Studia i Materiały Informatyki Stosowanej

Studia i Materiały Informatyki Stosowanej

Czasopismo młodych pracowników naukowych, doktorantów i studentów

Tom 13, Nr 1, 2021

Bydgoszcz 2021

Studia i Materiały Informatyki Stosowanej
Czasopismo młodych pracowników naukowych, doktorantów i studentów

Patronat naukowy:

Instytut Informatyki
Uniwersytet Kazimierza Wielkiego
ul. Chodkiewicza 30
85-064 Bydgoszcz
tel. (052) 325 76 11
e-mail: simis@ukw.edu.pl

ISSN 1689-6300

Projekt okładki: Łukasz Zawadzki (StudioStrzelec.pl)
DTP: Dawid Ewald

Wydawca:

Instytut Informatyki
Uniwersytet Kazimierza Wielkiego
Dyrektor:
dr hab. inż. Izabela Rojek, prof. uczelni
ul. Chodkiewicza 30
85-064 Bydgoszcz
tel. +48 52 325 76 11
email: izarojek@ukw.edu.pl

Kontakt:

dr inż. Jacek Czerniak, prof. uczelni
dr hab. inż. Marek Macko, prof. uczelni
Uniwersytet Kazimierza Wielkiego
ul. Chodkiewicza 30
85-064 Bydgoszcz
e-mail: jczerniak@ukw.edu.pl
mackomar@ukw.edu.pl

Druk (ze środków sponsora):
Oficyna Wydawnicza MW

Nakład 250 egz.

Bydgoszcz 2021

Studies and Materials in Applied Computer Science

Journal of young researchers,
PhD students and students

Vol. 13, No.1, 2021

Bydgoszcz 2021

Studies and Materials in Applied Computer Science
Journal of young researchers, PhD students and students

Scientific patronage:
Institute of Informatics
Kazimierz Wielki University
ul. Chodkiewicza 30
85-064 Bydgoszcz, Poland
tel. +48 52 325 76 11
e-mail: simis@ukw.edu.pl

ISSN 1689-6300

Cover designed by: Łukasz Zawadzki (StudioStrzelec.pl)
DTP by: Dawid Ewald

Publisher:

Institute of Informatics
Kazimierz Wielki University
Head:
Izabela Rojek, PhD DSc Eng Assoc. Prof.
ul. Chodkiewicza 30
85-064 Bydgoszcz, Poland
tel. + 48 52 325 76 11
e-mail: izarojek@ukw.edu.pl

Contact:

Jacek Czerniak, PhD. Eng. Assoc. Prof.
Marek Macko, PhD. DSc. Eng., Assoc Prof
Kazimierz Wielki University
ul. Chodkiewicza 30
85-064 Bydgoszcz, Poland
e-mail: jczerniak@ukw.edu.pl
mackomar@ukw.edu.pl

Printing (funded from non-prifit programme**):**
Oficyna Wydawnicza MW

Edition of 250 copies

Bydgoszcz 2021

Studia i Materiały Informatyki Stosowanej

czasopismo młodych pracowników naukowych, doktorantów i studentów

patronat: Polskie Towarzystwo Informatyczne

SPIS TREŚCI

OD REDAKCJI

Szanowni Czytelnicy,

Pierwszy tegoroczny numer naszego czasopisma poświęcony jest szerokiemu spektrum zagadnień z obszaru nieodległej przyszłości informatyki: od zastosowań interfejsów mózg-komputer w połączeniu z systemami rzeczywistości wirtualnej poprzez zastosowania sztucznej inteligencji (zarówno sztucznych sieci neuronowych, jak i inteligencji roju) aż po zastosowania technologii blockchain. Pokazujemy w ten sposób różnorodność dzisiejszej informatyki i jej zastosowań, ale również jest powszechność, a w nadchodzącej dekadzie: zapewne wszechobecność. Mamy nadzieję, że niniejszy numer naszego czasopisma przyniesie Czytelnikom wiele ciekawych przemyśleń, jak również zainspiruje ich do dalszych badań.

Redaktorzy Naczelni SiMIS,
dr inż. Jacek Czerniak,
dr hab. inż. Marek Macko, prof. nadzw.

Studia i Materiały Informatyki Stosowanej, Tom 13, Nr 1, 2021
str. 5-10

BCI w VR: imersja sposobem na sprawniejsze wykorzystywanie interfejsu mózg-komputer

Adrianna Piszcz[1]

[1] *Uniwersytet Kazimierza Wielkiego, Instytut Informatyki, Kopernika 1, 85-074 Bydgoszcz*

Streszczenie: *Celem eksperymentów było zbadanie czy rzeczywistość wirtualna usprawnia korzystanie z interfejsu mózg-komputer. Do badania wykorzystano autorski system informatyczny, który umożliwia rysowanie kształtów na ekranie komputera. Przygotowane stanowisko badawcze składa się z komputera z niezbędnym oprogramowaniem, z mobilnych gogli wirtualnej rzeczywistości Esperanza EMV300 ze smartfonem Samsung Galaxy A40 oraz interfejsu mózg-komputer Emotiv Epoc. Wykazano, że imersja pozwala zwiększyć poziom koncentracji i sprawniej korzystać z interfejsu mózg-komputer. Taki rodzaj zanurzenia w rzeczywistość wirtualną może zapoczątkować całą serię aplikacji obsługiwanych w sposób intuicyjny, za pomocą komend myślowych, w wykreowanym wirtualnym świecie.*

Słowa kluczowe: *BCI, interfejs mózg-komputer, EEG, VR, rzeczywistość wirtualna, rysowanie*

BCI in VR: an immersive way to make the brain-computer interface more efficient

Abstract: *The purpose of the experiments was to investigate whether virtual reality improves the use of the brain-computer interface. The study used a custom computer system that allows drawing shapes on the computer screen. The prepared test stand consists of a computer with the necessary software, Esperanza EMV300 mobile virtual reality goggles with a Samsung Galaxy A40 smartphone and Emotiv Epoc brain-computer interface. It was shown that immersion allows to increase the level of concentration and use the brain-computer interface more efficiently. This kind of immersion in virtual reality could initiate a whole series of applications operated intuitively, via thought commands, in a created virtual world.*

Keywords: *BCI, brain-computer interface, EEG, VR, virtual reality, painting*

1. Wprowadzenie

Obecny świat stał się miejscem wiecznego biegu. Każdy z nas w ciągu całego dnia wykonuje kilkaset małych zadań, czynności o których musi pamiętać. Większość z nich znajduje się w świecie wirtualnym, świecie Internetu. Istnieje coraz więcej rozwiązań mających na celu zminimalizowanie konieczności fizycznej interakcji z urządzeniami komputerowymi. Możliwość komunikacji z komputerem za pośrednictwem myśli staje się coraz bliższa. Interfejs mózg-komputer w połączeniu z rzeczywistością wirtualną może stać się sposobem na wykonywanie tego o czym pomyślimy. Sposobem załatwiania spraw, bez konieczności wychodzenia z domu, bez konieczności uczenia się interfejsu kolejnej aplikacji, serwisu czy programu.

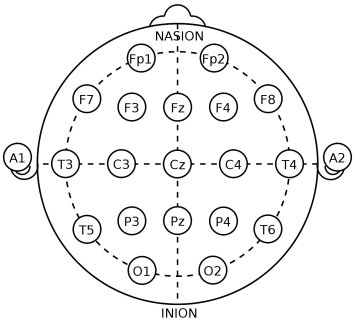
W artykule przedstawiono badania, które pozwoliły określić czy wykorzystanie interfejsu mózg-komputer wsparte rzeczywistością wirtualną, dzięki imersji pozwala zwiększyć poziom koncentracji i sprawniej korzystać z urządzenia.

2. Elektroencefalografia

Elektroencefalografia (EEG) jest to rejestrowanie elektrycznej aktywności kory mózgowej, przede wszystkim potencjałów postsynaptycznych. EEG mierzy potencjał polowy w przestrzeni wokół neuronów, a zatem jest to średnia aktywność elektryczna wielu neuronów na znacznym obszarze. Pomiar dokonywany jest za pośrednictwem elektrod, poprzez które potencjały korowe przewodzone są do urządzenia wzmacniającego sygnał. Rozmieszczenie elektrod na głowie jest standaryzowane Między-

narodowym Systemem 10-20 (Rys. 1).

Większość rejestrowanego sygnału EEG jest związana z aktywnością neuronów. Głównym generatorem sygnału są potencjały postsynaptyczne, które trwają od 50 do 200 ms. Potencjały czynnościowe trwają 10 ms lub mniej, a zatem mają bardzo ograniczone pole elektryczne. Potencjał czynnościowy przemieszcza się wzdłuż aksonu do zakończeń nerwowych, aby za pośrednictwem neuroprzekaźnika powodować zmiany stopnia polaryzacji błony komórkowej.



Rysunek 1: Lokalizacja elektrod

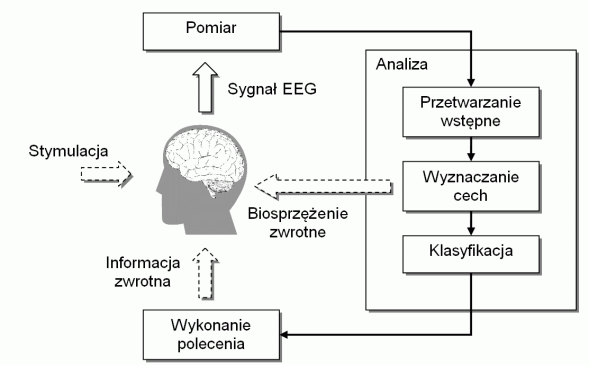
Źródło: https://www.researchgate.net/figure/The-10-20-International-system-of-EEG-electrode-placement_fig1_324361441

3. Interfejs mózg-komputer

Interfejs mózg-komputer stanowi sposób komunikacji komputera z człowiekiem. Użytkownik wykorzystując urządzenie i dedykowane do niego oprogramowanie jest w stanie, po odpowiedniej kalibracji, sterować komputerem w sposób podobny, jak z użyciem myszki lub klawiatury. Poszczególne stany wywoływane w naszym mózgu dają się rozpoznawać przez oprogramowanie urządzenia i mogą być przypisane do danego klawisza klawiatury lub do kliknięcia przyciskiem myszy.

Działanie interfejsu mózg-komputer (Rys. 2) inicjuje użytkownik, poprzez rozpoczęcie zadanie lub podjęcie określonej akcji. Następnie następuje akwizycja sygnałów mózgowych oraz przetwarzanie wstępne, na które składa się między innymi oczyszczenie sygnału z występujących w nim artefaktów. Z zebranego i przetworzonego sygnału wyodrębnione zostają cechy charakterystyczne. Cechy te zostają sklasyfikowane, a następnie zostaje obliczony sygnał wyjściowy. Ostatnim etapem jest prezentacja podjętego zadania.

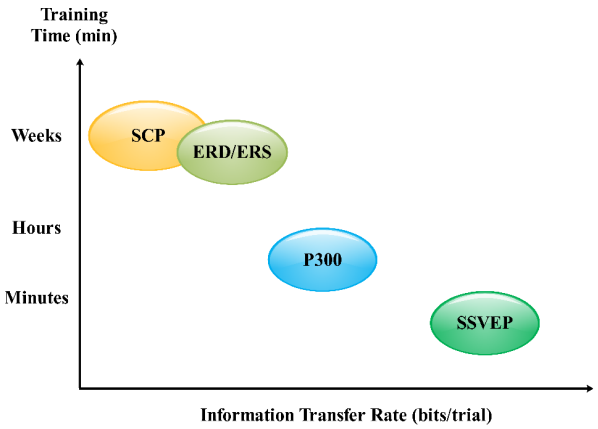
Aby w pełni wykorzystywać możliwości interfejsu mózg-komputer konieczna jest kalibracja urządzenia oraz trening. Trening odbywa się z wykorzystaniem symulatorów. Użytkownik na symulatorze uczy się świadomego manipulowania daną cechą charakterystyczną. Czas trwania treningu zależy od metody oraz od tego jaka cecha charakterystyczna ma być wykrywana podczas tego treningu.



Rysunek 2: Schemat działania interfejsu mózg-komputer

Źródło: http://www.eletel.p.lodz.pl/programy/vepcom/

Najszybsze działanie wykazują interfejsy synchroniczne, w których dokonuje się uśrednienia sygnału, co znacząco ułatwia identyfikację. Interfejsy asynchroniczne rozpoznają spontaniczną aktywność kory mózgowej, powstałą na skutek wykonywania pewnych czynności myślowych (np. wyobrażenie ruchu kończyną), są znacznie wolniejsze i trudniejsze do wykrycia. Wybrana cecha charakterystyczna sygnału ma wpływ na to jakie metody przetwarzania sygnału rozstaną wykorzystane, jakie będą możliwe zastosowania i ograniczenia oraz jaki będzie czas treningu użytkownika (Rys. 3). Cechy charakterystyczne sygnałów mózgowych mogą być wywoływane przez bodziec zewnętrzny lub mogą być kontrolowane świadomie.



Rysunek 3: Porównanie cech charakterystycznych pod względem czasu uczenia i szybkości przesyłania informacji

Źródło: https://kar.kent.ac.uk/66503/

4. Metody

Badania zostały wykonane w warunkach domowych na przygotowanym stanowisku badawczym. Stanowisko badawcze wykorzystane w eksperymencie składa się z trzech głównych elementów: komputera (laptopa Nitro 5), do którego podłączony jest dodatkowy monitor i zewnętrzna mysz komputerowa, interfejsu mózg-komputer (Emotiv EPOC) oraz mobilnych gogli VR wyposażonych w smartfon (gogle VR Esperanza EMV300 z telefonem Samsung Galaxy A40) (Rys. 4).

Rysunek 4: (1) laptop, (2) interfejs mózg-komputer, (3) gogle VR, (4) płyn do nawilżenia elektrod, (5) mysz komputerowa, (6) pudełko na elektrody, (7) dodatkowy monitor, (8) źródło zasilania, (9) odbiornik radiowy USB

Źródło: własne

W trakcie badania na ekranie komputera wyświetlane są instrukcje oraz zadania. Zadania polegają na odwzorowaniu geometrycznych kształtów według szablonu wykorzystując polecenie myślowe do przyłożenia wirtualnego pisaka. Podczas, gdy zarejestrowana zostanie aktywność kory mózgu odpowiadająca skalibrowanej komendzie myślowej w aplikacji zostanie wygenerowany dźwięk świadczący o opuszczeniu pisaka i możliwości rysowania odpowiedniego kształtu. Aplikacja mierzy czas wykonywania danego zadania, aby możliwe było zestawienie wyników badania.

Wszystkie urządzenia biorące udział w eksperymencie połączone są ze sobą za pośrednictwem komputera. Wykorzystywany jest laptop Nitro 5 z systemem Windows 10. Dodatkowo do laptopa podłączony jest zewnętrzny monitor, aby ułatwić śledzenie przebiegu badania.

Urządzenie BCI wykorzystane w badaniu to Emotiv EPOC. Jest to 16-kanałowy interfejs mózg-komputer wyposażony w elektrody mokre. Do prawidłowego działania urządzenia niezbędna jest zatem sól fizjologiczna, która będzie nawilżała elektrody i zapewniała odpowiednie przewodnictwo. W zestawie do urządzenia dołączony jest odbiornik USB, który należy podłączyć do komputera, aby uzyskać połączenie.

Do imersji wykorzystano proste mobilne gogle VR Esperanza EMV300 wraz ze smartfonem Samsung Galaxy A40 z systemem Android 10. W wyborze uwzględniono takie czynniki jak waga urządzenia oraz sposób montażu na głowie, a także mobilność ze względu na sposób połączenia telefonu z aplikacją desktopową na komputerze. Istotne było, aby osoba badana czuła się komfortowo przy jednoczesnym użytkowaniu interfejsu mózg-komputer i gogli wirtualnej rzeczywistości. Gogle Esperanza zbudowane są z lekkiego plastiku, wyposażone są w elastyczne pasy mocujące z możliwością regulacji. Możliwość ustawienia odległości soczewek sprawiła, że dostosowanie pola widzenia było bardzo proste i nie przedłużało czasu badania.

Do urządzenia Emotiv EPOC wykorzystano oprogramowanie EPOC Control Panel, które pozwala na kalibracje i przypisanie określonych stanów myślowych do klawiszy na klawiaturze lub myszce i wykonywania akcji w aplikacji autorskiej.

Połączenie z aplikacją autorską na komputerze realizowane jest poprzez oprogramowanie Trinus. Aplikacja dokonuje projekcji aktualnie aktywnej aplikacji z komputera na smartfon w czasie rzeczywistym. Wyświetlany obraz jest już odpowiednio podzielony i przygotowany do odbioru poprzez gogle VR.

Badanie było nagrywane za pośrednictwem programu OBS Studio, dzięki któremu oprócz ekranu komputera nagrywana była jednocześnie twarz osoby badanej z kamery, w którą wyposażony jest laptop. Oprócz tego program umożliwia śledzenie aktywności z klawiatury i myszki, co również jest uwzględniane w nagraniach. Dzięki temu analizując badania można określić kiedy zostało wywołane kliknięcie myszki za pośrednictwem interfejsu mózg-komputer.

5. Testy

Badanie zostało przeprowadzone w ciągu dwóch dni. Osobą badaną była 25-letnia kobieta. Warunki przeprowadzonego badania były jednakowe w każdym dniu – odbywały się w ciszy, o tej samej porze dnia. Osoba badana była wyspana (po 8 godzinach snu) i najedzona.

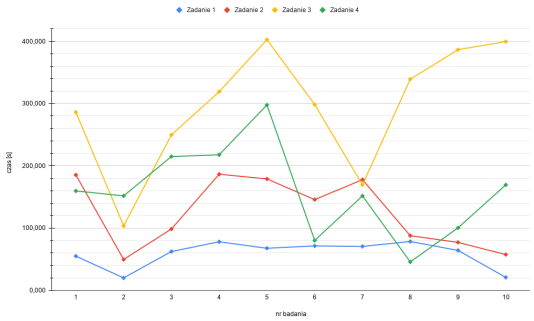
Dzień pierwszy badań polegał na przeprowadzeniu 10 prób bez wykorzystywania gogli wirtualnej rzeczywistości. Każde badanie rozpoczynało się od prawidłowego zamontowania interfejsu mózg-komputer na głowie osoby badanej. Elektrody należało dokładnie namoczyć roztworem soli fizjologicznej lub płynem do soczewek, a następnie założyć na głowę zgodnie z instrukcją widoczną w programie.

Drugi dzień badań obejmował wykorzystanie go-

gli wirtualnej rzeczywistości. Należało powtórzyć montaż oraz kalibrację urządzenia BCI, a dodatkowo poprawnie skonfigurować i założyć gogle VR. W programie *Trinus Cardboard* na komputerze należało ustawić opcję „Sensor Mode" na *No sensor*, po to, aby nadal sterować kursorem za pomocą podłączonej myszy komputerowej, a nie za pomocą żyroskopu wbudowanego w telefonie. Po wciśnięciu „START" w aplikacji *Trinus* na telefonie oraz na komputerze smartfon rozpoczyna transmisję aktywnej aplikacji z komputera. Na tym etapie należało upewnić się czy wszystkie elektrody w programie nadal są zielone i czy osoba badana widzi poprawnie obraz i kursor. Kiedy wszystko było gotowe nagranie było uruchamiane, a osoba badana mogła rozpocząć badanie w dowolnej chwili naciskając klawisz Enter.

6. Rezultaty

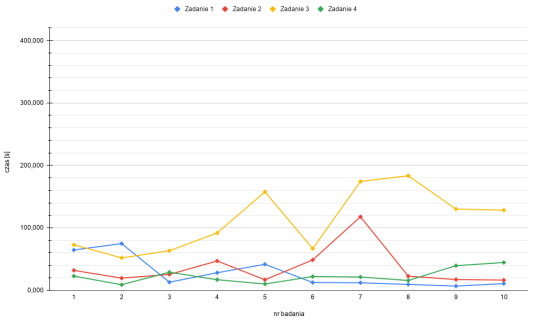
Dane z eksperymentu zostały zestawione w programie *Excel*, a następnie na potrzeby generowania wykresów czas trwania poszczególnych zadań został zamieniony na sekundy. Dla każdego zadania zostały również wyliczone średnie czasy zebrane z dziesięciu prób.



Rysunek 5: Dane dla każdego zadania z 10 prób bez VR

Źródło: własne

Sporządzono wykresy liniowe, które prezentują zmiany czasu wykonywania zadania w poszczególnych próbach (Rys. 5 oraz Rys. 6). Z wykresów można odczytać, że zadanie o niskiej złożoności (Zadanie 1) bez wykorzystania gogli VR zajmowało podobną ilość czasu w każdej z prób i był to czas najkrótszy. Natomiast im większa była złożoność i precyzja wykonywanego zadania, tym większy był ogólny czas trwania wykonywanego zadania oraz wahnięcia tego czasu. Dla zadania pierwszego średni czas wynosił niecałe 59 sekund, a amplituda między najkrótszym, a najdłuższym czasem wykonywania tego zadania wynosiła ok. 58 sekund. Natomiast dla zadania trzeciego, które okazało się najbardziej skomplikowane czas był 5-krotnie dłuższy, tj. odpowiednio średni czas – 295 sekund oraz amplituda – 299 sekund.

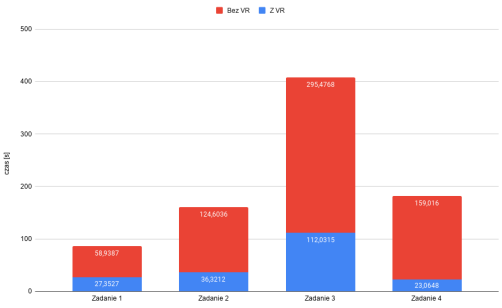


Rysunek 6: Dane dla każdego zadania z 10 prób z VR

Źródło: własne

Dane zebrane w badaniach, w których wykorzystano gogle wirtualnej rzeczywistości są mniej zróżnicowane w poszczególnych próbach. Widać wyraźnie, że odchylenia pojawiają się rzadziej i tylko w zadaniach o wyższej złożoności. Czas w jakim wykonywane są zadania jest podobny dla wszystkich czterech zadań, a zatem wykorzystanie gogli VR sprawiło, że złożoność zadania nie była aż tak istotna. Zadanie, które było wykonywane najszybciej to rysowanie koła. Średni czas wynosił 23 sekundy, natomiast różnica między najkrótszym, a najdłuższym czasem wykonywania tego zadania wyniosła ok. 36 sekund. Zadaniem wykonywanym najdłużej ponownie okazało się zadanie 3, czyli rysowanie sześcianu. Tutaj średni czas wynosił 112 sekund, a różnica – niecałe 132 sekundy.

Zestawiając uzyskane średnie wyniki można jednoznacznie stwierdzić, że wyniki uzyskane w próbach z wykorzystaniem gogli VR są zdecydowanie krótsze (Rys. 7). Poziom koncentracji osiągany był szybciej i łatwiej było go utrzymać. Największą po-



Rysunek 7: Zestawienie średnich wyników

Źródło: własne

prawę wyników czasowych udało się uzyskać dla zadania 4 (rysowanie koła) i jest to niemal 6-krotnie mniejszy średni czas wykonywania tego zadania. Zadanie 2 uzyskało ok. 3,5-krotnie niższy czas, natomiast zadania 3 i 1 niewiele gorzej - ponad 2-krotnie krótszy czas.

7. Wnioski

Przeprowadzone badania pozwoliły określić, że czas wykonywania zadań polegających na rysowaniu kształtów geometrycznych o różnej złożoności zmniejsza się znacząco, gdy zadania wykonywane są z wykorzystaniem gogli wirtualnej rzeczywistości. Łączny średni czas wykonywania zadań zmniejszył się ponad 3-krotnie. Oznacza to, że szybciej osiągany był odpowiedni poziom skupienia oraz możliwe było utrzymywanie go przez dłuższy czas. Dodatkową obserwacją jest fakt, że zadanie wymagające największej precyzji okazało się o wiele łatwiejsze, gdy było wsparte rzeczywistością wirtualną.

Istnieje realny wpływ wykorzystywania środowiska wirtualnego na użytkowanie interfejsu mózg-komputer. Rozwiązanie to jest bardziej intuicyjne i nie wymaga tak długiego treningu, jak w przypadku tradycyjnego wykorzystywania BCI. A zatem może być z powodzeniem wykorzystane w projektowaniu i tworzeniu nowych koncepcji wizualnych, między innymi przy tworzeniu rysunków poglądowych, szkiców, a także modelowaniu brył w przestrzeni, sculptingu (cyfrowym rzeźbieniu).

Taki rodzaj zanurzenia w rzeczywistość wirtualną może zapoczątkować całą serię aplikacji obsługiwanych w sposób intuicyjny, za pomocą komend myślowych, w wykreowanym wirtualnym świecie. Pozwoliłoby to wykonywać nużące i powtarzalne czynności w odmieniony, prosty, a zarazem bardziej rozrywkowy sposób. Takie rozwiązanie przeznaczone byłoby nie tylko dla osób zdrowych, lecz także dla osób z dysfunkcjami ruchowymi, które w ten sposób byłyby bardziej niezależne.

Kolejnym możliwym zastosowaniem są inteligentne domy. Odpowiedni interfejs wykorzystujący rzeczywistość rozszerzoną pozwoliłby sterować wieloma inteligentnymi urządzeniami poprzez komendy myślowe.

Istotnym torem rozwoju tych połączonych technologii są różnego rodzaju symulacje i sesje treningowe. Przykładem może być obsługiwanie wózka inwalidzkiego sterowanego poprzez BCI lub sterowanie dronem na przygotowanej symulacji w wirtualnym środowisku bez ponoszenia ryzyka uszkodzenia sprzętu oraz urazów fizycznych użytkownika i/lub osób postronnych.

Literatura

[1] J. Amores, X. Benavides, and P. Maes, "PsychicVR: Increasing mindfulness by using virtual reality and brain computer interfaces," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2016, pp. 2–2.

[2] E. A. Curran and M. J. Stokes, "Learning to control brain activity: A review of the production and control of EEG components for driving brain–computer interface (BCI) systems," *Brain and cognition*, vol. 51, no. 3, pp. 326–336, 2003.

[3] W. J. Greenleaf, "Virtual reality applications in medicine," in *Proceedings of WESCON'95*. IEEE, 1995, p. 691.

[4] W.-Y. Hsu, "Brain–computer interface connected to telemedicine and telecommunication in virtual reality applications," *Telematics and Informatics*, vol. 34, no. 4, pp. 224–238, 2017.

[5] G. Jayabhavani, N. Raajan, and R. Rubini, "Brain mobile interfacing (BMI) system embedded with wheelchair," in *2013 IEEE Conference on Information & Communication Technologies*. IEEE, 2013, pp. 1129–1133.

[6] M. Kołodziej, R. J. Rak, and A. Majkowski, "Interfejs mózg-komputer – wybrane problemy rejestracji i analizy sygnału EEG," *PRZEGLĄD ELEKTROTECHNICZNY*, 2009. [Online]. Available: https://www.researchgate.net/publication/308596494

[7] P. Kucharski, A. Rybicki, and M. Kopaczyńska, "Połączenie mózg-komputer jako metoda komunikacji z niereagującymi pacjentami – przegląd literatury," *Acta Bio-Optica et Informatica Medica. Inżynieria Biomedyczna*, vol. 21, no. 3, 2015.

[8] R. Leeb, R. Scherer, C. Keinrath, G. Pfurtscheller, D. Friedman, F. Y. Lee, H. Bischof, and M. Slater, "23 Combining BCI and Virtual Reality: Scouting Virtual Worlds," *Toward brain-computer interfacing*, p. 393, 2007.

[9] E. Mikołajewska and D. Mikołajewski, "Interfejsy mózg-komputer jako rozwiązania dla osób niepełnosprawnych z uszkodzeniami układu nerwowego," *Niepełnosprawność–zagadnienia, problemy, rozwiązania*, vol. 3, no. 4, pp. 19–36, 2012.

[10] M. M. Moore, "Real-world applications for brain-computer interface technology," *IEEE*

Transactions on Neural Systems and Rehabilitation Engineering, vol. 11, no. 2, pp. 162–165, 2003.

[11] S. Paszkiel, "Control based on brain-computer interface technology for video-gaming with virtual reality techniques," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 10, 2016.

[12] S. Paszkiel, "Using BCI and VR Technology in Neurogaming," in *Analysis and Classification of EEG Signals for Brain–Computer Interfaces*. Springer, 2020, pp. 93–99.

[13] Y. Renard, F. Lotte, G. Gibert, M. Congedo, E. Maby, V. Delannoy, O. Bertrand, and A. Lécuyer, "OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments," *Presence: teleoperators and virtual environments*, vol. 19, no. 1, pp. 35–53, 2010.

[14] M. Van Gerven, J. Farquhar, R. Schaefer, R. Vlek, J. Geuze, A. Nijholt, N. Ramsey, P. Haselager, L. Vuurpijl, S. Gielen *et al.*, "The brain–computer interface cycle," *Journal of neural engineering*, vol. 6, no. 4, p. 041001, 2009.

[15] A. Wulff-Abramsson, A. Lopez, and L. A. M. Cerda, "Paint with Brainwaves – A Step Towards a Low Brain Effort Active BCI Painting Prototype," in *Mobile Brain-Body Imaging and the Neuroscience of Art, Innovation and Creativity*. Springer, 2019, pp. 183–188.

[16] D. Zapała, P. Francuz, E. Zapała, N. Kopiś, P. Wierzgała, P. Augustynowicz, A. Majkowski, and M. Kołodziej, "The impact of different visual feedbacks in user training on motor imagery control in BCI," *Applied psychophysiology and biofeedback*, vol. 43, no. 1, pp. 23–35, 2018.

Studia i Materiały Informatyki Stosowanej, Tom 13, Nr 1, 2021
str. 11-16

Efektywność klasyfikacji mrugnięcia z wykorzystaniem wybranych sieci neuronowych

Krzysztof Galas[1]

[1] *Instytut Informatyki, Uniwersytet Kazimierza Wielkiego w Bydgoszczy*

Streszczenie: *Głównym celem badania było porównanie i wykazanie, która z przedstawionych typów sieci neuronowych najlepiej sklasyfikuje pobierany sygnał EEG mierzony przez headset Emotiv EPOC. Przedstawione sieci neuronowe są stosowane w szerokim zakresie przetwarzania danych. Została wybrana sieć splotowa oraz sieć Kohonena. Parametry sieci, takie jak ilość przejść danych uczących w jednej sesji uczącej zostały modyfikowane. Badanie uwzględnia stopień błędu klasyfikacji sygnału przez sieć oraz ilość czasu potrzebna do trening modelu. Wartością porównywalną jest stosunek czasu treningu do stopnia dokładności klasyfikacji. Otrzymane wyniki zostały przedstawione jako wykresy zależności w/w wartości do parametrów dotyczących uczenia modelu sieci.*

Słowa kluczowe: *Sieci neuronowe, sztuczna inteligencja, headset, BCI, EEG.*

Effectiveness of blink classification using selected neural networks

Abstract: *The main objective of this study was to compare and demonstrate which of the presented neural network types will best classify the extracted EEG signal measured by the Emotiv EPOC headset. The presented neural networks are used in a wide range of data processing. A convolutional network and a Kohonen network have been selected. The network parameters such as number of learning data transitions in one learning session have been modified. The study considers the degree of signal classification error by the network and the amount of time required to train the model. The comparative value is the ratio of training time to classification accuracy. The obtained results are presented as plots of the relation of the above-mentioned values to the parameters concerning the learning of the network model.*

Keywords: *Neural networks, artificial intelligence, headset, BCI, EEG.*

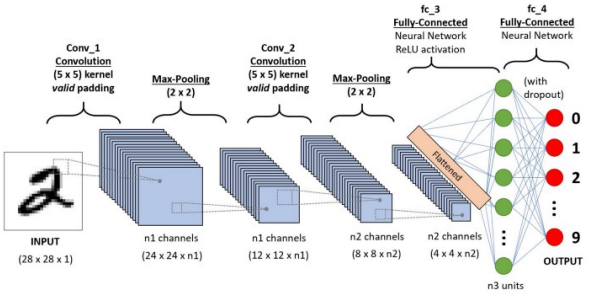
1. Wprowadzenie

Głównym celem badania jest wybranie najskuteczniejszej sieci neuronowej w rozpoznawaniu artefaktu, jakim jest mrugnięcie. Celem szczegółowym jest dogłębna analiza wykorzystywanych do tych celów sieci neuronowych oraz ich skuteczności w wykrywaniu artefaktów sygnału EEG. Drugim celem szczegółowym jest wybranie sieci, która zapewni największą skuteczność i efektywność tego procesu.

W pracy zostanie dowiedzione czy wybrana sieć gwarantowała wykrywanie mrugnięć oczami. Sygnał wejściowy zostanie pobrany za pomocą urządzenia EMOTIV EPOC. Pobrane sygnały będą klasyfikowane w dwóch kategoriach: sygnały błędne (nieposiadające oczekiwanego kształtu) oraz sygnały prawidłowe (z kształtem występującym w trakcie mrugnięcia) przeznaczone do treningu. Mrugnięcia zostaną wyselekcjonowane ręcznie z sygnału treningowego, znany będzie zakres czasowy wystąpienia mrugnięcia w pobranym sygnale. Następnie zostanie przeprowadzony trening sieci neuronowej pod kątem wykrycia anomalii. Wytrenowana sieć zostanie sprawdzona na sygnale testowym. Oczekiwany efekt to zaznaczenie występujących mrugnięć na obrazach. Program zostanie napisany w języku Python z wykorzystaniem dostępnych bibliotek m.in. OpenCV oraz TensorFlow. Na podstawie wyników działania programu zostanie stwierdzone czy wybrana sieć gwarantowała wykrywanie mrugnięć oczami.

2. Wybrane sieci neuronowe

Sieć splotowa W literaturze nazywana jest konwolucyjną siecią neuronową lub splotową siecią neuronową. Należy do głębokich sieci neuronowych (ang. Deep Neural Network). Jest to rodzaj sieci składająca się z dużej ilości warstw biorących udział w przetwarzaniu sygnałów co widać na przedstawionym rysunku (Rys. 1). Wykorzystywane są dla złożonych elementów takich jak zdjęcia czy inne obrazy. Skomplikowany pod względem analitycznym sygnał wejściowy można podzielić na jednostkowe zestawy cech, które składają się na dane wejściowe. Cechą charakterystyczną tego rodzaju sieci jest idea początkowego ustalania wag jako liczby o losowych, małych wartościach. Ten zabieg ma za zadanie usprawnić proces uczenia się złożonej w hierarchii sieci neuronowej.



Rysunek 1: Schemat CNN.

Źródło: towardsdatascience.com/ a-comprehensive-guide-to-convolutional-" neural-networks-the-eli5-way-3bd2b1164a53

Zadaniem sieci jest redukcja obrazu do formy uproszczonej do zminimalizowanej formy, która jednak nie modyfikuje obrazu do formy uniemożliwiającej działanie pozostałych elementów sieci. Neurony w pierwszej warstwie splotowej jako wejścia przyjmują wartości związane tylko z przetwarzanym obrazem.

Następnym krokiem jest wykonanie operacji uśredniania wartości obrazu. Wykorzystywane są trzy rodzaje redukcji max, min oraz average. W operacji max wybierana jest największa wartość z danego obszaru składającego się na cały obraz. Odpowiednio dobrze działają filtry min, a operacja average - uśrednia cały obszar do jednej wartości. Operacje te powodują odpowiednio zmianę wielkości obrazu. W przypadku, gdy maska wychodzi poza obszar obrazu, puste przestrzenie (brak danych) jest wypełniane zerami.

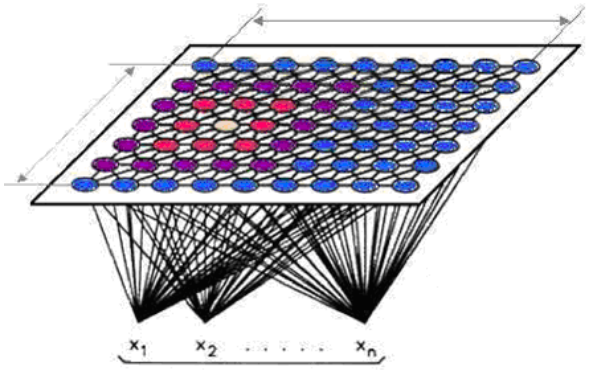
Przefiltrowany obraz (po wyjściu z każdej ukrytej warstwy neuronowej) otrzymujemy obraz nazywanym mapą cech. Posiada on trzy atrybuty, takie jak wysokość, szerokość oraz głębia. Sieć neuronowa wykorzystuje wyszukuje filtry najodpowiedniejsze do danego wyodrębnienia cech. Najprostszą cechą do wyodrębnienia jest wykrycie przez filtr krawędzi pionowych oraz poziomych. W rezultacie zastosowania takiego filtru otrzymujemy charakterystyczny obraz o dużym kontraście. Dzięki zastosowaniu filtrów możemy wyodrębnić poziome oraz pionowe linie. Jednak nie zawsze spotyka się linie idealnie pionowe lub poziome, widoczne na rysunku linie ukośne także są rozpoznawane jako przynależne do jednego i drugiego zbioru. Wynika to z tego, że filtr klasyfikuje krótkie odcinki prostej, które mogą być poziome lub pionowe, jednak występować na różnych wysokościach względem siebie. Rezultatem może być ich widoczność po przejściu tych dwóch filtrów.

Sieć Kohonena Kohonen jako pierwszy zastosował uczenie sieci z rywalizacją w literaturze spotyka się również określenie *sieci samoorganizujące się*. Odnosi się to do sieci, których zadaniem jest uczenie się z dowolnych relacji wektorów wejścia i wyjścia. W odróżnieniu od podstawowego uczenia jest to, że wektor wejściowy X zostaje poddany procesowi normalizacji tak, aby jego długość była równa 1 ($||X|| = 1$).

W tym rodzaju sieci uczeniu nie są poddawane wszystkie neurony wejściowe, tylko ten, który posiada największy sygnał wyjściowy oznaczany $y_m^{(j)}$. Z powodu zastosowania takiego rozwiązania sieci Kohonena nazywane są jako sieci uczone metodą rywalizacji. Przejawia się przez rywalizację neuronów po przejściu sygnału o najwyższy wynik za co jest nagradzany w postaci uczenia czyli zmodyfikowaniu wagi. Dla licznej ilości zróżnicowanych obiektów wejściowych, neurony będą wykazywać inne wartości co skutkuje coraz lepszym dopasowaniem obiektów podobnych do wzorcowego.

Przez pryzmat operowania na wielowymiarowych wartości wejściowych tworzona jest mapa topologiczna, która reprezentuje wynik uczenia się neuronów (odpowiedź na zadane dane). Każdemu z tych wyjść przypisana jest odpowiedź na daną wartość sygnału wejściowego oznacza to, że siła odpowiedzi jest wprost proporcjonalna dla podobieństwa do zestawu uczącego. Mapa topologiczna (Rys. 2) przybiera formę dwuwymiarowej płaszczyzny, których wartości można przedstawiać za pomocą gradientu. Neurony przedstawiono odpowiednio od x_1 do x_n. Dzięki takiej reprezentacji wyników łatwiej jest wyłonić zwycięzcę.

Neurony posiadają własne numery i zostają one



Rysunek 2: Mapa topologiczna sieci Kohonena.

Źródło: researchgate.net/figure/Kohonen-topological-map˙fig8˙304161914

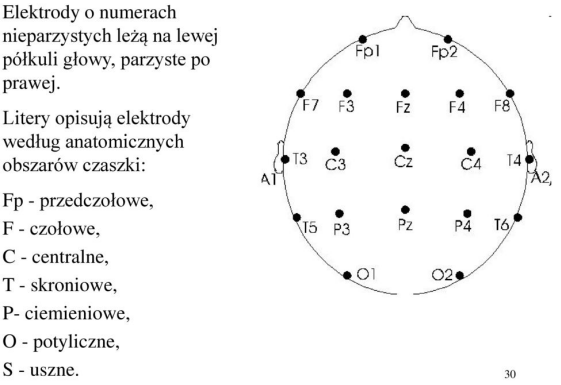
uporządkowane (przedstawia się to za pomocą współczynnika m). Całą metodę uczenia można rozszerzyć do neuronów sąsiadujących ze zwycięzcą przez co usprawniamy metodę uczenia przedstawianą wcześniej. Na przedstawionym schemacie (Rys. 2) widać gradient na warstwie wyjściowej od wyjścia najbardziej dopasowanego poprzez do neuronów sąsiednich. Sąsiedztwo może być określane do 4 lub 8 neuronów najbliższych, mogą mieć kształt zwykłego kwadratu $3x3$ (środkowy neuron wyjścia jako zwycięzca). W przypadku czterech sąsiadujących neuronów stosuje się tzw. sąsiedztwo Von Neumanna, gdzie jego zasięg wynosi 2. Gdy znany jest neuron o najwyższej wartości, realizowane jest uczenie realizowane poprzez zmianę wag dla sygnałów wejściowych realizowane jest to wzorem (1). Sam proces jest wielokrotnie powtarzany samoczynnie do uzyskania zadowalających wyników.

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta^{(j)} h(m, m^*)(\dot{x}_i^{(j)} - w_i^{(m)(j)}) \quad (1)$$

gdzie:
$\dot{x}$ - wejście podlegające uczeniu,
m - numer porządkowy neuronu,
j - liczba iteracji/przejść,
η - stopień uczenia się w danej iteracji,
h - stopień nauczania ze względu na odległość od neuronu od zwycięzcy (m^*).

3. EEG oraz artefakty

Sygnał EEG Sygnał EEG (Elektroencefalografia) jest to zebrane przez elektrodę wzbudzenie potencjałów elektrycznych wywoływanych poprzez zmianę potencjałów wywoływanych przez pracę neuronów. Punkt kontaktu elektrody z głową użytkownika lub osób badanych pobiera sygnały nie tylko związane z aktywnością jednego neuronu, tylko z całego obszaru mózgu zaangażowanego w wykonanie jakiejś czynności. Źródłem sygnału jest kora mózgowa, a rozkład elektronów jest determinowany przez określony ich rozkład na powierzchni głowy.



Rysunek 3: Schemat przedstawiający jednostkową elektrodę pobierającą sygnał.

Źródło: Kołodziej M. „Przetwarzanie, analiza i klasyfikacja sygnału EEG na użytek interfejsu mózg-komputer".

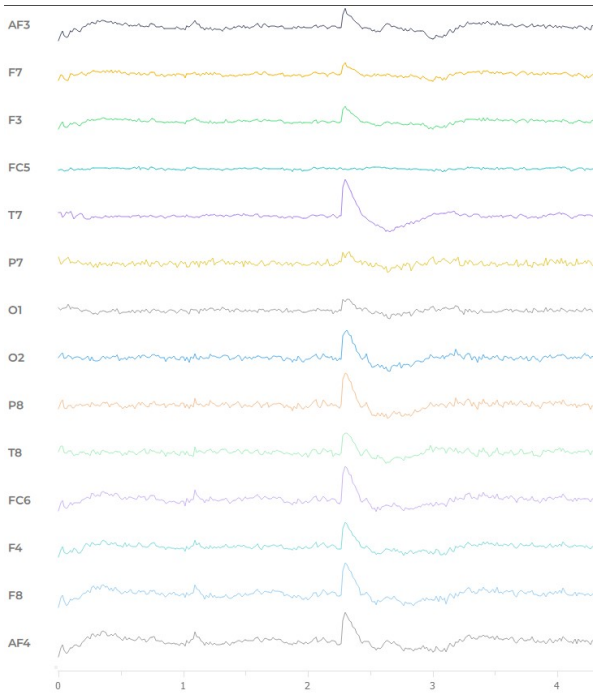
Na przedstawionym schemacie (Rys. 3) widoczny w układzie pomiarowym jest wzmacniacz, poprawiający wydobycie zmiany potencjałów wywoływanych przez neurony. Powodem jego zastosowania jest słaba siła sygnału, który musi zostać odebrany przez różne warstwy głowy (skórę, czaszkę, płyn mózgowo rdzeniowy). Sama siła sygnału uzyskanego jest wielkością rzędu kilkudziesięciu miliwoltów.

Można zauważyć, że sygnał nie jest generowany przez typowe neurony, tylko te znajdujące się w warstwie kory mózgowej znajdującej się na powierzchni mózgu. Znajdują się tam komórki piramidalne, które wzbudzone generują skierowaną różnicę potencjałów ułatwiające odczyt wartości wzbudzenia.

Kora mózgowa dzieli się na obszary odpowiadające za ruchy całego ciała. Poszczególne partie naszego ciała generują aktywność, której siła jest wprost proporcjonalna do obszaru kory mózgowej za nie odpowiedzialną.

Artefakty Wspomniane wcześniej obszary mózgu odpowiedzialne za elementy ruchowe lub zmysłowe nie zaprzestają aktywności, mogą być wykonywane ruchy podświadomie lub być odruchami. Powoduje to powstawaniem, zakłóceń o różnej dłu-

gości. Mogą one być powiązane również z uszkodzeniem diody czy używaniem różnego rodzaju sprzętu elektrycznego. Mogą być związane z zastosowaniem wzmocnienia sygnału lub nawet przesunięcie elektrod.



Rysunek 4: Mrugnięcie numer 1 (od 2 do 3 sekundy).

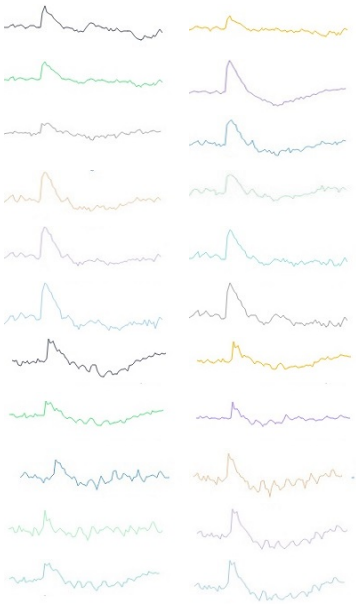
Źródło: Własne.

Analiza częstotliwościowa wskazuje na najwyższą wartość ok. 20 Hz. W obrębie charakterystyki sygnału należy przyjrzeć się wykresów z zapisów poszczególnych elektrod. Zapis występuje cyklicznie o krótkim zakresie czasu, występuje głównie w elektrodzie oznaczonej nazwą F_{p2}. Podczas badania można pominąć całe okno czasowe, które zawiera ten artefakt.

Przy wykorzystywaniu wzmacniacza dla akwizycji sygnału możliwe jest powstawanie fal elektromagnetycznych związanych z przepływem prądu. Mogą one być wychwytywane przez elektrodę, występują charakterystyczne artefakty o dużej częstotliwości związanej z częstotliwością przepływu prądu, posiadają one charakterystykę o wartości 50 Hz (widoczne jest to również na wykresie analizy częstotliwościowej poprzez duży skok w okolicach 50 Hz. Występujący na całej długości zapisywanego sygnału. W tym przypadku może pomóc zaprzestanie badania czy pomiaru aby naprawić ewentualną awarię oraz ponownie przystąpić do pomiaru.

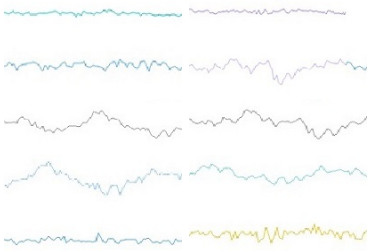
4. Stworzenie zbioru uczącego i kontrolnego

Zbiór mrugnięć Jako zestaw danych wejściowych zostało wytypowanych 32 zdjęć sygnałów (z czego 22 przedstawiające mrugnięcie (rys.5) oraz 10 bez mrugnięcia (rys.6).



Rysunek 5: Zestaw przedstawiający mrugnięcie.

Źródło: własne



Rysunek 6: Zestaw przedstawiający stan neutralny.

Źródło: własne

Zbiorem kontrolnym jest część zbioru z sygnałem, w którym występuje mrugnięcie czyli 10 losowych fragmentów. Do tego zbioru zalicza się też kilka elementów ze zbioru ze stanem neutralnym zawierającym 4 losowo wybrane wycinki sygnału.

5. Topologia sieci

Sieć splotowa Zgodnie z założeniami sieć splotowa została dopasowana odpowiednio do potrzeb. Z myślą o znanej wielkości analizowanych fragmentów sygnału pierwsza warstwa wejściowa została stworzona o wielkości wejściowej warstwy 200 x 60 (jako fragment sygnału), później odpowiednio warstwy maxPool o wielkości 2 x 2 i ich odpowiednio zwiększających się wartości wyjściowych, odpowiednio 16, 32, 64. Ostatnia warstwa posiada 512 wejść. Funkcja aktywacji warstwy splotowej to tak zwana funkcja rectifier (skrót to ReLU). Ostatnim neuronem tej sieci jest neuron o sigmoidalnej funkcji aktywacji ze względu na klasyfikacje sygnału jako mrugnięcie lub brak mrugnięcia co zważając na zakres obejmowany przez tę funkcje może klasyfikować jako brak mrugnięcia sygnał w niewielkiej mierze podobny do tego, w którym mrugnięcie występuje.

Sieć Kohonena Sieć została stworzona z 1000 neuronów wejściowych, których metoda rywalizacji została dobrana jako metoda sąsiedztwa trójkątnego. Program ma za zadanie klasyfikację na mapie wielkości 4 x 5 przyporządkować do zbiorów przedstawiony fragment jako mrugnięcie lub stan neutralny.

6. Rezultaty

Wyniki sieci splotowej Dla przedstawionej wcześniej sieci splotowej wykonano trzy różne sesje uczące dla różnych ilości przejść danych w jednej sesji uczącej sieci neuronowej. Odpowiednio dla 3 przejść po 10 danych wejściowych, w konfiguracji 5 sesji uczących zawierających 6 pakietów danych oraz 10 sesjach po 3 zdjęcia wejściowe. W dalszej części pracy oraz w zapisie z programów odpowiadają temu nazwy epoch (dane przetwarzane są raz przez całą sieć neuronową), steps per epoch (ilość kroków w sesji uczenia, gdzie podawany jest jeden zestaw danych).

Tabela 1: Dokładność klasyfikacji.

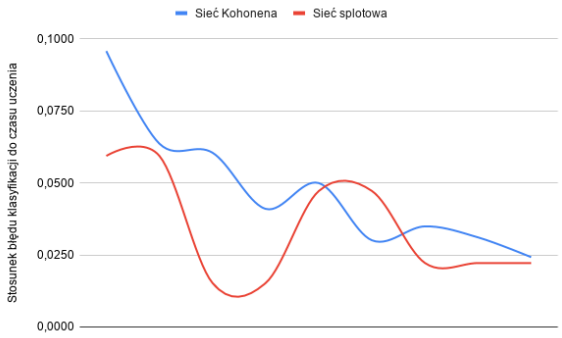
Parametry	Czas uczenia	Dokładność
E:2 S:16	00:00:10	0.5940
E:3 S:10	00:00:45	0.6870
E:5 S:5	00:00:15	0.7087
E:10 S:3	00:00:30	0.6667

Wyniki sieci Kohonena Przeprowadzono pomiary zaczynając od 500 iteracji, zwiększając ich ilość stopniowo o 250. Otrzymane wyniki przedstawione są w tabeli poniżej.

Tabela 2: Wyniki klasyfikacji sieci Kohonena.

Iteracje	Czas uczenia	Dokładność
500	00:00:05	0.4790
750	00:00:08	0.5092
1000	00:00:10	0.6051
1250	00:00:15	0,6154
1500	00:00:15	0.7496
1750	00:00:20	0.6023
2000	00:00:20	0,6988
2250	00:00:25	0,7778
2500	00:00:30	0.7273

Zestawienie wyników Wskazane wyniki w postaci tabel zostały zestawione ze sobą w sposób obliczenia stosunku czasu uczenia do ich dokładności w celu jednoznacznego porównania wyników otrzymanych przez obie sieci. Przedstawiono je w postaci wykresów odpowiadających sieci splotowej oraz sieci Kohonena. W celu łatwiejszej interpretacji wyników zestawiono je ze sobą na jednym wykresie (rys.7). Aby porównać obie sieci powielono wyniki sieci splotowej w taki sposób, aby ilość parametrów była równa.



Rysunek 7: Zestawienie wykresów.

Źródło: Własne.

7. Wnioski

Celem głównym przeprowadzonego badania było wybranie najskuteczniejszej sieci neuronowej w rozpoznaniu artefaktu występującego, gdy użytkownik mruga. Podczas tego działania widoczny jest

charakterystyczny zapis sygnału, który odpowiada otwarciu i zamknięciu powieki. Jest to łatwe do wywołania zaburzenie stanu neutralnego. Sygnał został pobrany za pomocą urządzenia EMOTIV EPOC. Podczas pomiaru wprowadzono celowo artefakt, który następnie został odpowiednio przygotowany do analizy przez sieć neuronową.

Celem szczegółowym była analiza wykorzystywanych do tych celów sieci neuronowych oraz ich skuteczności w wykrywaniu artefaktów sygnału EEG. Jako sieci analizujące dostępny typ danych wybrano sieci Kohonena oraz sieć RCNN. Zostały one stworzone na potrzeby eksperymentu zgodnie z ich przeznaczeniem. Kolejnym celem szczegółowym było określenie, która z wytypowanych sieci, osiągnie największą dokładność przy możliwie jak najniższym czasie treningu. Wybrane elementy sygnału zostały przeanalizowane przez obie sieci, a ich osiągi czasowe wraz z dokładnością zostały przedstawione w tabelach. Wyliczona została zależność między długością treningu, a dokładnością wskazania odpowiedniego fragmentu sygnału.

Obie sieci zadziałały poprawnie oraz wykonały zakładane przez Autora zadania. Najlepszym wskaźnikiem założonym w eksperymencie okazała się sieć splotowa, co potwierdza tezę zawartą w pracy dyplomowej.

Literatura

[1] D. L. Felten, R. F. Józefowicz, J. A. Craig, C. A. Machado, and J. A. Perkins, *Atlas neuroanatomii i neurofizjologii Nettera*. Elsevier Urban & Partner, 2012.

[2] L. Grad, "Przykład optymalizacji struktury sztucznej sieci neuronowej metodą algorytmów genetycznych," *Biuletyn Instytutu Automatyki i Robotyki*, vol. 12, no. 23, pp. 27–36, 2006.

[3] N. F. Güler, E. D. Übeyli, and I. Güler, "Recurrent neural networks employing lyapunov exponents for eeg signals classification," *Expert systems with applications*, vol. 29, no. 3, pp. 506–514, 2005.

[4] M. Jukiewicz, M. Buchwald, and A. Cysewska-Sobusiak, "Usuwanie artefaktów z sygnałów sterujących interfejsem mózg-komputer," *Poznan University of Technology Academic Journals. Electrical Engineering*, 2017.

[5] M. Kołodziej, R. J. Rak, and A. Majkowski, "Interfejs mózg-komputer – wybrane problemy rejestracji i analizy sygnału EEG," *PRZEGLĄD ELEKTROTECHNICZNY*, 2009.

[6] A. Krizhevsky, I. Sutskever, and G. H. Hinton, "Imagenet classification with deep con- volutional neural networks," *Adv. Neural Inf. Process. Syst*, vol. 25, 2012.

[7] R. Leszek, "Metody i techniki sztucznej inteligencji," *PWN, Warszawa*, 2005.

[8] D. Matthew and R. Fergus, "Visualizing and understanding convolutional neural networks," in *Proceedings of the 13th European Conference Computer Vision and Pattern Recognition, Zurich, Switzerland*, 2014.

[9] D. Mikołajewski, E. Tomaszewska, and M. Karczmarek, "Interfejsy mózg-komputer w sterowaniu urządzeniami i systemami mechatronicznymi," *Studia i Materiały Informatyki Stosowanej*, vol. 10, no. 2, pp. 4–9, 2018.

[10] S. Paszkiel, "Wykorzystanie metody PCA i ICA do analizy sygnału EEG w kontekście usuwania zakłóceń," *Pomiary Automatyka Kontrola*, vol. 59, no. 3, pp. 204–207, 2013.

[11] S. M. Plis, D. R. Hjelm, R. Salakhutdinov, E. A. Allen, H. J. Bockholt, J. D. Long, H. J. Johnson, J. S. Paulsen, J. A. Turner, and V. D. Calhoun, "Deep learning for neuroimaging: a validation study," *Frontiers in neuroscience*, vol. 8, 2014.

[12] T. Pracki and D. Pracka, "Elektroencefalografia cyfrowa," *Sen*, vol. 4, pp. 71–77, 01 2004.

[13] R. Tadeusiewicz, *Sieci neuronowe*. Akademicka Oficyna Wydawnicza Warszawa, 1993, vol. 180.

[14] R. Tadeusiewicz and M. Szaleniec, *Leksykon sieci neuronowych*. Projekt Nauka. Fundacja na rzecz promocji nauki polskiej, 2015.

[15] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*. IEEE, 2010.

[16] J. Żurada, M. Barski, and W. Jędruch, *Sztuczne sieci neuronowe: podstawy teorii i zastosowania*. Wydawnictwo Naukowe PWN, 1996.

Studia i Materiały Informatyki Stosowanej, Tom 13, Nr 1, 2021
str. 17-32

Porównanie wybranych algorytmów wilczego stada stosowanych w rozwiązaniach problemów optymalizacji

Belco Sangho[1]

[1] *Uniwersytet Kazimierza Wielkiego, Instytut Informatyki, Kopernika 1, 85-074 Bydgoszcz*

Streszczenie: *Algorytmy optymalizacyjne zyskały uznanie jako szybki i konsekwentny sposób rozwiązywania problemów optymalizacyjnych. W ostatnim czasie wilki są coraz częściej wykorzystywane jako inspiracja do tworzenia algorytmów, jak i w projektach używających tych algorytmów. W niniejszej pracy opisano sześć wybranych algorytmów. Następnie zaimplementowano je w języku R i porównano z pomocą sześciu funkcji porównujących, tzw. benchmarków. Wyniki trzydziestu testów na każdej z funkcji zaprezentowano za pomocą średniego wyniku, odchylenia standardowego wyniku, średniego czasu oraz odchylenia standardowego czasu. Dodatkowo zaprezentowano wykres zbieżności na dwóch z funkcji porównujących. Uzyskane wyniki algorytmów często różniły się od tych zaprezentowanych w publikacjach, jednak skuteczność części z nich była lepsza bądź porównywalna z PSO[1], DE[2] i GA[3]. Najlepszym wilczym algorytmem okazał się Grey Wolf Optimizer[4].*

Słowa kluczowe: *Optymalizacja, Algorytmy rojowe, Algorytmy wilcze, Wilki, Funkcje porównujące*

Comparison of selected wolf pack algorithms used in solving optimization problems

Abstract: *Optimization algorithms have gained recognition as a fast and consistent way to solve optimization problems. Recently, wolves have been increasingly used as inspiration for algorithms as well as in projects using these algorithms. In this paper, six selected algorithms are described. They were then implemented in R and compared using six comparison functions, called benchmarks. The results of thirty tests on each function were presented by mean score, standard deviation of the score, mean time and standard deviation of the time. Additionally, a convergence plot on two of the benchmark functions was presented. The algorithm results obtained often differed from those presented in the publications, but the performance of some of the algorithms was better or comparable to PSO[1], DE[2], and GA[3]. The best wolf algorithm was found to be Grey Wolf Optimizer[4].*

Keywords: *Optimization, Swarm algorithms, Wolf herd algorithm, Wolfs, Benchmarks*

1. Wprowadzenie

Problemy optymalizacyjne są kwestią, którą prawie wszyscy z nas znają szkoły podstawowej. Rozwiązanie funkcji X,Y zazwyczaj znajdowało się w miejscu zerowym, na przecięciu się wykresu wartości z osią X. Pozwalało nam to zazwyczaj określić ile owoców ma kupić sprzedawca aby jak najlepiej wykorzystać wzmożony popyt, oraz nie zamrozić funduszy w przypadku zbyt dużego zakupu. Dokładnie takimi problemami zajmują się algorytmy optymalizacyjne, które szukają jak najmniejszego kosztu w dostępnym nam obszarze poszukiwania. Teren ten jest zazwyczaj stworzony za pomocą pewnej liczby zmiennych bądź wyborów, których wartości musimy odpowiednio dostosować aby uzyskać najlepszy wynik.

Jednak skoro została tutaj przywołana szkoła, to czemu nie skorzystać np. z pochodnych, bądź liniowo obliczyć wszystkie wartości? Cóż, algorytmy oparte na pochodnych istnieją (takie jak), ale okazało się, że są one zbyt czasochłonne/wadliwe aby dało się z nich korzystać w coraz to bardziej złożonych funkcjach. Dokładnie z tego samego powodu nie używa się liniowych obliczeń. Mimo iż moc komputerowa rośnie w zastraszającym tempie (zgodnie z prawem Moora głoszącym o podwajaniu się liczby tranzystorów co około dwa lata), jeszcze szybciej rosła złożoność problemów, które należało rozwiązać. Dodatkowo niektóre problemy wymagają

szybkiej reakcji, jak na przykład obliczenie długości kroku motoru w robocie.

Dziedziną zajmującą się szybkim rozwiązywaniem problemów optymalizacyjnych, jednak bez gwarancji znalezienia najlepszego rozwiązania, jest heurystyka. Nauka ta nie bierze pod uwagę jedynie poprawności rozwiązania, ale też prostotę samego algorytmu i prędkość działania. Aby zobrazować sens tej nauki, weźmy na przykład problem komiwojażera. Występuje w nim pewna liczba miast, oraz trasy je łączące o określonej długości. Celem jest takie pokonanie wszystkich dróg, aby uzyskać najmniejszy wynik. Z uwagi na ilość kombinacji dróg (których liczba wynosi silnię z $(n-1)/2$, co dla dziesięciu miast daje nam liczbę 181440) obliczenie tego jedynego najlepszego rozwiązania jest niezwykle czasochłonne. Jeśli jednak zamiast znalezienia najlepszego wyniku zadowolimy się tylko wartością zbliżoną do niego, to algorytm heurystyczny znajdzie rozwiązanie w przeciągu sekund.

Algorytmy optymalizacyjne zazwyczaj działają w sposób losowy, co z jednej strony pozwala na lepsze przeszukiwanie obszaru (niedeterministyczne kolejne kroki pozwalają na mniejszą szansę zakopania się w rozwiązaniach lokalnych). Jest to szczególnie ważne kiedy obszar funkcji ma bardzo nieprzewidywalną strukturę. Z drugiej strony jednak sprawia, że każde rozwiązanie zazwyczaj różni się od poprzedniego, co w przypadku dużych różnic między kolejnymi wynikami sprawia, że skuteczność algorytmu staje pod znakiem zapytania.

Funkcjonowanie algorytmów jest oparte na iteracjach, to jest takich samych krokach, które są podejmowane tak długo, aż nie osiągniemy pewnego celu. Sprawia to, że algorytmy te są zazwyczaj stosunkowo proste. Pozwala to na łatwą implementację i modyfikację działania. Jest to ważne, gdyż bardzo często algorytmy są pisane z myślą o wszystkich rodzajach funkcji. Zazwyczaj jednak zależnie od projektu w którym mają one być użyte, różne są kryteria odnośnie dokładności, bądź dostosowania do konkretnego typu problemów. Prosta wypływa też na prędkość działania.

Przeważająca większość algorytmów korzysta z populacji (wielu agentów posiadających własne rozwiązanie), chociaż możliwe jest oparcie działania na tylko jednym rozwiązaniu. Generalnie rzecz biorąc, najczęściej wyróżnia się trzy główne podgatunki algorytmów optymalizacyjnych:

- Genetyczne - inspirujące się procesami ewolucji zachodzącymi w naturze. Funkcjonuje w nich populacja, która w każdej iteracji jest sortowana za pomocą funkcji dopasowania, aby następnie lepsza część członków (zazwyczaj połowa) została wykorzystana aby stworzyć potomków. Gorsza część populacji zostaje usunięta. Przykładowe algorytmy to *Genetic Algorithm* [3] oraz *Differential evolution* [2].

- Fizyczne - oparte na zjawiskach fizycznych takich jak grawitacja bądź wyżarzanie. Ich działanie z uwagi całkowicie odmienne działanie różnych procesów może bardzo mocno się od siebie różnić. *Big Bang Big Crunch* [5] oparto na teorii początku i końca wszechświata, to jest Wielkiego Wybuchu, oraz Wielkiego Krachu. Inicjalne jest generowane wiele rozwiązań rozprzestrzenionych po obszarze poszukiwania, aby następnie je zgromadzić do pojedynczego wyniku. Innymi algorytmami fizycznymi są np. *Gravitational Search Algorithm* [6] bądź *Simulated Annealing* [7].

- Rojowe - symulowane jest w nich najczęściej zachowanie stad, ławic bądź grup zwierząt jednego gatunku, które poszukują pożywienia. Z uwagi na mnogość zwierząt i ich stosunkowo proste zachowania, są to najczęściej tworzone algorytmy. Przykładowe rozwiązania to kolonia mrówek [8] oraz rój cząstek [1].

Wilcze algorytmy należą do ostatniej wymienionej grupy algorytmów. Używanie tych zwierząt jako inspiracji zaczęło się stosunkowo niedawno, ponieważ wszystkie opisane tu prace są zostały stworzone po roku 2010. Najstarsze rozwiązanie używające tych ssaków znalezione przez autora tej pracy wystąpiło w 2007 roku [9]. Wilki polują w małych grupach, najczęściej rodzinach. Ich taktyka na obalanie dużych zwierząt jest stosunkowo prosta i opiera się na kilku krokach:

1. Wykrycie ofiary z pomocą ich wyczulonego zmysłu węchu. Wilki regularnie przeszukują swój obszar łowny, który jest zazwyczaj stały. Pozwala im to na znalezienie zarówno kopytnych, jak i też innych źródeł pożywienia.

2. Zmuszenie pojedynczego osobnika do ucieczki. Jako że wilki nie podejmą walki z całym stadem, to spróbują one odizolować ich ofiarę od reszty. W przypadku spotkania samotnej ofiary też podejmą tę próbę, aby ją zmęczyć.

3. Okrążenia i próba obalenia zwierzęcia przez stado wilków. Proces ten może trwać nawet kilka godzin.

Zachowanie to jest przedstawione na zdjęciu 1. Zasady te są na tyle proste, że z powodzeniem można przenieść je na algorytm. Z uwagi jednak na istnienie kilkunastu algorytmów wilczych, rodzi się pytanie który z nich tak naprawdę najlepiej się nadaje do zastosowań komercyjnych bądź naukowych.

Rysunek 1: Zachowanie wilków podczas polowania na dużego kopytnego. Na części A przedstawiono pogoń, na B do D widoczne jest okrążanie i nękanie. Zdjęcie E pokazuje ostateczną walkę.

Niniejsza praca postara się odpowiedzieć na to pytanie. W następnym rozdziale przedstawiono 6 wilczych rozwiązań, które następnie przetestowano na 6 wykresach.

2. Wilcze algorytmy

2..1 Grey Wolf Optimizer (GWO)

Grey Wolf Optimizer [4] ukazał się na łamach czasopisma *Advances in Engineering Software* w 2014 roku. W publikacji przedstawiono inspirację autorów, schemat działania algorytmu, testy na 29 benchmarkach i 3 problemach technicznych. Pojawiło się w niej również zastosowanie algorytmu do rozwiązania problemu w inżynierii optycznej.

Algorytm oparto na zachowaniu stadnym wilków, w którym dzielą się one na poszczególne grupy:

- Alfa - para dowodząca. Podejmuje najważniejsze decyzje, takie jak wybór czasu snu bądź polowania.
- Beta - część pomagająca alfom w podejmowaniu i egzekwowaniu decyzji.
- Omega - najniższa ranga w hierarchii, najczęściej zajmująca się doglądaniem młodych. Omegi często są kozłami ofiarnymi dla silniejszych wilków.
- Delta - pozostałe wilki, zazwyczaj są to starsi, opiekunowie najmłodszych, strażnicy bądź zwiadowcy.

Co ciekawe autorzy GWO nie zaprezentowali w swojej pracy dowodów pochodzących z literatury naukowej, które potwierdzałyby takie twarde podziały żywych wilków. Badania na stadach w dziczy [10] [11] wydają się tym twierdzeniom twardo zaprzeczać. Przedstawiają one wilki raczej jako rodzinę, w której alfa są partnerami mającymi młode.

Alfę(α), betę(β) oraz deltę(δ) przeniesiono do algorytmu jako trzy najlepsze wilki. Pozostałe są nazywane omegami(ω) To na ich podstawie oblicza się kroki wszystkich wilków w trakcie okrążania ofiary, za pomocą następnych równań:

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X}_p(t) - \overrightarrow{X}(t)| \quad (1)$$

$$\overrightarrow{X}_p(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A} \cdot \overrightarrow{D} \quad (2)$$

$$\overrightarrow{A} = 2\overrightarrow{a} \cdot \overrightarrow{r}_1 - \overrightarrow{a} \quad (3)$$

$$\overrightarrow{C} = 2 \cdot \overrightarrow{r}_2 \quad (4)$$

gdzie

$\overrightarrow{X}_p(t)$	: Oznacza pozycję ofiary, a w przełożeniu na działanie algorytmu średnia z trzech najlepszych wilków.
$\overrightarrow{X}$	: Pozycja wilka w przestrzeni poszukiwań.
$\overrightarrow{A}, \overrightarrow{D}, \overrightarrow{C}$	: Wektory współczynników
t	: Konkretna iteracja pętli.
$\overrightarrow{a}$	: Współczynnik który zmniejsza się z 2 do 0 na w trakcie iteracji, co ma zwiększać znaczenie eksploatacji podczas działania algorytmu.
$\overrightarrow{r}_1, \overrightarrow{r}_2$	: Liczby generowane losowo z przestrzeni [0,1].

Wzór 2 pozwala nam na obliczenie ruchu wilka, korzystając z pozycji ofiary oraz wprowadzając pewną losowość z pomocą $\overrightarrow{A}, \overrightarrow{C}$. Zmniejszanie wartości $\overrightarrow{a}$ pozwala z czasem zmniejszyć przedział wartości "skoku" wilka. Z uwagi na to, aby zapewnić że losowość będzie występować nawet w końcowych krokach algorytmu, wprowadzono dodatkowo $\overrightarrow{C}$, który zawsze daje wartości z przedziału $[2r, 0]$. Jako że ofiara nie jest obecna w algorytmie, jej pozycję symulujemy przy pomocy alfy, omegi i delty. Dzieje się tak na mocy następujących równań:

$$\begin{aligned} \overrightarrow{D}_\alpha &= |\overrightarrow{C}_1 \cdot \overrightarrow{X}_\alpha - \overrightarrow{X}| \\ \overrightarrow{D}_\beta &= |\overrightarrow{C}_2 \cdot \overrightarrow{X}_\beta - \overrightarrow{X}| \\ \overrightarrow{D}_\delta &= |\overrightarrow{C}_3 \cdot \overrightarrow{X}_\delta - \overrightarrow{X}| \end{aligned} \quad (5)$$

$$\begin{aligned} \overrightarrow{X}_1 &= \overrightarrow{X}_\alpha - \overrightarrow{A}_1 \cdot \overrightarrow{D}_\alpha \\ \overrightarrow{X}_2 &= \overrightarrow{X}_\beta - \overrightarrow{A}_2 \cdot \overrightarrow{D}_\beta \\ \overrightarrow{X}_3 &= \overrightarrow{X}_\delta - \overrightarrow{A}_3 \cdot \overrightarrow{D}_\delta \end{aligned} \quad (6)$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X}_1 + \overrightarrow{X}_2 + \overrightarrow{X}_3}{1} \quad (7)$$

Działanie tych wzorów można zobaczyć na rysunku 2. Średnia z pozycji trzech najlepszych wilków pozwala nam uzyskać teoretyczną pozycję ofiary. Jak widać więc działanie algorytmu nie jest zbyt złożone. W algorytmie 1 przedstawiono pseudokod GWO, który odzwierciedla kod wykorzystany w testach.

Algorithm 1 GWO

1: Stwórz członków $NumPopulation$ wilczego stada X_i, generując ich w losowych miejscach przestrzeni
2: $a = 2$
3: Oblicz i posortuj na podstawie wartości funkcji dopasowania dla każdego wilka
4: $X_\alpha, X_\beta, X_\delta$ są równe odpowiednio pierwszemu, drugiemu i trzeciemu wilkowi
5: **for** $mIter = 1, 2, \ldots, maxIteration$ **do**
6: **for** $i = 1, 2, \ldots, NumPopulation$ **do**
7: $A \leftarrow 2 * a * rand(0,1) - a$
8: $C \leftarrow 2 * rand(0,1)$
9: Oblicz nową pozycję wilka korzystając z równań 5, 6 i 7
10: **if** $X_i < X_\alpha$ **then**
11: $X_\alpha \leftarrow X_i$
12: **end if**
13: **if** $X_i > X_\alpha$ & $X_i < X_\beta$ **then**
14: $X_\beta \leftarrow X_i$
15: **end if**
16: **if** $X_i > X_\beta$ & $X_i < X_\delta$ **then**
17: $X_\delta \leftarrow X_i$
18: **end if**
19: **end for**
20: $a \leftarrow 2 - t * ((2)/mIter)$
21: **end for**
22: return X_α

2..2 Grey Wolf Optimizer with Evolutionary population (GWO-EPD)

Modyfikacja *Grey Wolf Optimizer* stworzona przez tych samych autorów została opublikowana rok po oryginalnym algorytmie w 2015 roku w czasopiśmie *Neural Computing and Applications*. Zmiana w nim zaprezentowana jest stosunkowo niewielka. Opiera się ona na dodaniu jeszcze jednego kroku, który eliminuje gorszą połowę populacji. Następnie tworzy ją znów, opierając się na pozycji jednego z trzech przewodzących wilków, bądź losowo w całej przestrzeni. Działanie przetestowano w publikacji na kilkunastu funkcjach testujących, ale co ciekawe porównano go tylko z oryginalnym algorytmem.

Działanie to opiera się na wzorach:

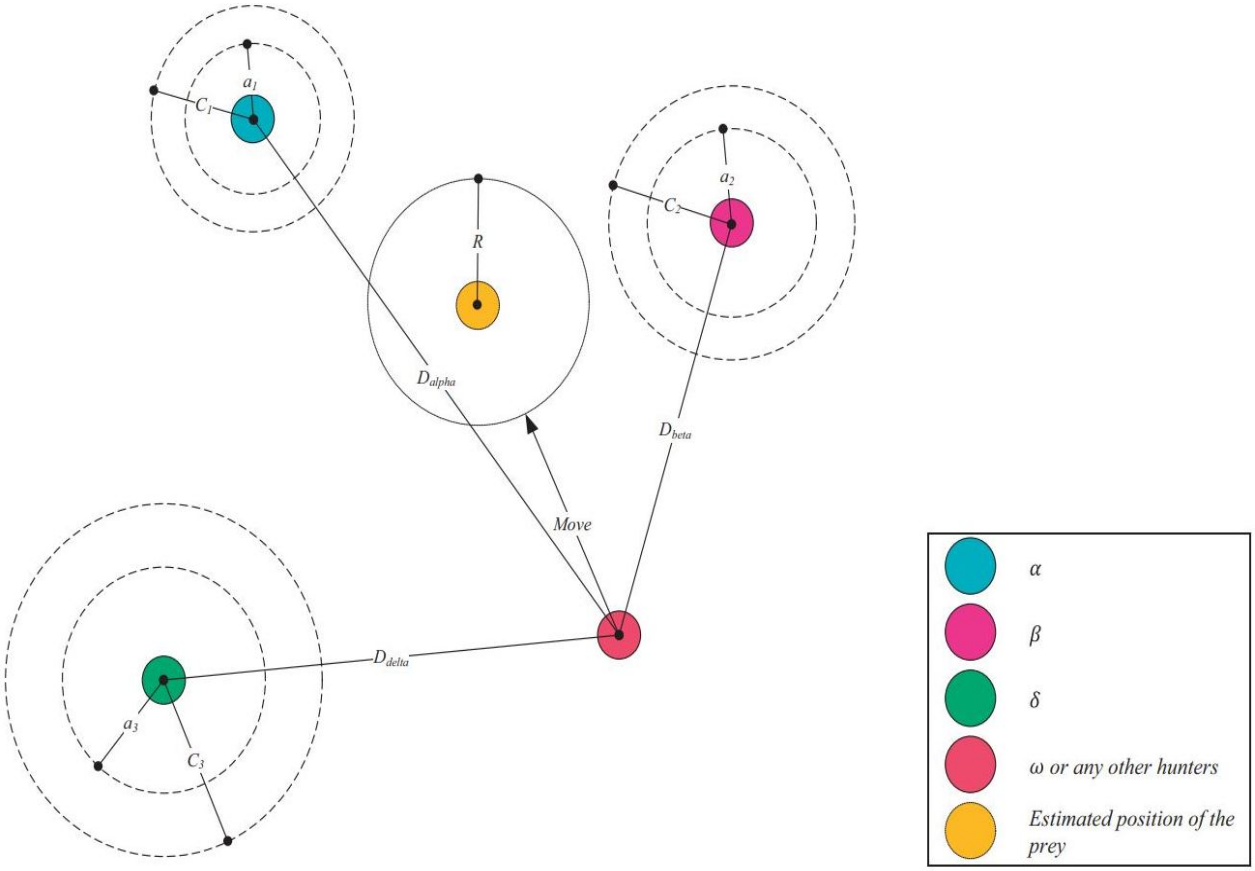
$$\overrightarrow{X}_p(t+1) = \overrightarrow{X}_\alpha(t) \pm (ub - lb \cdot r + lb) \quad (8)$$

$$\overrightarrow{X}_p(t+1) = \overrightarrow{X}_\beta(t) \pm (ub - lb \cdot r + lb) \quad (9)$$

$$\overrightarrow{X}_p(t+1) = \overrightarrow{X}_\delta(t) \pm (ub - lb \cdot r + lb) \quad (10)$$

$$\overrightarrow{X}_p(t+1) = (ub - lb \cdot r + lb) \quad (11)$$

Gdzie:



Rysunek 2: Działanie wzorów 5, 6, 7, rysunek z pracy *Grey Wolf Optimizer* [4]

$\overrightarrow{X}_p(t)$	: Oznacza pozycję w następnej iteracji.
$\overrightarrow{X}_\alpha, \overrightarrow{X}_\beta, \overrightarrow{X}_\delta$	: Pozycja wilka alfa, beta i delta.
ub, lb	: Respektywnie górna oraz dolna granica przestrzeni poszukiwania rozwiązania.
r	: Liczba generowane losowo z z przestrzeni [0,1].

Można zauważyć, że z uwagi na użycie górnej i dolnej wartości przedziału, oraz wykorzystanie losowej zmiennej z przedziału $[0, 1]$, umiejscowienie wilka może być dalekie od jednego z przewodzących wilków. Kod modyfikacji przedstawiono w algorytmie 2. Należy go umieścić na końcu każdej iteracji w GWO.

Algorithm 2 GWO-EDP

1: **for** $i = 1, 2, \ldots, NumPopulation/2$ **do**
2: $ub \leftarrow$ Górna granica przedziału
3: $lb \leftarrow$ Górna granica przedziału
4: $r \leftarrow rand(0, 1)$
5: Z równym prawdopodobieństwem, użyj jednego z wzorów 8, 9, 10 bądź 11
6: **end for**

2..3 Wolf Colony (WC)

Algorytm *The Wolf Colony Algorithm and Its Application* [12] zaprezentowano w czasopiśmie *HCTL Open International Journal of Technology Innovations and Research* w roku 2011. W publikacji pojawiły się testy na 4 funkcjach. Dodatkowo zaimplementowano algorytm na jednym problemie wyznaczania trasy.

Działanie algorytmu opiera się na zachowaniach zainspirowanych tymi z natury: zwiadach, oblężeniu i śmierci z powodu braku pożywienia. W algorytmie wykorzystano dość dużo parametrów, których objaśnienie wraz z wartościami użytymi w publikacji znajduje się poniżej:

$n = 200$	: Liczba stada.
$mark$	: Maksymalna liczba iteracji.
d	: Liczba wymiarów funkcji do obliczenia.
$\overrightarrow{X}_i$	: Wektor pozycji wilka i.
$q = 5$	: Liczba wilków przeprowadzająca zwiady.

$h = 4$: Ilość propozycji ruchów wygenerowana podczas zwiadów.
$maxdh = 15$: Maksymalna liczba wykonanych kroków podczas zwiadów.
$stepa = 1.5$: Wartość wykorzystywana do kontroli maksymalnej długości kroku podczas zwiadów.
$stepb = 0.9$: Wartość wykorzystywana do kontroli maksymalnej długości kroku podczas oblężenia.
$m = 5$: Ilość wilków usuwana podczas śmierci na koniec każdej iteracji.

Zwiady są przeprowadzane jedynie przez część stada, i są oparte na równaniu:

$$Y_j = \overrightarrow{X}_i + randn \cdot stepa \tag{12}$$

W którym Y_j równa się wartości funkcji dopasowania, a *randn* jest liczbą losową z przedziału [-1,1]. Naraz jest generowanych h takich rozwiązań, a najlepsze z nich jest porównywane do najlepszego wyniku w ogóle znalezionego przez algorytm. Jeśli jest lepsze, to zwiady dla tego konkretnego wilka kończą się. W przeciwnym wypadku jeśli ta pozycja jest lepsza od aktualnej pozycji, to następuje ruch wilka, a następnie zachowanie to jest powtarzane maksymalnie *maxdh* razy.

Podczas oblegania całe stado angażuje się w eksploatację potencjalnego rozwiązania globalnego, wyznaczanego przez pozycję najlepszego wilka. Jeśli w wyniku ruchu nowy wilk stanie się najlepszym, dalszy atak jest prowadzony w jego stronę. Dzieje się tak na mocy równania:

$$\overrightarrow{X}_i^{k+1} = \overrightarrow{X}_i^k + rand \cdot stepb \cdot (\overrightarrow{X}_{best} - \overrightarrow{X}_i^k) \tag{13}$$

Gdzie:

$\overrightarrow{X}_i^{k+1}$: Pozycja i wilka po wykonaniu ruchu.
$\overrightarrow{X}_i^k$: Pozycja i wilka przed wykonaniem ruchu.
$\overrightarrow{X}_{best}$: Pozycja najlepszego wilka.
$stepb$: Wartość wykorzystywana do kontroli maksymalnej długości kroku.
$rand$: Losowa liczba z przedziału [0,1].

Ostatnim krokiem jest śmierć głodnych wilków. Zdecydowano się tutaj ustawić tą wartość na $m = 5$, która jest stosunkowo niska biorąc pod uwagę wielkość stada użytego w publikacji. Na miejsce martwych wilków generowane są nowe losowo w przestrzeni. Pseudokod przedstawiony w algorytmie 3 obrazuje zachowanie kodu użytego w testach.

Algorithm 3 WC

```
Stwórz członków NumPopulation wilczego stada X_i, generując ich w losowych miejscach przestrzeni
s ← 0.12
q ← 5
h ← 4
maxdh ← 15
stepA ← 1.5
stepB ← 0.9
Oblicz i posortuj na podstawie wartości funkcji dopasowania dla każdego wilka
X_best ← pierwszy wilk
for mIter = 1, 2, ..., maxIteration do
    for i = 1, 2, ..., q do
        for j = 1, 2, ..., maxdh do
            Wygeneruj h X_scout nowych pozycji wilka na podstawie równania 12
            X_best−scout ← X_scout
            if X_best−scout < X_best then
                X_best ← X_best−scout
                X_i ← X_best−scout break
            else
                if X_best−scout < X_i then
                    X_i ← X_best−scout
                end if
            end if
        end for
    end for
    for i = 2, ..., NumPopulation do
        X_i ← wartość obliczona zgodnie z równaniem 13
    end for
    Wygenerowanie m wilków losowo w przestrzeni i zastąpienie nimi m najgorszych wilków
end for
```

2..4 Wolf Pack Algorithm (WPA)

Algorytm WPA [13] ukazał się na łamach czasopisma *Mathematical Problems* w 2014 roku. W publikacji opisano testy na 8 funkcjach porównawczych. Mimo iż nie jest to tam wprost opisane, algorytm ten jest mocno inspirowany, żeby nie powiedzieć oparty, na poprzednim algorytmie, *Wolf Colony*. W porównaniu do niego trochę zmieniono równanie kroku zwiadu, jak i oblegania i śmierci, oraz dodano dodatkowy krok, zwoływanie. Różnice dotknęły także zmiennych sterujących, które opisano w sposób następujący:

N : Liczba wilków.
$\overrightarrow{X}_1$: Pozycja wilka w pierwszej iteracji generowana losowo.
Y_{lead} : Najlepsza/najwyższa wartość wyznaczona przez funkcję optymalizowaną dla najlepszego wilka.
k_{max} : Maksymalna liczba iteracji.

$S = 0.12$: Parametr odpowiadający za długość kroku wilka.
$L_{near} = 0.08$: Minimalna odległość od wilka prowadzącego podczas zwoływania.
$T_{max} = 8$: Maksymalna liczba powtórzeń zachowania zwiadowczego.
$\beta = 2$: Liczba wpływająca na ilość przeżywających wilków.

Zwiady są wykonywane przez całe stado, poza najsilniejszym (najlepszym) członkiem. Równanie wykorzystane do symulacji tego stanowiska to:

$$\overrightarrow{X}_i^p = \overrightarrow{X}_i + sin(2\pi \cdot \frac{p}{h}) \cdot S, \quad p = [1, 2, ..., h] \quad (14)$$

Gdzie:

$\overrightarrow{X}_i^p$: Pozycja wilka po podjęciu kroku w stronę p.
h : wartości z przedziału $[h_{min}, h_{max}]$ będące liczbą całkowitą, niestety proponowany przedział nie został podany przez twórców.

Zasada na aktualizację bądź kontynuowanie tego procesu jest taka sama jak w WC, zmieniła się jedynie nazwa zmiennej sterującej maksymalną ilością kroków na T_{max}. Z uwagi na podjęcie zwiadów przez całe stado, oraz teoretycznej górnej wartości tych kroków, znacząco wzrósł też czas działania algorytmu.

Zwoływanie ma za zadanie przeprowadzić konwersję wszystkich wilków w pobliże ofiary. Wilk prowadzący gra rolę ofiary. Pozycja ta jest oznaczana jako $\overrightarrow{X}_{lead}$. Równanie przedstawiające to zachowanie to:

$$\overrightarrow{X}_{i+1} = \overrightarrow{X}_i + \frac{S}{2} \cdot \frac{(\overrightarrow{X}_{lead} - \overrightarrow{X}_i)}{|\overrightarrow{X}_{lead} - \overrightarrow{X}_i|} \quad (15)$$

W przypadku uzyskania lepszej wartości funkcji dopasowania przez aktualnego wilka to on staje się ofiarą. W przeciwnym wypadku ruch jest powtarzany do osiągnięcia wartości granicznej, tj. $L(\overrightarrow{X}_{lead}, \overrightarrow{X}_i) < L_{near}$. Z równaniem tym jest jednak jeden problem. W przypadku uzyskania na dole równania zera (kiedy wilk jak i ofiara stoją w tym samym miejscu), mamy do czynienia z wartością nieskończoną. Z uwagi na brak opisu tego problemu przez autorów algorytmu, podczas implementacji usunięto dolną część ułamka aby temu zapobiec. Nie jest jasne jak wpłynęło to na skuteczność algorytmu, ale było to wymagane do testów.

Obleganie jest prawie identyczne jak w przypadku poprzedniego algorytmu, i jest dane na mocy poniższego równania:

$$\overrightarrow{X}_{i+1} = \overrightarrow{X}_i + \lambda \cdot 2S \cdot |\overrightarrow{X}_{lead} - \overrightarrow{X}_i| \quad (16)$$

λ oznacza wartość losową z przedziału [-1,1]. Jeśli wartość funkcji dopasowania wilka X_{i+1} jest lepsza po kroku, jest ona aktualizowana.

Zmiany dotknęły także śmierci stada, w której to bierze udział większa liczba członków. R wilków ginie, a wartość ta jest generowana losowo z przedziału $[n/(2 \cdot \beta), n/\beta]$. Pozycje nowych wilków wyznacza równanie:

$$\overrightarrow{X} = \overrightarrow{X}_{lead} \cdot rand \quad (17)$$

Gdzie *rand* jest liczbą z przedziału $[-0.1, 0, 1]$. Pseudokod został podany w algorytmie 4

2..5 Wolf search algorithm

The Wolf search algorithm with ephemeral memory opublikowano w 2012 roku podczas konferencji *Seventh International Conference on Digital Information Management*. Jego działanie sprawdzono na ośmiu funkcjach w dwóch różnych wariantach. Niestety w pracy nie podano przyjętych parametrów sterujących algorytmem, co należy zaliczyć na duży minus. Mechanizm działania jest tutaj inny od pozostałych wilczych stad. Zamiast polować na ofiarę symulowaną przez lepszego członka, zwierzęta wypatrują lepszego miejsca/ofiary w sposób przypadkowy. Ruch ten jest podany następującym równaniem:

$$\overrightarrow{X}_i = \overrightarrow{X}_i + \alpha \cdot v \cdot rand() \quad (18)$$

Gdzie:

α : Parametr sterujący prędkością wilka.
v : Parametr sterujący zasięgiem wzroku
$rand()$: Liczba generowana losowo z przedziału [-1,1]

Krok zostanie wykonany tylko, jeśli wygenerowane miejsce jest lepsze od aktualnie zajmowanego. Jako że ssaki te mają dobrą pamięć, autorzy zasymulowali ten fakt poprzez dodanie sprawdzenia czy dane miejsce już zostało zwiedzone w poprzednich krokach. Ilość miejsc wstecz zapamiętanych przez wilka może zostać zmieniona. Jednakże gdy poruszamy się w przestrzeni ciągłej, możliwość wylosowania dwa razy tego samego miejsca, tj. powrotu do uprzednio zwiedzonego miejsca, jest co najmniej znikoma. Z tego powodu tego typu rozwiązania są zazwyczaj stosowane w problemach dyskretnych, jak problem komiwojażera. Z uwagi na to, że testy zaprezentowane w tej publikacji są oparte na przestrzeniach ciągłych, ten fakt pominięto. Generując liczbę losową *rand*() tysiąc razy w środowisku R, ani razu nie powtórzyła się ta sama wartość, co symuluje możliwość powrócenia w to samo miejsce. Oczywiście jeśli miejsc w okolicy wilka byłoby więcej (zależnie od długości pamięci wilka oraz ich

Algorithm 4 WPA

Stwórz członków $NumPopulation$ wilczego stada X_i, generując ich w losowych miejscach przestrzeni
$s \leftarrow 0.12$
$L \leftarrow 0.08$
$T \leftarrow 8$
$beta \leftarrow 2$
Oblicz i posortuj na podstawie wartości funkcji dopasowania dla każdego wilka
$X_{best} \leftarrow$ pierwszy wilk
for $mIter = 1, 2, \ldots, maxIteration$ **do**
  for $i = 1, 2, \ldots, NumPopulation$ **do**
    $h \leftarrow rand(4, 12)$
    for $i = 1, 2, \ldots, h$ **do**
      Wygeneruj X_{scout} nowych pozycji wilka na podstawie równania 14
      if $X_{scout} < X_{best}$ **then** $X_{best} \leftarrow X_{scout}$
        $X_i \leftarrow X_{scout}$ break
      else
        if $X_{scout} < X_i$ **then**
          $X_i \leftarrow X_{scout}$
        end if
      end if
    end for
  end for
  for $i = 1, 2, \ldots, NumPopulation$ **do**
    while $X_{best} - X_i > L$ **do**
      Zmień pozycję wilka zgodnie z równaniem 15
      if $X_i < X_{best}$ **then** $X_{best} \leftarrow X_i$ break
      end if
    end while
  end for
  for $i = 1, 2, \ldots, NumPopulation$ **do** $X_{siege} \leftarrow$ wartość oblicznona zgodnie z równaniem 16
    if $X_{siege} < X_{best}$ **then** $X_{best} \leftarrow X_{siege}$
      $X_i \leftarrow X_{siege}$ break
    else
      if $X_{siege} < X_i$ **then** $X_i \leftarrow X_{siege}$
      end if
    end if
  end for
  $ilMarWil \leftarrow$ liczba losowa z przedziału $(1, (numPopulation/(2 * beta), numPopulation/beta))$
  for $i = 1, 2, \ldots, ilMarWil$ **do**
    $X_i \leftarrow$ wartość oblicznona zgodnie z równaniem 17
    if $X_i < X_{best}$ **then**
      $X_{best} \leftarrow X_i$
    end if
  end for
end for

bliskości) szansa ta by wzrastała, ale dalej byłaby ona pomijalna.

Krokiem, który ma zapewnić konwergencję, jest wzajemne wypatrywanie się wilków. Mają one określony zasięg wzroku, z którego wybierają najlepszego widocznego członka stada. Jeśli jego pozycja jest lepsza, podejmą one próbę ruchu w jego kierunku. Dzieje się tak na mocy równania:

$$\overrightarrow{X}_i = \overrightarrow{X}_i + \beta e^{-r^2}(\overrightarrow{X}_j - \overrightarrow{X}_i) \tag{19}$$

Gdzie:

β : Oznacza różnicę pomiędzy wartością funkcji dopasowania dla wilka $\overrightarrow{X}_i i \overrightarrow{X}_j$
r^2 : odległość pomiędzy wilkiem a jego towarzyszem podniesiona do kwadratu.
e : Stała Eulera.

Im dalej jeden wilk jest od drugiego, tym mniejszy będzie ruch. Zachowanie to ma promować eksplorację. Ostatnim krokiem układanki jest dodanie szansy na wyrwanie się z optimum lokalnego poprzez nagły i daleki skok wilka. Z pewną szansą, przyjętą w testach przedstawionych w tej pracy na 25%, zwierzęta uciekną z zajmowanej aktualnie pozycji zgodnie z równaniem:

$$\overrightarrow{X}_i = \overrightarrow{X}_i + \alpha \cdot s \cdot escape() \tag{20}$$

Gdzie funkcja escape zwróci wartość większą od zasięgu wzroku wilka, a mniejszą od połowy obszaru przeszukiwanego. s oznacza długość kroku. Pseudokod jest widoczny na algorytmie 5. Zawarto w nim też użyte wartości parametrów sterujących.

2..6 Gaussian Guided Self-Adaptive Wolf Search Algorithm (GSAW-SA)

Modyfikacja poprzedniego algorytmu, *Gaussian Guided Self-Adaptive Wolf Search Algorithm Based on Information EntropyTheory* ukazała się na łamach czasopisma *Entropy* w 2018 roku. Dotyczy wprowadzenia zmiany parametrów sterujących ruchami wilków podczas działania algorytmu. Prędkość kroku, długość kroku, zasięg wzroku oraz szansa na ucieczkę zostały podporządkowane tej logice. Aby to uzyskać wykorzystano mapę chaosu *Gaussian* (pol. Guassa). Nowe zmienne są generowane na koniec każdej iteracji przy pomocy równania:

$$x_{n+1} = exp(-\alpha \cdot x_n^2) + \beta \quad = e^{-\alpha x_n^2} + \beta \tag{21}$$

α ustawiono na 5.4, a β na -0,52. X oznaczono zmienną której wartość jest zmieniana. Można zauważyć, że wartości te są bardzo podatne na początkowo przyjętą liczbę. Niestety autorzy nie podali w jaki sposób zainicjować zmienne. Aby wykonać testy, zdecydowano się na ustawienie wszystkich zmiennych na 0.25. Parametry sterujące nie są

Algorithm 5 WSA

Stwórz członków $NumPopulation$ wilczego stada X_i, generując ich w losowych miejscach przestrzeni
$\alpha \leftarrow 0.25$
$v \leftarrow 1$
$s \leftarrow 0.8$
$p_a \leftarrow 0.75$
Oblicz i posortuj na podstawie wartości funkcji dopasowania dla każdego wilka
$X_{best} \leftarrow$ pierwszy wilk
for $mIter = 1, 2, \ldots, maxIteration$ **do**
    for $i = 1, 2, \ldots, NumPopulation$ **do**
        $X_i \leftarrow$ wartość obliczona zgodnie z równaniem 18
        Wartość ta zostaje przypisana tylko jeśli $X_i < X_{i+1}$
        $BliskieWilki \leftarrow$ Wilki w zasięgu wzroku v
        $NajWilk \leftarrow$ Najlepszy z bliskich wilków
        if $NajWilk < X_i$ **then**
            Wykonaj równanie 19 dla wilka i
        else
            Powtórz krok pierwszy, tj równanie 18 i ewentualną podmianę pozycji
        end if
        if $LosowaLiczba[0, 1] > p_a$ **then**
            Wykonaj ucieczkę zgodnie z równaniem 20
        end if
    end for
end for

jednak zmieniane za każdą iteracją. Wprowadzono test, w którym najpierw wybierane są 4 różne losowe wilki. Następnie przy pomocy jednego z czterech mechanizmów genetycznych widocznych w tabeli 1, sprawdzamy najlepsze rozwiązanie z tym wygenerowanym. Jeśli jest lepsze, zmieniamy parametry. Cały mechanizm można zobaczyć w pseudokodzie widocznym na algorytmie 6. W testach użyto równania genetycznego oznaczonego jako RDE2.

3. Testy

3..1 Narzędzia

Do testów wykorzystano popularny język skryptowy R, w wersji 4.0.3. Jest on głównie używany do statystyki oraz pracy z dużymi ilościami danych, takimi jak *data mining*. Z uwagi na wykorzystaną przy nim licencję GNU, można z niego korzystać darmowo. Został on użyty we wszystkich zaprezentowanych tutaj publikacjach wilczych algorytmów.

Z języka R korzysta się z konsoli. Aby ułatwić pi-

Algorithm 6 GSAWSA

Stwórz członków $NumPopulation$ wilczego stada X_i, generując ich w losowych miejscach
$accelerate, p_a, s, v \leftarrow 0.25$
$P_{update} \leftarrow 0.75$
$\alpha \leftarrow -5.4$
$\beta \leftarrow -0.52$
Oblicz i posortuj wilki
$X_{best} \leftarrow$ pierwszy wilk
for $mIter = 1, 2, \ldots, maxIteration$ **do**
    for $i = 1, 2, \ldots, NumPopulation$ **do**
        $X_i \leftarrow$ wartość obliczona zgodnie z równaniem 18
        Wartość ta zostaje przypisana tylko jeśli $X_i < X_{i+1}$
        $BliskieWilki \leftarrow$ Wilki w zasięgu wzroku v
        $NajWilk \leftarrow$ Najlepszy z bliskich wilków
        if $NajWilk < X_i$ **then**
            Wykonaj równanie 19 dla wilka i
        else
            Powtórz krok pierwszy, tj równanie 18 i ewentualną podmianę pozycji
        end if
        if $LosowaLiczba[0, 1] > p_a$ **then**
            Wykonaj ucieczkę zgodnie z równaniem 20
        end if
    end for
    if $losowa(0, 1) > P_{update}$ **then**
        Wylosuj 4 pozycje różnych wilków
    end if
    for $mIter = 1, 2, \ldots, params$ **do**
        Oblicz wartość RDE2 1.
        if $X_{evolve} < X_{best}$ **then**
            for $j = 1, 2, \ldots, 4$ **do**
                if $losowa(0, 1) > P_{update}$ **then**
                    $param_j \leftarrow$ wartość z równania 21
                end if
            end for
        end if
    end for
end for

Tabela 1: Funkcje krzyżujące użyte w SAWSA

Nazwa	Wzór
DE1	$y = best_{gloable} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$
DE2	$y = x_{r1} + F \cdot (best_{gloable} - x_{r2}) - F \cdot (x_{r3} - x_{r4})$
DE3	$y = best_{gloable} + F \cdot (x_{r1} - x_{r2})$
RDE1	$y = best_{selected} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$
RDE2	$y = x_{r1} + F \cdot (best_{selected} - x_{r2}) - F \cdot (x_{r3} - x_{r4})$
RDE3	$y = best_{selected} + F \cdot (x_{r1} - x_{r2})$
RDE4	$y = x_{r1} + F \cdot (x_{r2} - x_{r3})$

sanie skryptów, użyto narzędzia R studio. Pozwala ono w prosty sposób uzyskać dostęp do konsoli, kodu, plików. Widoczne są w nim również załadowane zmienne środowiskowe, funkcje czy inne obiekty.

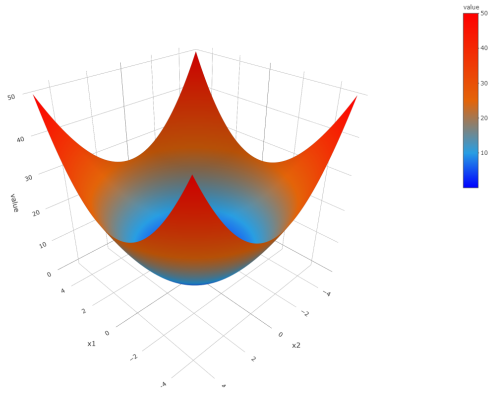
Jako że publikacja GWO wskazywała pakiet R, do którego został dodany kod źródłowy algorytmu, postanowiono z niego skorzystać. W paczce tej zawarto też inne klasyczne rozwiązania problemów optymalizacyjnych (np. PSO czy GA). Korzystanie z niej jest darmowe. Nazywa się ona *metaheuristicOpt*. Wyniki GWO oraz pozostałych klasycznych algorytmów zostały uzyskane przy pomocy tego pakietu.

Resztę wilczych algorytmów autor zaimplementował samodzielnie. W porównaniu do oryginalnych kodów, dokonano pewnych zmian w WSA, GSAWSA oraz WPA. Powody te zostały opisane w poprzednim rozdziale, i wskazują na potrzebę dokładnego opisania użytych parametrów sterujących, czy niepoprawnych wartości wynikających z niektórych równań. Pozostałe algorytmy zostały zaimplementowane zgodnie z myślą oryginalnych autorów.

Funkcje testowe należą do klasycznych problemów optymalizacyjnych. Aby uniknąć potrzeby pisania ich kodu, zostały one pobrane z strony *Optimization Test Functions and Datasets* [14], dostępnej pod adresem https://www.sfu.ca/ ssurjano/optimization.html. Z pośród kilkudziesięciu dostępnych tam funkcji ciągłych wybrano 6.

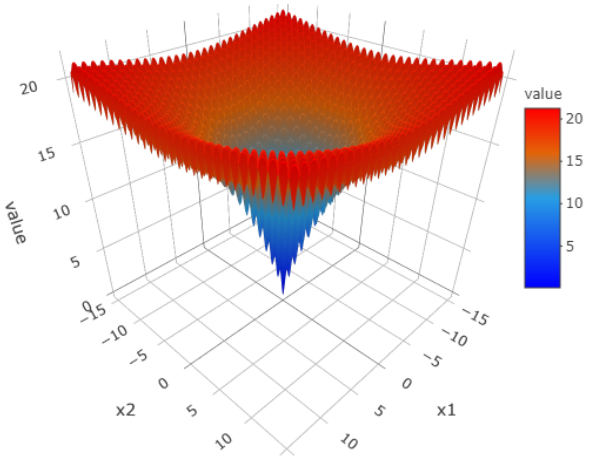
3..2 Funkcje testowe

Wzory, miejsca zerowe, wymiary oraz przestrzeń przeszukiwana zostały zawarte w tabeli 2. Wykresy funkcji zostały wygenerowane przy pomocy funkcji *plot_ly* z pakietu plotly. Pozwala on na tworzenie ładnych i wyraźnych wykresów. Pierwszą funkcją wykorzystaną do testów jest *Sphere*. Jest ona stosunkowo prosta, i bada głownie eksploatację algorytmów. Stanowi ona niejako test, którego niezaliczenie równa się z pewną porażką stworzonego rozwiązania. Wersja w trzech wymiarach jest widoczna na rysunku 3.



Rysunek 3: Wykres 3D funkcji Sphere

Drugą wykorzystaną funkcją jest Ackley. Zawiera ona wiele lokalnych optimum. Ma kształt gąbczastego leja. Parametry z których ona korzysta ustawiono na $a = 20, b - 0.2ic = 2\pi$. Jej wykres widać na rysunku 4.



Rysunek 4: Wykres Ackley

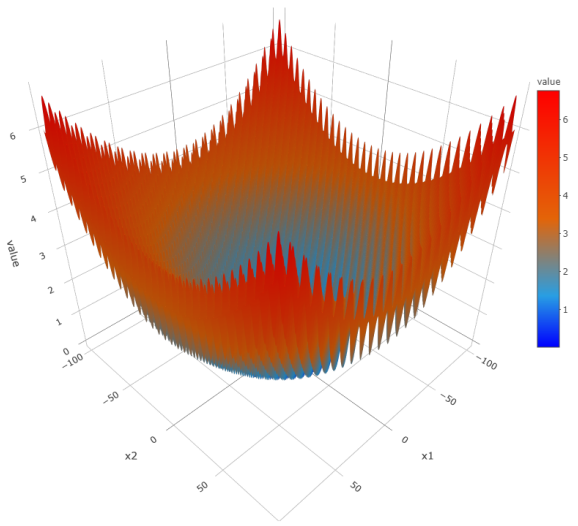
Kolejny, podobny wykres posiada Griewank. Różni się ona od Ackley wyglądem ogólnym, ale tak samo jest najeżona lokalnymi rozwiązaniami funkcji. Widać ją na rysunku 5.

Czwartym benchmarkiem jest Levy. Ma ona o wiele mniej miejsc zerowych, ale za to nie jest tak regularna. Jej wykres można zobaczyć na rysunku 6.

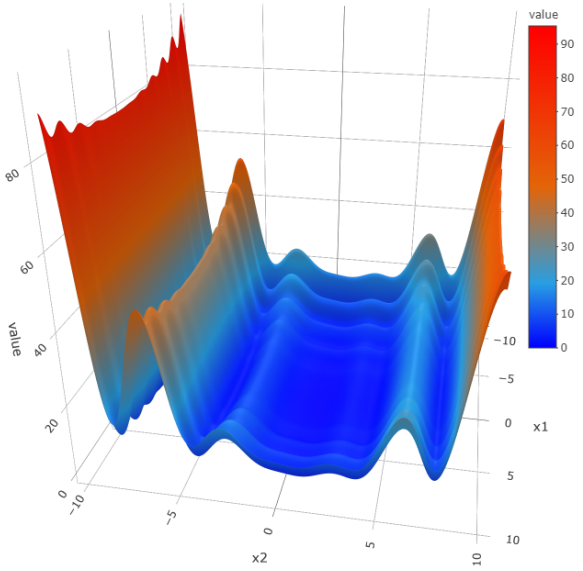
Rastrigin ponownie posiada wiele lokalnych rozwiązań, i podobny wygląd do Griewanka, ale ma mniejszą połać terenu. Wykres widać na rysunku 7.

Tabela 2: Funkcje wykorzystane do porównania skuteczności algorytmów

Funkcja	Formuła	f_{min}	Przestrzeń	Wymiar
Sphere	$F(\overrightarrow{x}) = \sum_{i=1}^{n} x_i^2$	0	(-5.12,5.12)	10
Ackley	$F(\overrightarrow{x}) = -aexp(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2} - exp(\frac{1}{d}\sum_{i=1}^{d} cos2\pi x_1) + a + exp(1))$	0	(-32.768, 32.768)	10
Griewank	$F(\overrightarrow{x}) = \frac{1}{4000}\sum_{i=1}^{n} x^2 - \Pi_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	0	(-600,600)	10
Rastrigrin	$F(\overrightarrow{x}) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]$	0	(-5.12,5.12)	10
Levy	$F(\overrightarrow{x}) = sin^2(\pi\omega_1) + \sum_{i=1}^{d}[1 + 10sin(\pi\omega_i + 1)] + (\omega_d - 1)[1 + sin^2(2\pi\omega_d)]$, $gdzie\ \omega_i = 1 + \frac{x_i - 1}{4}$, $dla\ \ i = 1, \cdot, d$	0	(-10,10)	10
Schwefel	$F(\overrightarrow{x}) = 418.9829d - \sum_{i=1}^{d}[x_i sin(\sqrt{\lvert x_i \rvert})]$	0	(-500,500)	10



Rysunek 5: Wykres Griewank



Rysunek 6: Wykres Levy

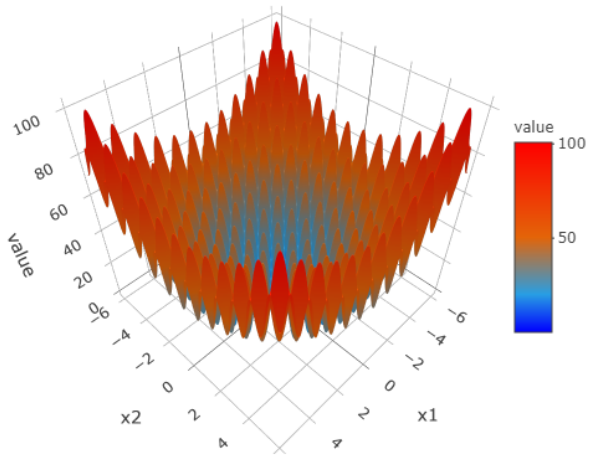
Ostatnim elementem układanki, oraz najtrudniejszym problem optymalizacyjnym jest Schwefel. Ma ona głębokie i rozległe optimum lokalne. Widać to wyraźnie na rysunku 8. Algorytmy genetyczne poradziły sobie o wiele lepiej, co jest po części związane z ich działaniem. Eksploracja jest w nich zazwyczaj o wiele lepsza, jednak często nie radzą sobie one tak dobrze z eksploatacją.

3..3 Wyniki

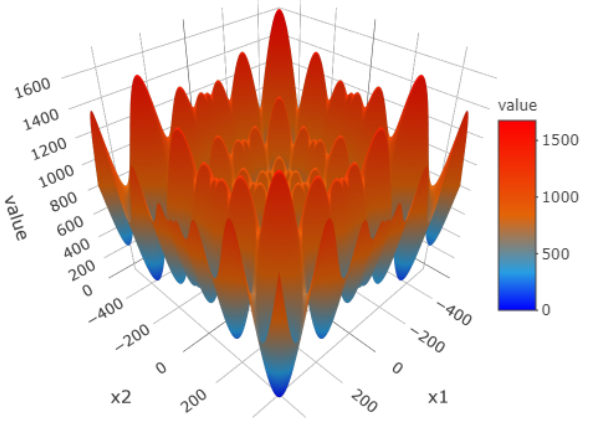
W celu porównania wyników wilczych algorytmów z innymi rozwiązaniami, postanowiono skorzystać z trzech innych algorytmów. Użyto *Genetic Algorithm* (GA) [3], *Particle swarm optimization* (PSO) [1] oraz *Differential evolution* (DE) [2]. Parametry w GA zostały zainicjalizowane w następujący sposób:

0.8 : Współczynnik krzyżowania
0.1 : Szansa na mutację

W *Particle swarm optimization* występują 4 parametry. Ich wartość ustawiono na:



Rysunek 7: Wykres Rastrigin



Rysunek 8: Wykres Schwefel

$Vmax = 2$: Prędkość maksymalna
$ci = 1.49445$: Współczynnik dążenia do najlepszego lokalnego rozwiązania
$cg = 1.49445$: Współczynnik dążenia do najlepszego globalnego rozwiązania

$w = 0.729$: Bezwładność, określająca wpływ rozwiązania znalezionego w poprzednim kroku na następny krok

Strategia i współczynniki DE wybrano następująco:

Best/1 : Strategia generowania nowych członków
0.5 : Współczynnik mutacji
0.8 : Współczynnik amplifikacji (ang. *scaling vector*)

Wyniki wszystkich testów są widoczne w tabeli 3 oraz 4. Każdy z algorytmów został przetestowany 30 razy na każdej z funkcji. Następnie wyliczono średni wynik oraz odchylenie standardowe wyników. Dodatkowo obliczono średni czas w sekundach oraz odchylenie standardowe od tych wartości. Jedynie WPA uzyskał wyniki czasu w sekundach. Każdy z algorytmów działał na ustawionym tysiącu iteracji.

Aby dalej poszerzyć wiedzę o jakości wyników, przygotowano wykresy konwergencji na funkcji Griewank. Wynik GWO, GWO-EPD, WC, WPA oraz GA widnieje na rysunku 9, a pozostałych na rysunku 10. Na pierwszym wykresie widnieje jedynie 100 iteracji, z uwagi na szybką zbieżność. Rysunki wykonano na podstawie jednego testu.

Algorytm *Grey Wolf Optimizer* jest najlepszy z obecnych tu rozwiązań wilczych. Nie poradził sobie jedynie z ostatnią funkcją. Mimo różnicy w czasie na korzyść niektórych z innych wilków, wykres zbieżności pokazuje że jedynie WPA jest od niego szybszy i to jedynie pod względem iteracji. Poza tym, oferuje lepsze rozwiązania w funkcji Levy i Rastrigin. Działa lepiej od *Particle swarm optimization*, który jest uznanym algorytmem. Jego jedynymi rywalami są oba algorytmy genetyczne, od których odróżnia go precyzja. Lepiej eksploatuje on rozwiązania globalne. W funkcji Schwefel poradziły sobie one lepiej od niego, choć jedynie GA w sposób znaczny (o dwie wielkości). Ogólnie jednak rzecz biorąc, widać dlaczego jest to najbardziej cytowany algorytm wilczy z wybranych w tej pracy.

Grey Wolf Optimizer with Evolutionary population jest modyfikacją niewiele różniącą się od oryginału. Choć autorzy mogą się w teorii powołać na *No free lunch theorems for optimization* [15], który traktuje o tym, że każdy algorytm aby zyskać coś w jakiejś klasie problemów musi coś poświęcić w innej, to nie widać aby w tym przypadku się to udało. Wyniki obu algorytmów są prawie nierozróżnialne, z czego największa różnica jest w ostatniej funkcji. Niestety na niekorzyść modyfikacji względem oryginalnego algorytmu. Należałoby potraktować to raczej jako porażkę. W końcu celem zmian ma zazwyczaj być poprawa jakości.

Wolf colony sprawił się jako trzeci z wilczych rozwiązań. Zaoferował on bardzo podobne wyniki jak PSO, jednak często od niego gorsze. Mimo tego znalazł się na tym miejscu z uwagi na jego w miarę normalny czas, mimo faktu iż był wolniejszy od wszystkich oprócz WPA. Dodatkowo zaoferował podobną zbieżność jak GWO, nieznacznie mu ustępując. Był też do tego o wiele szybszy w konwergencji niż *Particle swarm optimization*. Poradził sobie też lepiej w ostatniej funkcji od pozostałych wilków i roju cząstek. Ogólnie więc mówiąc, jest to dobry algorytm, jednak nie można mówić tu o znacznej

różnicy względem klasycznego roju.

Biorąc pod uwagę dobre wyniki swojego oryginału, nie można się dziwić, że ktoś spróbował polepszyć algorytm wilczej kolonii. Jednak droga podjęta przez twórców *Wolf pack* jest definitywnie problematyczna. Na pierwszy rzut oka widać ogromny wzrost czasu względem innych algorytmów. Mimo iż teoretycznie zbieżność jest osiągana już w dosłownie kilku iteracjach, jest to okupione potwornym czasem wykonania w ustalonej ilości pętli. Dodatkowo, algorytm ten najbardziej poległ w ostatniej funkcji. Ciężko więc mówić o dużym sukcesie. Dodatkowo, różnica między zerem a 0.1 nie jest tak duża aby pozwolić sobie na taką różnice w czasie. W Levy algorytm też nie uzyskał najlepszego wyniku. Mimo tego, został on uznany za lepszego od pozostałych dwóch algorytmów wilczych.

Wolf search algorithm boryka się z dużą ilością problemów. Brak zawartych domyślnych wartości parametrów sterujących mógł wpłynąć na wynik, i nie sposób stwierdzić jak duże miało to znaczenie. Dodatkowo, z uwagi na mechanizm raczej należący do domen algorytmów optymalizujących funkcje dyskretne, można wyżej przeczytać o braku jego implementacji. Sam styl napisania publikacji też pozostawia wiele do życzenia w kwestii wytłumaczenia różnych operacji tego rozwiązania. Finalnie, wyniki tu uzyskane są najgorsze z obecnych, i zawsze słabsze niż klasyczny algorytm, PSO. Nie ma aż tak ogromnej różnicy żeby mówić o kompletnej porażce, ale nie ma tez powodu żeby z tego rozwiązania korzystać. Zbieżność widoczna na wykresie konwergencji jest bardzo skokowa, co dalej podkopuje pozycję WSA.

Modyfikacja WSA, *Gaussian Guided Self-Adaptive Wolf Search Algorithm*, nie różni się wiele od swojego oryginału. Popełniła ona podobne błędy jeśli chodzi o zawarcie wartości parametrów sterujących i tak samo ominęła dokładne wytłumaczenie pewnych kluczowych operacji. Zastosowanie ciekawego podejścia, jakim jest zmiana parametrów, nie przyniosło większych sukcesów. Wyniki względem oryginału są raz lepsze a raz gorsze. Dodatkowo, zapewne przez wprowadzenie dodatkowych obliczeń, algorytm jest wolniejszy. Ocena więc należy się podobna, jeśli nie wręcz gorsza od oryginału - w obecnym stanie brak powodów aby z tego rozwiązania korzystać.

4. Wnioski

Jak widać, wilcze algorytmy są w stanie zaoferować lepsze bądź podobne wyniki w porównaniu z innymi klasycznymi rozwiązaniami. Niewątpliwe najlepszy okazał się *Grey Wolf Optimizer*. Jedynie algorytmy ewolucyjne w jednym przypadku podważyły jego umiejętności. Jednak mimo tego, podobnie jak PSO, dalej nadaję się on do wykorzystania w realnych problemach optymalizacyjnych. Biorąc pod uwagę jego wysokie wskaźniki cytowań, inni badacze bez wątpienia zgadzają się z tym faktem. Modyfikacja GWO, dodająca mechanizm ewolucyjny nie wniosła za wiele do tematu, nieznacznie pogarszając wyniki.

Kolejne dwa algorytmy, *Wolf Colony* oraz *Wolf Pack* uzyskały gorszą ocenę, głównie ze względu na ich czas wykonania, i gorsze osiągi w części funkcji testowych. Mimo tego są porównywalne bądź lepsze od ich konkurenta rojowego, PSO.

Największą porażką na pewno odznaczyły się Wilk szukający oraz jego modyfikacja Gaussa. Wyraźne nagłe skoki w znajdywaniu lepszych rozwiązań są negatywnym zjawiskiem w konwergencji. Nie pozwala to dobrze określić optymalnego czasu w którym algorytm znajdzie rozwiązanie. Ogólnie rzecz biorąc wyniki tych rozwiązań nie są takie złe, ale po prostu istnieją lepsze alternatywy. Na przykład te wskazane w niniejszej pracy.

Podsumowując, algorytmy wilczego stada nadają się do rozwiązywania problemów optymalizacyjnych. Widać wyraźnie czemu te drapieżniki są często podejmowane jako inspiracja do tworzenia nowych rozwiązań bądź modyfikowania już istniejących. Algorytmy te pokazują nowe i ciekawe podejścia do od dawna znanego problemu.

Bibliografia

[1] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[2] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. 11(4), 1997.

[3] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–73, 1992.

[4] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46 – 61, 2014.

[5] Osman K. Erol and Ibrahim Eksin. A new optimization method: Big bang–big crunch. *Advances in Engineering Software*, 37(2):106 – 111, 2006.

[6] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232 – 2248, 2009. Special Section on High Order Fuzzy Sets.

Tabela 3: Wyniki algorytmów optymalizacyjnych.

Funkcja	Algorytm	Średnia	Odchylenie	Średni Czas[s]	Odchylenie[s]
Sphere $f_{min}(\overrightarrow{x}) = 0$	GWO	2.0553e-177	0	10.5816	2.5157
	GWO-EPD	7.7542e-176	0	11.8804	2.4582
	WC	0.2376	0.1363	15.7010	0.1884
	WPA	0	0	1.1865min	0.1166min
	WSA	3.7366	1.2116	4.7903	0.1740
	GSAWSA	8.8820	2.4832	6.5562	0.7882
	PSO	5.5897e-56	1.1653e-55	6.9985	1.2662
	DE	1.7554e-42	2.4270e-42	0.4618	0.0437
	GA	0.0015	0.0011	1.1592	0.1334
Ackley $f_{min}(\overrightarrow{x}) = 0$	GWO	4.9441e-15	1.5979e-15	15.6035	1.3637
	GWO-EPD	4.8257e-15	1.5283e-15	15.0345	1.4891
	WC	1.5461	0.5477	20.6302	0.3285
	WPA	4.4408e-16	0	1.7590min	0.2256min
	WSA	17.0793	1.1482	5.6244	0.3050
	GSAWSA	16.3092	1.1823	7.2237	0.2136
	PSO	3.9968e-15	0	7.3	0.4276
	DE	3.9968e-15	0	0.7036	0.0693
	GA	0.4647	0.1954	1.4450	0.1222
Griewank $f_{min}(\overrightarrow{x}) = 0$	GWO	0.0141	0.0160	13.8200	1.8674
	GWO-EPD	0.0225	0.0323	12.3534	1.8143
	WC	2.3631	1.3632	19.1972	1.6045
	WPA	0	0	1.4091min	0.1103min
	WSA	53.9747	16.0915	6.0211	0.2803
	GSAWSA	41.0102	9.9708	7.9068	0.3944
	PSO	0.5386	0.3019	6.7454	0.109
	DE	0.0441	0.0192	1.0485	0.2405
	GA	0.5803	0.2304	1.2187	0.07
Rastrigin $f_{min}(\overrightarrow{x}) = 0$	GWO	0.3323	1.2917	11.3405	1.0982
	GWO-EPD	0.4318	1.3261	11.4035	1.1253
	WC	20.3965	7.1699	17.8619	0.2876
	WPA	0	0	1.2134min	0.0215min
	WSA	60.0253	4.8182	5.6589	0.3012
	GSAWSA	65.6314	5.9479	8.2285	1.5928
	PSO	5.5054	2.5831	6.2371	0.2680
	DE	2.1557	1.3344	0.8224	0.1119
	GA	0.3053	0.2346	1.0348	0.1174

Tabela 4: Wyniki algorytmów optymalizacyjnych cz. 2.

Funkcja	Algorytm	Średnia	Odchylenie	Średni Czas[s]	Odchylenie[s]
Levy $f_{min}(\overrightarrow{x}) = 0$	GWO	0.0376	0.05	12.8763	0.9715
	GWO-EPD	0.0478	0.0512	11.7281	1.1759
	WC	0.1735	0.1340	23.8095	0.3820
	WPA	0.5393	0.2329	1.7894min	0.04061min
	WSA	8.2483	2.0257	6.001	0.2625
	GSAWSA	8.5194	2.2857	7.4072	0.2403
	PSO	1.4996e-32	0	8.4826	0.3885
	DE	1.4996e-32	0	1.0607	0.1201
	GA	0.0032	0.0021	1.1665	0.0920
Schwefel $f_{min}(\overrightarrow{x}) = 0$	GWO	969.3719	284.6110	12.2941	0.3728
	GWO-EPD	1096.2342	252.9833	11.3166	1.9551
	WC	1426.7238	320.7987	16.6173	0.3092
	WPA	2358.2336	246.4415	1.2656min	0.0299min
	WSA	1905.5797	147.8068	5.5269	0.3443
	GSAWSA	1731.03105	145.8400	6.7911	0.1325
	PSO	2053.4427	426.3038	6.0361	0.1060
	DE	687.7272	355.6033	0.7139	0.0712
	GA	1.9594	1.0994	1.0676	0.1832

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[8] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[9] C. Yang, X. Tu, and J. Chen. Algorithm of marriage in honey bees optimization based on the wolf pack search. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*, pages 462–467, 2007.

[10] Rolf Peterson, Amy Jacobs, Thomas Drummer, L. Mech, and Douglas Smith. Leadership behavior in relation to dominance and reproductive status in gray wolves, canis lupus. *Canadian Journal of Zoology-revue Canadienne De Zoologie - CAN J ZOOL*, 80:1405–1412, 08 2002.

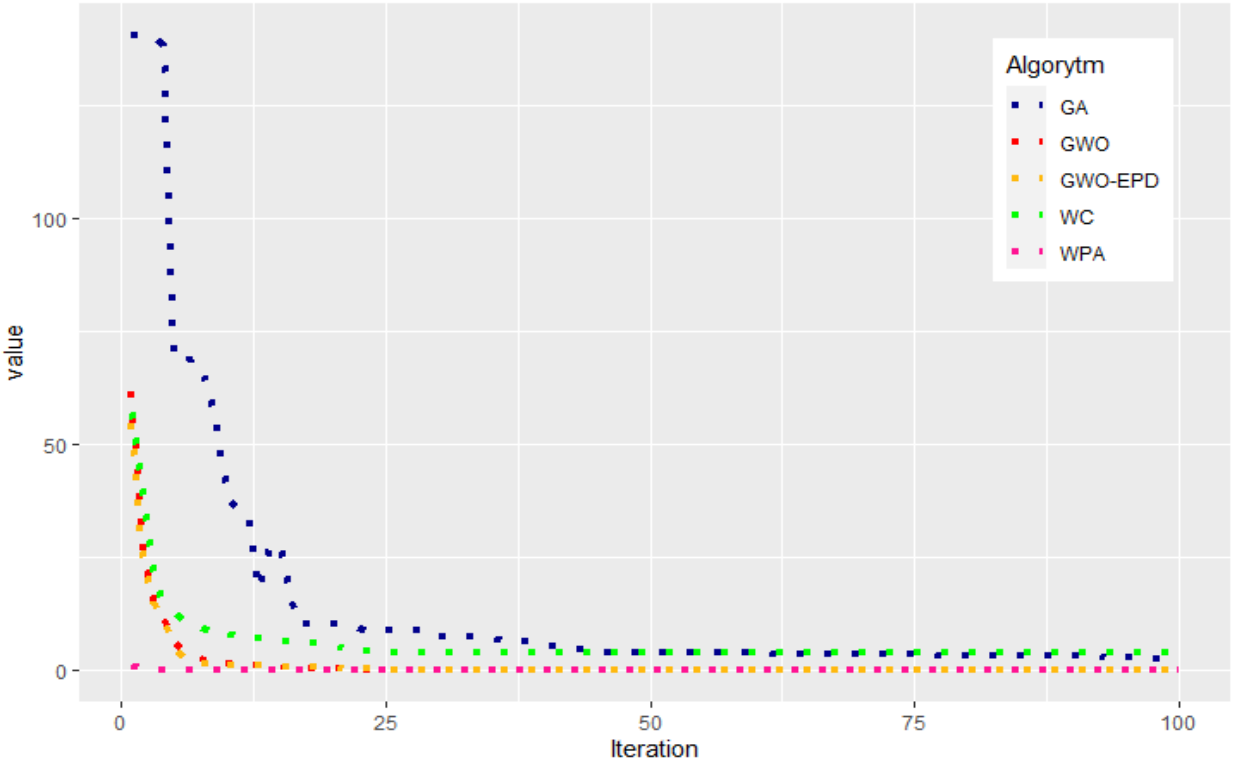
[11] L. Mech. Alpha status, dominance, and division of labor in wolf packs. *Canadian Journal of Zoology*, 77:1196–1203, 1999.

[12] C.-Y Liu, X.-H Yan, and H. Wu. The wolf colony algorithm and its application. *Chinese Journal of Electronics*, 20:212–216, 04 2011.

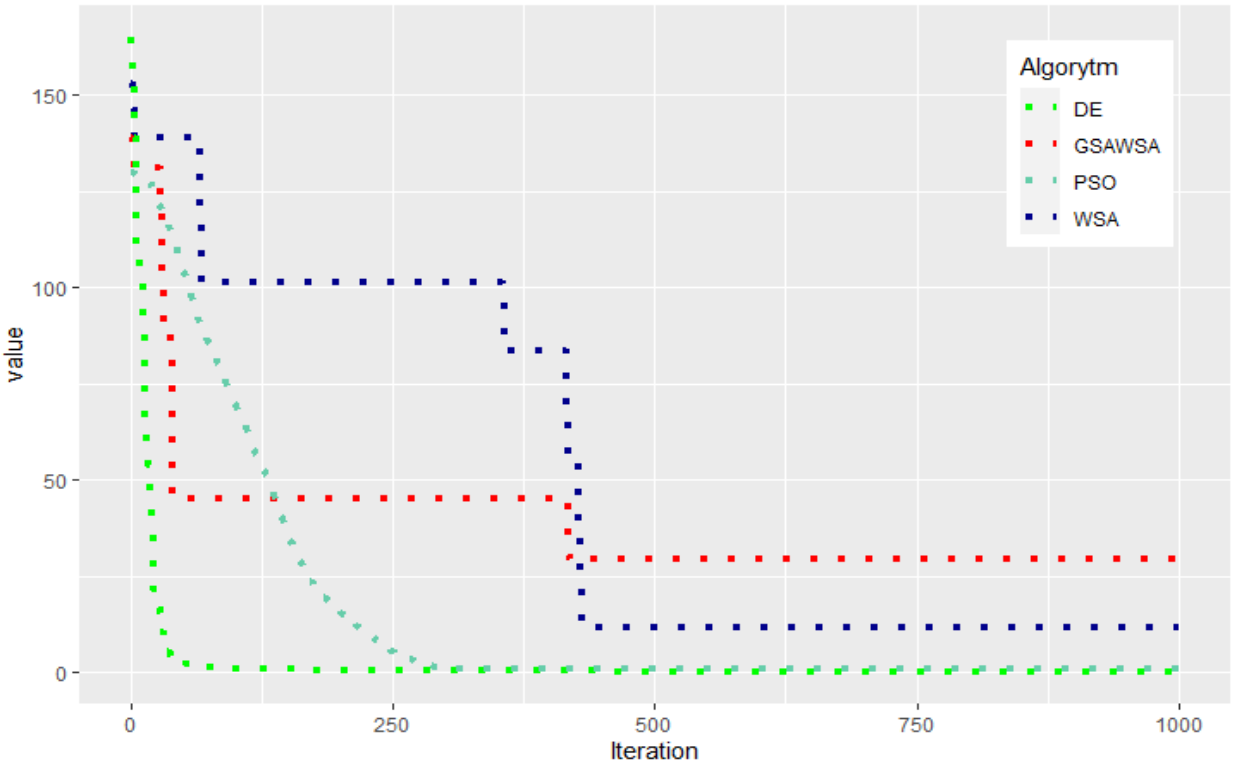
[13] Hu-Sheng Wu and Feng-Ming Zhang. Wolf pack algorithm for unconstrained global optimization. *Mathematical Problems in Engineering*, 2014:1–17, 03 2014.

[14] Derek Bingham Sonja Surjanovic. *Optimization Test Functions*, 2013.

[15] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.



Rysunek 9: Wyniki GWO, GWO-EPD, WC, WPA, GA w funkcji Griewank.



Rysunek 10: Wyniki WSA, GSAWSA, PSO oraz DE w funkcji Griewank.

Studia i Materiały Informatyki Stosowanej, Tom 13, Nr 1, 2021
str. 37-41

TECHNOLOGIA BLOCKCHAIN

Maciej Sitko[1], Mieczysław Jagodziński[2]

[1] *Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, 44-101 Gliwice, ul. Akademicka 16*
[2] *Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, 44-101 Gliwice, ul. Akademicka 16*
email: mieczyslaw.jagodzinski@polsl.pl

Streszczenie: *W artykule przedstawiono zasadę działania blockchainu na podstawie Bitcoina oraz smart kontraktów Ethereum. Zostały omówione najistotniejsze zalety technologii blockchain. Zwrócono szczególną uwagę na transparentność, decentralizację oraz bezpieczeństwo proponowanego rozwiązania. Opisano szczegóły funkcjonalne oraz techniczne. Przedstawiono przykładowe zastosowanie blockchainu.*

Słowa kluczowe: *Blockchain, Bitcoin, Ethereum, decentralizacja, inteligentne kontrakty*

Blockchain technology

Abstrac*t: The article presents the principle of blockchain operation based on Bitcoin and Ethereum smart contracts. The most important advantages of blockchain technology were discussed. Particular attention was paid to transparency, decentralization and security of the proposed solution. The functional and technical details are described. An example of the use of blockchain is presented*

Key words: *Blockchain, Bitcoin, Ethereum, decentralization, smart contracts*

1. Wprowadzenie

Technologia blockchain zyskuje w ostatnich latach coraz większą popularność. Początkowo łańcuch bloków znajdował swoje zastosowanie przede wszystkim w branży finansowej w postaci kryptowalut, czyli wirtualnych pieniędzy. Decentralizacja jaką zapewnia blockchain stworzyła alternatywę dla tradycyjnych systemów bankowych i zaproponowała rozwiązanie, które eliminuje potrzebę ingerencji organów trzecich w wymianę środków pomiędzy pojedynczymi podmiotami. Blockchain zapewnia także wysoki poziom bezpieczeństwa przechowywanych danych. Specjalnie zaprojektowana struktura blockchainu nie pozwala na zmianę zapisanych w nim informacji – możliwy jest jedynie ich odczyt. Ponadto wszystkie akcje podejmowane w blockchainie udostępniane są w sieci publicznej. Zapewnia to transparentność wykonywanych transakcji, z zachowaniem anonimowości podmiotów biorących w nich udział. Wraz z rozwojem tej technologii znaleziono także rozwiązania pozafinansowe, do których można zaimplementować architekturę łańcucha bloków. Było to możliwe za sprawą inteligentnych kontraktów (ang. smart contracts).

Blockchain to technologia, która opiera swoje działanie na publicznym, rozproszonym rejestrze danych zachowującym chronologię wykonanych operacji. Rejestr udostępniany jest w sieci o architekturze peer – to – peer. Sam termin „blockchain" oznacza „łańcuch bloków". Bloki to proste struktury danych, które są podstawą działania blockchainu. Przechowuje się w nich transakcje, będące bezpośrednimi dowodami każdej akcji podjętej w sieci. Każdy kolejny stworzony blok posiada referencję do bloku poprzedniego, stąd określenie „łańcuch bloków".

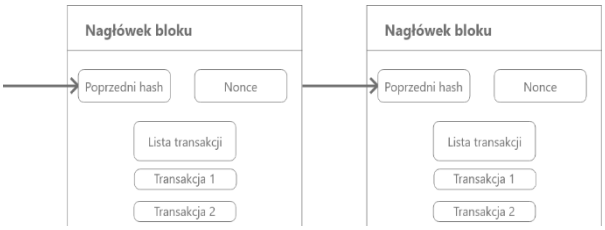
Pierwsze próby stworzenia blockchainu datuje się na końcówkę XX wieku. Mimo to pierwsze w pełni działające rozwiązanie zostało opublikowane w 2008 roku przez Satoshiego Nakamoto – założyciela Bitcoina [1]. Stworzył on kryptowalutę, która opierała swoje działanie na łańcuchu bloków. Kolejny przełom w tej dziedzinie odbył się za sprawą Vitalika Buterina, który w 2013 roku udoskonalił rozwiązanie Nakamoto o smart kontrakty, czyli krótkie programy skryptowe, które znacznie rozszerzają możliwości zarządzania transakcjami w blockchainie. W ten sposób powstało Ethereum, które umożliwiło tworzenie pierwszych zdecentralizowanych aplikacji. W artykule zostanie opisane działanie blockchainu na podstawie Bitcoina oraz krótko scharakteryzowane zostaną smart kontrakty.

2. Blok

Blok to podstawowa struktura danych w blockchainie, któremu technologia ta zawdzięcza

swoją nazwę. Dołączanie kolejnych bloków do łańcucha to podstawowe zadanie, które wykonują użytkownicy w sieci. Uproszczony schemat bloków przedstawiono na rys. 1. W rzeczywistości bloki składają się z kilkunastu pól zawierających informacje niezbędne do funkcjonowania systemu. Najważniejszymi z nich są:

- numer hasz poprzedniego bloku,
- czas utworzenia bloku,
- trudność bloku,
- liczba „nonce”,
- lista transakcji.

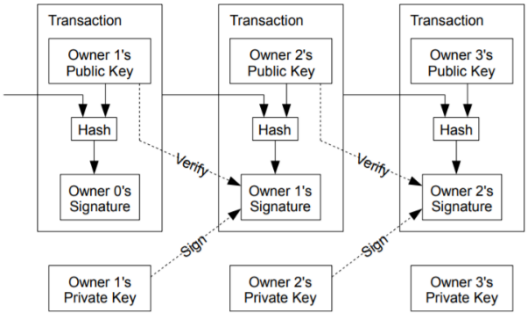


Rysunek 1. Uproszczony schemat bloków w blockchainie. [opracowanie własne]

Każdy blok posiada swój unikalny numer, który powstaje przez zahaszowanie zawartości tego bloku. Funkcja haszująca w Bitcoinie opiera się o szyfrowanie SHA-256, które dowolnie dużej liczbie przyporządkowuje ciąg znaków o stałej długości 256 bitów. Jest to funkcja jednokierunkowa, tzn. prawie niemożliwe jest odkodowanie zaszyfrowanej liczby. Ponadto nawet najmniejsza zmiana w liczbie zaszyfrowanej powoduje całkowitą zmianę wyniku szyfrowania. Obliczenie numeru hasz nazywa się „wykopywaniem bloku” (ang. mining), a osoby podejmujące się tego zadania określa się mianem „górników” (ang. miners). Za wykonanie tego zadania jako pierwszy, górnik otrzymuje nagrodę pieniężna, na którą składają się opłaty z zawartych w bloku transakcji. Wykopanie bloku stanowi dowód pracy (ang. proof-of-work), świadczący o uczciwości użytkownika sieci. Musi on bowiem poświęcić moc obliczeniową swojego urządzenia na wyliczenie numeru hasz, który warunkowany jest przez ustaloną trudność. Trudność bloku to szacunkowa liczba prób obliczenia haszu, jaką musi podjąć użytkownik, aby numer ten spełniał postawione przez blok wymagania. Wymaganiem jest, aby numer hasz był mniejszy niż ustalona liczba, tzw. target. Funkcja haszująca jest na tyle złożona, że niemożliwe jest obliczenie haszu mniejszego niż target innym sposobem niż metodą prób i błędów. Bitcoin zwiększa trudność bloku co 2016 wykopanych bloków, czyli co około 2 tygodnie. Obecnie trudność wynosi 21,448,277,761,059 (stan na 10.03.2021r.). Wykopywanie bloku polega na inkrementacji liczby „nonce”, która jest jednym z wejść funkcji haszującej. Wartości pozostałych wejść są niezmienne, dlatego należy znaleźć taką liczbę „nonce”, która spowoduje, że wyjście funkcji haszującej będzie mniejsze niż określony target. Połączenie kolejnych bloków poprzez referencje do numerów hasz gwarantuje niezmienność danych zapisanych w blockchainie. Zmiana dowolnej wartości zapisanej transakcji powoduje całkowitą zmianę numeru hasz bloku. Oznacza to, że kolejne hasze bloków, które referowały do poprzedniego haszu są błędne i taki atak jest od razu zauważalny przez sieć.

3. Transakcje

Bitcoin został stworzony jako elektroniczna waluta. Podstawową operacją w tym blockchainie jest transfer pieniędzy, który realizowany jest poprzez transakcje. Aby transakcja mogła dojść do skutku, adresat oraz nadawca transakcji muszą należeć do sieci blockchain, tzn. muszą posiadać swój klucz publiczny i prywatny. Klucz publiczny służy do identyfikacji użytkownika i jest niejako adresem, który używany jest do korespondowania w sieci. Klucz prywatny stosuje się do podpisywania transakcji. Na jego podstawie oblicza się podpis cyfrowy, który służy do weryfikacji adresata. Podobnie jak blok, każda transakcja posiada swój numer hasz. Jest on obliczany na podstawie numeru hasz poprzedniej transakcji, danych obecnej transakcji oraz klucza publicznego odbiorcy. Można więc stwierdzić, że kolejne transakcje, podobnie jak bloki, również są połączone w łańcuch. Schemat takiego połączenia przedstawiono na rys. 2.



Rysunek 2. Schemat połączenia kolejnych transakcji [opracowanie własne na podstawie [1]]

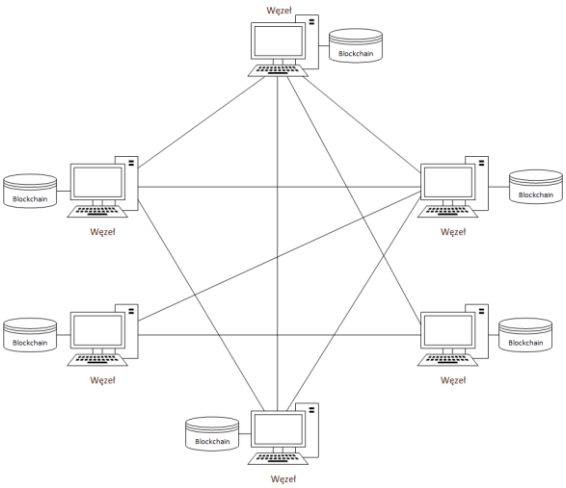
Początkowo transakcje można wyobrazić sobie jako transfer pewnej kwoty x od użytkownika A do użytkownika B. Wydaje się więc, że po udanej transakcji, kwota x zostaje odjęta od stanu konta użytkownika A, a następnie dodana do stanu konta użytkownika B. Jednak w Bitcoinie mechanizm ten funkcjonuje zupełnie inaczej. Transakcje polegają

na przesyłaniu środków jako monet (ang. coin), które tworzone są na podstawie poprzednich transakcji. Nowa transakcja na wejściu otrzymuje referencje do poprzednich transakcji i na ich podstawie tworzy wyjście. Transakcje, podobnie jak rzeczywiste monety, są niepodzielne. W przypadku, gdy łączna wartość użytych monet jest większa niż wartość wysyłana, nadawcy zwracana jest reszta. Można więc powiedzieć, że portfel Bitcoin to nie pojedyncza liczba stanowiąca saldo konto, a łańcuch wszystkich przeprowadzonych transakcji.

Po wysłaniu przez użytkownika operacji, transakcja trafia do puli. Za weryfikację poprawności transakcji odpowiedzialne są węzły sieci. Czas oczekiwania na zatwierdzenie transakcji w Bitcoinie to około 10 minut. Czas ten może być krótszy, jeśli nadawca dołączy do transakcji dodatkową opłatę transakcyjną. Taka operacja jest znacznie bardziej atrakcyjna dla górników, którzy otrzymają za nią większą nagrodę.

4. Sieć

Blockchain to publiczny rejestr, który dostępny jest dla każdego użytkownika. Umożliwia to sieć o architekturze peer – to – peer, przedstawiona na rys. 3.



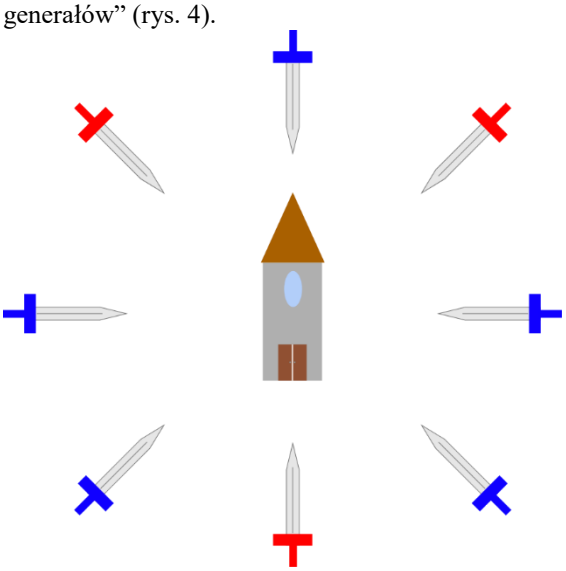
Rysunek 3. Schemat sieci peer – to – peer podłączonej do blockchainu. [opracowanie własne na podstawie [1]]

Każdy użytkownik może pobrać kopię rejestru na swoje urządzenie. Zapewnia to transparentność dokonywanych transakcji. Użytkowników sieci blockchain można podzielić na kilka typów:

- pełny węzeł,
- lekki węzeł,
- górnik.

Najważniejsze dla sprawnego funkcjonowania sieci są pełne węzły (ang. full nodes). Przechowują one pełną kopię blockchainu. Odpowiadają także za weryfikację wszystkich transakcji i bloków oraz dystrybucję tych informacji w całej sieci. Z uwagi na duży rozmiar pełnej kopii łańcucha (obecnie jest to około 140 GB danych), coraz popularniejsze są lekkie węzły (ang. light nodes). Nie przechowują one pełnej kopii łańcucha, ale same nagłówki bloków o maksymalnej wielkości 80 bajtów, co znacznie redukuje eksploatację sprzętu. Ostatnim typem użytkowników są górnicy, którzy pobierają tylko dane ostatnie bloku, żeby wyliczyć nowy hasz.

Jednak w rozwiązaniu opierającym się na topologii sieci peer – to - peer pojawia się problem ze zgodnością przechowywanych wersji. W jaki sposób dojść do porozumienia w sprawie kolejności transakcji oraz ich poprawności. Sytuacja ta określana jest jako „problem bizantyjskich generałów" (rys. 4).



Rysunek 4. Wizualizacja „problemu bizantyjskich generałów".[opracowanie własne na podstawie [3]]

Problem został sformułowany następująco przez Marshalla Pease'a, Leslie Lamporta i Roberta Shostaka:

„Grupa armii bizantyjskich otacza miasto nieprzyjaciela. Rozkład sił jest taki, że jeśli wszystkie armie zaatakują razem, to będą w stanie zdobyć miasto. Innym sposobem uniknięcia porażki jest odwrót wszystkich armii. Generałowie poszczególnych armii mają zaufanych posłańców, którzy z powodzeniem dostarczą każdy komunikat od jednego generała do innego. Jednak niektórzy generałowie mogą być zdrajcami usiłującymi doprowadzić do porażki armii bizantyjskich. Należy opracować algorytm, który umożliwi wszystkim wiernym generałom uzgodnienie pewnego planu działania. Ostateczna decyzja powinna być z grubsza taka, jaka zostałaby podjęta w drodze głosowania większościowego nad decyzjami poszczególnych generałów. W przypadku

nierozstrzygnięcia głosowania końcową decyzją ma być odwrót". [3]

Jako pierwszy, zadanie to rozwiązał Satoshi Nakamoto – twórca Bitcoina. Dowodem uczciwości uczestników sieci jest dowód pracy (proof – of – work) wykonanej przy obliczaniu numeru hasz. Ponadto transakcja zostaje dodana do bloku dopiero wtedy, gdy ponad połowa uczestników sieci ją zaakceptuje, a także zakłada się, że co najmniej połowa użytkowników sieci jest uczciwa. W takiej sytuacji nieopłacalne staje się bycie nieuczciwym, ponieważ szansa na oszustwo jest bardzo niska a każda taka próba obarczona kosztem eksploatacji sprzętu oraz Internetu. Przypadek gdy ponad połowa użytkowników sieci jest nieuczciwa określa się jako „atak 51%".

5. Bezpieczeństwo

Innym problemem sieci rozproszonej jest „problem podwójnego wydatkowania" (ang. double spending problem). Powstaje on w systemach, gdzie nie ma jednostki centralnej, która określa kolejność wykonanych transakcji. W celu rozwiązania tego problemu Satoshi Nakamoto wprowadził znaczniki czasowe (ang. timestamps), które jednoznacznie definiują chronologię wykonanych transakcji. W przypadku gdy kilka bloków zostanie stworzonych w tym samym momencie, część użytkowników może wybrać inny blok jako ten stworzony według nich najszybciej z uwagi na różny czas propagacji informacji w sieci. Przy tworzeniu kolejnego bloku szansa na wystąpienie takiej sytuacji jest jeszcze mniejsza, więc zachowany zostanie ten blok, który wraz z kolejnymi blokami będzie tworzył najdłuższy łańcuch, tzw. reguła najdłuższego łańcucha. Polega ona na cyklicznym sprawdzaniu, który łańcuch w blockchainie jest najdłuższy, a dokładniej, który łańcuch ma największą sumaryczną trudność. Połączenie rejestru znaczników czasu transakcji i rejestru dopasowującego klucze publiczne do prywatnych było kluczowym czynnikiem działania systemu stworzonego przez Satoshiego.

Należy rozpatrzeć także sytuację, gdy nieuczciwy użytkownik sieci dokona próby dołączenia własnej kopii łańcucha. Zgodnie z regułą najdłuższego łańcucha, musi stworzyć łańcuch o co najmniej jeden blok dłuższy niż aktualny blockchain. Problem, z którym musi się zmierzyć to tzw. problem ruiny gracza (ang. Gambler's ruin). Nieuczciwy użytkownik zawsze zaczyna wyścig o obliczanie nowego bloku z pewnym deficytem. Musi on bowiem obliczyć hasze wszystkich poprzednich bloków, a także hasz nowego bloku wcześniej niż wszyscy użytkownicy sieci obliczą i zatwierdza nowy blok. W dokumencie technicznym Satoshi Nakamoto obliczył, że dla łańcucha składającego się z 1 bloku, prawdopodobieństwo takiego zdarzenia wynosi 20%, przy 5 blokach wynosi już 0,09%. Przy obecnej liczbie bloków Bitcoina sytuacja ta staje się praktycznie niemożliwa, jednak wciąż stanowi niebezpieczeństwo dla mniejszych blockchainów.

6. Smart kontrakty

Smart kontakty to krótkie programy skryptowe, które stanowią podstawę działania blockchainu Ethereum. Są także głównym czynnikiem wyróżniającym Ethereum na tle Bitcoina. Smart kontrakty umożliwiają zarządzanie funkcjonalnościami blockchainu za pośrednictwem dodania warunków logicznych do transakcji oraz interakcji pomiędzy kontraktami. Ich działanie opiera się o EVM (Ethereum Virtual Machine), czyli wirtualną maszynę autorstwa Ethereum, która odpowiada za wykonywanie instrukcji zapisanych w kontraktach.

Swoją popularność Ethereum zawdzięcza także tzw. tokenom. Za pośrednictwem smart kontraktów możliwe jest stworzenie wirtualnych żetonów, którym przypisuje się funkcje określone przez dane standardy. Obecnie najpopularniejszym z nich jest ERC-20. Używanie standardów pozwala na jednorodną implementację funkcjonalności, która jest niezbędna przy tak łatwym dostępie do stworzenia własnego tokenu. Token może pełnić rolę pieniądza, akcji bądź usługi

7. Podsumowanie

Decentralizacja działań w sieci jest tematem coraz częściej poruszanym w dzisiejszych czasach, a wizję Internetu pozbawionego kontroli jednostek centralnych określa się mianem Web 3.0. Rynek aplikacji opartych o blockchain rozwija się dynamicznie, a zastosowanie tego rozwiązania można zauważyć w wielu branżach pozafinansowych. Przykładem jest branża spożywcza, która umożliwia przechowywanie danych o dystrybuowanych produktach za pomocą blockchainu oraz monitorowanie daty przydatności do spożycia za pośrednictwem smart kontraktów. Innym przykładem jest branża nieruchomości. Za pomocą tokenów możliwe jest sprzedaż tylko części budynku czy ziemi, której wartość poddaje się tokenizacji, czyli przypisaniu tokenowi określonej części wartości danego dobra.

Literatura

1. Nakamoto S. Bitcoin: *A Peer-to-Peer Electronic Cash System* https://bitcoin.org/bitcoin.pdf [data dostępu: 2021-03-10]

2. Sitko Maciej, Praca inżynierska *„Smart contracts z wykorzytaniem technologii blockchain”*
3. Wikipedia, wolna encyklopedia. *Problem generałów bizantyjskich.* https://pl.wikipedia.org/wiki/Problem_bizantyjskich_genera%C5%82%C3%B3w [data dostępu: 2021-03-10]

Studia i Materiały Informatyki Stosowanej, Tom 13, Nr 1, 2021
str. 42-46

ZDECENTRALIZOWANE APLIKACJE Z WYKORZYSTANIEM TECHNOLOGII BLOCKCHAIN

Maciej Sitko[1], Mieczysław Jagodziński[2]

[1] *Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, 44-101 Gliwice, ul. Akademicka 16*
[2] *Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, 44-101 Gliwice, ul. Akademicka 16*
email: mieczyslaw.jagodzinski@polsl.pl

Streszczenie: *W artykule przedstawiono przykładową zdecentralizowaną aplikację wykorzystującą blockchain Ethereum. Aplikacja w sposób uproszczony symuluje działanie systemu zarządzającego łańcuchem dostaw, który do przechowywania danych produktu wykorzystuje smart kontrakty. Wskazano zalety wykorzystania technologii blockchain w podanym przypadku oraz zaproponowano możliwe scenariusze rozwoju aplikacji w przyszłości.*

Słowa kluczowe: *Blockchain, Ethereum, decentralizacja, inteligentne kontrakty, łańcuch dostaw*

Decentralized applications using blockchain technology

***Abstrac**t: The article presents an example of a decentralized application using the Ethereum blockchain. The application in a simplified way simulates the operation of the supply chain management system, which uses smart contracts to store product data. The advantages of using blockchain technology in the given case were indicated and possible application development scenarios in the future were proposed.*

***Key words:** Blockchain, Ethereum, decentralization, smart contracts, supply chain*

1. Wprowadzenie

Technologia blockchain swoją popularność w ostatnich latach zawdzięcza nie tylko licznym zastosowaniom finansowym, ale przede wszystkim rozwiązaniom wprowadzonym w branżach pozafinansowych. Przełom ten nastąpił w 2013 roku za sprawą Vitalika Buterina oraz założonego przez niego Ethereum, z czego skorzystały m.in. branża spożywcza, transportowa czy nieruchomości. Platforma pozwala deweloperom na tworzenie własnych aplikacji opierających swoje działanie o blockchain, czyli tzw. dApps (ang. decentralized applications). Zdecentralizowane aplikacje podłączone są do EVM (Ethereum Virtual Machine), która wykonuje instrukcje zawarte w inteligentnych kontraktach. Inteligentne kontrakty (ang. smart contracts) to krótkie programy skryptowe znacznie rozszerzające możliwości interakcji użytkownika z blockchainem. W porównaniu do Bitcoina, transakcje nie muszą oznaczać tylko transferu pieniędzy. Za pośrednictwem transakcji w Ethereum można przesyłać dowolne dane, które następnie zapisywane są w blokach. Chociaż z założenia Ethereum to platforma służąca do obsługi smart kontraktów, twórcy Ethereum wprowadzili Ether, który jest kryptowalutą. Głównym celem wprowadzenia własnej kryptowaluty w Ethereum nie jest transfer pieniędzy, lecz monetyzacja działań podjętych w zdecentralizowanych aplikacjach. Każda transakcja wykonana przez EVM zużywa gaz, czyli miarę zużycia pamięci mocy obliczeniowej, jaką maszyna musiała podjąć, aby wykonać zapisane w kontrakcie instrukcje. Dla prostych operacji opłaty za gaz są bardzo małe, jednak w ostatnim czasie ceny te znacznie wzrosły za sprawą dużego zainteresowania Ethereum i idącym za tym przeciążeniem sieci.

W artykule przedstawiona zostanie przykładowa aplikacja działająca z wykorzystaniem blockchainu Ethereum. Aplikacja ma na celu symulację systemu zarządzania łańcuchem dostaw z wykorzystaniem łańcucha bloków do zapisu i odczytu danych o wprowadzonych produktach.

2. Łańcuch bloków a łańcuch dostaw

Zaraz po finansach, branże spożywcza i transportowa stały się naturalnymi wyborami dla przedsiębiorstw zainteresowanych technologią bloków. Branże te wykorzystują wiele walorów blockchainu - przede wszystkim transparentność i bezpieczeństwo danych, ale także możliwości inteligentnych kontraktów..

Transparentność w branży spożywczej to coraz częściej pojawiający się trend. Klient chce mieć gwarancję, że informacje o kupowanym przez niego produkcie są prawdziwe. W klasycznych systemach sprzedaży musi on zaufać sprzedawcy w istotnych

kwestiach jak np. kraj pochodzenia danego produktu. Transparentność w takich sytuacjach może zapewnić blockchain. Jako że blockchain to rejestr publiczny, konsument może mieć wgląd we wszystkie dane wprowadzone do bloków. Żeby dane te nie pozostawały bez pokrycia, blockchain umożliwia także wprowadzenie niezbędnych plików z dokumentami potwierdzającymi autentyczność wprowadzanych informacji. Bezpieczeństwo i niezmienność tych danych jest zapewnione przez złożoną strukturę kryptograficzną blockchainu.

Pochodzenie produktu jest istotne nie tylko z perspektywy świadomości klientów o kupowanym towarze, ale także ich bezpieczeństwa i zdrowia. W klasycznych łańcuchach dostaw dane na poziomie poszczególnych ogniw dystrybucji są bardzo zróżnicowane. Każda firma prowadzi swoją własną bazę danych, a niektóre z nich wciąż są formie papierowej. W ten sposób dane o konkretnym produkcie na przestrzeni jednego łańcucha dostaw są niejednolite, co wprowadza chaos w dokumentacji. Chaos ten staje się szczególnie niebezpieczny, gdy podczas produkcji dojdzie np. do skażenia całej partii towaru. Gdy skażenie zostanie zauważone w dalszych etapach dostawy należy jak najszybciej zlokalizować miejsce, z którego wyszła wadliwa partia. Przebrnięcie przez wszystkie bazy danych każdej z firm, która przewoziła produkt, będzie trwać nawet kilka dni, co może przynieść tragiczne w skutkach konsekwencje. Przy ujednoliconym systemie zarzadzania łańcuchem dostaw w blockchainie informacje te można zdobyć w ciągu kilku sekund.

Działanie łańcuchów dostaw może również zostać usprawnione przez zastosowanie inteligentnych kontraktów. Skrypty zamieszczone w kontrakcie na bieżąco odczytują dane zapisane w blockchainie. W ten sposób możliwe jest np. kontrolowanie daty spożycia produktów. Gdy producent wprowadzi taką informację do systemu, skrypt zawarty w kontrakcie może sprawdzać ją automatycznie, a w przypadku przeterminowania produktu – wysłać komunikat do odpowiedniego podmiotu. Innym sposobem wykorzystania kontraktów jest kontrola warunków przechowywania produktów. Gdy przewożony towar wymaga specyficznych warunków, np. odpowiednią temperaturę przechowywania lub poziom wilgotności powietrza, smart kontrakt może odczytywać te wartości z odpowiednich czujników i porównywać je z wytycznymi producenta.

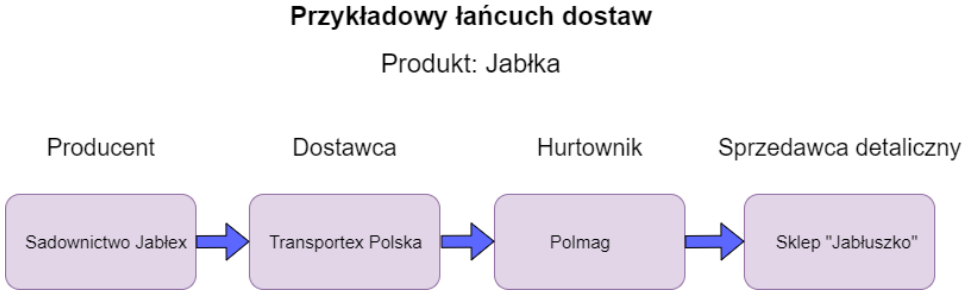
Zaprezentowana w artykule aplikacja to tylko uproszczony sposób wykorzystania blockchainu do zapisu i odczytu danych o wprowadzonym do systemu produkcie. Na rys. 1 przedstawiono schemat łańcucha dostaw zaproponowanego w aplikacji oraz przykładowy produkt wprowadzony w systemie.

3. Wykorzystane narzędzia

Do stworzenia strony internetowej wykorzystano Django, czyli otwarty framework napisany w języku Python. Django opiera swoje działanie na modelach ściśle powiązanych z bazą danych, widokach hermetyzujących logikę aplikacji oraz szablonach, które odpowiadają za renderowanie widoku strony. Zorientowana w ten sposób architektura nazywana jest MVT (Model – View – Template) i została wykorzystana w tym projekcie. Zastosowanie modeli pozwoliło na łatwą integrację aplikacji ze smart kontraktami dla każdego z ogniw łańcucha dostaw. Django posiada również rozbudowany system logowania i rejestracji z zaimplementowaną weryfikacją uzupełnianych pól oraz przejrzysty panel administratora, co znacznie ułatwiło pracę podczas częstego dodawania nowych użytkowników będących ogniwami łańcucha.

Smart kontrakty zostały napisane w języku Solidity, który jest jednym z najpopularniejszych wyborów programistów. Kod języka interpretowany jest przez maszynę EVM, która wykonuje zawarte w kontrakcie instrukcje. Kontrakty w Solidity są podobne do klas w językach zorientowanych obiektowo. Każdy uczestnik łańcucha dostaw w aplikacji posiada odpowiadający jego roli kontrakt z dedykowanymi funkcjonalnościami.

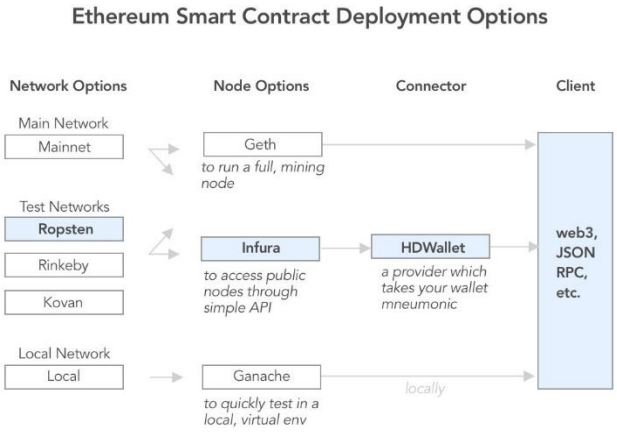
Do wdrożenia smart kontraktów do sieci Ethereum wykorzystano narzędzie Truffle. Truffle to rozbudowane środowisko ułatwiające programistom pracę z EVM w sieci głównej oraz sieciach testowych. Korzystanie z sieci testowych jest darmowe, natomiast przy łączeniu aplikacji z siecią główną, czyli Mainnetem, należy brać pod uwagę rzeczywiste opłaty za wykonywane operacje.



Rysunek 1. Schemat łańcucha dostaw dla produktu testowego

4. Połączenie aplikacji z blockchainem

Schemat połączenia z siecią Ethereum pokazano na rys. 2. Ethereum umożliwia dostęp do kilku sieci testowych (Rinkeby, Ropstein, Kovan, Goerli) oraz do głównej sieci Mainnet. Możliwe jest także uruchomienie własnego blockchainu na serwerze lokalnym za pośrednictwem aplikacji Ganache. Konta tworzone w sieciach testowych można doładowywać za darmo wirtualnym Etherem, który nie ma rzeczywistej wartości.



Rysunek 2. Schemat dostępnych możliwości połączenia aplikacji z blockchainem Ethereum. [3]

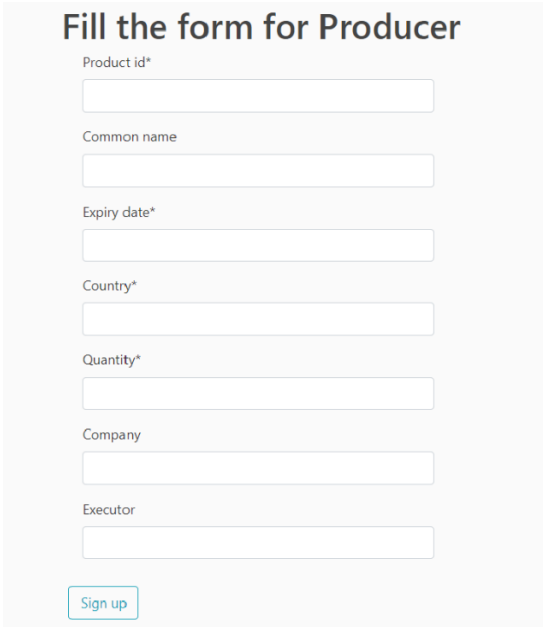
W aplikacji wykorzystano sieć Rinkeby, która wyróżnia się łatwym dostępem do wirtualnego Etheru, co było ważne z uwagi na duże zużycie gazu przy przeprowadzaniu testów. Do połączenia z siecią służy węzeł sieciowy (ang. node). W sieci Ethereum węzeł można zainstalować samodzielnie za pośrednictwem aplikacji Geth lub połączyć się z publicznym węzłem, np. za pośrednictwem Infury. Infura zapewnia użytkownikowi system monitorowania swojej aktywności w blockchainie oraz prywatny klucz dostępu do API. Do połączenia się z blockchainem wymagany jest także portfel. Portfel zawiera jedno lub więcej kont użytkownika, które posiadają swój klucz publiczny (adres) służący do komunikacji w sieci oraz klucz prywatny, który jest niezbędny do stworzenia cyfrowego podpisu. Ulepszoną wersją zwykłego portfela są tzw. HD-Wallety (Hierarchical Deterministic Wallet). HD–Wallet tworzy klucze prywatne i publiczne na podstawie sekwencji 12 do 24 losowych słów, nazwanej „mneumonic". Słowa te nie mogą być przypadkowe, tylko wybierane są z listy 2048 określonych słów, która jest zaakceptowana przez wszystkich użytkowników sieci. Ostatnią warstwę łączności z blockchainem stanowi biblioteka Web3, która posiada rozbudowane API umożliwiające m.in. wywoływanie funkcji smart kontraktów, wykonywanie transakcji czy odczytywanie informacji o wygenerowanych w blokach w łańcuchu. W aplikacji korzystano z wersji biblioteki dedykowanej językowi Python, czyli Web3py.

5. Widoki i funkcjonalności aplikacji

Strona główna aplikacji została zaprezentowana na rys. 3. Z pełnych możliwości strony mogą korzystać tylko zarejestrowani użytkownicy, czyli osoby pełniące pewną rolę w łańcuchu dostaw. Mogą oni dodawać i odczytywać dane z blokchainu. Formularz rejestracji przedstawiony jest na rys. 5. Aplikacji mogą używać także niezarejestrowani użytkownicy, czyli potencjalni klienci, którzy będą chcieli odczytać z blokchainu dane o zakupionym produkcie.

Rysunek 3. Strona domowa.

Dodawanie danych do blockchain za pośrednictwem API z biblioteki Web3py. Informacje o produkcie wpisywane są przez uprawnionych użytkowników za pośrednictwem formularzy dedykowanych poszczególnym rolom w łańcuchu dostaw. Na rys. 4 przedstawiono przykładowy formularz dla producenta.



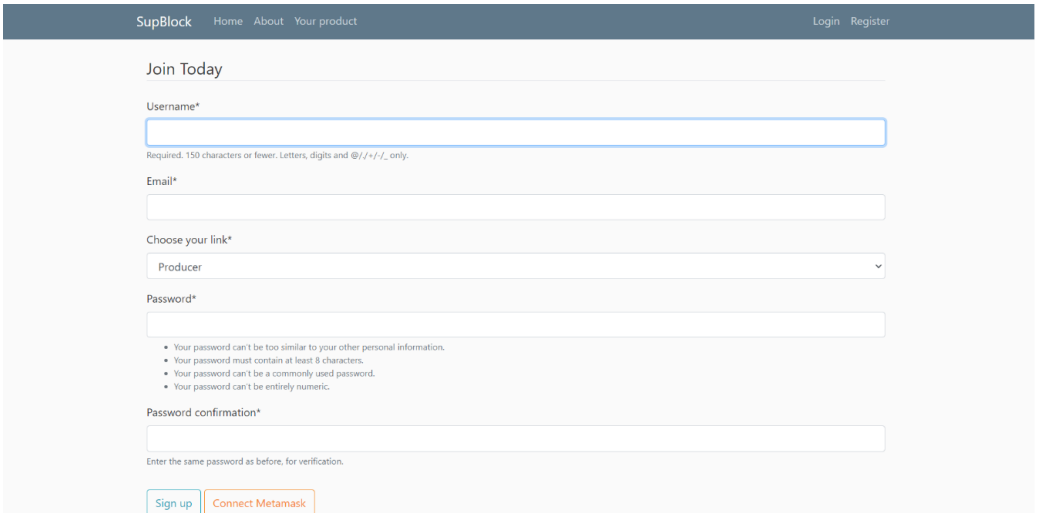
Rysunek 4. Przykładowy formularz dla producenta.

Po uzupełnieniu formularza, dane przekazywane są do smart kontraktów, które wcześniej zostały wdrożone do sieci. Kontrakty, podobnie jak konta użytkownika, posiadają swoje klucze publiczne (adresy), więc interakcja z kontraktem polega na wysłaniu specjalnej transakcji na konkretny adres. Zapis do blockchainu kosztuje pewną niewielką ilość Etheru, natomiast odczytywanie danych jest darmowe. Pozwala to niezalogowanym użytkownikom na darmowe korzystanie z aplikacji.

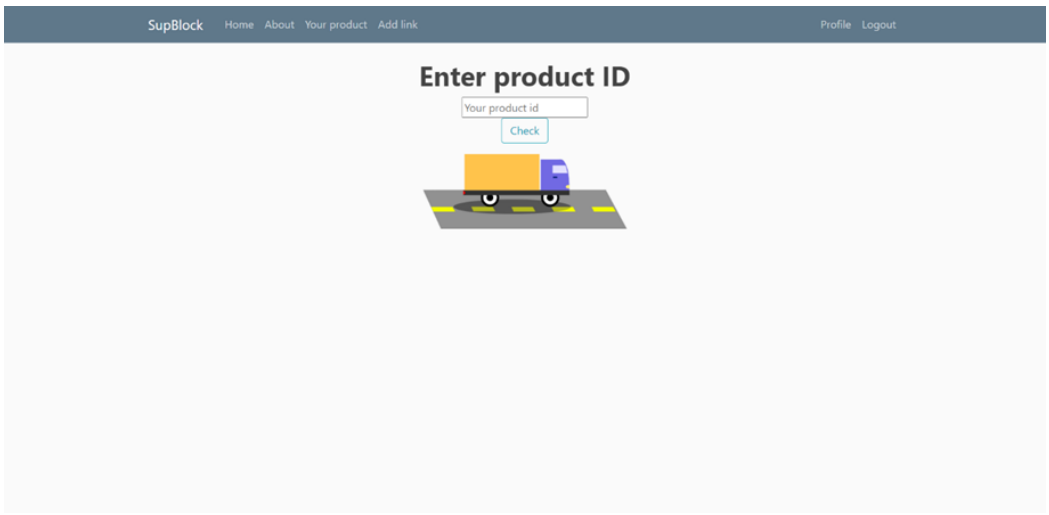
Sprawdzenie informacji o produkcie znajduje się w zakładce „Your product". W wyświetlanym polu (rys. 6) należy wpisać numer identyfikacyjny produktu. Zakłada się, że numer ten będzie wydrukowany na opakowaniu produktu zakupionego w sklepie. Jeśli produkt zostanie znaleziony w blockchainie aplikacja wyświetli pełną trasę, jaką pokonał towar podczas dostawy, od producenta do sprzedawcy detalicznego (rys. 7).

Na rys. 8 przedstawiono szczegółowy widok dla produktu testowego. Jako że każda akcja w blockchainie jest transparentna, wszystkie dokonane transakcje można zweryfikować na stronie www.ethscan.io, która wyświetla dane wszystkich operacji dla danego konta bądź kontraktu, tj.:

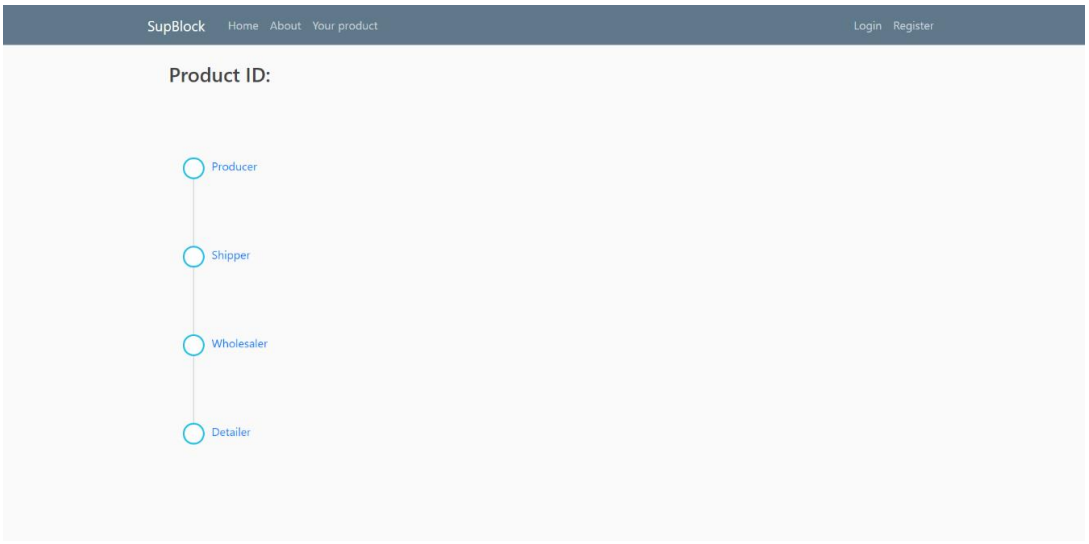
- Numer hash transakcji
- Numer bloku, do którego transakcja została zapisana
- Czas, który minął od zapisania transakcji do teraz
- Adres nadawcy
- Adres odbiorcy
- Ilość wysłanego Etheru
- Podatek za transakcję



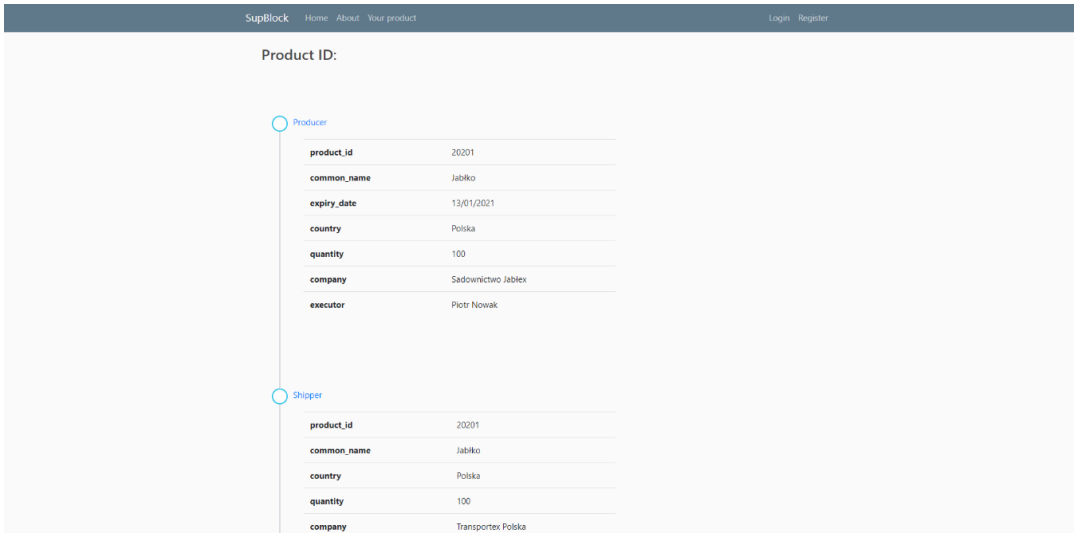
Rysunek 5. Strona rejestracji.



Rysunek 6. Widok wpisywania numeru ID produktu.



Rysunek 7. Lista ogniw łańcucha dostaw wyświetlona po wpisaniu ID produktu.



Rysunek 8. Szczegółowy widok produktu.

6. Podsumowanie

Przedstawiona w artykule aplikacja jest tylko uproszczoną symulacją systemu zarządzania łańcuchem dostaw. Mimo to dobrze prezentuje podstawowe mechanizmy pozwalające na wykorzystanie blockchainu w branży spożywczej. Wdrożenie aplikacji do rzeczywistego systemu wymaga dużo więcej pracy, przede wszystkim całkowitej reorganizacji struktury informatycznej firmy zainteresowanej takim rozwiązaniem. Dodatkowo, aplikacja powinna zostać rozszerzona o możliwość dodawania plików z dokumentami świadczącymi o prawdziwości wprowadzonych danych.

Literatura

1. Ethereum Foundation. *Ethereum Whitepaper*. https://ethereum.org/en/whitepaper/ [data dostępu: 2021-03-12]
2. Sitko Maciej, Praca inżynierska *Smart contracts z wykorzystaniem technologii blockchain*
3. Zhu Nicole. *5 minute guide to deploying smart contracts with Truffle and Ropsten* https://medium.com/coinmonks/5-minute-guide-to-deploying-smart-contracts-with-truffle-and-ropsten-b3e30d5ee1e [data dostępu: 2021-03-12]