WYBRANE ZAGADNIENIA SYMULACJI KOMPUTEROWEJ

REDAKCJA NAUKOWA

ZENON A. SOSNOWSKI

Politechnika Białostocka

WYBRANE ZAGADNIENIA SYMULACJI KOMPUTEROWEJ

pod redakcją naukową
Zenona A. Sosnowskiego

Politechnika Białostocka
OFICYNA WYDAWNICZA POLITECHNIKI BIAŁOSTOCKIEJ
BIAŁYSTOK 2024

Recenzenci:
dr hab. inż. Piotr Kisielewski, prof. PK
dr hab. inż. Tadeusz Nowicki, prof. WAT

Redaktor naukowy dyscypliny informatyka techniczna i telekomunikacja:
prof. dr hab. Jarosław Stepaniuk

Korekta językowa:
Edyta Chrzanowska

Zdjęcie na okładce:
u_3u3n7wt5sr,
https://pixabay.com/pl/illustrations/wykres-technologia-przysz%C5%82y-numer-8305514/
Chihiro23,
https://pixabay.com/pl/illustrations/matryca-przetwarzanie-danych-7178709/

Skład i okładka:
Marcin Dominów

ISBN 978-83-68077-56-8 (e-Book)
DOI: 10.24427/978-83-68077-56-8

Oficyna Wydawnicza Politechniki Białostockiej
ul. Wiejska 45C, 15-351 Białystok
e-mail: oficyna.wydawnicza@pb.edu.pl
www.pb.edu.pl

Spis treści

Wstęp

Modelowanie i symulacja komputerowa są kluczowymi elementami współczesnej metodologii badań naukowych. Symulacja komputerowa rozwija się dzięki współdziałaniu wiedzy oraz metodologii nauk matematycznych, przyrodniczych, technicznych, społecznych i obliczeniowych.

W niniejszej monografii zaprezentowano wybrane zagadnienia odnoszące się do wykorzystania metod modelowania i symulacji komputerowej w szerokim zakresie tematycznym związanym z: klasyfikacją wysokowymiarowych (w tym niezbalansowanych) zbiorów danych, systemami rekomendacyjnymi, biometrią behawioralną, przetwarzaniem języka naturalnego, systemami identyfikacji gatunkowej, wykrywaniem złośliwego oprogramowania, planowaniem tras na odcinku ostatniej mili, obliczeniami kwantowymi, zadaniem kodowania stanów automatu skończonego. Materiał obejmuje prace badawcze oraz o charakterze badawczo-rozwojowym.

W rozdziale pierwszym *Wierzchołkowe balansowanie wysokowymiarowych zbiorów danych* Leon Bobrowski przedstawił metody projektowania warstw klasyfikatorów liniowych (neuronów formalnych) opartych na wierzchołkach działających w różnych podprzestrzeniach wielowymiarowej przestrzeni cech. Zgodnie z zaproponowaną przez autora koncepcją szkodliwe efekty niezbalansowania liczb wektorów cech w poszczególnych zbiorach uczących można redukować za pomocą zbalansowanych warstw detektorów poszczególnych kategorii.

W rozdziale drugim *Klasyfikacja oparta na wielu warstwach detektorów*, autorstwa Pawła Zabielskiego, przedstawiono zagadnienia dotyczące możliwości skonstruowania wielu warstw detektorów, które służą do ostatecznej klasyfikacji obiektów. Wszystkie utworzone warstwy detektorów są wykorzystywane w ostatecznej klasyfikacji obiektów poprzez głosowanie. Ta zaawansowana metoda klasyfikacji może znaleźć zastosowanie w dziedzinach, w których istnieje zapotrzebowanie na skuteczne modele klasyfikacyjne na małych, ale wysokowymiarowych zbiorach danych.

Jerzy Krawczuk w rozdziale trzecim *Systemy rekomendacyjne* przedstawia przegląd metodologii i technik stosowanych w budowie zaawansowanych systemów rekomendacyjnych. Autor przeanalizował kluczowe koncepcje i algorytmy zarówno filtrowania opartego na treści, jak i filtrowania kolaboratywnego, podkreślając ich mocne strony oraz wyzwania, które niosą ze sobą w praktycznych implementacjach. Omówił też podejścia hybrydowe, które mogą być remedium na bolączki pojedynczych metod.

W rozdziale czwartym Daniel Grabowski i Aleksander Sawicki opisują *Zastosowanie klasyfikatorów hierarchicznych w biometrii behawioralnej chodu*, której celem

jest identyfikacja osób na podstawie sposobu poruszania się. Rezultaty wskazują, że wykorzystanie klasyfikatorów hierarchicznych, w których na pierwszym etapie następuje podział pełnego zbioru na podzbiory (według predykcji wytrenowanego klasyfikatora), może mieć pozytywny wpływ na skuteczność systemu biometrycznego.

Jakub Zaprzałka i Magdalena Topczewska, autorzy rozdziału piątego *Porównanie wybranych technik głębokiego przetwarzania języka naturalnego w kontekście uczenia nienadzorowanego*, przedstawiają kilka głębokich modeli uczenia maszynowego w różnych zadaniach przetwarzania języka naturalnego. Wybrali najszerzej wykorzystywane modele, takie jak RoBERTa czy BART, które są też stosowane w kontekście grupowania aglomeracyjnego. Przedstawili również użycie poszczególnych elementów dużych sieci neuronowych oraz wpływ ich wyboru na jakość grupowania.

W rozdziale szóstym *System do rozpoznawania gatunków muzycznych* Szymon Górski i Anna Łupińska-Dubicka objaśniają koncept gatunku muzycznego, definiując go w kontekście informatycznym, oraz wskazują sygnał dźwiękowy jako medium łączące teorię muzyki z informatyką. Autorzy przedstawili bazujący na sztucznej inteligencji system rozpoznawania gatunków muzycznych oraz ewaluację jego dokładności. Do modelowania predykcji wykorzystali konwolucyjne sieci neuronowe.

Patryk Ostrowski w rozdziale siódmym *Metody wykrywania złośliwego oprogramowania na platformie Android z wykorzystaniem algorytmów uczenia maszynowego* przedstawia metodę ekstrakcji danych z dużych zbiorów aplikacji mobilnych oraz określa wpływ wykorzystania różnych algorytmów selekcji zmiennych i optymalizacji hiperparametrów na skuteczność wykrywania złośliwego oprogramowania.

Jolanta Koszelew, Krzysztof Ostrowski, Grażyna Starzec i Mateusz Starzec, autorzy rozdziału *Optymalizacja procesu przydzielania zleceń transportowych oraz generowania optymalnych tras dostaw realizowanych na odcinku ostatniej mili*, omówili badania symulacyjne w procesie przygotowania i przeprowadzenia testów oryginalnego algorytmu rozwiązującego problem TOP (ang. *Team Orienteering Problem*) przy użyciu ogólnodostępnych benchmarków sieci transportowej. Przedstawili także wpływ wyników badań na realizację komercyjnego projektu B+R, którego efekty zostały wdrożone w systemie AutoHiver.

W rozdziale dziewiątym, *Symulacja obliczeń kwantowych*, Joanna Wiśniewska i Marek Sawerwain omawiają kilka autorskich pakietów wspomagających symulację wybranych zagadnień z obszaru obliczeń kwantowych, takich jak obwody kwantowe w małej skali z pełnymi możliwościami podglądu stanu, co naturalnie jest możliwe tylko w przypadku symulacji. Zaprezentowali pakiet do obliczeń związanych z metodą kwantowych trajektorii, pakiet EntDetector wspomagający pracę ze splątanymi stanami kwantowymi oraz pokrótce pakiet QDC zawierający wybrane metody uczenia maszynowego, jakie obecnie są dostępne w ramach modelu obliczeń kwantowych.

Valery Salauyou w rozdziale dziesiątym *Sekwencyjne metody kodowania stanów automatów skończonych* przedstawia autorskie metody, które są efektywne pod względem obszaru (kosztu implementacji) i wydajności (szybkości) oraz koncentrują się na implementacji automatów skończonych w CPLD (ang. *Complex Programmable Logic Device*), FPGA (ang. *Field Programmable Gate Array*) lub ASIC

(ang. *Application-Specific Integrated Circuit*). Metody te opierają się na sekwencyjnym wyborze stanu automatu do zakodowania i znalezieniu najbardziej odpowiedniego kodu do wybranego stanu. Wybór stanu uwzględnia jego relacje z już zakodowanymi stanami, liczbę przejść przychodzących i wychodzących oraz liczbę zmiennych wejściowych, które inicjują przejścia do danego stanu. Wybór kodu minimalizuje wzrost obszaru.

Zespół autorski niniejszej monografii ma nadzieję, że treści w niej zawarte zainteresują potencjalnych czytelników. Wiele osób pomogło w jej przygotowaniu. Chciałbym podziękować autorom poszczególnych rozdziałów, recenzentom z Polskiego Towarzystwa Symulacji Komputerowej za pomoc w procesie selekcji materiału, a Wydziałowi Informatyki PB za wspieranie naszych wysiłków.

Zenon A. Sosnowski
Białystok, lipiec 2024 r.

Rozdział I
Wierzchołkowe balansowanie wysokowymiarowych zbiorów danych

Leon Bobrowski

Wydział Informatyki, Politechnika Białostocka
Instytut Biocybernetyki i Inżynierii Biomedycznej, PAN w Warszawie

Streszczenie: W rozdziale opisana jest metoda projektowania warstw klasyfikatorów liniowych (neuronów formalnych) opartych na wierzchołkach działających w różnych podprzestrzeniach wielowymiarowej przestrzeni cech. Zgodnie z proponowaną koncepcją szkodliwe efekty niezbalansowania liczb wektorów cech w poszczególnych zbiorach uczących można redukować za pomocą zbalansowanych warstw detektorów poszczególnych kategorii.

Słowa kluczowe: wysokowymiarowe zbiory danych, niezbalansowanie zbiorów uczących, wierzchołki optymalne, detektory klas, zbalansowane warstwy detektorów

1. Wstęp

Klasyfikatory przypisujące wektory cech do wybranych klas (kategorii) mogą być budowane na podstawie zbiorów uczących [1]. Reprezentują one poszczególne klasy i składają się z wektorów cech wcześniej przypisanych przez ekspertów do tych klas.

Zbiory uczące są niezrównoważone (niezbalansowane), gdy ich liczności bardzo różnią się między sobą [2]. Niezrównoważone zbiory uczące utrudniają projektowanie klasyfikatorów o wystarczająco dobrej jakości. Niezbalansowanie tych zbiorów prowadzi do niekorzystnego zjawiska dominacji klasyfikatorów opartych na najliczniejszych zbiorach.

Podstawową i powszechnie stosowaną metodą balansowania zbiorów uczących jest obecnie metoda SMOTE, która polega na zwiększaniu lub zmniejszaniu liczby wektorów cech w poszczególnych zbiorach [2]. W rozdziale opisano alternatywną koncepcję balansowania kategorii. Niezbalansowana reprezentacja kategorii za pomocą wektorów cech jest zastępowana zbalansowaną reprezentacją przy użyciu detektorów klas opartych na wierzchołkach optymalnych w przestrzeni parametrów [3].

2. Wysokowymiarowe zbiory uczące

Bierzemy pod uwagę zbiory danych zbudowane z n-wymiarowych *wektorów cech* $\mathbf{x}_j = [x_{j,1},...,x_{j,n}]^T$ ($\mathbf{x}_j \neq \mathbf{0}$) [1]. Poszczególne składowe $x_{j,i}$ wektora cech $\mathbf{x}_j$ są liczbowymi wynikami pomiarów (*cech*) X_i obiektu O_j, gdzie $i = 1,..., n$ oraz $j = 1,..., m$. Wartości $x_{j,i}$ poszczególnych cech X_i mogą być liczbami rzeczywistymi ($x_{j,i} \in R^1$) lub binarnymi ($x_{j,i} \in \{0,1\}$). Zbiory danych są wysokowymiarowe, gdy liczba m wektorów cech $\mathbf{x}_j$ jest znacznie mniejsza niż wymiar n tych wektorów ($m << n$).

Zakładamy, że m obiektów O_j reprezentowanych przez wektory cech $\mathbf{x}_j$ zostało przypisanych zgodnie z dodatkową wiedzą a priori do poszczególnych kategorii (klas) ω_k ($k = 1,\ldots, K$). Na tej podstawie zostały sformowane tzw. *zbiory treningowe* $\boldsymbol{C}_k$ [4]:

$$(\forall k \in \{1,\ldots, K\}) \quad \boldsymbol{C}_k = \{\mathbf{x}_j(k): j \in J_k\} \tag{1}$$

gdzie J_k jest podzbiorem m_k indeksów j obiektów O_j przypisanych do kategorii ω_k.

Treningowy zbiór danych $\boldsymbol{C}_k$ (1) reprezentuje klasę ω_k za pomocą m_k wektorów cech $\mathbf{x}_j(k)$ powiązanych z k-tą kategorią ω_k na podstawie dodatkowej wiedzy eksperckiej. Zbiory $\boldsymbol{C}_k$ są niezbalansowane, jeżeli ich liczebności m_k znacznie różnią się między sobą. Niezbalansowanie treningowych zbiorów danych $\boldsymbol{C}_k$ (1) może obniżyć jakość klasyfikatorów projektowanych na ich podstawie. To zaś wiąże się ze szkodliwym pomniejszaniem znaczenia tych kategorii ω_k, które są reprezentowane przez małe liczby m_k wektorów cech $\mathbf{x}_j(k)$ w zbiorach treningowych $\boldsymbol{C}_k$ (1).

Na podstawie zbiorów danych $\boldsymbol{C}_k$ (1) tworzone są pary $\{\boldsymbol{G}_{k(l)}{}^{!+}, \boldsymbol{G}_{k(l)}{}^{-}\}$ zbiorów uczących $\boldsymbol{G}_{k(l)}{}^{+}$ i $\boldsymbol{G}_{k(l)}{}^{-}$ dla poszczególnych kategorii ω_k, gdzie $l = 1, \ldots, L_k$ [4]. Dodatni zbiór uczący $\boldsymbol{G}_{k(l)}{}^{+}$ zawiera $m_{k(l)}+$ ($m_{k(l)}{}^{+} \leq m_k$) przykładowych wektorów cech $\mathbf{x}_j(k)$ ze zbioru $\boldsymbol{C}_k$ (1)

$$(\forall k \in \{1,\ldots, K\}) (\forall l \in \{1,\ldots, L_k\}) \quad \boldsymbol{G}_{k(l)}+ \subset \boldsymbol{C}_k \tag{2}$$

Podobnie ujemny zbiór uczący $\boldsymbol{G}_{k(l)}{}^{-}$ zawiera $m_{k(l)}{}^{-}$ przykładowych wektorów cech $\mathbf{x}_j(k')$ z innych zbiorów treningowych $\boldsymbol{C}_{k'}$ ($k' \neq k$):

$$(\forall k \in \{1,\ldots, K\}) (\forall l \in \{1,\ldots, L_k\}) \quad \boldsymbol{G}_{k(l)}{}^{-} \subset \cup_{k' \neq k} \boldsymbol{C}_{k'} \tag{3}$$

Ujemny zbiór uczący $\boldsymbol{G}_{k(l)}{}^{-}$ może reprezentować l-ty *kontekst* kategorii ω_k.

Definicja 1: Zbiory uczące $\boldsymbol{G}_{k(l)}{}^{+}$ (2) i $\boldsymbol{G}_{k(l)}{}^{-}$ (3) są liniowo separowalne w n-wymiarowej przestrzeni cech $\boldsymbol{F}[n]$ ($\mathbf{x} \in \boldsymbol{F}[n]$), jeżeli poniższy układ nierówności liniowych ma rozwiązanie:

$$(\forall \mathbf{x}_j(k) \in \boldsymbol{G}_{k(l)}{}^{+}) \quad \mathbf{x}_j(k)^T \mathbf{w}_k \geq 1 \tag{4}$$

oraz

$$(\forall \mathbf{x}_j(k') \in \boldsymbol{G}_{k(l)}{}^{-}) \quad \mathbf{x}_j(k')^T \mathbf{w}_k \leq -1 \tag{5}$$

gdzie $\mathbf{w}_k = [w_{k,1},..., w_{k,n}]^T$.

Liniowa separowalność (4), (5) zbiorów uczących $\boldsymbol{G}_{k(l)}^{+}$ (2) i $\boldsymbol{G}_{k(l)}^{-}$ (3) oznacza, że zbiory te mogą być rozdzielone za pomocą poniższej hiperpłaszczyzny $\boldsymbol{H}(\mathbf{w}_k)$ w przestrzeni cech $\boldsymbol{F}[n]$:

$$\boldsymbol{H}(\mathbf{w}_k) = \{\mathbf{x}: \mathbf{w}_k^{T}\mathbf{x} = 0\} \tag{6}$$

Hiperpłaszczyzna $\boldsymbol{H}(\mathbf{w}_k)$ (6) rozdziela zbiory $\boldsymbol{G}_{k(l)}^{+}$ (2) i $\boldsymbol{G}_{k(l)}^{-}$ (3) z marginesem $\delta(\mathbf{w}_k)$ [5]:

$$(\forall \mathbf{x}_j(k) \in \boldsymbol{G}_{k(l)}^{+}) \quad \mathbf{w}_k^{T}\mathbf{x}_j(k) \geq \delta(\mathbf{w}_k) \tag{7}$$

oraz

$$(\forall \mathbf{x}_j(k') \in \boldsymbol{G}_k(l)^{-}) \quad \mathbf{w}_k^{T}\mathbf{x}_j(k') \leq -\delta(\mathbf{w}_k) \tag{8}$$

gdzie

$$\delta(\mathbf{w}_k) = 1 / \| \mathbf{w}_k \| \tag{9}$$

oraz $\| \mathbf{w}_k \|$ jest długością (*normą*) wektora $\mathbf{w}_k$ wyznaczającego hiperpłaszczyznę $\boldsymbol{H}(\mathbf{w}_k)$ (6).

Liniową separowalność (4), (5) zbiorów uczących $\boldsymbol{G}_{k(l)}^{+}$ (2) i $\boldsymbol{G}_{k(l)}^{-}$ (3) można weryfikować poprzez minimalizację perceptronowej funkcji kryterialnej zdefiniowanej na wektorach cech $\mathbf{x}_j(k)$ należących do tych zbiorów [3]. Perceptronowa funkcja kryterialna jest funkcją wypukłą i odcinkowo-liniową (ang. *convex and piecewise-linear*, CPL). Do rodziny funkcji typu CPL należą kolinearna funkcja kryterialna oraz funkcja regularyzacyjna z normą L_1 [5]. Praktyczną ważną właściwością funkcji kryterialnych typu CPL jest, że ich minimum globalne może być wyznaczone w jednym z wierzchołków $\mathbf{w}_k$ w przestrzeni parametrów.

3. Hiperpłaszczyzny dualne i wierzchołki w przestrzeni parametrów

Hiperpłaszczyzny dualne h_j^1 definiowane są za pomocą m ($m = m_1 + \ldots + m_K$) wektorów cech $\mathbf{x}_j$ tworzących zbiory uczące $\boldsymbol{C}_k$ (1):

$$(\forall j \in \{1,\ldots, m\}) \quad h_j^1 = \{\mathbf{w} \in \boldsymbol{R}^n: \mathbf{x}_j(k)^{T}\mathbf{w} = 1\} \tag{10}$$

Hiperpłaszczyzny dualne h_i^0 definiowane są w n-wymiarowej przestrzeni parametrów (wag) za pomocą n wektorów jednostkowych $\mathbf{e}_i$, gdzie $m << n$:

$$(\forall i \in \{1,\ldots, n\}) \quad h_i^0 = \{\mathbf{w} \in \boldsymbol{R}^n : \mathbf{e}_i^T \mathbf{w} = 0\} \tag{11}$$

Hiperpłaszczyzny dualne h_j^1 (10) wiązane są z perceptronową funkcją kryterialną [3]. Hiperpłaszczyzny h_i^0 (11) wiązane są z funkcjami regularyzacyjnymi opartymi na normie L_1 [5].

Definicja 1: Wierzchołek $\mathbf{w}_k(r)$ rzędu r jest punktem przecięcia się r hiperpłaszczyzn h_j^1 (10) oraz $n - r$ hiperpłaszczyzn h_i^0 (11).

Wierzchołek $\mathbf{w}_k(r)$ jest rozwiązaniem poniższego układu n równań liniowych:

$$(\forall j \in J_k(r)) \quad \mathbf{w}_k(r)^T \mathbf{x}_j = 1 \tag{12}$$

oraz

$$(\forall i \in I_k(n - r)) \quad \mathbf{w}_k(r)^T \mathbf{e}_i = 0 \tag{13}$$

gdzie $J_k(r))$ jest podzbiorem indeksów j takich hiperpłaszczyzn h_j^1 (10) wyznaczonych przez r liniowo niezależnych wektorów cech $\mathbf{x}_j$ ze zbioru uczącego $\boldsymbol{C}_k$ (1), które przechodzą przez wierzchołek $\mathbf{w}_k(r)$.

$I_k(n - r)$ jest podzbiorem $n - r$ indeksów i takich wektorów jednostkowych $\mathbf{e}_i$, które wyznaczają hiperpłaszczyzny h_i^0 (11) przechodzące przez wierzchołek $\mathbf{w}_k(r)$. Z równania (13) wynika, że $n - r$ składowych $w_{k,i}$ wektora $\mathbf{w}_k = [w_{k,1},..., w_{k,n}]^T$ (15) jest równe zeru

$$(\forall i \in I_k(n - r)) \; w_{k,i} = 0 \tag{14}$$

Równania (12) i (13) można przedstawić w postaci macierzowej:

$$\boldsymbol{X}(r)\, \mathbf{w}_k(r) = \mathbf{I}_k(r) \tag{15}$$

gdzie $\mathbf{I}_k(r) = [1,...,1,0,...,0]^T$ oraz $\boldsymbol{X}(r) = [\mathbf{x}_{j(1)},..., \mathbf{x}_{j(r)}, \mathbf{e}_{i(1),,,,,}, \mathbf{e}_{i(n-r)}]^T$ jest macierzą nieosobliwą (bazą) zbudowaną z r ($r \le m_k$) liniowo niezależnych wektorów cech $\mathbf{x}_j$ ze zbioru $\boldsymbol{C}_k$ (1) oraz z $n - r$ wektorów jednostkowych $\mathbf{e}_i$ ($i \in I_k(n - r)$ (13)).

Wierzchołek $\mathbf{w}_k(r)$ jest sumą r pierwszych kolumn $\mathbf{r}_i$ macierzy odwrotnej $\boldsymbol{X}(r)^{-1}$ $=[\mathbf{r}_j,...., \mathbf{r}_n]$:

$$\mathbf{w}_k(r) = \boldsymbol{X}(r)^{-1}\, \mathbf{I}_k(r) = \mathbf{r}_1 + ...+ \mathbf{r}_r \tag{16}$$

Z wierzchołkiem $\mathbf{w}_k = [w_{k,1},..., w_{k,n}]^T$ (16) można związać margines δ_{L1} (9) w normie L_1 [5]:

$$\delta_{L1} = \delta_{L1}(\mathbf{w}_k) = 1 \,/\, (\Sigma_i \mid w_{k,i} \mid) \tag{17}$$

Minimalizacja perceptronowej funkcji kryterialnej z regularyzacją typu L_1 pozwala wyznaczyć wierzchołek optymalny $\mathbf{w}_k^*(r)$ o największym marginesie $\delta_{L1}(\mathbf{w}_k^*(r))$ (17) spośród wszystkich wierzchołków $\mathbf{w}_k(r)$ (16)) rzędu r [4]

$$(\forall \mathbf{w}_k(r)\ (15))\ \ \delta_{L1}(\mathbf{w}_k^*(r)) \geq \delta_{L1}(\mathbf{w}_k(r))\ \} \tag{18}$$

Z wierzchołkiem optymalnym $\mathbf{w}_k^*(r)$ (17) związany jest optymalny podzbiór $F_k^*(r)$ r cech X_i:

$$F_k^*(r) = \{\ X_{i(1)},....,\ X_{i(r)}\} \tag{19}$$

Cechy X_i w podzbiorze $F_k^*(r)$ (19) wybierane są z początkowego zbioru $F_0(n) = \{X_1,\ldots, X_n\}$ n cech X_i ($F_k^*(r) \subset F_0(n)$). Podzbiór optymalny $F_k^*(r)$ (19) zawiera takie r cech X_i, które są związane z niezerowymi wagami $w_{k,i}^*$ ($w_{k,i}^* \neq 0$) (14) w wierzchołku optymalnym

$$\mathbf{w}_k^* = [w_{k,1}^*,..., w_{k,n}^*]^T \tag{17}$$

4. Wierzchołkowe reguły klasyfikacyjne

Zbiory treningowe $\boldsymbol{C}_k$ (1) pozwalają projektować różnego typu reguły klasyfikacyjne $d(\mathbf{x})$, które lokują obiekty O reprezentowane za pomocą wektorów cech $\mathbf{x}$ do poszczególnych klas ω_k:

$$(\forall \mathbf{x} \in \boldsymbol{F}(n))\ \ d(\mathbf{x}) \in \{1,\ldots, K\} \tag{20}$$

gdzie $\boldsymbol{F}(n)$ jest *przestrzenią cech* zawierającą n-wymiarowe wektory cech $\mathbf{x} = [x_1,...,x_n]^T$.

Reguła $d(\mathbf{x})$ (20) przypisuje wektor cech $\mathbf{x}$ ($\mathbf{x} \in \boldsymbol{F}(n)$) do kategorii ω_k o numerze $k = d(\mathbf{x})$. W zadaniach projektowania klasyfikatorów na podstawie zbiorów treningowych $\boldsymbol{C}_k$ (1) użyteczne są liniowe reguły klasyfikacyjne $d(\mathbf{w}_k^*; \mathbf{x})$ oparte na wierzchołkach optymalnych $\mathbf{w}_k^*$ (16) [5]:

$$(\forall \mathbf{x} \in \boldsymbol{F}(n))\ (\forall k \in \{1,\ldots, K\})$$
$$\textit{if}\ (\mathbf{w}_k^*)^T\mathbf{x} > 0\ \textit{then}\ d(\mathbf{w}_k^*; \mathbf{x}) = k, \textit{else}\ d(\mathbf{w}_k^*; \mathbf{x}) = 0 \tag{21}$$

gdzie $\mathbf{w}_k^* = [w_{k,1},..., w_{k,n}]^T \in \boldsymbol{R}^n$ jest wierzchołkiem reprezentującym k-tą kategorię ω_k w przestrzeni parametrów.

Warunek $d(\mathbf{w}_k; \mathbf{x}) = 0$ (21) oznacza, że reguła $d(\mathbf{w}_k; \mathbf{x})$ nie wyznacza k-tej kategorii ω_k. Reguła klasyfikacyjna $d(\mathbf{w}_k^*; \mathbf{x})$ (21) może wskazać dla wektora cech $\mathbf{x}$ więcej niż jedną kategorię ω_k. Wierzchołki optymalne $\mathbf{w}_k^*$ (18) reprezentujące w przestrzeni parametrów poszczególne kategorie ω_k wyznaczane są poprzez minimalizację perceptronowej funkcji kryterialnej z regularyzacją L_1.

5. Zbalansowane warstwy detektorów klas

Reguła klasyfikacyjna $d(\mathbf{w}_k^*; \mathbf{x})$ (21) charakteryzuje pojedynczy detektor k-tej kategorii ω_k oparty na jednym wierzchołku optymalnym $\mathbf{w}_k^*$ (18) związanym z podzbiorem $F_k^*(r)$ (19) *r cech aktywnych* X_i. W celu zwiększenia poprawności klasyfikacji można zastosować zbalansowane warstwy detektorów poszczególnych kategorii ω_k [4].

Załóżmy, że projektowana warstwa ma obejmować L_k detektorów k-tej kategorii ω_k opartych na wierzchołkach optymalnych $\mathbf{w}_{k(l)}^*$, projektowanych w kolejnych krokach l ($l = 1,\ldots, L_k$). Pierwszy detektor w warstwie oparty jest na wierzchołku optymalnym $\mathbf{w}_{k(1)}^*$ (18) związanym z optymalnym zestawem $F_{k(1)}^*(r)$ (19) r cech aktywnych $X_{i(l)}$. Zestaw $F_{k(1)}^*(r)$ (19) r cech aktywnych $X_{i(l)}$ wybrany został ze zbioru początkowego $F_0(n) = \{X_1,\ldots, X_n\}$ n cech X_i.

Optymalny zestaw $F_{k(2)}^*(r)$ (19) r cech aktywnych $X_{i(l)}$ w kroku drugim ($l = 2$) wybierany jest ze zredukowanego zestawu $F_1(n - r)$ $n - r$ cech X_i nadal dostępnych do zużycia:

$$F_1(n - r) = F_0(n) \setminus F_{k(1)}^*(r) = \{X_1,\ldots, X_n\}\setminus\{X_{i(1)},\ldots., X_{i(r)}\} \tag{22}$$

W kroku drugim obliczany jest podobnie wierzchołek optymalny $\mathbf{w}_{k(2)}^*$ (18) związany z optymalnym zestawem $F_{k(2)}^*(r)$ (19) r cech aktywnych $X_{i(l)}$. Wierzchołek optymalny $\mathbf{w}_{k(2)}^*$ (18) definiuje w projektowanej warstwie drugi detektor (21) k-tej kategorii ω_k. W kolejnych L_k krokach l projektowana jest warstwa L_k detektorów k-tej kategorii ω_k. Reguła decyzyjna $d_k(\mathbf{w}_{k(1)}^*,\ldots, \mathbf{w}_{k(Lk)}^*; \mathbf{x})$ k-tej warstwy L_k detektorów może zostać zdefiniowana jako wartość średnia reguł $d(\mathbf{w}_{k(l)}^*; \mathbf{x})$ (21):

$$(\forall \mathbf{x} \in \boldsymbol{F}(n)) \;\; (\forall k \in \{1,\ldots, K\})$$

$$d_k(\mathbf{w}_{k(1)}^*,\ldots, \mathbf{w}_{k(Lk)}^*; \mathbf{x}) = (d(\mathbf{w}_{k(1)}^*; \mathbf{x}) + \ldots + d(\mathbf{w}_{k(Lk)}^*; \mathbf{x})) / L_k \tag{23}$$

Proces projektowania warstwy L_k detektorów k-tej kategorii ω_k może zostać powtórzony dla każdej z K kategorii ω_k ($k = 1,\ldots, K$). W ten sposób może zostać zbudowany kompleksowy klasyfikator agregujący K warstw (23) detektorów poszczególnych kategorii ω_k [4].

Reguła decyzyjna klasyfikatora agregującego K warstw (23) może mieć poniższą postać większościową:

$$\begin{aligned} &\textit{Wektor cech } \mathbf{x}\ (\mathbf{x} \in \boldsymbol{F}(n)) \textit{ jest przypisany do kategorii } \omega_k{}^* \\ &\textit{o największej wartości średniej } d_k(\mathbf{w}_{k(1)}{}^*, \ldots, \mathbf{w}_{k(Lk)}{}^*; \mathbf{x})\ (23). \end{aligned} \tag{24}$$

Większościowa reguła klasyfikacyjna (24) może zostać zbalansowana ze względu na wszystkie kategorie ω_k. Każda z K kategorii ω_k jest reprezentowana w takim klasyfikatorze przez zbalansowaną warstwę o takiej samej liczbie L_k detektorów opartych na wierzchołkach optymalnych $\mathbf{w}_{k(l)}{}^*$ (18) tego samego rzędu r.

6. Uwagi końcowe

Niezbalansowanie treningowych zbiorów danych $\boldsymbol{C}_k$ (1) jest ważnym problemem, ponieważ może prowadzić do obniżenia jakości klasyfikatorów projektowanych na podstawie tych zbiorów. Wiąże się to z pomniejszaniem znaczenia lub nawet pomijaniem tych kategorii ω_k, które są ważne w praktyce, ale są reprezentowane przez małe liczby m_k wektorów cech $\mathbf{x}_j(k)$ w zbiorach danych [2].

Podstawową i powszechnie stosowaną metodą balansowania zbiorów danych jest obecnie metoda SMOTE (*synthetic minority over-sampling technique*), która polega na zwiększaniu lub zmniejszaniu liczby wektorów cech w zbiorach danych reprezentujących poszczególne kategorie ω_k [2].

W niniejszym rozdziale opisano alternatywną metodę reprezentacji kategorii ω_k. Niezbalansowana reprezentacja kategorii ω_k za pomocą wysokowymiarowych zbiorów wektorów cech $\mathbf{x}_j(k)$ może być zastąpiona reprezentacją w przestrzeni parametrów za pomocą zbalansowanych warstw detektorów klas (24) opartych na wierzchołkach optymalnych $\mathbf{w}_k{}^*(r)$ (17). Uzasadnienie proponowanej metody może być formułowane na bazie twierdzenia ergodycznego.

Bibliografia

[1] Duda O.R., Hart P.E., Stork D.G., *Pattern classification*, John Wiley, New York 2001.

[2] Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P., *SMOTE: Synthetic minority over-sampling technique*, „Journal of Artificial Intelligence Research" 2002, vol. 16, s. 321–357.

[3] Bobrowski L., *Data Exploration and Linear Separability*, Lambert Academic Publishing, Chisinau 2019.

[4] Bobrowski L., *Complex Layers of Formal Neurons*, w: *Engineering Applications of Neural Networks*, red. L. Iliadis, Ch. Jayne, A. Tefas, E. Pimenidis, Springer, Cham 2022, s. 81–89.

[5] Bobrowski L., *Computing on vertices in data mining*, w: *Data mining – Concepts and applications*, red. C. Thomas, Intech Open, London 2022, s. 1–19.

Vertex balancing of high-dimensional data sets

Summary: The article describes a method for designing layers of linear classifiers (formal neurons) based on vertices in various subspaces of a multidimensional feature space. According to the proposed concept, the harmful effects of unbalanced numbers of feature vectors collected in individual training sets can be reduced by using balanced layers of detectors of individual categories.

Keywords: high-dimensional data sets, unbalanced training sets, optimal vectors, class detectors, balanced layers of detectors

Rozdział II
Klasyfikacja oparta na wielu warstwach detektorów

Paweł Zabielski
Wydział Informatyki, Politechnika Białostocka

Streszczenie: W rozdziale omówiono zaawansowaną metodę klasyfikacji opartą na wielu warstwach detektorów. Jest ona stosowana do analizy zbiorów danych, w których każdy obiekt jest opisany za pomocą wielowymiarowych wektorów cech. Proces konstrukcji warstw detektorów rozpoczyna się od niezależnej budowy detektorów dla każdej z klas w zbiorze danych. Wszystkie warstwy detektorów są wykorzystywane w ostatecznej klasyfikacji obiektów poprzez głosowanie. Ta zaawansowana metoda klasyfikacji może znaleźć zastosowanie w dziedzinach, w których istnieje zapotrzebowanie na skuteczne modele klasyfikacyjne na małych, ale wysokowymiarowych zbiorach danych. Warto zwrócić uwagę na to, że ta technika może być użyteczna m.in. w medycynie czy analizie danych ekonomicznych.

Słowa kluczowe: klasyfikacja, wielowarstwowe detektory, wielowymiarowe wektory cech, funkcja kryterialna, analiza danych

1. Wprowadzenie

Klasyfikacja danych jest kluczowym zagadnieniem w dziedzinie uczenia maszynowego i analizy danych. W ostatnich latach zwiększyło się zapotrzebowanie na rozwijanie bardziej zaawansowanych technik klasyfikacyjnych, zwłaszcza w przypadku małych, ale wysokowymiarowych zbiorów danych [1]. Generalnie przetwarzanie dużych zbiorów jest trudnym zadaniem [2].

Zaprezentowana w tym opracowaniu metoda stanowi innowacyjne podejście w dziedzinie klasyfikacji opartej na wielu warstwach detektorów. W celu ułatwienia zrozumienia tego zaawansowanego podejścia niniejszy tekst podzielono na kilka części. We wprowadzeniu przedstawiono kluczową rolę klasyfikacji w analizie danych, co pozwoli lepiej zrozumieć kontekst, w jakim działa prezentowana metoda. W drugiej części artykułu przedstawiono różne techniki wykorzystywane w procesie klasyfikacji obiektów. Trzecia część zawiera rozważane typy zbiorów danych. Następnie omówiono funkcję kryterialną użytą w algorytmie. Piąta część poświęcona jest głównym zależnościom i konceptom wykorzystanym w konstrukcji wielowarstwowych detektorów – to kluczowy element tego podejścia. Szósta część opisuje przeprowadzone eksperymenty. Na zakończenie przedstawiono wnioski wyciągnięte na podstawie

obserwacji i eksperymentów związanych z prezentowaną metodą. Ponadto pokazano przyszłe perspektywy rozwoju tej zaawansowanej techniki klasyfikacji.

2. Techniki klasyfikacji

Klasyfikacja obiektów jest fundamentalnym zagadnieniem w analizie danych, które ma zastosowanie w różnych dziedzinach, od rozpoznawania obrazów po medycynę i ekonomię. Istotą klasyfikacji jest przypisanie obiektów do określonych kategorii lub klas na podstawie ich cech bądź właściwości.

W tej sekcji omówiono niektóre z popularnych technik klasyfikacji, które stanowiły inspirację dla omawianej zaawansowanej metody. Wśród podstawowych koncepcji, zalet i ograniczeń tych technik można wyróżnić:

- regresję logistyczną, czyli podstawową technikę klasyfikacji wykorzystywaną w problemach binarnej klasyfikacji. Za pomocą tej metody obiekty są przypisywane do jednej z dwóch klas bazujących na modelu logistycznym;
- drzewa decyzyjne, które pozwalają na tworzenie hierarchicznych drzew decyzyjnych prowadzących do klasyfikacji obiektów. Ta technika jest stosunkowo prostym narzędziem, które jednak może dostarczyć cennych informacji o strukturze danych;
- maszyny wektorów nośnych (SVM), które są potężnym narzędziem do klasyfikacji i pozwalają wyznaczyć hiperpłaszczyzny oddzielające różne klasy w przestrzeni cech. To technika szczególnie użyteczna w przypadku danych, które nie są liniowo separowalne.

Wybór odpowiedniej techniki klasyfikacji zależy od natury danych i celów analizy. W kolejnych rozdziałach omówione zostaną powody decyzji wykorzystania zaawansowanej metody opartej na wielowarstwowych detektorach.

3. Typ zbiorów danych

W artykule przedstawiono typ zbiorów danych, które są wykorzystywane podczas obliczeń. Są to wysokowymiarowe elementy. Każda wejściowa kolekcja składa się z m wektorów cech $\mathbf{x}_j = [x_{j1},...,x_{jn}]^T$ z przestrzeni n wymiarowej.

Wektory cech $\mathbf{x}_j$ reprezentują m obiektów, gdzie każdy opisany jest przez numeryczne wartości przedstawiające właściwości/cechy każdego pacjenta. Ważnym elementem jest to, iż liczba osobników w rozważanych zbiorach danych jest dużo mniejsza niż liczba cech każdego z nich ($m << n$).

Dane składają się z dodatnich elementów, które liczą m_k^+ wektorów cech $\mathbf{x}_j^+$ reprezentujących m_k^+ pacjentów należących do klasy ω_1. Podobnie negatywny zbiór składa się z m_k^- wektorów cech $\mathbf{x}_j^-$ reprezentujących pacjentów należących do kategorii ω_2 [3].

4. Funkcja kryterialna

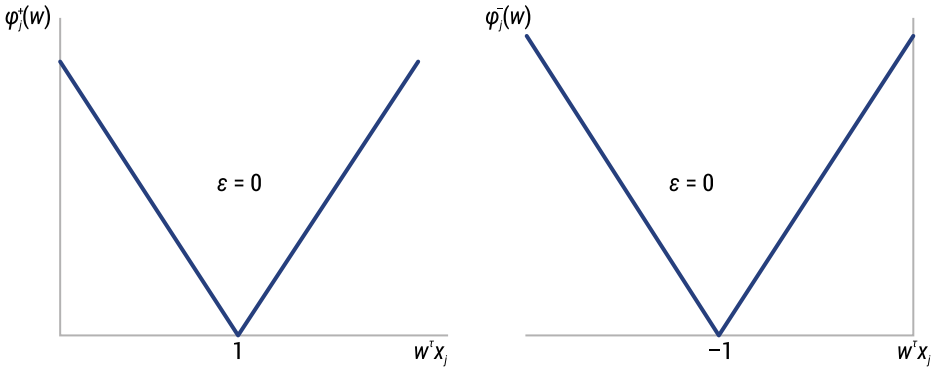
W zależności od obiektów odpowiedniej klasy zdefiniowana została właściwa funkcja kary. Wektory cech $\mathbf{x}_j^+$ ze zbioru elementów należących do klasy ω_1 umożliwiają utworzenie funkcji kary o następującym wzorze:

$$(\forall x_j^+) \quad \varphi_j^+(\mathbf{w}) = |1 - \mathbf{x}_j^T \mathbf{w}| \tag{4.1}$$

a elementy należące do drugiej klasy ω_2 pozwalają na zastosowanie podobnej funkcji kary wyrażonej w taki sposób:

$$(\forall x_j^-) \quad \varphi_j^-(\mathbf{w}) = |1 - \mathbf{x}_j^T \mathbf{w}| \tag{4.2}$$

Wykresy tych funkcji znajdują się na rys. 1.



RYS. 1. **Funkcje kary**

ŹRÓDŁO: **oprac. własne.**

Funkcja kryterialna elementów z klasy ω_1 oznaczona $\Phi_k^+(\mathbf{w})$ jest określona jako suma funkcji kar $\varphi_j^+(\mathbf{w})$ (4.1):

$$\Phi_k^+(\mathbf{w}) = \sum\nolimits_j \varphi_i^+(\mathbf{w}) \tag{4.3}$$

podobnie funkcja kryterialna wyliczona z obserwacji z klasy ω_2 oznaczona $\Phi_k^-(\mathbf{w})$ stanowi sumę funkcji kar $\varphi_j^-(\mathbf{w})$ (4.2):

$$\Phi_k^-(\mathbf{w}) = \sum\nolimits_j \varphi_i^-(\mathbf{w}) \tag{4.4}$$

Funkcje kryterialne $\Phi_k^+(\mathbf{w})$ (4.3) oraz $\Phi_k^-(\mathbf{w})$ (4.4) są wypukłe i odcinkowo-liniowe (typu CPL) [5].

Zaimplementowany algorytm wymiany rozwiązań bazowych ma na celu skuteczne znalezienie optymalnych wierzchołków w_k^+ oraz w_k^-. Wartości funkcji kryterialnych $\Phi_k^+(\mathbf{w})$ (4.3) oraz $\Phi_k^-(\mathbf{w})$ (4.4) osiągają w nich swoje minima [6].

5. Metoda klasyfikacji oparta na wielu warstwach detektorów

Proponowana metoda klasyfikacji opiera się na budowaniu wielu warstw detektorów, które są niezależnie tworzone na podstawie danych należących do odpowiednich klas. Liczba detektorów dla każdej klasy jest identyczna. Każdy z nich jest konstruowany przez minimalizację wypukłej i odcinkowo-liniowej funkcji kryterialnej (CPL) w celu znalezienia optymalnego wierzchołka [5]. Operują one na pomniejszonej przestrzeni cech, jednak wszystkie są tego samego wymiaru. Proces ten jest powtarzany dla każdej warstwy detektorów i kończy się po znalezieniu z góry określonej liczby detektorów. Wszystkie wykorzystane cechy w każdej warstwie są rozłączne.

Kluczowym elementem algorytmu jest to, że wszystkie warstwy detektorów są użyte w procesie ostatecznej klasyfikacji. Każda warstwa detektorów głosuje na przynależność obiektu do konkretnej klasy. Wartość głosowania reprezentuje odpowiedź danej warstwy w klasyfikacji danego obiektu.

6. Binarny klasyfikator

Na podstawie zbudowanych warstw detektorów ze zbiorów uczących należących odpowiednio do klas ω1 oraz ω2 następuje binarna klasyfikacja każdego z obiektów. Jako parametr algorytmu definiowana jest liczba L detektorów każdej klasy. Dlatego otrzymywane są $2L$ warstwy detektorów:

- L detektorów opartych na wierzchołkach zdegenerowanych wyliczonych na podstawie zbioru uczącego należącego do klasy ω_1;
- L detektorów opartych na wierzchołkach zdegenerowanych wyliczonych na podstawie zbioru uczącego należącego do klasy ω_2.

Każdy wektor podlegający klasyfikacji dzielony jest na podwektory zawierające cechy odpowiadające kolejnym warstwom detektorów. Wprowadzony na wejście każdej warstwy otrzymuje na wyjściu jedną z wartości: $Y_{j,l} \in \{1, 0, -1\}$. Dla warstw zbudowanych na podstawie pierwszej klasy zależy ona od następującego wzoru:

$$
\begin{aligned}
Y_{j,l}^+ &= 1, \; if \; \boldsymbol{w}_l^{+T} \boldsymbol{z}_l^+ > \varepsilon \\
Y_{j,l}^+ &= 0, \; if \; -\varepsilon \le \boldsymbol{w}_l^{+T} \boldsymbol{z}_l^+ \le \varepsilon \\
Y_{j,l}^+ &= -1, \; if \; \boldsymbol{w}_l^{+T} \boldsymbol{z}_l^+ < -\varepsilon
\end{aligned}
\tag{6.1}
$$

gdzie $\boldsymbol{w}_l^{+T}$ jest optymalnym wierzchołkiem danej warstwy klasy pierwszej, $\boldsymbol{z}_l^+$ kolejnym wektorem podlegającym klasyfikacji, ε marginesem, a rezultat $Y_{\mathrm{j},l}^+$ zwróconą wartością głosowania danej warstwy.

Podobnie warstwy zbudowane na podstawie drugiej klasy generują odpowiedzi zależne od następującego wzoru:

$$Y_{\mathrm{j},l}^- = -1, \textit{if } \boldsymbol{w}_l^{-T}\boldsymbol{z}_l^- > \varepsilon \tag{6.2}$$

$$Y_{\mathrm{j},l}^- = 0, \textit{if } -\varepsilon \le \boldsymbol{w}_l^{-T}\boldsymbol{z}_l^- \le \varepsilon$$

$$Y_{\mathrm{j},l}^- = 1, \textit{if } \boldsymbol{w}_l^{-T}\boldsymbol{z}_l^- < -\varepsilon$$

gdzie $\boldsymbol{w}_l^{-T}$ jest optymalnym wierzchołkiem danej warstwy klasy drugiej, $\boldsymbol{z}_l^-$ kolejnym wektorem podlegającym klasyfikacji, ε marginesem, a rezultat $Y_{\mathrm{j},l}^-$ zwróconą wartością głosowania danej warstwy.

Każda z warstw bierze udział w głosowaniu i wpływa na ostateczną klasyfikację. Wyliczana jest średnia wartość otrzymana na podstawie L warstw klasy pierwszej:

$$Y_{\mathrm{j}}^+(L) = (Y_{\mathrm{j},1}^+ + \ldots + Y_{\mathrm{j},L}^+)/L \tag{6.3}$$

W podobny sposób wyliczana jest średnia odpowiedź otrzymana z L detektorów klasy drugiej:

$$Y_{\mathrm{j}}^-(L) = (Y_{\mathrm{j},1}^- + \ldots + Y_{\mathrm{j},L}^-)/L \tag{6.4}$$

Zbiorcze odpowiedzi detektorów $Y_{\mathrm{j}}^+(L)$ (6.3) oraz $Y_{\mathrm{j}}^-(L)$ (6.4) biorą udział w ostatecznej decyzji, do której klasy będzie sklasyfikowany obiekt. Dla każdego z nich podejmowana jest decyzja na podstawie sformułowanych nierówności:

$$(\forall \mathbf{x}_{\mathrm{j}}) \text{ jeśli } \left(\left|Y_{\mathrm{j}}^+(L) - 1\right| < \left|Y_{\mathrm{j}}^-(L) + 1\right|\right), \text{ wtedy } \mathbf{x}_{\mathrm{j}} \text{ należy do klasy } \omega_1, \tag{6.5}$$

$$\text{jeśli } \left(\left|Y_{\mathrm{j}}^+(L) - 1\right| \ge \left|Y_{\mathrm{j}}^-(L) + 1\right|\right), \text{ wtedy } \mathbf{x}_{\mathrm{j}} \text{ należy do klasy } \omega_2.$$

Przedstawiona w artykule wersja algorytmu dostosowana jest do binarnej klasyfikacji (przynależności do jednej z dwóch klas ω_1 lub ω_2). Często jest to rozpatrywane jako przewidywanie chorób – pacjent chory lub pacjent zdrowy.

7. Eksperymenty i wyniki

Aby zweryfikować skuteczność proponowanej metody, przeprowadzono eksperymenty na zbiorze danych składającym się z obiektów z dwóch wielowymiarowych klas. Ich wyniki były obiecujące, co sugeruje, że metoda ta może być użyteczna w klasyfikacji danych o wysokim wymiarze.

Doświadczenia przeprowadzono z wykorzystaniem zbioru danych dotyczących raka jelita grubego (*Colon cancer* [7]). Ten zbiór danych obejmuje $m = 62$ próbki pobrane od pacjentów, podzielone na próbki nowotworowe i prawidłowe ($j = 1,\ldots, m$). Wśród nich znajduje się $m^+ = 40$ próbek nowotworowych i $m^- = 22$ próbki nienowotworowe. W rezultacie zbiór danych można podzielić na dwa podzbiory, dwie klasy – ω_1 reprezentuje pacjentów chorych na raka, a ω_2 składa się z wektorów cech reprezentujących pacjentów nienowotworowych. Każdy wektor cech ma wymiar $n = 2000$.

Celem eksperymentu było zbadanie rozpoznawania określonej klasy (choroby, ω_1) na podstawie zbiorów uczących, które składają się z niewielkiej liczby wielowymiarowych wektorów cech. Dane do zbiorów uczących dwóch klas ω_1 oraz ω_2 zostały wybrane w sposób losowy.

W eksperymentach wykorzystano dwie pary zbiorów uczących. Liczby losowo wybranych wektorów cech z każdej z klas do zbioru uczącego były następujące:

a) $m_1^+ = 5, m_1^- = 5,$

b) $m_2^+ = 10, m_2^- = 10.$

Liczbę warstw detektorów budowanych dla każdej z klas ustalono na $L = 50$.

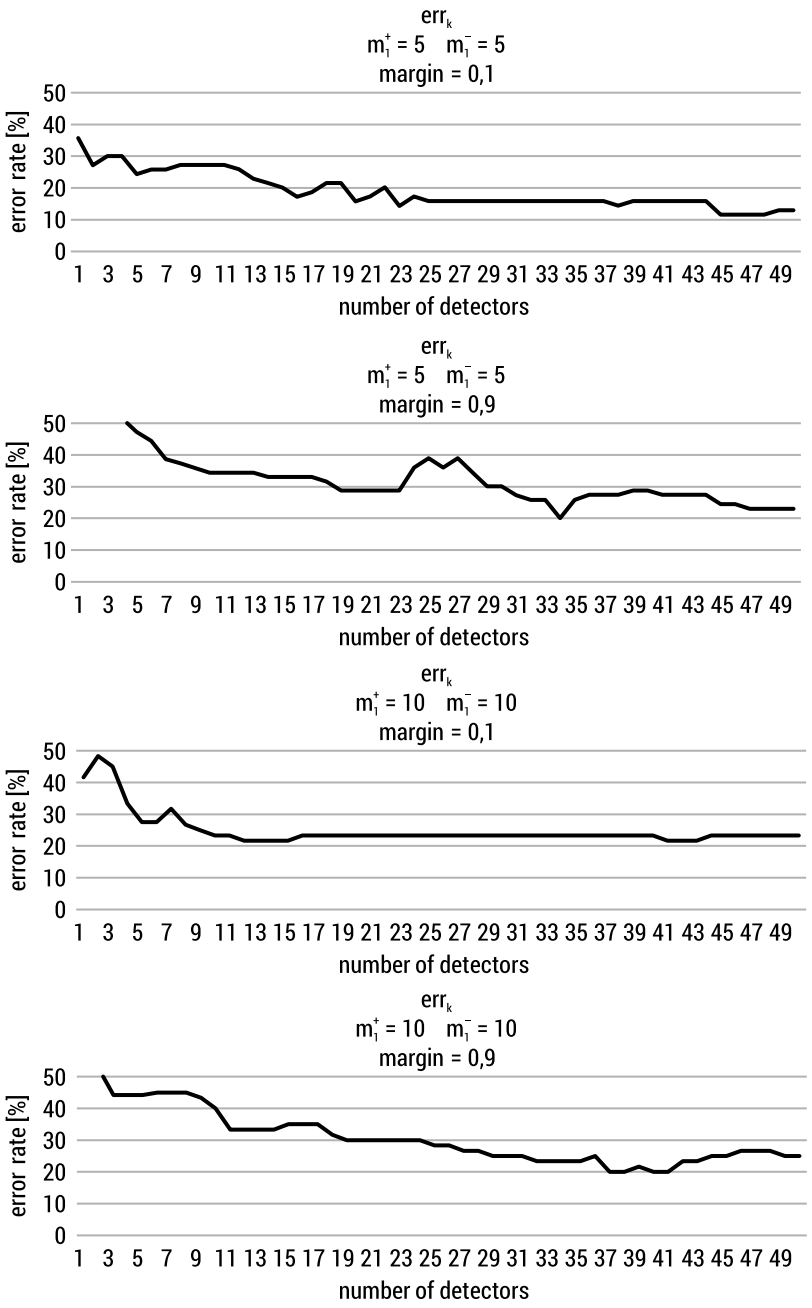
Ocena klasyfikatora (6.5) została przeprowadzona wyłącznie na wektorach cech, które nie były częścią zbiorów uczących. Gwarantuje to, że wektory cech użyte do projektowania klasyfikatora nie zostały wykorzystane do jego oceny.

Wartość błędu *err*k(L) wyliczono w następujący sposób:

$$(\forall k = 1,2)\; err_{\mathrm{k}}(L) = \left[\frac{\dfrac{m_{\mathrm{k}}^+(L)}{m^+ - m_{\mathrm{k}}^+} + \dfrac{m_{\mathrm{k}}^-(L)}{m^- - m_{\mathrm{k}}^-}}{2} \right] 100\% \tag{7.1}$$

gdzie $m_{\mathrm{k}}^+(L)$ oznacza liczbę błędnie sklasyfikowanych wektorów cech z podzbioru danych klasy pierwszej ω_1, które nie należą do zbioru uczącego, $m_{\mathrm{k}}^-(L)$ zaś liczbę błędnie sklasyfikowanych wektorów cech z podzbioru danych klasy drugiej ω_2, które nie należą do zbioru uczącego.

Na wykresie 1 pokazano związek pomiędzy malejącym poziomem błędu $err_{\mathrm{k}}(L)$ (7.1) a wzrostem liczby detektorów L. Na podstawie przeprowadzonych eksperymentów można wnioskować, że zwiększenie do pewnego stopnia liczby detektorów może zwiększyć zdolność reguły klasyfikacji.



WYKRES 1. Wyniki eksperymentów dla różnych zbiorów uczących oraz różnych wartości marginesu ε

ŹRÓDŁO: oprac. własne.

8. Podsumowanie

W rozdziale przedstawiono innowacyjny algorytm klasyfikacji oparty na wielu warstwach detektorów. Metoda ta pozwala na efektywną klasyfikację danych o wysokim wymiarze, co jest ważne w dzisiejszym świecie, w którym dostęp do nich jest coraz powszechniejszy. Eksperymenty potwierdziły skuteczność tej metody, co sugeruje, że może ona znaleźć zastosowanie w wielu dziedzinach, w których klasyfikacja danych jest kluczowym zadaniem.

Bibliografia

[1] Bobrowski L., *Small Samples of Multidimensional Feature Vectors*, w: *Advances in Computational Collective Intelligence*, red. M. Hernes, K. Wojtkiewicz, E. Szczerbicki, Springer, Cham 2020, s. 87–98.

[2] Johnson R.A., Wichern D.W., *Applied Multivariate Statistical Analysis*, Prentice-Hall, Inc., Englewood Cliffs, New York 2002.

[3] Duda O.R., Hart P.E., Stork D.G., *Pattern classification*, John Wiley, New York 2001.

[4] Bobrowski L., *Eksploracja danych oparta na wypukłych i odcinkowo-liniowych funkcjach kryterialnych*, Wydawnictwo Politechniki Białostockiej, Białystok 2005.

[5] Bobrowski L., *Discovering main vertexical planes in a multivariate data space by using CPL functions*, w: *Advances in Data Mining. Applications and Theoretical Aspects – 14th Industrial Conference, St. Petersburg, Russia, July 16–20, 2014. Proceedings*, red. P. Perner, Springer Verlag, Berlin 2014, s. 200–213.

[6] Bobrowski L., Zabielski P., *Feature* (*Gene*) *Clustering with Collinearity Models*, w: *Computational Collective Intelligence. 13th International Conference, Rhodes, Greece, September 29 – October 1, 2021, Proceedings*, red. N. Thanh Nguyen, L. Iliadis, I. Maglogiannis, B. Trawiński, Springer, Cham 2021, s. 403–415.

[7] Bobrowski L., Łukaszuk T., *Relaxed Linear Separability* (*RLS*) *Approach to Feature* (*Gene*) *Subset Selection*, w: *Selected Works in Bioinformatics*, red. X. Xuhua, INTECH, Rijeka 2011, s. 103–118.

Classification based on Multiple Detector Layers

Summary: The chapter discusses an advanced classification method based on multi-layer detectors. The primary goal of this technique is to construct multiple layers of detectors used for the ultimate classification of objects. This method is applied for analyzing datasets where each object is described using multi-dimensional feature vectors. The process of constructing detector layers begins with independently building detectors for each class in the dataset. All created detector layers are utilized in the final classification of objects through voting. This advanced classification method can find application in fields where there is a need for effective classification models on small but high-dimensional datasets. It's worth noting that this technique can be useful in various domains, including medicine and economic data analysis.

Keywords: classification, multi-layer detectors, multi-dimensional feature vectors, criterion function, data analysis

Rozdział III
Systemy rekomendacyjne

Jerzy Krawczuk
Wydział Informatyki, Politechnika Białostocka

Streszczenie: W erze cyfrowej, w której każda interakcja online – od kliknięć po przewijanie strony – jest rejestrowana, systemy rekomendacyjne stały się kluczowym narzędziem w zestawie nowoczesnych technologii informacyjnych. Te zaawansowane systemy wyróżniają się zdolnością do personalizowania treści, produktów i usług, dostosowując je do indywidualnych potrzeb użytkowników. W tekście przedstawiono metody ich implementacji, ukazano ich złożoność oraz subtelność działania. W rozdziale omówione zostały kluczowe koncepcje stojące za systemami rekomendacyjnymi, w tym dwa podstawowe podejścia: filtrowanie oparte na treści oraz filtrowanie kolaboratywne. To pierwsze analizuje cechy produktów i dopasowuje je do preferencji użytkownika, drugie zaś wykorzystuje dane o interakcjach użytkowników do przewidywania ich preferencji. Opisano zalety i ograniczenia tych metod oraz podejścia hybrydowe, które mogą być remedium na bolączki pojedynczych metod. W rozdziale posłużono się przykładami specjalistycznych metryk stosowanych w ocenie jakości rekomendacji, takich jak *Mean Reciprocal Rank* oraz *Normalized Discounted Cumulative Gain*.

Słowa kluczowe: systemy rekomendacyjne, filtrowanie treści, filtrowanie kolaboratywne, *cold start*, *Mean Reciprocal Rank*, *Normalized Discounted Cumulative Gain*

1. Wprowadzenie

Systemy rekomendacyjne odgrywają niezwykle istotną rolę w dzisiejszym cyfrowym świecie [1]. Są one nieodłącznym elementem platform streamingowych, serwisów e-commerce, a także wielu innych usług internetowych, w których personalizacja i dostosowanie treści do indywidualnych preferencji użytkowników stanowią klucz do sukcesu. Systemy te funkcjonują na styku zaawansowanych technologii, takich jak uczenie maszynowe, przetwarzanie języka naturalnego oraz analiza dużych zbiorów danych, a ich zdolność do przewidywania zainteresowań i preferencji może znacząco wpłynąć na doświadczenie użytkownika i wyniki biznesowe.

W dobie, gdy każda akcja online, od kliknięcia po przewijanie, jest monitorowana i analizowana, systemy rekomendacyjne wyrosły na kluczowe narzędzie nowoczesnej technologii informacyjnej. Ich zdolność do personalizacji treści, produktów, a nawet usług, by spełniać unikatowe potrzeby każdego użytkownika, jest tak samo imponująca,

jak skomplikowana. Niniejszy artykuł ma wprowadzić czytelnika w tę niezwykle bogatą tematykę, która dużo rzadziej pojawia się w literaturze, niż wynikałoby to z olbrzymiej ilości praktycznych zastosowań, z którymi spotykamy się każdego dnia.

Zacznijmy od zrozumienia podstaw. System rekomendacyjny to rodzaj inteligentnego systemu informatycznego sugerującego produkty, usługi lub informacje (w treści rozdziału często będziemy używać terminu „element"), które mogą być interesujące dla użytkownika. Te sugestie, czyli rekomendacje, generowane są zawsze w postaci listy posortowanej (rankingu) od rekomendacji, którą system z jakiegoś powodu uznał za najbardziej trafną, do tych mniej dopasowanych. Generowane są one na podstawie różnorodnych danych, takich jak historia zakupów, zachowanie przeglądania, oceny innych użytkowników, a nawet porównania z podobnymi profilami użytkowników. Działają one pod powierzchnią wielu platform, takich jak YouTube, Facebook, Netflix, Allegro, często są niewidoczne dla oka, lecz niezmiernie istotne dla naszego cyfrowego doświadczenia.

W tym opracowaniu zgłębimy metodyki stojące za budową systemów rekomendacyjnych. Przeanalizujemy tradycyjne podejścia, takie jak filtrowanie kolaboratywne i bazujące na treści, a także bardziej zaawansowane techniki hybrydowe łączące różne metody. Omówimy również wyzwania, m.in. trudności związane z rekomendowaniem produktów nowym użytkownikom lub rekomendowaniem nowych produktów (ang. *cold start problem*) [16].

2. Algorytmy filtrowania treści

Algorytmy bazujące na filtrowaniu treści (ang. *content-based filtering*) to jedna z dwóch klasycznych metod stosowanych w systemach rekomendacyjnych [23, 17, 24]. Opiera się ona na analizie i porównaniu cech/atrybutów elementów z cechami użytkowników w celu wygenerowania rekomendacji. W tym rozdziale przyjrzymy się bliżej, jak działają te algorytmy, jakie mają zalety i ograniczenia oraz jakie wyzwania niesie ich praktyczna implementacja.

2.1. Zasada działania

Podstawowym założeniem filtrowania bazującego na treści jest przekonanie, że użytkownik będzie zainteresowany produktami odpowiadającymi jego profilowi stworzonemu w systemie. Możemy wyróżnić cztery kluczowe kroki niezbędne do wdrożenia takiego systemu.

Ekstrakcja cech. Najpierw system musi zidentyfikować kluczowe atrybuty (cechy) produktów lub elementów, które są przedmiotem rekomendacji. Mogą to być gatunek filmu, autor książki, składniki potrawy czy specyfikacje techniczne produktu. W prostym przykładzie systemu rekomendującego filmy w tabeli 1 przedstawiono trzy cechy, które w skali od 0 do 1 określają, jak silnie film przynależy do danego gatunku.

TABELA 1. Cechy rekomendowanych produktów

	Komedia	Przygoda	Dramat
Film 1	0,6	0,9	0,0
Film 2	0,9	0,1	0,0
Film 3	0,1	0,7	0,9

ŹRÓDŁO: opracowanie własne.

Profilowanie użytkownika [4]. Następnie system tworzy profil użytkownika. Niektóre implementacje pozwalają bezpośrednio określić preferencje, ale większość robi to na podstawie interakcji użytkowników z różnymi elementami serwisu. Interakcje te mogą obejmować oceny, przeglądanie lub zakup produktów. Profil użytkownika zawiera wagi cech, które odpowiadają jego zainteresowaniom. W poniższym przykładzie (tabela 2) użytkownik A mógł bezpośrednio w ankiecie zaznaczyć, że jest zainteresowany gatunkami komedia i przygoda, lub mogliśmy to pośrednio wywnioskować z faktu, iż w serwisie ogląda tylko filmy z tych gatunków.

TABELA 2. Profile użytkowników

	Komedia	Przygoda	Dramat
Użytkownik A	0,9	0,9	0,0
Użytkownik B	0,2	0,5	0,9

ŹRÓDŁO: opracowanie własne.

Dopasowanie. Następnie algorytm oblicza podobieństwo między profilem użytkownika a cechami każdego z produktów [21]. Najczęściej wykorzystuje się do tego takie metody miary, jak odległość kosinusowa, współczynnik podobieństwa Jaccarda czy korelacja Pearsona.

Ranking. Na podstawie podobieństwa do profilu każdemu użytkownikowi tworzy się indywidualną posortowaną listę produktów (ranking). Na jej szczycie znajdują się elementy najbardziej podobne do profilu czy, mówiąc inaczej, najbliższe użytkownikowi zgodnie z przyjętą metryką. W naszym prostym przykładzie możemy zaobserwować, że ranking użytkownika A to filmy 1, 2, 3, a użytkownika B – filmy 3, 1, 2.

2.2. Zalety i ograniczenia

Zalety filtrowania bazującego na treści:

- Ponieważ rekomendacje są ściśle związane z indywidualnym profilem użytkownika, są one wysoce spersonalizowane.

- W przeciwieństwie do filtrowania kolaboratywnego systemy te nie wymagają danych od innych użytkowników, co czyni je efektywnymi nawet przy mniejszej liczbie użytkowników. Do rekomendacji wystarczą profil jednego użytkownika i cechy produktów.
- W niektórych zastosowaniach bardzo istotne jest podanie powodu rekomendacji. Systemy te mogą łatwo wyjaśnić użytkownikowi, dlaczego dany element został zarekomendowany, ponieważ rekomendacja jest oparta na konkretnych cechach produktów.

Wyzwania i **ograniczenia**:

- System może rekomendować elementy zbyt podobne do tych, które użytkownik już zna, co ogranicza odkrywanie nowych, nieznanych kategorii.
- Skuteczność systemu zależy od dostępności i jakości opisów elementów, co może być trudne do uzyskania dla niektórych typów produktów lub treści [2]. Ten rodzaj systemu rekomendacyjnego sprawdza się dobrze, gdy mamy pełną kontrolę i standaryzację informacji o elementach naszego serwisu.
- W przypadku nowych użytkowników może wystąpić *cold start problem*. Aby system mógł przygotować rekomendacje, potrzebny jest profil użytkownika. Gdy jest on budowany na podstawie zachowań użytkownika, w serwisie potrzeba czasu, który może być kluczowy na początku przygody użytkownika z serwisem. Często stosowanym rozwiązaniem jest pytanie użytkownika wprost o jego preferencje przy rejestracji lub o wskazanie kilku interesujących go elementów serwisu.

2.3. Wyzwania implementacyjne

Zaimplementowanie takiego systemu w praktyce zaczyna się od zebrania i oczyszczenia danych, co jest kluczowe dla późniejszych etapów analizy. Wymaga to stosowania technik przetwarzania języka naturalnego (NLP) dla tekstów, analizy obrazu dla multimediów oraz innych specjalistycznych metod dla różnych typów danych. Po utworzeniu solidnej bazy danych następuje projektowanie i trenowanie modelu, który będzie w stanie przewidywać zainteresowania użytkownika na podstawie jego dotychczasowej aktywności. W tym kontekście inżynierowie ds. danych i specjaliści ds. uczenia maszynowego stają przed wyzwaniem stworzenia systemów, które będą nie tylko efektywne i skalowalne, lecz także zdolne do adaptacji i ewolucji w czasie.

Podsumowując, algorytmy bazujące na filtrowaniu treści są podstawowym, ale bardzo silnym narzędziem w arsenale systemów rekomendacyjnych, oferującym personalizację i precyzję. Ich skuteczność w dużym stopniu zależy od głębokiego zrozumienia produktów i użytkowników, jak też od ciągłego dostosowywania do zmieniającego się środowiska.

3. Algorytmy filtrowania kolaboratywnego

Filtrowanie kolaboratywne jest jednym z najbardziej rozpowszechnionych i efektywnych podejść do budowania systemów rekomendacyjnych [22, 15]. W przeciwieństwie do filtrowania opartego na treści, które analizuje podobieństwa między cechami produktów a zainteresowaniami użytkownika, filtrowanie kolaboratywne polega na wykorzystaniu ocen produktów przez wielu użytkowników, aby przewidzieć, co może się spodobać wybranemu użytkownikowi. Jest to metoda, która wykorzystuje „mądrość tłumu" do przewidywania preferencji użytkowników. Nie używamy w niej cech produktów i profilu użytkownika, ale jedynie tzw. macierz ocen (tabela 3).

TABELA 3. Macierz ocen stosowana w filtrowaniu kolaboratywnym

	Film 1	Film 2	Film 3	Film 4
Użytkownik A	5	4	?	5
Użytkownik B	3	?	?	4
Użytkownik C	?	2	5	?
Użytkownik D	1	3	4	?

ŹRÓDŁO: opracowanie własne.

W takiej macierzy wiersze zazwyczaj odpowiadają poszczególnym użytkownikom, a kolumny poszczególnym elementom (takim jak filmy, książki, produkty itd.). Wartości w komórkach macierzy to oceny danego elementu przez użytkownika. Mogą one być wystawiane przez niego bezpośrednio, np. przez umożliwienie mu wyboru liczby gwiazdek w skali od 1 do 5 albo przycisków [lubię], [nie lubię]. Użytkownik może też oceniać dany element pośrednio, w momencie jego interakcji z produktem w portalu, kiedy pod uwagę bierze się np. czas spędzony na oglądaniu produktu, zadanie pytania czy wystawienie komentarza, dodanie produktu do ulubionych bądź koszyka zakupów i finalnie jego zakup.

Metoda polega na wypełnieniu macierzy w miejscach pustych (?) najbardziej prawdopodobnymi ocenami danego elementu przez użytkownika. Istnieje wiele algorytmów, które realizują to zadanie. Muszą one mierzyć się z wyzwaniem bardzo rzadkiej macierzy, gdyż dla większości rzeczywistych systemów rekomendacyjnych tylko niewielka część komórek jest wypełniona.

3.1. Zasada działania

W najbardziej ogólnej klasyfikacji algorytmy filtrowania kolaboratywnego możemy podzielić na te oparte na pamięci (ang. *memory-based*) i na modelach (ang. *model-based*), w których stosuje się techniki uczenia maszynowego.

Podejście oparte na **pamięci**:

- Filtrowanie oparte na użytkownikach (ang. *user-based*): system szuka użytkowników o podobnych preferencjach do danego użytkownika i na tej podstawie rekomenduje produkty lub treści, które się im podobały. W naszym przykładzie użytkownik D jest najbardziej podobny do użytkownika C, gdyż podobnie ocenił filmy 2 i 3. Bardzo słabo natomiast ocenił film 1, stąd system może wnioskować, że również nie spodoba się on użytkownikowi C.
- Filtrowanie oparte na przedmiotach (ang. *item-based*): system analizuje podobieństwa między różnymi produktami lub treściami na podstawie ocen użytkowników. Jeśli użytkownik ocenił wysoko pewien produkt, system może polecić mu inne produkty również ocenione wysoko przez użytkowników, którzy wysoko ocenili też ten sam produkt. Film 4 jest wysoko oceniony przez użytkowników A i B, dodatkowo użytkownik A wysoko ocenił film 2, co może być dobrą rekomendacją dla użytkownika B.

Podejście oparte na **modelach**:

- Faktoryzacja macierzy (ang. *matrix factorization*) jest jednym z najbardziej popularnych podejść i polega na dekompozycji macierzy ocen użytkownik–element w produkty dwóch niższych wymiarów: macierzy reprezentujących ukryte czynniki (ang. *latent factors*) dla użytkowników i elementów [14]. Jest to najbogatsza w algorytmy gałąź systemów rekomendacyjnych. Oprócz standardowych technik jak SVD (*Singular Value Decomposition*) oraz ALS (*Alternating Least Squares*) stosowane są również takie algorytmy jak *Incremental Factorization Machines* [12] czy *Randomized Matrix Decomposition* [18].
- Sztuczne sieci neuronowe: głębokie uczenie znalazło zastosowanie również w systemach rekomendacyjnych, w których mogą one odkrywać skomplikowane struktury i relacje w danych [10]. Sieci neuronowe są wykorzystywane do tworzenia tzw. osadzeń (ang. *embeddings*) dla użytkowników i elementów, które następnie służą do generowania rekomendacji.
- Algorytmy oparte na sąsiedztwie (ang. *neighbourhood models*): w modelach opartych na sąsiedztwie, takich jak probabilistyczne modele indeksowania semantycznego (*Probabilistic Latent Semantic Indexing*, PLSI) [11] czy *Latent Dirichlet Allocation* (LDA) [3], stosowane są techniki z teorii prawdopodobieństwa do grupowania użytkowników i elementów, co umożliwia generowanie rekomendacji na podstawie przynależności do tych grup. Metody te stosowane są szczególnie w sytuacjach, gdy mamy do czynienia z bogatymi danymi tekstowymi, takimi jak opisy produktów, recenzje czy artykuły pisane przez użytkowników.

3.2. Zalety i ograniczenia

Zalety filtrowania kolaboratywnego:

- Nie wymaga wiedzy o treści. Algorytmy te mogą efektywnie rekomendować elementy serwisu bez konieczności analizy treści produktów i określania ich cech, co jest szczególnie przydatne w przypadkach, gdy taka analiza jest trudna lub niemożliwa.
- Zdolność do odkrywania nowości. Systemy te są zdolne do rekomendowania produktów, których użytkownik nigdy by samodzielnie nie znalazł, ponieważ odkrycia nie są ograniczone do wybranego zestawu atrybutów. Jeżeli podobni mu użytkownicy byli zainteresowani innymi elementami, to pojawią się one w jego rekomendacjach i pozwolą mu odkryć nowe obszary, z którymi dotychczas nie miał styczności.

Wyzwania i **ograniczenia**:

- Problem rozruchu na zimno (ang. *cold start*) [19]. W przypadku tych algorytmów dotyczy on zarówno nowych użytkowników, jak i nowych produktów. Jedni i drugie potrzebują wielu ocen, aby zostały obsłużone przez tę metodę.
- Macierze ocen są często bardzo rzadkie, co oznacza, że większość możliwych ocen nie jest dostępna i utrudnia to znalezienie podobnych użytkowników lub produktów.
- Mimo że współczesne metody są skalowalne, klasyczne algorytmy mogą być zbyt wolne dla bardzo dużych systemów z milionami użytkowników i produktów, gdzie rekomendacje powinny być generowane na żywo podczas wizyt użytkownika na konkretnych stronach serwisu.
- Algorytm ten zależy od jakości i ilości danych oceniania; niejednolite dane lub manipulacja ocenami mogą znacząco wpłynąć na jakość rekomendacji.
- Słaba możliwość uzasadnienia przez system rekomendacji, szczególnie metod opartych na uczeniu maszynowym.

3.3. Wyzwania implementacyjne

W praktyce realizacja filtrowania kolaboratywnego często polega na wykorzystaniu zaawansowanych technik uczenia maszynowego, takich jak algorytmy oparte na najbliższych sąsiadach (k-NN), algorytmy oparte na modelach (np. SVD, NMF) czy nawet głębokich sieciach neuronowych. Przetwarzanie tej skali danych wymaga użycia zaawansowanych technologii oraz efektywnych algorytmów, które niemal zawsze muszą działać w rozproszonych systemach obliczeniowych.

Podsumowując, filtrowanie kolaboratywne jest potężną techniką w arsenale systemów rekomendacyjnych, która potrafi dostarczyć trafne rekomendacje dzięki analizie wzorców zachowań ogromnej liczby użytkowników. Wymaga jednak starannej implementacji oraz ciągłej optymalizacji, aby sprostać wyzwaniom, takim jak problem rozruchu na zimno i rzadkość danych.

4. Hybrydowe systemy rekomendacyjne

Hybrydowe systemy rekomendacyjne łączą w sobie różne techniki i algorytmy rekomendacyjne, aby wyeliminować ograniczenia poszczególnych metod i wzmocnić ich zalety [6]. Poniżej wymieniono i krótko scharakteryzowano pięć technik, co nie oznacza, że w praktyce musimy wybrać tylko jedną z nich – w przypadku dużych systemów często jest to kombinacja wielu takich podejść. Rozwiązania stosowane przez duże firmy, takie jak Facebook, Amazon czy YouTube, nie są upubliczniane[1], ale znamy rozwiązanie, które wygrało klasyczny już konkurs na system rekomendacyjny Netflix Prize 2006 [5]. Wyzwaniem tego konkursu było stworzenie algorytmu rekomendującego na podstawie udostępnionych 100 milionów recenzji filmów, który będzie lepszy o co najmniej 10% od tego stosowanego i opracowanego przez Netflix (o nazwie CineMatch). W 2009 roku nagrodę w wysokości miliona dolarów przyznano za rozwiązanie obejmujące aż 107 różnych algorytmów i łączące ich rekomendacje w zależności od okoliczności.

4.1. Łączenie wyników

Łączenie wyników to technika hybrydowa, która polega na wykorzystaniu kilku różnych systemów rekomendacyjnych i scaleniu ich wyjść, aby uzyskać końcowe zestawienie rekomendowanych pozycji [20]. Każdy z tych systemów pracuje niezależnie, generując własne propozycje. Są one następnie ważone i łączone na podstawie ustalonych kryteriów, takich jak precyzja, zaufanie do systemu czy świeżość rekomendacji.

W technice tej musimy określić sposoby ważenia rekomendacji dla poszczególnych systemów, a następnie sumowania wyników. Najprostszym sposobem ważenia jest określenie z góry stałego współczynnika wagowego każdemu z systemów, np. równego 1. Możemy też zaimplementować wagi dynamiczne, które mogą zależeć od różnych czynników, takich jak aktualność danych, zachowania użytkownika czy kontekst sytuacyjny (np. pora dnia). Wagi mogą być też różne dla poszczególnych użytkowników i związane z ich preferencjami.

Gdy mamy ustalone wagi, następuje proces agregacji wyników. Popularnym rozwiązaniem jest głosowanie większościowe, w którym pozycja musi zostać zarekomendowana przez określoną liczbę systemów, aby pojawić się w końcowym zestawieniu. Możemy też wyróżnić technikę polegającą na budowaniu rankingu, który uwzględnia ważenie na podstawie różnych cech, takich jak popularność czy nowość elementów.

Załóżmy, że mamy do czynienia z serwisem streamingowym, który chce zarekomendować filmy swoim użytkownikom. Serwis wykorzystuje trzy różne systemy rekomendacyjne:

[1] Wyjątkiem jest serwis Twitter, który po przejęciu przez Elona Muska w kwietniu 2022 roku upublicznił swój algorytm rekomendacyjny (https://github.com/twitter/the-algorithm, dostęp: 7.11.2024).

- filtrowanie kolaboratywne bazujące na ocenach innych użytkowników o podobnych gustach, które rekomenduje filmy A, B i C;
- filtrowanie oparte na treści, które analizuje cechy filmów, takie jak gatunek czy reżyser, a które zarekomendowało filmy B, C i D;
- system oparty na regułach, który uwzględnia aktualne trendy i promocje specjalne, i w tym akurat tygodniu są to filmy E i F.

Użytkownik jest nowy i nie ma jeszcze wielu ocen w systemie, więc system kolaboratywny otrzymuje niższą wagę. Filmy B i C są wspólne dla dwóch pierwszych systemów, dlatego otrzymują wyższe ważenie. Film E jest promowany i mimo że według dwóch pierwszych systemów nie jest zgodny z preferencjami użytkownika, otrzymuje dodatkowe punkty ze względu na strategię marketingową. Końcowa lista rekomendacji może wyglądać następująco: B, C, E, A, D, F.

4.2. Dodawanie cech

Systemy te integrują cechy (charakterystyki) produktów lub użytkowników z jednego źródła (np. z filtrowania opartego na treści) bezpośrednio do modelu innego systemu (np. filtrowania kolaboratywnego) [13]. Przykładowo cechy użytkowników, takie jak wiek, płeć, historia zakupów, czy atrybuty produktów, takie jak gatunek filmu czy reżyser, z systemu filtrowania opartego na treści mogą być wykorzystane w modelu filtrowania kolaboratywnego. Dzięki temu połączeniu możliwe jest uwzględnienie dodatkowych informacji o przedmiotach i/lub użytkownikach, co może wzbogacić rekomendacje i uczynić je bardziej trafnymi. Taki system rekomendacyjny wykorzystuje zarówno oceny filmów, jak i ich cechy, aby zarekomendować użytkownikowi filmy. Jeśli np. często ocenia on pozytywnie filmy z gatunku „akcja" i z udziałem konkretnego aktora, system może polecić mu nowy film z tego gatunku z tym aktorem, nawet jeśli nie został on jeszcze oceniony przez wielu użytkowników. W rezultacie użytkownik otrzymuje rekomendacje filmów, które są dopasowane nie tylko do jego ogólnego profilu w serwisie, lecz także do specyficznych cech filmów, które w przeszłości przykuły jego uwagę.

Tego typu systemy hybrydowe wymagają zmian w samym algorytmie i procesie jego budowy. Przykładem może być technika SVD++, która rozszerza klasyczną dekompozycję macierzy SVD o dodatkową macierz cech.

4.3. Podejście łańcuchowe

W tej metodzie wynik jednego systemu rekomendacyjnego jest używany jako wejście dla innego systemu [25]. Na przykład system filtrowania kolaboratywnego może być używany do zawężenia puli potencjalnych rekomendacji, a następnie filtrowanie oparte na treści może wybrać najbardziej odpowiednie produkty z tej ograniczonej listy.

W podejściu łańcuchowym proces rekomendacji odbywa się w krokach. Przykładowo pierwszy model może dokonywać selekcji ogólnej grupy przedmiotów, w przykładzie rekomendacji filmów może to być filtrowanie oparte na treści, które wybierze tytuły zgodne z ogólnymi preferencjami użytkownika, takie jak gatunek filmu czy reżyser. Następnie wykorzystujemy model filtracji kolaboratywnej, który analizuje wybory podobnych użytkowników, aby zawęzić listę tytułów do tych, które są popularne wśród osób o podobnych gustach. Ostatni model może być systemem uczenia maszynowego, który bierze pod uwagę subtelniejsze czynniki, takie jak nastroje wyrażane przez użytkownika w opisowych ocenach czy pory dnia, kiedy zwykle ogląda on filmy, i oferuje rekomendacje dostosowane do konkretnego kontekstu.

Z każdym krokiem rekomendacje są coraz bardziej „oczyszczane" lub dopracowane. Model na końcu łańcucha może wykorzystywać subtelne i zaawansowane techniki, aby dostarczyć bardzo spersonalizowanych sugestii.

4.4. Metamodel

Metamodel to podejście, w którym jeden system rekomendacyjny jest wykorzystywany do stworzenia cech, te zaś następnie są używane jako wejścia do drugiego systemu rekomendacyjnego [8]. Zazwyczaj budowanych jest wiele modeli, których wyjścia stanowią wejście dla metamodelu.

Przykładowo mamy kilka różnych systemów rekomendacyjnych: system oparty na filtrowaniu kolaboratywnym, na treści oraz wykorzystujący analizę sentymentu w recenzjach. Zamiast tworzenia prostego systemu hybrydowego przez połączenie wyników z tych modeli tworzymy metamodel. Przyjmuje on wyniki z podstawowych systemów rekomendacyjnych jako wejście i jest trenowany w celu zrozumienia, które kombinacje rekomendacji z podstawowych modeli najlepiej przewidują zakupy klientów. Model może odkryć, że rekomendacje z filtrowania kolaboratywnego są bardziej skuteczne dla nowych produktów, podczas gdy analiza sentymentu jest lepsza dla produktów z wieloma recenzjami.

4.5. Przełączanie modeli

System przełączający wybiera między różnymi strategiami rekomendacji w zależności od określonych warunków lub kontekstów [9]. Na przykład może stosować filtrowanie kolaboratywne, gdy dostępnych jest wiele danych o użytkowniku, a przełączać się na filtrowanie oparte na treści lub metody oparte na wiedzy, gdy danych jest mniej (np. w przypadku nowych użytkowników). Innym powodem przełączania może być aktualny kontekst użytkownika, taki jak pora dnia, lokalizacja czy obecny etap w tzw. *customer journey* – wówczas różne modele mogą dawać lepsze wyniki. Innym powodem może być wydajność poszczególnych systemów. Niektóre modele

mogą być bardziej kosztowne obliczeniowo od innych. Ich przełączanie pozwala na optymalizację wydajności poprzez stosowanie bardziej złożonych modeli tylko wtedy, gdy jest to uzasadnione.

5. Miary jakości

Do pomiaru jakości rekomendacji szeroko stosowane są tzw. metryki *top-k*, które oceniają jakość *k* pierwszych rekomendacji w rankingu zwracanym przez system rekomendacyjny [7]. Z reguły też same systemy nie zwracają użytkownikowi listy z wszystkimi elementami serwisu, a jest ona ograniczona do 5, 10 czy 20 elementów.

5.1. *Precision at k*

Metryka *precision at k* (precyzja przy *k*) w kontekście systemów rekomendacyjnych to sposób mierzenia ich skuteczności w zakresie dostarczania trafnych rekomendacji. Jest to miara koncentrująca się na pierwszych *k* pozycjach listy rekomendacji, pozwalająca ocenić, jak dużo z tych *k* rekomendowanych elementów jest rzeczywiście relewantnych dla użytkownika. Matematycznie jest to stosunek liczby relewantnych wśród *k* pierwszych rekomendacji do liczby *k*. Na przykład jeśli system rekomendacyjny generuje listę filmów, a użytkownik uznaje za interesujące (relewantne) 2 z pierwszych 5 filmów, to *precision at 5* wynosi 2/5. Pewnym ograniczeniem tej miary jest to, że nie uwzględnia ona pozycji, na której znalazły się te dwie relewantne rekomendacje, a intuicyjnie wiemy, że lepszym rankingiem jest ten, w którym te dwa elementy znalazły się na 1 i 2 pozycji, niż ten, w którym są one na pozycjach 4 i 5. Kolejne dwie miary biorą pod uwagę również pozycję na liście rekomendacji.

5.2. *Mean Reciprocal Rank*

Jest to metryka służąca do oceny jakości systemów rekomendacyjnych oraz systemów wyszukiwania informacji. Jest szczególnie przydatna w scenariuszach, w których ważna jest pierwsza trafna odpowiedź lub rekomendacja. *Mean Reciprocal Rank* jest odwrotnością rangi pierwszego odpowiedniego (relewantnego) wyniku na liście rekomendacji. Jeśli pierwszy odpowiedni element znajduje się na pozycji *k*, to odwrotna ranga wynosi *1/k*. Przykładowo, jeśli pierwszy odpowiedni element jest na pierwszym miejscu, odwrotna ranga wynosi 1. Jeśli pierwszy odpowiedni element jest na drugim miejscu, odwrotna ranga wynosi 1/2. Jeśli jest na trzecim miejscu, to odwrotna ranga wynosi 1/3 itd.

Mean Reciprocal Rank oblicza się jako średnią odwrotnych rang pierwszych odpowiednich rekomendacji dla wszystkich użytkowników w zbiorze.

5.3. *Normalized Discounted Cumulative Gain*

To zaawansowana metryka stosowana do oceny jakości systemów rekomendacyjnych, biorąca pod uwagę wszystkie rekomendacje i ich pozycje w rankingu. Im dalej na liście znajduje się element, tym mniejsza waga jest mu nadawana. Najczęściej stosowana jest waga logarytmiczna:

$$DCG_k = \sum_{i=1}^{k} relevance_i / \log_2(i+1)$$

gdzie $relevance_i$ to stopień dopasowania rekomendacji (relewantność); w praktyce często przyjmuje się jedynie wartości 0 i 1.

Metryka ta nie przyjmuje wartości z przedziału [0,1], dlatego też jest normalizowana przez maksymalną możliwą wartość NDCG. Wartość ta oznaczana jako iDCG obliczana jest jako NDCG dla idealnie posortowanego rankingu według relewancji.

5.4. Przykład

Poniżej przedstawiono przykład pięciu rekomendacji proponowanych użytkownikowi. Rekomendacje B na pozycji drugiej i E na pozycji 5 są relewantne, pozostałe zaś nie są uznane za dobre dla tego użytkownika.

Pozycja w rankingu:	1	2	3	4	5
Element:	a	B	c	d	E
Precision at 5: 0,40 = 2/5					
Reciprocal rank: 0,50 = 1/2					
NDCG : 0,62 = (1/log(1 + 2) + 1/log(1 + 5)) / (1/log(1 + 1) + 1/log(1 + 2))					

6. Podsumowanie

W niniejszym rozdziale przedstawiono przegląd metodologii i technik stosowanych w budowie zaawansowanych systemów rekomendacyjnych. Przeanalizowano kluczowe koncepcje i algorytmy filtrowania zarówno opartego na treści, jak i kolaboratywnego, podkreślając ich mocne strony oraz wyzwania, które niosą ze sobą w praktycznych implementacjach.

Filtrowanie oparte na treści wykazuje silne podstawy w kontekście rekomendacji personalizowanych, jednak to filtrowanie kolaboratywne jest techniką bardziej dynamiczną, która zdolna jest do odkrywania nieoczywistych wzorców preferencji użytkowników. Poprzez analizę zachowań i ocen dużej grupy użytkowników systemy te potrafią generować rekomendacje o wysokiej trafności, co jest szczególnie widoczne w platformach e-commerce, serwisach streamingowych czy w mediach społecznościowych.

Problemy takie jak *cold start* oraz rzadkość danych stanowią istotne wyzwania, którym projektanci systemów muszą stawić czoła. Dlatego też w praktyce często tworzone są systemy hybrydowe, które łączą zalety poszczególnych modeli i mogą pomóc przeciwdziałać ich ograniczeniom. W tym artykule omówiono kilka przykładowych podejść, jak łączenie wyników czy dodawanie cech.

Artykuł stanowi przewodnik po współczesnych metodach rekomendacji i służy za punkt odniesienia dla praktyków i badaczy zainteresowanych głębszym zrozumieniem tego szybko rozwijającego się obszaru informatyki.

*

Badania zostały zrealizowane w ramach pracy nr WZ/WI-IIT/4/2023 w Politechnice Białostockiej, sfinansowanej z subwencji badawczej przekazanej przez ministra edukacji i nauki.

Bibliografia

[1] Aggarwal Ch.C., *Recommender systems. The textbook*, vol. 1, Springer, Cham 2016.

[2] Ansari A., Mohammed M.H., *Content based video retrieval systems methods, techniques, trends and challenges*, „International Journal of Computer Applications" 2015, vol. 112(7), s. 13–22.

[3] Apaza R.G., Vera Cervantes E., Cruz Quispe L., Ochoa Luna J., *Online courses recommendation based on LDA*, w: *Proceedings of the 1st Symposium on Information Management and Big Data*, red. J.A. Lossio-Ventura, H. Alatrista-Salas, CEUR Workshop, 2014, s. 42–48.

[4] Au Yeung Ch.M., Gibbins N., Shadbolt N., *A study of user profile generation from folksonomies*, w: *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*, 2008.

[5] Bell R.M., Koren Y., Volinsky Ch., *All together now: A perspective on the Netflix Prize*, „Chance" 2010, vol. 23(1), s. 24–29.

[6] Burke R., *Hybrid recommender systems: Survey and experiments*, „User Modeling and User-Adapted Interaction" 2002, vol. 12, s. 331–370.

[7] Cremonesi P., Koren Y., Turrin R., *Performance of recommender algorithms on top-n recommendation tasks*, w: *Proceedings of the fourth ACM conference on Recommender systems*, Association for Computing Machinery, New York 2010, s. 39–46.

[8] Cunha T., Soares C., de Carvalho A., *Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering*, „Information Sciences" 2018, vol. 423, s. 128–144.

[9] Ghazanfar M., Prugel-Bennett A., *An improved switching hybrid recommender system using naive bayes classifier and collaborative filtering*, The 2010 IAENG International Conference on Data Mining and Applications, 2010.

[10] He X., Liao L., Zhang H., Nie L., Hu X., Chua T.S., *Neural collaborative filtering*, w: *Proceedings of the 26th international conference on World Wide Web*, red. E. Barrett, ACM, New York 2017, s. 173–182.

[11] Hofmann T., *Probabilistic latent semantic indexing*, w: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, red. F. Gey, M. Hearst, ACM, New York 1999, s. 50–57.

[12] Jakomin M., Bosnić Z., Curk T., *Simultaneous incremental matrix factorization for streaming recommender systems*, „Expert Systems with Applications" 2020, vol. 160, 113685.

[13] Koren Y., *Factorization meets the neighbourhood: a multifaceted collaborative filtering model*, w: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, red. Y. Li, B. Liu. S. Sarawagi, ACM, New York 2008, s. 426–434.

[14] Koren Y., Bell R., Volinsky Ch., *Matrix factorization techniques for recommender systems*, „Computer" 2009, vol. 42(8), s. 30–37.

[15] Koren Y., Rendle S., Bell R., *Advances in collaborative filtering*, w: *Recommender systems handbook*, red. F. Ricci, L. Rokach, B. Shapira, Springer, Cham 2022, s. 91–142.

[16] Lika B., Kolomvatsos K., Hadjiefthymiades S., *Facing the cold start problem in recommender systems*, „Expert Systems with Applications" 2014, vol. 41(4), s. 2065–2073.

[17] Lops P., De Gemmis M., Semeraro G., *Content-based recommender systems: State of the art and trends*, w: *Recommender systems handbook*, red. F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, Springer, Cham 2011, s. 73–105.

[18] Martinsson P.G., Rokhlin V., Tygert M., *A randomized algorithm for the decomposition of matrices*, „Applied and Computational Harmonic Analysis" 2011, vol. 30(1), s. 47–68.

[19] Natarajan S., Vairavasundaram S., Natarajan S., Gandomi A.H., *Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data*, „Expert Systems with Applications" 2020, vol. 149, 113248.

[20] Nguyen T.N., Nguyen A.T., *Weighing the role of multi-criteria communities for recommender systems*, „International Journal of Intelligent Engineering Informatics" 2015, vol. 3(4), s. 330–348.

[21] Stitini O., Kaloun S., Bencharef O., *Investigating different similarity metrics used in various recommender systems types: Scenario cases*, „The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences" 2022, vol. 48, s. 187–193.

[22] Su X., Khoshgoftaar T.M., *A survey of collaborative filtering techniques*, „Advances in Artificial Intelligence" 2009, 421425.

[23] Van Meteren R., Van Someren M., *Using content-based filtering for recommendation*, „Machine Learning ECML" 2000, vol. 30, s. 47–56.

[24] Wang D., Liang Y., Xu D., Feng X., Guan R., *A content-based recommender system for computer science publications*, „Knowledge Based Systems" 2018, vol. 157, s. 1–9.

[25] Wang S., Hu L., Wang Y., Cao L., Sheng Q.Z., Orgun M., *Sequential recommender systems: challenges, progress and prospects*, arXiv preprint arXiv:2001.04830, 2019.

Recommender systems

Summary: In the digital age, where every online interaction – from clicks to page scrolling – is recorded, recommendation systems have become a defining feature of the information technology stack. These advanced content systems are designed to personalize products and services, tailoring them to the individual needs of users. Ten articles aim to present the methods of implementing such systems, their complexity and the subtlety of operations. The article discusses key concepts that are related to recommendation systems, including two basic elements: content filtering and collaborative filtering. Content-to-content filtering analyzes product marks and matches them to user usage, while collaborative filtering uses user interaction data that predicts their usage. Discussed are the advantages and limitations of these methods, as well as hybrid

distributions that may be a remedy for the problems that disable the method. Specialized metrics used in the quality of recommendations, such as Mean Reciprocal Rank and Normalized Discounted Cumulative Gain, are described and illustrated.

Key words: recommender systems, content-based filtering, collaborative filtering, cold start, Mean Reciprocal Rank, Normalized Discounted Cumulative Gain

Rozdział IV
Zastosowanie klasyfikatorów hierarchicznych w biometrii behawioralnej chodu

Daniel Grabowski, Aleksander Sawicki
Wydział Informatyki, Politechnika Białostocka

Streszczenie: W rozdziale opisano zastosowanie klasyfikatorów hierarchicznych w biometrii behawioralnej chodu, której celem jest identyfikacja osób na podstawie sposobu poruszania się. W badaniu użyto danych zebranych za pomocą systemu śledzenia ruchu Perception Neuron 32, który monitorował chód stu uczestników podczas dwóch sesji. Wstępne przetworzenie tych danych obejmowało segmentację cykli chodu, interpolację, konwersję do globalnego układu odniesienia, filtrację częstotliwościową i normalizację. Następnie zostały one przeanalizowane przy użyciu sieci neuronowej typu CNN. Klasyfikacja hierarchiczna polegała na podziale danych według takich kryteriów jak płeć, waga, wzrost i BMI oraz identyfikacji osób w tych podzbiorach. Podejście to poprawiło skuteczność systemu, szczególnie przy podziale według płci – osiągnięto F1-score na poziomie 0,941. Podział danych według innych kryteriów nie przyniósł znaczącej poprawy. Wykazano, że klasyfikatory hierarchiczne mogą istotnie poprawić efektywność systemów biometrycznych poprzez podział danych na mniejsze, jednorodne podzbiory i zastosowanie specjalizowanych klasyfikatorów. Badania przeprowadzono w ramach grantu Politechniki Białostockiej i dzięki wsparciu finansowemu Ministerstwa Nauki i Szkolnictwa Wyższego w Polsce.

Słowa kluczowe: biometria behawioralna, chód, klasyfikatory hierarchiczne, identyfikacja osób, sieć neuronowa

1. Wprowadzenie

Biometria behawioralna chodu jest szybko rozwijającą się dziedziną nauki [1], której celem jest identyfikacja osób na podstawie ich sposobu poruszania się. W obszarze rozpoznawania wzorców chodu zastosowanie klasyfikatorów hierarchicznych może zwiększyć skuteczność systemu poprzez efektywniejsze zarządzanie złożonymi danymi wejściowymi oraz poprawę dokładności klasyfikacji.

W ramach niniejszej pracy zaproponowaliśmy zastosowanie klasyfikatorów hierarchicznych do identyfikacji osób na podstawie próbek ich chodu. Zebrane dane zostały przeanalizowane w kontrolowanych warunkach laboratoryjnych za pomocą systemu śledzenia ruchu Perception Neuron 32 [2]. Składa się on z zestawu 17 sensorów,

które umieszcza się na specjalnym kombinezonie. W skład każdego z sensorów IMU (ang. *Inertial Measurement Unit*) wchodziły trójosiowy akcelerometr oraz trójosiowy żyroskop.

Dane zebrano od stu uczestników podczas dwóch sesji, co umożliwiło przeprowadzenie walidacji *cross-day*, pozwalającej na trening modelu decyzyjnego z udziałem próbek zebranych w pierwszym dniu oraz testowanie z udziałem próbek zgromadzonych w innym dniu. Takie podejście pozwala na bardziej realistyczną ocenę skuteczności opracowanych metod, gdyż bardziej odzwierciedla rzeczywiste warunki działania systemu.

W pierwszym etapie klasyfikacji hierarchicznej pełny zbiór danych został podzielony na podzbiory według różnych kryteriów: płeć, waga, wzrost oraz wskaźnik masy ciała (BMI). W każdej z tych kategorii zastosowano nadrzędne klasyfikatory binarne oraz podrzędne klasyfikatory identyfikujące konkretne osoby. Takie rozwiązanie przyjęto w klasyfikatorze hierarchicznym, aby zwiększyć precyzję i efektywność klasyfikacji poprzez redukcję złożoności danych na każdym etapie analizy. Wyniki pokazują, że podejście hierarchiczne, szczególnie oparte na kryterium płci, znacząco zwiększa skuteczność systemu identyfikacji.

Jednakże stosowanie klasyfikatorów hierarchicznych wiąże się z pewnym ryzykiem. Nawet jeśli klasyfikatory podrzędne mają wysoką skuteczność, to jeżeli nadrzędny klasyfikator będzie miał niską skuteczność, cały system również będzie miał niską jakość klasyfikacji. Dlatego kluczowe jest odpowiednie wyszkolenie klasyfikatora nadrzędnego. Wyniki badań potwierdzają potencjał zastosowania klasyfikatorów hierarchicznych w biometrii behawioralnej chodu, podkreślając znaczenie precyzyjnego podziału zbioru danych na podzbiory oraz konieczność wysokiej skuteczności klasyfikatora nadrzędnego.

2. Przegląd literatury

Zastosowanie klasyfikatorów hierarchicznych zyskało uwagę w różnych dziedzinach nauki, także w biometrii. Dają one możliwość poprawy dokładności klasyfikacji poprzez dzielenie pełnego zbioru danych na mniejsze, bardziej jednorodne podzbiory. W niniejszym przeglądzie literatury skupiliśmy się na trzech publikacjach odnoszących się do wykorzystania klasyfikatorów hierarchicznych w dziedzinie biometrii autorstwa: Kaia Cao, Liaojuna Panga, Jimina Lianga i Jie Tiana (2013) [3], Remca Bouckaerta (2009) [4] oraz Jianpinga Fana, Ji Zhanga, Kuizhiego Meia, Jinye Penga i Linga Gao (2015) [5].

W pracy [3] zaproponowano hierarchiczny klasyfikator w zakresie rozpoznawania odcisków palców. Takie podejście do problemu umożliwiało przystępne zarządzanie różnorodnością i złożonością istniejących próbek. Główne etapy obejmowały wstępne przetwarzanie danych, ekstrakcję cech oraz naukę i walidację modelu decyzyjnego. W jego skład wchodził zestaw klasyfikatorów, które stopniowo dzieliły zbiory na podzbiory, a finalnie dawały możliwości rozpoznania konkretnych etykiet. Główną zaletą

prezentowanego podejścia jest poprawa dokładności klasyfikacji. Wynika ona z możliwości redukcji liczby etykiet do rozpoznawania w każdym etapie decyzyjnym. Wadą jest jednak konieczność dokładnego wyszkolenia każdego z klasyfikatorów nadrzędnych, co może być czasochłonne i wymagające dużej liczby próbek treningowych.

Z kolei w [4] przedstawiono podejście hierarchiczne w zadaniu rozpoznawania twarzy. Proces ten podzielony został na kilka części. Początkowo system klasyfikował obrazy twarzy według głównych cech, takich jak: kształt twarzy, kolor skóry, „struktura". Następnie w każdym z podzbiorów przeprowadzano bardziej szczegółową klasyfikację. Hierarchiczne podejście pozwoliło na znaczące zwiększenie efektywności systemu rozpoznawania twarzy oraz redukcję błędów poprzez stopniowe zawężanie liczby dostępnych etykiet. Autor jako główne zalety wymieniał poprawę dokładności i szybkości klasyfikacji. Natomiast główną wadą jest skomplikowana struktura, która wymaga starannego projektowania i dostrajania każdego poziomu klasyfikatora.

W pracy [5] skupiono się na uczeniu się kosztów wrażliwych drzew klasyfikacyjnych do dużej skali klasyfikacji obrazów oraz wykrywaniu nowych kategorii. Hierarchiczne drzewo klasyfikacyjne było używane do skuteczniejszego zarządzania dużymi zbiorami danych poprzez podział na mniejsze, bardziej jednorodne grupy. Do zalet tego podejścia należą zwiększona dokładność i elastyczność w klasyfikacji nowych kategorii, jednakże wadami są złożoność obliczeniowa i potrzeba dużej ilości danych do treningu, aby skutecznie modelować koszty błędów.

Autorzy powyższych prac wskazują na korzyści płynące z zastosowania podejścia hierarchicznego. Za ogólne zalety można uznać możliwość skuteczniejszego zarządzania dużymi i złożonymi zbiorami danych oraz finalną poprawę dokładności klasyfikacji poprzez redukcję liczby porównań i stopniowe zawężanie liczebności etykiet. Wśród wad są natomiast konieczność starannego projektowania i dostrajania każdego poziomu klasyfikatora, jak również potrzeba dużej liczby próbek uczących. Zwiększa to złożoność i czasochłonność procesu treningu modeli decyzyjnych. W kontekście biometrii behawioralnej chodu, jak pokazano w tym rozdziale, podejście hierarchiczne może poprawić skuteczność systemu poprzez podział danych na mniejsze, bardziej jednorodne podzbiory oraz zastosowanie wyspecjalizowanych klasyfikatorów.

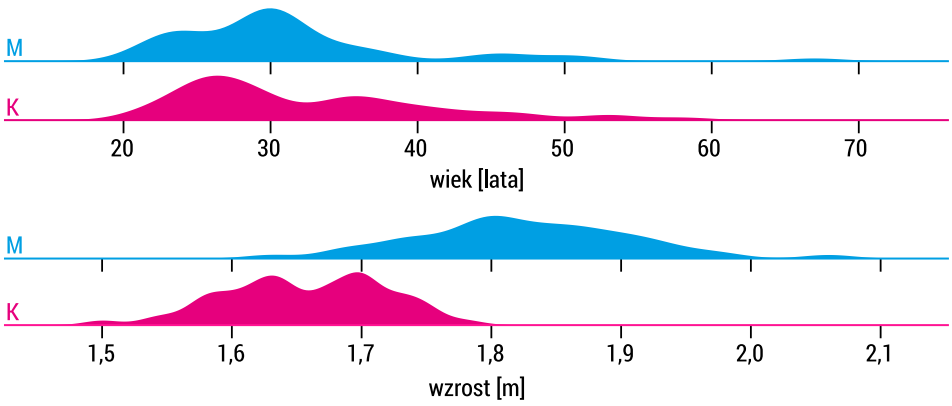
3. Opis korpusu danych

W niniejszej pracy wykorzystano autorski korpus danych zebrany na uniwersytecie macierzystym (który stanowi przedmiot wcześniejszych prac). W ramach przeprowadzonych w warunkach laboratoryjnych badań stu uczestników poproszono o naturalny chód na prostym odcinku długości około 3 metrów. W badaniach uczestniczyły osoby powiązane z białostockim środowiskiem akademickim, w tym byli i obecni studenci, pracownicy techniczni i administracyjni Politechniki Białostockiej, Uniwersytetu Medycznego w Białymstoku oraz Wyższej Szkoły Wychowania Fizycznego i Turystyki w Białymstoku. Osoby biorące udział w eksperymencie proszone były o jak najbardziej naturalne poruszanie się, bez narzuconego sposobu startu

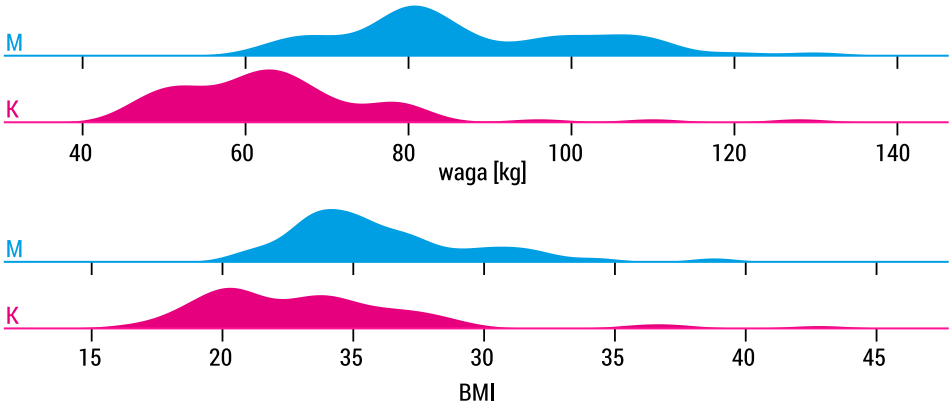
(rozpoczęcie prawą lub lewą kończyną) czy prędkości poruszania się. Każda z osób uczestniczących w badaniach wyraziła zgodę na udział w dwóch sesjach śledzenia ruchu w ciągu dwóch osobnych dni. Warunkiem wykluczającym w eksperymencie były przeszłe i aktualne urazy kończyn oraz brak chęci/możliwości uczestnictwa w dwóch sesjach śledzenia ruchu.

Podejście takie pozwoliło na wykonanie testów walidacyjnych w tzw. konfiguracji *cross-day*. Oznacza to, że trening modelu decyzyjnego przeprowadzony został z udziałem próbek zgromadzonych w ciągu jednego dnia, a ocena skuteczności w obrębie danych z dnia drugiego. Ta forma walidacji jest znacznie bardziej wymagająca od walidacji typu *single-day*, ale jednocześnie bliższa rzeczywistemu scenariuszowi działania docelowego systemu. W naszej opinii to wyniki walidacji typu *cross-day* mogą świadczyć o aplikacyjności rozwiązania, gdyż pozwalają na ocenę skuteczności w przypadku warunków bliższych docelowym warunkom działania systemu W procesie akwizycji wykorzystano system śledzenia ruchu Perception Neuron 32 firmy [2] oraz kamerę głębi Microsoft Kinect v 2.0 [6]. W badaniach skupiono się nad wykorzystaniem danych pochodzących z systemu czujników noszonych. Mimo że na specjalnym kombinezonie umieszczono 17 czujników ruchu, użyto jedynie danych pochodzących z sensorów umiejscowionych w okolicy prawego uda. Są one zbliżone do sensorów ruchu wbudowanych w telefon komórkowy i umiejscowionych w kieszeni spodni. Zarówno trójosiowy akcelerometr, jak i żyroskop miały trzy kanały pomiarowe w ramach tzw. ortogonalnych osi.

W eksperymencie uczestniczyło 47 mężczyzn oraz 53 kobiety. Średni wiek to 32,18 ± 8,73 lat; wzrost to 1,73 ± 0,11 m i waga 75 ± 19 kg. Jednak z uwagi na to, że typowe średnie wzrost i waga mogą zależeć od płci, na rys. 1 zaznaczono wykresy typu *violin* przedstawiające dane demograficzne z podziałem na płcie.



RYS. 1. Wykresy danych demograficznych w formie *violin* z podziałem na płcie



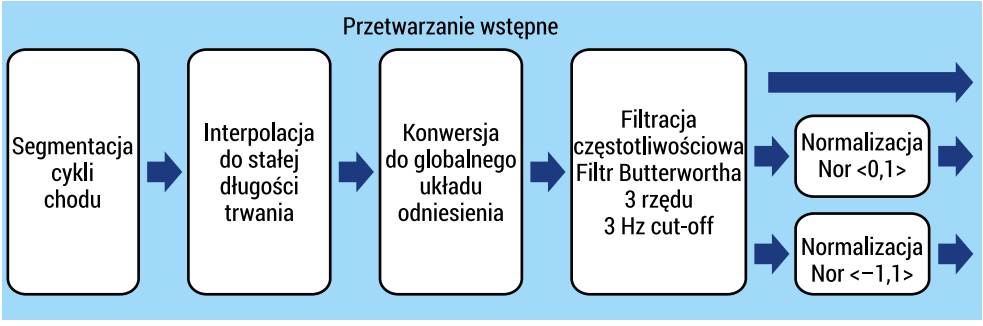
RYS. 1. Wykresy danych demograficznych w formie *violin* z podziałem na płcie (cd.)

ŹRÓDŁO: opracowanie własne.

Rysunek 1 składa się z czterech podwykresów obejmujących takie dane demograficzne jak wiek, waga oraz BMI. Kolorem niebieskim zaznaczono dane dla mężczyzn, a różowym dla kobiet. Z przedstawionego wykresu wynika, że istnieje wyraźne rozróżnienie wartości średnich wykorzystanych danych ze względu na płeć.

4. Przetwarzanie wstępne i klasyfikacja podstawowa

Przetwarzanie wstępne danych jest kluczowym etapem w analizie biometrii behawioralnej chodu, mającym na celu poprawę jakości danych i przygotowanie ich do dalszej analizy. Proces ten składa się z kilku kroków, które zostały zilustrowane na rys. 2.



RYS. 2. Widok typowych przesuwnych szaf na odzież w magazynie

ŹRÓDŁO: opracowanie własne.

Segmentacja cykli chodu

Pierwszym krokiem w przetwarzaniu wstępnym jest segmentacja cykli chodu. Dane zarejestrowane podczas chodu uczestników są podzielone na pojedyncze cykle. Cykl odpowiada jednej pełnej sekwencji ruchu, czyli od momentu, kiedy prawa pięta dotyka podłoża, do momentu, gdy ta sama pięta panowanie dotyka podłoża. W prezentowanym podejściu pojedynczy cykl chodu interpretowany jest jako pojedyncza próbka. W dniu pierwszym zgromadzono 3376 próbek uczących, a w dniu drugim 3321 próbek testowych.

Interpolacja do stałego wymiaru

Następnym krokiem jest interpolacja danych do stałego wymiaru. W zależności od wzrostu, wagi i stylu chodzenia uczestnicy mogą mieć różne czasy trwania cykli chodu. W celu zapewnienia stałej wymiarowości danych biorących udział w procesie klasyfikacji każdy cykl chodu (próbka) jest interpolowany do stałej długości trwania.

Konwersja do globalnego układu odniesienia

Dane pomiarowe sensorów noszonych natywnie reprezentowane są w lokalnym układzie odniesienia sensora zależnym od sposobu montażu. Aby zminimalizować wpływ montażu czujnika na wartości pomiarowe, dane są konwertowane do globalnego układu odniesienia. Umożliwia to porównywanie danych niezależnie od orientacji i pozycji uczestnika podczas rejestrowania ruchu.

Filtracja częstotliwościowa i normalizacja

Po konwersji do globalnego układu odniesienia dane poddawane są filtracji częstotliwościowej w celu usunięcia szumów i zakłóceń. Następnie opcjonalnie są normalizowane do jednego z dwóch przedziałów $\langle -1,1 \rangle$ lub $\langle 0,1 \rangle$. Proces normalizacji zapewnia, że wszystkie wartości znajdują się w określonym zakresie, co jest pomocne dla osiągnięcia wysokich miar skuteczności klasyfikatorów.

Po przetwarzaniu wstępnym dane wykorzystywane są przez algorytm klasyfikacyjny. W niniejszej pracy wykorzystano sztuczną sieć neuronową typu CNN (ang. *Convolutional neural network*) przedstawioną w tabeli 1. Klasyfikator składa się z kilku warstw splotowych występujących naprzemiennie z warstwą max-pooling oraz finalnie warstw liniowych.

TABELA 1. Architektura wykorzystanego klasyfikatora

Warstwa	Typ	Szczegóły
1	Convolution	in = 1, out = 32, ks = [1,9], p = [0,4], stride = [1,2]
2	Max pooling	ks = [1,2], stride = [1,2]
3	Batch normalisation	n_features = 32, act = RELU
4	Convolution	in = 32, out = 64, ks = [1,3], p = [0,1], stride = [1,1]
5	Convolution	in = 64, out = 128, ks = [1,3], p = [0,1], stride = [1,3]

Warstwa	Typ	Szczegóły
6	Max pooling	ks = [1,2], stride = [1,2]
7	Batch normalisation	n_features = 128, act = RELU
8	Convolution	in = 128, out = 128, ks = [6,1], p = valid
9	Batch normalisation	n_features = 128, act = RELU
10	Attention	channel = 128, reduction = 8
11	Dense	in = 2048, out = 100
12	Softmax	

ŹRÓDŁO: opracowanie własne.

Skuteczność klasyfikatorów niehierarchicznych z podziałem na typ przetwarzania wstępnego została zestawiona w tabeli 2. Przedstawione wyniki wskazują, że niezależnie od zastosowanego przedziału normalizacji najwyższe miary skuteczności osiągnięto dla danych filtrowanych i normalizowanych. Brak normalizacji powodował znaczący spadek osiąganych wyników klasyfikacji.

TABELA 2. Skuteczności klasyfikatorów niehierarchicznych w zależności od typu przetwarzania wstępnego

Typ przetwarzania wstępnego	Filtrowane	Filtrowanie normalizowane ⟨0,1⟩	Filtrowane i normalizowane ⟨−1,1⟩
Skuteczności F1-score	0,742	0,929	0,929

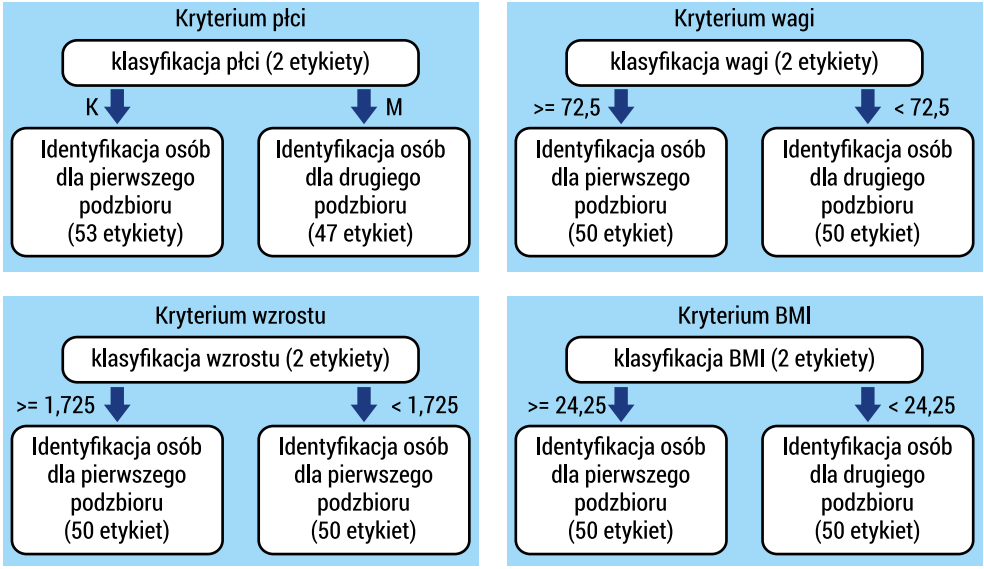
ŹRÓDŁO: opracowanie własne.

5. Klasyfikacja hierarchiczna

Jest to zaawansowane podejście w dziedzinie biometrii behawioralnej chodu, które polega na podziale pełnego zbioru danych na mniejsze, bardziej jednorodne podzbiory przed przeprowadzeniem klasyfikacji. Pozwala to na efektywniejsze zarządzanie złożonością danych oraz poprawę dokładności klasyfikacji poprzez zastosowanie specjalizowanych klasyfikatorów dla poszczególnych podzbiorów.

W prezentowanej pracy zastosowano klasyfikatory hierarchiczne oparte na różnych kryteriach podziału danych: płeć, waga, wzrost oraz wskaźnik masy ciała (BMI). Podział na podzbiory został przedstawiony na rys. 3. Proces klasyfikacji hierarchicznej składa się z dwóch głównych etapów:

- kryterium podziału – na podstawie wybranego kryterium pełny zbiór danych jest dzielony na podzbiory. Na przykład dane mogą być podzielone według płci (kobiety i mężczyźni) lub mediany wagi, wzrostu oraz BMI;
- klasyfikacja w podzbiorach – w każdym z nich stosowane są podrzędne klasyfikatory, które realizują zadanie identyfikacji osób na podstawie ich chodu.



RYS. 3. Schemat hierarchicznej klasyfikacji osób na podstawie różnych kryteriów: płci, wagi, wzrostu oraz wskaźnika masy ciała (BMI)

ŹRÓDŁO: opracowanie własne.

Wyniki klasyfikacji hierarchicznej zostały przedstawione w trzech scenariuszach przetwarzania wstępnego: filtracja danych wejściowych, filtracja z normalizacją w przedziale $\langle 0,1 \rangle$ oraz filtracja z normalizacją w przedziale $\langle -1,1 \rangle$.

TABELA 3. Skuteczność klasyfikatorów hierarchicznych w przypadku jedynie filtracji danych wejściowych

Kryterium podziału	**Płeć**		**Waga [kg]**		**Wzrost [m]**		**BMI**	
F1-score klasyfikatora binarnego	0,975		0,901		0,923		0,909	
Kryterium podzbioru	płeć = 'K'	płeć = 'M'	waga ≥ 72,5	waga < 72,5	wzrost ≥ 1,725	wzrost < 1,725	BMI ≥ 24,25	BMI < 24,25
F1-score identyfikacji osób w podzbiorze	0,773	0,776	0,816	0,807	0,805	0,805	0,787	0,813
F1-score identyfikacji osób dla klasyfikatora hierarchicznego	0,782		0,748		0,753		0,745	

ŹRÓDŁO: opracowanie własne.

TABELA 4. Skuteczność klasyfikatorów hierarchicznych w przypadku filtracji danych wejściowych i normalizacji w przedziale ⟨–1, 1⟩

Kryterium podziału	Płeć		Waga [kg]		Wzrost [m]		BMI	
F1-score klasyfikatora binarnego	0,974		0,926		0,933		0,949	
Kryterium podzbioru	płeć = 'K'	płeć = 'M'	waga ≥ 72,5	waga < 72,5	wzrost ≥ 1,725	wzrost < 1,725	BMI ≥ 24,25	BMI < 24,25
F1-score identyfikacji osób w podzbiorze	0,984	0,938	0,979	0,958	0,94	0,992	0,963	0,952
F1-score identyfikacji osób dla klasyfikatora hierarchicznego	0,941		0,892		0,900		0,914	

ŹRÓDŁO: opracowanie własne.

TABELA 5. Skuteczność klasyfikatorów hierarchicznych w przypadku filtracji danych wejściowych i normalizacji w przedziale ⟨0, 1⟩

Kryterium podziału	Płeć		Waga [kg]		Wzrost [m]		BMI	
F1-score klasyfikatora binarnego	0,973		0,925		0,929		0,945	
Kryterium podzbioru	płeć = 'K'	płeć = 'M'	waga ≥ 72,5	waga < 72,5	wzrost ≥ 1,725	wzrost < 1,725	BMI ≥ 24,25	BMI < 24,25
F1-score identyfikacji osób w podzbiorze	0,984	0,938	0,971	0,963	0,929	0,989	0,955	0,957
F1-score identyfikacji osób dla klasyfikatora hierarchicznego	0,937		0,890		0,890		0,908	

ŹRÓDŁO: opracowanie własne.

Zastosowanie klasyfikatorów hierarchicznych przyniosło zróżnicowane rezultaty w zależności od przyjętego kryterium podziału, jak również zaimplementowanej metody normalizacji. Przy braku normalizacji danych (tabela 3) osiągnięte miary skuteczności F1-score były najniższe. W pozostałych metodach przetwarzania wstępnego podział danych według kryterium płci znacząco poprawił skuteczność klasyfikacji. W przypadku normalizacji w zakresie ⟨–1, 1⟩ (tabela 4) klasyfikator binarny rozpoznający płeć osiągnął najwyższą skuteczność F1-score na poziomie 0,974. Identyfikacja osób w podzbiorze kobiet osiągnęła wynik 0,984 F1-score, a w przedziale mężczyzn 0,938. Zastosowanie podejścia hierarchicznego pozwoliło na osiągnięcie skuteczności 0,941. Stosując normalizację w zakresie ⟨0, 1⟩ (tabela 5) klasyfikator binarny rozpoznający płeć osiągnął najwyższą skuteczność F1-score na poziomie 0,973.

Identyfikacja osób w podzbiorze kobiet osiągnęła wynik 0,984 F1-score, a w przedziale mężczyzn 0,938. Użycie podejścia hierarchicznego pozwoliło na osiągnięcie skuteczności 0,937.

Zmiana skuteczności klasyfikatora nadrzędnego dla kryterium płci o tylko 0,01 (tabela 4 skuteczność 0,973, tabela 5 skuteczność 0,974) skutkowała zmianą całkowitej skuteczności o 0,04 (tabela 4 skuteczność 0,941, tabela 5 skuteczność 0,937). Wskazuje to jednoznacznie na bardzo duży wpływ zmian skuteczności klasyfikatora nadrzędnego na skuteczność klasyfikatora hierarchicznego. W przypadku normalizacji danych podejście hierarchiczne pozwoliło na podwyższenie skuteczności bazowego modelu jedynie w przypadku klasyfikacji na podstawie płci. Dla innych kryteriów podziału, takich jak waga, wzrost i BMI, nie zaobserwowano poprawy wyników. Natomiast przy braku normalizacji danych (tabela 3) podwyższenie skuteczności następowało dla każdego z kryterium, a najbardziej znaczące było dla kryterium płci (zmiana z 0,742 do 0,78 F1-score).

6. Konkluzja

W niniejszym rozdziale opisano wyniki prac podstawowych w dziedzinie biometrii behawioralnej chodu. Przedstawione rezultaty wskazują na to, że wykorzystanie klasyfikatorów hierarchicznych, w których w pierwszym etapie następuje podział pełnego zbioru na podzbiory (według predykcji wytrenowanego klasyfikatora), może mieć pozytywny wpływ na skuteczność systemu biometrycznego.

Opracowany model bazowy realizował zadanie klasyfikacji wieloetykietowej i zdolny był do poprawnego rozpoznania numeru identyfikacyjnego uczestnika (100 etykiet) ze skutecznością 0,929 F1-score dla danych znormalizowanych. W zaimplementowanym podejściu hierarchicznym klasyfikator binarny realizował zadanie rozpoznania płci uczestnika ze skutecznością 0,974 F1-score. Klasyfikatory podrzędne realizowały docelowe zadanie identyfikacji osób na podstawie chodu. Skuteczność systemu biometrycznego w przypadku kobiet (53 etykiety) wynosiła 0,984 F1-score, a w przypadku mężczyzn (47 etykiet) 0,938 F1-score. Zestaw klasyfikatorów hierarchicznych osiągnął miarę skuteczności 0,941 F1-score, co pozwoliło podwyższyć wynik bazowy.

Wzrost miary F1-score systemu biometrycznego wynika z faktu, iż opracowanie rozwiązania realizującego klasyfikację zbiorów 50-etykietowych przez dwa osobne klasyfikatory jest zagadnieniem mniej wymagającym niż rozpoznawanie 100 etykiet przez jeden model decyzyjny. Podejście takie wymaga jednak klasyfikatora nadrzędnego charakteryzującego się wysoką skutecznością. W analizowanym studium przypadku zastosowanie podejścia klasyfikatorów hierarchicznych miało korzystny wpływ na system biometryczny jedynie przy kryterium płci. W pozostałych badanych regułach progów wagi, wzrostu, wskaźnika BMI nie zaobserwowano wzrostu skuteczności identyfikacji.

*

Badania były prowadzone dzięki uzyskaniu grantu WZ/WI-IIT/5/2022 z Politechniki Białostockiej oraz finansowane ze środków Ministerstwa Nauki i Szkolnictwa Wyższego w Polsce.

Bibliografia

[1] Han J., Bhanu B., *Individual recognition using gait energy image*, „IEEE Transactions on Pattern Analysis and Machine Intelligence" 2006, vol. 28(2), s. 316–322.

[2] Perceptrion Neuron 32, https://neuronmocap.com/products/perception_neuron/ (dostęp: 16.01.2019) lub https://web.archive.org/web/20190116201850/https://neuronmocap.com/products/perception_neuron (dostęp: 24.03.2024).

[3] Cao K., Pang L., Liang J., Tian J., *Fingerprint classification by a hierarchical classifier*, „Pattern Recognition" 2013, vol. 46(12), s. 3186–3197.

[4] Bouckaert R.R., *A Hierarchical Face Recognition Algorithm*, w: *Advances in Machine Learning. First Asian Conference on Machine Learning, ACML 2009, Nanjing, China, November 2-4, 2009. Proceedings*, red. Z.H. Zhou, T. Washio, Springer, Berlin 2009, s. 38–50.

[5] Fan J., Zhang J., Mei K., Peng J., Gao L., *Cost-sensitive learning of hierarchical tree classifiers for large-scale image classification and novel category detection*, „Pattern Recognition" 2015, vol. 48(5), s. 1673–1687.

[6] Kinect for Windows, https://www.microsoft.com/en-us/kinectforwindows (dostęp: 28.05.2015) lub https://web.archive.org/web/20150522142846/http://www.microsoft.com/en-us/kinectforwindows/meetkinect/default.aspx (dostęp: 24.03.2023).

Application of Hierarchical Classifiers in Behavioral Gait Biometrics

Summary: The chapter discusses the application of hierarchical classifiers in behavioral gait biometrics, aimed at identifying individuals based on their walking patterns. The study utilized data collected using the Perception Neuron 32 motion tracking system, which monitored the gait of 100 participants during two sessions. These data were pre-processed through steps including gait cycle segmentation, interpolation, conversion to a global reference frame, frequency filtering, and normalization, and then analyzed using a convolutional neural network (CNN). Hierarchical classification involved dividing the data based on criteria such as gender, weight, height, and BMI, and identifying individuals within these subsets. This approach improved the system's effectiveness, particularly when dividing by gender, achieving an F1-score of 0.941. Data division based on other criteria did not yield significant improvements. It was shown that hierarchical classifiers can significantly enhance the efficiency of biometri:c systems by splitting data into smaller, homogeneous subsets and using specialized classifiers. The work was supported by Bialystok University of Technology and the Ministry of Science and Higher Education in Poland.

Keywords: behavioral biometrics, gait, hierarchical classifiers, person identification, neural network

Rozdział V
Porównanie wybranych technik głębokiego przetwarzania języka naturalnego w kontekście uczenia nienadzorowanego

Jakub Zaprzałka, Magdalena Topczewska
Wydział Informatyki, Politechnika Białostocka

Streszczenie: W ostatnich latach przetwarzanie języka naturalnego jest jedną z najszybciej rozwijających się dziedzin głębokiego uczenia maszynowego. Główny kierunek badań skupia się na generatywnych aspektach tego tematu, podczas gdy techniki uczenia nienadzorowanego z coraz lepszą skutecznością mogą się opierać na zakodowanej przez nie wiedzy. Grupowanie obrazów na podstawie ich opisów może być zrealizowane z wykorzystaniem wiedzy nauczonej przez modele tekstowe, w adekwatny sposób proponując klasy i jednocześnie będąc źródłem do lepszego zrozumienia danych. W ramach badań przeanalizowanych zostało kilka głębokich modeli uczenia maszynowego w różnych zadaniach przetwarzania języka naturalnego (NLP). Wybrane zostały najszerzej wykorzystywane modele, takie jak RoBERTa czy BART, które następnie użyto w kontekście grupowania aglomeracyjnego. W kolejnym kroku zastosowane zostały techniki służące do ewaluacji jakości grupowania – zarówno parametryczne, jak i subiektywne opinie eksperckie sugerujące liczbę klas w oryginalnych danych. Takie rozróżnienie pozwala przeanalizować wpływ zadań i złożoności opisów uznawanych za zbliżone oraz oddzielanie ich od siebie, a także użycie poszczególnych elementów dużych sieci neuronowych oraz wpływ ich wyboru na jakość grupowania.

Słowa kluczowe: uczenie nienadzorowane, przetwarzanie języka naturalnego, grupowanie aglomeracyjne

1. Wstęp

W ostatnich latach przetwarzanie języka naturalnego (ang. *natural language processing*, NLP) jest jedną z najszybciej rozwijających się dziedzin głębokiego uczenia maszynowego, co jest efektem przede wszystkim szybkiego rozwoju generatywnych modeli takich jak ChatGPT. Główny kierunek badań skupiony jest na generatywnych aspektach przetwarzania języka naturalnego, podczas gdy prostsze i bardziej ukierunkowane modele NLP, służące np. do klasyfikacji, mogą być z powodzeniem wykorzystywane jako źródło wiedzy w innych zastosowaniach.

Największym jednak problemem przy przetwarzaniu dużych zbiorów danych potrafi być ich zrozumienie. Grupowanie danych jest techniką wspomagającą i ułatwiającą ten proces, ale nie wszystkie zbiory danych są oparte na danych liczbowych bądź tabelarycznych. Jeżeli w zbiorze danych istotną informację niesie opis albo tekst, dobrym rozwiązaniem może się okazać przetwarzanie tego tekstu z użyciem wytrenowanego na większych korpusach danych modelu NLP.

W dalszej części rozdziału przedstawiono sposób zastosowania wybranych technik NLP w połączeniu z grupowaniem danych. Wykorzystano w tym celu zbiór obrazów z podpisami, aby lepiej zanalizować zawartość. Analizując dane, najczęściej oczekujemy pewnej liczby klas, niejednokrotnie jednak ich liczba nie jest optymalna i nie wskazuje jednoznacznych podziałów, dlatego zaproponowano również technikę sugerowania liczby klas dla zadanej metody.

2. Wykorzystane techniki i architektury modeli

W badaniach posłużono się klasycznymi modelami maszynowymi w połączeniu z głębokimi sieciami neuronowymi w architekturze *Transformer* wykorzystywanymi do różnych zadań w przetwarzaniu języka naturalnego.

2.1. Grupowanie hierarchiczne aglomeracyjne

Jako że zbiór danych nie zakłada liczby klas, do których należy je grupować, spośród klasycznych metod analizy skupień zdecydowano się na grupowanie aglomeracyjne. Jest ono jedną z najszerzej stosowanych metod analizy skupień przy nieznanej liczbie klas [1]. Przy realizacji tego algorytmu istnieją dwie jego wersje odwzorowujące dwa różne podejścia do zestawu danych – aglomeracyjne (*bottom-up*) oraz deglomeracyjne (*top-down*).

Grupowanie hierarchiczne aglomeracyjne polega na iteracyjnym łączeniu najbliższych sąsiadów w klasy. Na początku wszystkie elementy przyporządkowywane są do unikatowych, osobnych klastrów. W każdym kroku łączone są ze sobą dwa klastry oddalone o najmniejszą odległość. Proces kontynuowany jest do momentu uzyskania jednego klastra. Nieodłącznym elementem metody jest dendrogram pokazujący, które klasy zostały połączone, na jakim etapie i o ile były one od siebie oddalone. Normalizując dendrogram do wysokości jednostkowej z zachowaniem pozostałych proporcji, można, ustalając wartość odcięcia dendrogramu, otrzymać liczbę klas odpowiednio odwzorowującą zestaw danych.

W przedstawionym algorytmie należy przyjąć odpowiednie założenia, które można nazwać jego hiperparametrami – nie zależą one od konkretnego zestawu danych, ale pozwalają mieć lepszą kontrolę nad algorytmem. Jednym z nich jest metryka odległości. Definiuje ona, w jaki sposób mierzona jest odległość między dwoma punktami w przestrzeni N-wymiarowej. Odległość między klastrami również może być liczona

rozmaicie: za punkty do wyznaczenia odległości mogą być brane średnie dla klastrów (*average linkage*), najdalsze od siebie punkty (*complete linkage*) lub najbliższe (*single linkage*). Innym definiowanym dla tego algorytmu parametrem jest próg, dla którego dendrogram jest odcinany celem wygenerowania klastrów.

2.2. Przetwarzanie języka naturalnego

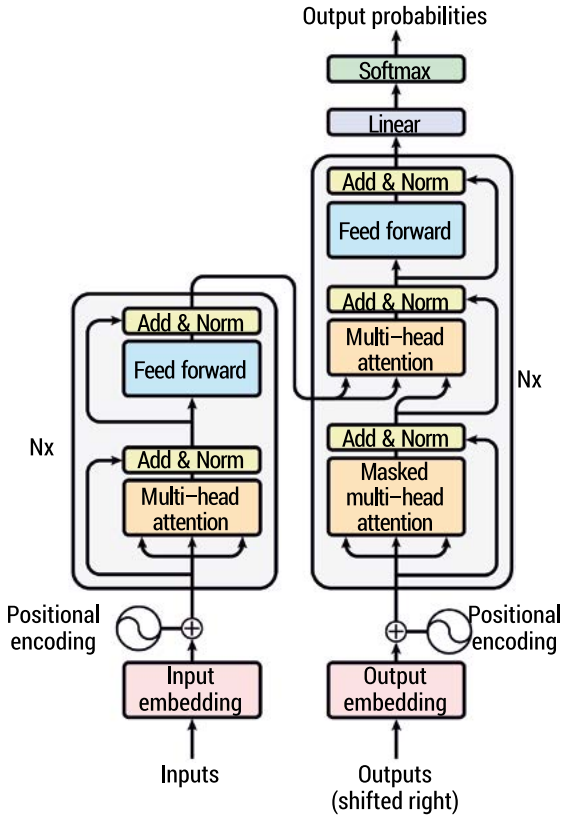
W ostatnich latach znacząco rozwinęło się przetwarzanie języka naturalnego (NLP). Rozumienie przez komputer tekstów oraz ich generowanie zyskało coraz więcej zastosowań, w miarę jak coraz więcej robotów bądź urządzeń IoT wymaga takich funkcjonalności. Ponieważ większość firm usługowych za pośrednictwem swoich stron internetowych oferuje 24-godzinne wsparcie zarówno aktualnych, jak i potencjalnych klientów, doprowadziło to do powstania niszy rynkowej w postaci tworzenia asystentów wspieranych metodami sztucznej inteligencji, tzw. chatbotów. Operują one właśnie w dziedzinie szeroko pojętego przetwarzania języka naturalnego.

Zainteresowanie tematem NLP istniało już wcześniej, niemniej jednak wyniki modeli były niezadowalające, a generowane teksty były opatrzone błędami związanymi z niejednoznacznymi regułami językowymi oraz wyjątkami od nich. Wszystkie dochodziły do pewnej bariery dokładności, głównie w aspekcie generowania tekstu czy rozumienia tekstu abstrakcyjnego. Blokującą rolę odgrywały niewystarczające moce obliczeniowe komputerów, głównie wydajność i rozmiary pamięci operacyjnych kart graficznych. Rozwój w tej dziedzinie w ciągu ostatnich dziesięciu lat niewątpliwie przyczynił się do ponownego zainteresowania tematyką sieci neuronowych wykorzystanych w problemach języka naturalnego.

Wspomnianą barierę przełamano w 2017 roku wraz z opublikowaniem przez Google architektury *Transformer* [2]. W artykule o znacząco brzmiącym tytule *Attention Is All You Need* [2] (z ang. „Uwaga to wszystko, czego potrzebujesz") zaprezentowano architekturę, z której od tej pory zaczęły korzystać wszystkie modele trafiające na listę SOTA w poszczególnych zadaniach z dziedziny NLP. Z czasem mechanizm uwagi będący podstawą architektury *Transformer* oraz różne jego odmiany pozwoliły modelom na osiąganie SOTA również w innych dziedzinach, np. w przetwarzaniu obrazów (przykładem jest model DETR [3]).

Mechanizm uwagi wykorzystywany w modelach z dziedziny *Transformer* zbudowany jest z modułów. Główną jego częścią jest pojedyncza warstwa 1-wymiarowej uwagi (ang. *Scaled Dot-Product Attention*). Moduł taki przetwarza trzy wejścia: wektor kluczy słownikowych K o wymiarze d_k, wektor odpowiadających kluczom wartości V o wymiarze d_v oraz macierz zapytań Q. Matematyczną reprezentację mechanizmu można przedstawić za pomocą równania (1) [2]:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$



RYS. 1. Architektura modelu *Transformer*

ŹRÓDŁO: [2].

Celem zwiększenia wydajności i jakości mechanizmu stosowanych jest wiele takich warstw dla wejść rzutowanych liniowo na ich kolejne wejścia (Q, V oraz K). Tak zmodyfikowana wielowarstwowa wariacja mechanizmu nazywana jest z ang. *Multi-Head Attention*.

W kolejnych sekcjach zaprezentowano modele wykorzystane w pracy. Wszystkie oparte są na architekturze *Transformer*.

2.2.1. BART

W 2019 roku Facebook AI (aktualnie Meta AI) zaprezentował model BART [4], który korzysta z opisanej wcześniej architektury *Transformer* i rozwija głównie możliwości modeli do tworzenia streszczeń (ang. *summarization*) – przede wszystkim abstrakcyjnych (ang. *abstractive summarization*) – oraz generowania tekstu i historii (ang. *story generation*). Model zaprezentował nową, łączącą najlepsze dotychczasowe aspekty

architekturę i jednocześnie wprowadził nowatorską metodę trenowania, dzięki czemu w aspekcie generowania streszczeń znajduje się na czołowych pozycjach listy SOTA.

Architektura modelu BART korzysta z dwóch czołowych architektur opartych na *Transformer*. Twórcy połączyli je w ten sposób, aby maksymalizując ich zalety oraz pożądane cechy, umiejętnie zminimalizować ewentualne wady bądź niechciane ograniczenia w działaniu.

Jako jeden z elementów kluczowych architektur wykorzystano moduł dwukierunkowego kodera. Jest on główną częścią modelu BERT [4] – jednego z pierwszych modeli opartych na *Transformer*, którego podstawowy szkielet znalazł zastosowanie w ponad połowie wszystkich późniejszych modeli korzystających z mechanizmu uwagi. Moduł pozwala na dobre odwzorowanie zależności dwukierunkowych, przewidywanych na podstawie poprzednich i następnych słów, jednak nie ma dobrych możliwości kodowania predykcji następnego słowa w trakcie generowania, czyli w implikacji generatywnego modelowania językowego.

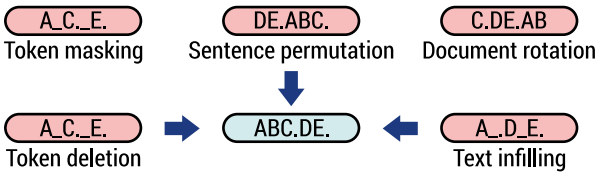
Drugi wykorzystany moduł – autoregresywny dekoder – pochodzi od modelu GPT [5], który został zaproponowany przez OpenAI w 2018 roku. Model ten w kolejnych wersjach wciąż używany jest do zadań związanych z generowaniem tekstu. Architektura dekodera pozwala na efektywne wytwarzanie wartości wstecznych oraz generatywne modelowanie języka, jednak do predykcji słów nie może używać relacji progresywnych i następnych ze względu na swoją budowę i musi ograniczyć się do relacji wstecznych. Nie jest to pożądane przy generowaniu streszczeń.

Połączenie tych dwóch modułów daje możliwości generowania tekstu z jednoczesną analizą dwukierunkową innego tekstu, na podstawie którego był on tworzony (jak w problemie generowania streszczeń). Pozwoliło to osiągnąć założony efekt polepszenia jakości modelu w tworzeniu streszczeń w porównaniu z poprzednimi najlepszymi, w czym pomogła jeszcze jedna innowacja wprowadzona przez twórców modelu: rozszerzenie sposobów augmentacji i przygotowywania danych do trenowania modelu sieci neuronowej i zastosowanie w tym aspekcie nowatorskich pomysłów.

Typowym sposobem modelowania językowego, w którym stosuje się permutacje wejściowe wyrazów (tokeny), jest tzw. MLM (ang. *Masked Language Modeling*). Aby nauczyć modele językowe charakterystyki języka, zamienia się pewien procent losowo wybranych tokenów odpowiadających wyrazom na z góry ustalony token maski (często w formie wyrazowej oznaczany jako [MASK]). Zadaniem modelu w procesie trenowania jest przewidzenie odpowiedniego wyrazu (tokenu) znajdującego się w miejscu maski.

W modelu BART oprócz opisanego standardowego podejścia zwanego maskowaniem tokenów (z ang. *Token Masking*) stosowane są inne transformacje, które poprawiają jakość modelu w tworzeniu syntetycznych podsumowań. Jednym z podejść jest usuwanie tokenów (ang. *Token Deletion*), które od zwykłego maskowania różni się tym, że miejsce usunięcia nie jest oznaczone tokenem maski. Inną wariacją na temat maskowania jest zastąpienie kilku sąsiadujących ze sobą tokenów jednym tokenem maski – twórcy modelu nazwali to wypełnianiem testu (z ang. *Text Infilling*). Pozostałe permutacje mają największy wpływ na tworzenie podsumowań, co jest głównym

zadaniem modelu. Zarówno permutacje zdań (ang. *Sentence Permutation*), jak i rotacja dokumentu (przesuwanie ostatniego lub kilku ostatnich tokenów na początek wejścia – ang. *Document Rotation*) pozwalają w procesie modelowania językowego lepiej wychwycić zależności z innych fragmentów wejścia (nazywanego w nomenklaturze modelowania językowego dokumentem), które w oryginalnym wejściu są od siebie znacznie oddalone. Wszystkie te podejścia – wraz z przykładami przedstawione na rysunku 2 – można ze sobą łączyć, tworząc kompozyty, które są również stosowane przy trenowaniu modelu BART.



RYS. 2. Transformacje ciągów tokenów stosowane w modelowaniu językowym przy trenowaniu modelu BART; w drugim wierszu na środku przedstawione jest wejście przed transformacjami

ŹRÓDŁO: [6].

Ze względu na mocne ukierunkowanie procesu trenowania modelu na wychwytywanie jak największej liczby zależności i referencji w dokumentach oraz zastosowanie dwóch przodujących technik w głębokim przetwarzaniu języka naturalnego model BART daje duże możliwości przy analizie różnego rodzaju tekstów. W związku z tym, mimo braku konkretnego ukierunkowania modelu na zadanie, wykorzystano go jako główny model do analizy języka naturalnego w podpisach obok modelu RoBERTa (opisanego w podrozdziale 2.2.2) – typowego modelu stosowanego w klasyfikacji dokumentów i zdań.

2.2.2. RoBERTa

Model RoBERTa [7] jest wariacją modelu BERT [4], trenowaną pod kątem klasyfikacji tekstu oraz analizy sentymentu. Został on zaproponowany przez naukowców z Uniwersytetu Allena, Uniwersytetu Stanu Waszyngton oraz Facebook AI (aktualnie Meta AI) w 2019 roku. Od bazowego modelu różni się sposobem trenowania – jego twórcy zoptymalizowali go pod kątem zadań klasyfikacyjnych.

Architektura modelu RoBERTa jest w pełni kompatybilna ze wzorcową architekturą modelu BERT [4] i autorzy artykułu nie wprowadzają do niej zmian. Głównymi cechami modelu są sposób trenowania i przygotowania danych, które znacznie poprawiają jego wyniki podczas porównania z metodami SOTA. Kluczową optymalizacją procesu trenowania wprowadzaną przez model RoBERTa jest jego dłuższe trenowanie na większej ilości danych. Ponadto model ten trenowany jest tylko na zadaniu maskowanego modelowania językowego (MLM), z użyciem dłuższych sekwencji

wejściowych. Dodatkowo znaczną optymalizację daje również ciągła zmiana procesu maskowania w trakcie treningu [7].

Dzięki zaawansowanym technikom optymalizacji treningu model ten jest głównie wykorzystywany do klasyfikacji tekstu, przewidywania następnego zdania (ang. *Next Sentence Prediction*, NSP) i innych flagowych zadań przetwarzania języka naturalnego. Zastosowane optymalizacje trenowania wykorzystuje się w większości analogicznych modeli językowych opartych na architekturze *Transformer* w większości języków. Pozwala to na uzyskanie znacznej poprawy wyników w klasyfikacji tekstu, analizie sentymentu i wielu innych.

3. Adaptacja modeli do potrzeb problemu

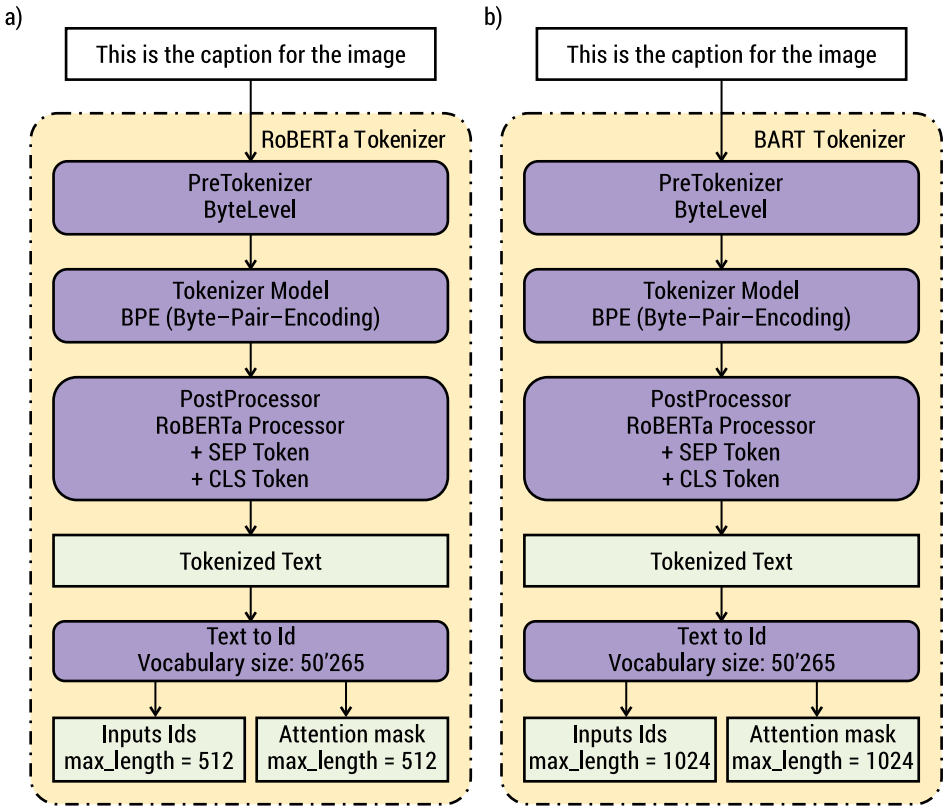
Powszechnymi praktykami są destylacja i transfer wiedzy takich modeli, aby na podstawie wiedzy nauczonej w procesie treningu i odpowiedniego przekształcenia w architekturze czy połączenia częściowo użyć ich do rozwiązywania nowych zadań. Opisane wyżej modele nie są przystosowane bezpośrednio do uczenia nienadzorowanego, jednak zostały odpowiednio wytrenowane. Aby zaadaptować je do przewidzianego w pracy zadania, wykorzystano grupowanie aglomeracyjne i różne przekształcenia.

W modelach tekstowych transformacji podlegać musi zarówno wejście, jak i wyjście. Istnieją różne podejścia do tej reprezentacji, jednak wejście jest najczęściej jednoznacznie definiowane przez model. Z kolei sposób interpretowania sygnału wyjściowego pozostawia dowolność użycia i jest uwarunkowany charakterem zadań wykonywanych przez model.

W przeciwieństwie do obrazów tekst nie jest jednoznacznie interpretowalny przez algorytmy opisujące sieci neuronowe. Nie może on być bezpośrednio przetwarzany przez sieć neuronową, bez uprzedniej jego konwersji na reprezentację liczbową. W związku z tym występują różne podejścia do jego kodowania do formy akceptowalnej przez modele sieci neuronowej. Proces zamiany ciągu tekstowego na sygnał w formie wektora liczbowego nosi nazwę tokenizacji, a moduły zapewniające jednoznaczną zamianę tekstu na wektor i wektora na tekst funkcjonują pod angielską nazwą *tokenizer*.

Tokenizer

Do zadań tokenizerów należą dwa zasadnicze elementy przetwarzania tekstu: jego podział na obecne w słowniku, atomowe i unikatowe elementy – tokeny – oraz ich zamiana na formę zrozumiałą dla komputera, czyli liczby. Podczas gdy drugi krok w pracy tokenizera jest oczywisty i nie różni się w zależności od metody (tokeny mają przypisaną liczbę naturalną odpowiadającą indeksowi danego tokena w tablicy jednoznacznie zdefiniowanej przez *tokenizer* i określanej jako słownik), metoda uzyskania wyjścia z pierwszego kroku zależy od podejścia i metody implementacji.



RYS. 3. Transformacje wejściowe modeli tekstowych – tokenizery modeli RoBERTa (a) i BART (b). Obydwa modele używają niemal identycznych transformacji, a modyfikują jedynie maksymalną długość wektora wejściowego do modelu max_length

ŹRÓDŁO: opracowanie własne.

Najprostszą formą tokenizacji jest podział na tokeny przeprowadzony za pomocą spacji w ciągu tekstowym. Tu pojawiają się jednak problemy ze znakami interpunkcyjnymi. Można je wydzielić jako osobne tokeny bądź zignorować, aby uzyskać bardziej jednoznaczną reprezentację słów, jednak przy tak przeprowadzonej tokenizacji rozmiar słownika może być rzędu 10^5, co może znacznie spowolnić obliczenia. Przykładem modelu wykorzystującego dzielenie dokumentu na tokeny za pomocą znaków interpunkcyjnych oraz innych reguł gramatycznych i syntaktycznych (moduły takie znane pod nazwą *rule-based tokenizers*) jest model *Transformer XL* [8].

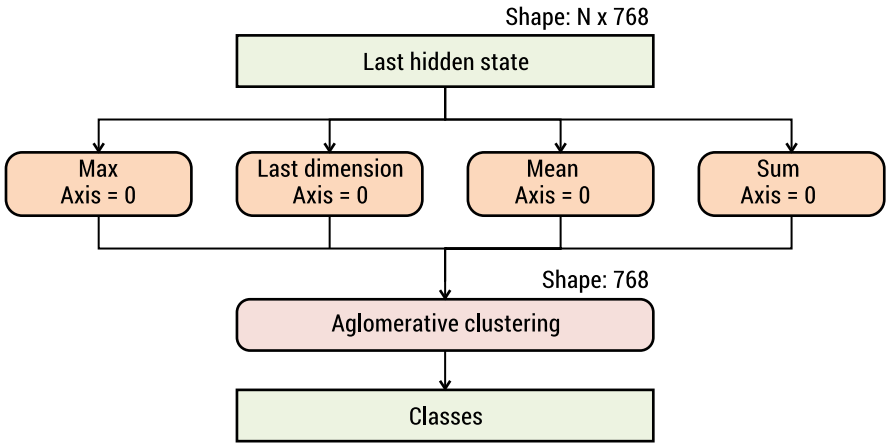
Wielkość słownika można zminimalizować, zamieniając na tokeny fragmentów słów. Te ostatnie są wybierane na podstawie korpusu tekstowego, na którym trenowany jest *tokenizer*. Niekoniecznie muszą być lematami czy bazową formą słowa, czy nawet odnosić się do tego samego słowa. Jest to niejednoznaczne i zależy od korpusu tekstowego oraz procesu treningu. W ten sposób tworzony i trenowany jest m.in. *tokenizer* modelu BERT [4].

Podczas analizy tekstu wejściowego można też napotkać znaki nieznane, takie jak emotikony, czy inne, nietypowe dla tekstu w rozumieniu lingwistycznym, jednak często stosowane w treściach w internecie – znaki UNICODE. Przy zastosowaniu dzielenia wyrazów z użyciem opisujących znaki bajtów można osiągnąć słownik wielkości 50 257. Jest to wykonalne bez konieczności używania tokenu oznaczonego <unk> – w innych tokenizerach oznaczającego nieznany w danym module symbol bądź znak. Takie podejście nosi nazwę *byte-level* BPE (ang. *byte-level Byte-Pair-Encoding*) i stosowane jest przykładowo w modelu GPT-2 [5].

Istnieją również metody tokenizacji, dla których długość fragmentu słowa jest z góry definiowana, a nie uczona na podstawie korpusu i różna dla różnych słów. Takie fragmenty o stałej długości nazywane są ogółem *n*-gramami, gdzie *n* jest liczbą znaków / długością fragmentu. Mogą one być nawet pojedynczymi znakami (unigramami). Model bazujący na konstrukcjach typu unigram jest najczęściej używany w połączeniu z innymi metodami w tłumaczeniu maszynowym ze względu na różnice między słowami w różnych językach [9].

Każdy z modeli wykorzystuje z góry zdefiniowany *tokenizer* zbudowany na podstawie przedstawionych tu metod oraz innych, niejednokrotnie innowacyjnych lub podstawowych technik. W odpowiedni sposób przygotowuje on dane pod kątem wejść konkretnego modelu oraz zapewnia spójność nauczonej przez model w procesie treningu zależności w kontekście poszczególnych słów, wyrażeń oraz ich znaczeń. Przygotowanie wejść oraz analizę wyjść dla każdego z modeli opisano w podrozdziałach 3.1 (model RoBERTa) oraz 3.2 (model BART).

Transformacje wyjściowe



RYS. 4. Transformacje sygnału wyjściowego w modelach tekstowych RoBERTa i BART. Używane są stany sygnału z ostatniej warstwy ukrytej sygnału z odpowiedniego modelu. Wymiar sygnału przetwarzanego przez model jest inny dla modeli i oznaczony na schemacie przez N. Dla modelu RoBERTa N = 512, z kolei dla modelu BART N = 1024

ŹRÓDŁO: opracowanie własne.

Kolejną częścią wspólną dla przetwarzania sygnału w rozpatrywanych modelach tekstowych są transformacje wyjściowe. Dla każdego modelu tekstowego (RoBERTa i BART) przetwarzanym wyjściem jest macierz odwzorowująca stan sygnału dla ostatniej warstwy (ang. *Last Hidden State*). Transformacje prowadzące do jej osiągnięcia opisano w poniższych podrozdziałach. Macierz ta poddana jest jednak wspólnym transformacjom.

3.1. RoBERTa

Model RoBERTa, jak wskazano w podrozdziale 2.2.2, ma identyczną zasadniczą strukturę jak model BERT. Jednakże jedną z rzeczy ulepszonych w nowszym modelu jest właśnie odpowiedni *tokenizer*, który nie stwarza problemów przy przetwarzaniu rozmaitych tekstów pobieranych z internetu, zawierających znaki spoza zakresu alfanumerycznego czy interpunkcyjne (najczęściej tzw. emotikony).

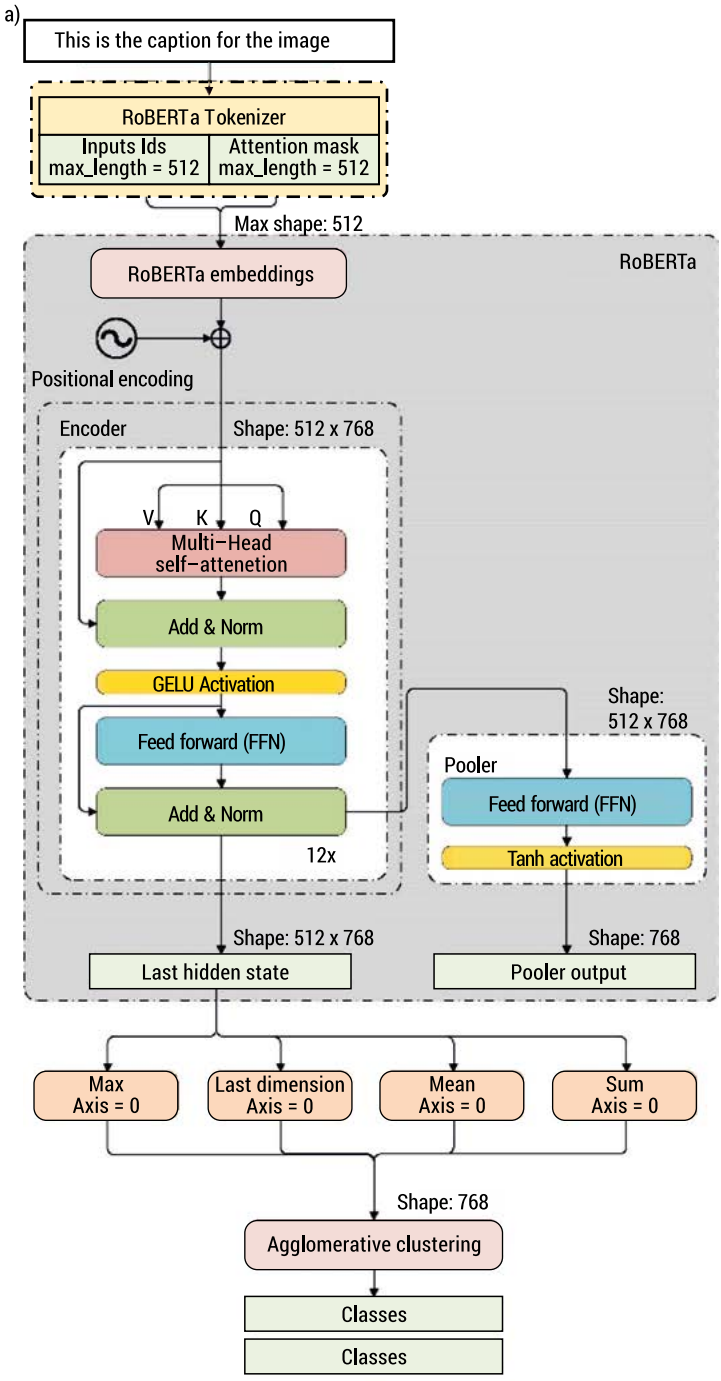
Tokenizerem najlepiej adresującym te problemy jest *byte-level* BPE i z tego względu użyty został do reprezentacji tokenów modelu RoBERTa (rys. 3a). Zapewnia on jednoznaczny odczyt tokenów na poziomie bajtów służących w komputerowej reprezentacji tekstu kodowaniu – najczęściej UTF-8, UTF-16 czy standardowym ASCII. Słownik tokenizera jest wielkości stałej 50 265 i definiuje on standardowe kodowanie tekstu na liczby. Na wyjściu tokenizera i zarazem wejściu do modelu otrzymujemy dwa wektory wymiaru 512 – wejście i maskę. Jako że wejścia modelu muszą być zawsze tego samego wymiaru, krótsze sekwencje w wektorze wejściowym *Input* wypełniane są tokenami reprezentowanymi przez puste wartości (najczęściej 0 lub –1) i oznaczanymi jako PAD_TOKEN. Z kolei wartości wektora maski uwagi (ang. *attention mask*) oznaczają miejsca, które są istotne dla modelu. Nieistotne wartości wektora oznaczone są wtedy wartościami 0, a istotne 1.

Model RoBERTa przetwarza przygotowany przez *tokenizer* sygnał w sposób przedstawiony na rys. 5a. Pierwszym elementem przetwarzania wektora wejściowego jest jego przekształcenie w specyficzną macierz zwaną macierzą zanurzeń (ang. *embedding matrix*). Transformacja ta zamienia wektor o maksymalnym wymiarze 512 na macierz 512 × 768. Zanurzenia służą odwzorowaniu powiązań znaczeniowych słów oraz ich pozycji w dokumencie i zakodowaniu ich na 768 wymiarach. Kolejne kroki operują na wygenerowanej macierzy zanurzeń w typowej architekturze modeli z grupy *Transformer*. Wykorzystanych jest dwanaście warstw modułu Encoder.

Jako funkcja aktywacji stosowana jest standardowa dla większości modeli z rodziny *Transformer* funkcja GELU (ang. *Gaussian Error Linear Units*) [10]. W zapisie matematycznym odwzorowuje ją równanie (2):

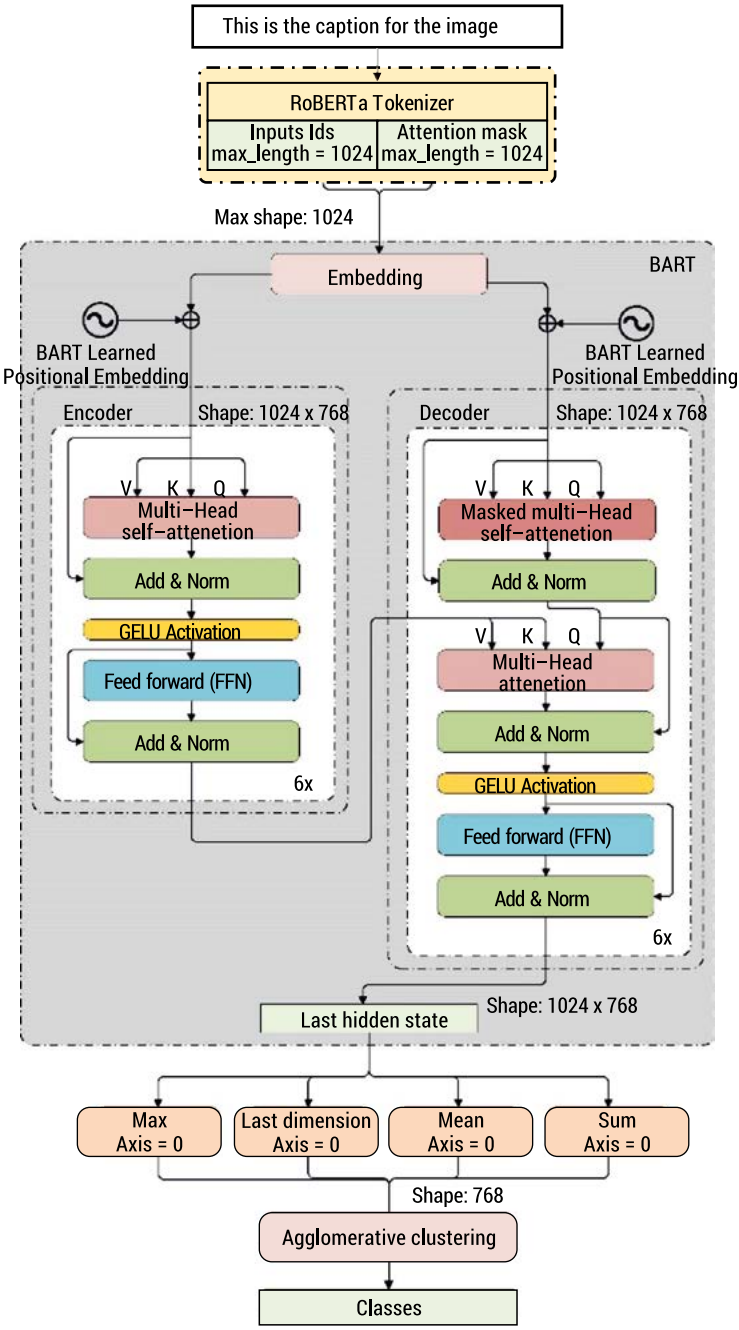
$$GELU(x) = xP(X \le x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{2}$$

przedstawiające wartość funkcji GELU w zależności od sygnału wejściowego x.



RYS. 5. Przepływ sygnału i transformacje w architekturze modeli RoBERTa (a)

b)



RYS. 5. **Przepływ sygnału i transformacje w architekturze modeli BART (b)**

ŹRÓDŁO: **oprac. na podstawie [7], [2].**

W tradycyjnym modelu RoBERTa wyjście z modułu Encoder przepuszczone jest jeszcze przez moduł Pooler. Wybiera on kolejne słowo do wylosowania w przypadku użycia modelu do generowania tekstu. Niemniej jednak w klasyfikacji bądź analogicznym zadaniu uczenia nienadzorowanego grupowania istotniejszym (gdyż lepiej odwzorowującym kontekst całego dokumentu) elementem jest samo wyjście z modułu Encoder – wyjścia (stany) jego ostatniej ukrytej warstwy [4]. Jest to macierz wymiaru analogicznego do macierzy zanurzeń. Poddaje się ją omówionym wspólnym transformacjom, aby uzyskać wektory wejściowe do modelu grupowania hierarchicznego.

3.2. BART

Model BART, jak opisano w podrozdziale 2.2.1, ma analogiczną do rodziny modeli BERT strukturę wejściową (część enkodera – ang. *Encoder*), jednak warstwa wyjściowa jest elementem struktury GPT (moduł dekodera – ang. *Decoder*).

Tokenizacja w modelu BART jest niemal identyczna ze schematem modelu RoBERTa opisanym w poprzednim podrozdziale. Jedyną różnicą w budowie tokenizera jest długość wektora wejściowego, który ma wymiar 1024. Wszystkie jego pozostałe funkcjonalności są identyczne, a pełną jego konstrukcję przedstawiono na rys. 3b.

Dalsze transformacje modelu BART zaprezentowano na rys. 5b. Zastosowano sześć warstw enkodera i sześć warstw dekodera. Ostatnia warstwa dekodera jest uznawana za ostatnią warstwę ukrytą (ang. *Last Hidden Layer*) i wyjście z niej w formie macierzy 1024×768 jest używane jako wejście do wspólnych dla obu modeli tekstowych transformacji wyjściowych.

4. Analiza i interpretacja wyników

W ramach prac badawczych przeprowadzono eksperymenty uczenia nienadzorowanego, w szczególności grupowania, na zbiorze danych wielodziedzinowych pozyskanych ze źródeł [11]. Zgromadzono 9138 obrazów w różnych rozmiarach wraz z podpisami. Z użyciem technik opisanych w podrozdziale 3 dokonano transformacji sygnałów wejściowych (podpisów w formacie tekstowym dla rekordów) do postaci wektorów.

Przy transformacji zastosowano wagi sieci neuronowych wyuczone na zdecydowanie większych korpusach tekstowych. Pochodzą one z przeróżnych źródeł i opisane są przez twórców modeli [6], [7]. W ten sposób wykorzystano technikę transferu wiedzy.

Celem zachowania przejrzystości wyników wprowadzono oznaczenia technik, które poddano eksperymentom. Oznaczenia mają charakter $\mathrm{NLP}_{\mathrm{model}}$, gdzie model jest wyróżnikiem konkretnej techniki w ramach modeli (BART lub RoBERTa).

Dla każdej z technik uruchomiono grupowanie hierarchiczne 150 razy, każdorazowo odpowiednio dobierając hiperparametry. Grupowanie oceniano następnie

za pomocą metryk pod względem jakości. Typy liczonych metryk wykorzystane w procesie grupowania opisano szerzej w podrozdziale 4.1.

Następnym krokiem zbierania i oceny wyników była analiza porównawcza najlepszych wyników grupowania dla każdej techniki uzyskania wektorów. Przeprowadzono analizę jakościową grupowania. Porównanie wydajnościowe nie miało uzasadnienia, gdyż rozmiary wyjściowe wektorów dla obu technik były identyczne.

4.1. Metryki grupowania

Aby poczynić wnioski na temat jakości grupowania, należało wprowadzić sposób jego oceny. Dlatego policzono odpowiednie metryki oraz zaadaptowano je pod kątem założonej liczby klastrów.

Grupowania w tej formie, w przeciwieństwie do klasyfikacji, nie można oceniać metrykami takimi jak dokładność czy czułość, gdyż nie ma możliwości podania rzeczywistych klas nawet dla pewnego podzbioru danych. Poprawne grupowanie może zwrócić kilka możliwych konfiguracji klas, z których każda może być poprawna. Kluczową miarą jakości tak postawionego problemu nie jest przyporządkowanie danych do konkretnych klas (charakteryzuje to problem klasyfikacji), ale zachowanie powiązań między obiektami przynależącymi do tej samej grupy – ścisłość klastra (ang. *cluster tightness*). Jednocześnie obiekty w jednym klastrze powinny być możliwie różne od obiektów w pozostałych klastrach – separacja klastra (ang. *cluster separation*).

$$t_i = \frac{1}{|C_i| - 1} \sum_{\forall \{j \in C_i : j \neq i\}} dsm(i, j) \tag{3}$$

jest miarą ścisłości klastra, a przyjmując za C_x zbiór elementów znajdujących się w innym klastrze,

$$s_i = \min \left\{ \left(\frac{1}{|C_x|} \sum_{\forall j \in C_x} dsm(i, j) \right) : C_x \neq C_i \right\} \tag{4}$$

jest miarą jego separacji. Wtedy:

$$sh_i = \frac{s_i - t_i}{\max\{t_i; s_i\}} \tag{5}$$

Metryka *silhouette*, liczona dla każdego elementu danych w zbiorze danych A (sh_i), oddaje liczbowo porównanie między ścisłością klastra a jego separacją (5). Końcowy indeks *silhouette* jest prostą średnią arytmetyczną wartości sh_i dla każdego elementu w zbiorze A, jak przedstawiono w równaniu (6):

$$sh = \frac{1}{|A|} \sum_{\forall i \in A} sh_i \tag{6}$$

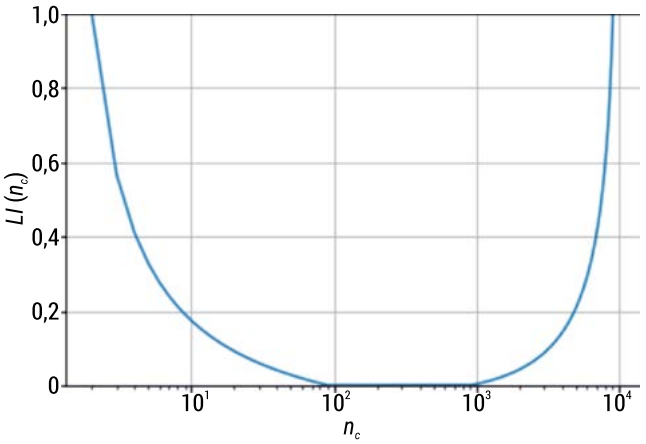
Metryka ta często stosowana jest celem wybrania optymalnej liczby klastrów w problemach grupowania ze względu na swoją uniwersalność w podejściu do problemu. Najczęściej daje ona również najbardziej adekwatne wyniki w porównaniu z innymi metodami dobierania klastrów z tej dziedziny. Z uwagi na wiele zalet została użyta na wyjściu konfiguracji hiperparametrów jako główny składnik funkcji straty.

Za pożądaną i optymalną liczbę klastrów przyjęto przedział [1–10%] liczności zbioru. Aby nie wymuszać takiej liczby grup, przygotowano funkcję, która działa jak mnożnik dla metryki grupowania. W połączeniu z nią może służyć jako funkcja straty.

Funkcja adaptująca funkcję straty (*LI*) pod założone kryteria zdefiniowana jest według przyjętych przedziałów. Przyjmując za *M* liczność zbioru, otrzymujemy odpowiednio $n_{LB} = [1\%M]$ oraz $n_{UB} = [10\%M]$. Wtedy funkcję adaptującą funkcję straty w funkcji liczby klastrów (n_C) przedstawia równanie (7)

$$LI(n_c) = \begin{cases} 0, & \text{dla} \quad n_c \in [n_{LB}; n_{UB}] \\ 1, & \text{dla} \quad n_c \in \{1; M\} \\ \dfrac{\ln n_{LB} - \ln n_c}{\ln n_c} \cdot \dfrac{2}{\ln n_{LB} - \ln 2}, & \text{dla} \quad n_c \in [2; n_{LB}) \\ \dfrac{\ln M - \ln(M + n_{UB} - n_c)}{\ln(M + n_{UB} - n_c)} \cdot \dfrac{\ln(x_{UB} + 1)}{\ln x_{MAX} - \ln(n_{UB} + 1)} & \text{dla} \quad n_c \in (n_{UB}; M) \end{cases} \tag{7}$$

Wykres funkcji *LI* przedstawiono na rys. 6. Zastosowano zbocza logarytmicznie opadające we wzorze funkcji, przez co drobne odchylenia od preferowanej liczby klastrów mają wartości bliskie 0. Niemniej jednak im dalej od preferowanego zakresu, tym bliższe 1 są wartości funkcji.



RYS. 6. **Wykres funkcji adaptującej funkcję starty *LI*(n_C)**

ŹRÓDŁO: **opracowanie własne.**

Funkcja *LI* w połączeniu z metryką *silhouette* tworzy funkcję straty *loss*. Jest ona kombinacją liniową tych dwóch wartości. Wzór funkcji straty przedstawia równanie (8):

$$loss = (-1) \times sh + 0,5 \times LI(n_c) \tag{8}$$

Tak przygotowana funkcja straty *loss* oparta na metryce *silhouette* przyjmuje wartość minimalną (*loss* = –1) dla optymalnego dopasowania liczby klastrów do zbioru oraz do przyjętych założeń. Funkcja *LI* zmienia wartość metryki o maksymalnie 0,5.

4.2. Analiza i interpretacja wyników

Dla każdej techniki procesów grupowania ze zmienionymi hiperparametrami przeprowadzono jakościową analizę porównawczą. Skorzystano z metryk grupowania, wartości funkcji straty oraz subiektywnej oceny wizualnej. Przeanalizowano eksperymenty dla technik opisanych w podrozdziale 3.

Również dla każdej z poszczególnych dziedzin przeprowadzono analizę porównawczą wyników pod względem jakościowym. W większości modeli skorzystano głównie z metryk *silhouette score* oraz funkcji straty *loss*.

Do analizy wykorzystane zostały wykresy pokazujące wartości metryk dla poszczególnych eksperymentów wraz z zaznaczeniem ich wartości optymalnych (minimalnych lub maksymalnych). Eksperymenty powiązane są również z liczbą klastrów powstałych w wyniku grupowania, co także zaznaczono na wykresach.

Przeanalizowano dwa modele wraz z powiązanymi z nimi eksperymentami NLP_{BART} oraz $NLP_{RoBERTa}$.

Adaptacja modeli będących podłożem tych eksperymentów przedstawiona w podrozdziale 3 definiuje dodatkowy hiperparametr – sposób liczenia wektora użytego do grupowania. Wybór jednej z opisanych czterech metod[2] przyjmowany jest jako hiperparametr i brany jest on pod uwagę przy wyborze optymalnej konfiguracji hiperparametrów grupowania.

TABELA 1. Wyniki dziesięciu najbardziej optymalnych pod względem wartości funkcji straty *loss* eksperymentów dla metody NLP_{BART} wraz z odpowiadającymi im hiperparametrami grupowania. Liczbę klastrów oznaczono przez n_c

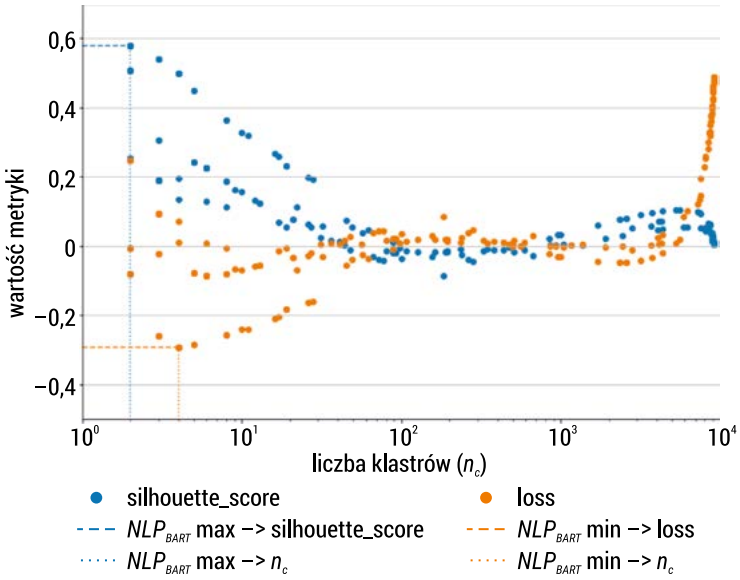
Parametry			**Metryki**		
distance threshold	***affinity***	**wybór wektora**	n_c	***loss***	***silhouette score***
0,2541	cosine	Last Dimension	**4**	**–0,2939**	0,4985
0,2451	cosine	Last Dimension	5	–0,2844	0,4480
0,2806	cosine	Last Dimension	**3**	–0,2592	**0,5411**
0,2315	cosine	Last Dimension	8	–0,2570	0,3631

[2] Metody adaptacji macierzy zanurzeń do wektora to: Max, Mean, Sum i Last Dimension.

Parametry			Metryki		
distance threshold	*affinity*	wybór wektora	n_c	*loss*	*silhouette score*
0,2204	cosine	Last Dimension	10	−0,2418	0,3288
0,2180	cosine	Last Dimension	11	−0,2395	0,3195
0,2078	cosine	Last Dimension	16	−0,2093	0,2662
0,2075	cosine	Last Dimension	17	−0,2044	0,2581
0,2042	cosine	Last Dimension	19	−0,1835	0,2318
0,1950	cosine	Last Dimension	26	−0,1643	0,1992

ŹRÓDŁO: opracowanie własne.

Jako pierwszą analizowano technikę opartą na modelu BART. Dla optymalnego uruchomienia wybrane zostały cztery klastry, a najlepsze wyniki osiągnięto przy użyciu odległości cosinusowej, co jest typowe dla modeli tekstowych. W analizie wykorzystano Last Dimension, element macierzy zanurzeń dającą wektor użyty przy tym uruchomieniu. Wartości poszczególnych metryk oraz liczbę klastrów dla dziesięciu najbardziej optymalnych uruchomień przedstawiono w tabeli 1, a zestawienie uruchomień grupowania zobrazowano na rys. 7. Na wykresie można również zauważyć wpływ modyfikacji realizowanych w ramach adaptacji funkcji straty na metrykę *silhouette score*, jak też wyniki eksperymentów równolegle dla innych wartości optymalnych wektora na wyjściu do algorytmu grupowania.



RYS. 7. Wartości metryk dla eksperymentów z techniki NLP_{BART}. Dodatkowo zaznaczono optymalną dla danej metryki (minimalną lub maksymalną) wartość spośród uruchomień, która odwzorowuje najlepszy wynik grupowania według danej metryki

ŹRÓDŁO: opracowanie własne.

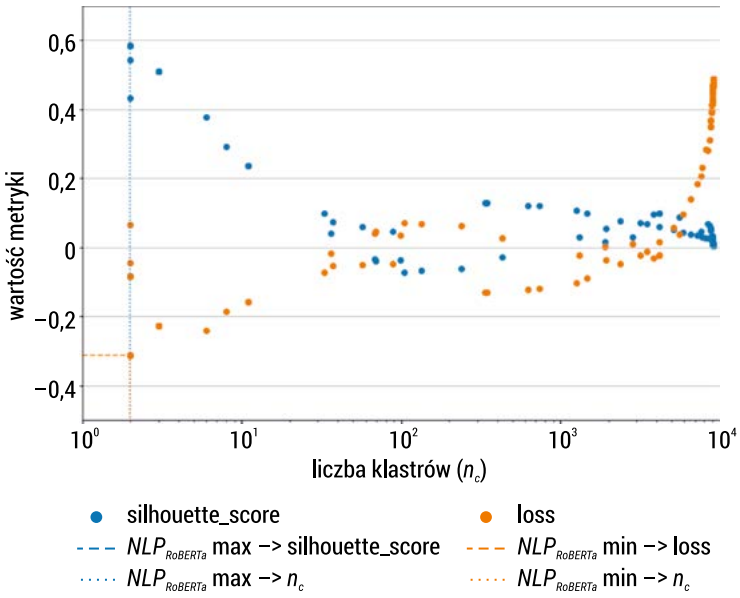
TABELA 2. Wyniki dziesięciu najbardziej optymalnych pod względem wartości funkcji straty *loss* eksperymentów dla metody $NLP_{RoBERTa}$ wraz z odpowiadającymi im hiperparametrami grupowania. Liczbę klastrów oznaczono przez n_c

Parametry			Metryki		
distance threshold	***affinity***	**wybór wektora**	n_c	***loss***	***silhouette score***
0,1165	cosine	Sum	**2**	**−0,3135**	**0,8135**
0,1699	cosine	Mean	2	−0,3135	0,8135
0,0568	cosine	Sum	**6**	**−0,2403**	0,3781
0,0679	cosine	Mean	3	−0,2278	0,5098
0,0715	cosine	Sum	3	−0,2278	0,5098
0,0535	cosine	Mean	8	−0,1856	0,2918
0,0502	cosine	Mean	11	−0,1581	0,2381
0,0507	cosine	Sum	11	−0,1581	0,2381
98,6042	l2	Sum	335	−0,1302	0,1302
98,1712	l2	Sum	348	−0,1294	0,1294

ŹRÓDŁO: opracowanie własne.

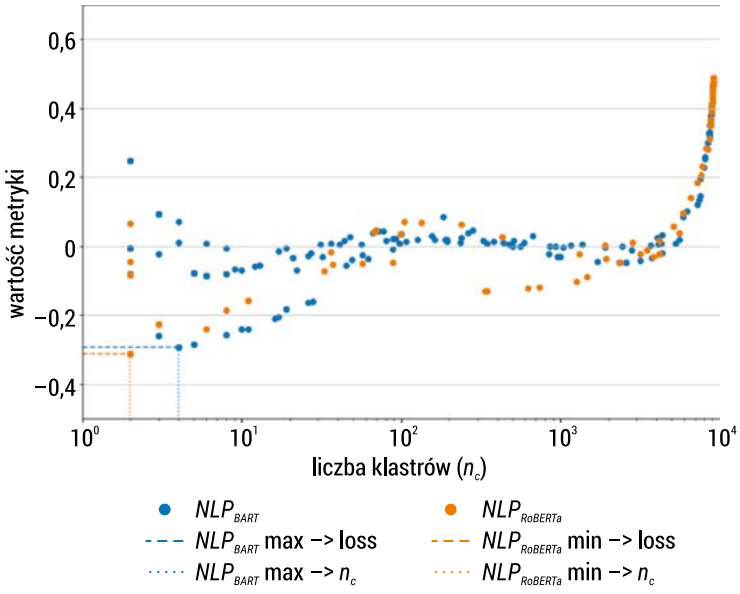
Podobne wartości funkcji straty uzyskuje model RoBERTa, dla którego optymalne uruchomienie skutkuje wygenerowaniem dwóch klastrów, a najlepsze wyniki również osiągnięto przy użyciu odległości cosinusowej. Niemniej jednak użytym elementem macierzy zanurzeń dającą wektor użyty przy tym uruchomieniu jest Sum lub Mean – obydwa dają ten sam wynik dla wartości funkcji straty *loss*. Wartości poszczególnych metryk dla dziesięciu najlepszych uruchomień przedstawia tabela 2, a wizualne zestawienie uruchomień grupowania pokazano na rys. 8. Na wykresie można również zauważyć nieznaczny wpływ modyfikacji realizowanych w ramach adaptacji funkcji straty na metrykę *silhouette score*.

Wszystkie wyniki dla modeli z dziedziny przetwarzania języka naturalnego agreguje wykres z rys. 9, pokazując wartości funkcji straty dla eksperymentów odpowiadających każdemu z modeli. Można na nim zaobserwować bardzo duże podobieństwo między wynikami uzyskanymi z pomocą modeli BART i RoBERTa. Z analizy poprzednich wykresów oraz tabel wynika, iż w przypadku eksperymentu NLP_{BART} adaptacja funkcji straty wobec metryki przyniosła zdecydowanie większy wpływ. W związku z tym, mimo że wartość *loss* dla eksperymentu $NLP_{RoBERTa}$ okazuje się nieznacznie mniejsza, za najlepszy model spośród modeli tekstowych uznano model BART. Był on bardziej skłonny do adaptacji do sugerowanej liczby klastrów, jednocześnie zachował zbliżoną wartość funkcji straty. Model ten został również oceniony jako najbardziej adekwatny w rozpatrywanym przypadku.



RYS. 8. Wartości metryk dla eksperymentów z techniki $NLP_{RoBERTa}$. Dodatkowo zaznaczono optymalną dla danej metryki (minimalną lub maksymalną) wartość spośród uruchomień, która odwzorowuje najlepszy wynik grupowania według danej metryki

ŹRÓDŁO: opracowanie własne.



RYS. 9. Porównanie technik NLP pod względem wartości funkcji straty *loss* dla rozpatrywanych eksperymentów z użyciem danego modelu. Dodatkowo zaznaczono minimalną wartość funkcji straty, która odwzorowuje najlepszy wynik grupowania dla danego modelu

ŹRÓDŁO: opracowanie własne.

5. Podsumowanie

W ramach prac badawczych poświęconych porównaniu technik głębokiego uczenia nadzorowanego w przetwarzaniu języka naturalnego w kontekście uczenia nienadzorowanego wykorzystano modele RoBERTa i BERT służące klasyfikacji bądź generowaniu tekstu. Na podstawie tych modeli oraz ich wyuczonych parametrów z użyciem transferu wiedzy wykorzystano je do analizy zbioru niemal 10 000 rekordów obrazów z podpisami (w szczególności w badaniach analizowane były podpisy).

Podczas eksperymentów przeprowadzono serię prób dopasowania modelu klastrowania hierarchicznego aglomeracyjnego ukierunkowanego na dwa cele – jak najlepsze dopasowanie do danych oraz osiągnięcie bądź zbliżenie się do przyjęto przedziału od 1% do 10% liczby klastrów przy zachowaniu możliwie najlepszego dopasowania. Pierwszy efekt osiągnięto, wykorzystując metrykę *silhouette score* badającą jakość dopasowania zestawu danych do osiągniętych klastrów. Dla połączenia z sugerowaną liczbą klastrów wykorzystano mnożnik funkcji straty kierunkowany liczbą klastrów.

Po otrzymaniu wyników grupowania przeprowadzono porównanie jakościowe metod oraz analizę wpływu mnożnika funkcji straty na liczbę klastrów. Podczas gdy klastrowanie dla modelu RoBERTa pozwala na lepszą jakość grupowania – o czym mówi metryka *silhouette score* – dla modelu BART efektywniejsze okazało się zastosowanie mnożnika i zasugerowanie metodzie wykorzystania czterech zamiast dwóch klastrów.

Na podstawie otrzymanych wyników można stwierdzić, że grupowanie z użyciem transferu wiedzy z modeli głębokich pozwala na zastosowanie wiedzy ze zbiorów internetowych i przeniesienie jej na analizowany problem. Funkcja mnożnika modyfikująca funkcję straty pozwala ponadto na sugerowanie modelowi grupowania optymalnych wartości dla liczby klastrów. Dzięki temu mamy większą kontrolę nad wstępnym grupowaniem przy analizowaniu dużych zbiorów danych i możemy wykorzystać to również w innych dziedzinach. Podobnie jak analiza grupowania z modelami z innych dziedzin przy przetwarzaniu innych zbiorów danych pozostaje to polem do dalszych badań.

Bibliografia

[1] Duda R.O., Hart P.E., Stork D.G., Pattern classification, Wiley-Interscience, New York 2000.

[2] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I., *Attention Is All You Need*, arXiv, 2017, https://arxiv.org/abs/1706.03762 (dostęp: 14.11.2024).

[3] Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S., *End-to-End Object Detection with Transformers*, arXiv, 2020, https://arxiv.org/abs/2005.12872 (dostęp: 14.11.2024).

[4] Lewis M., Liu Y., Goyal N., Ghazvininejad M., Mohamed A., Levy O., Stoyanov V., Zettlemoyer L., *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation*, Translation, and Comprehension, arXiv, 2019, https://arxiv.org/abs/1910.13461 (dostęp: 14.11.2024).

[5] Devlin J., Chang M.W., Lee K., Toutanova K., *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv, 2018, https://arxiv.org/abs/1810.04805 (dostęp: 14.11.2024).

[6] Radford A., Narasimhan K., Salimans T., Sutskever I., *Improving language understanding by generative pre-training*, OpenAI, 2018, https://openai.com/index/language-unsupervised/ (dostęp: 14.11.2024).

[7] Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V., *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, arXiv, 2019, https://arxiv.org/abs/1907.11692 (dostęp: 14.11.2024).

[8] Dai Z., Yang Z., Yang Y., Carbonell J., Le Q.V., Salakhutdinov R., *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*, arXiv, 2019, https://ar5iv.labs.arxiv.org/html/1901.02860 (dostęp: 14.11.2024).

[9] Kudo K., *Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates*, arXiv, 2018, https://arxiv.org/abs/1804.10959 (dostęp: 14.11.2024).

[10] Hendrycks D., Gimpel K., *Gaussian Error Linear Units (GELUs)*, arXiv, 2016, https://arxiv.org/abs/1606.08415 (dostęp: 14.11.2024).

[11] Dreamstime, *The Best Stock Images, Photos & Illustrations by Dreamstime*, 2020, https://www.dreamstime.com/photos-images/2020.html (dostęp: 14.11.2024).

Comparison of selected Deep Natural Language Processing Techniques in the context of unsupervised machine learning

Summary: In recent years, natural language processing has been one of the fastest growing areas of deep learning. The main focus of research has been on the generative aspects of this topic, while unsupervised learning techniques can rely on the knowledge encoded by these techniques with increasing efficiency. Grouping images based on their descriptions can be realized based on the knowledge learned by text models proposing classes in an adequate way classes and at the same time being a source for better understanding of the data. The research analyzed several deep learning models in various natural language processing (NLP) tasks. The most widely used models such as RoBERT or BART were selected and then used in the context of agglomerative clustering. Techniques for evaluating the quality of clustering were then applied – both parametric and subjective expert opinions suggesting the number of classes in the original data. Such a comparison makes it possible to analyze the impact of tasks and complexity of descriptions considered similar and separating them from each other. It also leaves the use of individual elements of large neural networks for further analysis and the impact of their selection on the quality of clustering.

Keywords: unsupervised learning, natural language processing, agglomerative clustering

Rozdział VI
System do rozpoznawania gatunków muzycznych

Szymon Górski, Anna Łupińska-Dubicka
Wydział Informatyki, Politechnika Białostocka

Streszczenie: Głównymi celami tych badań były utworzenie bazującego na sztucznej inteligencji systemu rozpoznawania gatunków muzycznych oraz ewaluacja jego dokładności. Na początku objaśniono koncept gatunku muzycznego, definiując go w kontekście informatycznym, oraz wskazano sygnał dźwiękowy jako medium łączące teorię muzyki z informatyką. Następnie opisano stan wiedzy dotyczący systemów identyfikacji gatunkowej, narzędzi, zbiorów danych i rozwiązań możliwych do wykorzystania w tworzonym systemie. Wybrano bazę GTZAN jako główny zbiór danych oraz konwolucyjne sieci neuronowe do modelu predykcji. Do implementacji systemu użyto Pythona, Librosa i TensorFlow. Celem opracowania było osiągnięcie predykcji gatunków muzycznych na poziomie równym lub lepszym niż pionierski artykuł George'a Tzanetakisa i Perry'ego Cooka z 2002 roku, który osiągnął skuteczność 61%. W eksperymencie bazowym uzyskano 61% dla zbioru walidacyjnego i 56% dla testowego. W celu poprawy wyników przeprowadzono dodatkowe eksperymenty z architekturą modelu i zbiorem danych. Zmiany hiperparametrów nie przyniosły pożądanych wzrostów, ale modyfikacje struktury modelu osiągnęły skuteczność 64% dla zbioru walidacyjnego i 61% dla testowego. Eksperymenty wykazały, że głównym problemem systemu był sam zbiór danych GTZAN. Po ponad 20 latach od jego wprowadzenia stwierdzono wady, takie jak słaba jakość próbek, niski bitrate, stratne metody nagrywania i błędy w tagowaniu.

Słowa kluczowe: rozpoznawanie gatunków muzycznych, konwolucyjne sieci neuronowe, sztuczna inteligencja

1. Wprowadzenie

Muzyka kształtowała i kształtuje różnorodne społeczności, stanowiła i stanowi ważną część rozwoju wielu cywilizacji, przewijała się i przewija przez różne epoki. Szeroka obecność muzyki w różnych kręgach pokazuje jedną z jej istotnych cech – ogromną rozpiętość. Potrzeba odnalezienia się w tejże podkreśliła konieczność porządkowania utworów oraz stworzenia kryteriów ich kategoryzacji. Jednym z nich są **gatunki muzyczne**, możliwe do opisania jako zbiory utworów o podobnych cechach, takich jak obecne konwencje, wykorzystywany styl, profil brzmieniowy oraz rytmiczny. Przez lata gatunki mocno

się zmieniały – ze sztywno zdefiniowanych zbiorów o niskiej zmienności do formy elastycznych struktur, pozwalających na łatwiejszą ewaluację konkretnych obiektów muzycznych (takich jak utwory, albumy, katalogi dzieł). Wraz z ich ewolucją pojawiła się pewna newralgiczna kwestia, spędzająca sen z powiek zarówno słuchaczom, jak i muzykom oraz badaczom: zacieranie się granic między gatunkami, prowadzące do rozważań na temat ich dalszej roli i istotności [1].

Rozmywanie się gatunkowych granic znacząco utrudnia kategoryzację utworów, co w konsekwencji prowadzi do problemów w określaniu ich cech, a co za tym idzie – wpływa negatywnie na rozpoznawanie gatunków. Sam proces kategoryzacji jest również czasochłonny – osiągnięcie satysfakcjonujących rezultatów przy dużej liczbie utworów oraz kompleksowości niuansów w nich zawartych bez pomocy zespołu ekspertów lub jakiejkolwiek automatyzacji (przetwarzania utworów, ekstrakcji cech z nich, obliczania rezultatów ich identyfikacji) okazuje się trudną i mozolną czynnością. Rozwiązaniem pozwalającym na eliminację wspomnianych przeszkód jest wykorzystanie wspomnianej automatyzacji procesu predykcji gatunków w kompozycjach poprzez utworzenie **systemów identyfikacji gatunkowej** bazujących na przetwarzaniu sygnałów i uczeniu maszynowym.

Osiągnięcie zadowalających rezultatów działania systemu opartego na powyższych elementach nie jest jednak trywialne – wymaga rozwiązania problemów, wśród których możemy wyszczególnić: wybór metody klasyfikacji, skonstruowanie skutecznego modelu, znalezienie odpowiednio opisanych baz utworów. Nie są to jedyne bariery możliwe do napotkania podczas rozwijania systemów rozpoznawania gatunków muzycznych, lecz ich pokonanie jest niezbędne do stworzenia systemu uzyskującego wyniki na równym lub lepszym poziomie niż w przypadku istniejących systemów tego typu. Opisana problematyka stanowi trzon tego artykułu, mającego na celu zbadanie zagadnień potrzebnych do identyfikacji gatunków muzycznych, zautomatyzowanie rozwiązania i utworzenie rozwiązania korzystającego z autorskiego modelu o skuteczności porównywalnej z już istniejącymi.

2. Gatunek muzyczny i jego reprezentacja informatyczna

Pochylenie się nad zagadnieniem rozpoznawania gatunków muzycznych wymaga zdefiniowania i objaśnienia pojęcia samego gatunku muzycznego. Włoski muzyk i muzykolog Franco Fabbri definiuje **gatunek muzyczny** na dwa sposoby: pierwsza z przedstawionych przez niego definicji określa gatunek muzyczny jako zbiór charakterystycznych muzycznych zdarzeń określany względem całego skończonego zbioru akceptowalnych społecznie zasad w muzyce [2], natomiast kolejna opisuje gatunek muzyczny jako rodzaj muzyki uznawany przez społeczność na podstawie konkretnego powodu, celu lub zestawu kryteriów muzycznych [3]. Warto podkreślić, że wspomniany badacz wyjaśnia **zdarzenie muzyczne** jako jakąkolwiek wykonywaną czynność skutkującą uzyskaniem dźwięku.

Powyższe definicje pomagają wyklarować znaczenie gatunku muzycznego w kontekście czysto muzycznym, lecz w przypadku przecięcia muzyki i informatyki przedstawione opisy mogą się okazać niewystarczające do utworzenia satysfakcjonującej definicji zgodnej ze specyfiką każdej z obu dziedzin. Reprezentacja utworów muzycznych w postaci uznanej przez muzykologów (np. zestawu konceptów lub zapisu nutowego) nie stanowi formy możliwej do bezpośredniego odczytania i wykorzystania przez komputer, co rodzi konieczność znalezienia formy pozwalającej na odczytywanie danych z nich przez zasoby obliczeniowe – taką reprezentację stanowi **sygnał dźwiękowy**.

Wymienione wcześniej pojęcie zdarzenia muzycznego można powiązać z sygnałem audio – przy wykonywaniu utworu otrzymywany jest zawierający w sobie pewną ilość zdarzeń muzycznych dźwięk, który po zarejestrowaniu może być przechowywany w formie pliku dźwiękowego, mającego takie cechy jak: barwa i wysokość dźwięków, struktura rytmiczna próbki, charakterystyka częstotliwościowa całości. Pozwalają one na wyszczególnienie charakterystycznych właściwości utworu oraz późniejsze porównanie tychże z innymi utworami. Prezentowana zależność została zauważona przez dwóch badaczy działających na pograniczu muzyczno-informatycznym – George'a Tzanetakisa i Perry'ego Cooka, którzy opisali ją w swej pracy z 2002 r., dotyczącej pozyskiwania informacji z sygnałów dźwiękowych w celach klasyfikacyjnych [4]. Przedstawione aspekty zawarte w sygnale audio można interpretować następująco:

- różne barwy i wysokości dźwięków mogą wskazywać na obecność brzmień bądź instrumentów typowych dla danych gatunków;
- konkretne struktury rytmiczne pozwalają określić intensywność instrumentów perkusyjnych właściwą konkretnym gatunkom;
- wyróżniające się charakterystyki częstotliwościowe ułatwiają późniejszą identyfikację wybranych gatunków.

Zarówno barwy i wysokości dźwięków, jak i charakterystyki częstotliwościowe mogą być przedstawione za pomocą **spektrogramów**, natomiast informacje o strukturach rytmicznych mogą przechowywać **histogramy**. Spektrogramy pomagają zobrazować zmiany częstotliwościowe w sygnale dźwiękowym względem czasu, histogramy zaś przedstawiają całościowy rozkład analizowanych danych – w tym przypadku obecność konkretnych częstotliwości z zakresu przypisywanego instrumentom perkusyjnym. Na podstawie wymienionych reprezentacji atrybutów sygnału audio możliwe jest rozróżnienie konkretnych gatunków muzycznych od siebie.

3. Rozpoznawanie gatunków muzycznych – stan wiedzy

Identyfikacja gatunków muzycznych z wykorzystaniem zasobów obliczeniowych jest rozległym zagadnieniem wymagającym zapoznania się z wiedzą dotyczącą nie tylko metod i kryteriów klasyfikacji, lecz także wykorzystywanych do tego celu zestawów

danych i technologii. Pojęcia niezbędne do poznania w tym kontekście to: narzędzia klasyfikacji, modele rozpoznawania gatunków, zestawy danych porównawczych i wykorzystywane technologie.

W ramach systemów klasyfikacji gatunków muzycznych wykorzystywane są narzędzia pozwalające na wyłuskiwanie żądanych informacji z plików dźwiękowych w celu uzyskania danych użytecznych przy grupowaniu utworów względem samych gatunków. Wśród tychże możemy wyróżnić trzy typy narzędzi najczęściej implementowanych w podobnych systemach: **spektrogramy** (przedstawiające zmiany częstotliwości w czasie), **parametry mel-cepstralne** (znane jako MFCC, obrazujące wspomniane zmiany częstotliwości w sposób podobny do działania percepcji słuchowej człowieka) oraz **histogramy** (ukazujące rozkład częstotliwości należących do konkretnych instrumentów w danych utworach).

Identyfikacja gatunków w utworach muzycznych wymaga korzystania z modeli uczenia maszynowego (naśladujących działanie ludzkiego umysłu z zakresu uczenia się) do zrealizowania dwóch żądanych akcji: poinstruowania systemu o tym, czym jest gatunek (tu: zbiorem podobnych wzorców lub cech w kontekście sygnału dźwiękowego), oraz dokonania predykcji gatunku przez system (na podstawie sygnału dźwiękowego przetworzonego do postaci obsługiwanej przez dany system). Modelami najczęściej używanymi do identyfikacji są **sieci neuronowe** (będące warstwowymi strukturami stosującymi sztuczne neurony, przypominające działaniem te biologiczne) oraz **maszyny wektorów nośnych** (stanowiące rozwiązanie z dziedziny uczenia nadzorowanego, wykorzystujące hiperpłaszczyzny do podziału danych na klasy).

Przy rozpoznawaniu gatunków muzycznych konieczne jest używanie zbiorów danych składających się z odpowiednio otagowanych utworów muzycznych – z ich pomocą realizowane są procesy: uczenia modeli identyfikacyjnych oraz klasyfikacji gatunkowej utworu. Wśród zestawów danych muzycznych użytecznych przy wymienionych procesach można wyróżnić m.in. **GTZAN** (będący jednym z najpopularniejszych tego typu zbiorów) [5], **MagnaTagATune** (bazujący na utworach wytwórni Magnatune oraz tagach zebranych przez aplikację TagATune) [6] oraz **Million Song Dataset** (składający się z informacji na temat cech miliona utworów popularnych w czasie jego tworzenia) [7].

Projektowanie i implementacja systemu rozróżniającego gatunki muzyczne stawiają wymóg wyboru odpowiednich technologii zarówno w zakresie stosowanego języka programowania, jak i bibliotek ułatwiających rozwijanie zbliżonych systemów. Wśród języków programowania wybieranych do podobnych rozwiązań można wyróżnić **Python** (z szeroko rozwiniętym ekosystemem rozwiązań zorientowanych wokół sztucznej inteligencji) oraz **C++** (często stanowiący bazę dla wspomnianych rozwiązań, dający szerokie możliwości optymalizacji i zarządzania pamięcią). Z kolei opcjami najczęściej wybieranymi spośród bibliotek są **Librosa** (tworzona przez zespół pod kierownictwem Briana McFee jako biblioteka języka Python) [8] oraz **Essentia** (rozwijana przez grupę badaczy z Uniwersytetu Pompeu Fabra w Barcelonie jako biblioteka języka C++, z możliwością korzystania z niej w języku Python) [9].

4. Prezentacja utworzonego systemu oraz autorskiego modelu klasyfikacji

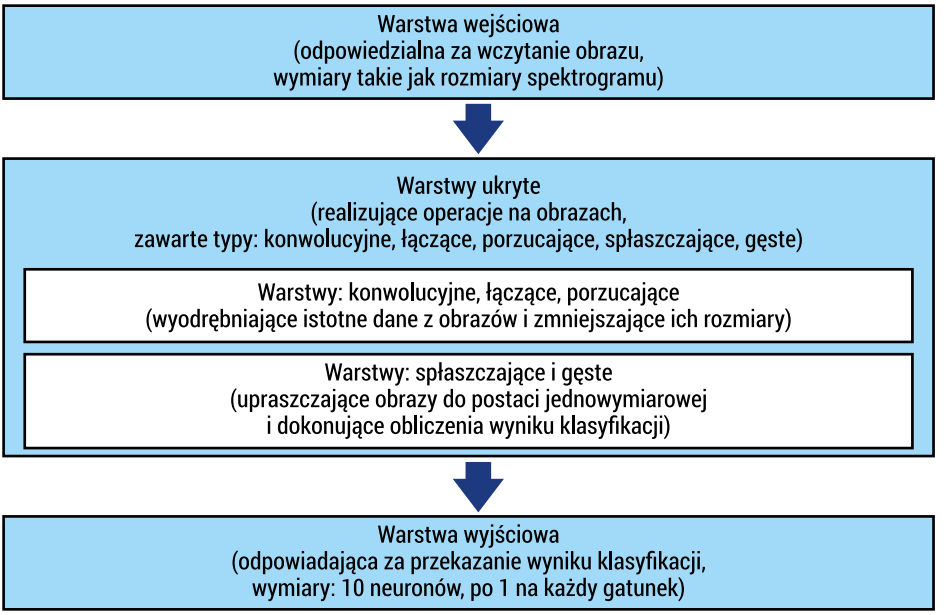
Stworzenie i wdrożenie systemu klasyfikującego gatunkowo treści muzyczne wymaga pochylenia się nad kilkoma zagadnieniami koniecznymi do utworzenia skutecznie funkcjonującego rozwiązania – prócz wybrania technologii stanowiących jego bazę uwagi wymagają także wcześniejsze zaprojektowanie architektury zarówno modelu pod kątem trafności predykcji, jak i aplikacji pod względem szybkości działania oraz zużycia zasobów. Każda decyzja podjęta podczas tworzenia tego typu systemów przekłada się znacząco na jakość ich funkcjonowania i uzyskiwane przez nie wartości parametrów, takich jak: dokładność i strata w procesie uczenia modelu, skuteczność predykcji gatunkowej, szybkość działania systemu.

Przy projektowaniu modelu stanowiącego bazę tworzonego na potrzeby tego artykułu rozwiązania istotnym zagadnieniem stało się przeanalizowanie różnych typów modeli w celu wybrania możliwie najlepszej podstawy rozwijanego systemu. Oparto się głównie na dwóch czynnikach: **elastyczności modyfikowania modelu** (zarówno podczas etapu projektowania, jak i implementacji, co pozwala na efektywne testowanie oraz szybkie wdrażanie zmian) i **skuteczności rozpoznawania gatunków** (podjętym już na początku prac założeniem było osiągnięcie wyników równych lub lepszych od rezultatów podanych w artykule Tzanetakisa i Cooka – wartość ta wynosi 61% [5]). Wstępne rozeznanie w temacie (wraz z początkową selekcją) pozwoliło na wyodrębnienie spośród wielu dostępnych wyjść trzech głównych opcji możliwych do zastosowania w ramach systemu, wśród których uwzględniono: maszyny wektorów nośnych (SVM, z ang. *support vector machines*), standardowe sieci neuronowe (ANN, z ang. *artificial neural networks*) i konwolucyjne sieci neuronowe (CNN, z ang. *convolutional neural networks*).

O ostatecznym wyborze metody zadecydowały chęć uzyskania zadowalających rezultatów w dość krótkim okresie i skorzystanie z dostępnego lokalnie sprzętu zawierającego dedykowane GPU. Maszyny wektorów nośnych są metodą przydatną podczas początkowych badań, przy których obciążenie nie stanowi większego problemu, lecz w ich dalszych, czasowo krytycznych etapach nie zapewniłaby ona oczekiwanej elastyczności, efektywności i wyników. Klasyczne sieci neuronowe dostarczają szerszego zaplecza technicznego i większej mobilności podczas rozwijania, lecz nie są w stanie wykorzystać pełni dostępnych zasobów obliczeniowych (głównie procesora graficznego). Konwolucyjne sieci neuronowe mogą zapewnić zarówno wysoką sprawność podczas funkcjonowania, jak i sporą plastyczność przy formowaniu ich architektury oraz dostrajaniu ich parametrów – dzięki tym cechom zostały one wybrane jako zwycięskie wyjście, pełniące funkcje bijącego serca projektu (analizującego utwory i świadczącego predykcje).

Model bazujący na wyselekcjonowanej opcji przyjmuje na wejście spektrogramy zawarte w ramach zbioru danych GTZAN (utworzone na podstawie udostępnionych w bazie 30-sekundowych urywków utworów). Następnie są one konwertowane

do formy obrazów – na ich podstawie możliwy jest nie tylko trening modelu, lecz także późniejsza klasyfikacja próbek. Wewnątrz niego są warstwy ukryte – odpowiadają one m.in. za realizowanie istotnych operacji obróbki obrazów, takich jak **konwolucja** (ekstrakcja istotnych danych z obrazu za pomocą filtrów) i **łączenie** (zmniejszanie rozmiaru obrazu celem uproszczenia jego dalszego przetwarzania) oraz za ich ostateczną predykcję. Wykonanie wszystkich czynności wieńczy przekazanie rezultatu klasyfikacji na warstwę wyjściową, informującą użytkownika o otrzymanym wyniku za pośrednictwem 10 neuronów – każdy z nich reprezentuje konkretny gatunek zadeklarowany w wykorzystywanym zestawie danych.

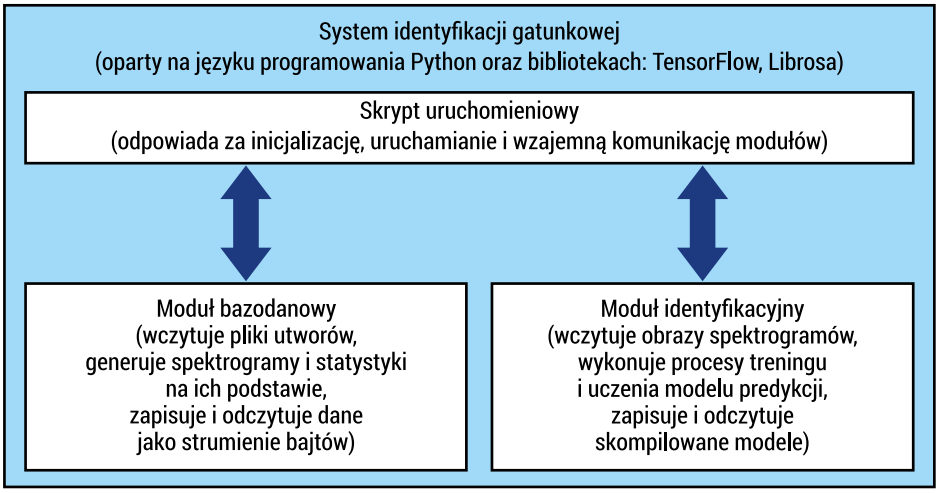


RYS. 1. Uproszczona architektura autorskiego modelu

ŹRÓDŁO: opracowanie własne.

Utworzony na potrzeby tej pracy system identyfikacji gatunkowej składa się z dwóch głównych modułów: **części bazodanowej**, obsługującej ładowanie i przetwarzanie bazy wykorzystywanej w projekcie, oraz **części identyfikacyjnej**, zajmującej się zarówno inicjalizacją, uczeniem i wizualizacją zaprojektowanych modeli, jak i samą identyfikacją gatunkową wraz z generowaniem rezultatów i wykresów je objaśniających. Pierwszy ze składników projektu wykorzystuje do swoich celów bibliotekę **Librosa**. Za jej pomocą otwiera pliki audio ze zbiorów, na ich podstawie generuje spektrogramy oraz oblicza dotyczące je dane statystyczne. Drugi z komponentów korzysta z biblioteki **TensorFlow**, dzięki której przygotowuje wcześniej zaprojektowany model do działania, sporządza macierz pomyłek obrazujących różnice w skuteczności predykcji, wyszukuje odpowiednie hiperparametry do doskonalenia

skuteczności predykcji. Obie części systemu mogą funkcjonować niezależnie od siebie, natomiast komunikacja między nimi zachodzi w ramach skryptu uruchomieniowego inicjującego system. Z jego poziomu określane są również parametry działania opisanych segmentów rozwiązania wraz z wywołaniami istniejących w nich metod. W ramach tego systemu możliwe są także dodatkowe operacje, jak np. mierzenie czasu działania programu przy wyborze danego wariantu modelu.



RYS. 2. Uproszczona architektura utworzonego systemu

ŹRÓDŁO: opracowanie własne.

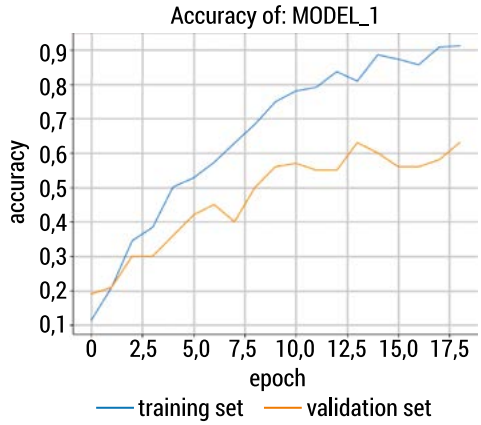
5. Eksperymenty na modelu – analiza i ewaluacja

Na potrzeby artykułu przeprowadzono eksperymenty zarówno na bazowej wersji modelu, jak i na jego zmodyfikowanych wariantach (głównie względem hiperparametrów i architektury modelu oraz proporcji podzbiorów przy zastosowanym zbiorze danych). Bazowy wariant modelu składa się z następujących części:

- warstwy wejściowej, przyjmującej na wejście spektrogramy przeskalowane do akceptowanej przez system rozdzielczości 128 × 660 (będącej dwukrotnym pomniejszeniem rozmiarów największego spektrogramu z bazy);
- trzech par warstw ukrytych, z których każda para zawiera po jednej warstwie konwolucyjnej (wyodrębniającej ze spektrogramu istotne cechy danego sygnału) i łączącej (zmniejszającej obraz poprzez wybranie z okna pikseli żądanej wartości, tu: największej z nich);
- kolejnej pary tych warstw – spłaszczającej (przekształcającej otrzymany tensor do postaci jednowymiarowego wektora) i gęstej (transformującej uzyskany wektor do znacząco pomniejszonej postaci);

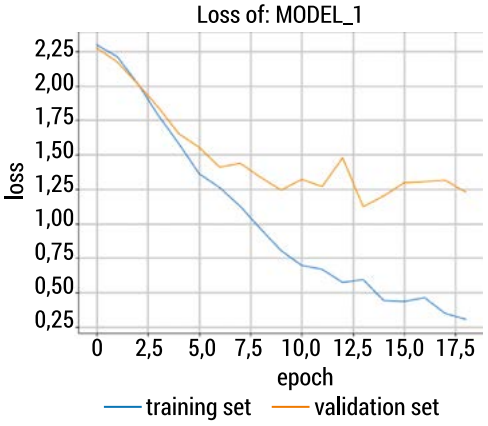
- warstwy wyjściowej, zwracającej ostateczny rezultat klasyfikacji spektrogramu względem gatunku za pośrednictwem 10 neuronów wyjściowych (po jednym na każdą klasę) w formie wartości binarnych.

W przypadku podstawowego wariantu skuteczność treningu autorskiego rozwiązania prezentowała się na poziomie 63% dla zbioru walidacyjnego oraz 58% dla zbioru testowego, natomiast strata osiągała wartości 1, 2297 przy wykorzystaniu zbioru walidacyjnego i 1, 2857 przy wykorzystaniu zbioru testowego.



RYS. 3. Zmienność wartości skuteczności w ramach podstawowego wariantu autorskiego modelu rozpoznawania gatunków muzycznych

ŹRÓDŁO: opracowanie własne.



RYS. 4. Zmienność wartości straty w ramach podstawowego wariantu autorskiego modelu rozpoznawania gatunków muzycznych

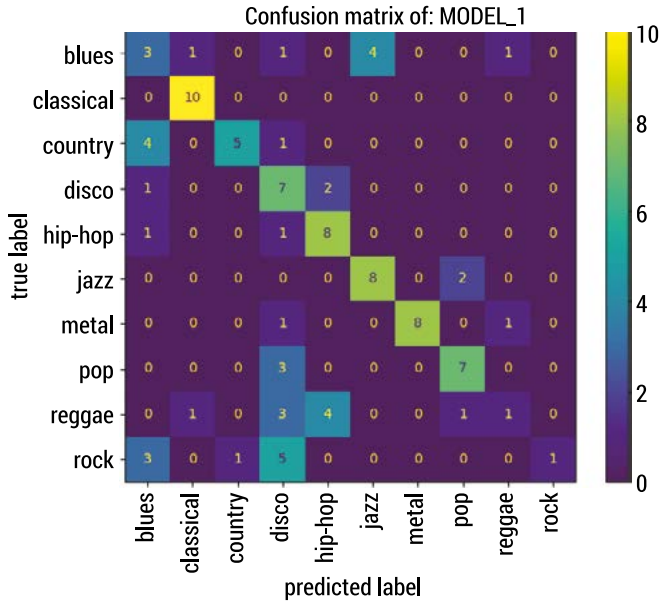
ŹRÓDŁO: opracowanie własne.

TABELA 1. Wartości skuteczności i straty w ramach podstawowego wariantu autorskiego modelu rozpoznawania gatunków muzycznych

Parametr	Zbiór walidacyjny	Zbiór testowy
Skuteczność	63%	58%
Strata	1,2297	1,2857

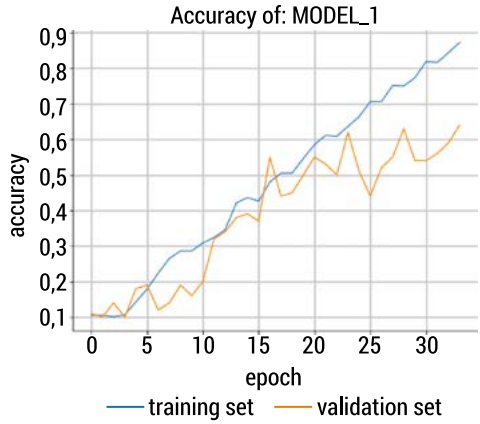
ŹRÓDŁO: opracowanie własne.

Otrzymana w ramach eksperymentu macierz pomyłek wskazywała na utrudnienia związane z rozpoznawaniem gatunków o mniej wyrazistych sygnaturach częstotliwościowych. Działający model radził sobie świetnie z muzyką poważną (mającą dość mało elementów perkusyjnych), metalem, hip-hopem (silnie opierającymi się na intensywnych sekcjach perkusyjnych) oraz jazzem (mieszającym intensywność gry w sekcji perkusyjnej), ale rozpoznanie przez niego innych gatunków stanowiło sporą trudność. Gatunki o podobnych sygnaturach często nie były poprawnie identyfikowane przez model, zwłaszcza w przypadku bluesa (często mylnie rozpoznawanego jako jazz lub rock), reggae (mocno mylonego z disco, hip-hopem oraz jazzem) i rocka (klasyfikowanego jako blues lub disco).



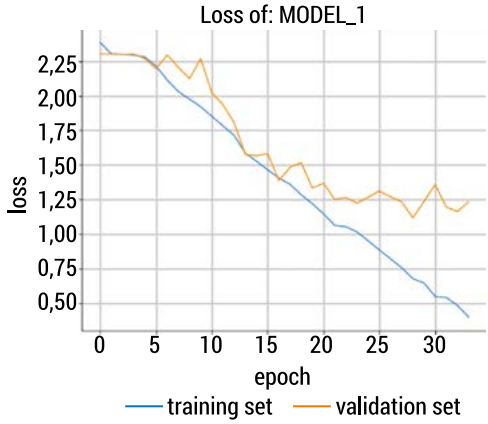
RYS. 5. Macierz pomyłek podstawowego wariantu autorskiego modelu rozpoznawania gatunków muzycznych

ŹRÓDŁO: opracowanie własne.



RYS. 6. Zmienność wartości skuteczności w ramach wariantu modelu o zmodyfikowanych warstwach gęstych

ŹRÓDŁO: opracowanie własne.



RYS. 7. Zmienność wartości straty w ramach wariantu modelu o zmodyfikowanych warstwach gęstych

ŹRÓDŁO: opracowanie własne.

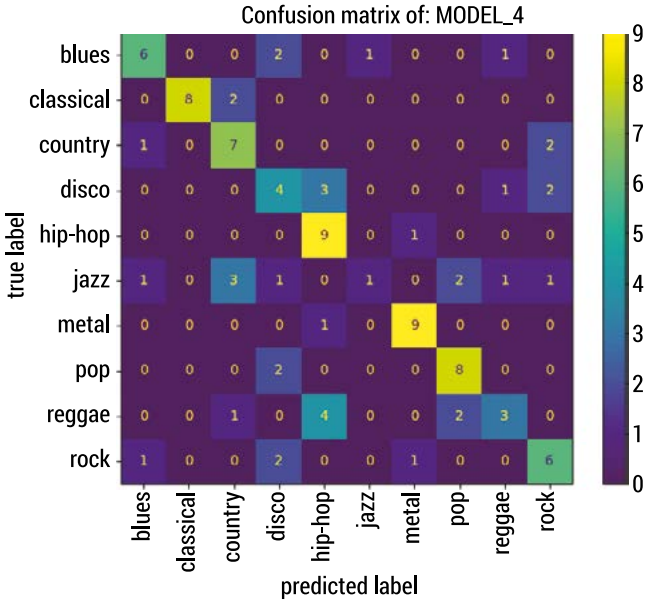
Po eksperymencie bazowym wykonano szereg badań na zmodyfikowanych wersjach modelu. Na początku zrealizowano eksperymenty na wariancie zmodyfikowanym względem wartości hiperparametrów (czyli zmiennych wykorzystywanych w procesie uczenia, takich jak liczba filtrów w warstwach konwolucyjnych czy liczba neuronów w poszczególnych warstwach gęstych) – zauważono, że nie tylko nie przyniosły one oczekiwanej poprawy wyników, lecz także doprowadziły do wystąpienia zjawiska overfittingu, świadczącego o nadmiernym dopasowaniu modelu względem danych treningowych. Później wykonano eksperymenty względem wariantu

o zmienionych proporcjach podzbiorów w bazie próbek (tu: GTZAN) – zmiany te znacząco pogorszyły skuteczność predykcji w porównaniu z bazowym modelem, sprowadzając ją do poziomu 40% w ramach wariantu o proporcjach: 50% – zbiór treningowy, 25% – zbiór testowy, 25% – zbiór walidacyjny. Na końcu przeprowadzono eksperymenty względem wariantu modelu o zmodyfikowanej architekturze i to właśnie one doprowadziły do oczekiwanej poprawy, przynosząc rezultaty lepsze o kilka punktów procentowych względem eksperymentu bazowego. Krokiem pozwalającym na uzyskanie lepszych wyników była ingerencja w warstwy gęste modelu.

TABELA 2. Wartości skuteczności i straty w ramach podstawowego wariantu autorskiego modelu rozpoznawania gatunków muzycznych

Parametr	Zbiór walidacyjny	Zbiór testowy
Skuteczność	64%	61%
Strata	1,2348	1,2408

ŹRÓDŁO: opracowanie własne.



RYS. 8. Macierz pomyłek podstawowego wariantu autorskiego modelu rozpoznawania gatunków muzycznych

ŹRÓDŁO: opracowanie własne.

6. Wnioski

Utworzenie pełnoprawnego systemu rozpoznawania gatunków muzycznych nie jest procesem łatwym, często wymaga podjęcia kroków idących w różnych kierunkach – poczynając od zrozumienia, co definiuje i cechuje gatunek jako pojęcie, na sposobach poprawienia rezultatów uczenia kończąc. Podczas podsumowywania tych kroków należy szczególnie pochylić się nad kilkoma z nich: doborem zbioru danych, procesem projektowania modelu oraz zakresem rozpoznawanych gatunków.

Zastosowany na potrzeby pracy zbiór GTZAN cechowały niski próg wejścia wynikający z małych rozmiarów (1000 próbek dla 10 gatunków, po 100 na każdy z nich) oraz prostota otagowania (każda próbka zawierała tylko jeden tag) – cechy te zdecydowały o wyborze tej bazy jako zbioru danych w ramach systemu. Niestety liczba wad odkrytych we wspomnianym zbiorze podczas dalszego rozwoju systemu (w tym błędy w tagowaniu sampli i anomalie jakościowe w próbkach) wpłynęła negatywnie na finalną skuteczność predykcji. Po głębszym przebadaniu okazało się, iż jest to kwestia trapiąca większość systemów predykcji gatunkowej bazujących na GTZAN [10].

Podobnie jak w przypadku omówionego przed chwilą zbioru, samodzielne rozwijanie modelu rozpoznawania gatunków muzycznych miało dość problematyczną naturę. Utworzenie sprawnie działającej architektury na własną rękę było znacznie bardziej interesującym wyzwaniem niż skorzystanie z działających już struktur, lecz prowadziło do częstego natrafiania na blokady wymagające ponadprogramowych zmian (takie jak np. nagłe obniżenie skuteczności uczenia do poziomu kilku punktów procentowych, wynikające z błędów w obsłudze próbek przez model lub nieprawidłowego ułożenia warstw ukrytych). Znacząco wydłużyło to czas potrzebny na finalizację tworzenia systemu.

Również zakres rozpoznawanych gatunków muzycznych dość znacznie może ograniczać system w przyszłych zastosowaniach. Liczba rozpoznawanych gatunków jest stosunkowo niewielka, zaledwie 10, i o bardzo szerokiej rozpiętości. Dość często mają one cechy wspólne (takie jak obecność intensywnych brzmień perkusyjnych w przypadku rocka i metalu czy też wysokie natężenie częstotliwości basowych w ramach hip-hopu i reggae), które w konsekwencji mogą prowadzić do pomyłek w predykcji, co było zauważalne podczas przeprowadzania eksperymentów względem systemu.

Mimo przedstawionych problemów spełniono pierwotnie postawione założenia dotyczące skuteczności identyfikacji gatunkowej – utworzony system miał skuteczność na poziomie zbliżonym do tego opisanego w pracy Tzanetakisa i Cooka. Wartość ta dla autorskiego systemu wyniosła 64%, co jest wynikiem o 3 punkty procentowe większym w porównaniu z rozwiązaniem stworzonym przez powyższych badaczy (w ich przypadku 61%).

Bibliografia

[1] Silver D., Lee M., Childress C.C., *Genre complexes in popular music*, „PLoS ONE" 2016, vol. 11, no. 5, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155471 (dostęp: 14.11.2024).

[2] Fabbri F., *A Theory of Musical Genres: Two Applications*, Popular Music Perspectives: Papers from the First International Conference on Popular Music Research, June 1981, https://www.tagg.org/xpdfs/ffabbri81a.pdf (dostęp: 14.11.2024).

[3] Fabbri F., *Browsing Music Spaces: Categories And The Musical Mind*, 1999 IASPM-UK/Ireland Branch Conference, 1999, https://www.tagg.org/xpdfs/ffabbri990717.pdf (dostęp: 14.11.2024).

[4] Tzanetakis G., Cook P., *Musical genre classification of audio signals*, „IEEE Transactions on Speech and Audio Processing" 2002, vol. 10, https://ieeexplore.ieee.org/document/1021072 (dostęp: 14.11.2024).

[5] Tzanetakis G., Website for GTZAN, Tzanetakis' dataset for music information retrieval purposes, Aug. 2021, https://web.archive.org/web/20220330025414/marsyas.info/downloads/datasets.html (dostęp: 14.11.2024).

[6] Lamere P., *Magnatagatune – a new research data set for MIR*, 2009, https://musicmachinery.com/tag/magnatagatune/ (dostęp: 14.11.2024).

[7] Bertin-Mahieux T., Ellis D.P., Whitman B., Lamere P., *The Million Song Dataset*, „Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)" 2011, vol. 1, https://www.ee.columbia.edu/ dpwe/pubs/BertEWL11-msd.pdf (dostęp: 14.11.2024).

[8] McFee B., Raffel C., Liang D., Ellis D.P., McVicar M., Battenberg E., Nieto O., *librosa: Audio and Music Signal Analysis in Python*, „Proceedings of the 14th Python In Science Conference" 2015, vol. 1, s. 18–25, https://conference.scipy.org/proceedings/scipy2015/pdfs/brian mcfee.pdf (dostęp: 14.11.2024).

[9] Bogdanov D., Wack N., Gomez E., Gulati S., Herrera P., Mayor O., Roma G., Salamon J., Zapata J., Serra X., *Essentia: an Audio Analysis Library for Music Information Retrieval*, „International Society for Music Information Retrieval Conference (ISMIR'13)" 2013, vol. 1, s. 493–498, https://repositori.upf.edu/bitstream/handle/10230/32252/essentia_ismir_2013.pdf?sequence=1isAllowed=y (dostęp: 14.11.2024).

[10] Sturm B.L., *The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use*, arXiv.org, 2013, https://arxiv.org/abs/1306.1461 (dostęp: 14.11.2024).

Music Genre Recognition System

Summary: The main objectives of this study were to create an artificial intelligence-based system for recognizing music genres and to evaluate its accuracy. Initially, the concept of a music genre was explained, defining it in an informational context and identifying the audio signal as the medium connecting music theory with informatics. Next, the current state of knowledge regarding genre identification systems, tools, datasets, and solutions that could be utilized in the developed system was described. The GTZAN database was selected as the primary data set, and convolutional neural networks were chosen for the prediction model. Python, Librosa, and TensorFlow were used for system implementation. The study aimed to achieve music genre predictions at a level equal to or better than the pioneering article by George Tzanetakis and Perry Cook from 2002, which

achieved a classification accuracy of 61%. In the baseline experiment, 61% accuracy was obtained for the validation and 56% for the test set. Additional experiments were conducted with the model's architecture and the dataset to improve these results. Changes in hyperparameters did not bring the desired improvements, but modifications to the model structure achieved an accuracy of 64% for the validation set and 61% for the test set. The experiments showed that the system's main problem was the GTZAN dataset itself. Over 20 years after its introduction, defects such as poor sample quality, low bitrate, lossy recording methods, and tagging errors were identified.

Keywords: music genre recognition, convolutional neural networks, artificial intelligence

Rozdział VII
Metody wykrywania złośliwego oprogramowania na platformie Android z wykorzystaniem algorytmów uczenia maszynowego

Patryk Ostrowski
Wydział Cybernetyki, Wojskowa Akademia Techniczna

Streszczenie: Wykrywanie złośliwego oprogramowania jest jednym z najważniejszych wyzwań w nowoczesnych systemach operacyjnych. Rozpoznanie zagrożenia przed jego realizacją to kluczowa kwestia ochrony prywatności użytkownika. W najpopularniejszym systemie operacyjnym Android problem ochrony danych użytkownika jest szczególnie istotny. Aplikacje wykorzystywane w nowoczesnych systemach mobilnych pozwalają obecnie na zarządzanie wrażliwymi danymi, takimi jak dane bankowe, zdrowotne i wirtualne dokumenty. Dlatego stają się one celem dla atakujących. Liberalna polityka instalacji oprogramowania na platformie Android zachęca do zwiększenia ochrony przed nieznanymi złośliwymi aplikacjami. W niniejszym rozdziale omówiono przygotowanie oprogramowania, które chroni użytkownika końcowego przed utratą danych wrażliwych. Wykorzystując różne podejścia analizy aplikacji na platformę Androida, zbadano skuteczność poszczególnych algorytmów uczenia maszynowego. W trakcie badań zaproponowano metodę ekstrakcji danych z dużych zbiorów aplikacji mobilnych oraz określono wpływ wykorzystania różnych algorytmów selekcji zmiennych i optymalizacji hiperparametrów na skuteczność wykrywania złośliwego oprogramowania. W rezultacie uzyskano dane pozwalające określić skuteczność implementacji, a wyniki pomogły wyznaczyć kierunek dalszych działań.

Słowa kluczowe: Android, *malware*, uczenie maszynowe

1. Wprowadzenie

System Android jest najpopularniejszym mobilnym systemem operacyjnym na świecie [1]. W 2022 roku obsługiwał ponad 72% używanych przenośnych systemów, co daje 3,1 mld użytkowników. Na skutek ataków są oni narażeni na utratę prywatnych danych. W trzecim kwartale 2022 roku Kaspersky Lab wykrył 438 035 złośliwych pakietów instalacyjnych [2]. Firma ta przygotowała również raport na temat typów wykrytego złośliwego oprogramowania oraz ich skutków dla użytkowników

systemu operacyjnego. Złośliwa aplikacja dąży do uzyskania dostępu do urządzenia użytkownika, a następnie napastnik realizuje swe cele. Wśród nich mogą być: kradzież danych (np. wiadomości SMS, plików, numerów telefonów, zdjęć), wyświetlanie reklam, nakłanianie do działań niekorzystnych dla użytkownika (np. wysyłanie wiadomości płatnych lub zlecanie subskrypcji), kradzież tożsamości (np. z bankowej aplikacji mobilnej). Jest to poważne zagrożenie dla użytkowników korzystających z platformy Android. Według statystyk w oficjalnym sklepie Google Play w połowie 2022 roku było prawie 2,6 mln aplikacji [3]. To ogromna liczba pakietów instalacyjnych, które muszą zostać przetestowane przed udostępnieniem ich użytkownikowi. Ze względu na tak dużą liczbę pakietów stosowane są zautomatyzowane metody wykrywania zagrożeń, takie jak analizy statyczna i dynamiczna. Metody te są skuteczne, ale niedoskonałe, co skutkuje pojawianiem się niebezpiecznych aplikacji w oficjalnych sklepach dystrybutorów. W 2022 roku badacze Bitdefender zidentyfikowali 35 niebezpiecznych aplikacji w sklepie Google Play, z których każda została pobrana ponad 100 000 razy [4].

Kolejnym problemem jest możliwość instalowania aplikacji mobilnych spoza oficjalnego sklepu producenta systemu. Istnieje wiele sklepów zewnętrznych, które nie zapewniają porównywalnego poziomu ochrony posiadaczom telefonu. Ponadto użytkownicy systemu Android mogą zainstalować aplikację poprzez kliknięcie na pobrany pakiet instalacyjny, co jest niemożliwe np. na platformie IOS. Tę technikę wykorzystuje się często w przypadku ataków phishingowych, których celem jest skłonienie odbiorcy wiadomości, np. SMS lub e-mail, do zainstalowania złośliwej aplikacji. Napastnik w celu wzbudzenia zaufania podaje się wtedy za instytucję znaną i godną zaufania – bank czy firmę kurierską.

2. Bezpieczeństwo użytkownika w systemie Android

Platforma Android opiera się na jądrze systemu Linux, wykorzystując jego mechanizmy bezpieczeństwa na poziomie separacji działających aplikacji [5]. Dodatkowo w celu ograniczenia liczby zasobów dostępnych dla oprogramowania do minimum zostały wprowadzone pozwolenia. Definiują one, czy dana aplikacji może się odwołać do elementu systemu, takich jak aparat, pamięć zewnętrzna czy wiadomości SMS [6].

Aby chronić prywatność użytkowników, system Android wymaga od każdej aplikacji mobilnej zdefiniowania niezbędnych uprawnień. Dochodzi do tego w pliku manifestu aplikacji AndroidManifest.xml, który można odczytać m.in po dekompilacji pliku instalacyjnego. Głównym celem pozwolenia jest zmniejszenie powierzchni ataku. Dana instancja może uzyskać dostęp tylko do zasobów telefonu, które zostały zdefiniowane w manifeście. Podczas instalacji użytkownik jest informowany o wymaganych uprawnieniach, a w przypadku uprawnień, które mogą w sposób szczególny naruszać jego prywatność, tj. odczyt pamięci telefonu, odczyt wiadomości SMS, użytkownik musi wyrazić świadomą zgodę podczas działania programu. System Android ma trzy rodzaje predefiniowanych grup pozwoleń: uprawnienia nadawane podczas instalacji,

podczas działania aplikacji oraz uprawnienia specjalne [7]. Pierwsza grupa zawiera uprawnienia, które są bezpieczne dla użytkownika lub aplikacja pochodzi z zaufanej strony. System automatycznie przyznaje je aplikacji, gdy użytkownik ją instaluje. Ta grupa zawiera dwie podgrupy: uprawnienia normalne oraz wymagające podpisu dostawcy. Te pierwsze mają niewielki wpływ na prywatność użytkownika Androida, dlatego ryzyko wykorzystania tych uprawnień przez niebezpieczną aplikację mobilną jest niskie. Uprawnienia przydzielane podczas działania aplikacji są bardziej niebezpieczne. Dzięki nim aplikacja może wykorzystać zasoby, które mogą mieć negatywny wpływ na bezpieczeństwo użytkownika. Takie uprawnienie definiowane jest w dwóch miejscach – w manifeście aplikacji oraz w kodzie źródłowym. Jeśli aplikacja będzie chciała wykorzystać je po raz pierwszy, użytkownik zostanie o tym poinformowany i będzie musiał wyrazić świadomą zgodę [8].

3. Metody wykrywania złośliwego oprogramowania oparte na analizie statycznej

Statyczna analiza kodu jest jedną z najważniejszych metod wykrywania złośliwego oprogramowania. Specjalna implementacja pobiera jako dane wejściowe plik binarny (w tym przypadku pakiet instalacyjny Androida .apk) i bada go bez jego uruchamiania [9]. Zautomatyzowane oprogramowanie sprawdza zdekompilowany kod oraz inne zasoby aplikacji mobilnych, takie jak pliki konfiguracyjne, zmienne itd. W przypadku aplikacji mobilnej analizowany jest również jej manifest. Jest to plik xml, który zawiera podstawowe informacje o aplikacji i konfiguracjach ważnych dla systemu Android, tj. uprawnienia, podstawową aktywność, obsługiwane wersje systemów. Podczas analizy kod nie jest uruchamiany, dlatego proces ten wymaga mniej zasobów na komputerze, na którym się ją przeprowadza. Jednak ten rodzaj analizy nie ocenia zachowania uruchomionej aplikacji, jak w analizie dynamicznej. Ponadto napastnik może starać się ukryć prawdziwe działania aplikacji, stosując różne metody. W przypadku analizy statycznej jest to wykorzystanie kryptografii whitebox oraz obfuskacji [10].

W najbardziej podstawowym ujęciu analizy porównuje się aplikacje oraz jej elementy do znanych zagrożeń. Do tego celu wykorzystuje się sygnatury, które powstają w wyniku działania funkcji skrótu [11]. Problemem tej opcji jest możliwość znalezienia tylko znanych już wcześniej zagrożeń, dla których przygotowano sygnatury. Dlatego warto analizować zasoby i wyróżnić dwa podejścia: oparte na analizie wyłącznie przydzielonych uprawnień oraz wykorzystujące wszystkie dane zgromadzone w pakiecie instalacyjnym, takie jak kod aplikacji czy zasoby. W obu przypadkach często wykorzystywane są algorytmy uczenia maszynowego. Przykładem zastosowania pierwszego podejścia jest projekt Crowdroid [12]. W tym projekcie uprawnienia każdej aplikacji są mapowane na wektor z wartościami binarnymi (wartość 1 – gdy uprawnienie zostało użyte, a 0 w przeciwnym razie), a każdemu wektorowi przypisywana jest klasa określająca, czy aplikacja jest bezpieczna. Autorzy projektu wykorzystali algorytm centroidów do klasyfikacji aplikacji, a następnie przy użyciu metody drzew

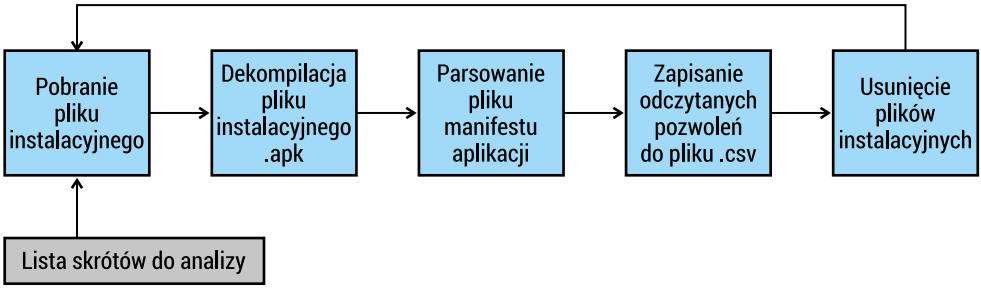
decyzyjnych i lasów losowych pakiety były analizowane, gdy klastry nakładały się na siebie. Eksperyment przeprowadzono na dwóch zestawach, większym z 500 wektorami i mniejszym z 200 wektorami, a wyniki scharakteryzowano w ocenie modelu. Podobne podejście przedstawiono w artykule *Lightweight Malware Detection based on Machine Learning Algorithms and the Android Manifest File* [13] – przeanalizowano w nim 1000 próbek aplikacji i porównano wyniki dla danych testowych, których zmienne objaśniające przedstawiały tylko uprawnienia przydzielane aplikacji. Przy zastosowaniu metody walidacji krzyżowej dokładność dla poszczególnych metod wynosiła: 82,6% – dyskryminatory liniowe, 91,4% – weighted KNN, 79% – KNN, 89,2% – Linear SVM i 89,3% dla drzewa decyzyjnego.

4. Badanie

Przygotowanie repozytoriów

W celu zbadania efektywności zastosowania algorytmów uczenia maszynowego wybrano repozytorium pakietów instalacyjnych, które następnie wykorzystano do przygotowania zbiorów uczących i testowych. W prowadzonych badaniach użyto repozytorium, które powstało w ramach projektu AndrooZoo [14]. W zasobie tym udostępnione zostały wyniki skanów 24 542 644 (23.06.2024) aplikacji mobilnych za pomocą aplikacji VirusTotal. Pliki źródłowe dostępne są do pobrania za pomocą określonego API. Do odpowiedniego zbioru danych zawierającego wyłącznie uprawnienia został dołączony skrypt (jego poglądowe działanie przedstawiono na rys. 1). Do jego zadań należą: pobranie pakietu instalacyjnego za pomocą Androzoo API, dekompilacja, parsowanie pliku manifestu i usunięcie paczki z dysku. Wszystkie odczytane wartości zostały zapisane w pliku csv.

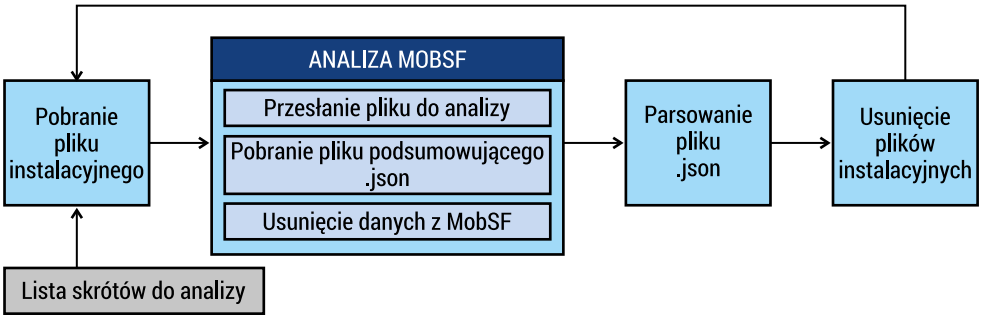
Ze względu na dużą liczbę pakietów aplikacji ważnymi elementami tego procesu były pobieranie i usuwanie aplikacji w czasie rzeczywistym.



RYS. 1. Analiza pozwoleń zadeklarowanych w pakietach instalacyjnych w celu przygotowania danych uczących

ŹRÓDŁO: opracowanie własne.

Na tym etapie badań wykorzystano jedynie uprawnienia. Drugi skrypt wyodrębnił więcej informacji za pomocą analizy statycznej. W tym celu wykorzystano oprogramowanie Mobile Security Framework. Analizuje ono pakiet poprzez analizę manifestu aplikacji, zdekompilowanego kodu źródłowego oraz innych zasobów, np. plików konfiguracyjnych (poglądowe działanie skryptu zostało przedstawione na rys. 2). Aby przyspieszyć analizę dużej liczby plików, na jednej maszynie wirtualnej uruchomiono pięć kontenerów Docker z oprogramowaniem MobSF. W pierwszym kroku skrypt pobierał pakiety instalacyjne z zasobów zdalnych. Następnie pakiet był wysyłany przez API do MobSF, gdzie był analizowany. Po tym procesie pobierany był plik json z wynikami skanowania. Po przeskanowaniu wszystkich plików kolejny skrypt analizował zebrane pliki json. Wynikiem tego działania był plik csv zawierający cechy aplikacji mobilnej. Potem wygenerowany zestaw uczący został wzbogacony o dodatkowe zmienne, dodano liczbę uprawnień, uprawnień neutralnych i uprawnień niebezpiecznych.



RYS. 2. Analiza pakietów instalacyjnych z wykorzystaniem oprogramowania MobSF w celu przygotowania danych uczących

ŹRÓDŁO: opracowanie własne.

Wykorzystane algorytmy uczenia maszynowego

K-najbliższych sąsiadów

K-Nearest Neighbors (KNN) to algorytm wykorzystywany w uczeniu maszynowym do zadań klasyfikacji i regresji [15]. Aby sklasyfikować nowy punkt danych, algorytm ten identyfikuje *k* najbliższych sąsiadów tego punktu w przestrzeni cech, obliczając odległości między nowym punktem a każdym punktem w treningowym zbiorze danych [16]. Typowe miary odległości obejmują odległość euklidesową oraz odległość Manhattan. Po obliczeniu odległości punkty uczące są sortowane w porządku rosnącym na podstawie ich odległości od nowego punktu. Wybieranych jest *k* punktów o najmniejszych odległościach. W przypadku zadań klasyfikacyjnych etykieta nowego punktu jest przypisywana poprzez głosowanie większościowe wśród *k* najbliższych sąsiadów. Etykieta, która pojawia się najczęściej wśród sąsiadów, jest przypisywana do nowego punktu.

Naiwny klasyfikator Bayesowski

Algorytm opiera się na podejściu biorącym pod uwagę oznaczony zbiór danych z wektorami cech $\mathbf{x}_i \in \mathbb{R}^d$ i odpowiadającymi im etykietami klas $y_i \in \mathscr{L}$L, gdzie $i = 1, 2, ..., n$, a n to całkowita liczba punktów danych [17]. Algorytm ten oblicza prawdopodobieństwo dla każdej etykiety klasy na podstawie występowania wartości cech. Zakłada on, że cechy są od siebie niezależne, co jest uproszczonym założeniem, ale często sprawdza się w praktyce. Podczas fazy uczenia algorytm szacuje wcześniejsze prawdopodobieństwa każdej klasy, zliczając częstotliwość występowania w zbiorze danych szkoleniowych. Szacuje również warunkowe prawdopodobieństwa wartości cech, biorąc pod uwagę etykiety klas. Aby sklasyfikować nowy punkt danych, algorytm oblicza prawdopodobieństwo następcze każdej etykiety klasy przy użyciu twierdzenia Bayesa i jako przewidywaną wybiera tę o najwyższym prawdopodobieństwie [18].

Drzewa decyzyjne

Drzewo decyzyjne to popularny algorytm wykorzystywany w uczeniu maszynowym do zadań klasyfikacji [19]. Buduje on drzewiasty model decyzji na podstawie wartości cech w celu przewidywania zmiennej docelowej. Algorytm stosuje podejście rekurencyjne, biorąc pod uwagę oznaczony zbiór danych z wektorami cech $\mathbf{x}_i \in \mathbb{R}^d$ i odpowiadającymi im etykietami y_i, gdzie $i = 1, 2, ..., n$, a n to całkowita liczba punktów danych. Algorytm drzewa decyzyjnego rozpoczyna się od wyboru najlepszej cechy do podziału danych na podstawie określonego kryterium, np. współczynnik Giniego. Jako cecha podziału wybierana jest ta, która zapewnia najbardziej znaczące zmniejszenie zanieczyszczenia lub maksymalizuje przyrost informacji [20]. Po jej określeniu zbiór danych, opierając się na możliwych wartościach tej cechy, jest dzielony na podzbiory. Proces ten jest powtarzany rekurencyjnie dla każdego podzbioru, tworząc strukturę przypominającą drzewo. Podział jest kontynuowany do momentu spełnienia kryterium zatrzymania, takiego jak osiągnięcie maksymalnej głębokości drzewa lub minimalnej liczby punktów danych w węźle liścia. W każdym węźle liścia klasa większościowa lub średnia wartość zmiennych docelowych w tym węźle jest przypisywana jako wartość przewidywana.

Lasy losowe

Algorytm lasów losowych działa poprzez tworzenie kolekcji drzew decyzyjnych. Każde drzewo jest konstruowane niezależnie przy użyciu losowego podzbioru danych szkoleniowych [21]. Ten losowy wybór pomaga wprowadzić różnorodność między drzewami. Podczas fazy uczenia każde drzewo decyzyjne buduje się przez rekurencyjny podział danych na podstawie wybranych cech. Podziały są określane przy użyciu algorytmów, takich jak ID3, CART lub C4.5, które znajdują najlepsze cechy i progi w celu rozdzielenia danych na różne klasy bądź przewidywane wartości ciągłych. Po skonstruowaniu drzew decyzyjnych prognozy przepuszczają nowy punkt danych przez każde drzewo. W przypadku zadań klasyfikacji etykieta klasy przewidywana przez każde drzewo jest liczona przez głosowanie, a najczęstsza staje się ostateczną prognozą. Lasy losowe znane są ze swojej odporności

i zdolności do dobrego uogólniania, co czyni je popularnym wyborem w zastosowaniach uczenia maszynowego [22].

Ekstremalne lasy losowe

Algorytm ekstremalnych lasów losowych działa poprzez tworzenie kolekcji drzew decyzyjnych. Każde drzewo jest budowane przy użyciu losowego podzbioru danych treningowych i losowego podzbioru cech. Podczas fazy uczenia każde drzewo decyzyjne jest konstruowane przez rekurencyjny podział danych wykorzystujący losowo wybrane cechy. Zamiast oceniać wiele potencjalnych podziałów i wybierać najlepszy z nich, algorytm losowo wybiera podział bez uwzględnienia jego jakości. Po skonstruowaniu drzew decyzyjnych prognozy przepuszczają nowy punkt danych przez każde drzewo. W przypadku zadań klasyfikacji liczona jest etykieta klasy przewidywana przez każde drzewo, a ta występująca najczęściej staje się ostateczną prognozą [23].

Wybór zmiennych objaśniających

Selekcja zmiennych objaśniających jest ważna w uczeniu maszynowym, poprawia wydajność modelu, eliminując nieistotne lub zbędne cechy, zmniejsza nadmierne dopasowanie oraz złożoność obliczeniową. Metoda KBest, znana również jako SelectKBest [24], to technika selekcji cech, która wybiera *k* najlepszych cech na podstawie miar statystycznych, takich jak np. chi-kwadrat. Przypisuje ona wyniki do każdej cechy, określając ilościowo ich istotność oraz znaczenie dla etykiety klasyfikującej. Metoda ta koncentruje model na najbardziej informatywnych zmiennych.

Optymalizacja hiperparametrów

Hiperparametry pozwalają na konfigurację algorytmu uczenia maszynowego, która nie jest ustawiana na podstawie danych, ale musi zostać zdefiniowana ręcznie przez użytkownika przed uczeniem modelu [25]. Parametry te wpływają na zachowanie i wydajność modelu. Optymalizacja hiperparametrów jest ważna, gdyż może znacznie wpłynąć na wydajność i zdolność uogólniania modelu. Skonfigurowanie nieodpowiednich lub nieoptymalnych wartości może prowadzić do słabej zdolności predykcji modelu, w tym nadmiernego bądź niedostatecznego dopasowania do danych. Jednym z popularnych sposobów wyboru jest metoda przeszukiwania siatki, którą powszechnie stosuje się do wyboru zmiennych. Polega na zdefiniowaniu macierzy możliwych wartości dla każdego analizowanego hiperparametru. Siatka jest kombinacją różnych wartości do zbadania. Algorytm wyszukiwania siatki następnie wyczerpująco przeszukuje wszystkie możliwe kombinacje hiperparametrów, trenując i oceniając model dla każdej kombinacji [26]. Dla każdego zestawu model jest trenowany, a jego wydajność jest oceniana przy użyciu wybranej metryki oceny, takiej jak dokładność lub średni błąd kwadratowy. Kombinacja wartości, która zapewnia najlepszą wydajność zgodnie z metryką oceny, jest wybierana jako optymalny zestaw hiperparametrów. Wyszukiwanie za pomocą tej metody jest prostym i systematycznym podejściem do optymalizacji hiperparametrów, ponieważ ocenia wszystkie możliwe kombinacje, jednak proces ten jest czasochłonny.

Ocena efektywności

W przypadku klasyfikacji miary jakości klasyfikatora są definiowane przez zgodność między szacowaną etykietą klasy a rzeczywistą etykietą – macierz konfuzji [27]. Macierz ta zawiera liczbę próbek sklasyfikowanych jako prawdziwie negatywne, fałszywie pozytywne, fałszywie negatywne i prawdziwie pozytywne [28].
Na tej podstawie obliczane są następujące wartości:

False Positive Rate

$$fpr = \frac{fp}{(fp + tn)}$$

False Negative Rate

$$fnr = \frac{fn}{(fn + tp)}$$

Precyzja

$$prec = \frac{tp}{(tp + fp)}$$

Recall

$$r1 = \frac{tp}{(tp + fn)}$$

W przypadku wykrywania złośliwego oprogramowania celem jest osiągnięcie najwyższego możliwego wskaźnika dokładności przy jednoczesnym zminimalizowaniu liczby przypadków prawdziwie negatywnych. Fałszywie pozytywne przypadki są bezpieczne dla użytkownika, ale jednocześnie irytujące i mogą spowodować, że nie będzie on korzystał z aplikacji. Przypadki prawdziwie negatywne są szczególnie niebezpieczne, ponieważ użytkownik instaluje i używa niebezpiecznej aplikacji, traktując ją nie tylko jako bezpieczną, lecz także zweryfikowaną przez zaufany podmiot. Dlatego ważne jest, aby znaleźć właściwą równowagę między tymi parametrami.

Implementacja

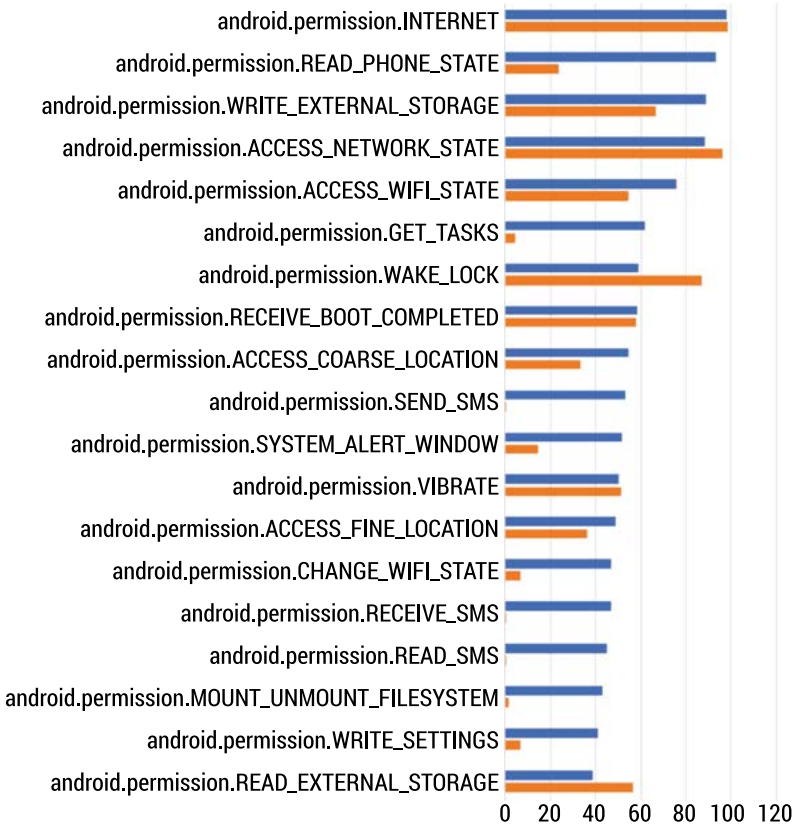
Przygotowane skrypty służą do trzech celów. Pierwsza grupa została zaimplementowana, aby uzyskać zbiory uczące na podstawie repozytorium. Skrypty te dekompilują pakiety instalacyjne, analizują właściwości aplikacji, zapisują do pliku csv oraz wybierają zmienne objaśniające wykorzystane w kolejnych krokach. Druga grupa wykorzystuje algorytmy uczenia maszynowego, a ostatnia ocenia efektywność ich działania. Skrypty do analizy danych oraz przeprowadzenia procesów uczenia maszynowego zostały napisane w języku Python, z wykorzystaniem bibliotek Scikit-Learn oraz Pandas.

5. Wyniki

Podczas analizy zbadano zgromadzony zestaw danych, aby porównać aplikacje bezpieczne z niebezpiecznymi. W tym celu wykorzystano skrypty i arkusz kalkulacyjny. W pierwszej kolejności porównano uprawnienia dla aplikacji sklasyfikowanych jako

bezpieczne i dla tych, które uznawane są za niebezpieczne. Wyniki przedstawiono na rys. 3 oraz w tabeli 1.

Według analizy oba typy aplikacji najczęściej otrzymywały dostęp do internetu, co jest normalne dla współczesnych aplikacji. Większość z nich potrzebuje internetu do poprawnego działania, np. logowania użytkownika. Rozbieżność zaczyna się na drugiej pozycji, gdzie złośliwe oprogramowanie wykorzystuje uprawnienie READ_PHONE_STATE do odczytu danych o telefonie. Ponadto złośliwe aplikacje często wykorzystują uprawnienie, aby dostać się do pamięci zewnętrznej telefonu. Natomiast najbardziej zauważalną różnicą jest uprawnienie GET_TASKS (wycofane), które pozwala aplikacji na pobieranie informacji o aktualnościach, oraz uprawnienie pozwalające na korzystanie z SMS-ów. W przypadku tych uprawnień bezpieczne aplikacje rzadko proszą o dostęp do tych zasobów.



RYS. 3. Uprawnienia aplikacji bezpiecznych i niebezpiecznych

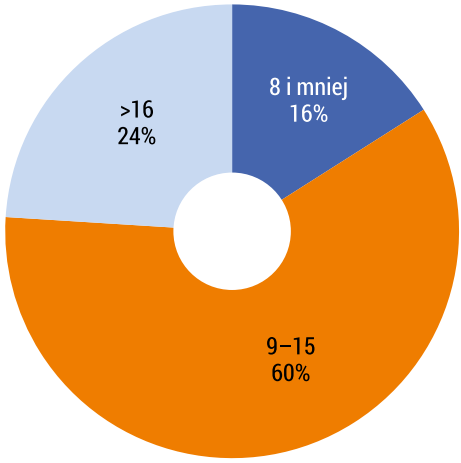
ŹRÓDŁO: opracowanie własne.

TABELA 1. Udział poszczególnych uprawnień w aplikacjach bezpiecznych oraz niebezpiecznych

Lp.	Pozwolenie	*Malware* [%]	Bezpieczne [%]
1.	android.permission.INTERNET	95,18987	98,66889
2.	android.permission.READ_PHONE_STATE	83,29114	30,78203
3.	android.permission.WRITE_EXTERNAL_STORAGE	73,92405	72,71215
4.	android.permission.ACCESS_NETWORK_STATE	90,63291	94,84193
5.	android.permission.ACCESS_WIFI_STATE	61,77215	43,7604
6.	android.permission.GET_TASKS	53,16456	12,31281
7.	android.permission.SEND_SMS	38,98734	2,828619
8.	android.permission.RECEIVE_SMS	36,20253	3,826955
9.	android.permission.RECEIVE_BOOT_COMPLETED	63,5443	28,78536
10.	android.permission.WAKE_LOCK	67,8481	61,23128

ŹRÓDŁO: opracowanie własne.

Dla tego samego zbioru danych zbadano występowanie minimalnego deklarowanego zestawu SDK wymaganego do prawidłowego działania dla aplikacji niebezpiecznych. W rezultacie stwierdzono (rys. 4), że większość aplikacji korzysta z minimalnych wersji SDK, od 9 do 15, co wiąże się z dotarciem do jak największej liczby użytkowników. Niska wartość minimalnego SDK jest oczywistą cechą, ponieważ atakujący ze swoją aplikacją chce dotrzeć do jak największej liczby odbiorców.



RYS. 4. Deklarowany minimalny SDK dla aplikacji niebezpiecznych

ŹRÓDŁO: opracowanie własne.

W tabeli 2 porównano również średnią i medianę liczby wymaganych zezwoleń zadeklarowanych w manifeście aplikacyjnym dla różnych aplikacji.

TABELA 2. Liczba pozwoleń wykorzystywanych przez różne typy aplikacji (mediana i średnia)

Aplikacje	Średnia	Mediana
Niebezpieczne	11,75	9
Bezpieczne	8,56	7

ŹRÓDŁO: opracowanie własne.

W kolejnym kroku przeanalizowano korelację między udzielonym pozwoleniem a przypisaną kategorią na podstawie danych ze zbioru danych (tabela 3). Z przeprowadzonych analiz wynika, że uprawnienia pozwalające na korzystanie z wiadomości SMS najsilniej korelują z klasą aplikacji.

TABELA 3. Korelacja pomiędzy zmiennymi objaśniającymi a etykietą

Pozwolenie	Współczynnik korelacji
SEND_SMS	0,545650
READ_PHONE_STATE	0,408542
RECEIVE_SMS	0,387995
READ_SMS	0,370009
WRITE_SMS	0,267264
WRITE_HISTORY_BOOKMARKS	0,242051
INSTALL_PACKAGES	0,235471
READ_HISTORY_BOOKMARKS	0,230836

ŹRÓDŁO: opracowanie własne.

Następnie do klasyfikacji pakietów instalacyjnych użyto algorytmów uczenia maszynowego. Zastosowano zmienną liczbę zmiennych objaśniających k = 10, 100 oraz wykorzystano wszystkie dane (wyniki przedstawiono w tabelach 4–5). Podczas badania zastosowano walidację krzyżową – aby zapewnić odpowiednią wiarygodność wyników, podzielono zbiór na pięć podzbiorów. Zbadano wyniki dla następujących parametrów: dokładność, precyzja, wskaźnik wyników fałszywie dodatnich.

TABELA 4. Wykorzystane algorytmy uczenia maszynowego dla różnej liczby zmiennych objaśniających dla zbioru wyłącznie z przydzielonymi uprawnieniami

Parametr	KN	SVM	NB	DT	RF	ERF	NN	Liczba zmiennych objaśniających
Dokładność	0,77	0,86	0,80	0,87	0,87	0,87	0,85	K = 10
	0,78	0,88	0,78	0,88	0,89	0,89	0,87	K = 100
	0,78	0,89	0,80	0,78	0,89	0,88	0,88	K = all

Parametr	KN	SVM	NB	DT	RF	ERF	NN	Liczba zmiennych objaśniających
Precyzja	0,68	0,86	0,80	0,90	0,89	0,90	0,90	K = 10
	0,78	0,85	0,81	0,92	0,90	0,91	0,91	K = 100
	0,77	0,86	0,82	0,91	0,90	0,91	0,88	K = all
False Positive Rate	0,26	0,10	0,03	0,02	0,02	0,09	0,08	K = 10
	0,21	0,11	0,02	0,02	0,01	0,09	0,07	K = 100
	0,25	0,12	0,12	0,12	0,11	0,08	0,17	K = all

ŹRÓDŁO: opracowanie własne.

TABELA 5. Wykorzystane algorytmy uczenia maszynowego dla różnej liczby zmiennych objaśniających dla zbioru stosującego pełną analizę statyczną

Parametr	KN	SVM	NB	DT	RF	ERF	NN	Liczba zmiennych objaśniających
Dokładność	0,53	0,93	0,85	0,94	0,90	0,90	0,93	K = 10
	0,98	0,98	0,87	0,98	0,98	0,98	0,98	K = 100
	0,98	0,97	0,79	0,99	0,99	0,99	0,98	K = all
Precyzja	0,52	0,94	0,86	0,94	0,94	0,94	0,92	K = 10
	0,98	0,99	0,98	0,98	0,99	0,99	0,99	K = 100
	0,99	0,99	0,99	1	1	1	0,99	K = all
False Positive Rate	0,96	0,09	0,006	0,09	0,09	0,09	0,08	K = 10
	0,02	0,01	0,02	0,01	0,01	0,01	0,01	K = 100
	0,002	0,03	0,04	0,001	0,001	0,001	0,001	K = all

ŹRÓDŁO: opracowanie własne.

6. Konkluzja

Stwierdzono, że dodatkowa ekstrakcja cech za pomocą głębokiej analizy pakietów ma dobry wpływ na uzyskane wyniki. W zależności od liczby zmiennych objaśniających użytych w procesie uczenia dokładność oraz precyzja (tabele 4–5) były wyższe dla zestawu, podczas gdy liczba fałszywie pozytywnych przypadków była niższa dla liczby zmiennych objaśniających 100. Pomimo stosunkowo wysokiej dokładności uzyskanych wartości należy zbadać możliwość wykorzystania innych metod wykrywania złośliwego oprogramowania, takich jak analiza behawioralna, w tym analiza ruchu sieciowego. Metody stosowane w celu ukrycia się przed analizą statyczną mogą niewątpliwie narazić użytkownika na dodatkowe ryzyko, co także warto rozważyć w dalszych pracach. W większości przypadków najwyższą dokładność uzyskano przy użyciu struktur drzewiastych, takich jak drzewa decyzyjne i lasy losowe.

Bibliografia

[1] Curry D., *Android Statistics* (*2024*), Business of Apps, https://www.businessofapps.com/data/android-statistics/ (dostęp: 15.11.2024).

[2] Shishkova T., Kibba A., *IT threat evolution in Q3 2022. Mobile statistics*, Securelist, https://securelist.com/it-threat-evolution-in-q3-2022-mobile-statistics/107978/ (dostęp: 15.11.2024).

[3] Ceci L., *Google Play Store: number of apps 2023*, Statista, https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ (dostęp: 15.11.2024).

[4] Palmer D., *Google Play malware: If you've downloaded these malicious apps, delete them immediately*, ZDNET, https://www.zdnet.com/article/google-play-malware-if-youve-downloaded-these-malicious-apps-delete-them-immediately/ (dostęp: 15.11.2024).

[5] *Platform architecture*, Android Developers, https://developer.android.com/guide/platform (dostęp: 15.11.2024).

[6] Fang Z., Han W., Li Y., *Permission based Android security: Issues and countermeasures*, „Computer & Security" 2014, vol. 43, s. 205–218.

[7] Johnson R., Wang Z., Gagnon C., Stavrou A., *Analysis of Android Applications' Permissions*, http://code.google.com/p/android-apktool/ (dostęp: 15.11.2024).

[8] Felt A.P., Ha E., Egelman S., Haney A., Chin E., Wagner D., *Android Permissions: User Attention, Comprehension, and Behavior*, w: *SOUPS'12. Proceedings of the Eighth Symposium on Usable Privacy and Security*, red. L.F. Cranor, Washington 20102, s. 1–14.

[9] Shabtai A., Fledel Y., Elovici Y., *Automated static code analysis for classifying android applications using machine learning*, w: *Proceedings – 2010 International Conference on Computational Intelligence and Security, CIS 2010*, Washington 2011, s. 329–333.

[10] Jóźwiak I.J., Michalski M., Pasieczny M., *Zaciemnianie kodu jako forma zabezpieczenia kodu źródłowego*, „Zeszyty Naukowe. Organizacja i Zarządzanie" 2012, z. 61, s. 159–169.

[11] Moser A., Kruegel C., Kirda E., *Limits of Static Analysis for Malware Detection*, w: *Proceedings of the IEEE 23rd Annual Computer Security Applications Conference, Florida, 10–14 December 2007*, Washington 2008, s. 421–430.

[12] Aung Z., Zaw W., *Permission-Based Android Malware Detection*, „International Journal of Scientific & Technology Research 2013", vol. 2, s. 228–234.

[13] Kumaran M., Li W., *Lightweight malware detection based on machine learning algorithms and the android manifest file*, w: *Proceedings 2016 IEEE MIT Undergraduate Research Technology Conference* (*URTC*), Washington 2018.

[14] Allix K., Bissyandé T.F., Klein J., Le Traon Y., *AndroZoo: Collecting Millions of Android Apps for the Research Community*, w: *MSR'16: Proceedings of the 13th International Conference on Mining Software Repositories*, New York 2016, s. 468–471.

[15] Potamias M., Bonchi F., Gionis A., Kollios G., *k-Nearest Neighbors in Uncertain Graphs*, „Proceedings of the VLDB Endowment" 2010, vol. 3(1), s. 997–1008.

[16] Peterson L.E., *K-nearest neighbor*, „Scholarpedia" 2009, vol. 4(2), s. 1883.

[17] Berrar D., *Bayes' Theorem and Naive Bayes Classifier*, w: D.B. Roitberg, *Reference Module in Life Sciences*, Elsevier 2016, s. 1–18.

[18] Ting S.L., Ip W.H., Tsang A.H.C., *Is Naïve Bayes a Good Classifier for Document Classification?*, „International Journal of Software Engineering and Its Applications" 2011, vol. 5(3), s. 37–46.

[19] Quinlan J.R., *Learning Decision Tree Classifiers*, „ACM Computing Surveys" 1996, vol. 28(1), s. 71–72.

[20] Zhou Z.H., Chen Z.Q., *Hybrid Decision Tree*, „Knowledge-Based System" 2002, vol. 15(8), s. 515–528.

[21] Rodriguez-Galiano V.F., Ghimire B., Rogan J., M. Chica-Olmo, Rigol-Sanchez J.P., *An assessment of the effectiveness of a random forest classifier for land-cover classification*, „ISPRS Journal of Photogrammetry and Remote Sensing" 2012, vol. 67, s. 93–104.

[22] Liaw A., Wiener M., *Classification and Regression by randomForest*, „R News" 2002, vol. 2(3), s. 18–22.

[23] *Uczenie maszynowe z użyciem Scikit-Learn, Keras i TensorFlow*, Google Scholar, https://scholar.google.com/scholar?hl=pl&as_sdt=0%2C5&q=Uczenie+maszynowe+z+u%C5%BCyciem+Scikit-Learn%2C+Keras+i+TensorFlow&btnG=#d=gs_cit&t=1719140333998&u=%2Fscholar%3Fq%3Dinfo%3AkX78h27W_foJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Dpl (dostęp: 15.11.2024).

[24] Kumar V., Minz S., *Smart Computing Review Feature Selection: A literature Review*, „Smart Computing Review" 2014, vol. 4(3), s. 211–229.

[25] *Automated Machine Learning Methods, Systems, Challenges*, red. F. Hutter, L. Kotthoff, J. Vanschoren, Cham 2019.

[26] Dagnew G., Shekar B.H., *Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data*, w: *Proceedings 2019 Second International Conference on Advanced Computational and Communication Paradigms*, Washington 2019.

[27] Sarker I.H., Kayes A.S.M., Watters P., *Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage*, „Journal of Big Data" 2019, vol. 6, 57.

[28] Alvarez S.A., *An exact analytical relation among recall, precision, and classification accuracy in information retrieval*, Technical Report BC-CS-2002-01, Computer Science Department, Boston College.

Detecting malware on the Android platform using machine learning algorithms

Summary: Detecting malware is one of the most important challenges in modern mobile operating systems. Recognizing the threat before it is realized is a key issue for protecting user privacy. In the most popular Android operating system, the problem of protecting user data is significant. The applications used in modern mobile systems now allow the management of sensitive data, such as banking data, health data, and virtual documents. Therefore, they are becoming an important target for attackers. The liberal software installation policy on the Android platform encourages increased protection against unknown malicious applications. Thus, one of the main goals of the work presented in the article is to prepare software that protects the end user from losing sensitive data. Using various approaches for analyzing applications on the Android platform, the effectiveness of various machine learning algorithms was investigated. In the course of the research, a method for extracting data from large sets of mobile applications was proposed, and the effect of using different variable selection and hyperparameter optimization algorithms on the effectiveness of malware detection was determined. As a result, data were obtained to determine the effectiveness of the implementation, and the results helped to determine the direction of further work.

Keywords: Android, Malware, Machine Learning

Rozdział VIII
Optymalizacja procesu przydzielania zleceń transportowych oraz generowania optymalnych tras dostaw realizowanych na odcinku ostatniej mili

Jolanta Koszelew[1,2], *Krzysztof Ostrowski*[1], *Grażyna Starzec*[2], *Mateusz Starzec*[2]
[1]*Wydział Informatyki, Politechnika Białostocka,* [2]*Sentio sp. z o.o. w Białymstoku*

Streszczenie: Systemy IT do zarządzania flotą transportową, nazywane TMS-ami, coraz częściej wyposażane są w tzw. planery tras, czyli funkcje optymalizujące proces przydzielania zleceń do pojazdów/kierowców oraz ustalania kolejności tych zleceń w trasie. W rozdziale skupiono się na jednej z kluczowych funkcji planera, która uwzględnia sytuację, gdy nie jest możliwe zrealizowanie wszystkich zleceń przy określonych założeniach i zasobach. Wówczas celem optymalizacji jest realizacja zleceń o najwyższych łącznych priorytetach, przy jednoczesnym przestrzeganiu limitów czasowych. Taka wersja problemu optymalizacyjnego nazywana jest *Team Orienteering Problem* (TOP). Autorzy opisali proces przygotowania i przeprowadzenia testów oryginalnego algorytmu rozwiązującego problem TOP przy użyciu ogólnodostępnych benchmarków sieci transportowej. Zaprezentowali także wpływ wyników badań na realizację komercyjnego projektu B+R, którego efekty zostały wdrożone w systemie AutoHiver (Autohiver.com).

Słowa kluczowe: planer tras, dostawy na odcinku ostatniej mili, strategia algorytmu mrówkowego, benchmarki, TRL, B+R

1. Wprowadzenie

Do zarządzania flotą pojazdów transportujących określone dobra lub realizujących założone zlecenia używa się dziś systemów informatycznych typu TMS (ang. *Transport Management System*) [10]. W ostatnich latach koszty finansowe i ekologiczne wykorzystania floty nabrały bardzo dużego znaczenia, dlatego coraz częściej do standardowych systemów TMS dodawane są funkcje optymalizujące koszty realizacji zleceń, tzw. planery trasy (ang. *Route Planners*) [17]. Nie tylko układają one optymalną kolejność punktów konkretnej trasy, lecz także tak dzielą zbiór zleceń na podzbiory przypisane do poszczególnych pojazdów, że spełnione są wszystkie założenia danego procesu logistycznego (np. okna czasowe, liczba dostępnych

pojazdów itp.), ale też trasy są optymalne pod kątem określonych kryteriów (np. czas realizacji czy ich długość).

Szczególnie ambitnym wyzwaniem jest opracowanie algorytmów realizujących funkcje planera tras w sytuacji, gdy zlecenia są już na odcinku ostatniej mili (ang. *Last Mile Delivery*, LMD) [17]. Trasy i zbiory zleceń w tego typu planerach, np. dla dostaw kurierskich, są dynamiczne. Zbiór zleceń zmienia się codziennie, trasy zawierają nawet po kilkaset punktów, a proces wykonywania zleceń wymaga stosowania wielu różnych kryteriów optymalizacji oraz spełnienia specyficznych warunków.

Planery tras to zestawy algorytmów rozwiązujących problemy optymalizacji wielokryterialnej, należące do rodzin VRP (ang. *Vehicle Routing Problem*) [16] i TOP (ang. *Team Orienteering Problem*) [13].

W niniejszym rozdziale zostaną przedstawione wyniki i wnioski z badań symulacyjnych przeprowadzonych na oryginalnym algorytmie rozwiązującym problem automatycznego i optymalnego przydziału zleceń i tras należący do klasy TOP. Uwzględniona wersja problemu dotyczy sytuacji, w której zasoby floty nie są, z założenia, wystarczające do realizacji całego zbioru zleceń, ale dzięki priorytetom (wagom) przypisanym zleceniom można wybrać te, które mają najwyższą łączną wartość priorytetów.

Badania zostały przeprowadzone celem wdrożenia opracowanego algorytmu w ramach kluczowych funkcji produktu AutoHiver. Jest to system TMS z funkcjami optymalizacji przydziału zleceń i tras, oferowany firmom transportowym realizującym zlecenia typu LMD (np. firmy kurierskie, dostawcy zakupów i posiłków na terenie aglomeracji miejskich).

Badania polegały na:

- opracowaniu oryginalnego algorytmu rozwiązującego wcześniej zdefiniowany problem klasy TOP dla LMD;
- przeprowadzeniu testów algorytmu na wybranych, najlepszych, publicznie dostępnych benchmarkach;
- porównaniu wyników realizacji algorytmu z wynikami osiągniętymi przez inne rozwiązania, które były testowane na tych samych benchmarkach co badany algorytm, a ich rezultaty są dostępne publicznie.

Przed wykonaniem badań zdefiniowano jakościowe parametry techniczne opracowanego algorytmu oraz ich pożądane minimalne wartości. Fakt osiągnięcia tych parametrów świadczył o spełnieniu kamieni milowych projektu B+R i przesądzał o jego dalszej realizacji. Było to warunkiem koniecznym, aby przejść do kolejnej fazy – testów algorytmów planera w warunkach rzeczywistych.

W kolejnych punktach rozdziału opisano problem, strategię algorytmu mrówkowego, która jest podstawą opracowanego algorytmu, oraz wyniki testów porównawczych osiągniętych na benchmarkach. Zakończono go podsumowaniem.

2. Problem klasy TOP w planerze tras dla zastosowań LMD

Team Orienteering Problem (TOP) [1] to zaawansowany problem optymalizacji wielokryterialnej w teorii grafów i logistyce, który polega na wyznaczeniu zestawu tras dla zespołu agentów (np. kurierów), tak aby maksymalizować łączną wartość zebranych punktów/priorytetów (np. dostarczonych przesyłek) przy jednoczesnym ograniczeniu czasu podróży do ustalonego limitu. Z założenia każda trasa zaczyna się i kończy w tej samej lokalizacji, a pokonujący ją ma do odwiedzenia zestaw punktów, z których każdy ma przypisaną wartość i ograniczenie czasowe.

Cele optymalizacyjne poszukiwanego rozwiązania to:

- maksymalizacja łącznej wartości punktów odwiedzonych przez agentów;
- efektywne wykorzystanie dostępnego czasu dla każdego z agentów.

Kluczowe elementy problemu:

- Wartości punktów – każdy punkt ma przypisaną wartość, która wpływa na decyzję o jego odwiedzeniu.
- Ograniczenie czasowe – każdy agent ma maksymalny czas na pokonanie trasy.
- Zespoły – problem dotyczy wielu agentów, co wprowadza dodatkową złożoność w planowaniu tras.

TOP należy do problemów trudnych obliczeniowo klasy NP. Chodzi o to, że przy względnie małym rozmiarze zadania złożoność obliczeniowa algorytmu dokładnego [2] jest nie do przyjęcia w wielu rozwiązaniach o znaczeniu praktycznym, np. kiedy agentów jest więcej niż 10, a liczba punktów w trasach przekracza 30.

Po raz pierwszy TOP sformułowano w 1996 roku [5] i od tego czasu powstało wiele rozwiązań opartych na metodach sztucznej inteligencji. Mają one zastosowanie przy planowaniu tras kurierskich, ale służą też np. do generowania efektywnych wycieczek turystycznych czy tras przejazdu środkami transportu publicznego w aglomeracjach miejskich [14].

Strategie wykorzystane do projektowania algorytmów dla problemu TOP to algorytmy ewolucyjne [12], genetyczne [9] czy mrówkowe [11]. Powstało również wiele hybrydowych rozwiązań łączących różne strategie [1]. Mimo sporego zbioru rozwiązań wciąż poszukiwane są kolejne. Wiąże się to m.in. z koniecznością osiągnięcia odpowiedniej jakości rozwiązania do określonego rozmiaru sieci transportowej i jej typu (np. dostawy LMD), a także z tym, że najlepsze rozwiązania są nieujawnione ze względu na wykorzystanie komercyjne.

W literaturze można znaleźć formalne definicje problemu TOP [2] zapisane w postaci zestawu formalnych formuł matematycznych. Ponieważ jednak głównym celem pracy jest przedstawienie wyników testów opracowanego algorytmu na benchmarkach, bardziej przydatna będzie poniższa, nieformalna specyfikacja problemu.

Dane wejściowe:

- Z – zbiór zleceń przewidzianych do realizacji danego dnia;
- n – liczba zleceń w zbiorze Z;
- O – punkt startowy i końcowy każdej z tras;
- max(m) – liczba dostępnych pojazdów, które mogą być wykorzystane do realizacji zleceń n przewidzianych na dany dzień, czyli liczba agentów (np. kierowców);
- max(t) – maksymalna długość każdej z tras realizacji zleceń. W praktyce dla zastosowań w planerach LMD może być to maksymalny czas pracy kierowcy w ciągu dnia;
- P_1, P_2, …, P_n – priorytety (punkty) przypisane poszczególnym zleceniom, które określają rangę realizacji zlecenia danego dnia. W szczególnym przypadku priorytety mogą przyjmować wartości 0 i 1, co oznacza odpowiednio brak konieczności realizacji zlecenia danego dnia i konieczność takiej realizacji.

Dane wynikowe:

- m tras, gdzie $m \leq \max(m)$ złożonych ze zleceń zbioru Z, dla których
- suma priorytetów P_i jest możliwie największa oraz
- każda z m tras ma długość mniejszą niż max(t), a także
- każde zlecenie może być realizowane co najwyżej raz w co najwyżej jednej trasie oraz
- wszystkie trasy zaczynają się i kończą w punkcie O.

3. Strategia algorytmu mrówkowego

Celem projektu B+R, w ramach którego został opracowany m.in. algorytm do problemu TOP dla LMD, było stworzenie całego zestawu algorytmów, które umożliwiają realizację funkcji rozwiązujących różne problemy optymalizacyjne, należące zarówno do klasy VRP, jak i TOP. Pierwszym krokiem projektu było opracowanie jednej, spójnej rdzennej strategii algorytmicznej dla wszystkich wariantów problemów występujących w planerze, którą w kolejnych fazach uzupełniano o specyficzne warunki, założenia itp. Ze względu na komercyjny charakter wyników projektu i związaną z tym ochronę własności intelektualnej autorzy nie mogą przedstawić algorytmu i poprzestają na przedstawieniu strategii algorytmicznej.

Po przeanalizowaniu specyfiki problemu badawczego planera dla LMD zastosowano strategię algorytmu mrówkowego ACO (ang. *Ant Colony Optimization*) [15]. Motywy wyboru były następujące:

- metaheurystyka ACO jest idealnie dopasowana do problemów, w których przeszukiwanie rozwiązań opiera się na przechodzeniu grafu;
- wszechstronność algorytmu: możliwość zakodowania wszystkich celów optymalizacji (długość trasy, czas jej trwania, priorytet trasy) w jednym miejscu algorytmu (podczas wyboru kolejnego wierzchołka przez mrówkę);
- łatwość zrównoleglenia algorytmu: podczas przeszukiwania każda mrówka buduje

rozwiązanie niezależnie od pozostałych i współdzieli z innymi mrówkami jedynie dane do odczytu;
- możliwość połączenia z dodatkowymi mechanizmami, np. uczenia maszynowego: algorytm mrówkowy w trakcie działania buduje strukturę tzw. feromonów, które efektywnie reprezentują przydatność krawędzi grafu do budowania dobrych rozwiązań. Ta właściwość algorytmu dobrze łączy się z założeniem przygotowania bardziej uniwersalnego modelu planera, który będzie w stanie dla różnych problemów przewidzieć wstępne wartości feromonów lub wartości funkcji heurystycznej. Te wartości w trakcie obliczeń mogą być dostosowane do konkretnej instancji problemu do rozwiązania.

Algorytm mrówkowy działa następująco: W jego kolejnych iteracjach trasy są budowane za pomocą wędrujących mrówek. Wybór następnego wierzchołka przez mrówkę jest losowy, ale prawdopodobieństwo wyboru zależy od lokalnej wiedzy o problemie (heurystyka, która określa, jak „dobra" jest dana krawędź, np. na podstawie jej kosztu) oraz od wiedzy stadnej (feromony zostawiane przez mrówki). Ta ostatnia jest aktualizowana w kolejnych iteracjach na podstawie krawędzi należących do najlepszych znalezionych rozwiązań. Dodatkowo rozwiązania przechodzą fazę lokalnych poszukiwań. Algorytmy optymalizacji globalnej (jak ACO) umożliwiają skuteczną eksplorację przestrzeni rozwiązań w kierunku obiecujących obszarów, a zastosowanie lokalnych poszukiwań (eksploatacja) pozwala na znalezienie najlepszych rozwiązań na danym obszarze.

Poniżej przedstawiono schemat działania algorytmu mrówkowego [15]. W zależności od wersji problemu (z klas VRP i TOP) poszczególne komponenty algorytmu (inicjalizacja feromonów, wybór wierzchołka i porównanie rozwiązań) działają specyficznie.

Parametry algorytmu:

K – liczba iteracji,
L – liczba mrówek (inaczej tras tworzonych w każdej iteracji),
α – parametr sterujący wpływem feromonów na trasy mrówek (wybór wierzchołka),
β – parametr sterujący wpływem wiedzy lokalnej (heurystyki) na trasy mrówek,
ρ – współczynnik parowania feromonów, steruje tempem odświeżania wiedzy stadnej.

Pseudokod algorytmu:

```
Zainicjuj tablicę feromonów (dla każdej krawędzi grafu)
Ng – zbiór najlepszych rozwiązań z całego przeszukiwania (początkowo pusty)
Powtórz K razy
{
Ni – zbiór najlepszych rozwiązań z danej iteracji (początkowo pusty).
Powtórz L razy {
                    Zainicjuj rozwiązanie R pustymi trasami
                    Dopóki możliwa jest rozbudowa rozwiązania R
                        {
```

Wybierz nieodwiedzony wierzchołek *v* i trasę *t* należącą do *R*

Dodaj v na koniec trasy *t*

}

Optymalizacja lokalna rozwiązania *R*

Zaktualizuj *Ni* o rozwiązanie *R*

}

Zaktualizuj zbiór *Ng* na podstawie rozwiązań z *Ni*

Zaktualizuj tablicę feromonów na podstawie rozwiązań z *Ni* i *Ng*

}

Zwróć najlepsze rozwiązania ze zbioru *Ng*

4. Wyniki przeprowadzonych testów na benchmarkach

Poniżej przedstawiono szczegółowe wyniki testów dla znanych sieci teoretycznych (benchmarków) algorytmu rozwiązującego problem klasy TOP dla LMD. Dla każdego przypadku testowego podano:

- wynik – podstawowy wynik algorytmu, czyli sumę priorytetów P_i zleceń ze zbioru Z, które zostały wybrane do trasy;
- %BP – procentowy błąd przybliżenia w stosunku do optymalnego/najlepszego znanego rozwiązania dla podstawowego wyniku, jaką jest suma priorytetów;
- m – liczba wygenerowanych tras, gdzie $m \leq max(t)$;
- czas – czas obliczeń (w sekundach, przy założeniu, że parametry mocy obliczeniowej są następujące: Oracle VM.Standard3.Flex – 40CPU 2,6 GHz + 120 GB RAM).

Testy przeprowadzono na dwóch zbiorach benchmarków – Chao i Fischettiego [18] – dla których rozwiązania problemu klasy TOP osiągnęły najlepsze znane z literatury rezultaty dla głównego wyniku działania algorytmu, czyli sumy priorytetów.

Wszystkie sieci testowe są publicznie dostępne [18], a najlepsze wyniki dla tych sieci zostały opublikowane [3, 6, 7, 8].

Poniżej przedstawiono wyniki testów porównawczych algorytmu wykonanych na powyższych sieciach.

TABELA 1. Wyniki dla sieci Chao64

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao64	64	2	15	192	0	2	0,2
Chao64	64	2	30	936	1,3	2	0,3
Chao64	64	2	35	1116	0	2	0,4

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao64	64	2	37,5	1188	0	2	0,4
Chao64	64	2	40	1260	0	2	0,4
Chao64	64	3	15	282	0	3	0,2
Chao64	64	3	18,3	606	5,6	3	0,3
Chao64	64	3	21,7	858	4	3	0,4
Chao64	64	3	23,3	996	0,6	3	0,3
Chao64	64	3	25	1080	0	3	0,4
Chao64	64	3	26,7	1152	1,5	3	0,4
Chao64	64	4	15	366	0	4	0,2
Chao64	64	4	16,2	516	2,3	4	0,3
chao64	64	4	17,5	648	6,9	4	0,3
				Średnio	**1,6**		**0,3**

ŹRÓDŁO: opracowanie własne.

TABELA 2. Wyniki dla sieci Chao66

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao66	66	2	20	410	0	2	0,2
Chao66	66	2	25	580	0	2	0,2
Chao66	66	2	27,5	660	1,5	2	0,2
Chao66	66	2	30	800	0	2	0,3
Chao66	66	2	32,5	850	1,2	2	0,3
Chao66	66	2	35	925	0	2	0,3
Chao66	66	2	37,5	990	2,9	2	0,3
Chao66	66	2	40	1150	0	2	0,3
Chao66	66	2	42,5	1180	1,3	2	0,4
Chao66	66	2	45	1260	0	2	0,3
Chao66	66	2	47,5	1300	3	2	0,4
Chao66	66	2	50	1400	0	2	0,4
Chao66	66	2	52,5	1440	1,4	2	0,4
Chao66	66	2	55	1485	1,3	2	0,4
Chao66	66	2	57,5	1560	0,3	2	0,4
Chao66	66	2	60	1610	0	2	0,5
Chao66	66	2	62,5	1640	0,3	2	0,5
Chao66	66	2	65	1680	0	2	0,5
Chao66	66	3	18,3	470	5,1	3	0,2

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao66	66	3	20	595	0	3	0,2
Chao66	66	3	23,3	735	2,6	3	0,3
Chao66	66	3	25	870	0	3	0,3
Chao66	66	3	28,3	1055	1,4	3	0,3
Chao66	66	3	30	1125	0	3	0,4
Chao66	66	3	31,7	1175	1,3	3	0,3
Chao66	66	3	33,3	1260	0	3	0,4
Chao66	66	3	35	1335	0,7	3	0,4
Chao66	66	3	36,7	1395	2,1	3	0,4
Chao66	66	3	38,3	1460	1,7	3	0,4
Chao66	66	3	40	1535	1,3	3	0,4
Chao66	66	3	41,7	1565	1,9	3	0,5
Chao66	66	3	43,3	1630	0,3	3	0,4
Chao66	66	4	20	765	0	4	0,3
Chao66	66	4	21,2	860	0	4	0,3
Chao66	66	4	22,5	900	6,3	4	0,4
Chao66	66	4	23,8	970	5,8	4	0,3
Chao66	66	4	25	1160	0	4	0,3
Chao66	66	4	26,2	1230	5,4	4	0,3
Chao66	66	4	27,5	1320	0	4	0,4
Chao66	66	4	28,8	1370	1,4	4	0,4
Chao66	66	4	30	1440	0,7	4	0,4
Chao66	66	4	31,2	1500	1,3	4	0,4
Chao66	66	4	32,5	1540	4,9	4	0,4
				Średnio	**1,3**		**0,4**

ŹRÓDŁO: opracowanie własne.

TABELA 3. Wyniki dla sieci Chao100

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao100	100	2	25	197	4,4	2	0,4
Chao100	100	2	30	327	4,1	2	0,2
Chao100	100	2	35	447	1,1	2	0,3
Chao100	100	2	40	521	1,9	2	0,3
Chao100	100	2	45	595	3,7	2	0,3
Chao100	100	2	50	667	2,9	2	0,4
Chao100	100	2	55	753	0,5	2	0,5

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao100	100	2	60	808	3,2	2	0,5
Chao100	100	2	65	872	5	2	0,5
Chao100	100	2	70	952	1,3	2	0,5
Chao100	100	2	75	1004	1,8	2	0,6
Chao100	100	2	80	1067	0,7	2	0,6
Chao100	100	2	85	1109	2	2	0,6
Chao100	100	2	90	1170	0,3	2	0,7
Chao100	100	2	95	1200	1,5	2	0,7
Chao100	100	2	100	1233	0,7	2	0,8
Chao100	100	2	105	1238	2,4	2	0,7
Chao100	100	2	110	1276	1,2	2	0,8
Chao100	100	2	115	1297	0,5	2	0,8
Chao100	100	2	120	1305	0,1	2	0,8
Chao100	100	3	23,3	182	5,7	3	0,3
Chao100	100	3	30	468	0	3	0,3
Chao100	100	3	33,3	559	3,5	3	0,3
Chao100	100	3	36,7	622	4,7	3	0,3
Chao100	100	3	40	720	1,2	3	0,4
Chao100	100	3	43,3	772	4,6	3	0,4
Chao100	100	3	46,7	838	2,7	3	0,4
Chao100	100	3	50	904	1,6	3	0,5
Chao100	100	3	53,3	951	2,9	3	0,5
Chao100	100	3	56,7	1012	4,8	3	0,6
Chao100	100	3	60	1070	4,5	3	0,6
Chao100	100	3	63,3	1118	4,6	3	0,6
Chao100	100	3	66,7	1170	4,3	3	0,6
Chao100	100	3	70	1215	3	3	0,7
Chao100	100	3	73,3	1251	1,7	3	0,7
Chao100	100	3	76,7	1277	1,4	3	0,7
Chao100	100	3	80	1292	1	3	0,8
Chao100	100	4	22,5	171	6,6	4	0,4
Chao100	100	4	25	324	0	4	0,2
Chao100	100	4	27,5	428	7,2	4	0,3
Chao100	100	4	30	571	0	4	0,4
Chao100	100	4	32,5	632	3,8	4	0,3
Chao100	100	4	35	720	1,6	4	0,4
Chao100	100	4	37,5	779	5,1	4	0,4

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao100	100	4	40	876	0,5	4	0,5
Chao100	100	4	42,5	904	1,6	4	0,5
Chao100	100	4	45	943	3,5	4	0,6
Chao100	100	4	47,5	1003	5,5	4	0,6
Chao100	100	4	50	1062	5,5	4	0,6
Chao100	100	4	52,5	1123	3,3	4	0,6
Chao100	100	4	55	1198	1,5	4	0,7
Chao100	100	4	57,5	1211	3,9	4	0,7
Chao100	100	4	60	1241	3,4	4	0,7
Chao100	100	4	60	1281	0,3	4	0,7
				Średnio	**2,7**		**0,5**

ŹRÓDŁO: opracowanie własne.

TABELA 4. Wyniki dla sieci Chao102

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao102	102	2	40	190	0	2	0,2
Chao102	102	2	50	289	0,3	2	0,2
Chao102	102	2	60	383	1	2	0,3
Chao102	102	2	70	457	0,4	2	0,3
Chao102	102	2	80	517	0,8	2	0,3
Chao102	102	2	90	579	0,2	2	0,4
Chao102	102	2	100	633	2	2	0,4
Chao102	102	2	110	700	0,7	2	0,4
Chao102	102	2	120	751	2,1	2	0,4
Chao102	102	2	130	809	2,2	2	0,5
Chao102	102	2	140	862	2,9	2	0,5
Chao102	102	2	150	919	2,8	2	0,5
Chao102	102	2	160	984	1,8	2	0,6
Chao102	102	2	170	1025	1,8	2	0,6
Chao102	102	2	180	1080	1,3	2	0,6
Chao102	102	2	190	1124	1,1	2	0,6
Chao102	102	2	200	1160	1,6	2	0,6
Chao102	102	2	200	1161	1,5	2	0,6
Chao102	102	3	53,3	417	1,9	3	0,3
Chao102	102	3	60	480	1,4	3	0,3

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
Chao102	102	3	66,7	539	4,4	3	0,3
Chao102	102	3	73,3	632	0,2	3	0,3
Chao102	102	3	80	681	0,4	3	0,4
Chao102	102	3	86,7	750	1,6	3	0,4
Chao102	102	3	93,3	784	4,4	3	0,4
Chao102	102	3	100	857	1,9	3	0,5
Chao102	102	3	106,7	916	1,4	3	0,5
Chao102	102	3	113,3	981	0,6	3	0,5
Chao102	102	3	120	1016	1	3	0,6
Chao102	102	3	126,7	1058	2,1	3	0,5
Chao102	102	3	133,3	1091	2,6	3	0,6
Chao102	102	4	35	217	0	4	0,2
Chao102	102	4	40	283	0,7	4	0,2
Chao102	102	4	45	359	1,9	4	0,3
Chao102	102	4	55	507	2,5	4	0,3
Chao102	102	4	60	569	3,6	4	0,3
Chao102	102	4	65	646	0	4	0,4
Chao102	102	4	70	725	0,7	4	0,4
Chao102	102	4	75	777	0,5	4	0,4
Chao102	102	4	80	823	2,7	4	0,4
Chao102	102	4	85	893	1,8	4	0,5
Chao102	102	4	90	965	0,5	4	0,5
Chao102	102	4	95	1018	0,4	4	0,6
Chao102	102	4	100	1044	3,1	4	0,5
				Średnio	**1,52**		**0,42**

ŹRÓDŁO: opracowanie własne.

W przypadku sieci Fischettiego istnieją trzy różne sposoby generowania priorytetów:

- generacja I – priorytety binarne: 1 – zlecenia zrealizowane, O – zlecenie bez możliwości realizacji danego dnia, czyli optymalizowana jest liczba zrealizowanych zleceń;
- generacja II – priorytety pseudolosowe z zakresu <1, 100>: $P_i = 1 + (7141^* i + 73) \; mod \; 100$;
- generacja III – priorytety z zakresu od 1 do 100, ale proporcjonalne do odległości od punktu startu O: $P_i = 1 + \lfloor 99 \cdot d(1, i) / d_{max} \rfloor$ (gdzie $d(1, i)$ to odległość wierzchołka i od startu O, natomiast d_{ma} to odległość ze startu do najdalszego wierzchołka).

TABELA 5. Wyniki dla sieci Fischettiego dla generacji I

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
bier127	127	2	29570,5	99	6,6	2	1,1
bier127	127	3	19713,7	96	6,8	3	0,9
pr136	136	2	24193	61	3,2	2	0,7
kroA200	200	4	3671	79	2,5	4	0,8
kroB200	200	2	7359,5	108	2,7	2	1,5
gil262	262	3	396,3	97	4	3	1,4
gil262	262	4	297,3	76	2,6	4	0,9
pr264	264	4	6142	105	1,9	4	1,1
pr299	299	2	12048	136	2,2	2	2,3
pr299	299	3	8032	107	3,6	3	1,4
pr299	299	4	6024	81	3,6	4	0,9
lin318	318	2	10522,5	168	6,7	2	3,7
lin318	318	3	7015	144	3,4	3	2,1
rd400	400	4	1910,3	202	5,2	4	4,7
				Średnio	**3,93**		**1,68**

ŹRÓDŁO: opracowanie własne.

TABELA 6. Wyniki dla sieci Fischettiego dla generacji II

Sieć	n	Max (m)	Max(t)	Wynik	%BP (wynik)	m	Czas
bier127	127	2	29570,5	5201	4,8	2	1
bier127	127	3	19713,7	5151	4,5	3	0,9
bier127	127	4	14785,2	4807	6,1	4	0,9
pr136	136	2	24193	3592	1,3	2	0,6
kroA150	150	2	6631	4215	2,8	2	0,9
rat195	195	2	581	5041	2,1	2	1,2
kroB200	200	2	7359,5	6017	2,7	2	1,4
kroB200	200	4	3679,8	4832	2,3	4	0,9
ts225	225	2	31661	5859	0	2	1,5
gil262	262	2	594,5	7109	5,2	2	2,2
gil262	262	3	396,3	5491	2,2	3	1,3
pr264	264	2	12284	6516	1,8	2	1,4
pr264	264	3	8189,3	6321	1,5	3	1,3
pr264	264	4	6142	5475	2	4	1,1
pr299	299	3	8032	5687	5,5	3	1,4
pr299	299	4	6024	4341	2,6	4	1

Sieć	n	Max (m)	Max(t)	Wynik	%BP (wynik)	m	Czas
lin318	318	2	10522,5	8947	6,3	2	3,5
lin318	318	3	7015	7530	3,3	3	2,1
rd400	400	4	1910,3	11742	2,4	4	4,6
				Średnio	**3,13**		**1,54**

ŹRÓDŁO: opracowanie własne.

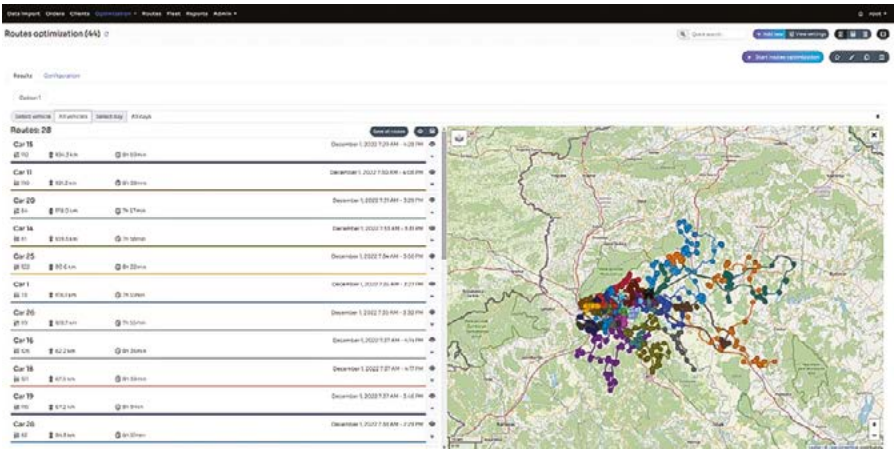
TABELA 7. Wyniki dla sieci Fischettiego dla generacji III

Sieć	n	Max (m)	Max (t)	Wynik	%BP (wynik)	m	Czas
kroA150	150	3	4420,7	2679	1,7	3	0,6
rat195	195	3	387,3	2511	2,4	3	0,8
kroB200	200	2	7359,5	4451	6,6	2	1,4
kroB200	200	3	4906,3	2991	1,2	3	1
gil262	262	2	594,5	7018	2,3	2	2,4
gil262	262	4	297,3	2419	3,5	4	0,9
pr264	264	3	8189,3	2675	3,5	3	1,4
pr299	299	2	12048	5613	2	2	2,3
pr299	299	3	8032	3491	4,5	3	1,5
pr299	299	4	6024	2224	1,9	4	1
lin318	318	2	10522,5	7752	2,3	2	3,8
lin318	318	4	5261,3	3735	1,6	4	1,5
rd400	400	4	1910,3	9727	6,6	4	4,9
				Średnio	**3,1**		**1,8**

ŹRÓDŁO: opracowanie własne.

Ze względu na fakt, że algorytm rozwiązujący problem TOP dla LMD został opracowany w celu zastosowania w komercyjnym systemie planera tras, wyniki testów wykonanych na benchmarkach były kluczowe do podjęcia decyzji. Należało znaleźć odpowiedź na pytanie: Czy jakość algorytmu jest wystarczająco dobra, aby przejść do testów w warunkach rzeczywistych, które wymagały zaangażowania dużo większych zasobów finansowych i organizacyjnych? W przypadku planera tras testy te wykonywane są bowiem przy intensywnym udziale przyszłych klientów systemu (np. firmy kurierskiej) oraz wymagają przygotowania jego prototypu umożliwiającego przeprowadzenie takich testów. W przypadku systemu AutoHiver należało przygotować prototyp panelu dyspozytora (rys. 1), który umożliwiał wywołanie algorytmów planera przez jego przyszłego użytkownika – dyspozytora zleceń. Ponadto niezbędne było stworzenie prototypu aplikacji mobilnej dla kierowców (rys. 2), która z jednej strony służy do nawigacji kierowców po punktach tras wygenerowanych

przez algorytm, ale z drugiej strony taka aplikacja ma też zbierać dane z realnych tras w celu potwierdzenia, że kierowcy rzeczywiście wykonali zlecenia w takiej kolejności, jaką wygenerował algorytm, zużywając przy tym tyle czasu, ile zostało przewidziane w wynikach algorytmu.



RYS. 1. Prototyp panelu dyspozytora

ŹRÓDŁO: opracowanie własne.

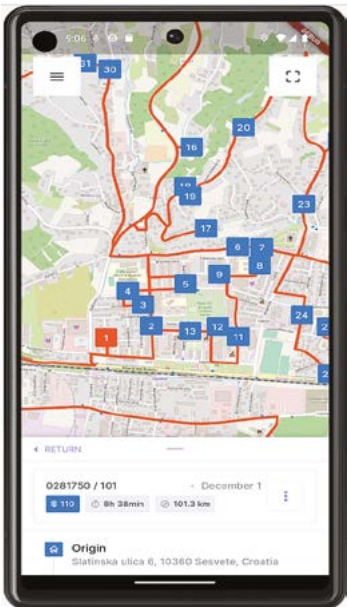
W projekcie B+R przewidziano następujące kamienie milowe dla algorytmu rozwiązującego problem klasy TOP dla LMD:

- *m* – liczba tras wygenerowanych przez algorytm w trakcie testów powinna wynieść co najmniej 700;
- %BP – błąd przybliżenia algorytmu nie powinien przekroczyć 10%;
- czas – czas realizacji jednego wywołania algorytmu nie powinien być dłuższy niż 5 sekund dla zbioru zleceń Z o rozmiarze $n \leq 500$, przy jednoczesnej liczbie aktywnych użytkowników równej lub nie mniejszej niż 100.

W testach przeprowadzonych na benchmarkach, których wyniki zamieszczono w tabelach 1–7, wygenerowano łącznie 1492 trasy, czyli ponad dwa razy więcej niż założono w kamieniu milowym dotyczącym parametru *m*.

W aspekcie błędu przybliżenia, czyli osiągniętej w testach wartości parametru %BP, zanotowano następujące wartości:

- %BP algorytmu dla sieci Chao64: 1,6%;
- %BP algorytmu dla sieci Chao66: 1,3%;
- %BP algorytmu dla sieci Chao100: 2,7%;
- %BP algorytmu dla sieci Chao102: 1,52%;
- % BP algorytmu dla sieci Fischettiego, generacja I: 1,68%;
- % BP algorytmu dla sieci Fischettiego, generacja II: 3,13%;
- % BP algorytmu dla sieci Fischettiego, generacja III: 3,1%.



RYS. 2. Prototyp aplikacji mobilnej

ŹRÓDŁO: opracowanie własne.

Czas realizacji jednego wywołania algorytmu w żadnym teście nie przekroczył 5 sekund. Dzięki zastosowaniu chmurowych systemów obliczeniowych można uruchomić dowolną liczbę instancji algorytmu optymalizującego i rozdzielać obliczenia za pomocą mechanizmu kolejkowego, co powoduje, że czas obliczeń jest niezależny od liczby aktywnych użytkowników.

Tym samym na podstawie rezultatów osiągniętych w testach można było jednoznacznie stwierdzić, że opracowany algorytm rozwiązujący problem TOP dla LMD osiągnął poziom TRL 4 [4]. Na tej podstawie można było podjąć decyzję o kontynuacji projektu B+R, czyli wykonaniu testów w warunkach rzeczywistych.

5. Podsumowanie

Celem niniejszej pracy było przedstawienie metodyki planowania i realizacji badań przemysłowych jako elementu projektu B+R, dążącego do rozwoju technologii z poziomu badań podstawowych, czyli TRL 2, do poziomu TRL 4. Cel ten został zrealizowany przez przedstawienie przykładu zastosowania badań symulacyjnych z użyciem danych benchmarkowych, których efektem było potwierdzenie spełnienia kamieni milowych. Przykład dotyczył algorytmu rozwiązującego problem optymalizacyjny klasy TOP dla LMD. Sieci benchmarkowe najczęściej wykorzystuje się, by potwierdzić hipotezy badawcze przedstawiane w artykułach naukowych. Opisany

w pracy przykład zastosowania benchmarków w projekcie B+R pokazuje dwie kluczowe możliwości takiego podejścia. Po pierwsze, obniża koszty prac rozwojowych w projekcie B+R w postaci testów technologii przeprowadzenia w warunkach rzeczywistych, gdyż na poziomie TRL 4, po testach na benchmarkach, algorytm jest już odpowiednio wysokiej jakości. Po drugie, daje możliwość upowszechniania wyników prac B+R w publikacjach naukowych bez konieczności ujawniania rozwiązań stosowanych komercyjnie.

Przedstawiony przykład badań symulacyjnych zakończył się sukcesem. Kamienie milowe etapu badań przemysłowych zostały osiągnięte, testy technologii w warunkach rzeczywistych zrealizowane i ostatecznie produkt, w którym zostały zastosowane wyniki projektu B+R, trafił na rynek na początku 2024 roku.

*

W pracy wykorzystano elementy projektu POIR.01.01.01-00-1484/20-00 „TRASA – opracowanie i weryfikacja algorytmów optymalizacji tras i przydziału zasobów do ich realizacji", współfinansowanego z Programu Operacyjnego Inteligentny Rozwój 2014–2020, Oś Priorytetowa Wsparcie prowadzenia prac B+R przez przedsiębiorstwa, Działanie Projekty B+R przedsiębiorstw, Poddziałanie Badania przemysłowe i prace rozwojowe realizowane przez przedsiębiorstwa.

Bibliografia

[1] Bederina H., Hifi M., *A hybrid multi-objective evolutionary algorithm for the team orienteering problem*, 4th International Conference on Control, Decision and Information Technologies (CoDIT), 2017.

[2] Boussier S., Feillet D., Gendreau M., *An exact algorithm for team orienteering problems*, „4OR. A Quarterly Journal of Operations Research" 2007, vol. 5, s. 211–230.

[3] Campos V., Martí R., Sánchez-Oro J., Duarte A., *GRASP with path relinking for the orienteering problem*, „Journal of the Operational Research Society" 2014, vol. 65, s. 1800–1813.

[4] Cobos M.P., Quezada V.F., Morloy L.I., Alvarez P.V., González S.P., *A model based on the Technology Readiness Level* (*TRL*) *scale to measure the maturity level of research projects that can become spinoffs in Higher Education Institutions*, Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI), 2021, s. 1–6.

[5] Chao I., Golden B.L., Wasil E.A., *The team orienteering problem*, „European Journal of Operational Research" 1996, vol. 88, s. 464–474.

[6] Dang D., Guibadj R.N., Moukrim A., *An effective PSO-inspired algorithm for the team orienteering problem*, „European Journal of Operational Research" 2013, vol. 229, s. 332–344.

[7] Fischetti M., Salazar-González J., Toth P., *Solving the Orienteering Problem through Branch-and-Cut*, „INFORMS Journal of Computing" 1998, vol. 10, s. 133–148.

[8] Gunawan A., Lau H.C., Lu K., *SAILS: Hybrid algorithm for the Team Orienteering Problem with Time Windows*, w: *MISTA 2015. Proceedings of the 7th Multidisciplinary International Scheduling Conference, Prague, Czech Republic, August 25–28, 2015*, red. Z. Hanzalek, G. Kendall, B. McCollum, P. Sucha, Prague 2015, s. 276–295.

[9] Koszelew J., Ostrowski K., *A Genetic Algorithm with Multiple Mutation which Solves Orienteering Problem in Large Networks*, w: *Computational Collective Intelligence. Technologies and Applications. 5th International Conference, ICCCI 2013, Craiova, Romania, September 11–13, 2013, Proceedings*, red. C. Bădică, N.T. Nguyen, M. Brezovan, Cham 2013, s. 356–366.

[10] Manuel David B.G., Zapana Ruiz Z.R., Meneses Claudio B.A., *Transportation Management and Distribution of Goods in a Transportation Company in the Department of Ancash*, „Southern Perspective / Perspectiva Austral" 2023, vol. 1, 4.

[11] Montemanni R., Weyland D., Gambardella L.M., *An Enhanced Ant Colony System for the Team Orienteering Problem with Time Windows*, w: *ISCCS'11. Proceedings of the 2011 International Symposium on Computer Science and Society*, Washington 2011, s. 381–384.

[12] Ostrowski K., *Parameters Tuning of Evolutionary Algorithm for the Orienteering Problem*, „Advances in Computer Science Research" 2015, vol. 12, s. 53–78.

[13] Ostrowski K., Koszelew J., Karbowska-Chilinska J., Zabielski P., *Evolution-inspired local improvement algorithm solving orienteering problem*, „Annals of Operations Research" 2017, vol. 253, s. 519–543.

[14] Ostrowski K., *An Effective Metaheuristic for Tourist Trip Planning in Public Transport Networks*, „Applied Computer Science" 2018, vol. 14(2), s. 5–19.

[15] Ostrowski K., Starzec G, Starzec M., *The Ant Colony Optimization Algorithm Applied in Transport Logistics*, „Computer Science" 2024, vol. 25(3), s. 1–20.

[16] Pilati F., Tronconi R., *Multi-objective optimisation for sustainable few-to-many pickup and delivery vehicle routing problem*, „International Journal of Production Research" 2024, vol. 62, s. 3146–3175.

[17] Tiwari K.V., Satyendra K.S., *An optimization model for vehicle routing problem in last-mile delivery*, „Expert Systems with Applications" 2023, vol. 222(C), 119789.

[18] Vansteenwegen P., *The Orienteering Problem: Test Instances*, KU Leuven, https://www.mech.kuleuven.be/en/cib/op#section-0 (dostęp: 15.11.2024).

Optimization of Allocating Transport Orders Process and Generating Optimal Routes Implemented in Last Mile Delivery

Summary: IT systems for managing a fleet transport, i.e. TMS, are increasingly equipped with so-called route planners, i.e. functions that optimize the process of separating orders into vehicles/drivers and arranging the order of assigned orders on the road. The work applies to one of the key features of such a planer, which takes into account the situation in which it is not possible to fulfill all orders with specific assumptions and resources. Then the purpose of the optimization is to carry out orders with the highest total priorities, while meeting the time limits imposed on the routes. Such version of the optimization problem is called Team Orienteering Problem (TOP). The aim of the work is to present the simulation test process on the original algorithm that solves TOP problem using the transport network benchmarks available publicly. The results of research and their impact on the implementation of the commercial R&D project, the effects of which were implemented in the AutoHiver (Autohiver.com) system were also presented.

Keywords: Route Planner, Last Mile Delivery, Ant Colony Algorithm Strategy, benchmarks, TRL, R&D

Rozdział IX
Symulacja obliczeń kwantowych

Joanna Wiśniewska[1], *Marek Sawerwain*[2]

[1]*Wydział Cybernetyki, Wojskowa Akademia Techniczna*
[2]*Instytut Sterowania i Systemów Informatycznych, Uniwersytet Zielonogórski*

Streszczenie: Symulacja wybranych modeli obliczeń kwantowych nadal pozostaje ważnym narzędziem w obszarze informatyki kwantowej. Mimo dostępu do kilkunastu rozwiązań sprzętowych, takich jak m.in. IBM Quantum Platform czy D-Wave (oparty na kwantowym wyżarzaniu), istniejące rozwiązania sprzętowe charakteryzują się istotnym wpływem szumu (ang. *noise*) na przeprowadzane obliczenia, co powoduje, że nie są one dokładne, a platformy sprzętowe nie zawsze są łatwo i szybko dostępne. W rozdziale omawiamy kilka autorskich pakietów wspomagających symulację wybranych zagadnień z obszaru obliczeń kwantowych, takich jak obwody kwantowe w małej skali z pełnym możliwościami podglądu stanu, co naturalnie jest możliwe tylko w przypadku symulacji. Przedstawiamy pakiet do obliczeń związanych z metodą kwantowych trajektorii, pakiet EntDetector wspomagający pracę ze splątanymi stanami kwantowymi oraz pokrótce pakiet o skrócie QDC, zawierający wybrane metody uczenia maszynowego, które obecnie są dostępne w ramach modelu obliczeń kwantowych.

Słowa kluczowe: obliczenia kwantowe, kwantowe splątanie, obliczenia równoległe

1. Wprowadzenie

W obecnych czasach nadal czekamy na komputery kwantowe będące w stanie przewyższyć superkomputery klasyczne pod kątem dostępności, stabilności i uniwersalności działania. Ogólnie dostępne są tzw. obliczenia w chmurze (np. IBM Quantum Platform, Azure Quantum, Rigetti Forest) na maszynach o stosunkowo niewielkiej liczbie kubitów, czyli kwantowych bitów. Maszyny te określane są jako NISQ (ang. *Noisy Intermediate-Scale Quantum*) i charakteryzują się dopuszczalnym do obliczeń poziomem zakłóceń, szczególnie przy zastosowaniu niewielkiej liczby bramek kwantowych, które pojawiają się na skutek zjawiska dekoherencji, czyli zaburzeniami wynikającymi z wpływu środowiska zewnętrznego na stan systemu kwantowego. Darmowy dostęp do wspomnianych rozwiązań chmurowych jest dość ograniczony, np. po założeniu konta na IBM Quantum Platform jest dostępnych 10 minut pracy na komputerze kwantowym miesięcznie (stan na lipiec 2024). Dodatkowo wiele osób korzysta z tych darmowych rozwiązań, co powoduje, że tworzą się długie kolejki i czasem trzeba

oczekiwać kilka tygodni na wyniki swojego eksperymentu. Pewnym rozwiązaniem wymienionych problemów jest możliwość skorzystania z programowych symulatorów komputerów kwantowych, np. IBM oferuje bogatą bibliotekę Qiskit w ramach języka Python – można z takiego symulatora skorzystać w chmurze lub na własnym komputerze. Korzystanie z własnego sprzętu niesie ze sobą ograniczenia w zakresie mocy obliczeniowej, ale poza tym gwarantuje dostęp do obliczeń bez kolejek oraz ujawniania wyników swoich eksperymentów. W tym artykule chcielibyśmy przedstawić kilka autorskich rozwiązań w zakresie symulacji obliczeń kwantowych, często z zastosowaniem obliczeń równoległych.

2. Wprowadzenie do mechaniki kwantowej

Ze względu na umożliwienie pełniejszego zrozumienia poniższych treści zacznijmy od wprowadzenia pewnych definicji, z których korzystamy w zastosowaniach informatyki kwantowej.

Obecnie istnieją dwa główne modele obliczeniowe, które doczekały się także implementacji sprzętowej w postaci maszyn NISQ (nie uwzględniamy kwantowej maszyny Turinga [5] ze względu na niską przydatność oraz modelu Adlemana–Liptona [35], który wykorzystuje do obliczeń sekwencję DNA), według których możemy klasyfikować realizacje komputerów kwantowych:

- model obwodowy [18] – obliczenia są realizowane za pomocą obwodu bramek kwantowych (np. komputer firmy IBM czy Rigetti);
- obliczenia adiabatyczne [3], w których następuje ewolucja stanu kwantowego opisywana przez tzw. hamiltonian. Podklasą tego typu obliczeń jest wyżarzanie kwantowe [8], które służy do rozwiązywania wybranych problemów optymalizacji, co jest realizowane przez komputer firmy D-Wave.

Niezależnie od przyjętego modelu obliczeń informacja w komputerze kwantowym opisywana jest przez stan kwantowy, który może być zapisany jako znormalizowany wektor kolumnowy o elementach będących liczbami zespolonymi, czyli wektor z przestrzeni Hilberta:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \dots \\ \alpha_x \end{bmatrix}, \text{ gdzie } \sum_{i=0}^{x} |\alpha_i|^2 = 1, \alpha_i \in \mathbb{C} \tag{1}$$

Jeżeli przez d oznaczymy stopień swobody, to możemy zdefiniować kubit jako jednostkę informacji z $d = 2$, czyli wektor opisujący stan pojedynczego kubitu będzie wektorem dwuelementowym ($d = x - 1$). Jednostkę kwantowej informacji można uogólnić i mówić wtedy o kudicie ze stopniem swobody $d \geq 2$, np. dla $d = 3$ mamy kutrit, który będzie wyrażony za pomocą wektora trzyelementowego, a dla kukuadu $d = 4$ i jest on opisywany czteroelementowym wektorem itd. Dodatkowo w równaniu (1) do opisu

wektora została zastosowana notacja Diraca. Według niej wektory kolumnowe zapisujemy jako „ket", czyli $|\cdot\rangle$, a wektory wierszowe, będące sprzężeniem kolumnowych, jako „bra" $\langle\cdot|$.

Oczywiście pojedynczy kudit nie jest zbyt użyteczny i czym więcej mamy kuditów, tym bardziej skomplikowane obliczenia jesteśmy w stanie przeprowadzić. Dlatego też potrzebujemy n-kuditowych rejestrów kwantowych, które są uporządkowanymi sekwencjami pojedynczych kuditów. Jeżeli łączymy kudity w rejestr, to na wektorach, które je opisują, musimy przeprowadzić operację iloczynu tensorowego, np. z połączenia dwóch stanów jednokubitowych powstaje dwukubitowy rejestr, opisywany przez wektor czteroelementowy z przestrzeni Hilberta:

$$\begin{bmatrix}1\\0\end{bmatrix}\otimes\frac{1}{\sqrt{2}}\begin{bmatrix}1\\i\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\i\\0\\0\end{bmatrix} \tag{2}$$

Oczywiście z matematycznego punktu widzenia możemy łączyć ze sobą w rejestr dowolną liczbę kuditów, nawet gdy mają różne stopnie swobody d.

Stany kwantowe mogą być także opisywane za pomocą równania superpozycji, co dobrze obrazuje wykładniczą pojemność informacyjną takiej struktury, ale najpierw musimy wprowadzić pojęcie bazy obliczeniowej. W klasycznych komputerach wykorzystujemy algebrę George'a Boole'a, więc każdą informację możemy zapisać za pomocą zer i jedynek. Podobnie w komputerach kwantowych potrzebujemy kontrastujących ze sobą stanów kwantowych, aby służyły nam jako odpowiedniki klasycznych zer i jedynek. Zbiór d^n znormalizowanych d^n-elementowych wektorów jest bazą obliczeniową dla układu n kuditów o stopniu swobody d, jeżeli wszystkie te wektory są wzajemnie ortogonalne. Oznacza to, że iloczyn wewnętrzny dla każdej pary wektorów z bazy jest równy zero (gdybyśmy obrazowali stany jednokubitowe za pomocą sfery Blocha, to byłyby one punktami na tej sferze, a każda para przeciwległych punktów stanowi parę wektorów ortogonalnych – widzimy więc, że liczba baz obliczeniowych jest nieskończona). Najpopularniejszą bazą obliczeniową jest tzw. baza standardowa, w której wektory bazowe składają się z jednej jedynki i d^n-1 zer:

$$\begin{bmatrix}1\\0\\0\\\dots\\0\end{bmatrix},\begin{bmatrix}0\\1\\0\\\dots\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\\\dots\\0\end{bmatrix},\dots,\begin{bmatrix}0\\0\\0\\\dots\\1\end{bmatrix} \tag{3}$$

Widzimy, że konstrukcja wektorów jest taka, że dowolny z nich, jeżeli zostanie sprzężony, to mnożony przez każdy z pozostałych da w wyniku zero, np. dla

pojedynczego kubitu mamy w bazie standardowej wektory oznaczane jako $|0\rangle$ i $|1\rangle$, które spełniają:

$$\langle 0|1\rangle = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = 0, \langle 1|0\rangle = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = 0 \tag{4}$$

Równanie superpozycji koduje stan n-kubitowy w odwołaniu do bazy standardowej w następujący sposób:

$$\begin{gathered}|\psi\rangle = \alpha_0|0\ldots000\rangle + \alpha_1|0\ldots001\rangle + \alpha_2|0\ldots010\rangle + \ldots + \alpha_{d^n-1}|1\ldots111\rangle \\ \text{gdzie} \sum_{i=0}^{d^n-1} |\alpha_i|^2 = 1\end{gathered} \tag{5}$$

Zapis ten jest równoważny z zapisem z równania (1), gdy przemnożymy wektory bazowe przez współczynniki α i dodamy je do siebie. Oczywiście równanie superpozycji można dostosować do dowolnej bazy i stanów o dowolnym $d \geq 2$ i $n \geq 1$. Współczynniki α nazywamy amplitudami prawdopodobieństwa, dlatego że gdyby stan zapisany jak w (5) był poddany kwantowemu pomiarowi w bazie standardowej (a pomiar kwantowy odbywa się w konkretnej bazie i polega na zrzutowaniu stanu na jeden z wektorów bazowych), to wartość $|\alpha_i|^2$ odpowiadałaby na pytanie, jakie jest prawdopodobieństwo otrzymania i-tego wektora bazowego jako wyniku.

Dzięki równaniu superpozycji możemy zobrazować wykładniczą pojemność informacyjną stanu kwantowego. Jeżeli wszystkie amplitudy prawdopodobieństwa będą sobie równe, np. dla stanu n-kubitowego:

$$|\theta\rangle = \frac{1}{\sqrt{2^n}}|0\ldots000\rangle + \frac{1}{\sqrt{2^n}}|0\ldots001\rangle + \frac{1}{\sqrt{2^n}}|0\ldots010\rangle + \ldots + \frac{1}{\sqrt{2^n}}|1\ldots111\rangle \tag{6}$$

to mamy do czynienia z sytuacją, gdy w tym samym momencie 2^n różnych wartości (np. liczb binarnych zapisanych w ketach) jest zapisanych w stanie kwantowym $|\theta\rangle$. Oczywiście wykonanie pomiaru spowoduje, że rejestr przyjmie tylko jedną z tych wartości, ale dopóki pomiar nie zostanie wykonany, to możemy przeprowadzać obliczenia równolegle na 2^n wartościach, co jest ogromną i niejedyną zaletą informatyki kwantowej względem informatyki klasycznej.

Kolejnym sposobem wyrażania stanów kwantowych są macierze/operatory gęstości. Stan kwantowy może być stanem czystym, czyli wektorowym, jak np. w równaniu (1). W naturze jednak czyste stany kwantowe interferują ze sobą, tworząc tzw. stany mieszane, które są na tyle złożone, że nie można ich już wyrazić za pomocą wektora. Jeżeli w obliczeniach mamy do czynienia ze stanami czystymi i mieszanymi, to dla ułatwienia wszystkie te stany zapisujemy za pomocą macierzy gęstości. W przypadku stanów wektorowych operator gęstości tworzymy poprzez iloczyn zewnętrzny danego stanu, czyli mnożymy stan przez jego sprzężnie, np. dla stanu $|a\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right)$ operator gęstości ρ_a obliczamy:

$$\rho_a = |a\rangle\langle a| = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right)\cdot\frac{1}{\sqrt{2}}\left(\langle 0| - i\langle 1|\right) = \frac{1}{2}\left(|0\rangle\langle 0| - i|0\rangle\langle 1| + i|1\rangle\langle 0| + |1\rangle\langle 1|\right) \quad (7)$$

Dla powyższego operatora możemy podać jego postać macierzową, czyli macierz gęstości:

$$\begin{aligned}\left[\rho_a\right] = |a\rangle\langle a| &= \frac{1}{2}\left(\begin{bmatrix}1\\0\end{bmatrix}\cdot\begin{bmatrix}1 & 0\end{bmatrix} - i\begin{bmatrix}1\\0\end{bmatrix}\cdot\begin{bmatrix}0 & 1\end{bmatrix} + i\begin{bmatrix}0\\1\end{bmatrix}\cdot\begin{bmatrix}1 & 0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}\cdot\begin{bmatrix}0 & 1\end{bmatrix}\right) = \\ &= \frac{1}{2}\left(\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} - i\begin{bmatrix}0 & 1\\0 & 0\end{bmatrix} + i\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix}1 & -i\\i & 1\end{bmatrix}\end{aligned} \quad (8)$$

Jeżeli mamy do czynienia ze stanem mieszanym, czyli statystyczną mieszanką stanów czystych, to obliczamy macierze gęstości stanów czystych, a następnie mnożymy je przez ich procent zawartości w stanie finalnym i sumujemy je ze sobą:

$$\left[\rho_c\right] = u_1\left[\rho_1\right] + u_2\left[\rho_2\right] + \ldots + u_x\left[\rho_x\right], \text{ gdzie} \sum_{i=1}^{x} u_i = 1,\ u_i \in \mathbb{R} \quad (9)$$

Macierz gęstości, niezależnie od tego, czy opisuje stan czysty, czy mieszany, jest zawsze macierzą hermitowską, określoną nieujemnie, a jej ślad jest równy jeden.

W niniejszej pracy głównie skupiamy się na obwodowym modelu obliczeń kwantowych. W takim podejściu rozumiemy algorytm jako obwód bramek kwantowych, które działają na stan kwantowy. Bramki kwantowe wyrażamy za pomocą macierzy unitarnych, co gwarantuje, że po przeprowadzeniu operacji bramkowej stan kwantowy pozostanie znormalizowany. Wymieńmy kilka podstawowych bramek jednokubitowych:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}, I = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, NOT = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}, Z = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \quad (10)$$

gdzie H to bramka Hadamarda (dzięki której możemy uzyskać efekt superpozycji), I jest bramką identyczności, NOT realizuje operację negacji, a Z jest bramką zmiany fazy lokalnej. Przykładami bramek dwukubitowych mogą być:

$$SWAP = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}, CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix} \quad (11)$$

Bramka $SWAP$ zamienia miejscami wartości kubitów w rejestrze dwuelementowym, np. $SWAP|01\rangle = |10\rangle$. Natomiast bramka $CNOT$ jest bramką kontrolowanej

negacji, czyli zaprzecza wartość drugiego kubitu, jeżeli pierwszy kubit ma wartość $|1\rangle$. Więcej informacji na temat innych bramek kwantowych można znaleźć w [19].

Matematycznie działanie bramki U na czysty stan kwantowy opisujemy jako mnożenie macierzy przez wektor $|\psi\rangle$:

$$U|\psi\rangle = |\psi'\rangle \tag{12}$$

W przypadku stanów zapisanych w formie macierzy gęstości $[\rho]$:

$$U[\rho]U^* = [\rho'] \tag{13}$$

gdzie U^* to sprzężenie macierzy unitarnej U. Widzimy zatem, że rozmiar macierzy symbolizujący bramkę musi być $d^n \times d^n$, jeżeli chcemy nią przetwarzać n-kuditowy stan o stopniu swobody d.

Unitarność bramek kwantowych gwarantuje nie tylko zachowanie poprawnego stanu kwantowego, lecz także odwracalność operacji obliczeniowych. Inaczej jest w przypadku kwantowego pomiaru, gdzie w najbardziej podstawowym modelu, zwanym pomiarem von Neumanna, stosujemy operatory rzutowania, które nie są unitarne. Ponieważ taki pomiar realizowany jest w konkretnej bazie, to operatory rzutowania konstruujemy, wykonując iloczyn zewnętrzny na każdym wektorze bazowym. Oczywiście żeby opisać wszystkie możliwe do otrzymania wyniki w przypadku n-kuditowego stanu o stopniu swobody d, potrzebujemy d^n operatorów rzutowania P_i. Na przykład dla pomiaru stanu pojedynczego kubitu w bazie standardowej, która składa się w tym przypadku z wektorów $|0\rangle$ i $|1\rangle$, operatory projekcji wyglądają następująco:

$$P_0 = |0\rangle\langle 0| = \begin{bmatrix}1\\0\end{bmatrix}\cdot\begin{bmatrix}1 & 0\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}, P_1 = |1\rangle\langle 1| = \begin{bmatrix}0\\1\end{bmatrix}\cdot\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} \tag{14}$$

Jako wynik pomiaru przez rzutowanie otrzymamy jeden ze stanów tworzących bazę pomiarową. Widzimy więc, że pomiar niszczy superpozycję, np. gdyby stan wektorowy $|\delta\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$ zmierzyć w bazie standardowej, to możemy tylko otrzymać wynik $|0\rangle$ lub $|1\rangle$. Ogólnie, gdy matematycznie modelujemy wynik pomiaru przez rzutowanie na stanie $|\psi\rangle$, to otrzymamy stan:

$$|\psi'\rangle = \frac{P|\psi}{\sqrt{\langle\psi|P|\psi\rangle}} \tag{15}$$

gdzie P opisuje wykonywaną operację rzutowania (jeżeli stan $|\psi\rangle$ jest wielokuditowy, to może to być pomiar na całym stanie lub wybranych kuditach, np. gdy w przypadku stanu trzykubitowego pomiar miałby polegać na zrzutowaniu pierwszego kubitu na $|0\rangle$,

drugiego na $|1\rangle$, a trzeci kubit nie byłby mierzony, to wtedy $P = |0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes I_{2\times 2}$, czyli na ostatnim kubicie wykonywana jest operacja identyczności opisywana macierzą o rozmiarach 2×2). W praktyce pomiar odbywa się np. w bazie Z, gdzie na operatorze Z, jak w (10), wykonujemy dekompozycję spektralną, aby otrzymać jego wartości i wektory własne. Wynikiem pomiaru eksperymentalnego będzie jedna z wartości własnych, a wynikiem w postaci stanu kwantowego wektor własny skojarzony z daną wartością własną. Procedura dekompozycji spektralnej polega na wyliczeniu wartości własnych z równania charakterystycznego, a następnie na wyznaczeniu wektorów własnych dla każdej wartości własnej. Dla operatora Z równanie charakterystyczne wygląda następująco:

$$|Z - \lambda I| = 0 \tag{16}$$

gdzie λ to wartości własne, a I jest macierzą identyczności o takich samych rozmiarach jak kwadratowa macierz Z. Otrzymamy tyle wartości λ, ile jest kolumn w macierzy Z. Jeżeli Z jest macierzą hermitowską, to możemy otrzymać niezdegenerowane wektory własne $|w\rangle$ – dla każdej wartości własnej obliczamy:

$$Z|w\rangle = \lambda|w\rangle \tag{17}$$

Wektory własne trzeba wyznaczyć w taki sposób, aby były znormalizowane. W zastosowanym przykładzie z operatorem Z otrzymamy: $\lambda_0 = 1, |w_0\rangle = |0\rangle$ oraz $\lambda_1 = -1, |w_1\rangle = |1\rangle\rangle$. Widzimy więc, że na skutek pomiaru możemy uzyskać wynik 1, który oznacza $|0\rangle$, lub wynik (–1), który wiąże się z $|1\rangle$.

Zauważmy też, że spodziewany średni wynik doświadczenia losowego przy jego wielokrotnym powtarzaniu nazywamy wartością oczekiwaną, a w pomiarach kwantowych mamy właśnie do czynienia z losowym wynikiem eksperymentu. Możemy więc zapisać, że wartość oczekiwana otrzymania stanu $|\psi\rangle$ na skutek zadziałania operatorem Z jest zdefiniowana w notacji Diraca dla stanów czystych jako:

$$\langle Z\rangle = \langle\psi|Z|\psi\rangle \tag{18}$$

W informatyce kwantowej nie można skonstruować uniwersalnej operacji kopiowania ze względu na charakter kwantowych pomiarów, które, jak już wspomniano, niszczą superpozycję stanu kwantowego. Jeżeli zachodzi potrzeba kopiowania stanu, to konstruuje się układy, które dla pewnych stanów mogą działać jako przybliżenie kopiowania i wtedy, aby sprawdzić, czy kopiowanie jest skuteczne, byłoby wskazane porównanie dwóch stanów. Podano wiele miar, które temu służą, a zaimplementowane są w pakiecie QDC [28]. Wspomnijmy tu o Fidelity, która dla stanów ϱ i δ zapisanych operatorem gęstości jest zdefiniowana następująco:

$$F(\varrho,\delta) = \left[Tr\left(\sqrt{\sqrt{\varrho}\,\delta\sqrt{\varrho}}\right)\right]^2, \text{ przy } \mathrm{F}(\varrho,\delta) \in [0,1] \tag{19}$$

Jeżeli ϱ i δ reprezentują ten sam stan, to $F(\varrho,\delta)=1$. Jeżeli stany ϱ i δ są ortogonalne, to $F(\varrho,\delta)=0$.

Zaimplementowane w pakiecie QCS spacery kwantowe są kwantową wersją błądzenia losowego. Jest to przykład prostego procesu stochastycznego określającego ruch losowy, w którym np. cząstka cieczy lub gazu przemieszcza się z jednego położenia do drugiego w sposób trudny do przewidzenia. Analogiczna sytuacja ma miejsce w kwantowych spacerach, aczkolwiek odpowiednikiem poruszającej się cząsteczki jest stan kwantowy, który może podlegać: superpozycji (zajmować kilka lokalizacji jednocześnie), operacjom unitarnym (czyli odwracalnym) i kwantowemu pomiarowi (operacja nieodwracalna).

Zjawiskiem, które oprócz superpozycji nadaje obliczeniom kwantowym ich specyficznego charakteru, jest splątanie. Polega ono na wprowadzeniu dwóch lub większej liczby kuditów w stan, w którym wykonywanie operacji na jednym kudicie powoduje też zmianę stanu innych splątanych z nim kubitów. Zjawisko to jest głównie wykorzystywane w przesyłaniu kwantowej informacji, np. w protokole kwantowej teleportacji. W badaniu kwantowego splątania napotykamy takie problemy, jak wykrywanie splątania i określanie jego poziomu, ponieważ splątanie może być silne lub słabe. Ogólnie czym więcej kuditów bierze udział w splątaniu, tym jest ono słabsze. Przykładem maksymalnego splątania dwóch kubitów może być stan:

$$|\phi\rangle=\frac{1}{\sqrt{2}}\left(|00\rangle+|11\rangle\right) \tag{20}$$

Jeżeli stan jest splątany, to nie możemy go przedstawić za pomocą iloczynu tensorowego pojedynczych kuditów tworzących stan. Jest wiele metod diagnozowania obecności splątania [12] zarówno dla stanów czystych (np. metoda analityczna), jak i mieszanych (np. świadkowie splątania), ale nie są to metody uniwersalne ani łatwe do implementacji numerycznej, które można łatwo skalować w zależności od rozmiaru stanu kwantowego czy stopnia swobody zastosowanych kuditów.

3. Symulator komputera kwantowego QCS i pakiet QTM

Około roku 2004 zaczął powstawać symulator komputera kwantowego QCS (ang. *Quantum Computer Simulator*) [32]. Motywacją do rozpoczęcia prac nad tym oprogramowaniem był ówczesny brak narzędzi, które mogłyby wspierać proces dydaktyczny w ramach ogólnopojętej informatyki kwantowej czy działań badawczych w tej dziedzinie [31]. Pakiet QCS oferuje takie funkcjonalności jak:

- symulacja działania obwodów zbudowanych ze znanych bramek kwantowych (można symulować bramki kuditowe, czyli takie, w których stopień swobody zapisywanej informacji jest równy 2 i więcej – ograniczeniem jest tu oczywiście wydajność komputera klasycznego, na którym odbywa się symulacja, bo zwiększanie stopnia swobody przetwarzanego stanu kwantowego będzie powodowało

zwiększenie jego pojemności informacyjnej, czyli pociągnie za sobą wymóg większej mocy obliczeniowej i nakładów pamięciowych do przetworzenia zadania);
- zapisywanie stanów kwantowych jako wektorów (tylko dla tzw. stanów czystych) oraz macierzy gęstości (stany czyste i mieszane, czyli takie, które są statystycznymi mieszankami stanów czystych);
- symulacja obwodów stabilizujących CHP [1] (m.in. w celu korekcji pojawiających się błędów czy symulacji działania układów wielokubitowych);
- symulacja tzw. kwantowych spacerów;
- możliwość wykonywania obliczeń symbolicznych na liczbach niewymiernych;
- funkcje pomocnicze, np. do operacji na macierzach, dekompozycja spektralna, sprawdzanie podobieństwa stanów kwantowych za pomocą miary Fidelity.

Dodatkowo z pakietu QCS można korzystać na różnych platformach (Windows, Linux) i wykonywać z jego pomocą obliczenia równoległe, co jest bardzo praktyczne ze względu na wykładniczą złożoność stanów kwantowych wynikającą ze zjawiska tzw. superpozycji.

Powyższe możliwości pakietu QCS zostały osiągnięte dzięki temu, że został on wykonany w standardzie ANSI C/C++ [13]. Języki programowania C i C++ gwarantują wieloplatformowość. Dodatkowo możliwość zastosowania pakietu SWIG pozwala na wykorzystywanie biblioteki QSC w innych popularnych językach programowania, np. Java, Python, Ruby. Obliczenia symboliczne wykonano w pakiecie GiNaC, a biblioteki LAPACK i BLAS wspierają operacje na macierzach i wektorach. Rozpraszanie obliczeń zostało zrealizowane za pomocą biblioteki MPI [17], a ich zrównoleglanie dzięki technologii CUDA [NVIDIA]. Repozytorium zawierające kod symulatora jest dostępne pod adresem [27].

Około 2013 roku rozpoczęły się prace nad dodatkiem QTM (ang. *Quantum Trajectory Method*) [33, 36] do symulatora QCS. Ówcześnie dostęp zdalny do obliczeń na komputerach kwantowych nie był tak korzystny jak teraz, głównie z powodu zbywania swojej własności intelektualnej względem zaprojektowanych obwodów i tak niską liczbą kubitów komputera kwantowego, że można ją było symulować na komputerze klasycznym, więc kuszące było posiadanie symulatora zjawisk fizycznych wykorzystywanego do prowadzenia obliczeń kwantowych. Dodatek wykorzystywał metodę kwantowych trajektorii, czyli podejście z grupy metod typu Monte-Carlo.

Metoda kwantowych trajektorii jest bardzo często używana w analizie tzw. otwartych układów kwantowych, czyli takich, gdzie na stan kwantowy może oddziaływać środowisko zewnętrzne. Stan układu kwantowego $|\psi\rangle$ jest wyrażony za pomocą funkcji falowej, która opisuje ewolucję układu w zależności od upływającego czasu t:

$$i\hbar \frac{d}{dt}|\psi\rangle = H|\psi\rangle \tag{21}$$

gdzie $\hbar$ to stała Planca, a H to tzw. hamiltonian. Zauważmy, że H jest operacją podobną do U, jak w (12), ale U, w przeciwieństwie do H, nie uwzględnia tego, co dzieje się

ze stanem kwantowym w trakcie przekształcania $|\psi\rangle$ w $|\psi'\rangle$, a hamiltonian opisuje ewolucję zachodzącą w układzie chwila po chwili, ze względu na zależność do t.

Dodatkowo w losowej chwili t może nastąpić tzw. kwantowy skok (ang. *quantum jump*), czyli właśnie zewnętrzny impuls powodujący zmianę aktualnego stanu kwantowego. Zapiszmy, że operator H działający na stan jest skonstruowany z operatora H_{wew}, który opisuje ewolucję układu bez zakłóceń i części przedstawiającej wpływ środowiska zewnętrznego w następujący sposób:

$$H = H_{wew} - \frac{i\hbar}{2}\sum_m C_m^\dagger C_m \tag{22}$$

gdzie C_m to operatory kolapsu opisujące kwantowe skoki, a $C_m^\dagger$ to sprzężenia tych operatorów.

W metodzie kwantowych trajektorii generowanych jest wiele stanów kwantowych $|\psi\rangle$, które właśnie nazywane są trajektoriami. O wystąpieniu kwantowego skoku w chwili t decyduje wylosowana wartość $r \epsilon (0,1)$. Skok występuje wtedy, gdy $r \le \langle\psi(t)|\psi(t)\rangle$ i stan kwantowy zostaje przekształcony w następujący sposób:

$$\left|\psi(t+\delta t)\right\rangle = \frac{C_m\left|\psi(t)\right\rangle}{\sqrt{\left\langle\psi(t)\right|C_m C_m\left|\psi(t)\right\rangle}} \tag{23}$$

czyli w chwili t na stan działa wylosowany operator kolapsu (z palety m takich operatorów) i po tym zdarzeniu otrzymujemy stan $\left|\psi(t+\delta t)\right\rangle$. Ze względu na podobieństwo (23) do (15) widzimy, że skok kwantowy jest nieodwracalną operacją, tak jak pomiar kwantowy.

Wygenerowane stany kwantowe uśrednia się, aby otrzymać informację o stanie całego układu. Obliczanie pojedynczych trajektorii wraz ze zmianami zachodzącymi na skutek kwantowych skoków można symulować w środowisku równoległym, ponieważ trajektorie powinny być od siebie niezależne. Wykorzystane tu oczywiście zostały mechanizmy MPI [33] oraz CUDA [36]. Dzięki temu, że w wielu wątkach generowane są trajektorie w tym samym czasie, można przyspieszyć obliczenia. Następnie trajektorie są grupowane i uśredniane, a potem wyniki z danej grupy są uśredniane z wynikami z innych grup, czyli do uśredniania trajektorii też stosowane są obliczenia równoległe według tzw. metody turniejowej [18].

Należy zauważyć, że dokładność symulacji systemu otwartego będzie definiowana przez liczbę generowanych trajektorii – czym ta liczba będzie większa, tym lepsze odwzorowanie rzeczywistego procesu fizycznego. Jeżeli symulowane są układy o niewielkich rozmiarach (w sensie liczby kubitów), to w przypadku technologii CUDA można wykorzystywać pamięć lokalną procesorów karty graficznej. W przeciwnym wypadku dla układów o większej liczbie kubitów obliczenia prowadzane są w pamięci głównej karty.

Dodatkowo do wygenerowania trajektorii z uwzględnionymi skokami kwantowymi potrzebne są generatory liczb losowych. W pakiecie QTM mamy w tym zakresie dwie dostępne opcje: można skorzystać z pliku zawierającego liczby prawdziwie losowe wygenerowane przez układ fizyczny [11] (wartości powinny być wcześniej przygotowane, co zależy od szybkości wykorzystywanego urządzenia – w czasach, gdy implementowany był pakiet, wydajność takich urządzeń była znacznie niższa niż obecnie) lub użyć generatorów liczb pseudolosowych. Ze względu na wykorzystywany pakiet CUDA skorzystano z dostępnego w nim rozwiązania cuRAND [21], które charakteryzuje się wystarczającą prędkością generacji liczb o dobrych parametrach statystycznych [22] z uwagi na zastosowanie takich algorytmów jak XORWOW [15] i MRG32k3A [14].

Wynikiem algorytmu QTM są wartości oczekiwane, będące liczbami rzeczywistymi, które otrzymuje się poprzez zastosowanie operatora wartości oczekiwanej do stanu układu po zakończeniu uśredniania trajektorii. Wektor stanu oraz macierze poszczególnych operatorów są strukturami, których elementy są liczbami zespolonymi – wykorzystujemy tu typ danych simpleComplex (jest to uproszczona definicja liczby zespolonej, jednakże jest ona wystarczająca, ponieważ wszystkie wartości są znormalizowane i można ominąć proces normalizacji, co oznacza, iż taki typ danych będzie bardziej wydajny niż cuComplex z pakietu CUDA). Wraz z zastosowaniem typu simpleComplex zostały przeciążone podstawowe operatory sumy, iloczynu itp., co pozwala na czytelny zapis operacji arytmetycznych. W podobny sposób zdefiniowany został typ uVector. Pozostałe pomocnicze elementy, jak typ reprezentujący pełną macierz uMatrix oraz macierz typu sparse uCSRMatrix, zostały również oparte na wzorcu uVector.

W metodzie QTM konieczne jest rozwiązanie problemu zagadnienia początkowego, do czego wykorzystujemy metodę Runge Kutta czwartego rzędu (RK4) oraz metodę BDF (ang. *Backward Differentiation Formula*). Zastosowanie typów simpleComplex czy uVector wraz z bogatym zestawem przeciążonych operatorów również pozwoliło na przejrzystą implementację metod RK4 i BDF. Repozytorium zawierające kod pakietu QTM jest dostępne pod adresem [29].

4. Pakiet EntDetector

Pakiet obliczeniowy EntDetector [7] zaczął powstawać w 2021 roku w celu analizy zjawiska kwantowego splątania. Zauważmy, że obecna wiedza dotycząca tego obszaru jest ciągle bardzo ograniczona – znamy koncepcje wykrywania splątania i pomiaru jego stopnia zaledwie dla układów dwu-, trzykubitowych [2] (ewentualnie dla układu składającego się z jednego kubitu i jednego kutritu [9, 24]). Początkowo więc EntDetector miał oferować procedury i funkcje wspomagające badanie zjawiska kwantowego splątania w układach dwucząsteczkowych. Ze względu na postępy w nauce (np. stwierdzenie splątania w układach o większej liczbie cząsteczek, dla kuditów o wyższym stopniu swobody czy dla stanów mieszanych) można powiedzieć, że pakiet ten będzie rozwijany jeszcze przez wiele lat.

Biblioteka EntDetector powstaje w języku Python. Pakiet ten udostępnia opcję generowania losowych stanów kwantowych, daje możliwość wprowadzania stanów użytkownika, a także skorzystania ze znanych typów stanów splątanych. Zaimplementowano takie przypadki splątania, jak: stany Bella (tzw. pary EPR), stany PPT, stany GHZ, stany W, stany izotropowe, stany Wernera, stany Gisina i inne. Pakiet pozwala na obsługę stanów zarówno czystych, jak i mieszanych.

Najważniejszą częścią biblioteki jest kod diagnozujący obecność splątania i wyliczający jego poziom, jeżeli takowe ma miejsce. Zaimplementowane funkcje wykrywające splątanie to: kryterium Peresa–Horodeckiego, dekompozycja Schmidta oraz autorska metoda SaSD [7] oparta na dekompozycji Schmidta i dekompozycji spektralnej. W ramach badania poziomu splątania wykorzystywane są miary Negativity i Concurrence oraz rząd Schmidta. Również metoda SaSD generuje tabelę, w której określa poziom splątania.

Dokładna informacja dotycząca typów stanów splątanych, kryteriów wykrywania splątania i określania jego poziomu może być znaleziona w [12]. Repozytorium zawierające kod pakietu EntDetector jest dostępne pod adresem [26].

5. Pakiet QDC wraz z modułem VQE

Ze względu na intensywny rozwój metod sztucznej inteligencji w obecnych czasach możemy także zauważyć podobny trend w informatyce kwantowej. Opracowywane są różne algorytmy kwantowe, które mają być odpowiednikami klasycznego uczenia maszynowego. Dlatego też, aby programistycznie wspierać ten kierunek rozwoju, od 2023 roku pracujemy nad pakietem QDC (ang. *Quantum Distance Classificator*). Jest on wykonywany w języku Python i zawiera wiele funkcji i procedur pomagających w rozwiązywaniu zadań klasyfikacji i segmentacji danych, które mogą być wczytywane z plików, a następnie konwertowane są do postaci stanów kwantowych. Konwersja ta, zgodnie z wyborem użytkownika, może być jedno- lub dwustopniowa. W konwersji jednostopniowej wykonywana jest tylko normalizacja do wartości jeden w ramach każdej obserwacji, tak aby finalnie otrzymać znormalizowany stan jak w (1). W konwersji dwustopniowej najpierw wykonywana jest klasyczna normalizacja w obrębie każdej zmiennej, żeby wszystkie wartości pochodziły z przedziału [0, 1]. Celem tej operacji jest oczywiście doprowadzenie do równowagi między zmiennymi, np. aby zmienna osiągająca pierwotnie wartości z przedziału [0, 1000] nie miała większego wpływu na trenowany model niż zmienna przyjmująca wartości z przedziału [–10, 10]. Następnie, jak w przypadku konwersji jednostopniowej, dokonywana jest normalizacja każdej obserwacji do postaci stanu kwantowego. Dodatkowo w pakiecie QDC można też generować losowo stany kwantowe. Takie zbiory danych zawierające obserwacje w postaci stanów kwantowych opartych na kubitach mogą być obrazowane jako punkty na sferze Blocha. Biblioteka zawiera też wiele różnych procedur do rysowania wykresów, co jest bardzo przydatne dla użytkownika w trakcie rozwiązywania zadań klasyfikacji czy segmentacji.

Jeżeli w analizowanym zbiorze danych mamy tzw. pary uczące, to przeprowadzamy klasyfikację, która jest metodą uczenia nadzorowanego, ponieważ w parze uczącej znajduje się informacja o tym, do jakiej klasy należy dana obserwacja. Natomiast segmentacja jest metodą uczenia nienadzorowanego, czyli wtedy mamy do czynienia z obserwacjami, które nie zostały w żaden sposób wcześniej poklasyfikowane. W obu przypadkach do porównywania obserwacji (w postaci stanów kwantowych) stosujemy metryki odległościowe. Na chwilę obecną w pakiecie zaimplementowanych jest dwanaście typów takich metryk, określanych też miarami podobieństwa stanów kwantowych, np. miara Fidelity, miara cosinusowa, odległość typu Manhattan.

W pakiecie zostały zaimplementowane takie metody kwantowego uczenia maszynowego jak: kwantowa klasteryzacja spektralna, klasteryzacja metodą energii potencjalnej, czy klasteryzacja danych kwantowych oraz algorytmy k-means i k-medoids z metryką odległości, naturalnie skonstruowaną dla stanów kwantowych. Repozytorium kodu pakietu QDC znajduje się pod adresem [28].

W ramach pakietu QDC można wyodrębnić rozwiązanie VQE (ang. *Variational Quantum Eigensolver*). Podejście VQE [16, 23] jest tzw. algorytmem hybrydowym – wykorzystuje ono zarówno komputer klasyczny, jak i kwantowy. Wynika to z tego, że pewne zadania, jak rozwiązanie równania Schrödingera czy znajdywanie wartości własnych, mogą być szybko rozwiązane przez komputer kwantowy, który nie został jeszcze wyposażony w algorytmy radzące sobie tak dobrze z zadaniami optymalizacji, jak robią to komputery klasyczne.

W naszym dodatku VQE wykorzystujemy algorytm hybrydowy do rozwiązywania zadania klasyfikacji. Okazuje się, że jest on wyjątkowo skuteczny, gdy klasy mają charakter rozproszony. Za pomocą analizy składowych głównych (funkcja PCA [25] z biblioteki Scikit-Learn dla języka Python) możemy zmniejszyć liczbę wymiarów konkretnego zadania i przedstawić obserwacje na wykresie dwuwymiarowym – to pozwala nam zobaczyć, czy obserwacje należące do tych samych klas tworzą wyraźne skupiska, czy dane są rozproszone.

Jak już wspomniano, ogólna idea algorytmu VQE jest taka, że pewne etapy zadania rozwiązujemy za pomocą układu bramek kwantowych zwanego ansatz, a inne z pomocą komputera klasycznego. W naszym pakiecie znormalizowane stany kwantowe przetwarzane są za pomocą obwodu kwantowego ansatz, którego działanie możemy opisać hamiltonianem H. Obwód ten zwraca stan kwantowy $|\psi\rangle$, który opisywany jest funkcją falową jak w (21). Niski poziom energii E_0 (tzw. groud state) hamiltonianu H to:

$$E_0 \leq \frac{\langle \psi | H | \psi \rangle}{\langle \psi | \psi \rangle} \tag{24}$$

Układ ansatz powinien działać tak, aby amplitudy zwracanego stanu $|\psi\rangle$ spełniały powyższe. Oznacza to, że wartość oczekiwana $\langle H \rangle = \langle \psi | H | \psi \rangle$ ma być minimalizowana. To zadanie również możemy wyrazić za pomocą operacji unitarnej U opisującej działanie układu ansatz. Wtedy zakładamy, że układ jest zbudowany

z x bramek obrotu kubitu o kąt φ (wyczerpujące informacje dotyczące konstruowania obwodów ansatz można znaleźć w pracach [6, 34]):

$$R(\varphi)=\begin{bmatrix}1 & 0\\ 0 & e^{i\varphi}\end{bmatrix} \tag{25}$$

Musimy wtedy znaleźć x wartości kątów (dla każdej bramki), tak aby cały układ realizował:

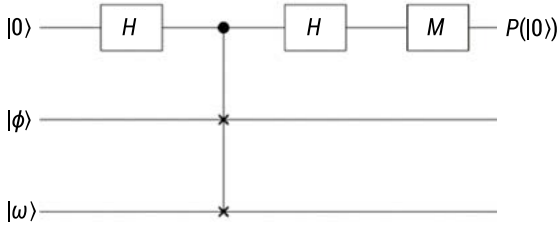
$$U|0\rangle=|\psi\rangle \tag{26}$$

Wartości kątów są optymalizowane przez komputer klasyczny, który minimalizuje funkcję kosztu k:

$$k(\psi^0,\psi)=\sum_{i=1}^{2^n-1}\left|p^0_{\alpha_i}-p_{\alpha_i}\right| \tag{27}$$

gdzie ψ^0 oznacza stan wchodzący na układ ansatz, a ψ to stan po operacji U. Parametr $p^0_{\alpha_i}$ opisuje prawdopodobieństwo otrzymania stanu $|i\rangle$ po pomiarze w bazie standardowej stanu ψ^0, a p_{α_i} to prawdopodobieństwo otrzymania stanu $|i\rangle$ po pomiarze w bazie standardowej stanu ψ.

W momencie, gdy zakończymy proces uczenia, czyli dostrajanie parametrów φ układu ansatz, możemy sprawdzić, czy nowy (niewykorzystywany w procesie uczenia) stan $|\phi\rangle$ należy do tej samej klasy co stan $|\omega\rangle$, który był „poznany" przez układ w trakcie uczenia. Porównujemy wtedy stany za pomocą układu zwanego SWAP-testem [4], który jest układem trzykubitowym składającym się z dwóch bramek Hadamarda, a pomiędzy nimi jest tzw. bramka Fredkina. Układ ten widoczny jest na rys. 1.



RYS. 1. Obwód bramek kwantowych realizujących SWAP-test

ŹRÓDŁO: opracowanie własne na podstawie [4].

Informację o tym, czy stany są podobne, otrzymujemy na skutek pomiaru w bazie standardowej na pierwszym kubicie układu SWAP-test (bramka z oznaczeniem M). Prawdopodobieństwo otrzymania stanu $|0\rangle$ na tym kubicie będzie przyjmowało wartości z przedziału $P|0\rangle\in[0,5;1]$. Jeżeli $P|0\rangle=0,5$, to stany nie są zupełnie do siebie podobne (tzw. stany ortogonalne); gdy $P|0\rangle=1$, to stany $|\phi\rangle$ i $|\omega\rangle$ są identyczne.

W dodatku VQE zaimplementowana została możliwość generowania układów ansatz. Naturalnie może on korzystać ze wszystkich funkcjonalności biblioteki QDC (np. porównywania stanów za pomocą układu SWAP-test). Repozytorium kodu pakietu VQE znajduje się pod adresem [30].

6. Konkluzja

Autorzy opisali swoje doświadczenia, które zdobyli w trakcie implementacji symulatorów obliczeń kwantowych. Można zauważyć, że w ciągu ostatnich dziesięciu lat zmieniały się potrzeby dotyczące funkcjonalności takiego oprogramowania, co oczywiście jest związane z rozwojem technologii. Początkowo komputery kwantowe miały mniej kubitów i były mniej dostępne, więc w procesie wytwarzania symulatorów bardziej koncentrowano się na imitowaniu działania systemów fizycznych. Obecnie wykonanie darmowych obliczeń w chmurze, za którą stoi ponad stukubitowy kwantowy komputer, jest w zasięgu każdej osoby, która założy konto w serwisie, np. IBM Quantum Platform [10]. Dlatego też oprogramowanie implementowane po roku 2020 w mniejszym stopniu symulować zjawiska fizyczne, a bardziej koncentruje się na wspieraniu konkretnych obliczeń, np. dotyczących analizy kwantowego splątania czy elementów związanych z kwantowym uczeniem maszynowym. Aczkolwiek należy też zachować ostrożność względem praw, które różne platformy mogą rościć do obwodów projektowanych przez użytkowników. W tej chwili np. IBM wyraźnie podkreśla, że obwody są własnością intelektualną ich twórców, ale nie zawsze tak było. Przy rejestracji na platformę oferującą obliczenia kwantowe oraz po każdej aktualizacji umowy licencyjnej warto więc czytać załączony dokument z warunkami korzystania z usług.

*

Chcielibyśmy podziękować za przydatne dyskusje z grupą Q-INFO w Instytucie Sterowania i Inżynierii Obliczeniowej (ISSI) Uniwersytetu Zielonogórskiego. Praca została sfinansowana przez Wojskową Akademię Techniczną w ramach projektu nr UGB 701/2024, a także z dotacji na realizację projektów badawczych w dyscyplinie informatyka techniczna i telekomunikacja Uniwersytetu Zielonogórskiego na rok 2024.

Bibliografia

[1] Aaronson S., Gottesman D., *Improved simulation of stabilizer circuits*, „Physical Review A" 2004, vol. 70, 052328.

[2] Acín A., Andrianov A., Costa L., Jané E., Latorre J.I., Tarrach R., *Generalized Schmidt Decomposition and Classification of Three-Quantum-Bit States*, „Physical Review Letters" 2000, vol. 85, 1560.

[3] Albash T., Lidar D.A., *Adiabatic quantum computation*, „Reviews of Modern Physics" 2018, vol. 90, 015002.

[4] Buhrman H., Cleve R., Watrous J., de Wolf R., *Quantum Fingerprinting*, „Physical Review Letters" 2001, vol. 87, s. 167902–167906.

[5] Chudy M., *Wprowadzenie do informatyki kwantowej*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2011.

[6] Fedorov D., Peng B., Govind N., Alexeev Y., *VQE method: a short survey and recent developments*, „Journal of Materials Science: Materials Theory" 2022, vol. 6(2), s. 1–21.

[7] Gielerak R., Sawerwain M., Wiśniewska J., Wróblewski M., *EntDetector: entanglement detecting toolbox for bipartite quantum states*, w: *Computational Science – ICCS 2021. ICCS 2021. Lecture Notes in Computer Science*, vol 12747, red. M. Paszynski, D. Kranzlmüller, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M.A. Sloot, P.M.A., Springer, Cham 2021.

[8] Grant E.K., Humble T.S., *Adiabatic Quantum Computing and Quantum Annealing*, w: *Oxford Research Encyclopedia of Physics*, red. B. Foster, Oxford 2020.

[9] Horodecki M., Horodecki P., Horodecki R., *Separability of mixed states: necessary and sufficient conditions*, „Physics Letters A" 1996, vol. 223(1–2), s. 1–8.

[10] IBM Quantum Platform, https://quantum.ibm.com (dostęp: 18.11.2024).

[11] Jacak M.M., Jóźwiak P., Niemczuk J., Jacak J.E., *Quantum generators of random numbers*, „Scientific Reports" 2021, vol. 11(1), 16108.

[12] Jurkowski J., *Korelacje nieklasyczne. Kwantowe splątanie i dyskord*, Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, Toruń 2014.

[13] Kernighan B., Ritchie D., *C Programming Language. Second Edition*, Prentice Hall, Englewood Cliffs 2002.

[14] L'Ecuyer P., *Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators*, „Operations Research" 1999, vol. 47(1), s. 159–164.

[15] Marsaglia G., *Xorshift RNGs*, „Journal of Statistical Software" 2003, vol. 8(14), s. 1–6.

[16] McClean J., Romero J., Babbush R., Aspuru-Guzik A., *The theory of variational hybrid quantum-classical algorithms*, „New Journal of Physics" 2016, vol. 18(2), s. 1–22.

[17] MPICH, https://www.mpich.org (dostęp: 18.11.2024).

[18] Neapolitan R., Naimipour K., *Podstawy algorytmów z przykładami w C++*, Helion, Gliwice 2004.

[19] Nielsen M.A., Chuang I., *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge 2010.

[20] NVIDIA, *CUDA C Programming guide*, https://developer.nvidia.com/cuda-toolkit (dostęp: 18.11.2024).

[21] NVIDIA, *CUDA Toolkit Documentation – cuRAND*, https://docs.nvidia.com/cuda/curand/index.html (dostęp: 18.11.2024).

[22] NVIDIA, *CUDA Toolkit Documentation – cuRAND Testing*, https://docs.nvidia.com/cuda/curand/testing.html#testing (dostęp: 18.11.2024).

[23] Peruzzo A., McClean J., Shadbolt P., Yung M.H., Zhou X.Q., Love P.J., Aspuru-Guzik A., O'Brien J.L., *A variational eigenvalue solver on a photonic quantum processor*, „Nature Communications" 2014, vol. 5(1), s. 1–7.

[24] Peres A., *Separability Criterion for Density Matrices*, „Physical Review Letters" 1996, vol. 77, 1413.

[25] Python, *Principal component analysis* (*PCA*), https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html (dostęp: 18.11.2024).

[26] Repozytorium EntDetector, https://github.com/qMSUZ/EntDetector (dostęp: 18.11.2024).

[27] Repozytorium QCS, https://github.com/qMSUZ/QCS (dostęp: 18.11.2024).

[28] Repozytorium QDC, https://github.com/qMSUZ/QDCLIB (dostęp: 18.11.2024).

[29] Repozytorium QTM, https://github.com/qMSUZ/QTM (dostęp: 18.11.2024).

[30] Repozytorium VQE, https://github.com/qMSUZ/QDCLIB/tree/main/examples-vqeclass (dostęp: 18.11.2024).

[31] Sawerwain M., *Symulacja obliczeń kwantowych – system QCS w zastosowaniach dla edukacji*, Poznańskie Warsztaty Telekomunikacyjne, Poznań 2006.

[32] Sawerwain M., *Selected Topics in Quantum Programming Theory*, University of Zielona Góra Press, Zielona Góra 2011.

[33] Sawerwain M., Wiśniewska J., *QTM: computational package using MPI protocol for quantum trajectories method*, „PLoS One" 2018, vol. 13(12), e0208263.

[34] Tilly J., Chen H., Cao S., Picozzi D., Setia K., Li Y., Grant E., Wossnig L., Rungger I., Booth G., Tennyson J., *The Variational Quantum Eigensolver: A review of methods and best practices*, „Physics Reports" 2022, vol. 986, s. 1–128.

[35] Wang Z., Ren X., Ji Z., Huang W., Wu T., *A novel bio-heuristic computing algorithm to solve the capacitated vehicle routing problem based on Adleman–Lipton model*, „Biosystems" 2019, vol. 184, 103997.

[36] Wiśniewska J., Sawerwain M., *Symulacja metody kwantowych trajektorii dla problemów optyki kwantowej oraz informatyki kwantowej*, „Symulacja w Badaniach i Rozwoju" 2015, vol. 6(1), s. 67–75.

Simulation of quantum computing

Summary: Simulation of selected quantum computing models still remains an important tool in the field of quantum computing. Despite an access to several hardware solutions, such as: IBM Quantum Platform or D-Wave (based on quantum annealing), existing hardware solutions are characterized by a significant impact of noise on the calculations performed, which results in inaccuracies in the performed calculations. Besides, hardware platforms are not always easily and quickly available. In the article, we discuss several packages, proposed by us, supporting the simulation of selected issues in the area of quantum computing, such as small-scale quantum circuits with full state preview capabilities, which is naturally only possible in the case of simulation. We present a package for calculations related to the quantum trajectory method and the EntDetector package supporting a work with quantum entangled states. We also briefly present the QDC package containing selected machine learning methods that are currently available within the quantum computing model.

Keywords: quantum computing, quantum entanglement, parallel computing

Rozdział X
Sekwencyjne metody kodowania stanów automatów skończonych

Valery Salauyou
Wydział Informatyki, Politechnika Białostocka

Streszczenie: Model automatu skończonego jest często używany do opisu funkcjonowania sprzętu cyfrowego. Podczas optymalizacji urządzeń cyfrowych bardzo ważne jest poprawienie właściwości automatów skończonych, takich jak obszar (koszt implementacji, liczba elementów logicznych), wydajność (szybkość) i zużycie energii.
Jednym z zadań, które należy rozwiązać podczas projektowania automatu skończonego, jest kodowanie jego stanów. W rozdziale zaproponowano sekwencyjne metody kodowania stanów, które zmniejszają obszar i zwiększają szybkość automatów skończonych, gdy automat zaimplementowany jest w układach FPGA (*Field Programmable Gate Array*), CPLD (*Complex Programmable Logic Device*) oraz ASIC (*Application-Specific Integrated Circuit*). Metody te opierają się na sekwencyjnym wyborze stanu automatu do zakodowania i znalezieniu najbardziej odpowiedniego kodu do wybranego stanu.
Zaproponowano kilka trybów wyboru stanu i kodu. Różnica między proponowanymi metodami kodowania stanu a już znanymi metodami polega na uwzględnieniu wewnętrznych relacji pomiędzy stanami automatu (zarówno z już zakodowanymi, jak i jeszcze niezakodowanymi stanami), liczby zmiennych wejściowych wpływających na przejścia do każdego stanu oraz uwzględnieniu specyfiki architektonicznej elementu elektronicznego, w którym zaimplementowany jest automat. Eksperymenty są przeprowadzane na przykładach wzorcowych automatów skończonych. Wyniki pokazują, że proponowane podejście prowadzi do średniego zmniejszenia obszaru automatów skończonych o 19,7% (dla niektórych przykładów o dwa razy) i średniego wzrostu wydajności automatów skończonych o 21,2% (dla niektórych przykładów o 69,3%) w porównaniu z trybem Sequential programu Quartus. Wnioski wskazują obiecujące kierunki rozwoju metod kodowania stanów automatów skończonych.

Słowa kluczowe: automat skończony (*Finite State Machine* – FSM), kodowanie stanów, metoda sekwencyjna, obszar, wydajność, układ FPGA (*Field Programmable Gate Array*), układ CPLD (*Complex Programmable Logic Device* – CPLD), układ ASIC (*Application-Specific Integrated Circuit* – ASIC)

1. Wprowadzenie

W urządzeniach i systemach cyfrowych automaty skończone najczęściej pełnią funkcję kontrolujące lub sterujące. Ponieważ większość urządzeń i systemów cyfrowych ma urządzenia sterujące i jest przez nie dokładnie scharakteryzowana, to automaty skończone są w nich bardzo szeroko stosowane. Dlatego ich synteza jest istotnym zagadnieniem w każdym nowym projekcie.

Jednym z zadań, które należy rozwiązać podczas projektowania automatu skończonego, jest kodowanie jego stanów. Jest to złożony problem logiczno-kombinatoryczny, który należy do klasy problemów NP-zupełnych. Problematyka kodowania stanów automatów skończonych przyciąga uwagę naukowców i inżynierów od wielu lat (od czasu pojawienia się tej teorii). Problem ten był najintensywniej badany pod koniec lat 80. W tym czasie zaproponowano wiele metod dwupoziomowej [1] i wielopoziomowej [2] syntezy automatów skończonych, które są bezpośrednio związane z kodowaniem ich stanów. Metody syntezy dwupoziomowej mają na celu implementację automatów skończonych w urządzeniach o strukturze dwóch programowalnych macierzy, które obejmują układy CPLD. Metody syntezy wielopoziomowej implementują automaty skończone w urządzeniach z funkcjonalnym generatorem LUT (*Look-Up Table*), które obejmują układy FPGA. Ostatnio automaty skończone są coraz częściej implementowane w układach FPGA, dlatego w niniejszym rozdziale rozważono kodowanie stanów automatów skończonych w FPGA.

Ponieważ problem kodowania stanów automatów skończonych jest NP-zupełny, to jego dokładne rozwiązanie jest możliwe tylko dla małych lub specyficznych automatów skończonych, takich jak liczniki. W celu rozwiązania tego problemu opracowano wiele algorytmów, które można podzielić na heurystyczne (deterministyczne) i genetyczne (ewolucyjne). Algorytmy heurystyczne pozwalają na znalezienie lokalnego minimum (lub maksimum) pewnego parametru automatu skończonego w praktycznie akceptowalnym czasie. Dają one dość dobre rozwiązania, które mogą się zbliżać do rozwiązania optymalnego. Dlatego też algorytmy heurystyczne są często wykorzystywane w praktyce. Jednak w niektórych przypadkach ich wyniki mogą znacznie się różnić od rozwiązania optymalnego. Główną wadą tych algorytmów jest trudność z wyjściem z lokalnych minimów (maksimów).

Tradycyjnym podejściem do rozwiązywania problemów NP-zupełnych jest wykorzystanie algorytmów genetycznych lub ewolucyjnych. Pozwalają one na wyjście z lokalnych minimów, rozszerzając tym samym przestrzeń poszukiwań rozwiązania. Algorytmy genetyczne są jednak kosztowne obliczeniowo, wymagają długiego czasu i nie zawsze prowadzą do optymalnych rozwiązań. Dlatego w praktyce wykorzystuje się je do kodowania stanów kluczowych automatów skończonych, a sukces dużych i złożonych projektów zależy od optymalizacji ich parametrów.

Znalezienie optymalnego rozwiązania problemu kodowania stanów nie gwarantuje jednak optymalnego rozwiązania problemu syntezy automatów skończonych. Chodzi o to, że metody tej syntezy oprócz kodowania stanów obejmują rozwiązywanie innych problemów, np. minimalizację, dekompozycję i faktoryzację systemu

funkcji boolowskich (który reprezentuje kombinacyjną część automatu skończonego). Kodowanie stanów wpływa głównie na realizację funkcji przejścia. Jednak metody syntezy mogą łączyć realizację funkcji przejścia i funkcji wyjścia, która wymaga zupełnie innych metod kodowania stanów.

Celem niniejszej pracy jest opracowanie sekwencyjnych metod kodowania stanów dla automatów skończonych, które są wydajne pod względem obszaru (kosztu implementacji) i wydajności (szybkości) oraz koncentrują się na implementacji automatów skończonych w CPLD, FPGA lub ASIC. Różnica między proponowanymi metodami kodowania stanu a już znanymi polega na:

- uwzględnieniu wewnętrznych relacji pomiędzy stanami automatu (zarówno z już zakodowanymi, jak i jeszcze niezakodowanymi stanami);
- uwzględnieniu liczby zmiennych wejściowych, wpływających na przejścia do każdego stanu;
- uwzględnieniu specyfiki architektonicznej elementu elektronicznego, w którym zaimplementowany jest automat.

2. Powiązane prace

Przegląd algorytmów i metod kodowania stanów automatów skończonych znajduje się w [3]. W niniejszym rozdziale dokonano zaś przeglądu publikacji z ostatnich dziesięcioleci dotyczących kodowania stanów automatów skończonych.

Wykorzystanie algorytmów ewolucyjnych (genetycznych) do kodowania stanów automatów skończonych jest badane w wielu pracach. W [4] do optymalizacji wykorzystywana jest metoda wyszukiwania kukułek (*cuckoo search*), która jest porównywana z binarnym algorytmem optymalizacji roju cząstek (*particle swarm*), algorytmem genetycznym oraz algorytmami deterministycznymi NOVA [5] i JEDI [6]. W [7] zaproponowano tradycyjną algebrę Boole'a oraz logikę Reeda–Mullera w celu zminimalizowania obszaru automatu skończonego. W [8] rozważany jest algorytm ewolucyjny o nazwie GP-ES, który wykazuje dobre wyniki w projektowaniu małych i średnich automatów skończonych. W [9] zaproponowano wykorzystanie prawdopodobieństw dla każdej pary podstawień kodowych. W [10] badana jest wielopopulacyjna strategia ewolucji (oznaczana jako MPES), która rozwiązuje problem przy użyciu wewnętrznej i zewnętrznej strategii ewolucji.

W [11] do zminimalizowania obszaru automatów skończonych Moore'a zaproponowano reprezentowanie kodów stanów jako konkatenacji pseudorównoważnych kodów klas stanów i zbiorów funkcji wyjściowych. Podobne podejście do kodowania stanów automatów skończonych Mealy'ego zaproponowano w [12].

W [13] do kodowania stanów zaproponowano wartości zmiennych wyjściowych, natomiast w [14] wykorzystano do tego celu wartości zmiennych wejściowych automatów skończonych. W [15] do używania wartości zmiennych wyjściowych jako kodów stanów zastosowano multipleksery.

W [16] zaproponowano kodowanie anty-ghono (*anti-racing coding*) stanów automatów skończonych do projektowania systemów odpornych.

Wiele prac poświęconych jest kodowaniu stanów automatów skończonych, co ma na celu ochronę przed niepożądanymi atakami. W [17] przedstawiono metodę bezpiecznego kodowania stanów (*secure state encoding*) automatów skończonych w celu ochrony przed atakami analizy mocy (*against power analysis attacks*). W [18] opisano metodę kodowania stanów automatów skończonych, która ma chronić przed atakami typu *Laser Fault Injection* (LFI). W [19] przedstawiono CAD do kodowania stanów automatów skończonych odpornych na wiele modeli LFI. W [20] rozważono analizę mocy różnicowej (*Differential Power Analysis* – DPA) i ataki typu FIA (*Fault Injection Attacks*) w automatach skończonych. Zaproponowano tu metodę kodowania stanu automatu skończonego, która zachowuje odległość Hamminga w celu obrony przed atakami DPA i FIA.

Wiele prac poświęconych jest kodowaniu stanów automatów skończonych dążących do zminimalizowania zużycia energii. W [21] przedstawiono przegląd znanych metod. W [22] zaproponowano sekwencyjny, a w [23] iteracyjny algorytm kodowania stanów automatów skończonych, żeby zminimalizować dynamiczne zużycie energii. W [24] zaproponowano zmianę długości kodów stanów w tym samym celu. W [25] przedstawiono metodę kodowania stanów, by zminimalizować zużycie energii poprzez binarną optymalizację roju cząstek i wybór flip-flopów (*binary particle swarm optimization and flip-flop selection*).

Kilka prac poświęconych jest projektowaniu automatów skończonych odpornych na błędy. W [26] zaproponowano heurystyczny algorytm kodowania stanu w celu zmniejszenia wpływu usterek przejściowych (*transient faults*). W [27] zaproponowano schemat kodowania stanów automatów skończonych za pomocą liczb stochastycznych.

Pomimo dużej liczby prac poświęconych kodowaniu stanów automatów skończonych nie ma badań nad takimi metodami, które są zorientowane na nowoczesną bazę elementów CPLD, FPGA i ASIC.

3. Sekwencyjny algorytm kodowania stanów automatów skończonych

Istotą sekwencyjnej metody kodowania stanów jest to, że stany automatu są wybierane sekwencyjnie w pewnej kolejności, a do wybranego stanu przypisywany jest określony kod. Poniższe hipotezy stanowią podstawę proponowanego sekwencyjnego algorytmu kodowania stanów automatów skończonych:

- **Hipoteza 1.** Wyniki kodowania stanów automatu skończonego zależą od kolejności, w jakiej kody są przypisywane do stanów.
- **Hipoteza 2.** Stany najbardziej powiązane ze stanami już zakodowanymi powinny być kodowane jako pierwsze.

Niech S = $\{s_0,\ldots,s_{M-1}\}$ będzie zbiorem stanów automatu skończonego, K = $\{k_0,\ldots,k_{Q-1}\}$ zbiorem wszystkich dostępnych kodów, gdzie M jest liczbą stanów automatu skończonego, Q jest liczbą wszystkich dostępnych kodów, s_0 jest początkowym stanem automatu skończonego. Niech $Q = 2^R$, gdzie R = $\lceil log_2 M \rceil$, $\lceil A \rceil$ jest najmniejszą liczbą całkowitą większą lub równą A.

Sekwencyjny algorytm kodowania stanów automatów skończonych można przedstawić w postaci następującego pseudokodu.

Algorytm 1.

1. Przypisz s_0 <= k_0, kod zerowy k_0 jest przypisany do stanu początkowego s_0; załóżmy S **:=** S \ { s_0 }, K **:=** K \ { k_0 }.
2. **While** (S ≠ Ø, gdzie Ø jest zbiorem pustym) **do**
3. Zgodnie z podanym trybem wyboru stanu w zbiorze S stan s_i jest wybierany do kodowania.
4. Zgodnie z algorytmem 2 ze zbioru K wybierany jest kod k_j, który zostanie przypisany do stanu s_i.
5. Przypisanie s_i <= k_j, kod k_j jest przypisywany do stanu s_i; S **:=** S \ { s_i }, K **:=** K \ { k_j }.
6. **end while**.

Najważniejszymi krokami w algorytmie 1 są wybór stanu s_i do zakodowania (krok 3) i znalezienie kodu k_j ze zbioru K, który jest najlepszym kodem przypisanym do stanu s_i (krok 4). Z kolei przy wyborze kodu k_j ważnym krokiem jest obliczenie wzrostu obszaru automatu w wyniku przypisania kodu k_j do stanu s_i.

3.1. Wybór stanu do zakodowania

Niech $A(s_i)$ będzie zbiorem stanów, w których kończą się przejścia ze stanu s_i; $B(s_i)$ będzie zbiorem stanów, których przejścia kończą się w stanie s_i; $t(s_i,s_j)$ będzie przejściem ze stanu s_i do stanu s_j ($s_i, s_j \in$ S). Zbiór przejść $\mathbf{P}(s_i)$ ze stanu s_i jest zdefiniowany przez wyrażenie (1):

$$P(s_i) = \left\{ t(s_i, s_j) \mid s_j \in A(s_i) \right\} \tag{1}$$

a zbiór $C(s_i)$ przejść do stanu s_i jest zdefiniowany przez wyrażenie (2):

$$C(s_i) = \left\{ t(s_j, s_i) \mid s_j \in B(s_i) \right\} \tag{2}$$

Niech ES (*Encoded States*) będzie zbiorem już zakodowanych stanów. Następnie zbiór $P_{ES}(s_i)$ przejść ze stanu s_i, które kończą się w stanach zbioru ES, jest zdefiniowany przez wyrażenie (3):

$$P_{ES}\left(s_i\right)=\left\{t\left(s_i,s_j\right)\mid s_j\in ES\right\} \tag{3}$$

a zbiór przejść $C_{ES}(s_i)$ ze stanów zbioru ES, które kończą się w stanie s_i, jest zdefiniowany przez wyrażenie (4):

$$C_{ES}\left(s_i\right)=\left\{t\left(s_i,s_j\right)\mid s_j\in ES\right\} \tag{4}$$

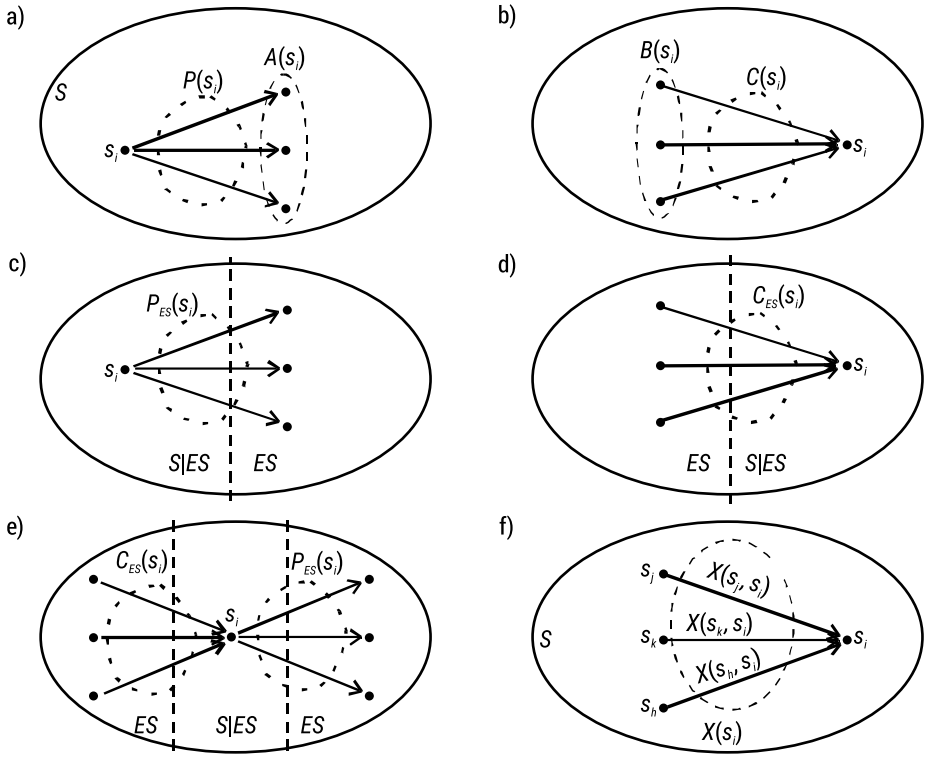
Niech $X(s_j,s_i)$ będzie zbiorem zmiennych wejściowych automatu skończonego, które inicjują przejście $t(s_j,s_i)$. Wtedy zbiór $X(s_i)$ zmiennych wejściowych wpływających na przejścia stanu s_i jest zdefiniowany przez wyrażenie (5):

$$X\left(s_i\right)=U_{s_j\in B(s_i)}X\left(s_j,s_i\right) \tag{5}$$

Strategia wyboru następnego stanu s_i do kodowania jest zdefiniowana przez następujące tryby:

- max_P, gdy wybierany jest stan s_i o maksymalnej wartości $\left|P\left(s_i\right)\right|$, gdzie |A| jest potęgą zbioru A (rys. 1a);
- max_C, gdy wybrany jest stan s_i z maksymalną wartością $\left|C\left(s_i\right)\right|$ (rys. 1b);
- P, gdy wybrany jest stan s_i z maksymalną wartością $\left|P_{ES}\left(s_i\right)\right|$ (rys. 1c);
- C, gdy wybrany jest stan s_i z maksymalną wartością $\left|C_{ES}\left(s_i\right)\right|$ (rys. 1d);
- P_C, gdy wybrany jest stan s_i z maksymalną wartością $\left|P_{ES}\left(s_i\right)\right|+\left|C_{ES}\left(s_i\right)\right|$ (rys. 1e);
- max_X, gdy wybrany jest stan s_i z maksymalną wartością $\left|X\left(s_i\right)\right|$ (rys. 1f).

Należy zauważyć, że tryby P, C i P_C uwzględniają relację wybranego stanu s_i z już zakodowanymi stanami zbioru ES, podczas gdy tryby max_C, max_P i max_X nie zależą od kolejności kodowania stanów.



RYS. 1. Wybór stanu s_i do kodowania: a) w trybie max_P; b) w trybie max_C; c) w trybie P; d) w trybie C; e) w trybie P_C; f) w trybie max_X

ŹRÓDŁO: opracowanie własne.

3.2. Wybór kodu do zakodowania następnego stanu

Wybór kodu k_j do zakodowania wybranego stanu s_i można przedstawić w postaci następującego algorytmu.

Algorytm 2.

1. Przyjmij min_cost := 100 000.
2. **for** (k_t dla wszystkich elementów zbioru K) **do**
3. Wykonywane jest próbne przypisanie kodu k_t do stanu s_i: $s_i <= k_t$.
4. Zgodnie z zadanym trybem wyboru kodu obliczana jest wartość obszaru automatu cost(k_t) w wyniku przypisania kodu k_t do stanu s_i.
5. **If** (cost(k_t) < min_cost) **do** min_cost := cost(k_t); $k_j := k_t$; **end if**.
6. Kod k_t zostaje odłączony od stanu s_i: $s_i => k_t$.
7. **end for**.
8. **return** (k_j).

Zauważmy, że w algorytmie 2 najważniejszy jest punkt 4, w którym obliczany jest obszar (koszt realizacji) funkcji przejścia automatu skończonego przy przypisaniu kodu k_t do stanu s_i.

Wcześniej eksperymentalnie ustalono, że kodowanie stanów wpływa tylko na koszt realizacji funkcji przejścia, nie wpływa zaś na koszt realizacji funkcji wyjścia. Dlatego w kroku 4 algorytmu 2 obszar jest obliczany tylko dla funkcji przejścia automatu skończonego.

3.3. Obliczanie obszaru automatu skończonego

W kroku 4 algorytmu 2 proponowane są następujące tryby obliczania obszaru dla funkcji przejścia: CPLD, FPGA i ASIC.

Niech d_r będzie pewną funkcją przejścia automatu skończonego. Gdy automat skończony jest zaimplementowany w FPGA opartym na LUT, obszar funkcji d_r jest określany za pomocą wyrażenia (6):

$$C_{FPGA}\left(d_r\right)=\left\{\begin{array}{c} 1\ przy\ q\le n \\ \dfrac{q-n}{n-1}+1\ przy\ q>n \end{array}\right\} \tag{6}$$

gdzie q jest rangą (liczbą argumentów) funkcji d_r, n jest liczbą wejść LUT.

Gdy automat skończony jest zaimplementowany w innych elementach elektronicznych, obszar można obliczyć inaczej. Na przykład, gdy automat jest zaimplementowany w układzie CPLD, obszar $C_{CPLD}(d_r)$ funkcji d_r jest równy liczbie sum w alternatywnej postaci normalnej (APN) funkcji logicznej d_r:

$$C_{CPLD}\left(d_r\right)=H\left(d_r\right) \tag{7}$$

gdzie $H(d_r)$ jest liczbą koniunkcji (mintermów) w APN funkcji d_r.

Podobnie, gdy automat jest zaimplementowany w ASIC, obszar $C_{ASIC}(d_r)$ funkcji d_r jest określony przez liczbę literałów w APN funkcji d_r, tj.

$$C_{ASIC}\left(d_r\right)=\sum_{i=1}^{H(d_r)}\left(X_i\left(d_r\right)+R\right)+H\left(d_r\right) \tag{8}$$

gdzie $X_i(d_r)$ jest liczbą zmiennych wejściowych zawartych w i-tej koniunkcji APN funkcji d_r; R jest liczbą zmiennych zwrotnych zawartych w i-tej koniunkcji APN funkcji d_r.

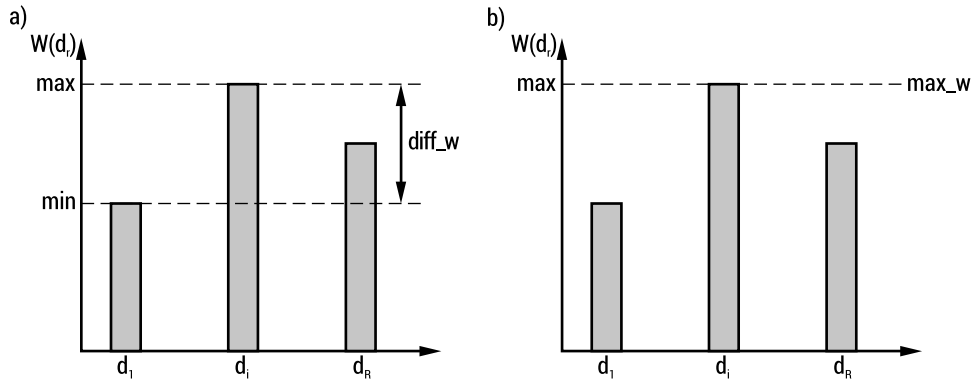
Podczas implementacji automatu skończonego w FPGA całkowity obszar automatu skończonego jest określany zgodnie z wyrażeniem (9):

$$C_{FSM}=\sum_{r=1}^{R}C_{FPGA}\left(d_r\right) \tag{9}$$

Należy zauważyć, że rzeczywisty obszar funkcji zależy od metod i algorytmów zastosowanych w narzędziu projektowym: metod syntezy logicznej, metod optymalizacji syntezy logicznej i fizycznej, metod mapowania logiki projektu na fizyczną strukturę elementu elektronicznego, metod przestrzegania ograniczeń czasowych projektu itp. Dlatego obszar obliczony według wzorów (6)–(9) często różni się od rzeczywistego obszaru zrealizowanego automatu skończonego.

3.4. Inne tryby wyboru kodu

Oprócz rozważanych trybów wyboru kodu zorientowanych na realizację automatów skończonych w określonym typie elementów elektronicznych proponowane są jeszcze dwa tryby – diff_w i max_w. W trybie diff_w minimalizowana jest różnica wag $W(d_r)$ funkcji przejścia (rys. 2a), a w trybie max_w minimalizowana jest maksymalna waga $W(d_r)$ funkcji przejścia (rys. 2b). Waga $W(d_r)$ każdej funkcji d_r jest równa liczbie sum w APN funkcji d_r. Tryb diff_w pozwala uśrednić koszt realizacji wszystkich funkcji przejścia, tj. zmniejszyć różnicę między najbardziej złożoną a najprostszą funkcją. Tryb max_w minimalizuje koszt implementacji najbardziej złożonej funkcji, czyli zwiększa szybkość działania automatu skończonego.



RYS. 2. Tryby wyboru kodu diff_w (a) i max_w (b)

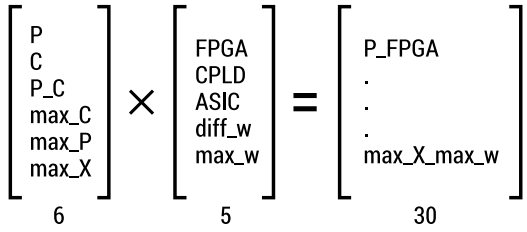
ŹRÓDŁO: opracowanie własne.

Można postawić następujące hipotezy:

- **Hipoteza 3.** Tryb diff_w pomaga zminimalizować koszt implementacji automatów skończonych.
- **Hipoteza 4.** Tryb max_w przyczynia się do zwiększenia wydajności automatów skończonych.

3.5. Budowanie sekwencyjnych metod kodowania stanów

Na podstawie algorytmów 1 i 2, a także proponowanych trybów wyboru stanu (6 trybów) i trybów wyboru kodu (5 trybów) można skonstruować 30 różnych sekwencyjnych metod kodowania stanu dla automatów skończonych (rys. 3).



RYS. 3. Budowa sekwencyjnych metod kodowania stanów

ŹRÓDŁO: opracowanie własne.

Tryby wyboru stanu (P, C, P_C, max_C, max_P i max_X) pozwalają na uwzględnienie złożonych zależności pomiędzy stanami automatu skończonego. Z kolei tryby wyboru kodu (FPGA, CPLD i ASIC) pozwalają uwzględnić cechy architektoniczne komponentów elektronicznych, a także cele optymalizacyjne – minimalizację obszaru (tryb diff_w) lub zwiększenie szybkości (tryb max_w) automatu skończonego.

Skonstruowane metody będą oznaczane jedną nazwą, łączącą tryb wyboru stanu i tryb wyboru kodu znakiem podkreślenia. Przykładowo w metodzie P_FPGA wybór stanów do kodowania odbywa się w trybie P, a koszt realizacji funkcji przejścia obliczany jest zgodnie z trybem FPGA.

4. Wyniki eksperymentalne

Badanie sekwencyjnych metod kodowania stanów automatów skończonych zostało przeprowadzone przy użyciu narzędzia Quartus 23.1 firmy Intel podczas implementacji przykładów wzorcowych centrum MCMC (Microelectronics Center of North Carolina) [28] w rodzinie układów FPGA Cyclone 10 LP. Wstępne badania eksperymentalne wykazały, że metody zbudowane przy użyciu trybu max_P nie działały dobrze, więc nie będą dalej rozważane.

Wyniki eksperymentalne podsumowano w tabelach 1 i 2, gdzie SS_mode (State Selection mode) to tryb wyboru stanu (P, C, P_C, max_C lub max_X); CS_mode (Code Selection mode) to tryb wyboru kodu (FPGA, CPLD, diff_w, max_w); Ls to wartość obszaru (liczba elementów logicznych FPGA), Fs to wartość wydajności (maksymalna częstotliwość pracy w MHz) przy użyciu trybu Sequential systemu Quartus; Lmin – minimalna wartość obszaru dla tego przykładu; Fmax – maksymalna wartość wydajności dla tego przykładu; Ls/Lmin i Fmax/Fs – stosunki odpowiednich parametrów; Best – liczba najlepszych rozwiązań; Unique – liczba unikatowych rozwiązań; Av – średnia arytmetyczna wartość parametru; Max – maksymalna wartość parametru.

TABELA 1. Obszar automatów skończonych przy użyciu sekwencyjnych metod kodowania stanu

SS_mode	P	P	P	P	C	C	C	C	P_C	P_C	P_C	P_C	max_C	max_C	max_C	max_C	max_X	Ls	Lmin	Ls/ Lmin
CS_mode	FPGA	CPLD	diff_w	max_w	FPGA	CPLD	diff_w	max_w	FPGA	CPLD	diff_w	max_w	FPGA	CPLD	diff_w	max_w	FPGA			
bbara	31	31	33	24	30	27	37	29	31	**22**	36	28	26	23	29	30	28	24	**22**	1,091
bbsse	58	60	65	62	58	64	62	57	61	62	65	**55**	61	64	57	62	68	60	**55**	1,091
beecount	**16**	22	19	22	**16**	22	19	22	**16**	22	19	22	17	**16**	**16**	**16**	**16**	**16**	**16**	1
dk14	51	41	40	43	37	44	46	43	47	42	56	43	41	45	42	42	45	**36**	**36**	1
*dk16	90	82	120	87	**80**	88	88	88	87	85	93	104	83	100	87	86	100	92	**80**	1,15
*dk512	22	16	15	16	16	18	15	15	15	15	17	18	**14**	15	17	17	15	15	**14**	1,071
ex2	67	75	71	76	58	72	74	67	74	61	81	71	56	62	62	64	62	**35**	**35**	1
ex3	26	28	32	29	27	23	30	26	23	23	34	28	28	23	23	26	23	**22**	**22**	1
ex5	20	24	24	26	21	21	27	20	**18**	19	22	23	22	23	23	22	23	21	**18**	1,167
ex7	25	34	27	33	22	28	33	31	20	24	41	26	20	25	25	27	25	**18**	**18**	1
lion9	**11**	15	15	18	**11**	15	15	18	**11**	15	15	18	**11**	15	18	18	15	19	**11**	1,727
pma	161	159	154	157	161	166	167	172	163	159	152	152	181	167	**139**	151	173	141	**139**	1,014
*s1	156	179	183	175	149	150	180	152	190	148	183	149	149	148	153	148	**147**	150	**147**	1,02
s208	69	73	75	71	57	73	61	57	73	69	72	72	64	64	74	49	64	**35**	**35**	1
s27	19	15	19	14	**9**	22	16	21	**9**	22	16	21	20	11	12	12	11	16	**9**	1,778
*s298	1024	1022	1063	969	969	965	1074	1000	**938**	958	1073	995	965	950	973	1008	943	1006	**938**	1,072
s386	58	60	55	53	59	56	59	57	57	50	53	53	58	56	57	55	58	**26**	**26**	1
s420	39	46	39	29	28	39	31	**19**	39	40	34	34	33	38	46	31	38	37	**19**	1,947
s820	144	133	141	130	132	125	138	126	127	127	147	134	127	127	**124**	132	134	138	**124**	1,113
shiftreg	**2**	4	3	4	4	4	4	4	4	**2**	4	**2**	**2**	4	4	4	4	4	**2**	2
sse	58	60	65	62	58	64	62	57	61	62	65	**55**	61	64	57	62	68	60	**55**	1,091
*styr	182	184	193	186	179	194	188	190	186	187	187	189	179	185	**177**	190	184	213	**177**	1,203
train11	28	28	29	27	28	**25**	26	31	28	**25**	26	31	28	**25**	31	31	**25**	**25**	**25**	1
Best	**3**				**4**	**1**		**1**	**5**	**3**		**3**	**3**	**2**	**4**	**1**	**3**	**8**		
Unique					**1**			**1**	**1**	**1**		**2**	**1**		**3**		**1**	**6**		
Av																				**1,197**
Max																				**2**

ŹRÓDŁO: opracowanie własne.

TABELA 2. Wydajność automatów skończonych wykorzystujących sekwencyjne metody kodowania stanu

SS_mode	P	P	P	P	C	C	C	C	P_C	P_C	P_C	P_C	max_C	max_C	max_C	max_C	max_X	Fs	Fmax	Fmax/ Fs
CS_mode	FPGA	CPLD	diff_w	max_w	FPGA	CPLD	diff_w	max_w	FPGA	CPLD	diff_w	max_w	FPGA	CPLD	diff_w	max_w	FPGA			
bbara	266	292	226	350	262	**396**	262	366	281	313	228	239	296	339	282	325	235	293	**396**	1,352
bbsse	215	184	199	217	221	186	185	195	182	168	187	208	184	213	**236**	218	190	212	**236**	1,113
beecount	326	273	318	273	326	273	318	273	326	273	318	273	**346**	344	344	344	344	317	**346**	1,091
dk14	251	222	232	249	217	261	248	268	246	220	212	**278**	225	235	262	262	235	249	**278**	1,116
*dk16	179	196	162	**212**	197	191	177	202	188	192	197	164	195	194	202	204	194	201	**212**	1,055
*dk512	201	314	324	337	321	313	328	347	320	350	282	**369**	325	339	330	310	339	329	**369**	1,122
ex2	160	183	169	190	206	209	172	177	179	184	172	181	172	**215**	208	164	215	152	**215**	1,414
ex3	269	250	165	172	199	181	199	191	226	253	191	185	159	212	179	**305**	212	200	**305**	1,525
ex5	310	244	280	193	214	242	228	282	256	300	244	303	232	220	**323**	209	220	249	**323**	1,297
ex7	260	210	**288**	163	236	212	202	238	259	201	188	242	275	252	221	208	252	248	**288**	1,161
lion9	389	**467**	428	373	389	**467**	428	373	389	**467**	428	373	389	**467**	373	373	**467**	294	**467**	1,588
pma	189	168	147	150	173	131	155	162	163	170	247	**276**	151	157	168	165	157	163	**276**	1,693
*s1	154	153	149	156	143	148	151	156	139	163	152	148	164	163	150	**176**	165	161	**176**	1,093
s208	222	218	169	172	253	213	191	254	219	200	191	213	230	238	222	**259**	238	253	**259**	1,024
s27	227	312	245	273	**389**	275	276	247	**389**	275	276	247	257	301	324	324	301	363	**389**	1,072
*s298	109	105	111	109	105	111	104	109	110	108	107	110	108	104	107	110	109	**113**	**113**	1
s386	196	189	192	211	215	205	217	221	185	203	226	**248**	197	223	226	212	167	185	**248**	1,341
s420	258	226	221	242	368	294	264	**373**	243	225	287	216	334	324	283	274	324	280	**373**	1,332
s820	170	167	160	174	166	**184**	141	161	156	160	159	164	178	166	162	159	156	165	**184**	1,115
shiftreg	847	935	611	935	924	926	938	926	926	847	**943**	847	847	935	935	935	935	935	**943**	1,009
sse	215	184	199	217	221	186	185	195	182	168	187	208	184	213	**236**	218	190	212	**236**	1,113
*styr	151	153	158	157	141	137	142	138	159	**170**	164	158	152	158	160	152	147	160	**170**	1,063
train11	212	206	179	**287**	184	216	260	241	184	216	260	241	184	216	241	241	216	244	**287**	1,176
Best		**1**	**1**	**2**	**1**	**3**		**1**	**1**	**2**	**1**	**4**	**1**	**2**	**3**	**3**	**1**	**1**		
Unique		**1**	**1**	**2**		**2**		**1**		**1**	**1**	**4**	**1**	**1**	**3**	**3**		**1**		
Av																				**1,212**
Max																				**1,693**

ŹRÓDŁO: opracowanie własne.

Sequential to tryb kodowania stanów automatów skończonych udostępniany przez system Quartus. W trybie Sequential stany automatu skończonego są kodowane sekwencyjnie za pomocą kodów binarnych o minimalnej długości. Tryb ten może być traktowany jako rozwiązanie losowe i jest wykorzystywany do porównywania wyników uzyskanych przez proponowane metody kodowania stanów.

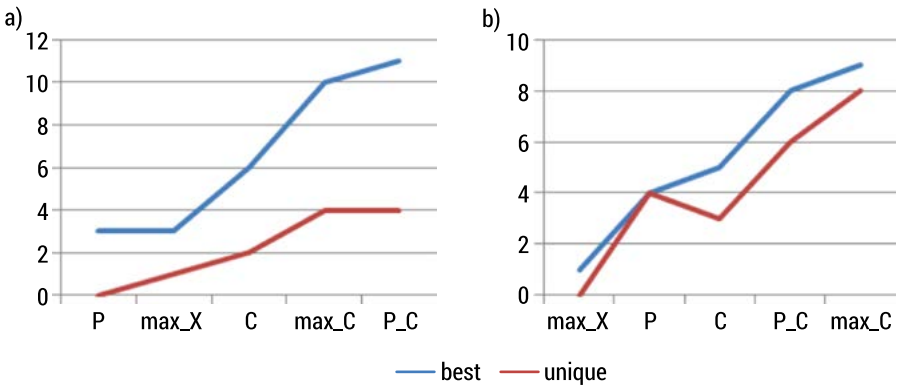
Dla rozważanych przykładów automatów skończonych zaimplementowanych w układach FPGA wyniki dla trybów obliczania kosztów implementacji CPLD i ASIC są takie same. Dlatego w tabelach 1–2 przedstawiono wyniki tylko dla CPLD. Tryb wyboru stanu max_X został zbadany jedynie w trybie wyboru kodu FPGA.

5. Dyskusje i rozważania

Analiza tabel 1–2 pokazuje, że w porównaniu z trybem sekwencyjnym rozważane metody kodowania stanów automatów zmniejszają obszar średnio o 19,7%, a dla niektórych przykładów dwukrotnie (przykład shiftreg), oraz zwiększają wydajność automatów średnio o 21,2%, a dla niektórych przykładów o 69,3% (przykład pma).

W tabelach 1–2 najlepsze rozwiązania zostały wyróżnione pogrubioną czcionką. Należy zauważyć, że można je uzyskać przy użyciu kilku metod kodowania. Dla niektórych metod najlepsze rozwiązanie jest unikatowe, tj. nie jest osiągane przy użyciu innych metod kodowania.

Na rys. 4 przedstawiono tryby wyboru stanu w zależności od liczby najlepszych i unikatowych rozwiązań pod względem obszaru i wydajności.



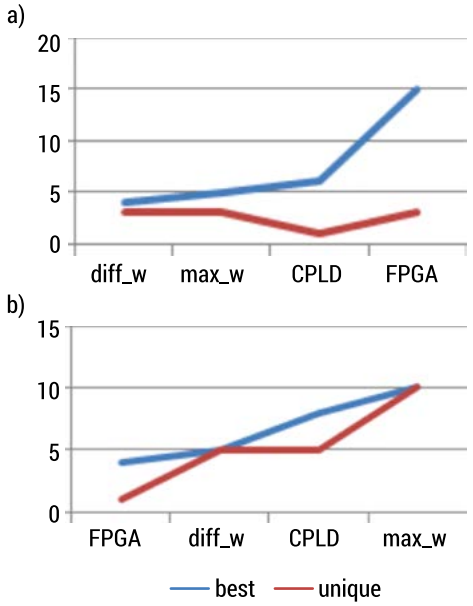
RYS. 4. Liczba najlepszych i unikatowych rozwiązań dla trybów wyboru stanu: a) według obszaru, b) według wydajności

ŹRÓDŁO: opracowanie własne.

Najlepszym trybem wyboru stanu według obszaru jest tryb P_C, a następnie tryb max_C (rys. 4a), a najlepszym trybem wyboru stanu według wydajności jest tryb max_C, a następnie tryb P_C (rys. 4b).

Wnioski. Najlepszego rozwiązania (zarówno pod względem obszaru, jak i wydajności) należy szukać przy użyciu trybów P_C i max_C.

Na rys. 5 przedstawiono wykresy trybów wyboru kodu w zależności od liczby najlepszych i unikatowych rozwiązań pod względem obszaru i wydajności.



RYS. 5. Liczba najlepszych i unikatowych rozwiązań dla trybów wyboru kodu: a) według obszaru, b) według wydajności

ŹRÓDŁO: opracowanie własne.

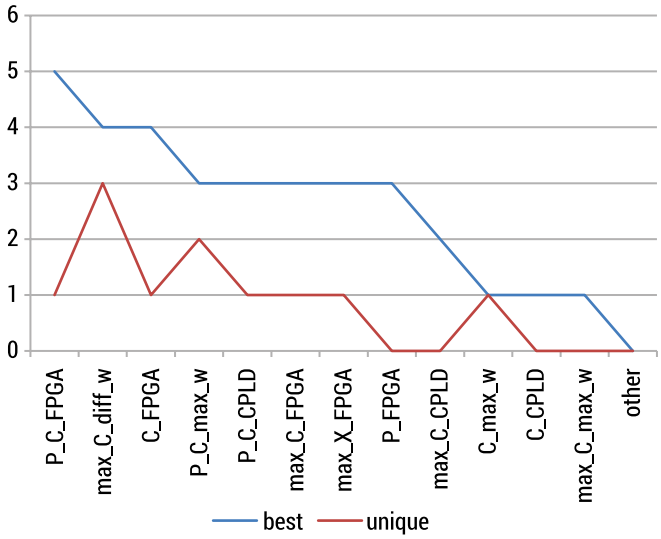
Tryb FPGA jest najlepszym trybem wyboru kodu pod względem obszaru, a tryb CPLD jest drugim najlepszym pod względem wydajności (rys. 5a). Jednak tryb FPGA jest najgorszym trybem pod względem wydajności (rys. 5b). Zgodnie z oczekiwaniami najlepszym trybem wyboru kodu pod względem wydajności jest tryb max_w, a drugim najlepszym pod względem wydajności jest tryb CPLD (tak samo jak w przypadku obszaru).

Wnioski. Jeśli ma zostać zoptymalizowany obszar, to należy użyć trybów wyboru kodu FPGA i CPLD. Jeśli zoptymalizowana ma zostać wydajność, należy użyć trybów wyboru kodu max_w i CPLD. Jeśli optymalizowane mają być zarówno obszar, jak i wydajność, to należy użyć trybu wyboru kodu CPLD.

Na rys. 6 pokazano liczbę najlepszych i unikatowych rozwiązań według obszaru, które zostały osiągnięte za pomocą zbudowanych sekwencyjnych metod kodowania stanu automatów skończonych.

Z rys. 6 wynika następująca kolejność najlepszych metod kodowania stanów ze względu na obszar:

1. P_C_FPGA;
2. max_C_diff_w;
3. C_FPGA;
4. P_C_max_w;
5. max_C_FPGA;
6. max_X_FPGA;
7. P_FPGA;
8. max_C_CPLD;
9. C_max_w;
10. C_CPLD;
11. max_C_max_w … .



RYS. 6. Liczba najlepszych i unikatowych w odniesieniu do obszaru rozwiązań uzyskanych za pomocą sekwencyjnych metod kodowania stanu

ŹRÓDŁO: opracowanie własne.

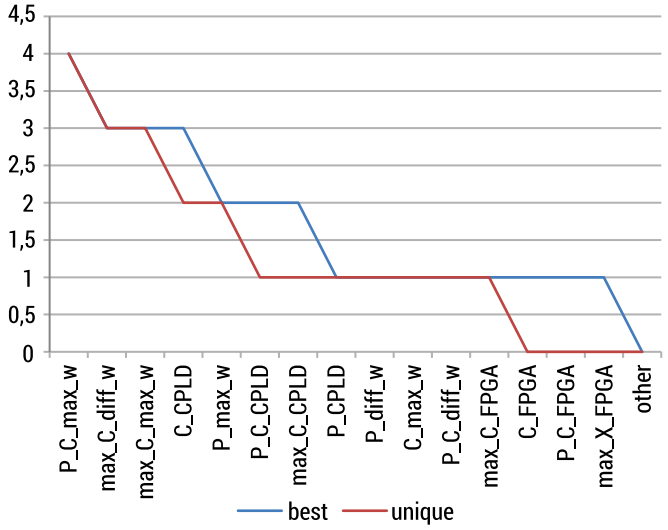
Należy zauważyć, że w najlepszych metodach ze względu na obszar wśród trybów wyboru stanu (oprócz trybów P_C i max_C) znaleziono również tryby C, max_X i P, a wśród trybów wyboru kodu (oprócz trybów FPGA i CPLD) znaleziono tryby diff_w i max_w.

Wnioski. Aby znaleźć najlepsze rozwiązanie obszarowe (oprócz zalecanych trybów P_C, max_C, FPGA i CPLD), należy również rozważyć tryby wyboru stanu C, max_X i P oraz tryby wyboru kodu diff_w i max_w.

Na rys. 7 pokazano liczbę najlepszych i unikatowych rozwiązań według wydajności, które zostały osiągnięte za pomocą zbudowanych sekwencyjnych metod kodowania stanu automatów skończonych.

Z rys. 7 wynika następująca kolejność najlepszych metod kodowania stanów ze względu na wydajność:

1. P_C_max_w;
2. max_C_diff_w;
3. max_C_max_w;
4. C_CPLD;
5. P_max_w;
6. P_C_CPLD;
7. max_C_CPLD;
8. P_CPLD;
9. P_diff_w;
10. C_max_w;
11. P_C_diff_w;
12. max_C_FPGA;
13. C_FPGA;
14. P_C_FPGA;
15. max_X_FPGA … .



RYS. 7. Liczba najlepszych i unikatowych rozwiązań uzyskanych przez sekwencyjne metody kodowania stanu w odniesieniu do wydajności

ŹRÓDŁO: opracowanie własne.

Należy zauważyć, że w najlepszych metodach pod względem wydajności wśród trybów wyboru stanu (oprócz trybów P_C i max_C) znaleziono również tryby C, P

i max_X, a wśród trybów wyboru kodu (oprócz trybów max_w i CPLD) znaleziono również tryby diff_w i FPGA.

Wnioski. Aby znaleźć najlepsze rozwiązanie pod względem wydajności (oprócz zalecanych trybów P_C, max_C, max_w i CPLD), należy również rozważyć tryby wyboru stanu C, P i max_X, a także tryby wyboru kodu diff_w i FPGA.

Chociaż tryb diff_w nie należy do najlepszych trybów wyboru kodu (zarówno pod względem obszaru, jak i wydajności), jest często spotykany w metodach kodowania stanów, które osiągają lepsze rozwiązania w zakresie obszaru i wydajności.

6. Zalecenia do zastosowań praktycznych

Rozważane metody sekwencyjnego kodowania stanów automatów skończonych zostały zaimplementowane w postaci programu *sequential_state_encoding*, który jest częścią pakietu uniwersyteckiego ZUBR do projektowania urządzeń cyfrowych [29].

Do kodowania stanów projektowanego automatu skończonego wykorzystywane są zwykłe metody narzędzia projektowego (np. w systemie Quartus są to: Sequential, Gray, Johnson, Minimal Bits i One-Hot). Jeśli nie zostanie znalezione odpowiednie rozwiązanie, to należy użyć programu *sequential_state_encoding*. W tym celu konieczne jest określenie sekwencyjnej metody kodowania stanu, która jest określana przez tryb wyboru stanu i tryb wyboru kodu.

Tryb wyboru stanu pozwala nam wziąć pod uwagę trudne do wykrycia połączenia między stanami automatu skończonego:

- tryb max_P wybiera stan, w którym rozpoczyna się maksymalna liczba przejść;
- tryb max_C wybiera stan, w którym kończy się maksymalna liczba przejść;
- tryb P wybiera stan, z którego wykonywana jest maksymalna liczba przejść do już zakodowanych stanów;
- tryb C wybiera stan, w którym kończy się maksymalna liczba przejść ze stanów już zakodowanych;
- tryb P_C wybiera stan, który ma maksymalną liczbę połączeń ze stanami już zakodowanymi;
- tryb max_X wybiera stan, na którego przejścia wpływa maksymalna liczba zmiennych wejściowych.

Tryb wyboru kodu pozwala na uwzględnienie cech architektonicznych elementu elektronicznego, w którym zaimplementowany jest automat skończony. Wyraża się to w sposobie obliczania kosztu realizacji funkcji przejścia automatu skończonego:

- w trybie FPGA obszar obliczany jest według wzoru (4);
- w trybie CPLD obszar jest obliczany według wzoru (5);
- w trybie ASIC obszar jest obliczany według wzoru (6).

Ponadto tryb wyboru kodu max_w minimalizuje maksymalną wagę funkcji przejścia, a tryb diff_w minimalizuje różnicę między maksymalną i minimalną wagą funkcji przejścia. W ten sposób tryb max_w optymalizuje wydajność automatu skończonego, podczas gdy tryb diff_w optymalizuje zarówno wydajność, jak i obszar automatu skończonego.

7. Podsumowanie

W artykule omówiono podejście do kodowania stanów automatów skończonych oparte na sekwencyjnym wyborze stanu do zakodowania i znalezieniu najodpowiedniejszego kodu, który przypisany jest do wybranego stanu. Zaproponowano kilka trybów wyboru stanu i wyboru kodu. Tryby wyboru stanu pozwalają na uwzględnienie wewnętrznych relacji między stanami i zmiennymi wejściowymi automatu, podczas gdy tryby wyboru kodu uwzględniają cechy architektoniczne elementu elektronicznego, w którym zaimplementowany jest automat. Różne sekwencyjne metody kodowania stanów automatów są konstruowane poprzez połączenie trybów wyboru stanu i trybów wyboru kodu.

Postawiono następujące hipotezy:

- **Hipoteza 1.** Wyniki kodowania stanów automatu skończonego zależą od kolejności, w jakiej stanom przypisywane są kody.
- **Hipoteza 2.** Stany najbardziej powiązane z już zakodowanymi stanami powinny być kodowane jako pierwsze.
- **Hipoteza 3.** Tryb diff_w przyczynia się do minimalizacji kosztu implementacji automatów skończonych.
- **Hipoteza 4.** Tryb max_w przyczynia się do zwiększenia szybkości działania automatów skończonych.

Hipotezy 1 i 4 zostały w pełni potwierdzone. Hipoteza 2 jest tylko częściowo potwierdzona, ponieważ tryb max_C, który bierze pod uwagę stany niezwiązane z już zakodowanymi stanami, również daje dość dobre rozwiązania. Hipoteza 3 nie została potwierdzona. Jednak tryb diff_w jest często spotykany w metodach kodowania stanów, które osiągają lepsze rozwiązania pod względem obszaru i wydajności.

Badania eksperymentalne wykazały, że rozważane metody kodowania stanów dla automatów zmniejszają obszar średnio o 19,7%, a dla niektórych przykładów dwukrotnie (przykład shiftreg) i zwiększają wydajność automatów średnio o 21,2%, a dla niektórych przykładów o 69,3% (przykład pma).

Przyszłe badania nad kodowaniem stanów automatów skończonych skupią się na opracowaniu algorytmów równoległych i iteracyjnych, a także na zmianie liczby bitów w kodach stanów. Połączenie realizacji funkcji wyjściowych i kodowania stanów automatów skończonych jest również postrzegane jako obiecujący kierunek badań.

*

Badania zostały zrealizowane w ramach pracy nr WZ/WI-III/5/2023 na Politechnice Białostockiej i sfinansowane z subwencji badawczej przekazanej przez ministra właściwego do spraw nauki.

Bibliografia

[1] De Micheli G., Brayton R.K., Sangiovanni-Vincentelli A., *Optimal state assignment for finite state machines*, „IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems" 1985, vol. 4(3), s. 269–285.

[2] Devadas S., Ma H.K., Newton A.R., Sangiovanni-Vincentelli A., *MUSTANG: State assignment of finite state machines targeting multilevel logic implementations*, „IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems" 1988, vol. 7(12), s. 1290–1300.

[3] Salauyou V., Klimowicz A., *Synteza logiczna układów cyfrowych w strukturach programowalnych*, Oficyna Wydawnicza Politechniki Białostockiej, Białystok 2010.

[4] El-Maleh A.H., Sait S.M., Bala A., *State assignment for area minimization of sequential circuits based on cuckoo search optimization*, „Computers & Electrical Engineering" 2015, vol. 44, s. 13–23.

[5] Villa T., Sangiovanni-Vincentelli A., *NOVA: State assignment of finite state machines for optimal two-level logic implementations*, w: *Proceedings of the 26th ACM/IEEE Design Automation Conference*, ACM, New York 1989, s. 327–332.

[6] Lin B., Newton A.R., *Synthesis of multiple level logic from symbolic high-level description languages*, w: *VLSI 89. Proceedings of the IFIP TC 10/WG 10.5 international conference on Very large scale integration, Munich, FRG, August 16–18, 1989*, red. G. Musgrave, U. Lauther, Amsterdam 1990, s. 187–196.

[7] Lin W., Wang L., Xia Y., *FSM dual logic synthesis targeting area optimization*, w: *2015 IEEE 16th International Conference on Communication Technology* (*ICCT*), 2015, s. 821–826.

[8] Tao Y., Zhang Q., Zhang L., Zhang Y., *A systematic EHW approach to the evolutionary design of sequential circuits*, „Soft Computing" 2016, vol. 20, s. 5025–5038.

[9] El-Maleh A.H., *A probabilistic pairwise swap search state assignment algorithm for sequential circuit optimization*, „Integration" 2017, vol. 56, s. 32–43.

[10] Tao Y., Zhang L., Wang Q., Chen R., Zhang Y., *A multi-population evolution strategy and its application in low area/power FSM synthesis*, „Natural Computing" 2019, vol. 18, s. 139–161.

[11] Mielcarek K., Barkalov A., Titarenko L., *Designing Moore FSM with unstandard representation of state codes*, w: *2016 5th International Conference on Modern Circuits and Systems Technologies* (*MOCAST*), 2016, s. 1–4.

[12] Barkalov A.A., Titarenko L., Mielcarek K., *Reducing LUT Count for Mealy FSMs With Transformation of States*, „IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems" 2021, vol. 41(5), s. 1400–1411.

[13] Solov'ev V.V., *Minimization of mealy finite-state machines by using the values of the output variables for state assignment*, „Journal of Computer and Systems Sciences International" 2017, vol. 56, s. 96–104.

[14] Salauyou V., Ostapczuk M., *State assignment of finite-state machines by using the values of input variables*, w: *Computer Information Systems and Industrial Management: 16th IFIP TC8 International Conference, CISIM 2017, Bialystok, Poland, June 16–18, 2017, Proceedings 16*, Springer, Cham 2017, s. 592–603.

[15] Senhadji-Navarro R., Garcia-Vargas I., *Mapping Outputs and States Encoding Bits to Outputs Using Multiplexers in Finite State Machine Implementations*, „Electronics" 2023, vol. 12(3), 502.

[16] Bychko V., Yershov R., Gulyi Y., Zhydko M., *Automation of anti-race state encoding of asynchronous FSM for robust systems*, w: *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology* (*PIC S&T*), red. D. Ageyev, IEEE, Washington 2020, s. 501–506.

[17] Agrawal R., Vemuri R., *On state encoding against power analysis attacks for finite state controllers*, w: *2018 IEEE International Symposium on Hardware Oriented Security and Trust* (*HOST*), IEEE, Washington 2018, s. 181–186.

[18] Jayasena A., Rani K., Mishra P., *Efficient Finite State Machine Encoding for Defending Against Laser Fault Injection Attacks*, w: *2022 IEEE 40th International Conference on Computer Design* (*ICCD*), IEEE, Washington 2022, s. 247–254.

[19] Choudhury M., Gao M., Tajik S., Forte D., TAMED: *Transitional Approaches for LFI Resilient State Machine Encoding*, w: *2022 IEEE International Test Conference* (*ITC*), IEEE, Washington 2022, s. 46–55.

[20] Konganapalle G., Shukla S., Singh V., *SMASh: A State Encoding Methodology Against Attacks on Finite State Machines*, w: *2023 IEEE 32nd Asian Test Symposium* (*ATS*), IEEE, Washington 2023, s. 1–6.

[21] Salauyou V., Grzes T., *FSM state assignment methods for low-power design*, w: *6th International Conference on Computer Information Systems and Industrial Management Applications* (*CISIM'07*), IEEE, Washington 2007, s. 345–350.

[22] Grzes T.N., Solov'ev V.V., *Sequential algorithm for low-power encoding internal states of finite state machines*, „Journal of Computer and Systems Sciences International" 2014, vol. 53, s. 92–99.

[23] Solov'ev V.V., Grzes T.N., *An iteration algorithm of coding internal states of finite-state machines for minimizing the power consumption*, „Russian Microelectronics" 2013, vol. 42, s. 189–195.

[24] Solov'ev V.V., *Changes in the length of internal state codes with the aim at minimizing the power consumption of finite-state machines*, „Journal of Communications Technology and Electronics" 2012, vol. 57, s. 642–648.

[25] Khatua K., *Low Power State Assignment of Sequential Circuits based on Binary Particle Swarm Optimization and Flip-Flop Selection*, w: *2022 International Conference on Electrical, Computer and Energy Technologies* (*ICECET*), IEEE, Washington 2022, s. 1–6.

[26] Ichihara H., Fukuda M., Iwagaki T., Inoue T., *State assignment for fault tolerant stochastic computing with linear finite state machines*, w: *2017 International Test Conference in Asia* (*ITC-Asia*), IEEE, Washington 2017, s. 156–161.

[27] Ichihara H., Maeda Y., Iwagaki T., Inoue T., *State encoding with stochastic numbers for transient fault tolerant linear finite state machines*, w: *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems* (*DFT*), IEEE, Washington 2019, s. 1–6.

[28] Yang S., *Logic Synthesis and Optimization Benchmarks user Guide. Version 3.0*, Microelectronics Center of North Carolina, Durham 1991.

[29] Salauyou V., Klimowicz A., Grzes T., Bulatowa I., Dimitrowa-Grekow T., *Synthesis methods of finite state machines implemented in package ZUBR*, w: *Proceedings of the Sixth International Conference Computer-Aided Design of Discrete Devices* (*CAD DD'7*), Minsk 2007, s. 53–56.

Sequential methods for encoding finite state machines

Summary: The finite state machine (FSM) model is often used to describe the functioning of digital hardware. When optimizing digital devices, it is essential to improve the properties of FSMs, such as area (implementation cost, number of logical elements), performance (speed), and power consumption. One of the tasks that must be solved when designing an FSM is encoding its states. This paper proposes sequential state encoding methods that reduce the area and increase the speed of FSMs when the FSM is implemented in FPGA (Field Programmable Gate Array), as well as in CPLD (Complex Programmable Logic Device) and ASIC (Application-Specific Integrated Circuit) devices. These methods are based on the sequential selection of the FSM state to be encoded and finding the most suitable code for the selected state. Several modes of state and code selection are proposed. The difference between the proposed state encoding methods and the already known ones is: taking into account the internal relations between the states of the FSM (with already encoded and not yet encoded states); taking into account the number of input variables that affect the transitions to each state; taking into account the architectural specificity of the electronic component in which the machine is implemented (FPGA, CPLD, or ASIC). The experiments have been carried out on the standard FSM benchmarks. The experimental results show that the proposed approach leads to an average reduction of the area of FSMs by 19.7% (for some examples by a factor of 2) and an average increase in the performance of finite state machines by 21.2% (for some examples on 69.3%) compared to the Sequential mode of system Quartus. The conclusions indicate promising directions for the development of state encoding methods of FSMs.

Keywords: finite state machine (FSM), state encoding, sequential method, area, performance, field programmable gate array (FPGA), complex programmable logic device (CPLD), application-specific integrated circuit (ASIC)

Spis tabel

Spis rysunków

Spis wykresów

Politechnika
Białostocka