17240
40	1,2,1	37623	85	2,4,7	32403	130	1,5,9	27538	175	2,6,10	17220
41	1,6,1	37567	86	1,8,6	32298	131	2,5,8	27515	176	2,3,12	17117
42	1,9,1	37496	87	2,2,7	32216	132	2,1,6	27439	177	2,4,12	16651
43	2,7,4	37483	88	2,3,7	31838	133	1,4,6	27328	178	2,2,10	16637
44	2,5,5	37386	89	2,8,9	31635	134	1,3,6	27266	179	2,1,11	16475
45	1,3,1	37359	90	2,9,6	31525	135	1,7,8	27239	180	1,3,15	16419

Tablica A.14 (c.d.)

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
181	2,2,12	16378	226	2,7,11	13038	271	1,2,13	9243
182	2,2,16	16296	227	1,1,16	12889	272	2,5,15	9102
183	1,1,15	16243	228	2,6,15	12845	273	1,3,13	9010
184	2,2,3	16171	229	1,4,9	12803	274	2,6,13	9000
185	2,7,15	16092	230	1,9,3	12775	275	2,6,16	8919
186	2,3,3	15791	231	1,3,11	12458	276	1,4,13	8794
187	2,3,16	15783	232	2,9,16	12363	277	1,1,14	8427
188	2,3,11	15767	233	1,4,11	12328	278	2,5,13	8341
189	2,4,11	15641	234	1,8,15	12226	279	1,6,16	8277
190	2,9,15	15609	235	2,6,3	12018	280	1,8,3	8108
191	1,1,12	15549	236	1,6,15	11845	281	1,3,14	7938
192	1,7,10	15478	237	1,8,11	11803	282	1,2,14	7922
193	2,1,3	15376	238	1,2,11	11769	283	1,9,13	7882
194	1,2,3	15319	239	1,9,11	11708	284	1,5,15	7818
195	1,1,10	15284	240	2,4,9	11652	285	1,8,13	7782
196	1,6,10	15280	241	2,8,16	11598	286	2,4,8	7782
197	1,3,10	15151	242	1,8,12	11538	287	1,6,13	7759
198	1,7,15	15149	243	2,7,16	11505	288	1,9,14	7757
199	2,1,16	14967	244	1,9,12	11407	289	2,5,16	7695
200	2,2,11	14780	245	2,1,13	11384	290	2,7,14	7588
201	2,7,3	14643	246	2,8,14	11287	291	1,4,8	7373
202	1,3,3	14493	247	2,1,14	11283	292	1,5,13	7113
203	1,2,10	14345	248	1,7,11	11244	293	2,5,14	7016
204	2,4,14	14215	249	1,5,12	11229	294	2,5,3	6803
205	1,3,12	14149	250	1,6,11	11216	295	2,6,14	6422
206	2,8,12	14100	251	1,5,11	11140	296	2,4,16	6255
207	2,5,12	14016	252	1,6,12	11037	297	1,5,16	6190
208	2,9,12	13933	253	1,7,12	10995	298	1,7,14	6082
209	2,9,3	13887	254	2,2,13	10985	299	1,4,16	5725
210	2,8,15	13867	255	1,6,3	10979	300	1,5,14	5472
211	1,2,16	13846	256	2,3,13	10913	301	1,6,14	5013
212	2,6,12	13810	257	2,3,14	10805	302	1,5,3	4851
213	1,4,14	13713	258	1,7,16	10753	303	2,4,15	4376
214	1,4,12	13697	259	2,4,13	10711	304	1,4,15	3340
215	2,8,11	13622	260	2,2,14	10607	305	2,4,3	3151
216	1,1,3	13564	261	2,7,13	10521	306	1,4,3	1679
217	2,7,12	13557	262	2,9,13	10428			
218	2,9,11	13510	263	2,8,3	9841			
219	1,2,12	13329	264	1,8,16	9831			
220	1,9,15	13320	265	1,8,14	9592			
221	1,7,3	13307	266	1,7,13	9480			
222	2,6,11	13274	267	1,1,13	9419			
223	1,3,16	13262	268	1,9,16	9371			
224	2,5,11	13192	269	2,8,13	9353			
225	1,1,11	13048	270	2,9,14	9324			

Źródło: opracowanie własne.

Tablica A.15

Wskaźniki dynamiki rang ścieżek WAP dla p=306 - zestaw danych nr 3

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
1	2,2,2	-	46	1,7,1	0,998	91	2,9,7	0,998	136	1,6,9	0,994
2	2,4,2	0,992	47	2,8,5	0,998	92	2,9,9	0,994	137	1,2,6	0,993
3	2,7,2	0,988	48	1,8,1	0,998	93	2,1,8	0,985	138	1,5,8	0,988
4	2,3,2	0,997	49	1,9,2	1,000	94	1,2,7	1,000	139	1,6,6	0,996
5	2,6,2	0,991	50	2,9,5	0,997	95	2,8,8	0,996	140	2,8,17	0,992
6	2,4,1	0,999	51	1,1,2	0,999	96	1,3,7	1,000	141	1,7,6	0,998
7	2,9,2	0,996	52	2,1,5	0,997	97	2,8,7	0,999	142	2,9,17	0,994
8	1,2,2	0,995	53	1,4,4	0,998	98	1,8,9	0,996	143	2,7,17	0,999
9	2,2,1	0,999	54	1,6,4	1,000	99	2,1,9	0,999	144	1,1,6	0,998
10	2,5,1	0,998	55	2,7,5	1,000	100	2,1,7	0,997	145	2,8,10	0,999
11	2,8,2	0,998	56	1,8,4	1,000	101	1,9,7	0,996	146	2,2,17	0,997
12	2,5,2	1,000	57	1,1,1	0,997	102	2,9,8	0,996	147	1,6,8	0,996
13	2,1,2	0,998	58	1,9,4	0,998	103	1,9,6	0,995	148	2,3,17	0,999
14	2,3,1	0,999	59	2,6,5	0,999	104	2,4,17	1,000	149	1,2,17	0,988
15	1,4,2	0,998	60	1,2,4	0,999	105	2,7,9	0,999	150	1,8,17	0,993
16	2,6,1	0,995	61	1,7,4	0,995	106	1,8,7	0,994	151	1,3,17	0,998
17	2,4,4	0,999	62	1,4,5	1,000	107	1,8,8	0,991	152	1,7,17	0,998
18	2,1,1	0,999	63	1,3,5	0,999	108	1,4,17	0,994	153	1,9,17	0,996
19	2,5,4	1,000	64	1,5,5	0,999	109	2,7,8	1,000	154	2,6,17	0,994
20	2,9,1	1,000	65	1,3,4	0,999	110	2,7,7	0,997	155	1,8,10	0,989
21	2,7,1	0,995	66	1,1,4	0,987	111	1,1,8	0,999	156	2,5,17	0,992
22	1,7,2	1,000	67	1,8,5	0,998	112	1,1,7	0,998	157	1,6,17	0,991
23	2,4,5	0,999	68	1,2,5	0,998	113	1,9,9	0,997	158	2,1,17	0,998
24	2,3,5	0,999	69	1,9,5	0,996	114	1,1,9	0,995	159	2,5,10	0,992
25	1,3,2	1,000	70	1,6,5	0,997	115	2,6,9	0,996	160	1,5,17	0,981
26	2,2,4	0,998	71	2,2,9	0,996	116	2,5,9	0,999	161	1,1,17	0,990
27	2,8,1	1,000	72	1,7,5	1,000	117	2,6,7	1,000	162	1,5,10	0,963
28	2,3,4	0,998	73	2,2,8	0,999	118	2,4,6	1,000	163	2,9,10	0,946
29	1,5,1	1,000	74	1,1,5	0,993	119	2,3,6	0,997	164	2,4,10	0,965
30	1,6,2	0,996	75	2,3,8	0,970	120	1,9,8	0,998	165	1,9,10	0,954
31	1,5,2	0,998	76	1,2,8	0,998	121	2,5,7	0,997	166	2,2,15	0,980
32	1,4,1	1,000	77	2,3,9	0,995	122	2,2,6	0,991	167	2,3,15	0,987
33	1,8,2	1,000	78	2,8,6	0,997	123	1,7,7	0,996	168	1,4,10	0,986
34	2,1,4	0,998	79	1,2,9	0,999	124	1,7,9	0,997	169	2,1,12	0,990
35	2,8,4	1,000	80	2,5,6	0,995	125	2,6,8	0,995	170	2,1,15	1,000
36	2,2,5	0,997	81	1,4,7	0,993	126	2,6,6	1,000	171	2,7,10	0,961
37	2,9,4	0,998	82	1,3,8	0,982	127	2,7,6	0,998	172	2,3,10	0,981
38	2,6,4	1,000	83	1,3,9	0,999	128	1,5,7	1,000	173	2,1,10	0,995
39	1,5,4	1,000	84	1,5,6	1,000	129	1,6,7	0,996	174	1,2,15	0,999
40	1,2,1	0,997	85	2,4,7	0,997	130	1,5,9	0,990	175	2,6,10	0,999
41	1,6,1	0,999	86	1,8,6	0,997	131	2,5,8	0,999	176	2,3,12	0,994
42	1,9,1	0,998	87	2,2,7	0,997	132	2,1,6	0,997	177	2,4,12	0,973
43	2,7,4	1,000	88	2,3,7	0,988	133	1,4,6	0,996	178	2,2,10	0,999
44	2,5,5	0,997	89	2,8,9	0,994	134	1,3,6	0,998	179	2,1,11	0,990
45	1,3,1	0,999	90	2,9,6	0,997	135	1,7,8	0,999	180	1,3,15	0,997

Tablica A.15 (c.d.)

Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}	Pozycja	Ścieżka	r_i/r_{i-1}
181	2,2,12	0,998	226	2,7,11	0,999	271	1,2,13	0,991
182	2,2,16	0,995	227	1,1,16	0,989	272	2,5,15	0,985
183	1,1,15	0,997	228	2,6,15	0,997	273	1,3,13	0,990
184	2,2,3	0,996	229	1,4,9	0,997	274	2,6,13	0,999
185	2,7,15	0,995	230	1,9,3	0,998	275	2,6,16	0,991
186	2,3,3	0,981	231	1,3,11	0,975	276	1,4,13	0,986
187	2,3,16	0,999	232	2,9,16	0,992	277	1,1,14	0,958
188	2,3,11	0,999	233	1,4,11	0,997	278	2,5,13	0,990
189	2,4,11	0,992	234	1,8,15	0,992	279	1,6,16	0,992
190	2,9,15	0,998	235	2,6,3	0,983	280	1,8,3	0,980
191	1,1,12	0,996	236	1,6,15	0,986	281	1,3,14	0,979
192	1,7,10	0,995	237	1,8,11	0,996	282	1,2,14	0,998
193	2,1,3	0,993	238	1,2,11	0,997	283	1,9,13	0,995
194	1,2,3	0,996	239	1,9,11	0,995	284	1,5,15	0,992
195	1,1,10	0,998	240	2,4,9	0,995	285	1,8,13	0,995
196	1,6,10	1,000	241	2,8,16	0,995	286	2,4,8	1,000
197	1,3,10	0,992	242	1,8,12	0,995	287	1,6,13	0,997
198	1,7,15	1,000	243	2,7,16	0,997	288	1,9,14	1,000
199	2,1,16	0,988	244	1,9,12	0,991	289	2,5,16	0,992
200	2,2,11	0,988	245	2,1,13	0,998	290	2,7,14	0,986
201	2,7,3	0,991	246	2,8,14	0,991	291	1,4,8	0,972
202	1,3,3	0,990	247	2,1,14	1,000	292	1,5,13	0,965
203	1,2,10	0,990	248	1,7,11	0,997	293	2,5,14	0,986
204	2,4,14	0,991	249	1,5,12	0,999	294	2,5,3	0,970
205	1,3,12	0,995	250	1,6,11	0,999	295	2,6,14	0,944
206	2,8,12	0,997	251	1,5,11	0,993	296	2,4,16	0,974
207	2,5,12	0,994	252	1,6,12	0,991	297	1,5,16	0,990
208	2,9,12	0,994	253	1,7,12	0,996	298	1,7,14	0,983
209	2,9,3	0,997	254	2,2,13	0,999	299	1,4,16	0,941
210	2,8,15	0,999	255	1,6,3	0,999	300	1,5,14	0,956
211	1,2,16	0,998	256	2,3,13	0,994	301	1,6,14	0,916
212	2,6,12	0,997	257	2,3,14	0,990	302	1,5,3	0,968
213	1,4,14	0,993	258	1,7,16	0,995	303	2,4,15	0,902
214	1,4,12	0,999	259	2,4,13	0,996	304	1,4,15	0,763
215	2,8,11	0,995	260	2,2,14	0,990	305	2,4,3	0,943
216	1,1,3	0,996	261	2,7,13	0,992	306	1,4,3	0,533
217	2,7,12	0,999	262	2,9,13	0,991			
218	2,9,11	0,997	263	2,8,3	0,944			
219	1,2,12	0,987	264	1,8,16	0,999			
220	1,9,15	0,999	265	1,8,14	0,976			
221	1,7,3	0,999	266	1,7,13	0,988			
222	2,6,11	0,998	267	1,1,13	0,994			
223	1,3,16	0,999	268	1,9,16	0,995			
224	2,5,11	0,995	269	2,8,13	0,998			
225	1,1,11	0,989	270	2,9,14	0,997			

Źródło: opracowanie własne.

Tablica A.16

Uporządkowanie ścieżek WAP dla p=10 - zestaw danych nr 3

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	767	46	1,5,1	47	91	1,8,7	4
2	2,4,7	636	47	2,1,4	45	92	2,8,4	4
3	2,2,2	332	48	2,6,1	44	93	2,8,6	4
4	1,6,2	310	49	1,1,1	40	94	1,2,4	3
5	2,4,2	301	50	1,9,1	39	95	1,5,6	3
6	1,2,2	297	51	1,2,8	38	96	2,5,4	3
7	1,7,2	282	52	1,4,5	36	97	2,7,4	3
8	2,7,2	279	53	1,3,5	35	98	2,8,9	3
9	2,6,2	249	54	1,5,5	35	99	1,1,8	2
10	1,3,2	242	55	1,6,5	35	100	1,3,9	2
11	1,4,2	235	56	1,3,1	34	101	1,8,8	2
12	2,3,2	208	57	1,4,1	34	102	1,2,9	1
13	2,1,2	207	58	1,2,1	30	103	1,8,6	1
14	1,1,2	191	59	2,3,4	29	104	2,7,8	1
15	2,4,1	188	60	1,1,5	27	pozostałe		0
16	1,4,9	169	61	2,9,5	27			
17	1,8,2	152	62	1,5,4	26			
18	1,5,2	131	63	2,8,5	26			
19	1,9,2	125	64	1,8,1	25			
20	2,9,2	120	65	2,7,1	25			
21	2,1,1	118	66	1,6,4	24			
22	2,4,5	106	67	2,5,5	24			
23	2,5,2	101	68	2,6,5	24			
24	2,1,5	100	69	1,2,5	22			
25	2,4,9	99	70	2,7,5	22			
26	2,3,5	94	71	2,3,8	18			
27	2,8,2	93	72	2,2,4	16			
28	2,3,1	83	73	2,5,1	16			
29	1,6,1	81	74	1,3,8	15			
30	2,3,16	80	75	2,2,9	15			
31	2,3,17	70	76	2,9,1	15			
32	2,2,5	67	77	2,1,8	14			
33	1,7,1	63	78	1,7,4	12			
34	1,7,5	60	79	1,8,9	11			
35	2,3,15	60	80	2,4,3	10			
36	1,4,17	59	81	2,8,1	9			
37	2,4,4	58	82	1,9,4	8			
38	1,8,5	56	83	2,6,8	8			
39	1,9,5	52	84	1,8,4	7			
40	2,2,1	51	85	1,1,4	6			
41	2,2,8	51	86	2,9,4	6			
42	2,4,6	50	87	2,1,6	5			
43	2,4,8	49	88	2,6,4	5			
44	2,4,17	48	89	1,3,4	4			
45	1,4,8	47	90	1,4,4	4			

Źródło: opracowanie własne.

Tablica A.17

Uporządkowanie ścieżek WAP dla p=5 - zestaw danych nr 3

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	350	46	1,2,8	9
2	2,4,7	259	47	1,3,5	9
3	1,6,2	98	48	1,4,1	8
4	2,2,2	86	49	1,5,1	8
5	1,7,2	83	50	1,9,1	8
6	2,4,2	82	51	1,1,1	7
7	1,2,2	71	52	1,3,1	7
8	1,4,9	69	53	2,9,5	7
9	2,7,2	68	54	2,1,4	6
10	2,4,1	67	55	2,2,1	6
11	1,3,2	56	56	1,2,5	5
12	2,1,2	53	57	1,8,1	5
13	2,6,2	53	58	2,6,5	5
14	1,4,2	52	59	1,2,1	4
15	1,1,2	40	60	1,3,8	4
16	1,9,2	40	61	2,5,5	4
17	2,3,2	40	62	2,8,5	4
18	1,8,2	38	63	1,5,5	3
19	2,1,5	37	64	1,6,5	3
20	2,4,9	36	65	2,3,8	3
21	1,5,2	33	66	2,6,8	3
22	2,3,16	30	67	2,7,1	3
23	2,9,2	28	68	1,5,4	2
24	2,1,1	24	69	1,6,4	2
25	2,8,2	24	70	2,1,8	2
26	2,3,5	21	71	2,4,4	2
27	2,4,5	21	72	2,5,1	2
28	2,5,2	21	73	2,7,5	2
29	1,6,1	20	74	2,9,1	2
30	2,3,17	20	75	1,7,4	1
31	1,4,17	18	76	2,2,4	1
32	2,4,17	17	77	2,3,4	1
33	1,7,5	15	78	2,8,1	1
34	2,3,1	15	pozostałe		0
35	2,2,5	14			
36	1,8,5	13			
37	2,2,8	13			
38	2,4,8	13			
39	1,4,8	12			
40	1,7,1	12			
41	1,4,5	10			
42	1,9,5	10			
43	2,3,15	10			
44	2,6,1	10			
45	1,1,5	9			

Źródło: opracowanie własne.

Tablica A.18

Uporządkowanie ścieżek WAP dla p=3 - zestaw danych nr 3

Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	192	46	1,2,8	2
2	2,4,7	131	47	1,3,8	2
3	1,6,2	41	48	1,9,5	2
4	2,2,2	35	49	1,1,1	1
5	2,4,1	33	50	1,2,1	1
6	2,4,2	33	51	1,2,5	1
7	1,4,9	32	52	1,3,1	1
8	1,7,2	30	53	1,5,1	1
9	2,1,2	22	54	1,6,5	1
10	2,7,2	22	55	1,8,1	1
11	1,2,2	21	56	2,2,1	1
12	2,1,5	19	57	2,3,8	1
13	1,3,2	18	58	2,6,5	1
14	2,6,2	18	59	2,6,8	1
15	1,9,2	17	60	2,7,1	1
16	1,4,2	16	61	2,9,5	1
17	1,8,2	16	pozostałe		0
18	2,4,9	16			
19	1,1,2	14			
20	2,3,2	14			
21	1,5,2	11			
22	2,3,16	10			
23	2,9,2	9			
24	1,4,17	8			
25	2,1,1	8			
26	2,4,17	8			
27	2,4,5	7			
28	2,8,2	7			
29	1,6,1	6			
30	2,2,8	6			
31	2,3,5	6			
32	1,7,1	5			
33	1,7,5	5			
34	2,2,5	5			
35	2,5,2	5			
36	1,1,5	4			
37	1,4,5	4			
38	1,8,5	4			
39	2,3,1	4			
40	2,4,8	4			
41	1,3,5	3			
42	1,4,1	3			
43	1,4,8	3			
44	1,9,1	3			
45	2,6,1	3			

Źródło: opracowanie własne.

Tablica A.19

Uporządkowanie ścieżek WAP dla p=1

Zestaw danych nr 1			Zestaw danych nr 2			Zestaw danych nr 3		
Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga	Pozycja	Ścieżka	Ranga
1	1,4,7	53	1	1,4,7	45	1	1,4,7	46
2	2,4,7	36	2	2,4,7	37	2	2,4,7	20
3	1,4,9	5	3	2,4,1	11	3	2,4,1	10
4	1,6,2	5	4	1,6,2	7	4	1,6,2	8
5	1,7,2	5	5	2,2,2	6	5	2,1,2	5
6	2,4,9	5	6	2,7,2	6	6	2,2,2	5
7	1,2,2	4	7	1,7,2	5	7	2,4,9	5
8	2,6,2	4	8	1,2,2	4	8	2,7,2	5
9	1,8,2	3	9	1,1,2	2	9	1,4,9	4
10	2,1,2	3	10	1,3,2	2	10	1,9,2	4
11	1,4,8	2	11	1,7,1	2	11	2,1,5	4
12	2,1,5	2	12	2,1,2	2	12	1,2,2	3
13	2,2,2	2	13	2,2,8	2	13	1,7,2	3
14	2,4,2	2	14	2,4,2	2	14	1,8,2	3
15	2,4,8	2	15	2,4,9	2	15	2,3,2	3
16	2,9,2	2	16	2,5,2	2	16	1,1,2	2
17	1,3,10	1	17	2,6,2	2	17	1,4,2	2
18	1,3,13	1	18	1,2,1	1	18	2,4,17	2
19	1,4,10	1	19	1,2,8	1	19	2,6,2	2
20	1,5,2	1	20	1,4,2	1	20	1,1,5	1
21	1,5,15	1	21	1,5,2	1	21	1,3,2	1
22	1,6,17	1	22	1,8,2	1	22	1,4,5	1
23	1,7,1	1	23	2,1,1	1	23	1,4,17	1
24	1,7,9	1	24	2,1,5	1	24	1,5,2	1
25	1,8,14	1	25	2,3,1	1	25	1,6,1	1
26	2,1,1	1	26	2,3,2	1	26	1,7,1	1
27	2,1,10	1	27	2,3,8	1	27	1,8,5	1
28	2,3,2	1	28	2,9,2	1	28	2,1,1	1
29	2,5,2	1	pozostałe		0	29	2,2,5	1
30	2,7,2	1				30	2,4,5	1
31	2,8,2	1				31	2,6,1	1
pozostałe		0				32	2,8,2	1
						33	2,9,2	1
						pozostałe		0

Źródło: opracowanie własne.

Tablica A.20

Układ 10 „najlepszych" ścieżek (150 zbiorów 1-go zestawu danych)

Numer zbioru	Numer i miejsce ścieżki									
	1	2	3	4	5	6	7	8	9	10
1	1,4,7	2,4,7	2,4,9	1,7,2	1,6,2	2,4,1	2,3,1	2,4,2	2,3,2	1,4,9
2	2,1,2	1,1,2	2,3,2	2,4,2	1,4,7	1,3,2	1,4,2	2,2,2	1,2,5	1,4,9
3	1,7,2	2,7,2	1,6,2	2,6,2	1,9,2	1,8,2	1,7,1	1,2,2	1,9,1	1,8,1
4	1,4,7	2,4,7	2,7,1	2,6,1	2,9,1	1,1,1	1,3,1	1,4,1	1,2,1	1,5,5
5	1,4,7	2,4,7	1,7,2	2,2,8	2,3,8	1,2,8	2,1,2	1,3,8	2,2,2	2,4,2
6	1,4,7	2,4,7	2,3,2	2,4,2	2,2,2	2,6,2	1,3,2	1,4,2	1,2,2	2,9,2
7	1,7,2	2,7,2	2,6,2	1,6,2	2,1,5	2,4,2	2,3,2	1,9,2	2,9,2	2,8,2
8	2,2,2	2,3,2	2,4,2	1,6,2	1,2,2	2,1,2	2,6,2	1,3,2	1,4,2	1,1,2
9	1,4,7	2,4,7	2,7,1	2,9,1	1,7,1	1,9,1	2,8,1	2,9,4	2,8,4	2,7,4
10	2,6,2	1,7,2	1,6,2	2,7,2	1,9,2	1,2,2	2,9,2	1,8,5	1,9,5	1,7,5
11	1,6,2	2,6,2	1,7,2	1,3,2	1,4,2	2,7,2	2,4,2	2,3,2	2,2,2	1,2,2
12	1,8,2	2,8,2	1,5,2	2,3,1	2,4,1	2,2,1	2,4,4	2,3,4	2,2,4	2,5,2
13	2,1,2	2,2,2	1,3,2	1,4,2	2,3,2	2,4,2	2,5,2	1,5,2	1,2,2	1,1,2
14	2,7,2	2,6,2	1,6,2	2,1,2	2,2,2	2,3,2	2,4,2	1,7,2	1,1,2	1,3,2
15	1,7,2	2,7,2	2,6,2	1,6,2	2,2,2	2,3,2	2,4,2	2,1,2	1,2,2	1,3,2
16	1,4,7	2,4,9	1,4,9	2,4,7	2,2,2	2,2,5	2,7,1	2,9,1	2,9,2	2,7,2
17	2,4,7	1,4,7	2,4,9	1,4,9	1,4,8	1,5,2	1,6,2	2,5,2	2,1,5	2,3,5
18	1,4,7	2,4,7	1,8,5	1,9,5	1,7,5	1,7,2	2,2,2	2,1,5	2,1,2	2,9,2
19	2,4,7	2,2,2	1,1,2	1,3,2	1,4,2	1,4,7	2,4,2	2,3,2	1,2,2	2,7,2
20	1,4,7	2,4,7	2,2,2	2,3,2	2,4,2	2,1,2	1,5,2	2,5,2	1,9,2	1,1,2
21	2,4,7	1,4,9	1,4,7	2,4,9	1,4,8	2,6,2	1,6,2	2,4,2	2,3,2	2,4,8
22	1,4,7	2,4,7	1,1,5	2,2,5	2,9,2	2,2,2	2,1,5	2,5,2	1,8,2	2,8,2
23	1,4,7	2,4,7	1,6,1	2,7,1	2,9,1	1,5,1	2,8,1	1,5,4	1,6,4	2,7,4
24	2,4,7	1,4,7	2,7,2	1,4,2	1,3,2	1,2,2	2,6,2	1,1,2	1,7,2	2,2,2
25	1,2,2	1,3,2	1,4,2	1,1,2	2,2,2	1,7,2	2,1,2	2,3,2	2,4,2	1,2,8
26	1,5,15	2,5,9	1,2,1	1,9,15	2,5,14	2,5,6	2,2,4	2,5,2	1,5,11	1,9,1
27	1,8,14	2,8,1	1,7,9	2,7,17	2,7,2	1,7,10	2,7,10	1,1,17	2,8,4	1,7,7
28	1,7,9	2,7,3	2,7,10	1,3,8	1,7,10	1,7,17	1,7,11	1,9,13	2,7,11	1,3,10
29	1,6,17	2,6,17	1,2,17	2,3,17	1,2,10	1,3,11	1,6,10	2,4,17	2,2,17	1,2,16
30	1,3,10	2,3,17	2,3,10	1,4,10	2,4,10	2,4,2	1,3,7	2,4,17	1,3,17	1,2,10
31	2,4,9	1,4,9	1,4,7	1,4,8	2,4,8	2,4,7	2,2,2	2,8,2	1,5,2	1,8,2
32	1,4,7	2,4,7	1,2,2	1,1,5	2,1,2	1,7,5	1,9,5	1,8,5	1,4,9	1,1,2
33	2,4,7	1,4,7	1,4,9	2,4,9	2,6,2	2,3,2	2,4,2	1,7,2	1,6,2	2,7,2
34	1,4,7	2,4,7	1,7,2	1,2,2	2,7,2	2,6,2	1,6,2	1,5,2	2,2,2	2,7,1
35	1,4,7	2,4,7	1,1,5	1,3,5	1,4,5	1,2,5	2,2,5	2,1,5	2,3,5	2,4,5
36	2,4,7	1,4,7	2,4,9	2,4,8	2,7,2	1,7,2	2,4,5	2,3,5	2,2,5	1,4,9
37	2,2,2	1,4,7	2,1,2	1,2,2	1,3,2	1,4,2	1,7,2	2,7,2	2,3,2	2,4,2
38	1,4,7	2,4,7	2,1,1	2,1,4	2,2,5	2,2,2	1,3,5	1,4,5	2,5,2	2,4,1
39	2,4,2	2,3,2	2,1,2	2,2,2	1,1,2	2,1,8	1,4,2	1,3,2	1,2,2	2,3,9
40	2,5,2	2,8,2	1,5,2	1,8,2	1,2,8	1,7,1	1,9,1	1,8,1	1,8,4	1,7,4
41	1,4,8	1,4,7	2,4,7	1,4,9	2,4,9	2,4,8	2,6,2	2,7,2	1,6,2	1,4,2
42	2,4,7	1,4,7	2,7,2	1,4,9	2,1,1	2,1,4	1,7,2	2,9,5	2,8,5	2,7,5
43	1,4,7	2,4,7	2,9,2	1,6,2	2,7,1	2,6,2	2,9,1	2,8,1	2,8,4	2,9,4
44	2,4,7	1,4,7	2,9,2	1,9,2	2,1,1	2,4,1	2,3,1	2,1,4	2,2,1	2,3,4
45	1,4,7	2,4,7	2,3,8	2,1,8	2,1,2	1,1,8	2,2,8	1,9,5	1,7,5	1,8,5
46	2,4,7	1,4,7	1,6,1	1,5,1	1,6,4	1,5,4	1,8,2	2,4,1	2,3,1	1,7,1
47	1,4,7	2,4,9	1,4,9	2,7,2	2,4,2	2,3,2	2,2,2	2,1,2	2,4,7	1,6,2
48	1,4,7	2,4,7	1,2,2	1,8,2	1,7,1	2,8,2	1,5,2	1,9,1	2,7,1	2,9,2
49	2,8,2	1,5,2	1,8,2	2,5,2	1,4,9	1,4,7	1,6,1	1,7,1	1,5,1	1,8,1
50	1,4,7	2,4,7	2,1,2	2,6,2	2,2,2	2,3,2	2,4,2	2,7,2	1,6,2	1,1,2
51	1,4,7	2,4,7	2,4,9	2,8,2	1,4,9	2,2,2	2,5,2	2,7,1	1,2,1	2,9,1
52	2,4,7	1,4,7	1,7,2	2,2,2	2,7,2	1,2,2	2,1,2	1,7,5	1,9,5	1,8,5
53	1,4,7	1,5,2	2,4,7	2,8,2	2,5,2	1,8,2	1,9,2	1,7,2	1,2,1	2,7,1
54	2,4,7	1,4,7	2,1,5	2,3,5	2,4,5	1,9,2	2,2,5	1,6,1	2,9,2	1,5,1
55	2,3,2	2,4,2	1,7,2	2,2,2	2,7,2	1,6,2	2,1,2	1,9,2	1,4,7	1,2,2
56	2,4,7	1,4,7	2,9,2	2,9,5	2,7,5	2,8,5	2,1,2	2,6,2	1,7,2	2,7,2
57	1,4,7	2,4,7	1,6,1	1,5,1	1,6,5	1,5,5	1,1,1	1,9,5	1,7,5	1,8,5
58	2,4,7	1,6,2	1,4,2	1,3,2	2,2,2	1,4,7	2,7,2	1,2,2	2,1,2	2,4,2
59	1,2,2	1,4,2	1,3,2	1,1,2	2,7,2	2,6,2	1,2,9	2,2,2	1,3,9	1,6,2
60	1,7,2	2,6,2	1,4,7	1,9,2	1,5,2	1,6,2	2,7,2	2,8,2	2,4,7	2,5,2

61	2,4,8	2,4,9	1,4,9	1,4,7	2,4,7	1,4,8	2,2,2	2,4,2	2,3,2	1,2,2
62	1,4,9	2,4,7	1,4,8	2,4,9	2,1,2	1,4,7	1,1,2	1,2,2	1,6,2	2,2,2
63	2,4,9	1,4,9	1,4,7	2,4,7	2,7,2	1,7,2	1,2,2	2,2,2	2,6,1	1,4,8
64	1,2,2	1,4,2	1,3,2	1,1,2	2,4,2	2,3,2	2,2,2	2,1,2	1,7,2	2,7,2
65	1,4,7	2,4,7	1,6,2	2,2,2	1,7,2	1,8,2	2,8,2	2,3,2	2,4,2	1,2,1
66	2,4,7	2,4,8	1,4,9	1,4,7	2,4,9	1,4,8	2,3,1	2,4,1	2,3,4	2,4,4
67	2,4,7	1,4,7	2,4,9	2,7,1	2,9,1	2,8,1	2,4,8	2,2,1	2,7,4	2,8,4
68	2,4,7	1,4,7	1,8,2	1,1,2	2,8,2	1,5,2	2,1,2	2,2,2	2,4,2	2,3,2
69	2,4,7	1,4,7	2,2,2	2,3,2	2,4,2	2,1,2	1,1,2	1,4,2	1,3,2	1,2,2
70	1,4,7	2,4,7	1,7,2	2,2,2	2,3,2	2,4,2	2,7,2	2,1,2	1,2,2	1,3,2
71	2,4,9	2,4,7	1,4,7	1,4,9	2,4,8	1,4,8	1,1,1	1,1,4	2,7,5	2,8,5
72	1,4,8	2,4,9	1,4,9	2,4,8	1,4,7	2,4,7	1,2,2	1,9,2	2,6,2	2,2,2
73	2,4,7	1,4,7	1,4,9	2,4,9	1,4,8	2,1,2	1,6,2	1,3,2	1,4,2	2,2,2
74	1,4,7	1,6,2	2,6,2	2,7,2	2,1,2	2,4,7	1,7,2	1,1,2	1,2,8	2,4,2
75	2,4,7	1,4,7	2,2,2	2,4,2	2,3,2	1,2,2	1,1,2	1,3,2	1,4,2	2,7,2
76	1,4,7	1,7,2	1,6,2	2,6,2	2,2,2	1,1,2	2,4,7	2,1,1	2,2,1	2,3,1
77	2,4,7	1,8,2	1,7,2	1,6,2	2,7,2	1,2,2	1,3,2	1,4,2	2,6,2	1,1,2
78	1,4,7	1,4,9	2,9,2	2,7,1	1,6,2	1,7,2	1,8,2	2,8,1	2,6,1	2,9,1
79	1,4,7	1,4,9	1,6,2	1,7,2	1,6,1	2,6,1	1,3,1	1,4,1	1,2,1	1,2,2
80	2,9,2	2,6,2	1,4,7	2,7,2	1,9,2	2,2,2	2,4,7	1,6,2	1,7,2	2,1,2
81	2,9,2	1,2,2	2,2,2	2,7,2	2,6,2	1,4,2	1,3,2	1,9,2	1,7,2	1,1,2
82	1,6,2	1,7,2	1,8,2	1,2,2	1,6,1	1,5,1	2,7,2	1,5,4	1,6,4	1,5,2
83	2,1,5	1,2,2	2,3,5	2,4,5	2,7,2	2,2,5	2,2,2	2,6,2	1,1,2	1,6,2
84	1,4,7	2,4,7	2,3,2	2,4,2	2,9,2	1,6,2	1,7,2	2,1,2	2,3,1	2,4,1
85	1,4,7	1,8,2	1,7,1	1,9,1	1,8,1	1,9,4	1,7,4	1,8,4	2,7,1	2,9,1
86	2,1,1	2,3,1	2,4,1	2,1,4	2,3,4	2,4,4	1,6,5	1,5,5	2,2,5	2,2,1
87	2,1,5	2,3,5	2,4,5	2,1,1	1,1,5	1,7,1	1,9,1	1,8,1	2,1,4	1,9,4
88	1,6,2	1,9,5	1,8,5	1,7,5	1,7,2	1,2,2	2,7,2	2,9,5	2,7,5	2,8,5
89	2,6,2	1,9,2	2,8,2	1,8,2	2,7,1	2,9,1	2,8,1	2,5,2	2,9,4	2,7,4
90	2,6,2	2,7,2	1,7,2	1,6,2	1,9,5	1,7,5	1,8,5	1,6,1	1,5,1	1,6,4
91	1,4,9	1,4,7	1,4,8	2,4,9	2,4,8	2,9,2	1,6,2	2,7,2	1,6,1	1,5,1
92	2,4,9	2,4,7	2,4,8	1,4,7	1,4,9	1,5,2	2,2,1	2,1,1	1,6,1	2,3,2
93	1,4,9	2,4,7	1,4,7	2,9,2	1,6,2	2,2,2	2,7,2	1,7,2	2,3,2	2,4,2
94	1,4,7	2,4,7	2,2,2	2,1,2	2,4,2	2,3,2	1,1,2	1,2,2	1,3,2	1,4,2
95	2,4,7	1,4,7	1,5,2	1,8,2	2,5,2	2,8,2	2,9,2	2,3,1	2,4,1	2,1,1
96	1,4,7	1,4,9	2,4,7	2,6,2	2,7,2	2,1,5	2,6,1	1,1,5	2,5,1	1,4,8
97	1,4,7	2,4,7	2,7,2	2,2,1	2,2,4	2,4,1	2,3,1	2,4,4	2,3,4	2,2,2
98	1,4,7	1,6,2	1,9,2	2,4,1	2,3,1	2,1,5	2,3,4	2,4,4	2,6,2	2,5,2
99	1,4,7	2,4,7	2,2,2	2,1,2	2,4,2	2,3,2	1,1,2	2,7,2	1,3,2	1,4,2
100	1,4,7	2,5,2	2,9,2	1,7,2	2,1,1	2,2,1	2,3,1	2,4,1	2,2,2	2,1,4
101	1,3,13	2,3,13	1,3,3	1,3,15	1,3,10	2,3,15	2,3,8	2,3,4	2,3,10	2,3,11
102	2,6,2	2,6,17	1,6,3	1,6,10	1,6,5	1,6,2	1,6,4	1,6,6	1,6,16	1,6,17
103	1,8,2	1,6,8	2,5,17	2,1,17	1,5,2	1,4,10	1,5,4	1,1,10	1,3,10	1,9,5
104	1,4,10	2,4,10	1,4,8	2,4,4	2,4,5	1,4,11	1,2,2	2,1,10	1,2,10	2,2,11
105	2,1,10	1,8,11	2,7,17	1,8,13	2,8,2	2,6,10	2,8,14	2,9,16	1,6,7	2,1,7
106	1,4,7	1,1,5	1,4,9	1,7,1	1,9,1	1,6,1	1,8,1	1,5,1	1,9,4	1,7,4
107	1,4,7	2,4,7	1,4,9	2,7,1	2,9,1	2,8,1	2,9,4	2,7,4	2,8,4	2,7,2
108	1,4,7	2,4,7	1,2,2	2,2,2	1,8,2	2,4,1	2,3,1	2,2,1	1,2,1	2,3,4
109	1,4,7	2,4,7	2,1,2	2,3,2	2,4,2	2,2,2	1,5,2	2,7,2	1,7,2	1,1,2
110	2,4,7	1,4,7	1,2,2	1,9,2	2,2,2	2,1,2	2,1,5	1,7,2	2,3,5	2,4,5
111	2,4,7	1,4,7	2,7,2	1,2,2	2,3,1	2,4,1	2,4,4	2,3,4	2,5,2	2,7,5
112	1,4,7	2,2,2	1,7,2	1,2,2	1,6,5	1,5,5	2,1,2	1,8,2	2,9,2	2,7,2
113	1,4,7	2,7,1	2,9,1	2,8,1	2,7,4	2,8,4	2,9,4	1,1,1	1,5,2	1,1,4
114	2,4,2	2,3,2	2,9,2	2,1,2	2,4,17	1,8,2	2,2,2	1,2,2	1,5,2	2,8,2
115	1,8,2	1,4,7	2,7,2	1,5,2	2,2,8	1,8,5	1,7,5	1,9,5	1,8,6	2,3,8
116	2,4,7	1,4,7	1,4,9	2,4,17	2,7,2	1,6,2	2,2,5	2,1,5	1,2,5	2,3,1
117	1,4,9	2,4,7	1,4,7	1,4,8	2,4,9	2,3,2	2,4,2	2,4,17	2,5,2	2,2,2
118	2,4,7	1,4,7	1,6,1	1,5,1	1,6,4	1,5,4	2,4,17	1,7,1	1,9,1	1,8,1
119	2,4,7	1,4,7	2,1,5	1,1,5	2,2,5	2,3,5	2,4,5	2,2,1	2,3,1	2,4,1
120	2,4,7	1,4,7	1,6,2	1,7,2	2,7,2	2,1,1	2,2,2	2,1,4	2,4,1	2,3,1
121	2,4,8	1,6,2	1,7,2	1,9,2	1,4,7	1,4,9	2,4,7	2,7,2	2,6,2	2,4,9
122	2,4,7	1,4,7	2,9,2	1,2,2	2,5,2	2,1,5	2,8,2	2,6,2	2,3,5	2,4,5
123	1,4,7	2,4,7	2,9,2	1,9,2	1,4,2	1,3,2	1,2,2	1,7,1	1,9,1	2,7,2

124	1,4,7	2,4,7	2,7,1	2,9,1	2,6,1	1,1,1	1,3,1	1,4,1	1,2,1	2,8,1
125	1,5,2	1,7,2	2,7,2	1,8,2	2,5,2	1,4,7	1,6,1	1,5,1	1,5,4	1,6,4
126	1,4,7	1,6,2	1,2,2	2,6,2	1,3,2	1,4,2	1,7,2	1,7,1	1,8,2	1,9,1
127	1,7,1	1,6,1	1,9,1	1,8,1	1,5,1	1,8,4	1,9,4	1,7,4	1,5,4	1,6,4
128	1,4,7	1,2,2	2,4,7	1,3,2	1,4,2	2,2,2	1,1,2	2,1,2	2,3,2	2,4,2
129	1,6,2	1,7,2	2,7,2	2,3,2	2,4,2	2,4,7	2,1,2	2,6,2	2,2,2	1,4,7
130	2,4,7	1,4,7	1,8,2	2,6,1	2,2,1	2,8,2	2,3,1	2,4,1	1,1,1	2,1,1
131	1,6,2	2,6,2	1,2,2	1,7,2	1,3,2	1,4,2	1,9,2	1,4,7	2,7,2	1,8,2
132	1,4,7	1,6,2	1,7,2	2,4,7	1,2,2	2,7,1	2,6,1	2,9,1	1,4,2	1,3,2
133	1,2,2	1,4,2	1,3,2	1,1,2	1,7,2	2,6,2	1,2,9	1,9,2	2,7,2	2,2,2
134	1,7,2	1,6,2	1,2,2	1,4,2	1,3,2	1,1,2	2,7,2	2,6,2	2,2,2	2,3,2
135	1,4,7	1,6,1	2,4,7	1,5,1	1,5,4	1,6,4	2,2,5	2,3,5	2,4,5	2,1,5
136	2,4,7	1,4,7	1,6,2	2,4,5	2,3,5	1,7,2	2,4,1	2,3,1	2,2,1	2,2,5
137	2,4,7	1,4,7	1,8,2	1,7,2	1,5,2	2,8,2	1,7,1	1,9,1	2,7,1	1,2,1
138	2,4,9	1,4,9	1,4,7	2,2,2	2,3,2	2,4,2	2,4,7	2,1,2	1,2,2	1,4,2
139	2,1,2	1,4,7	2,4,7	2,4,2	2,3,2	2,2,2	1,1,2	1,2,2	1,6,2	1,3,2
140	2,4,7	1,4,7	2,1,1	2,4,1	2,3,1	2,2,1	2,1,4	2,4,4	2,3,4	2,2,4
141	1,4,7	2,4,7	1,4,9	2,2,2	2,7,2	2,5,2	2,1,2	1,2,1	2,8,2	2,6,1
142	2,4,7	1,4,7	2,2,2	2,3,2	2,4,2	2,6,2	1,3,2	1,4,2	2,1,2	1,1,2
143	1,4,7	2,4,7	1,4,9	1,2,2	1,3,2	1,4,2	2,7,2	2,6,2	2,2,2	1,1,2
144	1,4,7	2,4,7	1,7,2	2,7,2	1,6,2	2,6,2	1,2,2	1,4,2	1,3,2	2,2,2
145	2,4,7	1,4,2	1,3,2	1,2,2	1,4,7	1,6,2	1,1,2	2,6,2	2,2,2	1,7,2
146	1,4,7	1,6,2	2,2,1	2,2,4	2,4,7	2,7,1	2,9,1	2,8,1	2,8,4	2,7,4
147	2,4,7	2,4,9	1,4,9	1,4,7	1,7,2	1,6,2	2,7,2	2,3,1	2,4,1	2,4,4
148	1,4,7	2,4,7	1,4,9	2,2,2	1,2,2	1,9,2	2,9,2	2,5,2	1,6,2	2,8,2
149	2,4,7	1,4,7	1,4,2	1,3,2	1,4,9	1,2,2	2,2,2	1,1,2	2,3,2	2,4,2
150	1,4,9	2,4,9	1,4,7	2,4,7	2,3,2	2,4,2	1,4,2	1,3,2	2,2,2	1,2,2

Źródło: opracowanie własne.

Tablica A.21

Układ 10 „najlepszych" ścieżek (150 zbiorów 2-go zestawu danych)

Numer	Numer i miejsce ścieżki									
zbioru	1	2	3	4	5	6	7	8	9	10
1	2,4,7	1,2,2	1,1,2	1,3,2	1,4,2	1,7,2	2,1,2	2,2,2	2,3,2	2,4,2
2	1,4,7	2,4,7	2,7,2	2,6,2	2,8,2	1,7,1	2,5,2	1,9,1	1,6,1	2,9,2
3	2,7,2	1,7,2	2,6,2	1,6,2	2,2,2	2,3,2	2,4,2	2,1,2	1,2,2	1,4,2
4	2,9,2	1,9,2	1,4,7	2,4,7	1,5,5	1,6,5	1,3,5	1,4,5	1,2,5	1,1,5
5	2,4,7	1,4,7	1,6,2	1,7,2	1,3,2	1,4,2	1,2,2	1,1,2	2,9,2	1,8,2
6	1,7,2	1,6,2	1,2,2	1,4,2	1,3,2	1,1,2	2,7,2	2,6,2	2,2,2	2,3,2
7	2,7,2	1,3,2	1,4,2	2,2,2	2,3,2	2,4,2	1,1,2	1,2,2	2,1,2	1,7,2
8	2,2,2	1,2,2	2,1,2	1,1,2	1,4,2	1,3,2	2,4,2	2,3,2	2,2,8	2,2,9
9	2,5,2	1,4,7	2,8,2	2,4,7	1,2,5	2,6,5	2,5,5	1,8,2	2,9,2	1,9,2
10	2,4,7	1,6,2	1,7,2	1,4,7	2,6,2	2,7,2	1,2,2	1,4,2	1,3,2	1,1,2
11	2,1,5	2,1,1	2,2,5	2,4,5	2,3,5	2,1,4	2,4,1	2,3,1	1,1,5	2,2,1
12	2,6,2	1,6,2	1,2,2	1,4,2	1,3,2	2,2,2	2,4,2	2,3,2	1,1,2	2,1,2
13	2,4,2	2,3,2	1,6,1	2,2,2	1,1,8	1,5,1	1,6,4	1,5,4	2,7,2	2,3,8
14	2,5,2	2,9,2	1,9,2	2,8,2	1,8,2	1,5,2	1,7,1	2,2,2	2,3,2	2,4,2
15	1,1,2	1,2,2	2,3,2	2,4,2	1,3,2	1,4,2	2,1,2	2,2,2	1,6,2	2,7,2
16	2,4,7	1,4,7	2,2,2	2,3,2	2,4,2	1,1,2	1,4,2	1,3,2	1,2,2	2,6,2
17	2,4,7	1,4,7	1,7,2	2,7,2	1,6,2	2,6,2	1,7,1	1,9,1	1,8,1	2,2,2
18	2,4,7	1,4,7	2,9,2	1,9,2	2,1,1	2,4,1	2,3,1	2,2,1	2,1,4	2,3,4
19	2,4,7	1,4,7	2,2,2	2,1,2	2,4,2	2,3,2	1,2,2	2,2,8	1,1,2	1,2,8
20	1,4,7	2,4,7	1,7,2	1,6,2	2,1,2	2,3,2	2,4,2	2,6,2	2,2,2	1,2,8
21	1,4,7	1,2,2	1,1,2	2,2,2	1,4,2	1,3,2	2,3,2	2,4,2	2,1,2	2,4,7
22	2,4,7	1,4,7	1,4,2	1,3,2	1,1,2	1,2,2	2,6,2	2,7,2	2,2,2	2,3,2
23	2,4,7	2,1,2	1,1,2	1,4,7	1,2,2	2,2,2	2,4,2	2,3,2	1,4,2	1,3,2
24	1,4,7	2,4,7	2,1,2	2,2,2	2,8,2	2,3,2	2,4,2	2,6,2	1,6,2	1,8,2
25	1,1,2	2,2,2	1,4,2	1,3,2	2,1,2	1,2,2	2,4,2	2,3,2	1,7,2	1,4,7
26	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
27	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
28	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
29	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
30	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
31	1,4,7	2,4,7	1,5,2	1,6,2	1,8,2	1,5,6	2,6,2	2,5,2	2,1,2	2,4,2

32	1,4,7	2,4,7	2,2,8	2,2,2	1,7,2	2,3,8	2,1,2	1,6,2	1,1,2	2,4,2
33	2,4,7	1,4,7	1,7,2	2,7,2	2,6,5	2,5,5	1,6,1	1,5,1	1,6,4	1,5,4
34	2,2,8	1,3,8	1,2,2	2,6,2	2,4,7	1,3,2	1,4,2	2,3,8	1,4,7	2,2,2
35	1,4,7	2,4,7	1,8,2	1,5,2	1,7,2	2,9,5	2,7,5	2,8,5	1,6,2	1,2,1
36	1,6,2	2,6,2	1,2,2	2,3,8	1,8,2	2,8,2	1,1,2	1,3,2	1,4,2	2,4,17
37	2,2,2	1,1,2	1,7,2	2,3,2	2,4,2	2,7,2	1,2,2	2,1,2	1,4,2	1,3,2
38	2,4,7	1,4,7	1,6,2	1,7,2	2,7,2	2,8,9	2,6,2	2,8,7	1,2,2	2,2,8
39	1,6,2	1,9,2	2,6,2	1,8,2	2,4,1	2,3,1	2,1,1	2,9,2	2,4,4	2,3,4
40	2,4,1	2,3,1	2,1,1	2,2,1	2,9,2	2,4,4	2,3,4	2,1,4	2,2,4	2,5,2
41	1,4,7	2,4,7	1,9,2	2,7,1	2,9,1	2,8,1	2,9,4	2,7,4	2,8,4	2,6,1
42	1,4,7	2,4,7	1,4,2	1,3,2	2,5,2	2,7,2	1,5,2	1,7,2	1,1,2	2,6,2
43	1,4,7	2,4,7	1,2,2	1,2,8	1,4,2	1,3,2	2,2,2	2,3,2	2,4,2	1,1,2
44	2,4,7	1,4,7	1,1,2	2,1,2	1,5,2	1,3,2	1,4,2	2,4,2	2,3,2	2,5,9
45	1,4,7	2,4,7	2,5,5	2,6,5	1,6,5	1,5,5	1,7,1	1,9,1	1,8,1	1,8,4
46	2,4,7	1,4,7	2,4,2	2,3,2	2,3,1	2,4,1	2,1,1	2,7,1	1,6,1	2,2,1
47	2,7,2	2,1,2	2,6,2	2,4,2	2,3,2	2,2,2	1,6,2	1,2,2	1,1,2	2,9,2
48	2,4,7	1,4,7	1,7,2	2,7,2	2,1,2	2,3,2	2,4,2	2,2,2	1,1,2	1,4,2
49	1,4,7	2,4,7	1,6,2	1,6,1	1,7,1	2,1,1	1,7,2	2,2,1	2,4,1	2,3,1
50	1,6,2	1,7,2	2,7,2	2,6,2	2,2,2	2,4,2	2,3,2	2,1,2	1,2,2	1,3,2
51	1,2,2	1,1,2	2,2,2	2,4,2	2,3,2	2,1,2	1,4,2	1,3,2	2,7,2	1,7,2
52	2,4,7	1,4,7	2,8,2	2,1,1	2,9,2	2,2,1	2,3,1	2,4,1	2,1,5	2,4,5
53	2,4,7	1,4,7	1,6,5	1,5,5	2,1,1	2,3,1	2,4,1	2,2,1	2,2,5	2,7,1
54	2,7,2	2,1,2	2,2,2	2,4,2	2,3,2	1,9,2	1,7,2	2,6,2	1,2,2	1,1,2
55	1,4,7	2,1,2	2,4,2	2,3,2	2,2,2	2,4,7	1,4,2	1,3,2	1,1,2	1,2,2
56	1,3,2	1,4,2	1,2,2	1,7,1	1,6,1	1,9,1	1,1,2	1,8,1	1,5,2	1,5,1
57	1,2,2	1,3,2	1,4,2	1,1,2	1,6,2	2,2,2	2,3,2	2,4,2	2,1,2	2,6,2
58	2,7,2	1,7,2	2,6,2	1,6,2	1,2,2	1,3,2	1,4,2	1,1,2	2,2,2	2,3,2
59	1,8,2	1,5,2	1,9,2	2,8,2	1,7,2	2,5,2	1,6,2	1,2,1	1,1,1	2,6,1
60	2,2,8	2,3,8	2,1,8	1,6,2	1,2,8	1,7,2	1,3,8	1,8,2	1,1,8	1,4,7
61	1,4,7	2,4,7	1,4,8	2,4,9	1,4,9	2,4,8	2,3,2	2,4,2	1,1,2	2,1,2
62	2,4,7	1,4,7	2,3,2	2,4,2	2,2,2	2,1,2	1,1,2	2,1,1	1,4,2	1,3,2
63	1,4,7	2,4,7	2,4,9	1,4,9	1,2,8	1,6,2	2,6,2	2,9,2	2,2,2	1,4,2
64	2,4,7	1,4,7	2,4,9	2,2,2	2,1,2	2,3,2	2,4,2	1,2,2	1,1,2	2,6,2
65	2,4,7	1,4,7	2,7,2	1,7,2	1,2,2	1,3,2	1,4,2	2,6,2	1,1,2	2,2,2
66	2,4,7	1,4,7	2,3,2	2,4,2	2,1,2	2,2,2	2,3,1	2,4,1	2,2,1	2,1,1
67	2,4,7	1,4,7	1,3,2	1,4,2	1,1,2	2,2,2	1,2,2	2,1,2	2,3,2	2,4,2
68	2,4,7	1,4,7	2,2,2	2,4,2	2,3,2	1,7,2	1,1,2	2,1,2	1,4,2	1,3,2
69	2,4,7	1,4,7	1,6,5	1,5,5	1,2,2	1,1,2	2,9,5	2,7,5	2,8,5	2,5,5
70	1,4,7	2,4,7	2,2,2	2,1,2	2,3,2	2,4,2	1,1,2	1,4,2	1,3,2	1,2,2
71	1,4,7	2,4,7	2,2,2	2,3,2	2,4,2	1,2,2	1,3,2	1,4,2	2,7,2	1,1,2
72	1,4,7	2,4,7	2,4,2	2,3,2	1,1,2	2,1,2	1,3,2	1,4,2	2,2,2	1,5,2
73	1,4,7	2,4,7	2,7,2	2,6,2	1,6,2	1,7,2	2,5,2	1,9,2	1,1,1	2,6,1
74	2,4,7	1,4,7	2,6,2	2,9,2	1,7,2	2,7,2	1,2,2	1,4,2	1,3,2	2,1,2
75	1,4,7	2,4,7	1,7,2	1,6,5	1,5,5	1,2,5	1,5,2	1,1,5	1,3,5	1,4,5
76	1,5,2	1,7,1	1,9,1	1,8,2	1,6,1	1,8,1	1,5,1	1,4,7	1,7,4	1,8,4
77	2,4,7	1,4,7	1,2,2	1,1,2	1,4,2	1,3,2	2,7,2	2,2,2	2,9,2	2,1,2
78	1,6,2	2,6,2	1,3,2	1,4,2	1,4,7	1,2,2	2,4,7	2,4,2	2,3,2	2,7,2
79	1,7,2	2,7,2	1,6,2	1,4,7	2,6,2	2,4,7	2,5,2	2,9,2	2,8,2	1,9,2
80	2,4,7	1,4,7	2,4,1	2,3,1	2,2,1	2,1,1	1,5,2	2,4,5	2,3,5	2,2,5
81	2,3,2	2,4,2	1,4,7	2,2,2	1,1,2	2,6,2	2,1,1	1,4,2	1,3,2	2,4,1
82	2,2,2	2,3,2	2,4,2	2,1,2	1,7,2	1,6,2	1,2,2	1,1,2	1,4,2	1,3,2
83	1,4,7	1,8,2	2,8,2	2,6,1	1,2,1	1,3,1	1,4,1	2,5,1	2,5,2	2,5,4
84	1,6,2	1,2,2	1,4,2	1,3,2	2,7,2	2,6,2	1,7,2	1,2,9	1,3,9	1,1,2
85	2,6,2	1,6,2	1,7,2	1,2,2	1,1,2	1,3,2	1,4,2	2,7,2	2,2,5	2,4,5
86	1,2,1	1,2,4	2,7,5	2,9,5	2,8,5	2,1,2	2,2,2	2,9,2	2,8,2	2,7,1
87	1,7,2	2,7,2	1,6,2	2,6,2	1,2,5	2,7,5	2,9,5	2,8,5	1,9,5	1,7,5
88	2,3,8	2,2,8	2,2,2	2,4,2	2,3,2	2,1,1	2,1,8	2,1,2	2,6,1	2,2,1
89	2,3,1	2,4,1	2,2,1	2,4,4	2,3,4	2,2,4	2,1,1	1,2,2	2,1,4	1,9,2
90	1,7,1	1,9,1	1,8,1	1,7,4	1,8,4	1,9,4	1,8,2	2,6,5	2,5,5	1,4,2
91	1,4,7	2,4,7	1,9,2	2,2,2	2,4,9	1,4,9	2,1,5	2,9,2	2,8,2	1,2,2
92	2,4,7	1,4,7	1,7,2	2,2,2	2,1,2	2,3,2	2,4,2	2,2,5	2,2,1	1,1,5
93	2,4,7	1,4,7	1,2,2	2,1,2	2,1,1	1,4,2	1,3,2	2,2,2	2,1,5	2,1,4
94	1,4,7	1,6,2	1,2,2	1,3,2	1,4,2	1,7,2	1,1,2	2,4,7	2,2,2	2,1,2

95	1,4,7	2,1,2	1,9,2	1,6,2	1,7,2	2,6,2	1,8,2	2,3,2	2,4,2	1,6,1
96	2,2,2	2,7,2	1,7,2	2,6,2	2,1,2	1,6,2	2,4,2	2,3,2	1,2,2	2,4,7
97	1,4,7	1,7,2	2,1,2	2,7,2	1,9,2	2,1,5	2,8,2	2,6,1	2,2,5	2,5,2
98	1,4,7	2,4,7	1,9,2	1,3,2	1,4,2	1,2,2	2,2,2	1,1,2	2,4,2	2,3,2
99	1,4,7	1,7,2	1,6,2	2,4,7	2,7,2	1,8,2	2,6,2	1,9,2	2,7,1	2,9,1
100	1,4,7	2,7,2	1,7,2	1,6,2	2,6,2	2,4,7	2,1,2	1,2,2	2,3,2	2,4,2
101	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
102	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
103	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
104	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
105	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
106	1,4,7	1,4,9	2,4,7	1,4,8	2,3,2	2,4,2	1,6,2	2,1,2	2,6,1	2,5,1
107	2,7,2	1,7,2	2,6,2	1,2,2	1,2,9	1,4,7	1,6,2	1,2,8	1,3,2	1,4,2
108	1,4,7	2,4,7	2,1,2	2,3,2	2,4,2	1,6,2	1,3,2	1,4,2	2,1,5	2,9,2
109	1,4,7	1,4,9	1,2,2	1,3,2	1,4,2	2,4,7	1,1,2	1,7,2	2,7,2	1,6,2
110	1,4,7	2,4,7	1,9,2	1,8,2	2,3,2	2,4,2	2,2,2	2,1,2	2,9,2	1,1,2
111	1,4,7	1,7,2	1,6,2	2,1,5	2,4,7	1,8,5	1,9,5	1,7,5	1,1,5	2,8,5
112	1,4,7	1,6,2	2,1,2	2,4,2	2,3,2	2,9,5	2,8,5	2,7,5	1,7,2	2,6,2
113	1,4,7	2,1,5	2,2,5	1,3,5	1,4,5	1,8,2	2,4,5	2,3,5	1,2,5	1,7,2
114	1,2,2	1,6,2	1,7,2	1,2,9	2,3,8	1,4,2	1,3,2	1,4,5	1,3,5	2,7,2
115	2,1,1	2,1,5	2,1,4	2,4,1	2,3,1	2,4,5	2,3,5	2,2,1	2,3,4	2,4,4
116	1,4,7	2,4,7	1,4,9	1,4,8	2,4,9	2,4,17	1,7,2	1,4,17	2,6,2	1,4,5
117	1,4,7	2,4,7	2,1,2	2,2,5	2,6,1	2,5,1	2,5,4	2,6,4	1,3,1	1,4,1
118	1,4,7	1,6,1	1,5,1	1,5,4	1,6,4	1,7,1	1,9,1	1,8,1	1,7,4	1,8,4
119	1,4,7	2,2,2	1,1,2	2,9,2	1,2,8	1,6,2	2,6,2	2,2,8	2,2,9	2,4,2
120	1,4,7	2,4,7	1,6,2	2,4,2	2,3,2	2,1,2	1,7,2	2,6,2	2,2,2	1,7,6
121	2,2,2	2,4,2	2,3,2	2,3,1	2,4,1	2,1,1	1,4,9	2,1,2	2,2,1	2,6,2
122	1,4,7	1,1,2	1,4,2	1,3,2	1,2,2	2,6,1	2,5,2	1,1,1	1,7,1	2,5,1
123	2,1,2	2,3,2	2,4,2	2,2,2	1,9,2	1,4,7	1,7,2	1,2,2	1,3,2	1,4,2
124	1,7,2	1,6,2	2,1,2	2,3,2	2,4,2	2,2,2	1,1,2	2,2,8	2,3,8	2,1,8
125	2,2,2	2,3,2	2,4,2	2,1,2	2,4,9	1,7,2	2,6,2	1,6,2	2,7,2	1,8,2
126	2,4,7	1,4,7	2,1,1	2,2,1	2,3,1	2,4,1	2,8,5	2,7,5	2,9,5	1,4,5
127	1,4,7	2,4,7	2,1,2	2,4,2	2,3,2	2,2,2	1,5,2	2,2,1	2,4,1	2,3,1
128	2,4,7	1,6,2	1,7,2	1,4,7	2,6,2	2,7,2	1,2,2	1,4,2	1,3,2	1,1,2
129	1,7,1	1,6,1	1,9,1	1,8,1	1,5,1	1,7,4	1,8,4	1,9,4	1,6,4	1,5,4
130	1,2,8	1,2,2	1,1,2	2,2,8	1,3,8	1,4,2	1,3,2	2,3,8	1,6,2	1,1,8
131	1,6,2	1,7,2	2,2,2	1,1,2	1,2,2	1,3,2	1,4,2	2,4,7	2,1,1	1,4,7
132	1,4,2	1,3,2	1,2,2	1,1,2	1,7,2	2,3,2	2,4,2	2,2,2	2,1,2	1,6,2
133	1,3,2	1,4,2	1,2,2	2,1,2	1,1,2	2,2,2	2,6,2	2,3,2	2,4,2	2,7,2
134	2,4,2	2,3,2	2,1,2	2,2,2	1,5,2	1,8,2	1,4,7	2,7,2	2,6,2	2,8,2
135	1,7,2	2,2,2	1,6,2	2,3,2	2,4,2	2,7,2	2,1,2	2,6,2	1,4,7	2,8,2
136	1,6,2	1,7,2	1,9,2	2,7,2	1,2,2	2,6,2	2,9,2	1,4,9	2,2,2	1,5,2
137	2,4,9	1,4,9	2,4,7	1,7,2	2,2,2	2,4,2	2,3,2	2,6,2	2,7,2	2,1,2
138	2,4,9	1,4,9	1,4,7	2,1,2	2,3,2	2,4,2	2,2,2	1,6,2	2,7,2	1,7,2
139	2,4,7	2,1,2	2,2,2	2,3,2	2,4,2	2,2,8	1,4,7	1,1,2	2,1,8	1,4,2
140	2,4,7	1,4,9	1,2,2	2,2,2	1,7,2	1,1,2	1,4,7	2,1,2	2,4,2	2,3,2
141	2,4,7	1,4,7	1,4,9	1,2,2	2,2,5	2,5,2	2,3,5	2,4,5	2,2,1	2,4,1
142	1,4,7	2,4,7	1,6,1	1,5,5	1,6,5	1,5,1	1,6,4	1,5,4	2,8,2	2,3,5
143	2,1,2	2,4,2	2,3,2	1,2,2	1,3,2	1,4,2	1,1,2	2,2,2	1,4,7	2,1,8
144	2,4,7	1,4,7	1,2,2	2,1,2	2,4,2	2,3,2	1,3,2	1,4,2	2,2,2	1,1,2
145	1,2,2	1,3,2	1,4,2	1,6,2	2,4,7	1,1,2	2,2,2	2,4,2	2,3,2	2,4,9
146	2,4,7	1,4,9	1,4,7	2,4,9	1,4,8	2,1,1	2,3,1	2,4,1	2,2,1	2,1,4
147	1,4,7	1,4,9	2,4,7	2,2,2	2,3,2	2,4,2	2,1,2	1,6,2	2,7,1	2,9,1
148	2,4,7	1,4,7	2,3,2	2,4,2	2,2,2	2,1,2	2,2,9	2,3,9	1,1,2	2,2,1
149	1,4,7	2,4,7	1,2,2	1,3,2	1,4,2	2,2,2	2,4,2	2,3,2	1,1,2	2,2,8
150	2,4,7	2,2,2	1,4,7	2,4,2	2,3,2	2,2,1	2,2,9	1,2,2	2,1,2	1,1,2

Źródło: opracowanie własne.

Tablica A.22

Układ 10 „najlepszych" ścieżek (150 zbiorów 3-go zestawu danych)

Numer zbioru	Numer i miejsce ścieżki									
	1	2	3	4	5	6	7	8	9	10
1	2,3,2	2,4,2	1,1,2	1,3,2	1,4,2	2,2,2	2,1,2	1,4,7	2,4,7	2,2,9
2	1,4,7	2,4,7	2,1,1	2,3,1	2,4,1	2,2,1	2,8,2	2,5,2	2,6,1	2,7,1
3	2,1,2	2,2,2	2,3,2	2,4,2	1,6,2	2,4,7	1,1,2	1,4,7	2,6,2	1,2,2
4	1,6,2	2,2,8	2,3,8	2,1,8	2,6,2	2,1,2	1,7,2	1,3,2	1,4,2	2,7,2
5	1,4,7	1,7,5	1,8,5	1,9,5	1,7,1	1,9,1	2,3,2	2,4,2	2,2,2	1,6,1
6	2,6,2	1,8,2	1,5,2	2,7,2	2,1,5	1,2,2	1,5,5	1,6,5	1,6,2	1,7,1
7	2,7,2	2,6,2	2,1,1	1,9,2	1,2,2	2,3,1	2,4,1	1,3,2	1,4,2	2,1,4
8	1,4,7	2,1,5	2,4,1	2,3,1	2,1,1	2,4,4	2,3,4	2,2,1	2,1,4	2,3,5
9	2,4,7	1,6,2	1,4,7	2,1,2	1,7,2	1,8,5	1,9,5	1,7,5	2,3,2	2,4,2
10	1,9,2	2,9,2	1,2,1	1,3,1	1,4,1	1,1,1	2,2,1	2,6,1	2,4,1	2,3,1
11	2,2,2	2,7,2	2,1,1	2,4,1	2,3,1	2,1,4	1,4,2	1,3,2	1,2,2	1,7,2
12	2,2,5	2,6,2	2,4,5	2,3,5	1,1,5	2,7,2	2,1,5	2,2,1	2,2,4	2,1,1
13	2,6,1	1,4,1	1,3,1	2,5,1	1,1,1	2,7,1	2,6,4	2,5,4	2,9,1	1,4,4
14	1,2,2	1,4,2	1,3,2	2,2,2	1,1,2	2,2,9	2,7,2	2,6,1	1,3,9	2,5,1
15	2,7,2	2,2,8	2,2,2	2,1,2	2,9,2	2,1,8	2,3,8	2,4,2	2,3,2	1,2,8
16	2,4,9	1,4,9	2,4,8	2,4,7	1,4,7	2,4,2	2,3,2	2,3,5	2,4,5	2,4,1
17	1,4,7	2,1,2	2,4,2	2,3,2	2,4,7	2,2,5	2,3,5	2,4,5	1,1,2	1,4,2
18	1,4,7	2,4,7	1,3,5	1,4,5	1,6,2	1,7,2	2,5,2	2,3,5	2,4,5	1,2,5
19	1,4,7	2,4,7	1,6,2	1,7,2	2,7,2	2,2,1	2,1,1	1,6,1	2,6,5	2,5,5
20	1,4,7	2,4,7	2,2,2	1,8,2	1,1,2	1,3,2	1,4,2	1,7,2	1,2,2	2,4,2
21	1,4,7	1,2,2	1,9,2	2,2,1	2,2,4	2,2,5	2,6,2	1,5,2	2,8,2	2,7,1
22	1,6,2	1,7,2	1,9,2	2,6,2	2,7,2	1,2,2	2,1,2	2,2,2	2,3,2	2,4,2
23	1,4,7	2,7,2	1,9,2	2,9,2	2,8,2	2,3,2	2,4,2	2,8,9	2,1,2	2,2,2
24	1,4,7	2,4,7	1,6,5	1,5,5	1,6,2	2,9,2	2,6,2	2,8,2	1,8,2	1,5,2
25	1,6,2	1,4,7	1,2,2	1,4,2	1,3,2	2,6,2	2,2,2	1,7,2	2,3,2	2,4,2
26	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
27	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
28	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
29	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
30	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
31	2,9,2	1,4,7	2,2,2	1,7,5	1,8,5	1,9,5	2,4,5	2,3,5	2,2,5	2,5,5
32	1,4,7	2,4,7	1,7,2	2,7,2	1,9,5	1,7,5	1,8,5	1,8,2	1,5,5	1,6,5
33	2,4,7	1,4,9	1,4,7	2,3,2	2,4,2	2,7,2	2,1,2	2,6,2	2,2,2	1,2,2
34	1,4,7	2,4,7	1,6,2	2,7,2	2,8,5	2,7,5	2,9,5	1,7,2	1,7,5	1,8,5
35	2,4,7	1,4,7	2,6,5	2,5,5	2,5,2	2,1,1	2,4,1	2,3,1	2,1,4	2,4,4
36	1,4,7	1,4,9	1,6,1	1,5,1	2,4,7	1,6,4	1,5,4	2,2,2	2,9,2	1,5,2
37	1,4,7	2,5,2	1,5,2	2,8,2	2,4,1	2,3,1	2,4,4	2,3,4	2,7,2	2,6,1
38	1,4,7	2,1,5	1,4,5	1,3,5	2,4,7	2,2,5	1,2,1	1,2,4	2,8,5	2,9,5
39	1,4,7	2,4,7	2,2,2	1,3,2	1,4,2	1,1,2	1,2,2	1,6,5	1,5,5	2,6,5
40	1,4,7	2,4,7	1,6,2	2,2,2	1,7,5	1,9,5	1,8,5	1,6,5	1,5,5	2,1,2
41	1,4,9	1,7,2	2,4,7	1,6,2	2,4,9	1,1,2	2,1,5	2,2,2	2,1,1	2,7,8
42	1,4,7	2,4,7	1,4,9	2,9,2	1,7,2	2,6,2	2,7,1	2,9,1	2,8,1	2,9,4
43	1,4,7	2,4,7	2,2,2	1,1,2	1,3,2	1,4,2	1,4,17	2,1,2	1,1,8	1,3,8
44	2,4,7	1,4,7	1,5,2	2,1,2	1,8,2	2,1,1	2,1,4	1,2,1	2,6,1	2,5,1
45	1,4,7	2,4,7	2,4,2	2,3,2	2,1,2	1,6,5	1,5,5	2,2,2	2,5,5	2,6,5
46	2,2,2	2,4,2	2,3,2	1,6,2	1,2,2	1,3,2	1,4,2	2,6,2	2,7,2	1,4,7
47	2,1,2	2,2,2	2,4,2	2,3,2	2,6,2	2,7,2	1,1,2	1,2,2	1,3,2	1,4,2
48	1,6,1	1,9,2	1,5,2	1,5,1	1,5,4	1,6,4	1,8,2	1,7,1	1,8,1	1,9,1
49	2,7,2	2,2,8	1,2,8	1,7,2	1,2,2	1,1,2	2,2,2	1,4,2	1,3,2	2,1,2
50	1,7,2	1,6,2	2,7,2	2,6,2	1,4,2	1,3,2	1,2,2	1,9,2	2,2,8	2,3,8
51	1,4,7	1,8,2	1,9,5	1,8,5	1,7,5	2,8,2	2,7,1	2,9,1	1,5,2	2,8,1
52	1,8,5	1,7,5	1,9,5	2,9,5	2,8,5	2,7,5	1,7,2	1,6,5	1,5,5	2,5,5
53	2,4,5	2,3,5	2,2,5	2,8,2	2,1,5	1,4,2	1,3,2	1,3,5	1,4,5	2,7,1
54	1,1,2	1,3,2	1,4,2	1,2,2	2,1,2	2,6,2	2,3,2	2,4,2	2,2,2	2,7,2
55	1,6,2	2,2,2	1,2,2	1,3,2	1,4,2	2,7,2	2,6,2	1,7,2	2,4,2	2,3,2
56	2,1,5	1,8,2	1,1,5	2,3,5	2,4,5	2,1,6	2,1,1	1,5,2	2,1,4	1,8,6
57	1,9,2	1,7,1	1,9,1	1,8,1	1,7,4	1,9,4	1,8,4	1,5,2	1,6,2	1,7,2
58	2,2,2	2,1,2	2,3,2	2,4,2	2,7,2	1,7,2	2,9,2	1,6,2	1,2,2	2,2,8
59	1,2,2	1,3,8	1,1,2	2,4,2	2,3,2	1,4,17	2,1,2	2,2,2	2,1,8	2,2,9
60	1,6,2	1,7,2	2,6,2	2,7,2	2,2,8	1,2,8	1,2,2	1,2,5	1,9,2	2,3,5

61	1,4,7	1,4,9	1,2,2	1,3,2	1,4,2	2,6,2	2,7,2	2,9,2	2,2,2	2,6,1
62	1,4,2	1,3,2	2,4,17	1,2,2	2,2,2	1,4,17	2,2,9	2,9,2	2,3,2	2,4,2
63	1,4,7	2,4,7	1,7,2	1,6,2	1,2,2	2,2,2	1,3,2	1,4,2	2,6,2	1,1,2
64	1,4,7	2,4,7	1,5,2	1,6,1	2,5,2	1,5,1	1,5,4	1,6,4	1,8,2	1,7,1
65	2,2,2	1,3,2	1,4,2	1,1,2	2,4,2	2,3,2	2,2,9	2,2,8	1,2,8	1,5,2
66	1,6,2	1,7,2	1,4,17	1,9,2	1,4,7	2,1,1	2,3,5	2,4,5	2,1,4	1,6,1
67	2,3,2	2,4,2	2,2,2	1,6,2	1,5,2	2,7,2	2,6,2	2,1,2	1,7,2	1,8,2
68	1,4,7	2,3,5	2,4,5	1,2,5	2,5,5	2,6,5	1,6,1	1,1,5	1,5,1	1,7,1
69	1,8,2	1,4,7	1,6,2	1,2,8	1,7,2	1,3,2	1,4,2	1,5,6	1,6,1	2,5,2
70	1,8,2	1,3,2	1,4,2	1,2,2	2,5,2	2,1,2	1,5,2	2,8,2	2,2,8	1,1,2
71	1,4,7	1,4,9	2,3,5	2,4,5	2,2,5	2,2,1	2,7,2	2,2,4	2,4,1	2,3,1
72	1,4,7	2,4,7	1,4,9	1,6,1	1,5,1	1,5,4	1,6,4	2,1,2	1,7,2	1,6,2
73	1,6,2	1,7,2	2,6,8	1,4,7	2,4,17	1,8,9	1,8,7	2,6,2	1,5,2	1,2,8
74	1,4,7	2,4,7	2,1,2	2,6,2	1,1,2	2,2,2	2,4,2	2,3,2	1,2,2	1,7,2
75	2,4,7	1,4,7	2,2,5	2,4,5	2,3,5	2,1,5	2,3,1	2,4,1	2,2,8	2,3,4
76	2,4,7	1,4,9	1,4,7	2,7,2	2,6,2	2,5,2	2,8,2	1,8,2	1,9,2	2,9,2
77	2,4,9	1,4,7	1,4,9	1,6,1	2,6,2	1,5,1	1,6,2	1,2,2	2,7,2	1,7,1
78	1,4,9	1,4,7	2,4,7	2,4,9	2,1,1	2,7,2	2,6,2	2,2,2	2,4,2	2,3,2
79	2,4,7	1,4,7	1,6,2	2,6,2	1,7,2	2,9,2	2,3,1	2,4,1	2,1,1	2,2,1
80	1,1,2	1,4,2	1,3,2	1,2,2	1,2,8	1,3,8	2,2,2	1,6,2	2,4,2	2,3,2
81	1,4,7	2,4,7	1,8,2	1,7,2	2,2,2	2,4,2	2,3,2	2,9,2	2,7,2	1,6,2
82	1,4,7	2,4,7	1,1,1	2,6,1	1,6,1	1,2,1	1,3,1	1,4,1	2,5,1	1,5,1
83	2,4,7	1,4,7	2,1,1	2,1,4	2,2,1	2,3,1	2,4,1	2,3,4	2,4,4	2,2,4
84	1,7,2	1,6,2	2,7,2	2,6,2	1,1,1	1,2,2	1,7,1	1,6,1	2,2,2	1,9,1
85	2,4,7	1,4,7	2,9,5	2,8,5	2,7,5	1,7,5	1,8,5	1,9,5	1,5,5	1,6,5
86	2,2,2	1,2,2	2,9,2	2,7,2	1,9,2	1,1,2	1,6,2	1,7,2	2,1,2	2,4,17
87	1,4,5	1,3,5	1,2,5	2,6,5	2,5,5	2,2,2	1,1,2	2,6,2	2,4,2	2,3,2
88	1,4,2	1,3,2	1,2,2	1,6,2	2,2,2	1,1,2	2,6,2	2,3,2	2,4,2	1,9,2
89	2,1,2	1,1,2	2,2,2	2,8,2	2,4,2	2,3,2	2,7,2	1,2,2	1,8,2	2,7,1
90	2,1,5	1,3,2	1,4,2	2,2,5	1,2,2	2,3,5	2,4,5	1,8,2	1,5,2	2,7,2
91	2,4,9	2,4,8	2,4,7	1,4,7	1,4,9	1,4,8	2,3,2	2,4,2	1,7,2	1,4,2
92	2,4,9	2,4,7	1,4,7	1,4,9	1,4,8	2,4,8	2,4,2	2,3,2	2,2,2	2,9,2
93	1,4,9	1,4,7	2,4,7	1,7,2	1,5,2	1,6,2	1,8,2	2,4,9	2,9,2	2,1,2
94	1,4,7	2,4,7	1,6,2	1,7,2	1,2,2	2,7,2	2,2,2	2,3,2	2,4,2	2,1,2
95	2,7,2	1,4,7	1,7,2	2,4,7	1,6,2	1,9,2	2,6,2	2,9,2	1,8,2	1,5,2
96	2,4,9	1,4,9	1,4,7	2,4,7	1,4,8	2,9,2	1,1,2	2,4,2	2,3,2	2,4,8
97	1,4,7	2,4,7	1,7,2	2,7,2	2,6,2	1,6,2	2,2,2	1,9,2	1,4,2	1,3,2
98	1,4,7	2,4,7	2,8,2	1,7,1	1,9,1	1,8,1	2,1,1	1,7,4	1,8,4	1,9,4
99	2,7,2	1,7,2	2,6,2	1,6,2	1,2,2	1,3,2	1,4,2	2,2,2	2,4,2	2,3,2
100	1,4,7	2,4,7	2,2,1	2,4,1	2,3,1	2,1,1	2,2,4	2,3,4	2,4,4	2,1,4
101	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
102	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
103	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
104	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
105	2,4,1	2,4,2	2,3,16	2,3,17	2,3,15	2,4,6	2,4,7	2,4,4	2,4,5	2,4,3
106	1,4,9	1,4,8	2,4,9	2,4,8	2,4,7	1,4,7	2,1,2	1,4,2	1,3,2	2,9,2
107	2,4,7	1,4,7	1,4,9	2,4,9	2,5,2	1,8,2	2,1,1	2,1,4	2,1,2	1,5,2
108	1,4,7	2,4,7	1,4,1	1,3,1	2,6,1	2,5,1	1,3,4	1,4,4	1,2,1	2,6,4
109	2,4,7	1,4,7	1,2,2	1,4,2	1,3,2	1,1,2	2,2,2	2,1,2	2,3,2	2,4,2
110	2,4,7	1,4,7	2,9,2	2,2,2	2,1,2	1,3,2	1,4,2	2,3,2	2,4,2	1,2,2
111	1,4,7	1,4,9	2,4,7	2,4,9	1,4,8	2,4,8	1,7,2	1,4,2	1,3,2	2,4,17
112	2,4,17	1,4,17	1,4,7	2,4,7	2,1,2	1,1,2	2,8,6	1,5,2	1,8,2	2,5,2
113	1,4,7	2,4,7	2,2,2	1,3,2	1,4,2	2,4,2	2,3,2	1,2,2	1,7,2	2,7,2
114	1,6,2	2,6,2	1,2,8	1,2,2	1,4,2	1,3,2	2,2,2	2,2,8	2,4,2	2,3,2
115	1,4,7	2,2,2	1,1,2	2,4,2	2,3,2	1,2,2	1,4,2	1,3,2	2,1,2	2,7,2
116	1,4,7	2,4,7	2,4,8	1,4,8	2,4,9	1,4,9	1,8,2	1,7,1	1,9,1	1,8,1
117	1,4,7	2,4,7	2,4,17	1,4,17	2,2,2	2,4,2	2,3,2	1,2,2	2,3,8	1,3,2
118	2,4,7	1,4,7	2,8,2	1,5,2	2,5,2	1,8,2	2,4,17	2,2,2	1,4,17	1,2,2
119	2,4,7	1,4,7	1,7,5	1,8,5	1,9,5	2,3,5	2,4,5	1,6,1	2,2,5	1,5,1
120	2,4,7	1,4,7	1,7,2	2,7,2	2,5,2	2,6,2	1,5,2	1,8,2	1,8,9	1,2,2
121	2,4,7	1,4,7	2,1,1	2,1,4	1,5,2	1,8,2	2,5,2	2,8,2	1,1,1	1,1,4
122	2,4,7	1,4,7	2,1,5	2,3,5	2,4,5	1,3,5	1,4,5	2,2,5	2,1,1	2,4,1
123	1,8,2	1,5,2	2,8,2	2,5,2	1,9,2	1,4,7	1,6,1	1,7,1	2,9,2	1,9,1

124	1,5,2	2,5,2	2,8,2	1,8,2	1,9,2	2,2,8	1,8,9	2,3,8	1,7,1	1,9,1
125	2,8,2	2,9,2	2,5,2	1,8,2	1,7,2	1,6,2	1,2,2	1,1,2	1,4,2	1,3,2
126	1,1,5	2,1,5	2,6,2	2,1,1	2,2,2	2,7,2	2,3,2	2,4,2	2,1,4	2,1,2
127	2,4,17	1,4,17	1,7,2	2,7,2	1,2,2	1,6,2	2,6,2	2,2,2	2,1,2	1,1,2
128	1,3,2	1,4,2	2,2,2	1,2,2	2,1,2	2,3,2	2,4,2	1,1,2	2,7,2	2,6,2
129	2,1,1	2,3,1	2,4,1	2,1,4	1,6,1	1,5,1	2,3,4	2,4,4	2,2,1	1,5,4
130	2,1,5	2,4,5	2,3,5	2,9,5	2,7,5	2,8,5	1,4,5	1,3,5	2,2,5	1,2,5
131	1,9,2	2,6,2	1,3,2	1,4,2	1,2,2	1,6,2	2,2,2	2,3,2	2,4,2	2,1,1
132	1,2,2	1,1,2	2,2,2	2,7,2	1,7,2	2,1,2	2,6,2	1,6,2	1,3,2	1,4,2
133	1,4,17	1,6,1	1,5,1	1,6,4	1,5,4	1,7,1	2,7,2	2,6,2	1,8,1	2,5,2
134	2,1,2	2,2,2	2,3,2	2,4,2	2,7,2	2,6,2	2,4,1	2,3,1	2,1,1	2,2,1
135	2,3,2	2,4,2	2,1,2	1,4,2	1,3,2	1,1,2	2,2,2	1,2,2	1,6,2	2,6,2
136	1,4,7	2,4,7	1,4,8	1,4,9	2,4,8	2,4,9	2,2,5	2,7,2	1,2,5	1,1,1
137	2,4,7	1,4,7	2,6,2	2,9,2	1,2,2	2,4,2	2,3,2	2,2,2	1,6,2	1,9,2
138	1,4,7	2,4,7	1,6,2	1,2,2	1,7,2	1,1,1	1,3,1	1,4,1	2,6,1	1,2,1
139	2,6,2	1,6,2	2,7,2	1,7,2	1,1,2	1,3,2	1,4,2	1,2,2	2,9,2	2,2,2
140	1,7,2	1,6,2	1,2,2	2,7,2	1,3,2	1,4,2	1,1,2	2,6,2	2,2,2	1,2,9
141	1,9,2	1,7,2	1,2,2	1,1,1	1,6,2	1,1,4	1,5,5	1,6,5	2,1,1	2,7,2
142	2,4,7	1,4,7	1,2,2	2,4,1	2,3,1	2,3,4	2,4,4	1,3,2	1,4,2	1,4,5
143	2,1,5	2,3,1	2,4,1	2,4,4	2,3,4	2,5,2	2,3,5	2,4,5	1,5,2	2,1,1
144	2,1,2	2,4,2	2,3,2	1,4,2	1,3,2	1,1,2	2,2,2	1,2,2	2,6,2	1,7,2
145	1,7,1	1,9,1	1,8,1	2,6,1	1,2,1	1,4,1	1,3,1	1,7,4	1,9,4	1,8,4
146	1,4,7	2,4,7	2,1,2	1,1,2	1,5,5	1,6,5	2,4,2	2,3,2	1,7,1	1,9,1
147	1,4,7	2,4,7	2,7,1	2,9,1	2,8,1	2,9,4	2,8,4	2,7,4	2,7,2	1,2,1
148	2,4,7	1,4,7	1,5,2	2,5,2	1,8,2	1,7,1	1,9,1	2,4,1	2,3,1	2,2,1
149	1,4,7	2,4,7	1,1,2	1,3,2	1,4,2	1,6,2	2,6,2	1,2,2	2,1,2	2,3,2
150	1,4,7	2,4,7	1,7,2	2,9,2	1,5,2	2,7,2	1,8,2	1,9,2	1,8,8	2,5,2

Źródło: opracowanie własne.

Tablica A.23

Współczynniki korelacji rang Kendalla

Współczynniki korelacji rang Kendalla									
Modyfikacja 1		Modyfikacja 2		Modyfikacja 3		Modyfikacja 4		Modyfikacja 5	
Ścieżka	τ	Ścieżka	τ	Ścieżka	τ	Ścieżka	τ	Ścieżka	τ
2,5,7	1,000	1,4,5	0,867	1,4,5	0,867	1,6,8	1,000	1,5,13	1,000
2,2,17	1,000	1,9,5	0,867	1,9,5	0,867	1,7,8	0,956	1,3,12	1,000
2,6,13	1,000	1,1,5	0,867	1,1,5	0,867	1,6,15	0,956	1,5,2	1,000
1,1,8	1,000	2,7,5	0,867	2,7,5	0,867	1,6,3	0,956	1,4,12	1,000
2,4,16	1,000	2,9,5	0,867	2,9,5	0,867	1,7,3	0,956	1,2,12	1,000
1,6,8	1,000	1,7,5	0,867	1,7,5	0,867	1,6,6	0,911	2,8,5	1,000
1,7,6	1,000	1,8,5	0,867	1,8,5	0,867	1,7,15	0,911	1,5,12	1,000
1,2,3	1,000	1,3,5	0,867	1,3,5	0,867	1,6,13	0,911	1,1,12	1,000
2,4,8	1,000	2,8,5	0,867	2,8,5	0,867	1,7,2	0,867	1,6,12	1,000
2,2,15	1,000	2,3,5	0,822	2,3,5	0,822	1,7,6	0,867	2,8,9	1,000
1,4,11	1,000	1,2,5	0,822	1,2,5	0,822	1,6,2	0,867	2,9,5	1,000
1,1,13	1,000	2,4,5	0,822	2,4,5	0,822	1,6,9	0,867	2,7,5	1,000
1,7,10	1,000	2,2,5	0,822	2,2,5	0,822	1,7,16	0,867	2,1,5	1,000
1,5,5	1,000	2,1,5	0,822	2,1,5	0,822	1,6,16	0,867	2,9,9	1,000
1,3,6	1,000	2,8,6	0,822	2,8,6	0,822	1,7,9	0,867	2,4,5	1,000
1,8,12	1,000	2,8,4	0,778	2,8,4	0,778	1,7,13	0,822	2,2,5	1,000
1,2,7	1,000	2,9,4	0,778	2,9,4	0,778	1,5,17	0,733	2,5,9	1,000
1,1,5	1,000	2,9,11	0,778	2,9,11	0,778	1,8,17	0,689	1,5,15	1,000
1,4,5	1,000	2,9,7	0,778	2,9,7	0,778	2,2,12	0,644	2,5,13	1,000
2,7,17	1,000	2,7,4	0,778	2,7,4	0,778	1,9,10	0,644	2,3,5	1,000
1,9,11	1,000	2,7,7	0,778	2,7,7	0,778	2,9,10	0,644	2,5,7	0,956
2,9,15	1,000	1,8,6	0,778	1,8,6	0,778	2,9,9	0,600	2,5,1	0,956
1,4,9	1,000	2,8,11	0,778	2,8,11	0,778	1,9,3	0,600	2,4,11	0,956
2,1,15	1,000	2,8,1	0,778	2,8,1	0,778	1,9,8	0,600	1,7,4	0,956
1,2,4	1,000	2,7,11	0,778	2,7,11	0,778	2,9,2	0,556	2,8,12	0,956
2,3,17	1,000	2,7,1	0,778	2,7,1	0,778	2,5,9	0,556	2,5,4	0,956
2,8,11	1,000	2,7,14	0,778	2,7,14	0,778	2,8,13	0,556	1,3,1	0,956
1,3,16	1,000	2,9,1	0,778	2,9,1	0,778	2,5,13	0,556	1,2,7	0,956
2,3,13	1,000	2,8,14	0,778	2,8,14	0,778	2,8,8	0,556	1,8,7	0,956
2,8,17	1,000	2,9,14	0,778	2,9,14	0,778	2,8,9	0,556	2,3,1	0,956
2,4,12	1,000	2,8,7	0,778	2,8,7	0,778	2,5,8	0,556	1,6,7	0,956
1,1,15	1,000	1,5,11	0,733	1,5,11	0,733	2,9,13	0,556	1,2,5	0,956
2,5,11	1,000	2,3,12	0,733	2,3,12	0,733	2,2,11	0,556	1,9,13	0,956
1,3,9	1,000	1,6,11	0,733	1,6,11	0,733	2,2,7	0,556	1,3,4	0,956
2,6,10	1,000	1,9,7	0,733	1,9,7	0,733	2,2,1	0,556	1,1,11	0,956
1,4,17	1,000	1,9,14	0,733	1,9,14	0,733	2,2,4	0,556	2,4,12	0,956
1,5,2	1,000	1,6,4	0,733	1,6,4	0,733	2,2,14	0,556	2,6,7	0,956
1,5,4	1,000	1,9,4	0,733	1,9,4	0,733	2,5,2	0,556	2,2,14	0,956
1,8,4	1,000	1,6,1	0,733	1,6,1	0,733	2,8,15	0,511	1,6,11	0,956
1,9,16	1,000	1,9,1	0,733	1,9,1	0,733	2,8,2	0,511	1,3,14	0,956
1,6,4	1,000	1,6,14	0,733	1,6,14	0,733	1,9,6	0,511	2,3,7	0,956
1,5,16	1,000	2,2,12	0,733	2,2,12	0,733	2,8,12	0,511	2,8,15	0,956
1,1,11	1,000	2,5,5	0,733	2,5,5	0,733	2,5,15	0,511	1,9,7	0,956
1,2,5	1,000	1,7,4	0,733	1,7,4	0,733	2,9,12	0,511	2,4,4	0,956
2,7,8	1,000	2,8,10	0,733	2,8,10	0,733	2,7,12	0,511	1,4,1	0,956
1,3,12	1,000	2,1,12	0,733	2,1,12	0,733	1,8,13	0,511	1,7,7	0,956
1,6,6	1,000	1,6,7	0,733	1,6,7	0,733	2,3,12	0,511	2,6,1	0,956
1,1,1	1,000	1,7,11	0,733	1,7,11	0,733	1,5,9	0,511	2,5,11	0,956
2,3,9	1,000	1,7,14	0,733	1,7,14	0,733	2,9,16	0,511	2,5,3	0,956
2,2,13	1,000	2,4,12	0,733	2,4,12	0,733	1,9,15	0,511	1,1,14	0,956
2,6,11	1,000	2,6,5	0,733	2,6,5	0,733	2,4,12	0,511	1,7,11	0,956
1,6,9	1,000	1,8,14	0,733	1,8,14	0,733	2,5,3	0,511	2,7,1	0,956
2,2,10	1,000	1,5,7	0,733	1,5,7	0,733	2,5,12	0,467	2,9,4	0,956
1,6,5	1,000	1,8,11	0,733	1,8,11	0,733	2,5,17	0,467	1,8,1	0,956
2,2,2	1,000	1,8,7	0,733	1,8,7	0,733	1,9,13	0,467	1,4,11	0,956
1,5,14	1,000	1,5,1	0,733	1,5,1	0,733	1,5,12	0,467	1,9,14	0,956
2,2,9	1,000	1,8,4	0,733	1,8,4	0,733	1,5,13	0,467	2,9,13	0,956
2,6,9	1,000	1,5,4	0,733	1,5,4	0,733	2,8,16	0,467	2,4,7	0,956

2,9,2	1,000	1,7,7	0,733	1,7,7	0,733	2,5,6	0,467	1,8,11	0,956
1,7,14	1,000	1,9,11	0,733	1,9,11	0,733	2,8,3	0,467	1,4,14	0,956
1,5,7	1,000	1,7,1	0,733	1,7,1	0,733	2,5,16	0,467	2,9,12	0,956
1,2,13	1,000	1,5,14	0,733	1,5,14	0,733	1,5,15	0,467	1,9,1	0,956
1,1,4	1,000	1,8,1	0,733	1,8,1	0,733	1,2,9	0,467	1,4,4	0,956
1,8,5	1,000	2,7,12	0,689	2,7,12	0,689	1,6,10	0,467	2,6,11	0,956
1,9,6	1,000	2,9,12	0,689	2,9,12	0,689	1,6,12	0,467	1,1,4	0,956
2,7,5	1,000	2,8,12	0,689	2,8,12	0,689	2,6,12	0,467	2,7,7	0,956
2,1,3	1,000	1,6,5	0,689	1,6,5	0,689	2,5,10	0,467	2,7,12	0,956
2,7,13	1,000	1,5,5	0,689	1,5,5	0,689	2,6,5	0,422	2,2,1	0,956
1,4,14	1,000	1,8,12	0,644	1,8,12	0,644	1,4,14	0,422	2,9,15	0,956
2,8,4	1,000	2,3,4	0,644	2,3,4	0,644	1,4,11	0,422	1,5,1	0,956
1,4,12	1,000	2,4,1	0,644	2,4,1	0,644	2,6,4	0,422	2,7,11	0,956
2,8,12	1,000	2,3,14	0,644	2,3,14	0,644	1,3,1	0,422	2,2,12	0,956
1,3,14	1,000	2,3,7	0,644	2,3,7	0,644	1,7,17	0,422	2,2,7	0,956
2,9,7	1,000	2,1,1	0,644	2,1,1	0,644	2,6,14	0,422	2,9,16	0,956
1,6,3	1,000	2,1,4	0,644	2,1,4	0,644	2,8,17	0,422	2,3,14	0,956
2,9,11	1,000	2,5,9	0,644	2,5,9	0,644	1,3,12	0,422	1,6,14	0,956
1,1,6	1,000	2,1,7	0,644	2,1,7	0,644	1,9,2	0,422	2,5,2	0,956
1,3,4	1,000	2,4,14	0,644	2,4,14	0,644	1,8,9	0,422	1,5,11	0,956
2,7,2	1,000	2,1,11	0,644	2,1,11	0,644	1,4,17	0,422	2,4,1	0,956
2,1,5	1,000	1,4,12	0,644	1,4,12	0,644	1,3,4	0,422	1,7,14	0,956
1,1,12	1,000	2,2,1	0,644	2,2,1	0,644	1,4,7	0,422	2,2,11	0,956
2,1,9	1,000	2,2,4	0,644	2,2,4	0,644	1,2,16	0,422	1,3,7	0,956
2,8,8	1,000	1,1,12	0,644	1,1,12	0,644	1,2,3	0,422	1,5,14	0,956
2,1,13	1,000	2,2,7	0,644	2,2,7	0,644	2,5,7	0,422	2,8,6	0,956
2,8,5	1,000	2,2,11	0,644	2,2,11	0,644	2,5,4	0,422	1,8,17	0,956
2,1,17	1,000	2,2,14	0,644	2,2,14	0,644	1,4,12	0,422	1,4,7	0,956
2,7,4	1,000	1,9,12	0,644	1,9,12	0,644	1,3,11	0,422	1,3,11	0,956
1,6,2	1,000	2,3,1	0,644	2,3,1	0,644	2,9,15	0,422	2,8,4	0,956
1,6,16	1,000	2,3,11	0,644	2,3,11	0,644	1,2,2	0,422	2,5,14	0,956
2,2,12	1,000	1,3,12	0,644	1,3,12	0,644	1,3,7	0,422	1,1,1	0,956
1,5,1	1,000	2,1,14	0,644	2,1,14	0,644	1,3,17	0,422	1,2,11	0,956
2,2,16	1,000	2,4,4	0,644	2,4,4	0,644	2,6,11	0,422	1,2,1	0,956
2,9,4	1,000	2,4,11	0,644	2,4,11	0,644	1,3,14	0,422	1,1,5	0,956
2,3,3	1,000	2,4,7	0,644	2,4,7	0,644	1,4,1	0,422	1,1,7	0,956
1,7,1	1,000	1,7,12	0,644	1,7,12	0,644	1,4,4	0,422	1,9,4	0,956
1,6,12	1,000	1,2,12	0,644	1,2,12	0,644	2,4,17	0,422	1,8,14	0,956
2,7,9	1,000	2,8,9	0,600	2,8,9	0,600	2,5,1	0,422	2,8,7	0,956
2,2,5	1,000	1,4,14	0,600	1,4,14	0,600	2,5,5	0,422	2,3,11	0,956
2,6,7	1,000	1,2,14	0,600	1,2,14	0,600	2,6,7	0,422	2,8,14	0,956
1,6,14	1,000	1,8,16	0,600	1,8,16	0,600	2,5,11	0,422	2,4,14	0,956
2,7,16	1,000	1,1,17	0,600	1,1,17	0,600	2,5,14	0,422	2,1,12	0,956
2,3,15	1,000	1,5,6	0,600	1,5,6	0,600	2,6,1	0,422	1,7,5	0,956
1,9,17	1,000	1,2,11	0,600	1,2,11	0,600	2,3,17	0,422	2,8,11	0,956
2,4,2	1,000	1,3,4	0,600	1,3,4	0,600	1,2,8	0,422	1,9,5	0,956
1,2,9	1,000	1,1,14	0,600	1,1,14	0,600	2,6,10	0,422	1,5,4	0,956
1,8,1	1,000	1,1,7	0,600	1,1,7	0,600	1,2,6	0,378	1,7,1	0,956
2,4,10	1,000	2,5,14	0,600	2,5,14	0,600	2,7,7	0,378	1,4,5	0,956
1,9,1	1,000	1,3,7	0,600	1,3,7	0,600	2,7,10	0,378	2,5,8	0,956
1,7,7	1,000	1,4,4	0,600	1,4,4	0,600	2,7,11	0,378	2,9,14	0,956
1,2,15	1,000	1,2,4	0,600	1,2,4	0,600	2,7,13	0,378	1,5,7	0,956
2,5,1	1,000	2,5,11	0,600	2,5,11	0,600	1,8,3	0,378	2,5,16	0,956
1,4,2	1,000	1,1,11	0,600	1,1,11	0,600	2,8,1	0,378	1,2,14	0,956
2,5,5	1,000	1,3,11	0,600	1,3,11	0,600	1,7,12	0,378	2,8,16	0,956
1,1,3	1,000	1,2,1	0,600	1,2,1	0,600	2,7,14	0,378	2,9,1	0,956
2,2,3	1,000	1,4,11	0,600	1,4,11	0,600	2,8,6	0,378	2,6,4	0,956
1,7,13	1,000	1,1,4	0,600	1,1,4	0,600	1,8,15	0,378	2,7,4	0,956
1,9,13	1,000	1,1,1	0,600	1,1,1	0,600	2,9,1	0,378	1,6,4	0,956
1,7,15	1,000	2,5,12	0,600	2,5,12	0,600	1,8,12	0,378	2,2,4	0,956
2,7,1	1,000	1,3,14	0,600	1,3,14	0,600	2,9,14	0,378	2,9,7	0,956
1,4,1	1,000	1,2,7	0,600	1,2,7	0,600	1,9,16	0,378	1,2,4	0,956
2,6,8	1,000	2,6,1	0,600	2,6,1	0,600	1,5,10	0,378	1,8,5	0,956
2,8,1	1,000	2,6,7	0,600	2,6,7	0,600	2,7,1	0,378	2,1,14	0,956

2,6,12	1,000	1,4,1	0,600	1,4,1	0,600	2,8,11	0,378	2,7,14	0,956
2,6,16	1,000	2,5,1	0,600	2,5,1	0,600	2,9,6	0,378	1,8,9	0,956
1,4,4	1,000	2,5,4	0,600	2,5,4	0,600	2,9,4	0,378	2,9,11	0,956
2,9,14	1,000	2,5,7	0,600	2,5,7	0,600	2,9,7	0,378	1,3,5	0,956
1,3,8	1,000	1,4,7	0,600	1,4,7	0,600	2,3,11	0,378	2,1,1	0,956
2,6,5	1,000	1,3,1	0,600	1,3,1	0,600	1,9,12	0,378	1,9,11	0,956
1,6,7	1,000	2,6,4	0,600	2,6,4	0,600	2,3,1	0,378	1,5,10	0,956
1,3,10	1,000	2,6,11	0,600	2,6,11	0,600	2,3,4	0,378	2,1,7	0,956
1,2,16	1,000	2,6,12	0,600	2,6,12	0,600	2,9,11	0,378	1,9,9	0,956
2,7,6	1,000	2,6,14	0,600	2,6,14	0,600	2,7,4	0,378	2,8,1	0,956
1,4,16	1,000	1,8,9	0,556	1,8,9	0,556	2,3,7	0,378	2,1,4	0,956
1,1,10	1,000	1,2,17	0,556	1,2,17	0,556	2,8,14	0,378	2,1,11	0,956
2,1,16	1,000	2,8,13	0,556	2,8,13	0,556	2,3,13	0,378	2,9,2	0,956
1,9,15	1,000	1,4,17	0,556	1,4,17	0,556	2,3,14	0,378	1,6,1	0,956
1,2,11	1,000	1,1,10	0,556	1,1,10	0,556	2,8,7	0,378	2,6,14	0,956
1,9,8	1,000	2,1,10	0,556	2,1,10	0,556	2,4,4	0,378	2,3,12	0,956
1,3,1	1,000	2,1,17	0,556	2,1,17	0,556	2,4,7	0,378	1,8,4	0,956
2,7,14	1,000	1,3,10	0,556	1,3,10	0,556	1,9,9	0,378	2,5,15	0,956
2,9,12	1,000	2,2,10	0,556	2,2,10	0,556	2,4,13	0,378	2,3,4	0,956
1,2,1	1,000	2,3,10	0,556	2,3,10	0,556	2,4,14	0,378	2,9,17	0,911
2,9,16	1,000	1,2,10	0,556	1,2,10	0,556	2,4,11	0,378	1,8,15	0,911
2,6,2	1,000	2,3,17	0,556	2,3,17	0,556	1,7,10	0,378	1,5,9	0,911
1,5,15	1,000	1,3,17	0,556	1,3,17	0,556	2,8,4	0,378	2,8,2	0,911
2,6,3	1,000	2,4,10	0,556	2,4,10	0,556	2,4,1	0,378	2,8,13	0,911
1,7,2	1,000	2,4,17	0,556	2,4,17	0,556	2,7,2	0,333	1,5,3	0,911
1,5,6	1,000	2,5,6	0,556	2,5,6	0,556	1,2,15	0,333	1,9,15	0,911
2,7,12	1,000	1,4,10	0,556	1,4,10	0,556	2,7,5	0,333	1,5,8	0,911
1,9,14	1,000	2,8,8	0,511	2,8,8	0,511	2,7,9	0,333	1,1,10	0,911
2,9,6	1,000	1,9,10	0,511	1,9,10	0,511	2,8,5	0,333	2,1,10	0,911
2,2,6	1,000	1,5,9	0,511	1,5,9	0,511	2,6,9	0,333	1,3,10	0,911
2,1,12	1,000	1,5,12	0,511	1,5,12	0,511	1,5,3	0,333	2,2,10	0,911
2,8,16	1,000	1,5,13	0,511	1,5,13	0,511	2,6,2	0,333	2,4,10	0,911
1,3,3	1,000	1,5,16	0,511	1,5,16	0,511	2,9,3	0,333	1,4,10	0,911
1,1,14	1,000	2,2,17	0,511	2,2,17	0,511	2,9,5	0,333	1,6,10	0,911
1,5,11	1,000	1,6,12	0,511	1,6,12	0,511	2,7,6	0,333	2,3,10	0,911
2,8,9	1,000	2,8,16	0,511	2,8,16	0,511	2,2,13	0,333	1,2,10	0,911
1,4,8	1,000	1,8,13	0,511	1,8,13	0,511	2,7,16	0,333	2,5,12	0,911
2,3,5	1,000	2,5,10	0,511	2,5,10	0,511	2,6,16	0,333	2,8,3	0,911
1,5,12	1,000	2,5,16	0,511	2,5,16	0,511	1,2,12	0,333	1,8,3	0,911
1,6,11	1,000	1,8,10	0,467	1,8,10	0,467	2,6,6	0,333	2,6,10	0,911
2,1,10	1,000	1,8,15	0,467	1,8,15	0,467	1,4,13	0,289	2,6,12	0,911
2,6,1	1,000	2,8,15	0,467	2,8,15	0,467	1,2,17	0,289	2,8,10	0,867
2,1,8	1,000	2,8,17	0,467	2,8,17	0,467	2,9,8	0,289	2,8,17	0,867
2,1,6	1,000	1,8,8	0,467	1,8,8	0,467	2,3,9	0,289	2,6,5	0,867
2,5,16	1,000	2,9,10	0,467	2,9,10	0,467	2,2,6	0,289	1,9,8	0,867
1,3,5	1,000	2,7,17	0,467	2,7,17	0,467	2,2,16	0,289	2,9,3	0,867
2,1,2	1,000	1,8,2	0,467	1,8,2	0,467	1,3,13	0,289	1,9,12	0,867
2,5,14	1,000	2,6,10	0,467	2,6,10	0,467	1,6,17	0,289	2,9,8	0,867
1,1,2	1,000	2,7,10	0,422	2,7,10	0,422	2,4,9	0,289	1,7,12	0,867
2,9,17	1,000	2,9,16	0,422	2,9,16	0,422	2,6,13	0,289	1,9,16	0,867
1,4,7	1,000	2,8,2	0,422	2,8,2	0,422	1,6,5	0,244	2,5,5	0,867
2,9,13	1,000	1,5,10	0,422	1,5,10	0,422	2,3,2	0,244	1,9,17	0,867
2,5,12	1,000	1,9,16	0,422	1,9,16	0,422	2,3,16	0,244	1,5,16	0,867
2,9,9	1,000	1,5,15	0,422	1,5,15	0,422	2,4,16	0,244	1,8,2	0,867
2,9,5	1,000	1,6,10	0,422	1,6,10	0,422	1,3,16	0,244	2,2,17	0,867
1,4,6	1,000	2,5,8	0,422	2,5,8	0,422	2,4,2	0,244	2,5,6	0,867
1,9,10	1,000	2,5,13	0,422	2,5,13	0,422	2,6,8	0,244	1,8,6	0,867
2,9,1	1,000	1,5,8	0,378	1,5,8	0,378	2,7,8	0,244	1,8,13	0,867
2,8,14	1,000	1,5,2	0,378	1,5,2	0,378	1,4,16	0,244	1,9,3	0,867
2,5,8	1,000	1,5,17	0,378	1,5,17	0,378	1,1,17	0,244	1,9,2	0,867
1,9,5	1,000	1,8,3	0,378	1,8,3	0,378	1,5,8	0,244	1,8,16	0,867
1,7,16	1,000	2,5,2	0,378	2,5,2	0,378	1,5,2	0,244	1,8,12	0,867
2,3,2	1,000	2,5,15	0,378	2,5,15	0,378	2,2,9	0,244	2,7,10	0,867
2,6,14	1,000	1,7,10	0,378	1,7,10	0,378	1,5,5	0,244	2,5,17	0,822

2,6,15	1,000	1,7,17	0,378	1,7,17	0,378	1,9,17	0,244	1,8,8	0,822
1,1,17	1,000	2,6,17	0,333	2,6,17	0,333	2,2,2	0,244	2,5,10	0,822
2,3,6	1,000	2,5,17	0,333	2,5,17	0,333	1,8,8	0,200	1,7,10	0,822
1,3,2	1,000	2,9,13	0,333	2,9,13	0,333	1,2,13	0,200	1,6,17	0,822
2,9,3	1,000	1,8,17	0,333	1,8,17	0,333	2,8,10	0,200	1,3,17	0,822
1,1,9	1,000	2,9,6	0,333	2,9,6	0,333	1,4,2	0,200	1,5,6	0,822
2,3,10	1,000	1,9,13	0,289	1,9,13	0,289	2,4,6	0,200	2,9,10	0,822
2,8,15	1,000	2,9,2	0,289	2,9,2	0,289	1,3,2	0,200	2,8,8	0,822
1,4,13	1,000	1,9,9	0,289	1,9,9	0,289	1,3,5	0,200	1,4,17	0,822
1,9,9	1,000	2,9,9	0,289	2,9,9	0,289	1,5,16	0,200	1,2,17	0,822
1,6,15	1,000	2,8,3	0,244	2,8,3	0,244	2,3,5	0,200	2,1,17	0,822
1,9,7	1,000	1,6,17	0,244	1,6,17	0,244	2,3,6	0,200	2,3,17	0,822
1,9,3	1,000	2,9,15	0,244	2,9,15	0,244	2,4,5	0,200	2,4,17	0,822
1,7,11	1,000	1,9,2	0,200	1,9,2	0,200	1,4,5	0,200	1,7,17	0,822
1,9,2	1,000	1,9,6	0,200	1,9,6	0,200	2,7,3	0,156	1,6,3	0,778
2,8,2	1,000	1,5,3	0,200	1,5,3	0,200	1,8,11	0,156	1,6,5	0,778
1,2,6	1,000	2,5,3	0,156	2,5,3	0,156	1,8,14	0,156	2,2,8	0,778
1,3,17	1,000	1,9,15	0,111	1,9,15	0,111	1,8,10	0,156	2,2,9	0,778
1,8,16	1,000	2,4,13	0,067	2,4,13	0,067	2,7,15	0,156	1,6,8	0,778
2,9,8	1,000	2,3,13	0,067	2,3,13	0,067	1,5,1	0,156	2,1,16	0,778
1,2,8	1,000	2,1,13	0,067	2,1,13	0,067	1,9,1	0,156	2,6,16	0,778
2,4,5	1,000	2,9,17	0,067	2,9,17	0,067	1,9,4	0,156	2,1,8	0,778
1,2,17	1,000	2,2,13	0,067	2,2,13	0,067	1,4,9	0,156	2,6,9	0,778
2,4,6	1,000	2,7,6	0,022	2,7,6	0,022	1,5,6	0,156	2,6,6	0,778
1,2,10	1,000	2,7,13	0,022	2,7,13	0,022	1,9,7	0,156	2,1,6	0,778
2,4,9	1,000	2,1,6	0,022	2,1,6	0,022	1,5,4	0,156	2,6,3	0,778
1,8,14	1,000	1,3,6	0,022	1,3,6	0,022	1,9,11	0,156	2,2,2	0,778
1,8,9	1,000	2,2,6	0,022	2,2,6	0,022	1,5,11	0,156	1,6,6	0,778
1,1,7	1,000	2,3,6	0,022	2,3,6	0,022	1,3,3	0,156	2,2,16	0,778
2,4,13	1,000	1,1,6	0,022	1,1,6	0,022	1,5,7	0,156	2,4,16	0,778
1,7,17	1,000	1,4,6	0,022	1,4,6	0,022	1,9,14	0,156	1,2,6	0,778
1,2,14	1,000	2,6,6	0,022	2,6,6	0,022	1,3,6	0,156	2,3,3	0,778
2,4,17	1,000	2,4,6	0,022	2,4,6	0,022	1,7,14	0,156	1,6,15	0,778
2,7,3	1,000	1,7,6	-0,022	1,7,6	-0,022	1,7,4	0,156	2,3,2	0,778
1,8,11	1,000	1,2,6	-0,022	1,2,6	-0,022	1,5,14	0,156	1,6,9	0,778
2,7,11	1,000	1,9,17	-0,022	1,9,17	-0,022	1,6,11	0,156	1,3,13	0,778
2,5,4	1,000	2,9,8	-0,022	2,9,8	-0,022	1,3,8	0,156	2,4,8	0,778
2,7,15	1,000	1,9,3	-0,067	1,9,3	-0,067	2,1,17	0,156	1,1,8	0,778
2,7,7	1,000	1,1,8	-0,067	1,1,8	-0,067	2,2,8	0,156	2,1,9	0,778
1,7,9	1,000	2,9,3	-0,067	2,9,3	-0,067	1,7,7	0,156	2,3,6	0,778
1,8,7	1,000	1,3,8	-0,067	1,3,8	-0,067	1,4,6	0,156	1,4,13	0,778
1,4,10	1,000	1,2,8	-0,067	1,2,8	-0,067	1,6,1	0,156	2,3,16	0,778
2,4,15	1,000	1,7,8	-0,067	1,7,8	-0,067	1,3,9	0,156	2,3,8	0,778
2,6,4	1,000	1,4,8	-0,067	1,4,8	-0,067	2,2,5	0,156	1,1,6	0,778
2,7,10	1,000	2,3,9	-0,111	2,3,9	-0,111	1,6,4	0,156	2,7,2	0,778
1,4,3	1,000	2,3,16	-0,111	2,3,16	-0,111	1,6,7	0,156	2,3,9	0,778
1,2,12	1,000	2,4,8	-0,111	2,4,8	-0,111	1,7,1	0,156	1,2,8	0,778
1,7,12	1,000	2,4,2	-0,111	2,4,2	-0,111	2,2,17	0,156	2,2,6	0,778
1,7,5	1,000	1,1,13	-0,111	1,1,13	-0,111	1,6,14	0,156	1,6,16	0,778
1,7,8	1,000	2,6,9	-0,111	2,6,9	-0,111	1,4,3	0,156	1,1,3	0,778
1,7,4	1,000	2,4,9	-0,111	2,4,9	-0,111	1,7,11	0,156	1,8,10	0,778
1,7,3	1,000	2,4,16	-0,111	2,4,16	-0,111	1,4,8	0,156	2,4,2	0,778
1,1,16	1,000	2,6,16	-0,111	2,6,16	-0,111	2,6,3	0,156	2,4,6	0,778
2,8,7	1,000	1,2,13	-0,111	1,2,13	-0,111	1,8,1	0,156	2,7,9	0,778
1,3,15	1,000	2,6,8	-0,111	2,6,8	-0,111	1,8,2	0,156	1,7,3	0,778
2,4,3	1,000	2,6,13	-0,111	2,6,13	-0,111	1,8,4	0,156	1,1,13	0,778
1,3,13	1,000	2,6,2	-0,111	2,6,2	-0,111	1,8,7	0,156	2,4,9	0,778
1,3,11	1,000	2,1,8	-0,111	2,1,8	-0,111	2,3,15	0,111	1,4,3	0,778
2,3,16	1,000	2,7,2	-0,111	2,7,2	-0,111	1,2,4	0,111	1,7,8	0,778
2,9,10	1,000	1,3,13	-0,111	1,3,13	-0,111	1,2,1	0,111	1,9,10	0,778
2,2,8	1,000	2,7,9	-0,111	2,7,9	-0,111	2,2,15	0,111	1,7,13	0,778
1,6,1	1,000	2,7,8	-0,111	2,7,8	-0,111	1,2,7	0,111	1,4,6	0,778
2,6,6	1,000	2,7,16	-0,111	2,7,16	-0,111	2,4,15	0,111	1,4,8	0,778
1,6,10	1,000	1,9,8	-0,111	1,9,8	-0,111	2,6,15	0,111	2,7,8	0,778

1,9,4	1,000	2,2,16	-0,111	2,2,16	-0,111	1,2,11	0,111	1,6,2	0,778
1,8,8	1,000	2,1,9	-0,111	2,1,9	-0,111	1,2,14	0,111	1,5,5	0,778
1,2,2	1,000	1,4,13	-0,111	1,4,13	-0,111	2,3,8	0,067	1,7,6	0,778
2,3,12	1,000	2,1,16	-0,111	2,1,16	-0,111	2,3,3	0,067	2,6,2	0,778
1,3,7	1,000	2,1,2	-0,111	2,1,2	-0,111	1,3,15	0,067	1,3,3	0,778
1,9,12	1,000	2,2,8	-0,111	2,2,8	-0,111	1,8,16	0,067	1,2,13	0,778
2,3,8	1,000	2,2,2	-0,111	2,2,2	-0,111	1,4,15	0,067	1,3,6	0,778
1,4,15	1,000	2,2,9	-0,111	2,2,9	-0,111	1,9,5	0,067	2,1,2	0,778
2,3,14	0,944	1,6,6	-0,111	1,6,6	-0,111	2,2,3	0,067	2,6,8	0,778
1,8,17	0,944	1,6,8	-0,111	1,6,8	-0,111	2,4,8	0,067	2,6,17	0,778
2,1,11	0,944	2,3,2	-0,111	2,3,2	-0,111	1,8,5	0,067	2,1,3	0,778
2,4,1	0,944	2,3,8	-0,111	2,3,8	-0,111	2,4,3	0,067	1,1,17	0,778
2,2,4	0,944	1,7,13	-0,156	1,7,13	-0,156	1,7,5	0,067	2,7,3	0,778
2,4,4	0,944	1,2,2	-0,200	1,2,2	-0,200	1,8,6	0,022	1,2,3	0,778
2,3,11	0,944	1,6,2	-0,200	1,6,2	-0,200	2,4,10	-0,022	1,3,8	0,778
1,6,13	0,944	1,6,9	-0,200	1,6,9	-0,200	2,9,17	-0,022	2,7,16	0,778
1,5,3	0,944	1,6,16	-0,200	1,6,16	-0,200	1,3,10	-0,022	2,2,3	0,778
2,6,17	0,944	1,7,9	-0,200	1,7,9	-0,200	2,3,10	-0,022	2,4,3	0,778
2,1,1	0,944	1,1,16	-0,200	1,1,16	-0,200	2,1,10	-0,022	1,7,9	0,733
2,8,3	0,944	1,7,2	-0,200	1,7,2	-0,200	1,4,10	-0,022	1,7,16	0,733
2,1,4	0,944	1,3,9	-0,200	1,3,9	-0,200	2,7,17	-0,111	1,1,2	0,733
1,8,15	0,944	1,4,2	-0,200	1,4,2	-0,200	2,2,10	-0,111	1,6,13	0,733
2,3,4	0,944	1,4,9	-0,200	1,4,9	-0,200	1,2,5	-0,111	1,2,15	0,733
2,5,10	0,944	1,1,9	-0,200	1,1,9	-0,200	2,6,17	-0,111	1,5,17	0,733
2,8,10	0,944	1,7,16	-0,200	1,7,16	-0,200	2,1,3	-0,200	2,1,15	0,733
2,5,9	0,944	1,2,9	-0,200	1,2,9	-0,200	2,1,8	-0,200	1,4,16	0,733
2,2,1	0,944	1,3,16	-0,200	1,3,16	-0,200	2,1,5	-0,200	1,2,2	0,733
2,4,7	0,944	1,3,2	-0,200	1,3,2	-0,200	1,2,10	-0,200	1,2,16	0,733
1,8,2	0,944	1,1,2	-0,200	1,1,2	-0,200	1,1,8	-0,244	1,7,15	0,733
2,8,6	0,944	1,2,16	-0,200	1,2,16	-0,200	2,1,6	-0,244	1,7,2	0,733
1,6,17	0,944	1,4,16	-0,200	1,4,16	-0,200	2,1,15	-0,289	2,6,13	0,733
1,8,6	0,944	2,7,15	-0,289	2,7,15	-0,289	2,1,16	-0,333	2,1,13	0,733
2,1,14	0,944	2,3,3	-0,289	2,3,3	-0,289	2,1,2	-0,378	2,3,13	0,733
2,2,11	0,944	2,6,3	-0,289	2,6,3	-0,289	1,1,5	-0,378	1,3,9	0,733
1,8,13	0,944	1,4,15	-0,289	1,4,15	-0,289	2,1,9	-0,422	1,3,16	0,733
2,2,7	0,944	1,6,13	-0,289	1,6,13	-0,289	1,1,6	-0,422	1,1,9	0,733
2,4,11	0,944	1,1,3	-0,289	1,1,3	-0,289	1,1,10	-0,422	1,2,9	0,733
2,2,14	0,944	1,2,3	-0,289	1,2,3	-0,289	1,1,16	-0,422	2,7,15	0,733
1,5,13	0,944	1,1,15	-0,289	1,1,15	-0,289	2,1,11	-0,467	2,7,13	0,733
2,1,7	0,944	2,6,15	-0,289	2,6,15	-0,289	2,1,14	-0,467	2,4,15	0,733
2,8,13	0,944	2,3,15	-0,289	2,3,15	-0,289	2,1,1	-0,467	1,1,16	0,733
2,5,3	0,944	1,4,3	-0,289	1,4,3	-0,289	1,1,2	-0,467	2,6,15	0,733
1,5,8	0,944	2,7,3	-0,289	2,7,3	-0,289	2,1,7	-0,467	2,2,15	0,733
2,5,13	0,944	2,2,3	-0,289	2,2,3	-0,289	1,1,3	-0,467	1,1,15	0,733
2,3,1	0,944	1,7,3	-0,289	1,7,3	-0,289	2,1,4	-0,467	2,7,17	0,733
2,3,7	0,944	2,1,3	-0,289	2,1,3	-0,289	1,1,9	-0,511	1,4,2	0,733
1,5,17	0,944	1,7,15	-0,289	1,7,15	-0,289	1,1,15	-0,511	2,3,15	0,733
2,5,6	0,944	1,3,15	-0,289	1,3,15	-0,289	1,1,12	-0,511	2,9,6	0,733
1,8,3	0,944	1,2,15	-0,289	1,2,15	-0,289	1,1,4	-0,556	1,4,9	0,733
2,5,2	0,944	2,1,15	-0,289	2,1,15	-0,289	1,1,14	-0,556	1,3,15	0,733
2,4,14	0,944	2,2,15	-0,289	2,2,15	-0,289	1,1,7	-0,556	1,4,15	0,733
2,5,15	0,889	2,4,3	-0,289	2,4,3	-0,289	1,1,11	-0,556	2,4,13	0,733
1,5,9	0,889	1,6,3	-0,289	1,6,3	-0,289	1,1,1	-0,556	2,7,6	0,733
1,8,10	0,833	2,4,15	-0,289	2,4,15	-0,289	1,1,13	-0,600	1,3,2	0,733
2,5,17	0,833	1,3,3	-0,289	1,3,3	-0,289	2,1,13	-0,644	2,2,13	0,689
1,5,10	0,833	1,6,15	-0,333	1,6,15	-0,333	2,1,12	-0,733	1,9,6	0,689

Źródło: opracowanie własne.

Tablica A.24

Struktura pliku PART.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_M	Numeric	3	0	liczba cech
2	F_N	Numeric	3	0	liczba obiektów
3	F_V	Numeric	3	0	liczba okresów
4	F_A	Numeric	19	6	granica przedziału
5	F_B	Numeric	19	6	granica przedziału
6	F_EX	Numeric	19	6	wartość oczekiwana
7	F_STDEV	Numeric	19	6	wariancja

Źródło: opracowanie własne.

Tablica A.25

Struktura pliku RNDAT.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_ID	Character	10	0	identyfikator zbioru
2	F_M	Numeric	3	0	liczba cech
3	F_N	Numeric	3	0	liczba obiektów
4	F_V	Numeric	3	0	liczba okresów
5	F_A	Numeric	19	6	granica przedziału
6	F_B	Numeric	19	6	granica przedziału
7	F_EX	Numeric	19	6	wartość oczekiwana
8	F_STDEV	Numeric	19	6	wariancja

Źródło: opracowanie własne.

Tablica A.26

Struktura pliku PS*.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_EX	Numeric	19	6	wektor średnich
2	F_MED	Numeric	19	6	wektor median
3	F_STDEV	Numeric	19	6	wektor odchyleń standardowych
4	F_VAR	Numeric	19	6	wektor współczynników zmienności
5	F_ASYM	Numeric	19	6	wektor współczynników asymetrii
6	F_CONC	Numeric	19	6	wektor współczynników koncentracji
7	F_DETCOR	Numeric	19	6	wyznacznik macierzy korelacji

Źródło: opracowanie własne.

Tablica A.27

Struktura pliku TD*.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	X1	Numeric	19	6	cecha nr 1
2	X2	Numeric	19	6	cecha nr 2
3	X3	Numeric	19	6	cecha nr 3
.	.	.	19	6	.
.	.	.	19	6	.
.	.	.	19	6	.
N	XN	Numeric	19	6	cecha nr N

Źródło: opracowanie własne.

Tablica A.28

Struktura pliku VC*.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	X1	Numeric	19	6	cecha nr 1
2	X2	Numeric	19	6	cecha nr 2
3	X3	Numeric	19	6	cecha nr 3
.	.	.	19	6	.
.	.	.	19	6	.
.	.	.	19	6	.
N	XN	Numeric	19	6	cecha nr N

Źródło: opracowanie własne.

Tablica A.29

Struktura pliku GAUGES.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_ID	Character	10	0	identyfikator
2	F_PATH	Numeric	5	0	ścieżka
3	F_G01	Numeric	19	6	miernik nr 1
4	F_G02	Numeric	19	6	miernik nr 2
5	F_G03	Numeric	19	6	miernik nr 3
6	F_G04	Numeric	19	6	miernik nr 4
7	F_G05	Numeric	19	6	miernik nr 5
8	F_G06	Numeric	19	6	miernik nr 6
9	F_G07	Numeric	19	6	miernik nr 7
10	F_G08	Numeric	19	6	miernik nr 8
11	F_G09	Numeric	19	6	miernik nr 9
12	F_G10	Numeric	19	6	miernik nr 10
13	F_G11	Numeric	19	6	miernik nr 11
14	F_G12	Numeric	19	6	miernik nr 12
15	F_GAGGR	Numeric	19	6	miernik agregatowy

Źródło: opracowanie własne.

Tablica A.30

Struktura pliku PATHS.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	PATH	Numeric	5	0	ścieżka
2	WEIGHT	Numeric	2	0	waga
3	NORMAL	Numeric	2	0	normalizacja
4	AGGREG	Numeric	2	0	agregacja

Źródło: opracowanie własne.

Tablica A.31

Struktura pliku RANGP.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_PATH	Numeric	5	0	ścieżka
2	F_RANG	Numeric	19	0	ranga

Źródło: opracowanie własne.

Tablica A.32

Struktura pliku RANGW.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_PATH	Numeric	5	0	ścieżka
2	F_RANG	Numeric	19	0	ranga

Źródło: opracowanie własne.

Tablica A.33

Struktura pliku RANGN.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_PATH	Numeric	5	0	ścieżka
2	F_RANG	Numeric	19	0	ranga

Źródło: opracowanie własne.

Tablica A.34

Struktura pliku RANGA.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_PATH	Numeric	5	0	ścieżka
2	F_RANG	Numeric	19	0	ranga

Źródło: opracowanie własne.

Tablica A.35

Struktura pliku PX.DBF

Lp.	Nazwa pola	Typ	Rozmiar	Precyzja	Przeznaczenie
1	F_PATH	Numeric	5	0	ścieżka
2	F_RANG	Numeric	19	0	ranga

Źródło: opracowanie własne.

Kody źródłowe programów

```
/*-------------------------------------------------------------------------
  PLIK: prov.hpp
     ?: plik naglowkowy klasy bazowej prov
-------------------------------------------------------------------------*/

#ifndef PROV_HPP
#define PROV_HPP

Code4 cb;
const long ITMAX = 10000000;

/*=========================================================================
FUNKCJA: out_mem - brak pamieci, funkcja globalna dla set_new_handler
PARAMETRY: void
WYWOLANIE: out_mem();
REZULTAT: void
-------------------------------------------------------------------------*/
void out_mem()
{
   cerr << "Insufficient memory or allocation error!" << endl;
   exit(1);
}

class prov
{
   public:

   // konstruktor
   prov();

   // destruktor
   ~prov() { }

   // funkcje
   void wait();
   int loc(int, int);
   int loc2(int, int, int);
   int length(char *);
   char *substr(char *, char *, int, int);
   char *id_str(Data4, char *);
   double round(double, int);

   // funkcje wirtualne

};

#endif PROV_HPP

/*-------------------------------------------------------------------------
  PLIK: nrand.hpp
     ?: plik naglowkowy klasy nrand
-------------------------------------------------------------------------*/

#ifndef NRAND_HPP
#define NRAND_HPP

class nrand : public prov
{
   public:
   long r;

   float u[97], c, cd, cm;
   int i97, j97, test;

   // konstruktor
   nrand( long s = 0 );
```

```
   // destruktor
   ~nrand() { };

   // funkcje
   void rget();
   void rseed( long );
   long ilot();
   double dlots();
   float flots();
   int ablot( int, int );
   double dablot( int, int );
   double nrnd();
   double trnd();
   double dxrnd( double, double );
   float fxrnd( float, float );
   void mvnd( array &, array &, array &, int, int );
   void ex_rnd( array &, int, int, int );
   void vc_rnd( array &, int, int, int );
   void rm_start();
   int rm_ini(int, int);
   void rm_seed(int *, int *);
   int rm_rnd(float *, int);
   float rm_random();

   // funkcje wirtualne

};

#endif NRAND_HPP

/*----------------------------------------------------------------------------
  PLIK: array.hpp
     ?: plik naglowkowy klasy array
----------------------------------------------------------------------------*/

#ifndef ARRAY_HPP
#define ARRAY_HPP

class array : public prov
{
   public:
   double *b;         // poczatek obszaru pamieci

   // konstruktor
   array(int d1, int d2 = 1, int d3 = 1);

   // destruktor
   ~array();

   // funkcje
   void alloc(int, int, int);          // funkcja przydzielajaca pamiec
   int loc3(int, int, int, int, int);
   void cub2v(array &, array &, int, int, int, int);
   void cub2o(array &, array &, int, int, int, int);
   void cub2t(array &, array &, int, int, int);

   // funkcje wirtualne

};

#endif ARRAY_HPP

/*----------------------------------------------------------------------------
  PLIK: mva.hpp
     ?: plik naglowkowy klasy mva
----------------------------------------------------------------------------*/

#ifndef MVA_HPP
#define MVA_HPP

class mva : public prov
{
   private:

   const int m;                  // liczba cech
   const int n;                  // liczba obiektow
   const int v;                  // liczba okresow
   const int w;                  // rozmiar pola
```

```
const int d;                    // liczba miejsc dziesietnych
const char t;                   // typ pola

public:

// konstruktory
mva( void ) : m(1), n(1), v(1), w(19), d(6), t('N') { }
mva( const int _m, const int _n, const int _v, const int _w = 19,
     const int _d = 6, const char _t = 'N' );

// destruktor
virtual ~mva() { }

// funkcje publiczne
void xcreate( Data4 &, char *, int, int );
void screate( Data4 &, char *, FIELD4INFO *, int );
void tcreate( Data4 &, char *, FIELD4INFO *, TAG4INFO *, int );
void ccreate( Data4 &, char *, int, int, int );
void use_dbf( Data4 &, char * );
int is_dbf(Data4 &, char *);
void use_idx( Data4 &, Index4 &, char *, TAG4INFO * );
char *gets( Data4 &, long int, int );
char *get_s( Data4 &, int );
void repl_s( Data4 &, int, char * );
void putd( Data4 &, long int, int, double );
void put_d( Data4 &, int, double );
void put_i( Data4 &, int, long int );
long int geti( Data4 &, long int, int );
long int get_i( Data4 &, int );
double getd( Data4 &, long int, int );
double get_d( Data4 &, int );
void disp( Data4 & );
void dbf2mat( Data4 &, array &, int, int );
void dbf2arr( Data4 &, array &, int, int, int );
void dbf2vec( Data4 &, array &, int );
void rec2vec(Data4 &, long int, array &, int);
void mat2dbf( Data4 &, array &, int, int );
void mv2dbf( Data4 &, array &, array &, int, int );
void vec2dbf( Data4 &, array &, int, int );
void v2dbf( Data4 &, array &, int, int );
void mtx_dup( array &, array &, int, int );
void list_m( ofstream &, array &, int, int, int, int );
void list_vh( ofstream &, array &, int, int, int );
void list_vv( ofstream &, array &, int, int, int );
void list_lwr( ofstream &, array &, int, int, int );
void list_upr( ofstream &, array &, int, int, int );
void list_t( ofstream &, array &, int, int, int );
void tr2vec( array &, array &, int );
void t2vec( array &, int );
int keep(int *, int, int);
int keep(array &, int, double);
void del_r(array &, array &, int *, int, int, int);
void del_c(array &, array &, int *, int, int, int);
void add_r(array &, array &, array &, int, int, int);
void add_c(array &, array &, array &, int, int, int);
void del_i(array &, array &, array &, int, int);
int index(array &, int, double);
void rang(array &, array &, array &, int);
void qsort_up( array &, int, int );
void qsort_dn( array &, int, int );
void qsort_dn( int *, int, int );
void qsort_dn( array &, int *, int, int );
void qsort_dn( array &, long int *, int, int );
void qsort_up( array &, long int *, int, int );
int odd( int );
void rep_v( array &, double, int );
void diag( array &, double, double, int, int );
void vdiag( array &, array &, double, int, int );
void transp( array &, int, int );
void mult( array &, array &, array &, int, int, int );
double norm_cor(array &, int);
double norm_frob(array &, int, int);
double max_v( array &, int );
double max_m( array &, int, int );
void max_x( array &, array &, int, int );
double min_v( array &, int );
double min_m( array &, int, int );
void min_x( array &, array &, int, int );
double mean_m( array &, int, int );
```

```
   double mean_v( array &, int );
   void mean_x( array &, array &, int, int );
   double stdev_m( array &, int, int );
   double stdev_v( array &, int );
   void stdev_x( array &, array &, int, int );
   double sum_v( array &, int );
   double sum_m( array &, int, int );
   void sum_x( array &, array &, int, int );
   void med_x( array &, array &, int, int );
   void asym_coef( array &, array &, int, int );
   void conc_coef( array &, array &, int, int );
   void var_coef( array &, array &, double *, int, int );
   void corr( array &, array &, int, int );
   void corr_xq( array &, array &, array &, int, int );
   void spearman( array &, array &, array &, int, int );
   double spearman(array &, array &, int);
   double kendall(array &, array &, int);
   void varcov( array &, array &, int, int );
   int gs_qr(array &, array &, array &, int, int);
   void rayleigh( array &, int n, array &, const int, const double, array &, int );
   void hotelling( array &, int, int, array &, array & );
   void loadings( array &, array &, array &, int );
   void loadings( array &, array &, array &, int, array & );
   void characters( array &, int );
   void dest2stym( array &, array &, int, int );
   void neg2pos( array &, int, int );
   void upperpole( array &, array &, int, int );
   void lowerpole( array &, array &, int, int );
   double det_eigen( array &, int );
   double gauge01( array &, array &, int, int );
   double gauge02( array &, array &, int, int );
   double gauge03( array &, array &, int, int );
   double gauge04( array &, array &, int, int );
   double gauge05( array &, array &, int, int );
   double gauge06( array &, array &, int, int );
   double gauge07( array &, array &, int, int );
   double gauge08( array &, array &, int, int );
   double gauge09( array &, int );
   double gauge10( array &, int );
   double gauge11( array &, array &, int, int );
   double gauge12( array &, array &, int, int );
   void dist_m01( array &, array &, int, int );
   void dist_v01( array &, array &, int );
   void normv01( array &, array &, int );
   void normv02( array &, array &N, int );
   void scale_q01( array &, int );
   void scale_q02( array &, int );
   void scale_q03( array &, int );
   void scale_q04( array &, int );
   int mlu_decomp( array &, array &, int );
   void lu_decomp( array &, int );
   double mdet( array &, int );
   double det( array &, int );
   void minverse( array &, int );
   void inverse( array &, int );

   // funkcje wirtualne
   virtual void weight( array &, int ) { };
   virtual void weight( array &, array &, int, int ) { };
   virtual void normal( array &, array &, int, int ) { };
   virtual void aggreg( array &, array &, array &, int, int ) { };
   virtual void aggreg( array &, array &, array &, array &, int, int ) { };

};

#endif MVA_HPP

/*-------------------------------------------------------------------------
  PLIK: weig01.hpp
     ?: plik naglowkowy klasy weig01
-------------------------------------------------------------------------*/

#ifndef WEIG01_HPP
#define WEIG01_HPP

class weig01 : public mva
{
   public:
```

```
   // konstruktory
   weig01( void ) { }   // bezparametrowy
   weig01(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~weig01();

   // funkcje

   // funkcje wirtualne
   void weight( array &, int );

};

#endif WEIG01_HPP

/*----------------------------------------------------------------------------
  PLIK: weig02.hpp
     ?: plik naglowkowy klasy weig02
----------------------------------------------------------------------------*/

#ifndef WEIG02_HPP
#define WEIG02_HPP

class weig02 : public mva
{
   public:

   // konstruktory
   weig02( void ) { }   // bezparametrowy
   weig02(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~weig02();

   // funkcje

   // funkcje wirtualne
   void weight( array &, array &, int, int );

};

#endif WEIG02_HPP

/*----------------------------------------------------------------------------
  PLIK: norm01.hpp
     ?: plik naglowkowy klasy norm01
----------------------------------------------------------------------------*/

#ifndef NORM01_HPP
#define NORM01_HPP

class norm01 : public mva
{
   public:

   // konstruktory
   norm01( void ) { }   // bezparametrowy
   norm01(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm01();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM01_HPP

/*----------------------------------------------------------------------------
  PLIK: norm02.hpp
     ?: plik naglowkowy klasy norm02
----------------------------------------------------------------------------*/
```

```
#ifndef NORM02_HPP
#define NORM02_HPP

class norm02 : public mva
{
   public:

   // konstruktory
   norm02( void ) { }   // bezparametrowy
   norm02(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm02();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM02_HPP

/*----------------------------------------------------------------------------
  PLIK: norm03.hpp
     ?: plik naglowkowy klasy norm03
----------------------------------------------------------------------------*/

#ifndef NORM03_HPP
#define NORM03_HPP

class norm03 : public mva
{
   public:

   // konstruktory
   norm03( void ) { }   // bezparametrowy
   norm03(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm03();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM03_HPP

/*----------------------------------------------------------------------------
  PLIK: norm04.hpp
     ?: plik naglowkowy klasy norm04
----------------------------------------------------------------------------*/

#ifndef NORM04_HPP
#define NORM04_HPP

class norm04 : public mva
{
   public:

   // konstruktory
   norm04( void ) { }   // bezparametrowy
   norm04(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm04();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );
```

```
};

#endif NORM04_HPP

/*-------------------------------------------------------------------------
  PLIK: norm05.hpp
     ?: plik naglowkowy klasy norm05
-------------------------------------------------------------------------*/

#ifndef NORM05_HPP
#define NORM05_HPP

class norm05 : public mva
{
   public:

   // konstruktory
   norm05( void ) { }   // bezparametrowy
   norm05(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm05();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM05_HPP

/*-------------------------------------------------------------------------
  PLIK: norm06.hpp
     ?: plik naglowkowy klasy norm06
-------------------------------------------------------------------------*/

#ifndef NORM06_HPP
#define NORM06_HPP

class norm06 : public mva
{
   public:

   // konstruktory
   norm06( void ) { }   // bezparametrowy
   norm06(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 7, const char _t = 'N');

   // destruktor
   ~norm06();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM06_HPP

/*-------------------------------------------------------------------------
  PLIK: norm07.hpp
     ?: plik naglowkowy klasy norm07
-------------------------------------------------------------------------*/

#ifndef NORM07_HPP
#define NORM07_HPP

class norm07 : public mva
{
   public:

   // konstruktory
   norm07( void ) { }   // bezparametrowy
   norm07(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');
```

```
   // destruktor
   ~norm07();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM07_HPP

/*----------------------------------------------------------------------------
  PLIK: norm08.hpp
     ?: plik naglowkowy klasy norm08 - deklaracje pol i funkcji
----------------------------------------------------------------------------*/

#ifndef NORM08_HPP
#define NORM08_HPP

class norm08 : public mva
{
   public:

   // konstruktory
   norm08( void ) { }   // bezparametrowy
   norm08(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm08();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM08_HPP

/*----------------------------------------------------------------------------
  PLIK: norm09.hpp
     ?: plik naglowkowy klasy norm09
----------------------------------------------------------------------------*/

#ifndef NORM09_HPP
#define NORM09_HPP

class norm09 : public mva
{
   public:

   // konstruktory
   norm09( void ) { }   // bezparametrowy
   norm09(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~norm09();

   // funkcje

   // funkcje wirtualne
   void normal( array &, array &, int, int );

};

#endif NORM09_HPP

/*----------------------------------------------------------------------------
  PLIK: aggr01.hpp
     ?: plik naglowkowy klasy aggr01
----------------------------------------------------------------------------*/

#ifndef AGGR01_HPP
#define AGGR01_HPP

class aggr01 : public mva
```

```
{
   public:

   // konstruktory
   aggr01( void ) { }   // bezparametrowy
   aggr01(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr01();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, int, int ) ;

};

#endif AGGR01_HPP

/*----------------------------------------------------------------------------
  PLIK: aggr02.hpp
     ?: plik naglowkowy klasy aggr02
----------------------------------------------------------------------------*/

#ifndef AGGR02_HPP
#define AGGR02_HPP

class aggr02 : public mva
{
   public:

   // konstruktory
   aggr02( void ) { }   // bezparametrowy
   aggr02(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr02();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, int, int ) ;

};

#endif AGGR02_HPP

/*----------------------------------------------------------------------------
  PLIK: aggr03.hpp
     ?: plik naglowkowy klasy aggr03
----------------------------------------------------------------------------*/

#ifndef AGGR03_HPP
#define AGGR03_HPP

class aggr03 : public mva
{
   public:

   // konstruktory
   aggr03( void ) { }   // bezparametrowy
   aggr03(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr03();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, int, int );

};

#endif AGGR03_HPP

/*----------------------------------------------------------------------------
```

```
  PLIK: aggr04.hpp
     ?: plik naglowkowy klasy aggr04
-------------------------------------------------------------------------------*/

#ifndef AGGR04_HPP
#define AGGR04_HPP

class aggr04 : public mva
{
   public:

   // konstruktory
   aggr04( void ) { }   // bezparametrowy
   aggr04(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr04();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR04_HPP

/*-------------------------------------------------------------------------------
  PLIK: aggr05.hpp
     ?: plik naglowkowy klasy aggr05
-------------------------------------------------------------------------------*/

#ifndef AGGR05_HPP
#define AGGR05_HPP

class aggr05 : public mva
{
   public:

   // konstruktory
   aggr05( void ) { }   // bezparametrowy
   aggr05(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr05();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR05_HPP

/*-------------------------------------------------------------------------------
  PLIK: aggr06.hpp
     ?: plik naglowkowy klasy aggr06
-------------------------------------------------------------------------------*/

#ifndef AGGR06_HPP
#define AGGR06_HPP

class aggr06 : public mva
{
   public:

   // konstruktory
   aggr06( void ) { }   // bezparametrowy
   aggr06(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr06();

   // funkcje
```

```
   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR06_HPP

/*-------------------------------------------------------------------------
  PLIK: aggr07.hpp
     ?: plik naglowkowy klasy aggr07
-------------------------------------------------------------------------*/

#ifndef AGGR07_HPP
#define AGGR07_HPP

class aggr07 : public mva
{
   public:

   // konstruktory
   aggr07( void ) { }   // bezparametrowy
   aggr07(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr07();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR07_HPP

/*-------------------------------------------------------------------------
  PLIK: aggr08.hpp
     ?: plik naglowkowy klasy aggr08
-------------------------------------------------------------------------*/

#ifndef AGGR08_HPP
#define AGGR08_HPP

class aggr08 : public mva
{
   public:

   // konstruktory
   aggr08( void ) { }   // bezparametrowy
   aggr08(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr08();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR08_HPP

/*-------------------------------------------------------------------------
  PLIK: aggr09.hpp
     ?: plik naglowkowy klasy aggr09
-------------------------------------------------------------------------*/

#ifndef AGGR09_HPP
#define AGGR09_HPP

class aggr09 : public mva
{
   public:

   // konstruktory
   aggr09( void ) { }   // bezparametrowy
```

```
   aggr09(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr09();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR09_HPP

/*----------------------------------------------------------------------------
  PLIK: aggr10.hpp
     ?: plik naglowkowy klasy aggr10 - deklaracje pol i funkcji
----------------------------------------------------------------------------*/

#ifndef AGGR10_HPP
#define AGGR10_HPP

class aggr10 : public mva
{
   public:

   // konstruktory
   aggr10( void ) { }   // bezparametrowy
   aggr10(const int _m, const int _n, const int _v, const int _w = 19,
            const int _d = 6, const char _t = 'N');

   // destruktor
   ~aggr10();

   // funkcje

   // funkcje wirtualne
   void aggreg( array &, array &, array &, array &, int, int );

};

#endif AGGR10_HPP


/*----------------------------------------------------------------------------
  PLIK: struct.hpp
     ?: plik naglowkowy - deklaracje struktur i indeksow
----------------------------------------------------------------------------*/

#ifndef STRUCT_HPP
#define STRUCT_HPP


   // struktura pliku GAUGES.DBF
   FIELD4INFO ga_struct[] =
   {
      { "F_ID",     'C', 10, 0 },
      { "F_PATH",   'N',  5, 0 },
      { "F_G01",    'N', 19, 6 },
      { "F_G02",    'N', 19, 6 },
      { "F_G03",    'N', 19, 6 },
      { "F_G04",    'N', 19, 6 },
      { "F_G05",    'N', 19, 6 },
      { "F_G06",    'N', 19, 6 },
      { "F_G07",    'N', 19, 6 },
      { "F_G08",    'N', 19, 6 },
      { "F_G09",    'N', 19, 6 },
      { "F_G10",    'N', 19, 6 },
      { "F_G11",    'N', 19, 6 },
      { "F_G12",    'N', 19, 6 },
      { "F_GAGGR",  'N', 19, 6 },
      { 0, 0, 0, 0 },
   };

   // indeks GAUGES.CDX
   TAG4INFO ga_taginfo[] =
   {
      { "ID", "F_ID",    0, 0, 0 },
```

```
   { "GP", "F_PATH", 0, 0, 0 },
   { 0, 0, 0, 0, 0 },
};

// struktura pliku PS*.DBF
FIELD4INFO ps_struct[] =
{
   { "F_EX",      'N', 19, 6 },
   { "F_MED",     'N', 19, 6 },
   { "F_STDEV",   'N', 19, 6 },
   { "F_VAR",     'N', 19, 6 },
   { "F_ASYM",    'N', 19, 6 },
   { "F_CONC",    'N', 19, 6 },
   { "F_DETCOR",  'N', 19, 6 },
   { 0, 0, 0, 0 },
};

// struktura pliku RNDAT.DBF
FIELD4INFO rd_struct[] =
{
   { "F_ID",     'C', 10, 0 },
   { "F_M",      'N',  3, 0 },
   { "F_N",      'N',  3, 0 },
   { "F_V",      'N',  3, 0 },
   { "F_A",      'N', 19, 6 },
   { "F_B",      'N', 19, 6 },
   { "F_EX",     'N', 19, 6 },
   { "F_STDEV",  'N', 19, 6 },
   { 0, 0, 0, 0 },
};

// indeks RNDAT.CDX
TAG4INFO rd_taginfo[] =
{
   { "ID", "F_ID", 0, 0, 0 },
   { 0, 0, 0, 0, 0 },
};

// struktura pliku RANG.DBF
FIELD4INFO rp_struct[] =
{
   { "F_PATH",   'N',  5, 0 },
   { "F_RANG",   'N', 19, 0 },
   { 0, 0, 0, 0 },
};

// indeks RANG.CDX
TAG4INFO rp_taginfo[] =
{
   { "RNGP", "F_RANG", 0, 0, r4descending },
   { 0, 0, 0, 0, 0 },
};

// struktura pliku RANGW.DBF
FIELD4INFO rw_struct[] =
{
   { "F_PATH",   'N',  5, 0 },
   { "F_RANG",   'N', 19, 0 },
   { 0, 0, 0, 0 },
};

// indeks RANGW.CDX
TAG4INFO rw_taginfo[] =
{
   { "RNGW", "F_RANG", 0, 0, r4descending },
   { 0, 0, 0, 0, 0 },
};

// struktura pliku RANGN.DBF
FIELD4INFO rn_struct[] =
{
   { "F_PATH",   'N',  5, 0 },
   { "F_RANG",   'N', 19, 0 },
   { 0, 0, 0, 0 },
};

// indeks RANGN.CDX
TAG4INFO rn_taginfo[] =
{
```

```
      { "RNGN", "F_RANG", 0, 0, r4descending },
      { 0, 0, 0, 0, 0 },
   };

   // struktura pliku RANGA.DBF
   FIELD4INFO ra_struct[] =
   {
      { "F_PATH",  'N',  5, 0 },
      { "F_RANG",  'N', 19, 0 },
      { 0, 0, 0, 0 },
   };

   // indeks RANGA.CDX
   TAG4INFO ra_taginfo[] =
   {
      { "RNGA", "F_RANG", 0, 0, r4descending },
      { 0, 0, 0, 0, 0 },
   };

   // struktura pliku OPTPTH.DBF
   FIELD4INFO op_struct[] =
   {
      { "F_PATH",  'N',  5, 0 },
      { "F_GAGGR", 'N', 19, 6 },
      { 0, 0, 0, 0 },
   };

   // indeks OPTPTH.CDX
   TAG4INFO op_taginfo[] =
   {
      { "AGR", "F_GAGGR", 0, 0, 0 },
      { "OPA", "F_PATH", 0, 0, 0 },
      { 0, 0, 0, 0, 0 },
   };

   // struktura pliku OPTPTH.DBF
   FIELD4INFO tp_struct[] =
   {
      { "F_PATH",  'N',  5, 0 },
      { "F_GAGGR", 'N', 19, 6 },
      { 0, 0, 0, 0 },
   };

   // struktura pliku GAX.DBF
   FIELD4INFO gx_struct[] =
   {
      { "F_DATA",  'N', 5, 0 },
      { "F_P01",   'N', 5, 0 },
      { "F_P02",   'N', 5, 0 },
      { "F_P03",   'N', 5, 0 },
      { "F_P04",   'N', 5, 0 },
      { "F_P05",   'N', 5, 0 },
      { "F_P06",   'N', 5, 0 },
      { "F_P07",   'N', 5, 0 },
      { "F_P08",   'N', 5, 0 },
      { "F_P09",   'N', 5, 0 },
      { "F_P10",   'N', 5, 0 },
      { 0, 0, 0, 0 },
   };

   // struktura pliku PX*.DBF
   FIELD4INFO px_struct[] =
   {
      { "F_PATH",  'N',  5, 0 },
      { "F_GAGGR", 'N', 19, 6 },
      { 0, 0, 0, 0 },
   };

   // struktura pliku KEN.DBF
   FIELD4INFO ken_struct[] =
   {
      { "F_PATH",  'N',  5, 0 },
      { "F_KENDALL", 'N', 19, 6 },
      { 0, 0, 0, 0 },
   };

#endif STRUCT_HPP

/*---------------------------------------------------------------------------
```

```
  PLIK: mvac.hpp
     ?: pliki programowe
-------------------------------------------------------------------------------*/

#ifndef MVAC_CPP
#define MVAC_CPP

#include "prov.cpp"
#include "array.cpp"
#include "nrand.cpp"
#include "mva.cpp"
#include "weig01.cpp"
#include "weig02.cpp"
#include "norm01.cpp"
#include "norm02.cpp"
#include "norm03.cpp"
#include "norm04.cpp"
#include "norm05.cpp"
#include "norm06.cpp"
#include "norm07.cpp"
#include "norm08.cpp"
#include "norm09.cpp"
#include "aggr01.cpp"
#include "aggr02.cpp"
#include "aggr03.cpp"
#include "aggr04.cpp"
#include "aggr05.cpp"
#include "aggr06.cpp"
#include "aggr07.cpp"
#include "aggr08.cpp"
#include "aggr09.cpp"
#include "aggr10.cpp"

#endif MVAC_CPP

/*-------------------------------------------------------------------------------
  PLIK: mvah.hpp
     ?: pliki naglowkowe
-------------------------------------------------------------------------------*/

#ifndef MVAH_HPP
#define MVAH_HPP

#include <conio.h>
#include <fstream.h>
#include <iomanip.h>
#include <iostream.h>
#include <math.h>
#include <new.h>
#include <time.h>
#include "d4all.hpp"
#include "prov.hpp"
#include "array.hpp"
#include "nrand.hpp"
#include "struct.hpp"
#include "mva.hpp"
#include "weig01.hpp"
#include "weig02.hpp"
#include "norm01.hpp"
#include "norm02.hpp"
#include "norm03.hpp"
#include "norm04.hpp"
#include "norm05.hpp"
#include "norm06.hpp"
#include "norm07.hpp"
#include "norm08.hpp"
#include "norm09.hpp"
#include "aggr01.hpp"
#include "aggr02.hpp"
#include "aggr03.hpp"
#include "aggr04.hpp"
#include "aggr05.hpp"
#include "aggr06.hpp"
#include "aggr07.hpp"
#include "aggr08.hpp"
#include "aggr09.hpp"
#include "aggr10.hpp"

#endif MVAH_HPP
```

```
/*-----------------------------------------------------------------------------
  PLIK: mv.cpp
     ?: program obliczajacy zmienne syntetyczne dla podanej sciezki
-----------------------------------------------------------------------------*/

#include "mvah.hpp"
#include "mvac.hpp"

extern unsigned _stklen = 10000;


int main(int argc, char *argv[])
{
   if (argc == 1)
   {
      cout << "call: mv name m n v p" << endl;
      cout << "name - database file name" << endl;
      cout << "m - number of variables" << endl;
      cout << "n - number of objects" << endl;
      cout << "v - number of terms" << endl;
      cout << "p - path's number" << endl;
      exit(0);
   }
   else
   {
      for (int c = 0; c < argc; c++) cout << "par:" << c << ":" << *(argv+c) << endl;
   }

   int m, n, v, p, err = 0;
   m = atoi(argv[2]);
   n = atoi(argv[3]);
   v = atoi(argv[4]);
   p = atoi(argv[5]);

   if ((m < 0) || (m > 25)) err = 1;
   if ((n < 0) || (n > 50)) err = 1;
   if ((v < 0) || (v > 15)) err = 1;
   if ((p < 0) || (p > 306)) err = 1;
   if (err)
   {
      cout << "invalid command ..." << endl;
      exit(0);
   }

   // zmienne i obiekty
   char *s_id = "ID00000000";
   int i, j, k, rec, weig, norm, aggr, nr_path, p_beg, p_end, g = 12;
   double det = 0.0, g_aggr = 0.0;
   Data4 rd_dbf, td_dbf, ga_dbf, pa_dbf, pr_dbf, dat_dbf;
   Index4 idx;
   mva wap;
   prov str;
   array vGAUG( g );              // mierniki jakosci cechy syntetycznej
   ofstream f("wap_arg.txt");

   // zmienne wskaznikowe
   mva *ptr;
   weig01 *pw01; weig02 *pw02;
   norm01 *pn01; norm02 *pn02; norm03 *pn03; norm04 *pn04; norm05 *pn05;
   norm06 *pn06; norm07 *pn07; norm08 *pn08; norm09 *pn09;
   aggr01 *pa01; aggr02 *pa02; aggr03 *pa03; aggr04 *pa04; aggr05 *pa05;
   aggr06 *pa06; aggr07 *pa07; aggr08 *pa08; aggr09 *pa09; aggr10 *pa10;

   // utworzenie plikow GAUGES.DBF i GAUGES.CDX - mierniki jakosci cech syntetycznych
   wap.tcreate( ga_dbf, "gauges.dbf", ga_struct, ga_taginfo, 0 );
   wap.use_dbf( ga_dbf, "gauges.dbf" );
   wap.use_idx( ga_dbf, idx, "gauges.cdx", ga_taginfo );

   // utworzenie unikalnego identyfikatora sesji
   str.id_str( ga_dbf, s_id );

   // otwarcie pliku PATHS.DBF - sciezki procedury WAP
   wap.use_dbf( pa_dbf, "paths.dbf" );

   // tablice
   array mDAT( m, n );            // dane pierwotne
   array mNRM( m, n );            // dane znormalizowane
   array vSD( m );                // wektor srednich
```

```
   array mVC( m, m );            // macierz wariancji-kowariancji / korelacji
   array vEVAL( m );             // wektor wartosci wlasnych
   array mEVEC( m, m );          // macierz wektorow wlasnych
   array vTMP( m );              // identyfikatory / bieguny
   array vWGT( m );              // wektor wag
   array vAGR( n );              // zmienna syntetyczna
//    array mXX( m, 3 );           // ladunki

   wap.use_dbf( dat_dbf, argv[1] );
   wap.dbf2mat( dat_dbf, mDAT, m, n );
   dat_dbf.close();

   /* -------------------------- procedura WAP -------------------------- */

   wap.corr( mDAT, mVC, m, n );
   f << "Macierz korelacji:" << endl;
   wap.list_m( f, mVC, m, m );

   wap.t2vec( mVC, m );

   // wyznaczenie charakteru cech metoda analizy czynnikowej
   wap.hotelling( mVC, m, m, mEVEC, vEVAL );
   wap.transp( mEVEC, m, m );
   f << "Wektor wartosci wlasnych:" << endl;
   wap.list_vh( f, vEVAL, m );

   wap.loadings( mEVEC, vEVAL, vSD, m );
// wap.loadings( mEVEC, vEVAL, vSD, m, mXX );
   f << "Ladunki czynnikowe:" << endl;
   wap.list_vh( f, vSD, m );

   // charakter cech
   wap.characters( vSD, m );
   f << "Wektor identyfikatorow cech:" << endl;
   wap.list_vh( f, vSD, m );

   // wyznacznik macierzy korelacji
   det = wap.det_eigen( vEVAL, m );
   f << "Wyznacznik z wartosci wlasnych: " << setw(25) << setprecision(15) << det << endl;

   if (p > 0)
      {
         pa_dbf.go(p);
         p_beg = p;
         p_end = p + 1;
      }
   else
      {
         pa_dbf.top();
         p_beg = 1;
         p_end = 307;
      }

   // petla przetwarzajaca kolejne sciezki procedury WAP
   for ( p_beg; pa_dbf.recNo() < p_end; pa_dbf.skip() )
   {
      // wczytanie konfiguracji WAP z pliku PATHS.DBF
      rec = pa_dbf.recNo();
      nr_path = wap.get_i( pa_dbf, 1 );
      weig = wap.get_i( pa_dbf, 2 );
      norm = wap.get_i( pa_dbf, 3 );
      aggr = wap.get_i( pa_dbf, 4 );

      cout << "id: " << s_id << "  p: " << rec << endl;
      f << "P: " << rec << "  W: " << weig << "  N: " << norm << "  A: " << aggr << "  m: " <<
m << "  n: " << n << "  v: " << v << endl;

      // wyznaczenie wag cech na danych pierwotnych
      switch( weig )
      {
         case 1: pw01 = new weig01; ptr = pw01; break;
         case 2: pw02 = new weig02; ptr = pw02; break;
      }
      if ( weig == 1 )
      {
         ptr->weight( vWGT, m );
      }
      else
      {
```

```
   ptr->weight( mDAT, vWGT, m, n );
}
f << "Wagi zmiennych: " << weig << endl;
wap.list_vh( f, vWGT, m );
delete ptr;

// normalizacja cech
switch( norm )
{
   case 1: pn01 = new norm01; ptr = pn01; break;
   case 2: pn02 = new norm02; ptr = pn02; break;
   case 3: pn03 = new norm03; ptr = pn03; break;
   case 4: pn04 = new norm04; ptr = pn04; break;
   case 5: pn05 = new norm05; ptr = pn05; break;
   case 6: pn06 = new norm06; ptr = pn06; break;
   case 7: pn07 = new norm07; ptr = pn07; break;
   case 8: pn08 = new norm08; ptr = pn08; break;
   case 9: pn09 = new norm09; ptr = pn09; break;
}

ptr->normal( mDAT, mNRM, m, n );
f << "Dane zormalizowane: " << norm << endl;
wap.list_m( f, mNRM, m, n, 15, 5 );

delete ptr;

// przeksztalcenie destymulant w stymulanty na danych znormalizowanych
wap.dest2stym( mNRM, vSD, m, n);
f << "Dane po przeksztalceniu D2S:" << endl;
wap.list_m( f, mNRM, m, n );

// eliminacja wartosci ujemnych i zerowych na danych znormalizowanych
wap.neg2pos( mNRM, m, n );
f << "Dane po eliminacji elementow <= 0:" << endl;
wap.list_m( f, mNRM, m, n );

// agragacja cech
switch( aggr )
{
   case  1: pa01 = new aggr01; ptr = pa01; break;
   case  2: pa02 = new aggr02; ptr = pa02; break;
   case  3: pa03 = new aggr03; ptr = pa03; break;
   case  4: pa04 = new aggr04; ptr = pa04; break;
   case  5: pa05 = new aggr05; ptr = pa05; break;
   case  6: pa06 = new aggr06; ptr = pa06; break;
   case  7: pa07 = new aggr07; ptr = pa07; break;
   case  8: pa08 = new aggr08; ptr = pa08; break;
   case  9: pa09 = new aggr09; ptr = pa09; break;
   case 10: pa10 = new aggr10; ptr = pa10; break;
   case 11: pa04 = new aggr04; ptr = pa04; break;
   case 12: pa05 = new aggr05; ptr = pa05; break;
   case 13: pa06 = new aggr06; ptr = pa06; break;
   case 14: pa07 = new aggr07; ptr = pa07; break;
   case 15: pa08 = new aggr08; ptr = pa08; break;
   case 16: pa09 = new aggr09; ptr = pa09; break;
   case 17: pa10 = new aggr10; ptr = pa10; break;
}
if ( aggr < 4 )
{
   ptr->aggreg( mNRM, vWGT, vAGR, m, n );
}
else if ( aggr > 3 && aggr < 11 )
{
   // wyznaczenie dolnego bieguna zbioru
   wap.lowerpole( mNRM, vTMP, m, n );
   f << "Dolny biegun zbioru:" << endl;
   wap.list_vh( f, vTMP, m );

   ptr->aggreg( mNRM, vWGT, vTMP, vAGR, m, n );
}
else if ( aggr > 10 )
{
   // wyznaczenie gornego bieguna zbioru
   wap.upperpole( mNRM, vTMP, m, n );
   f << "Gorny biegun zbioru:" << endl;
   wap.list_vh( f, vTMP, m );

   ptr->aggreg( mNRM, vWGT, vTMP, vAGR, m, n );
}
```

```
      delete ptr;

      f << "Zmienna syntetyczna: " << aggr << endl;
      wap.list_vh( f, vAGR, n );

      // obliczenie miernikow jakosci cech syntetycznych
      i = 0;
      *(vGAUG.b+i++) = wap.gauge01( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge02( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge03( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge04( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge05( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge06( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge07( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge08( mNRM, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge09( vAGR, n );
      *(vGAUG.b+i++) = wap.gauge10( vAGR, n );
      *(vGAUG.b+i++) = wap.gauge11( mDAT, vAGR, m, n );
      *(vGAUG.b+i++) = wap.gauge12( mDAT, vAGR, m, n );
      g_aggr = 0.0;
      for (i = 0; i < g; i++)
      {
         g_aggr = g_aggr + pow( *(vGAUG.b+i), 2 );
      }
      f << "Mierniki jakosci:" << endl;
      wap.list_vh( f, vGAUG, g );

      wap.scale_q01( vAGR, n );

      f << "Zmienna syntetyczna wyskalowana:" << endl;
      wap.list_vh( f, vAGR, n );
      wap.qsort_dn( vAGR, 0, n-1 );
      f << "Zmienna syntetyczna wyskalowana i posortowana:" << endl;
      wap.list_vh( f, vAGR, n );

      // zapisanie identyfikatora sesji, nr sciezki i miernikow jakosci
         cechy syntetycznej w pliku GAUGES.DBF
      ga_dbf.appendBlank();
      wap.repl_s( ga_dbf, 1, s_id );
      wap.put_i( ga_dbf, 2, nr_path );
      wap.vec2dbf( ga_dbf, vGAUG, 3, g );
      wap.put_d( ga_dbf, 15, sqrt(g_aggr) );
   }

   // destrukcja tablic
   mVC.~array(); vSD.~array(); mDAT.~array();
   mNRM.~array(); vTMP.~array(); vWGT.~array();
   vAGR.~array(); mEVEC.~array(); vEVAL.~array();

   f.close();
   cb.closeAll();
   cb.initUndo();

   return 0;
}


/* ------------------------------------------------------------------------
   PLIK: prov.cpp
      ?: definicje funkcji skladowych klasy bazowej prov
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: prov - konstruktor obiektow klasy prov
PARAMETRY: void
WYWOLANIE: prov pr();
REZULTAT: void
------------------------------------------------------------------------*/
prov::prov()
{

}

/*========================================================================
FUNKCJA: wait - oczekuje na znak z klawiatury
PARAMETRY: void
WYWOLANIE: wait();
REZULTAT: void
```

```
-------------------------------------------------------------------------*/
void prov::wait()
{
   cout << "Press any key to continue ..." << endl;
   getch();
}

/*=========================================================================
FUNKCJA: loc - przeksztalca dwuwymiarowy adres elementu macierzy na
               jednowymiarowy adres elementu wektora
PARAMETRY:
   i - nr wiersza w macierzy
   j - nr kolumny w macierzy
WYWOLANIE: loc(i, j);
REZULTAT: adres elementu w wektorze
-------------------------------------------------------------------------*/
int prov::loc(int i, int j)
{
   int q;
   i = i + 1;  j = j + 1;
   div_t k = div(i * (i - 1), 2);
   q = k.quot + j;
   return (q - 1);
}

/*=========================================================================
FUNKCJA: loc2 - przeksztalca dwuwymiarowy adres indeksowy elementu macierzy
                na jednowymiarowy adres wskaznikowy elementu wektora
PARAMETRY:
   i - nr wiersza w macierzy danych
   j - nr kolumny w macierzy danych
   m - liczba kolumn w macierzy danych
WYWOLANIE: loc2( i, j, m );
REZULTAT: adres wskaznikowy
-------------------------------------------------------------------------*/
int prov::loc2( int i, int j, int m )
{
   return ( i*m+j );
}

/*=========================================================================
FUNKCJA: length - oblicza dlugosc lancucha znakowego
PARAMETRY:
   s - lancuch znakow
WYWOLANIE: length( s );
REZULTAT: dlugosz lancucha
-------------------------------------------------------------------------*/
int prov::length(char *s)
{
 char *p = s;
 while(*p) p++;
 return p-s;
}

/*=========================================================================
FUNKCJA: substr - wycina fragment z lancucha znakow
PARAMETRY:
   str_s - zrodlowy lancuch znakow
   str_t - wyciety fragment
   p - pozycja startowa fragmentu
   len - dlugosc fragmentu
WYWOLANIE: substr( str_s, str_t, p, len );
REZULTAT: wyciety fragment lancucha
-------------------------------------------------------------------------*/
char *prov::substr( char *str_s, char *str_t, int p, int len )
{
   if( length(str_s) < p + len - 1 ) return NULL;
   char *t = str_t;
   char *s = str_s + p - 1;
   while( len-- ) *t++ = *s++;
   *t = '\0';
   return str_t;
}

/*=========================================================================
FUNKCJA: id_str - tworzy unikalny indentyfikator w postaci lancucha znakow
PARAMETRY:
   str_t - utworzony identyfikator
   dbf - identyfikator pliku z kluczem F_ID
```

```
WYWOLANIE: id_str( dbf, str_t );
REZULTAT: identyfikator
-------------------------------------------------------------------------------*/
char *prov::id_str( Data4 dbf, char *str_t )
{
   int rc;
   char *t = str_t, nr[11], s_id[] = "ID00000000";
   long int i = 1;
   while( i < ITMAX )
   {
      rc = dbf.seek( s_id );
      if( rc == 0 )
      {
         itoa( i++, nr, 10);
         substr( s_id, t, 1, (10 - length(nr)) );
         strcat( t, nr );
         strcpy( s_id, t );
      }
      else break;
   }
   return str_t;
}

/*=============================================================================
FUNKCJA: round - zaokragla liczbe rzeczywista typu double
PARAMETRY:
   x - liczba rzeczywista
   p - precyzja zaokraglenia
WYWOLANIE: round( x, p );
REZULTAT: liczb zaokraglona
-------------------------------------------------------------------------------*/
double prov::round( double x, int p )
{
   return floor( x * pow10( p ) ) * pow10( -p );
}

/* ---------------------------------------------------------------------------
   PLIK: nrand.cpp
   ?: definicje funkcji klasy nrand
--------------------------------------------------------------------------- */

/*=============================================================================
FUNKCJA: nrand - konstruktor obiektow klasy nrand
PARAMETRY:
   s - liczba typu long int (domyslnie r = 0)
WYWOLANIE: nrand rnd( s );
REZULTAT: void
-------------------------------------------------------------------------------*/
nrand::nrand( long s )
{
   r = s;
}

/*=============================================================================
FUNKCJA: rget - produkuje liczbe losowa na podstawie r metoda kogruencji
                liniowej
PARAMETRY:void
WYWOLANIE: rget();
REZULTAT: void
-------------------------------------------------------------------------------*/
void nrand::rget()
{
   r = r * 1103515245 + 12345;
}

/*=============================================================================
FUNKCJA: rseed - inicjuje generator liczba s
PARAMETRY:
   s - zarodek generatora
WYWOLANIE: rseed( s );
REZULTAT: void
-------------------------------------------------------------------------------*/
void nrand::rseed( long s )
{
   r = s;
}

/*=============================================================================
FUNKCJA: ilot - generuje liczbe losowa typu long int z przedzialu
```

```
                [0, LONG_MAX]
PARAMETRY: void
WYWOLANIE: ilot();
REZULTAT: wylosowana liczbu typu long int
-------------------------------------------------------------------------*/
long nrand::ilot()
{
   rget();
   return ( r & LONG_MAX );
}

/*=========================================================================
FUNKCJA: dlots - generuje liczbe losowa typu double z przedzialu [0, 1] przy
                 wykorzystaniu generatora rget o zakresie [0, LONG_MAX]
PARAMETRY: void
WYWOLANIE: dlots();
REZULTAT: wylosowana liczbu typu double
-------------------------------------------------------------------------*/
double nrand::dlots()
{
   rget();
   return ( r & LONG_MAX ) / ( double ) LONG_MAX;
}

/*=========================================================================
FUNKCJA: flots - generuje liczby pseudolosowe typu float z przedzialu [0,1]
                 przy wykorzystaniu generatora wbudowanego rand
                 o zakresie [0, INT_MAX]
PARAMETRY: void
WYWOLANIE: flots();
REZULTAT: liczba pseudolosowa typu float
-------------------------------------------------------------------------*/
float nrand::flots()
{
   float r;
   r = rand() & INT_MAX;
   return ( r / (float) INT_MAX );
}

/*=========================================================================
FUNKCJA: ablots - generuje liczbe losowa typu int z przedzialu [a, b]
PARAMETRY:
   a - dolna granica przedzialu
   b - gorna granica przedzialu
WYWOLANIE: ablots( a, b );
REZULTAT: wylosowana liczbu typu int
-------------------------------------------------------------------------*/
int nrand::ablot( int a, int b )
{
   return (int) ( a + ( b - a ) * dlots() );
}

/*=========================================================================
FUNKCJA: dablots - generuje liczbe losowa typu double z przedzialu [a, b]
PARAMETRY:
   a - dolna granica przedzialu
   b - gorna granica przedzialu
WYWOLANIE: dablots( a, b );
REZULTAT: wylosowana liczbu typu int
-------------------------------------------------------------------------*/
double nrand::dablot( int a, int b )
{
   return (double) ( a + ( b - a ) * dlots() );
}

/*=========================================================================
FUNKCJA: nrnd - generuje liczby pseudolosowe o rozkladzie normalnym
                z przedzialu [0,1] (algorytm 1)
PARAMETRY: void
WYWOLANIE: nrnd();
REZULTAT: liczba pseudolosowa
-------------------------------------------------------------------------*/
double nrand::nrnd()
{
   int i;
   double r, s = -6.0;
   for (i = 0; i < 12; i++)
   {
      s = s + dlots();
```

```
   }
   return s;
}

/*============================================================================
FUNKCJA: trnd - generuje liczby pseudolosowe o rozkladzie normalnym
                z przedzialu [0,1] metoda Teichroewa (G. Trybuś)
PARAMETRY: void
WYWOLANIE: trnd();
REZULTAT: liczba pseudolosowa
----------------------------------------------------------------------------*/
double nrand::trnd()
{
   int i;
   double r, s, t;
   do
   {
      s = -6.0;
      for (i = 0; i < 12; i++) s = s + dlots();
      s = 0.25 * s;
   }
   while( fabs( s ) > 1.0 );
   t = s * s;
   r = (((( 0.029899776 * t + 0.008353968) * t + 0.076542912) * t +
            0.252408784) * t + 3.949846138) * s;
   return r;
}

/*============================================================================
FUNKCJA: dxrnd - generuje liczby pseudolosowe typu double o rozkladzie
                 normalnym dla danej wartosci oczekiwanej ex i danego
                 odchylenia standardowego stdx
PARAMETRY:
   ex - wartosc oczekiwana
   stdx - odchylenie standardowe
WYWOLANIE: dxrnd( ex, stdx );
REZULTAT: liczba pseudolosowa
----------------------------------------------------------------------------*/
double nrand::dxrnd( double ex, double stdx )
{
   double r1, r2, a, b, x, s = 6.2831853;
/* r1 = dlots();
   r2 = dlots();*/
   r1 = rm_random();
   r2 = rm_random();
   a = log(r1);
   b = cos(s * r2);
   x = ex + sqrt(-2 * stdx * a) * cos(s * b);
   return ( x );
}

/*============================================================================
FUNKCJA: fxrnd - generuje liczby pseudolosowe typu float o rozkladzie
                 normalnym dla danej wartosci oczekiwanej ex i danego
                 odchylenia standardowego stdx
PARAMETRY:
   ex - wartosc oczekiwana
   stdx - odchylenie standardowe
WYWOLANIE: fxrnd( ex, stdx );
REZULTAT: liczba pseudolosowa
----------------------------------------------------------------------------*/
float nrand::fxrnd( float ex, float stdx )
{
   int i;
   float s = 0;
   for (i = 0; i < 12; i++)
   {
      s = s + flots();
   }
   return (s - 6.0) * stdx + ex;
}

/*============================================================================
FUNKCJA: mvnd - generuje wektory losowe o wielowymiarowym rozkladzie normalnym
                przy zadanym wektorze wartosci oczekiwanych i danej macierzy
                wariancji-kowariancji
PARAMETRY:
   mVC - macierz wariancji-kowariancji
   vE - wektor wartosci oczekiwanych
```

```
   mR - macierz obliczonych wektorow losowych
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: mvnd( mVC, vE, mR, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void nrand::mvnd( array &mVC, array &vE, array &mR, int m, int n )
{
   int i, j, k;
   double s, x;
   array vS(m);                        // wektor sum-iloczynow
   array vZ(m);                        // wektor generowany losowo
   array mD((m * (m + 1)) / 2);        // macierz trojkatna wyznaczana rekurencyjnie

   for (j = 0; j < m; j++)
   {
      *(mD.b+loc(j,0)) = *(mVC.b+loc(j,0)) / sqrt(*(mVC.b+loc(0,0)));
   }
   for (j = 1; j < m; j++)
   {
      s = 0.0;
      for (k = 0; k <= j-1; k++)
      {
         s = s + (*(mD.b+loc(j,k)) * *(mD.b+loc(j,k)));
      }
      x = *(mVC.b+loc(j,j)) - s;
      if (x < 0) { cout << "?" << endl; exit(0); }
      *(mD.b+loc(j,j)) = sqrt(x);
   }
    for (i = 1; i < m; i++)
    {
      for (j = 1; j < i; j++)
      {
         s = 0.0;
         for (k = 0; k <= j-1; k++) { s = s + (*(mD.b+loc(i,k)) * *(mD.b+loc(j,k))); }
         *(mD.b+loc(i,j)) = (*(mVC.b+loc(i,j)) - s) / *(mD.b+loc(j,j));
      }
    }
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { *(vS.b+j) = 0.0; *(vZ.b+j) = dxrnd(0, 1); }
      for (j = 0; j < m; j++)
      {
         for (k = 0; k <= j; k++)
         {
            *(vS.b+j) = *(vS.b+j) + *(mD.b+loc(j,k)) * *(vZ.b+j);
         }
      }
      for (j = 0; j < m; j++) { *(mR.b+i*m+j) = *(vS.b+j) + *(vE.b+j); }
   }
   mD.~array();
   vZ.~array();
   vS.~array();
}

/*============================================================================
FUNKCJA: ex_rnd - generuje wektor srednich
PARAMETRY:
   vEX - wektor srednich
   m - liczba kolumn (cech)
WYWOLANIE: ex_rnd( vEX, m );
REZULTAT: void
------------------------------------------------------------------------------*/
void nrand::ex_rnd( array &vEX, int m, int a, int b )
{
   int i, j;
   double c;

   for (i = 0; i < m; i++)
   {
      c = dablot( a, b );
      *(vEX.b+i) = c;
   }
}

/*============================================================================
FUNKCJA: vc_rnd - oblicza macierz wariancji-kowariancji
PARAMETRY:
   mV - macierz wariancji-kowariancji
```

```
   m - liczba kolumn (cech)
WYWOLANIE: vc_rnd( mV, m );
REZULTAT: void
------------------------------------------------------------------------------*/
void nrand::vc_rnd( array &mV, int m, int a, int b )
{
   int i, j, t = b;
   double c;
   for (i = 0; i < m; i++)
   {
      c = dablot( a, b );
      *(mV.b+m*i+i) = c;
   }
   for (j = 0; j < m; j++)
   {
      for (i = 0; i <= j; i++)
      {
         if ( i != j )
         {
            do
            {
               c = dablot( a, b );
               b--;
            }
            while ( (c*c) > *(mV.b+m*i+i) );
            *(mV.b+m*i+j) = c;
            *(mV.b+m*j+i) = c;
            b = t;
         }
      }
   }
}

/*============================================================================
FUNKCJA: rm_rnd - generuje wektor liczb losowych (G. Marsaglia, A. Zaman)
PARAMETRY:
   rvec - wektor liczb losowych
   len - rozmiar wektora
WYWOLANIE: rm_rnd( rvec, len );
REZULTAT: void
------------------------------------------------------------------------------*/
int nrand::rm_rnd(float rvec[], int len)
{
   float uni;
   int ivec;
   if ( !test )
   {
      return 1;
   }
   for ( ivec=0; ivec < len; ivec++)
   {
      uni = u[i97] - u[j97];
      if ( uni < 0.0 ) uni = uni + 1.0;
      u[i97] = uni;
      i97--;
      if ( i97 < 0 ) i97 = 96;
      j97--;
      if ( j97 < 0 ) j97 = 96;
      c = c - cd;
      if ( c < 0.0 ) c = c + cm;
      uni = uni - c;
      if ( uni < 0.0 ) uni = uni + 1.0;
      rvec[ivec] = uni;
   }
   return NULL;
}

/*============================================================================
FUNKCJA: rm_start - przygotowanie zarodkow generatora
PARAMETRY:
WYWOLANIE: rm_start();
REZULTAT: void
------------------------------------------------------------------------------*/
void nrand::rm_start()
{
   int ij, kl, *sx1, *sx2;
   rm_seed(sx1, sx2);
   ij = *sx1;  kl = *sx2;
   if (1 == rm_ini(ij, kl)) exit(1);
```

```
}

/*============================================================================
FUNKCJA: rm_ini - inicjalizacja generatora
PARAMETRY:
   ij, kl - zarodki generatora
WYWOLANIE: rm_ini( ij, kl );
REZULTAT: NULL
----------------------------------------------------------------------------*/
int nrand::rm_ini(int ij, int kl)
{
   float s, t;
   int i, j, k, l, m;
   int ii, jj;
   test = TRUE;
   if ( ( ij < 0 || ij > 31328 ) || ( kl < 0 || kl > 30081 ) )
   {
      return 1;
   }
   i = fmod(ij/177.0, 177.0) + 2;
   j = fmod(ij      , 177.0) + 2;
   k = fmod(kl/169.0, 178.0) + 1;
   l = fmod(kl      , 169.0);

   for ( ii=0; ii<=96; ii++ )
   {
      s = 0.0;
      t = 0.5;
      for ( jj=0; jj<=23; jj++ )
      {
         m = fmod( fmod(i*j,179.0)*k , 179.0 );
         i = j;
         j = k;
         k = m;
         l = fmod( 53.0*l+1.0 , 169.0 );
         if ( fmod(l*m,64.0) >= 32) s = s + t;
         t = 0.5 * t;
      }
      u[ii] = s;
   }
   c = 362436.0 / 16777216.0;
   cd = 7654321.0 / 16777216.0;
   cm = 16777213.0 / 16777216.0;
   i97 = 96;
   j97 = 32;
   test = TRUE;
   return NULL;
}

/*============================================================================
FUNKCJA: rm_seed - utworzenie zarodkow generatora
PARAMETRY:
   seed1, seed2 - zarodki
WYWOLANIE: rm_seed( seed1, seed2 );
REZULTAT: void
----------------------------------------------------------------------------*/
void nrand::rm_seed( int *seed1, int *seed2 )
{
   struct tm *tm_now;
   float s_sig, s_insig, maxs_sig, maxs_insig;
   long secs_now;
   int s, m, h, d, s1, s2;
   time(&secs_now);
   tm_now = localtime(&secs_now);
   s = tm_now->tm_sec + 1;
   m = tm_now->tm_min + 1;
   h = tm_now->tm_hour + 1;
   d = tm_now->tm_yday + 1;
   maxs_sig   = 60.0 + 60.0/60.0 + 24.0/60.0/60.0 + 366.0/24.0/60.0/60.0;
   maxs_insig = 60.0 + 60.0*60.0 + 24.0*60.0*60.0 + 366.0*24.0*60.0*60.0;
   s_sig      = s + m/60.0 + h/60.0/60.0 + d/24.0/60.0/60.0;
   s_insig    = s + m*60.0 + h*60.0*60.0 + d*24.0*60.0*60.0;
   s1 = s_sig   / maxs_sig   * 31328.0;
   s2 = s_insig / maxs_insig * 30081.0;
   *seed1 = s1;
   *seed2 = s2;
}

/*============================================================================
```

```
FUNKCJA: rm_random - wygeneratorowanie liczby typu float
PARAMETRY:
WYWOLANIE: rm_random();
REZULTAT: liczba losowa
-------------------------------------------------------------------------*/
float nrand::rm_random()
{
   float temp[1];
   int len = 1;

   if (1 == rm_rnd(temp, len)) exit(1);
   return temp[0];
}

/* ------------------------------------------------------------------------
   PLIK: array.cpp
   ?: definicje funkcji klasy array
------------------------------------------------------------------------ */

/*=========================================================================
FUNKCJA: array - konstruktor obiektow klasy array
PARAMETRY:
   d1 - liczba wierszy
   d2 - liczba kolumn
   d3 - liczba warstw
WYWOLANIE: array arr( d1, d2, d3 );
REZULTAT: void
-------------------------------------------------------------------------*/
array::array( int d1, int d2, int d3 )
{
   alloc( d1, d2, d3 );
}

/*=========================================================================
FUNKCJA: ~array - destruktor obiektow klasy array
PARAMETRY: void
WYWOLANIE: ~array();
REZULTAT: void
-------------------------------------------------------------------------*/
array::~array()
{
   if ( b ) delete b;
   b = NULL;
}

/*=========================================================================
FUNKCJA: alloc - przydziela pamiec operacyjna dla tworzonej tablicy
PARAMETRY:
   d1 - liczba wierszy
   d2 - liczba kolumn
   d3 - liczba warstw
WYWOLANIE: alloc( d1, d2, d3 );
REZULTAT: void
-------------------------------------------------------------------------*/
void array::alloc( int d1, int d2, int d3 )
{
   if( d1 < 1 || d2 < 1 || d3 < 1 )
   {
      cerr << "Dimension error!" ;
      exit(1);
   }
   long int size = d1 * d2 * d3;
   set_new_handler(&::out_mem);
   double *p = b = new double[size];
   while( size-- ) *p++ = 0;
   set_new_handler( 0 );
}

/*=========================================================================
FUNKCJA: loc3 - przeksztalca trojwymiarowy adres indeksowy elementu tablicy
                na jednowymiarowy adres wskaznikowy elementu wektora
PARAMETRY:
   i - nr wiersza w macierzy danych
   j - nr kolumny w macierzy danych
   k - nr warstwy w macierzy danych
   m - liczba kolumn w macierzy danych
   v - liczba warstw w macierzy danych
WYWOLANIE: loc3( i, j, m );
REZULTAT: adres wskaznikowy
```

```
-------------------------------------------------------------------------*/
int array::loc3( int i, int j, int k, int m, int n )
{
   return ( k*m*n+i*m+j );
}

/*=========================================================================
FUNKCJA: cub2o - wybiera z trojwymiarowej tablicy danych wszystkie
                 obserwacje dla j-tej cechy
PARAMETRY:
   arr - trojwymiarowa tablica danych
   mat - macierz obserwacji j-tej cechy o wymiarach nxv
   j - nr cechy
   m - liczba cech (kolumn)
   n - liczba obiektow (wierszy)
   v - liczba okresow (warstw)
WYWOLANIE: cub2o( arr, mat, j, m, n, v );
REZULTAT: void
-------------------------------------------------------------------------*/
void array::cub2o( array &arr, array &mat, int i, int m, int n, int v )
{
   int j, k;
   for(k = 0; k < v; k++)
   {
      for(j = 0; j < m; j++)
      {
         *(mat.b+loc2(k,j,m)) = *(arr.b+loc3(i,j,k,m,n));
      }
   }
}

/*=========================================================================
FUNKCJA: cub2t - wybiera z trojwymiarowej tablicy danych wszystkie
                 obserwacje dla i-tego obiektu
PARAMETRY:
   arr - trojwymiarowa tablica danych
   mat - macierz obserwacji i-tego obiektu o wymiarach mxv
   i - nr obiektu
   m - liczba cech (kolumn)
   n - liczba obiektow (wierszy)
WYWOLANIE: cub2t( arr, mat, i, m, n );
REZULTAT: void
-------------------------------------------------------------------------*/
void array::cub2t(array &arr, array &mat, int k, int m, int n)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
      {
         *(mat.b+loc2(i,j,m)) = *(arr.b+loc3(i,j,k,m,n));
      }
   }
}

/*=========================================================================
FUNKCJA: cub2v - wybiera z trojwymiarowej tablicy danych wszystkie
                 obserwacje dla k-tego okresu
PARAMETRY:
   arr - trojwymiarowa tablica danych
   mat - macierz obserwacji k-tego okresu o wymiarach nxm
   k - nr okresu
   m - liczba cech (kolumn)
   n - liczba obiektow (wierszy)
   v - liczba okresow (warstw)
WYWOLANIE: cub2v( arr, mat, k, m, n, v );
REZULTAT: void
-------------------------------------------------------------------------*/
void array::cub2v( array &arr, array &mat, int j, int m, int n, int v )
{
   int i, k;
   for(k = 0; k < v; k++)
   {
      for(i = 0; i < n; i++)
      {
         *(mat.b+loc2(k,i,n)) = *(arr.b+loc3(i,j,k,m,n));
      }
   }
}
```

```
/* ------------------------------------------------------------------------
   PLIK: mva.cpp
      ?: definicje funkcji klasy mva
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: mva - konstruktor obiektow klasy mva
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: mva dbf(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------*/
mva::mva(const int _m, const int _n, const int _v, const int _w, const int _d,
         const char _t) : m(_m), n(_n), v(_v), w(_w), d(_d), t(_t)
{

}

/*========================================================================
FUNKCJA: xcreate - tworzy pliki DBF
PARAMETRY:
   dbf - identyfikator pliku
   name - nazwa pliku
WYWOLANIE: xcreate(dbf, name, m, n);
REZULTAT: void
------------------------------------------------------------------------*/
void mva::xcreate(Data4 &dbf, char *name, int m, int n)
{
   int i;
   char f[4], p[3], *s;
   char **fname;                    // tablica wskaznikow do nazw pol
   char *type;                      // tablica typow pol
   int *len;                        // tablica rozmiarow pol
   int *dec;                        // tablica miejsc dziesietnych
   fname = new char *[m];           // tablica wskaznikow do nazw pol
   type = new char[m];              // tablica typow pol
   len = new int[m];                // tablica rozmiarow pol
   dec = new int[m];                // tablica miejsc dziesietnych
   cb.openError = 0;
   cb.safety = 0;
   Field4info xfields(cb);
   for (i = 0; i < m; i++)
   {
      itoa(i + 1, p, 10);
      strcpy(f, "X");
      strcat(f, p);
      s = new char[ strlen(f) ];
      fname[i] = strcpy(s, f);
      type[i] = t;
      len[i] = w;
      dec[i] = d;
   }
   for (i = 0; i < m; i++)
   {
      xfields.add(fname[i], type[i], len[i], dec[i]);
   }
   dbf.create(cb, name, xfields.fields()) ;
   for (i = 0; i < n; i++) { dbf.appendBlank(); }
   if (cb.errorCode) exit(0);
   else cout << dbf.fileName() << " created successfully" << endl;
   delete fname;
   delete type;
   delete len;
   delete dec;
   delete s;
   dbf.close();
}

/*========================================================================
FUNKCJA: screate - tworzy pliki DBF wg zadanej struktury fields i dopisuje
                   rec pustych rekordow
PARAMETRY:
   dbf - identyfikator pliku
```

```
   name - nazwa pliku
   fields - struktura pliku
   rec - liczba rekordow
WYWOLANIE: screate(dbf, name, fields, rec);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::screate(Data4 &dbf, char *name, FIELD4INFO *fields, int rec)
{
   cb.openError = 0;
   cb.safety = 0;
   cb.autoOpen = 0;
   dbf.create(cb, name, fields) ;
   for (int i = 0; i < rec; i++) { dbf.appendBlank(); }
   if (cb.errorCode) exit(0);
      else cout << dbf.fileName() << " created successfully" << endl;
   dbf.close();
}

/*============================================================================
FUNKCJA: tcreate - tworzy pliki DBF wg zadanej struktury fields, tworzy plik
                   indeksow wg tags i dopisuje rec pustych rekordow
PARAMETRY:
   dbf - identyfikator pliku
   name - nazwa pliku
   fields - struktura pliku
   tags - struktura pliku indeksow
   rec - liczba rekordow
WYWOLANIE: tcreate(dbf, name, fields, tags, rec);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::tcreate(Data4 &dbf, char *name, FIELD4INFO *fields, TAG4INFO *tags, int rec)
{
   cb.openError = 0;
   cb.safety = 0;
   cb.autoOpen = 0;
   dbf.create(cb, name, fields, tags) ;
   for (int i = 0; i < rec; i++) { dbf.appendBlank(); }
   if (cb.errorCode) exit(0);
      else cout << dbf.fileName() << " created successfully" << endl;
   dbf.close();
}

/*============================================================================
FUNKCJA: ccreate - tworzy pliki DBF dla trojwymiarowej tablicy danych
PARAMETRY:
   dbf - identyfikator pliku
   name - nazwa pliku
WYWOLANIE: ccreate(dbf, name);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::ccreate(Data4 &dbf, char *name, int m, int n, int v)
{
   xcreate(dbf, name, m, n * v);
}

/*============================================================================
FUNKCJA: use_dbf - otwiera plik DBF
PARAMETRY:
   dbf - identyfikator pliku
   name - nazwa pliku DBF
WYWOLANIE: use_dbf(dbf, name);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::use_dbf(Data4 &dbf, char *name)
{
   int rc;
   cb.openError = 0;
   cb.safety = 0;
   cb.autoOpen = 0;
   rc = dbf.open(cb, name);
   if (rc == r4noOpen) { cout << name << " not opened!" << endl; exit(1); }
   cb.exitTest();
   cout << dbf.fileName() << endl;
}
/*============================================================================
FUNKCJA: is_dbf - otwiera plik DBF
PARAMETRY:
   dbf - identyfikator pliku
   name - nazwa pliku DBF
```

```
WYWOLANIE: is_dbf(dbf, name);
REZULTAT: 0 || 1
-------------------------------------------------------------------------------*/
int mva::is_dbf(Data4 &dbf, char *name)
{
   cb.openError = 0;
   cb.safety = 0;
   cb.autoOpen = 0;
   dbf.open(cb, name);
   if (!dbf.isValid()) return 0; else return 1;
}

/*===============================================================================
FUNKCJA: use_idx - otwiera (tworzy jesli nie istnieje) plik indeksow CDX
PARAMETRY:
   dbf - identyfikator pliku DBF
   idx - identyfikator pliku CDX
   idx_name - nazwa pliku CDX
   tags - struktura pliku indeksow
WYWOLANIE: use_idx(dbf, idx, idx_ame, tags);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::use_idx(Data4 &dbf, Index4 &idx, char *idx_name, TAG4INFO *tags)
{
   int rc;
   cb.openError = 0;
   cb.safety = 0;
   cb.autoOpen = 0;
   rc = idx.open(dbf, idx_name);
   if (rc == r4noOpen)
   {
      cout << idx_name << " not opened. Creating ..." << endl;
      idx.create(dbf, idx_name, tags);
   }
   cb.exitTest();
   cout << idx.fileName() << endl;
}

/*===============================================================================
FUNKCJA: gets - pobiera zawartosc pola i zwraca w postaci lancucha znakow
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
WYWOLANIE: gets(dbf, r, f);
REZULTAT: string
-------------------------------------------------------------------------------*/
char *mva::gets(Data4 &dbf, long int r, int f)
{
   dbf.go(r);
   Field4 fld(dbf, f);
   return (fld.str());
}

/*===============================================================================
FUNKCJA: get_s - pobiera zawartosc pola i zwraca w postaci lancucha znakow
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
WYWOLANIE: get_s(dbf, f);
REZULTAT: string
-------------------------------------------------------------------------------*/
char *mva::get_s(Data4 &dbf, int f)
{
   Field4 fld(dbf, f);
   return (fld.str());
}

/*===============================================================================
FUNKCJA: repl_s - zapisuje w polu rekordu biezacego lancuch znakow
PARAMETRY:
   dbf - identyfikator pliku
   f - numer pola
   s - lancuch zankow
WYWOLANIE: repl_s(dbf, f, s);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::repl_s(Data4 &dbf, int f, char *s)
```

```
{
   Field4 fld(dbf, f);
   fld.assign(s);
}

/*============================================================================
FUNKCJA: putd - zapisuje w polu pliku wartosc numeryczna typu double
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
   x - liczba typu double
WYWOLANIE: putd(dbf, r, f, x);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::putd(Data4 &dbf, long int r, int f, double x)
{
   dbf.go(r);
   Field4 fld(dbf, f);
   fld.assignDouble(x);
}

/*============================================================================
FUNKCJA: put_d - zapisuje w polu rekordu biezacego wartosc numeryczna
                 typu double
PARAMETRY:
   dbf - identyfikator pliku
   f - numer pola
   x - liczba typu double
WYWOLANIE: put_d(dbf, f, x);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::put_d(Data4 &dbf, int f, double x)
{
   Field4 fld(dbf, f);
   fld.assignDouble(x);
}

/*============================================================================
FUNKCJA: put_i - zapisuje w polu rekordu biezacego wartosc numeryczna typu
                 long int
PARAMETRY:
   dbf - identyfikator pliku
   f - numer pola
   x - liczba typu double
WYWOLANIE: put_i(dbf, f, x);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::put_i(Data4 &dbf, int f, long int x)
{
   Field4 fld(dbf, f);
   fld.assignLong(x);
}

/*============================================================================
FUNKCJA: geti - pobiera z pola pliku wartosc numeryczna typu int
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
WYWOLANIE: geti(dbf, r, f);
REZULTAT: liczba typu int
----------------------------------------------------------------------------*/
long int mva::geti(Data4 &dbf, long int r, int f)
{
   dbf.go(r);
   Field4 fld(dbf, f);
   return ((long int) fld);
}

/*============================================================================
FUNKCJA: get_i - pobiera z pola rekordu biezacego wartosc numeryczna typu int
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
WYWOLANIE: get_i(dbf, r, f);
REZULTAT: liczba typu int
----------------------------------------------------------------------------*/
```

```
long int mva::get_i(Data4 &dbf, int f)
{
   Field4 fld(dbf, f);
   return ((long int) fld);
}

/*============================================================================
FUNKCJA: getd - pobiera z pola pliku wartosc numeryczna typu double
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
WYWOLANIE: getd(dbf, r, f);
REZULTAT: liczba typu double
----------------------------------------------------------------------------*/
double mva::getd(Data4 &dbf, long int r, int f)
{
   dbf.go(r);
   Field4 fld(dbf, f);
   return ((double) fld);
}

/*============================================================================
FUNKCJA: get_d - pobiera z pola pliku wartosc numeryczna typu double
PARAMETRY:
   dbf - identyfikator pliku
   r - numer rekordu
   f - numer pola
WYWOLANIE: get_d(dbf, f);
REZULTAT: liczba typu double
----------------------------------------------------------------------------*/
double mva::get_d(Data4 &dbf, int f)
{
   Field4 fld(dbf, f);
   return ((double) fld);
}

/*============================================================================
FUNKCJA: disp - wyswietla na ekranie monitora zawartosc pliku DBF
PARAMETRY:
   dbf - identyfikator pliku
WYWOLANIE: disp(dbf);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::disp(Data4 &dbf)
{
   int i, j;
   cout << dbf.fileName() << endl;
   for (i = 1; i <= n; i++)
   {
      for (j = 1; j <= m; j++) { cout << gets(dbf, i, j) << " " ; }
      cout << endl;
   }
}

/*============================================================================
FUNKCJA: dbf2mat - przepisuje zawartosc pliku DBF do macierzy
PARAMETRY:
   dbf - identyfikator pliku
   mD - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: dbf2mat(dbf, mD, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::dbf2mat(Data4 &dbf, array &mD, int m, int n)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { *(mD.b+loc2(i,j,m)) = getd(dbf, i+1, j+1); }
   }
}

/*============================================================================
FUNKCJA: dbf2arr - przepisuje zawartosc pliku DBF do macierzy
PARAMETRY:
   dbf - identyfikator pliku
   mD - macierz danych
```

```
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: dbf2arr(dbf, mD, m, n, v);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::dbf2arr(Data4 &dbf, array &mD, int m, int n, int v)
{
   int i, j;
   for (i = 0; i < n*v; i++)
   {
      for (j = 0; j < m; j++) { *(mD.b+loc2(i,j,m)) = getd(dbf, i+1, j+1); }
   }
}

/*============================================================================
FUNKCJA: dbf2vec
PRZEZNACZENIE: przepisuje zawartosc dolnego trojkata pliku DBF do wektora
PARAMETRY:
   dbf - identyfikator pliku
   vD - wektor danych
   n - rozmiar wektora
WYWOLANIE: dbf2vec(dbf, vD, n);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::dbf2vec(Data4 &dbf, array &vD, int n)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j <= i; j++) { *(vD.b+loc(i,j)) = getd(dbf, i+1, j+1); }
   }
}

/*============================================================================
FUNKCJA: rec2vec
PRZEZNACZENIE: przepisuje zawartosc biezacego rekordu pliku DBF do wektora
PARAMETRY:
   dbf - identyfikator pliku
   r - nr rekordu biezacego
   vD - wektor danych
   n - rozmiar wektora
WYWOLANIE: rec2vec(dbf, r, vD, n);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::rec2vec(Data4 &dbf, long int r, array &vD, int n)
{
   int j;
   dbf.go(r);
   for (j = 0; j < n; j++) { *(vD.b+j) = get_d(dbf, j+1); }
}

/*============================================================================
FUNKCJA: mat2dbf - zapisuje zawartosc macierzy w pliku DBF
PARAMETRY:
   dbf - identyfikator pliku
   mD - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: mat2dbf(dbf, mD, m, n);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::mat2dbf(Data4 &dbf, array &mD, int m, int n)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { putd(dbf, i+1, j+1, *(mD.b+loc2(i,j,m))); }
   }
}

/*============================================================================
FUNKCJA: mv2dbf - zapisuje zawartosc macierzy oraz wektora w pliku DBF
PARAMETRY:
   dbf - identyfikator pliku
   mD - macierz danych
   vD - wektor danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: mv2dbf(dbf, mD, vD, m, n);
```

```
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::mv2dbf(Data4 &dbf, array &mD, array &vD, int m, int n)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { putd(dbf, i+1, j+1, *(mD.b+loc2(i,j,m))); }
   }
   for (j = 0; j < m; j++) { putd(dbf, n+1, j+1, *(vD.b+j)); }
}

/*=========================================================================
FUNKCJA: vec2dbf - zapisuje zawartosc wektora w biezacym rekordzie pliku DBF
                   (start od pola nr f)
PARAMETRY:
   dbf - identyfikator pliku
   vD - wektor danych
   f - nr pierwszego pola
   n - rozmiar wektora vD
WYWOLANIE: vec2dbf(dbf, vD, r, f, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::vec2dbf(Data4 &dbf, array &vD, int f, int n)
{
   int j;

   for (j = 0; j < n; j++) { put_d(dbf, j+f, *(vD.b+j)); }
}

/*=========================================================================
FUNKCJA: v2dbf - zapisuje zawartosc wektora w biezacym rekordzie pliku DBF
                 (start od pola nr f)
PARAMETRY:
   dbf - identyfikator pliku
   vD - wektor danych
   f - nr pola
   n - rozmiar wektora vD
WYWOLANIE: v2dbf(dbf, vD, f, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::v2dbf(Data4 &dbf, array &vD, int f, int n)
{
   long int i;
   for (i = 0; i < n; i++) { putd(dbf, i+1, f, *(vD.b+i)); }
}


/*=========================================================================
FUNKCJA: mtx_dup - wykonuje duplikat tablicy S w tablicy D
PARAMETRY:
   mS - macierz zrodlowa
   mD - macierz-duplikat
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: mtx_dup(mS, mD, m, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::mtx_dup(array &mS, array &mD, int m, int n)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
      {
         *(mD.b+loc2(i,j,m)) = *(mS.b+loc2(i,j,m));
      }
   }
}

/*=========================================================================
FUNKCJA: list_m - zapisuje w pliku TXT zawartosc macierzy danych
PARAMETRY:
   f - identyfikator pliku
   mD - macierz danych
   m - liczba kolumn
   n - liczba wierszy
   w - rozmiar pola
   d - liczba miejsc dziesietnych
```

```
WYWOLANIE: list_m(f, mD, m, n, w, d);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::list_m(ofstream &f, array &mD, int m, int n, int w = 9, int d = 6)
{
   int i, j;
// This example sets scientific notation:
// cout<<setf(ios::scientific, ios::floatfield)<<f;
// f << f.setf(ios::scientific, ios::floatfield) << *(mD.b+loc2(i,j,m))<<" | ";
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
      {
         f << setw(w) << setprecision(d) << round(*(mD.b+loc2(i,j,m)), 3);
      }
      f << endl;
   }
}

/*============================================================================
FUNKCJA: list_vh - zapisuje w pliku TXT zawartosc wektora w ukladzie poziomym
PARAMETRY:
   f - identyfikator pliku
   vD - wektor danych
   n - dlugosc wektora
   w - rozmiar pola
   d - liczba miejsc dziesietnych
WYWOLANIE: list_vh(f, vD, n, w, d);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::list_vh(ofstream &f, array &vD, int n, int w = 10, int d = 2)
{
   int i;
   for (i = 0; i < n; i++)
   {
      f << setw(w) << setprecision(d) << setfill(' ') << round(*(vD.b+i), 2);
   }
   f << endl;
}

/*============================================================================
FUNKCJA: list_vv - zapisuje w pliku TXT zawartosc wektora w ukladzie pionowym
PARAMETRY:
   f - identyfikator pliku
   vD - wektor danych
   n - dlugosc wektora
   w - rozmiar pola
   d - liczba miejsc dziesietnych
WYWOLANIE: list_vv(f, vD, n, w, d);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::list_vv(ofstream &f, array &vD, int n, int w = 8, int d = 3)
{
   int i;
   for (i = 0; i < n; i++)
   {
      f << setw(w) << setprecision(d) << *(vD.b+i) << endl;
   }
}

/*============================================================================
FUNKCJA: list_lwr - zapisuje w pliku TXT zawartosc dolnego trojkata
                    symetrycznej macierzy danych
PARAMETRY:
   f - identyfikator pliku
   mD - macierz danych
   n - liczba wierszy macierzy
   w - rozmiar pola
   d - liczba miejsc dziesietnych
WYWOLANIE: list_lwr(f, mD, n, w, d);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::list_lwr(ofstream &f, array &mD, int n, int w = 8, int d = 3)
{
   int i, j;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j <= i; j++)
      {
```

```
         f << setw(w) << setprecision(d) << *(mD.b+i*n+j);
      }
      f << endl;
   }
}

/*============================================================================
FUNKCJA: list_upr - zapisuje w pliku TXT zawartosc gornego trojkata
                    symetrycznej macierzy danych
PARAMETRY:
   f - identyfikator pliku
   mD - macierz danych
   n - liczba wierszy macierzy
   w - rozmiar pola
   d - liczba miejsc dziesietnych
WYWOLANIE: list_upr(f, mD, n, w, d);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::list_upr(ofstream &f, array &mD, int n, int w = 8, int d = 3)
{
   int i, j, k;
   for (i = 0; i < n; i++)
   {
      for (k = 0; k < i*w; k++) f << " ";
      for (j = i; j < n; j++)
      {
         f << setw(w) << setprecision(d) << *(mD.b+i*n+j);
      }
      f << endl;
   }
}

/*============================================================================
FUNKCJA: list_t - zapisuje w pliku TXT zawartosc dolnego trojkata
                  symetrycznej macierzy danych
PARAMETRY:
   f - identyfikator pliku
   mD - macierz danych
   n - liczba wierszy macierzy
   w - rozmiar pola
   d - liczba miejsc dziesietnych
WYWOLANIE: list_t(f, mD, n, w, d);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::list_t(ofstream &f, array &mD, int n, int w = 8, int d = 3)
{
   int i, j, k = 0;
   for (i = 0; i < n-1; i++)
   {
      for (j = 0; j <= i; j++)
      {
         f << setw(w) << setprecision(d) << *(mD.b+k++);
      }
      f << endl;
   }
}

/*============================================================================
FUNKCJA: tr2vec - przepisuje dolny trojkat macierzy symetrycznej do wektora
PARAMETRY:
   mS - macierz symetryczna
   vD - wektor
   n - rozmiar wektora || stopien macierzy
WYWOLANIE: tr2vec(mS, vD, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::tr2vec(array &mS, array &vD, int n)
{
   int i, j, k = 0;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j <= i; j++)
      {
         *(vD.b+k) = *(mS.b+loc2(i,j,n));
         k++;
      }
   }
}
```

```
/*============================================================================
FUNKCJA: t2vec - przepisuje dolny trojkat macierzy symetrycznej do wektora
PARAMETRY:
   mS - macierz symetryczna
   vD - wektor
   n - rozmiar wektora || stopien macierzy
WYWOLANIE: t2vec(mS, vD, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::t2vec(array &mDAT, int n)
{
   int i, j, k = 0;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j <= i; j++)
      {
         *(mDAT.b+k) = *(mDAT.b+loc2(i,j,n));
         k++;
      }
   }
}

/*============================================================================
FUNKCJA: keep - pozostawienie (1) lub usuniecie (0) wiersza/kolumny
PARAMETRY:
   vH - wektor z numerami usuwanych wierszy/kolumn
   h - rozmiar wektora vH (liczba usuwanych wierszy/kolumn)
   x - numer biezacego wiersza/kolumny
WYWOLANIE: keep(vH, h, x);
REZULTAT: true or false
----------------------------------------------------------------------------*/
int mva::keep(int *vH, int h, int x)
{
   int i;
   for (i = 0; i < h; i++)
   {
      if (vH[i] == x) return 0;
   }
   return 1;
}

/*============================================================================
FUNKCJA: keep - pozostawienie (1) lub usuniecie (0) wiersza/kolumny
PARAMETRY:
   vH - wektor z numerami usuwanych wierszy/kolumn
   h - rozmiar wektora vH (liczba usuwanych wierszy/kolumn)
   x - numer biezacego wiersza/kolumny
WYWOLANIE: keep(vH, h, x);
REZULTAT: true or false
----------------------------------------------------------------------------*/
int mva::keep(array &vH, int h, double x)
{
   int i;
   for (i = 0; i < h; i++)
   {
      if (*(vH.b+i) == x) return 0;
   }
   return 1;
}

/*============================================================================
FUNKCJA: del_r - redukcja wierszy z macierzy A
PARAMETRY:
   mA - macierz zrodlowa
   mB - macierz zredukowana
   vH - wektor z numerami usuwanych wierszy
   m - liczba kolumn macierzy mA
   n - liczba wierszy macierzy mA
   h - rozmiar wektora vH (liczba usuwanych wierszy)
WYWOLANIE: del_r(mA, mB, vH, m, n, h);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::del_r(array &mA, array &mB, int *vH, int m, int n, int h)
{
   int i, j, k = 0;
   if (h > n) return;
   for (i = 0; i < n; i++)
   {
      if (keep(vH, h, i))
```

```
      {
         for(j = 0; j < m; j++)
         {
            *(mB.b+k*m+j) = *(mA.b+i*m+j);
         }
         k++;
      }
   }
}

/*============================================================================
FUNKCJA: del_c - redukcja kolumn z macierzy A
PARAMETRY:
   mA - macierz zrodlowa
   mB - macierz zredukowana
   vH - wektor z numerami usuwanych kolumn
   m - liczba kolumn macierzy mA
   n - liczba wierszy macierzy mA
   h - rozmiar wektora vH (liczba usuwanych kolumn)
WYWOLANIE: del_c(mA, mB, vH, m, n, h);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::del_c(array &mA, array &mB, int *vH, int m, int n, int h)
{
   int i, j, k = 0, g = m - h;
   if (h > m) return;
   for (j = 0; j < m; j++)
   {
      if (keep(vH, h, j))
      {
         for(i = 0; i < n; i++)
         {
            *(mB.b+i*g+k) = *(mA.b+i*m+j);
         }
         k++;
      }
   }
}

/*============================================================================
FUNKCJA: add_r - dolaczenie wierszy do macierzy A
PARAMETRY:
   mA - macierz zrodlowa
   mB - macierz powiekszona
   mH - macierz dolaczana
   m - liczba kolumn macierzy mA
   n - liczba wierszy macierzy mA
   h - liczba wierszy macierzy mH
WYWOLANIE: add_r(mA, mB, mH, m, n, h);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::add_r(array &mA, array &mB, array &mH, int m, int n, int h)
{
   int i, j, k = 0, g = n+h;
   mtx_dup(mA, mB, m, n);
   for (i = n; i < g; i++)
   {
      for(j = 0; j < m; j++)
      {
         *(mB.b+i*m+j) = *(mH.b+k*m+j);
      }
      k++;
   }
}

/*============================================================================
FUNKCJA: add_c - dolaczenie kolumn do macierzy A
PARAMETRY:
   mA - macierz zrodlowa
   mB - macierz powiekszona
   mH - macierz dolaczana
   m - liczba kolumn macierzy mA
   n - liczba wierszy macierzy mA
   h - liczba kolumn macierzy mH
WYWOLANIE: add_c(mA, mB, mH, m, n, h);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::add_c(array &mA, array &mB, array &mH, int m, int n, int h)
{
```

```
   int i, j, k = 0, g = m+h;
   for (j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++)
      {
         *(mB.b+i*g+j) = *(mA.b+i*m+j);
      }
   }
   for (j = m; j < g; j++)
   {
      for(i = 0; i < n; i++)
      {
         *(mB.b+i*g+j) = *(mH.b+i*h+k);
      }
      k++;
   }
}

/*============================================================================
FUNKCJA: del_i - redukcja elementow z wektora X
PARAMETRY:
   vX - wektor zrodlowa
   vY - wektor zredukowany
   vH - wektor z numerami usuwanych elementow
   n - rozmiar wektora vX
   h - rozmiar wektora vH (liczba usuwanych elementow)
WYWOLANIE: del_i(vX, vY, vH, n, h);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::del_i(array &vX, array &vY, array &vH, int n, int h)
{
   int i, j, k = 0;
   if (h > n) return;
   for (i = 0; i < n; i++)
   {
// cout<<"i: "<<i<<"   X: "<<*(vX.b+i)<<"   H: "<<vH[i]<<endl;
//    if (*(vX.b+i) != vH[i])
      if (keep(vH, h, *(vX.b+i)))
      {
         *(vY.b+k) = *(vX.b+i);
         k++;
      }
   }
}

/*============================================================================
FUNKCJA: index - zwaraca indeks elementu tablicy
PARAMETRY:
   vH - tablica danych
   x - element
   h - rozmiar tablicy vH
WYWOLANIE: index(vH, h, x);
REZULTAT: index || -1
----------------------------------------------------------------------------*/
int mva::index(array &vH, int h, double x)
{
   int i;
   for (i = 0; i < h; i++)
   {
      if (*(vH.b+i) == x) return i;
   }
   return -1;
}

/*============================================================================
FUNKCJA: rang - przypisanie rang
PARAMETRY:
   vP - tablica obiektow uporzadkowanych pierwotnie
   vW - tablica obiektow uporzadkowanych wtornie
   vR - tablica rang wtornych
   h - rozmiar tablic
WYWOLANIE: rang(vP, vW, vR, h);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::rang(array &vP, array &vW, array &vR, int h)
{
   int i;
   double x;
   for (i = 0; i < h; i++)
```

```
   {
      x = *(vP.b+i);
      *(vR.b+i) = index(vW, h, x) + 1;
   }
}

/*============================================================================
FUNKCJA: sort_up - sortuje niemalejaco tablice liczb typu double
PARAMETRY:
   arr - tablica
   b - indeks dolny
   t - indeks gorny
WYWOLANIE: sort_up(arr, b, t);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::qsort_up(array &arr, int l, int r)
{
   int i, j;
   double x, y;
   i = l;   j = r;
   x = *(arr.b+((l + r) / 2));
   do
   {
      while (*(arr.b+i) < x) i++;
      while (x < *(arr.b+j)) j--;
      if (i <= j)
      {
         y = *(arr.b+i);
         *(arr.b+i) = *(arr.b+j);
         *(arr.b+j) = y;
         i++;  j--;
      }
   }
   while (i < j);
   if (l < j) qsort_up(arr, l, j);
   if (i < r) qsort_up(arr, i, r);
}

/*============================================================================
FUNKCJA: sort_dn - sortuje nierosnaco tablice liczb typu double
PARAMETRY:
   arr - tablica
   b - indeks dolny
   t - indeks gorny
WYWOLANIE: sort_dn(arr, b, t);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::qsort_dn(array &arr, int l, int r)
{
   int i, j;
   double x, y;
   i = l;   j = r;
   x = *(arr.b+((l + r) / 2));
   do
   {
      while (*(arr.b+i) > x) i++;
      while (x > *(arr.b+j)) j--;
      if (i <= j)
      {
         y = *(arr.b+i);
         *(arr.b+i) = *(arr.b+j);
         *(arr.b+j) = y;
         i++;  j--;
      }
   }
   while (i < j);
   if (l < j) qsort_dn(arr, l, j);
   if (i < r) qsort_dn(arr, i, r);
}

/*============================================================================
FUNKCJA: sort_dn - sortuje nierosnaco tablice liczb typu int
PARAMETRY:
   arr - tablica
   b - indeks dolny
   t - indeks gorny
WYWOLANIE: sort_dn(arr, b, t);
REZULTAT: void
----------------------------------------------------------------------------*/
```

```
void mva::qsort_dn(int *arr, int l, int r)
{
   int i, j, x, y;
   i = l;   j = r;
   x = arr[ (l + r) / 2 ];
   do
   {
      while (arr[i] > x) i++;
      while (x > arr[j]) j--;
      if (i <= j)
      {
         y = arr[i];
         arr[i] = arr[j];
         arr[j] = y;
         i++;  j--;
      }
   }
   while (i < j);
   if (l < j) qsort_dn(arr, l, j);
   if (i < r) qsort_dn(arr, i, r);
}

/*============================================================================
FUNKCJA: sort_dn - sortuje nierosnaco tablice liczb typu double oraz ustawia
                   rangi obiektow
PARAMETRY:
   arr - tablica wartosci cech
   tab - tablica rang
   b - indeks dolny
   t - indeks gorny
WYWOLANIE: sort_dn(arr, tab, b, t);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::qsort_dn(array &arr, int *tab, int l, int r)
{
   int i, j;
   double x, y;
   i = l;   j = r;
   x = *(arr.b+((l + r) / 2));
   do
   {
      while (*(arr.b+i) > x) i++;
      while (x > *(arr.b+j)) j--;
      if (i <= j)
      {
         y = *(arr.b+i);
         *(arr.b+i) = *(arr.b+j);
         *(arr.b+j) = y;
         y = tab[i];
         tab[i] = tab[j];
         tab[j] = y;
         i++;  j--;
      }
   }
   while (i < j);
   if (l < j) qsort_dn(arr, tab, l, j);
   if (i < r) qsort_dn(arr, tab, i, r);
}

/*============================================================================
FUNKCJA: sort_dn - sortuje nierosnaco tablice liczb typu double oraz ustawia
                   rangi obiektow
PARAMETRY:
   arr - tablica wartosci cech
   tab - tablica rang
   b - indeks dolny
   t - indeks gorny
WYWOLANIE: sort_dn(arr, tab, b, t);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::qsort_dn(array &arr, long int *tab, int l, int r)
{
   int i, j;
   double x, y;
   i = l;   j = r;
   x = *(arr.b+((l + r) / 2));
   do
   {
      while (*(arr.b+i) > x) i++;
```

```
      while (x > *(arr.b+j)) j--;
      if (i <= j)
      {
         y = *(arr.b+i);
         *(arr.b+i) = *(arr.b+j);
         *(arr.b+j) = y;
         y = tab[i];
         tab[i] = tab[j];
         tab[j] = y;
         i++;  j--;
      }
   }
   while (i < j);
   if (l < j) qsort_dn(arr, tab, l, j);
   if (i < r) qsort_dn(arr, tab, i, r);
}

/*============================================================================
FUNKCJA: sort_up - sortuje niemalejaco tablice liczb typu double oraz ustawia
                   rangi obiektow
PARAMETRY:
   arr - tablica wartosci cech
   tab - tablica rang
   b - indeks dolny
   t - indeks gorny
WYWOLANIE: sort_up(arr, tab, b, t);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::qsort_up(array &arr, long int *tab, int l, int r)
{
   int i, j;
   double x, y;
   i = l;   j = r;
   x = *(arr.b+((l + r) / 2));
   do
   {
      while (*(arr.b+i) < x) i++;
      while (x < *(arr.b+j)) j--;
      if (i <= j)
      {
         y = *(arr.b+i);
         *(arr.b+i) = *(arr.b+j);
         *(arr.b+j) = y;
         y = tab[i];
         tab[i] = tab[j];
         tab[j] = y;
         i++;  j--;
      }
   }
   while (i < j);
   if (l < j) qsort_up(arr, tab, l, j);
   if (i < r) qsort_up(arr, tab, i, r);
}

/*============================================================================
FUNKCJA: odd - okresla, czy dana liczba jest nieparzysta
PARAMETRY:
   x - testowana liczba
WYWOLANIE: odd(x);
REZULTAT: 1-liczba nieparzysta, 0-liczba parzysta
----------------------------------------------------------------------------*/
int mva::odd(int x)
{
   int r;
   r = x % 2;
   return (r ? 1 : 0);
}

/*============================================================================
FUNKCJA: rep_v - zapisuje skalar d na elementach wektora vS
PARAMETRY:
   vS - wektor zrodlowa
   d - skalar
   n - rozmiar wektora
WYWOLANIE: rep_v(vS, d, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::rep_v(array &vS, double d, int n)
{
```

```
   int i;
   for (i = 0; i < n; i++) { *(vS.b+i) = d; }
}

/*============================================================================
FUNKCJA: diag - zapisuje skalar d na glownej przekatnej oraz liczbe x poza
                glowna przekatna macierzy mS
PARAMETRY:
   mS - macierz zrodlowa
   d - element diagonalny
   x - element poza glowna przekatna
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: diag(mS, d, x, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::diag(array &mS, double d, double x, int m, int n)
{
   int i, j;
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++)
      {
         if (i == j) *(mS.b+m*i+j) = d;    else *(mS.b+m*i+j) = x;
      }
   }
}

/*============================================================================
FUNKCJA: vdiag - zapisuje wektor vD na glownej przekatnej oraz liczbe x poza
                 glowna przekatna macierzy mS
PARAMETRY:
   mS - macierz zrodlowa
   vD - wektor
   x - element poza glowna przekatna
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: vdiag(mS, vD, x, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::vdiag(array &mS, array &vD, double x, int m, int n)
{
   int i, j;
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++)
      {
         if (i == j) *(mS.b+m*i+j) = *(vD.b+i); else *(mS.b+m*i+j) = x;
      }
   }
}

/*============================================================================
FUNKCJA: transp - wykonuje transpozycje macierzy
PARAMETRY:
   mDAT - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: transp(mDAT, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::transp(array &mDAT, int m, int n)
{
   int i, j;
   array mTMP(m, n);
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { *(mTMP.b+m*j+i) = *(mDAT.b+m*i+j); }
   }
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { *(mDAT.b+m*i+j) = *(mTMP.b+m*i+j); }
   }
   mTMP.~array();
}

/*============================================================================
FUNKCJA: mult - oblicza iloczyn macierzy C=A*B
PARAMETRY:
```

```
   mA - macierz A
   mB - macierz B
   mC - obliczony iloczyn macierzy A i B
   m - liczba wierszy macierzy A
   r - liczba kolumn macierzy A i liczba wierszy macierzy B
   n - liczba kolumn macierzy B
WYWOLANIE: mult(mA, mB, mC, m, r, n);
REZULTAT:
------------------------------------------------------------------------------*/
void mva::mult(array &mA, array &mB, array &mC, int m, int r, int n)
{
   int i, j, k;
   double s;
   for (i = 0; i < m; i++)
   {
      for (k = 0; k < n; k++)
      {
         s = 0.0;
         for (j = 0; j < r; j++) { s = s + *(mA.b+r*i+j) * *(mB.b+n*j+k); }
         *(mC.b+n*i+k) = s;
      }
   }
}

/*============================================================================
FUNKCJA: norm_cor - oblicza norme macierzy korelacji
PARAMETRY:
   mS - macierz danych
   m - stopien macierzy korelacji
WYWOLANIE: norm_cor(mS, m);
REZULTAT: norma macierzy mS
------------------------------------------------------------------------------*/
double mva::norm_cor(array &mS, int m)
{
   int i, j;
   double s = 0.0;
   for (i = 0; i < m; i++)
   {
      for (j = 0; j < i; j++)
      {
         if ( *(mS.b+i*m+j) > 0 ) { s = s + *(mS.b+i*m+j); }
      }
   }
   return ( 2 * s / (m * (m - 1)) );
}

/*============================================================================
FUNKCJA: norm_frob - oblicza norme Frobeniusa macierzy danych
PARAMETRY:
   mS - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: norm_frob(mS, m, n);
REZULTAT: norma macierzy
------------------------------------------------------------------------------*/
double mva::norm_frob(array &mS, int m, int n)
{
   int i, j;
   double s = 0.0;
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++) { s = s + *(mS.b+loc2(i,j,m)) * *(mS.b+loc2(i,j,m)); }
   }
   return (sqrt(s));
}

/*============================================================================
FUNKCJA: max_v - znajduje najwieksza wartosc w wektorze
PARAMETRY:
   vS - wektor danych
   n - liczba elementow wektora
WYWOLANIE: max_v(vS, n);
REZULTAT: najwiekszy element wektora vS
------------------------------------------------------------------------------*/
double mva::max_v(array &vS, int n)
{
   int i;
   double x = *(vS.b+0);
   for (i = 0; i < n; i++) { if (*(vS.b+i) > x) x = *(vS.b+i); }
```

```
   return (x);
}

/*============================================================================
FUNKCJA: max_m - znajduje najwieksza wartosc w macierzy
PARAMETRY:
   mS - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: max_m(mS, vM, m, n);
REZULTAT: najwiekszy element macierzy mS
----------------------------------------------------------------------------*/
double mva::max_m(array &mS, int m, int n)
{
   int i, j;
   double x = *(mS.b+0);
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++)
      {
         if (*(mS.b+loc2(i,j,m)) > x)  x = *(mS.b+loc2(i,j,m));
      }
   }
   return (x);
}

/*============================================================================
FUNKCJA: max_x - znajduje najwieksza wartosc w kazdej z kolumn macierzy mS
PARAMETRY:
   mS - macierz danych
   vM - wektor zawierajacy najwieksze wartosci poszczegolnych kolumn
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: max_x(mS, vM, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::max_x(array &mS, array &vM, int m, int n)
{
   int i, j;
   double x;
   for (j = 0; j < m; j++)
   {
      x = *(mS.b+j);
      for (i = 0; i < n; i++)
      {
         if (*(mS.b+loc2(i,j,m)) > x) x = *(mS.b+loc2(i,j,m));
      }
      *(vM.b+j) = x;
   }
}

/*============================================================================
FUNKCJA: min_v - znajduje najmniejsza wartosc w wektorze
PARAMETRY:
   vS - wektor danych
   n - liczba elementow wektora
WYWOLANIE: min_v(vS, n);
REZULTAT: najmniejszy element wektora vS
----------------------------------------------------------------------------*/
double mva::min_v(array &vS, int n)
{
   int i;
   double x = *(vS.b+0);
   for (i = 0; i < n; i++) { if (*(vS.b+i) < x) x = *(vS.b+i); }
   return (x);
}

/*============================================================================
FUNKCJA: min_m - znajduje najmniejsza wartosc w macierzy
PARAMETRY:
   mS - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: min_m(mS, m, n);
REZULTAT: najmniejszy element macierzy mS
----------------------------------------------------------------------------*/
double mva::min_m(array &mS, int m, int n)
{
   int i, j;
```

```
   double x = *(mS.b+0);
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++)
      {
         if (*(mS.b+loc2(i,j,m)) < x)  x = *(mS.b+loc2(i,j,m));
      }
   }
   return (x);
}

/*============================================================================
FUNKCJA: min_x - znajduje najmniejsza wartosc w kazdej z kolumn macierzy mS
PARAMETRY:
   mS - macierz danych
   vM - wektor zawierajacy najmniejsze wartosci poszczegolnych kolumn
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: min_x(mS, vM, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::min_x(array &mS, array &vM, int m, int n)
{
   int i, j;
   double x;
   for (j = 0; j < m; j++)
   {
      x = *(mS.b+j);
      for (i = 0; i < n; i++)
      {
         if (*(mS.b+loc2(i,j,m)) < x) x = *(mS.b+loc2(i,j,m));
      }
      *(vM.b+j) = x;
   }
}

/*============================================================================
FUNKCJA: mean_v - oblicza srednia arytmetyczna z elementow wektora
PARAMETRY:
   vS - wektor danych
   n - rozmiar wektora
WYWOLANIE: mean_v(vS, n);
REZULTAT: wartosc srednia typu double
----------------------------------------------------------------------------*/
double mva::mean_v(array &vS, int n)
{
   int i;
   double s = 0.0;
   for (i = 0; i < n; i++) { s = s + *(vS.b+i); }
   return (s / n);
}

/*============================================================================
FUNKCJA: mean_m - oblicza srednia arytmetyczna z elementow macierzy mS
PARAMETRY:
   mS - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: mean_m(mS, m, n);
REZULTAT: wartosc srednia typu double
----------------------------------------------------------------------------*/
double mva::mean_m(array &mS, int m, int n)
{
   int i, j;
   double s = 0.0;
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++) { s = s + *(mS.b+loc2(i,j,m)); }
   }
   return (s / (m*n));
}

/*============================================================================
FUNKCJA: mean_x - oblicza wektor srednich arytmetycznych z macierzy
PARAMETRY:
   mS - macierz danych
   vM - obliczony wektor srednich
   m - liczba kolumn
   n - liczba wierszy
```

```
WYWOLANIE: mean_x(mS, vM, m, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::mean_x(array &mS, array &vM, int m, int n)
{
   int i, j;
   double s;
   for (j = 0; j < m; j++)
   {
      s = 0.0;
      for (i = 0; i < n; i++) { s = s + *(mS.b+loc2(i,j,m)); }
      *(vM.b+j) = s / n;
   }
}

/*=========================================================================
FUNKCJA: stdev_v - oblicza odchylenie standardowe z elementow wektora vS
PARAMETRY:
   vS - wektor danych
   n - rozmiar wektora
WYWOLANIE: stdev_v(vS, n);
REZULTAT: wartosc odchylenia standardowego typu double
-------------------------------------------------------------------------*/
double mva::stdev_v(array &vS, int n)
{
   int i;
   double ex, s = 0.0;
   ex = mean_v(vS, n);      // srednia arytmetyczna z elementow wektora vS
   for (i = 0; i < n; i++) { s = s + pow(*(vS.b+i) - ex, 2); }
   return (sqrt(s/n));
}

/*=========================================================================
FUNKCJA: stdev_m - oblicza odchylenie standardowe z elementow macierzy mS
PARAMETRY:
   mS - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: stdev_m(mS, m, n);
REZULTAT: wartosc odchylenia standardowego typu double
-------------------------------------------------------------------------*/
double mva::stdev_m(array &mS, int m, int n)
{
   int i, j;
   double ex, s = 0.0;
   ex = mean_m(mS, m, n);  // srednia arytmetyczna z macierzy mS
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++) { s = s + pow(*(mS.b+loc2(i,j,m)) - ex, 2); }
   }
   return (sqrt(s / (m*n)));
}

/*=========================================================================
FUNKCJA: stdev_x - oblicza wektor odchylen standardowych z macierzy
PARAMETRY:
   mS - macierz danych
   vM - obliczony wektor odchylen standardowych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: stdev_x(mS, vM, m, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::stdev_x(array &mS, array &vM, int m, int n)
{
   int i, j;
   double s;
   array vE(m);                    // wektor srednich
   mean_x(mS, vE, m, n);
   for (j = 0; j < m; j++)
   {
      s = 0.0;
      for (i = 0; i < n; i++)
      {
         s = s + pow(*(mS.b+loc2(i,j,m)) - *(vE.b+j), 2);
      }
      *(vM.b+j) = sqrt(s / n);
   }
   vE.~array();
```

```
}

/*============================================================================
FUNKCJA: sum_v - oblicza sume elementow wektora
PARAMETRY:
   vS - wektor danych
   n - rozmiar wektora
WYWOLANIE: sum_v(vS, n);
REZULTAT: suma typu double
----------------------------------------------------------------------------*/
double mva::sum_v(array &vS, int n)
{
   int i;
   double s = 0.0;
   for (i = 0; i < n; i++) { s = s + *(vS.b+i); }
   return (s);
}

/*============================================================================
FUNKCJA: sum_m - oblicza sume elementow macierzy mS
PARAMETRY:
   mS - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: sum_m(mS, m, n);
REZULTAT: wartosc srednia typu double
----------------------------------------------------------------------------*/
double mva::sum_m(array &mS, int m, int n)
{
   int i, j;
   double s = 0.0;
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++) { s = s + *(mS.b+loc2(i,j,m)); }
   }
   return (s);
}

/*============================================================================
FUNKCJA: sum_x - oblicza wektor sum z macierzy mS (sumy kolumn)
PARAMETRY:
   mS - macierz danych
   vM - obliczony wektor sum
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: sum_x(mS, vM, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::sum_x(array &mS, array &vM, int m, int n)
{
   int i, j;
   double s;
   for (j = 0; j < m; j++)
   {
      s = 0.0;
      for (i = 0; i < n; i++) { s = s + *(mS.b+loc2(i,j,m)); }
      *(vM.b+j) = s;
   }
}

/*============================================================================
FUNKCJA: med_x - oblicza wektor median z macierzy (mediany kolumn)
PARAMETRY:
   mD - dana macierz
   vM - obliczony wektor median
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: med_x(mD, vM, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::med_x(array &mD, array &vM, int m, int n)
{
   int i, j;
   array vT(n);                // kopia j-tej kolumny z macierzy mD
   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++) { *(vT.b+i) = *(mD.b+m*i+j); }
      qsort_up(vT, 0, n-1);
      if (odd(n)) *(vM.b+j) = *(vT.b+(n/2));
```

```
         else *(vM.b+j) = (*(vT.b+((n-1)/2)) + *(vT.b+(n/2))) / 2.0;
   }
   vT.~array();
}

/*============================================================================
FUNKCJA: asym_coef - oblicza wektor wspolczynnikow asymetrii z macierzy
PARAMETRY:
   mD - macierz danych
   vA - obliczony wektor wspolczynnikow asymetrii
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: asym_coef(mD, mA, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::asym_coef(array &mD, array &vA, int m, int n)
{
   int i, j;
   double s, s1, s2;
   array vE(m);                        // wektor srednich
   array vS(m);                        // wektor odchylen standardowyvh
   mean_x(mD, vE, m, n);
   stdev_x(mD, vS, m, n);
   for (j = 0; j < m; j++)
   {
      s = 0.0;
      for (i = 0; i < n; i++)
      {
         s = s + pow(*(mD.b+m*i+j) - *(vE.b+j), 3);
      }
      s1 = s / n;
      s2 = pow(*(vS.b+j), 3);
      *(vA.b+j) = s1 / (s2 + fabs(s1));
   }
   vE.~array();
   vS.~array();
}

/*============================================================================
FUNKCJA: conc_coef - oblicza wektor wspolczynnikow koncentracji z macierzy
PARAMETRY:
   mD - macierz danych
   vC - obliczony wektor wspolczynnikow koncentracji
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: conc_coef(mD, mC, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::conc_coef(array &mD, array &vC, int m, int n)
{
   int i, j;
   double s, s1, s2;
   array vE(m);                        // wektor srednich
   array vS(m);                        // wektor odchylen standardowyvh
   mean_x(mD, vE, m, n);
   stdev_x(mD, vS, m, n);
   for (j = 0; j < m; j++)
   {
      s = 0.0;
      for (i = 0; i < n; i++)
      {
         s = s + pow(*(mD.b+m*i+j) - *(vE.b+j), 4);
      }
      s1 = (s / n) - pow(*(vS.b+j), 4);
      s2 = s / n;
      *(vC.b+j) = sqrt(s1 / s2);
   }
   vE.~array();
   vS.~array();
}

/*============================================================================
FUNKCJA: var_coef - oblicza wspolczynniki zmiennosci kolumn macierzy oraz
                    sume tych wpolczynnikow
PARAMETRY:
   mD - dana macierz
   vV - obliczony wektor wspolczynnikow zmiennosci
   sc - obliczona suma wspolczynnikow zmiennosci
   m - liczba kolumn
```

```
   n - liczba wierszy
WYWOLANIE: var_coef(mD, vV, &sc, m, n);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::var_coef(array &mD, array &vV, double *sc, int m, int n)
{
   int j;
   array vE(m);                        // wektor srednich
   array vS(m);                        // wektor odchylen standardowyvh
   mean_x(mD, vE, m, n);
   stdev_x(mD, vS, m, n);
   *sc = 0.0;
   for (j = 0; j < m; j++)
   {
      *(vV.b+j) = *(vS.b+j) / *(vE.b+j);
      *sc = *sc + (*(vS.b+j) / *(vE.b+j));
   }
   vE.~array();
   vS.~array();
}

/*============================================================================
FUNKCJA: corr - oblicza macierz wspolczynnikow korelacji liniowej Pearsona
PARAMETRY:
   mD - macierz danych
   mC - obliczona macierz korelacji
   m - liczba kolumn (cech)
   n - liczba wierszy (obserwacji)
WYWOLANIE: corr(mD, mC, m, n);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::corr(array &mD, array &mC, int m, int n)
{
   int i, j, k;
   double c, c1, c2;
   array vE(m);                        // wektor srednich
   mean_x(mD, vE, m, n);
   for (j = 0; j < m; j++)
   {
      for (k = 0; k <= j; k++)
      {
         c = 0.0; c1 = 0.0;   c2 = 0.0;
         for (i = 0; i < n; i++)
         {
            c = c + (*(mD.b+m*i+j) - *(vE.b+j)) * (*(mD.b+m*i+k) - *(vE.b+k));
            c1 = c1 + pow(*(mD.b+m*i+j) - *(vE.b+j), 2);
            c2 = c2 + pow(*(mD.b+m*i+k) - *(vE.b+k), 2);
         }
         *(mC.b+m*k+j) = c / sqrt(c1 * c2);
         *(mC.b+m*j+k) = c / sqrt(c1 * c2);
      }
   }
   vE.~array();
}

/*============================================================================
FUNKCJA: corr_xq - oblicza wspolczynniki korelacji liniowej pomiedzy zmienna
                   syntetyczna oraz j-ta zmienna diagnostyczna
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   vC - obliczony wektor wspolczynnikow korelacji
   m - liczba kolumn (cech)
   n - liczba wierszy (obserwacji)
WYWOLANIE: corr_xq(mD, vQ, vC, m, n);
REZULTAT: void
------------------------------------------------------------------------------*/
void mva::corr_xq(array &mD, array &vQ, array &vC, int m, int n)
{
   int i, j, k;
   double c, c1, c2, eq;
   array vE(m);
   eq = mean_v(vQ, n);          // srednia arytmetyczna realizacji zmiennej Q
   mean_x(mD, vE, m, n);        // wektor srednich zmiennych diagnostycznych
   for (j = 0; j < m; j++)
   {
      c = 0.0; c1 = 0.0;   c2 = 0.0;
      for (i = 0; i < n; i++)
      {
```

```
         c = c + (*(mD.b+m*i+j) - *(vE.b+j)) * (*(vQ.b+i) - eq);
         c1 = c1 + pow(*(mD.b+m*i+j) - *(vE.b+j), 2);
         c2 = c2 + pow(*(vQ.b+i) - eq, 2);
      }
      *(vC.b+j) = c / sqrt(c1 * c2);
   }
   vE.~array();
}

/*============================================================================
FUNKCJA: spearman - oblicza wspolczynniki korelacji rang Spearmana pomiedzy
                    zmienna syntetyczna oraz j-ta zmienna diagnostyczna
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   vS - obliczony wektor wspolczynnikow korelacji rang Spearmana
   m - liczba kolumn (cech)
   n - liczba wierszy (obserwacji)
WYWOLANIE: spearman(mD, vQ, vS, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::spearman(array &mD, array &vQ, array &vS, int m, int n)
{
   int i, j, k;
   double s;
   int *aRq = new int [ n ];          // rangi zmiennej syntetycznej
   int *aRx = new int [ n ];          // rangi j-tej zmiennej diagnostycznej
   array vVx(n);                      // kopia j-tej kolumny (cechy) z macierzy
                                      // obserwacji zmiennych diagnostycznych
   for (k = 0; k < n; k++) { *(aRq+k) = k+1; }
   qsort_dn(vQ, 0, n-1);
   for (j = 0; j < m; j++)
   {
      s = 0.0;
      for (k = 0; k < n; k++)
      {
         *(aRx+k) = *(aRq+k);
         *(vVx.b+k) = *(mD.b+m*k+j);
      }
      qsort_dn(vVx, aRx, 0, n-1);
      for (i = 0; i < n; i++) { s = s + pow(*(aRq+i) - *(aRx+i), 2); }
      *(vS.b+j) = 1 - ((6 * s) / (pow(n, 3) - n));
   }
   vVx.~array();
   delete aRq;
   delete aRx;
}

/*============================================================================
FUNKCJA: spearman - oblicza wspolczynnik korelacji rang Spearmana X z Y
PARAMETRY:
   vX - wektor X
   vY - wektor Y
   n - rozmiar wektorow
WYWOLANIE: spearman(vX, vY, n);
REZULTAT: void
----------------------------------------------------------------------------*/
double mva::spearman(array &vX, array &vY, int n)
{
   int i;
   double s;
   for (i = 0; i < n; i++) { s = s + pow(*(vX.b+i) - *(vY.b+i), 2); }
   return (1 - ((6 * s) / (pow(n, 3) - n)));
}

/*============================================================================
FUNKCJA: kendall - oblicza wspolczynniki korelacji rang Kendalla (tau)
                   pomiedzy wektorami uporzadkowan
PARAMETRY:
   vX - wektor 1
   vY - wektor 2
   n - liczba elementow
WYWOLANIE: kendall(vX, vY, n);
REZULTAT: double K
----------------------------------------------------------------------------*/
double mva::kendall(array &vX, array &vY, int n)
{
   int i, j, p = 0, q = 0;
   for (i = 0; i < n-1; i++)
```

```
   {
      for (j = i+1; j < n; j++)
      {
         if (((*(vX.b+i) - *(vX.b+j)) * (*(vY.b+i) - *(vY.b+j))) > 0)
            p++;
         else
            q--;
      }
   }
   return ((2.0 * (p + q)) / (n * (n - 1)));
}

/*============================================================================
FUNKCJA: varcov - oblicza macierz wariancji-kowariancji
PARAMETRY:
   mD - macierz danych
   mV - macierz wariancji-kowariancji
   m - liczba kolumn (cech)
   n - liczba wierszy (obserwacji)
WYWOLANIE: varcov(mD, mV, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::varcov(array &mD, array &mV, int m, int n)
{
   int i, j, k;
   double c;
   array vE(m);       // wektor srednich zmiennych diagnostycznych
   mean_x(mD, vE, m, n);
   for (j = 0; j < m; j++)
   {
      for (k = 0; k <= j; k++)
      {
         c = 0.0;
         for (i = 0; i < n; i++)
         {
            c = c + (*(mD.b+m*i+j) - *(vE.b+j)) * (*(mD.b+m*i+k) - *(vE.b+k));
         }
         *(mV.b+m*k+j) = c / (n-1);
         *(mV.b+m*j+k) = c / (n-1);
      }
   }
   vE.~array();
}

/*============================================================================
FUNKCJA: gs_qr - dekompozycja macierzy A na czynniki Q i R
                 metoda Grama-Schmidta
PARAMETRY:
   mA - macierz danych
   mQ - macierz kolumnami ortogonalna
   mR - gorna macierz trojkatna stopnia
   n - liczba wierszy
   m - liczba kolumn
WYWOLANIE: gs_qr(mA, mQ, mR, m, n);
REZULTAT: 0 || 1 || 2
----------------------------------------------------------------------------*/
int mva::gs_qr(array &mA, array &mQ, array &mR, int m, int n)
{
   int i, j, k, st;
   double ap, s;
   array vP(m);

   if ((m < 1) || (n < m)) { st = 1; return st; }

   for (j = 0; j < m; j++)
   {
      for (i = 0; i < n; i++) { *(mQ.b+m*j+i) = 0; }
      for (i = 0; i < m; i++) { *(mR.b+m*j+i) = 0; }
   }
   s = 0.0;
   for (j = 0; j < n; j++) { ap = *(mA.b+m*j+0); s = s + ap * ap; }
   s = sqrt(s);
   if (s == 0) { st = 2; return st; }
   st = 0;
   *(mR.b+0+0) = s;
   for (j = 0; j < n; j++) { *(mQ.b+m*j+0) = *(mA.b+m*j+0) / s; }
   for (i = 1; i < m; i++)
   {
      for (k = 0; k < i; k++)
```

```
      {
         s = 0;
         for (j = 0; j < n; j++) { s = s + *(mQ.b+m*j+k) * *(mA.b+m*j+i); }
         *(mR.b+m*k+i) = s;
      }

      for (j = 0; j < n; j++)
      {
         s = 0;
         for (k = 0; k < i; k++) { s = s + *(mQ.b+m*j+k) * *(mR.b+m*k+i); }
         *(vP.b+j) = *(mA.b+m*j+i) - s;
      }

      for (k = 0; k < i; k++)
      {
         s = 0;
         for (j = 0; j < n; j++) { s = s + *(mQ.b+m*j+k) * *(vP.b+j); }
         for (j = 0; j < n; j++) { *(vP.b+j) = *(vP.b+j) - *(mQ.b+m*j+k) * s; }
         *(mR.b+m*k+i) = *(mR.b+m*k+i) + s;
      }
      s = 0;
      for (j = 0; j < n; j++) { s = s + *(vP.b+j) * *(vP.b+j); }
      s = sqrt(s);
      if (s == 0) for (j = 0; j < n; j++) *(mQ.b+m*j+i) = 0;
         else for (j = 0; j < n; j++) *(mQ.b+m*j+i) = *(vP.b+j) / s;
      for (j = 0; j < i; j++) *(mR.b+m*i+j) = 0;
      *(mR.b+m*i+i) = s;
   }
   vP.~array();
   return (st);
}

/*============================================================================
FUNKCJA: rayleigh - wyznacza wartosc wlasna i wektor wlasny macierzy
                    symetrycznej metoda Reylaigh'a
PARAMETRY:
   mS - macierz/wektor danych
   n - stopien macierzy
   vX - wyznaczony wektor wlasny
   mx - maksymalna liczba iteracji
   dok - dokladnosc obliczen
   vL - wektor obliczanych wartosci wlasnych
WYWOLANIE: rayleigh(mS, n, vX, mx, dok, vL);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::rayleigh(array &mS, int n, array &vX, const int mx,
                   const double dok, array &vL, int r)
{
   int i, j, k, it;
   double b, la, ep;
   array vY(n);
   it = 0;
   *(vL.b+r) = 1.0e+12;
   do
   {
      la = 0.0;
      for (i = 0; i < n; i++)
      {
         *(vY.b+i) = 0.0;
         for (j = 0; j < n; j++)
         {
            if (i < j) k = loc(j,i); else k = loc(i,j);
            *(vY.b+i) = *(vY.b+i) + *(mS.b+k) * *(vX.b+j);
         }
         la = la + *(vY.b+i) * *(vY.b+i);
      }
      la = sqrt(la);
      for (i = 0; i < n; i++) { *(vY.b+i) = *(vY.b+i) / la; }
      b = la;
      la = 0.0;
      ep = 0.0;
      for (i = 0; i < n; i++)
      {
         la = la + *(vY.b+i) * *(vX.b+i);
         ep = ep + *(vX.b+i) * *(vX.b+i);
      }
      la = b * la / ep;
      for (i = 0; i < n; i++) { *(vX.b+i) = *(vY.b+i); }
      ep = fabs(*(vL.b+r) - la);
```

```
      *(vL.b+r) = la;
      it++;
   }
   while ((it < mx) || (ep > dok));
   vY.~array();
}

/*============================================================================
FUNKCJA: hotelling - wyznacza wszystkie wartosci wlasne oraz wektory wlasne
                     macierzy symetrycznej metoda Hotellinga
PARAMETRY:
   mS - macierz danych
   n - stopien macierzy
   k - liczba wyznacznych wartosci i wektorow wlasnych
   mB - macierz zawierajaca wyznaczone wektory wlasne
   vL - wektor obliczonych wartosci wlasnych
WYWOLANIE: hotelling(mS, n, k, mB, vL);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::hotelling(array &mS, int n, int k, array &mB, array &vL)
{
   const int mx = 40;
   const double dok = 1.0e-9;
   int i, j, r, m = (n*(n+1)) / 2;
   array vW(m);
   array vX(n);
   for (i = 0; i < n; i++) { *(vX.b+i) = 1.0;    }
   rayleigh(mS, n, vX, mx, dok, vL, 0);
   for (i = 0; i < n; i++) { *(mB.b+n*0+i) = *(vX.b+i); }
   for (i = 0; i < m; i++) { *(vW.b+i) = *(mS.b+i); }
   for (r = 1; r < k; r++)
   {
      for (i = 0; i < n; i++)
      {
         for (j = 0; j <= i; j++)
         {
            *(vW.b+loc(i,j)) = *(vW.b+loc(i,j)) - *(vL.b+r-1) * *(vX.b+i) *
                               *(vX.b+j);
         }
      }
      rayleigh(vW, n, vX, mx, dok, vL, r);
      for (i = 0; i < n; i++) { *(mB.b+n*r+i) = *(vX.b+i); }
   }
   vW.~array();
   vX.~array();
}

/*============================================================================
FUNKCJA: loadings - oblicza ladunki czynnikowe pierwszego czynnika glownego
PARAMETRY:
   mT - macierz wektorow wlasnych
   vL - wektor wartosci wlasnych
   vX - wektor ladunkow czynnikowych pierwszego czynnika glownego
   m - liczba cech
WYWOLANIE: loadings(mT, vL, vX, m);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::loadings(array &mT, array &vL, array &vX, int m)
{
   double p, w;
/* int j, k;
   for (k = 0; k < m; k++)
   {
      p = sqrt(fabs(*(vL.b+k)));
      for (j = 0; j < m; j++)
      {
         w = p * (*(mT.b+m*k+j));
         *(vX.b+m*k+j) = w;
      }
   }  */
   int j;
   p = sqrt(*(vL.b+0));
   for (j = 0; j < m; j++)
   {
      w = p * (*(mT.b+m*j));
      *(vX.b+j) = w;
   }
}
```

```
/*============================================================================
FUNKCJA: loadings - oblicza ladunki czynnikowe wszystkich czynnikow
PARAMETRY:
   mT - transponowana macierz wektorow wlasnych
   vL - wektor wartosci wlasnych
   mW - macierz obliczonych ladunkow czynnikowych
   m - liczba cech (dlugosc wektora L)
   mX - macierz zawierajaca procent wyjasnianej wariancji przez poszczegolne
        czynniki glowne oraz procent skumulowany i ladunki czynnikowe
        pierwszego czynnika glownego
WYWOLANIE: loadings( mT, vL, mW, m, mX );
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::loadings( array &mT, array &vL, array &mW, int m, array &mX )
{
   array vP( m );
   double s, x, q, p, w;
   int j, k;
   for (k = 0; k < m; k++)
   {
      s = 0;
      for (j = 0; j < m; j++) { s = s + *(mT.b+m*j+k) * *(mT.b+m*j+k); }
      *(vP.b+k) = s;
   }
   s = 0;
   for (k = 0; k < m; k++)
   {
      s = s + *(vL.b+k);
      for (j = 0; j < m; j++)
      {
         *(mW.b+m*j+k) = sqrt( fabs(*(vL.b+k)) ) * ( *(mT.b+m*j+k) ) /
                         sqrt( *(vP.b+k) );
      }
   }
   q = 0;
   for (k = 0; k < m; k++)
   {
      x = ( *(vL.b+k) / s ) * 100.0;
      q = q + x;
      *(mX.b+m*0+k) = x;
      *(mX.b+m*1+k) = q;
   }
   p = sqrt( fabs(*(vL.b+0)) );
   for (j = 0; j < m; j++)
   {
      w = p * ( *(mT.b+m*j) );
      *(mX.b+m*2+j) = w;
   }
   vP.~array();
}

/*============================================================================
FUNKCJA: characters - identyfikacja cech
PARAMETRY:
   vX - I: wektor ladunkow czynnikowych pierwszego czynnika glownego,
        O: wektor identyfikatorow cech 1-stymulanta, 0-destymulanta
   m - liczba cech
WYWOLANIE: characters(vX, m);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::characters(array &vX, int m)
{
   int j;
   for (j = 0; j < m; j++)
   {
      if (*(vX.b+j) < 0) *(vX.b+j) = 0; else *(vX.b+j) = 1;
   }
}

/*============================================================================
FUNKCJA: dest2stym - przeksztalca cechy-destymulanty na cechy-stymulanty
PARAMETRY:
   mDAT - macierz danych
   vX - wektor zawierajacy identyfikatory cech (1-stymulanta, 0-destymulanta)
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: dest2stym(mDAT, vX, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
```

```
void mva::dest2stym(array &mDAT, array &vX, int m, int n)
{
   int i, j;
   for (j = 0; j < m; j++)
   {
      if (*(vX.b+j) == 0)
      {
         for (i = 0; i < n; i++) { *(mDAT.b+m*i+j) = *(mDAT.b+m*i+j) * (-1); }
      }
   }
}

/*============================================================================
FUNKCJA: neg2pos - realizuje postulat dodatniosci znormalizowanej macierzy
PARAMETRY:
   mDAT - macierz danych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: neg2pos(mDAT, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::neg2pos(array &mDAT, int m, int n)
{
   int i, j;
   double s, xmin, std;
   xmin = min_m(mDAT, m, n);
   std = stdev_m(mDAT, m, n);
   if (xmin <= 0)
   {
      s = (-xmin + (std/5));        // element skalujacy
      for (j = 0; j < m; j++)
      {
         for (i = 0; i < n; i++) { *(mDAT.b+m*i+j) = (*(mDAT.b+m*i+j) + s); }
      }
   }
}

/*============================================================================
FUNKCJA: upperpole - wyznacza gorny biegun macierzy danych znormalizowanych
PARAMETRY:
   mS - macierz zawierajaca obserwacje zmiennych
   vU - wektor zawierajacy elementy wyznaczajace gorny biegun
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: upperpole(mS, vU, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::upperpole(array &mS, array &vU, int m, int n)
{
   int i, j;
   double xmax;
   for (j = 0; j < m; j++)
   {
      xmax = *(mS.b+j);
      for (i = 0; i < n; i++)
      {
         if (*(mS.b+m*i+j) > xmax) xmax = *(mS.b+m*i+j);
      }
      *(vU.b+j) = xmax;
   }
}

/*============================================================================
FUNKCJA: lowerpole - wyznacza dolny biegun macierzy danych znormalizowanych
PARAMETRY:
   mS - macierz zawierajaca obserwacje zmiennych
   vL - wektor zawierajacy elementy wyznaczajace dolny biegun
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: lowerpole(mS, vL, m, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::lowerpole(array &mS, array &vL, int m, int n)
{
   int i, j;
   double xmin;
   for (j = 0; j < m; j++)
   {
      xmin = *(mS.b+j);
```

```
      for (i = 0; i < n; i++)
      {
         if (*(mS.b+m*i+j) < xmin) xmin = *(mS.b+m*i+j);
      }
      *(vL.b+j) = xmin;
   }
}

/*============================================================================
FUNKCJA: det_eigen - oblicza wyznacznik macierzy korelacji na podstawie
                     wektora wartosci wlasnych tej macierzy
PARAMETRY:
   vL - wektor wartosci wlasnych
   n - rozmiar wektora vL
WYWOLANIE: det_eigen(vL, n);
REZULTAT: wyznacznik
----------------------------------------------------------------------------*/
double mva::det_eigen(array &vL, int n)
{
   int i;
   double det = 1.0;
   for (i = 0; i < n; i++)
   {
      det = det * *(vL.b+i);
   }
   return (det);
}

/*============================================================================
FUNKCJA: gauge01 - miernik zgodnosci odwzorowania
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: gauge01(mD, vQ, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge01(array &mD, array &vQ, int m, int n)
{
   int k, r = (n*(n-1)) / 2;
   double s1 = 0.0, s2 = 0.0;
   array vDZ(r);            // odleglosc w R cech diagnostycznych (trojkat)
   array vDQ(r);            // odleglosc w R cechy syntetycznej (trojkat)
   dist_m01(mD, vDZ, m, n);
   dist_v01(vQ, vDQ, n);
   for (k = 0; k < r; k++)
   {
      s1 = s1 + pow(*(vDQ.b+k) - *(vDZ.b+k), 2);
      s2 = s2 + pow(*(vDZ.b+k), 2);
   }
   vDZ.~array();
   vDQ.~array();
   return (s1 / s2);
}

/*============================================================================
FUNKCJA: gauge02 - miernik zgodnosci odwzorowania
PARAMETRY:
   aD - macierz obserwacji zmiennych diagnostycznych
   aQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: gauge02(mD, vQ, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge02(array &mD, array &vQ, int m, int n)
{
   int k, r = (n*(n-1)) / 2;
   double s = 0.0;
   array vDZ(r);            // odleglosc w R cech diagnostycznych
   array vDQ(r);            // odleglosc w R cechy syntetycznej
   dist_m01(mD, vDZ, m, n);
   dist_v01(vQ, vDQ, n);
   for (k = 0; k < r; k++)
   {
      s = s + pow((*(vDQ.b+k) - *(vDZ.b+k)) / *(vDZ.b+k), 2);
   }
   vDZ.~array();
   vDQ.~array();
   return ((2 * s) / (n * (n - 1)));
```

```
}

/*============================================================================
FUNKCJA: gauge03 - miernik zgodnosci odwzorowania
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: gauge03(mD, vQ, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge03(array &mD, array &vQ, int m, int n)
{
   int k, r = (n*(n-1)) / 2;
   double s1 = 0.0, s2 = 0.0;
   array vDZ(r);            // odleglosc w R cech diagnostycznych
   array vDQ(r);            // odleglosc w R cechy syntetycznej
   dist_m01(mD, vDZ, m, n);
   dist_v01(vQ, vDQ, n);
   for (k = 0; k < r; k++)
   {
      s1 = s1 + pow(*(vDQ.b+k) - *(vDZ.b+k), 2) / *(vDZ.b+k);
      s2 = s2 + *(vDZ.b+k);
   }
   vDZ.~array();
   vDQ.~array();
   return (s1 / s2);
}

/*============================================================================
FUNKCJA: gauge04 - miernik oparty na wspolczynniku korelacji liniowej Pearsona
                   miedzy zmienna syntetyczna oraz zmiennymi diagnostycznymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge04(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge04(array &mD, array &vQ, int m, int n)
{
   int j;
   double s = 0.0;
   array vC(m);                  // wektor wspolczynnikow korelacji
   corr_xq(mD, vQ, vC, m, n);
   for (j = 0; j < m; j++) { s = s + *(vC.b+j); }
   vC.~array();
   return (1 - (s/m));
}

/*============================================================================
FUNKCJA: gauge05 - miernik oparty na wspolczynniku korelacji liniowej Pearsona
                   miedzy zmienna syntetyczna oraz zmiennymi diagnostycznymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge05(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge05(array &mD, array &vQ, int m, int n)
{
   int j, l;
   double s = 0.0;
   array vC(m);                  // wektor wspolczynnikow korelacji
   corr_xq(mD, vQ, vC, m, n);
   for (j = 0; j < m; j++)
   {
      if (*(vC.b+j) >= 0.5) l = 0;
      if (*(vC.b+j) >=  0.0 && *(vC.b+j) < 0.5) l = 1;
      if (*(vC.b+j) >= -0.5 && *(vC.b+j) < 0.0) l = 2;
      if (*(vC.b+j) < -0.5) l = 3;
      s = s + l;
   }
   vC.~array();
   return (s/m);
}
```

```
/*============================================================================
FUNKCJA: gauge06 - miernik oparty na wspolczynniku korelacji rang Spearmana
                   miedzy zmienna syntetyczna oraz zmiennymi diagnostycznymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge06(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge06(array &mD, array &vQ, int m, int n)
{
   int j;
   double s = 0.0;
   array vS(m);                        // wektor wspolczynnikow korelacji rang
   spearman(mD, vQ, vS, m, n);
   for (j = 0; j < m; j++) { s = s + *(vS.b+j); }
   vS.~array();
   return (1 - (s/m));
}

/*============================================================================
FUNKCJA: gauge07 - miernik oparty na wspolczynniku korelacji rang Spearmana
                   miedzy zmienna syntetyczna oraz zmiennymi diagnostycznymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge07(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge07(array &mD, array &vQ, int m, int n)
{
   int j, l;
   double s = 0.0;
   array vS(m);                        // wektor wspolczynnikow korelacji rang
   spearman(mD, vQ, vS, m, n);
   for (j = 0; j < m; j++)
   {
      if (*(vS.b+j) >= 0.5) l = 0;
      if (*(vS.b+j) >= 0.0 && *(vS.b+j) < 0.5) l = 1;
      if (*(vS.b+j) >= -0.5 && *(vS.b+j) < 0.0) l = 2;
      if (*(vS.b+j) < -0.5) l = 3;
      s = s + l;
   }
   vS.~array();
   return (s/m);
}

/*============================================================================
FUNKCJA: gauge08 - miernik oparty na wspolczynniku korelacji rang Spearmana
                   miedzy zmienna syntetyczna oraz zmiennymi diagnostycznymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge08(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge08(array &mD, array &vQ, int m, int n)
{
   int i, j, k, l;
   double s = 0.0, x = 0.0;
   int *aRq = new int [n];        // rangi obiektow wg zmiennej syntetycznej
   int *aTq = new int [n];        // kolejnosc wyjsciowa obiektow dla Q
   int *aRx = new int [n];        // rangi obiektow wg j-tej cechy diagnostycznej
   int *aTx = new int [n];        // kolejnosc wyjsciowa obiektow dla X
   int *aMx = new int [m*n];      // macierz rang obiektow wg cech diagnostycznych
   array vVx(n);                  // kopia j-tej cechy z macierzy obserwacji
   array vTMP(n);
   for (k = 0; k < n; k++) { *(aTq+k) = k+1; }
   for (i = 0; i < n; i++) { *(vTMP.b+i) = *(vQ.b+i); }
   qsort_dn(vTMP, aTq, 0, n-1);
   for (k = 0; k < n; k++) { *(aRq + *(aTq+k) -1) = k+1; }
   for (j = 0; j < m; j++)
```

```
   {
      for (k = 0; k < n; k++)
      {
         *(aTx+k) = k+1;
         *(vVx.b+k) = *(mD.b+m*k+j);
      }
      qsort_dn(vVx, aTx, 0, n-1);
      for (k = 0; k < n; k++) { *(aRx + *(aTx+k) -1) = k+1; }
      for (k = 0; k < n; k++) { *(aMx+m*k+j) = *(aRx+k); }
   }
   s = 0.0;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { s = s + fabs(*(aMx+m*j+i) - *(aRq+i));  }
   }
   if (odd(n)) l = pow(n, 2) - 1; else l = pow(n, 2);
   vVx.~array();
   vTMP.~array();
   delete aRq;
   delete aTq;
   delete aRx;
   delete aTx;
   delete aMx;
   x = (double) m * l;
   return ((2*s) / x);
}

/*============================================================================
FUNKCJA: gauge09 - miernik zmiennosci zmiennej syntetycznej
PARAMETRY:
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: gauge09(vQ, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge09(array &vQ, int n)
{
   double ex, std;
   ex = mean_v(vQ, n);
   std = stdev_v(vQ, n);
   return (-(std/ex));
}

/*============================================================================
FUNKCJA: gauge10 - miernik zmiennosci zmiennej syntetycznej
PARAMETRY:
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: gauge10(vQ, n);
REZULTAT: wartosc miernika typu double
----------------------------------------------------------------------------*/
double mva::gauge10(array &vQ, int n)
{
   int i;
   double ex, std, s=0;
   array vP(n-1);    // wektor pierwszych roznic w realizacjach zmiennej Q
   array vTMP(n);
   for (i = 0; i < n; i++) { *(vTMP.b+i) = *(vQ.b+i); }
   qsort_up(vTMP, 0, n-1);
   for (i = 1; i < n; i++) { *(vP.b+(i-1)) = *(vTMP.b+i) - *(vTMP.b+(i-1)); }
   ex = mean_v(vP, n-1);
   std = stdev_v(vP, n-1);
   for (i = 0; i < n-1; i++) { s = s + *(vP.b+i); }
   vP.~array();
   vTMP.~array();
   return (std/ex);
}

/*============================================================================
FUNKCJA: gauge11 - miernik oparty na odleglosci taksonomicznej miedzy
                   standaryzowana cecha syntetyczna oraz standaryzowanymi
                   cechami diagnostycznymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge11(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
```

```
-------------------------------------------------------------------------*/
double mva::gauge11(array &mD, array &vQ, int m, int n)
{
   int i, j;
   double s = 0.0;
   array mN(m, n);       // macierz obserwacji standaryzowanych
   array vM(n);          // wektor standaryzowanych realizacji zmiennej Q
   norm05 nrm_d;
   nrm_d.normal(mD, mN, m, n);
   normv02(vQ, vM, n);      // standaryzacja wektora vQ
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { s = s + fabs(*(mN.b+m*j+i) - *(vM.b+i)); }
   }
   mN.~array();
   vM.~array();
   return (s / (n*m));
}

/*=========================================================================
FUNKCJA: gauge12 - miernik oparty na odleglosci taksonomicznej miedzy
                   standaryzowana zmienna syntetyczna oraz standaryzowanymi
                   zmiennymi pierwotnymi
PARAMETRY:
   mD - macierz obserwacji zmiennych diagnostycznych
   vQ - wektor realizacji zmiennej syntetycznej
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: gauge12(mD, vQ, m, n);
REZULTAT: wartosc miernika typu double
-------------------------------------------------------------------------*/
double mva::gauge12(array &mD, array &vQ, int m, int n)
{
   int i, j;
   double s = 0.0;
   array mN(m, n);       // macierz obserwacji standaryzowanych
   array vM(n);          // wektor standaryzowanych realizacji zmiennej Q
   norm05 nrm_d;
   nrm_d.normal(mD, mN, m, n);
   normv02(vQ, vM, n);      // standaryzacja wektora vQ
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++) { s = s + pow((*(mN.b+m*j+i) - *(vM.b+i)), 2); }
   }
   mN.~array();
   vM.~array();
   return (sqrt(s / (m*n)));
}

/*=========================================================================
FUNKCJA: dist_m01 - ogleglosc Euklidesa
PARAMETRY:
   mS - macierz danych znormalizowanych
   mD - macierz odleglosci
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: dist_m01(mS, mD, m, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::dist_m01(array &mS, array &mD, int m, int n)
{
   int i, j, k, r = 0;
   double s = 0.0;
   for (i = 0; i < n-1; i++)
   {
      for (k = i+1; k < n; k++)
      {
         s = 0.0;
         for (j = 0; j < m; j++)
         {
            s = s +  pow(*(mS.b+m*i+j) - *(mS.b+m*k+j), 2);
         }
         *(mD.b+r++) = sqrt(s);           // dolny trojkat
      }
   }
}

/*=========================================================================
FUNKCJA: dist_v01 - ogleglosc Euklidesa miedzy elementami wektora
```

```
PARAMETRY:
   mS - macierz danych znormalizowanych
   vD - wektor obliczonych odleglosci
   m - liczba cech
   n - liczba obserwacji
WYWOLANIE: dist_v01(mS, vD, m, n);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::dist_v01(array &mS, array &vD, int n)
{
   int i, k, r = 0;
   for (i = 0; i < n-1; i++)
   {
      for (k = i+1; k < n; k++)
      {
         *(vD.b+r++) = sqrt(pow(*(mS.b+i) - *(mS.b+k), 2));
      }
   }
}

/*===============================================================================
FUNKCJA: normv01 - normalizacja elementow wektora metoda przeksztalcenia
                   ilorazowego
PARAMETRY:
   vD - wektor danych pierwotnych
   vN - obliczony wektor wartosci znormalizowanych
   n - rozmiar wektora
WYWOLANIE: normv01( vD, vN, n );
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::normv01( array &vD, array &vN, int n )
{
   int i;
   double xmax;
   xmax = max_v( vD, n );         // wartosc max w wektorze vD
   for(i = 0; i < n; i++) { *(vN.b+i) = *(vD.b+i) / xmax; }
}

/*===============================================================================
FUNKCJA: normav02 - normalizacja elementow wektora metoda standaryzacji
PARAMETRY:
   vD - wektor danych pierwotnych
   vN - obliczony wektor wartosci znormalizowanych
   n - rozmiar wektora
WYWOLANIE: normav02( vD, vN, n );
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::normv02( array &vD, array &vN, int n )
{
   int i;
   double ex, sx;
   ex = mean_v( vD, n );
   sx = stdev_v( vD, n );
   for(i = 0; i < n; i++) { *(vN.b+i) = (*(vD.b+i) - ex) / sx; }
}

/*===============================================================================
FUNKCJA: scale_q01 - skalowanie wektora zmiennej syntetycznej
PARAMETRY:
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: scale_q01(vQ, n);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::scale_q01(array &vQ, int n)
{
   int i;
   double xmin, std, ex, s;
   array vP(n);
   xmin = min_v(vQ, n);
   for (i = 0; i < n; i++) { *(vP.b+i) = *(vQ.b+i) - xmin; }
   ex = mean_v(vP, n);
   std = stdev_v(vP, n);
   s = ex + 2 * std;
   for (i = 0; i < n; i++) { *(vQ.b+i) = *(vP.b+i) / s; }
   vP.~array();
}

/*===============================================================================
```

```
FUNKCJA: scale_q02 - skalowanie wektora zmiennej syntetycznej
PARAMETRY:
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: scale_q02(vQ, n);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::scale_q02(array &vQ, int n)
{
   int i;
   double xmax;
   xmax = max_v(vQ, n);
   for (i = 0; i < n; i++) { *(vQ.b+i) = (*(vQ.b+i) / xmax); }
}

/*===============================================================================
FUNKCJA: scale_q03 - skalowanie wektora zmiennej syntetycznej
PARAMETRY:
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: scale_q03(vQ, n);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::scale_q03(array &vQ, int n)
{
   int i;
   double std, ex, s;
   ex = mean_v(vQ, n);
   std = stdev_v(vQ, n);
   s = ex + 2 * std;
   for (i = 0; i < n; i++) { *(vQ.b+i) = 1.0 - (*(vQ.b+i) / s); }
}

/*===============================================================================
FUNKCJA: scale_q04 - normalizacja cech metoda unitaryzacji (9)
PARAMETRY:
   vQ - wektor realizacji zmiennej syntetycznej
   n - liczba obserwacji
WYWOLANIE: scale_q04(vQ, n);
REZULTAT: void
-------------------------------------------------------------------------------*/
void mva::scale_q04( array &vQ, int n )
{
   int i, j;
   double xmax, xmin;
   xmax = max_v(vQ, n);
   xmin = min_v(vQ, n);
   for(i = 0; i < n; i++)
   {
       *(vQ.b+i) = (*(vQ.b+i) - xmin) / (xmax - xmin);
   }
}

/*===============================================================================
FUNKCJA: mlu_decomp - rozklada macierz A na czynniki trojkatne LU wg algorytmu
                      Gaussa-Banachiewicza z wyborem elementu glownego
                      (czynniki LU zapisywane w miejsce zrodlowej macierzy A)
PARAMETRY:
   mA - na wejsciu macierz zrodlowa A, na wyjsciu zawiera czynniki L i U
   vP - wektor przestawien wierszy w macierzy A
   n - stopien macierzy A
WYWOLANIE: mlu_decomp(mA, vP, n);
REZULTAT: liczba przestawien wierszy w macierzy A
-------------------------------------------------------------------------------*/
int mva::mlu_decomp(array &mA, array &vP, int n)
{
   int i, j, k, l, t, p = 0;
   double tmp = 0.0, s = 0.0;
   for (i = 0; i < n; i++)
   {
      tmp = 0.0;
      for (j = i; j < n; j++)
      {
         s = 0.0;
         for (k = 0; k < i; k++)
         {
            s = s + *(mA.b+j*n+k) * *(mA.b+k*n+i);
         }
         *(mA.b+j*n+i) = *(mA.b+j*n+i) - s;
```

```
         if (fabs(*(mA.b+j*n+i)) > tmp)
         {
            tmp = fabs(*(mA.b+j*n+i));
            t = j;
         }
      }
      if (tmp == 0.0) { exit(1); }      // macierz osobliwa
      *(vP.b+i) = t;
      if (i < t)
      {
         for (l = 0; l < n; l++)
         {
            tmp = *(mA.b+i*n+l);
            *(mA.b+i*n+l) = *(mA.b+t*n+l);
            *(mA.b+t*n+l) = tmp;
         }
         p++;
      }
      for (l = i+1; l < n; l++)
      {
         s = 0.0;
         for (k = 0; k < i; k++)
         {
            s = s + *(mA.b+i*n+k) * *(mA.b+k*n+l);
         }
         *(mA.b+i*n+l) = *(mA.b+i*n+l) - s;
      }
      for (j = i+1; j < n; j++)
      {
         *(mA.b+j*n+i) = *(mA.b+j*n+i) / *(mA.b+i*n+i);
      }
   }
   return (p);
}

/*============================================================================
FUNKCJA: lu_decomp - rozklada macierz A na czynniki trojkatne LU wg algorytmu
                     Gaussa-Banachiewicza bez wyboru elementu glownego
                     (czynniki LU zapisywane w miejsce zrodlowej macierzy A)
PARAMETRY:
   mA - na wejsciu macierz zrodlowa A, na wyjsciu zawiera czynniki L i U
   n - stopien macierzy A
WYWOLANIE: lu_decomp(mA, vP, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::lu_decomp(array &mA, int n)
{
   int i, j, k, l;
   double s = 0.0;
   for (i = 0; i < n; i++)
   {
      for (j = i; j < n; j++)
      {
         s = 0.0;
         for (k = 0; k < i; k++)
         {
            s = s + *(mA.b+i*n+k) * *(mA.b+k*n+j);
         }
         *(mA.b+i*n+j) = (*(mA.b+i*n+j) - s);
      }
      for (l = i+1; l < n; l++)
      {
         s = 0.0;
         for (k = 0; k < i; k++)
         {
            s = s + *(mA.b+l*n+k) * *(mA.b+k*n+i);
         }
         *(mA.b+l*n+i) = (*(mA.b+l*n+i) - s) / *(mA.b+i*n+i);;
      }
   }
}

/*============================================================================
FUNKCJA: mdet - oblicza wyznacznik macierzy A na podstawie rozkladu
                trojkatnego LU otrzymanego z wyborem elementu glownego
PARAMETRY:
   mA - macierz zrodlowa A
   n - stopien macierzy A
WYWOLANIE: mdet(mA, n);
```

```
REZULTAT: wyznacznik macierzy A
-------------------------------------------------------------------------*/
double mva::mdet(array &mA, int n)
{
   int i, j, p;
   double det = 1.0;
   array vP(n);
   p = mlu_decomp(mA, vP, n);
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < n; j++)
      {
         if (i == j)
         {
            det = det * *(mA.b+i*n+j);
         }
      }
   }
   vP.~array();
   return (odd(p) ? -det : det);
}

/*=========================================================================
FUNKCJA: det - oblicza wyznacznik macierzy A na podstawie rozkladu trojkatnego
               LU otrzymanego bez wyboru elementu glownego
PARAMETRY:
   mA - macierz zrodlowa A
   n - stopien macierzy A
WYWOLANIE: det(mA, n);
REZULTAT: wyznacznik macierzy A
-------------------------------------------------------------------------*/
double mva::det(array &mA, int n)
{
   int i, j;
   double det = 1.0;
   lu_decomp(mA, n);
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < n; j++)
      {
         if (i == j)
         {
            det = det * *(mA.b+i*n+j);
         }
      }
   }
   return (det);
}

/*=========================================================================
FUNKCJA: minverse - oblicza macierz odwrotna na podstawie rozkladu trojkatnego
                    LU wyznaczonego z wyborem elementu glownego
PARAMETRY:
   mA - macierz danych
   n - stopien macierzy
WYWOLANIE: minverse(mA, n);
REZULTAT: void
-------------------------------------------------------------------------*/
void mva::minverse(array &mA, int n)
{
   int i, j, k, t;
   double r, tmp;
   array vL(n), vP(n);
   mlu_decomp(mA, vP, n);
   for (k = 0; k < n; k++)
   {
      *(mA.b+k*n+k) = 1.0 / *(mA.b+k*n+k);
      for (i = 0; i < k; i++)
      {
         *(mA.b+i*n+k) = - *(mA.b+i*n+k) * *(mA.b+k*n+k);
      }
      for (j = k+1; j < n; j++)
      {
         r = *(mA.b+k*n+j);
         *(mA.b+k*n+j) = 0;
         for (i = 0; i <= k; i++)
         {
            *(mA.b+i*n+j) = *(mA.b+i*n+j) + *(mA.b+i*n+k) * r;
         }
```

```
      }
   }
   for (k = n-2; k >= 0; k--)
   {
      for (i = k+1; i < n; i++)
      {
         *(vL.b+i) = *(mA.b+i*n+k);
         *(mA.b+i*n+k) = 0;
      }
      for (j = k+1; j < n; j++)
      {
         for (i = 0; i < n; i++)
         {
            *(mA.b+i*n+k) = *(mA.b+i*n+k) - *(mA.b+i*n+j) * *(vL.b+j);
         }
      }
   }
   for (j = n-2; j >= 0; j--)
   {
      t = *(vP.b+j);
      if (j < t)
      {
         for (i = 0; i < n; i++)
         {
            tmp = *(mA.b+i*n+j);
            *(mA.b+i*n+j) = *(mA.b+i*n+t);
            *(mA.b+i*n+t) = tmp;
         }
      }
   }
   vP.~array();
   vL.~array();
}

/*============================================================================
FUNKCJA: inverse - oblicza macierz odwrotna na podstawie rozkladu trojkatnego
                   LU wyznaczonego bez wyboru elementu glownego
PARAMETRY:
   mA - macierz danych
   n - stopien macierzy
WYWOLANIE: inverse(mA, n);
REZULTAT: void
----------------------------------------------------------------------------*/
void mva::inverse(array &mA, int n)
{
   int i, j, k, t;
   double r, tmp;
   array vL(n);
   lu_decomp(mA, n);
   for (k = 0; k < n; k++)
   {
      *(mA.b+k*n+k) = 1.0 / *(mA.b+k*n+k);
      for (i = 0; i < k; i++)
      {
         *(mA.b+i*n+k) = - *(mA.b+i*n+k) * *(mA.b+k*n+k);
      }
      for (j = k+1; j < n; j++)
      {
         r = *(mA.b+k*n+j);
         *(mA.b+k*n+j) = 0;
         for (i = 0; i <= k; i++)
         {
            *(mA.b+i*n+j) = *(mA.b+i*n+j) + *(mA.b+i*n+k) * r;
         }
      }
   }
   for (k = n-2; k >= 0; k--)
   {
      for (i = k+1; i < n; i++)
      {
         *(vL.b+i) = *(mA.b+i*n+k);
         *(mA.b+i*n+k) = 0;
      }
      for (j = k+1; j < n; j++)
      {
         for (i = 0; i < n; i++)
         {
            *(mA.b+i*n+k) = *(mA.b+i*n+k) - *(mA.b+i*n+j) * *(vL.b+j);
         }
```

```
      }
   }
   vL.~array();
}

/* ------------------------------------------------------------------------
   PLIK: weig01.cpp
   ?: definicje funkcji klasy weig01
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: weig01 - konstruktor obiektow klasy weig01
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: weig01 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------*/
weig01::weig01( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*========================================================================
FUNKCJA: ~weig01 - destruktor obiektow klasy weig01
PARAMETRY: void
WYWOLANIE: ~weig01();
REZULTAT: void
------------------------------------------------------------------------*/
weig01::~weig01()
{

}

/*========================================================================
FUNKCJA: weight - wyznaczenie wag zmiennych (wagi stale, rownomierne)
PARAMETRY:
   aW - wektor zawierajacy wyznaczone wagi poszczegolnych cech
WYWOLANIE: weight(aW, m);
REZULTAT: void
------------------------------------------------------------------------*/
void weig01::weight( array &aW, int m )
{
   int j;
   for(j = 0; j < m; j++) { *(aW.b+j) = 1.0 / m; }
}

/* ------------------------------------------------------------------------
   PLIK: weig02.cpp
   ?: definicje funkcji klasy weig02
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: weig02 - konstruktor obiektow klasy weig02
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: weig02 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------*/
weig02::weig02( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*========================================================================
FUNKCJA: ~weig02 - destruktor obiektow klasy weig02
PARAMETRY: void
WYWOLANIE: ~weig02();
```

```
REZULTAT: void
------------------------------------------------------------------------------*/
weig02::~weig02()
{

}

/*==============================================================================
FUNKCJA: weight - wyznaczenie wag zmiennych na podstawie wspolczynnika
                  zmiennosci
PARAMETRY:
   mD - macierz danych
   aW - wektor wag poszczegolnych cech
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: weight( mD, vW, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void weig02::weight( array &mD, array &vW, int m, int n )
{
   int j;
   double sc;                              // suma wspolczynnikow zmiennosci
   array vC( m );                          // wektor wspolczynnikow zmiennosci
   var_coef( mD, vC, &sc, m, n );          // wspolczynnik zmiennosci
   for(j = 0; j < m; j++) { *(vW.b+j) = *(vC.b+j) / sc; }
   vC.~array();
}


/* ---------------------------------------------------------------------------
   PLIK: norm01.cpp
   ?: definicje funkcji klasy norm01
--------------------------------------------------------------------------- */

/*==============================================================================
FUNKCJA: norm01 - konstruktor obiektow klasy norm01
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm01 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------------*/
norm01::norm01( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*==============================================================================
FUNKCJA: ~norm01 - destruktor obiektow klasy norm01
PARAMETRY: void
WYWOLANIE: ~norm01();
REZULTAT: void
------------------------------------------------------------------------------*/
norm01::~norm01()
{

}

/*==============================================================================
FUNKCJA: normal - normalizacja cech metoda przeksztalcenia ilorazowego (2)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void norm01::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vM( m );                 // wektor wartosci min w kolumnach
   min_x( mD, vM, m, n );
   for(j = 0; j < m; j++)
```

```
   {
      for(i = 0; i < n; i++) { *(mN.b+m*i+j) = *(mD.b+m*i+j) / *(vM.b+j); }
   }
   vM.~array();
}

/* ----------------------------------------------------------------------------
   PLIK: norm02.cpp
   ?: definicje funkcji klasy norm02
---------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: norm02 - konstruktor obiektow klasy norm02
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm02 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
----------------------------------------------------------------------------*/
norm02::norm02( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~norm02 - destruktor obiektow klasy norm02
PARAMETRY: void
WYWOLANIE: ~norm02();
REZULTAT: void
----------------------------------------------------------------------------*/
norm02::~norm02()
{

}

/*============================================================================
FUNKCJA: normal - normalizacja cech metoda przeksztalcenia ilorazowego (2)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
----------------------------------------------------------------------------*/
void norm02::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vM( m );                // wektor wartosci max w kolumnach
   max_x( mD, vM, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++) { *(mN.b+m*i+j) = *(mD.b+m*i+j) / *(vM.b+j); }
   }
   vM.~array();
}

/* ----------------------------------------------------------------------------
   PLIK: norm03.cpp
   ?: definicje funkcji klasy norm03
---------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: norm03 - konstruktor obiektow klasy norm03
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm03 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
----------------------------------------------------------------------------*/
```

```
norm03::norm03( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~norm03 - destruktor obiektow klasy norm03
PARAMETRY: void
WYWOLANIE: ~norm03();
REZULTAT: void
----------------------------------------------------------------------------*/
norm03::~norm03()
{

}

/*============================================================================
FUNKCJA: normal - normalizacja cech metoda przeksztalcenia ilorazowego (3)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
----------------------------------------------------------------------------*/
void norm03::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vE( m );                    // wektor wartosci ex w kolumnach
   mean_x( mD, vE, m, n );           // obliczenie wektora srednich
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++) { *(mN.b+m*i+j) = *(mD.b+m*i+j) / *(vE.b+j); }
   }
   vE.~array();
}



/* ---------------------------------------------------------------------------
   PLIK: norm04.cpp
   ?: definicje funkcji klasy norm04
--------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: norm04 - konstruktor obiektow klasy norm04
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm04 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
----------------------------------------------------------------------------*/
norm04::norm04( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~norm04 - destruktor obiektow klasy norm04
PARAMETRY: void
WYWOLANIE: ~norm04();
REZULTAT: void
----------------------------------------------------------------------------*/
norm04::~norm04()
{

}

/*============================================================================
FUNKCJA: normal - normalizacja cech metoda przeksztalcenia ilorazowego (4)
PARAMETRY:
   mD - macierz danych
```

```
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void norm04::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vS( m );                // wektor sum w kolumnach macierzy
   sum_x( mD, vS, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++) { *(mN.b+m*i+j) = *(mD.b+m*i+j) / *(vS.b+j); }
   }
   vS.~array();
}



/* ----------------------------------------------------------------------------
   PLIK: norm05.cpp
   ?: definicje funkcji klasy norm05
---------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: norm05 - konstruktor obiektow klasy norm05
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm05 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------------*/
norm05::norm05( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~norm05 - destruktor obiektow klasy norm05
PARAMETRY: void
WYWOLANIE: ~norm05();
REZULTAT: void
------------------------------------------------------------------------------*/
norm05::~norm05()
{

}

/*============================================================================
FUNKCJA: normal - normalizacja cech metoda standaryzacji (5)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void norm05::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vE( m );                // wektor srednich
   array vS( m );                // wektor odchylen standardowych
   mean_x( mD, vE, m, n );
   stdev_x( mD, vS, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++)
      {
          *(mN.b+m*i+j) = ( *(mD.b+m*i+j) - *(vE.b+j) ) / *(vS.b+j);
      }
   }
   vE.~array();
```

```
   vS.~array();
}



/* ------------------------------------------------------------------------
   PLIK: norm06.cpp
   ?: definicje funkcji klasy norm06
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: norm06 - konstruktor obiektow klasy norm06
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm06 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------*/
norm06::norm06( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*========================================================================
FUNKCJA: ~norm06 - destruktor obiektow klasy norm06
PARAMETRY: void
WYWOLANIE: ~norm06();
REZULTAT: void
------------------------------------------------------------------------*/
norm06::~norm06()
{

}

/*========================================================================
FUNKCJA: normal - normalizacja cech metoda standaryzacji (6)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
------------------------------------------------------------------------*/
void norm06::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vS( m );                // wektor odchylen standardowych
   stdev_x( mD, vS, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++) { *(mN.b+m*i+j) = *(mD.b+m*i+j) / *(vS.b+j); }
   }
   vS.~array();
}



/* ------------------------------------------------------------------------
   PLIK: norm07.cpp
   ?: definicje funkcji klasy norm07
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: norm07 - konstruktor obiektow klasy norm07
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm07 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
```

```
-------------------------------------------------------------------------*/
norm07::norm07( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*=========================================================================
FUNKCJA: ~norm07 - destruktor obiektow klasy norm07
PARAMETRY: void
WYWOLANIE: ~norm07();
REZULTAT: void
-------------------------------------------------------------------------*/
norm07::~norm07()
{

}

/*=========================================================================
FUNKCJA: normal - normalizacja cech metoda unitaryzacji (7)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
-------------------------------------------------------------------------*/
void norm07::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vMn( m );                  // wektor wartosci min
   array vMx( m );                  // wektor wartosci max
   min_x( mD, vMn, m, n );
   max_x( mD, vMx, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++)
      {
          *(mN.b+m*i+j) = *(mD.b+m*i+j) / ( *(vMx.b+j) - *(vMn.b+j) );
      }
   }
   vMn.~array();
   vMx.~array();
}



/* ------------------------------------------------------------------------
   PLIK: norm08.cpp
   ?: definicje funkcji klasy norm08
------------------------------------------------------------------------ */

/*=========================================================================
FUNKCJA: norm08 - konstruktor obiektow klasy norm08
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm08 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
-------------------------------------------------------------------------*/
norm08::norm08( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*=========================================================================
FUNKCJA: ~norm08 - destruktor obiektow klasy norm08
PARAMETRY: void
WYWOLANIE: ~norm08();
REZULTAT: void
-------------------------------------------------------------------------*/
norm08::~norm08()
{
```

```
}

/*=========================================================================
FUNKCJA: normal - normalizacja cech metoda unitaryzacji (8)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
-------------------------------------------------------------------------*/
void norm08::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vE( m );                    // wektor wartosci ex w kolumnach
   array vMn( m );                   // wektor wartosci min
   array vMx( m );                   // wektor wartosci max
   mean_x( mD, vE, m, n );
   min_x( mD, vMn, m, n );
   max_x( mD, vMx, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++)
      {
          *(mN.b+m*i+j) = (*(mD.b+m*i+j) - *(vE.b+j)) / (*(vMx.b+j) - *(vMn.b+j));
      }
   }
   vE.~array();
   vMn.~array();
   vMx.~array();
}



/* ------------------------------------------------------------------------
   PLIK: norm09.cpp
   ?: definicje funkcji klasy norm09
------------------------------------------------------------------------ */

/*=========================================================================
FUNKCJA: norm09 - konstruktor obiektow klasy norm09
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: norm09 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
-------------------------------------------------------------------------*/
norm09::norm09( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*=========================================================================
FUNKCJA: ~norm09 - destruktor obiektow klasy norm09
PARAMETRY: void
WYWOLANIE: ~norm09();
REZULTAT: void
-------------------------------------------------------------------------*/
norm09::~norm09()
{

}

/*=========================================================================
FUNKCJA: normal - normalizacja cech metoda unitaryzacji (9)
PARAMETRY:
   mD - macierz danych
   mN - obliczona macierz wartosci znormalizowanych
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: normal( mD, mN, m, n );
REZULTAT: void
```

```
-------------------------------------------------------------------------*/
void norm09::normal( array &mD, array &mN, int m, int n )
{
   int i, j;
   array vMn( m );                  // wektor wartosci min
   array vMx( m );                  // wektor wartosci max
   min_x( mD, vMn, m, n );
   max_x( mD, vMx, m, n );
   for(j = 0; j < m; j++)
   {
      for(i = 0; i < n; i++)
      {
          *(mN.b+m*i+j) =
                     (*(mD.b+m*i+j) - *(vMn.b+j)) / (*(vMx.b+j) - *(vMn.b+j));
      }
   }
   vMn.~array();
   vMx.~array();
}

/* -------------------------------------------------------------------------
   PLIK: aggr01.cpp
   ?: definicje funkcji klasy aggr01
------------------------------------------------------------------------- */

/*=========================================================================
FUNKCJA: aggr01 - konstruktor obiektow klasy aggr01
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr01 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
-------------------------------------------------------------------------*/
aggr01::aggr01( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*=========================================================================
FUNKCJA: ~aggr01 - destruktor obiektow klasy aggr01
PARAMETRY: void
WYWOLANIE: ~aggr01();
REZULTAT: void
-------------------------------------------------------------------------*/
aggr01::~aggr01()
{

}

/*=========================================================================
FUNKCJA: aggreg - bezwzorcowa zmienna syntetyczna wg sredniej arytmetycznej
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vQ, m, n );
REZULTAT: void
-------------------------------------------------------------------------*/
void aggr01::aggreg( array &mN, array &vW, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++) { s = s + *(vW.b+j) * *(mN.b+m*i+j); }
      *(vQ.b+i) = s;
   }
}
```

```
/* ------------------------------------------------------------------------
   PLIK: aggr02.cpp
   ?: definicje funkcji klasy aggr02
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: aggr02 - konstruktor obiektow klasy aggr02
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr02 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------*/
aggr02::aggr02( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*========================================================================
FUNKCJA: ~aggr02 - destruktor obiektow klasy aggr02
PARAMETRY: void
WYWOLANIE: ~aggr02();
REZULTAT: void
------------------------------------------------------------------------*/
aggr02::~aggr02()
{

}

/*========================================================================
FUNKCJA: aggreg - bezwzorcowa zmienna syntetyczna wg sredniej geometrycznej
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vQ, m, n );
REZULTAT: void
------------------------------------------------------------------------*/
void aggr02::aggreg( array &mN, array &vW, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 1;
      for(j = 0; j < m; j++) { s = s * pow( *(mN.b+m*i+j), *(vW.b+j) ); }
      *(vQ.b+i) = s;
   }
}



/* ------------------------------------------------------------------------
   PLIK: aggr03.cpp
   ?: definicje funkcji klasy aggr03
------------------------------------------------------------------------ */

/*========================================================================
FUNKCJA: aggr03 - konstruktor obiektow klasy aggr03
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr03 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------*/
aggr03::aggr03( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{
```

```
}

/*============================================================================
FUNKCJA: ~aggr03 - destruktor obiektow klasy aggr03
PARAMETRY: void
WYWOLANIE: ~aggr03();
REZULTAT: void
----------------------------------------------------------------------------*/
aggr03::~aggr03()
{

}

/*============================================================================
FUNKCJA: aggreg - bezwzorcowa zmienna syntetyczna wg sredniej harmonicznej
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vQ, m, n );
REZULTAT: void
----------------------------------------------------------------------------*/
void aggr03::aggreg( array &mN, array &vW, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++) { s = s + ( *(vW.b+j) / *(mN.b+m*i+j) ); }
      *(vQ.b+i) = s;
   }
}



/* ---------------------------------------------------------------------------
   PLIK: aggr04.cpp
   ?: definicje funkcji klasy aggr04
--------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: aggr04 - konstruktor obiektow klasy aggr04
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr04 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
----------------------------------------------------------------------------*/
aggr04::aggr04( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~aggr04 - destruktor obiektow klasy aggr04
PARAMETRY: void
WYWOLANIE: ~aggr04();
REZULTAT: void
----------------------------------------------------------------------------*/
aggr04::~aggr04()
{

}

/*============================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci Hamminga
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
```

```
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void aggr04::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++)
      {
         s = s + *(vW.b+j) * fabs( *(mN.b+m*i+j) - *(vP.b+j) );
      }
      *(vQ.b+i) = s;
   }
}



/* ----------------------------------------------------------------------------
   PLIK: aggr05.cpp
   ?: definicje funkcji klasy aggr05
---------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: aggr05 - konstruktor obiektow klasy aggr05
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr05 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------------*/
aggr05::aggr05( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~aggr05 - destruktor obiektow klasy aggr05
PARAMETRY: void
WYWOLANIE: ~aggr05();
REZULTAT: void
------------------------------------------------------------------------------*/
aggr05::~aggr05()
{

}

/*============================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci Euklidesa
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void aggr05::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++)
      {
         s = s + *(vW.b+j) * pow( *(mN.b+m*i+j) - *(vP.b+j), 2 );
```

```
      }
      *(vQ.b+i) = sqrt( s );
   }
}



/* ---------------------------------------------------------------------------
   PLIK: aggr06.cpp
   ?: definicje funkcji klasy aggr06
--------------------------------------------------------------------------- */

/*===========================================================================
FUNKCJA: aggr06 - konstruktor obiektow klasy aggr06
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr06 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
---------------------------------------------------------------------------*/
aggr06::aggr06( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*===========================================================================
FUNKCJA: ~aggr06 - destruktor obiektow klasy aggr06
PARAMETRY: void
WYWOLANIE: ~aggr06();
REZULTAT: void
---------------------------------------------------------------------------*/
aggr06::~aggr06()
{

}

/*===========================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci Jeffreysa-Matusita
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
---------------------------------------------------------------------------*/
void aggr06::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++)
      {
         s = s + *(vW.b+j) * pow(sqrt(*(mN.b+m*i+j)) - sqrt(*(vP.b+j)), 2);
      }
      *(vQ.b+i) = s;
   }
}



/* ---------------------------------------------------------------------------
   PLIK: aggr07.cpp
   ?: definicje funkcji klasy aggr07
--------------------------------------------------------------------------- */

/*===========================================================================
FUNKCJA: aggr07 - konstruktor obiektow klasy aggr07
PARAMETRY:
   _m - liczba cech (pol)
```

```
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr07 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------------*/
aggr07::aggr07( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~aggr07 - destruktor obiektow klasy aggr07
PARAMETRY: void
WYWOLANIE: ~aggr07();
REZULTAT: void
------------------------------------------------------------------------------*/
aggr07::~aggr07()
{

}

/*============================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci Braya-Curtisa
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void aggr07::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s1, s2;
   for(i = 0; i < n; i++)
   {
      s1 = 0;  s2 = 0;
      for(j = 0; j < m; j++)
      {
         s1 = s1 + *(vW.b+j) * fabs( *(mN.b+m*i+j) - *(vP.b+j) );
         s2 = s2 + *(vW.b+j) * fabs( *(mN.b+m*i+j) + *(vP.b+j) );
      }
      *(vQ.b+i) = s1 / s2;
   }
}



/* ---------------------------------------------------------------------------
   PLIK: aggr08.cpp
   ?: definicje funkcji klasy aggr08
--------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: aggr08 - konstruktor obiektow klasy aggr08
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr08 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------------*/
aggr08::aggr08( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
```

```
FUNKCJA: ~aggr08 - destruktor obiektow klasy aggr08
PARAMETRY: void
WYWOLANIE: ~aggr08();
REZULTAT: void
-------------------------------------------------------------------------*/
aggr08::~aggr08()
{

}

/*=========================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci Clarka
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
-------------------------------------------------------------------------*/
void aggr08::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++)
      {
         s = s + *(vW.b+j) * pow((*(mN.b+m*i+j) - *(vP.b+j))/(*(mN.b+m*i+j) + *(vP.b+j)),2);
      }
      *(vQ.b+i) = sqrt( s );
   }
}



/* -------------------------------------------------------------------------
   PLIK: aggr09.cpp
   ?: definicje funkcji klasy aggr09
------------------------------------------------------------------------- */

/*=========================================================================
FUNKCJA: aggr09 - konstruktor obiektow klasy aggr09
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr09 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
-------------------------------------------------------------------------*/
aggr09::aggr09( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*=========================================================================
FUNKCJA: ~aggr09 - destruktor obiektow klasy aggr09
PARAMETRY: void
WYWOLANIE: ~aggr09();
REZULTAT: void
-------------------------------------------------------------------------*/
aggr09::~aggr09()
{

}

/*=========================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci "Canberra"
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
```

```
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void aggr09::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s;
   for(i = 0; i < n; i++)
   {
      s = 0;
      for(j = 0; j < m; j++)
      {
         s = s + *(vW.b+j) * (fabs(*(mN.b+m*i+j) - *(vP.b+j))/(*(mN.b+m*i+j)
               + *(vP.b+j)));
      }
      *(vQ.b+i) = s ;
   }
}



/* ---------------------------------------------------------------------------
   PLIK: aggr10.cpp
   ?: definicje funkcji klasy aggr10
--------------------------------------------------------------------------- */

/*============================================================================
FUNKCJA: aggr10 - konstruktor obiektow klasy aggr10
PARAMETRY:
   _m - liczba cech (pol)
   _n - liczba obiektow (rekordow)
   _v - liczba okresow (warstw)
   _w - rozmiar pola (domyslnie _w = 19)
   _d - liczba miejsc dziesietnych (domyslnie _d = 6)
   _t - typ pola (domyslnie _t = 'N')
WYWOLANIE: aggr10 obj(_m, _n, _v, _w, _d, _t);
REZULTAT: void
------------------------------------------------------------------------------*/
aggr10::aggr10( const int _m, const int _n, const int _v, const int _w,
         const int _d, const char _t ) : mva( _m, _n, _v, _w, _d, _t )
{

}

/*============================================================================
FUNKCJA: ~aggr10 - destruktor obiektow klasy aggr10
PARAMETRY: void
WYWOLANIE: ~aggr10();
REZULTAT: void
------------------------------------------------------------------------------*/
aggr10::~aggr10()
{

}

/*============================================================================
FUNKCJA: aggreg - wzorcowa zmienna syntetyczna wg odleglosci katowej
PARAMETRY:
   mN - macierz danych znormalizowanych
   vW - wektor zawierajacy wagi poszczegolnych cech (m)
   vP - wektor zawierajacy wzorzec rozwoju
   vQ - wektor zawierajacy wyznaczone zmienne syntetyczne dla (n) obiektow
   m - liczba kolumn
   n - liczba wierszy
WYWOLANIE: aggreg( mN, vW, vP, vQ, m, n );
REZULTAT: void
------------------------------------------------------------------------------*/
void aggr10::aggreg( array &mN, array &vW, array &vP, array &vQ, int m, int n )
{
   int i, j;
   double s1, s2, s3;
   for(i = 0; i < n; i++)
   {
      s1 = 0;      s2 = 0;      s3 = 0;
      for(j = 0; j < m; j++)
      {
```

```
            s1 = s1 + *(vW.b+j) * *(mN.b+m*i+j) * *(vP.b+j);
            s2 = s2 + *(vW.b+j) * pow( *(mN.b+m*i+j), 2 );
            s3 = s3 + *(vW.b+j) * pow( *(vP.b+j), 2 );
        }
        *(vQ.b+i) = s1 / sqrt( s2 * s3 );
    }
}
```