ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE

WYDZIAŁ INFORMATYKI

mgr inż. Krzysztof Michał Lorenz

Metoda selekcji cech wykorzystująca paradygmat algorytmu genetycznego dostosowana do specyficznych charakterystyk interfejsów mózg komputer

Rozprawa doktorska

Promotor: dr hab. Izabela Rejer, prof. ZUT

Szczecin 2023

Składam serdeczne podziękowania
Mojej Żonie,
na którą zawsze mogłem liczyć,
za Jej wsparcie, wiarę w moje możliwości
oraz za niezwykłą cierpliwość,
z którą podchodziła do mnie podczas realizacji pracy doktorskiej.

Składam serdeczne podziękowania
mojej Promotor
Pani prof. dr hab. Izabeli Rejer,
bez której moja praca doktorska nie mogłaby powstać.

Pani Profesor,
dziękuję za inspirację i wsparcie
udzielone mi w trakcie przygotowywania publikacji
oraz wyrozumiałość, życzliwość
i pracę włożoną w obecny kształt niniejszej rozprawy.

Spis treści

Wstęp 6

Rozdział I Interfejs Mózg–Komputer 14

1.1 Rys historyczny 14

1.2. Obecne i przyszłe obszary zastosowań interfejsu mózg–komputer. 16

1.3 Interfejs oparty na wyobrażeniu ruchu 21

1.4 Struktura interfejsu BCI 23

Rozdział II Proces optymalizacji 30

2.1 Optymalizacja jedno i wielokryterialna 30

2.2 Deterministyczne metody optymalizacji 33

2.3 Stochastyczne metody optymalizacji 42

2.4 Optymalizacja wielokryterialna 48

2.5 Rozwiązania niezdominowane oraz Front Pareto 54

Rozdział III Selekcja cech metodami klasycznymi 57

3.1 Ogólne zagadnienia selekcji cech 57

3.2 Metody wrapper 60

3.3 Metody filtrujące 67

3.4 Metody wbudowane 77

3.5 Metody redukcji wymiarowości wektora cech a metody selekcji cech. 80

Rozdział IV Algorytmy genetyczne 84

4.1 Metody symulowanej ewolucji 84

4.2 Algorytm genetyczny – podstawowe pojęcia 88

4.3 Schemat klasycznego algorytmu genetycznego 91

4.4 Kodowanie rozwiązań do postaci osobników 95

4.5 Operatory genetyczne 99

4.6. Selekcja osobników 105

Rozdział V Algorytmy genetyczne w procesie selekcji cech 108
5.1. Klasyczny algorytm genetyczny 109
5.2. Klasyczny algorytm genetyczny z członem kary.....110
5.3. Klasyczny algorytm genetyczny – odmienne sposoby kodowania oraz operatory genetyczne.....110
5.4. Wielokryterialny Algorytm Genetyczny – metoda NSGA II.....115
5.5. Algorytm Genetyczny z ustaloną liczbą cech (algorytm Cullinga) 119
5.6. Algorytm genetyczny z agresywną mutacją (GAAM).....122
5.7. Algorytm genetyczny z agresywną mutacją i zmniejszającą się liczbą genów (melting-GAAM).....126
5.8. GAAMmf – Algorytm genetyczny z agresywną mutacją i malejącą liczbą cech 129
Rozdział VI Weryfikacja działania algorytmu GAAMmf 135
6.1. Eksperyment 1 – badanie odporności i stabilności algorytmu 136
6.1.1. Zbiory danych wykorzystane w eksperymencie 1 136
6.1.2. Etap 1 - Wpływ parametrów *accWeight* i *fsWeight* na wyniki algorytmu..140
6.1.3. Etap 2 - Wpływ prawdopodobieństwa mutacji na wyniki algorytmu.....144
6.1.4. Etap 3 - Badanie stabilności algorytmu.....147
6.1.5. Etap 4 - Zestawienie wyników algorytmu dla 5 zbiorów BCI 151
6.2. Eksperyment 2 - porównanie algorytmu GAAMmf z algorytmami referencyjnymi na 5 zbiorach BCI 154
6.3. Eksperyment 3 - porównanie algorytmu GAAMmf z algorytmami referencyjnymi na benchmarkowych zbiorach danych 159
Podsumowanie 166
Spis literatury.....169
Spis tabel.....220
Spis rysunków.....222

Wstęp

Interfejs mózg-komputer (ang. *Brain Computer Interface, BCI*) to system służący do komunikacji i kontroli, w którym polecenia sterujące i komunikaty przekazywane przez użytkownika nie są uzależnione od standardowych kanałów wyjściowych mózgu [1-3]. W praktyce oznacza to, że informacje nie są przekazywane przez nerwy i mięśnie, a aktywność neuromięśniowa nie jest konieczna do osiągnięcia określonych działań, takich jak np. sterowanie wózkiem inwalidzkim. Sterowanie odbywa się przy pomocy komend, które są ekstrahowane z bieżącej aktywności mózgowej użytkownika interfejsu dzięki zastosowaniu odpowiednich algorytmów. Interfejsy mózg-komputer umożliwiają więc komunikację między człowiekiem a komputerem za pomocą fal mózgowych [4-6].

Interfejs BCI stanowi innowacyjne rozwiązanie w dziedzinie technologii medycznych [7-17]. Jego głównym zastosowaniem jest poprawa jakości życia osób dotkniętych schorzeniami, takimi jak stwardnienie zanikowe boczne, czy stwardnienie rozsiane oraz osób, które uległy poważnym wypadkom komunikacyjnym czy udarowi mózgu. W Polsce, liczba osób cierpiących na stwardnienie zanikowe boczne wynosi około 3 osób na 100000 rocznie [18], natomiast na całym świecie liczba ta szacowana jest na około 400 000 osób [19]. Ponadto, każdego roku w Polsce notuje się ponad 90 tysięcy przypadków udaru mózgu [20] oraz ponad 20 tysięcy ofiar wypadków komunikacyjnych [21].

Niemniej jednak warto zauważyć, że interfejs mózg-komputer znajduje zastosowanie nie tylko w medycynie, ale także w wojsku, w branży rozrywkowo-multimedialnej oraz wielu innych dziedzinach życia [22]. W kontekście działań wojskowych, interfejs pozwala na bezpośrednią komunikację między żołnierzami na polu walki, zwiększając tym samym ich zdolności operacyjne i skuteczność działań, natomiast w branży rozrywkowo-multimedialnej, umożliwia realizację nowych koncepcji, które przyczyniają się do rozwoju tej dziedziny.

W fazie projektowania interfejsu mózg-komputer, kluczowym aspektem jest zapewnienie wysokiej dokładności klasyfikacji sygnałów elektroencefalograficznych (EEG) [23-25], aby poprawnie przyporządkować aktualnie rejestrowany sygnał do jednej

z predefiniowanych klas aktywności umysłowej. Błędna klasyfikacja może prowadzić do wykonywania niewłaściwych działań, co w przypadku interfejsów przeznaczonych do wsparcia użytkownika w komunikacji z otoczeniem może jedynie spowolnić komunikację, ale już w przypadku interfejsów, które mają służyć do sterowania (np. sterowania wózkiem inwalidzkim), błędy klasyfikacji mogą stanowić poważne zagrożenie dla zdrowia użytkownika.

Dlatego też badania nad interfejsami BCI skupiają się na poszukiwaniu metod i technik, które pozwolą na zwiększenie dokładności klasyfikacji. Wysoka dokładność klasyfikacji zależy od wielu decyzji, które są podejmowane na różnych etapach budowy interfejsu mózg-komputer, począwszy od wstępnego przetwarzania sygnału, przez ekstrakcję cech i ich selekcję, aż do finalnego wyboru klasyfikatora i jego parametrów. Ważnym elementem projektowania interfejsu BCI jest także odpowiednie dostosowanie go do indywidualnych potrzeb użytkownika. Interfejs powinien uwzględniać różnorodne sposoby myślenia użytkownika, a także uwzględniać zmienność sygnałów EEG w czasie, wynikającą z aktualnie odczuwanego stanu emocjonalnego, poziomu zmęczenia czy innych czynników zewnętrznych [26].

W literaturze naukowej prowadzona jest dyskusja na temat wpływu poszczególnych etapów działania interfejsu BCI na ostateczną dokładność klasyfikacji [26-29]. W tej kwestii trudno wskazać jednoznacznie najważniejszy etap, ponieważ błędy popełnione na każdym z nich mogą wpłynąć negatywnie na jakość klasyfikacji. Jednakże obecnie za najbardziej problematyczne uważa się etapy ekstrakcji i selekcji cech. Niniejsza praca stanowi wkład w rozwój drugiego z wskazanych etapów – etapu selekcji cech.

Selekcja cech to nic innego jak ograniczenie zbioru wszystkich cech opisujących badany system, do zestawu cech o jak najwyższych zdolnościach dyskryminacyjnych [4,30-39]. Etap selekcji cech jest niezmiennie istotny zwłaszcza w przypadku interfejsów wykorzystujących w procesie sterowania i komunikacji potencjały motoryczne związane z ruchem. Interfejsy tego rodzaju umożliwiają interakcję użytkownika z urządzeniami lub aplikacjami za pomocą aktywności elektroencefalograficznej generowanej przez mózg, podczas wyobrażania konkretnych ruchów, takich jak na przykład poruszanie ręką lub stopą, czy podnoszenie palca, bez fizycznego wykonywania tych ruchów. W trakcie wyobrażenia ruchu w mózgu generowane są charakterystyczne wzorce aktywności, które mogą być zarejestrowane za pomocą elektrod umieszczonych na głowie lub wewnątrz mózgu (implanty neuronalne). Zarejestrowane sygnały są następnie analizowane przy

użyciu zaawansowanych algorytmów, które identyfikują wzorce aktywności mózgowej związane z wyobrażeniem ruchu poszczególnych części ciała. Po zidentyfikowaniu wzorca, wskazującego na wyobrażenie konkretnego rodzaju ruchu, wzorzec jest tłumaczony na przypisane do niego polecenie sterujące, odpowiadające na przykład za przesunięcie kursora myszki na ekranie monitora [4,40].

Aktywność mózgowa generowana w trakcie wyobrażenia ruchu jest w dużym stopniu podmiotowo-zależna, co oznacza, że dokładne wzorce odpowiadające wyobrażeniu konkretnego ruchu różną się między użytkownikami. Stąd proces generowania cech potencjalnie opisujących poszczególne wzorce jest silnie nadmiarowy; w procesie tym generowanych jest bardzo dużo podobnych do siebie cech, z których później wybierane są jedynie te, które odzwierciedlają faktyczną charakterystykę wzorca danego użytkownika [4].

W przypadku interfejsu opartego na wyobrażeniu ruchu, wymiarowość przestrzeni cech możliwych do wyekstrahowania z sygnału EEG z łatwością może osiągnąć rząd 10^2-10^4 cech. Tak duża liczba cech jest problematyczna z kilku powodów. Po pierwsze, w celu wygodnego użytkowania, interfejs powinien działać w trybie zbliżonym do rzeczywistego. Konieczność ekstrakcji tysięcy cech w celu wydania każdej komendy znacznie utrudnia to zadanie. Ograniczenie zestawu cech do kilku-kilkunastu cech przyspiesza czas reakcji interfejsu. W przypadku interfejsu mózg-komputer jest to szczególnie ważne, ponieważ użytkownik powinien jak najszybciej zobaczyć efekt swojego działania, na przykład wyobrażenie ruchu, aby mózg mógł skutecznie powiązać wykonaną lub wyobrażoną czynność z widocznym na ekranie efektem. Niestety wzorce umysłowe zmieniają się powoli, np. aby zaobserwować podstawowe rytmy związane z wyobrażeniem ruchu w sygnale EEG, potrzeba około 6-7 sekund po wykonaniu/wyobrażeniu ruchu [41]. W związku z tym już sama rejestracja sygnału wprowadza duże opóźnienie reakcji interfejsu, dlatego nieakceptowalne jest dodatkowe zwiększanie tego opóźnienia poprzez ekstrahowanie niepotrzebnych cech z sygnału. Należy tutaj zaznaczyć, że szybkość działania interfejsu nie ma znaczenia na etapie jego budowy ani rekalibracji, które odbywają się w trybie offline, lecz ma kluczowe znaczenie w codziennej eksploatacji systemu, gdzie niezbędne jest szybkie i natychmiastowe rozpoznawanie sygnału, przez co nie ma czasu na wyznaczanie wartości cech o niskich zdolnościach dyskryminacyjnych.

Drugą z kluczowych przyczyn konieczności ograniczenia zbioru cech w interfejsach BCI jest znaczne ograniczenie liczby dostępnych danych pomiarowych.

Z uwagi na subiektywny charakter fal mózgowych, klasyfikator wykorzystywany w interfejsie jest często kalibrowany dla każdego użytkownika z osobna. Sesja kalibracyjna, podczas której następuje kalibracja parametrów klasyfikatora, nie może być zbyt długa ze względu na wiele czynników, takich jak zmęczenie użytkownika czy spłaszczanie fal mózgowych w wyniku wystąpienia efektu habituacji. Dlatego zbiór prób dostępnych dla procesu kalibracji interfejsu jest z reguły niewielki, najczęściej jest to około 100-200 prób. W porównaniu z tysiącami cech możliwych do ekstrakcji z sygnału EEG, tak mała liczba prób może prowadzić do nadmiernego dopasowania klasyfikatora do danych treningowych, co może negatywnie wpłynąć na jego zdolności uogólniające. W związku z tym, aby uniknąć efektu przeuczenia klasyfikatora, selekcja kilku-kilkunastu cech o najwyższych zdolnościach dyskryminacyjnych jest kluczowym etapem w budowie interfejsu opartego na wyobrażeniu ruchu.

Kolejnym aspektem, na który należy zwrócić uwagę jest fakt, że zastosowanie niewielkiego zbioru cech w procesie budowy interfejsu przynosi korzyści wynikające z łatwiejszej interpretacji reguł klasyfikacyjnych leżących u podstaw interfejsu. Redukcja liczby cech umożliwia także odkrycie nowych praw rządzących procesami umysłowymi. Kolejną korzyścią wynikającą z redukcji liczby cech jest niższy koszt oraz większa wygoda eksploatacji interfejsu. Wskazane korzyści te są widoczne szczególnie wtedy, kiedy w procesie selekcji cech odrzucone zostają wszystkie cechy wyznaczone dla konkretnego kanału rejestrującego dane. W takim przypadku możliwe jest ograniczenie liczby elektrod wykorzystywanych w późniejszej eksploatacji interfejsu.

Olbrzymia przestrzeń potencjalnych cech możliwych do wyekstrahowania z sygnału EEG w interfejsie opartym na wyobrażeniu ruchu praktycznie uniemożliwia zastosowanie procedury pełnego przeszukania przestrzeni cech, która jest jedyną procedurą pozwalającą na znalezienie globalnie optymalnego zbioru cech [12]. Stąd też w procesie selekcji cech stosowane są metody przeszukiwania przestrzeni potencjalnych cech dające rozwiązania jedynie lokalnie optymalne. Grupa tych metod obejmuje zarówno metody klasyczne, jak i metody heurystyczne, wzorowane na zachowaniach obserwowanych w przyrodzie. Wśród metod klasycznych możemy wyróżnić na przykład: metodę CFS (*Correlation based Feature Selection*) [4,38,42-50], metodę ReliefF [38,42,51-60], metodę Lasso (ang. *Least Absolute Shrinkage and Selection Operator*) [61-68], metodę selekcji w przód (ang. *Sequential Forward Selection, SFS*) [61,69-78], metodę selekcji w tył (ang. *Sequential Backward Selection, SBS*) [61,69,7173,77,79] oraz wiele innych. Charakteryzują się one szeregiem zalet. Po pierwsze, są stosunkowo

proste i zrozumiałe, co ułatwia ich implementację. Ponadto, niezależność od konkretnego modelu umożliwia ich zastosowanie w różnorodnych kontekstach analizy danych. Łatwość interpretacji wyników, pozwalająca na wyjaśnienie istotności poszczególnych cech w procesie decyzyjnym, stanowi kolejną ważną zaletę. Jeżeli chodzi o wady metody klasycznych, to trudno jest wskazać wady wspólne dla wszystkich metod – każda metoda ma pewne właściwe sobie niedogodności. Jedyną negatywną cechą wspólną dla większości metod klasycznych jest to, że nie uwzględniają one nieliniowych zależności między cechami, zwłaszcza w przypadku zbiorów złożonych z wielu cech, co może prowadzić do utraty informacji o ich wspólnym wpływie na proces analizy.

Oprócz metod klasycznych do selekcji cech wykorzystuje się także metody heurystyczne, a wśród nich szczególnie często algorytmy genetyczne oraz ostatnio również algorytmy optymalizacji za pomocą roju cząstek (ang. *Particle Swarm Optimisation, PSO*) [80]. Potrafią one działać w nieliniowych przestrzeniach cech, uwzględniać interakcje występujące między cechami oraz adaptować się do zmieniających się danych. Dzięki swojej zdolności do jednoczesnego eksplorowania różnych fragmentów przestrzeni cech, mają szerokie zastosowanie w procesie selekcji cech.

Istnieje wiele odmian algorytmów genetycznych, przy czym w procesie selekcji cech najczęściej stosowane są: algorytm genetyczny Hollanda [26,81-89], algorytm genetyczny Hollanda z funkcją kary [26,90] oraz algorytm genetyczny NSGA (ang. *Nondominated Sorting Genetic Algorithm*) [26,91-92,94]. Pomimo, że wszystkie trzy algorytmy są często wykorzystywane do selekcji cech w problemach rzeczywistych, to są one obarczone pewnymi wadami. W klasycznym algorytmie genetycznym Hollanda funkcja przystosowania opiera się wyłącznie na maksymalizacji dokładności klasyfikacji, a trening klasyfikatorów odbywa się przy wykorzystaniu dużej liczby cech, co jest procesem czasochłonnym. Chociaż dokładność klasyfikatorów jest często zbliżona do 100% już w pierwszej iteracji algorytmu, większość z nich ma ograniczone zastosowanie ze względu na swoje ograniczone zdolności uogólniające. Wskazany problem rozwiązano rozszerzając funkcję przystosowania algorytmu o tzn. człon kary, którego zadaniem jest karanie osobników posiadających nadmiarową liczbę cech [90]. Pomimo, że rozwiązanie to prowadzi do osiągnięcia zamierzonego celu, to proces redukcji cech jest bardzo powolny (zwłaszcza dla zbiorów o bardzo dużej liczbie potencjalnych cech), ponieważ rozpoczyna się od osobników kodujących około połowy wszystkich dostępnych cech. Ponadto, w przypadku zbiorów cech o niekorzystnym

stosunku liczby cech do liczby przykładów, optymalizacja może w ogóle nie być możliwa do przeprowadzenia z uwagi na brak możliwości estymacji parametrów klasyfikatorów dla osobników kodujących zbyt dużą liczbę cech.

Z kolei algorytm NSGA [94] już od pierwszej wersji optymalizuje jednocześnie dwa kryteria, tj. zwiększa dokładność klasyfikacji równocześnie starając się zmniejszyć liczbę cech zakodowanych w osobniku. Aby osiągnąć tę jednoczesną optymalizację, funkcja przystosowania algorytmu wykorzystuje zasadę dominacji, w której osobnikom przypisywane są rangi na podstawie poziomu ich dominacji. Z uwagi na to, że algorytm NSGA, podobnie jak algorytm Hollanda z członem kary, korzysta z binarnego schematu kodowania, problemy tu występujące są analogiczne. Pierwszym z nich jest wysoka czasochłonność procesu redukcji zbioru cech, drugim – trudności z procesem estymacji parametrów klasyfikatora, stanowiącego element funkcji przystosowania, w przypadku zbiorów o niekorzystnym stosunku liczby cech do liczby przykładów.

W odpowiedzi na problemy występujące podczas stosowania wymienionych powyżej algorytmów genetycznych, w niniejszej pracy zaproponowano nową wersję algorytmu genetycznego - algorytm genetyczny z agresywną mutacją i malejącą liczbą cech GAAMmf (ang. *Genetic Algorithm with Aggressive Mutation and Minimum Features*). W algorytmie tym wprowadzono szereg zmian, w stosunku do wskazanych powyżej algorytmów, mających na celu przystosowanie go w jak największym stopniu do zagadnienia selekcji cech. Między innymi zmieniono schemat kodowania, z kodowania binarnego na kodowanie za pomocą liczb całkowitych (geny kodują indeksy cech). Dodatkowo, zastosowano operator mutacji dostosowany do kodowania całkowitego (operator agresywnej mutacji). Ponadto wykorzystano, znany z algorytmu Hollanda, człon kary do karania osobników posiadających zbyt wiele cech. Następnie, w celu zrównoważenia wpływu dwóch kryterium optymalizacyjnych (liczby cech oraz dokładności klasyfikacji), wprowadzono funkcję przystosowania opartą na koncepcji rang, wykorzystywanej w algorytmie NSGA. Wszystkie te zmiany umożliwiły stosowanie w algorytmie osobników o dowolnej długości, począwszy od osobników zawierających tylko jeden gen, aż do osobników kodujących wszystkie cechy z przestrzeni cech. Dodatkowo, obu kryteriom optymalizacyjnym przypisano wagi umożliwiające sterowanie naciskiem selekcyjnym; wagi te pozwalają na wskazanie czy algorytm ma prowadzić obliczenia w kierunku zwiększenia dokładności klasyfikacji przy mniejszej redukcji cech, czy też dążyć do zminimalizowania liczby cech, ale kosztem spadku dokładności. Jak zostanie wykazane w pracy, dzięki zaproponowanym

modyfikacjom algorytm GAAMmf pozwala na znalezienie zbioru cech o wyższych zdolnościach dyskryminacyjnych i/lub mniejszej liczbie cech, aniżeli obecnie stosowane algorytmy.

Z zaprezentowanych rozważań wynika teza niniejszej rozprawy, którą można sformułować następująco: ***Opracowana metoda selekcji cech pozwoli na uzyskanie zbioru cech o wyższych lub porównywalnych zdolnościach dyskryminacyjnych, mierzonych precyzją klasyfikacji wzorców aktywności mózgowej, oraz o mniejszej liczbie cech, aniżeli wybrane metody referencyjne.***

Celem rozprawy jest natomiast: ***Opracowanie metody selekcji cech opartej na algorytmach genetycznych dostosowanej do specyficznych charakterystyk interfejsów mózg komputer, to jest do konieczności zachowania informacji o pierwotnej lokalizacji cechy oraz ograniczenia przestrzeni cech do jedynie kilku-kilkunastu istotnych cech.***

Wkład autorski autora niniejszej rozprawy stanowi **nowy algorytm genetyczny do selekcji cech GAAMmf,** umożliwiający dwukryterialną optymalizację, wykorzystującą kryterium maksymalizacji dokładności klasyfikacji oraz kryterium minimalizacji liczby cech kodowanych w osobniku. Badania empiryczne przeprowadzone na rzeczywistych zbiorach danych potwierdziły skuteczność algorytmu oraz jego przewagę w porównaniu do wybranych algorytmów selekcji cech stosowanych obecnie.

Z zaprezentowanych rozważań wynika teza niniejszej rozprawy, którą można sformułować następująco ***Opracowana metoda selekcji cech pozwoli na uzyskanie zbioru cech o wyższych lub porównywalnych zdolnościach dyskryminacyjnych, mierzonych precyzją klasyfikacji wzorców aktywności mózgowej, oraz o mniejszej liczbie cech, aniżeli wybrane metody referencyjne.***

Celem rozprawy jest natomiast: ***Opracowanie metody selekcji cech opartej na algorytmach genetycznych dostosowanej do specyficznych charakterystyk interfejsów mózg komputer, to jest do konieczności zachowania informacji o pierwotnej lokalizacji cechy oraz ograniczenia przestrzeni cech do jedynie kilku-kilkunastu istotnych cech.***

Wkład autorski autora niniejszej rozprawy stanowi **nowy algorytm genetyczny do selekcji cech GAAMmf,** umożliwiający dwukryterialną optymalizację, wykorzystującą kryterium maksymalizacji dokładności klasyfikacji oraz kryterium minimalizacji liczby cech kodowanych w osobniku. Badania empiryczne

przeprowadzone na rzeczywistych zbiorach danych potwierdziły skuteczność algorytmu oraz jego przewagę w porównaniu do wybranych algorytmów selekcji cech stosowanych obecnie.

Niniejsza rozprawa ma charakter metodyczny i wnosi istotny wkład w dziedzinę informatyki, rozumianą jako dyscyplina naukowa, zajmującą się przetwarzaniem informacji z użyciem komputerów [93]. Wkład ten, który stanowi przede wszystkim autorska metoda selekcji cech GAAMmf, zgodnie z systemem CCS (ang. *Computing Classification System*), opracowanym przez Stowarzyszenie na Rzecz Maszyn Obliczeniowych (ang. *Association for Computing Machinery*, ACM) [93] można umiejscowić w klasie *„Computing Methodologies”* („*Metodologie obliczeniowe*”), a w szczególności w podklasie *„Machine learning algorithms”* (*„Algorytmy uczenia maszynowego”*), w kategorii *„Feature selection”* (*„Selekcja cech”*). Pomimo, że opracowany algorytm został opracowany w odpowiedzi na problemy występujące w konkretnej dziedzinie (dziedzinie związanej z interfejsami mózg-komputer), ze względu na uniwersalność zagadnienia, które rozwiązuje - zagadnienia selekcji cech - może on znaleźć zastosowanie w wielu różnorodnych dziedzinach nauki i praktyki związanych z analizą danych, modelowaniem predykcyjnym czy ogólnie pojętą sztuczną inteligencją.

Niniejsza rozprawa składa się z sześciu rozdziałów. Rozdział pierwszy wprowadza w tematykę interfejsu mózg–komputer, obszary jego zastosowań oraz omawia podstawową strukturę interfejsu. Rozdział drugi opisuje procesy optymalizacji, stochastyczne oraz deterministyczne metody optymalizacji oraz optymalizację wielokryterialną. Rozdział trzeci jest poświęcony procesowi selekcji cech oraz klasycznym metodom wykorzystywanym w tym procesie. Czwarty rozdział zawiera opis oraz schemat klasycznego algorytmu genetycznego, a także definicje podstawowych pojęć z nim związanych. Rozdział piąty jest najważniejszym rozdziałem pracy. W rozdziale tym, na tle algorytmów genetycznych wykorzystywanych w procesie selekcji cech, przedstawiono algorytm genetyczny z agresywną mutacją GAAM (ang. *Genetic Algorithm with Aggressive Mutation*) [26,90,94-95] oraz oparty na nim algorytm genetyczny z agresywną mutacją i malejącą liczbą cech GAAMmf (ang. *Genetic Algorithm with Aggressive Mutation and minimum Feature*), stanowiący wkład autora niniejszej rozprawy. W rozdziale szóstym przedstawione zostały eksperymenty weryfikujące skuteczność zaproponowanego w rozprawie algorytmu GAAMmf.

Rozdział I
Interfejs Mózg–Komputer

1.1 Rys historyczny

Początków historii obecnych interfejsów mózg–komputer można doszukiwać się w XVIII wieku, kiedy to w 1786 r. Luigi Galvani wykazał, że skurcz mięśnia wypreparowanej kończyny żaby następuje nie tylko w sposób naturalny, ale również poprzez równoczesne dotknięcie tego mięśnia za pomocą dwóch połączonych ze sobą metali (elektrod). Kolejny krok na drodze do ustanowienia interfejsów pozwalających na sterowanie za pomocą fal mózgowych został wykonany dopiero około 100 lat później, kiedy to w 1875 r. doszło do pierwszego w historii opisu czynności elektrycznej mózgu. Dokonał tego Richard Caton, poprzez wykazanie korelacji pomiędzy zmianą potencjału w odpowiednich obszarach kory mózgowej kotów i królików, a prostymi czynnościami wykonywanymi przez te zwierzęta, np. ruchem głowy lub przeżuwaniem [23,96-97]. Richard Caton zapisał wynik badania w swoim raporcie, który jest dziś uznawany za pierwszy zapis elektroencefalogramu. Piętnaście lat później, polski naukowiec, Adolf Beck obronił rozprawę doktorską pt. *„Oznaczenie lokalizacyi w mózgu i rdzeniu za pomocą zjawisk elektrycznych”*, w której to opisał działanie elektroencefalogramu oraz lokalizację funkcji motorycznych w mózgu. Następnie, na polecenie profesora Nikodema Cybulskiego (Uniwersytet Jagielloński), dr Adolf Beck prowadził dalsze badania na zwierzętach (m.in. na psach, królikach, żabach, małpach), których streszczenie znalazło się w 1890 r. w publikacji: „*Zentralblatt fur Physiologie*” [23,97-98].

W 1913 roku Prawdicz–Niemiński (Rosja) opublikował pierwszą publikację, która zawierała zdjęcie z zapisem czynności elektrycznej mózgu, którą sam później określił jako elektrocerebrogramu [23,97-100]. Po kolejnych badaniach na zwierzętach opisał on także dwie fale [24], które to później zostały opisane przez Hansa Bergera jako fale alfa

i beta [25]. W roku 1925 Hans Berger za pomocą elektroencefalografu (EEG) zarejestrował aktywność mózgu z powierzchni czaszki swojego syna. Po dokładnej analizie wyników EEG, naukowiec zidentyfikował po raz pierwszy w historii fale występujące podczas działania mózgu, (fale alfa) nazwane później falami Bergera. Po czterech latach opublikował artykuł *„Über das Elektroenkephalogramm des Menschen”*, który stanowi do dziś klasykę elektroencefalografii klinicznej, a sam Berger uważany jest za twórcę elektrofizjologii klinicznej [23-25].

W 1934 r. Brian Mattews ze współpracownikami stworzył rejestrator pisakowy, który umożliwiał zapisywanie czynności bioelektrycznej mózgu na papierze przy użyciu zwykłego atramentu [101]. W 1935 r. Jan Friedrich Tonnies stworzył kolejny aparat EEG z atramentowymi rejestratorami, który nazwał Neurograf II. [97,102]. W 1935 r. powstało w Stanach Zjednoczonych urządzenie Grass Model I, zawierające zarówno rejestrator pisakowy jak i wzmacniacz różnicowy. Był to pierwszy elektroencefalograf z możliwością zapisu czynności EEG pobranej jednocześnie z trzech kanałów [97,101]. Do końca lat 70. XX wieku w budowie elektroencefalografów nastąpiły znaczące zmiany. Lampy elektronowe zostały zastąpione przez układy scalone i tranzystory oraz można już było dokonać wielokanałowego zapisu EEG (nawet z 23 kanałów) [103].

Rozwój technik pozwalających na rejestrację aktywności mózgowej utorował drogę interfejsom mózg–komputer. Pierwszy działający interfejs został stworzony w 1964 r. przez neurofizjologa Williama Graya Waltera [2,9,104-105]. Interfejs powstał w efekcie przeprowadzonego przez niego eksperymentu, w trakcie którego wszczepił on elektrody do obszaru kory motorycznej pacjenta, a następnie obserwował aktywność mózgu podczas przełączania przez pacjenta slajdów w prezentacji komputerowej. Następnie Walter przekonfigurował urządzenie tak, aby reagowało w sytuacji, kiedy badany tylko „wykazywał chęć” przełączenia slajdów. Wówczas doszło do ciekawego odkrycia. Slajdy zmieniały się zanim pacjent zdążył użyć przycisku.

Pięć lat po tym odkryciu Eberhard Fetz i jego koledzy z Regionalnego Centrum Badań Wydziału Fizjologii i Biofizyki na Waszyngtońskim Uniwersytecie Medycznym w Seattle prowadząc badania nad warunkowaniem instrumentalnym udowodnili, że małpy są w stanie nauczyć się zginać ramię robota jedynie odpowiednio modulując swoją aktywność neuronalną. Eksperyment, który przeprowadzili w tym celu, polegał na przekształcaniu napięcia pochodzącego z elektrod wszczepionych do kory motorycznej małp na napięcie konieczne do wychylenia wskazówek woltomierza. Igła woltomierza wychylała się wówczas, gdy częstotliwość powstawania sygnałów w korze motorycznej

była duża. Małpa w każdym momencie mogła patrzeć na woltomierz. W sytuacji, gdy wskazówka woltomierza osiągała lub przekraczała wyznaczony wcześniej poziom, wówczas zwierzę było nagradzane. W tej sytuacji małpa bardzo szybko nauczyła się, że aby otrzymać nagrodę, konieczne jest spowodowanie wychylenia wskazówki woltomierza, a zatem nauczyła się w jaki sposób kontrolować aktywność swojego mózgu. W eksperymencie pokazano, że małpy posiadają możliwość nauki świadomego modulowania aktywności poszczególnych części mózgu i sterowania w ten sposób urządzeniami zewnętrznymi [104,106].

W 1999 r. w Alabanie w Rensselarville Institute of Albany została zorganizowana pierwsza międzynarodowa konferencja poświęcona interfejsom mózg–komputer. Uczestnikami konferencji było ponad czterdziestu naukowców oraz inżynierów. Reprezentowali oni 22 różne grupy badające interfejsy mózg–komputer m.in. ze Stanów Zjednoczonych, Wielkiej Brytanii, Kanady, Niemiec oraz Włoch. Podczas tej historycznej konferencji dokonano przeglądu stanu badań nad interfejsem mózg–komputer do 1999 roku oraz określono kolejne cele badawcze. Zidentyfikowano również najważniejsze problemy techniczne, a także opracowano standardowe metody oceny interfejsów, algorytmy i procedury badawcze [104,107-108]. Konferencja ta okazała się przełomem w badaniach nad interfejsem mózg–komputer. Od tego czasu nastąpił coraz szybszy rozwój badań naukowych oraz komercyjnych projektów w zakresie interfejsów umożliwiających komunikację pomiędzy mózgiem człowieka a urządzeniami zewnętrznymi.

1.2. Obecne i przyszłe obszary zastosowań interfejsu mózg–komputer.

Istnieje wiele obszarów, w których interfejs mózg–komputer znajduje (bądź znajdzie w przyszłości) swoje zastosowanie. Głównie są to obszary medyczne, związane z poprawą, ulepszeniem bądź zastąpieniem pewnych funkcji (w większości funkcji motorycznych) układu nerwowego człowieka. Podchodząc do problemu w sposób bardziej systematyczny można wymienić 5 podstawowych celów stosowania interfejsów mózg–komputer [109]:

- całkowite zastąpienie funkcji układu nerwowego utraconych w wyniku choroby lub wypadku;

- przywrócenie utraconych funkcji układu nerwowego (np. poprzez stymulację mięśni u osoby sparaliżowanej);
- poprawa funkcji układu nerwowego (np. poprzez zastosowanie interfejsu w rehabilitacji poudarowej);
- wzmacnianie informacji wyjściowych z układu nerwowego (np. poprzez wykrywanie poziomu stresu lub spadku uwagi w przypadku wykonywania zadań wymagających wysokiej koncentracji);
- wsparcie badań naukowych.

Odnosząc się do pierwszej z wymienionych kategorii, całkowite zastąpienie funkcji motorycznych ma miejsce w przypadku osób niepełnosprawnych w ciężkich stanach chorób takich jak np. mózgowe porażenie dziecięce, stwardnienie rozsiane (SM), udar mózgowy podkorowy, stwardnienie zanikowe boczne czy też zespół Guillain–Barre'a [110]. U osób tych mózg nadal jest sprawny, ale nie są one w stanie komunikować się ze światem zewnętrznym. Stąd zadaniem interfejsu mózg–komputer jest zapewnienie możliwości komunikacji przy wykorzystaniu środków zastępczych. W chwili obecnej najczęściej stosuje się tutaj interfejsy najprostsze, pozwalające na wybór jednej z opcji prezentowanych użytkownikowi. Komunikacja za pomocą tego rodzaju interfejsów jest bardzo utrudniona, ponieważ po pierwsze wywołuje szybkie zmęczenie/znużenie użytkownika, po drugie jest dosyć wolna.

Wśród przykładów interfejsów opracowanych w celu zastąpienia utraconych funkcji układu nerwowego można podać np. interfejs opisany w 2002 r. przez Chenga pozwalający na wybór numeru telefonu za pomocą fal mózgowych [6]. Rok później Bayliss stworzył inteligentny dom w wirtualnym środowisku. Można w nim było zarządzać kilkoma urządzeniami właśnie w oparciu o sygnał pochodzący z fal mózgowych [111-113]. Na początku XXI wieku pojawił się system *IntendiX*, opracowany przez firmę g.tec. Był to system do zamiany sygnału EEG na konkretne czynności, np. do pisania na matrycy przypominającej wygląd klawiatury, a także edycji tekstu i kopiowaniu do innych plików bądź wiadomości e–mail w celu późniejszego wysłania takiego listu do odbiorcy. Trzy lata później firma g.tec wyprodukowała nowy moduł systemu *IntendiX*. Nazywał się on *Screen Overlay Control Interface* (*SOCI*) i pozwalał użytkownikom grać w wiele gier, m.in. *World of Warcraft* lub *Angry Birds* (2015, 2015). W 2006 r. Pfurtscheller wraz ze swoimi kolegami [114] wykorzystał interfejs w celu poruszania się w wirtualnej rzeczywistości. Osoby badane miały wyobrażać sobie ruch

rąk i nóg do dotykania lub chwytania obiektów, które były umieszczone w tej rzeczywistości.

W przyszłości planowane jest zastąpienie komunikacji korzystającej z wyboru jednej z dostępnych opcji komunikacją werbalną, dzięki której użytkownik interfejsu będzie w stanie formułować skomplikowane zdania oraz wyrażać swoje potrzeby i emocje. Zadanie to miałoby zostać zrealizowane poprzez wszczepienie do mózgu chorego interfejsu BCI połączonego z syntezatorem mowy. Urządzenie to miałoby wysyłać sygnały do komputera za pomocą sieci bezprzewodowej. Po treningu urządzenie byłoby w stanie odczytać z aktywności mózgu pacjenta początkowo pojedyncze litery, później całe słowa, a wreszcie całe zdania. Zaletami takiego urządzenia byłaby ciągła dostępność interfejsu, znacznie niższy poziom zmęczenia użytkownika, a także większa wygoda użytkowania. Jednakże wszczepienie takiego interfejsu pociągałoby za sobą pewne niedogodności, np. ryzyko popełnienia błędu przez chirurga wykonującego operację.

Interfejs mózg–komputer może nie tylko zastąpić utracone naturalne kanały wyjściowe centralnego układu nerwowego (ang. *Central Nervous System, CNS*) [7], ale może również je przywrócić. Interfejs tego rodzaju jest stosowany w przypadku utraty połączeń nerwowych np. w wyniku urazu rdzenia kręgowego (kora motoryczna działa prawidłowo, ale z uwagi na brak połączenia z mięśniami informacje nie są przesyłane między korą motoryczną, a mięśniami). Oczywiście BCI nie przywraca naturalnych kanałów wyjściowych CNS w ich pierwotnej postaci, tylko omija je, przywracając ich funkcje w inny sposób. Interfejs BCI w tym zastosowaniu to nic innego jak wszczepiony w korę motoryczną implant przekazujący impulsy elektryczne z mózgu do mięśni za pomocą neuroprotezy. Neuroproteza jest więc z jednej strony podłączona bezpośrednio do mózgu – do obszarów kory mózgowej zawiadujących ruchami danej kończyny, z drugiej strony zaś do mięśni, które ma stymulować. W fazie treningu pacjent stopniowo uczy się, w jaki sposób uaktywniać za pomocą neuroprotezy mięśnie stymulowanej kończyny, aby wykonać stawiane przed nim zadanie. Ten rodzaj interfejsu kierowany jest do około 25 tysięcy osób w całej Europie. Zaletą tego interfejsu jest umożliwienie choremu powrotu do społeczeństwa. Protezę taką jest bardzo łatwo założyć choremu, uaktywnić, a następnie wyłączyć i zdjąć. Wadą jest natomiast cena neuroprotezy (2 000 – 10 000 euro jednorazowo) oraz miesięczny koszt (około 8000 euro

miesięcznie)[1], a także to, że konieczne jest przystosowanie urządzenia do każdego pacjenta indywidualnie. Niezbędna jest również obecność innej osoby, która pomoże założyć a następnie zdjąć neuroprotezę [7,10,110,115-129].

Eksperymenty z wykorzystaniem tej metody prowadził już kilkanaście lat temu Pfurtscheller [130]. Uczył on chorą osobę, która miała przerwany rdzeń kręgowy. Eksperyment polegał na umieszczeniu na powierzchni ręki elektrody. Następnie po otrzymaniu właściwego impulsu z interfejsu, elektroda ta wywoływała skurcz mięśni dłoni.

Innym przykładem jest wykorzystanie interfejsu u osób, które utraciły część lub całkowicie słuch [118,122,131-137]. Dzięki zastosowaniu technologii dousznego implantu słuchowego razem z interfejsem BCI, możliwa jest automatyczna regulacja głośności. Do tej pory pacjenci musieli ręcznie, często za pomocą potencjometru, regulować głośność dźwięków jakie słyszeli, niekiedy nawet kilka lub kilkanaście razy dziennie. Przyczyną była sama budowa implantu. Po podłączeniu implantu do interfejsu BCI pacjent już nie musiał sam regulować dźwięku. Interfejs BCI monitorował jego aktywność mózgową i po wychwyceniu charakterystycznych dla danego pacjenta fal, zmniejszał lub zwiększał dźwięk, jaki słyszał pacjent. Interfejs BCI jest w tym przypadku płytką z elektrodami umieszczoną na głowie pacjenta, którą można w łatwy sposób założyć lub zdjąć. Urządzenie skierowane jest do bardzo dużej liczby odbiorców. Przykładowo w 2012 roku Światowa Organizacja Zdrowia (ang. *World Health Organization*, *WHO*) oszacowała na około 360 milionów liczbę osób mających problem z utratą słuchu.

Interfejsy mózg–komputer mogą również pomóc chorym pacjentom w częściowym lub pełnym odzyskaniu funkcji motorycznych [9]. Jako przykład można tu podać wykorzystanie BCI w rehabilitacji po udarze. W tym zastosowaniu kora mózgowa jest uszkodzona i interfejs jest stosowany jako narzędzie pomagające w naprawie tych uszkodzeń. Dokładniej mówiąc zadaniem interfejsu jest tu wykrycie i wzmocnienie sygnału pochodzącego z uszkodzonych obszarów kory mózgowej i jednoczesna stymulacja mięśni, aby zwiększyć zakres i/lub precyzję ruchu. W 2000 roku pacjentów, w przypadku których można by przyspieszyć rehabilitację poudarową za pomocą interfejsu BCI było ponad milion, a w 2025 r. szacuje się, że będzie ich około 1,5 mln. (szacunki WHO). Niepodważalną zaletą tego rozwiązania jest to, że stosowany jest tu

[1] Dane na sierpień 2016

nieinwazyjny interfejs BCI, natomiast jego główną wadą jest jak na razie konieczność długotrwałego uczestnictwa osoby nadzorującej rehabilitację [1,123,136,138-152].

Kolejnym celem stosowania interfejsów mózg–komputer jest wzmacnianie funkcji układu nerwowego. W tym zastosowaniu chodzi głównie o bieżące monitorowanie aktywności mózgowej użytkownika i reagowanie w przypadku wykrycia pożądanej lub niepożądanej aktywności. Wśród przykładów można podać tu monitorowanie aktywności mózgowej podczas przedłużających się zadań wymagających skupienia, takich jak np. prowadzenie samochodu. Interfejs nie tylko monitoruje tu aktywność, ale jego podstawowym zadaniem jest wykrywanie spadków koncentracji u użytkownika i informowanie go o takich sytuacjach w celu przywrócenia pełnej koncentracji. Interfejs może też monitorować inne rodzaje aktywności mózgowej np. poziom stresu, poziom relaksacji, poziom senności itp. W przypadku osób cierpiących na epilepsję interfejs tego rodzaju wykrywa stan aktywności mózgowej charakterystyczny dla zbliżającego się napadu epilepsji. Po wykryciu zbliżającego się ataku interfejs informuje użytkownika o problemie (np. za pomocą sygnału dźwiękowego) i jednocześnie generuje impuls powstrzymujący atak. Interfejs tego rodzaju kierowany jest do około 65 milionów ludzi na całym świecie, którzy cierpią na epilepsję [3,5,10-17,153-155].

Poza ostrzeganiem użytkownika w przypadku wykrycia niepożądanej aktywności mózgowej, interfejs mózg–komputer może być też wykorzystywany do sterowania działaniami użytkownika w przypadku wykrycia aktywności pożądanej. Najczęściej przytaczanym w tym miejscu przykładem jest nadzorowanie procesu nauki. W zastosowaniu tym interfejs miałby na celu monitorowanie bieżącej aktywności mózgowej użytkownika i takie sterowanie procesem nauki, aby w momentach największej koncentracji prezentować użytkownikowi najtrudniejsze zadania, natomiast w momentach całkowitej utraty koncentracji wprowadzać przerwy w procesie nauki.

Piątym obszarem zastosowań, w którym interfejsy mózg–komputer są bardzo szeroko wykorzystywane są badania naukowe, prowadzone np. nad zagadnieniem podejmowania decyzji, czy wolnej woli człowieka. Interfejs jest również wykorzystywany podczas medycznych eksperymentów, do testowania nowych leków czy terapii.

1.3 Interfejs oparty na wyobrażeniu ruchu

Jak dotąd wiele różnych wzorców aktywności mózgowej (potencjałów mózgowych) było badanych w celu znalezienia takich, które będą zarówno łatwe do automatycznego wykrycia, jak i w miarę łatwe do kontrolowania ich przez użytkownika interfejsu. Potencjały, które są wykorzystywane obecnie można podzielić na dwie główne kategorie: potencjały wywoływane oraz potencjały spontaniczne. Podstawowa różnica między tymi dwoma kategoriami jest taka, że potencjały wywoływane są generowane nieświadomie przez użytkownika w odpowiedzi na konkretny bodziec zewnętrzny (na konkretną stymulację), natomiast potencjały spontaniczne są dobrowolnie/świadomie generowane przez użytkownika, bez żadnej stymulacji zewnętrznej. Obecnie w interfejsach mózg-komputer wykorzystywane są najczęściej dwa rodzaje potencjałów wywoływanych: potencjały wywoływane stanu ustalonego (ang. *Steady State Evoked Potentials*, *SSEP*) [156-158] oraz potencjały P300 [159-161] i jeden rodzaj potencjałów spontanicznych - spontaniczne potencjały motoryczny generowane w trakcie wyobrażenia ruchu.

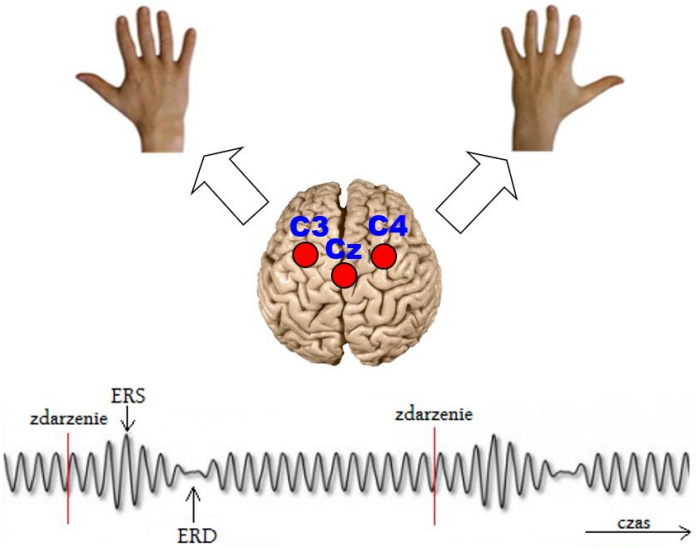
W interfejsach wykorzystujących potencjały SSVEP oraz potencjały P300 cechy stanowiące podstawę procesu klasyfikacji są z reguły nieliczne i dość dobrze określone, a więc proces selekcji cech najczęściej nie jest w ich przypadku potrzebny. Zupełnie inaczej ma się sprawa z interfejsami opartymi na potencjałach motorycznych generowanych w trakcie wyobrażenia ruchu, w przypadku których trudno jest z góry ustalić, które z tysięcy możliwych do wyekstrahowania cech będą miały największe zdolności dyskryminacyjne w odniesieniu do konkretnego użytkownika. Stąd też niniejsza rozprawa jest zorientowana wokół jednego konkretnego rodzaju interfejsów mózg-komputer – interfejsów opartych na spontanicznych potencjałach motorycznych generowanych w trakcie wyobrażenia ruchu.

Potencjały te generowane są nad korą motoryczną pod wpływem ruchu lub wyobrażenia ruchu (wyobrażenie ruchu wywołuje analogiczne zmiany w korze motorycznej jak w przypadku ruchu rzeczywistego [10,162-168]). Najczęściej w podejściu tym wykrywany jest zamierzony ruch prawą czy lewą ręką lub ruch stopami.

W interfejsach mózg–komputer opartych na detekcji potencjałów motorycznych, proces detekcji opiera się na pomiarze zmiany mocy sygnału EEG w danym paśmie częstotliwości. Najsilniejsze zmiany mocy sygnału wywołane rzeczywistym

lub wyobrażonym ruchem rejestrowane są nad korą motoryczną, na elektrodach C3, Cz i C4. Na przykład wyobrażenie ruchu lewej ręki powoduje zmianę mocy sygnału rejestrowanego z prawej półkuli mózgu (elektroda C4); wyobrażenie ruchu prawej ręki – zmianę mocy sygnału w lewej półkuli mózgu (elektroda C3), a wyobrażenie ruchu stóp – zmianę mocy sygnału nad szczeliną międzypółkulową (elektroda Cz) [10,162-165,168].

Wyobrażony lub rzeczywisty ruch wywołuje dwa efekty, stanowiące podstawę rozpoznawania intencji użytkownika interfejsu: desynchronizację rytmu μ (8–11Hz) (efekt zwany z ang. *Event–Related Desynchronization*, *ERD*) [169-171] w sygnale rejestrowanym przez elektrodę ulokowaną nad obszarem kory motorycznej zarządzającym daną kończyną oraz następczą do ruchu synchronizację w paśmie β (13–30Hz) (efekt zwany z ang. *Event–Related Synchronization*, *ERS*) [172-174] w sygnale rejestrowanym przez tę samą elektrodę (Rysunek 1.1).



Rysunek 1.1. Efekt ERD i ERS wywołany przez rzeczywisty lub wyobrażony ruch ręki. Źródło: Opracowanie własne.

Desynchronizacja rytmu μ przejawia się spadkiem mocy sygnału w tym paśmie i jest interpretowana jako aktywacja neuronów w danym fragmencie kory mózgowej, czyli jako ich wyjście ze stanu spoczynkowego i zaangażowanie w zamierzone działanie. Z kolei spadek aktywności neuronów przy przejściu ze stanu aktywnego do stanu spoczynkowego wywołuje synchronizację neuronów w paśmie μ. Rozważając z kolei pasmo β należy stwierdzić rzecz odwrotną – synchronizacja neuronów w tym paśmie (efekt ERS) związana jest ze wzrostem aktywności neuronów, natomiast desynchronizacja – ze spadkiem aktywności.

Podstawową zaletą interfejsów mózg–komputer wykorzystujących potencjały motoryczne jest to, że potencjały te generowane są spontanicznie przez użytkownika. Do ich wywołania nie jest więc potrzebna żadna zewnętrzna stymulacja, która jest niezbędna w przypadku dwóch wcześniej omówionych rodzajów potencjałów mózgowych sterujących interfejsem. Potencjały motoryczne dostarczają więc najbardziej „naturalnego" sposobu sterowania. Wadą tego paradygmatu jest natomiast niski stosunek sygnału do szumu, powodujący trudności z wykryciem rytmów motorycznych w rejestrowanym sygnale. Inną wadą jest konieczność prowadzenia długotrwałego treningu użytkownika, w trakcie którego uczy się on kontrolować amplitudy rytmów motorycznych μ i β [10,108,164,175-177].

Jako przykład można tu podać sterowanie wózkiem inwalidzkim. Podczas eksperymentu badany wyobraża sobie ruch np. lewą ręką, a poprawnie zaprogramowany interfejs odczytuje to jako „skręt w lewo" [26,69].

1.4 Struktura interfejsu BCI

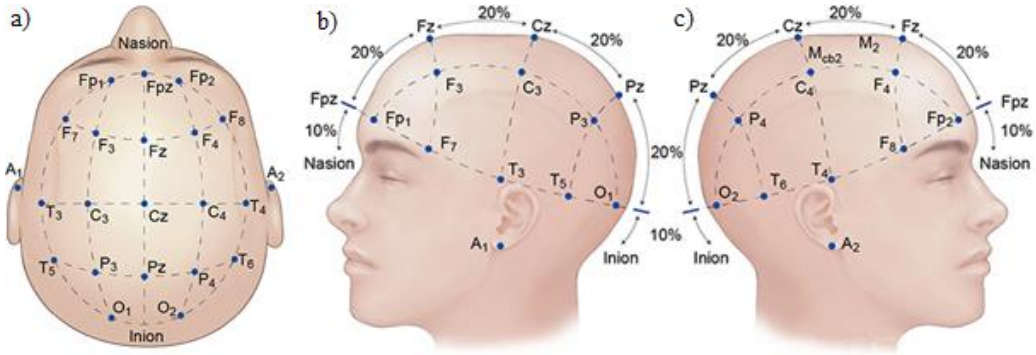
Interfejs mózg-komputer nie jest monolityczną strukturą, lecz jest złożony z kilku oddzielnych modułów. W sposób najbardziej ogólny w strukturze systemu BCI możemy wyodrębnić trzy moduły [178-181]:

1. moduł wejściowy, odpowiedzialny za pomiar aktywności mózgu;
2. moduł przetwarzający, odpowiedzialny za przetworzenie sygnału EEG, wyodrębnienie wzorców z tego sygnału i wybór odpowiedniej komendy sterującej;
3. moduł wyjściowy, odpowiedzialny za przesłanie komendy na zewnątrz interfejsu i tym samym dostarczaniu feedbacku do użytkownika.

Moduł pierwszy jest odpowiedzialny za rejestrację aktywności mózgowej użytkownika interfejsu. Realizacja tego modułu wymaga więc posiadania aparatu EEG oraz zestawu elektrod. Elektrody są mocowane na powierzchni czaszki, a następnie podłączane do aparatu EEG. W celu ułatwienia montażu elektrod w odpowiednich miejscach na powierzchni czaszki stosowane są czepki EEG (Rysunek 1.2). Elektrody są najczęściej umiejscawiane na powierzchni czaszki w miejscach wyznaczonych zgodnie z międzynarodowym systemem lokalizacji elektrod - Systemem 10-20, w którym ustalenie położenia poszczególnych elektrod odbywa się poprzez wyznaczenie

procentowych odległości między punktami charakterystycznymi (punktem *Nasion*, *Inion* oraz *punktami przedusznymi*) [97,166,182] (Rysunek 1.3). Standardem w pracowniach EEG jest korzystanie z 19 elektrod, natomiast w badaniach naukowych wykorzystywane są znacznie bardziej gęste siatki elektrod, zawierające 32, 64, 128, a nawet 1024 elektrody.

Rysunek 1.2. Przykład czepka do badań EEG.
Źródło: [183]



Rysunek 1.3. Położenie elektrod w systemie 10–20; a) widok z góry; b) widok z lewej strony; c) widok z prawej strony.
Źródło: [184]

Realizacja modułu drugiego, modułu przetwarzającego, jest znacznie bardziej skomplikowana [178-181]. Obejmuje ona cały zestaw czynności, które należy wykonać na pobranym z modułu pierwszego sygnale. Moduł ten jest najczęściej dzielony na szereg podmodułów, przy czym to, które z tych podmodułów zostaną wykorzystanie zależy od aktualnego trybu pracy interfejsu.

Generalnie interfejs mózg-komputer może działać w trybie uczenia (inaczej trybie kalibracji) lub w trybie użytkowania. Celem trybu uczenia jest kalibracja klasyfikatora

stanowiącego podstawę interfejsu i dostosowanie go do charakterystycznych cech jego aktualnego użytkownika. Z kolei tryb użytkowania jest wykorzystywany podczas pracy użytkownika z interfejsem.

W trybie użytkowania wykorzystywane są najczęściej trzy podmoduły modułu przetwarzającego [13,71]:

1. Moduł wstępnego przetworzenia sygnału, odpowiedzialny przede wszystkim za oczyszczenie sygnału z artefaktów, to jest z wszystkich sygnałów zebranych przez elektrody, ale mających swoje źródła poza mózgiem (np. EMG, EKG, EOG).
2. Moduł ekstrakcji cech, zadaniem tego modułu jest opisanie sygnału przez kilka najbardziej istotnych cech, np. moc sygnału w poszczególnych pasmach częstotliwości.
3. Moduł klasyfikacji, którego zadaniem jest wyznaczenie klasy odpowiadającej wartościom cech wyznaczonych w module ekstrakcji.

Zarejestrowanie czystego sygnału EEG jest praktycznie niemożliwe. Już Nunez i współpracownicy [185] udowodnili, że tylko około 50% sygnału rejestrowanego w badaniu EEG pochodzi z grupy neuronów znajdujących się pod elektrodą. Reszta to zakłócenia (nazywane artefaktami) wynikające z mrugania powiekami, bicia serca, samoistnego spinania się mięśni, sieci energetycznej itd. [163]. Wpływ niektórych artefaktów na rejestrowany sygnał EEG można ograniczyć odpowiednio przygotowując eksperyment badawczy oraz jego uczestnika. Jednak wielu artefaktów nie da się w ten sposób uniknąć i wiele z nich przenika do rejestrowanego sygnału. Aby w jak największym stopniu oczyścić sygnał z artefaktów w procesie wstępnego przetwarzania sygnału EEG wykorzystuje się całą gamę metod takich jak:

- filtracja pasmowo–przepustowa [186],
- filtry przestrzenne np. filtr Laplace'a (ang. *Laplace Filter*, *LF*) [187],
- metoda wspólnych wzorców przestrzennych (and. *Common Spatial Patterns*, *CSP*) [188],
- analiza składowych niezależnych (ang. *Independent Component Analysis*, *ICA*) [189],
- inne metody z grupy metod ślepej separacji sygnałów (ang. *Blind Signal Separation, BSS*) [190].

Przetworzony wstępnie sygnał EEG jest następnie przekazywany do drugiego z podmodułów modułu przetwarzającego – modułu ekstrakcji cech. Zadaniem tego modułu jest opisanie sygnału EEG zestawem cech, który następnie zostanie wykorzystany w procesie klasyfikacji. Cechy ekstrahowane z sygnału EEG są najczęściej związane z kształtem przebiegu sygnału w czasie, parametrami statystycznymi sygnału, parametrami modeli autoregresyjnych zbudowanych na podstawie czasowego przebiegu sygnału, składowymi częstotliwościowymi sygnału oraz jego parametrami czasowo–częstotliwościowymi. Ta różnorodność cech wykorzystywanych do opisu sygnału wynika z wielości metod, które można wykorzystać do wykonania tego zadania. Wśród metod najczęściej stosowanych w interfejsach mózg-komputer w procesie ekstrakcji cech należy wymienić:

1. Dyskretną Transformatę Fouriera (ang *Discrete Fourier Tansform, DFT*) [191].
2. Dyskretną Transformatę Falkowa (ang. *Discrete Wavelet Transform, DWT*) [191].
3. Krótkoczasową transformatę Fouriera (ang. *Short–Time Fourier Transform, STFT*) [192].
4. Modele Autoregresyjne (ang. *Autoregressive Model, AR model*) [193].
5. Korelacje kanoniczną (ang. *Canonical Correlation* A*nalysis, CCA*) [194].
6. Statystyki wyższych rzędów (ang. *Higher Order Statistics, HOS*) [195], np. wariancja, skośność (miara asymetrii), kurtoza (współczynnik spłaszczenia) itd.

Cechy wyekstrahowane z uprzednio zarejestrowanego i oczyszczonego sygnału EEG są przekazywane do modułu klasyfikacji, którego zadaniem jest wyznaczenie klasy odpowiadającej danemu zestawowi cech. Każda klasa odpowiada jednej komendzie sterującej bądź jednemu fragmentowi tworzonej wiadomości. W zależności od obszaru stosowania interfejsu klasy mogą odpowiadać: kierunkom ruchu robota (prawo, lewo, góra, dół), komendom dla wózka inwalidzkiego (jedź prosto, obróć się w lewo, stój), przyciskom w przeglądarce internetowej, cyfrom bądź literom na klawiaturze, odpowiedziom tak/nie, itd. Stąd po rozpoznaniu klasy następuje wykonanie przypisanej do niej komendy.

Naturalnie, żeby klasyfikator mógł wykonywać postawione mu zadanie, musi zostać odpowiednio skonstruowany. Klasyfikatory wykorzystywane w interfejsach opartych na wyobrażeniu ruchu, wymagają najczęściej adaptacji do aktywności mózgowej konkretnego użytkownika interfejsu, stąd do ich prawidłowej budowy wymagane jest wcześniejsze przeprowadzenie treningu wybranego modelu klasyfikatora.

Trening ten wymaga uruchomienia interfejsu w drugim z wspomnianych wcześniej trybów, trybie uczącym.

Moduł przetwarzania sygnału wykorzystywany w trybie uczenia obejmuje cztery podmoduły. Dwa z czterech są identyczne, jak w przypadku, kiedy interfejs działa w trybie użytkowania. Są to moduł wstępnego przetworzenia sygnału oraz moduł ekstrakcji cech. Poza nimi w trybie uczenia wykorzystywany jest dodatkowo moduł selekcji cech oraz moduł trenowania klasyfikatora. W zależności od stosowanej metody selekcji te dwa ostatnie moduły albo działają odrębnie (najpierw moduł selekcji cech, później moduł trenujący) albo symultanicznie, tzn. moduł trenowania klasyfikatora stanowi wewnętrzny element modułu selekcji cech.

Klasyfikator jest algorytmem, którego zadaniem jest ustalenie do której ze zdefiniowanych wcześniej klas należy przypisać aktualnie zarejestrowane obserwacje, opierając się na cechach (atrybutach) tych obserwacji. Aby klasyfikator mógł wykonywać poprawnie swoje zadanie, trzeba go najpierw zdefiniować korzystając ze zbioru obserwacji $x \in X$ oraz przypisanych do nich klas $y \in Y$, charakteryzującego rozwiązywany problem. W przypadku interfejsu BCI, x_i to wektor cech opisujących daną obserwację ($i=1...m$, gdzie m – liczba cech), a y to jedna z ustalonych wcześniej aktywności mózgowych wykorzystywanych do sterowania. Zbiór danych treningowych wymaganych na etapie definiowania parametrów klasyfikatora opisywany jest jako: $\{(x_1, y), (x_2, y), (x_3, y), \dots, (x_n, y)\}$.

Aby wytrenować klasyfikator, konieczne jest więc uprzednie pozyskanie danych. Dane te uzyskuje się w trakcie sesji uczącej, kiedy użytkownik wykonuje zdefiniowane wcześniej zadania. W przypadku interfejsu korzystającego z potencjałów motorycznych zadaniem użytkownika w trakcie sesji treningowej jest wyobrażanie sobie ruchu kolejno wskazywanymi częściami ciała (np. prawą i lewą ręką). Zestawy sygnałów zebranych w trakcie sesji treningowej (wraz z odpowiadającymi im etykietami klas opisującymi aktywność mózgową, która powinna zostać rozpoznana w każdym kolejnym sygnale) są wykorzystywane następnie w celu treningu klasyfikatora stanowiącego podstawę interfejsu.

Przeważającą grupę klasyfikatorów wykorzystywanych w interfejsach mózg–komputer opartych na wyobrażeniu ruchu stanowią klasyfikatory liniowe, a wśród nich przede wszystkim:

- Klasyfikator LDA (ang. *Linear Discriminant Analysis*) – Liniowa Analiza Dyskryminacyjna [196].

- Klasyfikator QDA (ang. *Quadratic Discriminant Analysis*) – Kwadratowa Analiza Dyskryminacyjna [197].
- Klasyfikator SVM (ang. *Support Vector Machine*) – Maszyna Wektorów Wspierających – z liniową funkcją jądra [198].

Znacznie rzadziej natomiast są stosowane klasyfikatory nieliniowe, takie jak:

- Klasyfikator SVM (ang. *Support Vector Machine*) – Maszyna Wektorów Wspierających – z nieliniową funkcją jądra [198].
- Klasyfikator MLP (ang. *Multilayer Perception*) – Sieci Neuronowe [199].
- Klasyfikator k–nn (ang. *k nearest neighbours)* – klasyfikator *k* najbliższych sąsiadów [200].

W przypadku gdy zestaw cech (o potencjalnie wysokich zdolnościach dyskryminacyjnych) wyekstrahowanych z sygnału EEG jest zbyt duży, aby zapewnić odpowiednio szybkie działanie interfejsu w trybie użytkowania, zestaw ten należy ograniczyć poprzez wybór cech najsilniej dyskryminujących zadane klasy. Do wykonania tego zadania służy ostatni z podmodułów modułu przetwarzającego – moduł selekcji cech. W module tym wykorzystywane są algorytmy, które mają za zadanie przeszukać przestrzeń potencjalnych cech i wskazać wśród nich te cechy, które pozwolą na otrzymanie możliwie najdokładniejszego rozwiązania postawionego problemu klasyfikacyjnego w założonym czasie. Wśród metod wykorzystywanych w niniejszym module można wymienić między innymi:

- Algorytm GA – *Genetic Algorithm* [4,22,40,69,83,90,94,95]*;*
- Algorytm FS – *Forward Selection* [61,69-78]*;*
- Algorytm BS – *Backward Selection* [61,69,7173,77,79]*;*
- Algorytm SFS – *Sequence Forward Selection* [61,69-78,95]*;*
- Algorytm BDS – *BiDirectional Selection* [61,69,71,73]*;*
- Algorytm SBS–SLASH – *Sequential Backward Selection SLASH* [287,309-310]*;*
- Algorytm LVW – *Las Vegas Wrapper* [287,311-312]*;*
- Algorytm LVF – *Las Vegas Filter* [287,311-312]*;*
- Algorytm *ReliefF* [38,42,51-60]*;*
- Algorytm IG – *Information Gain* [38,329-348]*;*
- Algorytm CFS – *Correlation based Feature Selection* [4,38,42-50]*;*
- Algorytm FCBF – *Fast Correlation–Based Filter* [42]*;*

- Algorytm CBFS – *Consistency–Based Feature Selection* [38,60,323,329,352-362]*;*
- Algorytm LASSO – *Least Absolute Shrinkage and Selection Operator* [61-68]*;*
- Algorytm *Elastic Net* [286,372-375];
- Algorytm RR – *Ridge Regression* [286,379-381]*;*
- Algorytm SVM–RFE – *Support Vector Machine – Recursive Feature Elimination* [198,307,367-369,372,376-378]*;*

Większość z wymienionych powyżej metod zostanie omówiona w dalszej części rozprawy.

Rozdział II
Proces optymalizacji

Problem selekcji cech na potrzeby procesu klasyfikacji należy rozpatrywać jako problem optymalizacyjny. Można zdefiniować go jako zadanie znalezienia minimalnego zbioru cech pozwalającego na zdefiniowanie klasyfikatora o dokładności satysfakcjonującej użytkownika. Stąd niniejszy rozdział rozpoczyna się od omówienia problemu optymalizacji jedno i wielokryterialnej. Następnie przedstawione zostały podstawowe metody optymalizacyjne, z podziałem na metody deterministyczne i stochastyczne. Oddzielny podrozdział poświęcono dokładniejszemu omówieniu zagadnieniu optymalizacji wielokryterialnej, skupiając się na rozwiązaniach pareto-optymalnych i na problemie dominacji jednych rozwiązań nad innymi.

2.1 Optymalizacja jedno i wielokryterialna

Optymalizacja [201-202] jest metodą ukierunkowaną na poszukiwanie najlepszego, z punktu widzenia przyjętego kryterium, rozwiązania postawionego zadania. Kryterium, zgodnie z którym prowadzony jest proces poszukiwania rozwiązania optymalnego może być np. cena, ilość, dokładność rozwiązania itd. Bardziej formalnie zadanie optymalizacyjne można zdefiniować jako zadanie polegające na znalezieniu takich wartości zmiennych decyzyjnych, dla których funkcja celu osiąga minimum lub maksimum w zbiorze dopuszczalnym, przy czym zbiór dopuszczalny to zbiór możliwych rozwiązań wyznaczony przez ograniczenia nałożone na zmienne decyzyjne [203]. Należy tu zauważyć, że przyjęcie złych lub niekompletnych założeń/ograniczeń może doprowadzić do uzyskania błędnego rozwiązania [204].

W ujęciu matematycznym celem procesu optymalizacji jest znalezienie najlepszych, z punktu widzenia funkcji celu F (wzór 2.1), elementów x^* z całego zbioru X [205]:

$$F = \{f_i : X \to Y_i : 0 < i \leq n, Y_i \in R\} \tag{2.1}$$

gdzie [205]:
F – funkcja celu (wyrażenie matematyczne lub złożony algorytm);
n – liczba kryteriów (dla algorytmów optymalizacji jednokryterialnej $n = 1$; dla algorytmów optymalizacji wielokryterialnej $n > 1$).

Jednokryterialna funkcja celu (inaczej funkcja kryterialna, wskaźnik jakości, kryterium decyzyjne) przyjmuje postać (wzór 2.2):

$$F = f(x_1, x_2, \dots, x_n) \tag{2.2}$$

W procesie optymalizacji wyznaczane są wartości zmiennych decyzyjnych, dla których funkcja celu będzie osiągać wartość maksymalną (wzór 2.3):

$$f(x_1, x_2, \dots, x_n) \to max \tag{2.3}$$

bądź minimalną (wzór 2.4):

$$f(x_1, x_2, \dots, x_n) \to min \tag{2.4}$$

Zmienne decyzyjne $x_1, x_2, \dots, x_n$ mogą posiadać ograniczenia, np. warunki brzegowe, np. $x_j \leq 100$ dla $j=1,\dots,n$.

Jeżeli funkcja celu jest funkcją jednej zmiennej, wówczas jest to najprostsze do rozwiązania zadanie optymalizacyjne, ponieważ dzięki teoretycznej i praktycznej znajomości zasad optymalizacji zadań z jedną zmienną opracowano już bardzo wiele algorytmów, które można wykorzystać do rozwiązania tego zadania. Poszczególne metody różnią się od siebie pod względem stawianych hipotez oraz przypuszczeń w stosunku do właściwości i postaci badanej funkcji celu [206]. Im więcej zmiennych występuje w optymalizowanej funkcji, tym zadanie optymalizacji staje się trudniejsze. Nawet przy jedynie kilku zmiennych może nie być już możliwości analitycznego wyznaczenia rozwiązania.

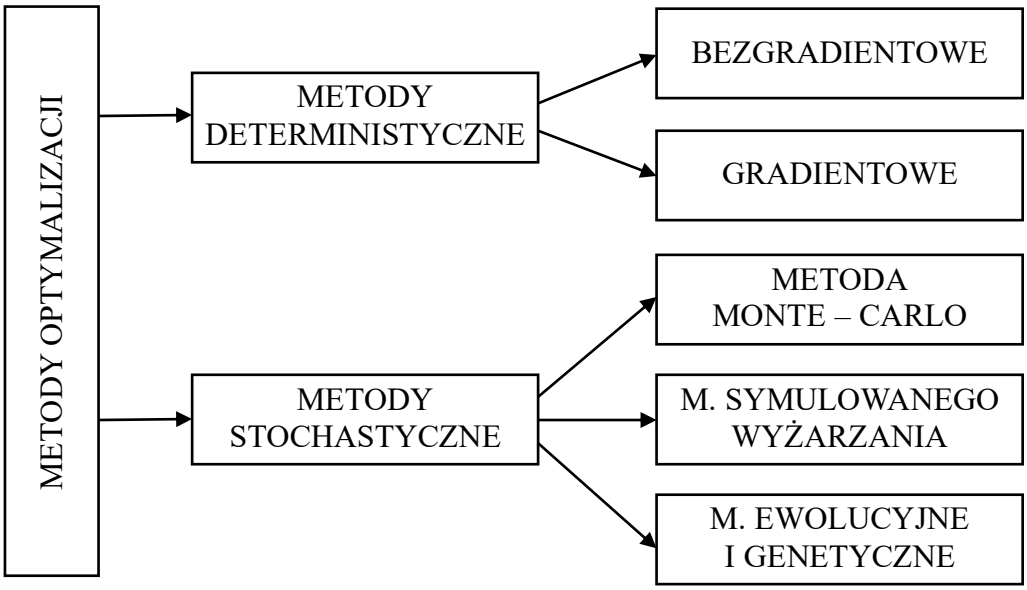
Poza podziałem zadań optymalizacyjnych na zadania z jedną i wieloma niewiadomymi (zmiennymi decyzyjnymi) można wyróżnić wiele innych podziałów. Wśród najczęściej spotykanych należałoby wymienić [203]:

1. Ze względu na występowanie ograniczeń:
    a. zadanie z ograniczeniami – zmienne decyzyjne muszą przyjmować wartości należące do zbioru dopuszczalnego;
    b. zadanie bez ograniczeń – zmienne decyzyjne mogą przyjmować dowolne wartości.
2. Ze względu na typ funkcji celu (oraz ograniczeń):
    a. zadanie optymalizacji liniowej – zarówno funkcja celu, jak i ograniczenia są liniowymi kombinacjami zmiennych decyzyjnych;
    b. zadanie optymalizacji nieliniowej – przynajmniej jedna spośród funkcji występujących w zadaniu (ograniczenia, funkcja celu) jest nieliniowa względem zmiennych decyzyjnych.
3. Za względu na dopuszczalne wartości zmiennych decyzyjnych:
    a. zadanie optymalizacji w zbiorze ciągłym – zmienne decyzyjne mogą przyjmować dowolne wartości ze zbioru dopuszczalnego (np. wartości rzeczywiste);
    b. zadanie optymalizacji w zbiorze dyskretnym – zmienne decyzyjne muszą przyjmować wartości ze zbioru dyskretnego.
4. Ze względu na liczbę funkcji celu:
    a. zadanie jednokryterialne – w zadaniu występuje tylko jedna funkcja celu (optymalizacja przebiega więc według jednego kryterium);
    b. zadanie wielokryterialne – w zadaniu występuje wiele funkcji celu (optymalizacja przebiega z uwzględnieniem wszystkich funkcji celu jednocześnie); z uwagi na to, że wyboru dokonuje się tu na podstawie co najmniej dwóch kryteriów, często istnieje konieczność uwzględnienia preferencji decydenta dotyczących poszczególnych kryteriów.

Istnieje olbrzymia różnorodność metod prowadzenia procesu optymalizacji, przy czym większość z nich w sposób bezpośredni odnosi się do rozwiązywania problemów jednokryterialnych. Jak pisze J. Stadnicki najczęściej pierwotne zadanie optymalizacji wielokryterialnej jest zastępowane zadaniem albo ciągiem zadań jednokryterialnych,

które rozwiązuje się za pomocą jednego z algorytmów optymalizacji jednokryterialnej [203].

W sposób najbardziej ogólny ogół metod optymalizacyjnych można podzielić [7] na metody deterministyczne i stochastyczne (Rysunek 2.1). Wśród metod deterministycznych wyróżnia się metody bezgradientowe (m.in. złotego podziału) oraz metody gradientowe [207-214] (m.in. metoda gradientu prostego, metoda największego spadku, metoda gradientu sprzężonego). Do metod stochastycznych zalicza się m.in. metodę Monte-Carlo, metodę symulowanego wyżarzania, metody ewolucyjne i genetyczne itp.



Rysunek 2.1. Podział metod optymalizacyjnych.
Źródło: [215]

2.2 Deterministyczne metody optymalizacji

Metody deterministyczne są metodami matematycznymi, w których nie występują żadne zmienne (elementy) losowe, a danemu na wejściu zdarzeniu jednoznacznie przypisuje się konkretny stan tj. wszystkie informacje o właściwościach sytemu i otoczenia są pewne. Termin „*deterministyczny*" jest terminem, który oznacza, „*że każde zdarzenie ma przyczynę, a wszelkie zjawiska zachodzą zgodnie z określonymi prawami*" [216]. Jego „*idea zakłada (...) stały porządek w związkach zachodzących między zjawiskami*" [217]. Termin „*deterministyczny*" wynika od „*determinizmu*" [218],

który jest „*koncepcją rzeczywistości, uznającą zasadę przyczynowego uwarunkowania wszystkich zjawisk*” [219-220]. Zgodnie z koncepcją determinizmu „*zajście jednego zjawiska pociąga za sobą powstanie innego zjawiska (tzw. związek przyczynowo – skutkowy). Na podstawie związku przyczynowego i znajomości przyczyny można przewidywać skutek, a ze znajomości skutku można wnioskować o przyczynie*” [218]. Istotne jest również to, że ewolucja układu w modelu deterministycznym jest z góry przesądzona i zależy wyłącznie od parametrów początkowych lub ich wartości poprzednich [221].

Metody deterministyczne służą do badania związków przyczynowo-skutkowych i są bardzo często wykorzystywane w procesie optymalizacji systemu przedstawionego w postaci układu równań różniczkowych bądź różnicowych [222]. Przykładem jest określenie czasu podróży do innego miasta na podstawie znajomości odległości do tego miasta i ograniczeń prędkości występujących na różnych trasach prowadzących do tego miasta.

Metody deterministyczne są przeciwieństwem metody stochastycznej, która jest metodą probabilistyczną (losową) ze zmienną czasową, gdzie wyniki końcowe opisywane są tylko z pewnym prawdopodobieństwem. Porównania wielu deterministycznych metod optymalizacyjnych dokonano w [223]. Poniżej przedstawiono przykłady metod deterministycznych:

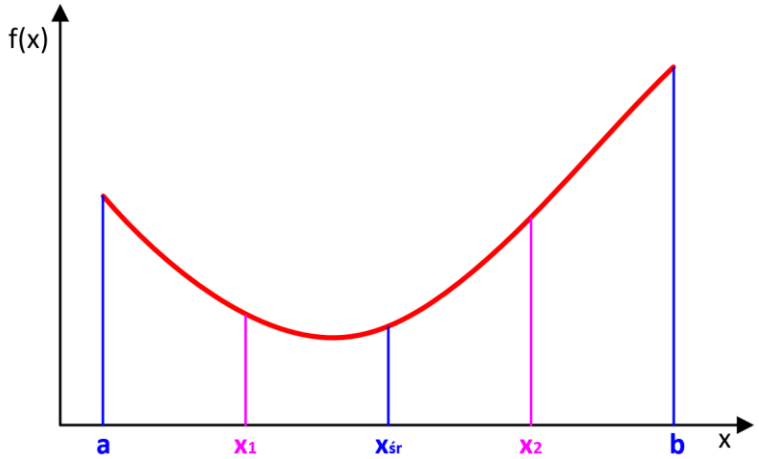
1. Metody bezgradientowe optymalizacji funkcji jednokryterialnej
    a. Metoda dzielenia przedziału na połowę [206];
    b. Metoda siecznych [206];
    c. Metoda złotego podziału [206];
    d. Metoda Fibonacciego [206];
    e. Metoda interpolacji kwadratowej [206].
2. Metody bezgradientowe optymalizacji funkcji wielokryterialnej
    a. Metoda simpleks [206];
    b. Metoda pełzającego simpleksu Neldera–Meada [206,214,224];
    c. Metoda Hooke’a–Jeevesa [206,214];
    d. Metoda Gaussa–Seidela [214];
    e. Metoda Powella [206,214];
    f. Metoda Rosenbrocka.
3. Metody gradientowe
    a. Metoda gradientu prostego [214];

b. Metoda największego spadku (Metoda Cauchy'ego) [206,214];
c. Metoda gradientów sprzężonych [206].
d. Metoda zmiennej metryki [206];
e. Metoda Newtona [206].

Główną zaletą metod deterministycznych jest ich prosta technika obliczeniowa (tj. multiplikatywna bądź addytywna formuła). Do ich wad można natomiast zaliczyć nieuwzględnianie czynników losowych. Poniżej opisano po jednym reprezentancie trzech wskazanych powyżej grup metod deterministycznych.

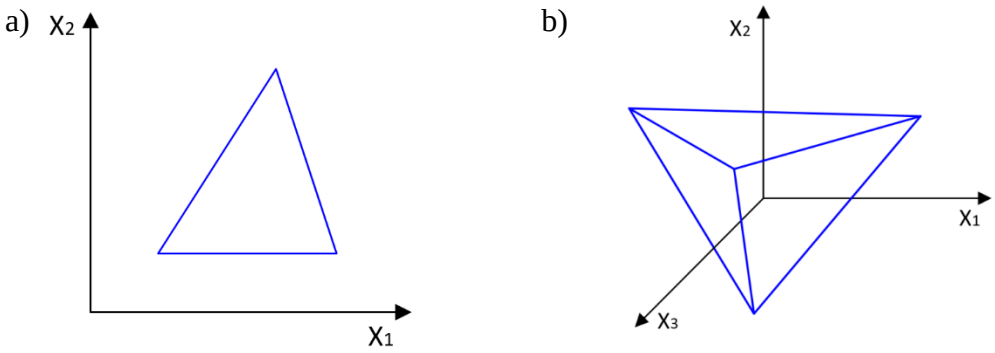
Metoda dzielenia przedziału na połowę (ang. *Interval Halving Method, Dichotomous Search Method*) (*Metoda równego podziału, Metoda połowienia, Metoda bisekcji, Metoda połowienia przedziału*) [206,225-229] należy do grupy metod poszukiwania ekstremum w zadanym przedziale.

W metodzie tej dany jest przedział (a, b) [225,230-231]. Z przedziału wybierane są trzy punkty $x_1, x_{śr}, x_2$ przy czym $x_{śr}$ jest punktem środkowym przedziału, punkt x_1 znajduje się dokładnie pośrodku pomiędzy punktem a i punktem $x_{śr}$, a punkt x_2 znajduje się dokładnie pośrodku pomiędzy punktem $x_{śr}$ i punktem b (Rysunek 2.2). Obliczana jest wartość funkcji w wyznaczonych punktach, dzięki czemu w każdej iteracji obszar poszukiwań minimum jest zmniejszany o połowę. Punkt środkowy wyznaczony w kolejnej iteracji zawsze jest jednym z punktów x_1 lub x_2, wyznaczonych we wcześniejszej iteracji. Stosując metodę dzielenia przedziału na połowę eliminuje się w każdej iteracji dokładnie pół przedziału. Metoda jest czasami nazywana metodą trzypunktowego poszukiwania w równych przedziałach, ponieważ wybiera się trzy próbne punkty, które są równomiernie rozłożone w przedziale poszukiwań.



Rysunek 2.2. Metoda dzielenia przedziału na połowę.
Źródło: Opracowanie własne na podstawie [225].

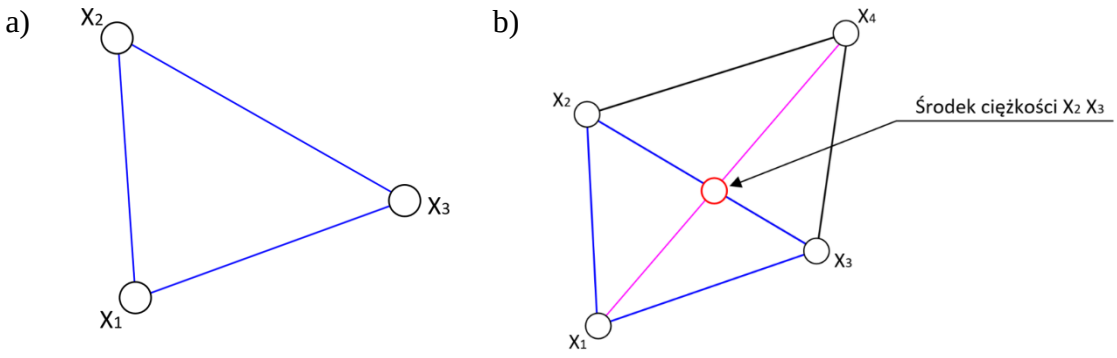
Metoda simpleks polega na znalezieniu rozwiązania zagadnienia optymalizacyjnego wśród wierzchołków wieloboku utworzonego przez nałożone na zadanie ograniczenia i warunki brzegowe. Jest stosowana do zadań z małą ilością zmiennych decyzyjnych. Metoda simpleks w procesie optymalizacji wykorzystuje modele nazywane simpleksami. Regularny simpleks to model zawierający najmniejszą liczbę punktów; regularny simpleks w N-wymiarowej przestrzeni jest wielościanem, który jest zobrazowany przez $N + 1$ równo odległych od siebie punktów (wierzchołków). Przykładowo, dla dwóch zmiennych (przestrzeń dwuwymiarowa) simpleks jest trójkątem równobocznym, a dla trzech zmiennych (przestrzeń trójwymiarowa) simpleks jest czworościanem (Rysunek 2.3) [206] .



Rysunek 2.3. Simpleks dla dwóch (a) i trzech (b) zmiennych.
Źródło: Opracowanie własne.

W algorytmie simpleks w pierwszym etapie poszukiwania optimum funkcji celu buduje się regularny simpleks w przestrzeni niezależnych zmiennych, ocenia się wartości funkcji celu w każdym z wierzchołków simpleksu oraz określa się wierzchołek, który posiada najwyższą wartość funkcji celu. Następnie wierzchołek ten przekształca się z wykorzystaniem środka ciężkości pozostałych wierzchołków simpleksu w nowy wierzchołek, który zostanie wykorzystany w nowo tworzonym simpleksie. Każdy nowy simpleks może zostać zbudowany na dowolnej krawędzi simpleksu początkowego, poprzez przeniesienie wybranego wierzchołka na odpowiednią odległość w dół wzdłuż prostej przechodzącej przez środek ciężkości pozostałych wierzchołków simpleksu początkowego. Nowy punkt otrzymany w ten sposób staje się wierzchołkiem nowego simpleksu, a wybrany podczas budowy wierzchołek simpleksu początkowego jest odrzucany. Przy przejściu do nowego simpleksu konieczne jest obliczenie tylko jednej wartości funkcji celu – dla nowo zdefiniowanego wierzchołka. Na Rysunku 2.4

przedstawiono proces budowy nowego simpleksu na płaszczyźnie dwuwymiarowej [206].



Rysunek 2.4. Budowa nowego simpleksu na płaszczyźnie dwuwymiarowej: (a) początkowy simpleks X_1, X_2, X_3, (b) nowy simpleks X_2, X_3, X_4.
Źródło: Opracowanie własne na podstawie [206]

Iteracje zmierzające do obliczania nowego simpleksu przeprowadzane są do czasu osiągnięcia punktu minimum lub do rozpoczęcia cyklicznego ruchu w co najmniej dwóch simpleksach. W przypadku wystąpienia jednej z powyższych sytuacji wykorzystuje się odpowiednią z poniższych reguł [206]:

Reguła 1. „Objęcie" punktu minimum. Zamiast wierzchołka, który był znaleziony w poprzedniej iteracji, a któremu odpowiadała największa wartość funkcji celu, bierze się wierzchołek, któremu odpowiada kolejna (ze względu na wielkość wartości) funkcja celu [206].

Reguła 2. Ruch cykliczny. Jeżeli któryś z wierzchołków simpleksu nie jest odrzucany w trakcie M iteracji, wówczas konieczne jest zmniejszenie rozmiaru simpleksu za pomocą współczynnika redukcji. Kolejnym krokiem jest zbudowanie nowego simpleksu poprzez wybór tego punktu, jako punktu bazowego, któremu odpowiada minimalna wartość funkcji celu. Niezbędne jest ustalenie wielkości współczynnika redukcji. Autor w [206] proponuje obliczenie liczby iteracji jako (wzór 2.5):

$$M = 1.65 \cdot N + 0.05 \cdot N^2 \tag{2.5}$$

gdzie:

M – liczba iteracji (zaokrągla się do wartości całkowitej);

N – wymiar przestrzeni.

Reguła 3. Kryterium zakończenia poszukiwań. Zakończenie poszukiwań następuje w momencie, gdy różnice pomiędzy wartościami funkcji lub rozmiar simpleksu są dostatecznie małe. Konieczne jest ustalenie wielkości parametru zakończenia poszukiwań.

Poniżej przedstawiono algorytm metody simpleks S^2 [206]:

Dane:

f(x) – minimalizowana funkcja wielu zmiennych, gdzie $x = [x_1, x_2, x_3, \dots, x_{N-1}, x_N]$;

N – wymiar przestrzeni;

$x^{(0)}$ – punkt startowy (bazowy) (dowolny);

α – współczynnik kroku;

m – liczba iteracji;

ε – wymagana dokładność.

1. Utworzyć w przestrzeni regularny simpleks, poprzez obliczenie współrzędnych wierzchołków według wzoru 2.6:

$$x_j^{(i)} = \begin{cases} x_j^{(i)} + \delta_1, \text{ gdy } i \neq j, \\ x_j^{(i)} + \delta_2, \text{ gdy } i = j, \end{cases} \tag{2.6}$$

Przyrosty δ_1 i δ_1 zależą tylko od N i wybranego współczynnika kroku, przy $\alpha = const.$ (wzór 2.7 i 2.8):

$$\delta_1 = \left[\frac{(N+1)^{\frac{1}{2}} + N - 1}{N \cdot \sqrt{2}}\right] \cdot \alpha \tag{2.7}$$

$$\delta_2 = \left[\frac{(N+1)^{\frac{1}{2}} - 1}{N \cdot \sqrt{2}}\right] \cdot \alpha \tag{2.8}$$

Dla $\alpha = 1$ żebra regularnego simpleksu mają długość jednostkową.

2. Obliczyć wartość funkcji celu dla poszczególnych wierzchołków.
3. Sprawdzić, czy odległość pomiędzy wierzchołkami simpleksu nie jest większa niż przyjęta dokładność ε. Jeżeli tak, wówczas następuje koniec obliczeń.
4. Znaleźć wierzchołek x_m, dla którego wartość funkcji celu jest największa $f(x_m)$. Jeżeli znaleziony punkt będzie punktem uzyskanym z ostatniej iteracji, wówczas należy wziąć punkt następny.
5. Znaleźć środek symetrii (wzór 2.9).

$$\mathrm{x_c} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i}}{\mathrm{N}} \tag{2.9}$$

$x^{(j)}$ – punkt poddawany odbiciu

x_c – środek ciężkości pozostałych N-punktów.

6. W celu uzyskania nowego simpleksu, wybrany punkt x_m nalezy odbić względem obliczonego środka symetrii (wzór 2.10):

$$\bar{\mathrm{x}} = \mathrm{x_m} + \lambda(\mathrm{x_c} - \mathrm{x_m}) \tag{2.10}$$

Dla $\lambda = 0$ – otrzymujemy punkt wyjściowy $x^{(j)}$;

Dla $\lambda = 1$ – środek ciężkości wynosi x_c.

Otrzymuje się nowy punkt, w którym wartość funkcji będzie mniejsza niż w poprzednim punkcie, otrzymując zarazem nowy simpleks.

7. Kontynuując proces, simpleks przesuwa się coraz bliżej minimum.
   Sprawdzić, czy istnieje wierzchołek, który nie został odrzucony w k kolejnych iteracjach. Jeżeli go nie ma, przejść do kroku 3. W przeciwnym razie zmniejszyć rozmiar simpleksu i zbudować nowy simpleks, przyjmując jako punkt bazowy ten punkt, w którym wartość funkcji jest najmniejsza. Należy określić nową bazę, obliczyć wartości funkcji dla nowych wierzchołków i przejść do kroku 3.
8. Obliczenia kończą się wówczas, gdy odległość pomiędzy wierzchołkami simpleksu w pobliżu poszukiwanego minimum będzie mniejsza lub równa od założonej dokładności ε.

Metoda jest bardzo często zbieżna po m (do $2 \cdot m$) krokach, dzięki systematycznemu zmniejszaniu wartości funkcji w każdym kroku.

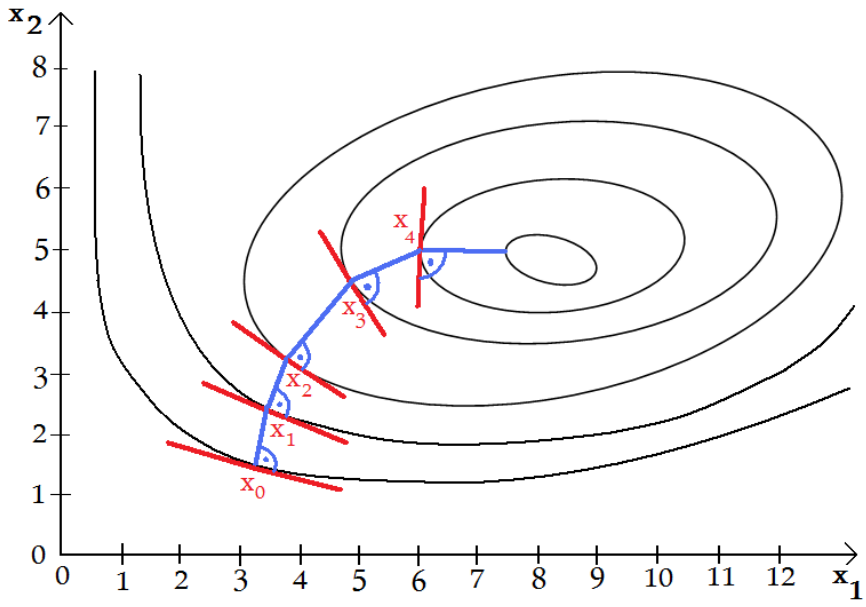
Metody gradientowe poszukiwania ekstremum funkcji wykorzystują informacje zarówno o wartości funkcji jak i o wartości jej gradientu. Ich zasada działania opiera się na wyznaczaniu kolejnego kierunku poszukiwań na podstawie znajomości gradientu funkcji celu w punkcie wyznaczonym w poprzednim kroku [232-233]. Zaletą metod gradientowych jest szybsza zbieżność w porównaniu do metod bezgradientowych. Ich podstawową wadą jest natomiast to, że można je stosować tylko przy znajomości gradientu funkcji, co oznacza, że funkcja celu musi posiadać postać analityczną oraz musi być ciągła i różniczkowalna, a także musi być wypukła w badanej dziedzinie.

Proces poszukiwania optimum funkcji celu w metodach gradientowych jest procesem iteracyjnym. W kolejnych iteracjach wykonywane są czynności zmierzające do przesunięcia bieżącego punktu x_i do kolejnego punktu x_{i+1}. Czynności te obejmują wyznaczenie kierunku poszukiwań oraz określenie minimum w poszukiwanym kierunku [232-233]. W metodach zaliczanych do tej grupy algorytm określania minimum w poszukiwanym kierunku jest identyczny, a różnice polegają tylko na różnym sposobie wyznaczania kierunków poszukiwań. Poniżej dla przykładu opisano jedną z najprostszych metod gradientowych – metodę gradientu prostego.

Zasada działania metody gradientu prostego oparta jest na wykonywaniu kroków o założonej długości w kierunku, który został określony przez wartość gradientu funkcji celu w kolejno wyznaczanych punktach [232-233]. (Rysunek 2.5). Aktualny kierunek poszukiwań jest wyznaczany w bieżącym punkcie (kierunek ma wartość ujemnego gradientu w tym właśnie punkcie). Po wyznaczeniu kierunku wykonywany jest krok o założonej długości „ε" wzdłuż tego kierunku. Procedura wykonywana jest do czasu spełnienia założonego wcześniej kryterium (tj. do czasu osiągnięcia minimum wyznaczonej funkcji celu) [232-233].

Założenia początkowe algorytmu:

- d – początkowa długość kroku;
- x_0 – wybrany punkt startowy;
- $\beta \in (0,1)$ – współczynnik korekcyjny długości kroku;
- ε - założona dokładność obliczeń minimum.



Rysunek 2.5. Poszukiwanie minimum metodą gradientu prostego.

Źródło: Opracowanie własne.

Algorytm obliczeń [232-233].

1. Znalezienie punktu startowego algorytmu x_0 w przestrzeni zmiennych decyzyjnych
2. Obliczenie wartości funkcji celu F_0 oraz jej gradientu g_0w punkcie x_0 (wzór 2.11 i 2.12):

$$F_0 = f(x_0) \tag{2.11}$$

$$g_0 = g(x_0) \tag{2.12}$$

3. Wyznaczenie kierunku poszukiwań ξ (wzór 2.13):

$$\xi = -g_0 \tag{2.13}$$

4. Wykonanie kroku o długości d wzdłuż kierunku poszukiwań ξ i określenie współrzędnych nowego punktu x_{i+1} (wzór 2.14).

$$x_{i+1} = x_i - d \cdot \xi \tag{2.14}$$

(dla pierwszej iteracji $x_i = x_0$)

5. Obliczenie wartości funkcji F oraz jej gradientu g w punkcie x_{i+1} (wzór 2.15 i 2.16):

$$F = f(x_{i+1}) \tag{2.15}$$

$$g = g(x_{i+1}) \tag{2.16}$$

6. Sprawdzenie czy (wzór 2.17):

$$F > F_0 \tag{2.17}$$

   - ✓ Jeżeli NIE → idź do punktu 3, gdzie w miejsce początkowej wartości gradientu g_0 wstaw aktualną wartość gradientu g
   - ✓ Jeżeli TAK → idź do punktu 7.

7. Zbadanie, czy zostało osiągnięte minimum z założoną dokładnością ε
   - ✓ Jeżeli NIE →
     a) zmniejsz długość kroku o β
     b) idź do punktu 4
   - ✓ Jeżeli TAK → Koniec algorytmu

Jedną z szerzej znanych modyfikacji metody gradientu prostego jest metoda najszybszego spadku (Metoda Cauchy'ego). Podstawowa różnica między tymi metodami tkwi w sposobie obliczania długości kolejnych kroków. W metodzie gradientu prostego bardzo często kolejne kroki przyjmują jedną stałą wartość ustaloną na starcie algorytmu (wartość ta jest czasami modyfikowana po przekroczeniu minimum funkcji celu). Z kolei w metodzie najszybszego spadku długość kroku jest zmienna i jest obliczana poprzez wykonywanie minimalizacji kierunkowej funkcji względem wartości d (wzór 2.18):

$$\min_{d>0} f(x_i - dg(x_i)) \tag{2.18}$$

2.3 Stochastyczne metody optymalizacji

Termin „stochastyczny" jest terminem heurystycznym oznaczającym dosłownie „*losowy*" („*statystyczny*", „*przypadkowy*") [221,234-235]. Z kolei określenie „*losowy*" oznacza „*związany z przypadkowością, brakiem możliwości przewidzenia rezultatu, wyniku*" [218]. Jako że termin „*stochastyczny*" nie jest terminem precyzyjnym, nie ma

żadnego uniwersalnego testu bądź miary samej „przypadkowości” lub „losowości”, stąd też określenie „stochastyczny” jest bardzo często wykorzystywane w codziennym języku [236].

Stochastyczne metody optymalizacji często znajdują zadowalające rozwiązania, gdy zawodzą metody deterministyczne albo gdy nie mogą one zostać zastosowane z uwagi na zbyt dużą liczbę zmiennych decyzyjnych lub zbyt wysoki stopień skomplikowania funkcji celu oraz ograniczeń nałożonych na zmienne decyzyjne [205]. Wśród najczęściej wykorzystywanych stochastycznych metod optymalizacyjnych można wymienić m.in.:

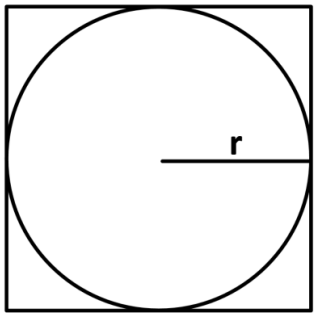
1. Metodę Monte Carlo [237];
2. Metodę symulowanego wyżarzania [238,241-243];
3. Algorytmy genetyczne i ewolucyjne [89;214,231];
4. Strategie ewolucyjne [30];
5. Algorytmy mrówkowe [85].

Poniżej omówiono dwie wybrane stochastyczne metody optymalizacyjne: metodę Monte Carlo oraz metodę symulowanego wyżarzania.

Metoda **Monte Carlo** służy do matematycznego modelowania procesów zbyt złożonych (np. obliczania całek), aby można było wyznaczyć ich wyniki za pomocą podejścia analitycznego [237]. Metoda ta może być stosowana wszędzie tam, gdzie badane zagadnienie można opisać teoretycznie w ujęciu stochastycznym, chociaż samo zagadnienie może mieć przy tym charakter ściśle deterministyczny. Zaletą metody Monte Carlo jest możliwość zastąpienia skomplikowanych obliczeń analitycznych prostym sposobem rozwiązania zadania oraz rozwiązania trudnych zadań i problemów. Metoda ta uwalnia od skomplikowanych wzorów oraz teorii i pozwala skupić się na istocie problemu.

Sposób postępowania w przypadku metody Monte Carlo zostanie przybliżony na przykładzie zadania polegającego na obliczeniu pola koła o promieniu *r* i środku w punkcie (0, 0) (Rysunek 2.6), zdefiniowanego nierównością (wzór 2.19):

$$x^2 + y^2 \leq r^2 \tag{2.19}$$



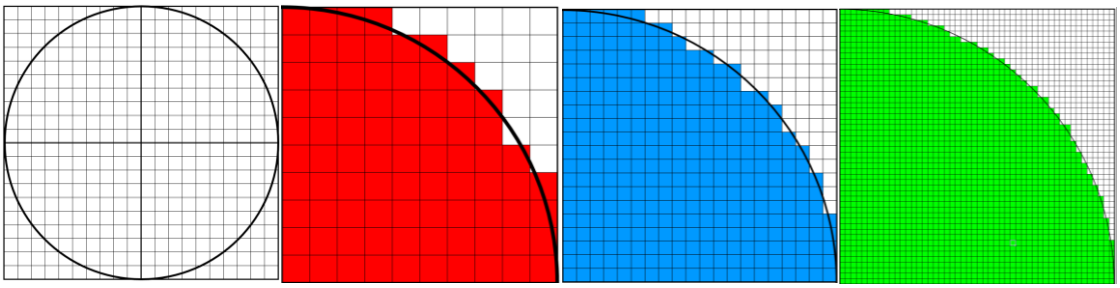
Rysunek 2.6 Koło o promieniu *r* i środku w punkcie (0, 0) wpisane w kwadrat.
Źródło: Opracowanie własne

Zasadniczym elementem metody Monte Carlo [237] jest losowanie (zgodnie ze znanym rozkładem) wielu wartości zmiennych charakteryzujących dany proces. Stąd algorytm rozwiązania postawionego zadania przybiera postać:

1. Wylosuj *n*-punktów z kwadratu opisanego na kole.
2. Dla każdego z punktów sprawdź, czy współrzędne punktu należą do koła, zatem czy spełniają nierówność (2.19).
3. Zlicz liczbę punktów należących do koła (*m*).
4. Oblicz stosunek liczby punktów należących do koła do liczby wszystkich punktów (*n*).
5. Oblicz pole koła (P_k) korzystając z wyznaczonego stosunku (*m/n*) oraz pola kwadratu opisanego na okręgu (*P*) (wzór 2.20):

$$\mathrm{P_k} = \mathrm{P}\frac{\mathrm{m}}{\mathrm{n}} \tag{2.20}$$

Dokładność wyniku w metodzie Monte Carlo zależy od liczby losowań oraz rodzaju wykorzystanego generatora liczb pseudolosowych [238] [239] [240]. Ważna jest także rozdzielczość losowania. Im większa rozdzielczość, tym można uzyskać dokładniejsze przybliżenie rozwiązania (Rysunek 2.7). Nie należy jednakże zapominać, że metoda Monte-Carlo jest metodą losową, zatem otrzymane wyniki mogą być czasami gorsze, a czasami lepsze, w zależności od uruchomienia i wylosowanych liczb.

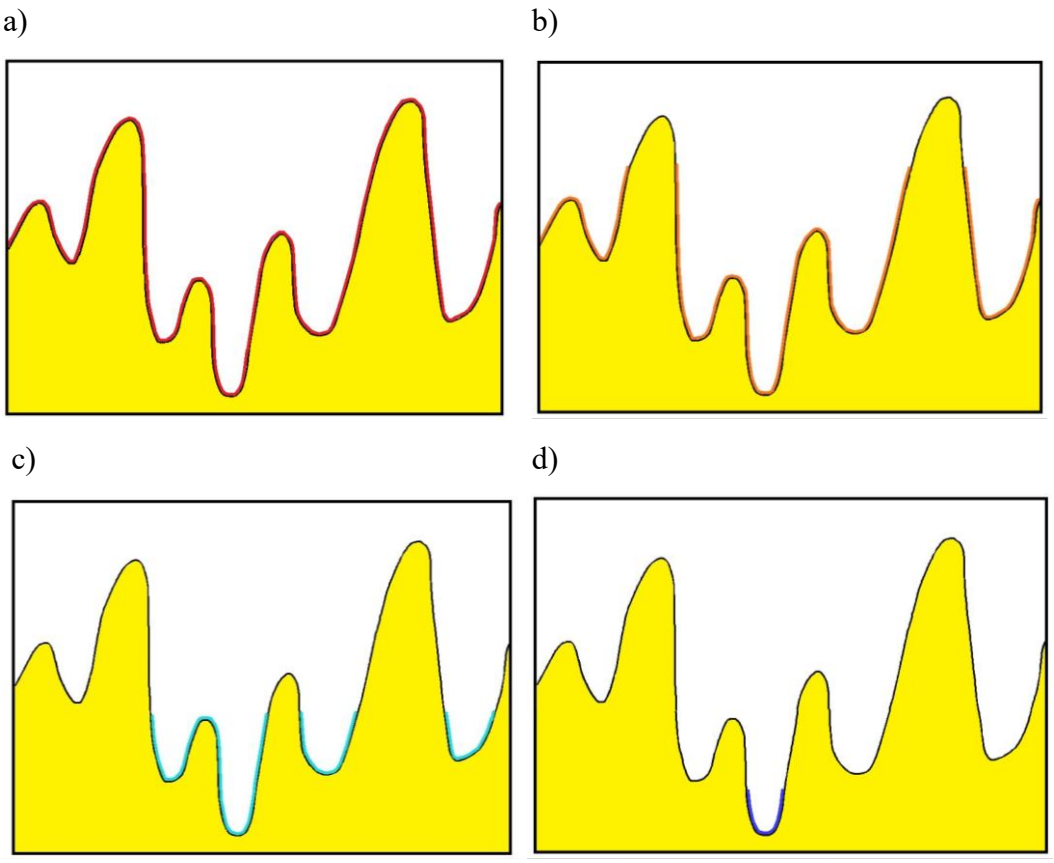


Rysunek 2.7. Zwiększanie rozdzielczości losowań poprzez podział promienia na coraz większą liczbę równych odcinków i uzyskanie lepszego przybliżenia rozwiązania.
Źródło: Opracowanie własne

Algorytm symulowanego wyżarzania [238,241-243] jest algorytmem, w którym rozwiązanie jest poprawiane w sposób ciągły, aż do osiągnięcia oczekiwanego rezultatu bądź aż do napotkania sytuacji, w której już nie można poprawić bieżącego rozwiązania. Matematyczny algorytm symulowanego wyżarzania jest wzorowany na procesie krystalizacji. Empirycznie dowiedziono, że w pewnej temperaturze wybrany materiał może znajdować się w różnych stanach stabilnych. Przykładem jest wymrażanie krzemu do postaci ciała stałego. Jeżeli nastąpi nagłe obniżenie temperatury płynnego krzemu, wówczas wytworzy się substancja amorficzna, której atomy są nieuporządkowane. Jeżeli schłodzenie następuje wolniej, wówczas krzem będzie mieć strukturę polikrystaliczną (zlepek mikrokryształków). Dopiero bardzo powolne wymrażanie doprowadzi do otrzymania kryształu krzemu z uporządkowaną strukturą atomów. Im bardziej uporządkowana jest struktura atomów, tym bardziej maleje energia kryształu krzemu. Tak więc tylko powolne obniżanie temperatury (odprężanie, wyżarzanie) pozwala, dzięki unikaniu wewnętrznych naprężeń, na uzyskanie globalnego minimum energii (odpowiada to sytuacji powstania idealnego kryształu krzemu). Poprzez zastosowanie różnych procesów schładzania można więc doprowadzić płynny krzem do różnych stanów, które odpowiadają różnym minimom lokalnym energii. Problemem optymalizacyjnym jest wyznaczenie najniższej energii potrzebnej do przejścia krzemu ze struktury płynnej do stałej.

Algorytm symulowanego wyżarzania, wzorowany na procesie wymrażania krzemu, jest algorytmem iteracyjnym, co oznacza, że polega na stopniowym polepszaniu początkowego rozwiązania, aż do znalezienia rozwiązania o satysfakcjonującej dokładności. Różni się on jednak od typowych metod iteracyjnych tym, że dopuszcza wybór w danym kroku rozwiązania gorszego od rozwiązania aktualnego. Wpływ

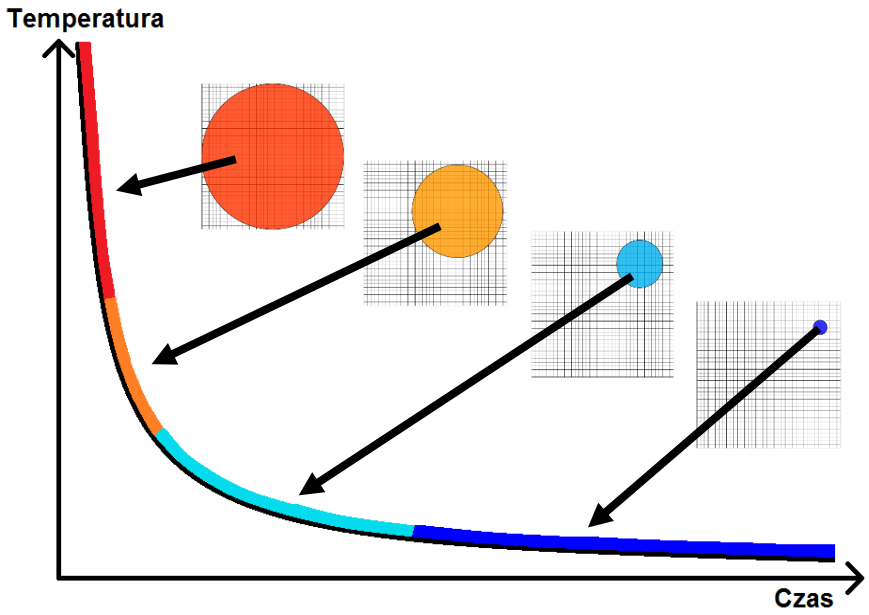
na prawdopodobieństwo wyboru gorszego rozwiązania ma podstawowy parametr algorytmu, którym jest temperatura wyżarzania (odpowiadająca temperaturze krystalizacji w fizycznym odpowiedniku algorytmu). Im wyższa temperatura, tym prawdopodobieństwo wyboru gorszego rozwiązania jest większe. Im temperatura niższa, tym algorytm jest bardziej zbliżony w działaniu do typowych metod iteracyjnych [244]. Przy bardzo wysokiej temperaturze algorytm **bardzo często** wybiera rozwiązania gorsze (Rysunek 2.8a). Przy spadku temperatury algorytm już tylko **czasami** wybiera rozwiązania gorsze (Rysunek 2.8b). Przy niskiej temperaturze algorytm **bardzo rzadko** wybiera gorsze rozwiązania, a rozwiązanie otrzymane stara się ulepszyć (Rysunek 2.8c). Przy bardzo niskiej temperaturze algorytm **nie wybiera** już gorszych rozwiązań, a rozwiązania otrzymane stara się maksymalnie ulepszyć (Rysunek 2.8d).



Rysunek 2.8. Poszukiwanie rozwiązania spośród wszystkich dostępnych rozwiązań.
Źródło: Opracowanie własne.

Na początku działania algorytmu temperatura jest wysoka i bardzo często algorytm dokonuje zmiany konfiguracji oraz wyboru rozwiązania. Często wybierane jest rozwiązanie gorsze, a wybór sąsiedniego rozwiązania odbywa się w sposób losowy.

W miarę kolejnych iteracji oraz zmniejszania temperatury, algorytm wybiera częściej rozwiązanie lepsze i je zapamiętuje. Pod koniec działania algorytmu temperatura jest już bardzo niska, co oznacza, że prawdopodobieństwo wyboru gorszego rozwiązania jest bardzo małe. Algorytm działa wtedy jak typowy algorytm iteracyjny, starając się ulepszyć aktualne rozwiązanie (Rysunek 2.9). Dzięki możliwości wyboru gorszego rozwiązania algorytm ma możliwość wyjścia z obszaru lokalnego optimum funkcji celu i kontynuowania optymalizacji w kierunku globalnego minimum. Temperatura ma wpływ nie tylko na prawdopodobieństwo chwilowego zaakceptowania gorszego rozwiązania, ale także na wielkość promienia okręgu, wewnątrz którego poszukiwane jest nowe optymalne rozwiązanie.



Rysunek 2.9. Schemat poszukiwania najlepszego rozwiązania w algorytmie symulowanego wyżarzania.
Źródło: Opracowanie własne.

Proces optymalizacji z wykorzystaniem algorytmu symulowanego wyżarzania przebiega w sposób następujący. Losowane jest startowe rozwiązanie X z przestrzeni możliwych rozwiązań problemu optymalizacyjnego. Następnie rozwiązanie to jest zaburzane, np. poprzez zmianę wartości jednej ze zmiennych decyzyjnych – w ten sposób powstaje nowe rozwiązanie Y. W kolejnym kroku z dwóch rozwiązań X i Y wybierane jest startowe rozwiązanie dla kolejnej iteracji. Rozwiązanie Y staje się rozwiązaniem startowym, jeżeli spełniony jest jeden z dwóch warunków: rozwiązanie Y jest lepsze

(w kategorii funkcji kryterialnej) od rozwiązania *X* lub rozwiązanie *Y* jest co prawda gorsze od rozwiązania *X*, ale wylosowana wartość prawdopodobieństwa jest mniejsza od wyznaczonej dla aktualnej iteracji wartości funkcji prawdopodobieństwa *P*, zależnej od aktualnej temperatury *T*. Jeżeli żaden z warunków nie jest spełniony, to rozwiązaniem startowym dla kolejnej iteracji pozostaje rozwiązanie *X*. Po wyborze rozwiązania startowego dokonywane jest obniżenie temperatury zgodnie z przyjętą funkcją *G*. Opisane działania wykonywane są wielokrotnie, aż do osiągnięcia ustalonego warunku zakończenia, którym najczęściej jest osiągnięcie ustalonej na starcie algorytmu temperatury minimalnej T_{min}.

Funkcja prawdopodobieństwa *P* jest przyjmowana najczęściej jako (wzór 2.21):

$$P = e^{-\frac{f(Y)-f(X)}{T}}, \tag{2.21}$$

gdzie: *X* – poprzednie rozwiązanie, *Y* – aktualne rozwiązanie, *f()* – funkcja kryterialna
Funkcja zmiany wartości temperatury *G* jest przyjmowana najczęściej jako (wzór 2.22):

$$\mathrm{G} = \alpha * \mathrm{T_m}, \tag{2.22}$$

gdzie α jest stałe i mieści się w przedziale $\alpha \in < 0{,}80; 0{,}99 >$; na ogół $\alpha = 0{,}95$, *m* – numer kolejnej iteracji algorytmu.

2.4 Optymalizacja wielokryterialna

Podczas rozwiązywania rzeczywistych problemów, związanych z wyborem najlepszego rozwiązania spośród wielu innych (podjęcia optymalnej decyzji), bardzo często dochodzi do sytuacji, kiedy wybór jest zależny równocześnie od wielu kryteriów. W zadaniu optymalizacji wielokryterialnej poszukuje się wektora $x = [x_1, x_2, \dots, x_i, \dots, x_n]^T$, minimalizującego zbiór kryteriów $q_1(x), q_2(x), \dots, q_l(x), \dots, q_p(x)$, w zbiorze dopuszczalnym $\Phi = \{x: g_j(x) \leq 0, j = 1, 2, 3, \dots, m\}$ [203].

Kryteria brane pod uwagę w problemie optymalizacji wielokryterialnej mogą być zgodne lub niezgodne. Kryteria są zgodne, jeżeli dla pary kryteriów $q_s(x)$ i $q_t(x)$ dla $s,t \in <1,2,3,\dots,p>$ zmianie wektora x towarzyszy poprawa obu kryteriów. Kryteria są zaś niezgodne, jeżeli dla pary kryteriów $q_s(x)$ i $q_t(x)$ dla $s,t \in <1,2,3,\dots,p>$ zmianie wektora x towarzyszy poprawa jednego kryterium kosztem pogorszenia drugiego kryterium. W przypadku niezgodnych kryteriów możliwe jest doprowadzenie do zgodności poprzez zmianę znaku jednego z kryteriów W ten sposób można zastąpić pierwotne zadanie z niezgodnymi kryteriami nowym zadaniem z kryteriami zgodnymi [203,245].

Ze względu na to, że teoria optymalizacji nie zajmuje się rozwiązywaniem zadań wielokryterialnych, dlatego takie zadania są zastępowane ciągiem zadań jednokryterialnych. Podczas budowy takiego zastępczego zadania jednokryterialnego wykorzystuje się wiele metod [203,246-251], takich jak np.:

- Metoda ważonego kryterium zbiorczego (ang. *Method of weighted aggregate criterion*) [203,246,257-259].
- Metoda programowania celowego (ang. *Goal programming method*) [247,261-262].
- Metoda leksykograficznego porządkowania kryteriów (ang. *Lexicographic ordering of criteria method*) [203,246,257,267-278].
- Metoda ograniczania kryteriów (ang. *Criteria limitation method, Trade-off method, ε-constraint method*) [203,257,263,282].

Metoda ważonego kryterium zbiorczego.

W przypadku, gdy nie wszystkie kryteria składowe $q_l(x)$ są ważne w takim samym stopniu, wówczas za pomocą funkcji $u_l(q_l)$ można każdemu z kryteriów składowych $q_l(x)$ przypisać wartość zależną od ważności danego kryterium l w stosunku do pozostałych kryteriów składowych [252-256]. Dla zbioru p kryteriów z hierarchią określoną w ten sposób, tworzy się kryterium zbiorcze (wzór 2.23) [203,246,257-259]:

$$Q = \sum_{l=1}^{p} u_l(q_l), \tag{2.23}$$

które następnie trzeba zminimalizować, aby znaleźć rozwiązanie optymalne zadania wielokryterialnego [203,248,257-259].

Dla funkcji liniowej (wzór 2.24) [258-260]:

$$u_l(q_l) = w_l q_l(x), \tag{2.24}$$

stąd po zastąpieniu zadania wielokryterialnego zadaniem jednokryterialnym (wzór 2.25) [257]:

$$Q(x) = \sum_{l=1}^{p} w_l q_l(x) \rightarrow min, \tag{2.25}$$

przy ograniczeniach: $q_l(x) \leq 0,\ j = 1, 2, 3, \dots, m$; mnożniki w_l (ponieważ są wagami przypisywanymi poszczególnym kryteriom), muszą spełniać dodatkowe wymagania: $w_l \geq 0, l = 1, 2, 3, \dots, p, \sum_{l=1}^{p} w_l = 1$]203,258-259].

Dla zadania z dwoma kryteriami (wzór 2.26) [258-259]:

$$Q(x) = w_1 q_1(x) + w_2 q_2(x), \tag{2.26}$$

Wadą tej metody jest to, że wagi w_l są określane na ogół przez zespół ekspertów, którzy opierają się na własnych preferencjach, czego skutkiem jest to, że wagi w_l mogą być wybierane nieobiektywnie. Ponadto należy znormalizować kryteria składowe q_l (ponieważ mogą różnic się od siebie nie tylko rzędem wielkości, ale też wartościami) [203,248].

Metoda programowania celowego.

Metoda programowania celowego wykorzystywana jest w przypadku, gdy znane są dokładne wartości poszczególnych kryteriów składowych $\tilde{q_l}$ [247,261-262]. Celem jest osiągnięcie punktu (wektora) idealnego $\tilde{Q} = [\tilde{q}_1, \dots, \tilde{q}_l, \dots, \tilde{q}_p]^T$ lub rozwiązania leżącego najbliżej tego punktu w przestrzeni kryteriów P^p [203,246].

Punktom, które należą do zbioru dopuszczalnych wartości kryteriów Ψ można przypisać liczby wyrażające ich odległości od punktu $\tilde{Q}$. Wówczas zadanie wielokryterialne zastępuje się zadaniem jednokryterialnym, w którym będzie minimalizowana odległość między punktami (wzór 2.27, 2.28 i 2.29) [203,247,257,263-265].

$$Q = [q_1, \dots, q_l, \dots, q_p]^T, \tag{2.27}$$

oraz

$$\tilde{Q} = [\tilde{q}_1, \dots, \tilde{q}_l, \dots, \tilde{q}_p]^T, \tag{2.28}$$

gdzie: $\rho(q_1, \dots, q_l, \dots, q_p) \rightarrow min$, (2.29)
przy ograniczeniach: $q_j(x) \leq 0, j = 1, \dots, m$.

Ogólny wzór (poniżej) opisuje przykładowe postaci metryk do pomiaru odległości pomiędzy punktami Q oraz $\tilde{Q}$ (wzór 2.30) [203,264,266] :

$$\rho_r(q_1, \dots, q_l, \dots, q_p) = [\sum_{l=1}^{p} |\tilde{q}_l - q_l(x)|^r]^{\frac{1}{r}}, \tag{2.30}$$

gdzie $1 \leq r \leq \infty$.

Konkretne postaci metryk zależą od przyjętego r. Na przykład dla często stosowanej metryki Euklidesa $r = 2$, a odległość pomiędzy punktami Q oraz $\tilde{Q}$ wynosi (wzór 2.31) [203,264,266]:

$$\rho_2(q_1, \dots, q_l, \dots, q_p) = [\sum_{l=1}^{p} |\tilde{q}_l - q_l(x)|^r]^{\frac{1}{2}} = \sqrt{\sum_{l=1}^{p} |\tilde{q}_l - q_l(x)|^2} \tag{2.31}$$

Wzór $\rho_r(q_1, \dots, q_l, \dots, q_p) = [\sum_{l=1}^{p} |\tilde{q}_l - q_l(x)|^r]^{\frac{1}{r}}$ można uzupełnić o wagi w_l (tworzone analogicznie jak w metodzie ważonego kryterium zbiorczego) i w rezultacie otrzymuje się (wzór 2.32) [203,264-265]:

$$\rho_r^w(q_1, \dots, q_l, \dots, q_p) = [\sum_{l=1}^{p} |w_l[\tilde{q}_l - q_l(x)]|^r]^{\frac{1}{r}} = \sqrt[r]{\sum_{l=1}^{p} |w_l[\tilde{q}_l - q_l(x)]|^r} \tag{2.32}$$

Zastępcze zadanie jednokryterialne po wykorzystaniu metryki Euklidesa wynosi (wzór 2.33) [203,264]:

$$Q(x) = \sqrt{\sum_{l=1}^{p} |w_l[\tilde{q}_l - q_l(x)]|^2} \rightarrow min \tag{2.33}$$

przy ograniczeniach: $q_j(x) \leq 0, j = 1, \dots, m$.

Przykładowo, dla zadania z dwoma kryteriami (wzór 2.34) [203,264-266]:

$$Q(x) = w_1[q_1(x) - \tilde{q}_1]^2 + w_2[q_2(x) - \tilde{q}_2]^2 \rightarrow min \tag{2.34}$$

Jeżeli $Q(x)$ zostanie potraktowane jako parametr $Q(x) = const.$, wówczas powyższe równanie przyjmie postać elipsy o środku w punkcie $[\tilde{q}_1\ \tilde{q}_2]^T$ i półosiach $\sqrt{\frac{const}{w_2}}$ oraz $\sqrt{\frac{const}{w_1}}$. W szczególnym przypadku, dla $w_1 = w_2 = w$ otrzymuje się okrąg o środku w punkcie $[\tilde{q}_1\ \tilde{q}_2]^T$ i promieniu $\sqrt{\frac{const}{w}}$ [203].

Metoda leksykograficznego porządkowania kryteriów.

Metoda leksykograficzna polega na posortowaniu kryteriów składowych w porządku ich ważności, gdzie indeks $q_1(x)$ to kryterium najważniejsze, a $q_p(x)$ to kryterium najmniej ważne [203,246,257,267-278]. Algorytm rozwiązywania zadań wielokryterialnych za pomocą metody leksykograficznej wygląda następująco [203]:

1. Znajdź minimum $q_1(x) \rightarrow min$ przy ograniczeniach $g_j(x) \leq 0$ oraz $j = 1, \dots, m$; Rozwiązanie oznacz jako x_1^* oraz $q_1^* = q_1(x_1^*)$.
2. Znajdź minimum $q_2(x) \rightarrow min$ przy ograniczeniach (wzór 2.35):

$$\begin{cases} g_j(x) \leq 0,\ j = 1, \dots, m \\ q_1(x) = q_1^* \end{cases}, \tag{2.35}$$

   Rozwiązanie oznacz jako x_2^* oraz $q_2^* = q_2(x_2^*)$
3. Znajdź kolejne minima aż do $q_l(x) \rightarrow min$ przy ograniczeniach (wzór 2.36):

$$\begin{cases} g_j(x) \leq 0,\ j = 1, \dots, m \\ q_l(x) = q_l^*,\ l = 1, \dots, l-1 \end{cases} \tag{2.36}$$

Podczas rozwiązywania ciągu zadań jednokryterialnych dla wszystkich $l = 1, \dots, p$ kryteriów, za rozwiązanie zadania wielokryterialnego przyjmuje się rozwiązanie ostatniego z zadań jednokryterialnych [203].

Zaletą tej metody jest to, że jest to metoda łatwa w implementacji. Wadą tej metody jest to, że poszczególne kryteria wymagają określenia ich ważności przez ekspertów opierających się na własnych preferencjach, stąd też metoda jest nieobiektywna. Ponadto kolejne implementacje (uruchomienia) metody, różniące się porządkiem układów kryteriów, dadzą wiele różnych wyników [203,257,266,269,273-275,277-281].

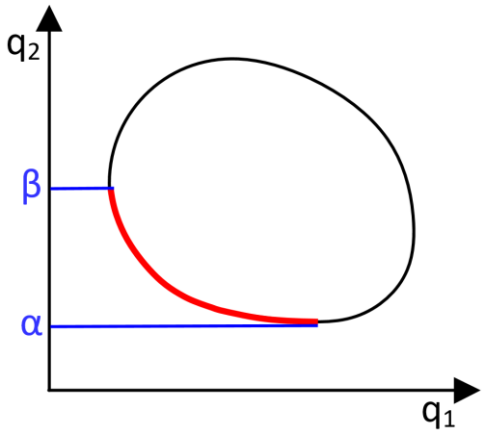
Metoda ograniczania kryteriów.

W metodzie tej na początku wybierane jest jedno składowe kryterium optymalizacyjne $q_s(x)$ (gdzie $s = 1, \dots, p$), uznane za najważniejsze. Następnie określane są dolne i górne granice zmienności pozostałych kryteriów $\alpha_l \leq q_l(x) \leq \beta_l$, $l = 1, \dots, s-1, s+1, \dots, p$. Kolejno przeprowadzana jest optymalizacja jednokryterialna ze względu na wybrane wcześniej kryterium; pozostałe kryteria włączane są do zbioru ograniczeń brzegowych, a zadanie jednokryterialne przyjmuje postać [203,257,263,282]:

Znajdź minimum $q_1(x) \rightarrow min$ przy ograniczeniach (wzór 2.37):

$$\begin{cases} g_j(x) \leq 0, \ \ j = 1, \dots, m \\ \alpha_l \leq q_l(x) \leq \beta_l, \ \ l = 1, \dots, s-1, s+1, \dots, p \end{cases} \tag{2.37}$$

Na Rysunku 2.10 przedstawiono graficzną ilustrację metody dla zadania dwukryterialnego, w którym kryterium q_1 jest uznane za kryterium ważniejsze, a kryterium q_2 jest ograniczone od góry i od dołu w sposób następujący $\alpha \leq q_2 \leq \beta$.



Rysunek 2.10. Graficzna ilustracja metody ograniczania kryteriów dla zadania dwukryterialnego.

Źródło: Opracowanie własne na przykładzie [203].

Istnieje jeszcze wiele innych metod rozwiązywania zadań optymalizacji wielokryterialnej, korzystających z metod iteracyjnych, heurystycznych, algorytmów ewolucyjnych [246,251,257], logiki rozmytej [283], itp.

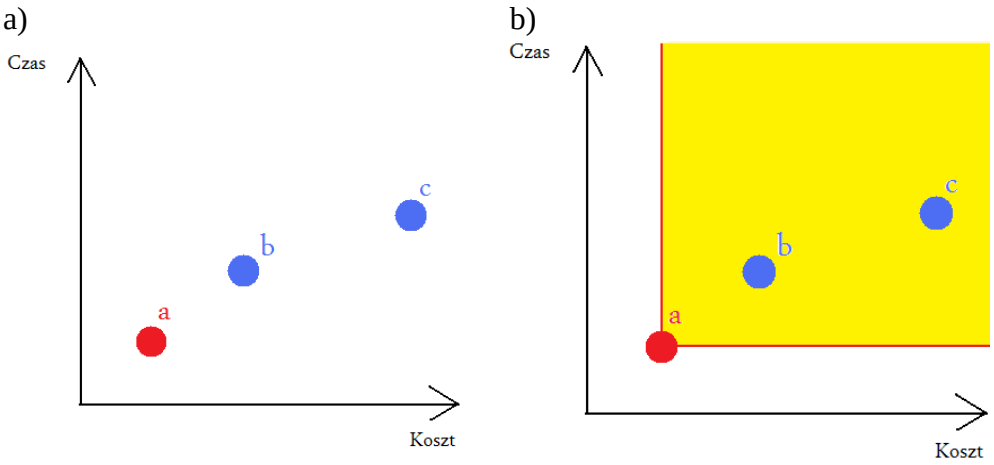
2.5 Rozwiązania niezdominowane oraz Front Pareto

Zestaw potencjalnych rozwiązań problemu optymalizacji wielokryterialnej może być podzielony na rozwiązania zdominowane oraz niezdominowane. Rozwiązaniem niezdominowanym nazwiemy każde rozwiązanie, które nie jest zdominowane przez żadne inne rozwiązanie dopuszczalne. Dla rozwiązań niezdominowanych nie ma rozwiązań co najmniej równych z uwagi na wszystkie kryteria i nie ma rozwiązania lepszego z uwagi na co najmniej jedno kryterium. Dla rozwiązań niezdominowanych nie można poprawić jednego kryterium, nie pogarszając jednocześnie któregoś z pozostałych kryteriów. Rozwiązania zdominowane to natomiast wszystkie te rozwiązania, które są gorsze przynajmniej z uwagi na jedno kryterium od rozwiązań niezdominowanych. W jednym problemie optymalizacyjnym może istnieć wiele rozwiązań niezdominowanych. Zbiór rozwiązań niezdominowanych nazywany jest także zbiorem kompromisów. Pojęcie rozwiązania niezdominowanego zostało sformułowane przez V. Pareto i dlatego rozwiązania niezdominowane nazywa się też rozwiązaniami Pareto-optymalnymi, a zbiory rozwiązań kompromisowych – zbiorami Pareto [203].

Z powyższego wynika, że rozwiązanie jest paretooptymalne wtedy, gdy nie można znaleźć rozwiązania lepszego od niego ze względu na wszystkie kryteria. Przykładem może być chęć zakupu towaru w jak najkrótszym czasie po jak najniższej cenie. W tak postawionym problemie funkcjami kryterialnymi będzie:

- *czas na zakup → min;*
- *koszt towaru → min.*

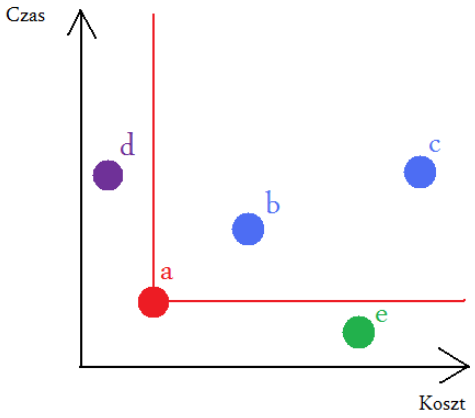
Przykładowe rozwiązanie paretooptymalne przy tak postawionym problemie zostało przedstawione na Rysunku 2.11a. Jak można zaobserwować na rysunku najlepszym wyborem spośród towarów *a, b* i *c* jest zakup towaru a, ponieważ posiada on zarówno najniższą cenę, jak i można go kupić w możliwie najkrótszym czasie. Rozwiązania b i c są więc zdominowane przez rozwiązanie a, które z kolei jest rozwiązaniem niezdominowanym. Obszar dominacji rozwiązania a nad pozostałym rozwiązaniami został przedstawiony na Rysunku 2.11b.



Rysunek 2.11. a) Dominacja rozwiązania a nad rozwiązaniem b i c; b) Obszar dominacji rozwiązania a.

Źródło: Opracowanie własne.

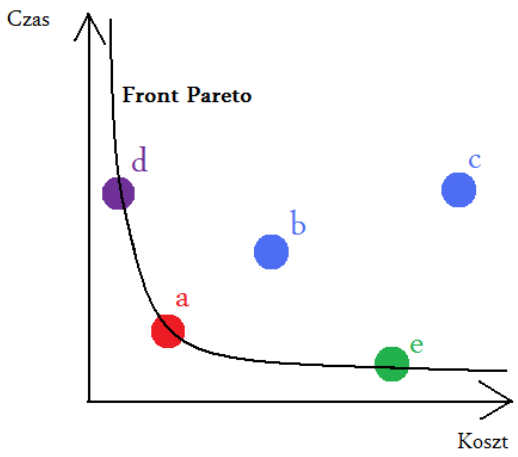
Bardzo często zdarza się, że nowe rozwiązanie jest lepsze w porównaniu do aktualnego rozwiązania tylko pod względem jednego czynnika. Sytuację taką przedstawiono na Rysunku 2.12.



Rysunek 2.12. Obszar dominacji rozwiązań d i e nad rozwiązaniem a pod względem jednego czynnika.

Źródło: Opracowanie własne.

W sytuacji przedstawionej na Rysunku 2.12 rozwiązanie a nie dominuje nad rozwiązaniami d ani e, ponieważ rozwiązanie d jest lepsze od rozwiązania a pod względem kosztów, natomiast rozwiązanie e jest lepsze od rozwiązania a pod względem czasu. Z tego też względu żadne z rozwiązań a, d i f nie dominują nad sobą wzajemnie. Tworzą tzw. Front Pareto (Rysunek 2.13).



Rysunek 2.13. Front Pareto.

Źródło: Opracowanie własne.

Zbiór optymalny w sensie Pareto, tzw. Front Pareto, jest zbiorem rozwiązań, które nie są zdominowane w całej dopuszczalnej przestrzeni poszukiwań. Są one zatem rozwiązaniami optymalnymi w danym wielokryterialnym problemie optymalizacyjnym.

Rozdział III
Selekcja cech metodami klasycznymi

W procesie selekcji cech obecnie bardzo często wykorzystuje się metody dokonujące selekcji w oparciu o wyniki procesu klasyfikacji. Metody te nie są jednak zbyt efektywne w przypadku wielowymiarowych zbiorów danych ze względu na wysoki koszt obliczeniowy. Znacznie bardziej wydajne są metody budujące ranking cech według kryterium niezależnego od klasyfikatora. Metody te są jednak znacznie mniej dokładne, ponieważ w procesie selekcji nie uwzględniają współzależności występujących między cechami. Z uwagi na wielość metod, które można wykorzystać w procesie selekcji cech, trudno jest odgórnie wskazać jedną najlepszą metodę, czy nawet ogólną strategię. Decyzja o tym, którą z możliwych strategii wybrać, jest zawsze zależna od posiadanego zbioru danych. W przypadku cech wyekstrahowanych z sygnału EEG, stosowanie niektórych strategii jest z góry pozbawione sensu z uwagi na małą liczbę obserwacji i jednocześnie na dużą liczbę cech, z których należy dokonać wyboru. W niniejszym rozdziale przybliżono pokrótce problem selekcji cech oraz dokonano przeglądu podstawowych metod selekcji w podziale na metody: opakowane, filtrujące, wbudowane oraz tzw. frappery. W ostatnim podrozdziale odniesiono się krótko do metod dokonujących transformacji pierwotnej przestrzeni cech.

3.1 Ogólne zagadnienia selekcji cech

Problem selekcji cech [4,30-39] jest problemem trudnym to rozwiązania w przypadku interfejsu mózg-komputer, ponieważ sygnał EEG jest sygnałem wielowymiarowym i w dodatku długotrwałym. Stąd do opisu surowego sygnału EEG nierzadko wykorzystuje się kilkaset a nawet i kilka tysięcy cech. Tak duża przestrzeń potencjalnych cech uniemożliwia zastosowanie procedury pełnego przeszukania przestrzeni cech, która jest jedyną procedurą pozwalającą na znalezienie globalnie

optymalnego zbioru cech [69]. Stąd też w procesie selekcji cech na potrzeby interfejsów mózg-komputer stosowane są algorytmy przeszukiwania przestrzeni potencjalnych cech dających rozwiązania jedynie lokalnie optymalne.

Algorytmy do selekcji cech (ang. *Feature Selection Algorithm, FSA*) [4,26] to algorytmy, których głównym celem jest identyfikacja takich zestawów cech, które pozwolą na otrzymanie optymalnej wartości funkcji oceny. W literaturze można znaleźć wiele algorytmów do selekcji cech, klasyfikowanych głównie w układzie trzech następujących kryteriów [284]:

- metody generowania podzbiorów cech;
- strategii przeszukiwania przestrzeni podzbiorów cech;
- strategii oceniania wybranych podzbiorów cech.

Ze względu na metodę generowania podzbiorów cech, algorytmy selekcji cech można podzielić na trzy grupy [27]:

- algorytmy dokonujące przeglądu zupełnego;
- algorytmy korzystające z podejścia losowego;
- algorytmy korzystające z podejścia heurystycznego.

W przypadku algorytmów dokonujących przeglądu zupełnego generowane są wszystkie możliwe podzbiory cech. Jest to metoda najmniej efektywna, jednakże jako jedyna zapewnia znalezienie najlepszego możliwego podzbioru cech. W algorytmach korzystających z podejścia losowego, cechy umieszczane w kolejnych podzbiorach wybierane są w sposób losowy, przy czym każda cecha wybierana jest w losowaniu bez powtórzeń z jednakowym prawdopodobieństwem. Jako wynik zwracany jest podzbiór cech najlepszy z punktu widzenia przyjętej funkcji oceny. Wreszcie w algorytmach wykorzystujących podejście heurystyczne cechy wprowadzane są do kolejnych podzbiorów zgodnie z wybraną heurystyką [27].

Heurystyczne algorytmy selekcji cech są algorytmami zdecydowanie najczęściej wykorzystywanymi w problemach klasyfikacyjnych opisanych przez dużą liczbę cech, takich jak problem klasyfikacji sygnału EEG wykorzystywanego do sterowania w interfejsie mózg komputer. Tego rodzaju algorytmy selekcji cech są złożone z czterech podstawowych etapów, najczęściej wykonywanych wielokrotnie [27-29]:

1. Generowanie następnego podzbioru/podzbiorów cech zgodnie z przyjętą heurystyką, przy czym punktem startowym algorytmu jest:
   - pełen zbiór cech albo
   - zbiór cech losowo wybranych albo
   - pusty zbiór cech.
2. Ocena wygenerowanego podzbioru/podzbiorów cech.
3. Porównanie wygenerowanego podzbioru/podzbiorów w kategoriach funkcji kryterialnej z wcześniej wygenerowanymi podzbiorami i wybór podzbioru najlepszego na danym etapie.
4. Sprawdzenie kryterium zatrzymania algorytmu, którym może być zarówno przyjęta wartość funkcji kryterialnej (np. dokładność klasyfikacji), jak i liczba iteracji algorytmu, bądź osiągnięcie założonego rozmiaru podzbioru cech

Rozpatrując drugie z wskazanych kryteriów klasyfikacji algorytmów selekcji cech – strategię przeszukiwania przestrzeni podzbiorów cech – również można wyróżnić trzy grupy algorytmów: algorytmy o złożoności wykładniczej (np. przeszukiwanie wyczerpujące), algorytmy sekwencyjne (np. przeszukiwanie w przód, przeszukiwanie w tył) oraz algorytmy stochastyczne (np. algorytmy genetyczne) [285].

Biorąc pod uwagę strategie oceniania wybranych podzbiorów cech (czyli relację z innymi algorytmami nadrzędnymi, np. algorytmem klasyfikacji) można wyróżnić trzy podstawowe grupy algorytmów: algorytmy opakowujące (wrappery), algorytmy filtrujące oraz metody wbudowane (embedded) [286].

Bez względu na strategię oceniania wybranych podzbiorów cech, bardzo ważnym elementem w procesie oceny jest funkcja kryterialna zastosowana do oceny kolejnych podzbiorów cech. W funkcji tej mogą być wykorzystywane różne miary [27,287-289], przy czym wśród najczęściej wykorzystywanych należy wymienić m.in.:

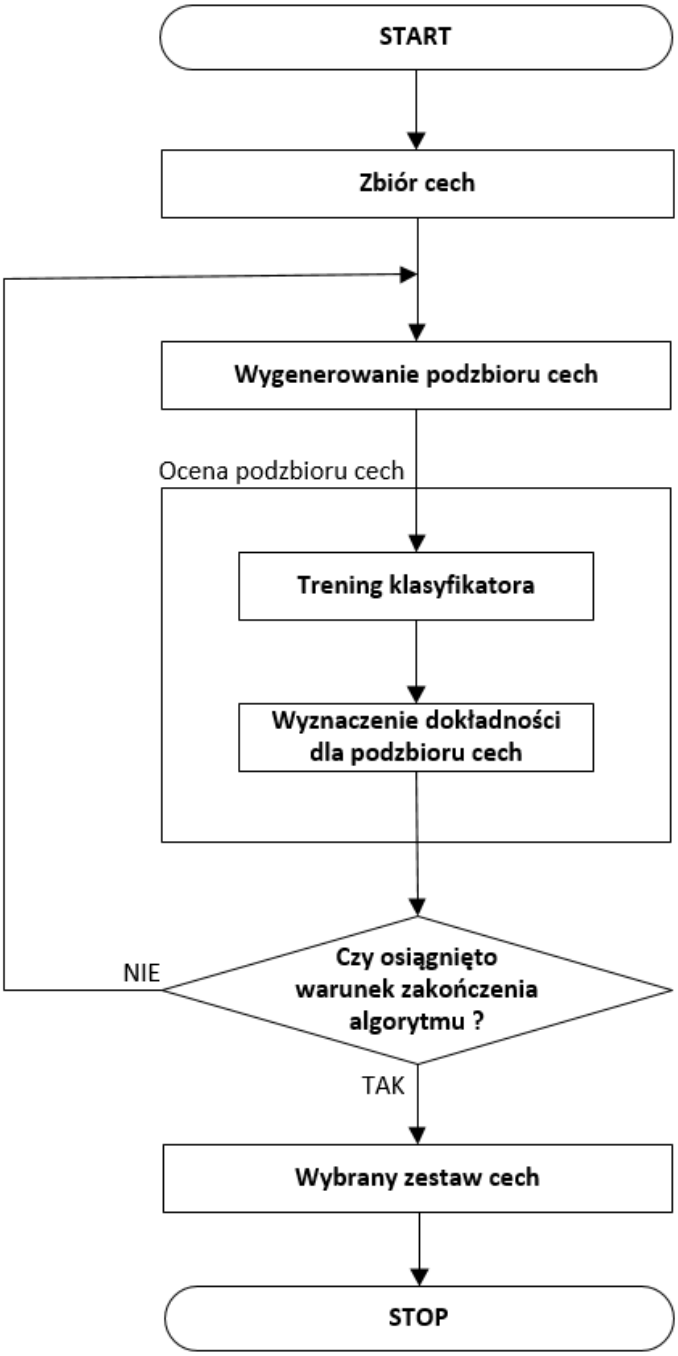
1. Błąd klasyfikacji (ang. *Classifier Error Rate)* [179]. Metody wykorzystujące błąd klasyfikacji do oceny podzbiorów cech charakteryzują się wysoką dokładnością, jednakże są mało efektywne obliczeniowo, ponieważ ich użycie wymaga, aby każdy analizowany podzbiór cech został wykorzystany w charakterze zmiennych wejściowych klasyfikatora. Oznacza to konieczność wyznaczenia parametrów klasyfikatora dla każdego badanego podzbioru cech, co jest procesem czasochłonnym.

2. Miarę odległości (ang. *Distance Measure)* [255,313] – w tym podejściu preferuje się tę podprzestrzeń cech (ten podzbiór cech), w której odległości między klastrami (reprezentującymi klasy) w przestrzeni cech są większe niż ma to miejsce w przypadku innych podprzestrzeni (podzbiorów) cech.
3. Miarę zależności (ang. *Dependence Measure*) [313,362] określającą zdolność przewidywania wartości jednej cechy na podstawie wartości innej cechy. Przykładem na wykorzystanie tej miary są współczynniki korelacji (np. współczynnik korelacji Pearsona) definiujące siłę korelacji pomiędzy cechami znajdującymi się w badanym zbiorze cech oraz między cechami a klasami (wybierane są podzbiory o słabej korelacji między cechami i jednocześnie silnej korelacji między cechami, a klasami).
4. Entropię, definiowaną jako najmniejsza średnia ilość jednostek informacji (bitów) potrzebna do zakodowania faktu zajścia pewnego zdarzenia (ze zbioru zdarzeń o danych prawdopodobieństwach). Dla równych prawdopodobieństw, kiedy każde ze zdarzeń zachodzi tak samo często, entropia jest największa. Entropia jest wykorzystywana w procesie selekcji cech poprzez preferowanie atrybutów dla których następuje najwyższy przyrost informacyjny mierzony właśnie spadkiem entropii [197,256].

3.2 Metody wrapper.

Metody wrapper (opakowane, ang. *wrappers*) [27,29,287,290-305] są metodami zawierającymi sprzężenie zwrotne między algorytmem selekcji cech a elementem decyzyjnym (najczęściej klasyfikatorem), którego efektywność jest wykorzystywana w funkcji celu realizowanego procesu optymalizacyjnego. Metody te w procesie porównywania kolejnych podzbiorów cech uwzględniają zarówno wyniki klasyfikacji jak i wiedzę o przynależności do danej klasy [70]. Wybrany algorytm uczący jest tu wykorzystywany już na etapie selekcji cech, a dokładniej mówiąc na etapie oceny wybranych podzbiorów. Ocena podzbiorów cech jest więc dokonywana z wykorzystaniem konkretnego klasyfikatora, a miarą jakości danego podzbioru cech jest efektywność wybranego klasyfikatora. Efektywność ta może być szacowana przy wykorzystaniu odrębnego zbioru weryfikacyjnego lub walidacji krzyżowej. Ten sam model jest wykorzystywany zarówno w trakcie selekcji cech, jak i później, po dokonaniu

wyboru najlepszego zestawu cech, do klasyfikacji kolejnych danych [61]. Do generowania podzbiorów cech można wykorzystywać wszystkie strategie opisane w poprzednim punkcie m.in. algorytmy genetyczne, symulowane wyżarzanie, wybór losowy itp. [292,306]. Schemat działania metod opakowujących został przedstawiony na Rysunku 3.1.



Rysunek 3.1. Model selekcji cech z wykorzystaniem metod wrapper.
Źródło: Opracowanie własne

Algorytm metody wrapper rozpoczyna się od wygenerowania podzbioru cech. Następnie dokonywana jest ocena podzbioru cech poprzez trening klasyfikatora i wyznaczenie jego dokładności dla tego podzbioru cech. Kolejnym krokiem jest sprawdzenie czy został osiągnięty warunek zakończenia algorytmu. Jeżeli warunek jest spełniony, wówczas wybrany zestaw cech jest zapisywany i algorytm kończy działanie. Jeżeli nie, następuje wygenerowanie kolejnego podzbioru cech, trening klasyfikatora, wyznaczenie dokładności dla podzbioru cech i sprawdzenie warunku zakończenia. Warunkiem zakończenia algorytmu jest najczęściej wykonanie określonej liczby iteracji, osiągnięcie pożądanej dokładności klasyfikacji lub brak poprawy rozwiązania przez określoną liczbę iteracji.

Główne zalety podejścia wrapper to dostosowanie wybranego podzbioru cech do wymagań algorytmu decyzyjnego oraz większa dokładność w porównaniu np. do metod filtrujących. Rozwiązanie daje dobre efekty dzięki zastosowaniu klasyfikatora już na etapie oceny wybranego podzbioru cech. Jeżeli w metodzie rankingowej dwie cechy będą znajdować się nisko w rankingu, to nie ma możliwości ich wyboru do szukanego podzbioru cech nawet wówczas, gdy po połączeniu ich ocena będzie bardzo wysoka. Z kolei metoda wrapper uwzględni te dwie cechy dzięki wykorzystaniu wyniku klasyfikacji jako miary użytej do oceny jakości podzbioru cech [307].

Do głównych wad metod typu wrapper można zaliczyć brak rzetelności w ocenie dokładności klasyfikacji oraz większą złożoność obliczeniową – dla każdego potencjalnego podzbioru cech konieczne jest odpowiednie nauczenie klasyfikatora (np. z wykorzystaniem k-krotnej walidacji krzyżowej). Metody opakowane są kosztowne obliczeniowo, ponieważ klasyfikator jest wykorzystywany do oceny każdego podzbioru cech. Zastosowanie klasyfikatora już na etapie oceny wybranego podzbioru cech sprawia, że wybrany podzbiór nie jest uniwersalny; przy wykorzystaniu innego algorytmu, należy powtórzyć proces selekcji cech [307].

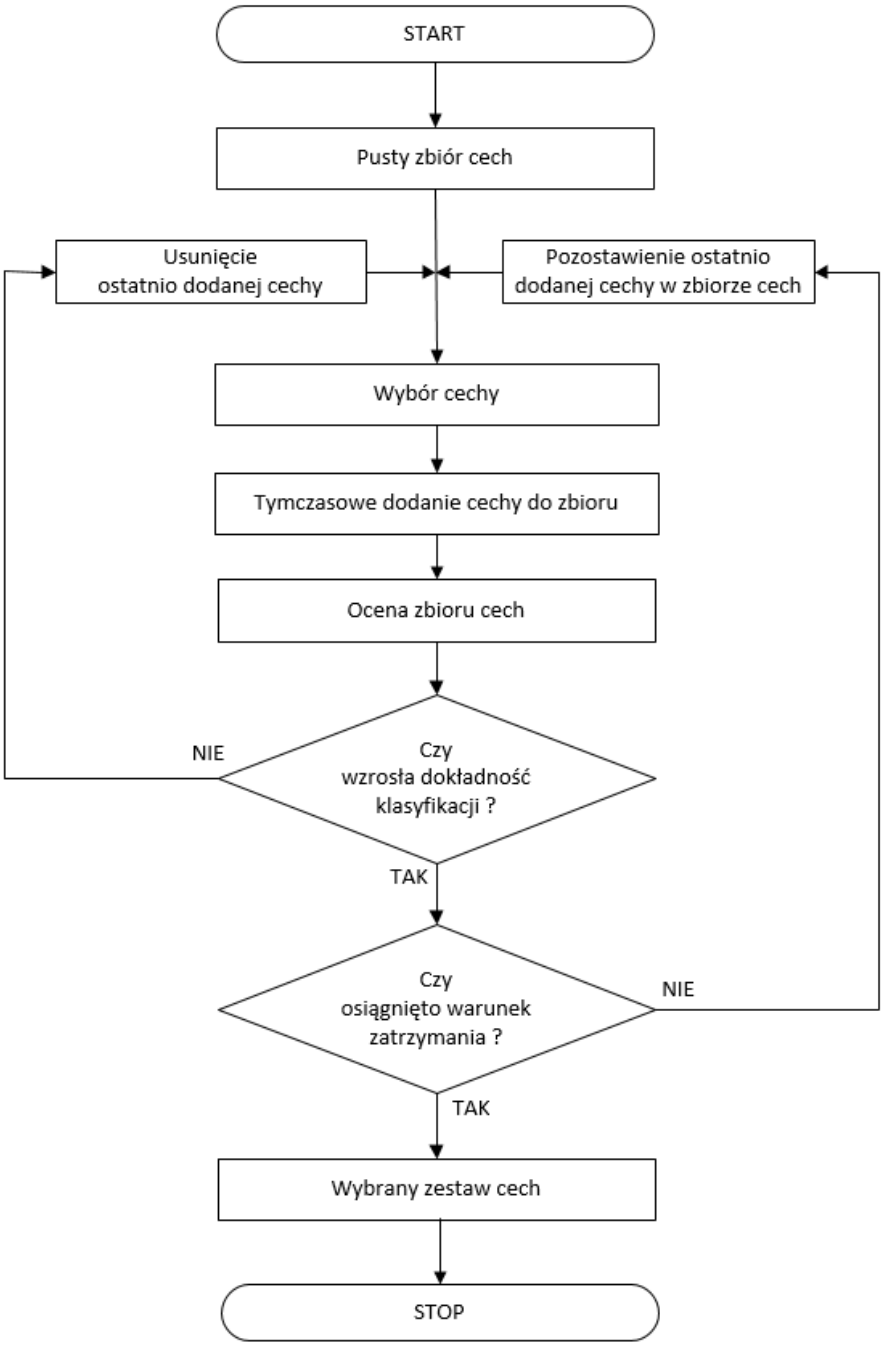
Liczba możliwych kombinacji podzbiorów cech rośnie wykładniczo wraz z liczbą cech w tym zbiorze. Dla kilku lub kilkunastu cech można przeszukać wszystkie możliwe kombinacje cech. Jednakże dla dwudziestu cech liczba wszystkich możliwych kombinacji będzie wynosić około milion, a dla trzydziestu cech, liczba wszystkich możliwych kombinacji wzrośnie do miliarda. W literaturze przedmiotu [308] można znaleźć różne strategie wykorzystywane w metodach opakowanych, zaprojektowane w taki sposób, aby znaleźć właściwe proporcje między czasem obliczeń a jakością

otrzymanego podzbioru cech. Najlepszym przykładem są tutaj różne odmiany strategii nazywanej selekcją krokową.

Metody selekcji krokowej należą do grupy algorytmów heurystycznych, które definiują całościową strategię prowadzenia procesu przeszukiwania przestrzeni możliwych rozwiązań. Metody te posiadają wiele odmian, z których w zastosowaniach praktycznych najczęściej stosowane są trzy: selekcja w przód, selekcja w tył oraz selekcja dwukierunkowa. Metody te działają na takiej samej zasadzie, a jedyną cechą je różniącą jest kierunek prowadzenia poszukiwań w przestrzeni cech [69].

W przypadku **selekcji w przód** (ang. *Forward selection*) [61,69-78] proces przeszukiwania przestrzeni cech rozpoczyna się od pustego zbioru cech, do którego, w każdym kolejnym kroku procedury, dodawana jest jedna cecha. Każdorazowo po dodaniu cechy do aktualnego zbioru cech wykonywane jest sprawdzenie czy dana cecha poprawia skuteczność algorytmu uczącego. W klasycznej odmianie algorytmu, jeżeli odpowiedź jest twierdząca, wówczas dana cecha zostaje w zbiorze cech; jeżeli odpowiedź jest negatywna, wówczas cecha ta odpada. W przypadku selekcji cech na potrzeby procesu klasyfikacji kryterium decydującym o tym, która cecha ma zostać dodana do zbioru cech jest najczęściej dokładność klasyfikacji. Cały proces selekcji kończy się więc w momencie, gdy żadna z pozostałych cech nie jest w stanie spowodować, aby wzrosła dokładność klasyfikacji albo w momencie, kiedy osiągnięto założoną wcześniej maksymalną liczbę cech [61,69-70,73]. Schemat działania klasycznej wersji metody selekcji w przód został przedstawiony na Rysunku 3.2.

Praca algorytmu rozpoczyna się pustego zbioru cech, do którego w kolejnych krokach dodawane są kolejno wybierane cechy. W każdej iteracji wybierana jest jedna cecha, która jest wprowadzana do klasyfikatora wraz z wszystkimi innymi wcześniej wybranymi cechami. Jeżeli klasyfikator charakteryzuje się wyższą dokładnością, aniżeli klasyfikator z kroku poprzedniego, to ostatnio dodana cecha jest pozostawiana na stałe w zbiorze cech, jeżeli nie, to jest ona usuwana. Następnie następuje sprawdzenie czy osiągnięto warunek zatrzymania. Jeżeli warunek jest spełniony, wówczas wybrany zestaw cech jest zapisywany i algorytm kończy działanie. Jeżeli nie, następuje dodanie kolejnej cechy do zbioru cech i ponowna ocena zbioru cech. Procedura jest kontynuowana do osiągnięcia warunku zatrzymania algorytmu.



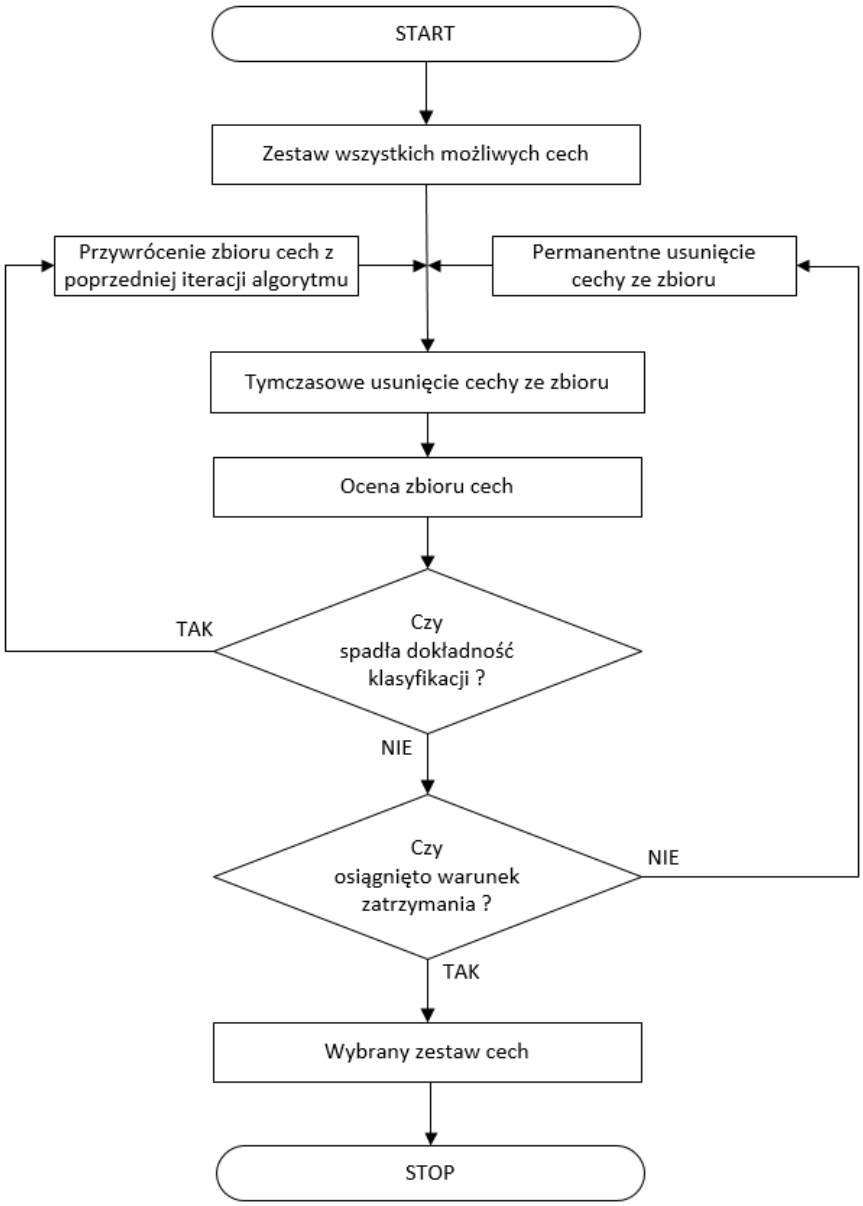
Rysunek 3.2. Schemat działania selekcji w przód.
Źródło: Opracowanie własne.

Poza klasyczną odmianą algorytmu selekcji w przód bardzo często stosowana jest modyfikacja polegająca na tym, że w każdym kolejnym kroku testowane są wszystkie cechy wcześniej nie wybrane (tworzony jest ranking cech), a cecha ostatecznie dodawana do zbioru ma tę właściwość, że posiada najkorzystniejszą wartość wybranej funkcji celu. Dla zbioru 250 cech proces selekcji w przypadku tak zmodyfikowanej strategii przebiegałby następująco. Najpierw zostałoby przetestowanych 250 zbiorów z 1 cechą –

wybrana zostałaby cecha o najkorzystniejszej wartości funkcji celu. Następnie przetestowano by 249 zbiorów o 2 cechach, każdy zawierałby cechę wybraną w kroku poprzednim i jedną z pozostałych 249 cech. Ponownie zostałby wybrany zbiór o najkorzystniejszej wartości funkcji celu. W ten sam sposób przetestowano by zbiory o 3 cechach, 4 cechach itd. Proces selekcji zostałby zakończony w momencie, gdy żaden ze zbiorów utworzonych w kroku kolejnym nie byłby lepszy (pod względem założonego kryterium) od najlepszego zbioru z kroku poprzedniego.

Algorytm selekcji w przód jest szybszy niż algorytm selekcji w tył lub selekcji dwukierunkowej. Jest to szczególnie widoczne podczas rozwiązywania problemów, które charakteryzują się wysoką wymiarowością wektora cech. Wynika to z tego, że najbardziej wymagający obliczeniowo jest proces wyznaczenia parametrów klasyfikatora. W metodzie selekcji w przód, gdzie algorytm najpierw testuje niskowymiarowe zbiory cech, koszt uruchomienia klasyfikatora jest znacznie mniejszy niż dla metody selekcji w tył lub selekcji dwukierunkowej, gdzie algorytm na początku bada zbiory o maksymalnej lub zbliżonej do maksymalnej liczbie cech [61]. Wadą algorytmu selekcji w przód (w wersji klasycznej) jest natomiast to, że cecha dodana do zbioru cech, nie może już zostać z niego usunięta, nawet, jeżeli w kolejnych etapach pojawią się cechy lepsze niż ta aktualna cecha [71]. Wada ta jest w dużym stopniu redukowana w opisanej powyżej odmianie algorytmu testującego na każdym etapie wszystkie jeszcze nie wybrane cechy.

W drugiej z wspomnianych metod selekcji krokowej – metodzie **selekcji w tył** (ang. *Backward Selection*) [61,69,71,73,77,79] proces przeszukiwania przestrzeni cech rozpoczyna się od zbioru zawierającego wszystkie możliwe cechy. Następnie, w każdym kolejnym kroku procedury ze zbioru cech usuwana jest jedna cecha. Podobnie jak w przypadku selekcji w przód, również tutaj możliwe są dwie wersje algorytmu. W wersji klasycznej ze zbioru usuwana jest pierwsza cecha, której usunięcie nie powoduje pogorszenia dokładności klasyfikacji. W wersji zmodyfikowanej, na każdym etapie test usunięcia cechy z aktualnego zbioru cech jest przeprowadzany z osobna dla wszystkich cech znajdujących się w zbiorze – ostatecznie ze zbioru cech eliminowana jest ta cecha, której usunięcie wywołuje najmniejszy lub żaden spadek dokładności klasyfikacji (Rysunek 3.3) [61,69,73,78].



Rysunek 3.3. Schemat działania selekcji w tył.
Źródło: Opracowanie własne

Podstawową wadą algorytmu selekcji w tył jest to, że cecha raz usunięta ze zbioru cech, nie może zostać do niego ponownie dodana, nawet jeżeli po usunięciu kolejnych cech okazałoby się, że jest ona jednak istotna dla procesu klasyfikacji, a wcześniej została usunięta, ponieważ jej istotność była znoszona przez obecność innych cech. Poza tym, metoda ta charakteryzuje się wyższym kosztem obliczeniowym, aniżeli metoda selekcji w przód. Dodatkowo, w przypadku, gdy wymiarowość wektora cech jest wyższa niż wymiarowość wektora danych, stosowanie algorytmu selekcji w tył może w ogóle nie być zasadne z uwagi na prawdopodobieństwo wystąpienia efektu przeuczenia klasyfikatora.

Algorytm rozpoczyna się od zbioru wszystkich możliwych cech. Ze zbioru tymczasowo usuwana jest jedna z cech, po czym następuje ocena zredukowanego zbioru cech. Jeżeli dokładność klasyfikacji nie uległa pogorszeniu, następuje sprawdzenie czy osiągnięto warunek zatrzymania. Jeżeli warunek jest spełniony, wówczas wybrany zestaw cech jest zapisywany i algorytm kończy działanie. Jeżeli nie, wówczas ostatnio analizowana cecha jest permanentnie usuwana ze zbioru cech i następuje powrót do kroku tymczasowego usunięcia kolejnej cechy. Procedura jest kontynuowana do osiągnięcia warunku zatrzymania algorytmu.

Selekcja dwukierunkowa (ang. *BiDirectional Selection, BDS*) [61,69,71,73] jest metodą, która powstała poprzez połączenie selekcji w przód i w tył. Zasada jej działania oparta jest na naprzemiennym stosowaniu obu strategii elementarnych, a rozwiązanie znalezione jest wspólnym rozwiązaniem dla obydwu algorytmów [61,69].

W celu zapewnienia zbieżności algorytm przeszukiwania w przód sprawdza dodatkowo, czy cecha, która ma być dodana do zbioru cech, nie została już wcześniej wyeliminowana przez algorytm przeszukiwania w tył. Z kolei algorytm przeszukiwania w tył nigdy nie usuwa cechy dodanej we wcześniejszych krokach przez algorytm przeszukiwania w przód. Warunek ten ma istotne znaczenie, ponieważ bez jego zastosowania mogłoby bowiem dojść do sytuacji, w której zarówno jeden algorytm jak i drugi w tej samej iteracji próbowałyby odpowiednio dodać i usunąć tę samą cechę. Tak więc wszystkie cechy, które zostały dodane lub usunięte ze zbioru cech, nie biorą udziału w selekcji w kolejnych krokach działania obydwu algorytmów [61].

Wśród innych metod z grupy wrapper można wymienić m.in.: metodę SBS-SLASH (ang. *Sequential Backward Selection SLASH*) [287,309-310], LVW (ang. *Las Vegas Wrapper* – wykorzystującą LVF – ang. *Las Vegas Filter Algorithm* – do wygenerowania początkowego zbioru cech) [287,311-312] i inne.

3.3 Metody filtrujące

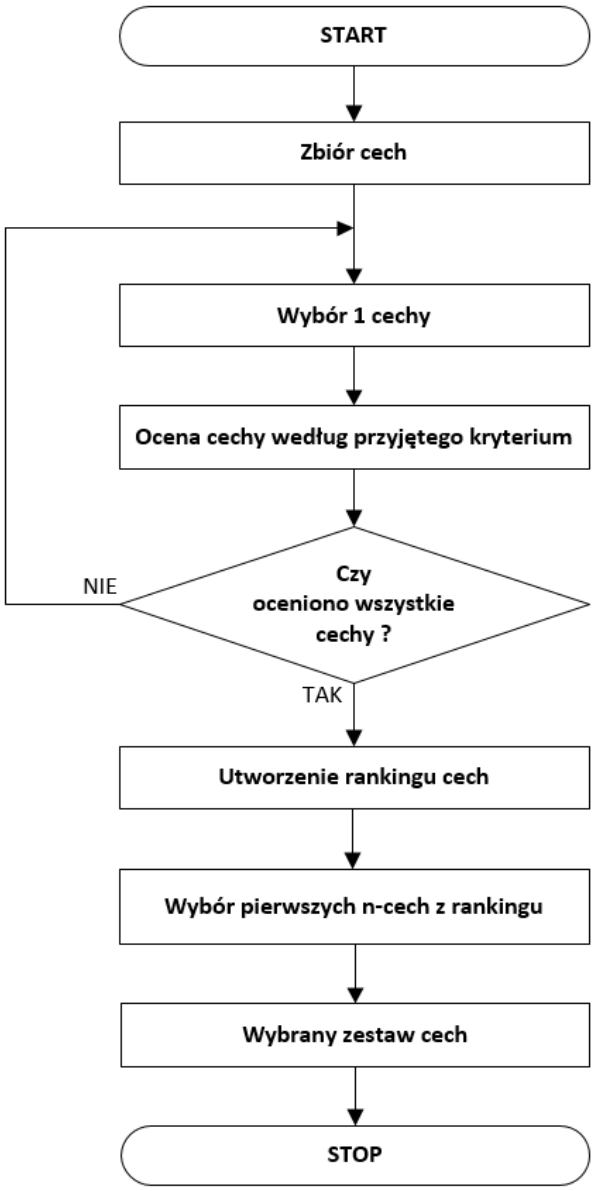
Metody filtrujące, inaczej filtry (ang. *Filter Approach*) [27,29,61,70,287-288,291-292,294,304-305,313-317] to metody, w których selekcja cech jest dokonywana w oderwaniu od procesu klasyfikacji. Cechy oraz zbiory cech oceniane są jedynie przy wykorzystaniu danych je opisujących [318]. Do oceny wykorzystywane są różnorodne

kryteria, oparte m.in. na: współczynniku korelacji, miarach odległości międzyklasowej [319], miarach wywodzących się z teorii informacji [320] i innych [321-322] .

Zasada działania filtrów opiera się na ocenie cech według przyjętego kryterium, a następnie na posortowaniu cech na podstawie wartości kryterium. Otrzymuje się wówczas ranking, czyli listę cech uporządkowaną według zadanego kryterium. Sam proces selekcji cech polega jedynie na wyborze najlepszych cech z utworzonego rankingu, przy czym punkt odcięcia (czyli kryterium decydujące o tym które cechy mają być wybrane) jest najczęściej ustalany albo jako liczba cech znajdujących się na najwyższych pozycjach w rankingu albo jako wartość kryterium, która musi zostać przekroczona przy ocenie kolejnych cech, aby zostały wybrane do zbioru cech. Ogólny schemat działania metod filtrujących został przedstawiony na Rysunku 3.4.

Algorytmy filtrujące nie odrzucają cech redundantnych. Istnieje możliwość skorygowania tej niepożądanej własności dzięki zastosowaniu dwuetapowej selekcji rankingowej, która w pierwszym kroku porządkuje cechy ze względu na ich istotność, a w drugim kroku grupuje cechy ze względu na ich redundancję [61,323]. Ostateczna selekcja cech jest następnie wykonywana zgodnie z rankingiem istotności, ale przy zachowaniu warunku wyboru tylko 1 cechy z każdej z grup.

Filtry wyznaczają kryterium i samodzielnie podejmują decyzje o wyborze cech, na podstawie niezależnego od klasyfikatora współczynnika. Jest to zarówno zaleta, jak i wada metod filtrujących. Wśród innych zalet tych metod można wymienić między innymi szybkość działania algorytmu. Ponieważ filtry są niezależne od klasyfikatora, zatem zmniejsza się złożoność obliczeniowa, a dzięki temu wzrasta szybkość działania oraz wydajność filtra. Szybkość filtra zależy oczywiście również od wykorzystanego algorytmu wyznaczania kryterium wyboru cech, jednakże algorytm ten jest bardzo często mniej kosztowny niż algorytm uczenia klasyfikatora. Kolejną zaletą jest uniwersalność metody, przejawiająca się możliwością wykorzystania dowolnego klasyfikatora w procesie klasyfikacji [61,307].



Rysunek 3.4. Model selekcji cech za pomocą metod filtrujących.
Źródło: Opracowanie własne

Poza wymienionymi zaletami, metody filtrujące posiadają również kilka wad. Podstawową z nich jest nieuwzględnianie zależności między cechami. W efekcie cechy, które stają się istotne dopiero w przypadku połączenia z innymi cechami, są odrzucane przez algorytm, ponieważ zajmują niskie miejsca w rankingu. Wadą jest także nieuwzględnianie specyfiki danego algorytmu klasyfikacji, co w przypadku niektórych klasyfikatorów może spowolnić proces uczenia lub pogorszyć wyniki klasyfikacji. Z uwagi na dwie wspomniane wady, klasyfikatory działające w oparciu o cechy wybrane za pomocą metod filtrujących cechują się z reguły niższą dokładnością od klasyfikatorów korzystających z cech zwróconych przez metody opakowane [61,307].

W klasycznej wersji metod filtrujących liczba cech wybieranych do zbioru cech jest ustalana na początku algorytmu jako jeden z jego podstawowych parametrów. Nie jest to do końca dobre rozwiązanie, ponieważ nie uwzględnia ono stopnia przyrostu dokładności klasyfikatora wywołanego przez kolejno dodawane cechy. W efekcie może okazać się, że wiele cech zostaje dodanych do zbioru cech w sposób nadmiarowy, nie przynosząc istotnego wzrostu dokładności klasyfikatora. W celu wyeliminowania tej niedogodności metody filtrujące łączy się czasami z metodami wrapper. Efektem tego połączenia są metody typu frapper [61,324]. W metodach tych algorytm decyzyjny (np. sieć neuronowa) optymalizuje liczbę wybranych cech Metody filtrów wykorzystuje się wówczas do utworzenia rankingu i selekcji cech, a metody wrappers do zmiany i dopasowania parametrów filtrów. Metody typu frapper łączą cechy zarówno pozytywne, jak i negatywne obu metod składowych. Z jednej strony są szybsze niż metody wrappers (ale też mniej od nich dokładne). Z drugiej natomiast, są bardziej dokładne niż filtry (ale jednocześnie znacznie od nich wolniejsze). Podstawową ich zaletą jest wysoki uniwersalizm – mogą one zostać wykorzystane w większości problemów klasyfikacyjnych [307].

Grupa metod filtrujących obejmuje wiele metod szczegółowych oraz kryteriów służących do budowy rankingu cech. Jednym z najprostszych i jednocześnie najczęściej stosowanych podejść jest podejście, w którym w charakterze kryterium, na podstawie którego tworzony jest ranking cech, wykorzystywany jest współczynnik korelacji Pearsona, zdefiniowany jako (wzór 3.1) [61,325-326]:

$$\mathrm{R(x,y)} = \frac{\mathrm{cov(x,y)}}{\sigma_{\mathrm{x}} \cdot \sigma_{\mathrm{y}}} \tag{3.1}$$

gdzie:

$R(x, y)$ – współczynnik korelacji Rearsona pomiędzy zmiennymi x i y,

$cov(x, y)$ – kowariancja pomiędzy zmiennymi x i y,

σ – odchylenie standardowe.

W przypadku, gdy współczynnik korelacji jest wyznaczany na podstawie danych z próby, to przyjmuje on następującą postać (wzór 3.2) [61]:

$$R(i) = \frac{\sum_{k=1}^{m}(x_{ki} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{ki} - \bar{x}_i)^2 \cdot \sum_{k=1}^{m}(y_k - \bar{y})^2}} \tag{3.2}$$

gdzie:

i – indeks cechy;

k – indeks obserwacji;

x_{ik} – wartości i–tej cechy w k-tej obserwacji;

y_k – wartość zmiennej wyjściowej dla k–tego wektora cech;

$\bar{x}_i$ – Wartości średnie i–tej cechy wyznaczone z wszystkich obserwacji;

$\bar{y}$ – Wartości średnie zmiennej wyjściowej wyznaczone z wszystkich obserwacji.

Interpretacja współczynnika korelacji Pearsona została przedstawiona w Tabeli 3.1. Siła korelacji wskazuje, jak dużo danych zachowuje się w sposób oczekiwany. Przykładowo, dla $r > 0$ wraz ze wzrostem x rośnie też y. Przykładowo dla $r=0.25$ tylko część danych spełnia zależność (widać pewną tendencję, ale zdarzają się także odstępstwa), a dla $r=0.95$ prawie wszystkie dane spełniają założenia (tendencja jest bardzo widoczna) [327].

Rodzaj korelacji	**Siła korelacji dla $\|r\|$**
r > 0 – korelacja dodatnia Gdy wartość x rośnie to wartość y również rośnie	$\|r\| < 0.2$ Brak związku liniowego
r = 0 – brak korelacji Gdy wartość x rośnie to wartość y czasem rośnie a czasem maleje	$0.2 \leq \|r\| < 0.4$ Słaba zależność
r < 0 – korelacja ujemna Gdy wartość x rośnie to wartość y maleje	$0.4 \leq \|r\| < 0.7$ Umiarkowana zależność
	$0.7 \leq \|r\| < 0.9$ Dość silna zależność
	$0.9 \leq \|r\|$ Bardzo silna zależność

Tabela 3.1. Interpretacja współczynnika korelacji liniowej Pearsona
Źródło: [327]

W przypadku wykorzystania współczynnika korelacji liniowej Pearsona jako kryterium selekcji cech, ranking cech jest budowany na podstawie bezwzględnej wartości współczynnika. Nie jest tu bowiem istotny kierunek związku występującego między badaną cechą, a klasą, lecz siła tego związku. Oczywiście na szczycie rankingu znajdują się cechy o maksymalnej bezwzględnej wartości współczynnika korelacji.

Inną często stosowaną metodą filtrującą jest **metoda Relief** oraz jej rozszerzenie dla problemów wieloklasowych – metoda ReliefF [38,42,51-60]. W metodzie tej atrybuty są oceniane według tego, jak dokładnie umożliwiają rozróżnianie podobnych obiektów, tj. obiektów znajdujących się blisko siebie w przestrzeni rozpatrywanej cechy. Idea ich działania opiera się, podobnie jak np. w przypadku klasyfikatora kNN [200] na pomiarze odległości pomiędzy obiektami. Procedura Relief (oraz ReliefF) wykorzystuje heurystykę, według której zadaniem dobrego atrybutu (cechy) jest rozróżnianie leżących blisko siebie obiektów należących do innych klas. Dobry atrybut powinien posiadać więc taką samą wartość dla obiektów, które znajdują się blisko siebie i należą do tej samej klasy oraz różną wartość dla obiektów znajdujących się blisko siebie, ale pochodzących z różnych klas [42].

Procedura ReliefF przebiega następująco. Dla każdej analizowanej cechy X_i oraz dla każdego obiektu r, znajdowanych jest k-obiektów $S_{1\ldots k}$, tej samej klasy co obiekt r, dla których wartość cechy X_i jest najbardziej zbliżona do wartości tej samej cechy w badanym obiekcie r. Następnie, dla każdej analizowanej cechy X_i, każdego obiektu r oraz każdej klasy innej niż klasa, do której przynależy obiekt r znajdowanych jest k-obiektów $S_{1\ldots k}$, dla których wartość cechy X_i jest najbliższa wartości cechy X_i w obiekcie r (w sumie znajdowanych jest więc $k \cdot (c-1)$ obiektów, gdzie c to liczba analizowanych klas) [42].

W najprostszym przypadku (tzn. przy założeniu $k = 1$ oraz klasyfikacji binarnej) przebieg algorytmu wyznaczania wartości indeksu Relief dla poszczególnych cech występujących w analizowanym zagadnieniu można opisać następująco [51,328]:

1. Zdefiniowanie, ile próbek μ będzie zawierał zbiór treningowy (dla $\mu = n$ wybierany jest cały zbiór próbek, dla $\mu < n$, zbiór treningowy wybierany jest losowo).
2. Dla każdego punktu x ze zbioru treningowego znajdowany jest najbliższy sąsiad z tej samej klasy (ang. $h = hit$) oraz z innej klasy (ang. $m = miss$).
3. Następnie dla każdej cechy f indeks Relief jest powiększany o odległość x od m (liczoną wzdłuż kierunku f) i pomniejszany o odległość x od h (wzór 3.3):

$$\mathrm{Relief}(\mathrm{f};\mu) = \frac{1}{\mu}\sum_{\mathrm{i}=1}^{\mu}(|\mathrm{x_i},\mathrm{m_i}|_{\mathrm{f}} - |\mathrm{x_i},\mathrm{h_i}|_{\mathrm{f}}) \tag{3.3}$$

gdzie

$|x_i, m_i|_f$ to odległość wzdłuż kierunku f x od m (wzór 3.4):

$$|x_i, m_i| \equiv |x_i[f] - m_i[f]| \tag{3.4}$$

$|x_i, h_i|_f$ to odległość wzdłuż kierunku f x od h (wzór 3.5):

$$|x_i, h_i| \equiv |x_i[f] - h_i[f]| \tag{3.5}$$

Algorytm wyznaczania indeksu ReliefF [51] [328] opiera się na modyfikacji powyższej procedury polegającej na uwzględnieniu k-sąsiadów z każdej z c klas. W przypadku najprostszym, czyli kiedy liczba sąsiadów jest ograniczona do 1 (k=1) modyfikacja indeksu ReliefF przebiega według wzoru (wzór 3.6):

$$\mathrm{ReliefF}(f; \mu, c) = \frac{1}{\mu \cdot c} \sum_{i=1}^{\mu} \sum_{j=1}^{c} (|x_i, m_{ij}|_f - |x_i, h_{ij}|_f) \tag{3.6}$$

Powyższe wzory mają zastosowanie, jeżeli cechy posiadają wartości numeryczne. W przypadku atrybutów symbolicznych, odległości $|\cdot,\cdot|_f$ trzeba zastąpić wartościami binarnymi, tzn. 1 – dla punktów mających taką samą wartość cechy f oraz 0 – dla punktów mających różną wartość cechy [51].

Wybór procedury ReliefF jest odpowiedni dla danych pochodzących z sygnału EEG, ponieważ bardzo dobrze funkcjonuje ona w przypadku zaszumionych sygnałów oraz wysoce zależnych (skorelowanych) cech. Ponadto, z powodu liniowej złożoności, procedura bardzo dobrze skaluje wielowymiarowe przestrzenie cech. Tak jak inne metody filtrujące procedura ReliefF buduje ranking wszystkich cech, a więc wymaga podania pewnej wartości progowej, decydującej o tym, które z cech powinny ostatecznie zostać wybrane [38].

Kolejną często stosowaną procedurą filtrującą jest procedura oparta na kryterium przyrostu informacji, nazywana w skrócie IG (ang. *Information Gain*) [38,329-348]. Procedura ta jest bardzo skuteczna podczas selekcji cech w przestrzeniach wielowymiarowych [60], stąd jest często wykorzystywana do selekcji cech na potrzeby klasyfikacji tekstu [333]. Z założenia metoda ta operuje na dyskretnych wartościach cech, stąd w przypadku stosowania jej do budowy rankingu cech o wartościach ciągłych, konieczna jest wcześniejsza dyskretyzacja cech.

Przyrost informacji można zdefiniować jako redukcję entropii, osiągniętą dzięki wykorzystaniu cechy f. Obliczany jest według wzoru 3.7, gdzie $IG(f)$ jest oczekiwaną redukcją entropii H wywołaną przez cechę f (wzór 3.7) [38,343-344]:

$$IG(f) = H(S) - \sum_i \frac{n_i}{n} \cdot H(S_i) \tag{3.7}$$

gdzie:
$IG(f)$ – przyrost informacji osiągnięty dzięki wykorzystaniu cechy *f*;
S – zbiór wszystkich przykładów uczących;
S_i – *i*-ty podzbiór przykładów uczących wygenerowany przez podział zbioru S ze względu na wartości cechy *f;*
n – liczność pełnego zbioru przykładów;
n_i – liczność *i*-tego podzbioru przykładów;
$H(S)$ – entropia pełnego zbioru przykładów;
$H(S_i)$ – entropia *i*-tego podzbioru przykładów;

Entropia, zarówno w przypadku pełnego zbioru przykładów, jak i kolejnych podzbiorów, jest wyznaczana jako (wzór 3.8):

$$H = -\sum_{i=1}^{k} P(C_i) \cdot \log P(C_i) \tag{3.8}$$

gdzie:
k – liczba klas,
C_i – zbiór przykładów należących do *i*-tej klasy, $i = \{1,2,\dots,k\}$,
$P(C_i)$ – prawdopodobieństwo wystąpienia przykładu z *i*-tej klasy (stosunek liczby przykładów reprezentujących *i*-tą klasę do liczby wszystkich przykładów).

Obliczenia są dokonywane dla każdej cechy, a następnie cechy są sortowane i układane w oparciu o ich wartość przyrostu informacji (IG); im wyższa wartość IG, tym bardziej cecha jest pożądana, ponieważ ma wyższe zdolności dyskryminacyjne.

Jeszcze inną metodą filtrującą jest **metoda LVF** (ang. *Las Vegas Filter*) [42,349-350]. Metoda ta wykorzystuje probabilistyczne podejście do określenia kierunku poszukiwań prawidłowego rozwiązania. Dzięki wykorzystaniu losowego przeszukiwania przestrzeni rozwiązań umożliwia znalezienie akceptowalnego rozwiązania także

w sytuacji, gdy w trakcie poszukiwań były podejmowane błędne decyzje [42,349]. Mark A. Hall wykazał, że metoda LVF umożliwia redukcję liczby cech co najmniej o połowę [45].

Algorytm metody LVF w każdej kolejnej rundzie generuje losowy podzbiór cech *S*. Jeżeli liczba cech *C* w podzbiorze *S* jest mniejsza niż liczba cech aktualnie uznana za najmniejszą C_{min}, to badana jest wartość kryterium niespójności dla zbioru cech *S*. Jeżeli wartość kryterium niespójności znajduje się poniżej predefiniowanego poziomu, to aktualna minimalna liczba cech C_{min} zostaje zastąpiona przez liczbę cech *C*, a aktualny minimalny podzbiór cech S_{min} zostaje zastąpiony przez podzbiór cech *S*. Domyślną wartością kryterium niespójności jest wartość zero, oznaczająca, że zbiór ze zredukowaną liczbą cech musi wykazywać się taką samą wartością współczynnika niespójności jak zbiór z pełną liczbą cech.

Współczynnik niespójności zbioru danych jest obliczany zgodnie z następującymi zasadami [349]:

- dwa przykłady są uznawane za niespójne, jeżeli mają identyczne wartości wszystkich cech, a różnią się jedynie wartościami klas;
- dla wszystkich przykładów o identycznych wartościach cech licznik niespójności jest liczbą przykładów pomniejszoną o liczbę wystąpień klasy dominującej w tym zbiorze przykładów;
- współczynnik niespójności jest sumą wszystkich liczników niespójności podzieloną przez całkowitą liczbę przykładów.

W sposób bardziej zwarty kryterium niespójności jest opisywane wzorem 3.9 [42]:

$$IncR = \frac{\sum_i (D_i - M_i)}{N} \tag{3.9}$$

gdzie:

D_i – liczba wystąpień *i*-tej kombinacji wartości cech;

M_i – Liczność obiektów klasy dominującej dla *i*-tej kombinacji cech;

N – liczba cech.

Kolejną popularną metodą selekcji cech działającą zgodnie z filozofią filtrów jest **metoda CFS** (ang. *Correlation based Feature Selection*) [38,42-50]. Metoda ta, zaproponowana przez M. A. Halla [48], opiera się na badaniu korelacji pomiędzy

cechami. Dokładniej mówiąc w metodzie CFS przy poszukiwaniu najlepszych podzbiorów cech uwzględniane są dwa kryteria. Pierwsze kryterium określa, jak dobrze poszczególne cechy przewidują klasę. Drugie kryterium wskazuje w jakim stopniu cechy wybrane zgodnie z kryterium pierwszym są skorelowane z innymi cechami. Dobre podzbiory cech zawierają więc cechy silnie skorelowane z daną klasą i nie skorelowane z innymi cechami [38].

W procedurze CFS często wykorzystuje się, opisaną we wcześniejszej części rozdziału, korelację liniową Pearsona. Dzięki zastosowaniu heurystyki, która mówi, że dobry podzbiór cech zawiera atrybuty, które są silnie skorelowane z określoną klasą obiektów, a nieskorelowane z innymi atrybutami możliwe jest odfiltrowanie cech, które w żadnym lub niewielkim stopniu opisują przynależność obiektu do określonej klasy oraz cech nadmiarowych, które są silnie powiązane z innymi cechami [42]. Wartość heurystyki obliczana jest z następującego wzoru 3.10 [46,48,50,351]:

$$Merit_S = \frac{k \cdot r_{cf}}{\sqrt{k + k \cdot (k-1) \cdot r_{ff}}} \tag{3.10}$$

gdzie:

$Merit_S$ – wartość heurystyki dla podzbioru *S*, który zawiera *k*-cech;

k – liczba cech;

r_{cf} – średnia wartość korelacji pomiędzy cechami i klasami;

r_{ff} – średnia korelacja wzajemna pomiędzy cechami.

Proces wyszukania optymalnego podzbioru cech w metodzie CFS jest najczęściej prowadzony zgodnie ze strategią *best first* (pierwszy najlepszy). Procedura rozpoczyna się więc od pustego zbioru cech, do którego dodawane są kolejne cechy. Każdorazowo po dodaniu cechy następuje sprawdzenie heurystyki (3.10). Jeżeli wyznaczona wartość heurystyki jest wyższa niż poprzednia, to poszukiwanie jest kontynuowane, jeżeli niższa, to jest kończone na zbiorze cech z poprzedniego kroku. Oczywiście stosowane są bardzo różne modyfikacje procedury podstawowej (np. wielokrotne rozpoczynanie procesu poszukiwań w różnych punktach, jednoczesne badanie wielu podzbiorów, zmiana kierunku poszukiwań itp.). Według Karegowda, Manjunath i Jayaram, procedura CFS daje lepsze rezultaty niż IG, jednakże wymaga o wiele dłuższych obliczeń komputerowych [334,335].

Następna z metod, **metoda FCBF** (ang. *Fast Correlation-Based Filter*) [42], jest oparta na współczynniku symetrycznej niepewności pary atrybutów (ang. *symmetrical uncertainty*). Symetryczna niepewność jest określona jako stosunek zawartości informacyjnej pary atrybutów do sumy entropii tych atrybutów (wzór 3.11) [42]:

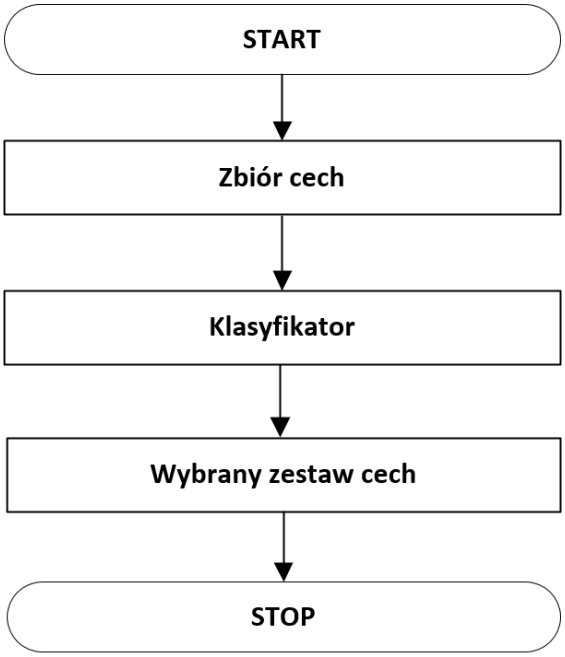
$$SU(X,Y) = 2 \cdot [\frac{IG(X|Y)}{H(X)+H(Y)}] \tag{3.11}$$

Na początku działania metody FCBF oblicza się dla każdej cechy współczynnik symetrycznej niepewności. Do dalszych obliczeń wybierane są tylko te cechy, których symetryczna niepewność jest większa od przyjętego progu. Z cech tych budowany jest ranking według malejącej wartości współczynnika symetrycznej niepewności. Następnie ranking jest sprawdzany pod względem występowania w nim cech nadmiarowych. Cechy nadmiarowe (w stosunku do cech umieszczonych na wyższych pozycjach rankingu) są usuwane z rankingu [42].

Wśród pozostałych metod z grupy filtrów można wymienić m.in. metodę CBFS (ang. *Consistency-Based Feature Selection*) [38,60,323,329,352-362], metodę 1RR (ang. *1–Rule Ranking*) [38,353,363-365], metodę indywidualnych wskaźników pojemności informacyjnej Hellwiga [42,366]) i wiele innych.

3.4 Metody wbudowane

Nazwa „*metody wbudowane*" (ang. *Embedded Approach*) [94-95, 307,367-369] wskazuje na podstawową cechę tego rodzaju metod selekcji, którą jest zintegrowanie procesu selekcji cech z procesem klasyfikacji. Przy czym nie chodzi tutaj o wykorzystanie wyników klasyfikatora jako kryterium selekcji cech (jak ma to miejsce w przypadku metod opakowanych), ale o faktyczną integrację procesu selekcji i klasyfikacji. W metodach opakowanych proces selekcji cech jest zewnętrzny w stosunku do procesu klasyfikacji - selekcja niejako „opakowuje" klasyfikację, która dzięki temu może być przeprowadzona przy wykorzystaniu dowolnego rodzaju metod klasyfikacyjnych. W przypadku metod wbudowanych to klasyfikacja jest zewnętrzna w stosunku do procesu selekcji – to klasyfikator dobiera cechy, które są optymalne według kryteriów ustalonych dla klasyfikatora (Rysunek 3.5). Stąd w metodach wbudowanych nie jest możliwe rozdzielenie procesu selekcji i procesu klasyfikacji w celu np. wykorzystania innej metody klasyfikacji.



Rysunek 3.5. Model selekcji cech za pomocą metod wbudowanych.
Źródło: Opracowanie własne.

Metody wbudowane wykorzystują cechy algorytmów uczących do automatycznej selekcji cech na etapie uczenia algorytmu decyzyjnego (np. drzewa decyzji), SVM, LDA itd. [307,367-369]. W procesie selekcji wykorzystywana jest najczęściej reguła, mówiąca o tym, że jeżeli liczba poprawnie sklasyfikowanych obserwacji zwiększa się dzięki wykorzystaniu danej cechy, wówczas wzrasta ocena istotności danej cechy w całym procesie [28,370].

Główne zalety metod wbudowanych to m.in. ich duża dokładność wynikająca z tego, że metody te projektowane są specjalnie dla konkretnych algorytmów klasyfikacji oraz duża szybkość – metoda selekcji cech jest już wbudowana w proces uczenia się klasyfikatora, co zapewnia oszczędność obliczeń na kolejnych etapach działania algorytmu. Ponadto proces selekcji cech nie wymaga wielokrotnego uruchamiania klasyfikatora dla każdego sprawdzanego podzbioru cech, jak ma to miejsce w metodach wrappers, co dalej zwiększa szybkość metod wbudowanych. Wadą metod wbudowanych jest natomiast to, że metody te mogą być wykorzystane tylko dla konkretnego algorytmu klasyfikacji [61,307].

Jedną z częściej stosowanych wbudowanych metod selekcji jest metoda **LASSO**. Metoda LASSO (ang. *Least Absolute Shrinkage and Selection Operator*) [61-68] została zaproponowana przez Tibshiraniego [67]. W metodzie tej minimalizuje się funkcję straty L dają wzorem 3.12 [61]:

$$\mathrm{L} = \sum_{\mathrm{i}}(\mathrm{y_i} - \sum_{\mathrm{p}} \beta_{\mathrm{p}} \cdot \mathrm{x_{ip}})^2 + \lambda \sum_{\mathrm{p}} |\beta_{\mathrm{p}}| \tag{3.12}$$

gdzie:

L – funkcja straty, która odzwierciedla ujemne konsekwencje błędnej decyzji, tj. zaliczenia próbki *i* do klasy *A*, gdy w rzeczywistości próbka ta należy do klasy *B*;

x_{ip} – wartość p-tej cecha *i*-tej obserwacji;

y_i – odpowiedź klasyfikatora dla *i*-tego obserwacji;

β_p – współczynnik regresji *p*-tej cechy;

λ – współczynnik kary.

Poprzez wykorzystanie członu z kosztami cech, tzn. $\lambda \sum_p |\beta_p|$, metoda LASSO prowadzi do rozwiązania, w którym większość współczynników β_p przyjmuje wartości zerowe. Stąd wniosek, że współczynniki cech redundantnych (ang. *redundant*) lub najmniej istotnych (ang. *irrelevant*) stają się zerowe [61]. Metoda LASSO jest bardzo efektywna, jeżeli w zbiorze uczącym znajduje się niewiele próbek i dużo cech nieistotnych [61,371]. Wadą metody LASSO jest założenie liniowości przestrzeni cech, co wiąże się z tym, że metoda osiąga gorsze rezultaty dla nieliniowych zależności, stąd też w [63] przedstawiono próbę rozwiązania tego problemu [61].

Wśród innych metod z grupy embedded można wymienić m.in.: metodę **Elastic Net** (metoda pokonuje ograniczenia funkcji kary występujące w metodzie LASSO poprzez dodanie do wzoru członu ($||\beta||^2$), znanego jako uregulowanie Tikhonov) [286,372-375]; metodę **SVM-RFE** (ang. *Support Vector Machine – Recursive Feature Elimination)* – jest to metoda rekurencyjnej eliminacji cech wykorzystywana m.in. w bioinformatyce, gdzie występują ogromne zbiory cech; metoda może nie być obiektywna przy silnie skorelowanych cechach) [372,376-378]; metodę **Regresji Grzbietowej** (ang. *Ridge Regression*) – metoda ta wprowadza do modelu pewną stałą, co rozwiązuje problem współliniowości zmiennych objaśniających oraz redukuje ich liczbę) [286,379-381] i inne.

3.5 Metody redukcji wymiarowości wektora cech a metody selekcji cech.

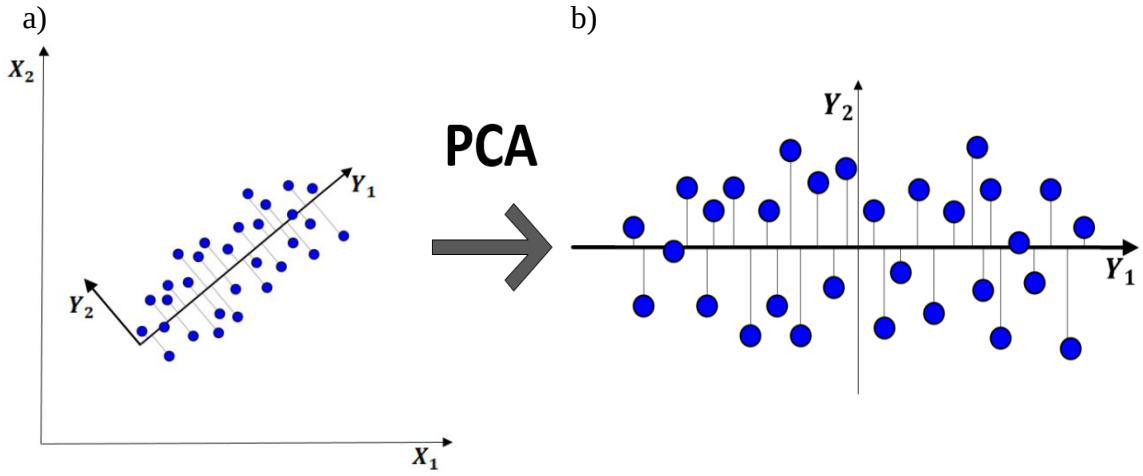
Metody selekcji cech stanowią podzbiór metod, które można ogólnie nazwać metodami redukcji wymiarowości przestrzeni cech. Poza metodami selekcji cech, dokonującymi wyboru cech najbardziej istotnych z punktu widzenia rozpatrywanego kryterium, do grupy metod redukcji wymiarowości przestrzeni cech należą metody dokonujące transformacji pierwotnej przestrzeni cech do innej przestrzeni, korzystniejszej z uwagi na postawione kryterium.

Podstawową różnicą w obu rodzajach metod redukcji przestrzeni cech jest to, że metody selekcji operują zawsze na pierwotnych cechach, natomiast metody transformacyjne operują na kombinacjach liniowych pierwotnych cech. W związku z tym, podczas gdy podzbiory cech wybierane za pomocą metody selekcji mają bezpośrednią interpretację rzeczywistą, podzbiory cech otrzymywane jako efekt zastosowania metod transformacyjnych, nie są interpretowalne w sposób bezpośredni (choć najczęściej można dokonać ich rzutowania na pierwotną przestrzeń cech). Tak więc w przypadku metod transformacyjnych nie jest możliwe np. utworzenie rankingu pierwotnych cech ani stwierdzenie, która z nich najlepiej spełnia warunki funkcji kryterialnej [382-383].

Najczęściej stosowaną metodą dokonującą transformacji pierwotnej przestrzeni cech jest Analiza Składowych Głównych (ang. *Principal Component Analysis*, *PCA*) [384-389]**,** znana także jako transformacja Karhunena-Loeve'go (ang. *Karhunen-Loeve Transformation*, *KLT*) [385,390]. Metoda ta polega na wyborze *k*-ortogonalnych wektorów *n*-wymiarowych, które najlepiej reprezentują dane, przy czym $k \leq n$. Zasada działania metody polega na rzutowaniu oryginalnych danych na przestrzeń rozpiętą przez *k* wybranych wektorów (składowe główne). W ten sposób uzyskuje się redukcję wymiaru wektorów cech z wymiaru *n* do wymiaru *k* [385]. Podczas działania metody PCA, na wejście algorytmu podawana jest macierz obserwacji, na podstawie których będą wyznaczane główne składowe (tj. wektory bazowe nowej przestrzeni). Algorytm w wyniku zwraca macierz współrzędnych znalezionych wektorów bazowych [391]. Kierunki kolejnych wektorów bazowych są dobierane w taki sposób, aby maksymalizować wariancję rzutowanych na nie danych [392] (Rysunek 3.6).

Zaletą metody PCA jest więc tworzenie nowych nieskorelowanych cech o wysokiej wariancji oraz redukcja wymiarowości uzyskana dzięki odrzuceniu cech z małą

wariancją. Wadą metody PCA jest to, że o tym, które cechy zostaną wykorzystane decydują najwyższe wartości wariancji, a to nie gwarantuje dobrych wyników klasyfikacji, ponadto po wyznaczeniu kombinacji liniowych pierwotnych cech tracone jest znaczenie cech [384]. Metoda ta ma wiele zastosowań, m.in. w kompresji sygnałów czy redukcji statystycznych zbiorów danych [391].



Rysunek 3.6. Kierunki wektorów bazowych maksymalizujące wariancję pierwotnego zbioru cech; a) pierwotna przestrzeń cech; b) przestrzeń cech po transformacji za pomocą metody PCA.
Źródło: Opracowanie własne

Algorytm metody PCA przebiega następująco [384-385,391]:

1. Unormowanie cech.
2. Wyznaczenie składowych głównych (wektorów własnych macierzy kowariancji cech).
3. Posortowanie składowych głównych od składowych o najwyższej wariancji do składowych o najniższej wariancji.
4. Wybór *k* składowych głównych o najwyższej wariancji i usunięcie pozostałych składowych.
5. Rzutowanie oryginalnych danych na przestrzeń rozpiętą na bazie *k* wybranych wektorów własnych (składowych głównych).

Na starcie algorytmu dane wejściowe (tj. wektory cech) są normowane w ten sposób, aby każda cecha przyjmowała wartości z tego samego przedziału zmienności. Można to uczynić za pomocą przesunięcia wartości średniej do zera oraz unormowanie

wariancji do jedności. Dzięki zastosowaniu takiego rozwiązania cechy szerzej rozłożone nie zdominują cech mocniej skoncentrowanych. W kroku kolejnym następuje obliczenie składowych głównych, poprzez wyliczenie *k*-ortonormalnych wektorów, które tworzą bazę dla unormowanych wcześniej danych wejściowych. Wektory te są wektorami jednostkowymi, prostopadłymi do pozostałych wektorów z utworzonej bazy [385,392]. W celu wyznaczenia wektorów własnych należy najpierw wyliczyć wartości własne λ_1^S, λ_2^S, λ_3^S, ... , λ_n^S macierzy rozproszenia danych, np. macierzy kowariancji *S* [385].

Zakładając, że dane wejściowe reprezentuje zestaw *n* wektorów cech $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(n)}\}$, macierz kowariancji przyjmuje postać (wzór 3.13) [384]:

$$cov = \begin{pmatrix} cov(X_1, X_1) & cov(X_1, X_2) & cov(X_1, X_3) & \cdots & cov(X_1, X_n) \\ cov(X_2, X_1) & cov(X_2, X_2) & cov(X_2, X_3) & \cdots & cov(X_2, X_n) \\ cov(X_3, X_1) & cov(X_3, X_2) & cov(X_3, X_3) & \cdots & cov(X_3, X_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ cov(X_n, X_1) & cov(X_n, X_2) & cov(X_n, X_3) & \cdots & cov(X_n, X_n) \end{pmatrix} \quad (3.13)$$

przy czym kowariancja między dwoma cechami (*X* i *Y*) jest wyznaczana z wzoru 3.14 [384]:

$$cov(X, Y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X}) \cdot (Y_i - \bar{Y})}{(n-1)} \quad (3.14)$$

gdzie:

$\bar{X}$- średnia wartość cechy *X,*

$\bar{Y}$- średnia wartość cechy *Y.*

Kowariancje, które leżą na przekątnej macierzy, są równe wariancjom poszczególnych cech. Po wyznaczeniu macierzy kowariancji obliczane są wartości własne macierzy kowariancji oraz odpowiadające im wektory własne. Z uwagi na to, że macierz kowariancji jest kwadratowa i ma wymiar $n \times n$, posiada ona n wektorów własnych. Wektory własne przedstawiane są w postaci jednostkowej [384].

Kolejnym krokiem jest posortowanie wektorów własnych według odpowiadających im wartości własnych, w kolejności od największej do najmniejszej wartości własnej. Następnie wybranych zostaje *k* pierwszych wektorów własnych, w celu umożliwienia transformacji danych pierwotnych do przestrzeni *k*-wymiarowej.

W ostatnim kroku algorytmu dane pierwotne są rzutowane na przestrzeń rozpiętą na wybranych wektorach własnych (wzór 3.15) [384]:

$$\text{Dane\_końcowe}^{\text{T}} = \text{Wektor\_cech}^{\text{T}} \times \text{Dane}^{\text{T}} \tag{3.15}$$

gdzie:

$Dane^T$ – macierz z danymi wejściowymi, w której każdy wiersz oznacza jeden z wymiarów (jedną cechę), a każda kolumna oznacza pojedynczy przypadek.

$Wektor\_cech^T$ – wektor składających się z wybranych wektorów własnych

$Dane\_końcowe$ – nowa macierz danych o wymiarach $m \times k$.

Rozdział IV
Algorytmy genetyczne

Algorytmy genetyczne i inne pokrewne im metody symulowanej ewolucji są przykładem jednej z częściej wykorzystywanych heurystycznych metod optymalizacyjnych. Z uwagi na to, że są one podstawą zaproponowanej i testowanej w dalszej części pracy metody selekcji cech na potrzeby interfejsu mózg-komputer, niniejszy rozdział zawiera krótkie omówienie podstawowych ich cech oraz zasad działania. Opis algorytmów genetycznych rozpoczęto od ogólnego wprowadzenia do tematyki metod symulowanej ewolucji wzorowanych na metodach doboru naturalnego i zasadach dziedziczenia. Następnie zdefiniowano podstawowe pojęcia wykorzystywane do opisu algorytmu genetycznego. W kolejnym podrozdziale przedstawiono schemat klasycznego algorytmu genetycznego opracowanego przez Johna Hollanda. W trzech ostatnich podrozdziałach przedstawiono trzy zadania realizowane w trakcie działania algorytmu: kodowanie, reprodukcję i selekcję.

4.1 Metody symulowanej ewolucji

W efekcie obserwacji procesów zachodzących w środowisku naturalnym rozwinięte zostały tak zwane metod symulowanej ewolucji. Metody te znalazły swoje zastosowanie głównie do przeszukiwania przestrzeni potencjalnych rozwiązań problemów optymalizacyjnych [27,82,89,393-397]. Metody symulowanej ewolucji rozwiązując problemy optymalizacyjne naśladują procesy ewolucji zachodzące w środowisku, korzystając przy tym z mechanizmów genetyki, dziedziczności oraz zasad doboru naturalnego [398]. Istnieje wiele metod symulowanej ewolucji, wykorzystujących różne parametry, operatory i techniki genetyczne. Wśród najczęściej stosowanych należy wymienić między innymi [85,89,393-394,399-401]:

1. Algorytmy genetyczne (ang. *Genetic Algorithms, GA*) [4,22,30,40,69,83,89-90,397,425].
2. Programowanie genetyczne (ang. *Genetic Programming, GP*); często programowanie genetyczne jest uważane za podgrupę algorytmów genetycznych [27,30,87,89,109,214]
3. Programowanie ewolucyjne (ang. *Evolutionary Programming, EP*) [251,401].
4. Strategie ewolucyjne (ang. *Evolutionary Strategies, ES*) [308,426].
5. Przeszukiwanie rozproszone (ang. *Scatter Search*) [308,440,447].
6. Neuroewolucje (ang. *Neuroevolution*) [5,27,57.9,107].

Algorytmy genetyczne [4,22,40,69,82-83,90,402-432] są jedną z heurystycznych metod rozwiązywania problemów optymalizacyjnych [433-434]. Algorytmy genetyczne są algorytmami wyszukiwania, opartymi na mechanizmach genetyki oraz naturalnej selekcji (doboru) organizmów. Łączą one w sobie mechanizmy przetrwania najsilniejszych osobników danego gatunku z losową wymianą informacji między tymi osobnikami [393].

Algorytmy genetyczne, zaproponowane przez Johna Hollanda [81], są to algorytmy probabilistyczne, które rozpoczynają się od zainicjowania początkowej populacji rozwiązań wybranych z wszystkich możliwych rozwiązań danego problemu optymalizacyjnego, a następnie rozwijają się poszukując coraz lepszych wersji rozwiązania postawionego problemu. Nowe rozwiązania (osobniki) są tworzone z wykorzystaniem operatorów genetycznych wzorowanych na procesach występujących w naturze.

Algorytmy genetyczne są globalnymi technikami optymalizacji, pozwalającymi uniknąć wielu wad, które są specyficzne dla wielu lokalnych technik wyszukiwania, wykorzystywanych w trudnych przestrzeniach wyszukiwania [435].

Głównym założeniem algorytmów genetycznych jest losowy wybór populacji punktów należących do przeszukiwanej przestrzeni zamiast przeszukiwania całej przestrzeni. Kolejnym krokiem jest modyfikacja tej przestrzeni podobna do procesów, jakie występują podczas ewolucji w środowisku naturalnym. Zatem w każdym pokoleniu (każdej iteracji), aktualnie sprawdzana populacja staje się populacją coraz lepiej przystosowanych osobników (osobniki te reprezentują rozwiązania, które są coraz bliższe rozwiązaniu optymalnemu [436]. W środowisku naturalnym mechanizmy ewolucyjne są adaptacyjne (umożliwiają dostosowanie się gatunków do nowych warunków).

Dzięki tym mechanizmom ewolucyjnym gatunki gorzej przystosowane wymierają, ustępując miejsca nowym gatunkom, które powstają i są lepiej dostosowane do warunków środowiskowych, w których żyją [437]. Analogicznie do środowiska naturalnego, „dobór naturalny" decyduje o szansach przeżycia sztucznych osobników oraz wydania przez nich potomstwa, zgodnie z zasadą, że im jest lepsze przystosowanie osobnika do danego środowiska, tym większe jest prawdopodobieństwo jego przeżycia i wydania potomstwa [27]. Stąd też, algorytm genetyczny jest ewolucją sztucznie wygenerowanych osobników Każdy z tych osobników jest potencjalnym rozwiązaniem odpowiednio zakodowanym, który żyje i ewaluuje w sztucznym środowisku.

Do głównych zalet algorytmów genetycznych należą m.in. [438]:

- rozpoczynanie przeszukiwania przestrzeni rozwiązań nie od pojedynczego rozwiązania, lecz z pewnej populacji rozwiązań;
- przetwarzanie wyłącznie zakodowanej postaci parametrów zadania;
- korzystanie tylko z funkcji celu (bez pomocniczych informacji lub pochodnych);
- wykorzystywanie probabilistycznych zamiast deterministycznych reguł wyboru;
- naśladowanie procesów zachodzących w środowisku naturalnym;
- brak konieczności korzystania z informacji pomocniczych.

Wadą algorytmów genetycznych jest to, że nie ma natomiast gwarancji znalezienia jednego rozwiązania optymalnego; możliwe jest tylko znalezienie lepszego lub gorszego przybliżenia rozwiązania optymalnego. Rezultat pracy algorytmów genetycznych „*może być punktem wyjściowym do poszukiwań końcowych przy pomocy klasycznych algorytmów optymalizacji"* [244,439]. Powszechnie uważa się, że niemożliwe jest stworzenie wyłącznie jednego algorytmu genetycznego, który byłby uniwersalny i umożliwiał rozwiązywanie wszelkich problemów i zagadnień optymalizacyjnych. Jednakże możliwe jest dopasowywanie algorytmu genetycznego do aktualnie rozwiązywanego problemu [440,441]. Pozwala to stworzyć algorytm genetyczny, który działa w sposób wystarczająco dobry dla pewnego konkretnego problemu lub też dla szczególnej grupy problemów [441].

Programowanie genetyczne to technika bardzo podobna do algorytmów genetycznych. Główna różnica między tymi dwoma metodami tkwi w reprezentacji rozwiązań problemu optymalizacyjnego. W algorytmie genetycznym rozwiązanie problemu reprezentowane jest za pomocą ciągu liczb, natomiast w programowaniu

genetycznym rozwiązaniem jest program o strukturze drzewiastej, składający się z węzłów i gałęzi. Rozwiązania są tutaj oceniane tak jak w algorytmie genetycznym na podstawie funkcji przystosowania, a operatory genetyczne krzyżowania i mutacji stosowane są na poszczególnych węzłach i gałęziach drzewa. Końcowym efektem algorytmu jest rozwiązanie reprezentujące najlepszy kod programu [400,442-445].

Programowanie ewolucyjne jest odmianą programowania genetycznego, charakteryzującą się wykorzystywaniem tylko operacji selekcji i mutacji, z pominięciem operacji krzyżowania. Idea programowania ewolucyjnego polega na tym, że w każdej populacji każde rozwiązanie rodzicielskie poprzez wykorzystanie operatora mutacji generuje swojego potomka (prawdopodobieństwo mutacji najczęściej ma rozkład równomierny). W trakcie selekcji wybierana jest pewna liczba najlepszych osobników z całego zbioru osobników rodzicielskich oraz ich potomków. Dzięki temu, że najlepszy osobnik przechowywany jest zawsze, nie jest on gubiony w procesie selekcji [442-443].

Strategie ewolucyjne proces poszukiwania rozwiązania opierają, podobnie jak programowanie ewolucyjne, przede wszystkim na zastosowaniu operatora mutacji i selekcji. Mutacja i selekcja decydują o przebiegu algorytmu i mają największy wpływ na jego zbieżność. Wyróżnia się kilka strategii ewolucyjnych, różniących się sposobem przeprowadzenia tych dwóch operacji. Wśród najczęściej stosowanych strategii ewolucyjnych można wymienić m.in. strategię: $(1+1)$, $(\mu+\lambda)$, (μ,λ), gdzie μ rodziców produkuje λ dzieci. Strategia $(\mu+\lambda)$ dopuszcza przeżycie zarówno rodziców jak i dzieci, a najlepszy osobnik zawsze kopiowany jest do następnej populacji (strategia elitarna). W strategii (μ,λ) najlepsze dzieci λ przeżywają i zastępują swoich rodziców (brak tych rodziców w następnej populacji) [400,442-443].

W przeszukiwaniu rozproszonym bardzo często wykorzystywane są metody deterministyczne zamiast stochastycznych. W strategii tej populację tworzą najlepsze osobniki lub ich fragmenty. Populacja powinna posiadać osobniki bardzo zróżnicowane, które nie muszą tworzyć rozwiązań dopuszczalnych. Tworzony jest także zbiór referencyjny (ang. *reference set*) zawierający najlepsze rozwiązania wybrane z populacji. Wielkość zbioru referencyjnego jest niewielka w porównaniu do całej populacji, która najczęściej jest kilkanaście razy liczniejsza niż zbiór referencyjny. W operacjach genetycznych (operacji krzyżowania i mutacji) biorą udział tylko osobniki ze zbioru referencyjnego (wszystkie osobniki znajdujące się aktualnie w zbiorze). Gdy liczba nowo utworzonych osobników przekroczy liczność aktualnej populacji, wówczas, aby zachować stałą liczność populacji (dodatkowy wymóg to zachowanie różnorodności

populacji), wybierane są tylko najlepsze osobniki z dotychczasowych i nowych osobników. Wybrane w ten sposób najlepsze osobniki zastępują osobniki z dotychczasowego zbioru referencyjnego (liczba osobników w zbiorze referencyjnym nie ulega zmianie) [446-449].

Charakterystyczną cechą ostatniej z wskazanych metod symulowanej ewolucji, nazywanej neuroewolucją, jest postać osobników występujących w populacji. Osobnikiem jest tutaj bowiem sztuczna sieć neuronowa. Przetwarzanie populacji osobników, czyli populacji sieci neuronowych, odbywa się na tych samych zasadach, jak w przypadku algorytmów genetycznych. W wyniku działania operatorów genetycznych zmianom mogą ulegać zarówno wartości wag i liczba warstw poszczególnych sieci neuronowych, jak i liczba neuronów w każdej z warstw. Najlepiej przystosowane osobniki (sieci neuronowe) stają się rodzicami nowego pokolenia. Metoda ta ma zastosowanie w tych przypadkach, kiedy trudno jest z góry zdefiniować całą strukturę sieci neuronowej, ale można ocenić poprawność sieci generowanych w kolejnych pokoleniach. Wyróżnia się dwa typy algorytmów neuroewolucyjnych: o stałej strukturze sieci neuronowych (zastępuje standardowe metody nauki z nauczycielem) oraz o zmiennej strukturze sieci neuronowych (wymagają określenia tylko stałej liczby wejść i wyjść sieci neuronowych; dzięki temu możliwe jest tworzenie struktur, które będą w stanie same ewaluować, tzw. *TWEANN* – ang. *Topology and Weight Evolving Artificial Neural Networks*) [450-452].

4.2 Algorytm genetyczny – podstawowe pojęcia

Do zdefiniowania algorytmu genetycznego zdolnego do rozwiązania postawionego zadania optymalizacyjnego konieczne jest ustalenie[4,22,40,69,83,90,453]:

1. Genetycznej reprezentacji rozwiązania rozwiązywanego problemu optymalizacyjnego.
2. Funkcji, która będzie wykorzystywana do oceny zwracanych przez algorytm rozwiązań.
3. Operatorów genetycznych, które zostaną wykorzystane do tworzenia nowych rozwiązań.
4. Metody wyboru rozwiązań przenoszonych do kolejnych pokoleń.
5. Stałych wartości dodatkowych parametrów algorytmu, takich jak np. wielkość populacji, prawdopodobieństwo stosowania kolejnych operatorów genetycznych, liczba pokoleń, itp.

W procesie definiowania wskazanych powyżej elementów algorytmu genetycznego wykorzystywane są pojęcia zaczerpnięte z genetyki, takie jak: gen, chromosom, allel, populacja, funkcja przystosowania itp. Dla zachowania jasności wywodu pojęcia te zostały zdefiniowane poniżej. Z kolei podstawowe metody stosowane w procesie kodowania rozwiązań, selekcji osobników oraz reprodukcji zostały pokrótce omówione w trzech ostatnich podpunktach niniejszego rozdziału.

Osobnik to „*zespół potencjalnych genów, tzw. genotyp*" [27]. Osobnik jest więc jednostką, która posiada *„cechy odróżniające ją od innych jednostek oraz zestaw genów opisujących (kodujących) te cechy"* [454]. W przypadku algorytmów genetycznych osobniki są punktami w przestrzeni (potencjalnych) rozwiązań przeszukiwana, a zatem pełnią rolę lepszego lub gorszego reprezentanta potencjalnego rozwiązania danego problemu. [434] [436].

Chromosom to „*łańcuch, ciąg uporządkowanych genów, zakodowana postać potencjalnego rozwiązania (punktu przestrzeni poszukiwań)"* [436]. Chromosom (genotyp) stanowi kod informacyjny, który przekazywany jest kolejnym pokoleniom. Kod ten określa proces budowy i rozwoju osobnika, począwszy od jego powstania, aż do jego śmierci. Definicja chromosomu w algorytmie genetycznym w dużym stopniu pokrywa się z definicją osobnika, ponieważ najczęściej chromosom jest równoważny osobnikowi. W przypadku bardziej skomplikowanych problemów optymalizacyjnych poszczególne cechy problemu są kodowane w odrębnych chromosomach. W takim przypadku osobnik stanowi pojedynczy punkt w przestrzeni rozwiązań i składa się z zespołu chromosomów, przy czym każdy z chromosomów koduje jedną cechę rozwiązania.

Gen jest pojedynczym elementem chromosomu [436], który *„może przyjmować jedną z kilku wartości zwanych allelami"* [238]. W klasycznym algorytmie genetycznym, w którym osobniki są kodowane binarnie, możliwe są dwa allele: 1 lub 0 [436]. W przypadku innych rodzajów kodowania mogą występować inne allele. Na przykład, gen kodujący cechę „*czas trwania procesu uczenia*", może mieć allele: bardzo długo, długo, średnio, krótko, bardzo krótko, gen kodujący płeć użytkownika interfejsu może mieć allele: żeński, męski, a gen kodujący rodzaj interfejsu może mieć allele: SSVEP, P300, MI itd.

Osobnik ma dwie odpowiadające sobie postacie: genotyp oraz fenotyp. **Genotyp** to *„ciąg symboli stanowiący kod genetyczny osobnika; zapis zbioru jego cech"* [238]. Genotyp stanowi więc indywidualny zbiór informacji osobnika i określa w sposób

bezpośredni jego strukturę genetyczną. W środowisku naturalnym genotyp osobnika jest kodem RNA lub DNA [454]. Z kolei **fenotyp**[2] to zestaw zewnętrznych cech osobnika [436]. Jako że *„zewnętrzne cechy (fenotyp) osobnika zależą od zestawu jego genów (genomu[3])* [455] [456], więc każdy postać fenotypu jest ściśle uwarunkowana przez genotyp osobnika.

Zbiór osobników o zadanej liczebności nazywany jest **populacją**. Populacja jest to więc zbiór reprezentacji potencjalnych rozwiązań danego problemu optymalizacyjnego [4,22,40,69,83,90,244,436,438,454] wybranych z pełnej przestrzeni rozwiązań. Osobniki znajdujące się w populacji oddziałują na siebie i rozmnażają się na danym etapie obliczeń (nazywanym **pokoleniem**).

Oceny jakości rozwiązań zakodowanych w osobnikach występujących w populacji dokonuje się za pomocą funkcji przystosowania [27]. **Funkcja przystosowania** jest funkcją informującą, które osobniki reprezentują lepsze, a które gorsze rozwiązania danego problemu optymalizacyjnego [437]. Zarówno funkcja przystosowania jak i dziedzina możliwych rozwiązań problemu optymalizacyjnego są zależne od konkretnego optymalizowanego problemu i są znane przed uruchomieniem algorytmu. Funkcja przystosowania powinna być określona dla każdego punktu dziedziny, którą stanowi zbiór potencjalnych osobników.

Funkcja przystosowania posiada bardzo ważną rolę weryfikującą, przez co wywiera znaczący wpływ na działanie algorytmu genetycznego. Funkcja przystosowania, która została niewłaściwie zdefiniowana, może być przyczyną zbyt szybkiej dominacji pojedynczego osobnika lub grupy osobników, uniemożliwiając działanie operatorów genetycznych, które mógłby wyprowadzić proces przeszukiwania przestrzeni rozwiązań z „doliny” optimum lokalnego zakodowanego w dominującym osobniku, co w następstwie może prowadzić do utraty zbieżności algorytmu do rozwiązania optymalnego. Definiując funkcję przystosowania powinno się brać pod uwagę także dodatkowe ograniczenia nakładane na rozwiązanie [4,22,40,69,83,90,244,439,457].

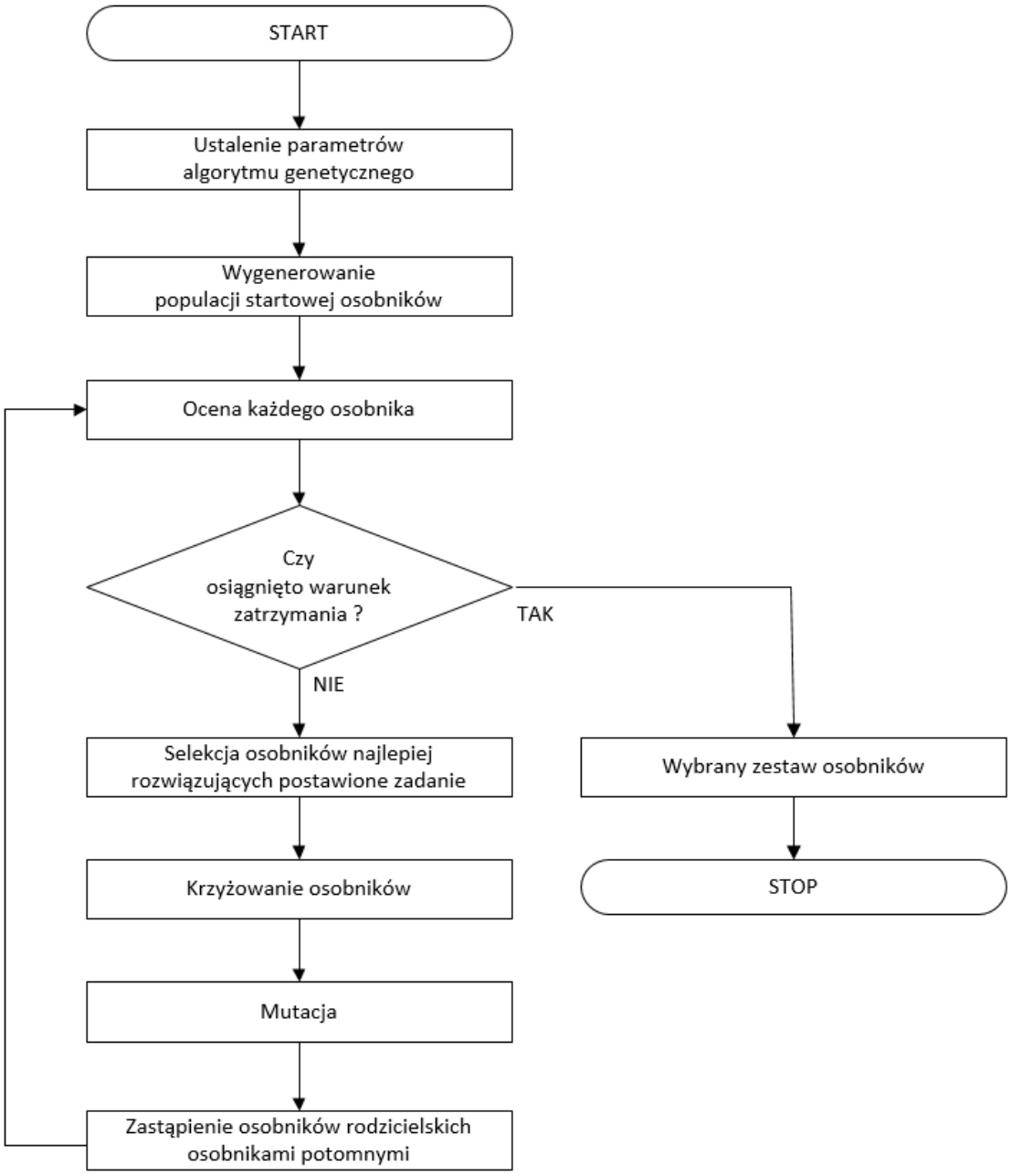
[2] Fenotyp (w biologii) – zespół cech organizmu (anatomicznych, fizjologicznych, morfologicznych i in.) ukształtowanych pod wpływem czynników dziedzicznych (genów) i warunków środowiska.
[3] Genom (w biologii) - podstawowy zespół chromosomów jądra komórkowego, zawierający pojedynczy zestaw genów.

4.3 Schemat klasycznego algorytmu genetycznego

Klasyczny algorytm genetyczny jest jednym z najczęściej wykorzystywanych algorytmów genetycznych. Przebieg klasycznego algorytmu genetycznego, tj. algorytmu Hollanda z 1975 r. [81] pokazany został na Rysunku 4.1 [4,22,40,69,83-90]. Algorytm ten był następnie rozwijany w kolejnych latach. Przebieg klasycznego algorytmu Hollanda można przedstawić w następujących krokach [81]:

1. Uruchom algorytm i ustal jego parametry.
2. Wygeneruj startową populację osobników.
3. Oceń każdego osobnika występującego w aktualnej populacji, zgodnie z przyjętą funkcją przystosowania.
4. Jeśli zostanie spełniony warunek zakończenia (np. zostanie wykonana zakładana liczba iteracji, albo zostanie osiągnięta założona dokładność rozwiązania), przerwij algorytm i wybierz najlepszego osobnika z aktualnej populacji. Jeżeli warunek zatrzymania nie został spełniony, przejdź do kroku 5.
5. Dokonaj selekcji osobników najlepiej rozwiązujących postawione zadanie, zgodnie z przyjętą metodą selekcji.
6. Utwórz nowe osobniki (osobniki potomne) poprzez przeprowadzenie operacji genetycznych (mutacji oraz krzyżowania) na wybranych osobnikach rodzicielskich. Zastąp osobniki rodzicielskie osobnikami potomnymi, a następnie wróć do kroku 3.

Zgodnie z zaprezentowanym schematem, przed rozpoczęciem właściwego algorytmu genetycznego należy wykonać dwa zadania: zdefiniować parametry algorytmu oraz wygenerować startową populację osobników. Wśród parametrów, które należy ustalić znajdują się: funkcja przystosowania, wielkość populacji, metoda selekcji, prawdopodobieństwo mutacji i krzyżowania, metoda mutacji i krzyżowania, metoda kodowania rozwiązań do postaci osobników oraz warunek zatrzymania algorytmu. Poszczególne parametry są wykorzystywane w kolejnych etapach algorytmu. Po zdefiniowaniu parametrów tworzona jest startowa populacja osobników. Populacja ta zawiera wybrane rozwiązania danego problemu optymalizacyjnego [458]. Populację początkową tworzy się na ogół losowo, zgodnie z rozkładem normalnym [244].



Rysunek 4.1. Schemat działania klasycznego algorytmu genetycznego.
Źródło: Opracowanie własne.

Osobniki tworzące populację początkową mogą występować w postaci fenotypów (rzeczywistych rozwiązań zadania), albo w postaci genotypów, czyli rozwiązań zakodowanych do postaci ciągu genów, zgodnie z przyjętym schematem kodowania. Jeżeli do populacji początkowej wybierane są rozwiązania rzeczywiste, stanowiące fenotypy osobników, to fenotypy te muszą zostać zakodowane do postaci genotypów przed wykonaniem operacji genetycznych. Jeżeli natomiast populacja początkowa składa się z genotypów osobników, to fenotypy muszą zostać wyznaczone na etapie oceny

przystosowania osobników. Do najważniejszych rodzajów kodowania rozwiązań do postaci genotypów można zaliczyć m.in. kodowanie binarne [4,22,40,69,83,90,438,453], kodowanie Graya [238,438,459], kodowanie zmiennopozycyjne itd. W przypadku klasycznego algorytmu genetycznego stosowane jest kodowanie binarne, czyli fenotyp osobnika jest przedstawiany za pomocą zero-jedynkowego ciągu genów.

W klasycznym algorytmie genetycznym przebieg procesu ewolucji osobników (rozwiązań) sterowany jest przy pomocy funkcji przystosowania. Osobniki o wysokich wartościach ustalonej funkcji przystosowania są przeważnie przenoszone do dalszych etapów ewolucji, osobniki o niskich wartościach tej funkcji najczęściej są eliminowane. Stąd po wybraniu osobników do populacji początkowej trzeba dokonać ich oceny, czyli określić jak dobrze rozwiązują one postawiony problem optymalizacyjny. Etap oceny osobników kończy się sprawdzeniem, czy został spełniony warunek zatrzymania algorytmu. Oczywiście najczęściej warunek zatrzymania nie jest spełniony na etapie oceny osobników z populacji startowej, niemniej jednak sytuacja taka jest możliwa, jeżeli warunkiem zatrzymania jest osiągnięcie ustalonej wartości funkcji przystosowania.

Jeżeli warunek zatrzymania nie został spełniony, następuje przejście do etapu selekcji osobników do kolejnej populacji (tzw. populacji rodzicielskiej). Osobniki do kolejnych populacji są wybierane na podstawie wyznaczonej w poprzednim etapie wartości funkcji przystosowania. Istotny jest tu fakt, że najczęściej wysoka wartość funkcji przystosowania osobnika nie oznacza automatycznego wyboru tegoż osobnika, a jedynie zwiększa prawdopodobieństwo jego wyboru. Wykorzystywany jest tu rozkład prawdopodobieństwa, ustalany na podstawie zdefiniowanych parametrów algorytmów, np. liczebności populacji czy tez liczby genów w osobniku. Wśród wielu standardowo stosowanych metod selekcji osobników można wymienić m.in.: metodę rangową [205] [27], metodę turniejową [27,205,460], metodę progową, metodę proporcjonalną (nazywaną powszechnie metodą koła ruletki) [27,81,436], metodę pojedynku itp.

W trakcie procesu selekcji osobniki słabe są usuwane ze zbioru rozwiązań, a w ich miejsce wstawiane są nowe osobniki, które są tworzone poprzez reprodukcję osobników o wysokich wartościach funkcji przystosowania w procesie krzyżowania i mutacji [461]. W algorytmach genetycznych operatory genetyczne spełniają przede wszystkim funkcje związane z badaniem przestrzeni poszukiwań oraz analizą obszarów ekstremów lokalnych [205].W klasycznym algorytmie genetycznym stosowane są operatory

najprostsze, to jest jednopunktowe krzyżowanie i jednogenowa mutacja. Krzyżowanie jednopunktowe rozpoczyna się od wyboru pary osobników z populacji rodzicielskiej [81]. Osobniki z populacji rodzicielskiej są kojarzone w pary w sposób losowy, zgodnie z zadeklarowanym na początku prawdopodobieństwem krzyżowania. Następnie dla każdej pary osobników jest losowana w pozycja genu (tzw. locus [436]), która określa punkt krzyżowania, czyli punkt rozcięcia osobników rodzicielskich. Operacja krzyżowania kończy się wymianą odciętych fragmentów genotypu między osobnikami rodzicielskimi. Powstałe w wyniku tej operacji osobniki potomne zastępują osobniki rodzicielskie w aktualnej populacji.

Krzyżowanie jest uważane za najważniejszy operator algorytmu genetycznego. Jest tak dlatego, bo proces krzyżowania może zarówno zachowywać korzystne geny dotychczasowych osobników potomnych, jak i tworzyć osobniki zupełnie odmienne od osobników rodzicielskich [244]. Operator krzyżowania ma swój odpowiednik w procesach biologicznych – jest inspirowany przez reprodukcję heteroseksualną.

Po przeprowadzeniu operacji krzyżowania, następuje przejście do operacji mutacji. Najczęściej mutacja polega na wykonaniu losowych zmian w genach występujących w populacji. Podobnie, jak w przypadku operacji krzyżowania, mutacja jest wykonywana tylko na tych genach, których prawdopodobieństwo mutacji jest mniejsze od pewnej ustalonej wartości. Mutacja jednogenowa, stosowana w algorytmie klasycznym polega na zamianie wylosowanego genu z „0” na „1” lub też z „1” na „0” [462]. Czasami mutacja genu jest zastępowana przez mutację całego fragmentu osobnika za pomocą losowego łańcucha binarnego (tzw. makromutacja) [463]. Celem mutacji jest wprowadzenie różnorodności w nowej populacji oraz badanie zupełnie nowych obszarów przestrzeni optymalizowanego problemu. Dzięki mutacji możliwe jest stworzenie osobnika, który może być zarówno bardzo podobny, jak i bardzo różny od swojego „oryginalnego” poprzednika, a więc również bardzo bliski lub odległy w kategoriach funkcji przystosowania.

Stworzenie populacji, w której wybrane osobniki rodzicielskie zostały zastąpione utworzonymi na ich podstawie osobnikami potomnymi kończy jedną iterację klasycznego algorytmu genetycznego. Opisane etapy oceny, selekcji oraz reprodukcji osobników są powtarzane wielokrotnie, aż do momentu znalezienia rozwiązania o satysfakcjonującej wartości funkcji przystosowania lub do momentu wykonania założonej liczby iteracji. Wielokrotne powtarzanie operacji oceny, selekcji, krzyżowania i mutacji prowadzi najczęściej do uzyskania populacji o wyższej średniej wartości funkcji

przystosowania osobników, aniżeli wybrana losowo populacja początkowa. Ostatecznym wynikiem algorytmu jest najczęściej osobnik o najwyższej wartości funkcji przystosowania z ostatniej (lub dowolnej wcześniejszej) iteracji.

4.4 Kodowanie rozwiązań do postaci osobników

Pisząc o sposobach kodowania rozwiązań zadania optymalizacyjnego w postaci osobników przetwarzanych w algorytmie genetycznym Goldberg stwierdza: *„W pewnym sensie wybór kodu dla zadania rozwiązywanego przy użyciu algorytmu genetycznego nie stanowi żadnego problemu, gdyż programista jest ograniczony głównie swoją własną wyobraźnią.(...) Patrząc z innego punktu widzenia, ta swoboda wyboru stanowi wątpliwe błogosławieństwo dla niedoświadczonego użytkownika; widok całych szeregów możliwych sposobów kodowania może zarówno dodawać otuchy, jak i oszałamiać”* [464] [465]. Cytat ten podkreśla mnogość i różnorodność możliwych sposobów kodowania osobników, które można zastosować przy rozwiązywaniu problemów optymalizacyjnych za pomocą algorytmu genetycznego.

W procesie wyboru metody kodowania osobników stosowany jest cały szereg zasad. Jedną z podstawowych jest zasada minimalnego alfabetu. Mówi ona, że należy wybrać najmniejszy alfabet, w którym rozpatrywane zadanie wyraża się w sposób naturalny [464]. Dlatego też możliwe jest przyjęcie struktur innych niż wektory (np. drzewa, permutacje, macierze itd.). Ponadto podczas wyboru metody kodowania należy upodobnić zakodowane osobniki do rzeczywistych rozwiązań. Należy również do wybranej metody kodowania dobrać odpowiednio zaprojektowane specjalistyczne operatory genetyczne [465].Wśród najczęściej stosowanych metod kodowania należy wymienić: kodowanie binarne, kodowanie Graya, logarytmiczne oraz zmiennopozycyjne.

W **kodowaniu binarnym** [89,438,453,466-467] genotyp osobnika stanowi ciąg zer i jedynek [468]. W klasycznym algorytmie genetycznym wykorzystuje się system dwójkowy, gdzie każdy kolejny bit w kodzie dwójkowym jest odpowiednikiem kolejnej potęgi liczby 2 w systemie dziesiętnym [438].

Kodowanie Graya [238,438,459] jest wykorzystywane jako alternatywa kodowania binarnego. Jego zaletą jest to, że przejście pomiędzy dwoma sąsiadującymi liczbami wymaga zmiany tylko i wyłącznie jednego bitu. W Tabeli 4.1 przedstawiono

kodowanie pierwszych ośmiu cyfr z alfabetu arabskiego za pomocą kodowania binarnego i kodowania Graya. Jak można zauważyć w Tabeli 4.1 kodowanie Graya jest oszczędniejsze niż kodowanie binarne, ponieważ przejście z liczby siedem do liczby osiem, w kodzie binarnym wymagające zmiany wszystkich czterech bitów, a w przypadku kodu Greya wymaga zmiany tylko jednego bitu.

Liczba w kodowaniu dziesiętnym	**Liczba w kodowaniu binarnym**	**Liczba w kodowaniu Greya**
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0

Tabela 4.1 Porównanie kodowania binarnego z kodowaniem Graya.
Źródło: Opracowanie własne.

Kodowanie logarytmiczne wykorzystuje funkcję wykładniczą (ekspotencjalną) w celu znacznego zmniejszenia długości chromosomów [436]. W tym kodowaniu fenotyp osobnika ma postać funkcji wykładniczej [468]. Kodowanie logarytmiczne wykorzystywane jest do rozwiązywania problemów, które mają duże przestrzenie poszukiwań oraz wiele parametrów. Kodowanie to ma dwie zalety. Po pierwsze fenotypy osobników mogą przyjmować wartości rzeczywiste (a nie tylko całkowite, jak w standardowym kodowaniu binarnym i kodowaniu Graya), po drugie znacznie krótsza jest długość genotypów w porównaniu do wcześniej omówionych sposobów kodowania. Kodowanie logarytmiczne jest realizowane zgodnie z następującym wzorem 4.1 [469-471]:

$$[\alpha\beta bin] = (-1)^{\beta} e^{(-1)^{\alpha}[bin]_{10}} \tag{4.1}$$

gdzie:
α – pierwszy gen chromosomu – znak liczby wyrażonej za pomocą funkcji wykładniczej;
β – drugi gen chromosomu – znak wykładnika;
$[bin]_{10}$ – wartość dziesiętna zakodowanego ciągu (łańcuch binarny wykładnika).

Pierwsze dwa bity (a i b) nie mają bezpośredniego powiązania z wartością, służą jedynie do ustalenia znaku liczby i znaku potęgi [471]. Za pomocą tego kodowania możliwe jest zakodowanie za pomocą 5 bitów wartości z przedziału $(-e^7, e^7)$, co daje $(\sim -1096, \sim 1096)$. Na przykład ciąg binarny [10101] w kodowaniu logarytmicznym wyraża wartość:

$$(-1)^0 e^{(-1)^1[101]_{10}} = (1)e^{(-1)\cdot 5} = 0{,}00673795,$$

a ciąg binarny [01110] – wartość:

$$(-1)^1 e^{(-1)^0[110]_{10}} = (-1)e^{(1)\cdot 6} = -403{,}42879349.$$

W przypadku **kodowania zmiennopozycyjnego** każdy osobnik zapisany jest jako wektor liczb zmiennopozycyjnych o tej samej długości jak wektor rozwiązania. Dokładność kodowanego rozwiązania przy zastosowaniu kodowania zmiennopozycyjnego jest o wiele wyższa niż przy zastosowaniu kodowania binarnego [471-472]. W kodowaniu zmiennopozycyjnym genotyp przyjmuje postać (wzór 4.2) [473]:

$$(\mathrm{x}) = \mathrm{x}' = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \dots, \mathrm{x}_{n-1}, \mathrm{x}_n), \tag{4.2}$$

gdzie $(x_1, x_2, \dots, x_n) \in R$.

Wartość liczby zmiennoprzecinkowej oblicza się według wzoru 4.3 [474]:

$$\mathrm{LZP} = \mathrm{m} \cdot \mathrm{p}^{\mathrm{c}}, \tag{4.3}$$

gdzie: LZP – Wartość liczby zmiennoprzecinkowej; p – podstawa; m – mantysa; c – cecha.

W systemie dwójkowym wszystkie trzy elementy *m*, *p* i *c* będą zapisane dwójkowo za pomocą odpowiednio dobranego systemu kodowania liczb. Podstawa p zawsze będzie równa 2 [474]. Boryczka [474] podaje przykład genotypu zbudowanego z 8 bitów ponumerowanych od 0 do 7, w którym mantysa jest zawarta w 4 pierwszych bitach (od b0 do b3), a cecha 4 ostatnich (od b4 do b7) (Rysunek 4.2). Cecha jest tu liczbą całkowitą ze znakiem w kodzie U2[4].

[4] Kod uzupełnień do dwóch (w skrócie U2 lub ZU2) – system reprezentacji liczb całkowitych w dwójkowym systemie pozycyjnym. Jest obecnie najpopularniejszym sposobem zapisu liczb całkowitych w

Rysunek 4.2. Przykład chromosomu zbudowanego z 8 bitów ponumerowanych od 0 do 7.
Źródło: [474].

Stąd wartość cechy obliczana jest wg wzoru 4.4:

$$c = b7 \cdot (-2^3) + b6 \cdot 2^2 + b5 \cdot 2^1 + b4 \cdot 2^0 = (-8) \cdot b7 + 4 \cdot b6 + 2 \cdot b5 + 1 \cdot b4 \quad (4.4)$$

Mantysa jest natomiast 4-bitową liczbą stałoprzecinkową w kodzie U2. Pozycja przecinka jest wyznaczona pomiędzy bitami b1 i b2. Zatem wartość mantysy obliczana jest według wzoru 4.5 oraz 4.6 [474]:

$$m = b3 \cdot b2 \quad (4.5)$$

$$m = b3 \cdot (-2^1) + b2 \cdot 2^0 + b1 \cdot 2^{-1} + b0 \cdot 2^{-2} = (-2) \cdot b3 + 1 \cdot b2 + \frac{1}{2} \cdot b1 + \frac{1}{4} \cdot b0 \quad (4.6)$$

Wartość liczby zmiennoprzecinkowej wynosi zatem (wzór 4.7):

$$LZP = m \cdot p^c = m \cdot 2^c \quad (4.7)$$

Ograniczeniem w przypadku kodowania zmiennopozycyjnego jest rodzaj procesora oraz systemu operacyjnego (32-bitowego lub 64-bitowego), natomiast w przypadku kodowania binarnego ograniczeniem jest przyjęta liczba bitów wektora [471]. W celu zwiększenia dokładności można wprowadzić większą liczbę bitów, jednakże im więcej bitów, tym wolniejsze jest działania algorytmu. Według Davida E. Goldberga [464], reprezentacja zmiennopozycyjna osobnika i operacje wykonywane przy jej zastosowaniu są szybsze i dają w różnych cyklach algorytmu bardziej zbliżone wyniki.

systemach cyfrowych. Jego popularność wynika z faktu, że operacje dodawania i odejmowania są w nim wykonywane tak samo jak dla liczb binarnych bez znaku (Źródło: https://pl.wikipedia.org/wiki/Kod_uzupełnień_do_dwóch).

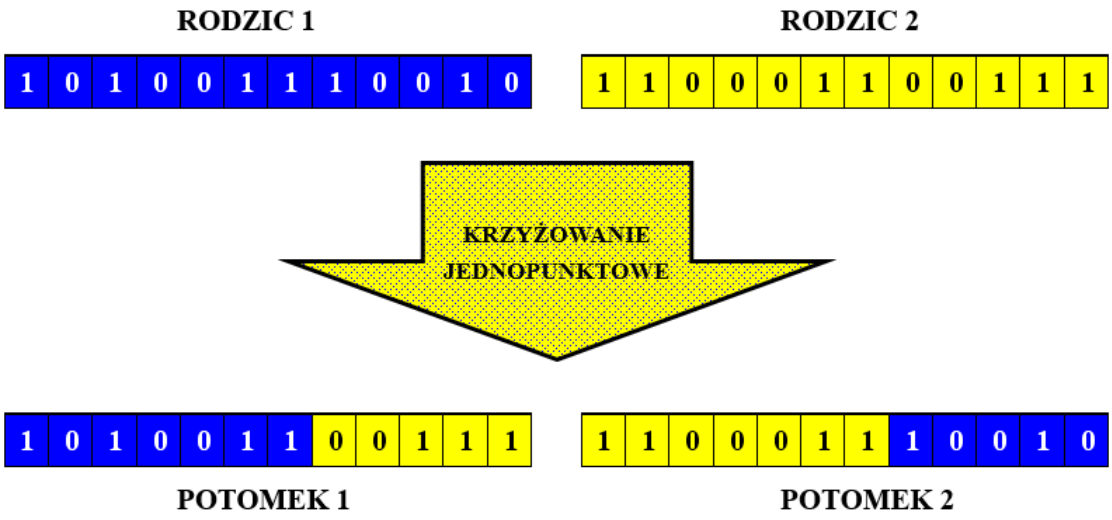
Ponadto możliwe jest uzyskanie lepszej dokładności, zwłaszcza w przypadku większych przestrzeni rozważań (kodowanie binarne wymaga zbyt długich reprezentacji osobnika). W celu uzyskania dużej dokładności (lepszej niż dla reprezentacji binarnych), możliwe jest wprowadzenie specjalistycznych operatorów genetycznych dostosowanych do mechanizmu kodowania zmiennopozycyjnego [473].

4.5 Operatory genetyczne

Dwie podstawowe operacje genetyczne wykorzystywane w klasycznym algorytmie genetycznym to krzyżowania i mutacji. Poza nimi czasami stosowana jest również operacja inwersji. W niniejszym punkcie zostanie podanych kilka podstawowych operatorów służących do wykonania wskazanych operacji.

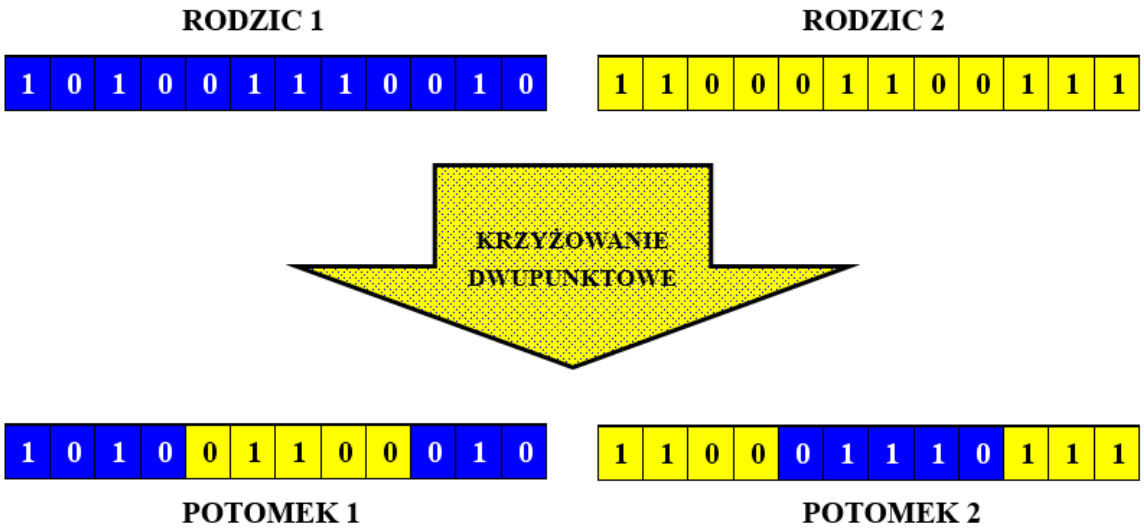
Wybór operatora krzyżowania jest uwarunkowany zastosowanym w algorytmie sposobem reprezentacji osobnika. Wśród częściej stosowanych można wymienić krzyżowanie: jednopunktowe, wielopunktowe, liniowe, rozproszone itp. Podczas operacji krzyżowania fragmenty kodu genetycznego pochodzące od dwóch osobników rodzicielskich podlegają wymianie; w ten sposób powstają dwa osobniki potomne. Krzyżowanie jest to więc *„proces rekombinacji genów, prowadzący do powstania nowych chromosomów (potomków) w wyniku wymiany fragmentów chromosomów rodziców”* [436]. Proces ten występuje także w środowisku naturalnym, gdy krzyżują się chromosomy rodziców, tworząc w ten sposób genom dzieci. Wynikiem tego jest to, że dziecko otrzymuje część genów od ojca i część genów od matki [454].

Krzyżowanie jednopunktowe (ang. *1-Point Crossover, Simple Arithmetic Crossover, One Point Crossover*) [475] [476] zostało już pokrótce opisane w podrozdziale 4.3. Opiera się ono na przecięciu genotypu osobników rodzicielskich w tym samym punkcie krzyżowania, a następnie na wymianie fragmentów genotypów znajdujących się po punkcie przecięcia (Rysunek 4.3).



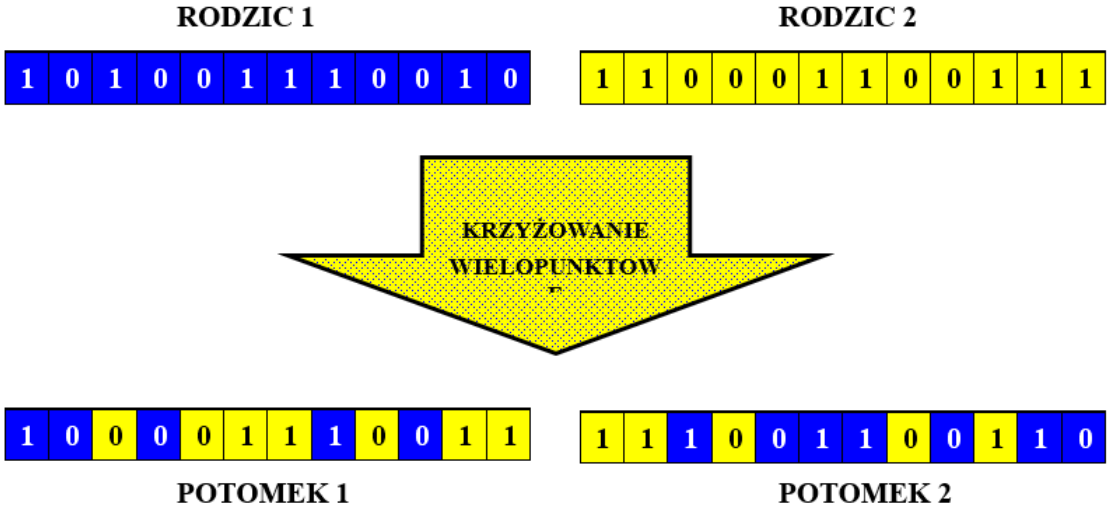
Rysunek 4.3. Przykład krzyżowania jednopunktowego.
Źródło: opracowanie własne.

Krzyżowanie dwupunktowe [81,238] (ang. *Two Point Crossover*) polega na przecięciu genotypu osobników rodzicielskich w dwóch punktach krzyżowania, a następnie na wymianie genów zawartych między punktami przecięcia (Rysunek 4.4).



Rysunek 4.4. Przykład krzyżowania dwupunktowego.
Źródło: opracowanie własne.

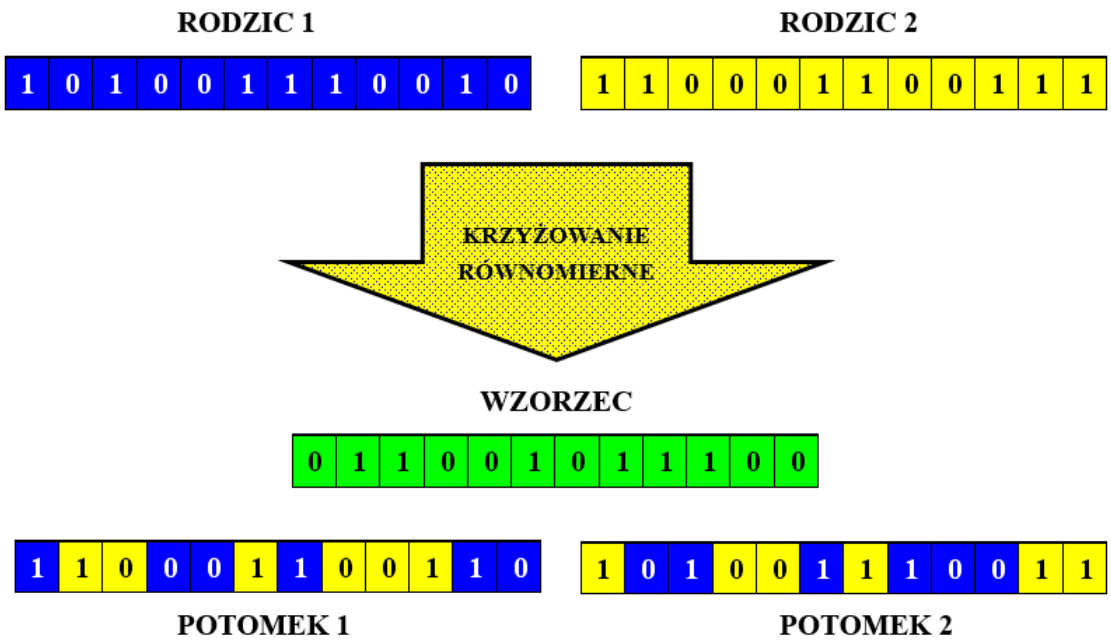
Krzyżowanie wielopunktowe [475-477] (ang. *k-Point Crossover, Multi* P*oint Crossover*) polega na przecięciu genotypu osobników rodzicielskich w wielu punktach krzyżowania (Rysunek 4.5). Punkty te wybierane są w sposób losowy bez powtórzeń. Zaletą tej formy krzyżowania jest usprawnienie procesu krzyżowania osobników o długich genotypach.



Rysunek 4.5. Przykład krzyżowania wielopunktowego.
Źródło: Opracowanie własne.

Krzyżowanie równomierne [475,478-481] (ang. U*niform Crossover*) polega na krzyżowaniu osobników zgodnie z wylosowanym wzorcem. To właśnie wzorzec decyduje, które geny będzie posiadał potomek (Rysunek 4.6). Przykładowo, gdy we wzorcu będzie występowała wartość 0, to pierwszy osobnik potomny otrzyma gen od pierwszego z osobników rodzicielskich, a gdy we wzorcu wystąpi wartość 1, wówczas pierwszy osobnik potomny otrzyma gen od drugiego z osobników rodzicielskich. W przypadku drugiego osobnika potomnego sytuacja będzie dokładnie odwrotna, tj. wartość 0 we wzorcu – geny od Rodzica 2, wartość 1 we wzorcu – geny od Rodzica 1.

Krzyżowanie równomierne podobne jest do krzyżowania wielopunktowego. Zasadniczą jednak różnicą jest to, w krzyżowaniu wielopunktowym miejsce krzyżowania wyznacza algorytm. Z kolei w krzyżowaniu równomiernym krzyżowanie odbywa się tylko na podstawie wylosowanego wzorca (maski krzyżowania). Wylosowany wzorzec musi być tej samej długości co osobniki. Na wymianę bitów między osobnikami można wpływać poprzez ustawienie pożądanego prawdopodobieństwa zmiany wartości bitu podczas tworzenia wzorca wymiany.



Rysunek 4.6 Przykład krzyżowania równomiernego.

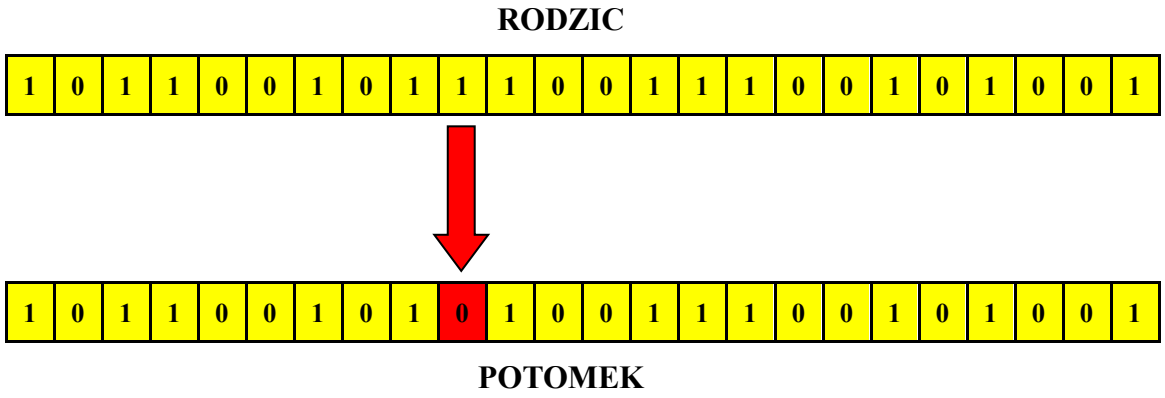
Źródło: opracowanie własne.

Poza czterema opisanymi operatorami krzyżowania, do popularnych metod krzyżowania można zaliczyć między innymi: krzyżowanie tasujące (ang. *Shuffle Crossover*) [475,482], krzyżowanie zastępujące (ang. *Reduced Surrogate Crossover*) [475], krzyżowanie niszczące (ang. *Heuristic Uniform Crossover/Highly Disruptive Crossover*) [475,479-4881], krzyżowanie uśredniające (ang. *Average Crossover*) [475-476,483], krzyżowanie ziarniste (ang. *Discrete Crossover*) [475,484-485], krzyżowanie płaskie (ang. *Flat Crossover*) [475,486], krzyżowanie heurystyczne-1 (ang. *Heuristic Crossover/Intermediate Crossover*) [475,484-486], krzyżowanie mieszające (ang. *Blend Crossover*) [475,486-487], krzyżowanie powielające podobieństwa (ang. *Random Respectful Crossover*) [475-476,488-490], krzyżowanie oparte na dominacji (ang. Masked Crossover) [475,491-494], krzyżowanie wyboru operatora (ang. *1 bit Adaptation Crossover*) [475,495-497], krzyżowanie wielowymiarowe (ang. *Multivariate Crossover*) [475,498-500], krzyżowanie homologiczne (ang. *Homologous Crossover*) [475,501-502], krzyżowanie zliczające-1 (ang. *Court-preserving Crossover-1*) [475,503-504] itd.

Działanie operatora mutacji polega na zmianie w genotypie osobnika wartości pojedynczego genu poprzez przypisanie do niego dowolnej innej wartości z dozwolonego zestawu alleli. W przypadku binarnej reprezentacji osobników mutacja polega na zamianie wartości genu z 0 na 1 lub odwrotnie [81,434,436]. Proces mutacji polega na

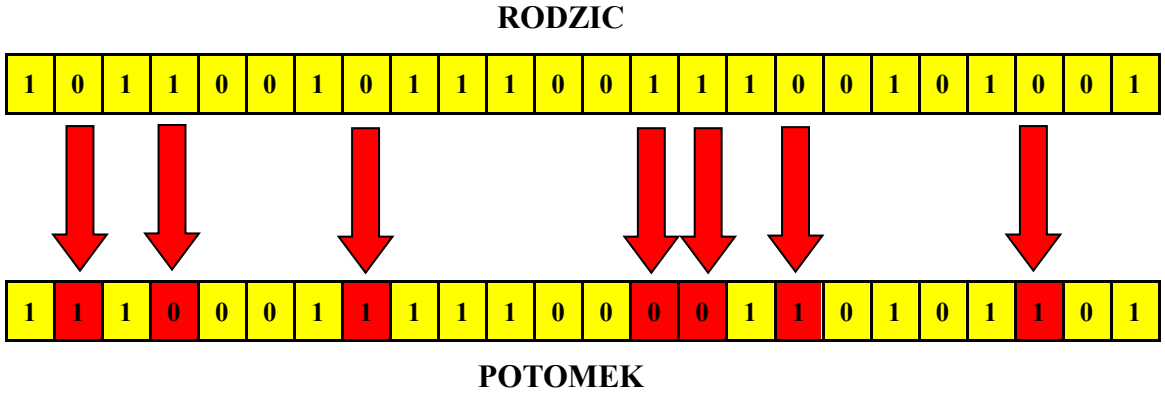
niezależnym losowaniu dla każdego genu prawdopodobieństwa mutacji. Współczynniki, które określają prawdopodobieństwo wystąpienia mutacji, powinny być dobrane w taki sposób, aby mutacja występowała przynajmniej raz na kilka nowych genotypów [477]. Najczęściej stosowanymi rodzajami mutacji jest mutacja jedno i wielopunktowa.

Mutacja jednopunktowa [505] (ang. *1-Point Mutation*, *One Point Mutation*) polega na utworzeniu nowego osobnika poprzez wprowadzenie losowej zmiany jednego genu w osobniku rodzicielskim (Rysunek 4.7).



Rysunek 4.7. Przykład mutacji jednopunktowej.
Źródło: Opracowanie własne.

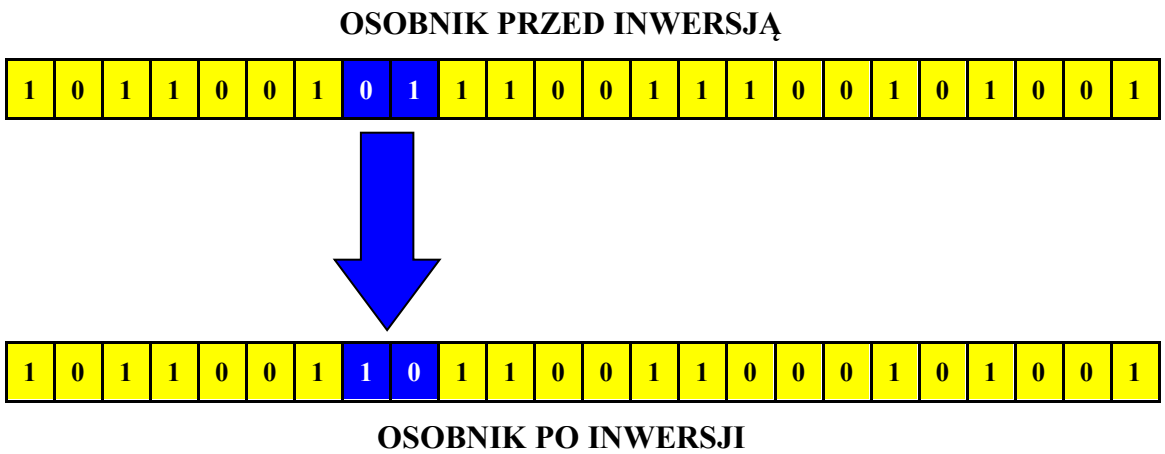
Mutacja wielopunktowa [505] (ang. *Multi Point Mutation*) polega na utworzeniu nowego osobnika poprzez wprowadzenie losowej zmiany w kilku genach osobnika rodzicielskiego (Rysunek 4.8).



Rysunek 4.8. Przykład mutacji wielopunktowej.
Źródło: Opracowanie własne.

Wśród innych rodzajów mutacji wyróżnić można m.in.: mutacja dynamiczna (ang. *Dynamic Mutation*) [505-511], mutacja najlepszym schematem (ang. *Best Schema Mutation*) [505,512-516], adaptacyjna mutacja asymetryczna (ang. *Adaptive Asymmetric Mutation*) [505,517-519], adaptacyjna mutacja ukierunkowana na geny (ang. *Gene-based Adaptive Mutation* [505,520-524], mutacja kontrolowana logiką rozmytą (ang. *Fuzzy Logic Controlled Mutation*) [505,525-534], mutacja bazująca na statystykach pozycji (ang. *Locus Statistics-based Mutation*) [505,535-537], adaptacyjna mutacja metodą stałego przyrostu i redukcji (ang. *Constant Gain & Declining Adaptive Mutation*) [505,538], mutacja zliczająca (ang. *Count-pressing Mutation-1*) [505,539-542], mutacja infekcją wirusową (ang. *Virus Infection Operators*) [505,543-546], mutacja lokalnie zachłanna (ang. *Local GreedyMutation*) [505,547-550], mutacja z decydentem (ang. *Half Sibling and Clone*) [505,551-552], samoadaptacyjna mutacja Gaussa (ang. *Self-adaptive Gaussian Mutation*) [505,553-557].

Ostatni z operatorów genetycznych, stosowany czasami w klasycznym algorytmie genetycznym, to operator inwersji. Działanie operatora inwersji polega na zamianie kolejności genów (inwersji) w losowo wybranym podciągu genów osobnika (Rysunek 4.9) [81]. Wprowadzenie operatora inwersji ma za zadanie wprowadzenie znacznych różnic pomiędzy populacją rodzicielską a kolejnym pokoleniem, a co za tym idzie, działa korzystnie na przeszukiwanie przestrzeni rozważań, wyprowadzając algorytm z obszarów oddziaływania lokalnych ekstremów funkcji celu. Więcej informacji na temat operatora inwersji znajduje się w [558].



Rysunek 4.9. Przykład inwersji.

Źródło: Opracowanie własne.

4.6. Selekcja osobników

Każdy organizm biologiczny przebywa, rozwija i rozmnaża się w określonym środowisku, w obrębie populacji osobników tego samego gatunku. Osobniki te różnią się od siebie. Jedne osobniki lepiej radzą sobie w danym środowisku (np. są szybsze, zwinniejsze, mądrzejsze), a inne są gorzej przystosowane do tego środowiska. Osobniki należące do pierwszej grupy mają większe szanse na przeżycie i wydanie potomstwa niż osobniki należące do drugiej grupy. W konsekwencji wraz z upływem kolejnych pokoleń cała populacja osobników jest coraz lepiej przystosowana do danego środowiska. Środowisko dokonuje oceny poszczególnych osobników danego gatunku na podstawie zewnętrznych cech osobników, czyli ich fenotypów [455-456]. Podobnie sytuacja wygląda w algorytmie genetycznym. Tutaj również osobniki są oceniane na podstawie informacji zawartych w ich fenotypach. Osobniki o fenotypach najsilniejszych, z punktu widzenia przyjętej funkcji celu, mają największe szanse na przetrwanie, czyli wybranie do kolejnego pokolenia, natomiast osobniki o fenotypach najsłabszych najczęściej zostają wyeliminowane z dalszych etapów przetwarzania.

W procesie selekcji osobników wykorzystywanych jest wiele metod. Wśród najczęściej stosowanych należy wymienić metodę próbkowania deterministycznego [27], metodę deterministyczną, metodę stochastyczną (losową) według reszt z powtórzeniami [27], metodę stochastyczną (losową) według reszt bez powtórzeń [27], metodę selekcji progowej, metodę rangową (rankingową) [27,205], metodę stochastyczną równomierną, metodę koła ruletki (metodę proporcjonalną) [27,81,205,436], metodę pojedynku oraz metodę turniejową [27,205,460].

Metoda koła ruletki [27,81,436] była bardzo często wykorzystywana w pierwszych algorytmach genetycznych [238]. W metodzie budowane jest wirtualne koło ruletki. Koło jest dzielone na szereg części odpowiadających poszczególnym osobnikom. Wielkość wycinków koła przypisanych do osobników jest proporcjonalna do ich oceny wyznaczonej na podstawie funkcji przystosowania. Tak więc osobniki lepiej dopasowane mają przypisany szerszy wycinek koła niż osobniki o niższych wartościach funkcji przystosowania. W bardziej matematycznym ujęciu procedury, wycinki koła, odpowiadają prawdopodobieństwu wylosowania osobnika do kolejnej populacji, zdefiniowanemu jako (wzór 4.8) [559]:

$$P_w = \frac{WP_O}{\sum WP_{WO}} \tag{4.8}$$

gdzie:

P_W – prawdopodobieństwo wylosowania danego osobnika;

WP_O – wartość przystosowania danego osobnika;

$\sum WP_{WO}$ – suma wartości przystosowania wszystkich osobników w populacji.

Po zdefiniowaniu prawdopodobieństw dla poszczególnych osobników wykonywane jest *n*-krotne losowanie osobników z aktualnej populacji (gdzie *n* to liczba osobników w populacji). Wylosowane osobniki tworzą populację rodzicielską, czyli populację, która zostanie poddana operacjom genetycznym. W metodzie koła ruletki istnieje wysokie prawdopodobieństwo, że lepsze osobniki zostaną wybrane wielokrotnie, gorsze natomiast zostaną wyeliminowane. Podstawowymi wadami tej metody jest to, że algorytm bardzo słabo odróżnia od siebie osobniki podobne, ponieważ każdemu z nich jest przypisywane takie same prawdopodobieństwo wylosowania oraz to, że *„przypadek odgrywa zbyt wielką rolę w tej metodzie [ponieważ], jedno pechowe pokolenie, dla którego ważny, dobrze przystosowany osobnik nie został wybrany, mogłoby zahamować postęp w następnych pokoleniach."* [238].

W **metodzie rankingowej** [27,205] dla każdego osobnika obliczana jest funkcja oceny, a następnie osobniki są sortowane i ustawiane w szeregu według malejących wartości funkcji przystosowania. Osobniki najlepsze znajdują się na początku listy i właśnie one są wybierane do populacji rodzicielskiej, osobniki najgorsze są natomiast usuwane z populacji. Wadą metody rankingowej jest jej niewrażliwość na różnice pomiędzy kolejnymi osobnikami w rankingu. Może się okazać, że rozwiązania sąsiadujące ze sobą w rankingu mają znacząco różne wartości funkcji przystosowania, ale generują prawie taką samą ilość potomstwa.

W **metodzie turniejowej** [27,205,460] losowana jest grupa osobników (tzw. grupa turniejowa), z której następnie wybierany jest jeden lub kilka osobników o największych wartościach funkcji przystosowania. Wybrane osobniki są przekazywane do populacji rodzicielskiej. Losowanie ma charakter rozkładu równomiernego. Losowanie grup turniejowych oraz wybór osobników najlepszych powtarza się wielokrotnie,

aż do osiągnięcia populacji o zadanej liczebności. Liczba losowanych osobników oraz liczba osobników wybieranych jest ustalana jako parametr turnieju. Metoda turniejowa jest uznawana za jedną z lepszych metod selekcji z tego powodu, że dokonuje wyboru korzystając jedynie z informacji o przewadze jednego rozwiązania nad innym. Dzięki temu, że nie jest w jej przypadku wymagana maksymalizacja funkcji oceny, zmniejsza ona radykalnie prawdopodobieństwo utknięcia algorytmu w optimum lokalnym funkcji celu [462,559]. Jak pisze Cierniak *„Doświadczenia wykazują znacznie lepsze efekty działania algorytmu genetycznego, w którym zamiast metody ruletki dokonuje się selekcji turniejowej. Warto podkreślić, że selekcję turniejową można także wykorzystać w zadaniach optymalizacji wielokryterialnej”* [436].

Rozdział V
Algorytmy genetyczne w procesie selekcji cech

Jednym z popularnych obszarów zastosowań algorytmów genetycznych jest selekcja cech. Podstawową zaletą algorytmów genetycznych, sprawiającą, że są one chętnie stosowane w procesie selekcji cech jest to, że oceniają one równolegle wiele zestawów cech, zamiast oceniać każdy z nich pojedynczo. Ponadto, nie mają one skłonności do utykania w lokalnych minimach i nie muszą zakładać interakcji między cechami. Z tych powodów algorytmy genetyczne są często wykorzystywane do znajdowania optymalnych zestawów cech w problemach wielowymiarowych i nieliniowych.

W pierwszych pięciu punktach niniejszego rozdziału opisano algorytmy genetyczne stosowane obecnie najczęściej w procesie selekcji cech: klasyczny algorytm genetyczny Hollanda [4,22,40,69,83,90], wariacje klasycznego algorytmu genetycznego (odmienne sposoby kodowania i reprodukcji, odmienna funkcja przystosowania), wielokryterialny algorytm genetyczny NSGA II [4,22,40,69,83,90,94] oraz algorytm genetyczny z ustaloną liczbą cech (algorytm Cullinga) [94,95]. Dwa kolejne algorytmy, algorytm genetyczny z agresywną mutacją [94,95] oraz algorytm genetyczny z agresywną mutacją i zmniejszającą się liczbą genów, to algorytmy [94,95] , które zostały w ostatnich latach opracowane w zespole badawczym, w skład którego wchodzi autor niniejszej rozprawy. Wreszcie algorytm opisany w punkcie wieńczącym niniejszy rozdział, algorytm genetyczny z agresywną mutacją i malejącą liczbą cech, to autorska propozycja połączenia klasycznego algorytmu Hollanda z funkcją kary oraz wielokryterialnego algorytmu genetycznego NSGA II z algorytmem genetycznym z agresywną mutacją.

5.1. Klasyczny algorytm genetyczny

W klasycznym algorytmie genetycznym każdy osobnik jest złożony z genów, które mogą przyjmować dwie wartości – 1 lub 0. Najbardziej oczywistym podejściem w przypadku wykorzystania tego rodzaju algorytmu w procesie selekcji cech jest więc podejście, w którym kolejne geny osobnika odpowiadają kolejnym cechom z wektora cech, a cały osobnik odpowiada wybranemu podzbiorowi cech. Podstawowa reguła kodowania jest następująca: jeżeli cecha *i* występuje w osobniku, wówczas gen *i* przyjmuje wartość „1", natomiast jeżeli osobnik nie posiada cechy *i*, wówczas gen *i* przyjmuje wartość „0". Liczba genów w osobniku jest więc równa liczbie wszystkich cech, przy czym niektóre z genów przyjmują wartość 1 (oznaczającą wystąpienie cechy w podzbiorze cech zakodowanym w danym osobniku), pozostałe – wartość 0 – oznaczającą brak danej cechy (Rysunek 5.1).

Rysunek 5.1. Podstawowa reguła kodowania cech w klasycznym algorytmie genetycznym: 0 – cecha nie występuje w osobniku, 1 – cecha występuje w osobniku.
Źródło: Opracowanie własne.

Podzbiór cech zakodowany w osobniku jest oceniany zgodnie z kryterium dokładności klasyfikacji uzyskanej po wprowadzeniu tego podzbioru cech do klasyfikatora. Populacja osobników (czyli zestaw podzbiorów cech) jest następnie poddawana klasycznym operacjom krzyżowania i mutacji, w trakcie których następuje wymiana cech między osobnikami (krzyżowanie) oraz usuwanie/dodawanie cech w obrębie danego osobnika (mutacja). Po tych etapach poszczególne osobniki nadal mają wyjściową liczbę genów, jednakże liczby cech mogą być zupełnie inne niż na początku.

Podejście to jest bardzo proste w implementacji, ponieważ liczba genów osobnika jest równa liczbie wszystkich cech występujących w wektorze cech. Metoda ta posiada jednak jedną bardzo niekorzystną cechę, tzn. przy zastosowaniu klasycznej postaci funkcji przystosowania danej dokładnością klasyfikacji, proces optymalizacji jest prowadzony w kierunku uzyskania osobników o 100% dokładności klasyfikacji, bez względu na liczbę cech występujących w rozwiązaniu.

5.2. Klasyczny algorytm genetyczny z członem kary

Najczęściej stosowanym podejściem pozwalającym na złagodzenie wskazanego powyżej problemu jest dołączenie do funkcji przystosowania oceniającej poszczególne osobniki tzw. członu kary. Zadaniem tego dodatkowego członu jest karanie osobników o dużej liczbie cech. Cel ten zostaje osiągnięty poprzez zmniejszanie wartości przystosowania tychże osobników (wzór 5.1) [90].

$$A = 0.5\, D_K + 0.5\, \frac{(F_{All} - F_O)}{F_{All}} \tag{5.1}$$

gdzie:

A – funkcja przystosowania;

D_K – dokładność klasyfikacji;

F_{All} – całkowita liczba cech;

F_O – liczba cech w osobniku.

Liczba cech, które są zawarte w osobniku, wpływa na wartość członu kary funkcji przystosowania, tzn. im większa liczba cech, tym większa jest wartość kary nakładanej na osobnika. W najprostszym przypadku zarówno człon oparty na dokładności klasyfikatora, jak i człon kary uwzględniane są z taką samą wagą [90], ale możliwe jest również przypisanie obu członom różnych wartości współczynnika wagowego.

5.3. Klasyczny algorytm genetyczny – odmienne sposoby kodowania oraz operatory genetyczne

Innym sposobem ograniczenia liczby cech w rozwiązaniach zwracanych przez klasyczny algorytm genetyczny jest zmiana sposobu kodowania osobników. W [560] zostało zaproponowane podejście CBRG (ang. *Case-Based Repair Generation*), w którym pojedynczy osobnik koduje jednocześnie trzy różne podzbiory cech. Każdy osobnik składa się tu z czterech chromosomów, każdy o długości równej liczbie wszystkich cech. Chromosom pierwszy, przyjmujący wartości rzeczywiste, zawiera wagi przypisane do każdej z cech. Trzy kolejne chromosomy (ściśle odpowiadające kodowanym podzbiorom cech) zawierają natomiast wartości binarne informujące o tym,

czy cecha f_i wchodzi w skład danego podzbioru (gen = 1), czy nie (gen = 0). Osobnik w populacji jest więc reprezentowany jak na Rysunku 5.2. Wykorzystanie trzech zbiorów selekcjonowanych zmiennych umożliwia sprawne wyrównanie przestrzeni funkcji przystosowania poprzez redukcję wpływu zmian w statusie selekcji cech na funkcję przystosowania chromosomów [561].

$$m = \{w_0, \ldots, w_{F-1} | f_{0,0}, \ldots, f_{0,F-1}, f_{1,0}, \ldots, f_{1,F-1}, f_{2,0}, \ldots, f_{2,F-1}\}$$

Rysunek 5.2. Przykład odmiennego sposobu kodowania cech w osobniku. Geny $f_{0,i}$, $f_{1,i}$,$f_{2,i}$ zawierają informację, czy cech i wchodzi w skład podzbioru cech zakodowanego w osobniku, natomiast gen w_i informuje o stopniu ważności tej cechy.
Na rysunku $w_0, \ldots, w_{F-1}$ oznaczają wagę cech; $f_{0,0}, \ldots, f_{0,F-1}, f_{1,0}, \ldots, f_{1,F-1}, f_{2,0}, \ldots, f_{2,F-1}$ oznaczają zbiór wyselekcjonowanych cech.
Źródło: Opracowanie własne na podstawie [560].

Algorytm rozpoczyna się uszeregowania cech pod względem ich istotności dla rozwiązywanego problemu i przypisaniu każdej cesze indywidualnej wagi. Im cecha jest ważniejsza dla rozwiązywanego zadania, tym ma wyższą wagę. Następnie generowana jest startowa populacja osobników, przy czym dla każdego osobnika losowo ustalane są wartości trzech ostatnich chromosomów (kodujących podzbiory cech), natomiast wartości wag przypisanych do kolejnych cech są wyznaczane zgodnie z dwoma zasadami. Pierwsza z nich mówi o tym, że cecha skojarzona z wyznaczaną wagą musi być obecna przynajmniej w dwóch z trzech kodowanych podzbiorów, czyli przynajmniej dwa binarne geny odpowiadające tej cesze muszą mieć wartość 1, aby została uznana za występującą w osobniku. Jeżeli warunek ten nie jest spełniony, to waga tej cechy jest ustalana na 0 i tym samym cecha ta jest wykluczana z osobnika [560]. Jeżeli natomiast cecha występuje w przynajmniej dwóch z trzech kodowanych podzbiorów, to jej wagą jest indywidualna waga ustalona na początku algorytmu.

Wyznaczenie wartości funkcji przystosowania pojedynczego osobnika przebiega zgodnie z następującym schematem. Najpierw dokonywana jest normalizacja genów zawierających wagi poszczególnych cech w taki sposób, aby całkowita wartość sumy wag osobnika wynosiła jeden. Następnie osobnik jest konwertowany na wektor ciężkości (z pominięciem tych cech, których wartość wagi jest ustawiona na zero), który

wykorzystywany jest do obliczania dokładności klasyfikacji [560]. Etap selekcji osobników oraz operacji genetycznych są wykonywane w algorytmie CBRG za pomocą metod klasycznych.

Inny sposób kodowania osobników dostosowany do potrzeb selekcji cech przedstawiono w artykule [562]. W zaproponowanym podejściu do reprezentowania chromosomu wybrano system kodowania binarnego, przy czym samo kodowanie chromosomu polega tu na jego podziale na 3 części kodujące kolejno: wartość kary C, parametry jądra funkcji radialnej γ oraz maskę cech *f* [562] (Rysunek 5.3).

$$g_C^1, \ldots g_C^i \ldots, g_C^{n_C} \Big| g_\gamma^1, \ldots g_\gamma^j \ldots, g_\gamma^{n_\gamma} \Big| g_f^1, \ldots g_f^k \ldots, g_f^{n_f}$$

Rysunek 5.3. Wygląd chromosomu składającego się z trzech części: *C*, γ i maski funkcji. Źródło: Opracowanie własne na podstawie [562]

Poszczególne symbole z Rysunku 5.3 oznaczają:

C – parametr kary (wartości od g_C^1 do $g_C^{n_C}$);

n_C – liczba bitów reprezentująca parametr C;

γ – parametry jądra funkcji radialnej (ang. *Radial Basis Function, RBF*) w klasyfikatorze SVM (wartości od g_γ^1 do $g_\gamma^{n_\gamma}$);

n_γ – liczba bitów reprezentująca parametr γ;

f – maska cech (wartości od g_f^1 do $g_f^{n_f}$);

n_f – liczba bitów reprezentująca parametr *f*.

Przedstawiony sposób kodowania jest podobny do sposobu wykorzystywanego w algorytmie klasycznym w tym, że stosowana jest ta sama maska cech, to jest gen o wartości „1" reprezentuje cechę obecną w osobniku, a gen o wartości „0" cechę pominiętą. Poza maską cech do osobnika włączone są jednak również inne aspekty problemu, nie występujące w kodowaniu klasycznym, czyli człon kary oraz parametry funkcji radialnej stosowanej w klasyfikatorze.

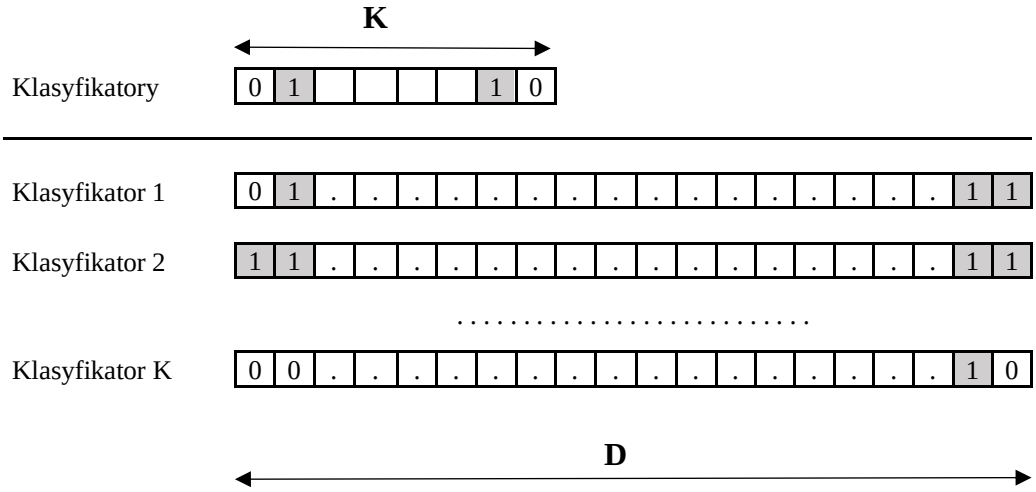
Inny sposób kodowania osobników, łączący selekcję cech z doborem klasyfikatora, został zaprezentowany w artykule [563]. Każdy osobnik składa się tutaj z dwóch części

(Rysunek 5.4). Pierwsza część zawiera K-genów (K odpowiada liczbie klasyfikatorów wykorzystanych w procesie klasyfikacji) zakodowanych w systemie binarnym, zgodnie z poniższą regułą (wzór 5.2) [563]:

$$Gen(k) = \begin{cases} 1, \ jezeli\ został\ wybrany\ k-ty\ klasyfikator \\ 0\ w\ pozostałych\ przypadkach \end{cases} \tag{5.2}$$

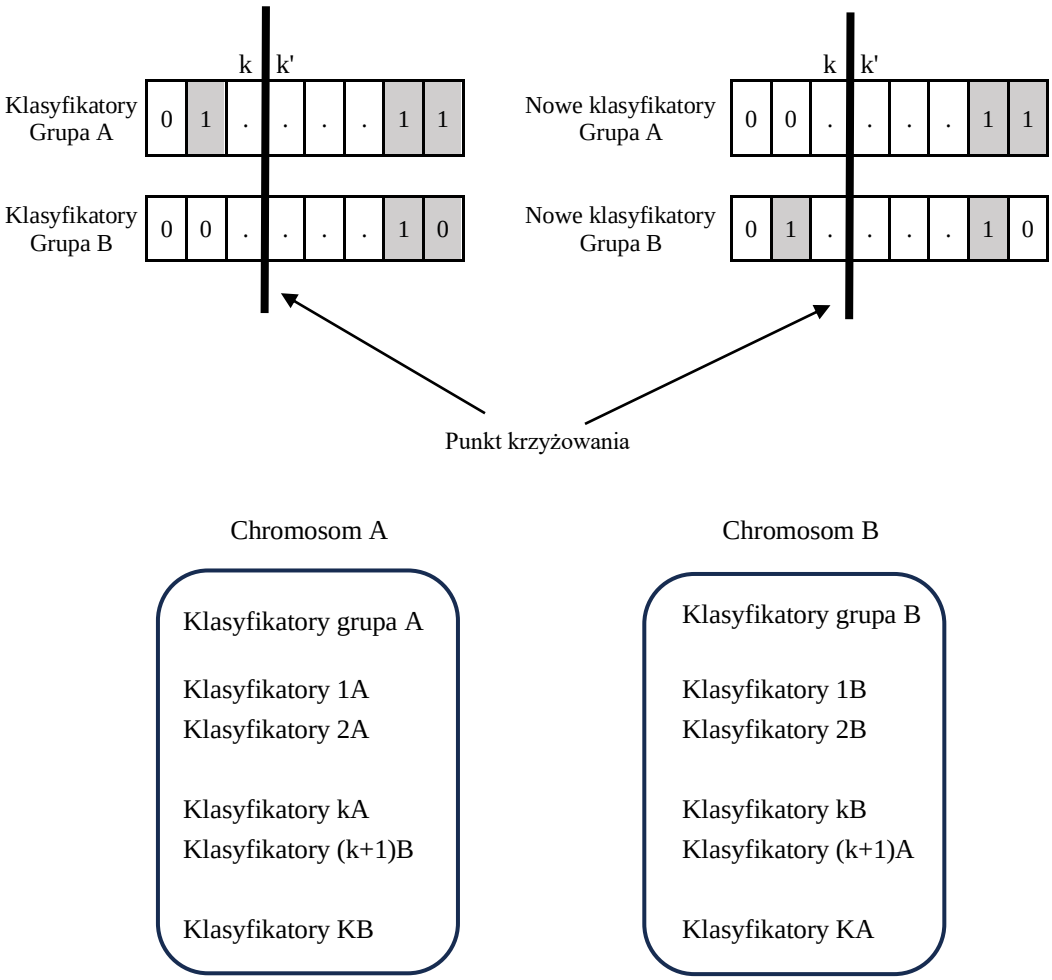
Druga część osobnika zawiera informację o D-cechach wykorzystanych w K-tym klasyfikatorze. Również ta część kodowana jest w sposób binarny zgodnie z analogiczną regułą. Dla *k=1....K* i *d=1...D* otrzymuje się (wzór 5.3) [563]:

$$Gen_k(d) = \begin{cases} 1, \ jezeli\ d-cecha\ została\ wybrana\ przez\ k-ty\ klasyfikator \\ 0\ w\ pozostałych\ przypadkach \end{cases} \tag{5.3}$$



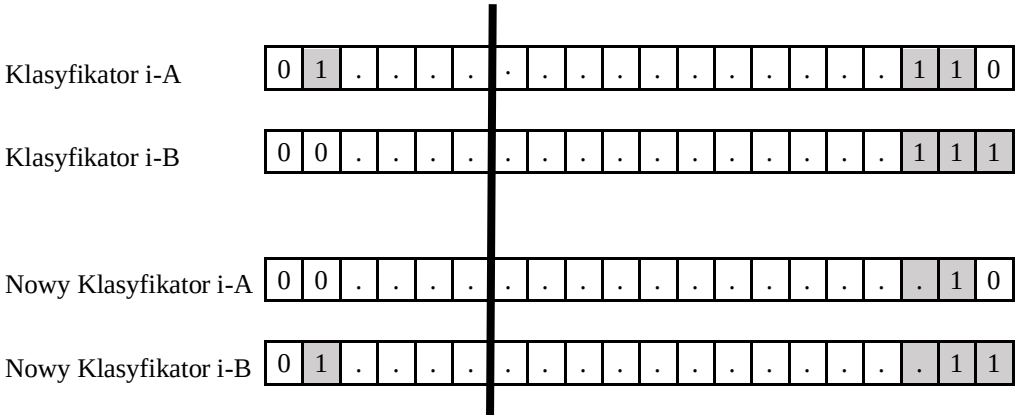
Rysunek 5.4. Struktura chromosomu.
Źródło: Opracowanie własne na podstawie [563].

Taki dwu-chromosomowy układ osobnika wymaga wprowadzenia zmian również w procesie krzyżowania, który musi odbywać się dwustopniowo. W pierwszym etapie krzyżowanie jest wykonywane na części osobnika kodującej klasyfikatory (Rysunek 5.5), w drugiej – na części osobnika kodującego cechy każdego z klasyfikatorów (Rysunek 5.6). Oba poziomy krzyżowania korzystają z zasady krzyżowania jednopunktowego.



Rysunek 5.5. Krzyżowanie jednopunktowe pary osobników A i B w części kodującej klasyfikatory, gdzie *k'=k+1*

Źródło: Opracowanie własne na podstawie [563].



Rysunek 5.6. Krzyżowanie jednopunktowe pary osobników A i B w części kodującej cechy.

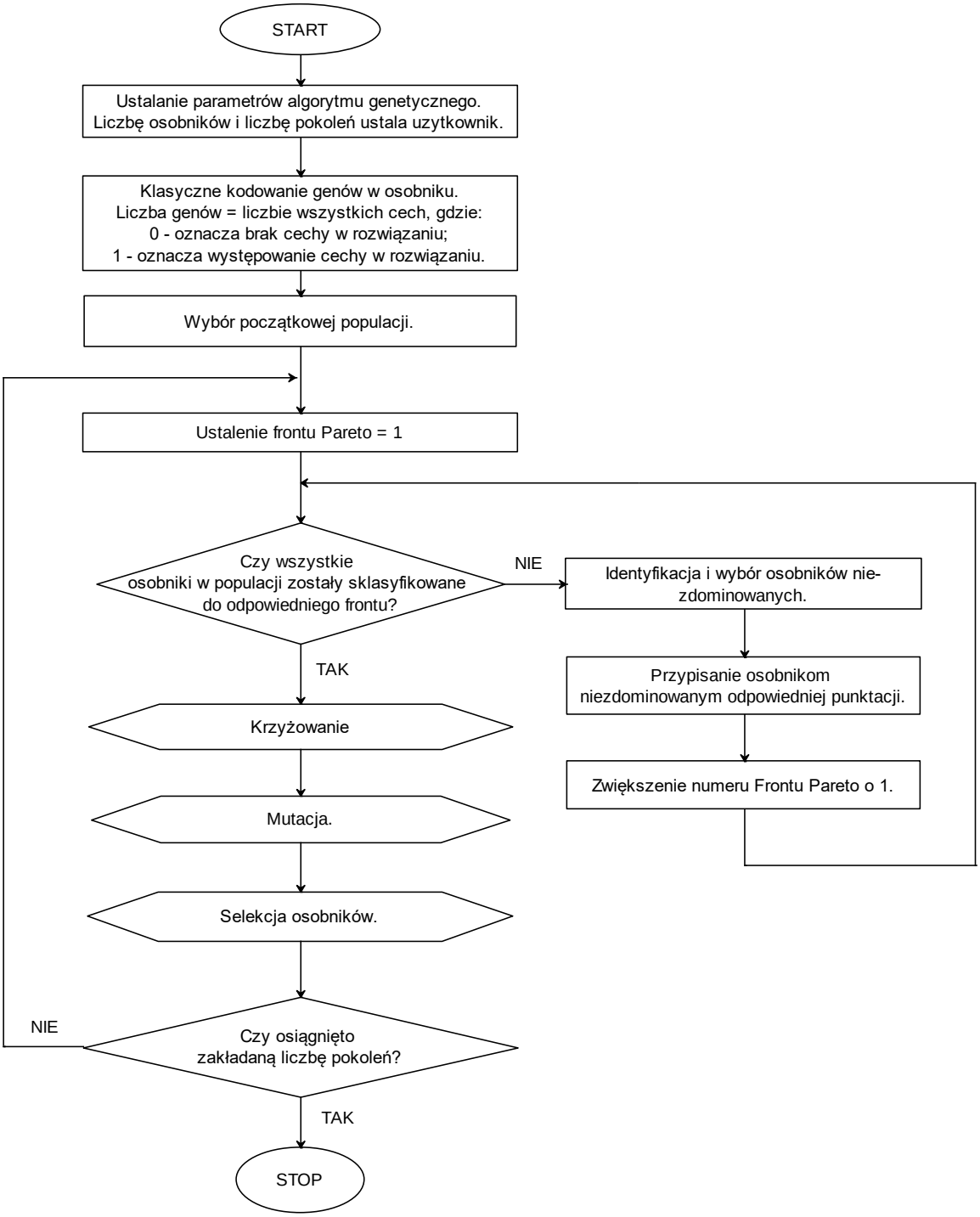
Źródło: Opracowanie własne na podstawie [563]

5.4. Wielokryterialny Algorytm Genetyczny – metoda NSGA II

Wybór ostatecznego rozwiązania następuje bardzo często w oparciu o kilka kryteriów. Na przykład, na ostateczną decyzję konsumenta poszukującego nowego telefonu komórkowego, będzie miała wpływ nie tylko cena urządzenia, ale także inne czynniki, takie jak: producent, wielkość matrycy, kolor, system operacyjny, prędkość działania i dostępu do Internetu itd. Jedną z metod pozwalających na uwzględnienie wielu kryteriów w procesie optymalizacji jest optymalizacja wielokryterialna z wykorzystaniem frontu Pareto. Podejście to realizowane jest w dziedzinie algorytmów genetycznych najczęściej w postaci algorytmu NSGA II (ang. *Nondominated Sorting Genetic Algorithm II*) lub jego wcześniejszej wersji NSGA.

Poza tymi dwoma algorytmami optymalizacja wielokryterialna z użyciem frontu Pareto może być realizowana również za pomocą algorytmu VEGA (ang. *Vector Evaluated Genetic Algorithm*), stworzonego przez D. Shaffera w 1984 r. [564-565], algorytmu HLGA (ang. *Hajela's and Lin's Weighting-based Genetic Algorithm*) [566], algorytmu FFGA (ang. *Fonseca's and Fleming's Multiobjective Genetic Algorithm*) [567], algorytmu NPGA (ang. *The Niched Pareto Genetic Algorithm*) [568], algorytmu SPEA (ang. *The Strength Pareto Evolutionary Algorithm*) [569-571], algorytmu SPEA 2 (ang. *The Strength Pareto Evolutionary Algorithm 2*) [572-574] i innych.

Algorytm NSGA po raz pierwszy został zaprezentowany przez Deb i Srinivas w 1994 r. [91], natomiast algorytm NSGA II osiem lat później w roku 2002 [92]. Algorytm NSGA II jest implementowany jako klasyczny algorytm Hollanda ze specyficzną funkcją przystosowania [92]. Funkcja ta wykorzystuje zasadę dominacji jednych osobników nad pozostałymi. Osobniki niezdominowane (paretooptymalne) to osobniki, dla których w danej populacji nie można znaleźć osobników lepszych z uwagi na wartość co najmniej jednej funkcji kryterialnej bez pogorszenia z uwagi na funkcje pozostałe. Schemat działania algorytmu NSGA II przedstawiono na Rysunku 5.7.



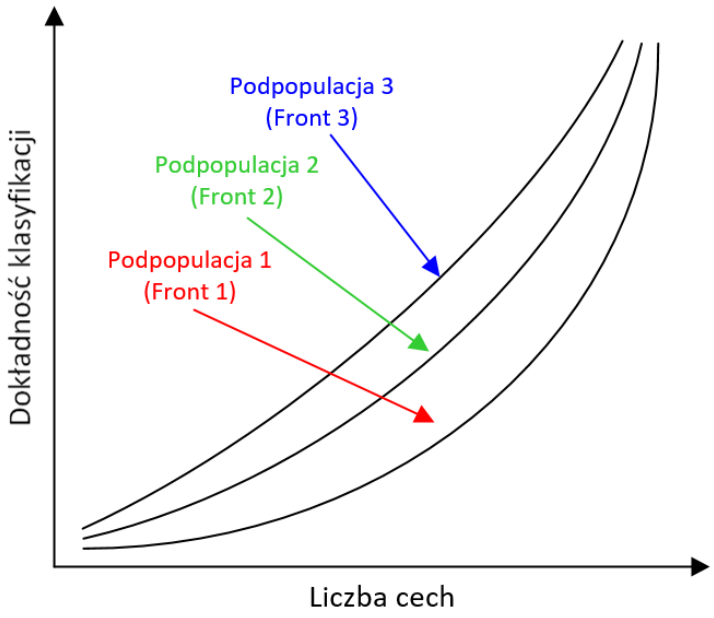
Rysunek 5.7. Schemat działania algorytmu NSGA II.

Źródło: Opracowanie własne na podstawie [26].

Proces oceny osobników w metodzie NSGA II przebiega w dwóch etapach. W etapie pierwszym dla każdego z osobników występujących w aktualnie ocenianej populacji wyznaczane są wartości wszystkich funkcji kryterialnych. W przypadku, kiedy algorytm jest wykorzystywany do selekcji cech, dominację jednych osobników

nad innymi ustala się najczęściej w oparciu o dwie funkcje kryterialne: dokładność klasyfikacji i liczbę cech zakodowanych w osobniku. Po wyznaczeniu wartości wszystkich funkcji kryterialnych osobniki są sortowane (z osobna dla danego kryterium) i nadawane im są rangi (liczba rang osobnika jest równa liczbie funkcji kryterialnych). Porządek sortowania wynika z kierunku optymalizacji, czyli dla minimalizowanego kryterium liczby cech jest rosnący, a dla maksymalizowanego kryterium dokładności klasyfikacji jest malejący.

Drugi etap oceny polega na stopniowym wydzielaniu z populacji głównej kolejnych podpopulacji, zgodnie z zasadą dominacji. Każda kolejna podpopulacja odpowiada kolejnemu frontowi Pareto (Rysunek 5.8) i zawiera osobniki niezdominowane przez osobniki nie zaklasyfikowane do wcześniej wydzielonych podpopulacji. Jako że kolejne fronty Pareto ustalane są zgodnie z zasadą dominacji, więc pierwszy front jest całkowicie niezdominowanym zbiorem osobników w populacji. Drugi front jest zdominowany tylko przez osobniki z frontu pierwszego, trzeci front jest zdominowany tylko przez osobniki z frontu drugiego itd. Przy tak zdefiniowanym procesie oceny, ocenę osobnika stanowi numer podpopulacji (frontu Pareto), do której został on zakwalifikowany. Osobnikom z pierwszego frontu nadawana jest wartość funkcji przystosowania równa 1, osobniki z drugiego frontu otrzymują wartość przystosowania równą 2, osobniki z frontu trzeciego otrzymują wartość przystosowania równą 3 itd.



Rysunek 5.8. Fronty Pareto

Źródło: Opracowanie własne.

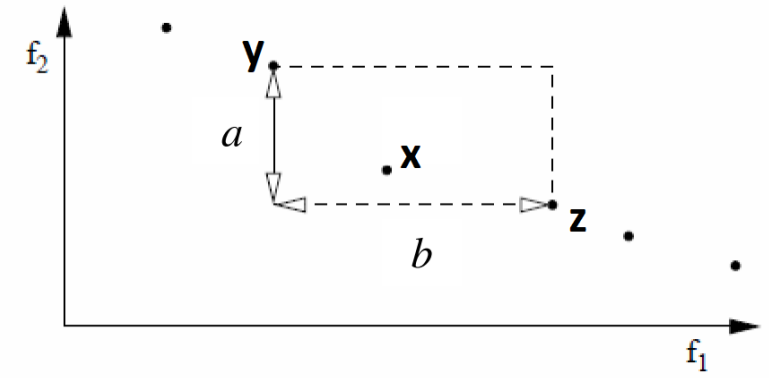
W algorytmie NSGA II [94] do wartości funkcji przystosowania każdego osobnika dodawany jest nowy parametr, nazywany „*crowding distance*”. Wartość tego parametru zależy od tego, jak blisko oceniany osobnik znajduje się swoich najbliższych sąsiadów. Duża średnia wartość tego parametru wyznaczona dla całej populacji oznacza wyższą różnorodność osobników w populacji.

Wartość parametru *Crowding Distance* (dla uproszczenia w problemie optymalizacji dwukryterialnej) jest wyznaczana następująco [575]:

1. Dla osobnika x w przestrzeni kryteriów jest znajdowanych jego „dwóch najbliższych sąsiadów” y i z (Rysunek 5.9), gdzie pierwszy sąsiad y ma być lepszy od osobnika x względem pierwszego kryterium, natomiast drugi sąsiad z ma być lepszy od osobnika x względem drugiego kryterium.
2. Osobniki y i z znajdują się na przeciwległych wierzchołkach prostokąta o długości boków prostokąta równych a i b.
3. Wartość parametru *Crowding Distance* (CR) dla osobnika x, wynosi (wzór 5.4):

$$CR(x) = a + b \tag{5.4}$$

Jeżeli jeden z sąsiadów osobnika nie istnieje, to przyjmuje się, że odpowiedni bok prostokąta ma długość nieskończoną, czyli $CR(x) = \infty$.



Rysunek 5.9. Prostokąt rozpięty na dwóch najbliższych sąsiadach osobnika w danym froncie Pareto.

Źródło: Opracowanie własne na podstawie [575].

O wyborze osobników do kolejnych populacji decyduje zarówno numer frontu Pareto osobnika i jak wyznaczona dla niego wartość parametru *crowding distance*. Osobnik jest rzadziej wybierany, jeżeli znajduje się w wyższym froncie Pareto lub jeżeli

wartość jego parametru *crowding distance* jest mniejsza niż u innych osobników. Wybrana populacja osobników generuje potomstwo najczęściej za pomocą klasycznych operatorów krzyżowania i mutacji. Po rozmnożeniu populacji, populacja rodzicielska jest łączona z populacją potomną, po czym wszystkie osobniki z obu populacji są przypisywane do kolejnych frontów Pareto zgodnie z kryterium dominacji. Operacja selekcji przeprowadzana jest zgodnie z metodą rangową, przy czym ranking osobników tworzony jest według dwóch kryteriów:

- Rangi osobnika – czyli numeru frontu Pareto do którego należy (im niższy numer frontu Pareto, tym lepiej).
- Wartości parametru *Crowding Distance* (*CR*) osobnika (im większa odległość, tym lepiej).

W celu utworzenia rankingu definiowana jest relacja na zbiorze osobników. Przykładowo dla dwóch osobników x i y będzie ona zdefiniowana następująco:

- Jeżeli $rang(x) < rang(y)$ to $x < y$
- Jeżeli $rang(x) > rang(y)$ to $y < x$
- Jeżeli $rang(x) = rang(y)$ to
  - Jeżeli $CR(x) > CR(y)$ to $x < y$
  - Jeżeli $CR(x) < CR(y)$ to $y < x$
  - Jeżeli $CR(x) = CR(y)$ to *dowolnie*

Z powyższej relacji wynika, że osobniki do kolejnej populacji wybierane są zgodnie z numerem frontu. Dopiero w przypadku, kiedy liczba osobników w danym froncie jest większa niż liczba miejsc do obsadzenia w kolejnej populacji, osobniki w tym froncie są szeregowane zgodnie z malejącą kolejnością wartości parametru *crowding distance i* w takiej też kolejności dodawane do populacji, aż jej liczebność nie osiągnie N.

5.5. Algorytm Genetyczny z ustaloną liczbą cech (algorytm Cullinga)

Klasyczny algorytm genetyczny z członem kary działa prawidłowo, jeżeli współczynniki wagowe przypisane do obu członów funkcji przystosowania zostaną właściwie dobrane. Dobranie ich nie jest jednak prostą sprawą zwłaszcza, jeżeli występuje duża nierównowaga między liczbą wszystkich cech w zbiorze cech, a liczbą

cech w podzbiorach kodowanych w osobnikach. W przypadku zbiorów cech o wysokiej wymiarowości, jeżeli klasyfikator ma działać jedynie na kilku wybranych cechach, znacznie lepszym rozwiązaniem jest ustalenie dopuszczalnej liczby cech i przeszukiwanie przestrzeni rozwiązań ograniczonej do podzbiorów cech o wybranej liczebności. Takie podejście wymaga oczywiście zmiany sposobu kodowania osobników.

Jednym z możliwych sposobów kodowania uwzględniających stałą liczbę cech w osobniku jest losowy wybór grup cech (np. grupa cech złożona z 7 cech) ze wszystkich dostępnych cech, które będą kolejno użyte do klasyfikacji (algorytm Culling) [576]. Algorytm Cullinga jest bardziej agresywny podczas przeszukiwania przestrzeni cech niż algorytm klasyczny [26], dzięki czemu jest bardziej efektywny w procesie selekcji cech. Zasada kodowania osobników w algorytmie Cullinga oparta jest na losowym wyborze grup cech o ustalonej liczebności (N) spośród wszystkich dostępnych cech (wzór 5.5) [26,577].

$$N_G = \frac{N_W}{N} \tag{5.5}$$

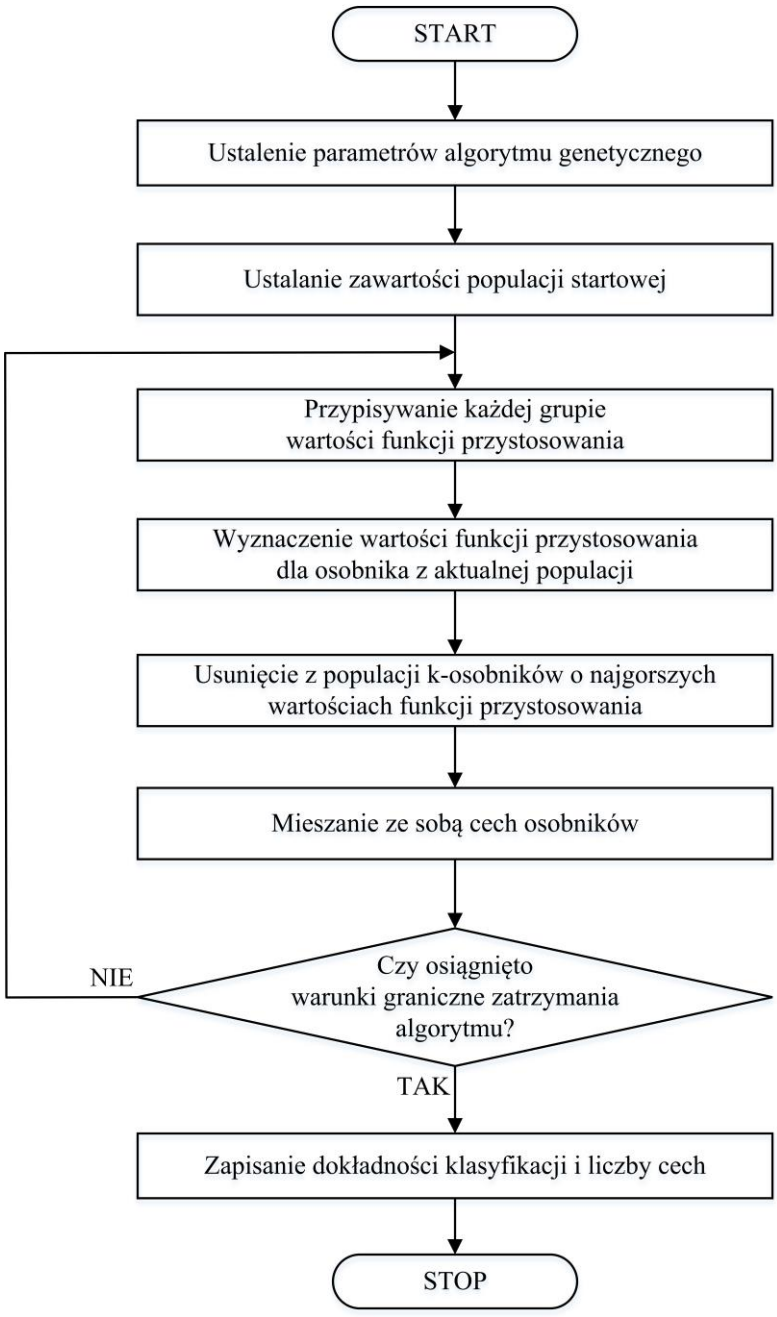
gdzie:

N_G – liczba grup cech (liczba osobników); N_G jest zaokrąglane w górę do wartości całkowitej;

N_W – zbiór wszystkich cech;

N – liczba cech wchodzących w skład jednej grupy (liczba genów w osobniku).

Na początku działania algorytmu (Rysunek 5.10) ustalana jest zawartość populacji startowej poprzez utworzenie N_G–losowych grup cech (jedna grupa jest jednym osobnikiem algorytmu). Losowanie cech do kolejnych osobników jest wykonywane bez zwracania, co oznacza, że w populacji startowej występują wszystkie cechy. Jeżeli pierwotnie obliczona wartość N_G (przed zaokrągleniem) była niecałkowita, to ostatni osobnik uzupełniany jest zerami tak, aby zawierał N genów. Każdej grupie przypisywana jest wartość funkcji przystosowania wynikająca z oceny działania klasyfikatora zawierającego cechy występujące w tej grupie. Następnie z populacji usuwanych jest k osobników o najgorszych wartościach funkcji przystosowania. Cechy występujące w osobnikach pozostałych w populacji są mieszane ze sobą. Proces ustalania kolejnej populacji osobników przebiega tak, jak w przypadku populacji startowej (geny są losowo dobierane do poszczególnych osobników).



Rysunek 5.10. Schemat algorytmu genetycznego z ustaloną liczbą cech (algorytm Cullinga).

Źródło: Opracowanie własne na podstawie [26].

Podstawowa różnicą występującą między pierwszą i kolejnymi populacjami jest taka, że w pierwszej populacji osobniki ustalane są na podstawie wszystkich możliwych cech, natomiast w kolejnych populacjach liczba cech się zmniejsza (w związku z usuwaniem osobników najsłabszych). Jako że rozmiar populacji pozostaje stały (każda populacja ma N_G osobników), osobniki w kolejnych populacjach zawierają powtarzające

się cechy. Algorytm kończy działanie najczęściej po ustalonej z góry liczbie iteracji, a jego ostatecznym wynikiem jest osobnik (zbiór cech), który uzyskał najwyższą wartość funkcji przystosowania spośród wszystkich osobników, które zostały utworzone w trakcie wszystkich iteracji algorytmu. Algorytm genetyczny Cullinga wyróżnia się spośród innych algorytmów genetycznych krótkim czasem działania oraz tym, że osiąga w ewolucji największe spodziewane zyski funkcji przystosowania [26].

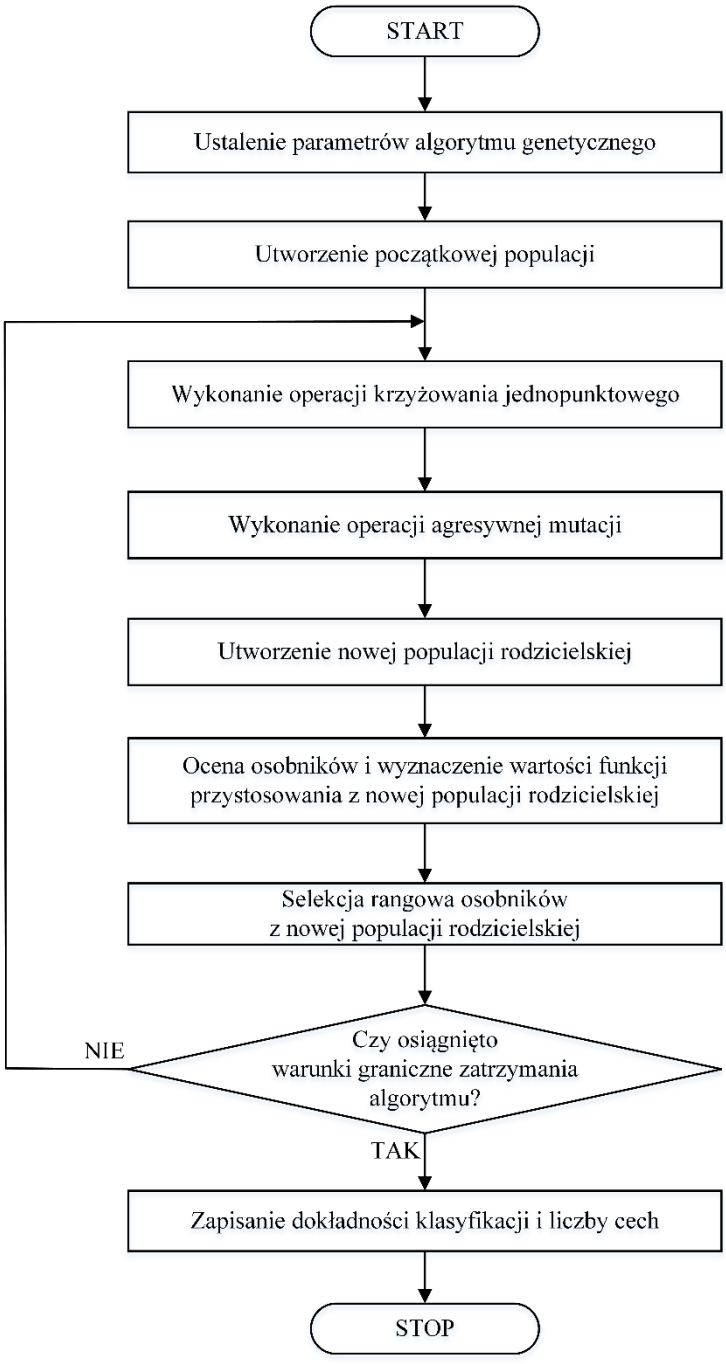
5.6. Algorytm genetyczny z agresywną mutacją (GAAM)

Algorytm genetyczny z agresywną mutacją GAAM (ang. *Genetic Algorithm with Aggressive Mutation*) [94,95] jest oparty, podobnie jak algorytm Cullinga, na strategii przeszukiwania podprzestrzeni cech o z góry ustalonej liczbie cech [578]. W algorytmie GAAM osobnik złożony jest z liczby genów równej założonej liczbie cech. Poszczególne geny osobnika przyjmują wartości całkowite ze zbioru $\{0, 1, 2, ..., F\}$, gdzie F jest wymiarem zbioru cech, a kolejne elementy odpowiadają indeksom cech w zbiorze wszystkich cech. Wartość zero, będąca pierwszym elementem zbioru, nie odpowiada żadnej z cech. Została ona wprowadzona po to, żeby umożliwić dodatkowe zmniejszenie wymiarowości osobnika. Za każdym razem, kiedy którykolwiek z genów przyjmie wartość zero, liczba wejść klasyfikatora dokonującego oceny osobnika jest zmniejszana o jeden [26,90,94]. Analogiczny efekt spowoduje wystąpienie w osobniku genów o takiej samej wartości cechy. Ujmując rzecz dokładniej, proces oceny osobnika o powtarzającym się genie rozpocznie się od ustawienia wartości takich samych genów (z wyjątkiem jednego) na wartość zero. W rezultacie do klasyfikatora zostanie wprowadzonych mniej cech i tym samym możliwe będzie sprawdzenie, czy przy wykorzystaniu mniejszej liczby cech można uzyskać taką samą dokładność klasyfikacji [95].

Podstawową operacją w algorytmie GAAM jest mutacja. Podczas tego etapu, mutacji poddawany jest nie tylko każdy osobnik w populacji, ale także każdy gen wspomnianego przed chwilą każdego osobnika. W wyniku takiego działania ogromna liczba nowej informacji genetycznej wprowadzana jest do populacji w kolejnych iteracjach algorytmu. Agresywna forma mutacji zastosowana w algorytmie GAAM odpowiedzialna jest za zmianę kolejności dwóch podstawowych kroków klasycznego algorytmu genetycznego, tj. kroku selekcji oraz reprodukcji. W zaproponowanym podejściu z agresywną mutacją, najpierw dokonywana jest reprodukcja, a następnie

selekcja. Podczas etapu reprodukcji populacja rodzicielska jest powiększana poprzez dodawanie nowych osobników, które zostały utworzone podczas etapu mutacji i krzyżowania. Następnie, poprzez wybór osobników z największą wartością funkcji przystosowania, populacja jest zmniejszana do początkowego rozmiaru.

Schemat działania algorytmu genetycznego z agresywną mutacją został przedstawiony na Rysunku 5.11.



Rysunek 5.11. Schemat algorytmu genetycznego z agresywną mutacją (GAAM).
Źródło: Opracowanie własne na podstawie [579].

Główne kroki algorytmu GAAM przedstawiają się następująco [579]:

1. Na początku działania algorytmu GAAM należy wyznaczyć parametry algorytmu: *T* – liczba pokoleń, *M* – liczba osobników w populacji, *N* – liczba genów w osobniku, *tourN* – liczba osobników w każdym turnieju na etapie selekcji.
2. Po ustawieniu parametrów algorytmu, losowana jest początkowa populacja rodzicielska. Populacja składa się z *M* losowo wybranych osobników, gdzie każdy osobnik składa się z *N* genów. Liczba genów w osobniku (parametr *N*) jest ustalana w zależności od liczby obserwacji, liczby klas i ogólnej charakterystyki danego problemu. Ponieważ każda cecha jest zakodowana przez jeden gen, zatem ten sam parametr określa jednocześnie liczbę cech poszukiwanych przez algorytm. Kolejne geny są losowane ze zbioru *{0, 1, ..., F}*, gdzie *F* to zbiór wszystkich możliwych cech.
3. Główna pętla algorytmu rozpoczyna się od wykonania dwóch operacji reprodukcji na osobnikach z populacji rodzicielskiej. Pierwsza operacja to klasyczne jednopunktowe krzyżowanie na osobnikach z poprzedniej populacji, wykonywane ze stałym prawdopodobieństwem równym jeden.
4. Druga operacja reprodukcji to agresywna mutacja – koncepcja specyficzna dla algorytmu GAAM. Podczas agresywnej mutacji każdy gen każdego osobnika jest mutowany indywidualnie przez ustawienie jego wartości na wskaźnik jednej losowo wybranej cechy ze zbioru cech. W wyniku operacji agresywnej mutacji, jeden osobnik rodzicielski posiada zbiór *N* osobników potomstwa, z których każde powstało poprzez zmutowanie innego genu tego osobnika. Pseudokod schematu agresywnej mutacji przedstawiono na Rysunku 5.12. Dodatkowo, dla dokładniejszej ilustracji procesu mutacji, na Rysunku 5.13 zaprezentowano zestaw wszystkich osobników powstałych w wyniku przeprowadzenia operacji mutacji na osobniku złożonym z 10 genów.
5. W kolejnym kroku tworzona jest nowa populacja rodzicielska poprzez połączenie ze sobą kolejno: *M* osobników z poprzedniej populacji rodzicielskiej, *M* (lub *M-1* dla nieparzystego *M*) osobników powstałych w wyniku operacji krzyżowania oraz *NxM* osobników powstałych w trakcie agresywnej mutacji.
6. Wszystkie *M(2+N)* (lub *M(2+N)-1*) osobniki z populacji rodzicielskiej są następnie oceniane przy wykorzystaniu dokładności klasyfikacji jako funkcji przystosowania. Podzbiór cech zakodowany w osobniku staje się danymi wejściowymi do wybranego modelu klasyfikatora, który jest następnie trenowany na dostępnym

zbiorze danych. Dokładność klasyfikatora wyznaczona w procesie walidacji krzyżowej jest przypisywana osobnikom jako wartość ich funkcji przystosowania. Ponieważ algorytm dopuszcza powielanie cech (które mogą pojawić się w wyniku operacji genetycznych lub losowej inicjalizacji populacji początkowej), przed wytrenowaniem klasyfikatora wszystkie identyczne geny (oprócz jednego) są ustawiane na zero, co powoduje wyeliminowanie ich z procesu klasyfikacji.

7. W trakcie ostatniej operacji, operacji selekcji, następuje odrzucenie osobników $M+MN$ o najmniejszej dokładności klasyfikacji. Po tym kroku w populacji pozostaje tylko M osobników zapewniających najwyższą dokładność klasyfikacji. Ponieważ w procesie selekcji biorą udział również wszystkie osobniki z populacji rodzicielskiej, zatem najlepszy osobnik w obecnej populacji ma co najmniej taką samą wartość funkcji przystosowania, jaka wystąpiła w populacji poprzedniej.

Algorytm kończy działanie po osiągnięciu określonej liczby iteracji (parametr T). W tym momencie osobniki o najwyższej wartości funkcji przystosowania z ostatniej populacji rodzicielskiej są zwracane jako wynik działania algorytmu GAAM. Dodatkowo, zapisywana jest dokładność klasyfikacji, czas pracy algorytmu oraz liczba cech osobnika.

Algorytm jest wykonywany w pętli T–razy. Trzy główne części tej pętli, tj. krzyżowanie, mutacja i ewaluacja, determinują złożoność całego algorytmu. Ponieważ złożoność krzyżowania dla pojedynczego pokolenia wynosi $O(M)$, a złożoność mutacji i ewaluacji wynoszą $O(NM)$, stąd też ogólna złożoność dla całego algorytmu wynosi $O(TNM)$ [95].

```
j = 0
 for i = 1 to M
        for g = 1 to N
                new = population ( i, 1 : N )
                new(g)=random( { 0,1,...,F } )
                j = j + +
                population (M + j ) = new
        end
 end
```

Rysunek 5.12. Pseudokod schematu agresywnej mutacji; $population$ – macierz osobników; *new* – wektor zawierający wszystkie geny *i*-tego osobnika; *random({0,1,...,F})* – funkcja zwracająca losową cechę ze zbioru cech.

Źródło: Opracowanie własne na podstawie [579].

X	2	3	4	5	6	7	8	9	10
1	X	3	4	5	6	7	8	9	10
1	2	X	4	5	6	7	8	9	10
1	2	3	X	5	6	7	8	9	10
1	2	3	4	X	6	7	8	9	10
1	2	3	4	5	X	7	8	9	10
1	2	3	4	5	6	X	8	9	10
1	2	3	4	5	6	7	X	9	10
1	2	3	4	5	6	7	8	X	10
1	2	3	4	5	6	7	8	9	X

Rysunek 5.13. Przykład zbioru wszystkich osobników potomnych powstałych z tego samego osobnika rodzicielskiego; *X* – gen zmieniony w osobniku potomnym
Źródło: Opracowanie własne.

5.7. Algorytm genetyczny z agresywną mutacją i zmniejszającą się liczbą genów (melting-GAAM)

Główną wadą metody algorytmu genetycznego z agresywną mutacją oraz podejścia Cullinga [4,22,40,69,83,90,94,95] jest to, że procesy te przetwarzają osobniki o ilości cech zdefiniowanej na początku algorytmu. Ponieważ oba algorytmy nie mogą tworzyć osobników z mniejszą liczbą cech w kolejnych pokoleniach, dlatego też nie mogą zmniejszać danego rozmiaru wektora cech. Co prawda, w algorytmie genetycznym z agresywną mutacją, teoretycznie możliwe jest uzyskanie mniejszego wektora cech, gdy jeden lub więcej genów danego osobnika przyjmuje wartość zero, jednakże rzeczywiste prawdopodobieństwo takiego zdarzenia jest bardzo niskie. Stąd można stwierdzić, że żaden z dotychczas opisanych algorytmów nie posiada narzędzi, które wspierałyby dalszą redukcję cech w osobniku. O ile w przypadku niektórych aplikacji możliwe jest kilkakrotne uruchomienie algorytmu, za każdym razem z mniejszą liczbą genów w osobniku i manualnym wyborem końcowej liczby cech, to w badaniach nad interfejsem BCI takie podejście jest niemożliwe do zrealizowania. Jest to związane z tym, że sam algorytm BCI musi być samowystarczalny. Oznacza to, że system BCI podczas rekalibracji nie może wymagać ingerencji inżyniera obsługującego interfejs. W związku z tym, odpowiednim algorytmem genetycznym do selekcji cech w systemie BCI jest taki algorytm, w którym proces optymalizacji nie tylko maksymalizuje dokładność

klasyfikatora, ale także minimalizuje liczbę cech. Poniżej przedstawiono na Rysunku 5.14 schemat algorytmu genetycznego spełniającego obydwa wymagania [22]:

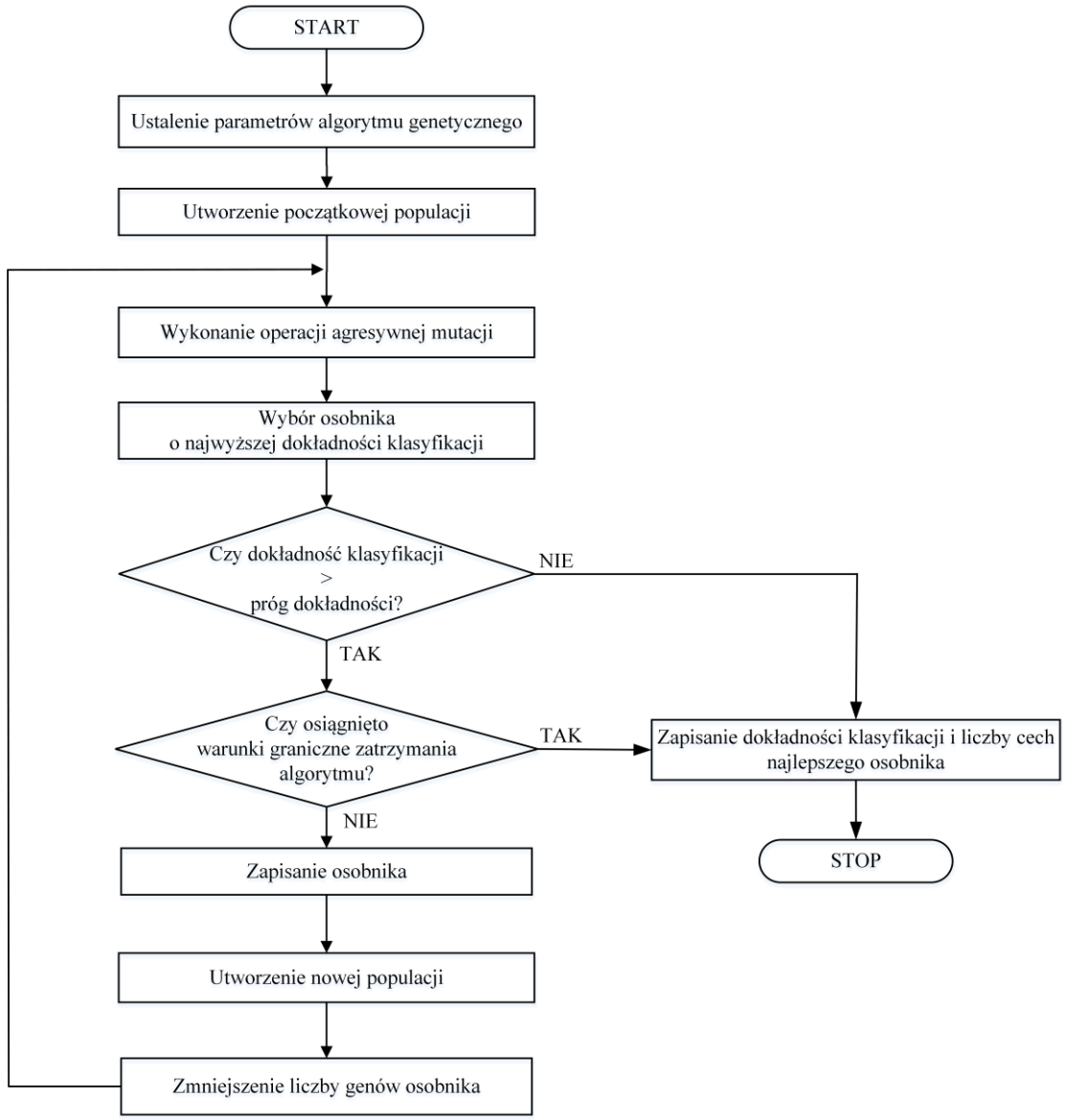
1. Określenie dwóch parametrów algorytmu:
   - maksymalnej liczby genów osobnika N_{max}
   - progu dokładności T_C

   Ustawienie aktualnej liczby genów osobnika od N do N_{max} $(N = N_{max})$
2. Uruchomienie algorytmu z agresywną mutacją dla M osobników, gdzie każdy zawiera N genów.
3. Wybór osobnika o najwyższej dokładności klasyfikacji T_h.

   Jeśli dokładność T_h jest większa lub równa progowi dokładności T_C, tzn. $T_h \geq T_C$, zachowaj osobnika i przejdź do punktu 4.

   Jeśli dokładność T_h jest mniejsza niż próg dokładności T_C, tzn. $T_h < T_C$, zakończ algorytm i zwróć:
   - najlepszego osobnika z obecnej populacji (jeżeli $N = N_{max}$),
   - najlepszego osobnika z poprzedniej populacji (jeżeli $N < N_{max}$),
4. Zmniejszenie liczbę genów, $N = N - 1$.
5. Utworzenie nowej populacji poprzez skopiowanie osobników z populacji o największej średniej funkcji przystosowania otrzymanej z poprzedniego uruchomienia algorytmu genetycznego z agresywną mutacją. W celu uzyskania osobników o zmniejszonej liczbie genów, należy losowo wybrać i odrzucić jeden gen od każdego osobnika.
6. Powtórzenie kroków 2 - 5.

Dane wyjściowe opisanego algorytmu mogą zostać wyprowadzone w dwóch postaciach: zarówno w postaci pojedynczego osobnika o najwyższej dokładności klasyfikacji i najmniejszej liczbie cech, jak i w postaci zbioru osobników o najwyższej dokładności klasyfikacji otrzymanych przez kolejne uruchomienia algorytmu genetycznego z agresywną mutacją. W związku z tym ostatecznego wyboru najlepszego zbioru cech można dokonać zarówno w trybie nadzorowanym jak i nienadzorowanym.



Rysunek 5.14. Schemat algorytmu genetycznego ze zmniejszającą się liczbą genów w osobniku (Melting-GAAM)

Źródło: Opracowanie własne na podstawie [22].

W rzeczywistości oba tryby są używane w interfejsie BCI. Tryb nadzorowany jest zalecany do wstępnej kalibracji, ponieważ nadal obecny jest inżynier obsługujący interfejs, który w każdej chwili może przejrzeć raport, a następnie wybrać cechy przy zachowaniu najlepszego kompromisu między liczbą cech a dokładnością klasyfikacji. W związku z tym próg klasyfikacji można zdefiniować bardziej precyzyjnie. Z kolei w trakcie rekalibracji interfejsu przeprowadzanej najczęściej każdorazowo po uruchomieniu interfejsu, już bez zewnętrznego nadzoru, możliwe jest prowadzenie procesu selekcji cech w sposób całkowicie automatyczny.

5.8. GAAMmf – Algorytm genetyczny z agresywną mutacją i malejącą liczbą cech

Głównym celem wprowadzenia nowego algorytmu, nazwanego GAAMmf (ang. *Genetic Algorithm with Aggressive Mutation and minimum Feature*) było zapewnienie pełnej automatyzacji procesu doboru cech, pozwalającej na rzeczywistą optymalizację dwukryterialną. W tym celu w algorytmie zaimplementowano nową funkcję przystosowania, która jednocześnie uwzględnia dwa kryteria: dokładność klasyfikacji oraz liczbę cech kodowanych w osobniku. Funkcja ta została zaprojektowana w taki sposób, aby umożliwić balans pomiędzy jakością klasyfikacji a liczbą cech. Aby połączyć oba kryteria, w algorytmie zastosowano koncepcję rang, znaną z algorytmu NSGA-II. Rangi są tu przypisywane dwukrotnie, najpierw w odniesieniu do dokładności klasyfikacji, następnie w odniesieniu do liczby cech. Wyniki przeprowadzonych testów wykazały, że wprowadzenie funkcji przystosowania uwzględniającej liczbę cech znacznie zmniejszyło liczbę zwracanych cech, przy jednoczesnym utrzymaniu wysokiej dokładności klasyfikacji.

Funkcja przystosowania zastosowana w algorytmie GAAMmf dokonuje oceny każdego osobnika w odniesieniu do całej aktualnej populacji osobników. Jej dokładny schemat jest następujący. Na początku dla każdego osobnika budowany jest klasyfikator wyposażony w zestaw cech zakodowanych w tymże osobniku. Klasyfikatory są następnie trenowane na posiadanym zbiorze danych (zgodnie z 10-krotną walidacją krzyżową). Po zakończeniu procesu uczenia średnie dokładności klasyfikacji kolejnych klasyfikatorów, wyznaczone z dziesięciu zbiorów testowych, są przypisywana do odpowiadających im osobników. W kolejnym kroku osobniki z aktualnej populacji są sortowane według rosnącej dokładności klasyfikacji i następuje przydzielenie im rang (*accRank*) zgodnie z regułą, że osobniki o identycznej dokładności (zaokrąglonej do wartości całkowitych) otrzymują tę samą rangę, przy czym ranga numer 1 (najgorsza) jest przydzielana osobnikom o najniższej dokładności.

Następnie dla każdego osobnika zliczana jest liczba zawartych w nich cech, po czym osobniki są sortowane według malejącej liczby cech oraz przypisywane są im rangi (*fsRank*) zgodnie z zasadą, że osobniki o takiej samej liczbie cech otrzymują tę samą rangę, przy czym ranga numer 1 (najgorsza) jest przypisana osobnikom o największej liczbie cech. Aby uniknąć problemów związanych z różnicą w zakresie rang dla obu

kryteriów, przed przypisaniem rang dokonuje się normalizacji wartości obu kryteriów z wykorzystanie pseudo normalizacji min-max. Termin „pseudo min-max" oznacza, że wartości kryteriów nie są normalizowane w odniesieniu do rzeczywistej minimalnej i maksymalnej dokładności czy liczby cech ustalonych dla aktualnej populacji, lecz w odniesieniu do stałych wartości granic, ustalonych na 0 i 100 dla dokładności klasyfikacji oraz na 1 i N dla liczby cech. Wzory 5.12 oraz 5.13 przedstawiają zastosowane procedury normalizacyjne, przy czym wzór 5.12 pokazuje normalizację kryterium dokładności klasyfikacji, a wzór 5.13 normalizację kryterium liczby cech.

$$accRankN(i) = \frac{accRank(i)}{100} \tag{5.12}$$

gdzie: $accRank(i)$ – ranga osobnika po posortowaniu osobników według rosnącej dokładności klasyfikacji (ranga numer 1 jest przydzielana osobnikom o najniższej dokładności); $accRankN(i)$ – ranga osobnika po normalizacji (wartość 0 przypisywana jest osobnikom o najniższej dokładności, a wartość 1 – osobnikom o dokładności najwyższej).

$$fsRankN(i) = \frac{fsRank(i) - 1}{N - 1} \tag{5.13}$$

gdzie: *fsRank(i)* – ranga osobnika po posortowaniu osobników według malejącej liczby cech (ranga numer 1 jest przypisana osobnikom o największej liczbie cech); *fsRankN(i)* – ranga osobnika po normalizacji (wartość 0 przypisywana jest osobnikom o najmniejszej liczbie cech, a wartość 1 osobnikom o największej liczbie cech).

Ostateczna wartość przystosowania osobnika $fit(i)$ jest wyznaczana poprzez zsumowanie znormalizowanych wartości rang, zgodnie z wzorem 5.14.

$$fit(i) = accWeight * accFactor * accRankN(i) + fsWeight * fsFactor * fsRankN(i) \tag{5.14}$$

gdzie: *accWeight* – waga kryterium dokładności klasyfikacji; *accFactor* – stała wyznaczona z wzoru 5.15; *fsWeight*– waga kryterium liczby cech; *fsFactor*– stała wyznaczona z wzoru 5.16.

Jak można zaobserwować we wzorze 5.14, znormalizowane wartości rang wyznaczone dla danego osobnika nie są sumowane bezpośrednio, lecz są najpierw modyfikowane przy pomocy dwóch par modyfikatorów. Para pierwsza *accWeight* i *fsWeight* została dodana po to, aby zapewnić możliwość sterowania udziałem obu kryteriów w funkcji przystosowania. Domyślnie wartości obu wag są ustalone na 1, ale użytkownik algorytmu może je dowolnie modyfikować.

Druga para modyfikatorów *accFactor* i *fsFactor* to dwie stałe, które są wyznaczane na początku algorytmu. Stałe te mają na celu zapewnienie równego udziału obu kryteriów (dokładności klasyfikacji oraz liczby cech) w całkowitej wartości funkcji przystosowania. Ich wartości obliczane są na podstawie liczby możliwych poziomów dokładności i liczby możliwych cech w osobniku. Domyślnie w algorytmie zakłada się, że dokładność zmienia się od 0% do 100%, a liczba cech od 1 do *N* (dla *N*-1 poziomów), więc stałe $accFactor$ i $fsFactor$ są obliczane zgodnie z wzorami 5.15 i 5.16. Naturalnie w przypadku założenia innego zakresu zmian dokładności bądź liczby cech, przedstawione wzory powinny zostać adekwatnie dostosowane.

$$accFactor = 100\frac{100}{\mathrm{N}-1} \tag{5.15}$$

$$fsFactor = (N-1)\frac{100}{\mathrm{N}-1} = 100 \tag{5.16}$$

Podsumowując, w algorytmie GAAMmf wykorzystano całkowicie nową funkcję przystosowania, która uwzględnia zarówno znormalizowane rangi dotyczące dokładności osobników, jak i znormalizowane rangi dotyczące liczby cech w osobnikach wraz z odpowiadającymi im wagami. Ostatecznie, wyniki uzyskane dla obu kryteriów są sumowane, co stanowi kluczowy element procesu optymalizacji. Przedstawione podejście wykazuje wysoką skuteczność i pozwala na uzyskanie optymalnych wyników w selekcji cech w kontekście badanych kryteriów.

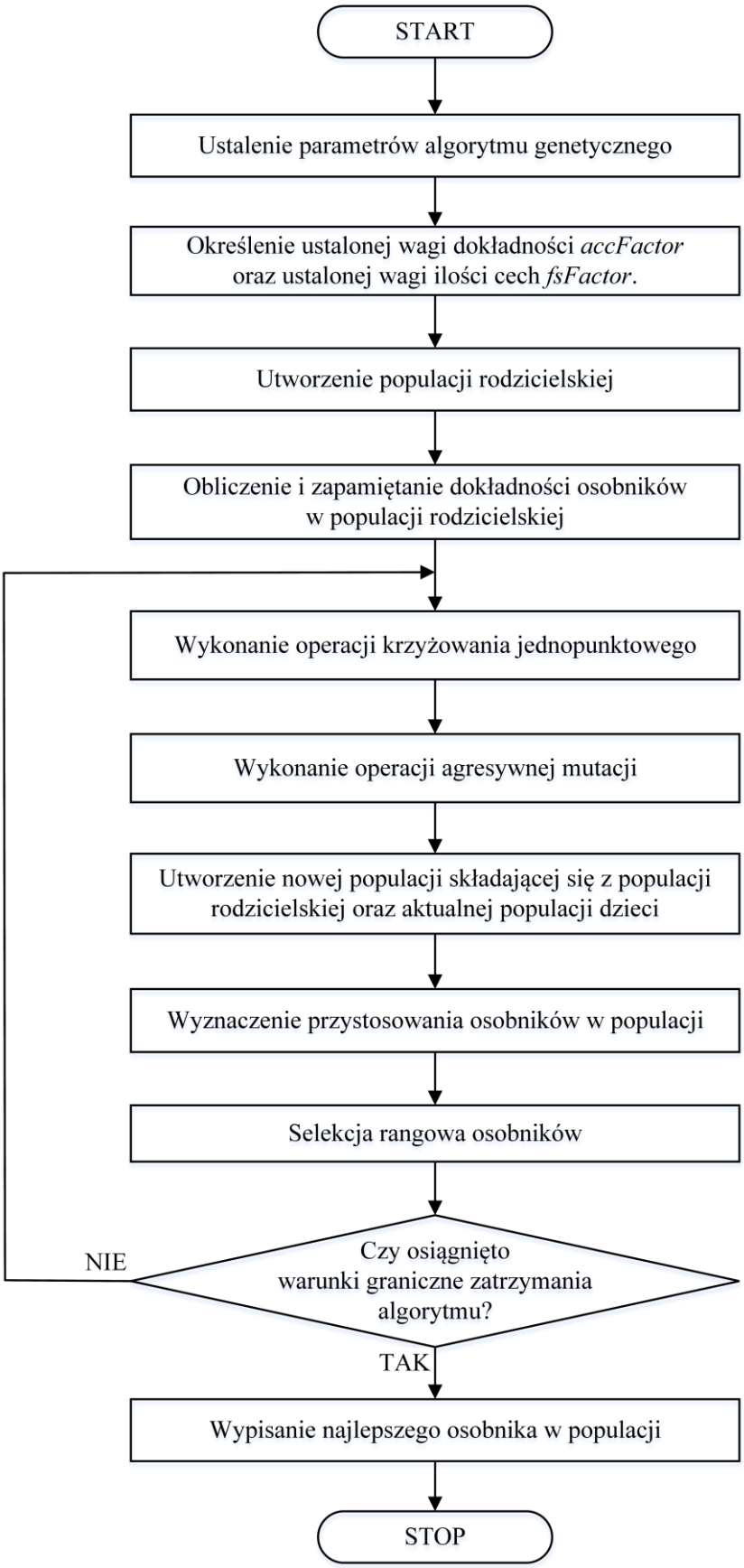
Poza nową funkcją przystosowania, w algorytmie GAAMmf zastosowano również kilka innych drobnych modyfikacji. Przede wszystkim parametr *N,* określający liczbę genów osobnika odnosi się teraz tylko do osobników z populacji startowej. Wartość tego

parametru musi być wartością całkowitą z przedziału $\{1, ..., F\}$, gdzie F oznacza liczbę wszystkich cech. Dokładna wartości tego parametru nie wpływa na prawidłowość działania algorytmu (będzie on pracował prawidłowo dla każdej wartości z podanego przedziału), jednakże wpływa na czas, w którym zostanie znalezione rozwiązanie. Stąd, wartość parametru N powinna zostać ustawiona w przybliżeniu na maksymalną akceptowalną liczbę cech.

Kolejną drobną modyfikacją jest zmiana procedury selekcji z selekcji turniejowej (która była wykorzystana w algorytmie GAAM) na selekcję opartą na rankingu, co znacznie zmniejsza czas obliczeń. W selekcji rangowej populacja jest najpierw sortowana w porządku malejącym na podstawie wartości funkcji przystosowania, a każdemu osobnikowi jest przypisywana ranga na podstawie jego pozycji w posortowanej populacji. Im lepsza wartość funkcji przystosowania, tym niższa ranga. Osobniki o tej samej wartości funkcji przystosowania mają taką samą rangę. Ostatecznie, k osobników o najniższych rangach jest przenoszonych do następnej populacji.

W algorytmie GAAMmf zdecydowano się również wykorzystać parametr „*prawdopodobieństwa mutacji*" który został wcześniej zastosowany w algorytmie GAAM. Warto podkreślić, że wielkość tego parametru jest ustalana na początku działania algorytmu, co umożliwia kontrolowanie intensywności procesu mutacji. Wykorzystanie tego rozwiązania przyczynia się do zwiększenia skalowalności algorytmu oraz umożliwia jego efektywne wykorzystanie w problemach opisanych w przestrzeni o wysokiej wymiarowości. Zastosowanie parametru „*prawdopodobieństwo mutacji*" pozwala na zachowanie równowagi pomiędzy intensywnością procesu mutacji a zróżnicowaniem populacji, co z kolei prowadzi do osiągnięcia lepszej jakości rozwiązań.

Szczegółowy schemat algorytmu GAAMmf przedstawiono na Rysunku 5.15.



Rysunek 5.15. Algorytm GAAMmf.

Źródło: Opracowanie własne.

Działania wykonywane na jego kolejnych etapach są następujące [579]:

1. Na początku należy wyznaczyć podstawowe parametry algorytmu: *T* – liczba pokoleń, *M* – liczba osobników w populacji, *N* – startowa liczba genów w osobniku, *p_mutacji* – prawdopodobieństwo mutacji, *accWeight* – waga kryterium dokładności, *fsWeight* – waga kryterium liczby cech w osobniku.
2. Wyznaczenie wartości modyfikatorów *accFactor* oraz *fsFactor*.
3. Wylosowanie *M* osobników o *N* genach do populacji rodzicielskiej.
4. Wyznaczenie dokładności osobników wchodzących w skład populacji rodzicielskiej.
5. Przeprowadzenie operacji krzyżowania na osobnikach należących do populacji rodzicielskiej (klasyczne krzyżowanie jednopunktowe, przeprowadzane ze stałym prawdopodobieństwem równym jeden)
6. Przeprowadzenie operacji agresywnej mutacji (zgodnie z ustalonym na początku prawdopodobieństwem *p_mutacji*).
7. Utworzenie nowej populacji poprzez połączenie: *M* osobników z poprzedniej populacji rodzicielskiej, *M* (lub *M-1* dla nieparzystego *M*) osobników powstałych w wyniku operacji krzyżowania oraz wszystkich osobników uzyskanych w trakcie operacji agresywnej mutacji.
8. Wyznaczenie wartości funkcji przystosowania (zgodnie z wzorem 5.14) dla wszystkich osobników w połączonej populacji.
9. Wybór *M* osobników do kolejnej populacji rodzicielskiej zgodnie z zasadami selekcji rangowej.
10. Sprawdzenie, czy osiągnięto warunek zatrzymania algorytmu (czyli czy wykonano zadaną liczbę iteracji). Jeśli nie, to następuje powrót do punktu 5. Jeśli tak, to z ostatniej populacji wybierany jest osobnik o najwyższej wartości funkcji przystosowania, którego parametry (dokładność klasyfikacji oraz wybrane cechy) są wypisywane jako wynik działania algorytmu, po czym algorytm kończy działanie.

Rozdział VI
Weryfikacja działania algorytmu GAAMmf

W celu oceny skuteczności algorytmu GAAMmf, przeprowadzono trzy eksperymenty badawcze. Eksperyment pierwszy przeprowadzono na 5 zbiorach zarejestrowanych podczas pracy z interfejsem mózg – komputer. Celem eksperymentu było zbadanie jaki wpływ na działanie algorytmu mają zmiany wartości jego parametrów. Badane tutaj były przede wszystkim zmiany dwóch parametrów kontrolujących znaczenie zastosowanych kryteriów optymalizacyjnych: dokładności klasyfikacji (parametr *accWeight*) oraz liczby cech (parametr *fsWeight*). Dodatkowo w eksperymencie pierwszym zbadano stabilność algorytmu dla kilku uruchomień oraz wpływ na wyniki algorytmu różnych poziomów prawdopodobieństwa mutacji.

W ramach drugiego eksperymentu porównano wyniki osiągnięte przez algorytm GAAMmf z wynikami uzyskanymi przy użyciu kilku innych algorytmów genetycznych, tj. algorytmu GAAM, algorytmu Melting GAAM, algorytmu NSGA II oraz algorytmu Hollanda z funkcją kary. Ponadto, wyniki zaproponowanego algorytmu zostały porównane z wynikami uzyskanymi przez zastosowanie czterech niegenetycznych metod selekcji cech, tj. Lasso, CFS, ReliefF oraz SFS. W eksperymencie wykorzystano ten sam zestaw zbiorów, co w eksperymencie pierwszym.

W ramach trzeciego eksperymentu przeprowadzono badania mające na celu wykazanie, że zaproponowany algorytm może być z powodzeniem wykorzystywany nie tylko w procesie selekcji cech wyekstrahowanych ze zbiorów sygnałów EEG, lecz również w przypadku standardowych zadań klasyfikacyjnych. W tym celu wyznaczono wyniki algorytmu GAAMmf dla jedenastu benchmarkowych zbiorów danych różniących się między sobą liczbą cech, klas oraz przykładów, a następnie porównano je z wynikami metod referencyjnych. W porównaniu wykorzystano ten sam zestaw metod, co w eksperymencie drugim, czyli algorytmy genetyczne: GAAM, Melting-GAAM, NSGA II, Holland z funkcją kary oraz metody niegenetyczne: Lasso, CFS, ReliefF oraz SFS.

We wszystkich eksperymentach w procesie klasyfikacji wykorzystano klasyfikator LDA (ang. *Linear Discrimination Analysis*). Dokładność klasyfikatora szacowano przy wykorzystaniu 10-krotnej walidacji krzyżowej. Miarą dokładności klasyfikatora była średnia wartość obliczona na podstawie dokładności klasyfikacji uzyskanej dla 10 zestawów walidacyjnych.

Wszystkie eksperymenty zostały przeprowadzone na komputerze z systemem operacyjnym Windows 10 Pro x64, z procesorem AMD Ryzen 5 1400 z czterema rdzeniami (Quad-Core) o częstotliwości 3,20 GHz i 16 GB pamięci RAM.

6.1. Eksperyment 1 – badanie odporności i stabilności algorytmu

Celem eksperymentu pierwszego było zbadanie jaki wpływ na działanie algorytmu mają zmiany wartości jego parametrów. Badania przeprowadzone w eksperymencie podzielono na 4 etapy:

- Etap 1 – badanie wpływu zmian wartości parametrów *accWeight* oraz *fsWeight* na wyniki algorytmu.
- Etap 2 – badanie wpływu zmian wartości parametru *probM* na wyniki algorytmu.
- Etap 3 – badanie stabilności wyników algorytmu.
- Etap 4 – badanie wpływu charakterystyki zbioru danych na wyniki algorytmu.

6.1.1. Zbiory danych wykorzystane w eksperymencie 1

Eksperyment pierwszy został przeprowadzony na pięciu zbiorach BCI. Zbiory pochodziły ze strony internetowej projektu „*Brain/Neural Computer Interaction*" *BNCI Horizon 2020* [580], który wspiera współpracę i komunikację między użytkownikami w dziedzinie interfejsów mózg-komputer oraz udostępnia repozytorium z linkami do publicznie dostępnych zbiorów BCI [581], a także ze strony internetowej organizacji *Berlin Brain-Computer Interface* (*BBCI*) [582], na której publikowane są informacje na temat konkursów BCI oraz ogólnodostępne zbiory BCI [583]. Ogólna charakterystyka zbiorów, wraz z procedurą zastosowaną podczas ich rejestracji została zaprezentowana poniżej.

Zbiór 1 (motor imagery, 13 kanałów EEG): Opisywany zbiór powstał w trakcie pracy z interfejsem opartym na wyobrażeniu ruchu. Sygnał EEG rejestrowano z 13 kanałów: FC3, FCz, FC4, C5, C3, C1, Cz, C2, C4, C6, CP3, CPz, CP4 za pomocą urządzenia GAMMAsys z systemem aktywnych elektrod, rejestrującego dane z częstotliwością 512 Hz. Sterowanie w interfejsie odbywało się za pomocą wyobrażenia ruchu prawej ręki lub obu stóp. Każda próba rozpoczynała się od sygnału dźwiękowego oraz wyświetlenia na ekranie monitora jednego z dwóch symboli graficznych: strzałki w prawo (oznaczającej wyobrażenie ruchu prawej ręki) lub strzałki w dół (oznaczającej wyobrażenie ruchu stóp). Symbol graficzny był wyświetlany w okresie od t=3 s, do t=4.25 s. Okres aktywności, w którym użytkownicy otrzymywali informację zwrotną, trwał od sekundy 4 do 8 (pięciosekundowy okres wyobrażenia ruchu). Pomiędzy próbami występowała losowa przerwa trwająca od 2 do 3 sekund. Zarejestrowany zbiór zawiera dane pochodzące od 12 osób. Dla każdej osoby zarejestrowano 199 prób należących do 2 klas: klasy „1" oraz klasy „2". Każda próba jest opisana przez 13 sygnałów EEG, każdy zawierający 2560 próbek (5 sekund x 512 próbek/sekundę).

Zbiór 2 (slow cortical potentials, 6 kanałów EEG): Zbiór zawiera dane zarejestrowane w trakcie korzystania z interfejsu opartego na paradygmacie wolnych potencjałów korowych (ang. *slow cortical potentials*). Zadaniem osoby badanej było przesuwanie na ekranie komputera kursora w górę i w dół poprzez wywołanie dodatnich oraz ujemnych zmian wolnych potencjałów korowych. Jeżeli sygnał był ujemny, wówczas kursor przesuwał się na ekranie w dół, w przypadku sygnału dodatniego, kursor poruszał się w górę. Podczas rejestrowania sygnału EEG badany otrzymywał wizualną informację zwrotną. Każda próba trwała 6 sekund, a wizualna informacja zwrotna była prezentowana badanemu w czasie od t=2 s. do t=5.5 s. Dane zarejestrowane zostały w 6 kanałach (2 kanały były zlokalizowane w okolicy Cz, 2 kanały w okolicy C3 oraz 2 kanały w okolicy C4), a częstotliwość próbkowania wynosiła 256 Hz. W dwóch różnych dniach zarejestrowano i zmieszano losowo 268 prób (168 prób pochodziła z pierwszego dnia, a pozostałe 100 prób z drugiego dnia). Zbiór zawiera dane ze 135 prób należących do klasy 0 oraz 133 prób należących do klasy 1. Każda próba jest opisana przez 6 sygnałów EEG, każdy zawierający 896 próbek (3.5 sekundy x 256 próbek/sekundę).

Zbiór 3 (motor imagery, 64 kanały ECoG): Trzeci zbiór odnosi się, podobnie jak zbiór pierwszy, do interfejsu opartego na wyobrażeniu ruchu i obejmuje sygnały zarejestrowane u tej samej osoby podczas wykonywania tego samego zadania, ale w dwóch różnych dniach z około tygodniową przerwą. Podczas eksperymentu badany musiał wyobrażać sobie ruchy języka lub małego palca lewej ręki. Do badania wykorzystano siatkę 64 elektrod ECoG (siatka 8x8 elektrod), którą umieszczono nad prawą korą ruchową. Przyjęto, że siatka pokrywa całkowicie prawą korę ruchową, jednak ze względu na swoje rozmiary (ok. 8x8cm) częściowo pokrywała również otaczające obszary kory. Wszystkie badania wykonano z częstotliwością próbkowania 1000 Hz, a wyniki zapisano w mikrowoltach. Pojedyncza próba (jedno wyobrażenie ruchu językiem lub małym palcem lewej ręki) trwała 3 sekundy. Aby uniknąć odzwierciedlenia w danych wywoływanych potencjałów wzrokowych, zapis rozpoczynano w 0.5 sekundy po zakończeniu wyświetlania wskazówki. Zbiór zawiera dane opisujące 278 prób należących do 2 klas: klasy „-1" oraz klasy „1". Każda próba jest opisana przez 64 sygnałów ECoG, każdy zawierający 2500 próbek (2.5 sekund x 1000 próbek/sekundę).

Zbiór 4 (motor imagery, 60 kanałów EEG, 4 klasy): Zbiór opisuje paradygmat wyobrażenia ruchu i przedstawia dane zarejestrowane u uczestnika, którego zadaniem było wyobrażanie ruchów lewej ręki, prawej ręki, stopy lub języka. Schemat pojedynczej próby był następujący. W drugiej sekundzie generowany był sygnał akustyczny sygnalizujący początek próby oraz na ekranie wyświetlany był krzyżyk „+", na którym osoba badana skupiała swoją uwagę. Następnie, w sekundzie trzeciej na ekranie wyświetlana była strzałka o zwrocie w lewo, w prawo, w górę lub w dół. Zadaniem badanego było wyobrażanie sobie (przez kolejne 4 sekundy) ruchu zgodnego ze strzałką, czyli odpowiednio ruchu lewej ręki, prawej ręki, języka lub stopy (klasy: „1", „2", „3" i „4"). Rejestracji danych EEG dokonano za pomocą 64-kanałowego wzmacniacza EEG firmy Neuroscan, Zarejestrowano sygnały z 60 kanałów EEG. Liczba poprawnie zarejestrowanych prób wynosiła 180. Próbkowanie danych wynosiło 250 Hz, zatem dla każdej próby i każdego kanału uzyskano 1000 próbek sygnału (4 sekundy x 250 próbek/sekundę).

Zbiór 5 (motor imagery, 3 kanały EEG): Zbiór danych zawiera dane zarejestrowane w trakcie korzystania z interfejsu opartego na paradygmacie potencjałów motorycznych. Zbiór ten został zarejestrowany w trakcie eksperymentu z 25-letnią kobietą. W trakcie

eksperymentu kobieta została poproszona o wykonanie zadania polegającego na przesuwaniu paska widocznego na ekranie w lewą lub prawą stronę, wykorzystując wyobrażone ruchy lewej i prawej ręki, zgodnie z losowymi wskazówkami wyświetlanymi na ekranie. Eksperyment składał się z 280 prób, podzielonych na 7 serii po 40 prób każda, przeprowadzonych w ciągu jednego dnia. Czas trwania każdej próby wynosił 9 sekund, z czego w pierwszych dwóch sekundach nic się nie działo, a w trzeciej sekundzie generowany był sygnał akustyczny wskazujący początek próby. W tym samym czasie na ekranie wyświetlany był krzyżyk "+". Następnie w czasie t=3 sekundy na ekranie pojawiała się strzałka wskazująca, czy badany powinien wyobrazić sobie ruch lewej czy prawej ręki, a tym samym przesunąć widoczny na ekranie pasek w lewą lub prawą stronę. W trakcie eksperymentu sygnały EEG zostały zarejestrowane w trzech bipolarnych kanałach EEG (C3, Cz i C4) z próbkowaniem 128 Hz. Przed analizą, sygnał został przefiltrowany w paśmie częstotliwości 0.5 - 30 Hz. Zbiór danych, składający się z 280 prób, został podzielony na dwa równe podzbiory: pierwszy przeznaczony do szkolenia klasyfikatora i drugi przeznaczony do zewnętrznego testowania klasyfikatora. W niniejszej pracy wykorzystano tylko pierwszy podzbiór, który zawierał wartości docelowe (1 - lewa ręka, 2 - prawa ręka). Pojedyncza próba trwała 9 sekund a próbkowanie danych wynosiło 128 Hz, więc uzyskano 1152 próbki sygnału dla każdej próby i kanału (9 sekund x 128 próbek/sekundę).

Oryginalne zbiory danych zostały przekształcone przy wykorzystaniu pasmowo-przepustowego filtru Butterwortha na zbiory cech, reprezentujące moc sygnału w badanych pasmach częstotliwości. W badaniu wykorzystano następujące pasma częstotliwości:

- pasmo delta (1-4 Hz) oraz dwa podpasma pasma delta (1-2 Hz, 2-4 Hz);
- pasmo theta (4-8 Hz) oraz dwa podpasma pasma theta (4-6 Hz, 6-8 Hz);
- pasmo alfa (8-13 Hz) oraz pięć podpasm pasma alfa (8-9 Hz, 9-10 Hz, 10-11 Hz, 11-12 Hz, 12-13 Hz);
- pasmo beta (13-30 Hz) oraz pięć podpasm pasma beta (13-17 Hz, 17-20 Hz, 20-23 Hz, 23-26 Hz, 26-30 Hz).

Wartości cech wyznaczone dla każdego ze zbiorów znormalizowano stosując normalizację min-max (wzór 6.1). W ten sposób przeskalowano wartości cech do przedziału <0,1> unikając tym samym sytuacji, w której jedna cecha mogłaby

dominować nad innymi, co mogłoby prowadzić do błędnych wniosków lub fałszywych wyników analizy. Następnie w każdym zbiorze zidentyfikowano pary cech, których liniowa korelacja (mierzona współczynnikiem Pearsona [584-586]) przekraczała 99% i usunięto po jednej cesze z każdej takiej pary. Szczegółowa charakterystyka zbiorów cech przed i po eliminacji cech jest przedstawiona w Tabeli 6.1.

$$x_{zn} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{6.1}$$

gdzie: x_{zn} – znormalizowana wartość cechy x; x – pierwotna wartość cechy x; *min(x)* – minimalna wartość cechy x; *max(x)* – maksymalna wartość cechy x.

Zbiór danych	Liczba cech	Liczba klas	Liczba prób	Źródło
Zbiór 1 BCI	1170/**1165**	2	199	[587]
Zbiór 2 BCI	324/**312**	2	268	[588]
Zbiór 3 BCI	2304/**1579**	2	278	[589]
Zbiór 4 BCI	4320/**4218**	4	180	[590]
Zbiór 5 BCI	486/**404**	2	140	[591]

Tabela 6.1. Charakterystyka zbiorów danych użytych w badaniu; wartości pogrubione odnoszą się do liczby cech, które pozostały w zbiorach danych po eliminacji cech o zbyt wysokiej korelacji.
Źródło: Opracowanie własne.

6.1.2. Etap 1 - Wpływ parametrów *accWeight* i *fsWeight* na wyniki algorytmu

Jak wskazano w rozdziale V, funkcja optymalizacyjna wykorzystywana w algorytmie GAAMmf bierze pod uwagę dwa kryteria, kryterium dokładności klasyfikacji oraz kryterium liczby cech. Zadaniem parametrów *fsWeight* oraz *accWeight,* analizowanych w niniejszym punkcie, jest regulowanie udziału obu parametrów w ostatecznej wartości funkcji przystosowania. Celem opisywanego eksperymentu było wykazanie, że procedura optymalizacji zastosowana w algorytmie GAAMmf prowadzi do zmniejszania liczby cech przy jednoczesnym zwiększaniu dokładności klasyfikacji przy różnych wartościach obu parametrów.

Do analizy wpływu wartości parametrów *fsWeight* oraz *accWeight* na działanie zaproponowanego algorytmu wykorzystano pierwszy z opisanych w poprzednim podpunkcie zbiorów (Zbiór 1). Na zbiorze tym pięciokrotnie uruchomiono algorytm GAAMmf, za każdym razem podstawiając inne wartości obu parametrów. W pierwszym uruchomieniu oba parametry ustawiono na wartość 1 (*accWeight=1, fsWeight=1)*. W drugim i czwartym uruchomieniu podwojono znaczenie jednego z kryteriów, otrzymując: *accWeight=1, fsWeight=2*, w przypadku drugiego uruchomienia oraz *accWeight=2, fsWeight=1*, w przypadku czwartego uruchomienia. W trzecim i piątym uruchomieniu znaczenie jednego z kryteriów oceny zwiększono czterokrotnie, otrzymując: *accWeight=1, fsWeight=4*, w przypadku trzeciego uruchomienia oraz *accWeight=4, fsWeight=1*, w przypadku piątego uruchomienia. Wartości pozostałych parametrów algorytmu były identyczne dla każdego uruchomienia i kształtowały się następująco:

- liczba osobników w populacji startowej: 10;
- startowa liczba genów: 10;
- prawdopodobieństwo mutacji (*probM*): 1;
- prawdopodobieństwo krzyżowania: 1;
- liczba iteracji: 1000.

Tabela 6.2 oraz Rysunek 6.1 przedstawiają wyniki uzyskane w trakcie każdego z pięciu uruchomień algorytmu. Wykresy przedstawione na Rysunku 6.1 pokazują dokładność klasyfikacji (wykresy po lewej stronie) oraz liczbę cech (wykresy po prawej stronie) zwrócone przez algorytm w kolejnych iteracjach. Tabela prezentuje te same dane, ale w bardziej zwartej postaci. Dla każdej liczby cech osiągniętych przez algorytm pokazany jest tutaj osobnik o najwyższej dokładności. Dodatkowo, pogrubioną czcionką oznaczono dokładność i liczbę cech osobników zwróconych w ostatniej iteracji algorytmu jako ostateczny wynik jego działania.

Jak można zaobserwować w Tabeli 6.2 najwyższe dokładności klasyfikacji algorytm osiągnął przy podwojonej wadze dokładności klasyfikacji *accWeight=2* (*fsWeight=1*). Dokładność ta wynosiła ponad 97.40 % zarówno dla 10, jak i 9 oraz 8 cech. Podobny poziom dokładności algorytm osiągnął przy zastosowaniu czterokrotnie większej wagi dokładności klasyfikacji *accWeight=4* w porównaniu do wagi liczby cech *fsWeight=1*. Dokładność ta wynosiła ponad 97.52 % zarówno dla liczby cech równej 10, jak i 9. Zbyt duży nacisk na dokładność klasyfikacji nie pozwolił jednak na dodatkowe

zmniejszenie liczby cech do 8, jak to miało miejsce w przypadku podwojonej wagi kryterium dokładności.

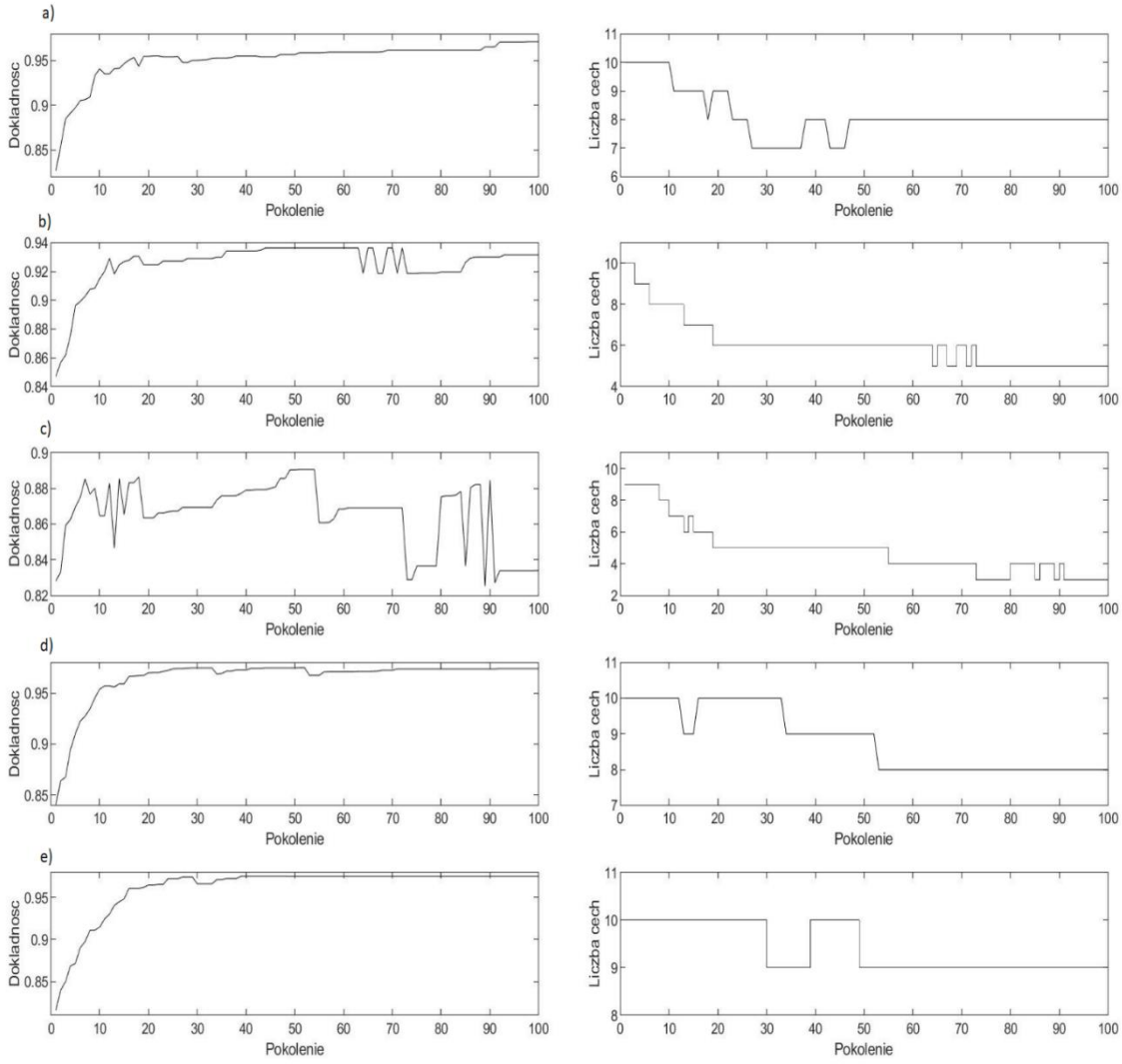
accWeight=1, fsWeight=1		***accWeight=1, fsWeight=2***		***accWeight=1, fsWeight=4***		***accWeight=2, fsWeight=1***		***accWeight=4, fsWeight=1***	
Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech
				83.65%	**3**				
				88.46%	4				
		93.17%	**5**	89.06%	5				
		93.65%	6	88.64%	6				
95.42%	7	93.07%	7	88.54%	7				
97.13%	8	92.92%	8	88.01%	8	**97.40%**	**8**		
95.53%	9	89.63%	9	88.53%	9	97.49%	9	**97.52%**	**9**
94.08%	10	85.68%	10			97.47%	10	97.54%	10

Tabela 6.2. Najwyższa dokładność osiągnięta dla różnej liczby cech dla zbioru 1. Wyniki pochodzą z pięciu uruchomień algorytmu GAAMmf przeprowadzonych z różnymi wartościami parametrów *accWeight i fsWeight*.
Źródło: Opracowanie własne.

Jeżeli chodzi o liczbę cech, to największa redukcja miała miejsce, tak jak można się było spodziewać, dla najwyższej wartości parametru *fsWeight (fsWeight=4, accWeight=1*). Algorytm osiągnął tutaj osobnika składającego się jedynie z 3 genów (3 cech). Niestety, dokładność klasyfikacji wyznaczona na podstawie cech zakodowanych w tym osobniku była o 14% mniejsza niż dla wcześniej omówionych uruchomień. Trochę niższą redukcję cech, za po przy znacznie wyższej dokładności, algorytm osiągnął przy podwojonej wadze kryterium liczby cech *(fsWeight=2, accWeight=1*). Liczbę cech udało się tu zredukować do 5 cech przy dokładności wynoszącej 93.17%, a więc tylko o 4% mniejszej niż najwyższa wartość jaką udało się osiągnąć we wszystkich uruchomieniach wynosząca 97.52%. Wartość ta została osiągnięta dla uruchomienia piątego (*accWeight=4, fsWeight=1*) dla prawie dwukrotnie większej liczby cech (9 cech). Wreszcie, w przypadku uruchomienia pierwszego, w którym założono równy udział obu kryteriów w funkcji przystosowania *(accWeight=1, fsWeight=1*), algorytm zredukował liczbę cech do 7 przy dokładności klasyfikacji równej 95.42%, czyli uzyskał wyniki pośredni.

Z porównania wyników zawartych w Tabeli 6.2 można wyciągnąć wniosek, że algorytm zachowuje się prawidłowo w każdym z zaprezentowanych przypadków, tzn. dla każdego z przedstawionych zestawów parametrów dąży do zmniejszenia liczby

cech przy jednoczesnym zwiększeniu dokładności klasyfikacji. Dodatkowo, w przypadku zwiększenia wagi kryterium dokładności algorytm zwraca rozwiązania o nieco wyższej liczbie cech, ale jednocześnie wyższej dokładności, natomiast w przypadku zwiększenia wagi kryterium liczby cech zwraca rozwiązania o nieco niższej dokładności, ale również niższej liczbie cech.



Rysunek 6.1. Wykresy przedstawiające wyniki GAAMmf dla zbioru 1; wykresy po lewej stronie prezentują dokładność najlepszego osobnika zwróconego w każdej iteracji, natomiast wykresy po prawej stronie przedstawiają liczbę cech zakodowanych w tym osobniku; wiersze wykresów prezentują wyniki dla różnych wartości parametrów *accWeight i fsWeight*: a) *accWeight=1, fsWeight=1*; b) *accWeight=1, fsWeight=2*; c) *accWeight=1, fsWeight=4*; d) *accWeight=2, fsWeight=1*; e) *accWeight=4, fsWeight=1*. Źródło: Opracowanie własne.

Ponadto, jak ilustruje Rysunek 6.1 dla prawie wszystkich badanych zestawów parametrów (poza zestawem *accWeight=1, fsWeight=4*) algorytm zachowywał właściwy kierunek zbieżności dla obu badanych kryteriów, tj. monotoniczny wzrost z drobnymi fluktuacjami dla kryterium dokładności oraz monotoniczny spadek dla kryterium liczby cech. Kierunek ten nie został zachowany w odniesieniu do kryterium dokładności w uruchomieniu trzecim, co wynikało z bardzo dużej wagi przypisanej kryterium liczby cech. Oczywiście tempo zbieżności algorytmu różniło się w zależności od uruchomienia i było zależne od wartości parametrów. Wyższe wartości *fsWeight* skutkowały szybszym zmniejszaniem liczby cech, ale z pewnymi fluktuacjami dokładności (Rysunek 6.1b oraz Rysunek 6.1c). Z kolei dla wyższych wartości *accWeight* (Rysunek 6.1d Rysunek 6.1e) algorytm wykazywał wysoką stabilność pod kątem zbieżności do jak najwyższej dokładności, choć z wolniejszym tempem redukcji cech.

Z przedstawionych badań wynika, że o ile nie ma ku temu ważnych przesłanek, proponowany algorytm należy stosować z jednakowymi wagami przypisanymi do obu kryteriów optymalizacyjnych *(accWeight=1, fsWeight=1*). Taki zestaw parametrów zostanie więc przyjęty we wszystkich kolejnych eksperymentach.

6.1.3. Etap 2 - Wpływ prawdopodobieństwa mutacji na wyniki algorytmu

W drugim etapie eksperymentu zbadano wpływ wartości parametru określającego prawdopodobieństwo mutacji (*probM*) na wyniki zwracane przez algorytm GAAMmf. W celu wykonania tego zadania trzykrotnie uruchomiono algorytm dla trzech różnych poziomów parametru *probM,* równych: 0.1, 0.5 oraz 1. Z uwagi na to, że wartość parametru *probM* silnie wpływa na liczbę osobników ocenianych w każdej iteracji algorytmu, wraz ze zwiększaniem wartości parametru *probM* zmniejszano liczbę iteracji algorytmu. W ten sposób zapewniono porównywalną sumaryczną liczbę osobników ocenianych w kolejnych uruchomieniach algorytmu. Dla pierwszego uruchomienia algorytmu z prawdopodobieństwem mutacji równym 0.1 przeprowadzono 400 iteracji, dla drugiego uruchomienia z prawdopodobieństwem mutacji równym 0.5 wykonano 172 iteracji, natomiast dla trzeciego uruchomienia z prawdopodobieństwem mutacji równym 1.0 wykonano 100 iteracji. W Tabeli 6.3 pokazano szacunkową liczbę osobników, która zostałaby oceniona w każdym z trzech rozpatrywanych przypadków przy założeniu, że liczba osobników w każdej populacji rodzicielskiej będzie stała

(równa 10). Oczywiście, faktyczne liczby osobników ocenionych w trakcie każdego z trzech przebiegów algorytmu różniły się nieznacznie od wartości podanych w tabeli, ponieważ liczba osobników generowanych w kolejnych iteracjach algorytmu jest zależna od liczby genów w każdym z mutowanych osobników.

Prawdopodobieństwo mutacji	Liczba iteracji	Szacunkowa liczba ocenianych osobników *(S+K+(S*M*probM)*N*
0.1	4000	(10+10+(10*10*0.1))*4000 = 120 000
0.5	1720	(10+10+(10*10*0.5))*1720 = 120 040
1.0	1000	(10+10+(10*10*1))*1000 = 120 000

Tabela 6.3. Liczba iteracji dla trzech uruchomień algorytmu dla różnego parametru prawdopodobieństwa mutacji; *S* – liczba osobników w populacji startowej, *K* – liczba osobników krzyżowanych, *M* – liczba mutowanych genów, *probM* – prawdopodobieństwo mutacji, *N* – liczba iteracji.
Źródło: Opracowanie własne.

Wartości pozostałych parametrów algorytmu były identyczne dla każdego uruchomienia i kształtowały się następująco:

- liczba osobników w populacji startowej: 10;
- startowa liczba genów: 10;
- prawdopodobieństwo krzyżowania: 1;
- *accWeight=1*;
- *fsWeight=1*.

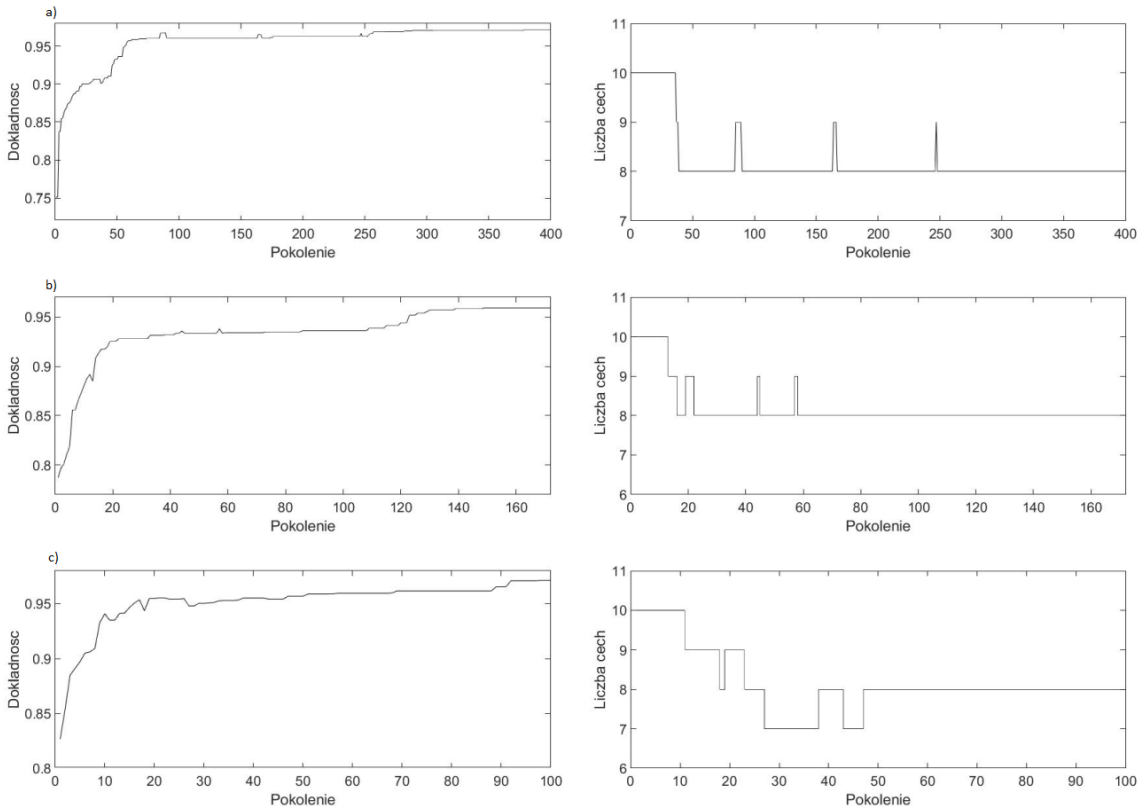
W Tabeli 6.4 zaprezentowano wyniki zwrócone przez algorytm w kolejnych uruchomieniach. Podobnie jak w Tabeli 6.2 dla każdej liczby cech osiągniętych przez algorytm przedstawiono osobnika o najwyższej dokładności klasyfikacji, pogrubioną czcionką oznaczając dokładność i liczbę cech osobników zwróconych w ostatniej iteracji algorytmu. Dodatkowo, na Rysunku 6.2 zilustrowano dokładność klasyfikacji oraz liczbę cech zwracane przez algorytm w każdej kolejnej iteracji.

Jak można zaobserwować w Tabeli 6.4 dla wszystkich trzech wartości parametru *probM* uzyskano taką samą redukcję cech w ostatnim pokoleniu, wynoszącą 8 cech. Pomimo, że przy *probM=1* algorytm w trakcie swojego działania zmniejszył liczbę cech do 7, to rozwiązanie to nie przetrwało do końca algorytmu z uwagi na niższą (o prawie

2%) dokładność klasyfikacji w porównaniu do zbioru o 8 cechach. Jeżeli chodzi o dokładność klasyfikacji uzyskaną dla zbiorów cech zwróconych w ostatniej iteracji algorytmu, to ona również była podobna przy wszystkich ustawieniach parametru *probM*, chociaż w przypadku *probM=0.5* była o około 1% niższa niż dla pozostałych ustawień.

***probM* = 0.1**		***probM* = 0.5**		***probM* = 1.0**	
Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech
				95.42%	7
97.16%	**8**	**95.91%**	**8**	**97.13%**	**8**
96.79%	9	93.79%	9	95.53%	9
90.64%	10	89.17%	10	94.08%	10

Tabela 6.4. Najwyższa dokładność klasyfikacji osiągnięta w trakcie działania algorytmu dla różnej liczby cech przy każdym z badanych współczynników mutacji.
Źródło: Opracowanie własne.



Rysunek 6.2. Wykresy przedstawiające wyniki GAAMmf dla zbioru 1; wykresy po lewej stronie prezentują dokładność najlepszego osobnika zwróconego w każdej iteracji, wykresy po prawej stronie przedstawiają liczbę cech zakodowanych w tym osobniku; wiersze wykresów prezentują wyniki dla różnych wartości parametru *probM*: a) *probM = 0.1;* b) *probM = 0.5;* c) *probM = 1*.
Źródło: Opracowanie własne.

Wykresy przedstawione na Rysunku 6.2 pokazują dodatkowo, że proces optymalizacyjny we wszystkich trzech przypadkach przebiegał prawidłowo, wykazując wysoce monotoniczny wzrost dokładności klasyfikacji oraz trochę bardziej fluktuujący, spadek liczby cech. Dodatkowo, wykresy ilustrujące wartości obu kryteriów optymalizacyjnych są w dużym stopniu podobne dla wszystkich trzech przebiegów algorytmu.

Porównanie przedstawione w Tabeli 6.4 pokazuje, że każda przyjęta wartość prawdopodobieństwa mutacji pozwoliła na osiągnięcie satysfakcjonującej liczby cech oraz dokładności klasyfikacji. Porównanie to pokazuje również, że najlepsze wyniki są osiągane w przypadku korzystania z pełnego schematu mutacji, zakładającego mutację każdego genu każdego osobnika. Z tego wynika, że jeżeli jest to tylko możliwe, należy stosować algorytm GAAMmf z parametrem *probM* ustawionym na wartość 1.

6.1.4. Etap 3 - Badanie stabilności algorytmu

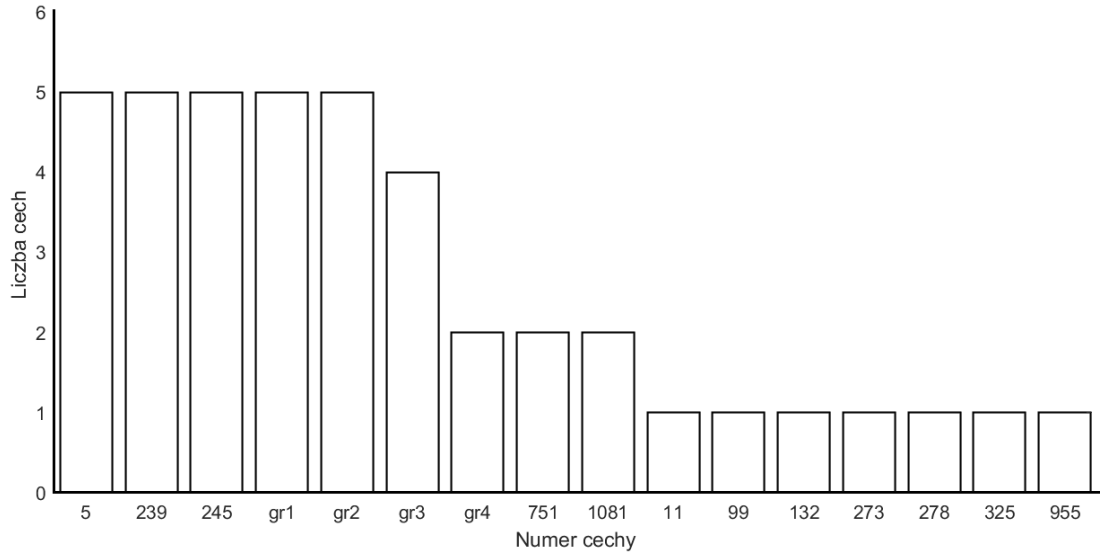
W trzecim etapie eksperymentu przeprowadzono badanie stabilności algorytmu. W tym celu wykonano pięć uruchomień algorytmu z takimi samymi parametrami: *accWeight=1*, *fsWeight=1*, *probM=1* oraz *liczba_iteracji=1000*. Wyniki algorytmu przedstawiono w Tabelach 6.5-6.6 oraz na Rysunku 6.3. Tabela 6.5 pokazuje liczbę cech oraz dokładność klasyfikacji osobnika o najwyższej wartości funkcji przystosowania, zwróconego jako wynik działania każdego z pięciu uruchomień algorytmu. Z kolei Tabela 6.6 przedstawia opis cech zakodowanych w osobnikach wynikowych.

Jak można zaobserwować w Tabeli 6.5, w każdym z pięciu uruchomień algorytm osiągnął porównywalne rezultaty, osiągając redukcję do 8 lub 9 cech przy zachowaniu dokładności powyżej 96%. Dla 8 cech uzyskano dokładność klasyfikacji w przedziale od 96.30% do 97.51%, a dla 9 cech wyniki oscylowały w przedziale od 96.57% do 97.90%.

Poza wysoką stabilnością w odniesieniu do zwracanych wyników liczbowych określających dokładność klasyfikacji oraz liczbę cech, algorytm wykazał również stabilne działanie w aspekcie jakościowym, co przejawiało się wysokim podobieństwem zbiorów cech wybieranych w kolejnych uruchomieniach. Jak można zaobserwować w Tabeli 6.6 oraz na Rysunku 6.3 prawie połowa zbioru cech była identyczna we wszystkich pięciu uruchomieniach - cechy o numerach 5, 239 oraz 245 powtórzyły się w każdym zbiorze. Cechy te mają również sensowną interpretację w odniesieniu

do wyobrażenia ruchu, ponieważ wszystkie trzy reprezentują moc sygnału w paśmie alfa obserwowaną nad elektrodami centralnymi:

- cecha nr 5 – moc sygnału zarejestrowanego w sekundzie 1 nad elektrodą C3 wyznaczoną w paśmie 8-13 Hz;
- cecha nr 239 – moc sygnału zarejestrowanego w sekundzie 2 nad elektrodą C3 wyznaczoną w paśmie 8-13 Hz;
- cecha nr 245 – moc sygnału zarejestrowanego w sekundzie 2 nad elektrodą CP3 wyznaczoną w paśmie 8-13 Hz.



Rysunek 6.3. Badanie stabilności algorytmu podczas 5 uruchomień; gr1 – cechy 1005, 1006; gr2 – cechy 795, 799, 800, 805; gr3 – cechy 332, 333, 335; gr4 – cechy 69, 72. Źródło: Opracowanie własne.

Pozostałe cechy zawarte w zbiorach cech zwracanych w kolejnych uruchomieniach nie były już identyczne, ale nadal większość z nich była w dużym stopniu podobna. Pierwsza grupa cech podobnych zwróconych przez algorytm we wszystkich uruchomieniach, to cechy o numerach 1005 i 1006. Cechy te reprezentują moc sygnału w paśmie 12-13 Hz wyznaczoną w 5 sekundzie z elektrody C5 (cecha 1005) oraz C3 (cecha 1006).

Druga grupa cech podobnych, występujących we wszystkich uruchomieniach algorytmu, to cechy o numerach 795, 799, 800 i 805. Cechy te zostały wyznaczone z sygnału zarejestrowanego w 4 sekundzie eksperymentu i reprezentują moc sygnału wyznaczoną dla pasma 13-17 Hz. Cechy z tej grupy różniły się od siebie jedynie

elektrodą, na której został zarejestrowany sygnał; cecha 795 została wyznaczona z sygnału zarejestrowanego na FCz, cecha 799 na elektrodzie C1, cecha 800 – na elektrodzie Cz, a cecha 805 - na elektrodzie CPz.

Wreszcie trzecia z grup cech o wysokim podobieństwie to grupa obejmująca cechy o numerach 332, 333 i 335, reprezentujące moc w paśmie 13-17 Hz wyznaczoną dla 2 sekundy. Cechy z tej grupy ponownie różniły się od siebie tylko położeniem elektrody rejestrującej sygnał EEG; cecha 332 została wyznaczona z sygnału zarejestrowanego na elektrodzie Cz, cecha 333 - na elektrodzie C2, a cecha 335 - na elektrodzie C6. Cechy z tej grupy wystąpiły w 4 uruchomieniach algorytmu.

Spośród pozostałych cech, cechy o numerach 751 (4 sekunda, pasmo 10-11 Hz, elektroda C6) i 1081 (5 sekunda, pasmo 26-30 Hz, elektroda FCz) wystąpiły w dwóch zbiorach cech. Również cechy o numerach 69 i 72 różniące się jedynie położeniem elektrod rejestrujących sygnał (cecha 69 - elektroda C5, cecha 72 - elektroda Cz) reprezentujące moc sygnału w paśmie 12-13 Hz wyznaczoną z pierwszej sekundy eksperymentu wystąpiły w dwóch zbiorach cech.

Podsumowując trzeci etap eksperymentu pierwszego należy zauważyć, że w wyniku pięciu uruchomień algorytmu zaobserwowano, że na 8-9 cech zawartych w zbiorach cech zwracanych w kolejnych uruchomieniach:

- trzy cechy były identyczne w każdym uruchomieniu;
- trzy cechy reprezentowały po jednej z 3 grup cech o wysokim stopniu podobieństwa i występowały we wszystkich (gr1 i gr2) lub prawie wszystkich (gr3) uruchomieniach;
- dwie – trzy pozostałe cechy powtarzały się w kilku uruchomieniach lub były unikatowe.

Z przeprowadzonego eksperymentu wynika, że algorytm wykazał wysoką powtarzalność zwracanych wyników, zarówno w odniesieniu do dokładności klasyfikacji, jak i rozmiaru zwracanego zbioru cech oraz wybieranych cech. Należy tu podkreślić, że na 42 cechy zwrócone w kolejnych uruchomieniach, jedynie 7 cech było unikatowych. Ta obserwacja sugeruje, że algorytm jest stabilny i konsekwentny w identyfikowaniu powtarzających się cech, co sugeruje wysoką niezawodność i powtarzalność wyników.

Uruchomienie 1		Uruchomienie 2		Uruchomienie 3		Uruchomienie 4		Uruchomienie 5	
Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech
97.13%	8	97.51%	8	97.90%	9	96.30%	8	96.57%	9

Tabela 6.5. Najwyższa osiągnięta dokładność dla różnej liczby cech dla zbioru 1. Wyniki pochodzą z pięciu uruchomień algorytmu GAAMmf przeprowadzonych z wartościami parametrów: *accWeight=1, fsWeight=1*, probM=1, *probM=1* oraz *liczba_iteracji=100*.

Źródło: Opracowanie własne.

Uruchomienie 1				Uruchomienie 2				Uruchomienie 3				Uruchomienie 4				Uruchomienie 5			
Numer cechy	Pasmo	Sek.	Elektroda	Numer cechy	Pasmo	Sek.	Elektroda	Numer cechy	Pasmo	Sek.	Elektroda	Numer cechy	Pasmo	Sek.	Elektroda	Numer cechy	Pasmo	Sek.	Elektroda
5	8-13	1	C3	5	8-13	1	C3	5	8-13	1	C3	5	8-13	1	C3	5	8-13	1	C3
239	8-13	2	C3	**11**	**8-13**	**1**	**CP3**	69	12-13	1	C5	**99**	**13-17**	**1**	**C2**	72	12-13	1	Cz
245	8-13	2	CP3	239	8-13	2	C3	239	8-13	2	C3	**132**	**23-26**	**1**	**FCz**	239	8-13	2	C3
333	13-17	2	C2	245	8-13	2	CP3	245	8-13	2	CP3	239	8-13	2	C3	245	8-13	2	CP3
751	10-11	4	C6	**273**	**9-10**	**2**	**CP4**	333	13-17	2	C2	245	8-13	2	CP3	**278**	**10-11**	**2**	**C3**
799	13-17	4	C1	332	13-17	2	Cz	751	10-11	4	C6	335	13-17	2	C6	**325**	**13-30**	**2**	**CP4**
1005	12-13	5	C5	799	13-17	4	C1	800	13-17	4	Cz	795	13-17	4	FCz	805	13-17	4	CPz
1081	26-30	5	FCz	1005	12-13	5	C5	1005	12-13	5	C5	1006	12-13	5	C3	**955**	**8-9**	**5**	**C1**
								1081	26-30	5	FCz					1005	12-13	5	C5

Tabela 6.6 Uzyskane cechy dla pięciu uruchomień algorytmu. Dla każdego uruchomienia ustawiono parametry: *accWeight=1, fsWeight=1*, *probM=1* oraz *liczba_iteracji=100*.

Źródło: Opracowanie własne.

6.1.5. Etap 4 - Zestawienie wyników algorytmu dla 5 zbiorów BCI

W etapie 4 eksperymentu pierwszego przeprowadzono ponownie badania z etapu 1 (badanie wpływu parametrów *accWeight* i *fsWeight* na wyniki algorytmu GAAMmf) oraz etapu 2 (badanie wpływu parametr *probM* na wyniki algorytmu GAAMmf) na wszystkich zbiorach BCI. Dla każdego zbioru algorytm został uruchomiony 15-krotnie, przy czym w każdym przypadku zastosowano różne wartości parametrów *accWeight, fsWeight* oraz *probM*. Pozostałe parametry algorytmu zostały ustawione na następujących poziomach:

- liczba osobników w populacji startowej: 10;
- startowa liczba genów: 10;
- prawdopodobieństwo krzyżowania: 1;
- liczba iteracji: 4000 dla *probM=0.1*; 1720 dla *probM=0.5*; 1000 dla *probM=1.0.*

W Tabeli 6.7 oraz na Rysunku 6.4 przedstawiono porównanie wyników zwróconych przez algorytm dla każdego zbioru w każdym uruchomieniu. Porównanie zostało wykonane, tak jak w poprzednich etapach, zarówno w kontekście dokładności klasyfikacji, jak i liczby cech zawartych w ostatecznym rozwiązaniu. Jak można zaobserwować w Tabeli 6.7 oraz na Rysunku 6.4 dla *probM*=0.1 najwyższe dokładności klasyfikacji (uśrednione po 5 zbiorach) algorytm osiągnął przy jednostkowej *accWeight=1,* podwojonej *accWeight=2* oraz czterokrotnie zwiększonej *accWeight=4* wadze dokładności. Średnia dokładność uzyskana dla tych trzech przypadków wynosiła ponad 84.5% przy średnio 9 cechach. Z kolei przy zastosowaniu podwojonej wagi liczby cech *fsWeight=2*, średnia dokładność spadła do 82.72%, ale algorytm zredukował liczbę cech do 7, a więc uzyskał zbiór zawierający o 2 cechy (28%) mniej niż poprzednio. Przy zastosowaniu czterokrotnie większej wagi dla liczby cech *fsWeight=4*, średnia dokładność spadła do 77.95% przy średnio 5 cechach (czyli przy o 57% mniejszym zbiorze cech). Zbyt duży nacisk na liczbę cech spowodowało więc znaczną redukcję zbiorów cech (średnio z 9 do 5), niestety kosztem znacznego spadku dokładności (z około 84% do około 78%)

Dla *probM=0.5* najwyższe średnie dokładności klasyfikacji algorytm również osiągnął przy pojedynczej *accWeight=1,* podwojonej *accWeight=2* oraz czterokrotnie zwiększonej *accWeight=4* wadze dokładności. Średnia dokładność uzyskana dla tych trzech przypadków była bardzo podobna i wynosiła od 85.95% do 86.36% przy średnio 9 cechach. Z kolei przy zastosowaniu podwojonej wagi liczby cech *fsWeight=2*,

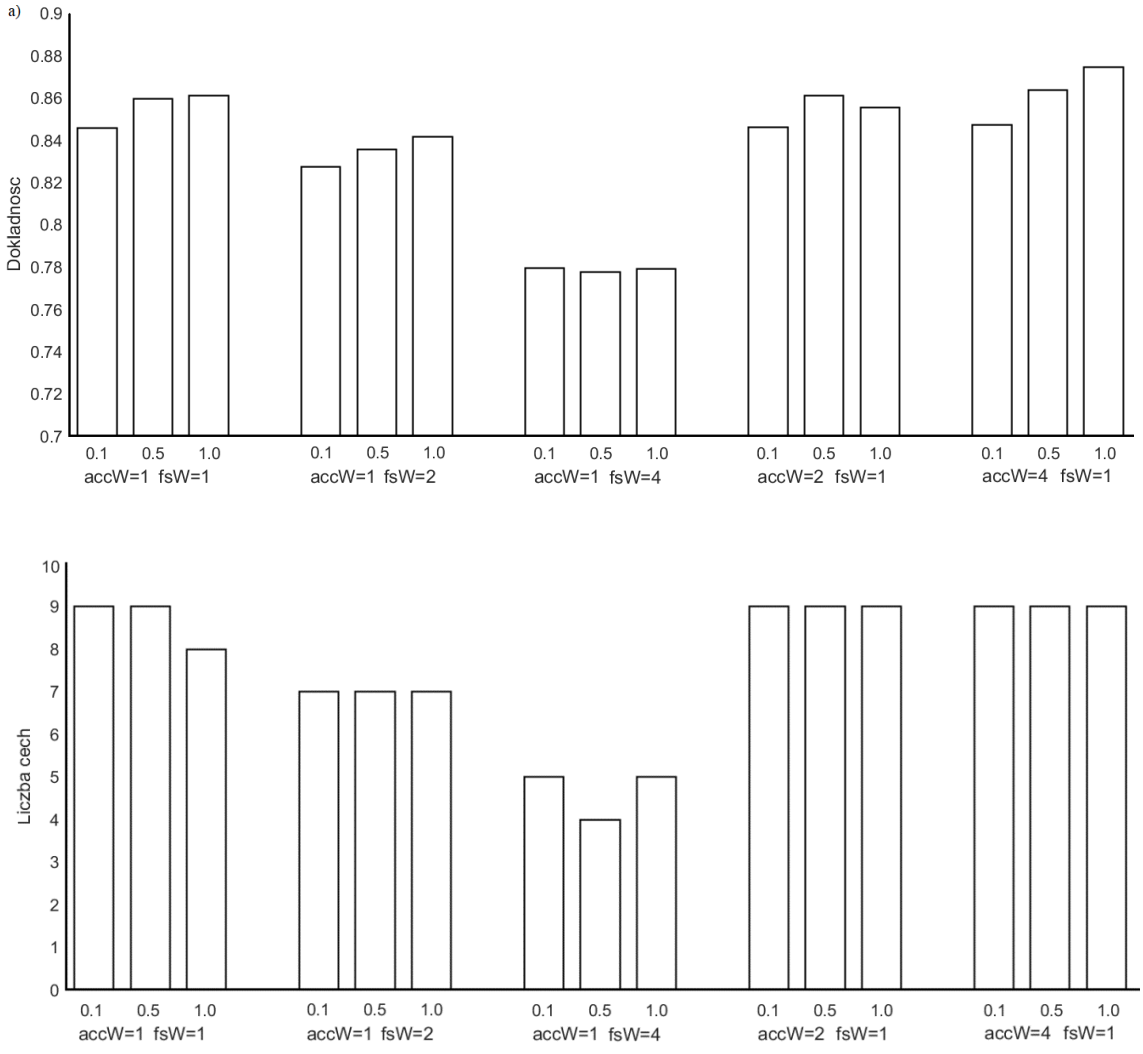
dokładność spadła do 83.54%, ale zbiór cech zmniejszył się do 7 cech. Przy zastosowaniu czterokrotnie większej wagi liczby cech *fsWeight=4*, średnia dokładność spadła do 77.75% przy 4 cechach (czy przy o 71% mniejszym zbiorze cech).

Prawdopodobieństwo mutacji=0.1										
	accWeight=1, fsWeight=1		*accWeight=1, fsWeight=2*		*accWeight=1, fsWeight=4*		*accWeight=2, fsWeight=1*		*accWeight=4, fsWeight=1*	
Zbiór	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech	Dokładność	Liczba cech
1	94.44%	7	93.97%	7	86.44%	4	95.96%	9	95.85%	9
2	86.30%	9	80.22%	6	77.86%	4	84.25%	8	84.87%	8
3	74.21%	9	72.09%	8	64.70%	5	71.07%	9	71.71%	10
4	73.52%	10	74.88%	9	69.85%	7	77.38%	10	76.02%	10
5	94.29%	10	92.42%	7	90.91%	6	94.29%	10	95.16%	9
Średnia	**84,55%**	**9**	**82,72%**	**7**	**77,95%**	**5**	**84,59%**	**9**	**84,72%**	**9**
Prawdopodobieństwo mutacji=0.5										
1	95.98%	8	94.26%	6	86.51%	3	95.99%	8	96.85%	9
2	85.42%	8	82.15%	6	79.97%	4	84.89%	6	87.17%	8
3	73.44%	10	71.65%	9	63.04%	4	74.62%	10	73.80%	10
4	78.83%	10	78.21%	9	70.56%	7	79.17%	10	79.85%	10
5	96.07%	9	91.43%	5	88.65%	4	95.71%	9	94.13%	10
Średnia	**85,95%**	**9**	**83,54%**	**7**	**77,75%**	**4**	**86,08%**	**9**	**86,36%**	**9**
Prawdopodobieństwo mutacji=1.0										
1	97.13%	8	93.17%	5	83.65%	3	97.40%	8	97.52%	9
2	86.01%	8	84.17%	5	82.05%	5	86.60%	9	87.21%	9
3	75.18%	10	72.97%	8	62.87%	4	72.24%	10	77.80%	10
4	77.96%	10	78.09%	10	73.89%	7	76.54%	10	79.20%	10
5	94.13%	8	92.38%	7	87.10%	4	94.92%	9	95.48%	9
Średnia	**86,08%**	**9**	**84,16%**	**7**	**77,91%**	**5**	**85,54%**	**9**	**87,44%**	**9**

Tabela 6.7. Wyniki algorytmu GAAMmf dla 5 zbiorów BCI. W tabeli przedstawiono dokładność klasyfikacji oraz liczbę cech zwrócone na zakończenie każdego z 15 uruchomień algorytmu przy różnych wartościach parametrów *probM*, *accWeight* oraz *fsWeight*.

Źródło: Opracowanie własne.

Dla *probM*=1.0 najwyższą średnią dokładność klasyfikacji, wynoszącą 87.44%, algorytm osiągnął przy poczwórnej *accWeight=4* wadze dokładności (przy średnio 9 cechach). Trochę niższa dokładność (86.08%) została uzyskana przy pojedynczej wadze dokładności i liczby cech (*accWeight=1 fsWeight=1*), także przy 9 cechach. Dla podwojonej *accWeight=2* wagi dokładności średnia dokładność klasyfikacji wynosiła 85.54%, również dla 9 cech. Z kolei przy zastosowaniu podwojonej wagi liczby cech *fsWeight=2*, dokładność spadła do 84.16%, ale algorytm zwrócił o 2 cechy mniej (7 cech). Przy zastosowaniu poczwórnej wagi liczby cech *fsWeight=4*, średnia dokładność spadła do 77.91%, przy jednoczesnym spadku liczby cech z 9 do 5.



Rysunek 6.4. Wyniki zwrócone przez algorytm GAAMmf przy różnej wartości parametrów *probM, accWeight* oraz *fsWeight* uśrednione po 5 zbiorach BCI; a) średnia dokładność klasyfikacji; b) średnia liczba cech.

Źródło: Opracowanie własne

Bardziej globalnej analizy średnich wyników algorytmu można dokonać obserwując wykresy przedstawione na Rysunku 6.4. Na rysunku tym widać wyraźnie, że podobnie jak zaobserwowano w eksperymencie 1 podczas analizy pierwszego zbioru, wartość parametru *probM* nie ma dużego wpływu na wyniki osiągane przez algorytm. Zarówno dokładność klasyfikacji, jak i liczba cech zwracanych przez algorytm są na bardzo podobnych poziomach, bez względu na konkretną wartość tego parametru (chociaż nieco wyższa dokładność jest uzyskiwana przy zastosowaniu pełnego schematu agresywnej mutacji, czyli przy *probM=1*). Z drugiej strony, wartości obu parametrów wagowych *accWeight* i *fsWeight* mają, jak można się było spodziewać, duży wpływ na ostateczny wynik zwracany przez algorytm. Zwiększenie wartości parametru *accWeight* w stosunku do parametru *fsWeight* powoduje otrzymanie zbioru cech pozwalającego na uzyskanie wyższej dokładności klasyfikacji (ale mogącego zawierać więcej cech), z kolei zwiększenie wartości parametru *fsWeight* w stosunku do parametru *accWeight* powoduje zwrócenie zbioru cech o niższej wielkości, ale przy tym również o niższej dokładności.

6.2. Eksperyment 2 - porównanie algorytmu GAAMmf z algorytmami referencyjnymi na 5 zbiorach BCI

Celem eksperymentu drugiego było porównanie skuteczności algorytmu GAAMmf na tle innych metod selekcji cech. W charakterze metod referencyjnych wykorzystano 8 metod, wśród których znalazły się cztery algorytmy genetyczne (algorytm GAAM, algorytm Melting GAAM, algorytm NSGA II, algorytm Hollanda z funkcją kary) oraz cztery metody klasyczne (metoda Lasso, metoda CFS, metoda ReliefF oraz metoda SFS – metoda selekcji w przód). Wszystkie metody zostały opisane w teoretycznej części pracy, metody należące do grupy algorytmów genetycznych – w rozdziale V, natomiast metody klasyczne – w rozdziale III. Porównanie wykonano na 5 zbiorach pochodzących z konkursów BCI, omówionych podczas opisu eksperymentu 1. Wartości parametrów algorytmu GAAMmf oraz poszczególnych metod referencyjnych zostały ustawione dla wszystkich zbiorów na takich samych poziomach, kształtujących się następująco:

- GAAMmf – *liczba pokoleń*: 1000; *ProbM*: 1; *M* (liczba osobników w populacji startowej): 10; *N* (startowa liczba genów): 20; *accWeight*: 1; *fsWeight*: 2.
- GAAM – *liczba pokoleń*: 1000; *ProbM*: 1; *M:* 10; *N*: 20.

- Melting – *liczba pokoleń*: 1000; *ProbM*: 1; *M*: 10; *N*: 20; *zakładana_dokładność*: 90%.
- NSGA II – *liczba pokoleń*: 1000; *ProbM*: 1; *M*: 10; *N*: 20.
- Holland – *liczba pokoleń*: 1000; *ProbM*: 1; *M*: 10; *N*: 20.
- Lasso – *W* (wymagana liczba cech): 20.
- CFS – *W*: 20.
- ReliefF – *W:* 20.
- SFS – brak.

Wyniki uzyskane dla każdej z 9 metod oraz każdego z 5 zbiorów zostały przedstawione w Tabeli 6.8. Dodatkowo na Rysunku 6.5 przedstawiono średnią dokładność klasyfikacji oraz liczbę cech (uśrednioną po 5 zbiorach), natomiast na Rysunku 6.6 - średnią dokładność klasyfikacji oraz liczbę cech (uśrednioną po 9 algorytmach).

Dokładność klasyfikacji [%]										
	GAAMmf	GAAM	Melting	NSGA II	Holland	Lasso	CFS	ReliefF	SFS	**Średnia**
Zbiór 1	97.13	98.52	89.00	-	-	93.50	74.90	80.40	97.00	**90.06**
Zbiór 2	86.01	91.30	89.05	-	-	80.20	72.00	65.70	87.26	**81.65**
Zbiór 3	75.18	79.98	76.06	-	-	73.00	51.40	51.80	67.95	**67.91**
Zbiór 4	85.19	92.28	89.85	-	-	62.78	39.40	27.80	76.67	**67.71**
Zbiór 5	94.13	96.75	89.48	-	-	91.43	64.30	55.00	92.86	**83.42**
Średnia	**87.53**	**91.77**	**86.69**	**-**	**-**	**80.18**	**60.40**	**56.14**	**84.35**	
Liczba cech										
	GAAMmf	GAAM	Melting	NSGA II	Holland	Lasso	CFS	ReliefF	SFS	**Średnia**
Zbiór 1	8	20	2	-	-	12	20	20	10	**13**
Zbiór 2	8	20	13	-	-	10	20	20	12	**15**
Zbiór 3	10	18	20	-	-	19	20	20	9	**17**
Zbiór 4	15	20	20	-	-	12	20	20	12	**17**
Zbiór 5	8	20	4	-	-	12	20	20	8	**13**
Średnia	**10**	**20**	**12**	**-**	**-**	**13**	**20**	**20**	**10**	

Tabela 6.8. Porównanie wyników zwróconych przez algorytm GAAMmf z wynikami 8 metod referencyjnych dla 5 zbiorów BCI.

Źródło: Opracowanie własne.

Przyglądając się Tabeli 6.8 można zauważyć, że 2 algorytmy stosowane najczęściej w procesie selekcji cech (algorytm Hollanda z członek kary oraz algorytm NSGA II) nie wygenerowały wyniku dla żadnego z analizowanych zbiorów. Przyczynę takiego stanu rzeczy można znaleźć w Tabeli 6.1 przedstawiającej liczbę cech, klas i przykładów zawartych w każdym z analizowanych zbiorów. Jak można zauważyć, każdy ze zbiorów jest opisywany przez bardzo niewielką liczbę przykładów (co jest cechą charakterystyczną zbiorów gromadzonych w czasie sesji z interfejsem mózg-komputer). Dodatkowo liczba ta jest znacznie mniejsza niż wymiar przestrzeni cech. Z uwagi na fakt, że oba wskazane algorytmy wykorzystują binarny sposób kodowania cech, charakteryzujący się tym, że przy losowym doborze cech proces optymalizacyjny rozpoczyna się od osobników zawierających około połowy wszystkich cech, nawet proste algorytmy liniowe nie radzą sobie z doborem parametrów klasyfikatora.

W oparciu o wyniki przedstawione w Tabeli 6.8, można zauważyć, że dla zbioru 1 najwyższą dokładność klasyfikacji osiągnął algorytm GAAM (98.52%). Nieco niższą dokładność uzyskał algorytm GAAMmf (97.13%). Porównując te dwa wyniki można jednak zauważyć, że algorytm GAAM potrzebował aż 20 cech, aby osiągnąć wskazaną dokładność, podczas gdy algorytmowi GAAMmf wystarczyło do tego zaledwie 8 cech, czyli ponad dwa razy mniej. Należy tutaj zwrócić dodatkowo uwagę na algorytm Melting GAAM, który dokonał największej redukcji zbioru cech, osiągając zbiór zawierający jedynie 2 cechy, jednak uzyskana przez niego dokładność była znacznie niższa od dokładności zwróconej przez dwa wskazane powyżej algorytmy i wyniosła jedynie 89%. Metoda SFS zwróciła zbiór cech pozwalający na uzyskanie dokładności (97%) zbliżonej do dokładności algorytmu GAAMmf, ale do osiągnięcia tego efektu potrzebowała o 2 cechy więcej. Pozostałe metody osiągnęły niższą dokładność klasyfikacji niż algorytm GAAMmf, przy jednoczesnym wykorzystaniu większej liczby cech.

Podobna sytuacja występuje w przypadku zbioru 2. Tutaj również algorytm GAAM osiągnął największą dokładność (91.30%), ale ponownie potrzebował do tego 20 cech, podczas gdy algorytm GAAMmf osiągnął dokładność o 5% niższą (86.01%), ale przy użyciu zaledwie 8 cech. Algorytm Melting uzyskał tym razem dokładność (89.05%) nieco wyższą niż algorytm GAAMmf, ale przy o ponad 50% większej liczbie cech (GAAMmf – 8 cech, Melting GAAM - 13 cech). Pozostałe metody uzyskały dokładność klasyfikacji niższą niż GAAMmf i jednocześnie wymagały większej liczby cech.

Dla zbioru 3 algorytmy GAAM oraz Melting GAAM ponownie osiągnęły dokładność wyższą aniżeli algorytm GAAMmf (GAAM – 79.98%, Melting GAAM –

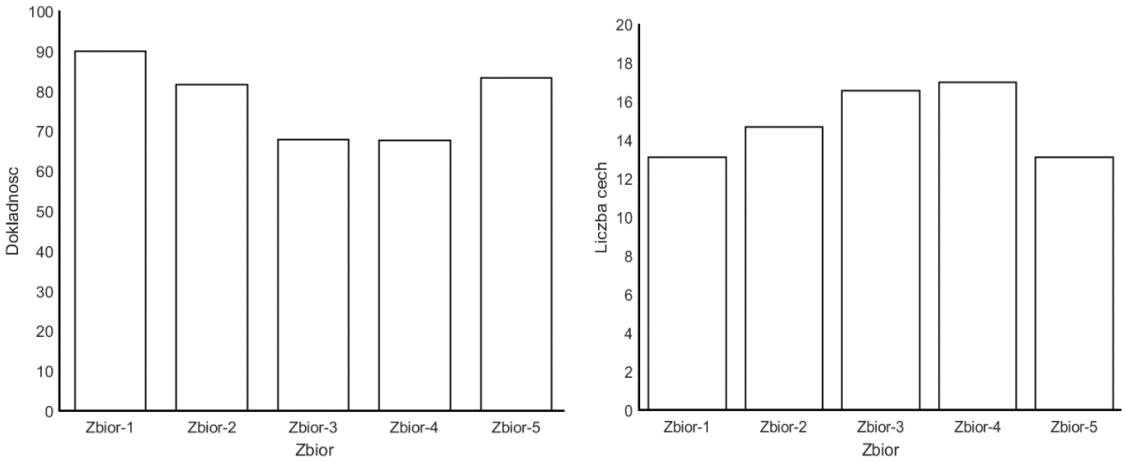
76.06%, GAAMmf – 75.18%). Należy tu jednak zauważyć, że podczas gdy oba wskazane algorytmy referencyjne nie dokonały praktycznie żadnej redukcji cech, GAAMmf zakończył proces poszukiwania rozwiązania na zbiorze składającym się jedynie z 10 cech. Pozostałe algorytmy, tak jak w przypadku poprzednich zbiorów, uzyskały niższą dokładność klasyfikacji niż algorytm GAAMmf i wymagały większej liczby cech (poza metodą SFS, która zakończyła proces selekcji cech na zbiorze składającym się z 9 cech, ale z dokładnością klasyfikacji równą jedynie 67.95%).

Po analizie wyników dla zbioru 4 widać, że algorytmy GAAM oraz Melting GAAM osiągnęły dokładność klasyfikacji wynoszącą odpowiednio 92.28% oraz 89.85%, przy wykorzystaniu aż 20 cech. Algorytm GAAMmf osiągnął niższą dokładność (85.19 %), ale przy mniejszej liczbie cech (15 cech). Metody Lasso i SFS, tym razem dokonały większej redukcji cech niż GAAMmf (12 cech), ale miały znacznie gorszą dokładność klasyfikacji odpowiednio 62.78% oraz 76.67%. Dokładność dwóch pozostałych metod była bardzo niska, a liczba cech oczywiście równa przyjętej (20 cech).

Dla zbioru 5 można zauważyć, że algorytm GAAM uzyskał nieznacznie lepszą dokładność klasyfikacji (96.75%) w porównaniu do algorytmu GAAMmf (94.13%), ale potrzebował do tego ponad dwukrotnie więcej cech (20 cech) niż algorytm GAAMms (8 cech). Metody Lasso i SFS uzyskały nieznacznie gorszą dokładność klasyfikacji (odpowiednio 91.43 % oraz 92.86 %), ale potrzebowały do tego większej liczby cech (odpowiednio 12 cech oraz 8 cech). Algorytm Melting co prawda dokonał największej redukcji zbioru cech, uzyskując 4 cechy, ale jego specyfika nie pozwalała mu uzyskać dokładności klasyfikacji większej niż 90% (z uwagi na wartość parametru *zakladana_dokładność* równą 90%). Pozostałe metody uzyskały dokładność klasyfikacji niższą niż algorytm GAAMmf przy większej liczbie cech.

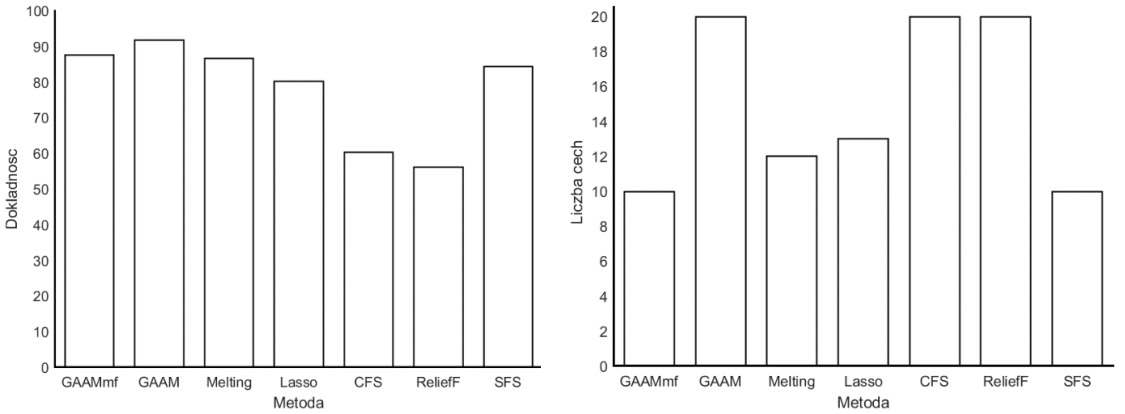
Powyżej przedstawiono porównanie wyników z osobna dla każdego z badanych zbiorów. Bardziej ogólne wnioski można wyciągnąć dokonując analizy Rysunku 6.5 przedstawiającego te same wyniki, ale uśrednione po pięciu zbiorach. Analizując Rysunek 6.5 można stwierdzić, że najwyższą średnią dokładność klasyfikacji osiągnął algorytm GAAM (91.77%), wymagając jednak do tego, aż 20 cech. Algorytm GAAMmf uzyskał średnią dokładność klasyfikacji zaledwie o około 4% niższą (87.53%) niż algorytm GAAM, ale przy dwukrotnie mniejszej liczbie cech. Średnie wyniki najbardziej zbliżone do wyników algorytmu GAAMmf osiągnął algorytm Melting GAAM oraz metoda SFS. Średnia dokładność klasyfikacji algorytmu Melting GAAM wyniosła 86.69%, a więc była tylko o około 1% niższa niż średnia dokładność uzyskana przy

wykorzystaniu algorytm GAAMmf (84.88%), a liczba cech jedynie o 20% wyższa (12 cech) w porównaniu do algorytmu GAAMmf (10 cech). Z kolei w przypadku metody SFS, liczba cech zawartych w ostatecznym zbiorze cech była co prawda taka sama, jak w przypadku algorytmu GAAMmf, ale średnia dokładność klasyfikacji była o ponad 3% niższa. Dwie pozostałe metody CFS oraz ReliefF osiągnęły zarówno dużo niższą dokładność klasyfikacji, jak i dużo wyższą liczbę cech, aniżeli algorytm GAAMmf. Z przedstawionego porównania wynika, że algorytm GAAMmf okazał się być najlepszy, ponieważ uzyskał wysoką średnią dokładność klasyfikacji przy najmniejszej liczbie cech.



Rysunek 6.5. Średnia dokładność klasyfikacji oraz średnia liczba cech dla kolejnych zbiorów BCI (uśrednienie po 9 metodach).

Źródło: Opracowanie własne.



Rysunek 6.6. Średnia dokładność klasyfikacji oraz średnia liczba cech dla kolejnych metod (uśrednienie po 5 zbiorach).

Źródło: Opracowanie własne.

6.3. Eksperyment 3 - porównanie algorytmu GAAMmf z algorytmami referencyjnymi na benchmarkowych zbiorach danych

W eksperymencie 2 wykazano, że rozwiązania zwrócone przez algorytm GAAMmf dla zbiorów BCI miały co prawda nieco niższą dokładność klasyfikacji, ale jednocześnie zawierały mniejszą liczbę cech w porównaniu do rozwiązań wygenerowanych przez większość innych algorytmów genetycznych (z wyjątkiem algorytmu Melting GAAM) oraz pozostałych metod referencyjnych. Celem eksperymentu 3 było ustalenie, czy ta obserwacja będzie nadal obowiązywać w przypadku rozszerzenia badania na zbiory opisujące różne zagadnienia klasyfikacyjne, charakteryzujące się różną liczbą cech, klas i przykładów. Aby sprostać temu zadaniu wybrano 11 benchmarkowych zbiorach danych powszechnie wykorzystywanych przy testowaniu algorytmów klasyfikacyjnych (*Adult, Coil100, Dermatology, Gisette, Gli_85, Humanactivity, Orl_32x32, Orlraws10P, Pima-Indians-diabetes, WarpAR10P, Yale_32x32*). Ogólna charakterystyka zbiorów została zaprezentowana poniżej.

Adult - Zbiór zawiera 48842 rekordy informujące czy dana osoba zarabia więcej niż 50 tysięcy dolarów rocznie (klasa 1) czy mniej niż 50 tysięcy dolarów rocznie (klasa 0). Każdy rekord opisany jest przez 14 cech (głównie nominalnych) dostarczających informacji na temat wieku, rodzaju pracy, poziomu wykształcenia, lat edukacji, stanu cywilnego, zawodu, relacji, rasy, płci, zysku kapitałowego itp. Celem badania, w trakcie którego powstał zbiór, była analiza wpływu wartości poszczególnych cech na poziom dochodu.

Coil100 - Zbiór zawiera 100 zdjęć różnych obiektów, gdzie każdy z obiektów odpowiada jednej klasie. Zdjęcia każdego z obiektów zostały wykonane podczas jego obracania o 5 stopni na toczeń (urządzenie służące do obracania przedmiotów wokół swojej osi). Dla każdego obiektu wykonano 72 zdjęcia. Rozmiar każdego zdjęcia wynosi 32x32 pikseli z 256 poziomami szarości na piksel. Zatem każde zdjęcie reprezentowane jest przez wektor o 1024 wymiarach (cechach). Celem badania, w trakcie którego powstał zbiór, była analiza struktury i wzorców występujących na zdjęciach oraz klasyfikacja obiektów na podstawie ich zdjęć.

Dermatology - Zbiór zawiera 366 przypadków choroby Eryhemato-Squamous, podzielonych na 6 klas: łuszczyca (klasa 1), łojotokowe zapalenie skóry (klasa 2), liszaj płaski (klasa 3), róża pęcherzykowata (klasa 4), przewlekłe zapalenie skóry (klasa 5) oraz łojotokowe zapalenie skóry z rumieniem i łuską (klasa 6). Każdy przypadek opisany jest za pomocą 12 klinicznych i 22 histopatologicznych cech. Celem badania, w trakcie którego powstał zbiór, była klasyfikacja przypadków choroby Eryhemato-Squamous na podstawie ich cech klinicznych i histopatologicznych.

Gisette - Zbiór zawiera 7000 obrazów z ręcznie pisanymi cyframi „4" oraz „9" (6000 obrazów znajduje się w zbiorze treningowym i 1000 znajduje się w zbiorze testowym), które są opisane przez 5000 cech. Zestaw cech składa się z: 2500 prawdziwych cech (to piksele wybrane losowo z górnej środkowej części ustalonego obrazu o wymiarach 28x28, a także cechy wyższego rzędu utworzone jako iloczyny tych pikseli) oraz 2500 cech dystrakcyjnych, nie mających mocy predykcyjnej. Celem analizy, w trakcie której powstał zbiór, było rozpoznawanie cyfr „4" oraz „9" na podstawie obrazów z cyframi ręcznie pisanymi.

Gli85 – Zbiór zawiera 85 próbek pochodzących od 74 pacjentów z rozpoznaniem nowotworu mózgu (glejakiem). Każda próbka składa się z 22283 cech, dostarczających informacji na temat różnych cech molekularnych glejaka, podtypu glejaka (1,2,3 lub 4) oraz stopniu jego złośliwości (wartości od 1 do 4). Zbiór został utworzony w celu klasyfikacji nowotworów na podstawie analizy cech i składa się z dwóch klas: nowotworów drobnokomórkowych oraz nowotworów nie drobnokomórkowych.

Humanactivity – Zbiór zawiera 24075 obserwacji dotyczących pięciu różnych fizycznych aktywności człowieka: 1 - siedzenie, 2 - stanie, 3 - chodzenie, 4 - bieganie, 5 - taniec. Każda obserwacja opisana została przez 60 cech pobranych z akcelerometru w smartfonie. Zbiór cech obejmuje m.in. cechy informujące o przyspieszeniu, pozycji GPS, żyroskopie, poziomie światła, polu magnetycznym i poziomie dźwięku.

Orl_32x32 – Zbiór zawiera 400 zdjęć twarzy pochodzących od 40 różnych osób (10 zdjęć na osobę), co daje 40 różnych klas. Zdjęcia te są w skali szarości i mają rozmiar 32x32 pikseli (jeden piksel to jedna cecha), co daje w sumie 1024 cechy. Wszystkie zdjęcia zostały wykonane przy użyciu tej samej kamery, oświetlenia i tła. Zbiór jest często

wykorzystywany w zadaniach rozpoznawania twarzy oraz podczas analizy algorytmów uczenia maszynowego przeznaczonych do analizy obrazów.

Orlraws10P – Zbiór zawiera 100 obrazów twarzy pochodzących z bazy danych ORL (ang. *Olivetti Research Laboratory*). Wykorzystywany jest w badaniach z dziedziny rozpoznawania twarzy i zawiera dane na temat 10 osób (10 klas) uczestniczących w badaniu. Każdy z obrazów jest opisany za pomocą 10304 cech dotyczących rozpoznanych wyrazów twarzy (oczy otwarte lub zamknięte, uśmiech lub brak uśmiechu) oraz szczegółów twarzy (okulary lub bez okularów) każdej z osób.

Pima Indians diabetes – Zbiór składa się z 768 rekordów opisujących pacjentów z plemienia Pima Indians, którzy mieszkają w okolicach Phoenix w Arizonie. Dane zawierają 8 cech opisujących każdego pacjenta, w tym m.in. ciśnienie skurczowe serca, ilość fałdów skórnych, stężenie glukozy we krwi na czczo, wskaźnik masy ciała BMI, wiek oraz stężenie insuliny. Każdy pacjent został sklasyfikowany jako posiadający (klasa 1) lub nieposiadający (klasa 0) cukrzycę typu 2. Celem badania, w trakcie którego powstał zbiór, była klasyfikacja przypadków wystąpienia cukrzycy typu 2.

WarpAR10P – Zbiór zawiera 130 obrazów twarzy o rozmiarze 32x32 pikseli, przedstawiających 10 różnych klas obiektów. Każdy obraz jest reprezentowany przez 1024 cechy (32 x 32 piksele). Zbiór jest podzielony na 10 równolicznych klas, każda klasa odpowiada jednemu z rozpoznawanych obiektów.

Yale_32x32 – Zbiór zawiera 165 obrazów w skali szarości w formacie GIF o rozmiarze 32x32 pikseli przedstawiających 15 różnych osób (15 klas). Każdy obraz jest reprezentowany przez 1024 cechy (32 x 32 piksele). Dla każdej osoby dostępne jest 11 obrazów (15 osób x 11 obrazów = 165 próbek), przedstawiających różne wyrazy twarzy lub konfiguracje, np. zdjęcie osoby szczęśliwej, zaskoczonej, sennej, smutnej bez okularów, w okularach, mrugającej itd. Zbiór danych może być wykorzystywany w zadaniach związanych z klasyfikacją obrazów, rozpoznawaniem twarzy lub przy badaniu skuteczności algorytmów uczenia maszynowego.

Przed wykorzystaniem zbiorów danych do porównania wybranych algorytmów selekcji cech, zastosowano następujące procedury wstępnego przetwarzania zbioru cech:

1. usunięcie wszystkich rekordów zawierających wartości NaN;
2. usunięcie zbędnych cech (cech, które miały tę samą wartość dla każdego rekordu);
3. identyfikacja par cech, których liniowa korelacja przekraczała 99%, a następnie odrzucenie jednej cechy z każdej pary.

Szczegółowa charakterystyka zbiorów danych przed i po zastosowaniu procedur wstępnego przetwarzania jest przedstawiona w Tabeli 6.9.

Zbiór danych	Liczba cech	Liczba klas	Liczba prób	Źródło
Adult	14	2	48842/**45222**	[592]
Coil100	1024	100	7 200	[593]
Dermatology	34	6	366/**358**	[594]
Gisette	5000/**4891**	23	7 000	[595]
Gli_85	22283/**22259**	2	85	[596]
Humanactivity	60/**57**	5	24 075	[597]
Orl_32x32	1024/**1023**	40	400	[598]
Orlraws10P	10 304	10	100	[599]
Pima-Indians-diabetes	8	2	768	[600]
WarpAR10P	2400/**2251**	10	130	[601]
Yale_32x32	1024	15	165	[602]

Tabela 6.9. Demografia zbiorów danych wykorzystanych w eksperymencie; wartości zapisane pogrubioną czcionką dotyczą liczby cech/przykładów, które pozostały w zbiorach danych po etapie wstępnego przetwarzania.
Źródło: Opracowanie własne

W celu ustalenia czy algorytm GAAMmf jest w stanie zwracać zbiory cech o satysfakcjonujących charakterystykach (w sensie dokładności klasyfikacji oraz wielkości zbioru cech) bez względu na liczbę klas, cech oraz przykładów zawartych w zbiorze danych wyniki uzyskane dla każdego zbioru porównano z wynikami uzyskanymi przy zastosowaniu 8 metod referencyjnych, tych samych, które wykorzystano w eksperymencie 2. Parametry poszczególnych metod pozostawiono na takich samych poziomach, jak w punkcie 6.2.

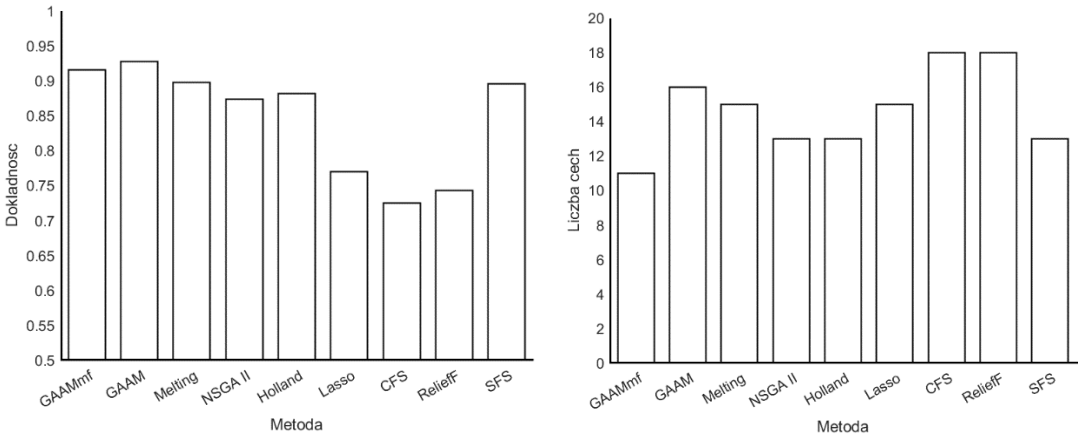
W Tabeli 6.10 znajduje się porównanie wyników osiągniętych przez algorytm GAAMmf oraz osiem metod referencyjnych, tj. GAAM, Melting GAAM, NSGA-II, Holland, Lasso, CFS, ReliefF i SFS dla jedenastu zbiorów danych opisanych w Tabeli 6.9.

Dokładność klasyfikacji [%]										
	GAAMmf	GAAM	Melting	NSGA II	Holland	Lasso	CFS	ReliefF	SFS	**Średnia**
Adult	79.06	79.55	79.27	79.41	79.50	97.27	79.27	79.27	79.07	**81.30**
Coil100	74.98	75.53	75.66			48.53	56.30	52.70	72.85	**65.22**
Dermatology	96.55	98.51	97.24	97.11	97.86	92.75	93.90	95.30	97.21	**96.27**
Gisette	92.97	93.99	90.13			91.44	63.00	76.31	93.40	**85.89**
Gli_85	98.60	100.00	96.68			96.39	75.30	71.80	97.78	**90.94**
Humanactivity	96.70	97.79	96.81	97.34	97.61	90.79	92.40	96.20	93.25	**95.43**
Orl_32x32	97.97	99.61	91.04			78.00	81.80	80.50	95.50	**89.20**
Orlraws10P	98.00	100.00	97.00			76.00	81.10	89.00	99.95	**91.58**
Pima-Indians-diabetes	76.92	77.79	77.80	75.70	77.58	76.70	76.70	76.70	77.74	**77.07**
WarpAR10P	99.62	100.00	93.63			70.00	51.50	41.50	95.39	**78.81**
Yale_32x32	95.77	97.47	92.42			47.24	46.70	58.20	83.68	**74.50**
Średnia	**91.56**	**92.75**	**89.79**	**87.39**	**88.14**	**77.01**	**72.54**	**74.32**	**89.62**	
Liczba cech										
	GAAMmf	GAAM	Melting	NSGA II	Holland	Lasso	CFS	ReliefF	SFS	**Średnia**
Adult	2	8	14	6	5	14	14	14	4	**9**
Coil100	20	20	20			18	20	20	14	**19**
Dermatology	8	18	16	15	14	15	20	20	13	**15**
Gisette	16	20	12			10	20	20	20	**17**
Gli_85	4	16	12			18	20	20	5	**14**
Humanactivity	9	20	20	25	26	18	20	20	14	**19**
Orl_32x32	17	20	18			17	20	20	20	**19**
Orlraws10P	5	17	12			8	20	20	8	**13**
Pima-Indians-diabetes	4	5	5	4	6	8	8	8	6	**6**
WarpAR10P	13	17	12			17	20	20	12	**16**
Yale_32x32	19	20	20			20	20	20	18	**20**
Średnia	**11**	**16**	**15**	**13**	**13**	**15**	**18**	**18**	**13**	

Tabela 6.10. Porównanie wyników zwróconych przez algorytm GAAMmf z wynikami 8 metod referencyjnych dla 11 benchmarkowych zbiorów danych.
Źródło: Opracowanie własne

Analizując wyniki przedstawione w Tabeli 6.10, można zauważyć, że algorytm GAAM osiągał lepszą dokładność klasyfikacji w porównaniu do pozostałych metod dla niemal wszystkich zbiorów. Tylko w przypadku dwóch zbiorów (tj. *Pima-Indians-diabetes* i *Coil100*) algorytm Melting GAAM zwrócił zbiór cech o nieznacznie wyższej dokładności. Warto jednak zauważyć, że wyższa dokładność uzyskiwana przez algorytm GAAM wiązała się ze znacznym rozszerzeniem zbioru cech kodowanych w osobnikach. Opierając porównanie na dokładności klasyfikacji uśrednionej po 11 zbiorach (Rysunek

6.7), na pierwszym miejscu znajduje się oryginalny algorytm GAAM (92.75%). Zaraz za nim, z dokładnością o około 1% niższą, plasuje się zaproponowany algorytm GAAMmf. Trzecie miejsce dzielą algorytm Melting GAAM (89.79%) oraz metoda SFS (89.62). Następne w kolejności są algorytm Hollanda z członem kary (88.14%) oraz algorytm NSGA II (87.39%). Wreszcie zdecydowanie niższą dokładnością, nawet o 10-15%, charakteryzują się zbiory cech zwracane przez metody Lasso (77.01%), ReliefF (74.32%) oraz CFS (72.54%).



Rysunek 6.7. Średnia dokładność klasyfikacji oraz średnia liczba cech dla kolejnych metod (uśrednienie po 11 zbiorach).
Źródło: Opracowanie własne.

Chociaż metody Holland i NSGA-II zwracały osobniki o dokładności tylko trochę niższej niż metody oparte na algorytmie GAAM, nie były one w stanie wygenerować wyników dla wszystkich zbiorów danych. W przypadku zbiorów o niekorzystnym stosunku liczby cech do liczby przykładów (zbiory *Orlraws10P, Gisette, Coil100, Gli 85, Orl 32x32, WarpAR10P* oraz *Yale 32x32*) obydwa algorytmy, napotkały problemy w procesie treningu klasyfikatorów, podobnie jak miało to miejsce w eksperymencie 2. W przeciwieństwie do wskazanych algorytmów, rozpoczynających proces przeszukiwania przestrzeni rozwiązań od zbiorów zawierających w przybliżeniu połowę z całkowitej liczby cech, wszystkie trzy algorytmy oparte na algorytmie GAAM (tj. GAAMmf, GAAM oraz Melting GAAM), pozwalające na początkowe określenie liczby genów (tj. cech) w osobnikach, nie są dotknięte tzw. „przekleństwem wielowymiarowości" i dlatego były w stanie wygenerować wyniki dla wszystkich badanych zbiorów danych, bez względu na ich charakterystykę.

Ponadto, należy zauważyć, że klasyczne metody selekcji cech, wykazywały znacznie niższą skuteczność w przypadku zbiorów danych o wielu klasach (poza metodą SFS). Na przykład, jak można zauważyć w Tabeli 6.10, metoda Lasso oraz oba filtry (CFS i ReliefF), osiągały znacznie gorsze wyniki w przypadku większości zbiorów wieloklasowych (z wyjątkiem zbiorów *Dermatology* i *Gisette* badanych za pomocą metody Lasso). Największy spadek dokładności klasyfikacji, pomiędzy algorytmem GAAMmf a wymienionymi wcześniej metodami, można zaobserwować dla zbiorów *Orlraws10P, Coil100, Orl 32x32, WarpAR10P* i *Yale 32x32*.

W drugiej części Tabeli 6.10, prezentującej liczbę cech zawartych w wybranych zbiorach cech, zaobserwowano znacznie większą różnicę między algorytmem GAAMmf, a metodami referencyjnymi. Zdecydowanie najmniejszą średnią liczbę cech zawierały zbiory cech wybrane przez algorytm GAAMmf (11 cech). Drugie miejsce pod kątem średniej liczby cech zajęła metoda SFS (13 cech), która również osiągała porównywalną średnią dokładność klasyfikacji jak GAAMmf. Taką samą liczbę cech osiągnęły obydwa algorytmy bazujące na kodowaniu binarnym (Holland i NSGA II), jednak branie ich pod uwagę w porównaniu nie do końca jest właściwe, z uwagi na to, że zwróciły one wyniki jedynie dla 4 spośród 11 zbiorów. Zbiory danych zwrócone przez 5 pozostałych metod charakteryzowały się już większą liczbą cech: Lasso i Melting GAAM – 15 cech; GAAM – 16 cech, CFS oraz ReliefF – 18 cech.

Podsumowanie

Podjęta w rozprawie doktorskiej problematyka *Opracowania metody selekcji cech opartej na algorytmach genetycznych dostosowanej do specyficznych charakterystyk interfejsów mózg komputer, to jest do konieczności zachowania informacji o pierwotnej lokalizacji cechy oraz ograniczenia przestrzeni cech do jedynie kilku-kilkunastu istotnych cech* stanowiła duże wyzwanie badawcze. Wynikało to przede wszystkim z tego, że z uwagi na specyficzność fal mózgowych występujących u poszczególnych użytkowników interfejsu mózg-komputer konieczne jest ekstrahowanie bardzo dużej liczby nadmiarowych cech, spośród których następnie wybierane są jedynie te, które dla danego użytkownika wykazują najwyższą zdolność dyskryminacyjną. Stąd liczba cech ekstrahowanych z sygnału EEG na etapie estymacji parametrów klasyfikatora stanowiącego rdzeń interfejsu może być liczona w setkach, a nawet w tysiącach.

Z drugiej strony, aby interfejs mógł działać w trybie zbliżonym do rzeczywistego, konieczne jest jak największe ograniczenie liczby cech, które trzeba ekstrahować z sygnału przy generowaniu każdej kolejnej instrukcji sterującej. Dodatkowym problemem jest to, że z uwagi na wspomnianą powyżej wysoką specyficzność fal mózgowych, zbiór danych gromadzonych w trakcie sesji trenującej (na podstawie którego estymowane są następnie parametry klasyfikatora) wymaga najczęściej przeprowadzenia sesji z każdym z użytkowników z osobna. To powoduje, że zbiór danych dostępnych dla procesu kalibracji interfejsu jest z reguły niewielki i wynosi najczęściej od 100 do 200 rekordów, co jest bardzo niewielką liczbą w porównaniu z tysiącami cech możliwych do ekstrakcji z sygnału EEG. W związku z tym selekcja kilku-kilkunastu cech o najwyższych zdolnościach dyskryminacyjnych jest niezmiernie ważnym, a jednocześnie trudnym do przeprowadzenia, etapem w budowie interfejsu opartego na wyobrażeniu ruchu.

W niniejszej rozprawie przedstawiono autorski algorytm genetyczny GAAMmf przeznaczony do selekcji cech, dostosowany do wskazanych powyżej ograniczeń, tzn. pozwalający na prowadzenie procesu selekcji bez względu na dostępną liczbę cech, przykładów oraz klas.

Jak wykazano w rozdziale VI rozprawy opracowany algorytm:

- działa prawidłowo dla różnych ustawień parametrów *accWeight, fsWeight* oraz *probM* (podpunkt 6.1.2, 6.1.3 oraz 6.1.5);
- jest stabilny, zarówno w odniesieniu do osiąganego stopnia redukcji (czyli zwracanej liczby cech) oraz dokładności klasyfikacji, jak i w odniesieniu do konkretnych wybieranych cech (podpunkt 6.1.4);
- umożliwia zachowanie informacji o interpretacji cechy (podpunkt 6.1.4);
- w przeciwieństwie do algorytmu Hollanda z członem kary oraz algorytmu NSGA II działa prawidłowo bez względu na liczbę cech, klas i przykładów zawartych w zbiorze danych (podpunkt 6.2 oraz 6.3);
- osiąga co prawda nieco niższą dokładność niż oryginalny algorytm GAAM, ale za to pozwala dokonać nie tylko selekcji, ale również redukcji zbioru cech, osiągając zbiory cech o czasami kilkukrotnie mniejszej liczbie cech, niż w przypadku algorytmu GAAM (podpunkt 6.2 oraz 6.3).

Ponadto parametry występujące w zaproponowanym algorytmie pozwalają dodatkowo sterować procesem selekcji. W przypadku, gdy potrzebne jest rozwiązanie o wyższej dokładności, przy nieco niższej redukcji cech, wystarczające jest zwiększenie wagi przypisanej do kryterium dokładności *accWeight* w stosunku do wagi przypisanej do kryterium liczby cech *fsWeight*. I odwrotnie, jeżeli wyższy nacisk ma być położony na jak największą redukcję zbioru cech, wtedy zwiększona powinna zostać waga kryterium liczby cech *fsWeight.*

Należy tu zwrócić jeszcze uwagę na fakt, że pomimo, iż algorytm GAAMmf został opracowany w odpowiedzi na specyficzne aspekty występujące w procesie selekcji cech na potrzeby interfejsów mózg-komputer, to, jak wykazano w eksperymencie 3 (podpunkt 6.3), może on być z powodzeniem stosowany również w innych klasach zadań. W eksperymencie tym pokazano, że spośród 8 zastosowanych metod referencyjnych, jedynie oryginalny algorytm GAAM uzyskał dokładność klasyfikacji (uśrednioną po 11 benchmarkowych zbiorach danych) o około 1% wyższą, aniżeli algorytm zaproponowany w rozprawie, jednak wymagał do tego zbioru cech zawierającego o

ponad 30% więcej cech, niż algorytm GAAMmf. Średni wynik wszystkich pozostałych metod referencyjnych był niższy zarówno pod kątem dokładności klasyfikacji, jak i osiągniętej redukcji cech.

Jak wynika z powyższych uwag cel rozprawy, którym było ***opracowanie metody selekcji cech opartej na algorytmach genetycznych dostosowanej do specyficznych charakterystyk interfejsów mózg komputer, to jest do konieczności zachowania informacji o pierwotnej lokalizacji cechy oraz ograniczenia przestrzeni cech do jedynie kilku-kilkunastu istotnych cech*** został osiągnięty, a teza rozprawy mówiąca, że ***opracowana metoda selekcji cech pozwoli na uzyskanie zbioru cech o wyższych lub porównywalnych zdolnościach dyskryminacyjnych, mierzonych precyzją klasyfikacji wzorców aktywności mózgowej, oraz o mniejszej liczbie cech, aniżeli metody referencyjne*** została udowodniona.

Spis literatury

[1] A. Cudo, E. Zabielska i B. Bałaj, „Wprowadzenie w zagadnienie interfejsów mózg-komputer.," Studia z Psychologii w KUL., tom 17, pp. 189-211, 2011.

[2] B. Allison, B. Z. Pfurtscheller i G. Graimann, Brain Computer Interfaces Revolutionizing Human Computer Interaction, New York: Springer, 2010.

[3] N. Birbaumer, „Breaking the silence: Brain–computer interfaces (BCI) for communication and motor control.," Psychophysiology., tom 43, pp. 517-532, 2006.

[4] I. Rejer, „Genetic Algorithms in EEG Feature Selection for the Classification of Movements of the Left and Right Hand," w Advances in Intelligent and Soft Computing, Springer, 2013.

[5] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller i T. M. Vaughan, „Brain–computer interfaces for communication and control.," Clinical Neurophysiology., tom 113, nr 6, pp. 767-791, 2002.

[6] M. G. Cheng, „Design and implementation of a brain-computer interface with high transfer rates.," IEEE Transactions on Biomedical Engeneering., tom 49, nr 10, pp. 1181-1186, 2002.

[7] T. Kowalczyk, „Interfejsy mózg-komputer jako rozwiązanie dla osób niepełnosprawnych," [Online]. Available: http://ww2.ii.uj.edu.pl/~smietans/bio-sem/BCI.pdf. [Data uzyskania dostępu: 10 05 2017].

[8] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller i T. M. Vaughan, „Brain–computer interfaces for communication and control.," Clinical Neurophysiology., tom 113, nr 6, pp. 767-791, 2002.

[9] N. Birbaumer i L. G. Cohen, „Brain-computer interfaces: communication and restoration of movement in paralysis," Journal of Physiology., tom 579, pp. 621-636, 2007.

[10] A. Broniec-Wójcik, Zastosowanie wybranych epizodów elektroencefalograficznych jako sygnału sterującego w interfejsie człowiek-

maszyna., Kraków: Akademia Górniczo-Hutnicza w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej., 2013.

[11] A. Bruzzo, Seizure prediction and control in epilepsy., Bologna.: Università degli Studi di Bologna., 2008.

[12] S.-F. Liang, F.-Z. Shaw, C.-P. Young, D.-W. Chang i Y.-C. Liao, „A closed-loop brain computer interface for real-time seizure detection and control.," w Conference proceedings Engineering in Medicine and Biology Society., Buenos Aires, Argentina, 2010.

[13] M. Kołodziej, Przetwarzanie, analiza i klasyfikacja sygnału EEG na użytek interfejsu mózg-komputer., Warszawa: Politechnika Warszawska, Wydział Elektryczny, 2011.

[14] S. R. Mousavi, M. Niknazar i B. V. Vahdat, „Epileptic Seizure Detection using AR Model on EEG Signals.," Biomedical Engineering Conference, CIBEC 2008. Cairo International, pp. 1-4, 2008.

[15] E. A. Larsen, Classification of EEG Signals in a Brain-Computer Interface System, Gloshaugen, Trondheim: Norwegian University of Science and Technology, Department of Computer and Information Science., 2011.

[16] A. Subasi, A. Alkan, E. Koklukaya i M. K. Kiymik, „Wavelet neural network classification of EEG signals by using AR model with MLE preprocessing.," Neural Networks., tom 18, nr 7, pp. 985-997, 2005.

[17] L. Huang i G. van Luijtelaar, „Brain Computer Interface for Epilepsy Treatment.," w Brain-Computer Interface Systems - Recent Progress and Future Prospects., 2013, pp. 239-252.

[18] I. Hubner, J. Hubner , I. Witek i S. Kroczka, „Stwardnienie zanikowe boczne z objawami piramidowo pozapiramidowymi oraz jako zespół paranowotworowy - opis dwóch przypadków," Polski Przegląd Neurologiczny, tom 13, nr 3, pp. 137-143, 2017.

[19] „Stwardnienie Zanikowe Boczne (ALS)," 12 05 2023. [Online]. Available: https://www.pratia.pl/s/resources/item/stwardnienie-zanikowe-boczne-als. [Data uzyskania dostępu: 12 05 2023].

[20] Ź. P. Medycyny, „Źródło: Puls Medycyny," 12 05 2023. [Online]. Available: https://pulsmedycyny.pl/w-polsce-odnotowuje-sie-90-tys-udarow-niedokrwiennych-rocznie-971045. [Data uzyskania dostępu: 12 05 2023].

[21] K. G. Policji, „Wypadki drogowe w Polsce w 2022 r.” 15 05 2023. [Online]. Available: https://statystyka.policja.pl/st/ruch-drogowy/76562,Wypadki-drogowe-raporty-roczne.html. [Data uzyskania dostępu: 23 05 2023].

[22] I. Rejer, „Genetic Algorithms for Feature Selection for Brain Computer Interface.,” Journal of Artificial Intelligence Research., tom 1, pp. 1-15, 2015.

[23] M. A. B. Brazier, A history of the electric al activity of the brain. The first half-century., London: Pitman, 1961.

[24] E. Niedermeyer i F. Lopes de Silva, „Historical aspect.,” w Elektroencephalography. Basic principles, clinical applications and related Fields., Williams & Wilkins., 1999, pp. 1-14.

[25] H. Berger, „Uber das Elektroenkephalogramm des Menschen.,” Archiv fur Psychiatrie Und Nervenkrankheiten., tom 89, pp. 527-570, 1929.

[26] K. Lorenz, „Przegląd algorytmów genetycznych stosowanych w procesie selekcji cech wyekstrahowanych z sygnału EEG,” w Młodzi naukowcy dla polskiej nauki (monografia), Kraków, pod red. M. Kuczera, 2013.

[27] H. Kwaśnicka, Ewolucyjne projektowanie sieci neuronowych., Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej, 2007.

[28] R. Kurzacz, Metody rankingowe oparte o maszynę wektorów podpierających w zarządzaniu danymi finansowymi i bio-medycznymi., Katowice: Politechnika Śląska, 2006.

[29] G. H. John, R. Kovahi i K. Pfleger, Irrelevant Features and the Subset Selection Problem, Computer Science Department Stanford University, 1994.

[30] Z. Michalewicz, Algorytmy genetyczne + struktury danych = programy ewolucyjne, Warszawa : Wydawnictwa Naukowo – Techniczne, 1995.

[31] D. Garrett, D. A. Peterson, C. Anderson i M. H. Thaut, „Comparison of Linear, Nonlinear, and Feature Selection Methods for EEG Signal Classification.,” IEEE Transactions on Neural Systems and Rehabilitation Engineering., tom 11, nr 2, pp. 141-145, 2003.

[32] D. Koller i M. Sahami, „Toward optimal feature selection,” w Proceedings Machine Learning, pp.284-292, 1996.

[33] G. Pfurtscheller, D. Flotzinger i J. Kalcher, „Brain-computer interface-a new communication device for handicapped persons.,” Journal of Microcomputer Application, tom 16, pp. 293-299, 1993.

[34] D. A. Peterson, J. N. Knight, M. J. Kirby, C. W. Anderson i M. H. Thaut, „Feature Selection and Blind Source Separation in an EEG-Based Brain-Computer Interface.," EURASIP Journal on Applied Signal Processing, tom 19, pp. 3128-3140, 2005.

[35] E. Yom-Tov i G. F. Inbar, „Feature Selection for the Classification of Movements From Single Movement-Related Potentials," IEEE Transactions on Neural Systems and Rehabilitation Engineering, tom 10, nr 3, pp. 170-177, 2002.

[36] M. Kołodziej, A. Majakowski i J. R. Rak, „A new Method of EEG Classification for BCI with Feature Extraction Based on Higher Order Statistics of Wavelet Components and Selection with Genetic Algorithms.," w ICANNGA 2011, Part I LNCS 6593, Berlin, Springer-Verlag Berlin, 2011, pp. 280-289.

[37] P. S. Hammon i V. R. de Sa, „Preprocessing and meta-classification for brain-computer interfaces," IEEE Transactions on Biomedical Engineering, tom 54, nr 3, pp. 518-525, 2007.

[38] I. Koprinska, „Feature Selection for Brain-Computer Interfaces," w T. Theeramunkong et al. (Eds.): PAKDD Workshops 2009, LNAI 5669, Berlin, Springer-Verlag Berlin Heidelberg, 2010, pp. 100-111.

[39] M. Sewell, „Feature Selection," 09 02 2017. [Online]. Available: https://pdfs.semanticscholar.org/aacd/187f333a60718387d4f42a18929d18203e0e.pdf.

[40] I. Rejer, „Genetic Algorithms in EEG Feature Selection for the Classification of Movements of the Left and Right Hand.," Advances in Intelligent and Soft Computing, 2013.

[41] J. Decety, D. Perani, M. Jeannerod, V. Bettinardi, B. Tadardy , R. Woods, J. Mazziotta i F. Fazio, „Mapping motor representations with positron emission tomography.," Nature, tom 37, pp. 600-602, 1994.

[42] P. Ziemba, „Redukcja wymiarowości i selekcja cech w zadaniach klasyfikacji i regresji z wykorzystaniem uczenia maszynowego.," Studia Informatica, tom 30 (733), pp. 221-236, 2012.

[43] J. Korbicz i J. M. Kościelny, Modelowanie, diagnostyka i sterowanie nadrzędne procesami., Warszawa: Wydawnictwa Naukowo-Techniczne, 2009.

[44] P. Lajmert i B. Kruszyński, „Zastosowanie Metod Eksploracji Danych Do Nadzorowania Procesu Szlifowania Kłowego Wałków.," Mechanik, XXXVI Naukowa Szkoła Obróbki Ściernej., Tomy %1 z %28-9, pp. 291-298, 203.

[45] M. A. Hall, Correlation-based Feature Selection for Machine Learning., Hamilton, NewZealand: The University of Waikato, Department of Computer Science., 1999.

[46] O. S. Soliman i A. Rassem, „Correlation Based Feature Selection Using Quantum Bio Inspired Estimation of Distribution Algorithm.," w International Workshop on Multi-disciplinary Trends in Artificial Intelligence. MIWAI 2012: Multi-disciplinary Trends in Artificial Intelligence., Berlin Heidelberg., Springer-Verlag., 2012, pp. 318-329.

[47] M. Doshi i D. S. Chaturvedi, „Correlation Based Feature Selection (CFS) Technique To Predict Student Perfromance.," International Journal of Computer Networks & Communications (IJCNC), tom 6, nr No. 3, pp. 197-206, 2014.

[48] M. A. Hall, „Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning.," w ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning., San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2000, pp. 359-366.

[49] R. Wald, T. M. Khoshgoftaar i A. Napolitano, „Using Correlation-Based Feature Selection for a Diverse Collection of Bioinformatics Datasets.," w IEEE 14th International Conference on Bioinformatics and Bioengineering, IEEE Computer Society, 2014, pp. 156-162.

[50] M. A. Hall i L. A. Smith, „Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper.," w Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference., AAAI Press., 1999, pp. 235-239.

[51] J. P. Piątkowski, Analiza i rozwój metod selekcji cech dla dużych problemów klasyfikacyjnych., Toruń: Uniwersytet Mikołaja Kopernika, Wydział Fizyki, Astronomii i Informatyki Stosowanej, 2006.

[52] S. F. Rosario i D. K. Thangadurai, „RELIEF: Feature Selection Approach.," International Journal of Innovative Research & Development, tom 4, nr 11, pp. 218-224, 2015.

[53] M. Robnik-Sikonja i I. Kononenko, „An adaptation of Relief for attribute estimation in regression.," Proceeding ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning., pp. 296-304, 1997.

[54] R. P. Durgabai, „Feature Selection using ReliefF Algorithm.,” International Journal of Advanced Research in Computer and Communication Engineering., tom 3, nr 10, pp. 8215-8218, 2014.

[55] K. Kira i L. A. Rendell, „A Practical Approach to Feature Selection.,” Proceeding ML92 Proceedings of the ninth international workshop on Machine learning., pp. 249-256, 1992.

[56] I. Kononenko, E. Simec i M. Robnik- Sikonja, „Overcoming the myopia of inductive learning algorithms with RELIEFF.,” Applied Intelligence, tom 7, pp. 39--55, 1997.

[57] A. Zafra, M. Pechenizkiy i S. Ventura, „ReliefF-MI: An extension of ReliefF to multiple instance learning.,” Neurocomputing, tom 75, nr 1, pp. 210-218, 2012.

[58] A. Zafra, M. Pechenizkiy i S. Ventura, „Feature Selection is the ReliefF for Multiple Instance Learning.,” In Proceedings of ISDA 2010 Conference, IEEE Computer Society, pp. 525-532, 2010.

[59] I. Kononenko, „Estimating Attributes: Analysis and Extensions of Relief.,” 7th International Conference on Machine Learning (ICML)., pp. 171-182, 1994.

[60] I. H. Witten i E. Frank, Data Mining: Practical Machine Learning Tools and Techniques 2nd edition, San Francisco: Morgan Kaufmann Publishers, 2005.

[61] W. Chmielnicki, Efektywne metody selekcji cech i rozwiązywania problemu wieloklasowego w nadzorowanej klasyfikacji danych., Kraków: Instytut Podstawowych Problemów Techniki Polskiej Akademii Nauk., 2012.

[62] M. Osborne, B. Presnell i B. Turlach, „On the LASSO and its dual,” Journal of Computational and Graphical Statistics., tom 9, pp. 319-337, 1999.

[63] V. Roth, „The Generalized LASSO,” IEEE Transactions On Neural Networks., tom 15, nr No. 1, pp. 16-28, 2004.

[64] W. J. Fu, „Penalized Regressions: The Bridge Versus the Lasso.,” Journal of Computational and Graphical Statistics,, tom 7, nr No. 3, p. 397–416, 1998.

[65] U. Libal, „Wavelet Decomposition of Signal and Feature Selection by LASSO for Pattern Recognition,” Przegląd elektrotechniczny, tom 4, nr ISSN 0033-2097, pp. 89-91, 2013.

[66] J. Kim, Y. Kim i Y. Kim, „A Gradient-Based Optimization Algorithm for LASSO,” Journal of Computational and Graphical Statistics, tom 17, pp. 994-1009, 2008.

[67] R. Tibshirani, „Regression Shrinkage and Selection via the Lasso.,” Journal of the Royal Statistical Society. Series B (Methodological)., tom 58, nr No. 1, pp. 267-288, 1996.

[68] „Lasso,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Lasso_(statistics). [Data uzyskania dostępu: 14 02 2017].

[69] I. Rejer i K. Lorenz, „Selekcja cech wyekstrahowanych z sygnału EEG odzwierciedlającego wyobrażenie ruchu lewą i prawą ręką,” w Wybrane zagadnienia informatyki medycznej., Szczecin, Stowarzyszenie Przyjaciół Wydziału Informatyki w Szczecinie, 2012, pp. 55-65.

[70] M. Kołodziej, Przetwarzanie, analiza i klasyfikacja sygnału EEG na użytek interfejsu mózg-komputer., Warszawa: Politechnika Warszawska, 2011.

[71] P. Boniński, M. Kazubek, A. Przelaskowski i A. Wróblewska, „Zastosowanie Kwantyzacji Wektorowej LVQ W Procesie Klasyfikacji Mikrozwapnień W Cyfrowej Mammografii,” w IV Sympozjum "Techniki Przetwarzania Obrazu"., Warszawa, 2002.

[72] „Automatyczna Selekcja Zmiennych,” [Online]. Available: http://algolytics.pl/wp-content/uploads/docs/pl/bk01pt04ch29s02.html. [Data uzyskania dostępu: 15 02 2017].

[73] K. Gałda, Zastosowanie algorytmów genetycznych do optymalizacji modelu SVM procesu stalowniczego., Katowice: Wydział Inżynierii Materiałowej i Metalurgii, Zakład Informatyki w Procesach Technologicznych, Katedra Elektrotechnologii, 2009.

[74] P. M. Szczypiński, Komputerowa analiza obrazów z endoskopu bezprzewodowego dla diagnostyki medycznej., Łódź: Politechnika Łódzka, 2012.

[75] P. Pudil i P. Somol, „Current Feature Selection Techniques in Statistical Pattern Recognition.,” Computer Recognition Systems, tom 30, pp. 53-68, 2005.

[76] „Forward selection,” [Online]. Available: https://www.stat.ubc.ca/~rollin/teach/643w04/lec/node41.html. [Data uzyskania dostępu: 15 02 2017].

[77] P. Pudil, J. Novovičová i J. Kittler, „Floating search methods in feature selection.,” Pattern Recognition Letters., tom 15, nr No. 11, p. 1119–1125, 1994.

[78] L. Yu i H. Liu, „Toward Integrating Feature Selection Algorithms for Classification and Clustering.,” IEEE Transactions on Knowledge & Data Engineering., tom 17, pp. 491-502, 2005.

[79] „Backward selection,” [Online]. Available: https://www.stat.ubc.ca/~rollin/teach/643w04/lec/node42.html. [Data uzyskania dostępu: 15 02 2017].

[80] S. Mirjalili, „Particle Swarm Optimisation,” Evolutionary Algorithms and Neural Networks, tom 780, p. 15–31, 2018.

[81] L. Davis, Handbook Of Genetic Algorithms, New York: Van Nostrand Reinhold, 1991.

[82] E. Yom-Tov i G. F. Inbar, „Feature Selection for the Classification of Movements From Single Movement-Related Potentials.,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, tom 10, nr 3, pp. 170-177, 2002.

[83] Rejer, I., „Wprowadzenie do algorytmów genetycznych.,” w Studia Informatica, tom 18, Szczecin, 2006.

[84] J. Heitkoetter, D. Beasley i Eds., „The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ).,” 1994. [Online]. Available: comp.ai.genetic.rtfm.mit.edu:/pub/usenet/news.answers/ai-faq/genetic/.

[85] J. Arabas, Wykłady z algorytmów ewolucyjnych, Warszawa: Wydawnictwa Naukowo-Techniczne, 2001.

[86] Anon, „How Do Genetic Algorithms Work?,” 2013. [Online]. Available: http://www.generation5.org/.

[87] J. Shen, Genetic algorithms and genetic programming, Germany: University Bielefeld, 2000.

[88] Hill, Seamus, O'Riordan i Colm, Analysis of the performance of Genetic Algorithmsand their Operators using Kauffman's NK Model, NUI, Galway, Ireland: Dept. of Information Technology, 2002.

[89] Z. Michalewicz, Algorytmy genetyczne + struktury danych = programy ewolucyjne., Warszawa: Wydawnictwa Naukowo – Techniczne, 1995.

[90] I. Rejer i K. Lorenz, „Classic genetic algorithm vs. genetic algorithm with aggressive mutation for feature selection for brain-computer interface,” Przegląd Elektrotechniczny, tom 91, nr 2, pp. 98-102, 2015.

[91] N. Srinivas i K. Deb, „Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms,” Evolutionary Computation, tom Vol. 2, nr No. 3, pp. 221-248, 1994.

[92] K. Deb, A. Pratap, S. Agarwal i T. Meyarivan, „A fast and elitist multiobjective genetic algorithm: NSGA-II.," IEEE Transactions on Evolutionary Computation (doi:10.1109/4235.996017), tom 6, nr 2, pp. 182 - 197, April April 2002.

[93] „ACM," 12 05 2023. [Online]. Available: https://dl.acm.org/ccs. [Data uzyskania dostępu: 12 05 2023].

[94] K. Lorenz i I. Rejer, „Feature selection with NSGA and GAAM in EEG signals domain.," 8th International Conference on Human System Interaction (HSI)., pp. 94-98, 2015.

[95] I. Rejer, „Genetic Algorithm with Aggressive Mutation for Feature Selection in BCI Feature Space.," Formal Pattern Analysis & Applications., 2014.

[96] B. E. Swartz, „Timeline of the history of EEG and associated fields.," Elektroencephalography and Clinical Neurophysiology, tom 106, pp. 173-176, 1988.

[97] J. Majkowski, „Elektryczność, elektrofizjologia i elektroencefalografia – rys historyczny," w Elektroencefalografia kliniczna, Warszawa, PZWL, 1989, pp. 2-19.

[98] K. Jus i A. Jus, „Rys historyczny," w Elektroencefalografia kliniczna, Warszawa, PZWL, 1967, pp. 11-17.

[99] V. A. Niemiński, „Ein Versuch der Registrierung der elektrischen Gehirnerscheinungen.," Zentralblatt fur Physiologie., tom 27, pp. 951-960, 1913.

[100] M. A. B. Branzier, Czynność elektryczna układu nerwowego., Warszawa: PZWL, 1964.

[101] A. M. Grass, The electroencephalographic heritage until 1960., Quincy: GRAS Instrument Company, 1984.

[102] R. Werner, R. Ullman i H. Fritzsch, „Die bedeutung es Elektroenzephalographie unter Berucksitigung des EEG-Gerates.," w Handbuch medizinscher Elektronik., Berlin., VEB Verlag Technik., 1965, pp. 1-59.

[103] R. Cooper, J. W. Osselton i J. C. Shaw, EEG Technology, London, 1969.

[104] A. Cudo, E. Zabielska, B. Bałaj, Wprowadzenie w zagadnienie Interfejsów mózg - komputer, ISBN 978-83-7702-473-7 Lublin, Wyd. KUL 2011, s. 189-211, 2011.

[105] N. Birbaumer, „Breaking the silence: Brain-computer interfaces (BCI) for communication and motor control.," Psychophysiology., tom 43, nr 6, pp. 517-532, 2006.

[106] D. M. Taylor, ME. Stetner, Intracortical BCIs: A brief history of neural timing, 2010.

[107] J.R. Wolpaw, D.J. McFarland, „Brain-computer interfaces for communication and control," Clinical Neurophysiology, tom 113(6), pp. 767-791, 2002.

[108] J. R. Wolpaw, N. Birbaumer, W. J. Heetderks, D. J. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. A. Quatrano, J. C. Robinson i T. M. Vaughan, „Brain-computer interface technology: A review of the first international meeting.," IEEE Transactions on Rehabilitation Engineering., tom 8, nr 2, pp. 161-173, 2000.

[109] BNCI Horizon 2020, Roadmap, The future In brain/neural-computer interaction: HORIZON 2020., ISBN 978-3-85125-379-5. red., Graz University of Technology.: European Commission within the 7th Framework Programme, Project runtime: Nov 2013–Apr 2015., 2015.

[110] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer i J. R. Wolpaw, „BCI2000: a general purpose brain computer interface (BCI) system.," IEEE Transactions on Biomedical Engineering., tom 51, nr 6, pp. 1034-1043, 2004.

[111] J. D. Bayliss, Use of the evoked potential p3 component for control in a virtual apartment., Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 11(2), 113-116. 1534-4320, 2003.

[112] J. I. Bayliss, „Changing the P300 brain computer interface.," Cyberpsychology and Behaviour., tom 7, nr 76, pp. 694-704, 2004.

[113] G. P. Edlinger, R. Prueckl, G. Krausz, C. Holzner i C. Guger, „P4-24 P300 and SSVEP based brain-computer interface for control of a smart home virtual environment.," Clinical Neurophysiology., tom 121, nr 1, p. 126, 2010.

[114] G. Pfurtscheller, R. Leeb, C. Keinrath, D. Friedman, C. Neuper, C. Guger i M. Slater, „Walking from thought.," Brain Research, tom 1071, pp. 145-152, 2006.

[115] E. Mikołajewska i D. Mikołajewski, „Neuroprostheses for Increasing Disabled Patients' Mobility and Control," Advances in Clinical and Experimental Medicine, tom 21, nr 2, pp. 263-272, 2012.

[116] E. Mikołajewska i D. Mikołajewski, „Problemy techniczne i medyczne w zakresie szerszego wykorzystania neuroprotez u pacjentów neurologicznych," Pielęgniarstwo Neurologiczne i Neurochirurgiczne., tom 1, nr 3, pp. 119-123, 2012.

[117] L. Kiersnowski, „Możliwości stosowania neuroprotez w uszkodzeniach układu nerwowego.," Aktualne Problemy Biomechaniki,, tom 1, pp. 105-110, 2007.

[118] E. Mikołajewska i D. Mikołajewska, „Wybrane rozwiązania automatyki i robotyki w wózkach dla niepełnosprawnych.," Postępy Rehabilitacji, tom 1, pp. 11-18, 2011.

[119] E. Mikołajewska i D. Mikołajewski, „Zastosowania automatyki i robotyki w wózkach dla niepełnosprawnych i egzoszkieletach medycznych.," Pomiary Automatyka Robotyka., tom 5, pp. 58-63, 2011.

[120] E. Nowak, „Doskonałość. Z genealogii human enhancement.," Humaniora. Czasopismo Internetowe., tom Nr 1(5), pp. 77-102, 2014.

[121] A. Cegielska i M. Olszewski, „Nieinwazyjny interfejs mózg–komputer do zastosowań technicznych," Pomiary Automatyka Robotyka., tom 19, nr 3, pp. 5-14, 2015.

[122] K. Białkowski, „Człowiek i części zamienne - czym dysponujemy?," w VI Konkurs Wiedzy Politechnicznej., Kwidzyn, II Liceum Ogólnokształcące im. Stanisława Wyspiańskiego w Kwidzynie., 2016.

[123] R. Kinalski, „Sprzężenie neurofizjologii klinicznej z teradiagnostyką: perspektywa nowego zawodu fizjoterapeuty?," Zeszyty Promocji Rehabilitacji, Ortopedii, Neurofizjologii i Sportu - IRONS - Kwartalnik Naukowy., tom 1, 2012.

[124] S. Cygan i K. Wildner, „Biomechanika Inżynierska," Instytut Metrologii i Inżynierii Biomedycznej, Politechnika Warszawska, [Online]. Available: http://zib.mchtr.pw.edu.pl/downloads/Przedmioty/BIOME/BIOME2017-wyklad-1-Cygan.pdf. [Data uzyskania dostępu: 10 05 2017].

[125] M. Błaszak, A. I. Brzezińska i Ł. Przybylski, „Strategie podwyższania jakości życia osób niepełnosprawnych: perspektywa neurokognitywistyki rozwojowej.," w Nauka., tom 1, Poznań., Polska Akademia Nauk. Oddział w Poznaniu, 2010, pp. 115-137.

[126] M. R. Popovic, D. B. Popowic i T. Keller, „Neuroprostheses for grasping," Neurological Research; A Journal of Progress in Neurosurgery, Neurology and Neurosciences, tom 24, nr 5, pp. 443-452, 2013.

[127] E. Mikołajewska i D. Mikołajewski, „Płaszczyzny współpracy specjalistów medycznych oraz inżynierów biomedycznych i biocybernetyków.," Studia Medyczne., tom 29, nr 1, pp. 121-128, 2013.

[128] J. Chapin i K. Moxon, Neural Prostheses for Restoration of Sensory and Motor Function, United States of America: CRC Press LLC, 2001.

[129] F. Kulaszyński, „Dziś i jutro inteligentnych robotów. Interfejs mózg-komputer.,” w Neurokognitywistyka w patologii i zdrowiu., Szczecin, Pomorski Uniwersytet Medyczny w Szczecinie., 2009-2011, p. 134–142.

[130] G. Pfurtscheller, G. R. Muller, J. Pfurtscheller, H. J. Gerner i R. Rupp, „'Thought' – control of functional electrical stimulation to restore hand grasp in a patient with tetraplegia.,” Neuroscience Letters., tom 351, pp. 33-36, 2003.

[131] J. Zielińska, Wybrane techniki obrazowania sygnałów w perspektywie pedagogiki specjalnej. Przykłady zastosowania w praktyce diagnostyczno - terapeutycznej., Kraków: Wydawnictwo Naukowe Uniwersytetu Pedagogicznego., 2016.

[132] I. Pietralik i S. Łagan, „Aparaty słuchowe a implanty słuchu. Stan obecny i perspektywy rozwoju.,” Aktualne Problemy Biomechaniki, tom 5, pp. 127-132, 2011.

[133] J. Zielińska, „Interfejs mózg – komputer w teorii i praktyce.,” Sympozjum: Człowiek - Media - Edukacja, tom 10, 2013.

[134] P. Durka, „Mózg, maszyny i manipulacje.,” [Online]. Available: http://www.fuw.edu.pl/~durka/var/Mozg_maszyny_manipulacje_NI_Durka.pdf. [Data uzyskania dostępu: 10 05 2017].

[135] M. Konarska-Król, M. J. Kacperska, K. Jastrzębski, M. Radek i B. Tomasik, „Zaburzenia słuchu w praktyce neurologa.,” Aktualności Neurologiczne., tom 14, nr 1, pp. 61-69, 2014.

[136] E. Mikołajewska i D. Mikołajewski, „Interfejsy mózg-komputer jako rozwiązania dla osób niepełnosprawnych z uszkodzeniami układu nerwowego.,” Niepełnosprawność – zagadnienia, problemy, rozwiązania., tom 3, nr 4, pp. 19-36, 2012.

[137] T. Przewoźny, Stan słuchu u chorych we wczesnym okresie udaru niedokrwiennego mózgu., Gdańsk: Akademia Medyczna w Gdańsku, 2007.

[138] P. Triponyuwasin i Y. Wongsawat, „Brain-Computer Interface Based Stroke Rehabilitation For Hemiplegia.,” w The 2014 Biomedical Engineering Intern ational Conference (BMEiCON-2014)., 2014.

[139] F. Pichiorri, F. Cincotti, F. De Vico Fallani, I. Pisotta, G. Morone, M. Molinari i D. Mattia, „Towards a Brain Computer Interface-Based Rehabilitation: from Bench to Bedside.,” 5th International BCI Conference 2011, pp. 268-271, 2011.

[140] A. Sójka, K. Janiak, A. Sikorska i K. Smółka, „Interfejsy BCI jako nowy kanał komunikacji dla osób z niepełnosprawnością ruchową.," Rocznik Kolegium Analiz Ekonomicznych, tom 42, pp. 385-403, 2016.

[141] S. R. Soekadar, N. Birbaumer, M. W. Slutzky i L. G. Cohen, „Brain–machine interfaces in neurorehabilitation of stroke," Neurobiology of Disease., tom 83, pp. 172-179, 2015.

[142] U. Chaudhary, N. Birbaumer i M. R. Curado, „Brain-Machine Interface (BMI) in paralysis.," Annals of Physical and Rehabilitation Medicine, tom 58, pp. 9-13, 2015.

[143] M. Arvaneh, C. Guan, K. K. Ang, T. E. Ward, K. S. G. Chua, C. W. K. Kuah, G. J. E. Joseph, K. S. Phua i C. Wang, „Facilitating motor imagery-based brain–computer interface for stroke patients using passive movement.," Neural Computing and Applications., pp. 1-14, 2016.

[144] E. Mikołajewska i D. Mikołajewski, „Implikacje wykorzystania interfejsów mózg-komputer u pacjentów z zaburzeniami świadomości.," International Letters of Social and Humanistic Sciences., tom 8, pp. 86-90, 2013.

[145] K. K. Ang i C. Guan, „Brain-Computer Interface in Stroke Rehabilitation," Journal of Computing Science and Engineering., tom 7, nr 2, pp. 139-146, 2013.

[146] R. Ortner, D.-C. Irimia, J. Scharinger i C. Guger, „A Motor Imagery based Brain-Computer Interface for Stroke Rehabilitation.," Studies in health technology and informatics, 2012.

[147] P. Kucharski, A. J. Rybicki i M. Kopaczyńska, „Połączenie mózg-komputer jako metoda komunikacji z niereagującymi pacjentami - przegląd literatury.," Inżynieria biomedyczna., tom 23, nr 3, pp. 148-157, 2015.

[148] J. J. Daly i J. R. Wolpaw, „Brain–computer interfaces in neurological rehabilitation.," Lancet Neurology., tom 7, nr 11, pp. 1032-1043, 2008.

[149] F. Nijboer, „Technology transfer of brain-computer interfaces as assistive technology: Barriers and opportunities.," Annals of Physical and Rehabilitation Medicine., tom 58, pp. 35-38, 2015.

[150] L. E. H. van Dokkum, T. Ward i I. Laffont, „Brain computer interfaces for neurorehabilitation - its current status as a rehabilitation strategy post-stroke.," Annals of Physical and Rehabilitation Medicine, tom 58, pp. 3-8, 2015.

[151] M. Zych, „Funkcjonalna elektrostymulacja kontrolowana za pomocą interfejsu mózg - komputer w rehabilitacji ręki po udarze – przegląd literatury.," w IX

wiosna z fizjoterapią; cykliczne sympozjum studenckich kół naukowych; aktualne kierunki rozwoju fizjoterapii i rehabilitacji., Warszawa, Warszawski Uniwersytet Medyczny, 2015.

[152] A. Cegielska i M. Olszewski, „Nieinwazyjny interfejs mózg–komputer do zastosowań technicznych.," Pomiary Automatyka Robotyka., tom 19, nr 3, pp. 5-14, 2015.

[153] Y. U. Khan, O. Farooq, M. Tripathi, P. Sharma i P. Alam, „Automatic detection of non-convulsive seizures using AR modeling.," w 2nd International Conference on Power, Control and Embedded Systems (ICPCES)., Allahabad, India, 2012.

[154] H. Min i L. Sun, „EEG signal classification for epilepsy diagnosis based on AR model and RVM.," w International Conference Intelligent Control and Information Processing (ICICIP), 2010.

[155] S.-F. Liang, F.-Z. Shaw, C.-P. Young, D.-W. Chang i Y.-C. Liao, „A Closed-loop Brain Computer Interface for Real-time Seizure Detection and Control.," 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)., pp. 4950-4953, 2010.

[156] D. Zhu, J. Bieger, G. G. Molina i R. M. Aarts, „A survey of stimulation methods used in SSVEP-based BCIs," Computational Intelligence and Neuroscience, tom 1, pp. 1-12, 2010.

[157] R. Zerafa, T. Camilleri, O. Falzon i K. P. Camilleri, „To train or not to train? A survey on training of feature extraction methods for SSVEP-based BCIs," Journal of Neural Engineering, tom 15, nr 5, 2018.

[158] C. Lin, C. Zhang, W. Wu i X. Gao, „Frequency Recognition Based on Canonical Correlation Analysis for SSVEP-Based BCIs," IEEE Transactions on Biomedical Engineering, tom 53, nr 12, pp. 2610-2614, 2006.

[159] R. Fazel-Rezai, Z. Brendan, C. Guger, E. Sellrs, S. Kleih i A. Kubler, „P300 brain computer interface: current challenges and emerging trends," Frontiers in Neuroengineering, tom 5, 2012.

[160] A. Barachant i M. Congedo, „A Plug&Play P300 BCI Using Information Geometry," Machine Learning, 2014.

[161] R. C. Panicker, S. Puthusserypady i Y. Sun, „An Asynchronous P300 BCI With SSVEP-Based Control State Detection," IEEE Transactions on Biomedical Engineering, tom 58, nr 6, pp. 1781-1788, 2011.

[162] M. Górska i M. Olszewski, „Interfejs mózg–komputer w zadaniu sterowania robotem mobilnym,” Pomiary Automatyka Robotyka, nr 3, pp. 15-24, 2015.

[163] A. Górski, „Zastosowanie Elektroencefalografii (EEG) w Technikach Multimedialnych.,” [Online]. Available: https://sound.eti.pg.gda.pl/student/tm/tm14_EEG.pdf. [Data uzyskania dostępu: 19 05 2017].

[164] S. Jankowska, „Potęga umysłu, czyli doping doskonały,” w Neurokognitywistyka w patologii i i zdrowiu., Szczecin, Pomorski Uniwersytet Medyczny w Szczecinie, 2011-2013, p. 99–104.

[165] A. Grabska-Barwińska, Model zmian synchronizacji czynności EEG związanych z wykonywaniem ruchu., Warszawa: Uniwersytet Warszawski, Międzywydziałowe indywidualne studia matematyczno-przyrodnicze, 2004.

[166] P. Wołowik, Zastosowanie sygnału EEG w interfejsach BCI łączących człowieka z komputerem., Poznań: Poznanskie Warsztaty Telekomunikacyjne., 2004, pp. 1-5.

[167] J. Lipiec, Porównanie wyników badań neurofizjologicznych, klinicznych i obrazowych u chorych po częściowym urazie szyjnego i piersiowego rdzenia kręgowego., Poznań: Uniwersytet Medyczny w Poznaniu, 2015.

[168] A. Cudo, E. Zabielska i D. Zapała, „Interfejsy mózg-komputer oparte o techniki elektroencefalograficzne.,” Studia z Psychologii w KUL, tom 18, pp. 195-216, 2012.

[169] G. Pfurtscheller, „EEG event-related desynchronization (ERD) and synchronization (ERS),” Electroencephalography and Clinical Neurophysiology, 1997.

[170] „Event-related desynchronization (ERD) during visual processing,” Event-related desynchronization (ERD) during visual processing, tom 16, nr 2-3, pp. 147-153, 1994.

[171] B. Z. Allison, C. Brunner, C. Altstätter, I. Wagner, S. Grissmann i C. Neuper, „A hybrid ERD/SSVEP BCI for continuous simultaneous two dimensional cursor control,” Journal of Neuroscience Methods, tom 209, nr 2, pp. 299-307, 2012.

[172] C. S. Nam, Y. Jeon, Y.-J. Kim, I. Lee i K. Park, „Movement imagery-related lateralization of event-related (de)synchronization (ERD/ERS): Motor-imagery duration effects.,” Clinical Neurophysiology, tom 122, nr 3, pp. 567-577, 2011.

[173] Y. Jeon, C. S. Nam, Y.-J. Kim i M. C. Whang, „Event-related (De)synchronization (ERD/ERS) during motor imagery tasks: Implications for brain–computer interfaces," International Journal of Industrial Ergonomics, tom 41, nr 5, pp. 428-436, 2011.

[174] G. Pfurtscheller, „Event-related synchronization (ERS): an electrophysiological correlate of cortical areas at rest," Electroencephalography and Clinical Neurophysiology, tom 83, nr 1, pp. 62-69, 1992.

[175] C. Neuper, G. R. Muller i A. Kubler, „Clinical application of an EEG - based brain - computer interface: a case study in a patient with severe motor impairment.," Clinical Neurophysiology., tom 114, nr 3, pp. 399-409, 2003.

[176] B. Obermaier, C. Neuper i C. Guger, „Information transfer rate in a five - classes brain-computer interface.," IEEE Transactions on Neural Systems and Rehabilitation Engineering., tom 9, nr 3, pp. 283-288, 2001.

[177] G. Pfurtscheller i C. Neuper, „Motor imagery and direct brain-computer communication.," Proceedings of the IEEE ., tom 89, nr 7, pp. 1123 - 1134, 2001.

[178] D. Zhang, C. Hansen, F. De Frene, S. P. Kergaard, W. Qian i K. Chen, „Automated labeling and online evaluation for self-paced movement detection BCI," Knowledge-Based Systems, tom 265, 2023.

[179] D. Delisle-Rodriguez, H. L. de Oliveira, J. C. da Silva, M. L. de Souza, T. Bastos, E. M. Nakamura-Palacios i A. Frizera-Neto, „Multi-channel EEG-based BCI using regression and classification methods for attention training by serious game," Biomedical Signal Processing and Control, tom 85, 2023.

[180] D. Zhang, C. Hansen, F. De Frène, S. P. Kargaard, W. Qian i K. Chen, „Automated labeling and online evaluation for self-paced movement detection BCI," Knowledge-Based Systems, tom 265, 2023.

[181] D. Pawar i S. Dhage, „EEG-based covert speech decoding using random rotation extreme learning machine ensemble for intuitive BCI communication," Biomedical Signal Processing and Control, tom 80, nr 2, 2023.

[182] H. G. Niebeling i R. Degen, Einfuhrung in die Elektroencephalograpie., Leipzig: Barth, 1968.

[183] „Czepak do badań EEG," 12 05 2023. [Online]. Available: https://sc04.alicdn.com/kf/Hc57c3bd368cc445b96703e62bc3a3f56j.jpg. [Data uzyskania dostępu: 12 05 2023].

[184] „Położenie elektrod 10-20,” 12 05 2023. [Online]. Available: https://skullbase.bwh.
harvard.edu/wp-content/uploads/2022/03/1.4-10-20-system-composite.jpg. [Data uzyskania dostępu: 12 05 2023].

[185] P. L. Nuneza, R. Srinivasanb, A. F. Westdorpa, R. S. Wijesinghea, D. M. Tuckerb, R. B. Silbersteine i P. J. Cadusche, EEG coherency: I: statistics, reference electrode, volume conduction, Laplacians, cortical imaging, and interpretation at multiple scales., Electroencephalogr Clin Neurophysiol., 1997.

[186] M. Kumngern, F. Khateb, T. Kulej, D. Arbet i M. Akbari, „Fully differential fifth-order dual-notch low-pass filter for portable EEG system,” AEU - International Journal of Electronics and Communications, tom 146, 2022.

[187] M. Sharma, S. Patel i U. R. Acharya, „Automated detection of abnormal EEG signals using localized wavelet filter banks,” Pattern Recognition Letters, tom 133, pp. 188-194, 2020.

[188] S. Zhang, S. Zhu, B. Zhang, B. Feng, T. Yu, Z. Li, Z. Zhang, G. Huang i Z. Liang, „Overall optimization of CSP based on ensemble learning for motor imagery EEG decoding,” Biomedical Signal Processing and Control, tom 77, 2022.

[189] B. A. Coffman, M. A. Hunter, A. P. Jones, H. A. Saxon, K. Kolodjeski, B. Lockmiller, O. Khan, T. Collar, J. M. Stephen i V. P. Clark, „Using independent components analysis (ICA) to remove artifacts associated with transcranial direct current stimulation (tDCS) from electroencephalography (EEG) data: A comparison of ICA algorithms„” Brain Stimulation, tom 7, nr 2, 2014.

[190] C. Stergiadis, V.-D. Kostaridou i M. A. Klados, „Which BSS method separates better the EEG Signals? A comparison of five different algorithms,” Biomedical Signal Processing and Control, tom 72, nr A, 2022.

[191] D. Sunaryono, R. Sarno i J. Siswantoro, „Gradient boosting machines fusion for automatic epilepsy detection from EEG signals based on wavelet features,” Journal of King Saud University - Computer and Information Sciences, tom 34, nr 10B, pp. 9591-9607, 2022.

[192] M. K. Kıymık, I. Huler, A. Dizibuyuk i M. Akin, „Comparison of STFT and wavelet transform methods in determining epileptic seizure activity in EEG signals for real-time application„” Computers in Biology and Medicine, tom 35, nr 7, pp. 603-616, 2005.

[193] Z. Liu, L. Wang, S. Xu i K. Lu, „A multiwavelet-based sparse time-varying autoregressive modeling for motor imagery EEG classification," Computers in Biology and Medicine, tom 155, 2023.

[194] [194] P. Sheoran i J. S. Saini, „A New Method for Automatic Electrooculogram and Eye Blink Artifacts Correction of EEG Signals using CCA and NAPCT," Procedia Computer Science, tom 167, pp. 1761-1770, 2020.

[195] S.-M. Zhou, J. Q. Gan i F. Sepulveda, „Classifying mental tasks based on features of higher-order statistics from EEG signals in brain–computer interface," Information Sciences, tom 178, nr 6, pp. 1629-1640, 2008.

[196] Y. Zhang, W. Chen, C.-L. Lin, Z. Pei, J. Chen i Z. Chen, „Boosting-LDA algriothm with multi-domain feature fusion for motor imagery EEG decoding," Biomedical Signal Processing and Control, tom 70, 2021.

[197] G. Roy, A. K. Bhoi i S. Bhaumik, „A Comparative Approach for MI-Based EEG Signals Classification Using Energy, Power and Entropy," IRBM, tom 43, nr 5, pp. 434-446, 2022.

[198] L. Duan, Z. Hongxin, M. S. Khan i M. Fang, „Recognition of motor imagery tasks for BCI using CSP and chaotic PSO twin SVM," The Journal of China Universities of Posts and Telecommunications, tom 24, nr 3, pp. 83-90, 2017.

[199] W. Chang, W. Meng, G. Yan, B. Zhang, H. Luo, R. Gao i Z. Yang, „Driving EEG based multilayer dynamic brain network analysis for steering process," Expert Systems with Applications, tom 207, 2022.

[200] T. Kayikcioglu i O. Aydemir, „A polynomial fitting and k-NN based approach for improving classification of motor imagery BCI data," Pattern Recognition Letters, tom 31, nr 11, pp. 1207-1215, 2010.

[201] H. Górecki, Optymalizacja systemów dynamicznych., 1993: Wydawnictwo Naukowe PWN, Warszawa.

[202] W. Chmielowski, Zastosowanie optymalizacji w gospodarce wodnej., Kraków: Wydawnictwo Politechniki Krakowskiej, 2005.

[203] J. Stadnicki, Teoria i praktyka rozwiązywania zadań optymalizacji z przykładami zastosowań technicznych., Warszawa: Wydawnictwa Naukowo-Techniczne, 2006.

[204] J. R. Dixon i P. Corrado, Engineering design and design for manufacturing. A structured approach., Field Stone Publishers, 1995.

[205] I. Gościniak, Nowe ujęcie wybranych zagadnień optymalizacji., Katowice: Wydawnictow Uniwersytetu Śląskiego, 2014.

[206] A. Ostanin, Optymalizacja liniowa i nieliniowa., Białystok: Wydawnictwo Politechniki Białostockiej, 2005.

[207] J. Korbicz, K. Patan i A. Obuchowicz, „Dynamie neural networks for process modelling in fault detection and isolation.," International Journal of Applied Mathematics and Computer Science., tom vol. 9 no. 3, pp. 519-546, 1999.

[208] S. Wan i L. Banta, „Parameter incremental learning algorithm for neural networks.," IEEE Transactions on Neural Networks, tom vol. 17 no 6, pp. 1424-1438, 2006.

[209] J. Bilski, „Uczenie nadzorowane wielowarstwowych sieci neuronowych.," w Ewolucja czy rewolucja. Nowoczesne techniki informatyczne., Częstochowa, Politechnika Częstochowska, 2003, pp. 59-101.

[210] S. Kollias i D. Anastassiou, „An adaptive least squares algorithm for the efficient training of artificial neural networks.," IEEE Trans on Circuits Syst., tom vol. 36, pp. 1092-1101, 1989.

[211] B. Wilamowski, N. Cotton, O. Kaynak i G. Dundar, „Computing gradient vector and Jacobian matrix in arbitrarily connected neural networks.," IEEE Transaction Ind. Electron., tom vol. 55, nr no. 10, pp. 3784-3790, 2008.

[212] N. Ampazis i S. Perantonis, „Two highly efficient second-order algorithms for training feedforward networks.," IEEE Transaction on Neural Networks., tom vol. 13, nr no. 5, pp. 1064-1074, 2002.

[213] M. Kordos i W. Duch, „Multilayer perceptron trained with numerical gradient.," Proceedings of the International Conference on Artificial Neural Networks and International Conference on Natural Information Processing, tom Istambul, 2003.

[214] M. Skowronek, Modelowanie cyfrowe. Opis, algorytmy i środki programowe., Gliwice: Wydawnictwo Plitechniki Śląskiej, 2004.

[215] K. M. Gawrylczyk, „Metody_bezgradientowe.pdf," [Online]. Available: http://kmg.zut.edu.pl/ftp/opt/ . [Data uzyskania dostępu: 05 10 2016].

[216] A. Caba, Wielki Słownik Wyrazów Obcych, Kraków: Krakowskie Wydawnictwo Naukowe, 2008.

[217] H. Zgółkowa, Praktyczny Słownik Współczesnej Polszczyzny. Tom 8., Poznań: Wydawnictwo Kurpin, 2003.

[218] B. Dunaj, Słownik współczesnego języka polskiego, Warszawa: Wilga, 1996.

[219] E. Sobol, Podręczny słownik języka polskiego., Warszawa: Wydawnictwo Naukowe PWN, 1996.

[220] E. Sobol, L. Dralik, A. Kabiak–Sokół i L. Wiśniewska, Słownik języka polskiego PWN. Wydanie Nowe., Warszawa: Wydawnictwo Naukowe PWN, 2006.

[221] M. Bańka, Wielki słownik wyrazów obcych PWN, Warszawa: Wydawnictwo Naukowe PWN, 2003.

[222] Wikipedia, „Model_deterministyczny," pl.wikipedia.org, [Online]. Available: https://pl.wikipedia.org/wiki/Model_deterministyczny. [Data uzyskania dostępu: 20 04 2016].

[223] K. Schittkowski, C. Zillober i R. Zotemantel, „Numerical comparison of nonlinear programming algorithms for structural optimization.," Structural Optimization, tom 7, pp. 1-19, 1994.

[224] C. Cichoń i M. Detka, Wybrane zagadnienia programowania liniowego, Kielce: Wydawnictwo Politechniki Świętokrzyskiej, 2004.

[225] J. Forenc, Podstawy informatyki 2 - Wykład nr 9., Białystok: Politechnika Białostocka - Wydział Elektryczny, 2007.

[226] G. Garai i B. B. Chaudhurib, „A novel genetic algorithm for automatic clustering.," Pattern Recognition Letters., tom 25, nr 2, pp. 173-187, 2004.

[227] S. C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists.Third Edition., Natick, MA, USA: McGraw-Hill, 2012.

[228] A. Azaron, C. Perkgoz i M. Sakawa, „A genetic algorithm approach for the time-cost trade-off in PERT networks.," Applied Mathematics and Computation, tom 168, nr 2, p. 1317–1339, 2005.

[229] T. N. Bui i B. R. Moon, „Genetic Algorithm and Graph Partitioning.," IEEE Transactions on Computers., tom 45, nr 7, pp. 841-855, 1996.

[230] M. Zawisza, „Optymalizacja funkcji jednej zmiennej.," Zakład Wspomagania i Analizy Decyzji, Instytut Ekonometrii, Szkoła Główna Handlowa, [Online]. Available: http://akson.sgh.waw.pl/~mz35109/MO/z2/zajecia2.pdf. [Data uzyskania dostępu: 08 04 2016].

[231] M. Lewandowski, „Metody optymalizacji – teoria i wybrane algorytmy," 15 01 2012. [Online]. Available: https://web.sgh.waw.pl/~mlewan1/Site/MO_files/mo_skrypt_21_12.pdf. [Data uzyskania dostępu: 01 02 2017].

[232] W. Findeisen, J. Szymanowski i A. Wierzbicki, Metody obliczeniowe optymalizacji., Warszawa: Wydawmictwa Politechniki Warszawskiej, 1973.

[233] K. Amborski, Podstawy metod optymalizacji., Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej., 2009.

[234] J. Bralczyk, Słownik 100 tysięcy potrzebnych słów, Warszawa: Wydawnictwo Naukowe PWN, 2005.

[235] A. Stankiewicz, L. Wiśniakowska, B. Wróbel i B. Pakosz, Słownik wyrazów obcych. Wydanie Nowe., Warszawa: Wydawnictwo Naukowe PWN, 1996.

[236] A. Trzesimiech, Układy stochastyczne., Uniwersytet Śląski: Instytut Informatyki, 2009.

[237] B. Kuipers, Qualitative Reasoning. Modeling and Simulation with Incomplete Knowledge, Cambridge, Massachusetts, London, England: The MIT Press, 1994.

[238] T. Masters, Sieci Neuronowe w Praktyce., Warszawa: Wydawnictwa Naukowo-Techniczne, 1996.

[239] W. H. Press, B. Flannery, S. Teukolsky i W. Vetterling, Numerical Recipes in C., New York: Cambridge U niversity Press, 1988.

[240] D. Knuth, Seminumerical Algorithms, Massachusetts: Addison-Wesley Rading, 1981.

[241] E. Aarts i P. Laarhoven, Simulated Annealing: Theory and Practice., New York: John Wiley and Sons, 1987.

[242] R. Azencott, Simulated Annealing: Parallelization Techniąues., New York: John Wiley and Sons, 1992.

[243] M. A. Stybliński i T.-S. Tang, „Experiment in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing.," Neural Networks, nr 3, pp. 467-483, 1990.

[244] J. Korbicz, A. Obuchowicz i D. Uciński, Sztuczne Sieci Neuronowe. Podstawy i Zastosowania., Warszawa: Akademicka Oficyna Wydawnicza PLJ, 1994.

[245] A. Smoliński, „Klasyfikacja i porównanie metod klasycznej optymalizacji wielokryterialnej.," Informatyczne Systemy Zarzadzania., tom 5, pp. 117-133, 2009.

[246] P. Chołda, Programowanie dynamiczne i optymalizacja wielokryterialna., Kraków.: Akademia Górniczo-Hutnicza w Krakowie, 2016.

[247] R. Ryńca i D. Kuchta, „Propozycja integracji programowania celowego i zrównoważonej karty wyników – ujęcie modelowe i studium przypadku," Zeszyty Naukowe Uniwersytetu Szczecińskiego., tom 73, nr 854, pp. 239-251, 2015.

[248] P. Krzemień i A. Gajek, „Wpływ postaci funkcji jakości oraz wag kryteriów cząstkowych na wyniki optymalizacji zderzenia metodą genetyczną.,” Czasopismo Techniczne Politechniki Krakowskiej., tom 10, nr 109, pp. 173-183, 2012.

[249] Z. Osiński i J. Wróbel, Teoria konstrukcji maszyn., Warszawa: PWN, 1995.

[250] W. Tarnowski, Podstawy projektowania technicznego., Warszawa: WNT, 1997.

[251] A. Osyczka, Evolutionary algorithms for a single and multicriteria design optimization., Heidelberg, Physica-Verlag, 2002.

[252] J. Malczewski, „Ordered weighted averaging with fuzzy quantifiers: GIS-based multicriteria evaluation for land-use suitability analysis.,” International Journal of Applied Earth Observation and Geoinformation., tom 8, pp. 270-277, 2006.

[253] P. V. Gorsevski, K. R. Donevska, C. D. Mitrovski i J. P. Frizado, „Integrating multi-criteria evaluation techniques with geographic information systems for landfill site selection: A case study using ordered weighted average.,” Waste Management, tom 32, pp. 287-296, 2012.

[254] R. R. Yager, „On ordered weighted averaging aggregation operators in multicriteria decision making.,” IEEE Transactions on Systems, Man, and Cybernetics, tom 8, pp. 183-190, 1988.

[255] H. Jiang i R. J. Eastman, „Application of fuzzy measures in multi-criteria evaluation in GIS,” International Journal of Geographical Information Systems., tom 14, pp. 173-184, 2000.

[256] B. S. Ahn, „Compatible weighting method with rank order centroid: Maximum entropy ordered weighted averaging approach.,” European Journal of Operational Research., tom 212, nr 3, pp. 552-559, 2011.

[257] Z. Sekulski, Wybrane problemy optymalizacji wielokryterialnej we wstępnym projektowaniu konstrukcji kadłuba statków morskich., Szczecin: Zachodniopomorski Uniwersytet Technologiczny w Szczecinie., 2012.

[258] E. Roszkowska, „Rank ordering criteria weighting methods – a comparative overview.,” Optimum. Studia Ekonomiczne., tom 5, nr 56, pp. 14-33, 2013.

[259] C. K. Makropoulos i D. Butler, „Spatial ordered weighted averaging: incorporating spatially variable attitude towards risk in spatial multi-criteria decision-making.,” Environmental Modelling & Software., tom 21, pp. 69-84, 2006.

[260] J.-R. Chou, „A Weighted linear combination ranking technique for multi-criteria decision analysis.,” South African Journal of Economic and Management Sciences., tom 16, nr 5, 2013.

[261] R. Miłaszewski i J. Śleszyński, „Zastosowanie programowania celowego do modelowania działalności inwestycyjnej w zakresie ochrony wód.,” Ekonomia i Środowisko, nr 2, pp. 103-115, 1992.

[262] K. Krupińska, „Zalety i wady modelu programowania celowego.,” Prace Naukowe Akademii Ekonomicznej we Wrocławiu. Ekonometria., tom 11, nr 981, pp. 156-173, 2003.

[263] W. Bura, Wielokryterialne, mrowiskowe algorytmy optymalizacji w nawigacji samochodowej., Sosnowiec: Uniwersytet Śląski, Wydział Informatyki i Nauki o Materiałach., 2014.

[264] G. Xiong, Q. Pan, H. Zhang i N. Ji, „A Satisfactory Coordination Solution of Multiple-Person Cooperative Game.,” International Conference on Wireless Communications, Networking and Mobile Computing., 2007.

[265] T. Błaszczyk i M. Nowak, „Wielokryterialne wspomaganie decyzji w harmonogramowaniu projektów.,” Decyzje., tom 13, pp. 27-54, 2010.

[266] W. Ogryczak, Wielokryterialna optymalizacja liniowa i dyskretna; modele preferencji i zastosowania do wspomagania decyzji, Warszawa: Instytut Informatyki, Uniwersytet Warszawski., 1997.

[267] „Porządek leksykograficzny.,” Wikipedia., [Online]. Available: https://pl.wikipedia.org/wiki/Porządek_leksykograficzny. [Data uzyskania dostępu: 14 07 2017].

[268] C. Schneeweis, „Hierarchical structures in organisations: A conceptual framework .,” European Journal of Operational Research., tom 85, pp. 4-31, 1995.

[269] W. Ogryczak i T. Śliwiński, „On solving linear programs with the ordered weighted averaging objective.,” European Journal of Operational Research., tom 148, pp. 80-91, 2003.

[270] A. Giarlotta, „Multicriteria compensability analysis.,” European Journal of Operational Research., tom 133, p. 190–209, 2001.

[271] W. Ogryczak i T. Śliwiński, „Lexicographic Max-Min Optimization for Efficient and Fair Bandwidth Allocation.,” [Online]. Available: https://www.euro-online.org/enog/inoc2007/Papers/author.100/paper/paper.100.pdf. [Data uzyskania dostępu: 15 07 2017].

[272] R. M. Salles i J. A. Barria, „Lexicographic Maximin Optimisation for Fair Bandwidth Allocation in Computer Networks.,” [Online]. Available: https://pdfs.semanticscholar.org/4875/83c1a64572006bfc468e72e6ef574f7cdd3d.pdf. [Data uzyskania dostępu: 15 07 2017].

[273] „Lexicographical order,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Lexicographical_order. [Data uzyskania dostępu: 14 07 2017].

[274] R. T. Marler i J. S. Arora, „Survey of multi-objective optimization methods for engineering.,” Structural and Multidisciplinary Optimization., tom 26, pp. 369-395, 2004.

[275] L. M. Kirilov i N. B. Dokev, „A Lexicographic Ordering Approach in Multiple CriUeria Decision Support.,” First International IEEE Symposium "Intelligent Systems"., pp. 314-319, 2002.

[276] I. Yevseyeva, K. Miettinen i P. Rasanen, „Verbal ordinal classification with multicriteria decision aiding.,” European Journal of Operational Research., tom 185, p. 964–983, 2008.

[277] K.-W. Jee, D. L. McShan i B. A. Fraass, „Lexicographic ordering: intuitive multicriteria optimization for IMRT.,” Physics in Medicine and Biology., tom 52, p. 1845–1861, 2007.

[278] K. Miettinen, Nonlinear Multiobjective Optimization., Boston, MA.: Kluwer, 1999.

[279] A. Ameljańczyk, „Metoda podziału zbioru obiektów na wielokryterialne klastry jakościowe.,” Biuletyn Instytutu Systemów Informatycznych., tom 12, pp. 1-7, 2013.

[280] „Platforma e-learningowa SGH.,” [Online]. Available: https://www.e-sgh.pl/niezbednik/plik.php?id=27244571&pid=1797. [Data uzyskania dostępu: 15 07 2017].

[281] P. C. Fishburn, „Exceptional Paper - Lexicographic Orders, Utilities and Decision Rules: A Survey.,” Management Science, tom 20, nr 11, pp. 1442-1471, 1974.

[282] N. H. Zardari, K. Ahmed, S. M. Shirazi i Z. B. Yusop, Weighting Methods and their Effects on Multi-Criteria Decision Making Model Outcomes in Water Resources Management., Springe rBriefs in Water Science and Technology., 2014.

[283] Y. Collette i P. Siarry, Multiobjective optimization., Springer Verlag, 2003.

[284] L. Ladha i T. Deepa, „Feature Selection Methods and Algorithms.,” International Journal on Computer Science and Engineering (IJCSE), tom 3, nr 5, pp. 1787-1797, 2011.

[285] K. Mączka, Metody filtrów dla danych finansowych i bio-medycznych., Katowice: Politechnika Śląska, 2005.

[286] J. Brownlee, „An Introduction to Feature Selection.,” w Machine Learning Process, 2014.

[287] M. Dash i H. Liu, „Feature selection for classification,” Intelligent Data Analysis, tom Volume 4, nr Issues 1-4, pp. 131-156, 1997.

[288] B. Raman i T. R. Ioerger, Enhancing Learning using Feature and Example selection., Texas A&M University, 2003.

[289] B. Kostek i P. Szczuko, Sztuczna inteligencja w medycynie (Skrypt do przedmiotu)., Gdańsk: Politechnika Gdańska, 2015.

[290] J. Bala, K. De Jong, Huang J., H. Vafaie i H. Wechsler, „Visual Routine For Eye Detection Using Hybrid Genetic Architectures.,” w Proceedings of the International Conference on Pattern Recognition., Vienna, Austria, 1996.

[291] R. Caruana i D. Freitag, „Greedy attribute selection.,” w In Proc. of 11th International Machine Learning Conference , New Brunswick, New York, Morgan Kaufmann, 1994, pp. 28-36.

[292] M. Dong i R. Kothari, Feature Subset Selection Using a New Definition of Classifiability, Indian Institute of Technology, 2003.

[293] D. Koller i M. Sahami, Toward Optimal Feature Selection., Computer Science Department Stanford University, 2006.

[294] M. Komori, H. Abe i T. Yamaguthi, „A New Feature Selection Method Based on Dynamic Incremental Extension of Seed Features.,” w Faculty of Information., Shizuoka University, 2002.

[295] H. Kwaśnicka i K. Michalak, „Correlation based Feature Selection Strategy in Classification Problems.,” International Journal of Applied Mathematics and Computer Science - Technical University in Zielona Góra, tom 16, nr 4, pp. 503-511, 2006.

[296] H. Kwaśnicka i K. Michalak, „Correlation based Feature Selection Strategy in Neural Classification.,” w IEEE Computer Society, Los Alamitos, Washington, Tokyo, 2006.

[297] P. Langley, Selection of Relevant Features in Machine Learning, Pat Langley., Institute for the Study of Learning and Expertise., 1994.

[298] E. Pekalska, A. Harol, C. Lai i R. P. Duin, „Pairwise Selection of Features and Prototypes.," w Computer Recognition Systems. in Proceeding of 4th International Conference on Computer Recognition Systems - CORES'2005, 2005.

[299] Z. Sun, G. Bebis, X. Yuan i S. J. Louis, Genetic feature Subset selection for gender Classification: A Comparison Study., Nevada: University of Nevada, 2002.

[300] H. Vafaie i K. De Jong, Genetic Algorithms as a Tool for Feature Selection in Machine Learning., Center for Artificial Intelligence George Mason University., 1992.

[301] H. Vafaie i K. De Jong, Genetic Algorithms as a Tool for Restructuring Feature Space Representations., Center for Artificial Intelligence George Mason University., 1995.

[302] H. Vafaie i K. De Jong, Improvising A Rule Induction System Using Genetic Algorithms., Center for Artificial Intelligence George Mason University., 1994.

[303] H. Vafaie i K. De Jong, Robust Feature Selection Algorithm., Center for Artificial Intelligence George Mason University., 1993.

[304] O. Uncu i I. B. Turksen, „A novel feature selection approach: Combining feature wrappers and filters.," Information Sciences, tom 177, p. 449–466, 2007.

[305] L. Talavera, „An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering.," w Advances in Intelligent Data Analysis VI., Berlin Heidelberg, Springer-Verlag, 2005, pp. 440-451.

[306] H. Kwaśnicka i P. Orski, „Genetic Algorithm as an Attributes Selection Tool for Learning Algorithms.," w Intelligent Information Systems 2004, New Trends in Intelligent Information Processing and Web Mining, Proceedings of the Intelligent Information Systems: Intelligent Information Processing and Web Mining 2004 Conference, Springer, 2004, pp. 449-453.

[307] K. Szelągowski, Interfejs WWW dla biblioteki Infosel++., Dąbrowa Górnicza: Wyższa Szkoła Biznesu w Dąbrowie Górniczej., 2010.

[308] J. Reunanen, Search strategies for wrapper methods In: Feature extraction, foundations and applications., Springer, 2006.

[309] P. Devijver i J. Kitler, Pattern Recognition: A Statistical Approach, 1982.

[310] A. Chouchoulas, Incremental Feature Selection Based on Rough Set Theory., Edinburgh: Centre for Intelligent Systems and their Applications, Division of Informatics, The University of Edinburgh, 2001.

[311] L. C. Molina, L. Belanche i À. Nebot, „Feature Selection Algorithms: A Survey and Experimental Evaluation," w Data Mining, 2002. ICDM 2003. Proceedings, Barcelona, Spain, 2002.

[312] H. Liu i R. Setiono, „Feature Selection for Classification: a Probabilistic Wrapper Approach.," w 9th International Conference on Industrial and Engineering Applications of AI and ES., Morgan Kaufmann, 1996, pp. 129-135.

[313] H. Liu, H. Motoda i M. Dash, „A Monotonic Measure for Optimal Feature Selection," w Department of Information Systems and Computer Science, National University of Singapore, Singapore, 1998.

[314] R. Kovahi i G. H. John, „Wrappers for Features Subset Selection, Data Mining and Visualization Silicon Graphics.," w Inc. Global Business Intelligent Solutions, Almaden Research Center, 1997.

[315] I. Guyon i A. Elisseeff, „An Introduction to Variable and Feature Selection," Journal of Machine Learning Research, tom 3, 2003.

[316] K. Mehmed, Data Mining: Concepts, Models, Methods, and Algorithms, 2nd Edition., Wiley-IEEE Press, 2011.

[317] P. Yang, B. B. Zhou, Z. Zhang i A. Y. Zomaya, „A multi-filter enhanced genetic ensemble system for gene selection and sample classification of microarray data.," BMC Bioinformatics, tom 11, 2010.

[318] H. Liu i H. Motoda, „Feature Extraction, Construction, and Selection," A Data Mining Perspective, p. ISBN: 079238198X, 1998.

[319] R. Setiono i H. Liu, „Chi2: Feature Selection And Discretization Of Numeric Attributes.," Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, nr 1082-3409/95 $04.00 © 1995 IEEE, pp. 388-391, 1995.

[320] K. Torkkola, „Information-theoretic methods for feature selection and construction.," Feature Extraction, Foundations and Applications., nr Springer, 2005.

[321] G. Forman, „An extensive empirical study of feature selection metrics for text classification.," Journal of Machine Learning Research, tom 3, pp. 1289-1305, 2003.

[322] W. Duch, „Filter methods In: Feature extraction, foundations and applications.,” w Studies in Fuzziness and Soft Computing., Berlin Heidelberg, Springer-Verlag, 2006, p. 89–117.

[323] L. Yu, H. Liu i I. Guyon, „Efficient feature selection via analysis of relevance and redundancy.,” Journal of Machine Learning Research, tom 5, p. 1205–1224, 2004.

[324] M. Blachnik, „Metody selekcji cech.,” [Online]. Available: http://www.mblachnik.pl/lib/exe/fetch.php/dydaktyka/zajecia/wyklady/metody_selekcji_cech.pdf. [Data uzyskania dostępu: 05 04 2017].

[325] M. James, Pattern recognition., New York, USA: Wiley-Interscience ©1988.

[326] „Wspolczynnik korelacji R Pearsona,” [Online]. Available: http://www.naukowiec.org/wzory/statystyka/wspolczynnik-korelacji-r-pearsona_22.html. [Data uzyskania dostępu: 13 02 2017].

[327] „Odchylenie Standardowe,” [Online]. Available: http://www.naukowiec.org/wzory/statystyka/odchylenie-standardowe_5.html. [Data uzyskania dostępu: 13 02 2017].

[328] S. Chikhi i S. Benhammada, „ReliefMSS: a variation on a feature ranking ReliefF algorithm.,” International Journal of Business Intelligence and Data Mining., tom 4(3), pp. 375-390, 2009.

[329] L. H. Patil i M. Atique, „A Multistage Feature Selection Model for Document Classification Using Information Gain and Rough Set.,” International Journal of Advanced Research in Artificial Intelligence (IJARAI), tom 3, nr 11, pp. 14-20, 2014.

[330] T. A. Alhaj, M. M. Siraj, A. Zainal, H. T. Elshoush i F. Elhaj, „Feature Selection Using Information Gain for Improved Structural-Based Alert Correlation.,” PLoS ONE, tom 11, nr 11, pp. 1-18, 2016.

[331] C. M. Lai, W. C. Yeh i C. Y. Chang, „Gene selection using information gain and improved simplified swarm optimization.,” Neurocomputing., tom 218, pp. 331-338, 2016.

[332] G. Forman, „An Extensive Empirical Study of Feature Selection Metrics for Text Classification,” Journal of Machine Learning Research, tom 3, pp. 1289-1305, 2003.

[333] Y. Yang i J. O. Pedersen, „A Comparative Study on Feature Selection in Text Categorization.,” w International Conference on Machine Learning (ICML)., Morgan Kaufmann Publishers, 1997, pp. 412--420.

[334] A. G. Karegowda, A. S. Manjunath i M. A. Jayaram, „Comparative study of attribute selection using gain ratio and correlation based feature selection.,” International Journal of Information Technology and Knowledge Management, tom 2, nr 2, p. 271–277, 2010.

[335] O. Villacampa, Feature Selection and Classific ation Methods for Decision Making: A Comparative Analysis., Nova Southeastern: College of Engineering and Computing at Nova Southeastern University, 2015.

[336] Z. Zheng, X. Wu i R. Srihari, „Future Selection for Text Catogarization on Imbalanced Data.,” ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets., tom 6, nr 1, pp. 80-89, 2004.

[337] B. Sui, Information Gain Feature Selection Based On Feature Interactions., Houston: University of Houston, Computer Science, College of Natural Sciences and Mathematics., 2013.

[338] B. Azhagusundari i A. S. Thanamani, „Feature Selection based on Information Gain.,” International Journal of Innovative Technology and Exploring Engineering (IJITEE)., tom 2, nr 2, pp. 18-21, 2013.

[339] D. Roobaert, G. Karakoulas i N. V. Chawla, „Information Gain, Correlation and Support Vector Machines.,” w StudFuzz 207, Berlin Heidelberg, Springer-Verlag, 2006, p. 463–470.

[340] R. B. Pereira, A. Plastino, B. Zadrozny i L. H. Merschmann, „Information Gain Feature Selection for Multi-Label Classification.,” Journal of Information and Data Management., tom 6, nr 1, pp. 48-58, 2015.

[341] F. Aiolli, Sistemi Informativi, Dip. di Matematica, Pura ed Applicata, 2007.

[342] C. Lee i G. G. Lee, „Information gain and divergence-based feature selection for machine learning-based text categorization.,” Information Processing and Management, tom 42, p. 155–165, 2006.

[343] J. Xu i H. Jiang, „An Improved Information Gain Feature Selection Algorithm for SVM Text Classifier.,” w CYBERC '15 Proceedings of the 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Washington, DC, USA, IEEE Computer Society, 2015, pp. 273-276.

[344] S. Lei, „A Feature Selection Method Based on Information Gain and Genetic Algorithm.,” w ICCSEE '12 Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering - Volume 02, Washington, DC, USA, IEEE Computer Society, 2012, pp. 355-358.

[345] D. Mladeni, Machine Learning on non-homogeneous, distributed text data., Slovenia: University of Ljubljana., 1998.

[346] D. Mladeni i G. Marko, „Feture selection for unbalanced class distribution and naive bayes.,” The Sixteenth International Conference on Machine Learning, p. 258–267, 1999.

[347] H. Ng, W. Goh i K. Low, „Feature selection, perceptron learning, and a usability case study for text categorization.,” ACM SIGIR Conference on Research and Developement in Information Retrieval., p. 67–73, 1997.

[348] F. Sebastiani, „Machine learning in automated text categorization.,” ACM Computing Surveys, tom 34, nr 1, pp. 1-47, 2002.

[349] H. Liu i R. Setiono, „A Probabilistic Approach to Feature Selection: a Filter Solution.,” 13th Int. Conference on Machine Learning (ICML)., p. 319–327, 1996.

[350] P. Ziemba i R. Budziński, „Selekcja kryteriów oceny jakości serwisów internetowych z wykorzystaniem pojemności informacyjnej Hellwiga.,” Metody ilościowe w badaniach ekonomicznych., tom 13, nr 3, pp. 302-311, 2012.

[351] E. E. Ghiselli, Theory of Psychological Measurement., New York: McGraw-Hill Education; First Edition first Printing edition, 1964.

[352] M. Dash, H. Liu i H. Motoda, „Consistency Based Feature Selection.,” w PADKK '00 Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications., London, UK, Springer-Verlag, 2000, pp. 98-109.

[353] G. Holmes i C. G. Nevill-Manning, „Feature Selection via The Discovery of Simple Classification Rules.,” In Proceedings of the International Symposium on Intelligent Data Analysis., 1995.

[354] M. Dash i H. Liu, „Consistency-based search in feature selection.,” Artificial Intelligence., tom 151, p. 155–176, 2003.

[355] R. Ruiz, J. C. Riquelme i J. S. Aguilar-Ruiz, „Heuristic Search over a Ranking for Feature Selection.,” w IWANN'05 Proceedings of the 8th international conference

on Artificial Neural Networks: computational Intelligence and Bioinspired Systems., Berlin, Heidelberg, Springer-Verlag, 2005, pp. 742-749.

[356] U. Maulik, „Fuzzy Preference Based Feature Selection and Semisupervised SVM for Cancer Classification.," IEEE Transactions On Nanobioscience, tom 13, nr 2, pp. 152-160, 2014.

[357] U. Maulik, „Gene-Expression-Based Cancer Subtypes Prediction Through Feature Selection and Transductive SVM.," IEEE Transactions On Nanobioscience, tom 60, nr 4, pp. 1111-1117, 2013.

[358] K. M. Shazzad i J. S. Park, „Optimization of Intrusion Detection through Fast Hybrid Feature Selection.," Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)., 2005.

[359] J. S. Park, K. M. Shazzad i D. S. Kim, „Toward Modeling Lightweight Intrusion Detection System Through Correlation-Based Hybrid Feature Selection.," w International Conference on Information Security and Cryptology (CISC), Berlin Heidelberg, Springer-Verlag, 2005, pp. 279-289.

[360] Y. Chen, Y. Li, X. Q. Cheng i L. Guo, „Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System.," w Inscrypt'06 Proceedings of the Second SKLOIS conference on Information Security and Cryptology., Berlin, Heidelberg, Springer-Verlag, 2009, pp. 153-167.

[361] P. Lin, „A Framework for Consistency Based Feature.," w Masters The ses & Specialist Projects, Western Kentucky University., 2009.

[362] K. Shin, D. Fernandes i S. Miyazaki, „Consistency Measures for Feature Selection: A Formal Definition, Relative Sensitivity Comparison and a Fast Algorithm.," w IJCAI'11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two , Barcelona, Catalonia, Spain, AAAI Press, 2011, pp. 1491-1497.

[363] A. Gupta, R. K. Agrawal i B. Kaur, „Performance enhancement of mental task classification using EEG signal: a study of multivariate feature selection methods," Methodologies And Application, Soft Computing, tom 19, p. 2799–2812, 2015.

[364] R. C. Holte, „Very Simple Classification Rules Perform Well on Most Commonly Used Datasets.," Machine Learning, tom 11, nr 1, p. 63–90, 1993.

[365] S. J. Cunningham i B. Summers, „Applying Machine Learning to Subject Classification and Subject Description for Information Retrieval.," w ANNES '95 Proceedings of the 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems., Washington, DC, USA, IEEE Computer Society, 1995.

[366] P. Ziemba, „Zastosowanie korelacji i metody wielokryterialnego podejmowania decyzji przy doborze kryteriów oceny serwisów internetowych.," Ekonometria., tom 34, pp. 154-164., 2011.

[367] I. Guyon, J. Weston, S. Barnhill i V. Vapnik, „Gene selection for cancer classification using support vector machines.," Machine Learning, tom 46(1/3), pp. 389-422, 2002.

[368] O. Chapelle i S. Keerthi, „Multi-Class Feature Selection with Support Vector Machines.," w In Proceedings of the American Statistical Association., 2008.

[369] S. Maldonado, R. Weber i J. Basak, „Simultaneous feature selection and classification using kernel-penalized support vector machines," Information Sciences., tom 181, pp. 115-128, 2011.

[370] A. I. Blum i P. Langley, „Selection of relevant features and examples in machine learning.," w Artificial Intelligence, 1997.

[371] A. Y. Ng, „Feature selection, L1 vs. L2 regularization, and rotation invariance.," w Proceedings of the 21st International Conference on Machine Learning., Banff, Canada, The International Machine Learning Society, 2004, pp. 78-86.

[372] M. Kubus, „Rekurencyjna eliminacja cech w metodach dyskryminacji.," w Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu nr. 384. Taksonomia 24. Klasyfikacja i analiza danych – teoria i zastosowania, Wrocław, Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu, 2015, pp. 154-162.

[373] „Elastic net regularization.," Elastic net regularization., [Online]. Available: https://en.wikipedia.org/wiki/Elastic_net_regularization. [Data uzyskania dostępu: 19 04 2017].

[374] H. Zou i T. Hastie, „Regularization and variable selection via the elastic net.," Journal of the Royal Statistical Society: Series B (Statistical Methodology)., tom 67, nr 2, p. 301–320, 2005.

[375] „Lasso and Elastic Net," Lasso and Elastic Net, [Online]. Available: https://www.mathworks.com/help/stats/lasso-and-elastic-

net.html?requestedDomain=www.mathworks.com. [Data uzyskania dostępu: 05 04 2017].

[376] M. Plechawska-Wójcik, „Biological interpretation of the most informative peaks in the task of mass spectrometry data classification.," Studia Informatica, tom 32, nr 2A (96), pp. 213-228, 2011.

[377] S. Barnhill, V. Vapnik, I. Guyon i J. Weston, „Gene selection for cancer classification using support vector machines.," Machine Learning., pp. 389-422, 2002.

[378] „Feature selection with SVM-RFE," Feature selection with SVM-RFE, [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/50701-feature-selection-with-svm-rfe. [Data uzyskania dostępu: 19 04 2017].

[379] J. Trzęsiok, „Nieklasyczne metody regresji a problem odporności.," w Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu Nr 385. Taksonomia 25. Klasyfikacja i analiza danych – teoria i zastosowania., Wrocław, Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu., 2015, pp. 296-304.

[380] P. Breheny, „Ridge Regression.," [Online]. Available: http://web.as.uky.edu/statistics/users/pbreheny/764-f11/notes/9-1.pdf. [Data uzyskania dostępu: 05 04 2017].

[381] J. Trzęsiok, „Porównanie zdolności predykcyjnych modelu regresji grzbietowej z wybranymi nieparametrycznymi modelami regresji.," w Studia Ekonomiczne nr 191 Zastosowanie metod matematycznych w ekonomii i zarzadzaniu., Katowice, Uniwersytet Ekonomiczny w Katowicach., 2014, pp. 65-74 .

[382] „Zmniejszanie wymiarowości przestrzeni poprzez selekcje cech.," Zmniejszanie wymiarowości przestrzeni poprzez selekcje cech., [Online]. Available: http://books.icse.us.edu.pl/runestone/static/ai/MetodyRedukcjiWymiarowosciPrzestrzeniCech/ZmniejszanieWymiarowosciPrzestrzeniPoprzezSelekcjeCech.html. [Data uzyskania dostępu: 24 03 2017].

[383] „Metody redukcji wymiarowości przestrzeni cech.," Metody redukcji wymiarowości przestrzeni cech., [Online]. Available: http://books.icse.us.edu.pl/runestone/static/ai/MetodyRedukcjiWymiarowosciPrzestrzeniCech/Wstep.html. [Data uzyskania dostępu: 24 03 2017].

[384] P. Włodarczyk, Wizualizacja danych, Toruń: Wydział Matematyki i Informatyki, Uniwersytet Mikołaja Kopernika w Toruniu, 2007.

[385] U. Libal, Redukcja wymiaru - metoda PCA., Wrocław: Politechnika Wrocławska, 2015.

[386] „Metoda składowych głównych (PCA),” [Online]. Available: http://books.icse.us.edu.pl/runestone/static/ai/MetodyRedukcjiWymiarowosciPrzestrzeniCech/ZmniejszanieWymiarowosciPrzestrzeniPoprzezEkstrakcjeCech.html#metoda-skladowych-glownych-pca. [Data uzyskania dostępu: 08 03 2017].

[387] I. T. Jolliffe, Principal Component Analysis. Second Edition., DOI: 10.1007/b98835 red., Springer, 2002.

[388] J. Koronacki i J. Ćwik, Statystyczne systemy uczące się., Warszawa: Wydawnictwo Naukowo - Techniczne., 2005.

[389] W. W. Cooley i P. R. Lohnes, Multivariate Data Analysis., New York: John Wiley & Sons Inc., 1971.

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

[391] „Analiza głównych składowych,” Wikipedia, wolna encyklopedia, [Online]. Available: https://pl.wikipedia.org/wiki/Analiza_głównych_składowych. [Data uzyskania dostępu: 02 03 2017].

[392] A. Webb, Statistical Pattern Recognition. Second Edition., ISBNs: 0-470-84513-9 red., John Wiley & Sons, LTD., 2002.

[393] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Alabama: The University of Alabama, 1989.

[394] H. Kwaśnicka, Obliczenia ewolucyjne w sztucznej inteligencji., Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej, 1999.

[395] D. B. Fogel, Evolving Artificial Intelligence, San Diego: University of California, 1992.

[396] R. Galar, Miękka selekcja w losowej adaptacji globalnej w "R". Próba biocybernetycznego ujęcia rozwoju, Wrocław: Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej Nr 84, 1990.

[397] M. Rutkowska, L. Piliński i L. Rutkowski, Sieci neuronowe, algorytmy genetyczne, systemy rozmyte, Warszawa: PWN, 1997.

[398] P. Cichosz, Systemy uczące się, Warszawa: Wydawnictwa Naukowo-Techniczne, 2006.

[399] L. J. Fogel, Artificial Intelligence Through Sumulated Evolution., Chichester, UK.: John Wiley, 1996.

[400] G. Wilczewski, „Algorytmy ewolucyjne - Wprowadzenie,” 2005. [Online]. Available: http://zasada.zut.edu.pl/ag/Wilczewski.pdf; www.mat.uni.torun.pl/~gracjan. [Data uzyskania dostępu: 30 12 2016].

[401] D. Ashlock, Evolutionary Computation for Modeling and Optimization, Guelph, Ontario: Springer, ISBN 0-387-22196-4, 2006.

[402] D. A. Peterson, J. N. Knight, M. J. Kirby, C. W. Anderson i M. H. Thaut, „Feature Selection and Blind Source Separation in an EEG-Based Brain-Computer Interface.,” EURASIP Journal on Applied Signal Processing, nr 19, pp. 3128-3140, 2005.

[403] M. Kołodziej, A. Majakowski i J. R. Rak, „A new Method of EEG Classification for BCI with Feature Extraction Based on Higher Order Statistics of Wavelet Components and Selection with Genetic Algorithms.,” ICANNGA, pp. 280-289, 2011.

[404] D. E. Goldberg, Optimal initial population size for binary-coded genetic algorithms., University of Alabama, 1985.

[405] D. E. Goldberg, „The genetic algorithm approach: Why, how, and what next?,” w Adaptive and learning systems: Theory and Applications, New York, Plenum Press, 1986, pp. 247-253.

[406] D. E. Goldberg, „Simple genetic algorithms and the minimal deceptive problems.,” w Genetic algorithms and simulated annealing,, London, Pitman, 1987, pp. 74-88.

[407] D. E. Goldberg i C. H. Kuo, „Genetic algorithms in pipeline optimization.,” Journal of Computers in Civil Engineering., tom 1(2), pp. 128-141, 1987.

[408] D. E. Goldberg, SGA: A simple genetic algorithm., University of Michigan, 1982.

[409] D. E. Goldberg, „Computer-aided pipeline operation using genetic algorithms and rule learning.,” w API Pipeline Cybernetics Symposium, Houston, 1984.

[410] D. E. Goldberg i M. P. Samtani, „Engineering optimization via genetic algorithm.,” w Proceedings of the 9th Conference on Elelctronic Computation., pp. 471-482, 1986.

[411] D. E. Goldberg, „Controlling dynamics systems with genetic algorithms and rule learning.,” w Proceedngs of the 4th Yale Workshop on Applicatioons of Adaptive Systems Theory., pp. 91-97, 1985.

[412] D. E. Goldberg i J. Richardson, „Genetic algorithms with sharing for multimodalfunction optimization.,” w Genetic algorithms nad their applications:

Proceedings of 2nd International Conference on Gentic Algorithms. , pp.41-49, 1987.

[413] D. E. Goldberg i R. E. Smith, „Nonstationery function optionsimization using genetic algorithms with dominance and diploidy.," w Genetic algorithms nad their applications: Proceedings of the 2nd Internetional Conference in Genetic Algorithms., pp.59-68., 1987.

[414] D. E. Goldberg, „Dynamic system control using rule learning and genetic algorithms.," w Proceedings of the 9th Internetional Jount Conference on Artificial Intelligence, pp. 588-592, 1985.

[415] D. E. Goldberg, „A tale of two problems: Broad and efficient optimization using genetic algorithms.," w Proceedings of the 1986 Summer Computer Simulation Conference, pp. 44-48, 1986.

[416] D. E. Goldberg i A. L. Thomas, Genetic algorithms: A bibliography 1962-1986, University of Alabama, 1986.

[417] J. H. Holland, „Genetic algorithms and the optimal allocations of trials.," w SIAM Journal of Computing., pp. 88-105., 1973.

[418] M. D. Vose, The Simple Genetic Algorithm: Foundation and Theory., Cambridge: MIT Press, 1999.

[419] J. R. Sampson i A. Brindle, „Genetic algorithms for function optimization," w Proceedings of the 9th Minitoba Conference on Numerical Mathematics and Computing, pp. 31-47, 1979.

[420] J. H. Holland, „Genetic algorithms and classifier systems: Foundations and futur directions.," Genetic algorithms and their applications: Proceedings of the 2nd International Conference on Genetic Algorithms, pp. pp.82-87, 1987.

[421] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, New York: Addison-Wesley Publishing Company, Inc., 1989.

[422] J. H. Holland, „Genetic algorithms and adaptation.," w Technikal Report No. 34., University of Michigan, Departament of Computer and Communication Sciences., 1981.

[423] J. H. Holland, Adaptation in Natural and Artificial Systems., University of Michigan Press, 1975.

[424] G. E. Liepins i M. R. Hilliard, „Genetic algorithms as a paradigm for machine learning," w ORSA/TIMS Joint Natural Meeting, Miami, 1986.

[425] A. K. Minga, „Genetic algorithms in aerospace design.,” w AIAA Sutheastern Regional Student Conference, Huntsville, 1986.

[426] D. Quagliarella, J. Periaux, C. Poloni i G. Winter, Genetic Algorithms and Evolution Strategies in Engineering and Computer Science., West Sussex, England: John Wiley & Sons, 1998.

[427] J. Cytowski, Algorytmy genetyczne., Warszawa: Akademicka Oficyna Wydawnicza PLJ, 1996.

[428] K. De Jong, „Learning with Genetic Algorithms: An Overview,” w Genetic Algorithms., Los Angeles, California, IEEE Computer Society Press Technology Seriwes, 1992, pp. 30-48.

[429] R. Tanese, „Parallel genetic algorithms for a hypercube.,” w Genetic algorithms and their applications: Proceedings of the 2nd International Conference on genetic algorithms., pp. 177-183., 1987.

[430] J. D. Schaffer i J. J. Grefenstette, „Multi-objective learning via genetic algorithms.,” w Proceedings of the 9th Internetional Joint Conference on Artificial Intelligence., pp. 593-595, 1985.

[431] R. Schaefer, Podstawy genetycznej optymalizacji globalnej., Kraków: Wydawnictwo Uniwersytetu Jagiellońskiego, 2002.

[432] J. D. Schaffer i A. Morishima, „An adaptive crossover distribution mechanism for genetic algorithms.,” w Genetic algorithms and their applications: Proceedings of the 2nd Internetional Conference on Genetic Algorithms., pp. 36-40., 1987.

[433] M. Mitchell, An Introduction To Genetic Algorithms., Cambridge: MIT Press, 1996.

[434] M. Flasiński, Wstęp do sztucznej inteligencji., Warszawa: Wydawnictwo Naukowe PWN, 2011.

[435] B. P. Buckles, F. E. Petry i J. J. Grefenstette, „Optimization of Control Parameters for Genetic Algorithms,” w Genetic Algorithms, Los Alamitos, California, IEEE Computer Society Press, 1994.

[436] R. Cierniak, Ewolucja czy rewolucja. Nowoczesne techniki informatyczne. Księga jubileuszowa poświecona Profesorowi Leszkowi Rutkowskiemu., Częstochowa: Katedra Inżynierii Komputerowej Politechniki Częstochowskiej, 2003.

[437] J. J. Mulawka, Systemy ekspertowe, Warszawa: Wydawnictwa Naukowo-Techniczne, 1996.

[438] J. Drabarek, Metody sztucznej inteligencji w diagnostyce urządzeń elektronicznych., Koszalin: Wydawnictwo Uczelniane Politechniki Koszalińskiej, 2011.

[439] W. Findeisen, J. Szymanowski i A. Wierzbicki, Teoria i metody obliczeniowe optymalizacji., Warszawa: PWN, 1980.

[440] D. Wolpert i W. Macready, „No free lunch theorems for search.," Sante Fe Institute technical report, SI'M-TR-02-010, Santa Fe, 1995.

[441] P. Kieś, Dobór parametrów binarnego algorytmu genetycznego metodą off-line, Warszawa: Instytut Naukowo-Badawczy ZTUREK, 2000.

[442] E. Figielska, „Algorytmy ewolucyjne i ich zastosowania.," w 81-92, Warszawa, Warszwska Wyższa Szkoła Informatyki, 2006, p. Zeszyty Naukowe 1/2006.

[443] V. Odugawa, A. Tivari i R. Roy, Evolutionary computing in manufacturing industry: an overview of recent applications, Applied Soft Computing, 2005.

[444] J. R. Koza, Genetic Programming, MIT Press, 1992.

[445] K. E. Kinnear, Advances in Genetic Programming, Cambridge: MIT Press, 1994.

[446] M. Kasprzak, Zaawansowane programowanie. Wykład 4: jeszcze o metaheurystykach, Poznań: Politechnika Poznańska, Instytut Informatyki, 2012.

[447] F. Glover, „Scatter Search and path relinking," w New Ideas in Optimization, tom 8, McGraw Hill, 1999, pp. 297-316.

[448] T. James, C. Rego i F. Glover, „Sequential and Parallel Path-Relinking Algorithms for the Quadratic Assignment Problem.," IEEE Intelligent Systems, pp. 58-65, 2005.

[449] W. Bożejko i M. Wodecki, „Równoległy algorytm scatter search dla problemu przepływowego z kryterium C_sum," Automatyka, tom 11, nr Zeszyt 1-2, pp. 53-59, 2007.

[450] M. Łącki, „Model środowiska wieloagentowego w neuroewolucyjnym sterowaniu statkiem.," Zeszyty Naukowe Akademii Morskiej w Gdyni, tom 67, pp. 31-36, 2010.

[451] P. Boryczko, Neuroewolucja topologii sieci neuronowych z wykorzystaniem algorytmów genetycznych., Kraków: Politechnika Krakowska, 2011.

[452] K. O. Stanley, D. B. Bobby i R. Miikkulainen, „Real-Time Neuroevolution in the NERO Video Game," IEEE Transactions On Evolutionary Computation, tom 9, nr 6, pp. 653-668, 2005.

[453] L. Bolc i J. Cytowski, Search Methods for Artificial Intelligence, Warszawa: Academic Press, 1992.

[454] I. Buałynicki-Birula i I. Białaynicka-Birula, Modelowanie rzeczywistości. Jak w komputerze przegląda się świat., Warszawa: Wydawnictwa Naukowo - Techniczne, 2007.

[455] R. Dawkins, Rzeka genów., Warszawa: Wydawnictwo CIS, Oficyna Wydawnicza MOST., 1995.

[456] A. Hoffman, Wokół ewolucji., Warszawa: Biblioteka Myśli Współczesnej, 1983.

[457] L. Bolc i J. Zaremba, Wprowadzenie do uczenia się maszyn., Warszawa: Akademicka Oficyna Wydawnicza RM., 1992.

[458] S. Goonatilake i S. Khebbal, Intelligent Hybrid Systems, University College London, UK: John Wiley & Sons, 1995.

[459] W. D. Mauer, The programmer's introduction to LISP, 1972.

[460] „http://zasoby.open.agh.edu.pl," [Online]. Available: http://zasoby.open.agh.edu.pl.

[461] H. Piech, Wykorzystanie narzędzi stochastycznych do realizacji algorytmów genetycznych., Częstochowa: Politechnika Częstochowska, 2003.

[462] K. Lorenz, „Przegląd algorytmów genetycznych stosowanych w procesie selekcji cech wyekstrahowanych z sygnału EEG.," w Nowe trendy w naukach inżynieryjnych, Kraków, Monografia pod red. M. Kuczera, 2013.

[463] T. Jones, „Crossover, macromutation, and population-based Search.," w Proceedings of the 6th International Conference on Genetic Algorithms., San Mateo, California, Morgan Kaufmann, 1996.

[464] D. E. Goldberg, Algorytmy genetyczne i ich zastosowania., Warszawa: Wydawnictwa Naukowo - Techniczne, 1998.

[465] M. Bereta i P. Jarosz, „Algorytmy genetyczne.," 15 10 2016. [Online]. Available: michalbereta.pl/dydaktyka/ae/lab_genetyczne/laboratorium_genetyczne.pdf.

[466] H. Lakany i B. A. Conway, Understanding intention of movement from electroencephalograms., Expert Sys-tems, 24, 5, 2007.

[467] S. T. Welstead, Neural Network And Fuzzy Logic Applications In C/C++, John Wiley & Sons, Inc., 1994.

[468] M. Kurzyński, Metody sztucznej inteligencji dla inżynierów., Legnica: Seria Wydawnicza Państwowej Wyższej Szkoły Zawodowej im. Witelona w Legnicy., 2008.

[469] A. Bołtuć, „Optymalizacja globalna.,” 15 10 2016. [Online]. Available: ii.uwb.edu.pl/~aboltuc/images/stories/OG/wykad_3.pdf.

[470] P. Urbanek, „Algorytmy genetyczne.,” 15 10 2016. [Online]. Available: ftp://zly.kis.p.lodz.pl/pub/people/P.Urbanek/Ostr%F3w%202011/Algorytmy%20genetyczne%20wyk%B3ad%202_2010.pdf.

[471] Z. P. Król, Przegl¡d i komputerowa implementacja algorytmów genetycznych w oprogramowaniu edukacyjnym, Warszawa: Politechnika Warszawska, Wydział Elektryczny, Instytut Eleltrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych, 2006.

[472] R. L. Haupt i S. E. Haupt, Practical genetic algorithms., John Wiley & Sons Ltd., 1998.

[473] U. Boryczka, „Techniki ewolucyjne - algorytm genetyczny i nie tylko,” 15 10 2016. [Online]. Available: http://155.158.112.34/~algorytmyewolucyjne/materialy/reprezentacja_binarna.pdf.

[474] M. Boryczka, „Algorytmy Genetyczne,” 15 10 2016. [Online]. Available: zsi.ii.us.edu.pl/~mboryczka/pliki/algorytmy.ppt.

[475] T. G. Gwiazda, Algorytmy Genetyczne Kompendium Tom 1 Operator Krzyżowania Dla Problemów Numerycznych, Warszawa: Wydawnictwo Naukowe PWN SA, 2007.

[476] T. Nomura, An Analysis on Crossovers for Real Number Chromosomesin an Infinite Population Size, ATR Humań Information Processing Research Laboratones Evolutionary Systems Department, 1997.

[477] P. Wawrzyński, Podstawy Sztucznej Inteligencji, Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 2014.

[478] R. Belea i L. Beldiman, „A new method of gene coding for a genetic algorithm designed for parametric optimization,” w The Annals of "Dunarea De Jos", Galati, University of Galati, 2003, pp. 66-71.

[479] W. M. Spears i K. A. De Jong, „On the Virtues of Parameterized Uniform Crossover,” w Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman, Morgan Kaufman, 1991, pp. 230-236.

[480] O. Cordón, S. Damas i J. Santamaria, „A CHC Evolutionary Algorithm for 3D Image Registration,” w 10th International Fuzzy Systems Association World

Congress (IFSA'03), Lecture Notes in Artificial Intelligence 2715, 2003, pp. 404-411.

[481] C. Cotta i J. M. Troya, „Using Dynastic Exploring Recombination to Promote Diversity in Genetic Search," w Parallel Problem Sohingfrom Naturę - 6th International Conference, Springer Verlag, 2000, pp. 16-20.

[482] F. J. Burkowski, „Shuffle Crossover and Mutual Information," Congress on Evolutionary Computation, pp. 1574-1580, 1999.

[483] C. Fernandes, R. Tavares, C. Munteanu i A. Rosa, „Using Assortative Mating in Genetic Algorithms for Vector Quantization Problems," w Proceedings of the 2001 ACM symposium on Applied computing, 2001, pp. 361-365.

[484] H.-M. Voigt, H. Muhlenbein i D. Cvetković, „Fuzzy recombination for the Breeder Genetic Algorithm," w Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufman, 1995, pp. 104-111.

[485] H. Muhlenbein i D. Schlierkamp-Voosen, „Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization," w Evolutionary Computation vol. 1, 1993, pp. 25-49.

[486] F. Herrera, M. Lozano i J. L. Verdegay, „Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioral Analysis," Artificial Intelligence Review vol. 12, nr vol. 12, pp. 254-319, 1998.

[487] M. Takahashi i H. Kita, „A Crossover Operator Using Independent Component Analysis for Real-Coded Genetic Algorithm," Proceedings of the 2001 Congress on Evolutionary Computation, pp. 643-649, 2001.

[488] N. J. Radcliffe, „Forma Analysis and Random Respectful Recombination," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 222-229, 1991.

[489] R. A. Watson i J. B. Pollack, „Recombination Without Respect: Schema Combination and Disruption in Genetic Algorithm Crossover," w Proceedings of GECCO 2000, Morgan Kaufman, 2000, pp. 112-119.

[490] T. Ozugur, „Hierarchical Provisioning for Cellular networks," IEEE Transactions on Wireless Communications, tom 4(2), pp. 775-791, 2005.

[491] S. J. Louis i G. J. Rawlins, „Designer Genetic Algorithms: Genetic Algorithms in Structure Design," w Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman, 1991, pp. 53-60.

[492] H. Maini, K. Mehrota, C. Mohan i S. Ranka, „Knowkdge-Based Nonuniform Crossover,” w Proceedings of IEEE International Conference on Evolutionary Computation, Orlando, 1994.

[493] K. Vekaria i C. Clack, „Biases Introduced by Adaptive Recombination Operators,” w Proceedings of the Evolutionary Computation Conference, 1999, pp. 670-677.

[494] C. Ch.-H i C. J.-N., „Genetic Algorithms: initialization schemes and genes extraction,” w Proceedings of the Ninth IEEE International Conference on Fuzzy Systems, 2000, pp. 965-968.

[495] S. W.M., „Adapting Crossover in Evolutionary Algorithms,” w Technical Report AIC-92-025, Naval Research Laboratory, Navy Center for Applied Research on Artificial Intelligence, p. 1992.

[496] D. Vrajitoru, „Intra and Extra-Generation Schemes for Cómbining Crossover Operators,” w Proceedings of the Fifteenth Midwest Artificial Intelligence and Cognitive Science Conference MAICS 2004, 2004, pp. 86-91.

[497] F Herrera, M Lozano, AM Sánchez, „Hybrid Crossover Operators for Real-Coded Genetic Algorithms: An Experimental Study,” Soft Computing-A Fusiort of Foundations Methodologies and Applications, tom 9(4), pp. 280-298.

[498] AH Konstam, SJ Hartley, WL Carr „Optimization in a Distributed Processing Environment using Genetic Algorithm with Multivariate Crossover,” Proceedings of the 1992 ACMannual Conference on Communications, pp. 109-116, 1992.

[499] W Antennas., „A Hybrid Algorithm using Genetic Algorithm and Gradient-Based Algorithm for Iterative Microwave Inverse Scattering,” IEEE International Conference on Evolutionary Computation, pp. 450-455, 1995.

[500] K Deb, M Goyal, „Optimizing Engineering Designs Using a Combined Genetic Search,” w Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufman, 521-528.

[501] J Park, J Park, CH Lee, MS Han., „Robust and Efficient Genetic Crossover Operator: Homologous Recombination,” w Proceedings of 1993 International Joint t Conference on Neural Networks, 1993, pp. 2975-2978.

[502] H.-G. Beyer, „Toward a Theory of Evolution Strategies: On the Benefits of Sex the (μ/μλ) Theory,” w Evolutionary Computation 3(1), 1995, pp. 81-111.

[503] SJ. Hartley, AH. Konstam,„Using Genetic Algorithms to Generate Steiner Triple Systems,” w ACMConference on Computer Science, 1993, pp. 366-371.

[504] YC Hou, YH Chang, „A New Efficient Encoding Mode of Genetic Algorithms for the Generalized Plant Allocation Problem," w Journal of Information Science And Engineering vol. 20, 2004, pp. 1019-1034.

[505] T. G. Gwiazda, Algorytmy Genetyczne Kompendium Tom 2 Operator Mutacji Dla Problemów Numerycznych, Warszawa: Wydawnictwo Naukowe PWN SA, 2007.

[506] T.-P. Hong i H.-S. Wang, „A Dynamic Mutation Genetic Algorithm.," w IEEE International Conference on Systems, Man and Cybemetics, vol. 3, 1996.

[507] T.-P. Hong, H.-S. Wang i W.-C. Chen, „Simultaneously Applying Multiple Mutation Operators in Genetic Algorithms.," w Journal o f Heuristics vol. 6(4), 2000.

[508] T.-P. Hong, H.-S. Wang i T. Y. Juang, „A dynamic crossover genetic algorithm.," w National Computer Symposium., Taiwan, 1995.

[509] H.-S. Yoon i B. R. Moon, „An Empirical Study on the Synergy of Multiple Crossover Operators.," w IEEE Transactions on Evolutionary Computation, vol. 6(2), 2002.

[510] T. Sasaki, C.-C. Hsu, H. Fujikawa i S. Yamada, „A Multi-operator Self-tuning Genetic Algorithm for Optimization.," w 23rd International Conference on Industrial Electronics, Control and Instrumentation, vol. 3, 1994.

[511] A. Acan, H. Altincay, Y. Tekol i A. Unveren, „A Genetic Algorithm with Multiple Crossover Operators for Optimal Freąuency Assignment Problem.," w Proceeding of The 2003 Congress on Evolutionary Computation., 2003.

[512] Z. Gang, P. Chuwu i Z. Mingshu, „A Modified Genetic Algorithm Based on the Best Schema and Its Application for Function Optimization," w Proceedings of the 3rd World Congress on Intelligent Control and Automation, 2000.

[513] K. Sastry i D. E. Goldberg, „Designing Competent Mutation Operators via Probabilistic Model Building of Neighborhoods.," IlliGAL Report no. 2004006, 2004.

[514] K. Sastry i D. E. Goldberg, „Let's get Ready to Rumbie: Crossover Versus Mutation Head to Head.," IlliGAL Report no. 2004005, 2004.

[515] Y. Iwasaki i F. Yonezawa, „Genetic EvoIution through Stochastic Mutation Dynamics.," w Proceedings of IEEE International Conference on Evolutionary Computation, 1996 .

[516] A. Agapie i H. Dediu, „GA for Deceptive Problems: Inverting Schemata by a Statistical Approach.," w Proceedings of International Conference on Evolutionary Computation, 1996.

[517] L. Yang, D. H. Widyantoro, T. loerger i J. Yen, „An Entropy-based Adaptive Genetic Algorithm for Leaming Classification Rules.," w Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, 2001.

[518] W. Paszkowicz, „Properties of a genetic algonthm extended by a random self-leaming operator and asymmetric mutations: A convergence study for a task of powder-pattem indexing.," w Analytica Chimica Acta, vol. 566, 2006.

[519] L. Yang i J. Yen, „An adaptive simplex genetic algonthm.," w Proceedings of the Genetic and Evolutionary Algorithms Conference, 2000.

[520] S. Uyar, S. Sariel i G. Eryigit, „A gene based adaptive mutation strategy for genetic algorithms.," w Lecture Notes in Computer Science, 3103, 2004.

[521] P. Hartono, S. Hashimoto i M. Wahde, „Labeled-GA with Adaptive Mutation Rate," w 2004 Congress on Evolutionary Computation, vol. 2, 2004.

[522] S. Yang i S. Uyar, „Adaptive mutation with fitness and allele distribution correlation for genetic algorithms.," w Proceedings of the 2006 ACM symposium on Applied computing, 2006.

[523] R. Tinós i A. C. de Carvalho, „A genetic algorithm with gene dependent mutation probability for non-stationary optimization problems.," w Congress on Evolutionary Computation, 2004.

[524] S. Uyar i G. Eryigit, „Improvements to penalty-based evolutionary algorithms for the multi-dimensional knapsack problem using a gene-based adaptive mutation approach.," w Proceedings of the 2005 Conference on Genetic and Evolutionary Computation., 2005.

[525] Y. H. Song, G. S. Wang, A. T. Johns i P. Y. Wang, „Improved Genetic Algorithms with Fuzzy Logic Controlled Crossover and Mutation," w UKACC International Conference on CONTROL'96 vol. 1, 1996.

[526] T. Zhang i W. A. Gruver, „Fuzzy-Neural Tuned Genetic Algorithm Applied to Large-Space Constraint Satisfaction," w 1998 IEEE International Conference on Systems, Man and Cybemetics vol.4, 1998.

[527] Y. Shi, R. C. Eberhart i Y. Chen, „Implementation of Evolutionary Fuzzy Systems.," w IEEE Transactions on Fuzzy Systems vol. 7(2), 1999.

[528] R. Subbu i P. P. Bonissone, „A Retrospective View of Fuzzy Control of Evolutionary Algorithm Resources," w IEEE International Conference on Fuzzy Systems vol. 1, 2003.

[529] J. Zhang, H. S. Chung i B. J. Hu, „Adaptiye Probabilities of Crossover and Mutation in Genetic Algorithms Based on Clustering Techniąue," w Congress on Evolutionary Computation, vol. 2, 2004.

[530] Z. Bingul, A. Sekmen i S. Zein-Sabatto, „Evolutionary Approach to Multi-0bjective Problems Using Adaptive Genetic Algorithms," w IEEE International Conference on Systems, Man and Cybenetics vol.3, 2000.

[531] F. Herrera i M. Lozano, „Adaptive Genetic Operators Based on Coevolution with Fuzzy Behaviors.," w IEEE Transactions on Evolutionary Computation vol.5(2), 2001.

[532] L. Ren i Y. Ding, „Parameter Optimization for a Class of General TS Fuzzy Controllers via a New DNA-Based Genetic Algorithm," w Proceedings of the 5th World Congress on Intelligent Control and Automation, vol. 3, 2004.

[533] K. Wang, „A New Fuzzy Genetic Algorithm based on Population Diversity," w Proceedings of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2001.

[534] L. Hongjie, W. Xiufeng, Z. Weicum i X. Guohua, „Market Clearing Price Forecasting Based on Dynamic Fuzzy System," w International Conference on Power Systems Technology, vol. 2, 2002.

[535] X. Pan, J. Zhang i K. Y. Szeto, „Application of Mutation Only Genetic Algonthm for the Extraction of Investment Strategy in Financial Time Series.," w International Conference on Neural Networks and Brain, vol. 3, 2005.

[536] K. Y. Szeto i J. Zhang, „Adaptive Genetic Algorithm and Quasi-Parallel Genetic Algorithm: Application to Knapsack Problem.," w Lecture Notes in Computer Science vol. 3743, 2005.

[537] S. Uyar, S. Sariel i G. Eryigit, „A gene based adaptive mutation strategy for genetic algorithms.," w Lecture Notes in Computer Science, 3103, 2004.

[538] D. Thierens, „Adaptive mutation rate control schemes in genetic algorithms," w Proceedings o f the 2002 Congress on Evolutionary Computation, vol. 1,, 2002.

[539] S. J. Hartley i A. H. Konstam, „Using Genetic Algorithms to Generate Steiner Triple Systems," w ACM Conference on Computer Science, 1993.

[540] A. T. Fuller i B. Nowrouzian, „A novel technique for optimization over the canonical signed-digit number space using genetic algorithms,” w The 2001 International IEEE Symposium of Systems and Circuits, vol. 2, 2001.

[541] F. Ashrafzadeh i B. Nowrouzian, „Crossover and Mutation in Genetic Algorithms Employing Canonical Signed-Digit Number System.,” w Proceedings of the 4th Midwest Symposium on Circuits and Systems, vol. 2, 1997.

[542] Y.-C. Hou i Y.-H. Chang, „ A New Efficient Encoding Mode of Genetic Algorithms for the Generalized Plant Allocation Problem.,” w Journal of Information Science and Engineering, vol. 20, 2004.

[543] N. Kubota, T. Arakawa i T. Fukuda, „Trajectory Generation for Redundant Manipulator Using Virus Evolutionary Genetic Algorithm,” w Proceedings of the 1997 IEEE International Conference on Robotics and Automation, 1997.

[544] I. Harvey, „The Mcrobial Genetic Algorithm.,” w School for Cognitive and Computing Sciences., Brighton, UK, University of Sussex, 1996.

[545] H. Kanoh, K. Hasegawa i N. Kato, „Solving Constraint Satisfaction Problems by a genetic Algorithm Adopting Viral Infection.,” w Proceedings of IEEE International Joint Symposia on Intelligence and Systems, 1996.

[546] P. Smith, „Conjugation - A Bacterially Inspired Form of Genetic Recombination.,” w Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University,, 1996.

[547] G. Schiitz i F. M. Pires, „A genetic based algorithm for thequadratic 0-1 problem.,” w Investigacao Operacional, vol. 23,, 2003.

[548] Y. Cui i Q. Huang, „Extracting characters of license plates from video seąuences.,” w Machine Vision and Applications, vol. 10, 1998.

[549] C.-C. Hsu, H. Takahashi, K. Shida, H. Fujikawa i S. Yamada, „An Eugenic Mutations for Optimum Problems,” w Proceedings of the 1995 IEEE IECON 21st International Conference on Industrial Electronics, Control and Instrumentation, vol. 2, 1995.

[550] L. Yao, W. A. Sethares i D. C. Kammer, „ Sensor Placement for On-orbit Modal Identification on Large Space Structure via a Genetic Algorithm.,” w IEEE International Conference on Systems Engineering., 1992.

[551] D. Gantala, A. S. Abdel-Aty-Zohdy i R. L. Ewing, „New Genetic Algorithm Approach For Dynamic Biochemical Sensor Measurements Characterization.,” w

Proceedings of The 2002 45th Midwest Symposium on Circuits and Systems., 2002.

[552] M. S. Sharawi, A. S. Abdel-Aty-Zohdy i R. L. Ewing, „Optimal-weights sensors-measurement fusion using genetic algorithms.," w Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems, 2001.

[553] R. Hinterding, „Gaussian Mutation and Self-adaptation for Numeric Genetic Algorithms.," w IEEE International Conference on Evolutionary Computation, 1996.

[554] V. Cutello, G. Nicosia, M. Pavone i J. Timmis, „An Immune Algorithm for Protein Structure Prediction on Lattice Models.," w IEEE Transactions on Evolutionary Computation., 2007.

[555] G. Rudolph, „Local Convergence rates of Simple Evolutionary Algorithms with Cauchy Mutations.," w IEEE Transactions on Evolutionary Computation, vol. 1(4), 1997.

[556] I. De Falco, A. Iazzetta, E. Tarantino i A. Della Cioppa, „The Effectiveness of Co-mutation in Evolutionary Algorithms: the Mijn operator.," w Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'98), Anchorage, 1998.

[557] T. C. Jones, „Crossover, Macromutation, and Population-based Search.," w Proceedings of the Sixth International Conference on Genetic Algorithms, 1995.

[558] E. Falkenauer, Genetic algorithms and grouping problems, West Sussex, England: John Wiley & Sons Ltd., 1998.

[559] „http://www.k0pper.republika.pl/geny.htm," [Online]. Available: http://www.k0pper.republika.pl/geny.htm.

[560] G. R. Beddoe i S. Petrovic, „Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering.," European Journal of Operational Research., tom 175, p. 649–671, 2006.

[561] M. I. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn i A. K. Jain, „Dimensionality reduction using genetic algorithms.," IEEE Transactions on Evolutionary Computation., tom 4, p. 164–171, 2000.

[562] C.-L. Huang i C.-J. Wang, „A GA-based feature selection and parameters optimization for support vector machines.," Expert Systems with Applications, tom 31, p. 231–240, 2006.

[563] T. T. Nguyen, A. W.-C. Liew, X. C. Pham, M. T. Tran i M. P. Nguyen, „A Novel Genetic Algorithm Approach for Simultaneous Feature and Classifier Selection in Multi Classifier System.," IEEE Congress on Evolutionary Computation (CEC), pp. 1698-1705, 2014.

[564] J. D. Schaffer, Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms., Nashville, TN: Vanderbilt University, 1984.

[565] D. J. Schaffer, „Multiple objective optimization with vector evaluated genetic algorithms," w International Conference on Genetic Algorithms and Their Applications, 1985.

[566] P. Haleja i C.-Y. Lin, „Genetic search strategies in multicriterion optimal design," Structural optimization, tom Volume 4, nr Issue 2, p. 99–107, June 1992.

[567] C. M. Fonseca i P. J. Fleming, „Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," w Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufmann, 1993, pp. 416–423.

[568] J. Horn, N. Nafplitois i D. E. Goldberg, „A niched Pareto genetic algorithm for multiobjective optimization," w Proceedings in the 1st IEEE Conference Evolutionary Computation, IEEE Service Cente, 0-7803-1899-4/94, 1994, p. 82–87.

[569] E. Zitzler i L. Thiele, „An evolutionary algorithm for multiobjective optimization: The strength pareto approach," w Technical Report 43, Zurich, Switzerland, Swiss Federal Institute of Technology (ETH), Computer Engineering and Networks Laboratory (TIK), 1998.

[570] E. Zitzler i L. Thiele, „Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," IEEE Transactions On Evolutionary Computation, tom 3, nr 4, pp. 257-271, November 1999.

[571] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Aachen, Germany: Institut für Technische Informatik und Kommunikationsnetze, Computer Engineering and Networks Laboratory, November 11, 1999.

[572] E. Zitzler, M. Laumanns i L. Thiele, „SPEA2: Improving the strength Pareto evolutionary algorithm," Zurich, Switzerland, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich, May 2001.

[573] E. Zitzler, M. Laumanns i L. Thiele, „SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization,” w Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), 2002.

[574] E. Zitzler, M. Laumanns i S. Bleuler, „A Tutorial on Evolutionary Multiobjective Optimization,” w Metaheuristics for Multiobjective Optimisation, Zurich, Switzerland, Springer, 2004, p. 3–37.

[575] P. Lipiński, „Wykład z algorytmów ewolucyjnych.,” [Online]. Available: http://www.ii.uni.wroc.pl/~lipinski/AE2017/lecture04.pdf. [Data uzyskania dostępu: 15 11 2017].

[576] E. Yom-Tov i G. F. Inbar, „Feature Selection for the Classification of Movements From Single Movement-Related Potentials,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, tom vol 10, nr no. 3, pp. 170-177, 2002.

[577] E. Yom-Tov i G. F. Inbar, „Feature Selection for the Classification of Movements From Single Movement-Related Potentials.,” IEEE Transactions on Neural Systems and Rehabilitation Engineering., tom vol 10, nr no. 3, pp. 170-177, 2002.

[578] I. Rejer, „Genetic Algorithms in EEG Feature Selection for the Classification of Movements of the Left and Right Hand.,” Advances in Intelligent and Soft Computing, CORES 2013, tom 226, pp. 579-589, 2013.

[579] I. Rejer, „Genetic Algorithm with Aggressive Mutation for Feature Selection in BCI Feature Space.,” Pattern Analysis & Applications., 2014.

[580] B. H. 2020, „BNCI Horizon 2020,” 18 02 2022. [Online]. Available: http://bnci-horizon-2020.eu/. [Data uzyskania dostępu: 18 02 2022].

[581] B. H. 2020, „BNCI Horizon 2020,” 18 02 2022. [Online]. Available: http://bnci-horizon-2020.eu/database/data-sets. [Data uzyskania dostępu: 18 02 2022].

[582] BBCI, „Berlin Brain-Computer Interface,” 18 02 2022. [Online]. Available: https://www.bbci.de/. [Data uzyskania dostępu: 18 02 2022].

[583] B. B.-C. Interface, „Berlin Brain-Computer Interface,” 18 02 2022. [Online]. Available: https://www.bbci.de/activities. [Data uzyskania dostępu: 18 02 2022].

[584] J. Czaja i E. Preweda, „Analiza statystyczna zmiennej losowej wielowymiarowej,” Geodezja, tom 6, nr 2, pp. 129-145, 200.

[585] P. Stefanów, „Ograniczenia stosowania współczynnika korelacji liniowej Pearsona.,” w Przedsiębiorcze aspekty rozwoju organizacji i biznesu, Kraków, Oficyna Wydawnicza Kraków, 2011, pp. 240-262.

[586] P. Peternek i M. Kosny, „Kilka uwag o testowaniu istotności współczynnika korelacji.,” Zeszyty Naukowe Wyższej Szkoły Bankowej we Wrocławiu., nr 20, pp. 341-350, 2021.

[587] „Autocalibration and recurrent adaptation (001-2015),” 15 12 2022. [Online]. Available: http://bnci-horizon-2020.eu/database/data-sets. [Data uzyskania dostępu: 15 12 2022].

[588] „Data sets Ia / Ib,” 15 12 2022. [Online]. Available: https://www.bbci.de/competition/ii/. [Data uzyskania dostępu: 15 12 2022].

[589] „Data set I,” 15 12 2022. [Online]. Available: https://www.bbci.de/competition/iii/#data_set_i. [Data uzyskania dostępu: 15 12 2022].

[590] „Data sets IIIa,” 15 12 2022. [Online]. Available: https://www.bbci.de/competition/iii/#data_set_i. [Data uzyskania dostępu: 15 12 2022].

[591] „Data set III,” 15 12 2022. [Online]. Available: https://www.bbci.de/competition/ii/. [Data uzyskania dostępu: 15 12 2022].

[592] „Zbiór Adult,” 22 05 2022. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Adult. [Data uzyskania dostępu: 22 05 2022].

[593] „Zbiór Coil100,” 20 02 2023. [Online]. Available: https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php. [Data uzyskania dostępu: 20 02 2023].

[594] „Zbiór Dermatology,” 22 05 2022. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/. [Data uzyskania dostępu: 22 05 2022].

[595] „Zbiór Gisette,” 22 05 2022. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Gisette. [Data uzyskania dostępu: 22 05 2022].

[596] „Zbiór Gli_85,” 20 02 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE4412. [Data uzyskania dostępu: 20 02 2023].

[597] „Zbiór Humanactivity (Matlab 2019b),” 22 05 2022. [Online]. Available: https://www.mathworks.com/products/new products/release2019b.html. [Data uzyskania dostępu: 22 05 2022].

[598] „Zbiór Orl_32x32,” 20 02 2023. [Online]. Available: http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html. [Data uzyskania dostępu: 20 02 2023].

[599] „Zbiór Orlraws10P,” 22 05 2022. [Online]. Available: https://jundongl.github.io/scikit-feature/datasets.html. [Data uzyskania dostępu: 22 05 2022].

[600] „Zbiór Pima-indians-diabetes,” 22 05 2022. [Online]. Available: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. [Data uzyskania dostępu: 22 05 2022].

[601] „Zbiór WarpAR10P,” 20 02 2023. [Online]. Available: https://jundongl.github.io/scikit-feature/datasets.html. [Data uzyskania dostępu: 20 02 2023].

[602] „Zbiór Yale_32x32,” 20 02 2023. [Online]. Available: http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html. [Data uzyskania dostępu: 20 02 2023].

Spis tabel

Tabela 3.1. Interpretacja współczynnika korelacji liniowej Pearsona.

Tabela 4.1. Porównanie kodowania binarnego z kodowaniem Graya.

Tabela 6.1. Charakterystyka zbiorów danych użytych w badaniu; wartości pogrubione odnoszą się do liczby cech, które pozostały w zbiorach danych po eliminacji cech o zbyt wysokiej korelacji.

Tabela 6.2. Najwyższa dokładność osiągnięta dla różnej liczby cech dla zbioru 1. Wyniki pochodzą z pięciu uruchomień algorytmu GAAMmf przeprowadzonych z różnymi wartościami parametrów *accWeight* i *fsWeight*.

Tabela 6.3. Liczba iteracji dla trzech uruchomień algorytmu dla różnego parametru prawdopodobieństwa mutacji; *S* – liczba osobników w populacji startowej, *K* – liczba osobników skrzyżowanych, *M* – liczba mutowanych genów, *probM* – prawdopodobieństwo mutacji, *N* – liczba iteracji.

Tabela 6.4. Najwyższa dokładność klasyfikacji osiągnięta w trakcie działania algorytmu dla różnej liczby cech przy każdym z badanych współczynników mutacji.

Tabela 6.5. Najwyższa osiągnięta dokładność dla różnej liczby cech dla zbioru 1. Wyniki pochodzą z pięciu uruchomień algorytmu GAAMmf przeprowadzonych z wartościami parametrów: *accWeight=1, fsWeight=1,* probM=1, *probM=1* oraz *liczba_iteracji=100*.

Tabela 6.6. Uzyskane cechy dla pięciu uruchomień algorytmu. Dla każdego uruchomienia ustawiono parametry: *accWeight=1*, *fsWeight=1*, *probM=1* oraz *liczba_iteracji=100*.

Tabela 6.7. Wyniki algorytmu GAAMmf dla 5 zbiorów BCI. W tabeli przedstawiono dokładność klasyfikacji oraz liczbę cech zwrócone na zakończenie każdego z 15 uruchomień algorytmu przy różnych wartościach parametrów *probM*, *accWeight* oraz *fsWeight*.

Tabela 6.8. Porównanie wyników zwróconych przez algorytm GAAMmf z wynikami 8 metod referencyjnych dla 5 zbiorów BCI.

Tabela 6.9. Demografia zbiorów danych wykorzystanych w eksperymencie; wartości zapisane pogrubioną czcionką dotyczą liczby cech/przykładów, które pozostały w zbiorach danych po etapie wstępnego przetwarzania.

Tabela 6.10 Porównanie wyników zwróconych przez algorytm GAAMmf z wynikami 8 metod referencyjnych dla 11 benchmarkowych zbiorów danych.

Spis rysunków

Rysunek 1.1. Efekt ERD i ERS wywołany przez rzeczywisty lub wyobrażony ruch ręki.

Rysunek 1.2. Przykład czepka do badań EEG.

Rysunek 1.3. Położenie elektrod w systemie 10–20; a) widok z góry; b) widok z lewej strony; c) widok z prawej strony.

Rysunek 2.1. Podział metod optymalizacyjnych.

Rysunek 2.2. Metoda dzielenia przedziału na połowę.

Rysunek 2.3. Simpleks dla dwóch (a) i trzech (b) zmiennych.

Rysunek 2.4. Budowa nowego simpleksu na płaszczyźnie dwuwymiarowej: (a) początkowy simpleks X_1, X_2, X_3, (b) nowy simpleks X_2, X_3, X_4.

Rysunek 2.5. Poszukiwanie minimum metodą gradientu prostego.

Rysunek 2.6. Koło o promieniu r i środku w punkcie (0, 0) wpisane w kwadrat.

Rysunek 2.7. Zwiększanie rozdzielczości losowań poprzez podział promienia na coraz większą liczbę równych odcinków i uzyskanie lepszego przybliżenia rozwiązania.

Rysunek 2.8. Poszukiwanie rozwiązania spośród wszystkich dostępnych rozwiązań.

Rysunek 2.9. Schemat poszukiwania najlepszego rozwiązania w algorytmie symulowanego wyżarzania.

Rysunek 2.10. Graficzna ilustracja metody ograniczania kryteriów dla zadania dwukryterialnego.

Rysunek 2.11. a) Dominacja rozwiązania a nad rozwiązaniem b i c; b) Obszar dominacji rozwiązania a.

Rysunek 2.12. Obszar dominacji rozwiązań d i e nad rozwiązaniem a pod względem jednego czynnika.

Rysunek 2.13. Front Pareto.

Rysunek 3.1. Model selekcji cech z wykorzystaniem metod wrapper.
Rysunek 3.2. Schemat działania selekcji w przód.
Rysunek 3.3. Schemat działania selekcji w tył.
Rysunek 3.4. Model selekcji cech za pomocą metod filtrujących.
Rysunek 3.5. Model selekcji cech za pomocą metod wbudowanych.
Rysunek 3.6. Kierunki wektorów bazowych maksymalizujące wariancję pierwotnego zbioru cech; a) pierwotna przestrzeń cech; b) przestrzeń cech po transformacji za pomocą metody PCA.

Rysunek 4.1. Schemat działania klasycznego algorytmu genetycznego.
Rysunek 4.2. Przykład chromosomu zbudowanego z 8 bitów ponumerowanych od 0 do 7.
Rysunek 4.3. Przykład krzyżowania jednopunktowego.
Rysunek 4.4. Przykład krzyżowania dwupunktowego.
Rysunek 4.5. Przykład krzyżowania wielopunktowego.
Rysunek 4.6. Przykład krzyżowania równomiernego.
Rysunek 4.7. Przykład mutacji jednopunktowej.
Rysunek 4.8. Przykład mutacji wielopunktowej.
Rysunek 4.9. Przykład inwersji.

Rysunek 5.1. Podstawowa reguła kodowania cech w klasycznym algorytmie genetycznym: 0 – cecha nie występuje w osobniku, 1 – cecha występuje w osobniku.
Rysunek 5.2. Przykład odmiennego sposobu kodowania cech w osobniku. Geny $f_{0,i}$, $f_{1,i}$,$f_{2,i}$ zawierają informację, czy cech *i* wchodzi w skład podzbioru cech zakodowanego w osobniku, natomiast gen w_i informuje o stopniu ważności tej cechy. Na rysunku $w_0, \ldots, w_{F-1}$ oznaczają wagę cech; $f_{0,0}, \ldots, f_{0,F-1}, f_{1,0}, \ldots, f_{1,F-1}, f_{2,0}, \ldots, f_{2,F-1}$ oznaczają zbiór wyselekcjonowanych cech.
Rysunek 5.3. Wygląd chromosomu składającego się z trzech części: *C*, γ i maski funkcji.
Rysunek 5.4. Struktura chromosomu.

Rysunek 5.5. Krzyżowanie jednopunktowe pary osobników A i B w części kodującej klasyfikatory.
Rysunek 5.6. Krzyżowanie jednopunktowe pary osobników A i B w części kodującej cechy.
Rysunek 5.7. Schemat działania algorytmu NSGA II.
Rysunek 5.8. Fronty Pareto
Rysunek 5.9. Prostokąt rozpięty na dwóch najbliższych sąsiadach osobnika w danym froncie Pareto.
Rysunek 5.10. Schemat algorytmu genetycznego z ustaloną liczbą cech (algorytm Cullinga).
Rysunek 5.11. Schemat algorytmu genetycznego z agresywną mutacją (GAAM).
Rysunek 5.12. Pseudokod schematu agresywnej mutacji; $population$ – macierz osobników; *new* – wektor zawierający wszystkie geny *i*-tego osobnika; *random({0,1,...,F})* – funkcja zwracająca losową cechę ze zbioru cech.
Rysunek 5.13. Przykład zbioru wszystkich osobników potomnych powstałych z tego samego osobnika rodzicielskiego; X – gen zmieniony w osobniku potomnym.
Rysunek 5.14. Schemat algorytmu genetycznego ze zmniejszającą się liczbą genów w osobniku (Melting-GAAM).
Rysunek 5.15. Algorytm GAAMmf.

Rysunek 6.1. Wykresy przedstawiające wyniki GAAMmf dla zbioru 1; wykresy po lewej stronie prezentują dokładność najlepszego osobnika zwróconego w każdej iteracji, natomiast wykresy po prawej stronie przedstawiają liczbę cech zakodowanych w tym osobniku; wiersze wykresów prezentują wyniki dla różnych wartości parametrów *accWeight i fsWeight*: a) *accWeight=1, fsWeight=1*; b) *accWeight=1, fsWeight=2*; c) *accWeight=1, fsWeight=4*; d) *accWeight=2, fsWeight=1*; e) *accWeight=4, fsWeight=1*.
Rysunek 6.2. Wykresy przedstawiające wyniki GAAMmf dla zbioru 1; wykresy po lewej stronie prezentują dokładność najlepszego osobnika zwróconego w każdej iteracji, wykresy po prawej stronie przedstawiają liczbę cech zakodowanych w tym osobniku; wiersze wykresów prezentują wyniki

dla różnych wartości parametru *probM*: a) *probM = 0.1;* b) *probM = 0.5;* c) *probM = 1*.

Rysunek 6.3. Badanie stabilności algorytmu podczas 5 uruchomień; gr1 – cechy 1005, 1006; gr2 – cechy 795, 799, 800, 805; gr3 – cechy 332, 333, 335; gr4 – cechy 69, 72.

Rysunek 6.4. Wyniki zwrócone przez algorytm GAAMmf przy różnej wartości parametrów *probM*, *accWeight* oraz *fsWeight* uśrednione po 5 zbiorach BCI; a) średnia dokładność klasyfikacji; b) średnia liczba cech.

Rysunek 6.5. Średnia dokładność klasyfikacji oraz średnia liczba cech dla kolejnych zbiorów BCI (uśrednienie po 9 metodach).

Rysunek 6.6. Średnia dokładność klasyfikacji oraz średnia liczba cech dla kolejnych metod (uśrednienie po 5 zbiorach).

Rysunek 6.7. Średnia dokładność klasyfikacji oraz średnia liczba cech dla kolejnych metod (uśrednienie po 11 zbiorach).