Imię i nazwisko autora rozprawy: **mgr inż. Piotr Dryja**
Dyscyplina naukowa: **Informatyka Techniczna i Telekomunikacja**

ROZPRAWA DOKTORSKA

Tytuł rozprawy w języku polskim: **Optymalizacja zasobów chmury obliczeniowej z wykorzystaniem inteligentnych agentów w zdalnym nauczaniu**

Tytuł rozprawy w języku angielskim: **Optimization of cloud computing resources using intelligent agents in remote teaching**

Promotor *podpis*
dr hab. inż. Jerzy Balicki, prof. uczelni

Gdańsk, rok 2022

STRESZCZENIE

Rozprawa dotyczy optymalizacji zasobów chmury obliczeniowej, w której zastosowano inteligentne agenty w zdalnym nauczaniu. Zagadnienie jest istotne w edukacji, gdzie wykorzystuje się nowoczesne technologie, takie jak Internet Rzeczy, rozszerzoną i wirtualną rzeczywistość oraz deep learning w środowisku chmury obliczeniowej. Zagadnienie jest istotne również w sytuacji, gdy pandemia wymusza stosowanie zdalnego nauczania na dużą skalę lub rosnące koszty efekty ograniczają naukę stacjonarną.

W dysertacji opracowano model alokacji inteligentnych agentów pedagogicznych, modułów systemu zdalnego nauczania oraz solwerów do serwerów w chmurze. Sformułowano zagadnienie wyznaczania reprezentacji rozwiązań Pareto-optymalnych. Opracowano wielokryterialny algorytm ewolucji różnicowej, za pomocą którego minimalizuje się cztery kryteria takie, jak: obciążenie CPU newralgicznego hosta, obciążenie komunikacyjne krytycznego węzła, koszt serwerów oraz łączny pobór mocy elektrycznej.

W celu eksperymentalnej weryfikacji poprawności modelu zaprojektowano demonstrator chmury edukacyjnej w oparciu o wybrane zasoby Politechniki Gdańskiej oraz Politechniki Warszawskiej. Przeprowadzono wielowariantowe symulacje migracji inteligentnych agentów w środowisku Moodle na platformie OpenStack. Eksperymenty numeryczne potwierdziły, że wykorzystanie wielokryterialnej metaheurystyki opartej na ewolucji różnicowej umożliwia wyznaczenie wyższej jakości rozwiązań niezdominowanych niż rozwiązania uzyskane za pomocą innych metaheurystyk, takich jak: algorytm ewolucyjny, programowanie genetyczne, algorytm wyszukiwania harmonii oraz algorytm roju cząstek.

ABSTRACT

The dissertation concerns the optimization of cloud computing resources with the use of intelligent agents in remote learning. The issue is important in education, where modern technologies such as the Internet of Things, augmented and virtual reality and deep learning are increasingly used in an environment of a properly configured cloud computing. The issue is also important in a situation where the pandemic necessitates the use of distance learning on a large scale.

In the dissertation, a model of the allocation of intelligent pedagogical agents to servers in the computing cloud was developed. The problem of determining the representation of Pareto-optimal solutions has been formulated. A multi-criteria differential evolution algorithm was developed, by means of which the four criteria of the educational cloud are minimized, such as: critical CPU host load, critical node communication load, server cost and electricity consumption.

To experimentally verify the model, a computing cloud was designed based on selected resources of the Gdańsk University of Technology and the Warsaw University of Technology. Multivariate simulations of the migration of intelligent pedagogical agents in the Moodle environment on the OpenStack platform were conducted. Numerical experiments have confirmed that the use of multi-criteria metaheuristics based on differential evolution enables the determination of higher-quality solutions that are not dominated than solutions obtained using other metaheuristics, such as: evolutionary algorithm, genetic programming, harmony search algorithm and particle swarm algorithm.

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH AKRONIMÓW I OZNACZEŃ 4
WPROWADZENIE 6
1. GENEZA SMART EDUKACJI 9
1.1 Rys historyczny 9
1.2 Wybrane zastosowania zdalnego nauczania 13
1.3 Inteligentne systemy agentowe 15
1.4 Agenty w e-learningu 18
1.5 Wnioski i uwagi 20
2. WYBRANE METODY SZTUCZNEJ INTELIGENCJI W E-KSZTAŁCENIU 21
2.1 Programowanie genetyczne 21
2.2 Sztuczne sieci neuronowe 22
2.3 Maszyna wektorów wspierających 24
2.4 Ewolucja różnicowa 25
2.5 Wnioski i uwagi 32
3. CHMURY OBLICZENIOWE W ZDALNYM NAUCZANIU 34
3.1 Chmura obliczeniowa 34
3.2 Koncepcja agentowego modelu zdalnego nauczania 39
3.3 Metody optymalizacji zasobów chmur obliczeniowych 43
3.4 Wnioski i uwagi 47
4. METODA OPTYMALIZACJI ZASOBÓW CHMURY EDUKACYJNEJ 49
4.1 Podstawowe kryteria oceny oraz ograniczenia 49
4.2 Sformułowanie problemu optymalizacji wielokryterialnej 56
4.3 Metoda ewolucji różnicowej do migracji maszyn wirtualnych 59
4.4 Wnioski i uwagi 70
5. EKSPERYMENTY NUMERYCZNE 72
5.1 Rekomendacje wartości metaparametrów ewolucji różnicowej 72
5.2 Porównanie algorytmu DEMCA z innymi metaheurystykami 75
5.3 Modernizacja infrastruktury chmury obliczeniowej 77
5.4 Projekt rozbudowy demonstratora chmury edukacyjnej 80
5.5 Wnioski i uwagi 93
PODSUMOWANIE 95
BIBLIOGRAFIA 97
SPIS RYSUNKÓW 106
SPIS TABEL 108
DODATEK. Aplikacja algorytmu ewolucji różnicowej DEMCA 109

WYKAZ WAŻNIEJSZYCH AKRONIMÓW I OZNACZEŃ

Cp – próg akceptacji współrzędnych potomków dla krzyżowania w algorytmie ewolucji różnicowej;

dm_n – (ang. *disc memory*) wielkość pamięci dyskowej DM (HDD lub półprzewodnikowych SDD) [TB] w komputerze *n*-tego rodzaju, $n = \overline{1, N}$;

F – funkcja oceny rozwiązań w problemie optymalizacji wektorowej;

F_1 – obciążenie CPU newralgicznego hosta (cechuje się największym obciążeniem CPU w chmurze) [s];

F_2 – obciążenie transmisją danych krytycznego węzła (cechuje się największym obciążeniem w chmurze) [s];

F_3 – łączny koszt zakupu lub dzierżawy hostów w chmurze [JM], JM – jednostka monetarna;

F_4 – moc poboru energii elektrycznej przez hosty [wat];

$H_1, \ldots, H_n, \ldots, H_N$ – rodzaje hostów;

H_n – host *n*-tego rodzaju, $n = \overline{1, N}$;

$h = [h_1, \ldots, h_k, \ldots, h_K]$ – wektor reprezentujący zapotrzebowanie maszyn wirtualnych na wielkość pamięci dyskowej DM w hostach, $k = \overline{1, K}$;

$N_1, \ldots, N_m, \ldots, N_M$ – węzły chmury obliczeniowej;

$r = [r_1, \ldots, r_k, \ldots, r_K]$ – wektor reprezentujący zapotrzebowanie maszyn wirtualnych z inteligentnymi agentami pedagogicznymi, modułami zdalnego nauczania i solwerami na wielkość pamięci RAM w hostach;

ram_n – wielkość pamięci RAM [GB] w komputerze *n*-tego rodzaju;

T_j – zadany wektor czasów pracy maszyn wirtualnych z inteligentnymi agentami i modułami systemu Moodle na komputerze *n*-tego rodzaju dla reprezentatywnych zestawów danych, $n = \overline{1, N}$;

$VM_1, \ldots, VM_k, \ldots, VM_K$ – maszyny wirtualne z agentami pedagogicznymi, modułami systemu Moodle lub solwerami;

VM_k – *k*-ta maszyna wirtualna, $k = \overline{1, K}$;

$\boldsymbol{X}$ – zbiór rozwiązań dopuszczalnych w problemie optymalizacji;

$x = (X^H, X^{VM})$ – rozwiązanie reprezentowane za pomocą pary macierzy dwuwymiarowych z elementami o wartościach 0 lub 1;

x^{Pareto} – rozwiązanie optymalne w sensie Pareto;

x^p – rozwiązanie kompromisowe dla zadanego parametru *p*;

$\bar{x} = (\bar{X}^H, \bar{X}^{VM})$ – rozwiązanie reprezentowane za pomocą pary wektorów, których współrzędne są liczbami naturalnymi;

$X^{VM} = [x_{km}^{VM}]_{K \times M}$ – binarna macierz rozmieszczenia maszyn wirtualnych w chmurze obliczeniowej; x_{vm}^{VM}=1, jeśli *k*-ta maszyna wirtualna jest przydzielona do hosta w węźle nr *m*;

$\bar{X}_k^{VM}$ – zmienna decyzyjna reprezentująca numer węzła, w którym do hosta przydzielona jest *k*-ta maszyna wirtualna;

$X^H = [x_{mn}^H]_{M \times N}$ – binarna macierz reprezentująca przydział rodzajów hostów do węzłów w chmurze obliczeniowej; x_{mn}^H = 1, jeśli *n*-ty rodzaj hosta przydzielono do węzła nr *m*;

$\bar{X}_m^H$ – zmienna decyzyjna reprezentująca indeks rodzaju hosta, który jest eksploatowany w węźle nr *m*;

$X_N^{\leqq}$ – zbiór rozwiązań niezdominowanych w zbiorze rozwiązań dopuszczalnych $\boldsymbol{X}$ dla relacji porządku $\succ_{\leq}$ w przestrzeni wielokryterialnej $\boldsymbol{Y}$;

$\boldsymbol{Y}$ – przestrzeń wielokryterialna (obraz zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$ dla funkcji oceny F); $\boldsymbol{Y}=F(\boldsymbol{X})$;

β_n – maksymalna moc poboru energii przez komputer n-tego rodzaju [wat];

μ_{min}^{DM} – wielkość wolnej pamięci dyskowej DM krytycznego hosta, który ma najmniejszą wielkość wolnej pamięci dyskowej ze wszystkich hostów w chmurze [TB];

μ_{min}^{RAM} – wielkość wolnej pamięci RAM newralgicznego hosta, który ma najmniejszą wielkość wolnej pamięci ze wszystkich hostów w chmurze [GB];

q – współczynnik skalujący różnicę wektorów w mutacji różnicowej;

$\succ_{\leq}$ – relacja dominowania w przestrzeni $\boldsymbol{R}^4$ modelująca minimalizację czterech kryteriów skalarnych w problemie optymalizacji wielokryterialnej.

WPROWADZENIE

Proces kształcenia na odległość rozpoczął się około 1700 roku korespondencyjną nauką języka angielskiego, a w 1883 roku rozpoczął kształcenie studentów Uniwersytet Nauki Korespondencyjnej w Nowym Jorku. Wraz z rozwojem nowoczesnych technologii, nauczanie na odległość cechowało się wykorzystaniem radia, telewizji, systemów satelitarnych i Internetu. Obecnie dyskutowane są koncepcje odnoszące się do stosowania chmur obliczeniowych wraz z usługami opartymi na sztucznej inteligencji. Ponadto możliwości, jakie stwarzają Internet Rzeczy, modele maszynowego uczenia głębokiego, czy rozszerzona i wirtualna rzeczywistość, stanowią fundament nowego podejścia do edukacji, którą nazywa się inteligentną edukacją (ang. *smart education*) [6].

Dynamiczny rozwój smart edukacji związany jest również z rozpowszechnieniem się pandemii koronawirusa SARS-CoV-2 (ang. *Severe Acute Respiratory Syndrome CoronaVirus 2*) w latach 2020-2021. Kryzys wymusił stosowanie zdalnej edukacji na niespotykaną dotąd skalę: nie tylko w uczelniach, ale również w szkołach średnich, a nawet w szkołach podstawowych. W rezultacie uczniowie pierwszych klas wielu szkół podstawowych uczestniczyli w tego rodzaju zajęciach, korzystając z platformy MS Teams nie tylko na laptopach, ale również na tabletach i smartfonach.

Dlatego niezwykle ważne jest efektywne wykorzystanie nowoczesnych platform zdalnego nauczania przy ograniczonych zasobach informatycznych. Na tym tle, warto zauważyć, że sztuczna inteligencja wspomaga znacząco smart edukację zarówno bezpośrednio za pomocą stosowania modeli uczenia maszynowego, jak i pośrednio za pomocą stosowania metaheurystyk do optymalizacji infrastruktury systemu nauczania.

Efektywne nadzorowanie edukacyjnej chmury obliczeniowej, w której zarządzane są setki węzłów, hostów i maszyn wirtualnych jest dla administratora zadaniem prawie niewykonalnym. Jednakże odpowiednio skonfigurowane środowisko chmury pozwala na osiągnięcie wysokiej wydajności oraz niskich kosztów eksploatacji hostów. W tym celu pomocne mogą być metaheurystyki, np. ewolucja różnicowa. Implementacja wyznaczonych rozwiązań może wprowadzić oszczędności, zmniejszone zużycie energii elektrycznej, zredukowane obciążenie newralgicznych procesorów oraz mniejszą intensywność transmisji danych w sieci komputerowej.

W ramach modelu smart edukacji przyjmuje się, że proces nauczania na uczelni może być wspierany za pomocą inteligentnych agentów pedagogicznych. Modele uczenia głębokiego mogą prognozować końcowe oceny z przedmiotów, stopień zainteresowania

studentów treściami nauczania oraz sprawność ukończenia studiów. Wykładowcy mogą zatem reagować wcześniej na sytuacje, które nie są wskazane w procesie nauczania.

Ze względu na fakt, że maszyna wirtualna z niektórymi agentami może się przemieszczać między hostami chmury, należy rozwiązać dylemat doboru adekwatnych hostów dla maszyn wirtualnych. W celu weryfikacji przyjętych założeń, zbudowano demonstrator chmury edukacyjnej w oparciu o wybrane zasoby Politechniki Gdańskiej oraz Politechniki Warszawskiej, a także komputery autora. Chmurę nazwano GUT-WUT (akronim od Gdańsk University of Technology - Warsaw University of Technology), a zdobyte doświadczenia mogą umożliwić zbudowanie w przyszłości Polskiej Chmury Edukacyjnej uczelni wyższych.

Opracowano zestaw symulowanych agentów pedagogicznych działających w środowisku Moodle, które to agenty funkcjonują na maszynach wirtualnych w ramach platformy OpenStack. Eksperymenty numeryczne potwierdziły, że wykorzystanie metaheurystyki opartej na ewolucji różnicowej umożliwia wyznaczanie wyższej jakości rozwiązań niezdominowanych niż warianty otrzymane za pomocą innych metaheurystyk. Warto zaznaczyć, że rozwiązanie reprezentuje rozmieszczenie maszyn wirtualnych na hostach, uwzględniając wydajność procesora, wielkość pamięci RAM, pojemność dysku, pobór energii elektrycznej, czy przepustowość karty sieciowej.

Celem badań ze względów na przyjętą metodologię jest model i algorytm ewolucji różnicowej do optymalizacji wektorowej zasobów chmury obliczeniowej, w której stosuje się inteligentne agenty do zdalnego wspomagania nauczania.

Problem badawczy polega na opracowaniu modelu, sformułowaniu adekwatnego problemu polioptymalizacji oraz skonstruowaniu efektywnego algorytmu ewolucji różnicowej do wyznaczenia miejsc migracji maszyn wirtualnych do hostów, które cechują się odpowiednimi wielkościami zasobów komputerowych. Na podstawie tak określonego celu badań oraz sformułowanego problemu badawczego, postawiono następującą hipotezę naukową.

Optymalizacja lokalizacji hostów migracji maszyn wirtualnych z inteligentnymi agentami pedagogicznymi, modułami systemu zdalnego nauczania Moodle i solwerami, za pomocą wielokryterialnego algorytmu ewolucji różnicowej, może poprawić jakość działania chmury poprzez redukcję obciążenia CPU newralgicznego hosta i obciążenia transmisją danych krytycznego węzła, a także poprzez obniżenie kosztu hostów i łącznej mocy poboru energii elektrycznej przez serwery.

Zarysowana koncepcja metodologiczna wpłynęła na przyjęcie następującego zakresu pracy. W rozdziale pierwszym scharakteryzowano zdalne nauczanie w kontekście smart edukacji. Omówiono wybrane zastosowania edukacji online oraz sklasyfikowano systemy zdalnego nauczania. Ponadto scharakteryzowano kluczowe trendy w e-learningu. Nawiązano do inteligentnych agentów w edukacji. Na końcu rozdziału oraz po każdym kolejnym zamieszczono wnioski i uwagi.

Zastosowanie wybranych metod sztucznej inteligencji w zdalnym nauczaniu przedstawiono w rozdziale drugim. W szczególności, opisano algorytm ewolucji różnicowej. Kluczową procedurą algorytmu ewolucji różnicowej jest mutacja, która oparta jest na modyfikacji różnicy wybranych wektorów. Mutacja i krzyżowanie pozwalają osiągnąć odpowiednią różnorodność osobników w populacji, a funkcja sprawności zapewnia wysoką jakość wyznaczonych rozwiązań.

W rozdziale trzecim scharakteryzowano platformę OpenStack dla chmur obliczeniowych. Przedstawiono koncepcję demonstratora chmury edukacyjnej w oparciu o platformę Moodle, inteligentne agenty pedagogiczne oraz solwery opierające się na wielokryterialnym algorytmie ewolucji różnicowej. Ponadto omówiono modele i metody dotyczących migracji maszyn wirtualnych w chmurze obliczeniowej.

Czwarty rozdział zawiera sformułowanie problemu optymalizacji wielokryterialnej, w którym minimalizowane są cztery skalarne kryteria oceny rozwiązań: obciążenie procesorów newralgicznego hosta, obciążenie transmisją danych newralgicznego węzła, koszt serwerów oraz łączna moc poboru energii elektrycznej. Wprowadzono również ograniczenia na migrację maszyn wirtualnych oraz na dobór hostów. Scharakteryzowano wielokryterialny algorytm ewolucji różnicowej DEMCA (ang. *Differential Evolution Method for Cloud Agents*) do wyznaczania rozwiązań Pareto-optymalnych.

W rozdziale piątym opisano implementację demonstratora chmury edukacyjnej GUT-WUT, której hosty użytkowano w Warszawie, Gdańsku, Gdyni oraz Kościerzynie w latach 2020-2021. Omówiono wyniki eksperymentów numerycznych obejmujące dobór metaparametrów algorytmu ewolucji różnicowej, a także porównanie z innymi metaheurystykami. Nawiązano do możliwych rozszerzeń demonstratora chmury edukacyjnej uczelni wyższych.

Na zakończenie zamieszczono podsumowanie, bibliografię, wykaz rysunków, wykaz tabel, a także dodatek, w którym scharakteryzowano aplikację DEMCA.

1. GENEZA SMART EDUKACJI

W 2016 roku pojawiła się w literaturze przedmiotu koncepcja inteligentnej edukacji (ang. *smart education*). W tym kontekście Zhu, You i Riezbos omówili problemy badawcze związane z rozwojem smart edukacji [187]. W szczególności Jagtap, Bodkhe, Gaikwad i Kalyana zaproponowali zastosowanie w edukacji modeli uczenia maszynowego w oparciu o media społecznościowe [69]. Dopiero jednak przeglądowa praca Martína, Alario-Hoyosa i Kloosa z 2019 roku ugruntowała pozycję smart edukacji [98]. Następnie Alam i Saiyed scharakteryzowali wykorzystanie chmury obliczeniowej w edukacji [6].

1.1 Rys historyczny

Intensywnemu napływowi imigrantów do Stanów Zjednoczonych towarzyszył deficyt nauczycieli języka angielskiego, co wymusiło rozwój nauczania korespondencyjnego. Pod koniec XIX wieku na Uniwersytecie Nauki Korespondencyjnej w Nowym Jorku rozpoczęto kształcenie na wybranych kierunkach, a doskonalącym język angielski umożliwiono naukę w Międzynarodowej Szkole Korespondencyjnej [179].

Rozwój technologii komunikacyjnych umożliwił wprowadzenie synchronicznego modelu edukacji na odległość, w którym tysiące studentów oddalonych o setki kilometrów od nauczycieli mogą uczestniczyć w zajęciach o ustalonej porze. W 1925 roku wyemitowano pionierskie edukacyjne audycje radiowe prowadzone przez wykładowców z Uniwersytetu Iowa. Wykłady radiowe stały się popularne również poza Ameryką Północną, na słabo zaludnionych i rozległych obszarach Australii. Również w Europie około pięćdziesiąt lat temu edukacja radiowa cieszyła się powodzeniem, szczególnie na terenach wiejskich [84].

Wykłady telewizyjne rozpowszechniono po zakończeniu II Wojny Światowej. Telewizyjny model edukacji na odległość był na tyle interesujący, że rozpoczęto emisję wielogodzinnych kursów. Warto podkreślić, że odbiorniki TV były na wyposażeniu niemal wszystkich szkół w Polsce na początku XXI wieku [179].

Modele edukacji w oparciu o telewizję czy radio nie cechują się bezpośrednią interaktywnością podczas zajęć, co nie sprzyja sprawnemu przyswajaniu wiedzy przez studentów. Z tego powodu, w 1962 roku, rozpoczęto stosowanie wideokonferencji, wykorzystując satelitę telekomunikacyjnego Telstar1. Wideokonferencje satelitarne stosowano w rejonach cechujących się surowym klimatem. Na Uniwersytecie Anchorage wprowadzono je do wspierania zajęć dydaktycznych, co było znaczącym usprawnieniem

ze względu na problemy z dojazdami na zajęcia podczas mrozów panujących na Alasce. Warto zaznaczyć, że wizualizacja treści nauczania wraz z interaktywnym uczeniem się zwiększa zaangażowanie studentów o około 40% [84].

W latach sześćdziesiątych ub. wieku zaczęto stosować komputery w edukacji. Pionierski system edukacyjny *Computer Managed Instruction* (CMI) monitorował postępy studentów, co umożliwiło opracowanie indywidualnych ścieżek kształcenia. Zauważono również istotną zaletę komputerowego wspierania indywidualnych stylów uczenia się [51]. Badania potwierdzają, że ok. 90% uczących się w trybie indywidualnym osiąga wyniki należące do zbioru 20-25% najlepszych ocen [84].

Zaprojektowano rozproszony system informatyczny PLATO, który zapewnił na Uniwersytecie Illinois współdzielenie cyfrowych zasobów edukacyjnych. W szczególności język programowania Tutor umożliwił opracowanie interaktywnych kursów edukacyjnych, a inne aplikacje zapewniły przesyłanie wiadomości oraz prowadzenie wideokonferencji. Następnie wprowadzono bardziej zaawansowaną generację systemów zdalnego nauczania, takich jak Blackboard i WebCT [51].

Przełom nastąpił wraz z rozpowszechnieniem Internetu na początku lat dziewięćdziesiątych XX wieku. Zauważono, że edukacja webowa umożliwia powszechny dostęp do multimedialnych materiałów [86]. Doceniono zalety, takie jak likwidowanie barier w dostępie do edukacji oraz uwzględnianie indywidualnych możliwości i oczekiwań studentów. Istotnym jest fakt, że internetowy model nauczania pozwala ograniczyć wydatki na kształcenie [2].

W 2019 roku wiele uczelni oferowało nauczanie na odległość jako uzupełnienie kształcenia. Z tego względu e-learning intensywnie rozwija się, tym bardziej, że wspierany jest przez nowoczesne technologie, takie jak: szerokopasmowy Internet oraz bezprzewodowa komunikacja, w tym standard telefonii mobilnej 5G [148]. Nowa sytuacja wystąpiła na wiosnę 2020 roku, kiedy pandemia Covid-19 wymusiła powszechne zastosowanie zdalnego nauczania. Kolejne fale pandemii spowodowały konieczność eksploatacji systemów zdalnego kształcenia. Po ustąpieniu pandemii zarysowała się tendencja prowadzenia zajęć w trybie hybrydowym, w którym niektórzy studenci uczestniczą w zajęciach zdalnie. Natomiast wysokie koszty energii elektrycznej w 2022 roku spowodowały, że niektóre uczelnie mogą szukać oszczędności, przechodząc w miesiącach zimowych na tryb zdalny w roku akademickim 2022/23.

Ponadto upowszechnienie Internetu Rzeczy (ang. *Internet of Things,* IoT) wzbogaciło sposób wykorzystania e-learningu. IoT umożliwia łączenie obiektów fizycznych (rzeczy)

z rzeczywistego świata za pośrednictwem Internetu, tworząc wirtualny model. Dotyczy to również sensorów, które pozwalają monitorować proces nauczania. Efektywne łączenie urządzeń ma wiele zastosowań w prawie każdej dziedzinie, takiej jak: opieka zdrowotna, biznes, transport, energetyka czy zarządzanie [102].

W połowie 2022 roku działało na świecie kilkadziesiąt miliardów urządzeń IoT, 7,3 mld smartfonów i 2 mld komputerów osobistych (https://www.statista.com/). Z tego powodu oczekiwanie dostępu do wiedzy online przez prawie 4 mld internautów spośród 8 mld ludzi wymusza opracowanie nowych modeli edukacji. Niewątpliwie rośnie również znaczenie zajęć asynchronicznych ze względu na postulaty indywidualnego przeglądania treści dydaktycznych w dogodnym czasie dla studenta.

Ze względu na wysoki poziom technologii, ocena postępów studenta może zostać przeprowadzona zdalnie za pomocą sprawdzenia wyników uzyskanych podczas samodzielnego rozwiązywania problemu projektowego. Nawet bardzo złożone zadania można wykonać poza uczelnią, przy czym wiąże się to z koniecznością zintensyfikowania interaktywności między wykładowcą a studentami. Ponieważ pojawiają się nowe zawody do realizacji zdalnego nauczania, można oczekiwać efektywniejszej adaptacji treści dydaktycznych do stylu uczenia się studentów.

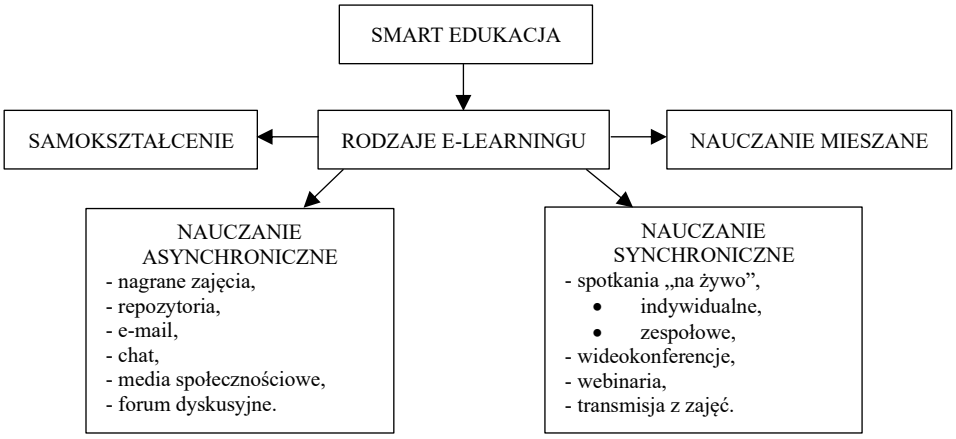
Globalizacja e-learningu powoduje, że szerokie rzesze studentów mogą korzystać z kursów online [100]. Natomiast rozwój uczenia maszynowego zaowocował powstaniem zaawansowanych modeli, które mogą wspomagać proces dydaktyczny w znacznie większym stopniu niż dotychczas. W szczególności, analizując emocje studenta na podstawie napisanego tekstu, obrazu twarzy, analizy dźwięku, w tym mowy i hałasu, a także analizując ruchy i gesty, można oszacować zainteresowanie treścią zajęć [148].

Inteligentne agenty pedagogiczne mogą pełnić również rolę asystentów studentów. W szczególności wpływają na większe zaangażowanie uczącej się osoby oraz na zwiększenie poziomu interaktywności [71]. Ponadto wirtualna i rozszerzona rzeczywistość umożliwia weryfikację wymaganych umiejętności za pomocą symulatorów 3D [76]. Sztuczna inteligencja wspiera także kontrolę podczas zajęć. Aplikacje mogą śledzić postępy studenta, wychwytywać moment, w którym jest konieczna pomoc, a także szacować zaangażowanie w rozwiązywanie zadań [150].

Chmury obliczeniowe wspierają zdalne nauczanie specjalistycznego oprogramowania lub trening obsługi sprzętu w oddalonym laboratorium [48, 91]. Można przewidywać oceny, przy czym dane o postępach, zaangażowaniu i sposobie uczenia się mogą być zbierane w wielu miejscach kampusu w ramach obowiązujących przepisów [6, 80].

Warto podkreślić, że również gry komputerowe aktywizują studenta za pomocą rywalizacji i kooperacji [148]. Uczący się mogą zdobywać punkty za ukończenie fragmentu kursu, a także za jakość napisanych postów na forach [97]. Natomiast media społecznościowe wspierają informowanie lokalnej społeczności akademickiej, w tym wyjaśniają procedury administracyjne oraz prezentują informacje o funkcjonowaniu uczelni [42].

Rodzaje e-learningu w inteligentnej edukacji przedstawiono na rysunku 1. Asynchroniczne zdalne nauczanie pozwala na kontakt z nauczycielem poprzez grupy dyskusyjne, pocztę elektroniczną, biuletyny oraz media społecznościowe. Natomiast nauczanie hybrydowe łączy tradycyjne metody nauczania z nauką online. Samokształcenie pozwala uczestnikowi na dostosowanie terminu zajęć i tempa nauki do własnych możliwości. W nauczaniu mobilnym (ang. *m-learning*) wykorzystywane są bezprzewodowe urządzenia mobilne (smartfony, tablety i laptopy) [86].

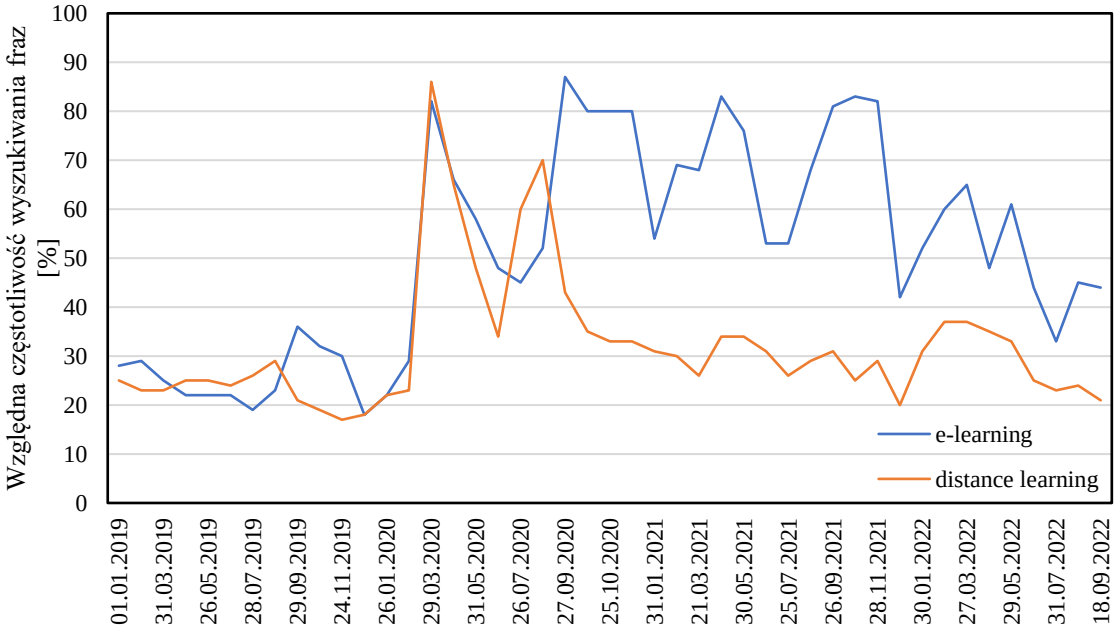


Rysunek 1 Rodzaje e-learningu w inteligentnej edukacji [179].

W krajach wysoko rozwiniętych ponad pięćdziesiąt procent studentów uczestniczy zdalnie w zajęciach [54]. Przed pandemią COVID-19 większość z nich wybierała kursy e-learningowe z powodu pracy zawodowej, uciążliwych dojazdów czy doskonalenia profesjonalnych umiejętności na dodatkowych zajęciach [54]. Rozpowszechnienie się zdalnego nauczania oraz różnorodność platform spowodowało wprowadzenie standardu zapisu kursów SCORM (ang. *Sharable Content Object Reference Model*).

Na rysunku 2 ukazano intensywność wyszukiwania przez internautów pojęć *distance education* i *e-learning* w wyszukiwarce Google. Widoczne jest znaczące zwiększenie zainteresowania *e-learningiem* w latach 2020-2021, kiedy ze względu na pandemię uczelnie intensywnie poszukiwały bezpiecznych sposobów kontynuacji kształcenia.

Wg OECD (*Organization for Economic Cooperation and Development*) smart edukacja istotnie wspiera również gospodarkę opartą na wiedzy, w której proces edukacji towarzyszy intensywnie pracy zawodowej [54].



Rysunek 2 Względna częstotliwość wyszukiwania fraz *Distance Education* oraz *E-learning* za pomocą wyszukiwarki Google w latach 2019 – 2022 [60].

1.2 Wybrane zastosowania zdalnego nauczania

Nowoczesne technologie informatyczne w wielu wypadkach powstają dla potrzeb militarnych, a zdalne szkolenie odgrywa priorytetową rolę w wojskach NATO. W portalu NATO School w Oberammergau z kursów online korzysta ponad 12 000 użytkowników [4]. Zdalne nauczanie realizuje się również na Uniwersytecie Bundeswery oraz w Szkole Lotniczej w Bückeburgu, w której wdrożono system zdalnego szkolenia dla pilotów śmigłowców zintegrowany z trójwymiarowym symulatorem pilotażu [129].

W portalu edukacyjnym AKO (ang. *Army Knowledge Online*) zarejestrowano ponad 2 miliony użytkowników, a dzienna liczba logowań dochodzi do miliona [4]. Portal udostępnia kilkadziesiąt serwisów US Army, w których zaindeksowano ponad 3 miliony stron webowych [126]. Opierając się na profilu użytkownika oraz stopniu wojskowym, oprogramowanie rekomenduje odnośniki do zasobów [126]. Możliwy jest dostęp do ponad 5 400 kursów z zakresu wojskowości, informatyki, języków obcych lub biznesu. Natomiast dla rodzin żołnierzy dostępnych jest około 17 000 książek i 2 800 kursów online, w tym oferowanych przez Cisco i Microsoft [126]. W szczególności, firma Rosetta Stone szkoli w zakresie kilkudziesięciu języków obcych [126].

Natomiast w armii brytyjskiej za pomocą portalu DLP (ang. *Defence Learning Portal*) udostępniono ponad tysiąc kursów [127]. Z kolei armia francuska realizuje zdalne nauczanie przy pomocy oprogramowania SAP Business Objects Knowledge Accelerator. Należy dodać, że rozwiązania firmy SAP wykorzystywane są także w zdalnym szkoleniu sił powietrznych USA, szwedzkich sił zbrojnych oraz hiszpańskiej straży cywilnej [128]. W armii kanadyjskiej Centrum Wsparcia Zdalnego Nauczania zatrudnia ponad 2 000 instruktorów, deweloperów oraz administratorów [125].

W uczelniach cywilnych Kanady, która liczy ok. 28 mln mieszkańców, ponad 1 600 szkół oferuje ok. dwóch tysięcy kursów. Atutem tej klasy uczelni wirtualnej jest mniejsze obciążenie tradycyjnej infrastruktury, a student odwiedza siedzibę szkoły tylko podczas laboratoriów, egzaminów, zaliczeń czy zjazdów. W ten sposób studiuje również ok. 200 000 studentów w The Open University, który jest największym pod względem liczby studentów w Wielkiej Brytanii [160]. Budżet uniwersytetu to ok. pół miliarda funtów.

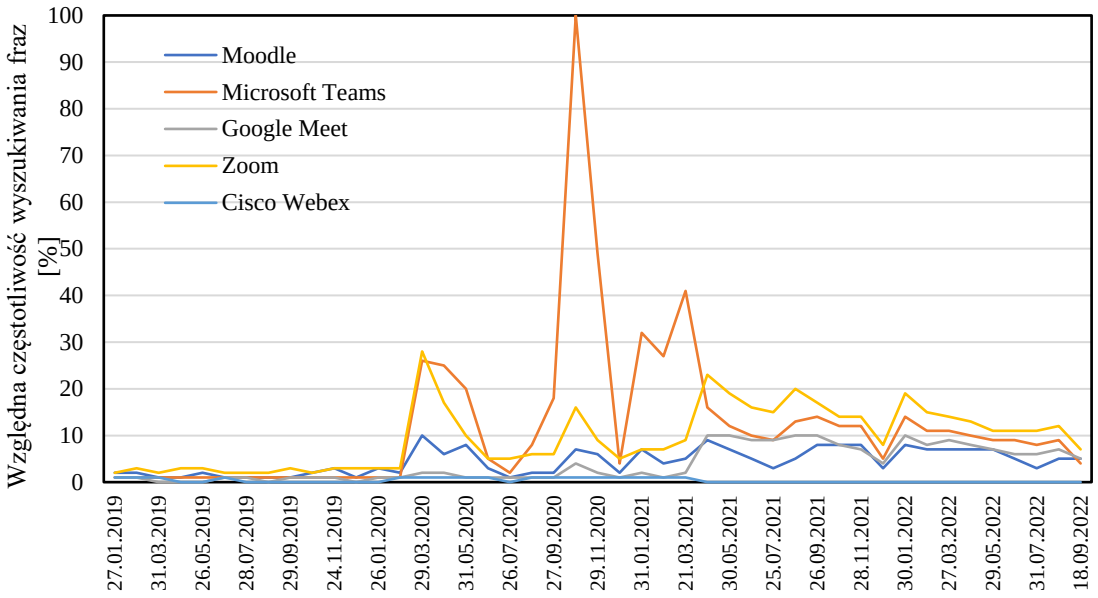
W Polsce ważną rolę odgrywa Ośrodek Kształcenia Na Odległość OKNO Politechniki Warszawskiej [130]. Tematyka e-learningu jest również przedmiotem wydawnictwa Szkoły Głównej Handlowej E-mentor [131]. Natomiast system SPRINT oferuje materiały dydaktyczne, zdalne rozwiązywanie zadań, zadania projektowe oraz konsultacje [130].

Do popularnych platform zdalnego nauczania należą: MS Teams [99], Zoom [188], Google Meet [59] oraz Cisco Webex [31]. Ponadto stosowano Blackboard [51], Oracle iLearning [121], Manhattan Virtual Classroom [120], a także WBTServer [123]. Niektóre z platform są wycofywane, w tym Claroline [116], ATutor [115], Ilias [118] oraz WebCT [51]. W szkolnictwie coraz rzadziej wykorzystuje się również systemy klasy Pelp [122], Schoology [145], Skype [147] oraz Dokeos [36]. Sporadycznie stosowane są platformy Myicourse [105], Sakailms [142], Fle3 [117] oraz Lotus [119].

Przed pandemią COVID-19 dużym zainteresowaniem cieszyły się platformy Cisco WebEx, Zoom oraz Moodle (rys. 3). Podczas pandemii zainteresowanie platformą Moodle wzrosło, jednak platformy umożliwiające wideokonferencje zyskały jeszcze większą popularność.

Szkolenie na odległość pracowników w Wielkiej Brytanii jest bardzo popularne. Na portalu Ministerstwa Pracy zarejestrowanych jest ok. pół miliona użytkowników. W Polsce warto wymienić projekt E-Pracownik [157], w którym zastosowanie technologii 3D pozwala wszechstronnie przedstawić środowisko szkolenia [43]. Do nauczania na odległość wykorzystywane są również komunikatory internetowe, takie jak Skype

[106]. Dobrym pomysłem jest łączenie wybranych kanałów, np. wykorzystanie komunikatora WhatsApp w połączeniu z systemem Moodle [104].



Rysunek 3 Zainteresowanie wybranymi platformami zdalnego nauczania [60].

Podczas pandemii w polskich szkołach podstawowych i ponadpodstawowych zajęcia prowadzono w klasach wirtualnych w chmurze obliczeniowej w oparciu o MS Office 365 lub Google G Suit. Wykorzystywano również dziennik elektroniczny [124]. Najbardziej efektywną formą zdalnego nauczania okazała się wideokonferencja, realizowana za pomocą MS Teams, Zoom lub Google Meet. Dla uczniów, którzy nie mogli stosować kamer, przygotowano podcasty przy użyciu Audacity lub Free Audio Editor. Sporadycznie stosowano media społecznościowe i tablice interaktywne [124].

1.3 Inteligentne systemy agentowe

Podstawową cechą agenta jest odbieranie bodźców ze środowiska za pomocą sensorów, a także wpływanie na środowisko poprzez efektory [53, 79]. Maes określa agenta jako system obliczeniowy w dynamicznym środowisku, który odczuwa i reaguje autonomicznie. Agent realizuje postawione cele w oparciu o własny plan [19]. Ponadto agent wykonuje następujące funkcje: postrzega zmiany w środowisku, podejmuje akcje, wyciąga wnioski, a także rozwiązuje zauważone problemy [53].

Wooldridge i Jennings zaproponowali „słabe" pojęcie agenta, wg którego agent cechuje się: autonomicznością, zdolnością socjalną, reaktywnością oraz proaktywnością. Autonomiczność to posiadanie kontroli na własnym stanem oraz podejmowanie działań.

Zdolność socjalna umożliwia komunikację z innym agentem. Reaktywność to obserwacja środowiska i reagowanie na zauważone zmiany w zadanym czasie. Agent proaktywny ogranicza się do działań prowadzących do osiągnięcia celu [169]. Natomiast „silne" pojęcie agenta nawiązuje do stosowania koncepcji, którymi zazwyczaj posługuje się człowiek. Agenta cechuje wiedza, intencjonalizm i przestrzeganie obowiązków [169]. Powyższe potwierdza potencjał aplikacyjny agentów w edukacji.

Powodem braku powszechnie akceptowalnej definicji agenta jest fakt, że „pojęcie to jest narzędziem analizy systemów, a nie absolutną charakterystyką dzielącą świat na agentów i nie-agentów" [79]. Franklin stwierdza, że „jedynie pojęcie matematyczne pozwala na ustalenie ostrych granic kategorii" [53].

Do istotnych cech agenta zalicza się również usytuowanie (ang. *situatedness*) związane z odczytywaniem danych ze środowiska oraz podejmowaniem akcji. Responsywność to możliwość obserwowania środowiska i reagowania na zmiany pojawiające się w nim [167]. Kolejną ważną cechą jest umiejętność uczenia się, za pomocą której agent może zmienić swoje zachowanie, opierając się na doświadczeniach [79]. Natomiast racjonalność (ang. *rationality*) to dążenie do osiągnięcia postawionych celów za pomocą swojej wiedzy [166]. Ważną cechą agenta jest mobilność – zdolność przenoszenia się z jednej maszyny na inną [53]. Maszyny wirtualne, które mogą przemieszczać się między hostami w chmurze obliczeniowej, są nowym sposobem przenoszenia się agentów.

W wypadku podziału agentów ze względu na ich cechy biologiczne, można wyróżnić agenty biologiczne, autonomiczne roboty i agenty obliczeniowe [53]. Ten ostatni rodzaj agentów obejmuje agenty sztucznego życia i agenty komputerowe (ang. *software agents*). Natomiast agenty specjalne i wirusy są podklasą agentów komputerowych [53].

Inteligentny agent jest zdolny do podejmowania decyzji na podstawie poprzednich doświadczeń, wykorzystując umiejętność uczenia się [67]. W systemach wieloagentowych (ang. *multi-agent systems,* MAS) zadania realizowane są za pomocą rozproszonej sztucznej inteligencji (ang. *distributed artificial intelligence,* DAI) [79]. MAS to sieć agentów, potrafiących współpracować ze sobą [74]. Zazwyczaj agent ma niekompletną informację o sposobie rozwiązania problemu, a także nie funkcjonuje scentralizowany system nadzoru. Dane są zdecentralizowane, a obliczenia – asynchroniczne [74]. Wyróżnia się sześć rodzajów agentów: współpracujące, interfejsowe, mobilne, informacyjne, reaktywne oraz hybrydowe [107].

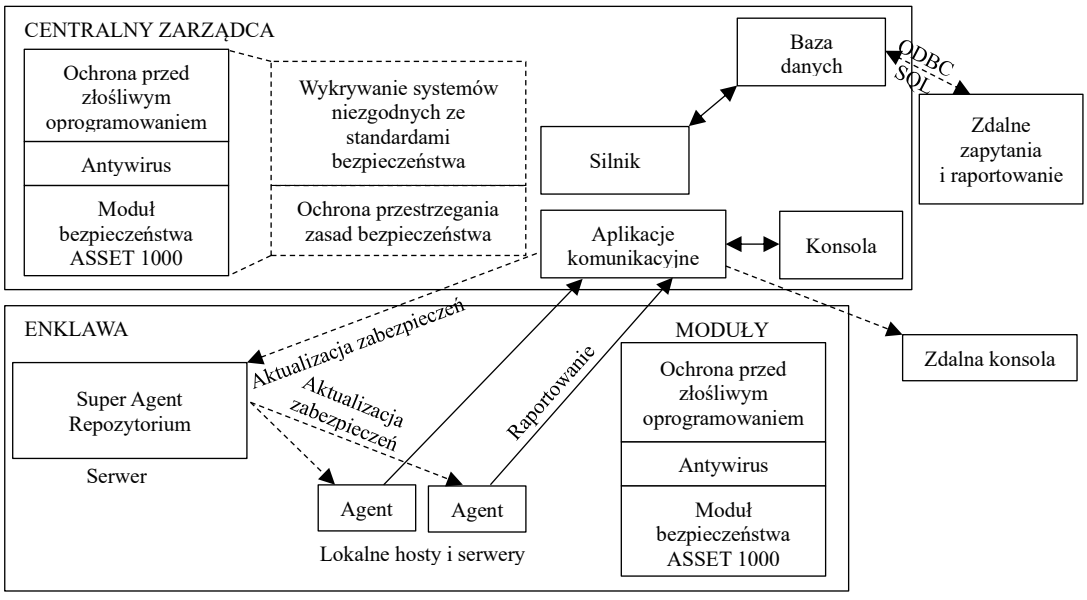
Agenty współpracujące cechują się autonomicznością, komunikatywnością i ograniczonymi zdolnościami uczenia się. [64]. Negocjacje pozwalają osiągnąć wzajemnie

akceptowalne porozumienie [107]. Dominującymi cechami agentów interfejsowych są autonomiczność oraz umiejętność uczenia się. Zadaniem agentów może być wspieranie i zapewnienie proaktywnej pomocy podczas nauki programowania. Agent obserwuje akcje studenta, a także sugeruje efektywny sposób wykonania zadania. Zdobywa wiedzę również na podstawie implementacji innych studentów [107].

Agenty mobilne mogą przemieszczać się do innych hostów w celu zbierania informacji, które „po powrocie do domu" są pomocne do realizacji zadań. oprogramowanie może przemieścić się w celu, np. poszukiwania najbardziej preferowanego filmu edukacyjnego. Agent po przejrzeniu i ocenie najbardziej wartościowego filmu, może przesłać skompre-sowaną kopię studentowi. Ponadto zbliżenie się do źródeł strumieni danych pozwala na redukcję transmisji danych w sieci [46, 107].

Agenty informacyjne potrafią zarządzać, manipulować lub porównywać informacje z różnych źródeł. Stosują wyszukiwarki i roboty internetowe do realizacji zadań, np. agent w przeglądarce internetowej [107]. Natomiast agenty reaktywne potrafią działać zgodnie z regułą „bodziec-reakcja" i odwzorowywać stan środowiska. Składają się z autonomicznych modułów, które zazwyczaj operują na danych z sensorów [107].

Przykładem systemu agentowego jest HBSS (ang. *Host-Base Security System*) firmy McAfee do ochrony zasobów chmury obliczeniowej (rys. 4). Agenty monitorują stan sprzętu i oprogramowania, sprawdzając, czy nie zostały naruszone zasady polityki bezpieczeństwa. W wypadku wykrycia anomalii na komputerze lokalnym, system może autonomicznie podjąć decyzję o jego odłączeniu [179].



Rysunek 4 Architektura systemu agentowego HBSS [179].

1.4 Agenty w e-learningu

Student podczas intensywnej nauki może odczuwać niewystarczające wsparcie, gdyż nie może uzyskać konsultacji od nauczyciela o dowolnej porze [29]. Ponadto, może nie być motywowany do systematycznej nauki. W tej sytuacji rolę asystenta uczącej się osoby przejmuje inteligentny agent pedagogiczny. W szczególności agent konwersacyjny może o dowolnej porze odpowiadać na pytania studentów [168]. Inteligentne agenty pedagogiczne mogą działać w symulatorach 3D. Przykładowo Open Wonderland jest oprogramowaniem, które umożliwia przeprowadzenie eksperymentów z fizyki [148].

Zastosowanie gier komputerowych w e-learningu aktywizuje uczących się, wprowadza element rywalizacji i kooperacji [43]. Inteligentne awatary ukierunkowane mogą być na podtrzymywanie aktywności studenta. Jeśli zatem student interesuje się muzyką, to eksperyment w wirtualnej pracowni chemicznej może przeprowadzić w towarzystwie ulubionego awatara-muzyka.

Inteligentny agent edukacyjny powinien spełniać dwa kryteria. Po pierwsze, powinien realizować inteligentne czynności, w tym wspomagać indywidualny tryb nauczania. Po drugie, powinien cechować się racjonalnym sprawstwem [43]. W stacjonarnym trybie studiowania dopasowanie sposobu nauki studenta do tempa przyswajania wiedzy nie jest zazwyczaj możliwe przy zachowaniu masowości przekazu, gdyż wymagałoby to indywidualnego podejścia nauczyciela do studenta [171]. Natomiast agenty mogą odwzorować zdobytą wiedzę przez studenta [55]. Pozwala to na określenie, co nie zostało jeszcze przyswojone, a następnie wyznaczenie obszarów kształcenia uzupełniającego.

Jeśli nie jest możliwy bezpośredni kontakt nauczyciela ze studentem, trudno jest ocenić postępy [55]. Natomiast agenty mogą wskazać, którzy studenci wymagają dodatkowej uwagi. Wnioski te wyciągane są na podstawie czynności wykonanych przez najlepszych uczących się, którzy ukończyli przedmiot [181]. Warto wspomnieć o technicznym ograniczeniu, gdyż smartfony posiadają limity odnośnie do szybkości przetwarzania danych i wielkości przechowywania danych, co wymaga dostosowania treści do możliwości urządzenia. W systemie ISABELE (ang. *Information Software Agent-Base E-Learning*) udostępnia się treści w odpowiednim dla urządzeń formacie Learning Objects [56].

Ważną funkcjonalnością jest przygotowanie nowych testów. Egzaminator po wpisaniu słów kluczowych otrzymuje test przygotowany przez agenta, który uwzględnia poziom trudności oraz stanu wiedzy studentów. Po rozwiązaniu testu, agent weryfikuje oceny, wysyła je do studentów i nauczyciela wraz ze wskazówkami merytorycznymi [30].

Rozpoznawanie emocji studentów podczas zajęć, w tym zadowolenia z odpowiedzi, zdenerwowania, czy zaskoczenia treścią pytania, pomaga w procesie nauczania. Emocje wywierają wpływ na procesy przyswajania wiedzy, w tym: zwiększoną uwagę, zapamiętywanie, czy sprawne podejmowanie decyzji. Agenty mogą powiadomić wykładowcę o potrzebie reakcji, gdyż, np. poziom koncentracji studentów jest niewystarczający. Agent może informować również studenta o jego aktualnym stanie koncentracji w stosunku do stanu, w którym uzyskał najlepsze rezultaty [12].

Podsumowując, agentom pedagogicznym można przypisać następujące role: kordynator, egzaminator, obserwator, osobisty nauczyciel lub indywidualny konsultant (rys. 5). Koordynator nadzoruje przebieg nauczania, reaguje, jeśli student nie może przyswoić materiału.

		Tryb nauczania	
		Nauczanie synchroniczne	**Nauczanie asynchroniczne**
Miejsce	**To samo: nauczanie kolokacyjne**	Tradycyjne nauczanie	Koordynator Osobisty Egzaminator
	Inne: nauczanie na odległość	Koordynator Osobisty Obserwator	Obserwator Osobisty Egzaminator

Rysunek 5 Agenty pedagogiczne w e-learningu [43].

Egzaminator nadzoruje przebieg zdalnego egzaminu. Obserwator śledzi proces kształcenia, analizuje, ile czasu student poświęcił na przyswojenie wymaganej wiedzy. Osobisty asystent dostarcza studentowi materiał uzupełniający i wspomaga podczas rozwiązywania zadań. Natomiast indywidualny konsultant dobiera najbardziej efektywny sposób nauki. W analogiczny sposób można stosowanie agenty do informowania mieszkańców inteligentnych miast. Udostępniane mogą być w sposób przystępny informacje o sytuacji w mieście, w tym o usługach publicznych, wprowadzonych zmianach w ruchu, sposobie aplikowania o dokumenty, czy obsłudze automatów dostępnych w infrastrukturze miejskiej [39, 44].

1.5 Wnioski i uwagi

Rozpowszechnienie Internetu i telefonii mobilnej zniosło bariery w dostępie do edukacji oraz umożliwia uwzględnienia indywidualnych możliwości i oczekiwań studentów. Dużym sukcesem zdalnej edukacji jest fakt, że niezbyt doskonałe jeszcze narzędzia podczas pandemii koronawirusa umożliwiły przeprowadzenie zajęć na wszystkich poziomach edukacji. Jednym z ciekawszych niekomercyjnych systemów zdalnego nauczania jest Moodle, który umożliwia wprowadzenie inteligentnych agentów pedagogicznych.

Agenty pedagogiczne zwiększają efektywność nauczania. Materiały dydaktyczne w formie papierowej są niewystarczające w niektórych dziedzinach. Powodem tego jest konieczność wizualizacji 3D oraz zwiększająca się ilość informacji, które należy przyswoić. Agenty mogą filtrować treści pod kątem potrzeb każdego studenta, wykrywać niedostatki zasobów, dostosowywać materiał do urządzeń, a także rozpoznawać emocje towarzyszące nauczaniu.

Agent może być koordynatorem przebiegu nauczania, reagując w wypadku niedostatecznego przyswojenia materiału. Ponadto może nadzorować egzaminy, przygotować pytania egzaminacyjne, ocenić odpowiedzi, wskazać materiały uzupełniające, informować o ocenach, a także przygotować raporty z pogłębioną analizą.

Agent monitorujący może śledzić proces studiowania, w tym analizować, ile czasu student poświęcił na przyswojenie wymaganej wiedzy. Agent może być również asystentem studenta, dostarczając materiał uzupełniający i wspomagając rozwiązywanie zadań. Agent konsultant cechuje się umiejętnością doboru najbardziej efektywnego sposobu nauki dla studenta.

Warto również podkreślić, że implementacja agentów pedagogicznych opiera się na wykorzystaniu ważnego obszaru sztucznej inteligencji, jakim jest uczenie maszynowe. Rozwój oprogramowania klasy TensorFlow oraz PyTorch umożliwia względnie sprawną implementację procesu treningu i testowania adekwatnych modeli edukacyjnych opierających się na klasyfikacji, regresji lub predykcji. Jednakże kluczowym problemem pozostaje skonstruowanie zbiorów danych, na podstawie których wytrenowane zostaną modele wspierające działanie agentów pedagogicznych.

Kolejnym ważnym zagadnieniem jest integracja wytrenowanych modeli i zasobów edukacyjnych w chmurze obliczeniowej. Ze względu na skalę uczelni wymagana jest optymalizacja zasobów komputerowych z wykorzystaniem maszyn wirtualnych. Istotne jest również zapewnienie mobilności agentów pedagogicznych.

2. WYBRANE METODY SZTUCZNEJ INTELIGENCJI W E-KSZTAŁCENIU

Sztuczna inteligencja w nauczaniu na odległość umożliwia wspieranie procesu studiowania. Dostosowuje proces nauczania do indywidualnego profilu uczestników. Pozwala wykładowcom na szybszą identyfikację potencjalnych problemów oraz obiektywne ocenianie. Ponadto umożliwia dopasowanie zasobów chmury obliczeniowej do zapotrzebowania agentów pedagogicznych.

2.1 Programowanie genetyczne

Programowanie genetyczne w e-learningu zastosowano do analizy procesu nauczania. W trybie asynchronicznym nauczyciel nie ma możliwości bezpośredniej komunikacji ze studentami i nie może obserwować ich reakcji podczas przyswajania materiału. Wówczas pomocna jest analiza interakcji studentów, która umożliwia dostosowanie treści nauczania do poziomu wiedzy uczących się [141].

Zastosowano reguły, które mogą wskazywać na poziom opanowania materiału przez studenta. Reguły mogą być reprezentowane za pomocą zdań typu jeżeli-to wyznaczonych za pomocą programowania genetycznego opartego o gramatykę. Uwzględnia się czas spędzony na studiowaniu wybranych materiałów oraz oceny z testów. Na podstawie wyznaczonych reguł można modyfikować przedmiot, aby zwiększyć skuteczność nauczania i dostosować go do poziomu wiedzy studentów [141].

Za pomocą programowania genetycznego można również prognozować oceny na podstawie analizy interakcji studentów. Dane badane są pod kątem ich wpływu na zaliczenie przedmiotu. Uwzględnienie poziomu trudności wykonanych zadań oraz czasu spędzonego na ich realizacji są kluczowe. Szacuje się również nakład pracy niezbędny do uzyskania zaliczenia [180].

Za pomocą programowania genetycznego wyznacza się również ścieżki uczenia uzupełniającego na podstawie nieprawidłowych odpowiedzi w testach. Agent może opracować spersonalizowane treści nauczania, uwzględniając poziomy trudności przedmiotów. Podejście to pozwala na opracowanie wysokiej jakości indywidualnych ścieżek edukacyjnych [28]. Programowanie genetyczne może również być wykorzystane jako metoda treningu modeli głębokich sieci neuronowych w innych zastosowaniach dydaktycznych. Ponadto metodę programowania genetycznego zastosowano do optymalizacji zasobów w chmurach obliczeniowych.

2.2 Sztuczne sieci neuronowe

Zdolność sztucznych sieci neuronowych (ANN) do modelowania złożonych relacji jest wykorzystywana do prognozowania efektów kształcenia [82, 143]. Wynikiem predykcji może być ocena, czy student zaliczy wybrany przedmiot. Ponadto można prognozować ocenę końcową. Susena zaproponowała trzy metody klasyfikacji studentów na podstawie liczby punktów osiągniętych w teście. W pierwszej metodzie skala ocen jest od 1 do 10, w drugiej rozważa się trzy klasy: „słaby", „dobry" lub „bardzo dobry", a w trzeciej metodzie − dwie klasy: pozytywna i negatywna [156].

Zbiór obserwacji zawierał 476 wyników testu składającego się z 25 pytań wielokrotnego wyboru, a za poprawną odpowiedź dodawano punkt. Wyniki zebrano w ciągu dwóch lat. Podczas testu student miał nie więcej niż dwie próby, aby udzielić odpowiedź. W modelu rozważa się pięć zmiennych objaśniających: liczbę poprawnych odpowiedzi, liczbę poprawnych odpowiedzi przy pierwszej próbie, liczbę prób potrzebnych do uzyskania poprawnych odpowiedzi, łączny czas trwania testu oraz łączny czas udzielania poprawnych odpowiedzi. Porównano model wielowarstwowego perceptronu MLP (ang. *Multi Layer Percepton*) z efektywniejszą radialną siecią wielowarstwową, w której ukryta warstwa neuronów cechowała się radialnymi funkcjami aktywacji RBF (ang. *Radial Basis Function*). Wytrenowany model umożliwił klasyfikację studenta wg wybranej skali na podstawie przeprowadzonego testu [156].

W modelu Sayeda-Bakera występuje 56 zmiennych objaśniających. Zmienne te cechują zaangażowanie studenta w naukę przy wykorzystaniu platformy zdalnego nauczania, a także samodzielność studenta. Uwzględnia się wykształcenie oraz wprowadzane treści do systemu zdalnego nauczania. W modelu analizuje się jakość przedmiotu, a także wsparcie uczelni wyrażone za pomocą liczby postów wysłanych do studenta. Rozważa się dostępność uczącego się oraz przepustowość systemu. Kluczowe są również zmienne objaśniające proces edukacji oraz zaangażowanie osoby odpowiedzialnej za przedmiot. Dataset zawierał dane 1879 studentów, dla których prognozowano średnią ocenę z przedmiotu. Model opiera się na sieci MLP z 50 neuronami w warstwie ukrytej [143].

Interesującym sposobem wykorzystania ANN w procesie kształcenia na odległość jest użycie ich do analizy aktywności uczącego się. Kolekar, Sanjeevi i Bormane charakteryzują zachowanie studenta za pomocą liczby logowań, czasów sesji, a także preferowanego stylu uczenia, w tym liczby wypowiedzi, liczby odpowiedzi, czy liczby zadań. Model wspomaga przygotowanie spersonalizowany materiału z przedmiotu [81].

Wykorzystuje się cztery metody klasyfikacji stylu uczenia się. Pierwsza metoda klasyfikuje spostrzegawczość jako sensomotoryczną lub semantyczno-operacyjną. Druga metoda klasyfikuje sposób przyswajania wiedzy: werbalnie lub z wykorzystaniem wizualizację. Trzecia metoda klasyfikuje studentów na podstawie zaangażowania: aktywne zaangażowany lub refleksyjny. Czwarta metoda rozważa sposób zrozumienia materiału: uczenie się sekwencyjnie lub uczenie się przez przeskoki między partiami materiału [81]. Natomiast Wu et al. zaproponowali model głębokiej sieci neuronowej do rekomendacji sposobu przyswajania wiedzy w systemie zdalnego nauczania. Model uczy się zasad przekazywania wiedzy w systemie zdalnego nauczania [170].

Wykorzystując głębokie uczenie, Abdessamad i Nour-Eddine skonstruowali model, który dostosowuje zawartość przedmiotu do wiedzy studenta tak, aby poświęcił on założony czas na przyswojenie wiedzy na najwyższym poziomie. Analizowane są cechy procesu poznawczego studenta na podstawie jego aktywności podczas realizacji zadań. Wykorzystano dane o interakcji studenta z systemem oraz dane z dziennika zdarzeń [27].

Warto podkreślić, że Zhang et al. zastosowali sieci głębokie do klasyfikacji stylu uczenia się [183]. Do treningu wykorzystano dane z platformy StarC, na której udostępniono dwieście kursów od marca 2015 roku do sierpnia 2016 roku. Spośród zarejestrowanych 85 345 studentów, 32 987 aktywnie korzystało z platformy [183].

Splotowa sieć neuronowa (ang. *convolutional neural network,* CNN) może rozpoznawać emocje studenta podczas zajęć. Zazwyczaj wykorzystuje się zbiory obserwacji: CK+, JAFFE, NVIE i FER13, przy czym zbiór danych CK+ zawiera 593 zdjęć twarzy 123 studentów. Klasyfikuje się osiem rodzajów emocji, w tym: neutralność, szczęście, zdziwienie, złość, wstręt, strach, smutek i pogardę [155].

Natomiast zbiór JAFFE zawiera 213 zdjęć ekspresji twarzy o rozdzielczości 256 x 256 dla 10 japońskich kobiet w odniesieniu do siedmiu rodzajów emocji. Z kolei zbiór NVIE zawiera zdjęcia profili twarzy 100 studentów z mimiką cechującą sześciu rodzajów emocji. Zdjęcia są pozowane lub spontaniczne, a wykonano je w zakresie widzialnym lub podczerwonym. Do wykonania zdjęć użyto kamery zwykłej, kamery termowizyjnej i kamery na podczerwień. Warto wspomnieć, że zbiór FER13 zawiera około 28 000 obrazów w formacie 48×48 w odniesieniu do siedmiu emocji. Natomiast zbiór KDEF (*The Karolinska Directed Emotional Faces*) jest zestawem 4 900 zdjęć 70 osób w formacie 562×762 [155].

Xu et al. do rozpoznawania koncentracji studenta podczas zajęć zaproponowali model CNN, który analizował położenie głowy względem kamery. Jeżeli student spoglądał

w kierunku kamery, to oceniano, że jest skupiony na zajęciach. Należy zauważyć, że przetwarzanie obrazów podczas treningu modelu wymaga dużej wielkości pamięci RAM rzędu kilkudziesięciu GB [172].

Jedną z tendencji w sztucznej inteligencji jest łączenie kilku metod, a następnie sprawdzanie skuteczności metody hybrydowej. Z tego nurtu badawczego wyłoniły się metody neuro-ewolucyjne [150]. Trzy główne zastosowania tych metod to: uczenie ze wzmocnieniem, kooperacja inteligentnych agentów w wirtualnym środowisku oraz symulatory biologii ewolucyjnej. Metody można zastosować do wspomagania zdalnego nauczania w analogiczny sposób jak sztuczne sieci neuronowe. W szczególności rozgrywają minigry ze studentem, co aktywizuje uczącego się, powodując efektywniejsze zapamiętanie materiału [81].

W grze NERO (ang. *Neuroevolving Robotic Operatives*) na platformie OpenNERO uczy się awatary omijania stanowisk ogniowych przeciwnika za pomocą algorytmu neuro-ewolucyjnego [151]. Sztuczne sieci neuronowe generowane są tak, aby modelowały zasady odpowiadające zdefiniowanej na początku maszyny stanów. Późniejszy proces treningu sieci uwzględnia nietypowe aktywności uczącego się [32]. Metody neuro-ewolucyjne mogą identyfikować niedostatki wiedzy studenta i rekomendować treści komplementarne.

2.3 Maszyna wektorów wspierających

Maszyna wektorów wspierających (ang. *Support Vector Machine,* SVM*)* jest metodą uczenia maszynowego wykorzystywaną do konstruowania klasyfikatorów, w tym modeli do klasyfikacji obrazów, rozpoznawania twarzy, identyfikacji emocji, kategoryzacji tekstu, czy rozpoznawania mowy.

Celem SVM jest wyznaczenie hiperpłaszczyzny, która separuje dwie klasy punktów wejściowych z maksymalnym możliwym marginesem. Dzięki zastosowaniu triku kernelowego, hiperpłaszczyzna separująca może zostać wyznaczona dla wielowymiarowej przestrzeni cech o zwiększonej liczbie wymiarów, co pozwala na separację punków, które w przestrzeni zbioru uczącego nie są liniowo separowalne. Algorytm wyznacza rozwiązania adekwatnego zagadnienia optymalizacji dla procesu treningu. Badania empiryczne pokazały, że maszyna wektorów nośnych efektywnie trenuje klasyfikatory dla niezbyt licznego zbioru uczącego oraz jest odporna na problem nadmiernego dopasowania [58].

Gong i Wang zastosowali SVM w celu dostosowania treści nauczania do preferencji uczącego się [58, 86]. Niektórzy studenci preferują graficznie przedstawiane informacje, podczas gdy inni najefektywniej przyswajają wiedzę w formie tekstu. Niektórzy wolą podejście od szczegółowych informacji do ogólnych, pozostali − odwrotnie. Zmienne objaśniające, jakie brano pod uwagę to: rodzaj treści wybierany przez studenta (np. tekst, wideo, grafika, czy audio), sekwencja wybieranych treści (np. treści szczegółowe, treści ogólne) oraz czas nauki. Wytrenowany model ułatwia przygotowanie prezentacji wykładu, która odpowiada wyznaczonemu stylowi uczenia się studenta [58].

Pawar i Sonkar wykorzystali SVM do skonstruowania modelu klasyfikacji pytań tekstowych zadawanych przez studentów [113]. Nauczyciel ze względu na ograniczenia czasowe nie jest w stanie odpowiedzieć na wszystkie z nich, co uzasadnia potrzebę selekcji najbardziej istotnych dylematów. Cechami branymi pod uwagę są: zgodność treści pytania z tematem wykładu oraz istotność pytań studenta. Uwzględnia się również podobieństwa między pytaniami. Zastosowanie modelu Pawara-Sonkara może w znaczący sposób podnieść jakość komunikacji między nauczycielem a studentem [113].

2.4 Ewolucja różnicowa

Algorytmy ewolucyjne stosowane są w optymalizacji i uczeniu maszynowym [92]. W szczególności algorytm ewolucji różnicowej (ang. *Differential Evolution,* DE), który opracowali Price i Storn wielokrotnie uznany został za najlepszy w konkursach International Contest on Evolutionary Optimization [33, 152].

Niech populacja składa się z L wektorów (chromosomów), przy czym każdy element zawiera I współrzędnych będących liczbami rzeczywistymi. Główną operacją odróżniającą algorytm ewolucji różnicowej od innych algorytmów ewolucyjnych jest mutacja różnicowa [66]. Niech populacja początkowa (j=0) jest generowana losowo. Współrzędne wektorów można wyznaczyć za pomocą poniższej zależności [126]:

$$x_{ml}(j) = rand_l(0,1) * (b_l^H - b_l^L) + b_l^L, \quad i = \overline{1,I};\ l = \overline{1,L}, \tag{2.4.1}$$

gdzie:

$x_{il}(j)$ – i-ta współrzędna l-tego wektora x w populacji numer j,

$rand_l(0,1)$ – liczba losowa z przedziału [0; 1] wygenerowana dla l-tego wektora,

b_l^H – górna granica l-tej współrzędnej,

b_l^L – dolna granica l-tej współrzędnej.

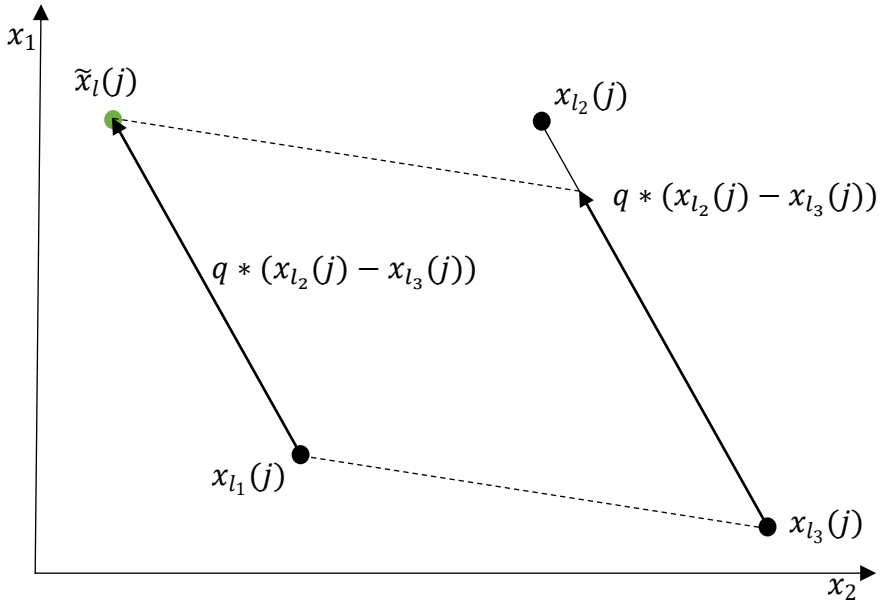
W podstawowej strategii mutacji różnicowej po wylosowaniu z populacji rozwiązania bazowego nr l_1, losuje się bez powtórzeń dwa inne warianty nr l_2 oraz l_3. Następnie do wektora nr l_1 dodaje się wyskalowaną różnicę wektorów nr l_2 oraz l_3 (rys. 6). Z uwagi na przyjęty równomierny rozkład prawdopodobieństwa losowania, może wystąpić sytuacja, w której pewne rozwiązanie nie zostanie wylosowane do mutacji [152]. Wynik mutacji różnicowej można wyznaczyć następująco:

$$\tilde{x}_l(j) = x_{l_1}(j) + q * \left(x_{l_2}(j) - x_{l_3}(j)\right), \quad l \neq l_1 \neq l_2 \neq l_3, \tag{2.4.2}$$

gdzie:

$\tilde{x}_l(j)$ – l-ty wektor po mutacji różnicowej w populacji numer j,

q – zadany współczynnik skalujący w mutacji różnicowej.



Rysunek 6 Graficzna interpretacja mutacji różnicowej w przestrzeni dwuwymiarowej [152].

Istotną rolę w zapewnieniu odpowiedniej różnorodności populacji odgrywa współczynnik skalujący $q > 0$, który reguluje wpływ różnicy dwóch wylosowanych wektorów na wektor bazowy. Rekomendowana jest wartość współczynnika $q \in [0{,}4; 1]$ [33, 152].

Po przeprowadzeniu mutacji rozważa się populację rodziców $P(j)$ oraz populację $PM(j)$ tymczasowych rozwiązań wyznaczonych za pomocą mutacji (rys. 7). Pary rozwiązań z obu populacji krzyżuje się tak, że dla i-tej współrzędnej l-tego konstruowanego wektora $\hat{x}_l$ do populacji $PC(j)$ losuje się liczbę $rand_{il} \in [0; 1], i = \overline{1,I}, l = \overline{1,L}$. Jeżeli $rand_{il}$ jest mniejsza od progu akceptacji współrzędnej $Cp \in [0;\ 1]$, to i-tą współrzędną wektora $\hat{x}_l$ jest i-ta współrzędna l-tego wektora z populacji $PM(j)$ po mutacji.

W przeciwnym wypadku jest akceptowana współrzędna rodzica, jak niżej [153]:

$$\hat{x}_{il}(j) = \begin{cases} x_{il}(j) \text{ jeśli } rand_{il} > Cp, \\ \tilde{x}_{il}(j) \text{ w przeciwnym wypadku,} \end{cases} \tag{2.4.3}$$

gdzie:

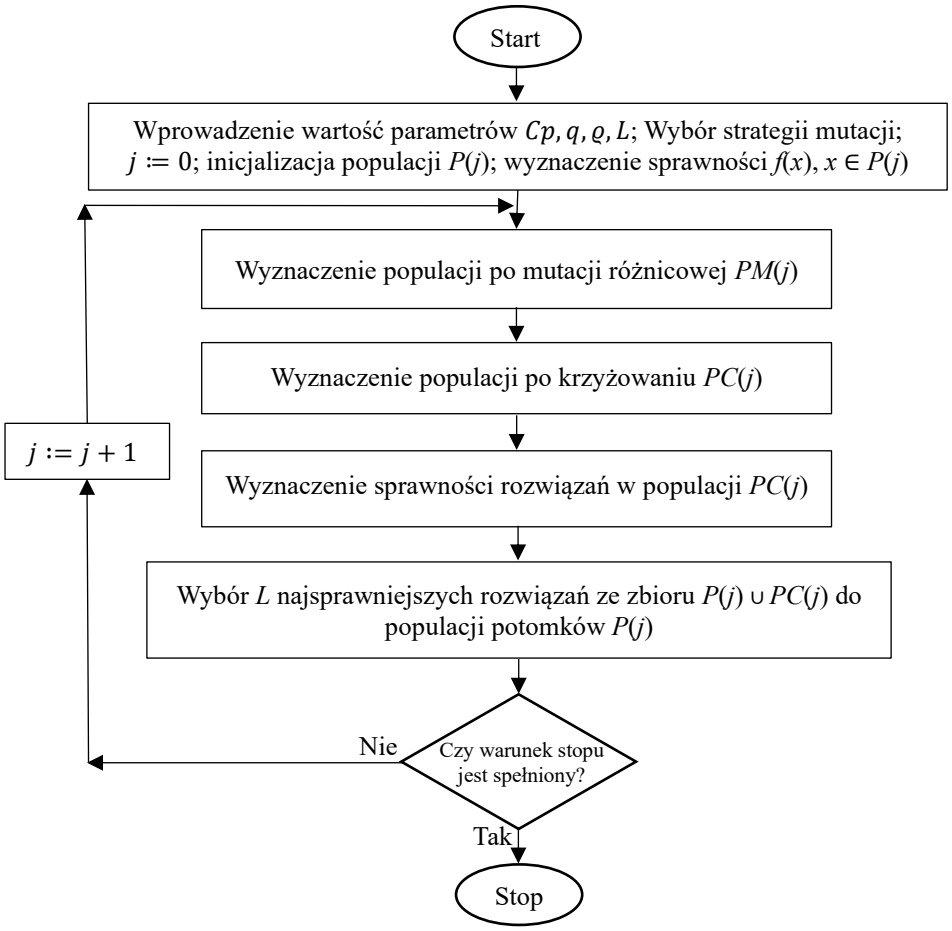
$\hat{x}_{il}(j)$ – i-ta współrzędna l-tego rozwiązania populacji potomnej $PC(j)$,

Cp – próg akceptacji współrzędnych potomków w krzyżowaniu.

Jeśli podczas krzyżowania nie zostanie wybrana żadna współrzędna wektorów z populacji PM, to losuje się zadaną liczbę współrzędnych ϱ z PM. Zapewnia to minimalny poziom różnorodności podczas eksploracji przestrzeni przeszukiwań [176].

Po mutacji i krzyżowaniu następuje sukcesja polegająca na skonstruowaniu populacji potomnej. Porównuje się oceny rodziców z ocenami potomków o tym samym indeksie. Do nowej populacji przechodzą rozwiązania o wyższej wartości funkcji sprawności f:

$$x_l(j+1) = \begin{cases} x_l(j) \text{ jeśli } f\big(x_l(j)\big) > f\big(\hat{x}_l(j)\big), \\ \hat{x}_l(j) \text{ w przeciwnym razie,} \end{cases} \quad l = \overline{1, L}, \;\; j = \overline{1, J_{max}}. \tag{2.4.4}$$



Rysunek 7 Diagram algorytmu ewolucji różnicowej do optymalizacji jednokryterialnej.
Źródło: opracowanie własne na podstawie [108, 152].

Proces ewolucyjnego konstruowania populacji jest zatrzymywany, jeśli wyznaczone zostanie rozwiązanie optymalne lub osiągnięta zostanie maksymalna liczba populacji K_{max}. Innym warunkiem stopu jest przekroczenie czasu obliczeń lub brak poprawy wartości uśrednionej funkcji sprawności populacji podczas założonej liczby generacji [152].

W celu osiągnięcia wysokiej jakości rozwiązań dobiera się parametry q i Cp [152]. Eksperyment dostrajający te parametry w ramach rozprawy polegał na wielokrotnym rozwiązywaniu instancji problemu wielokryterialnej optymalizacji, przy czym badano jakość wyznaczanych rozwiązań Pareto-optymalnych dla wybranych wartości parametrów q oraz Cp (rys. 8).

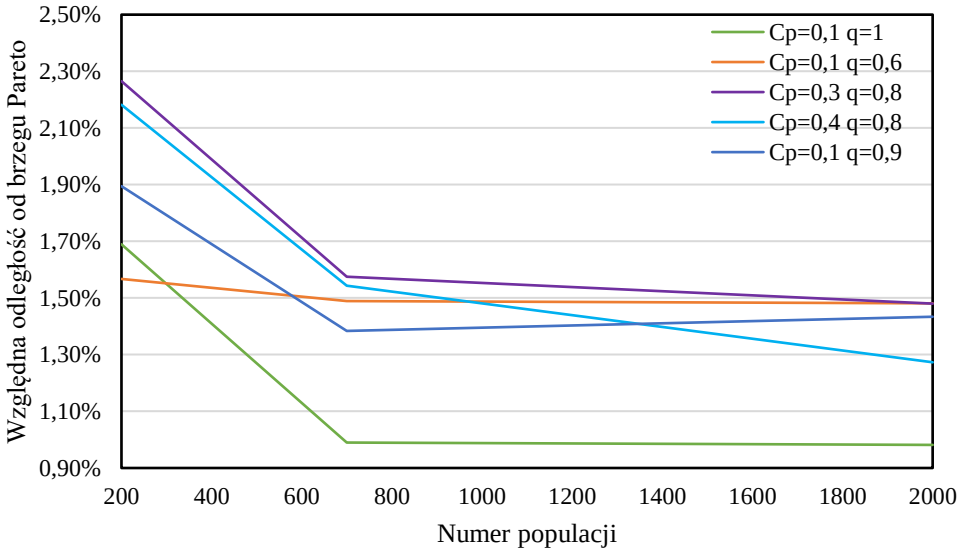
Warto podkreślić, że mutacja może być realizowana za pomocą różnych strategii. Zależność (2.4.2) opisuje strategię podstawową DE/rand/1, gdzie rand oznacza losowy sposób wyboru rozwiązania bazowego, a 1 – różnicę dwóch wybranych wariantów $\left(x_{l_1}(k) - x_{l_2}(k)\right)$. Natomiast strategia DE/rand/2 polega na uwzględnieniu dwóch różnic wektorów, które konstruuje się za pomocą czterech wylosowanych wektorów z populacji:

$$\tilde{x}_l(j) = x_{l_1}(j) + q\left(x_{l_2}(j) - x_{l_3}(j)\right) + q\left(x_{l_4}(j) - x_{l_5}(j)\right), \quad l_1 \neq l_2 \neq l_3 \neq l_4 \neq l_5. \tag{2.4.5}$$

Interesująca jest strategia DE/best/1 uwzględniająca „lidera” populacji jako rozwiązanie bazowe w następujący sposób:

$$\tilde{x}_l(j) = x_{best}(j) + q\left(x_{l_1}(j) - x_{l_2}(j)\right), \quad best \neq l_1 \neq l_2, \tag{2.4.6}$$

gdzie $x_{best}(j)$ – rozwiązanie o najwyższej sprawności w j-tej populacji.



Rysunek 8 Względna odległość rozwiązań efektywnych od brzegu Pareto dla wybranych wartości metaparametrów ewolucji różnicowej.
Źródło: opracowanie własne.

Natomiast wariant mutacji różnicowej DE/best/2 opisuje następująca zależność:

$$\tilde{x}_l(j) = x_{best}(j) + q\left(x_{l_1}(j) - x_{l_2}(j)\right) + q\left(x_{l_3}(j) - x_{l_4}(j)\right), \quad best \neq l_1 \neq l_2 \neq l_3 \neq l_4 \tag{2.4.7}$$

Strategia DE/current to best/2 uwzględnia najlepsze rozwiązanie do obliczania wybranej różnicy dwóch wariantów, jak niżej:

$$\tilde{x}_l(j) = x_{l_1}(j) + q\left(x_{best}(j) - x_{l_2}(j)\right) + q\left(x_{l_3}(j) - x_{l_4}(j)\right), \quad l_1 \neq best \neq l_2 \neq l_3 \neq l_4 \tag{2.4.8}$$

Niektóre modyfikacje podstawowego algorytmu ewolucji różnicowej omówiono w [38, 108]. W szczególności Li et al. zastosowali archiwum najsprawniejszych rozwiązań, z którego losowo wyznacza się dwa warianty do obliczenia różnicy wektorów [88, 89]. W [23] rozważa się losowy dobór wartości parametrów *q* i *Cp* oraz ich modyfikacje podczas obliczeń. W [24] uwzględnia się zmienną liczbę rozwiązań w populacji, a w [137] – zmieniające się strategie mutacji. Leon et al. porównali strategie mutacji ze względu na jakość rozwiązań, rekomendując strategię DE/current to best/1 [87].

Ponieważ wielokryterialny problem optymalizacji migracji maszyn wirtualnych z agentami pedagogicznymi w chmurze obliczeniowej jest zagadnieniem optymalizacji kombinatorycznej, to przed przystąpieniem do eksperymentów zweryfikowano działanie algorytmu ewolucji różnicowej za pomocą wybranej instancji problemu komiwojażera (ang. *Traveling Salesman Problem* - TSP) [161]. Przyjęto, że maksymalna liczba populacji J_{max}=1500; *Cp* = 0,9 oraz *q* = 0,5. Tabela 1 przedstawia uzyskane wyniki, a rysunek 9 – minimalizację długości trasy dla instancji z 14 miastami. Rekomendowanymi strategiami mutacji są dla tej instancji DE/best/1 i DE/rand/1 (tab. 1).

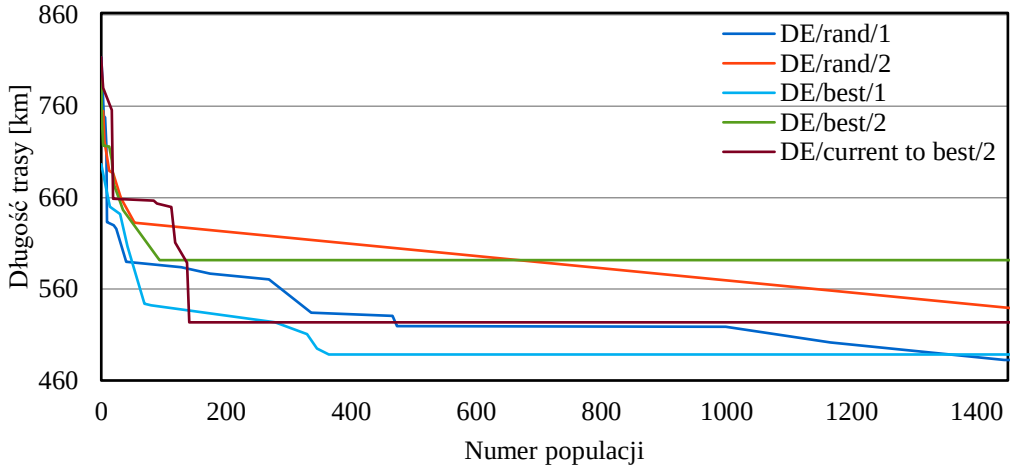
Tabela 1 Długości tras wyznaczone za pomocą algorytmu ewolucji różnicowej dla instancji problemu TSP.

Strategia mutacji różnicowej	Długość trasy	Numer populacji
DE/rand/1	482,47	1444
DE/best/1	488,44	364
DE/current to best/2	523,61	141
DE/rand/2	523,95	1467
DE/best/2	591,66	93

Źródło: opracowanie własne.

Ze względu na optymalizację wielokryterialną warto nawiązać do jednego z pierwszych algorytmów ewolucyjnych tej klasy, którym był Vector Evaluated Evolutionary Algorithm (VEEA) z wagami przydzielonymi kryteriom [144, 173]. Goldberg

zaproponował metodę wyznaczania rang w celu preferowania rozwiązań efektywnych w populacji [57]. Ponadto Fonseca i Fleming opracowali alternatywną wersję przydzielania rang, którymi były liczby dominujących osobników w algorytmie MOGA (ang. *Multi-Objective Genetic Algorithm*) [52].



Rysunek 9 Minimalizacja długości trasy w instancji problemu TSP za pomocą algorytmu ewolucji różnicowej.
Źródło: opracowanie własne.

Rysunek 10 przedstawia przykładowe rangi za pomocą procedury Fonseci-Fleminga w wypadku minimalizacji obciążanie CPU newralgicznego hosta oraz obciążenia transmisją danych newralgicznego węzła. Interesujący wielokryterialny algorytm ewolucji różnicowej opracowali Abbas i Sarker [1], w którym do mutacji różnicowej wykorzystuje się rozwiązania niezdominowane. Również Jamali et al. zaproponowali, aby podczas mutacji różnicowej losować reprezentanta rozwiązań efektywnych [70]. Warto zauważyć, że strategię mutacji DE/best/1 można przekształcić do strategii mutacji wielokryterialnej DE/Pareto/1, jak niżej:

$$\tilde{x}_l(j) = x^{effective}(j) + q\left(x_{l_1}(j) - x_{l_2}(j)\right), \quad l_1 \neq l_2, \tag{2.4.9}$$

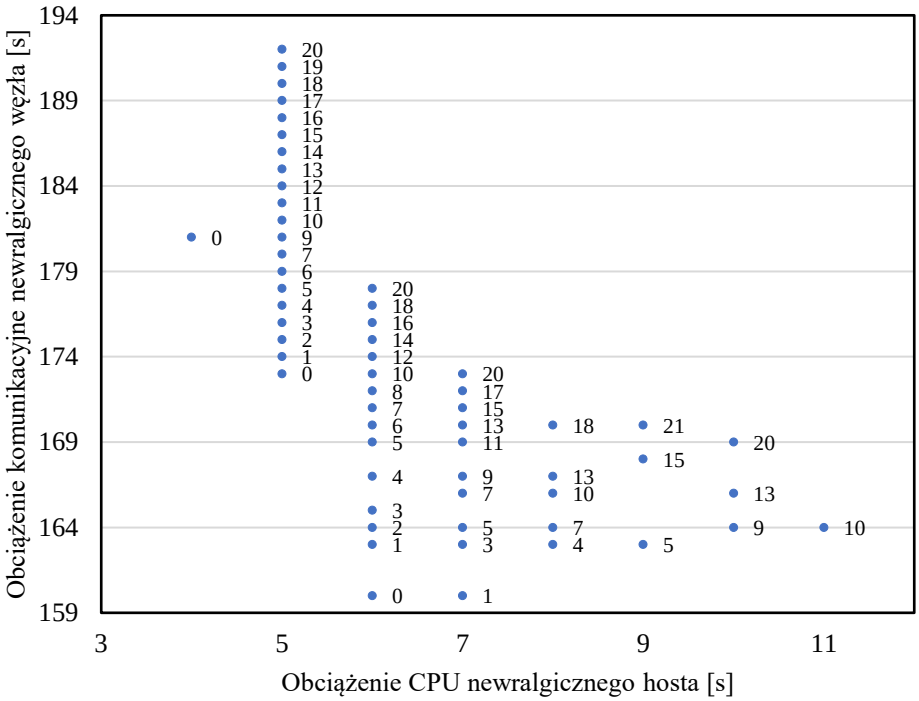
gdzie $x^{effective}(j)$ – wylosowane rozwiązanie niezdominowane z j-tej populacji.

Natomiast z wariantu mutacji DE/best/2 otrzymujemy mutację DE/Pareto/2:

$$\tilde{x}_l(j) = x^{effective}(j) + q\left(x_{l_1}(j) - x_{l_2}(j)\right) + q\left(x_{l_3}(j) - x_{l_4}(j)\right), \quad l_1 \neq l_2 \neq l_3 \neq l_4. \tag{2.4.10}$$

Z kolei DE/current to best/2 przekształcamy do strategii DE/current to Pareto/2:

$$\tilde{x}_l(j) = x_{l_1}(j) + q\left(x^{effective}(j) - x_{l_2}(j)\right) + q\left(x_{l_3}(j) - x_{l_4}(j)\right), \quad l_1 \neq l_2 \neq l_3 \neq l_4. \tag{2.4.11}$$

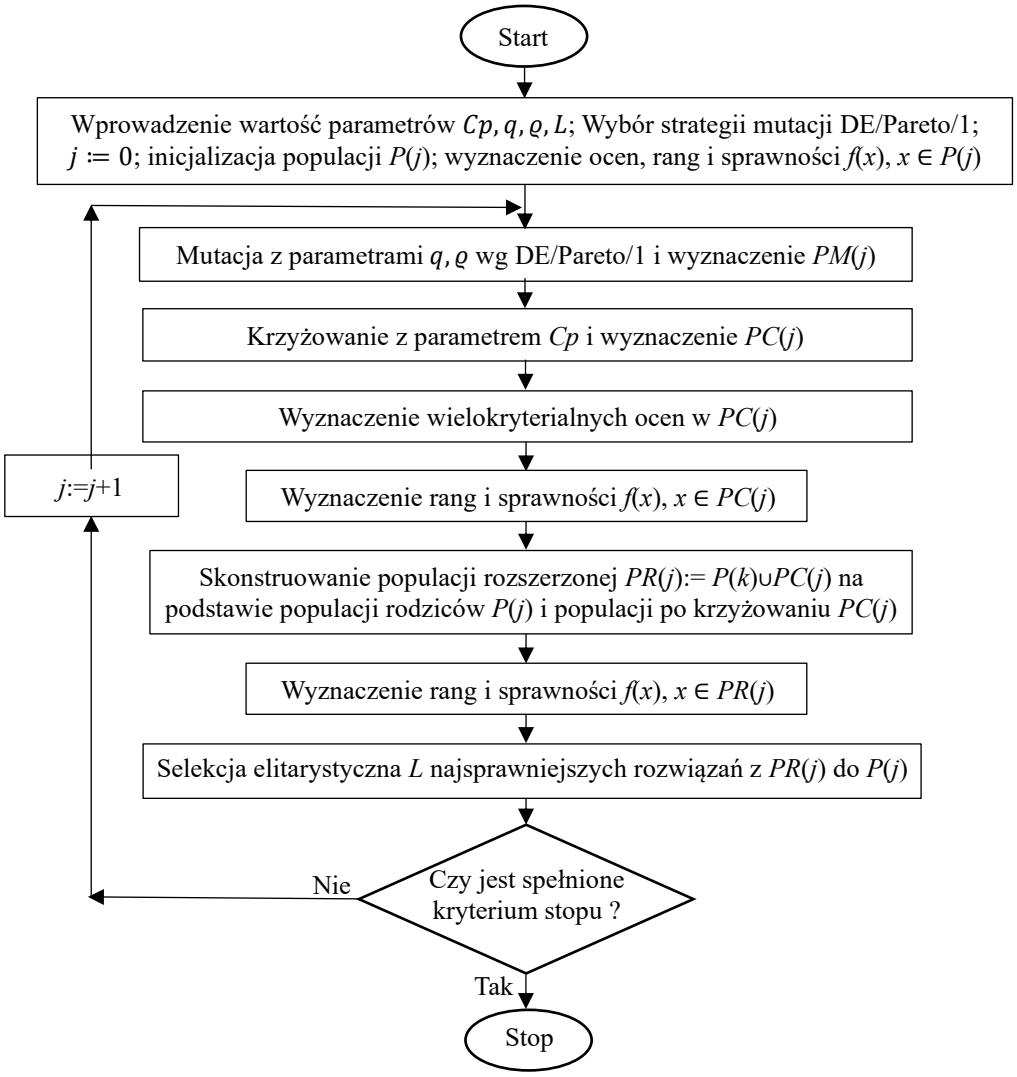


Rysunek 10 Ilustracja wyznaczania rang za pomocą procedury Fonseci-Fleminga. Źródło: opracowanie własne.

W algorytmie Abbasa-Sarkera przyjmuje się, że jeżeli rozwiązań niezdominowanych jest więcej niż wynosi liczebność populacji, to redukuje się te, które cechują się zbliżonymi ocenami (rys. 11). Ponadto Hancer zaproponował łączenie populacji rodziców z populacją po mutacji, a następnie sukcesję na podstawie sortowania rang [62]. W wypadku algorytmu Hancera nie stosuje się krzyżowania.

Warto podkreślić, że ewolucję różnicową wykorzystano również do wspomagania zdalnego nauczania. Wang et al. zaproponowali algorytm ewolucji różnicowej do przygotowania testów egzaminacyjnych. Ponieważ liczba możliwych testów rośnie wykładniczo wraz z liczbą pytań, to opracowanie testu jest zadaniem czasochłonnym. W modelu przyjęto pięć atrybutów pytania: stopień dyskryminacji, stopień trudności, czas potrzebny na odpowiedź, liczba punktów oraz typ pytania [164]. Ponadto Wang et al. zastosowali algorytm ewolucji różnicowej do szeregowania zadań w rozproszonym systemie zdalnego nauczania [163]. Celem była minimalizacja maksymalnego czasu realizacji zadań oraz maksymalizacja przepustowości systemu. Wyznaczone rozwiązania pozwalały na równoważenie obciążeń [111].

Na rysunku 11 zobrazowano wielokryterialny algorytm ewolucji różnicowej Abbasa-Sarkera [1]. Warto zauważyć, że rozwiązania w populacji rodziców cechują się innymi sprawnościami niż te same rozwiązania w populacji rozszerzonej *PR*. Z tego powodu wyznacza się ponownie sprawności rozwiązań w *PR*.



Rysunek 11 Diagram wielokryterialnego algorytmu ewolucji różnicowej Abbasa-Sarkera. Źródło: opracowanie własne na podst. [1].

2.5 Wnioski i uwagi

Algorytmy sztucznej inteligencji mogą być stosowane do wspomagania zdalnego nauczania, a modele uczenia maszynowego mogą dokonywać predykcji osiągnięcia efektów nauczania. Do przewidywania ocen studentów można wykorzystać programowanie genetyczne i sztuczne sieci neuronowe. Ponadto dydaktyczne gry komputerowe mogą być wspierane przez algorytmy neuro-ewolucyjne. Z kolei modele wytrenowane za pomocą maszyny wektorów wspierających umożliwiają dostosowanie treści nauczania do preferencji studenta. Natomiast inteligentna selekcja pytań pozwala na pełniejsze wyjaśnienie kwestii dydaktycznych.

Ważnym zastosowaniem algorytmów ewolucji różnicowej jest możliwość przygotowania testów z przedmiotu na podstawie obszernej bazy pytań. Algorytm ten również

wykorzystano do wyznaczania rozwiązań w NP-trudnym szeregowaniu zadań dla systemu e-learningowego. Interesującą dyskusję obejmującą najnowsze modele i metody szeregowanie zadań przedstawiono w [22].

W mutacji algorytmu ewolucji różnicowej wykorzystuje się współczynnik skalowania różnicy wektorów, a w krzyżowaniu − próg preferencji genów potomków. Intensywność mutacji i krzyżowania jest znacznie wyższa niż w wypadku innych algorytmów ewolucyjnych, co umożliwia rozszerzoną penetrację sąsiedztwa rozwiązań z populacji rodziców. W szczególności mutacja różnicowa wykorzystuje co najmniej dwa inne warianty z populacji do wyznaczenia poprawki dla wylosowanego rozwiązania podstawowego z populacji rodziców. Stosowane są różnorodne strategie mutacji, z których niektóre uwzględniają najlepiej przystosowane osobniki. W rozprawie zaproponowano strategie wykorzystujące niezdominowane alternatywy w wypadku zagadnień optymalizacji wielokryterialnej.

Krzyżowanie jest ważnym etapem modyfikacji rozwiązania po mutacji, gdyż współrzędne dobierane są losowo na podstawie genów rodzica i potomka po mutacji dla zadanego poziomu preferencji genów potomka. Przyjmuje się, że potomek ma zagwarantowaną minimalną liczbę genów, które zostaną zaakceptowane po mutacji. Natomiast sukcesja związana jest z wyborem między rodzicem a „dorosłym" potomkiem na podstawie wartości przystosowania.

W kontekście tematyki rozprawy ważną rolę odgrywają wielokryterialne algorytmy ewolucji różnicowej, które opierają się na procedurach stosowanych w wielokryterialnych algorytmach ewolucyjnych. W szczególności można zastosować procedurę nadawania rang Goldberga lub procedurę Fonseci-Flaminga. Ważną rolę pełni sukcesja oparta na sortowaniu rozwiązań niezdominowanych wg rang.

Warto jednak podkreślić, że nie stosowano dotychczas wielokryterialnego algorytmu ewolucji różnicowej do tak złożonego problemu, jakim jest optymalizacja zasobów pedagogicznej chmury obliczeniowej z czterema kryteriami. Ponadto istniejące algorytmy wielokryterialnej ewolucji różnicowej nie uwzględniają rozwiązań niedopuszczalnych i nie stosują funkcji zagęszczenia w wypadku rozwiązań o tej samej randze. Nie opracowano do tej pory algorytmu ewolucji różnicowej, w którym wykorzystuje się metaheurystykę tabu search jako dodatkową mutację.

3. CHMURY OBLICZENIOWE W ZDALNYM NAUCZANIU

W chmurach obliczeniowych możliwa jest migracja wielu maszyn wirtualnych między serwerami w czasie milisekund, co nie powoduje zauważalnych perturbacji w użytkowaniu aplikacji e-learningowych. Migracja jest konieczna ze względu na wyłączanie hostów ulegającym awariom, przerwy w dostawach prądu, planowane oszczędności zużycia energii, a także niezbędne konserwacje. W celu zapewnienia ciągłości nauczania możliwe zatem jest udostępnienie studentom chmury edukacyjnej wspieranej przez modele uczenia maszynowego.

Niewątpliwym wyzwaniem jest zaprojektowanie rozproszonego systemu zdalnego nauczania, który byłby platformą dla wielu polskich uczelni, a także uczelni zagranicznych, np. filii polskich uczelni na Litwie czy Ukrainie. Zaletą rozważanej chmury obliczeniowej wspierającej edukację jest możliwość efektywnego współdzielenia kosztownych zasobów dydaktycznych. Natomiast jednym z kluczowych problemów jest optymalizacja rozdziału zasobów ze względu na wybrane kryteria, do których zalicza się: obciążenie procesorów newralgicznego hosta, obciążenie transmisją danych krytycznego węzła, łączną moc zużycia energii elektrycznej, a także łączne koszty zakupu lub dzierżawy komputerów.

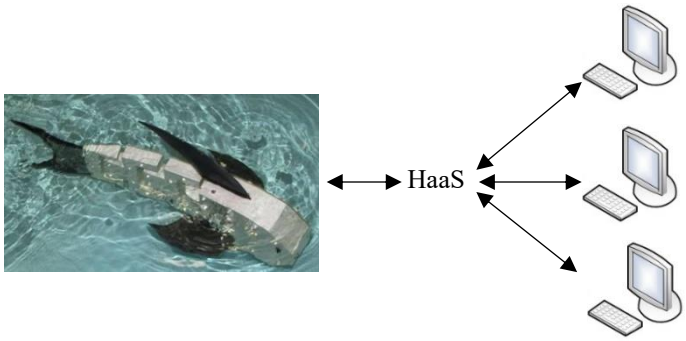
3.1 Chmura obliczeniowa

Zazwyczaj podczas prowadzenia zajęć dydaktycznych z wykorzystaniem komputerów występuje deficyt pamięci RAM lub pamięci zewnętrznej (dyskowej lub półprzewodnikowej). Ponadto studenci często narzekają na fakt, że moc obliczeniowa komputerów lub przepustowość transmisji danych są niedostateczne. Dylemat można rozwiązać, stosując maszyny wirtualne z zasobami dopasowanymi do wymogów zadań dydaktycznych [49]. Ważną cechą tego podejścia jest fakt, że zespoły inteligentnych agentów umieszczone w wirtualnych maszynach mogą przemieszczać się między hostami, aby zmniejszyć opóźnienia w transmisji danych, zwiększyć dostępną moc obliczeniową lub zmniejszyć zużycie energii elektrycznej.

Warto podkreślić, że chmura obliczeniowa umożliwia efektywne wspieranie smart edukacji, w której wykładowca ma dodatkowe opcje efektywnego nadzoru nad realizacją zadań przez studentów. Może również reagować na niepowodzenia lub opóźnienia podopiecznych w zdobywaniu wiedzy [48, 9]. Cechą charakterystyczną modelu usługi IaaS (ang. *Infrastructure as a Service*) jest zapewnienie dostępu użytkownikom

w chmurze obliczeniowej do wystarczającej ilości zasobów. Należy jednak wcześniej oszacować, ile zasobów będzie koniecznych. W smart edukacji rozwijane są usługi typu VlaaS (ang. *Virtual Laboratories as a Service*), za pomocą których udostępnia się wirtualne laboratoria [48].

W smart edukacji stosuje się także inne modele chmury obliczeniowej: SaaS (ang. *Software as a Service*) oraz HaaS (ang. *Hardware as a Service*) [158]. W modelu HaaS zdalnie udostępniany jest specjalistyczny sprzęt, za pomocą którego uczący się może przeprowadzić eksperymenty w wirtualnym laboratorium, a także nauczyć się konfigurowania i obsługi specjalistycznego sprzętu, np. routera czy drona podwodnego (rys. 12) [11]. Natomiast za pomocą modelu SaaS udostępnia się zestaw aplikacji webowych, co zwiększa możliwości nauczania [42].



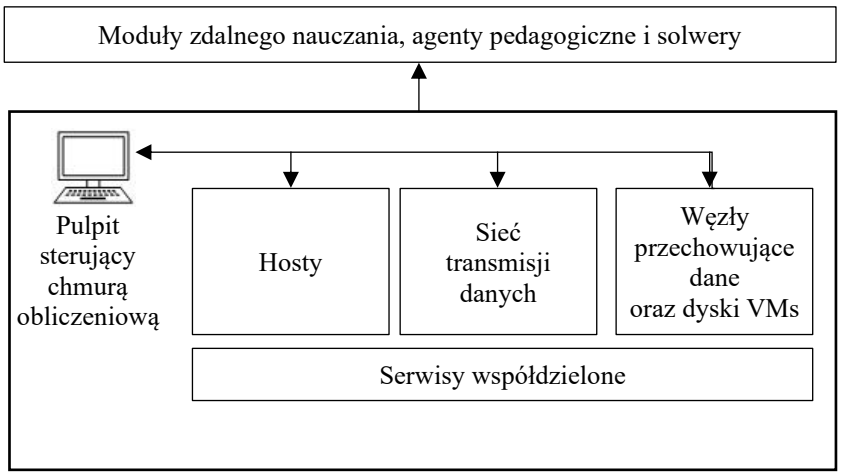
Rysunek 12 Komunikacja z podwodnym dronem za pomocą usługi Hardware as a Service.
Źródło: opracowanie własne.

W środowisku chmury obliczeniowej można przeprowadzić wykłady i zajęcia laboratoryjne [43], a także opracować ewaluację nauczania, w tym ocenę prac dydaktycznych [177]. Rekomendowaną platformą prywatnej chmury obliczeniowej w uczelni jest oprogramowanie OpenStack, które amerykańska firma Rackspace zaprojektowała w 2010 roku, a następnie udostępniła kod platformy na zasadach otwartego oprogramowania [138]. Od 2012 roku projektem zarządza fundacja OpenStack Foundation [109].

Architekturę oprogramowania OpenStack przedstawiono na rysunku 13, w tym usługi, które umożliwiają zarządzanie maszynami wirtualnymi, projektowanie wirtualnych sieci komputerowych oraz specyfikację serwisów przechowywania danych. Do sterowania usługami służą: dedykowana aplikacja webowa, interfejs programistyczny API (ang. *application programming interface*) typu REST (ang. *representational state transfer*), a także linia poleceń [109].

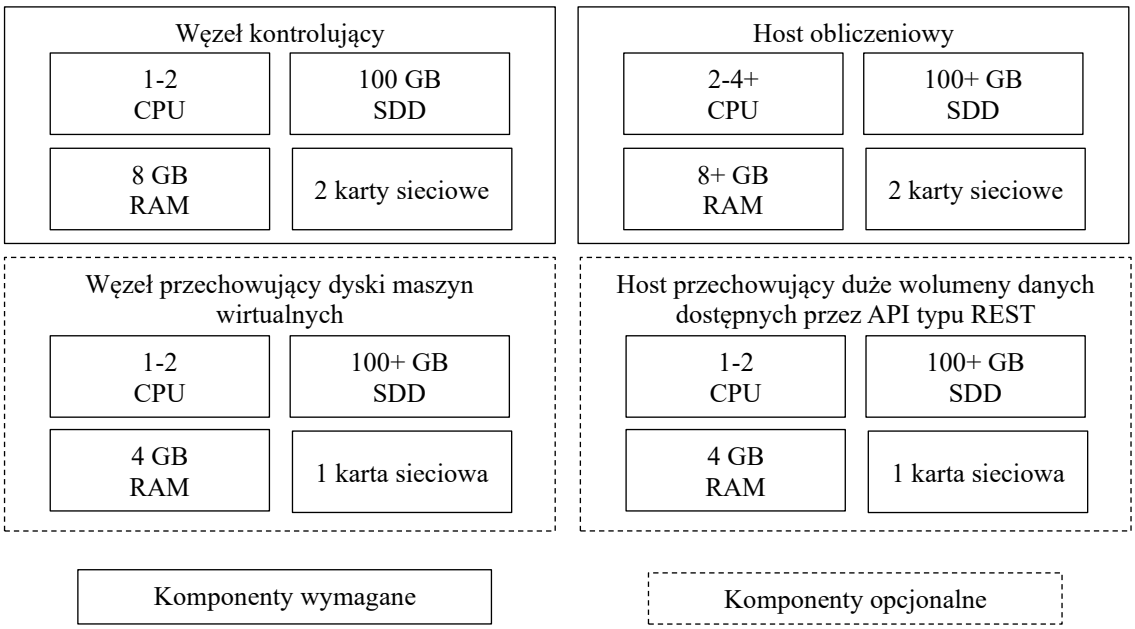
Natomiast przykładowe węzły chmury opartej na platformie OpenStack zilustrowano na rysunku 14. Kontroler zarządza zasobami chmury, a hosty umożliwiają pracę maszyn

wirtualnych. Ponadto wyróżnić można węzły do przechowywania zawartości dysków maszyn wirtualnych. Zarządzanie węzłami przechowującymi dane odbywa się za pomocą wybranego API. Przy doborze hostów dla maszyn wirtualnych kluczową decyzją jest określenie typów procesorów. W hostach stosowane są procesory, które posiadają zazwyczaj do 64 rdzeni [109, 20].



Rysunek 13 Agenty pedagogiczne na platformie OpenStack.
Źródło: opracowanie własne na podstawie [109].

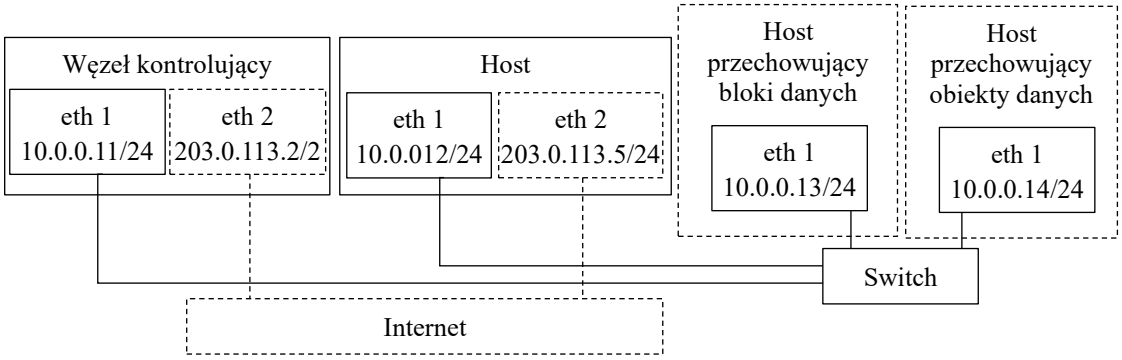
Można przydzielić większą liczbę procesorów oraz większą wielkość pamięci do maszyny wirtualnej niż pozwalają na to zasoby w wybranym węźle obliczeniowym. W przypadku liczby rdzeni procesorów współczynnik rozszerzenia wynosi 16:1, a w wypadku wielkości pamięci RAM − 1,5:1.



Rysunek 14 Rodzaje hostów i węzły pomocnicze w środowisku obliczeniowym OpenStack.
Źródło: opracowanie własne na podstawie [109].

W rezultacie można przydzielić zamiast jednego fizycznego rdzenia procesora w hoście aż 16 rdzeni w maszynie wirtualnej. Jeżeli host dysponuje 32 GB RAM, to maszyny wirtualne w tym węźle mogą wykorzystać 48 GB wirtualnej pamięci RAM. W platformie nie można jednak przekroczyć łącznej wielkości dostępnych zasobów w chmurze [83].

Hosty pracują pod systemami operacyjnymi klasy Linux. Natomiast maszyny wirtualne mogą emulować pracę różnorodnych systemów operacyjnych, np. Linux, Windows, MacOS lub Android. Zazwyczaj konstruuje się co najmniej dwie podsieci transmisji danych, przy czym pierwsza zapewnia komunikację między usługami, a druga – łączy chmurę z jej otoczeniem (rys. 15). Przykładowo, sieć o puli adresów 10.0.0.0/24 zapewnia przesyłanie danych w ramach chmury. Natomiast sieć o adresach 203.0.113.0/24 służy do połączenia chmury z Internetem [109]. W związku z migracją maszyn wirtualnych należy zagwarantować odpowiednią przepustowość sieci.

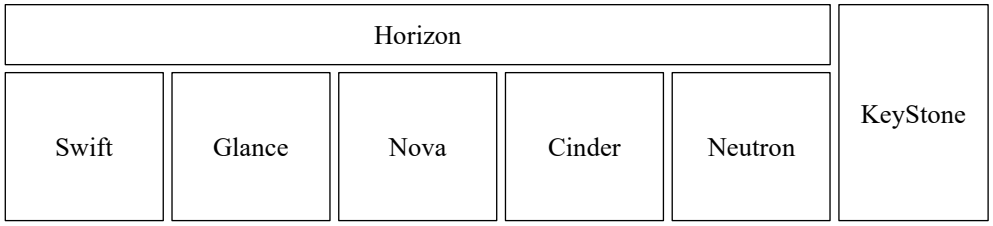


Rysunek 15 Adresowanie komponentów chmury opartej na platformie OpenStack.
Źródło: opracowanie własne na podstawie [109].

Kluczowe usługi oraz aplikacje platformy OpenStack przedstawiono na rysunku 16. Do komunikacji między aplikacjami w języku Python można wykorzystać API platformy lub system pośredniczący w wymianie komunikatów. Natomiast do kluczowych usług platformy zalicza się: KeyStone, Glance, Nova i Neutron. Usługa KeyStone zarządza katalogiem usług, zapewnia dostęp do chmury, uwzględniając uwierzytelnianie i autoryzację użytkowników i aplikacji [109].

Glance jest usługą w modelu IaaS pozwalającą na przechowywanie obrazów systemów operacyjnych, a także obrazów dysku lub serwera. Obrazy systemów operacyjnych składowane są w pamięci podręcznej hostów, co pozwala na redukcję czasu uruchomienia maszyny wirtualnej [109]. Zadaniem usługi Nova jest zarządzanie maszynami wirtualnymi. Natomiast usługa Neutron zarządza infrastrukturą sieci komputerowej chmury, w tym zaporą ogniową i wirtualnymi sieciami prywatnymi (ang. *Virtual Private Networks*, VPNs) oraz równoważy obciążenia.

Realizację wybranych topologii sieci komputerowych zapewniają routery i wirtualizacja sieci. Wyróżnia się dwa rodzaje sieci: zewnętrzną oraz wewnętrzną. W sieci zewnętrznej wykorzystywane są interfejsy fizyczne OpenStack, za pomocą których możliwe jest łącznie się z sieciami VPN. Interfejsy te mogą również służyć do łączenia sieci VPN z Internetem. Do łączenia sieci wewnętrznych wykorzystuje się routery, które za pomocą bramy mają dostęp do sieci zewnętrznej. Reguły bezpieczeństwa w chmurze służą do blokowania zagrożonych portów, ustalania zakresów portów lub blokowania rodzajów transmisji danych dla wybranej sieci [109].



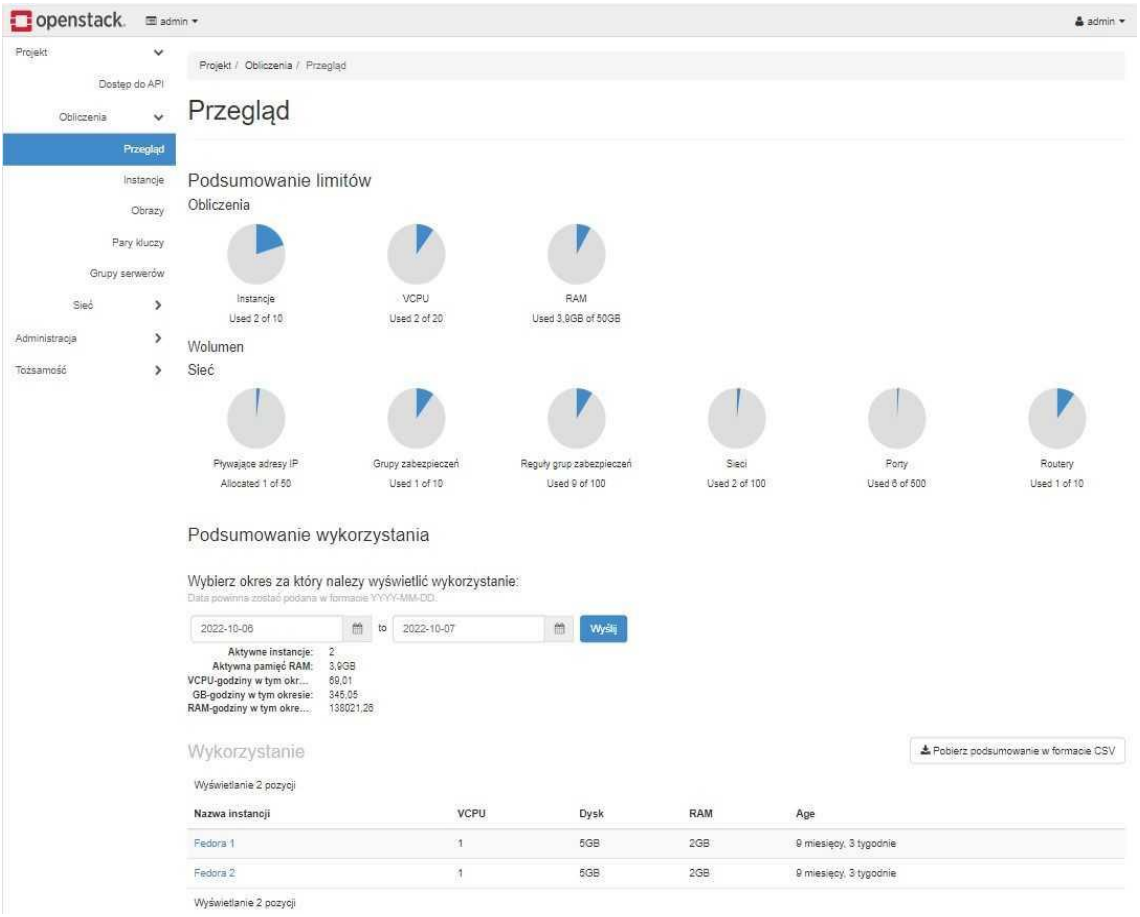
Rysunek 16 Diagram zarządzania usługami przez KeyStone.
Źródło: opracowanie własne na podstawie [109].

Opcjonalna usługa Neutron Load Balancing as a Services (Neutron LBaaS) równoważy obciążenia. Pierwsza wersja tej usługi stosowała agenta, który zarządza konfiguracją serwera pośredniczącego HAProxy [109]. Drugą rekomendowaną wersją jest Octavia, która za pomocą interfejsu programistycznego API wykorzystuje serwer HAProxy na maszynach wirtualnych [63]. Serwer analizuje ruch żądań, które napływają do hostów i równoważy ich obciążenia za pomocą wybranego algorytmu. Administrator przypisuje wirtualnej maszynie wybraną liczbę μ jednocześnie obsługiwanych żądań z przedziału od jeden do dwieście pięćdziesiąt sześć. Nie każdy algorytm równoważący obciążenie wykorzystuje wartości μ, np. algorytm karuzelowy (ang. *round robin*) reaguje dynamicznie na zmianę obciążeń serwerów.

Usługa Cinder zarządza przestrzenią dyskową maszyn wirtualnych, co umożliwia migrację maszyny z uszkodzonego hosta na sprawny host. Umożliwia również zmiany zapotrzebowania na zasoby, w tym na wielkość pamięci RAM, wielkość pamięci dyskowej, czy liczbę rdzeni wirtualnego procesora. Warto podkreślić, że nie ma możliwości współdzielenia przestrzeni dyskowej między maszynami wirtualnymi.

Usługa Swift zarządza dużymi wolumenami danych za pośrednictwem API HTTP typu REST. Natomiast moduł Horizon udostępnia interfejs webowy (rys. 17) do zarządzania usługami chmury obliczeniowej, wykorzystując serwer HTTP Apache [109]. W węźle kontrolera komponent Chrony synchronizuje usługi oraz bazę danych typu

MariaDB, MySQL lub PostgreSQL. Istotny jest również mechanizm kolejkowania wiadomości oraz mechanizm buforowania pamięci podręcznej. Natomiast wygenerowane przez Keystone klucze służą do bezpiecznej komunikacji między usługami. W przypadku pozostałych węzłów konieczna jest również instalacja usługi Chrony [109].



Rysunek 17 Szacowanie wykorzystania zasobów za pomocą usługi Horizon.
Źródło: opracowanie własne.

Wszystkie usługi można zainstalować na jednym komputerze dla niewielkich prywatnych chmur dydaktycznych. Warto również wspomnieć o projekcie RDO, który obejmuje OpenStack na pojedynczym komputerze dla systemów CentOS Stream i Red Hat Enterprise Linux [134].

3.2 Koncepcja agentowego modelu zdalnego nauczania

Rozszerzenie chmury edukacyjnej możliwe jest nie tylko w odniesieniu do jednego wydziału uczelni, ale również może objąć zasoby kilku uczelni. Każdej z grup interesariuszy można przydzielić zespoły agentów, których zadaniem będzie wspomaganie realizacji zadań dydaktycznych. Pierwszy zespół wspomaga studentów w zakresie opanowania wymaganego materiału, odciążając lub uzupełniając wykładowców [5].

Następna grupa agentów pedagogicznych gromadzi informacje odnośnie do realizacji zadań przez studentów, a następnie rekomenduje skuteczne działania studentom [181]. Kolejny zespół dostosowuje zawartości wykładów do używanych urządzeń [56]. Następna grupa ma za zadanie określić poziom skupienia studentów na zajęciach. Agent może również ewaluować poziom wiedzy studenta i wskazać braki, nad którymi należy jeszcze popracować. Kolejny zespół agentów może przewidywać oceny z wybranego przedmiotu na podstawie interakcji studentów oraz wyników testów cząstkowych. Ważną grupę stanowią agenty, które pozwalają na klasyfikację preferencji studenta odnośnie do sposobu przyswajania wiedzy, co zapewnia adekwatne prezentowanie wiedzy na zajęciach.

Wykładowcy powinni mieć możliwość śledzenia postępów studentów, co przy dużej liczbie studentów jest niezwykle trudne. W tym wypadku agenty określają, jaka część materiału była dla studenta najmniej zrozumiałą, analizując emocjonalne zaangażowanie oraz czynności wykonywane przez niego [12]. Przygotowanie testów to kolejne zadanie, które może być realizowane przez agenty [12, 30].

Ważnym elementem zaprojektowanego demonstratora chmury edukacyjnej jest system zdalnego nauczania Moodle [132], który umożliwia zarządzanie zasobami dydaktycznymi, w tym wirtualnymi salami i kursami online [3]. System w wersji 4.04 jest instalowany w rekomendowanym środowisku serwerowym PHP w wersji 8.0 [103]. Tabela 2 przedstawia oprogramowanie bazodanowe wspierające system Moodle.

W systemie możliwe jest efektywne zarządzanie użytkownikami i ich rolami, a także istnieje możliwość projektowania stron przedmiotów. Dostępnych jest dziewięć ról, przy czym możliwe jest definiowanie również innych ról (tab. 3). Ponadto wyróżnia się czternaście podstawowych rodzajów aktywności studenta [103].

Tabela 2 Wersje baz danych wspierane przez system Moodle w wersji 3.11 [103].

Baza danych	Wersja minimalna
Postgres	10
MySQL	5.7
MariaDB	10.2.29
Microsoft SQL Server	2017
Oracle	11.2

Ewaluację zadań wspiera przesyłanie rozwiązanych zadań w formie tekstu lub plików. Nauczyciel może wskazać dodatkowych wykładowców do oceny zadań. Przydatną opcją

jest wyznaczenie ostatecznego terminu na przesłanie zadania. Ocenę zadania można wyrazić, pisząc komentarz, przesyłając plik, wpisując punkty lub ocenę [103]. Ponadto w ramach przedmiotu czat umożliwia dyskusję tekstową zainteresowanych osób. Natomiast ankieta pozwala na głosowanie i weryfikowanie wiedzy. Wyniki aktywności są prezentowane dla wykładowcy w formie raportu [103].

Tabela 3 Standardowe role dostępne w systemie Moodle [103].

Poziom uprawnień	Opis
Administrator	Posiada wszystkie uprawnienia do zarządzanie systemem
Menedżer systemu	Rola posiada domyślnie wszystkie uprawnienia administratora, przy czym administrator może edytować uprawnienia do zarządzania systemem
Projektant kursów	Umożliwia zakładanie kursów
Nauczyciel	Pozwala na zarządzanie kursem, umożliwia dodawanie zawartości do kursu
Nauczyciel bez prawa edycji	Pozwala na wystawie ocen studentom, uniemożliwia edycję i dodawanie zawartości kursu
Student	Pozwala na dostęp do kursów oraz uczestniczenie w nich
Gość	Pozwala na przeglądanie kursów bez możliwości uczestniczenia w nich
Uwierzytelniony użytkownik kursu strony głównej	Umożliwia zakładanie i uczestniczenie w zajęciach kursu strony głównej

Baza danych pozwala wykładowcom i studentom na wprowadzanie, wyświetlanie i przeszukiwanie wpisów. Formatem wpisów może być obraz, plik, adres URL, liczba lub tekst. Natomiast serwis do zarządzania opiniami umożliwia przeprowadzenie ankiet, np. w celu oceny kursu lub nauczyciela. Forum pozwala na wymianę poglądów w formie tekstowej lub multimedialnej. Natomiast słownik umożliwia opracowanie zbioru definicji, a linki do pojęć ze słownika mogą być umieszczone w tekście materiałów edukacyjnych. Uczestniczyć w opracowaniu słownika mogą studenci [103].

Niektóre narzędzia pozwalają na interakcje w standardzie Learning Tools Interoperability między platformą Moodle a platformą MS Teams. Natomiast quiz pozwala wykładowcy na przeprowadzenie testu cechującego się dużą liczbą pytań, które mogą być wielokrotnego wyboru, prawda-fałsz, krótka tekstowa odpowiedź, wybór rysunku lub tekstu. Pytania mogą być przechowywane w banku zapytań i ponownie wykorzystane w innym quizie [103].

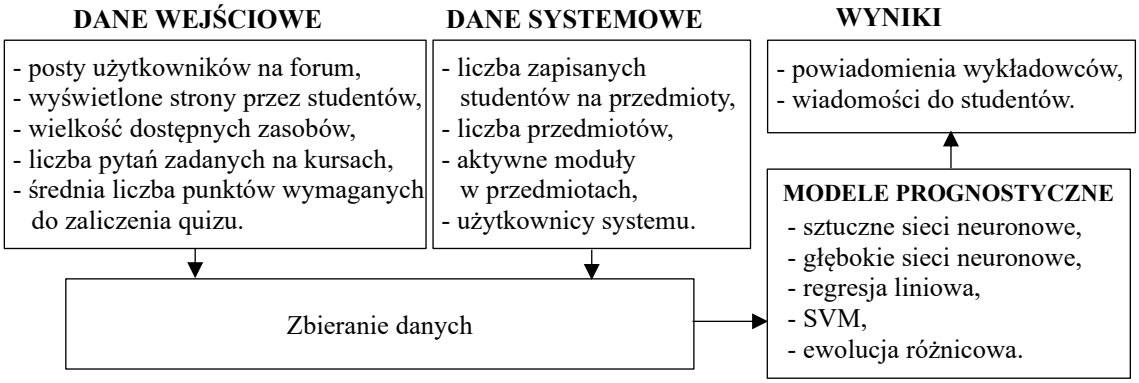
Moodle umożliwia dodawanie kursów w formacie SCORM (ang. *Sharable Content Object Reference Model*) lub AICC (ang. *Aviation Industry Computer-Based Training Committee)* [103]. Ponadto dostępna jest ankieta o konstruktywistycznym środowisku uczenia online (ang. *Constructivist On-Line Learning Environment Survey, COLLES*)

i ankieta o wybranych aspektach rozumowania (ang. *Attitudes to Thinking and Learning Survey, ATTLS)*, które są przydatne w stymulowaniu studentów do nauki. Wykładowcy mogą używać ankiet do zbierania danych w celu dostosowania zawartości kursu do oczekiwań studentów [103].

Serwis wiki jest zbiorem stron internetowych zaprojektowanych przez wykładowców i studentów. Możliwa jest w nim praca wielu autorów nad jednym dokumentem, co można zastosować w pracy zespołowej. Wirtualny warsztat umożliwia studentom ocenić swoją pracę, a następnie ocenić prace innych studentów. Wykładowca może zdecydować o widoczności tożsamość autora pracy podczas oceniania. Może również przekazać wytyczne o kryteriach oceny zrealizowanych zadań [103].

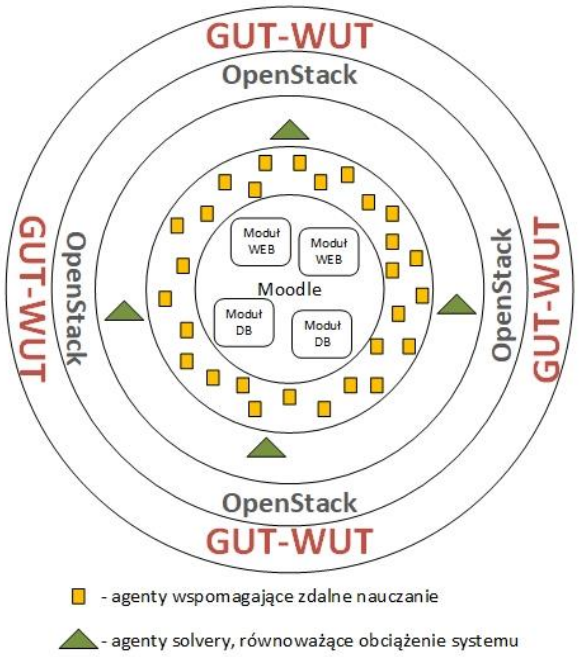
W modelu analizy postępów studenta (ang. *analitics*) dokonuje się predykcji osiągnięć studenta. Analiza dotyczy studentów zagrożonych nieukończeniem przedmiotu oraz studentów nieaktywnych. Dokonuje się predykcji w oparciu o zaangażowanie studenta, pozwalające przewidzieć, kto nie ukończy przedmiotu. Pytania o zaangażowanie studenta kierowane są do pozostałych osób. Istotne jest, czy student dzieli się wiedzą z innymi oraz czy prezentuje pomysły związane z uczeniem się. Pytania dotyczą tego, czy student przedstawia swoje poglądy odnośnie do zdobytej wiedzy oraz czy chętnie współpracuje z pozostałymi. Ważne są również pytania o jego aktywność na zajęciach: czy inicjuje dyskusję i zadaje pytania. Uwzględnia się również aktywność w poprzednich kursach.

Ograniczeniem omawianego rozwiązania jest konieczność wytrenowania modelu na podstawie ukończonych przedmiotów, co wymagana przechowywania danych w systemie. Nie wszystkie serwisy systemu Moodle to obecnie realizują. W konsekwencji, modele mogą uczyć się na niepełnej liczbie aktywności studentów, co prowadzi do niedokładności w prognozach. Warto podkreślić, że za pomocą modelu regresji logistycznej prognozuje się przedmioty, którymi nie będą zainteresowani studenci (rys. 18).



Rysunek 18 Akwizycja danych i prognoza wyników w systemie zdalnego nauczania.
Źródło: opracowanie własne.

Demonstrator chmury zaimplementowano w środowisku GUT-WUT opartym o platformę OpenStack. W celu optymalizacji zasobów zastosowano agenty-solwery (rys. 19).



Rysunek 19 Wybrane elementy demonstratora chmury edukacyjnej.
Źródło: opracowanie własne.

Moodle jest aplikacją webową, które wykorzystuje serwer oraz bazę danych. Pozwala to podzielić aplikację na moduł WEB zawierający serwer wykonujący kod aplikacji oraz moduł DB zawierający bazę danych (rys. 19). Moduły mogą działać na różnych maszynach wirtualnych. Możliwe jest skalowanie w poziomie, czyli dodawanie maszyn wirtualnych z modułami WEB i DB do obsługi większego obciążenia dydaktycznego.

3.3 Metody optymalizacji zasobów chmur obliczeniowych

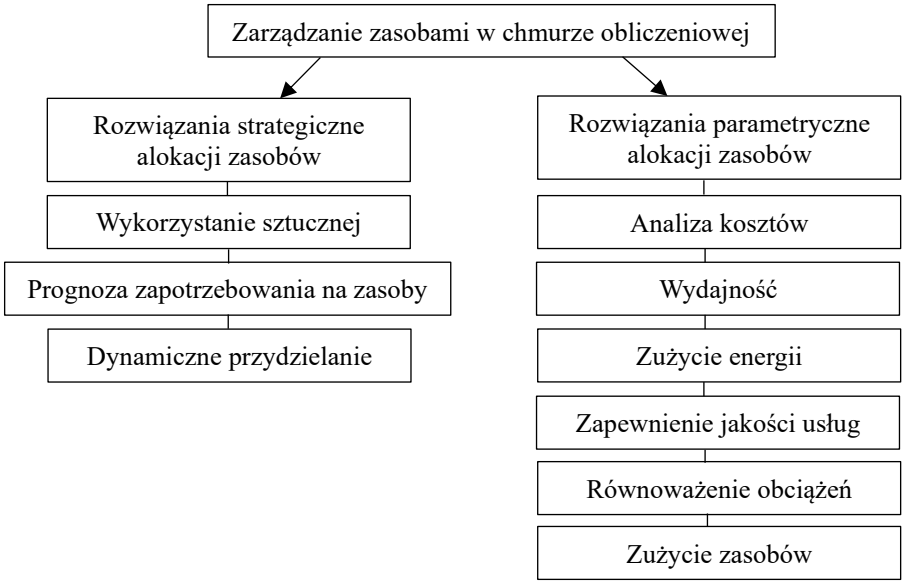
Chmury obliczeniowe ze względu na sposób udostępnienia zasobów mogą być publiczne lub prywatne. W chmurze publicznej przechowuje się i przetwarza dane w centrach danych za pomocą wydajnych hostów. Dostawca usług odpłatnie udostępnia zasoby zgodnie z regułą *pay-per-use,* dążąc zazwyczaj do maksymalizacji zysków. Natomiast usługobiorca preferuje minimalizację opłat za wykorzystanie zasobów z zachowaniem odpowiedniej jakości usług. W chmurze prywatnej usługi udostępnia się we własnej infrastrukturze organizacji, co preferowane jest przez uczelnie w Polsce ze względu na wrażliwe dane, których utrata może zaszkodzić procesowi dydaktyczno-badawczemu.

Zastosowanie modeli i metod sztucznej inteligencji do wspierania usług dydaktycznych w chmurze uczelni wymaga optymalizacji zasobów. W szczególności dążenie do

redukcji kosztu, zużycia energii i czasu reakcji systemu zmusza do implementacji rozwiązań optymalnych [40, 41]. Ważną kwestią jest również dostępność systemu [45].

W chmurze obliczeniowej optymalizacja zasobów obejmuje zarządzanie wirtualnymi maszynami, mocą obliczeniową hostów, przepustowością kanałów transmisji danych, wielkością pamięci dysków i RAM, a także alokacją zbioru aplikacji i usług [73]. Szczególne istotne jest uwzględnienie wymagań maszyn wirtualnych, w których działają zespoły agentów pedagogicznych, moduły systemów zdalnego nauczania oraz solwery [93, 146, 184]. Systemy zdalnego nauczania mogą ze sobą współpracować, co dotyczy Moodle, MS Teams, USOSa i innych. Ponadto urządzenia mobilne współdzielą obciążenie z serwerami chmury obliczeniowej, co „przesuwa" przetwarzanie danych na „obrzeża" chmury w celu skrócenia opóźnień [47, 96].

Metody alokacji zasobów w chmurze obliczeniowej można podzielić na metody wyznaczające rozwiązania strategiczne oraz metody wyznaczające rozwiązania oparte o parametryczną alokację zasobów (rys. 20) [94]. Ponadto sztuczna inteligencja może być wykorzystana do prognozowania zapotrzebowania na zasoby w trakcie wykonywania aplikacji [50].



Rysunek 20 Podział metod zarządzania zasobami w chmurze obliczeniowej [94].

W szczególności rozważane są w literaturze przedmiotu modele uwzględniające alerty związane z rozpoczęciem dzierżawy zasobów [112]. Wówczas do alokacji maszyn wirtualnych stosuje się sztuczne sieci neuronowe, których wagi wyznaczono za pomocą algorytmu genetycznego. Na podstawie zarejestrowanego obciążenia hosta przewiduje się zapotrzebowanie na zasoby w celu alokacji maszyn wirtualnych do hostów tak, aby

spełnić prognozowane zapotrzebowanie [139]. W modelu Lianga-Rui-Xu zastosowano algorytm mrówkowy, za pomocą którego prognozuje się zapotrzebowanie transmisji danych na przepustowość łączy oraz jakość sieci, w tym czas jej reagowania [90].

Ważną kwestią w infrastrukturach chmur obliczeniowych jest redukcja zużycia energii, co pozwala na znaczące oszczędności. Wang et al. zaproponowali metodę dynamicznej alokacji maszyn wirtualnych do hostów w celu minimalizacji zużycia energii [167] oraz zapewnienia należytej jakości usług [72]. Metoda dynamicznej alokacji oraz prognozowanie intensywności zapytań może poprawić wydajność pracy maszyn wirtualnych. Natomiast za pomocą ewolucyjnej sieci neuronowej prognozowano liczbę żądań na serwerze webowym NASA [8]. Ponadto za pomocą algorytmu mrówkowego szacowano dostępność zasobów w hostach [65].

Przewidywanie zapotrzebowania odnośnie do wielkości zasobów może poprawić jakość pracy maszyn wirtualnych, w tym obciążenie procesorów w hostach [68]. Do treningu modelu zastosowano zbiór danych specyfikujących przestoje hostów. W rezultacie model przydziela zgłoszenia do najmniej obciążonych hostów, co skutkuje zmniejszeniem zużycia energii elektrycznej [162].

Redukcja kosztów usługobiorcy jest również kluczową kwestią [172]. Mohana et al. do optymalizacji kosztów oraz czasu realizacji zadań zaproponowali algorytm optymalizacji rojem cząstek PSO (ang. *Particle Swarm Optimization*) [101]. Efektywność alokacji zasobów pomaga również zwiększyć przepustowość systemu oraz skrócić czas przydzielenia zasobów [178].

Zmiana wymagań użytkownika jest trudna do realizacji w czasie rzeczywistym. Yang et al. zastosowali algorytm hybrydowy, łącząc algorytm kolonii mrówek z algorytmem optymalizacji rojem cząstek [175]. Mechanizmy alokacji zasobów sprawdzają się w wypadku redukcji wytwarzanego ciepła oraz zużycia energii elektrycznej w centrach danych [26]. Szczególnie zużycie energii w wypadku pracy nieobciążonego hosta jest bardzo niepożądane [19]. Dashti et al. zastosowali zmodyfikowany algorytm PSO do migracji maszyn wirtualnych z bardziej obciążonych węzłów na mniej obciążone w centrach danych, co zredukowało zużycie energii, skróciło czas odpowiedzi oraz poprawiło jakość serwisów [34].

W modelu Kansal-Chana rozważa się alokację zasobów do zmniejszenia zużycia energii przy uwzględnieniu założonej wydajność aplikacji użytkowników. Za pomocą algorytmu roju pszczół wyznaczono optymalne rozmieszczenie maszyn wirtualnych na hostach [76]. Wymagania w zakresie jakości usług QoS obejmują wymagane poziomy:

dostępności usług, tolerancji błędu, niezawodności, przepustowości i czasu przywrócenie usług po awarii [2, 14].

Kumar et al. zastosowali uczenie maszynowe do przewidywania i alokacji zasobów w celu nowych żądań [85]. Natomiast Pillai and Rao rozważali model gry w wieloagentowym środowisku, w którym każdy z hostów pełnił rolę agenta, co pozwoliło na redukcję używanych zasobów oraz skrócenie czasu przydzielania zadań [114].

Analiza złożoności obliczeniowej różnorodnych wersji problemów równoważenia obciążeń w chmurach obliczeniowych, wykazała, że wiele z nich należy do klasy problemów NP-trudnym [15, 78 i 165]. Natomiast Sridevi et al. przedstawili klasyfikację ważniejszych algorytmów równoważących obciążenia (tab. 4) [149].

Tabela 4 Klasyfikacja algorytmów równoważących obciążenie [149].

Kryterium klasyfikacji	Typ modelu	Opis
Środowisko chmury obliczeniowej	Statyczny	Ustalona polityka propagacji zadań
	Dynamiczny	Progresywna polityka propagacji zadań
	Adaptacyjny	Hybrydowa polityka propagacji zadań
Sposób rozproszenia zadań	Scentralizowany	Scentralizowane zarządzanie zadaniami
	Rozproszony	Niezależna realizacja zadań
	Hierarchiczny	Globalny i lokalny poziom równoważenia obciążeń
	Częściowo rozproszony	Wybrane zadania są zarządzane centralnie, a pozostałe są realizowane niezależnie
Zależność między zadaniami	Zależny	Szeregowanie zadań
	Niezależny	Wykonywanie zadań zależnych od dostępności danych
Inicjacja procesu równoważenia obciążeń	Nadawca	Przeciążone hosty inicjują proces równoważenia obciążeń
	Odbiorca	Mało obciążone hosty zgłaszają możliwość przejęcia dodatkowego obciążenia
	Symetryczny	Model hybrydowy

W modelu dynamicznym możliwa jest zmiana reguł przydzielania obciążeń podczas każdego przychodzącego żądania. Page i Naughton zastosowali algorytm genetyczny do dynamicznego równoważenia obciążeń, optymalizując czas wykonania zadań. Przyjęto, że zasoby każdego hosta są zmienne w czasie. Ponadto obciążenie jest również zmienne w czasie [110]. W modelu scentralizowanym zalecana jest architektura typu master-slave [77], a w modelu rozproszonym − architektura *peer-to-peer* [25].

W modelu dynamicznym zakłada się, że niepodzielne zadania przychodzą do kolejki, z której są pobierane i przydzielane do hostów przez agenta-planistę [110]. Każdy „bezczynny” host zgłasza gotowość do realizacji zadań.

Makasarwala et al. zaproponowali algorytm genetyczny, uwzględniając priorytety zadań podczas ich wykonywania. Zadania, które cechują się krótszym czasem wykonania mają wyższy priorytet [95]. Natomiast Tingting et al. zastosowali algorytm genetyczny, w którym populacja początkowa rozwiązań wyznaczana jest za pomocą algorytmu zachłannego [166]. Interesującym sposobem kodowania rozwiązania jest struktura drzewiasta, w której korzeń reprezentuje procedurę identyfikującą hosty, poziom drugi − hosty, a liście − maszyny wirtualne [61].

Jyoti et al. do równoważenia obciążenia zaproponowali agenty, które prognozowały zapotrzebowanie na zasoby. Zastosowany algorytm dekompozycji równoważył obciążenie, przydzielając zadania do maszyn wirtualnych na podstawie priorytetu [75]. Z kolei Yang et al. rozważali optymalizację chmury prywatnej opartej o system OpenStack. Zastosowano trzy kryteria: obciążenie procesorów, zajętość pamięci oraz zużycie energii. Powyższe wielkości mierzono dla maszyn wirtualnych oraz hostów. Zaproponowano algorytm alokacji maszyn wirtualnych, stosując regułę przydziału maszyny wirtualnej o najwyższym priorytecie do najmniej obciążonego hosta [174].

Natomiast Begam et al. rozważali zadania ograniczone terminem zakończenia przy ograniczonych zasobach [18]. Warto również wspomnieć o algorytmie ewolucyjnym do minimalizacji czasu realizacji zadań i minimalizacji zużycia energii [17, 186].

Reasumując, przegląd literatury przedmiotu pod kątem metod alokacji maszyn wirtualnych do hostów w chmurze obliczeniowej wykazał, że zastosowano różne metaheurystyki, za pomocą których można zarządzać maszynami wirtualnymi i dostępnymi zasobami. Nie sformułowano jednak problemu optymalizacji z czterema kryteriami, w tym minimalizacji zużycia energii, obciążenia procesorów newralgicznego hosta, transmisji danych najbardziej obciążonego węzła oraz kosztu zakupu, dzierżawy lub eksploatacji hostów. Nie zastosowano również algorytmu ewolucji różnicowej do wyznaczania reprezentacji rozwiązań Pareto-optymalnych.

3.4 Wnioski i uwagi

Hybrydowy model nauczania opierający się na chmurach obliczeniowych cechuje się istotnymi zaletami. W szczególności wirtualne laboratoria z wykorzystaniem symulatorów 3D umożliwiają precyzyjną weryfikację umiejętności studentów. Niebagatelne jest również obniżenie kosztów edukacji, a także redukcja zużycia energii elektrycznej. Redukcja kosztów obejmuje cząstkowe koszty zależne od czasu obliczeń na procesorach, opłaty za wykorzystanie pamięci RAM lub pamięci dyskowej, a także koszty transmisji

danych. Alternatywnie można wykorzystać bezpłatną platformę OpenStack oraz oprogramowanie zdalnego nauczania Moodle do budowy edukacyjnej chmury obliczeniowej ma wydziale akademickim. Skonstruowany szkielet chmury edukacyjnej na wydziale można rozszerzyć na kolejne wydziały uczelni, a następnie − na inne uczelnie.

Platforma OpenStack jest napisana w języku Python i składa się z wielu usług komunikujących się ze sobą poprzez interfejs programistyczny API typu REST. Wykorzystywane jest również system pośredniczący w wymianie komunikatów. Zarządzanie chmurą realizowane jest za pomocą aplikacji webowej, wygodnych narzędzi lub linii poleceń. W celach szkolenia administratorów można wykorzystać projekt RDO, który umożliwia instalację platformę OpenStack na pojedynczym komputerze.

Środowisko OpenStack umożliwiło zaprojektowanie chmury GUT-WUT, która jest demonstratorem chmury edukacyjnej. Agenty pedagogiczne mogą rekomendować sposób realizacji zadań na podstawie wzorców aktywności studentów, którzy opanowali materiał w stopniu celującym. Ponadto materiał dydaktyczny może być dopasowany do indywidualnego stylu nauki i oczekiwań studenta. Inteligentne agenty mogą śledzić zaangażowanie studenta i weryfikować tempo przyswajania wiedzy. Możliwa jest analiza postępów uczącego się i reagowanie w wypadku zbyt wolnego tempa uczenia się. Agenty mogą również prognozować ocenę końcową na podstawie aktywności studentów. Ponadto testy mogą być przygotowywane przez agenty pedagogiczne.

4. METODA OPTYMALIZACJI ZASOBÓW CHMURY EDUKACYJNEJ

Decyzje dotyczące teleportacji maszyn wirtualnych oraz doboru adekwatnych rodzajów hostów mogą wpłynąć na zmianę wartości kryteriów cechujących inteligentną chmurę edukacyjną. Ze względu na występujące konflikty między wybranymi parami kryteriów sformułowano zagadnienie wyznaczania reprezentacji rozwiązań Pareto-optymalnych [13], spośród których można wybrać alternatywę kompromisową dla zadanego parametru *p*. Ponieważ sformułowany problem jest NP-trudny, zastosowano wybrane metaheurystyki do wyznaczania rozwiązań Pareto-optymalnych. W szczególności opracowano oryginalną wersję wielokryterialnego algorytmu ewolucji różnicowej DEMCA (ang. *Differential Evolution Method for Cloud Agents*) do wyznaczania rozwiązań sprawnych. Na podstawie przeprowadzonych eksperymentów numerycznych można wnioskować, że algorytm ewolucji różnicowej wyznacza zazwyczaj najwyższej jakości rozwiązania w porównaniu do kluczowych wielokryterialnych metaheurystyk, takich jak algorytm optymalizacji rojem cząstek MOPSO, algorytm ewolucyjny AMEA, algorytm programowania genetycznego MGP, czy algorytm poszukiwania harmonii MOHS.

4.1 Podstawowe kryteria oceny oraz ograniczenia

W modelu decyzyjnym przyjmuje się, że minimalizuje się następujące kryteria: F_1 – obciążenie procesorów newralgicznego hosta w chmurze [s], a także F_2 – obciążenie transmisją danych newralgicznego węzła w chmurze [s]. Ponadto minimalizacji podlega F_3 – łączny koszt hostów [JM – jednostka monetarna] oraz F_4 – łączna moc poboru energii elektrycznej przez hosty [wat]. Wprawdzie komputery zużywają coraz mniej energii na wykonanie zadanej liczby instrukcji, to warto podkreślić, że energooszczędność innych urządzeń, w tym urządzeń domowych poprawia się znacznie szybciej.

Najszybszy superkomputer Frontier z Oak Ridge National Laboratory w USA cechuje się wydajnością 1 102 Exaflop/s dla testu High Performance Linpack (HPC). W czerwcu 2022 roku Frontier zdystansował japońskiego Fugaku nie tylko pod względem wydajności, gdyż wydajność Fugaku to 442 PFlop/s. Frontier zużywa 1 wat energii elektrycznej do obliczenia aż 52,23 GFlops (wydajność energetyczna), co przewyższa znacząco 14,78 GFlops cechujących Fugaku. Frontier jest wyposażony w 8 730 112 rdzeni procesorów AMD EPYC 64C 2GHz, a komunikacja odbywa się za pomocą gigabitowej sieci Ethernet.

Niech wartości minimalizowanych kryteriów mają zadane górne ograniczenia, które stanowią kluczowe wymagania ilościowe w projekcie, jak niżej:

$$F_1(x) \leq F_1^{max}, \tag{4.1.1}$$

$$F_2(x) \leq F_2^{max}, \tag{4.1.2}$$

$$F_3(x) \leq F_3^{max}, \tag{4.1.3}$$

$$F_4(x) \leq F_4^{max}, \tag{4.1.4}$$

gdzie *x* reprezentuje decyzje projektowe w chmurze, które to decyzje dotyczą destynacji migrujących maszyn wirtualnych oraz doboru rodzajów hostów.

Ponadto istotne są ograniczenia na wielkości wybranych rodzajów pamięci. Niech μ_{min}^{RAM} reprezentuje wielkość wolnej pamięci RAM hosta, który cechuje się najmniejszą wolną wielkością pamięci RAM spośród wszystkich hostów w chmurze [GB], a μ_{min}^{DM} – wielkość wolnej pamięci dyskowej DM hosta, który ma najmniej tej pamięci spośród wszystkich hostów [TB]. Hosty newralgiczne ze względu na μ_{min}^{RAM} oraz μ_{min}^{DM} mogą być różne ze względu na zapotrzebowania przydzielonych maszyn wirtualnych. Hosty z największym deficytem pamięci RAM i DM powinny spełniać poniższe ograniczenia:

$$\mu_{min}^{RAM}(x) \geq 0, \tag{4.1.5}$$

$$\mu_{min}^{DM}(x) \geq 0. \tag{4.1.6}$$

Na maszynach wirtualnych w chmurze edukacyjnej rozważa się wykonywanie agentów pedagogicznych, modułów systemu Moodle oraz solwerów alokacji maszyn wirtualnych DEMCA. Maszyny wirtualne oznaczamy jako $VM_1, \ldots, VM_k, \ldots, VM_K$. Przyjmuje się, że w maszynie wirtualnej zainstalowano i uruchomiono wybrane agenty pedagogiczne, niektóre moduły Moodle oraz wyselekcjonowane solwery DEMCA. Maszyna wirtualna VM_v może przemieszczać się między hostami umieszczonymi w zadanych węzłach chmury $N_1, \ldots, N_m, \ldots, N_M$ [185, 182]. Natomiast do podstawowych decyzji projektowych w chmurze należy rozmieszczenie maszyn wirtualnych po migracji, co można zapisać za pomocą macierzy $X^{VM} = [x_{km}^{VM}]_{K \times M}$, gdzie:

$$x_{km}^{VM} = \begin{cases} 1 \text{ jeśli } VM_k \text{ przydzielono do hosta w } N_m, \\ 0 \text{ w przeciwnym razie,} \end{cases} \qquad k = \overline{1,K},\ m = \overline{1,M}.$$

Do poprawnej pracy maszyn wirtualnych niezbędne są zasoby komputerowe o wystarczającej wielkości. Niech $H_1, \ldots, H_n, \ldots, H_N$ reprezentują dostępne z punktu widzenia projektanta chmury edukacyjnej rodzaje komputerów, które mogą być usytuowane również w pozostałej części chmury obliczeniowej poza *M* węzłami. Hosty te można zakupić, wydzierżawić lub mogą być już użytkowane w chmurze edukacyjnej. Niech

zbiór $\boldsymbol{N} = \{1, \dots, n, \dots, N\}$ zawiera indeksy rodzajów hostów, które można zainstalować w chmurze. Projektowany przydział serwerów jest reprezentowany za pomocą macierzy $X^H = [x_{mn}^H]_{M\times N}$, gdzie:

$$x_{mn}^H = \begin{cases} 1 \text{ jeśli host } H_n \text{ przydzielono do węzła } N_m, \\ 0 \text{ w przeciwnym razie,} \end{cases} \qquad m = \overline{1, M}, \quad n = \overline{1, N}.$$

Decyzje projektowe w chmurze edukacyjnej mogą zatem być reprezentowane jako para macierzy $x = (X^H, X^{VM})$, które to macierze zawierają elementy binarne. Projektant chmury edukacyjnej może wymagać, aby eksploatowano zadaną liczbę ψ_n hostów *n*-tego rodzaju. Taka sytuacja występuje, jeśli pewne komputery cechują się wysoką obliczeniową wydajnością energetyczną. Wówczas projektant może zdecydować o konieczności zastosowania właśnie takich hostów. Ponadto, jeśli koszt dzierżawy wybranych komputerów jest niski, to również można preferować eksploatację tej klasy serwerów. Wybrane hosty, które należą do instytucji dydaktycznej, mogą być udostępnione bezpłatnie w projektowanej chmurze edukacyjnej.

Liczba rekomendowanych hostów nie może być większa niż liczba węzłów *M*. Niech $\boldsymbol{N}^*$ oznacza zbiór indeksów rekomendowanych rodzajów hostów. Zachodzi $\boldsymbol{N}^* \subset \boldsymbol{N}$. Rozważane wymaganie projektowe można opisać za pomocą następującej zależności:

$$\sum_{m=1}^{M} x_{mn}^H = \psi_n, \ n \in \boldsymbol{N}^* \subset \boldsymbol{N}. \tag{4.1.7}$$

Jeśli nie zgłoszono postulatu odnośnie do wymaganej liczby egzemplarzy hostów *n*-tego rodzaju, to przyjmuje się, że $\psi_n = 0$. Oznacza to, że komputery *n*-tego rodzaju mogą wystąpić w dowolnej liczbie węzłów nie większej niż *M*. Jeśli wybrany rodzaj hosta nie powinien być uwzględniony w projekcie, to zbiór indeksów $\boldsymbol{N}$ jest redukowany o indeks tego typu komputera.

Jeśli $\psi_n = I$, to komputery w chmurze powinny być wyłącznie *n*-tego rodzaju. Wówczas rozważa się optymalizację migracji maszyn wirtualnych, w której nowe lokalizacje maszyn reprezentuje poszukiwana macierz X^{VM}. Natomiast w $X^H = [x_{mn}^H]_{M\times N}$ elementami niezerowymi są zmienne $x_{mn}^H = 1$ dla $m = \overline{1, M}$ w odniesieniu do ustalonego indeksu *n*.

W specyfikacji zadanego wektora $\psi = [\psi_1, \dots, \psi_n, \dots, \psi_N]$ liczebności preferowanych rodzajów komputerów zachodzi:

$$\sum_{n=1}^{N} \psi_n \le M. \tag{4.1.8}$$

Reasumując, host *n*-tego rodzaju jest opisany za pomocą następującej dziewiątki:

$$H_n = (T_n, ram_n, dm_n, \tau_n, \beta_n, c_n, \psi_n, \chi_n, \varphi_n), \ n = \overline{1, N}, \tag{4.1.9}$$

gdzie:

T_n – zadany wektor czasów pracy *K* maszyn wirtualnych na komputerze *n*-tego rodzaju dla reprezentatywnych zestawów danych; elementami wektora są uśrednione czasy t_{kn} wykonania *k*-tej maszyny wirtualnej na hoście *n*-tego rodzaju, wyrażone w [s],

ram_n – wielkość pamięci RAM [GB] komputera *n*-tego rodzaju,

dm_n – wielkość pamięci dyskowej DM [TB] komputera *n*-tego rodzaju,

τ_n – zadana pięciowymiarowa macierz uśrednionych czasów transmisji danych $\left[\tau_{kumjl}^{n}\right]_{K\times K\times M\times M\times N}$ dla hosta *n*-tego rodzaju. Niech *m* to numer węzła, do którego przydzielono hosta *n*-tego rodzaju, a *j* to numer węzła z hostem *l*-tego rodzaju. Natomiast maszyna wirtualna *k* pracuje na hoście *n*-tego rodzaju, a maszyna *u* – na hoście *l*-tego rodzaju. Element macierzy τ_{kumjl}^{n} to średni czas transmisji danych między maszyną wirtualną nr *k* na hoście *n*-tego rodzaju w *m*-tym węźle a maszyną *u* na komputerze *l*-tego rodzaju w *j*-tym węźle. Element macierzy jest wyrażony w [s],

β_n – moc poboru energii elektrycznej przez komputer *n*-tego rodzaju [wat],

c_n – koszt zakupu hosta *n*-tego rodzaju [JM],

χ_n – intensywność awarii hosta *n*-tego rodzaju [JC^{-1}], JC – jednostka czasu,

φ_n – wydajność hosta *n*-tego rodzaju dla wybranego testu, np. Linpack.

Danymi wejściowymi do rozważanego problemu optymalizacji chmury edukacyjnej są również:

- wektor wielkości pamięci RAM hostów $ram = [ram_1, \dots, ram_n, \dots, ram_N]$ [GB],
- wektor wielkości dysków hostów $dm = [dm_1, \dots, dm_n, \dots, dm_N]$ [TB],
- wektor mocy poboru energii elektrycznej przez różne rodzaje hostów $\beta = [\beta_1, \dots, \beta_n, \dots, \beta_N]$ [wat],
- wektor kosztów zakupu hostów $c = [c_1, \dots, c_n, \dots, c_N]$ [JM].

Do pomiaru wydajności komputerów mogą być wykorzystane benchmarki, takie jak Linpack [37]. Ponieważ nie odzwierciedlają one specyfiki pracy agentów pedagogicznych, proponuje się uwzględnienie oszacowań wymaganych czasów t_{kn} pracy procesorów w maszynach wirtualnych z zadanym zespołem agentów, a także z modułami Moodle

lub solwerami w zaprojektowanej chmurze eksperymentalnej. Bez utraty ogólności rozważań przyjmuje się w dalszej części rozprawy, że wektory T_n średnich czasów pracy maszyn wirtualnych na komputerach wybranych kategorii są kolumnami dwuwymiarowej macierzy $T = [t_{kn}]_{K\times N}$. Natomiast *k*-ty wiersz macierzy T_k zawiera uśrednione czasy wykonania *k*-tej maszyny wirtualnej na poszczególnych rodzajach hostów.

Macierz uśrednionych czasów transmisji danych $\tau^{ext} = \left[\tau_{kumjnl}\right]_{K\times K\times M\times M\times N\times N}$ zawiera elementy reprezentujące łączny czas transmisji między maszyną wirtualną nr *k* a maszyną nr *u*, przy czym *m* to numer węzła z komputerem *n*-tego rodzaju, w którym pracuje maszyna nr *k*, a *j* to numer węzła z komputerem *l*-tego rodzaju, w którym wykonuje się maszyna wirtualna nr *u*. Macierz sześciowymiarowa τ powstaje z *N* pięciowymiarowych macierzy τ_n.

W praktyce macierz τ^{ext} redukuje się do macierzy dwuwymiarowej $\tau = [\tau_{ku}]_{K\times K}$, jeśli można założyć, że przepustowości podsystemów transmisji danych w hostach są jednakowe. Ponadto jednakowe powinny być przepustowości połączeń między węzłami. Wówczas *k*-ta kolumna tej macierzy τ_k specyfikuje obciążenie komunikacyjne wytworzone przez maszynę wirtualną nr *k*.

Zapotrzebowanie maszyn wirtualnych na wielkość rezerwowanej pamięci RAM opisuje wektor $r = [r_1, \dots, r_k, \dots, r_K]$ [GB], a wektor $h = [h_1, \dots, h_k, \dots, h_K]$ [TB] – zapotrzebowanie maszyn na pamięć dyskową. W chmurach obliczeniowych, w których wykorzystuje się komponenty niskiej jakości, powinno się uwzględnić ograniczenia związane z migracją maszyn wirtualnych. Niech θ_k oznacza czas niedostępności *k*-tej maszyny dla użytkowników i aplikacji podczas migracji między aktualnym *j*-tym węzłem a docelowym *n*-tym. Dla zadanej maksymalnej wartości θ_{max} łącznych czasów niedostępności maszyn podczas migracji można nałożyć następujące ograniczenie:

$$\sum_{k=1}^{K} \theta_k \sum_{j=1}^{M} \sum_{\substack{m=1 \\ m\neq j}}^{M} s_{kj}^{VM} x_{km}^{VM} \leq \theta_{max}, \tag{4.1.10}$$

gdzie zadana wartość $s_{kj}^{VM} = \begin{cases} 1 \text{ jeśli } VM_k \text{ migruje z hosta w } N_j, \\ 0 \text{ w przeciwnym razie,} \end{cases} \quad k = \overline{1,K}, \;\; j = \overline{1,M}.$

Niech $\bar{c}_k$ oznacza koszt transmisji 1 GB danych podczas migracji *k*-tej maszyny między aktualnym *j*-tym węzłem a docelowym *m*-tym. Dla zadanej maksymalnej wartości $\bar{c}_{max}$ kosztów migracji maszyn wprowadza się następujące ograniczenie:

$$\sum_{k=1}^{K} \bar{c}_k h_k \sum_{j=1}^{M} \sum_{\substack{m=1 \\ j\neq m}}^{M} s_{kj}^{VM} x_{km}^{VM} \leq \bar{c}_{max}. \tag{4.1.11}$$

Przyjmuje się, że zadany jest wektor $\Theta = [\theta_1, \dots, \theta_k\ , \dots, \theta_K]$ czasów niedostępności maszyn wirtualnych podczas zmiany hostów oraz macierz alokacji maszyn wirtualnych przed migracją $S = \left[s_{kj}^{VM}\right]_{V \times I}$. Znany jest również wektor kosztów transmisji 1 GB danych podczas transferu maszyn wirtualnych $\bar{C} = [\bar{c}_1, \dots, \bar{c}_k, \dots, \bar{c}_K]$. Ograniczenie (4.1.11) obowiązuje dla migracji maszyn wirtualnych, a przy projektowaniu nowej chmury z początkową alokacją – nie.

Warto podkreślić, że rozwiązania dopuszczalne powinny cechować się wysokim stopniem rozproszenia maszyn wirtualnych $\eta(x)$. Wymagania odnośnie do decentralizacji maszyn wirtualnych można rozszerzyć w ten sposób, aby co najmniej $\eta_{\min}$ par maszyn przydzielić do różnych węzłów, przy czym $1 \le \eta_{\min} \le M(M–1)$:

$$\eta(x) = \sum_{k=1}^{K} \sum_{\substack{u=1 \\ u>v}}^{K} \sum_{i_1=1}^{M} \sum_{\substack{i_2=1 \\ i_1 \neq i_2}}^{M} x_{ki_1}^{VM}\, x_{ui_2}^{VM} \ge \eta_{min}. \qquad (4.1.12)$$

W wypadku działania maszyn wirtualnych istotne są dwa wymagania związane z tolerowaniem uszkodzeń (ang. *fault-tolerance*). Po pierwsze, prawdopodobieństwo zakończenia działań maszyn wirtualnych przed założonymi terminami powinno być nie mniejsze niż wymagany próg. Po drugie, dostępność systemu $\tilde{\mu}$ również powinno nie być mniejsze niż zadany poziom $\tilde{\mu}_{min}$. Załóżmy, że hosty chmury obliczeniowej ulegają uszkodzeniom niezależnie z intensywnością wykładniczą $\chi_n, n = \overline{1, N}$. Nie rozważa się naprawy hostów w rozpatrywanym czasie pracy chmury, gdyż zazwyczaj czas napraw znacząco przewyższa czas pracy maszyn wirtualnych. Natomiast maszyny mogą migrować do hostów, których awarie są mniej prawdopodobne. Nie rozważa się również kopiowania stanu maszyny wirtualnej na wielu hostach, co wymaga użycia znaczącej ilości zasobów. Ograniczenie na dostępność maszyn wirtualnych można zapisać dla zadanego wektora $\chi = [\chi_1, \dots, \chi_k\ , \dots, \chi_K]$, jak następuje [179]:

$$\tilde{\mu}(x) = \prod_{k=1}^{K} \prod_{m=1}^{M} \prod_{n=1}^{N} e^{\left(-\chi_n t_{km} x_{km}^{VM} x_{mn}^{H}\right)} \ge \tilde{\mu}_{min}, \qquad (4.1.13)$$

gdzie e – stała Eulera, ok. 0,5772156649.

Niech niepodzielnym zadaniem jest praca maszyny wirtualnej. Przyjmuje się, że między zadaniami mogą zachodzić ograniczenia kolejnościowe. Każde zadanie powinno zakończyć się przed wymaganym terminem δ_k. Czas zakończenia zadania σ_k można wyznaczyć, symulując wykonanie wszystkich zadań dla ustalonego rozwiązania *x*.

Ze względu na logikę zależności między zadaniami niektóre pary zadań (α_k, α_u) implikowanych działaniem maszyn wirtualnych (VM_k, VM_u) mogą być wykonywane

w trybie OR, w którym zadanie α_k wykonuje się z prawdopodobieństwem $\tilde{p}_k$, a zadanie α_u – (1-$\tilde{p}_k$). Wybrane inne zadania mogą być realizowane co najwyżej L_k razy w pętli (tryb LOOP), przy czym każde powtórzenie realizowane jest z prawdopodobieństwem $\hat{p}_k$ dla *k*-tego zadania. Pozostałe zadania wykonują się deterministycznie.

Jeśli w zbiorze maszyn wirtualnych są dwie maszyny (VM_k, VM_u) pracujące w trybie OR oraz jedna maszyna VM_j pracująca w trybie LOOP, to prawdopodobieństwo wykonania instancji nr λ, w której wykona się maszyna VM_k oraz maszyna VM_j l razy $1 \le l \le L_j$ wynosi $p_\lambda = \tilde{p}_k \hat{p}_j{}^l$. Symulując wykonanie zadań dla instancji nr λ harmonogramu, można wyznaczyć czasy zakończenia zadań $\sigma_k(x), k = \overline{1,K}$. Jeśli ograniczenie $\sigma_k(x) \le \delta_k$ jest spełnione, to $\varsigma_k(x) = 1$, a $\varsigma_k(x) = 0$ w przeciwnym razie. Niech w instancji nr λ rozważa się maszyny wirtualne należące do zbioru $\boldsymbol{VM}_\lambda, \lambda = \overline{1,\Lambda}$, przy czym $\boldsymbol{VM}_\lambda \subseteq \boldsymbol{VM}, \lambda = \overline{1,\Lambda}$. Prawdopodobieństwo, że zadania zostaną wykonane w terminach dla Λ instancji powinno być nie mniejsze niż wymagany poziom p_{ter}^{min}, co można zapisać:

$$p_{ter}(x) = \sum_{\lambda=1}^{\Lambda} p_\lambda \prod_{\alpha_k \in \boldsymbol{VM}_\lambda} \varsigma_k(x) \ge p_{ter}^{min}. \tag{4.1.14}$$

Niech ustalone są dla *k*-tej maszyny wirtualnej: zbiór zadań poprzedzających *k*-te zadanie $\boldsymbol{VM}_k^0 \subset \boldsymbol{VM}$, wymagany termin zakończenia pracy δ_k, tryb pracy ϑ_k oraz czas realizacji zadania t_{kn} w trybie deterministycznym ($\vartheta_k = 1$). Jeśli maszyna pracuje w trybie OR ($\vartheta_k = 2$), to znane jest $\tilde{p}_k$ oraz α_u. W wypadku trybu LOOP ($\vartheta_k = 3$) specyfikacja dotyczy wartości $\hat{p}_k$ oraz L_k. Zadanie nr *k* cechuje uporządkowana piątka $G_k = \left(\boldsymbol{VM}_k^0, \delta_k, \vartheta_k, p_k, \varkappa_k\right)$, przy czym $p_k = \tilde{p}_k$ oraz $\varkappa_k = \alpha_u$ dla $\vartheta_k = 2$. Natomiast $p_k = \hat{p}_k$ oraz $\varkappa_k = L_k$ dla $\vartheta_v = 3$. Zadany jest zatem zbiór $\boldsymbol{G} = \{G_1, \dots, G_k, \dots, G_K\}$.

Ponadto ważnym wymaganiem jest to, aby $\varphi(x)$ – łączna wydajność hostów dla wybranego testu nie była mniejsza niż oczekiwany próg $\varphi_{min.}$, co można zapisać, jak niżej:

$$\varphi(x) = \sum_{m=1}^{M} \sum_{n=1}^{N} \varphi_n x_{mn}^H \ge \varphi_{min.} \tag{4.1.15}$$

Reasumując, zbiór rozmieszczonych agentów pedagogicznych, modułów systemu Moodle oraz solwerów działających w maszynach wirtualnych, które należą do zbioru $\boldsymbol{VM} = \{VM_1, \dots, VM_k, \dots, VM_K\}$ modeluje się za pomocą uporządkowanej ósemki:

$$\boldsymbol{VM} = (T, \tau, r, h, \Theta, \bar{C}, S, \boldsymbol{G}). \tag{4.1.16}$$

Natomiast *k*-ta maszyna wirtualna jest scharakteryzowana za pomocą następującej uporządkowanej ósemki:

$$VM_k = (T_k, \tau_k, r_k, h_k, \theta_k, \bar{c}_k, s_k, G_k), \;\; k = \overline{1,K}. \tag{4.1.17}$$

4.2 Sformułowanie problemu optymalizacji wielokryterialnej

Na podstawie modelu migracji maszyn wirtualnych oraz alokacji hostów w chmurze edukacyjnej, można sformułować problem polioptymalizacji wykorzystania zasobów chmury edukacyjnej przez agenty, moduły zdalnego nauczania i solwery, jak niżej.

Dla danych wejściowych:

- liczb: $\bar{c}_{max}, F_n^{max}, \overline{1,4}$; *K, M, N*, $\tilde{\mu}_{min,}, \eta_{min}, \theta_{max}$;
- wektorów: $c, \bar{c}, h, hdd, ram, r, \;\beta, \psi, \Theta, \chi$;
- macierzy: T, S, τ;
- zbioru $\boldsymbol{G}$,

należy wyznaczyć reprezentację rozwiązań Pareto-optymalnych $\boldsymbol{X}^{Pareto}$ zadania optymalizacji wektorowej, które jest uporządkowaną trójką:

$$(\boldsymbol{X}, F, \succ_{\leq}), \tag{4.2.1}$$

gdzie:

1) $\boldsymbol{X}$ – zbiór rozwiązań dopuszczalnych *x*, które spełniają ograniczenia (4.1.1) – (4.1.7), (4.1.10) – (4.1.15) oraz następujące ograniczenia formalne:

$$\sum_{n=1}^{N} x_{mn}^{H} = 1, \;\; m = \overline{1,M}, \tag{4.2.2}$$

$$\sum_{m=1}^{M} x_{km}^{VM} = 1, \;\; k = \overline{1,K}, \tag{4.2.3}$$

2) F – funkcja kryterium

$$F: \boldsymbol{X} \to \boldsymbol{R}^4, \tag{4.2.4}$$

$$F(x) = [F_1(x), F_2(x), F_3(x), F_4(x)],$$

3) $\succ_{\leq}$ – relacja dominowania w $\boldsymbol{R}^4$

$$\succ_{\leq} = \{(y,z) \in \boldsymbol{Y} \times \boldsymbol{Y} | y_n \leq z_n, \;\; n = \overline{1,4}\}, \;\; \boldsymbol{Y} = F(\boldsymbol{X}). \tag{4.2.5}$$

Warto podkreślić, że na podstawie założeń modelowych otrzymujemy następujące zależności dotyczące kryterium cząstkowego F_1 [111]:

$$F_1(x) = \max_{m=1,M} F_{1m}(x), \tag{4.2.6}$$

$$F_{1m}(x) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn}\, x_{km}^{VM} x_{mn}^{H}, \;\; k = \overline{1,M}, \tag{4.2.7}$$

W wypadku kryterium cząstkowego F_2 zachodzi, jak niżej [111]:

$$F_2(x) = \max_{n=1,N} F_{2n}(x), \tag{4.2.8}$$

$$F_{2n}(x) = \sum_{k=1}^{K} \sum_{\substack{u=1 \\ u \neq v}}^{K} \sum_{m=1}^{M} \sum_{\substack{j=1 \\ m \neq j}}^{M} \sum_{n=1}^{N} \sum_{\substack{l=1 \\ n \neq l}}^{N} \tau_{kunjnl}\, x_{km}^{VM} x_{mn}^{H} x_{uj}^{VM} x_{ul}^{H}, \tag{4.2.9}$$

Wartości kryterium cząstkowego F_3 wyznacza się następująco:

$$F_3(x) = \sum_{m=1}^{M} \sum_{n=1}^{N} c_n\, x_{mn}^{H}, \tag{4.2.10}$$

Natomiast wartości F_4 oblicza się za pomocą zależności, jak następuje:

$$F_4(x) = \sum_{m=1}^{M} \sum_{n=1}^{N} \beta_n\, x_{mn}^{H}, \tag{4.2.11}$$

Do wyznaczenia wartości μ_{min}^{RAM} stosuje się poniższe formuły:

$$\mu_{min}^{RAM}(x) = \min_{m=1,M} \{\mu_m^{RAM}(x)\}, \tag{4.2.12}$$

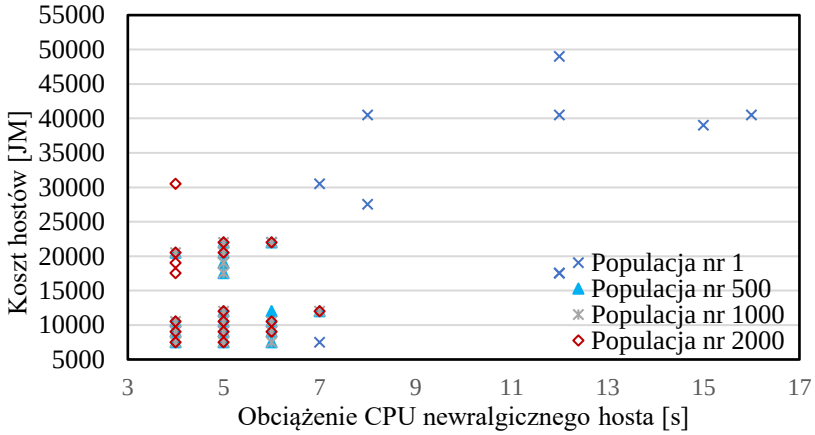
$$\mu_m^{RAM}(x) = \sum_{n=1}^{N} ram_n\, x_{mn}^{H} - \sum_{k=1}^{K} r_k\, x_{km}^{VM}, \quad m = \overline{1, M}, \tag{4.2.13}$$

Dostępną pamięć dyskową μ_{min}^{RAM} w newralgicznym hoście wyznacza się, jak niżej:

$$\mu_{min}^{DM}(x) = \min_{m=1,M} \{\mu_m^{DM}(x)\}, \tag{4.2.14}$$

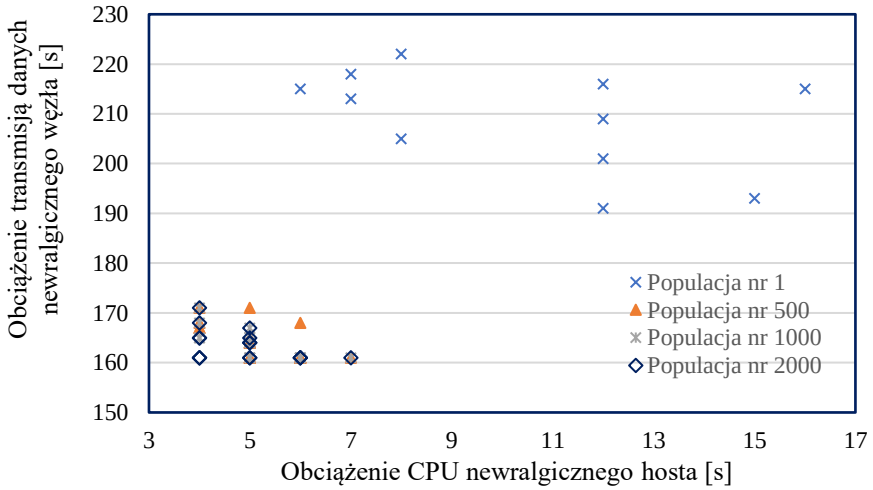
$$\mu_m^{HDD}(x) = \sum_{n=1}^{N} dm_n\, x_{mn}^{H} - \sum_{k=1}^{K} h_k\, x_{km}^{VM}, \quad n = \overline{1, N}. \tag{4.2.15}$$

Na rysunkach 21 – 26 przedstawiono dwuwymiarowe przekroje wybranych ocen rozwiązań dopuszczalnych ze zbioru $\boldsymbol{Y} \subset \boldsymbol{R}^4$ dla instancji problemu optymalizacji wielokryterialnej (4.2.1) – (4.2.5). W celu przejrzystej wizualizacji, oceny z przestrzeni czterowymiarowej są prezentowane na sześciu możliwych przekrojach 2D.

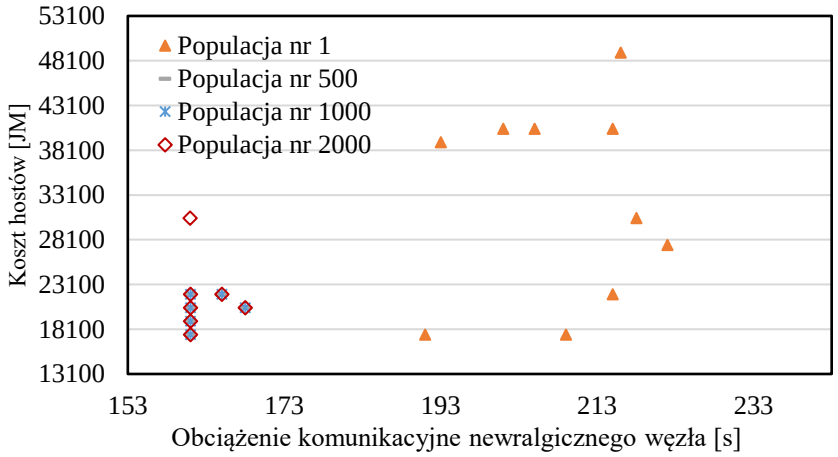


Rysunek 21 Wybrane oszacowania obciążenia CPU newralgicznego hosta oraz kosztów hostów dla instancji B90.
Źródło: opracowanie własne.

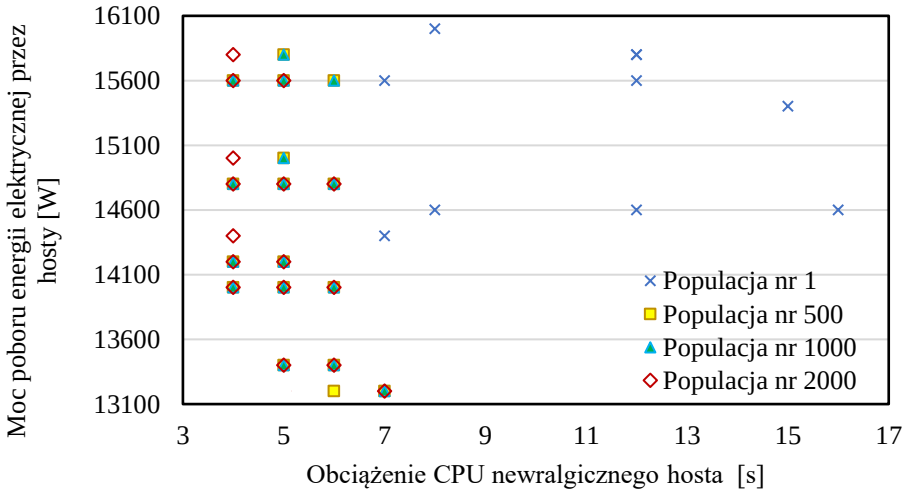
Na rysunku 21 rozważa się dwa kryteria: obciążenie CPU newralgicznego hosta oraz koszt hostów. Dane wejściowe dla rozważanej instancji B90 zapisano w pliku benchmark90.txt [154]. Oceny wyznaczono za pomocą algorytmu DEMCA.



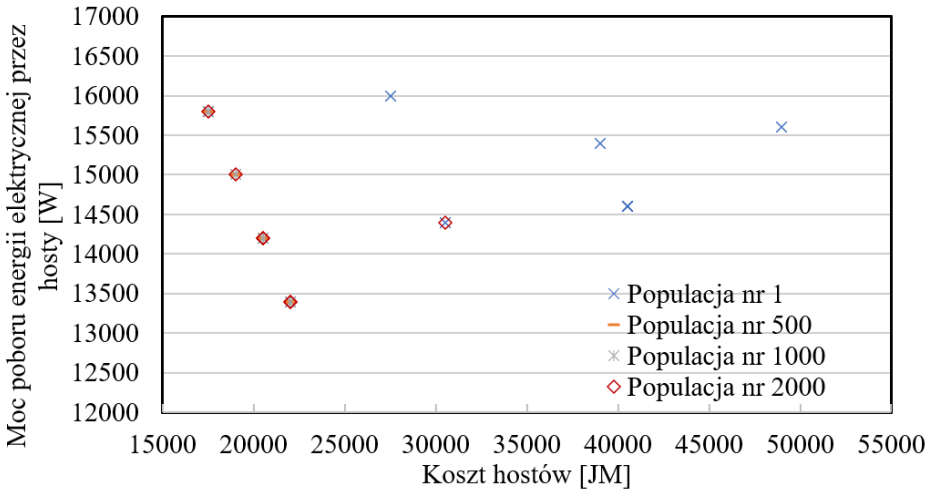
Rysunek 22 Wybrane oszacowania obciążenia CPU newralgicznego hosta oraz obciążenia transmisją danych newralgicznego węzła dla instancji B90.
Źródło: opracowanie własne.



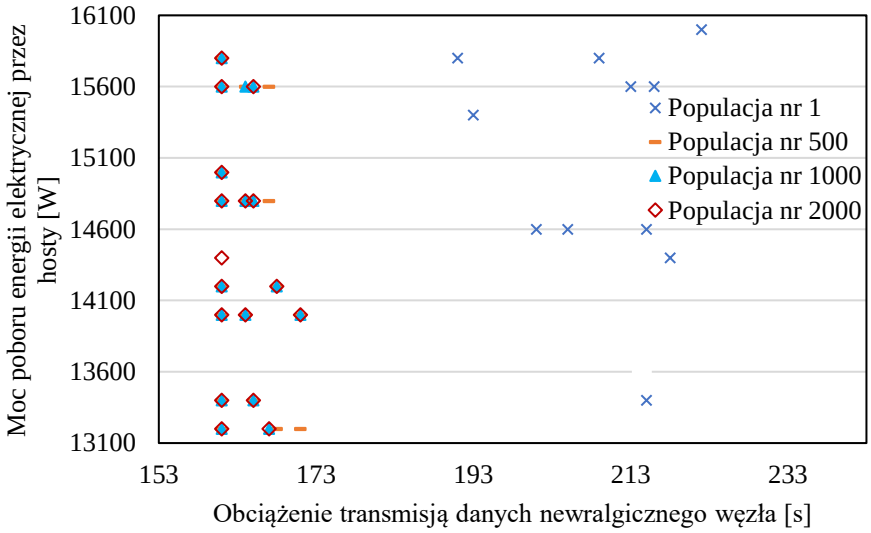
Rysunek 23 Wybrane oszacowania obciążenia komunikacyjnego newralgicznego węzła oraz kosztu hostów dla instancji B90.
Źródło: opracowanie własne.



Rysunek 24 Wybrane oszacowania obciążenia CPU newralgicznego hosta oraz łącznej mocy poboru energii elektrycznej przez hosty chmury dla instancji B90.
Źródło: opracowanie własne.



Rysunek 25 Wybrane wartości kryteriów skalarnych F_3 oraz F_4 w wypadku instancji B90. Źródło: opracowanie własne.



Rysunek 26 Wybrane wartości kryteriów skalarnych F_2 oraz F_4 w wypadku instancji B90. Źródło: opracowanie własne.

4.3 Metoda ewolucji różnicowej do migracji maszyn wirtualnych

W zagadnieniu optymalizacji wielokryterialnej (4.2.1) – (4.2.5) występuje sześć konfliktów między parami kryteriów skalarnych. Przykładowo, rozwiązanie może cechować się mniejszym obciążeniem procesorów newralgicznego hosta, ale za to większym obciążaniem krytycznego hosta pod względem transmisji danych w sytuacji, gdy maszyny wirtualne są bardziej rozproszone.

Podstawowe pojęcia dotyczące optymalizacji wielokryterialnej, w tym rozwiązania Pareto-optymalne, alternatywy dominujące oraz warianty kompromisowe zdefiniowano w [10]. Omówiono w niej również relacje dominowania w przestrzeni kryterialnej oraz kresy zbiorów uporządkowanych.

Analizując specyfikę problemu optymalizacji wielokryterialnej (4.2.1) – (4.2.5), można udowodnić następujący lemat i twierdzenie. Niech $\boldsymbol{X}_N^{\leqslant}$ oznacza zbiór rozwiązań optymalnych w sensie Pareto (sprawnych, efektywnych) [7, 10].

Lemat 4.1
Zbiór rozwiązań optymalnych w sensie Pareto $\boldsymbol{X}_N^{\leqslant}(2) \subseteq \boldsymbol{X}$ dla dwóch wybranych kryteriów skalarnych F_1 oraz F_2 w problemie (4.2.1) – (4.2.5) $(\boldsymbol{X}, F, \succ_{\leq})$ zawiera się w zbiorze rozwiązań Pareto $\boldsymbol{X}_N^{\leqslant}(3) \subseteq \boldsymbol{X}$ dla trzech kryteriów skalarnych F_1, F_2 oraz F_3, zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$ oraz relacji dominowania $\succ_{\leq}$ w $\boldsymbol{R}^3$.

Dowód: Niech $\boldsymbol{X}_N^{\leqslant}(2)$ jest zbiorem rozwiązań optymalnych w sensie Pareto dla kryteriów skalarnych F_1 oraz F_2 [7]. Jeśli rozważa się dodatkowo trzecie kryterium cząstkowe F_3, rozwiązania ze zbioru $\boldsymbol{X}_N^{\leqslant}(2)$ są rozwiązaniami Pareto-optymalnymi bez względu na ich wartości wyznaczone za pomocą kryterium skalarnego F_3. Ponieważ może istnieć rozwiązanie dopuszczalne $x \in \boldsymbol{X}, x \notin \boldsymbol{X}_N^{\leqslant}(2)$, które ze względu na wartość $F_3(x)$ jest rozwiązaniem niezdominowanym w sensie relacji dominowania $\succ_{\leq}$ w $\boldsymbol{R}^3$, to zachodzi $\boldsymbol{X}_N^{\leqslant}(2) \subseteq \boldsymbol{X}_N^{\leqslant}(3)$, co kończy dowód. □

Twierdzenie 4.1
Zbiór rozwiązań optymalnych w sensie Pareto $\boldsymbol{X}_P^{\leqslant}(I) \subseteq \boldsymbol{X}$ dla I ($I \geq 2$) kryteriów skalarnych w problemie optymalizacji wielokryterialnej (4.2.1) - (4.2.5) $(\boldsymbol{X}, F, \succ_{\leq})$ zawiera się w zbiorze rozwiązań Pareto-optymalnych $\boldsymbol{X}_P^{\leqslant}(I + l) \subseteq \boldsymbol{X}$ dla I+l kryteriów cząstkowych, l=1, 2, 3, …, zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$, a także relacji dominowania $\succ_{\leq}$ w $\boldsymbol{R}^{I+l}$.

Dowód przeprowadza się w oparciu o lemat 4.1.

Wniosek 4.1
Problem optymalizacji (4.2.1) – (4.2.5) $(\boldsymbol{X}, F, \succ_{\leq})$ jest problemem klasy NP.

Można przekształcić uproszczony problem migracji maszyn wirtualnych dla wybranego rodzaju hosta do NP-zupełnego problemu plecakowego [21]. Ze względu na fakt, że decyzyjny problem odpowiadający zagadnieniu (4.2.1) – (4.2.5) jest NP-zupełny, zagadnienie optymalizacji wektorowej (4.2.1) – (4.2.5) jest NP-trudne.

Spośród kryteriów cząstkowych zagadnienia (4.2.1) – (4.2.5) najbardziej kosztowne obliczeniowo jest wyznaczenie obciążenia komunikacyjnego F_2, a złożoność obliczeniowa dla tego kryterium wynosi $O(n^7)$, przy czym n = max {*K, M, N*}. Z kolei wyznaczenie obciążenia procesorów newralgicznego hosta cechuje się złożonością $O(n^3)$. W przypadku kryteriów F_3 oraz F_4 złożoność obliczeniowa to $O(n^2)$.

W rozprawie zastosowano metodę ewolucji różnicowej DEMCA do wyznaczania reprezentacji Pareto-optymalnych rozwiązań problemu (4.2.1) – (4.2.5). Rozwiązanie jest reprezentowane w nieco inny sposób niż para macierzy $x = (X^H, X^{VM})$. Wprowadza się następującą parę wektorów:

$$\bar{x} = (\bar{X}^H, \bar{X}^{VM}), \tag{4.3.1}$$

gdzie:

$\bar{X}^H = [\bar{X}_1^H, \dots, \bar{X}_m^H, \dots \bar{X}_M^H]$, przy czym $\bar{X}_m^H = n,$ jeśli $x_{mn}^H = 1,$

$\bar{X}^{VM} = [\bar{X}_1^{MV}, \dots, \bar{X}_k^{MV}, \dots \bar{X}_K^{MV}]$, przy czym $\bar{X}_k^{MV} = m,$ jeśli $x_{km}^{MV} = 1.$

Uzyskuje się dwie istotne zalety z powodu zmiany specyfikacji rozwiązania z formatu macierzowego x na format wektorowy $\bar{x}$. Po pierwsze, spełnione są ograniczenia formalne (4.2.2) oraz (4.2.3). Po drugie, zmniejsza się znacząco przestrzeń przeszukiwań algorytmu ewolucji różnicowej.

Zmienna decyzyjna $\bar{X}_k^{VM}$ reprezentuje numer destynacji maszyny wirtualnej nr *k*. Zmienna przyjmuje wartości należące do podzbioru liczb naturalnych od 1 do *M*, co zapewnia spełnienie ograniczenia (4.2.2). Powyższa reguła kodowania ułatwia generowanie populacji początkowej, a także implementację mutacji różnicowej czy krzyżowania. Podobne kodowanie zachodzi dla zmiennej decyzyjnej $\bar{X}_m^H$, którą jest indeks rodzaju hosta nr *n*. Zmienna przyjmuje wartości z przedziału od 1 do *N*, co zapewnia spełnienie ograniczenia (4.2.3). Aby współrzędne $\bar{X}_k^{VM}$ po mutacji różnicowej nie były większe od *K* oraz $\bar{X}_m^H$ – od *N*, stosowana jest operacja modulo *K* lub operacja modulo *N*, której wynik jest zwiększony o 1, co zapewnia spełnienie ograniczeń (4.2.2) i (4.2.3).

Dla $K, N, M > 1$ liczba binarnych zmiennych decyzyjnych $M_{bin} = M(K + N)$ jest większa niż liczba zmiennych decyzyjnych $\bar{x}$ $M_{int} = K + M$. Liczebność binarnej przestrzeni przeszukiwań $2^{M(K+N)}$ jest znacząco większa niż liczebność całkowitoliczbowej przestrzeni przeszukiwań $M^K N^M$. Dla dużych wartości K, M, N zachodzi eksplozja kombinatoryczna, która polega na wielokrotnym zwiększeniu liczebności przestrzeni spowodowanym niewielkim zwiększeniem K, M lub N.

W algorytmie DEMCA zastosowano procedurę nadawania rang, którą Fonseca i Fleming zaproponowali w [52]. Procedura ta cechuje się pewną wadą, gdyż oceny należące do brzegu Pareto mogą otrzymać jednakową rangę, mimo, że niektóre z nich są usytuowane w zagęszczonych regionach. Aby rozwiązać ten problem wprowadzono geometryczną miarę zagęszczenia *GCD* (ang. *geometric crowding distance*) w znormalizowanej przestrzeni wielokryterialnej. Niech w populacji znajduje się *J* rozwiązań o tej

samej randze. Wartość *GCD* to średnia odległość Euklidesa w znormalizowanej przestrzeni wielokryterialnej między *j*-tą oceną a pozostałymi ocenami, które cechują się tą samą rangą. Znormalizowaną wartość *GCD* dla *j*-tej oceny w przestrzeni *I*-wymiarowej wyznacza się za pomocą następującej zależności [111]:

$$GCD(j) = \frac{1}{(J-1)\sqrt{I}} \sum_{\substack{l=1 \\ l \neq j}}^{J} \sqrt{\sum_{i=1}^{I} \left(\frac{F_i(j) - F_i^{min}}{F_i^{max} - F_i^{min}} - \frac{F_i(l) - F_i^{min}}{F_i^{max} - F_i^{min}}\right)^2}, \quad j = \overline{1,J};\ i = \overline{1,I}, \quad (4.3.2)$$

gdzie:
F_i^{min}– minimalna wartość *i*-tej współrzędnej ocen w populacji,
F_i^{max} – maksymalna wartość *i*-tej współrzędnej ocen w populacji.

Znormalizowana wartość *GCD* należy do przedziału domkniętego $[0; 1]$. Rangi oraz wartości miary zagęszczenia umożliwiają sortowanie rozwiązań dopuszczalnych w populacji za pomocą funkcji sprawności *f*. Natomiast sortowanie umożliwia zastosowanie strategii elitarystycznej podczas sukcesji [57]. Wybór strategii elitarystycznej zapobiega utracie efektywnych rozwiązań podczas wyznaczania nowej populacji.

W wypadku rozwiązań należących do zbioru wariantów niedopuszczalnych, zazwyczaj obniża się wartość funkcji sprawności *f* o karę za niespełnienie ograniczeń (4.1.1) – (4.1.7) oraz (4.1.10) – (4.1.15). Warto podkreślić, że ograniczenia (4.2.2) i (4.2.3) są spełnione w każdym przypadku ze względu na sposób kodowania chromosomu. Natomiast ograniczenia nierównościowe (4.1.1) – (4.1.6) oraz (4.1.10) – (4.1.15) można przedstawić za pomocą układu dziewięciu nierówności, jak niżej:

$$g_l(x) \leq g_l^{max}, \quad l = \overline{1,12}. \quad (4.3.3)$$

W tym wypadku nierówność $\eta(x) \geq \eta_{min}$ zachodzi $-\eta(x) \leq -\eta_{min}$. Natomiast układ ograniczeń równościowych (4.1.7) $\sum_{m=1}^{M} x_{mn}^H = \psi_n,\ n \in \boldsymbol{N}^* \subset \boldsymbol{N}$ zawiera równania, w których są dodatnie wartości ψ_n. Nie rozważa się ograniczeń, w których $\psi_n = 0$. Ograniczenia (4.1.7) można zapisać następująco:

$$g_{12+n}(x) = \psi_n, \quad n = \overline{1, N^{max}}, \quad n \in \boldsymbol{N}^* \subset \boldsymbol{N}, \quad (4.3.4)$$

gdzie N^{max} – liczebność zbioru rekomendowanych rodzajów hostów $\boldsymbol{N}^*$, $N^{max} \leq N$.

W wypadku niespełnienia *l*-tego ograniczenia nierównościowego (4.3.3) zachodzi $g_l(x) > g_l^{max}$. Zatem dąży się do minimalizacji różnicy $P_l(x) = g_l(x) - g_l^{max} > 0$, która jest karą za niespełnienie *l*-tego ograniczenia (4.3.3). Niespełnienie *n*-tego ograniczenia równościowego powoduje, że $|g_{12+n}(x) - \psi_n| > 0$. W tym wypadku dąży się do minimalizacji kary $P_n(x) = |g_{12+n}(x) - \psi_n|$.

Zastosowanie kar za przekroczenie ograniczeń jest metodą obarczoną wadą ze względu na konieczność skalowania kar, co wymaga wyznaczania maksymalnych wartości funkcji kar w celu skalaryzacji do przedziału domkniętego $[0; 1]$. Kolejną wadą jest konieczność doboru wag dla funkcji kar.

W algorytmie zastosowano podejście opierające się na rangach rozwiązań niedopuszczalnych, które nie ma powyższych wad. Jeśli w populacji istnieje pojedyncze rozwiązanie niedopuszczalne $x \notin \boldsymbol{X}$, to jego ranga $\varphi(x) = 0$. Jeśli jednak istnieją co najmniej dwa rozwiązania niedopuszczalne, to można je porównać zgodnie z procedurą nadawania rang Fonseci-Fleminga. W tym wypadku nie rozważa się wektorowej funkcji celu *F*, lecz wektorową funkcję kar *P*, taką że:

$$P(x) = [P_1(x), \dots, P_k(x), \dots, P_{12}(x), \dots, P_{12+n}(x), \dots, P_{12+N^{max}}(x)], \;\; x \notin \boldsymbol{X}]. \quad (4.3.5)$$

Skalarne funkcje kar dla ograniczeń nierównościowych definiuje się, jak niżej:

$$P_l(x) = \begin{cases} 0, & dla \;\; g_l(x) \leq g_l^{max} \\ g_l(x) - g_l^{max}, & dla \;\; g_l(x) > g_l^{max} \end{cases}, \quad x \notin \boldsymbol{X}, \;\; l = \overline{1{,}12}. \quad (4.3.6)$$

Natomiast skalarne funkcje kar dla ograniczeń równościowych to:

$$P_n(x) = \begin{cases} 0, & dla \;\; g_{12+n}(x) = \psi_n \\ |g_{12+n}(x) - \psi_n|, & dla \;\; |g_{12+n}(x) - \psi_n| > 0 \end{cases}, \quad x \notin \boldsymbol{X}, \;\; n = \overline{1, N^{max}}. \quad (4.3.7)$$

Im mniejsza wartość wyznaczonej rangi $\varphi(x), x \notin \boldsymbol{X}$, tym mniej rozwiązań dominuje alternatywę niedopuszczalną x w $12+N^{max}$ wielowymiarowej przestrzeni kar. Jeśli rozwiązania niedopuszczalne cechują się tą samą rangą, to uwzględnienie GCD^{con} powoduje preferowanie wariantów niedopuszczalnych z obszarów mniej zagęszczonych.

W wypadku *J* rozwiązań niedopuszczalnych w populacji, znormalizowaną wartość GCD^{con} dla *j*-tej oceny w przestrzeni kar wyznacza się, jak niżej:

$$GCD^{con}(j) = \frac{1}{(J-1)\sqrt{12 + J^{max}}} \sum_{\substack{l=1 \\ l \neq j}}^{J} \sqrt{\sum_{i=1}^{12+J^{max}} \left(\frac{P_i(m) - P_i^{min}}{P_i^{max} - P_i^{min}} - \frac{P_i(l) - P_i^{min}}{P_i^{max} - P_i^{min}} \right)^2},$$

$$i = \overline{1{,}12 + N^{max}}, \quad (4.3.8)$$

gdzie:
P_i^{min} – minimalna wartość *i*-tej funkcji kary w populacji,
P_i^{max} – maksymalna wartość *i*-tej funkcji kary w populacji.

W wypadku mutacji różnicowej można empirycznie dobrać najbardziej efektywną strategię, np. DE/rand/1. Dodatkowo w celu poprawienia wybranych kryteriów

cząstkowych wprowadzono mutację tabu (rys. 27). Wartość parametrów *Tabu*1 i *Tabu*2 wyznaczana jest metodą prób i błędów, przy warunkach: $0 < Tabu1 < K$ oraz $0 < Tabu2 < M$. Jeżeli długość restrykcji wyrażona w iteracjach jest zbyt mała, to trajektoria przeszukiwania rozwiązań wpadnie w niepożądany cykl. Cykle zazwyczaj nie wyznaczają rozwiązań wysokiej jakości i kończą przeszukiwanie w przestrzeni możliwych wariantów. Wyznaczenie wartości obu parametrów jest szczególnie istotne w wypadku dużej liczby maszyn wirtualnych oraz rodzajów komputerów ($K > 20$, $N > 20$). Dla instancji z 30 maszynami wirtualnymi wartość $Tabu1=20$ kilkakrotnie obniżył średni błąd względny w stosunku do wartości $Tabu1=10$.

Zaakceptowany ruch powodujący migrację *k*-tej maszyny wirtualnej z *m*-tego węzła na *l*-ty powoduje „zamrożenie" możliwości wykonania tego ruchu przez następne *Tabu1* iteracji. Dodatnia wartość elementu macierzy restrykcji krótkoterminowych STMV $stmv_{kml}$ nie zezwala na wykonanie tej migracji. Przyjęto, że $Tabu1 < K$, aby nie nałożyć restrykcji na przemieszczanie się wszystkich maszyn wirtualnych. Niedodatnia wartość $stmv_{kml}$ zezwala na migrację. Również element macierzy restrykcji wymiany hostów w węźle $stmi_{mni}$ zezwala lub nie na wymianę hostów. Wstrzymanie wymiany zadanej pary hostów w ustalonym węźle nie może być dłuższe niż *Tabu2* wymian.

Ważną rolę odgrywa również pamięć długoterminowa *LTM*, której stan opisuje para macierzy (*LTMV*, *LTMH*). Element $ltmv_{vil}$ trójwymiarowej macierzy *LTMV* pamięci długoterminowej to liczba migracji *k*-tej maszyny wirtualnej z *m*-tego węzła na *l*-ty. Natomiast element $ltmh_{mni}$ macierzy *LTMH* pamięci długoterminowej to liczba wymian w *m*-tym węźle komputera *n*-tego rodzaju na *i*-ty. Przy dużej liczbie iteracji zbyt częste powtarzanie tych samych ruchów nie jest efektywne, gdyż eksploruje te same regiony.

Kodowanie rozwiązania $\bar{x}$ umożliwia wyznaczenie sumarycznej liczby iteracji wykonanych ruchów powiązanych z $\bar{x}$. Jeśli $\bar{X}_k^{MV} = l$, to sumaryczną liczbę migracji *k*-tej maszyny wirtualnej do *l*-tego węzła podczas przeszukiwania można wyznaczyć następująco:

$$VM(\bar{X}_k^{MV}) = \sum_{\substack{m=1 \\ l \neq i}}^{M} ltmv_{kml}\,,\ x \in \boldsymbol{C}(STM, x^{now}). \tag{4.3.9}$$

1. Wybór rozwiązania niezdominowanego
(A) Wybór rozwiązania niezdominowanego z aktualnej populacji $x^{now} \in P(t)$
(B) Wybór kryterium cząstkowego $F_i, i \in \{1,2,3,4\}$
(C) Zapamiętanie x^{now} jako najlepszego rozwiązania $x^{best} := x^{now}$
(D) Zapamiętanie $F_i(x^{now})$ jako najlepszej wartości funkcji celu $best\_cost := F_i(x^{now})$
(E) Wyzerowanie macierzy restrykcji $STM = (STMV, STMI)$ dla pamięci krótkoterminowej
(F) Ustalenie czasu wstrzymania migracji maszyn wirtualnych $Tabu1$, $0 < Tabu1 < K$
(G) Ustalenie czasu wstrzymania wymian rodzajów hostów $Tabu2$, $0 < Tabu2 < N$
(H) Wyzerowanie macierzy częstości zmian LTM dla pamięci długoterminowej

2. Wybór pretendenta i warunek stopu
(A) Wyznaczenie zbioru kandydatów $\boldsymbol{C}(STM, x^{now})$ jako podzbiór sąsiedztwa $\boldsymbol{N}(x^{now})$
(B) Wybór pretendenta $x^{next} \in \boldsymbol{C}(STM, x^{now})$ z niepustego zbioru kandydatów za pomocą minimalizacji funkcji wyboru S
(C) **Kryterium aspiracji.** Jeżeli wszystkie rozwiązania z $\boldsymbol{C}(STM, x^{now})$ są oznaczone jako tabu, to jako x^{next} kwalifikowany jest wariant, który poprawia wartość $0{,}75best\_cost$
(D) **Ewolucyjna procedura intensyfikacji.** Jeżeli nie wybrano x^{next} , to wyznacza się parę potomków za pomocą krzyżowania dwóch najlepszych, wariantów x^{best}, x^{bis}. Rozwiązaniem x^{next} jest alternatywa o mniejszej wartości funkcji wyboru S
(E) Jeżeli zostanie przekroczony limit iteracji lub czasu, to zwróć x^{best} do populacji ewolucji różnicowej. STOP.

3. Aktualizacja pamięci krótkoterminowej i pamięci długoterminowej
(A) Zapamiętanie wybranego rozwiązania $x^{now} := x^{next}$
(B) Jeżeli $best\_cost > F_n(x^{now})$, to $x^{bis} := x^{best}$ oraz wykonaj 1(C)
(C) Aktualizacja macierzy restrykcji krótkoterminowej $STM := STM - 1$
$stmv_{kml} := Tabu1$ po wykonaniu migracji k-tej maszyny z m-tego węzła na l-ty
$stmi_{mni} := Tabu2$ po wykonaniu zmiany w m-tym węźle z n-tego rodzaju hosta na i-ty
(D) Aktualizacja macierzy częstości zmian LTM dla pamięci długoterminowej. Powrót do 2

Rysunek 27 Diagram mutacji tabu search w ewolucji różnicowej.
Źródło: opracowanie własne.

Natomiast łączną liczbę migracji maszyn wirtualnych oraz wymian rodzajów hostów prowadzących do skonstruowania rozwiązania $\bar{x}$ można wyznaczyć, jak niżej:

$$Pressure(\bar{x}) = \sum_{k=1}^{K} \sum_{\substack{m=1 \\ l \neq i}}^{M} ltmv_{kml} + \sum_{m=1}^{M} \sum_{\substack{n=1 \\ i \neq n}}^{N} ltmh_{mni}, x \in \boldsymbol{C}(STM, x^{now}) \quad (4.3.10)$$

Presja zapobiegająca częstym wykonywaniu ruchów, które umożliwiają skonstruowanie rozwiązania $\bar{x}$ zmienia zakres przeszukiwań. Wartość funkcji selekcji wariantu jest zwiększana o znormalizowaną wartość miary presji $\overline{Pressure}(\bar{x})$, jak niżej:

$$S(x) = \bar{F}_n(x) + \overline{Pressure}(\bar{x}), \quad x \in \boldsymbol{C}(STM, x^{now}). \quad (4.3.11)$$

Warto podkreślić, że mutacja tabu search jest realizowana z niewielkim tempem. Wybór kryterium cząstkowego może odbywać się losowo lub niektóre kryteria mogą być preferowane. Dotyczy to zwłaszcza obciążenia transmisją danych newralgicznego węzła, którego minimalizacja nastręcza najwięcej kłopotów za pomocą algorytmu ewolucji różnicowej.

Po wyznaczeniu rang w populacji znana jest wartość rangi maksymalnej r_{max} dla rozwiązań dopuszczalnych oraz wartość rangi maksymalnej φ_{max} dla rozwiązań niedopuszczalnych. W algorytmie DEMCA wartość maksymalizowanej funkcji sprawności rozwiązania z populacji zdefiniowano następująco:

$$f(x) = \begin{cases} \varphi_{max} + r_{max} - r(x) + \frac{1}{2}GCD(x), & dla\ \ x \in \boldsymbol{X}, \\ \varphi_{max} - \varphi(x) + \frac{1}{2}GCD(x), & dla\ \ x \notin \boldsymbol{X}. \end{cases} \tag{4.3.12}$$

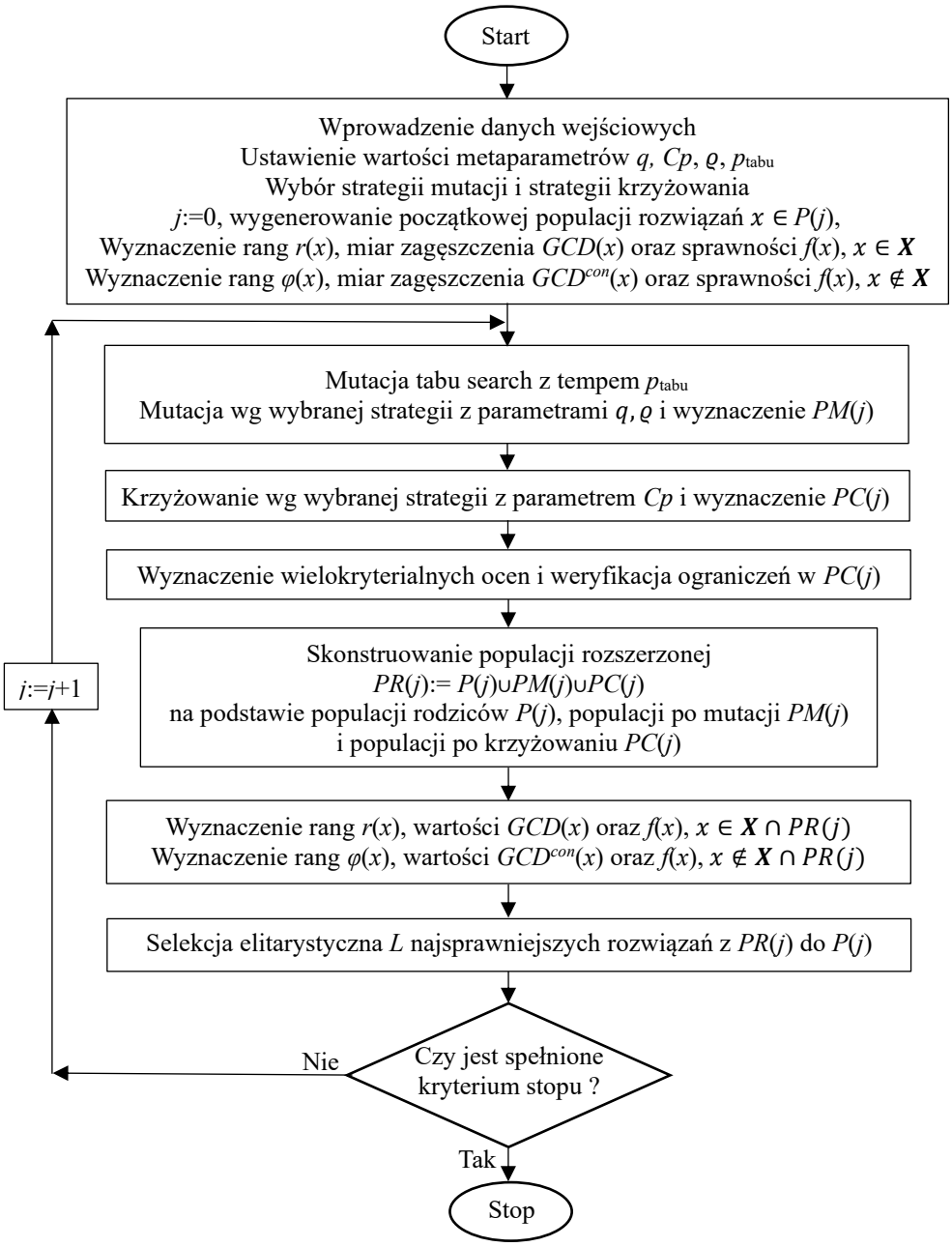
Strategia elitarystyczna stosowana jest w populacji pośredniej, która zawiera rozwiązania należące do populacji po mutacji i krzyżowaniu, a także rozwiązania należące do populacji rodziców. Populacja pośrednia jest sortowana od najlepszego rozwiązana do najgorszego L-tego rozwiązania w sensie funkcji sprawności. Procedura wyznaczania nowej populacji powtarzana jest co najwyżej T_{max} razy.

Oszacowanie złożoności obliczeniowej mutacji tabu search sprowadza się do wyznaczenia złożoności procedury wyboru pretendenta, która jest realizowana co najwyżej $\tilde{T}_{max}$ razy. Przyjęto, że maksymalna liczba iteracji w mutacji tabu $\tilde{T}_{max}$= 5(K+M). Podczas wyboru pretendenta obliczenie funkcji celu i testowanie ograniczeń odbywa się K+M razy. Ponieważ $O(n^7)$ to złożoność obliczeniowa najbardziej czasochłonnej weryfikacji ograniczenia (4.1.2), to złożoność mutacji tabu jest $O(n^9)$, gdzie n = max {K, M, N}.

Na rysunku 28 przedstawiono diagram wielokryterialnego algorytmu ewolucji różnicowej DEMCA, który znacząco różni się od algorytmu Abbasa-Sarkera [1] oraz od algorytmu Jamali-Mallipeddi-Salehpoura-Bagheri [70]. Pierwsza różnica polega na możliwości wyboru procedury nadawania rang między procedurą Goldberga a procedurą Fonseci-Flaminga. Ponadto uwzględnienia się miarę GCD, preferując rozwiązania efektywne w mniej „zagęszczonych" obszarach przestrzeni przeszukiwań. Kolejna różnica polega na zastosowaniu powyższych procedur dla rozwiązań niedopuszczalnych, na które nałożono kary za przekroczenie ograniczeń. Istotną różnicą jest uwzględnienie mutacji tabu w celu poprawy wybranego kryterium cząstkowego.

Warto również rozważyć nowe strategie mutacji, które zaproponowano w algorytmie DEMCA. Oprócz strategii klasycznych: DE/best/1, DE/best/2, DE/current to best/2 oraz strategii DE/Pareto/1 znanej z algorytmu Abbasa-Sarkera, opracowano pięć nowych strategii: DE/Pareto/2 (2.4.10), DE/current to Pareto/2 (2.4.11), DE/2Pareto/1, a także DE/2Pareto/2 oraz DE/current to 2Pareto/2. Niech dwa różne rozwiązania efektywne x^{ef1}, x^{ef2} będą losowane z bieżącej populacji $P(k)$:

$$x^{ef1} \neq x^{ef2};\quad x_l, x^{ef1}, x^{ef2} \in P(k), j = \overline{1, J_{max}}. \tag{4.3.13}$$



Rysunek 28 Diagram wielokryterialnego algorytmu ewolucji różnicowej DEMCA.
Źródło: opracowanie własne.

W strategii mutacji DE/2Pareto/1 uwzględnia się rozwiązania sprawne, jak niżej:

$$\tilde{x}_l = x_{l} + q(x^{ef1} - x^{ef2}), \tag{4.3.14}$$

Natomiast mutacja DE/2Pareto/2 polega na modyfikacji x_{l} za pomocą dwóch różnic:

$$\tilde{x}_l = x_{l} + q\left(x^{ef1} - x_{l_1}\right) + q\left(x^{ef2} - x_{l_2}\right),\ l_1 \neq l_2. \tag{4.3.15}$$

Z kolei strategia DE/current to 2Pareto/2 jest następująca:

$$\tilde{x}_l = x_{l_1} + q\left(x^{ef1} - x_{l_2}\right) + q\left(x^{ef2} - x_{l_3}\right), \qquad l_1 \neq l_2 \neq l_3. \tag{4.3.16}$$

W algorytmie DEMCA zastosowano ekstensywną eksplorację przestrzeni przeszukiwań podczas generowania populacji. Niech populacja rozszerzona $PR(j)$ jest sumą

zbiorów $P(j)$, $PM(j)$ i $PC(j)$, co zapobiega pomijaniu rozwiązań niezdominowanych, które wyznaczono podczas mutacji. Wprawdzie Hancer zaproponował łączenie populacji rodziców z populacją po mutacji, a następnie sukcesję na podstawie sortowania rang [62], ale nie stosował krzyżowania. Natomiast w algorytmie Abbasa-Sarkera pomija się populację po mutacji podczas sukcesji [1].

Wadą dotychczasowych podejść jest również „uboga" operacja krzyżowania. Przy niewłaściwie dobranym parametrze preferowania genów potomków efekty krzyżowania mogą być niewielkie. Jeśli Cp jest małe, to praktycznie geny potomków nie są wybierane. Z kolei zbyt duża wartość progu powoduje, że współrzędne rozwiązań rodziców mają niewielkie szanse na akceptację. Wartość Cp=0,5 powoduje losowy dobór genów. Ponieważ w początkowym etapie przeszukiwań rozwiązania z bieżącej populacji są zazwyczaj niskiej jakości, to wskazana jest intensywna eksploracja przestrzeni przeszukiwań z dużą wartością progu akceptacji genów potomków. Natomiast w końcowej fazie przeszukiwań rozwiązania niezdominowane w populacji rodziców powinny być już wysokiej jakości i krzyżowanie powinno odgrywać mniejszą rolę dla niskiego progu akceptacji genów potomków. Proponuje się zmianę progu w zależności od numeru populacji, jak niżej:

$$Cp(k) = \frac{j}{J_{max}} Cp_0, \tag{4.3.17}$$

gdzie Cp_0 – początkowy próg wyboru współrzędnych potomków w krzyżowaniu.

W celu zachowania różnorodności populacji stosuje się regułę w mutacji, że jeśli nie zostanie wybrana współrzędna wektorów z populacji *PM*, to akceptuje się wymaganą liczbę ϱ losowo wybranych współrzędnych dla tego wektora.

W ewolucji różnicowej złożoność obliczeniowa procedury nadawania rang $O(n^2)$ nie jest większa od złożoności weryfikacji ograniczeń $O(n^7)$. Zatem złożoność metaheurystyki DEMCA odpowiada złożoności mutacji tabu search, która wywoływana jest zadaną liczbę razy. Dla rozważanego problemu optymalizacji wielokryterialnej algorytm DEMCA cechuje się złożonością obliczeniową $O(n^9)$, gdzie n = max $\{K, L, M, J_{max}, L\}$.

Do rozwiązania problemu polioptymalizacji (4.2.1) – (4.2.5) można zastosować również inne metaheurystyki, takie jak: Multi-objective Harmony Search (MOHS), Multi-objective Genetic Programming with Archive (MOGPA), Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization (NSGA-II) oraz Multiple-objective Particle Swarm Optimizers (OMOPSO) [83, 111]. Porównanie jakości wyznaczonych rozwiązań za pomocą algorytmu DEMCA z innymi metaheurystykami wykazało, że za pomocą DEMCA wyznacza się wyższej jakości rozwiązania niezdominowane niż za

pomocą pozostałych metaheurystyk. Wyniki adekwatnych eksperymentów numeryczną zostaną omówione w dalszej części rozprawy.

Możliwe jest zastosowanie zespołu solwerów klasy DEMCA, które różnią się progiem krzyżowania, współczynnikiem skalującym w mutacji, rodzajem mutacji różnicowej, procedurą poprawy rozwiązań niedopuszczalnych oraz procedurą nadawania rang. W zespołowej metodzie DEMCA-E (ang. *DEMCA in Ensemble*) można zastosować także algorytm ewolucji różnicowej do wyznaczenia parametrów i procedur metody.

Za pomocą algorytmu DEMCA wyznaczane są rozwiązania suboptymalne w sensie Pareto, co oznacza, że nie ma gwarancji, że wyznaczone rozwiązania są Pareto-optymalne dla każdej instancji. Jednakże w porównaniu do innych metaheurystyk można oczekiwać, że jakość wyznaczonych wariantów za pomocą ewolucji różnicowej będzie przewyższała jakość pozostałych alternatyw. Wyłania się jednak dylemat, którą migrację maszyn wirtualnych należy zastosować w zbiorze hostów. Do wyboru rozwiązania z reprezentacji zbioru Pareto można użyć dodatkowego kryterium, np. prawdopodobieństwo ukończenia pracy maszyn wirtualnych w terminach lub stopień dostępności chmury edukacyjnej. W demonstratorze chmury edukacyjnej wyznacza się rozwiązanie kompromisowe dla parametru p=2 za pomocą procedury wyboru z zadanej reprezentacji zbioru alternatyw Pareto-optymalnych $\boldsymbol{X}^{Pareto}$ zadania (4.2.1) – (4.2.5).

Współrzędne punktu idealnego wyznacza się następująco:

$$P_i^{inf} = \min_{x \in \boldsymbol{X}^{Pareto}} F_i(x),\ i = \overline{1,4}. \tag{4.3.18}$$

Do normalizacji przestrzeni ocen wykorzystuje się również współrzędne punktu nadir:

$$N_i^* = \max_{x \in \boldsymbol{X}^{Pareto}} F_i(x),\ i = \overline{1,4}. \tag{4.3.19}$$

Selekcja rozwiązania kompromisowego x^p dla zadanej wartości parametru p ze zbioru $\boldsymbol{X}^{Pareto}$ zachodzi dla najmniejszej wartości p-normy L_p [10]:

$$L_p(x^p) = \min_{x \in \boldsymbol{X}^{Pareto}} L_p(x). \tag{4.3.20}$$

Dla p=2 wartość $L_2(x^{p=2})$ jest najkrótszą odległością w sensie Euklidesa w znormalizowanej przestrzeni ocen, jak niżej:

$$L_2(x^{p=2}) = \min_{x \in \boldsymbol{X}^{Pareto}} \sqrt{\sum_{i=1}^{4} \left(\frac{F_i(x) - P_i^{inf}}{N_i^* - P_i^{inf}} \right)^2}. \tag{4.3.21}$$

Dla p=1 wartość $L_1(x^{p=1})$ jest minimalną sumą znormalizowanych odchyłek współrzędnych oceny $F(x^{p=1})$ od środka układu współrzędnych:

$$L_1(x^{p=1}) = \min_{x \in \boldsymbol{X}^{Pareto}} \sum_{i=1}^{4} \left(\frac{F_i(x) - P_i^{inf}}{N_i^* - P_i^{inf}} \right). \tag{4.3.22}$$

Natomiast dla $p \to \infty$ wartość $L_\infty(x^{p \to \infty})$ jest minimalną wartością największej odchyłki współrzędnej od środka układu współrzędnych:

$$L_\infty(x^{p \to \infty}) = \min_{x \in \boldsymbol{X}^{Pareto}} \max_{i=1,4} \left\{ \frac{F_i(x) - P_i^{inf}}{N_i^* - P_i^{inf}} \right\}. \tag{4.3.23}$$

Solwer w demonstratorze chmury edukacyjnej współpracuje z menedżerem rekonfigurującym RMC, który wspomaga rozmieszczenie maszyn wirtualnych i wymianę rodzajów hostów na podstawie rozwiązania kompromisowego.

4.4 Wnioski i uwagi

NP-trudny problem optymalizacji chmury obliczeniowej (4.2.1) – (4.2.5) dotyczy minimalizacji obciążenia CPU newralgicznego hosta w chmurze oraz obciążenia transmisją danych krytycznego węzła. Ponadto minimalizuje się koszt eksploatacji komputerów oraz moc poboru energii elektrycznej. Należy nadmienić, że wartości kryteriów mają ograniczenia dolne i górne. Ponadto rozważa się ograniczenia na pamięć operacyjną RAM oraz pamięć dyskową DM. Narzucono również ograniczenia dotyczące preferowania hostów w rozwiązaniu dopuszczalnym. Uwzględnia się wymagania dotyczące prawdopodobieństwa zakończenia pracy maszyn wirtualnych w terminach oraz dostępności chmury edukacyjnej. Czas niedostępności maszyn wirtualnych podczas migracji nie może być przekroczony. Rozważa się także ograniczenia formalne na konieczność przydziału wszystkich maszyn wirtualnych oraz alokację hostów.

Sformułowano oryginalne zagadnienie wyznaczenia reprezentacji rozwiązań optymalnych w sensie Pareto z czterema kryteriami. W zagadnieniu występują konflikty między sześcioma parami kryteriów. Przykładowo rozwiązania cechujące się mniejszym obciążeniem newralgicznych hostów mogą być bardziej obciążone pod względem transmisji danych w sytuacji, gdy maszyny wirtualne są bardziej rozproszone. Natomiast skupienie maszyn wirtualnych w hoście cechującym się mało wydajnymi procesorami, redukuje obciążenie transmisją danych, ale zazwyczaj zwiększa obciążenie procesorów tego hosta.

Wykazano, że zbiór rozwiązań optymalnych w sensie Pareto $\boldsymbol{X}_N^{\leqslant}(2) \subseteq \boldsymbol{X}$ dla dwóch wybranych kryteriów skalarnych F_1 oraz F_2 zawiera się w zbiorze rozwiązań sprawnych

$\boldsymbol{X}_N^{\lesseqgtr}(3) \subseteq \boldsymbol{X}$ dla trzech kryteriów skalarnych F_1, F_2 oraz F_3 dla zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$ oraz relacji dominowania $\breve{\succeq}_{\leq}$ w $\boldsymbol{R}^3$. Stwierdzono również, że zbiór rozwiązań optymalnych w sensie Pareto $\boldsymbol{X}_P^{\lesseqgtr}(I) \subseteq \boldsymbol{X}$ dla I ($I \geq 2$) kryteriów skalarnych zawiera się w zbiorze rozwiązań efektywnych $\boldsymbol{X}_P^{\lesseqgtr}(N+k) \subseteq \boldsymbol{X}$ dla $I+l$ kryteriów cząstkowych, l=1, 2, 3, …, zbioru $\boldsymbol{X}$, a także relacji dominowania $\breve{\succeq}_{\leq}$ w $\boldsymbol{R}^{I+l}$.

Do wyznaczania reprezentacji rozwiązań Pareto-optymalnych zagadnienia (4.2.1) – (4.2.5) opracowano wielokryterialny algorytm ewolucji różnicowej DEMCA, w którym zakodowano chromosom tak, aby zredukować liczebność elementów w przestrzeni przeszukiwań i spełnić ograniczenia formalne. Zastosowano również procedurę nadawania rang Fonseci-Fleminga wraz z geometryczną miarą zagęszczenia w wypadku rozwiązań z aktualnej populacji. Nowością jest procedura nadawania rang w oparciu o funkcje kar za niespełnienie ograniczeń dla rozwiązań niedopuszczalnych. Algorytm ewolucji różnicowej współpracuje z metaheurystką przeszukiwania tabu, którą zastosowano jako dodatkową mutację rozwiązań. Mutacja tabu search znacząco redukuje wartość obciążenia transmisją danych newralgicznego węzła, co powoduje najwięcej kłopotów w optymalizacji za pomocą algorytmu ewolucji różnicowej.

Dla rozważanego problemu optymalizacji wielokryterialnej algorytm DEMCA cechuje się złożonością obliczeniową $O(n^9)$, gdzie n = max $\{K, M, N, J_{max}, L\}$. Do wyznaczania rozwiązania optymalnych w sensie Pareto można zastosować również inne metaheurystyki, takie jak: MOHS, MOGPA, NSGA-II oraz OMOPSO. Porównanie jakości wyznaczonych rozwiązań za pomocą algorytmu DEMCA z tymi metaheurystykami wykazało, że algorytm DEMCA wyznacza wyższej jakości warianty niezdominowane niż wspomniane metaheurystyki.

W zespołowej metodzie DEMCA-E rozważa się zespół solwerów DEMCA, które różnią się progiem krzyżowania, współczynnikiem skalującym w mutacji, rodzajem mutacji różnicowej, procedurą poprawy rozwiązań niedopuszczalnych oraz procedurą nadawania rang. W zespołowej metodzie można zastosować także algorytm ewolucji różnicowej do wyznaczenia parametrów i procedur algorytmów DEMCA.

Aby podjąć decyzję, które przeprowadzić migracje maszyn wirtualnych w zaprojektowanym demonstratorze chmury edukacyjnej wyznacza się rozwiązanie kompromisowe dla parametru p=2 za pomocą wyboru z reprezentacji zbioru alternatyw Pareto-optymalnych. Zaimplementowany solwer w demonstratorze współpracuje z menedżerem rekonfiguracyjnym, który wspomaga rozmieszczenie maszyn wirtualnych i dobór rodzajów komputerów w chmurze obliczeniowej.

5. EKSPERYMENTY NUMERYCZNE

Model i metodę optymalizacji zasobów edukacyjnej chmury obliczeniowej zweryfikowano eksperymentalnie ze względu na jakość wyznaczonych rozwiązań oraz możliwości implementacji demonstratora chmury edukacyjnej, wykorzystując hosty gridu Comcute na Politechnice Gdańskiej oraz komputery na Politechnice Warszawskiej. Uruchomiono laboratoryjną wersję demonstratora chmury na platformie OpenStack. Dokonano również obliczeń za pomocą algorytmu DEMCA na superkomputerze Prometheus w chmurze PL-Grid. W celu weryfikacji i powtórzenia obliczeń przez innych badaczy umieszczono na stronie WWW rozprawy opracowane zestawy danych źródłowych [154], które również dostępne są na załączonej płycie DVD.

5.1 Rekomendacje wartości metaparametrów ewolucji różnicowej

W literaturze przedmiotu rekomenduje się w optymalizacji jednokryterialnej kluczowe metaparametry ewolucji różnicowej o wartościach: Cp=1 i $q \in [0{,}4; 1]$ [133]. Celem pierwszego eksperymentu jest wyznaczenie rekomendowanych wartości współczynnika skalującego q oraz progu akceptacji współrzędnych potomków w krzyżowaniu Cp na podstawie rozwiązań dla instancji B90 zagadnienia optymalizacji wielokryterialnej (4.2.1) – (4.2.5), której dane zapisano w pliku benchmark90.txt [154]. Przyjęto następujące wartości ograniczeń: $F_1 = 25$ [s], $F_2 = 222$ [s], $F_3 = 16\,000$ [wat] oraz $F_4 = 80\,000$ [$].

Instancja cechuje się 90 binarnymi zmiennymi decyzyjnymi, a przestrzeń przeszukiwań dla kodowania zero-jedynkowego rozwiązań zawiera $1{,}24 * 10^{27}$ elementów. W wyniku kodowania całkowitoliczbowego rozwiązań liczebność przestrzeni przeszukiwań redukuje się do $1{,}59 * 10^{12}$ elementów.

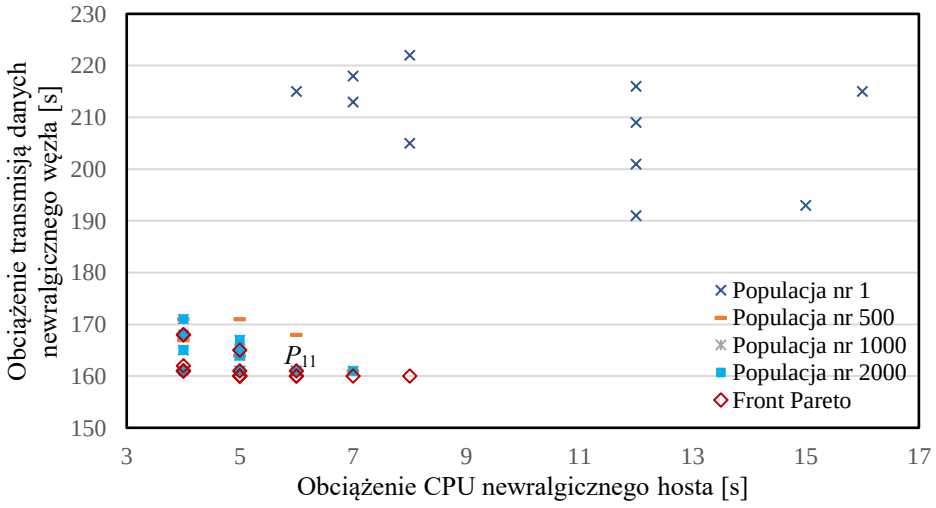
Współczynnik q zwiększano od 0,1 do 1 z przyrostem 0,1, a następnie dla $q > 1$ – o 1. Próg akceptacji współrzędnych potomków Cp modyfikowano od 0,1 do 1 o 0,1. Wykonano po 30 symulacji dla każdej pary parametrów. Liczebność populacji wynosiła 60. Przyjęto, że maksymalna liczba konstruowanych populacji to 2000, co wymagało 63 sekund obliczeń na komputerze DELL Latitude E7470 z procesorem dwurdzeniowym Intel Core i5-6300U 2,5GHz. Wyznaczono 16 rozwiązań niezdominowanych (tab. 5). W przestrzeni znormalizowanej rozwiązaniem kompromisowym dla p=2 jest wariant numer 10 o odległości L_2=0,435675 od oceny idealnej $y^{\inf}$.

Tabela 5 Zbiór rozwiązań Pareto-optymalnych dla instancji B90.

Lp.	X^{VM}	X^{H}	$F_1[s]$	$F_2[s]$	F_3 [\$]	F_4 [W]	L_2
1	[1;6;4;5;1;2;4;3;3;6;2;5]	[1;1;3;1;3;3]	4	161	7500	15600	0,931502
2	[1;6;4;5;1;2;4;3;3;6;2;5]	[1;1;1;1;3;3]	4	161	9000	14800	0,631329
3	[1;6;4;5;3;2;4;1;5;6;2;3]	[1;1;2;1;2;3]	4	161	30500	14400	1,108442
4	[1;6;4;5;2;2;4;3;5;6;3;1]	[1;1;11;2;3]	4	162	20500	14200	0,727942
5	[6;6;4;3;1;1;4;5;5;2;2;3]	[1;1;3;1;1;1]	4	168	10500	14000	1,054366
6	[6;6;4;5;1;1;4;3;5;2;2;3]	[1;1;1;1;2;1]	4	168	22000	13400	1,184637
7	[5;6;6;1;1;3;5;4;4;2;3;6]	[2;1;3;1;3;3]	5	160	17500	15800	1,118721
8	[5;6;6;1;1;3;5;4;4;2;3;6]	[2;1;3;1;1;3]	5	160	19000	15000	0,889826
9	[5;6;6;1;1;3;5;4;4;2;3;6]	[2;1;1;1;1;3]	5	160	20500	14200	0,727942
10	[5;3;6;5;6;1;4;2;2;3;1;4]	[1;1;3;1;1;1]	**5**	**161**	**10500**	**14000**	**0,435675**
11	[3;6;6;3;2;2;5;1;4;4;1;5]	[1;1;1;1;1;1]	5	165	12000	13200	0,701003
12	[5;6;6;1;1;3;5;4;4;2;3;6]	[3;1;3;1;1;3]	6	160	7500	15600	1,049796
13	[5;6;6;1;1;3;5;4;4;2;3;6]	[3;1;1;1;1;3]	6	160	9000	14800	0,795583
14	[1;6;4;5;3;2;4;1;5;6;2;3]	[1;1;1;1;1;1]	6	161	12000	13200	0,551276
15	[5;6;6;1;1;3;5;4;4;2;3;6]	[1;1;1;1;1;3]	7	160	10500	14000	0,821089
16	[5;6;6;1;1;3;5;4;4;2;3;6;]	[1;1;1;1;1;1]	8	160	12000	13200	1,018960

Źródło: opracowanie własne.

Na rysunku 29 przedstawiono 10 ocen dotyczących 16 rozwiązań Pareto-optymalnych. Wybrane rozwiązania niezdominowane cechują się identycznymi wartościami kryteriów F_1, F_2. Ocena P_{11}=(5 [s], 165 [s], 12 000 [zł], 13 200 [W]) jest zdominowana przez cztery oceny pod względem kryteriów F_1, F_2, ale ze względu na najniższy pobór mocy F_4 jest oceną Pareto-optymalną (tab. 5). W rezultacie dwuelementowy zbiór ocen efektywnych dla kryteriów F_1, F_2 został powiększony w wypadku czterech kryteriów F_1, F_2, F_3, F_4 do dziesięciu elementów.



Rysunek 29 Migracja ocen lokalnie niezdominowanych w kierunku brzegu Pareto na wybranych etapach przebiegu algorytmu DEMCA dla q=1, Cp=0,1.
Źródło: opracowanie własne.

W tabeli 6 przedstawiono uśrednioną procentową odległość od brzegu Pareto wyznaczonego za pomocą algorytmu DEMCA dla wybranych strategii zmian metaparametrów.

Oceny Pareto-optymalne uzyskano za pomocą zmodyfikowanej metody przeglądu (tab. 6). Dla wartości $Cp = 0{,}1$ i $q = 1$ algorytm działał najbardziej efektywnie, wyznaczając zbiór Pareto za każdym przebiegiem. Pary ($Cp = 0{,}6$; $q = 0{,}6$) i ($Cp = 0{,}9$; $q = 0{,}6$), również umożliwiają osiągnięcie wyników cechujących się uśrednioną procentową odległością od brzegu Pareto na poziomie 0,98%, przy czym zbieżność do tych ocen jest nieco wolniejsza niż dla pary (Cp = 0,1; q = 1).

Zweryfikowano trzy strategie modyfikacji metaparametrów. Pierwsza strategia $\Delta TCp{-}_{400}$ polega na zmniejszaniu wartości Cp o połowę co 400 iteracji, przy czym wartości początkowe to Cp = 0,9 oraz q = 1. Druga strategia $\Delta Tq{-}_{400}$ wykorzystuje redukcję współczynnika skalowania q = 0,9 oraz $Cp = 0{,}1$. W trzeciej strategii $\Delta Tq\ {+}_{400}$, zaczynając od q = 0,2 zwiększa się wartość q dwukrotnie przy zadanym $Cp = 0{,}1$. Natomiast strategia $\Delta Tq{-}_{400}\ \Delta TCp{-}_{400}$ polega na zmniejszaniu wartości q i Cp o połowę dla początkowych wartości Cp = 0,9 i q = 0,9. Z kolei w strategii $\Delta Tq\ {+}_{400}\ \Delta TCp{-}_{400}$ wartość parametru q podwaja się, a Cp zmniejsza o połowę dla początkowych wartości Cp = 0,9 i q = 0,2.

Tabela 6 Uśredniona procentowa odległość rozwiązań wyznaczonych za pomocą algorytmu DEMCA od brzegu Pareto dla wybranych strategii zmian metaparametrów Cp i q.

Nazwa strategii zmian wartości metaparametrów	Cp	q	Procentowa odległość od brzegu Pareto w zależności od numeru populacji		
			100	1000	2000
$\Delta TCp{-}_{400}$	0,9	0,1	36,76%	3,02%	2,19%
$\Delta Tq{-}_{400}$	0,1	0,9	6,08%	3,62%	3,29%
$\Delta Tq\ {+}_{400}$	0,1	0,2	13,99%	7,53%	4,62%
$\Delta Tq{-}_{400}\ \Delta TCp{-}_{400}$	0,9	0,9	8,72%	5,90%	4,01%
$\Delta Tq\ {+}_{400}\ \Delta TCp{-}_{400}$	0,9	0,2	7,48%	3,96%	2,29%
DE/rand/1	**0,1**	0,2	10,05%	10,82%	10,25%
		0,4	7,83%	10,39%	10,04%
		0,6	2,49%	1,53%	1,48%
		0,8	2,20%	1,62%	1,43%
		1,0	2,18%	1,29%	**0,98%**
	0,2	1,0	2,73%	1,05%	1,05%
	0,3	0,8	3,82%	1,39%	1,48%
	0,4	0,8	3,49%	1,38%	1,27%
	0,5	1,0	6,34%	1,19%	1,13%
	0,6	**0,6**	2,97%	1,48%	**0,98%**
	0,7	1,0	7,30%	2,64%	1,56%
	0,8	0,6	4,17%	4,26%	3,80%
	0,9	**0,6**	4,55%	1,36%	**0,98%**
	1,0	0,8	7,02%	5,87%	2,35%
	0.6	2.0	10,08%	1,70%	1,45%
	0,7	3.0	8,42%	7,64%	7,64%

Źródło: opracowanie własne.

5.2 Porównanie algorytmu DEMCA z innymi metaheurystykami

Celem drugiego eksperymentu jest porównanie efektywności algorytmu DEMCA w odniesieniu do innych metaheurystyk [83, 111]. Rozważa się problem optymalizacji dwukryterialnej polegający na minimalizacji obciążenia procesorów w newralgicznym hoście oraz na minimalizacji obciążenia transmisją danych krytycznego węzła [82]. Uwzględnia się średnią odległość wyznaczonych wyników sprawnych do frontu Pareto. Niech $d_s(y_l, \omega, k)$ to odległość Euklidesa między *l*-tą oceną Pareto-optymalną y_l a najbliższą oceną niezdominowaną ω wyznaczoną przez *s*-tą metaheurystykę w populacji nr *k*. Średnią odległość wyznaczonych ocen przez *s*-ty algorytm od frontu Pareto można oszacować, jak niżej:

$$Con_s(j) = \sum_{l=1}^{L} \min_{\omega \in \Omega_m} d_s(y_l, \omega, k)\,,\ s = \overline{1,S},\ j = \overline{1,J_{max}}, \qquad (5.2.1)$$

gdzie:
Ω_s – zbiór ocen efektywnych wyznaczonych za pomocą *s*-tego algorytmu,
J_{max} – maksymalna liczba populacji.

Jeśli ω jest najbliższa w sensie odległości Euklidesa do y_l, to nie jest rozpatrywana w wypadku innej oceny y_m. Jeśli $Con_s(j) = 0$, to wszystkie wyniki Pareto-optymalne wyznaczono za pomocą *s*-tego algorytmu w *j*-tej iteracji.

Instancja B306 cechuje się 306 zero-jedynkowymi zmiennymi decyzyjnymi, a binarna przestrzeń przeszukiwań zawiera $1{,}3 * 10^{92}$ elementów. Liczba wariantów, które spełniają warunki formalne wynosi $7{,}9 * 10^{31}$, przy czym nie wszystkie z nich są dopuszczalne. Dane wejściowe udostępniono w pliku benchmark306.txt [154].

Obliczenia przeprowadzono na komputerze DELL Inspiron 17 5000 (procesor czterordzeniowy Intel Core i7-7500U 2,7 GHz, 16 GB pamięci RAM). Przyjęto, że liczebność populacji wynosi 60, współczynnik skalujący $q = 1$, a $Cp = 0{,}1$. Algorytm MOGPA opierający się na programowaniu genetycznym wyznaczył zbiór rozwiązań niezdominowanych cechujący się ocenami bliższymi ocenom Pareto-optymalnym w sensie średniej odległości niż zbiór wariantów wyznaczony za pomocą algorytmu poszukiwania harmonii MOHS [83, 111]. Algorytmowi MOGPA obliczenia zajęły na podobnej klasie komputerze ok. 50 minut, a algorytmowi MOHS – ok. 95 minut.

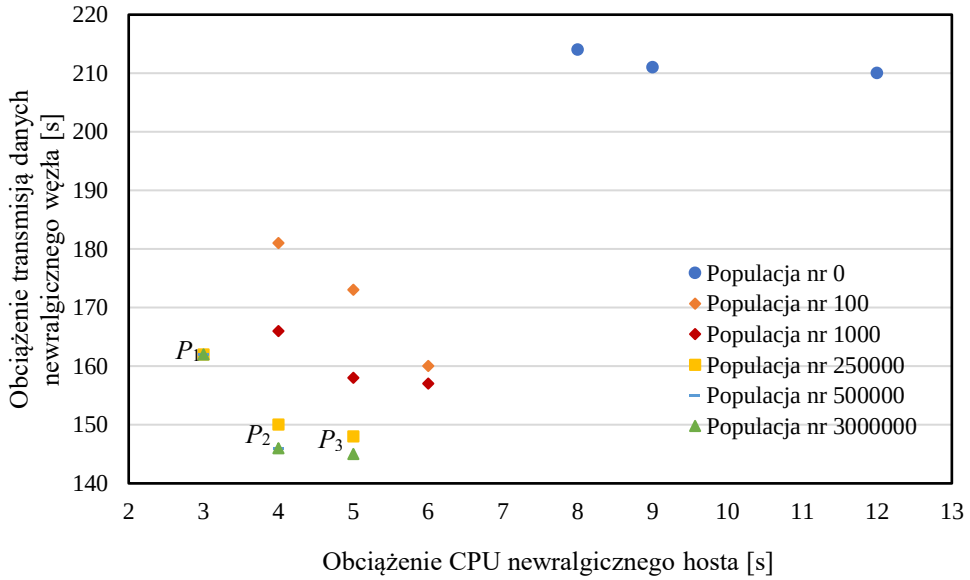
Na rysunku 30 przedstawiono migrację rozwiązań niezdominowanych dla algorytmu DEMCA, który w przestrzeni dwukryterialnej związanej z funkcjami F_1, F_2 wyznaczył zbiór $\{P_1 = (3\,[s]; 162\,[s]),\ P_2 = (4\,[s]; 146\,[s]),\ P_3 = (5\,[s]; 145[s])\}$ (tab. 7). Na rysunku 31 zobrazowano oceny rozwiązań wyznaczone również za pomocą innych

metaheurystyk: MOHS [111], MOGPA [82], NSGA-II [7] i OMOPSO [83, 111]. Punkt P_2 dominuje oceny wyznaczone przez inne metaheurystyki.

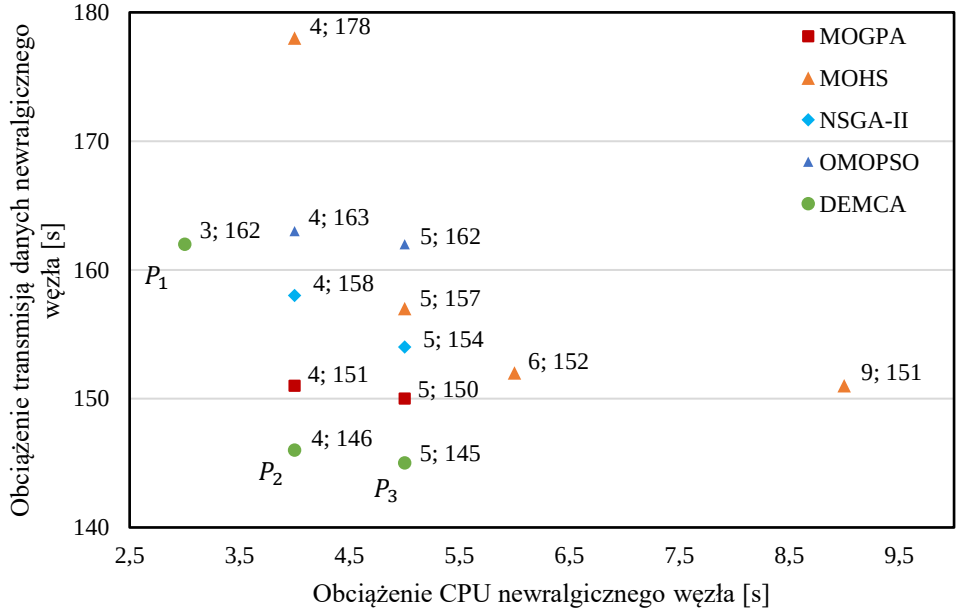
Tabela 7 Specyfikacja rozwiązań i ocen sprawnych wyznaczonych na pomocą algorytmu DEMCA.

P	X^{MV}	X^{H}	F_1 $[s]$	F_2 $[s]$	F_3 [\$]	F_4 $[W]$	L_2
1	[6;1;7;8;2;6;3;9;1;7;2;5;4;4;8;4;3;5;9;8;7;6;2;5]	[2;7;10;10;6;10;8;10;9]	3	162	46848	14450	1,19
2	[9;4;5;2;4;6;8;7;2;1;9;9;5;5;3;3;6;1;7;8;3;6;4;5]	[10;10;10;10;9;9;7;10;9]	4	146	55572	16700	1,50
3	[6;1;3;2;8;5;2;4;1;9;7;7;3;3;8;7;8;6;9;5;4;3;6;9]	[7;5;9;9;6;10;8;9;3]	5	145	40324	12300	1,00

Źródło: opracowanie własne.

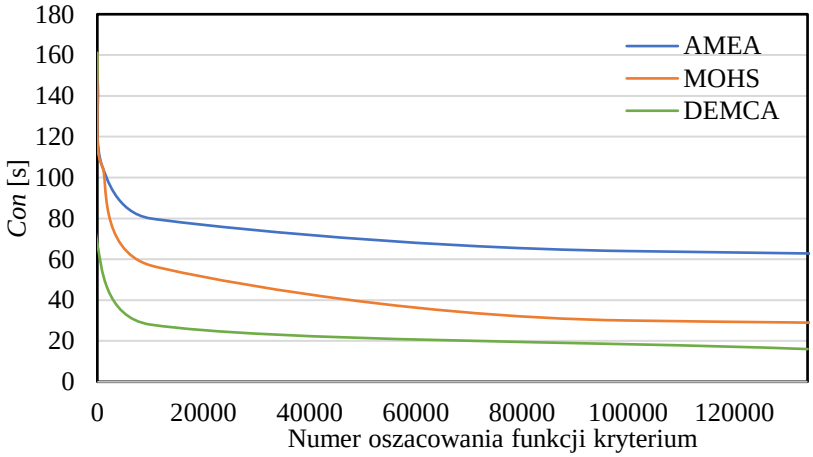


Rysunek 30 Zbieżność ocen rozwiązań niezdominowanych wyznaczonych za pomocą ewolucji różnicowej dla instancji B306.
Źródło: opracowanie własne.



Rysunek 31 Oceny rozwiązań wyznaczone przez wybrane metaheurystyki dla instancji B306.
Źródło: opracowanie własne.

Niech $P_1 = F(A)$, $P_2 = F(B)$, $P_3 = F(C)$. Wówczas $\eta(A) = 255$, $\eta(B) = 254$, $\eta(C) = 254$, co spełnia ograniczenie (5.2.3) dla $\eta_{min} = 200$. Na rysunku 32 przedstawiono uśrednioną zbieżność wyników wyznaczonych za pomocą algorytmu MOHS, adaptacyjnego algorytmu ewolucyjnego AMEA [16] oraz algorytmu DEMCA w oparciu o minimalizację średniej odległości od frontu Pareto dla 30 przebiegów.



Rysunek 32 Uśredniona zbieżność ocen dla wybranych wielokryterialnych metaheurystyk. Źródło: opracowanie własne.

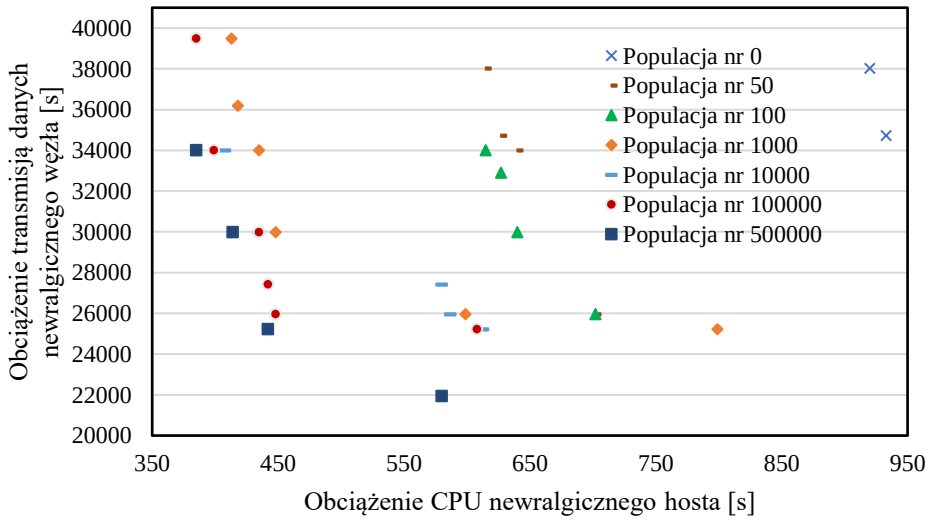
5.3 Modernizacja infrastruktury chmury obliczeniowej

Eksperyment trzeci polegał na przygotowaniu alokacji maszyn wirtualnych i zasobów dla modernizowanej infrastruktury systemu laboratoryjnego Comcute, który eksploatowano na Politechnice Gdańskiej w latach 2012–2021. Rozważa się zwiększenie liczby hostów obliczeniowych z 9 do 15 oraz maszyn wirtualnych z 6 do 45 [111]. Niech koszt zakupu komputerów nie będzie większy niż $F_3^{max} = 350\,000$ zł, a górną granicą poboru mocy elektrycznej będzie $F_4^{max} = 25$ kW. Nałożono również ograniczenie na minimalny stopień rozproszenia $\eta_{min} \geq 500$ [154].

Na wybranym hoście typu H_{11} zmierzono skumulowany czas obciążeń procesorów, który w wypadku maszyny wirtualnej typu *W* wyniósł 320,288 [s], a w wypadku maszyny wirtualnej typu *S* – 306,830 [s]. Pozostałe elementy macierzy T oszacowano za pomocą testu CPU Mark. Skumulowany czas transmisji danych od maszyny klasy *W* do maszyny klasy *S* na hoście typu H_{11} to 27,602 [s]. Natomiast skumulowany czas komunikacji od maszyny klasy *S* do maszyny klasy *W* – 337,926 [s]. Dane komunikacji między maszynami posłużyły do wyznaczenia macierzy τ. Niech dla minimalizowanych kryteriów F_1 i F_2 oraz zadanych punktów y^{inf} *i* y^{sup} należy wyznaczyć rozwiązanie kompromisowe $x^{p=2}$ (4.3.21).

W instancji B855 binarna przestrzeń przeszukiwania zawiera $2{,}4 * 10^{257}$ elementów. Natomiast przestrzeń przeszukiwań dla zakodowanych rozwiązań zawiera $1{,}3 * 10^{69}$ elementów. W wypadku kodowania całkowitoliczbowego jest zaledwie 60 zmiennych decyzyjnych zamiast 855. Przyjęto, że maksymalna liczba konstruowanych populacji to 500 000, co zajęło ok. 180 minut na komputerze DELL Inspiron 17 5000.

Za pomocą algorytmu MOHS wyznaczono $0{,}9 * 10^{6}$ aktualizacji tablicy rozwiązań (tablica wariantów odpowiada populacji), a za pomocą algorytmu MOGPA i pozostałych algorytmów maksymalna liczba populacji wynosiła 500 000. W algorytmie DEMCA zwiększono liczbę osobników w populacji do 60. Na rysunku 33 przedstawiono migrację ocen lokalnie niezdominowanych. Wraz ze wzrostem liczby rozwiązań widoczna jest poprawa jakości ocen aż do populacji nr 500 000. Wyznaczono cztery punkty należące do zbioru Pareto (rys. 33), których specyfikację przedstawiono w tab. 8.



Rysunek 33 Oceny niezdominowane na wybranych etapach ewolucji różnicowej dla instancji B855.
Źródło: opracowanie własne.

Tabela 8 Oceny rozwiązań sprawnych wyznaczonych za pomocą algorytmu DEMCA dla instancji B855.

Lp.	X^{VM}	X^{H}	F_1 [s]	F_2 [s]	F_3 [$]	F_4 [W]	L_2
1	[5; 13; 6; 5; 4; 7; 3; 4; 2; 7; 9; 11; 6; 10; 8; 1; 9; 12; 12; 12; 10; 2; 15; 14; 3; 8; 14; 9; 1; 15; 2; 15; 14; 8; 10; 14; 3; 11; 12; 1; 1; 13; 11; 13; 15]	[3; 3; 10; 10; 4; 3; 6; 7; 10; 3; 3; 3; 10; 3; 3]	580	21931	349578	19915	1,50
2	[12; 15; 14; 8; 6; 9; 11; 7; 7; 1; 5; 3; 10; 4; 13; 1; 6; 9; 1; 4; 12; 5; 9; 11; 14; 2; 10; 14; 1; 13; 15; 8; 6; 15; 15; 12; 12; 10; 4; 2; 13; 11; 8; 10; 9]	[8; 4; 2; 3; 4; 3; 4; 7; 10; 9; 3; 10; 3; 3; 8]	**442**	**25221**	**343563**	**20450**	**1,08**
3	[6; 14; 4; 6; 14; 15; 11; 2; 7; 9; 13; 12; 5; 13; 3; 10; 15; 3; 2; 10; 15; 5; 10; 7; 1; 15; 12; 6; 5; 2; 4; 3; 3; 6; 1; 15; 10; 5; 2; 2; 8; 5; 3; 10; 9]	[3; 10; 9; 5; 10; 9; 3; 11; 3; 10; 12; 3; 3; 3; 8]	414	29973	349189	19380	1,16
4	[2; 6; 4; 7; 10; 6; 11; 1; 7; 9; 2; 12; 5; 13; 5; 10; 15; 3; 2; 2; 15; 5; 6; 1; 15; 14; 12; 6; 4; 10; 10; 3; 2; 14; 5; 15; 10; 3; 13; 3; 8; 5; 3; 6; 9]	[3; 10; 9; 3; 10; 10; 3; 11; 3; 10; 12; 3; 3; 3; 8]	385	33994	347846	19760	1,28

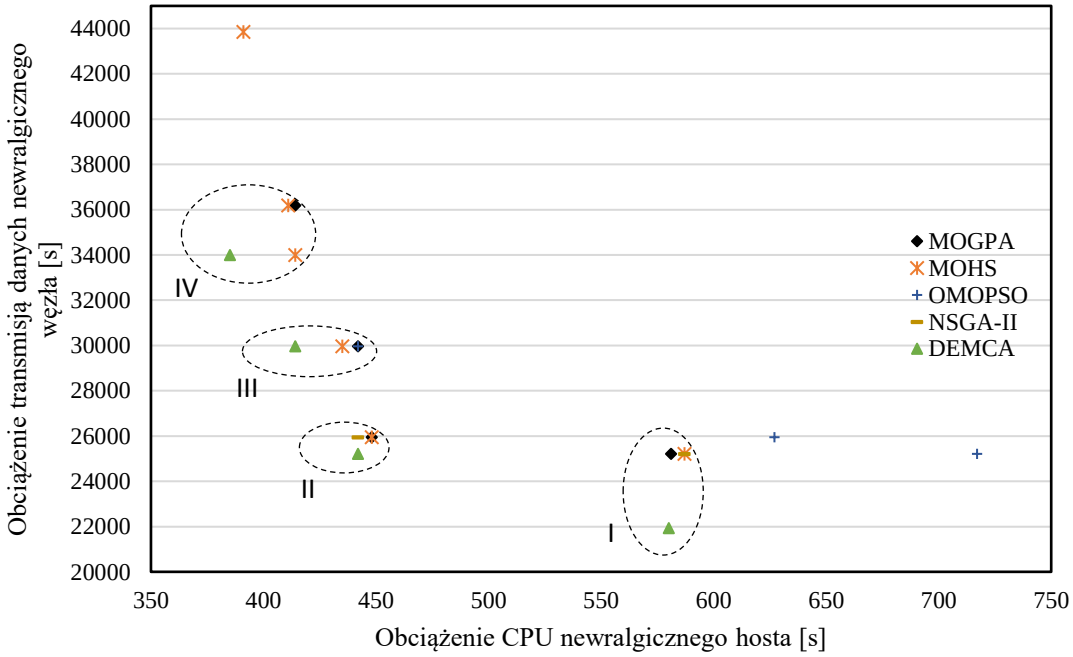
Źródło: opracowanie własne.

Rozwiązania wyznaczone za pomocą algorytmu DEMCA porównano z alternatywami otrzymanymi za pomocą algorytmów: MOHS, MOGPA, NSGA-II i OMOPSO (rys. 34, tab. 9). Zaobserwować można, że za pomocą algorytmu DEMCA wyznaczono punkty, które dominują oceny uzyskane za pomocą pozostałych algorytmów.

Tabela 9 Oceny wyznaczone za pomocą algorytmów DEMCA, MOGPA, MOHS, NSGA-II i OMOPSO.

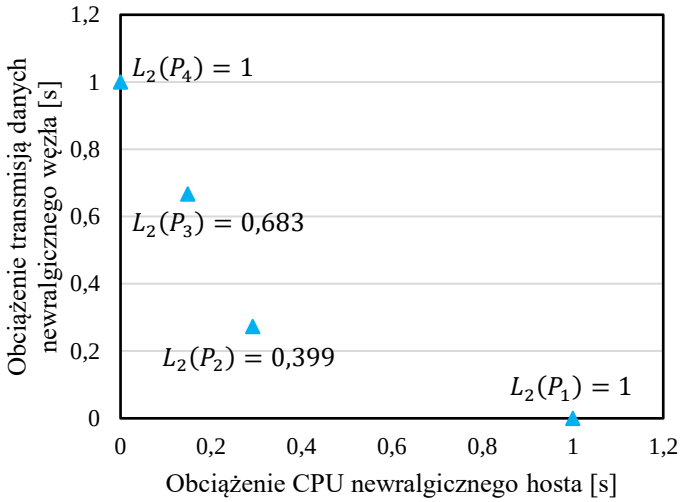
Obszar	Ocena rozwiązań $F(x) = [F_1(x), F_2(x)]^T$				
	DEMCA	MOGPA	NSGA-II	OMOPSO	MOHS
I	P_1: (580; 21931)	(581; 25221)	(587; 25221)	–	(587; 25221)
II	P_2: (442; 25221)	(448; 25952)	(442; 25952)	–	(448; 25952)
III	P_3: (414; 29973)	(442; 29973)	–	(442; 29973)	(435; 29973)
IV	P_4: (385; 33994)	(414; 36187)	–	–	(411; 36187)

Źródło: opracowanie własne na podstawie [83].



Rysunek 34 Porównanie wyników wyznaczonych za pomocą algorytmu DEMCA z ocenami otrzymanymi za pomocą innych metaheurystyk.
Źródło: opracowanie własne.

W celu modernizacji rozszerzenia chmury edukacyjnej, należy wybrać rozwiązanie kompromisowe ze zbioru Pareto w przestrzeni znormalizowanej (rys. 35). Oceny rozwiązania idealnego $y^{inf} = (385[s]; 21931[s])$ i punktu nadir $y^{sup} = (580[s]; 33994[s])$ wyznaczono na podstawie wartości kresu dolnego i kresu górnego zbioru ocen rozwiązań niezdominowanych. Punkt P_2 cechuje się najkrótszą odległością od punktu idealnego w znormalizowanej przestrzeni kryterialnej, przy czym odległość $L_2(P_2) = 0{,}399$.



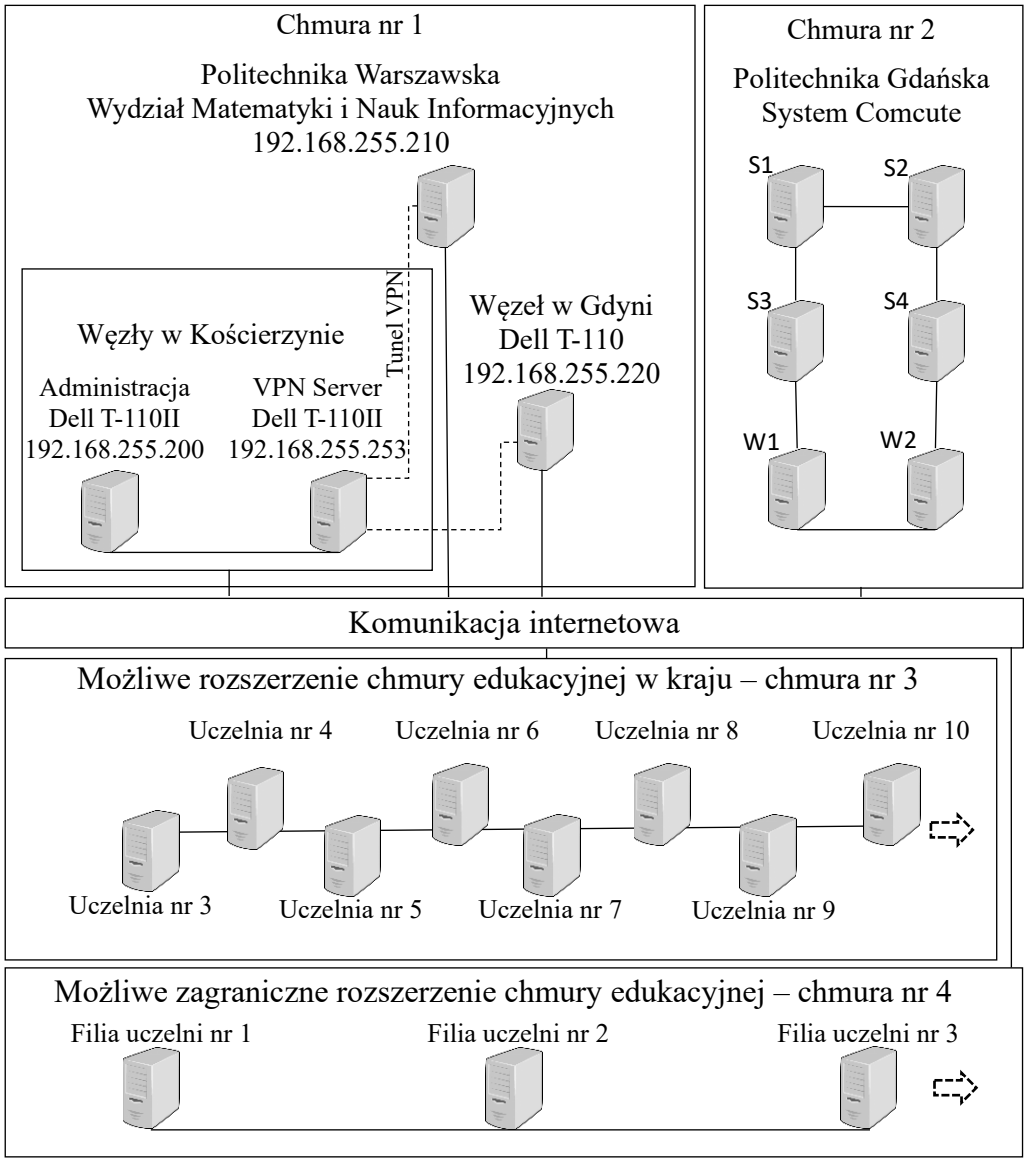
Rysunek 35 Wybór rozwiązania kompromisowego ze zbioru Pareto po normalizacji względem punktu idealnego i punktu nadir.
Źródło: opracowanie własne.

5.4 Projekt rozbudowy demonstratora chmury edukacyjnej

Celem eksperymentu czwartego jest wyznaczenie rozwiązania kompromisowego za pomocą solwera DEMCA w celu rozbudowy demonstratora chmury edukacyjnej. Rozważa się połączenie dwóch chmur z platformami OpenStack. Pierwsza obejmowała dwa komputery umiejscowione na Wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej, a także komputery w Gdyni i w Kościerzynie, a druga zlokalizowana była w laboratorium Katedry Architektury Systemów Komputerowych Wydziału Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej.

Do komunikacji między chmurami zastosowano protokoły HTTPS oraz WebSocket. Chmura edukacyjna nr 1 składała się z trzech hostów połączonych za pomocą wirtualnej sieci prywatnej z protokołem OpenVPN (rys. 36). Umożliwiło to dołączenie węzłów z sieci lokalnych wykorzystujących translację adresów sieciowych NAT (ang. *Network Address Translation*). W demonstratorze chmury edukacyjnej zainstalowano system Moodle w wersji 3.7, przy czym moduł webowy VM_1 współpracował z serwerem Apache w wersji 2.0 z interpreterem języka PHP w wersji 7.3. Natomiast moduł dostępu do danych VM_2 obejmował serwer bazodanowy MariaDB w wersji 10.4.8.

Chmura edukacyjna nr 2 składała się z 6 węzłów, w których zainstalowano serwery klasy DELL E5640. Serwery S1, S2, S3 i S4 posiadały 24 GB pamięci RAM, a serwery W1 i W2 były wyposażone w 48 GB pamięci RAM (rys. 36). Ponadto jednostka centralna każdego serwera składa się z dwóch procesorów Intel Xeon E-5640 2,66 GHz, które miały po 4 rdzenie. Natomiast wspólna macierz dyskowa serwerów miała wielkość 18 TB.



Rysunek 36 Rozlokowanie hostów demonstratora chmury edukacyjnej oraz możliwe rozszerzenia. Źródło: opracowanie własne.

Demonstrator chmury edukacyjnej nazwano GUT-WUT (ang. *Gdańsk University of Technology - Warsaw University of Technology*). Dobór hostów dotyczył serwerów z tabeli 10. Inteligentne agenty zaimplementowano z wykorzystaniem mikroserwisów z formatem wymiany danych JSON (ang. *JavaScript Object Notation*). Sześć agentów do predykcji oceny studenta na koniec semestru w oparciu o model sztucznej sieci neuronowej Sayeda-Bakera umieszczono w maszynach wirtualnych: VM_3, VM_4, VM_{21}, VM_{22}, VM_{39}, VM_{40}. Również sześć agentów, które adaptują treści nauczania do preferencji studenta za pomocą modelu Gonga-Wanga stosującego SVM, rozmieszczono w maszynach: VM_5, VM_6, VM_{23}, VM_{24}, VM_{41}, VM_{42}. Agenty do szacowania ocen wykładowcy za pomocą ankiet usytuowano w maszynach wirtualnych: VM_7, VM_8, VM_{25}, VM_{26}, VM_{43}, VM_{44}. Agenty do rekomendacji uczącym się aktywności edukacyjnych zainstalowano

w: VM_9, VM_{10}, VM_{27}, VM_{28}, VM_{45}, VM_{46}. Agenty typu asystent studentów do personalizowania materiału dydaktycznego za pomocą modelu Kolekar-Sanjeevi-Bormane rozlokowano w: VM_{11}, VM_{12}, VM_{29}, VM_{30}, VM_{47}, VM_{48}.

Tabela 10 Parametry wybranych rodzajów serwerów.

Nr n	Nazwa serwera H_n	Specyfikacja procesorów	ram_n – wielkość RAM [GB]	dm_n – wielkość dysku [TB]	β_n – max moc zasilania [W]	c_n – koszt komputera [zł] / liczba dostępnych maszyn*	Wydajność **
1	DELL E5640 v1	2xIntel Xeon E-5640 (4-core, 2.66GHz)	7	0,16	900	0 / 4	5250
2	DELL E5640 v2	2xIntel Xeon E-5640 (4-core , 2,66GHz)	7	0,16	900	0 / 2	5250
3	DELL T110 II	Intel Xeon E3-1220V2 (4-core, 3,10GHz)	6	0,25	305	0 / 1	6753
4	DELL T110	Intel Xeon X3430 (4-core, 2.40 GHz)	4	0,25	305	0 / 1	3339
5	Fujitsu Amilo Notebook Pi 3660	Intel Pentium Dual-Core T4300 (2-core, 2.1 GHz)	4	0,32	90	0 / 1	1232
6	Dell PowerEdge R740	Intel Xeon Silver-4110 (3.0GHz, 8-core)	16	0,12	1500	10922 / > I	11689
7	Dell PowerEdge R240	Intel Xeon E3-1240 (3.3GHz, 4-core)	8	1	250	3773 / > I	7980
8	HPE ProLiant DL360 Gen10 6130 125W 2P 64G-2R	Intel Xeon Gold 6130 (16-core, 2.1 GHz)	64	3,2	1600	69729 / > I	19643
9	HPE ProLiant DL360 Gen10	Intel Xeon Silver 4110 (8-core, 2.1 GHz)	16	2	1000	8550 / > I	11689
10	RX2520M5 v4208 Fujitsu	Intel Xeon Silver 4208 (8-core, 2.1 GHz)	16	10	450	14491 / > I	13116
11	RX2520M5 v5218 Fujitsu	Intel Xeon Gold 5218 (16-core, 2.3 GHz)	32	50	900	28797 / > I	17384
12	Inspur Server SNF5280M5003	Intel Xeon Silver 4208 (8-core, 2.1GHz)	32	100	800	32501 / > I	17904

* Zerowy koszt oznacza, że host dostępny jest bezpłatnie w modernizowanej chmurze
** wg testu *CPU Mark, źródło:* [159]

Źródło: opracowanie własne.

Agenty do asystowania wykładowcy, które wybierają pytania od studentów na podstawie modelu Susnea umieszczono w maszynach: VM_{13}, VM_{14}, , VM_{31}, VM_{32}, VM_{49}, VM_{50}. Agenty do analizy zainteresowania studenta materiałem i dostosowania materiału za pomocą modelu Romero-Bentura-Hervás-Gonzáleza rozmieszczono w: VM_{15}, VM_{16}, VM_{33}, VM_{34}, VM_{51}, VM_{52}. Agenty adaptujące treści przeglądanego materiału do możliwości smartfonu, oparte na modelu ISABELE, ulokowano w: VM_{17}, VM_{18}, VM_{35}, VM_{36}, VM_{53}, VM_{54}. Agenty do przygotowania egzaminu na podstawie modelu Wanga zainstalowano w maszynach: VM_{19}, VM_{20}, VM_{37}, VM_{38}, VM_{55}, VM_{56}.

Uruchomienie maszyny wirtualnej na komputerze w wybranym węźle następuje po wykonaniu komendy *openstack server migrate 'id maszyny wirtualnej' 'nazwa docelowego węzła'*. Alternatywnie możliwe jest wykorzystanie webowego API. Dane do maszyn wirtualnych, które migrują w obrębie chmury są transmitowane za pomocą protokołu WebSocket. Procedury komunikacji między agentami napisano w języku Scala w wersji 2 oraz wykorzystano środowisko uruchomieniowe Play Framework w wersji 5. Dla każdego agenta przygotowano obraz maszyny wirtualnej z systemem operacyjnym Linux Fedora Server w wersji 30. Moduł webowy i moduł bazodanowy Moodle ulokowano w oddzielnych maszynach wirtualnych. Warto podkreślić, że migracja typu live polega na przeniesieniu maszyny wirtualnej między hostami bez przerwania pracy tej maszyny. Natomiast podczas migracji non-live maszyna wirtualna jest wyłączana, następnie jest przenoszona na inny komputer, na którym jest uruchamiana. W tabeli 11 przedstawiono domyślne wymagania na zasoby dla wybranych dystrybucji systemu Linux/Unix na platformie OpenStack.

Tabela 11 Wymagane zasoby dla wybranych obrazów maszyn wirtualnych na platformie OpenStack.

Nazwa dystrybucji	Zajętość pamięci dyskowej przez skompresowany obrazu systemu operacyjnego [MB]	Domyślna wielkość wirtualnego dysku [GB]	Wielkość zajętej przestrzeni wirtualnego dysku przez system operacyjny [GB]	Wielkość rezerwowanej pamięci RAM przez system operacyjny [MB]
Fedora 30	338,00	4,0	1,10	89
Debian 10	512,55	2,0	0,96	47
openSUSE 15.1	215,90	0,8	0,64	59
CentOS 7	815,25	8,0	0,82	100
Ubuntu 18.04.4 LTS	318,19	2,2	0,94	56

Źródło: opracowanie własne.

W demonstratorze chmury edukacyjnej uruchomiono aplikację InteligentAgent, która imituje zapotrzebowanie na zasoby wybranych agentów, w tym czasy zajętości procesora oraz czasy transmisji danych między maszynami wirtualnymi. Aplikacja ma trzy parametry: *loop* – liczba iteracji, *amount.of.samples* – liczba próbek oraz *agent.url* – adres agenta, z którym rozważany agent się komunikuje. Przyjęto, że liczba użytkowników to 1 001 000, z czego 1 000 000 to liczba studentów, a 1 000 – liczba nauczycieli. Założono, że zajęcia prowadzi 200 nauczycieli dla 300 000 studentów każdego dnia. Ponadto każdy student rozwiązuje wybrane zadanie.

Uśrednione wyniki pomiarów czasu zajętości procesora na hoście referencyjnym klasy H_3, zapotrzebowania na pamięć RAM oraz wielkość pamięci dyskowej dla maszyn wirtualnych przedstawiono w tabeli 12. Do symulacji przyjęto zapotrzebowanie na

zasoby dla przedmiotu Platformy Technologiczne, który prowadzony był na platformie zdalnego nauczania Politechniki Gdańskiej w 2021 roku. Plik z materiałem dydaktycznym zajmuje ok. 19 MB.

Tabela 12 Uśrednione zapotrzebowanie na zasoby dla maszyn wirtualnych na hoście referencyjnym H_3.

VM_k	ram_n [GB]	dm_n [TB]	T_{k3} [s]
1	2	0,2400	659,744
2	2	0,0250	334,965
3	4	0,1000	2797,453
4	4	0,1000	2797,453
5	3	0,0900	2556,440
6	3	0,0900	2556,440
7	2	0,0600	1695,169
8	2	0,0600	1695,169
9	2	0,0500	1438,183
10	2	0,0500	1438,183
11	1	0,0400	1110,126
12	1	0,0400	1110,126
13	1	0,0100	273,934
14	1	0,0100	273,934
15	1	0,0150	411,691
16	1	0,0150	411,691
17	2	0,0250	676,062
18	2	0,0250	676,062
19	3	0,0175	488,855
20	3	0,0175	488,855
Σ	**42**	**1,0800**	**23890,535**

Źródło: opracowanie własne.

Natomiast plik z rozwiązaniem zadania w formacie zip zajmuje 19 KB, czas jego wczytania to 0,209 s, a wielkość bazy danych jest zwiększana o 4 KB dla każdego rozwiązania. Niezbędne jest co najmniej 38 GB pamięci na pliki studentów oraz 4 GB na bazę danych. Uwzględniając inne przedmioty, oszacowano, że baza danych wymaga 25 GB pamięci dyskowej, a pliki – 240 GB. Przyjęto, że baza danych Moodle obsługuje 150 połączeń jednocześnie, a moduł webowy – 256 połączeń.

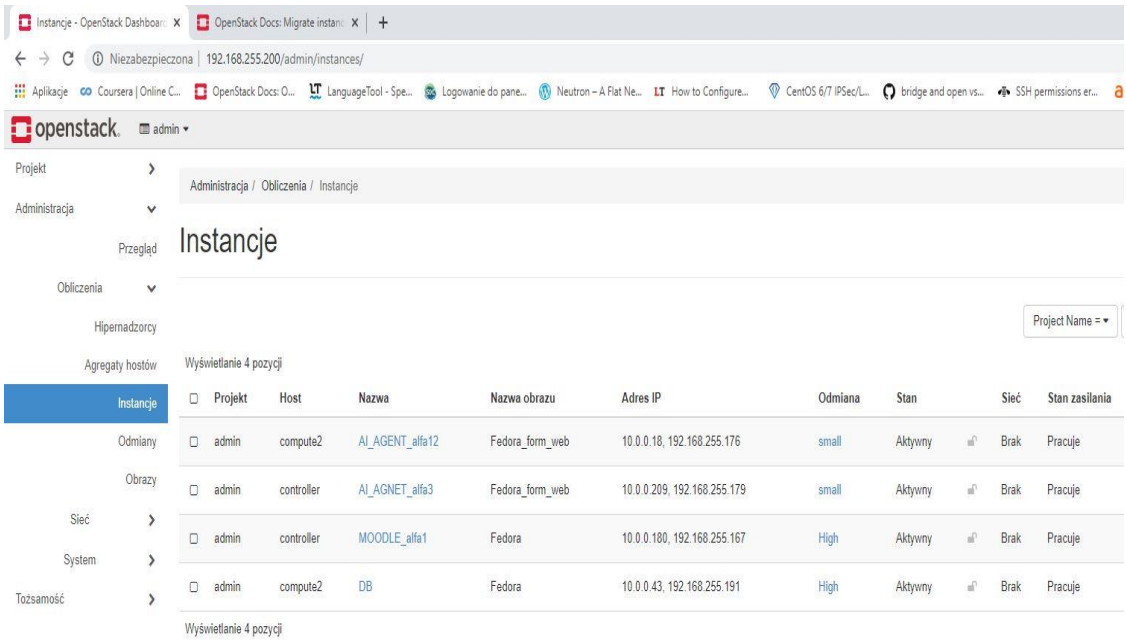
Rozmieszczenie dwudziestu maszyn wirtualnych z modułami systemu Moodla oraz agentami pedagogicznymi w demonstratorze chmury edukacyjnej przedstawiono w tabeli 13. Natomiast na rysunkach 37 - 38 widoczne są kopie ekranów z aplikacji Horizon, która zarządza platformą OpenStack. Przykładowo, maszynę wirtualną VM_{12} z agentem

pedagogicznym nazwano AI_Agent_alfa12. Demonstrator cechuje się następującymi wartościami kryteriów: $F_1 = 15\,333\ [s]$, $F_2 = 12\,973\ [s]$, $F_3 = 0\ [\$]$, $F_4 = 6100\ [\mathrm{W}]$. Rozważając rozbudowę demonstratora chmury, zakłada się, że będzie ona obsługiwać trzykrotnie więcej użytkowników. Oszacowano, że moduł webowy wymaga 5 TB przestrzeni dyskowej oraz 5 GB pamięci RAM. Dla modułu bazy danych zarezerwowano 5 GB przestrzeni pamięci RAM oraz 1 TB pamięci dyskowej.

Tabela 13 Rozmieszczenie dwudziestu maszyn wirtualnych na dziewięciu hostach w demonstratorze chmury edukacyjnej przed modernizacją.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
DELL E5640 S1													x		x			x	x	
DELL E5640 S2				x													x			
DELL E5640 S3						x														x
DELL E5640 S4							x	x								x				
DELL E5640 W1									x	x				x						
DELL E5640 W2					x						x									
DELL T110 II V3	x																			
DELL T110 V3		x										x								
Fujitsu Amilo Notebook Pi 3660 V1			x																	

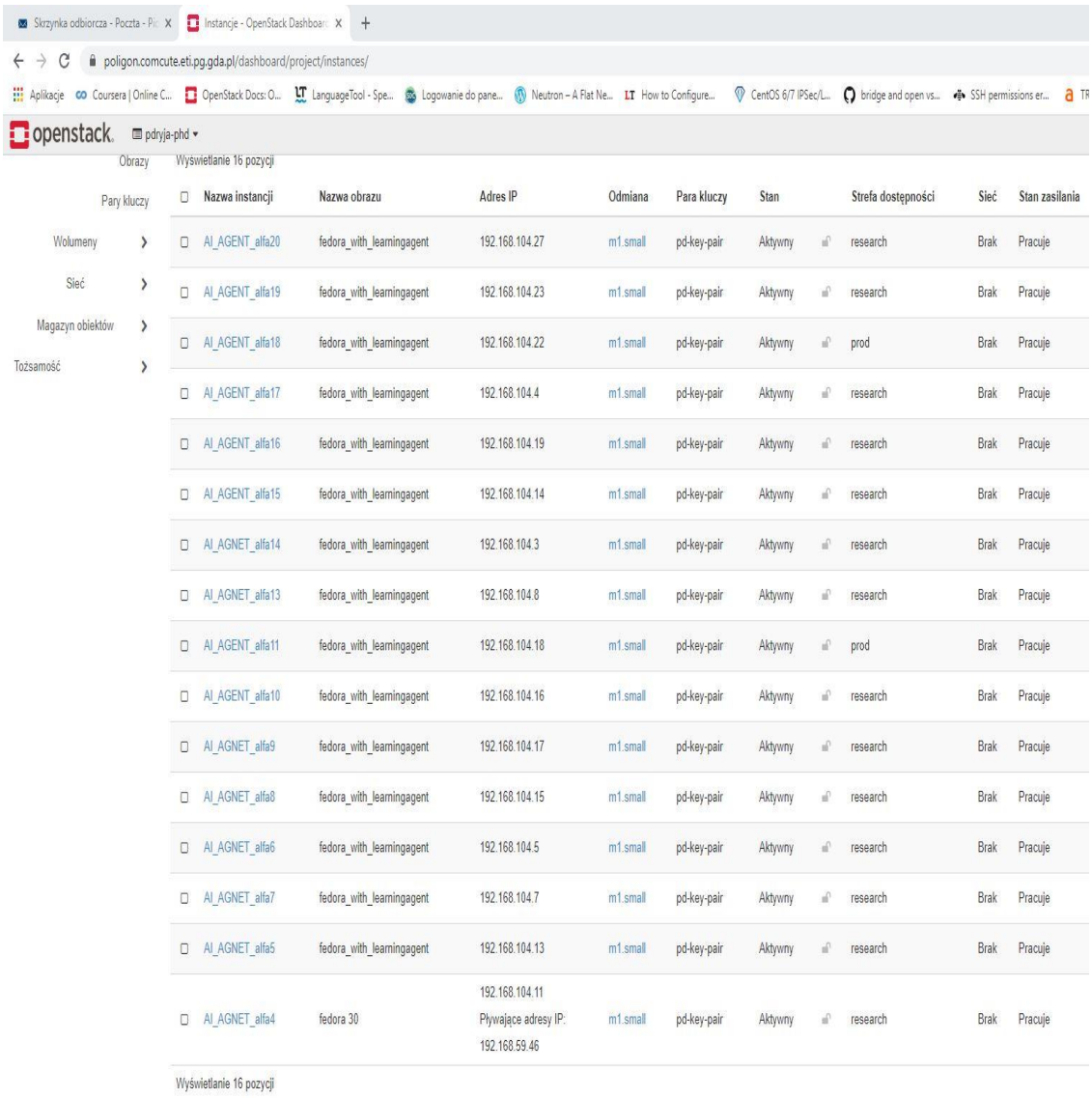
Źródło: opracowanie własne.



Rysunek 37 Rozmieszczenie czterech maszyn wirtualnych na dwóch hostach w chmurze nr 1 demonstratora. Źródło: opracowanie własne na podst. ekranu z aplikacji Horizon.

Niech liczba agentów pedagogicznych zostanie zwiększona z 18 do 54, co wymaga łącznie około 352 GB pamięci RAM oraz około 19 TB przestrzeni pamięci dyskowej. Ponadto przyjmuje się ograniczenie na maksymalny koszt modernizacji chmury edukacyjnej $F_3^{max} = 300\,000\ \$$. Niech liczba węzłów zostanie zwiększona z 9 do 15. W instancji zagadnienia optymalizacji czterokryterialnej (4.2.1) – (4.2.5) preferuje się

minimalizację F_1, F_2, F_3 oraz F_4. Przyjęto, że maksymalne obciążenie procesorów newralgicznego hosta $F_1^{max} = 3\,300\ [s]$, maksymalne obciążenie transmisją danych krytycznego węzła $F_2^{max} = 36\,000\ [s]$ i maksymalny pobór mocy energii elektrycznej $F_4^{max} = 8\,500\ [W]$. Dla $p = 2$ należy wyznaczyć rozwiązanie kompromisowe.



Rysunek 38 Rozmieszczenie szesnastu maszyn wirtualnych w chmurze nr 2 demonstratora. Źródło: opracowanie własne opracowanie własne na podst. ekranu z aplikacji Horizon.

Instancja B1020 cechuje się 1020 zero-jedynkowymi zmiennymi decyzyjnymi, a binarna przestrzeń przeszukiwań zawiera $1{,}12 * 10^{307}$ rozwiązań. Alternatyw spełniających warunki formalne jest $1{,}11 * 10^{82}$. Wielkość populacji wynosi 100, a pozostałe parametry są takie, jak w eksperymencie z instancją B855. Algorytm DEMCA przeszukał łącznie 180 000 populacji w czasie około 10 133 [s] na komputerze marki DELL Inspiron 17 5000. Wyznaczono 29 rozwiązań Pareto (tab. 14-15). Oceny wyznaczonych rozwiązań przedstawiono na rysunkach 39-44.

Tabela 14 Specyfikacja rozwiązań Pareto-optymalnych dla instancji B1024 – część 1.

Lp.	X^{VM}	X^{H}	F_1 $[s]$	F_2 $[s]$	F_3 [\$]	F_4 $[W]$	L_2
1	[7; 13; 11; 4; 3; 5; 5; 14; 6; 6; 2; 8; 10; 5; 3; 4; 14; 8; 1; 5; 2; 9; 14; 15; 10; 11; 12; 10; 9; 9; 12; 8; 4; 8; 14; 7; 15; 1; 2; 7; 11; 8; 1; 10; 7; 15; 7; 4; 7; 12; 15; 14; 10; 10; 14; 10]	[7; 12; 7; 10; 10; 7; 12; 10; 10; 12; 12; 3; 7; 11; 10]	2658	35592	246351	7655	1,06
2	[7; 13; 11; 4; 3; 5; 5; 14; 6; 6; 11; 8; 15; 15; 3; 6; 14; 11; 11; 5; 2; 9; 14; 15; 10; 11; 12; 10; 9; 8; 3; 8; 4; 5; 14; 9; 15; 1; 2; 7; 7; 8; 1; 10; 7; 15; 10; 12; 10; 8; 10; 14; 10; 2; 2; 9]	[7; 12; 7; 7; 10; 7; 12; 10; 10; 12; 12; 3; 7; 11; 10]	2810	35592	235633	7455	1,35
3	[3; 13; 9; 3; 15; 10; 14; 5; 3; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 1; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 1; 11; 6; 12; 2; 8; 10; 10; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 8; 8; 1; 8; 2; 12; 4; 1]	[10; 10; 12; 3; 10; 10; 3; 10; 3; 10; 7; 12; 10; 3; 12]	3133	35592	202716	7020	1,26
4	[3; 13; 9; 3; 9; 10; 14; 5; 15; 8; 6; 2; 12; 5; 3; 10; 10; 2; 1; 1; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 7; 11; 15; 12; 14; 8; 10; 14; 12; 15; 12; 6; 4; 1; 9; 5; 15; 8; 14; 8; 1; 1; 2; 12; 3; 4]	[10; 10; 12; 3; 10; 10; 7; 10; 11; 10; 7; 12; 10; 7; 12]	2708	35592	239060	7505	1,38
5	[3; 13; 9; 3; 9; 10; 14; 5; 15; 8; 6; 2; 12; 5; 3; 10; 10; 2; 1; 7; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 1; 11; 15; 12; 14; 8; 10; 1; 12; 15; 12; 6; 4; 1; 9; 5; 15; 8; 14; 8; 1; 4; 2; 12; 3; 4]	[10; 10; 12; 3; 10; 10; 7; 10; 11; 10; 7; 12; 10; 3; 12]	2708	35592	235287	7560	1,35
6	[3; 13; 9; 3; 15; 10; 14; 5; 14; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 1; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 7; 11; 6; 12; 5; 8; 10; 1; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 14; 8; 1; 8; 2; 12; 4; 12]	[10; 10; 12; 3; 10; 10; 3; 10; 3; 10; 7; 12; 10; 9; 12]	2856	35592	211266	7715	1,22
7	[3; 13; 9; 3; 15; 10; 14; 5; 1; 8; 6; 2; 12; 5; 3; 10; 10; 2; 1; 3; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 4; 11; 6; 12; 8; 8; 10; 12; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 11; 8; 1; 5; 2; 12; 4; 3]	[10; 10; 12; 3; 10; 10; 3; 10; 3; 10; 7; 12; 10; 7; 12]	2949	35592	206489	6965	1,12
8	[3; 13; 9; 3; 15; 10; 14; 5; 14; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 1; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 7; 11; 6; 12; 5; 8; 10; 1; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 14; 8; 1; 8; 2; 12; 4; 12]	[10; 10; 12; 3; 10; 10; 7; 10; 3; 10; 7; 12; 10; 9; 12]	2856	35592	215039	7660	1,22
9	[3; 13; 9; 3; 15; 10; 14; 5; 1; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 7; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 4; 11; 6; 12; 8; 8; 10; 12; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 14; 8; 1; 7; 2; 12; 4; 3]	[10; 10; 12; 3; 10; 10; 7; 10; 3; 10; 7; 12; 10; 7; 12]	2949	35592	210262	6910	1,14
10	[3; 13; 9; 3; 15; 10; 14; 5; 12; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 5; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 1; 11; 9; 11; 8; 8; 10; 7; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 5; 8; 1; 2; 2; 12; 3; 4]	[10; 10; 12; 3; 10; 10; 7; 10; 7; 10; 7; 12; 10; 7; 12]	2947	35592	214035	6855	1,16
11	[3; 13; 9; 3; 15; 10; 14; 5; 14; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 1; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 4; 11; 6; 12; 5; 8; 10; 1; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 11; 8; 1; 9; 2; 12; 4; 12]	[10; 10; 12; 3; 10; 10; 7; 10; 7; 10; 7; 12; 10; 10; 12]	2856	35592	224754	7055	1,20
12	[3; 13; 9; 3; 15; 10; 14; 5; 14; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 1; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 4; 11; 6; 12; 5; 8; 10; 1; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 11; 8; 1; 9; 2; 12; 4; 12]	[10; 10; 12; 3; 10; 10; 3; 10; 7; 10; 7; 12; 10; 10; 12]	2856	35592	220981	7110	1,17
13	[3; 13; 9; 3; 15; 10; 14; 5; 14; 8; 6; 2; 12; 10; 3; 10; 10; 2; 1; 3; 11; 2; 7; 3; 5; 1; 6; 8; 10; 15; 7; 11; 6; 12; 5; 8; 10; 1; 12; 15; 12; 6; 4; 1; 14; 5; 15; 8; 11; 8; 1; 1; 2; 12; 4; 12]	[10; 10; 12; 3; 10; 10; 3; 10; 3; 10; 7; 12;10;10;12]	2856	35592	217208	7165	1,15
14	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 15; 14; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 9; 11; 11; 3; 10; 10; 10; 11; 11; 10; 3]	3072	35313	214462	8465	1,25
15	[13; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 3; 11; 10; 3; 15; 3; 13; 10; 4; 13; 14; 12; 13; 2; 11; 3; 6; 12; 12; 5; 14; 2; 15; 4; 10; 11; 15; 4; 10; 6; 6; 4; 8; 9; 1; 10; 9; 8; 12; 12; 8; 10; 11; 3; 6]	[3; 10; 11; 10; 3; 11; 7; 3; 10; 10; 10; 11; 11; 3; 7]	3293	35592	195194	7570	1,48

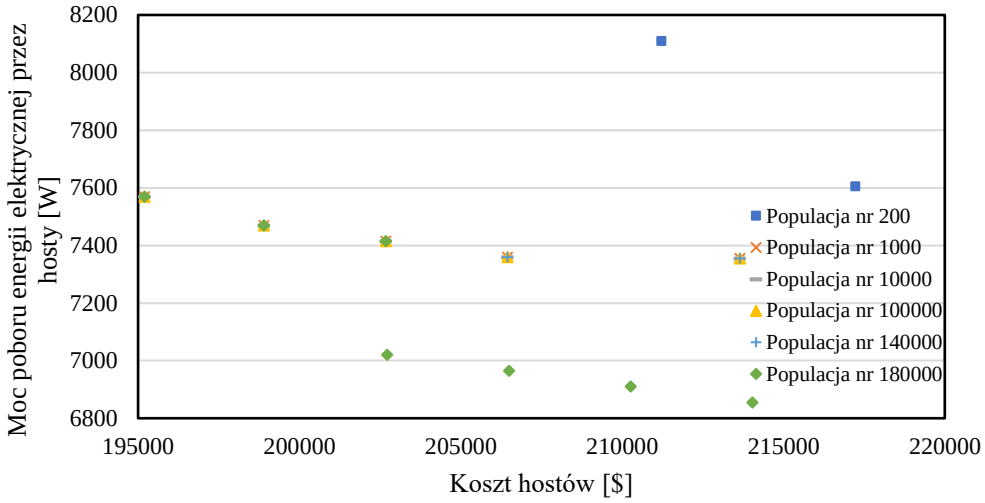
Źródło: opracowanie własne.

Tabela 15 Specyfikacja rozwiązań Pareto-optymalnych dla instancji B1024 – część 2.

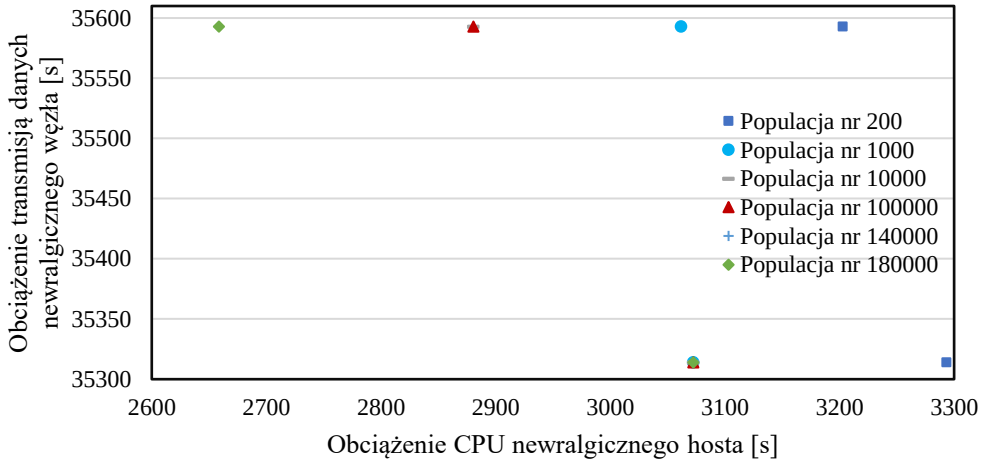
Lp.	X^{VM}	X^{H}	$F_1\ [s]$	$F_2\ [s]$	F_3 [\$]	$F_4\ [W]$	L_2
16	[13; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 3; 11; 10; 3; 15; 3; 13; 10; 4; 13; 14; 12; 13; 2; 11; 3; 6; 12; 12; 5; 14; 2; 15; 4; 10; 11; 15; 4; 10; 6; 6; 4; 8; 9; 1; 10; 9; 8; 12; 12; 8; 10; 11; 3; 6]	[3; 10; 11; 10; 3; 11; 7; 3; 10; 10; 10; 11; 12; 3; 7]	3293	35592	198897	7470	1,47
17	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 14; 8; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 9; 11; 11; 3; 10; 10; 10; 11; 12; 10; 3]	3072	35313	218166	8365	1,23
18	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 13; 14; 15; 4; 10; 11; 15; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 5; 11; 3; 6]	[7; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 12; 10; 7]	3072	35313	231653	7705	1,10
19	[13; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 3; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 3; 6; 12; 12; 5; 14; 2; 15; 4; 10; 11; 15; 4; 10; 6; 6; 4; 8; 9; 1; 10; 9; 8; 12; 12; 8; 15; 11; 3; 6]	[7; 10; 11; 10; 3; 11; 7; 3; 10; 10; 10; 11; 12; 3; 7]	3293	35592	202670	7415	1,46
20	[13; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 3; 11; 10; 3; 15; 3; 13; 10; 4; 13; 14; 12; 13; 2; 11; 3; 6; 12; 12; 5; 13; 8; 15; 4; 10; 11; 15; 4; 10; 14; 6; 4; 8; 9; 1; 10; 9; 8; 12; 12; 8; 15; 11; 3; 6]	[3; 10; 11; 10; 3; 11; 7; 3; 10; 10; 10; 10; 11; 11; 9; 7]	3093	35592	203744	8265	1,50
21	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 15; 14; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 12; 10; 7]	3072	35313	227880	7760	1,07
22	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 15; 14; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 9; 11; 11; 3; 10; 10; 10; 11; 10; 10; 3]	3104	35313	200156	8015	1,01
23	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 14; 8; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 12; 10; 3]	3072	35313	224107	7815	1,05
24	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 12; 14; 15; 4; 10; 11; 15; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 5; 11; 3; 6]	[7; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 10; 10; 7]	3104	35313	213643	7355	0,85
25	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 15; 14; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 10; 10; 7]	3104	35313	209870	7410	0,83
26	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 15; 14; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 11; 10; 3]	3072	35313	220404	7915	1,05
27	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 14; 5; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 8; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 10; 11; 11; 3; 10; 10; 10; 11; 10; 10; 3]	**3104**	**35313**	**206097**	**7465**	**0,83**
28	[7; 7; 5; 9; 11; 2; 12; 2; 6; 3; 15; 10; 11; 10; 3; 15; 3; 13; 1; 4; 13; 14; 12; 13; 2; 11; 5; 6; 12; 12; 5; 15; 14; 15; 4; 10; 11; 9; 4; 10; 14; 6; 4; 8; 6; 1; 10; 9; 8; 12; 12; 8; 15; 11; 3; 6]	[3; 10; 7; 10; 9; 11; 11; 3; 10; 10; 10; 11; 10; 10; 7]	3104	35313	203929	7960	1,00
29	[7; 11; 15; 4; 1; 9; 13; 5; 6; 8; 14; 8; 10; 9; 15; 15; 12; 15; 7; 3; 4; 3; 13; 10; 6; 9; 7; 7; 6; 3; 15; 7; 14; 12; 13; 14; 13; 15; 12; 14; 3; 8; 7; 12; 8; 2; 9; 9; 9; 5; 7; 2; 6; 3; 5; 8]	[3; 3; 12; 10; 3; 10; 12; 12; 11; 3; 7; 10; 10; 10; 10]	2880	35592	217023	7470	1,20

Źródło: opracowanie własne.

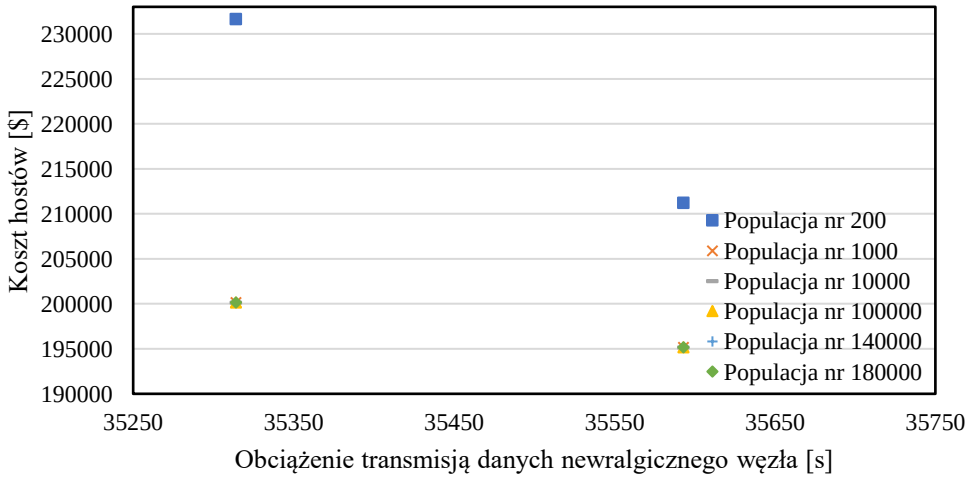
W zbiorze rozwiązań Pareto-optymalnych rozwiązaniem kompromisowym dla parametru p=2 jest punkt P_{27}, który cechuje się następującymi ocenami: $F_1 = 3\,104\ [s]$, $F_2 = 35\,313\ [s]$, $F_3 = 206\,097$ [\$] oraz $F_4 = 7\,465\ [W]$ (tab. 15).



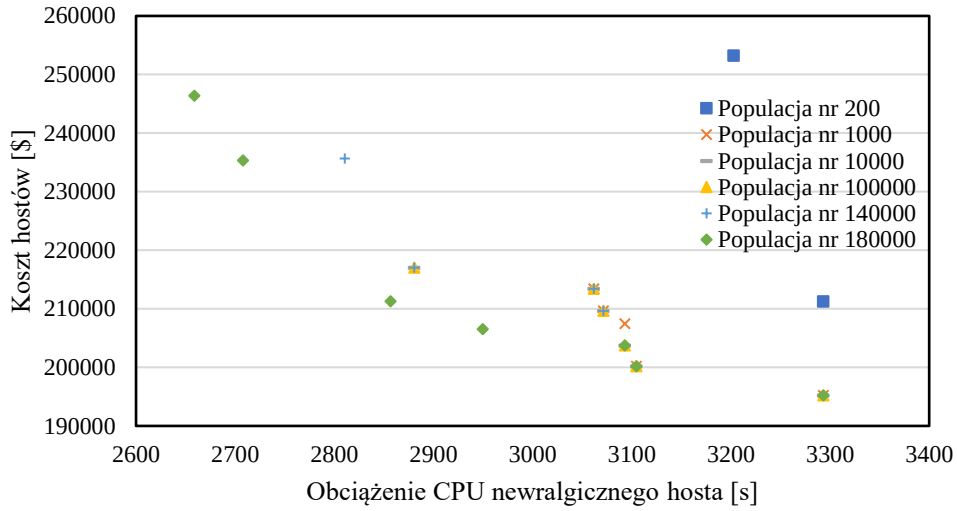
Rysunek 39 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do kosztu hostów i mocy poboru energii elektrycznej przez hosty.
Źródło: opracowanie własne.



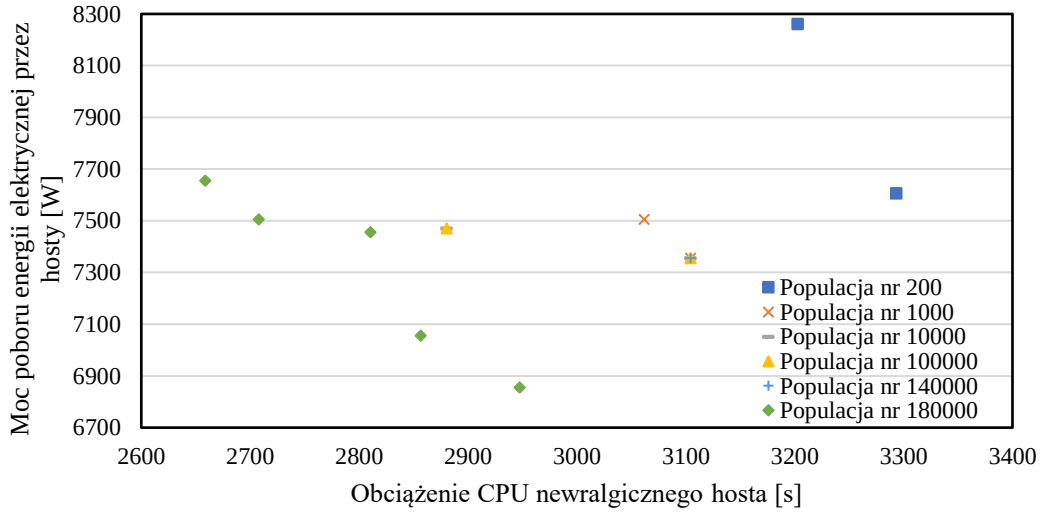
Rysunek 40 Migracja ocen rozwiązań sprawnych na wybranych etapach ewolucji w odniesieniu do obciążenia CPU newralgicznego hosta i obciążenia transmisją danych newralgicznego węzła.
Źródło: opracowanie własne.



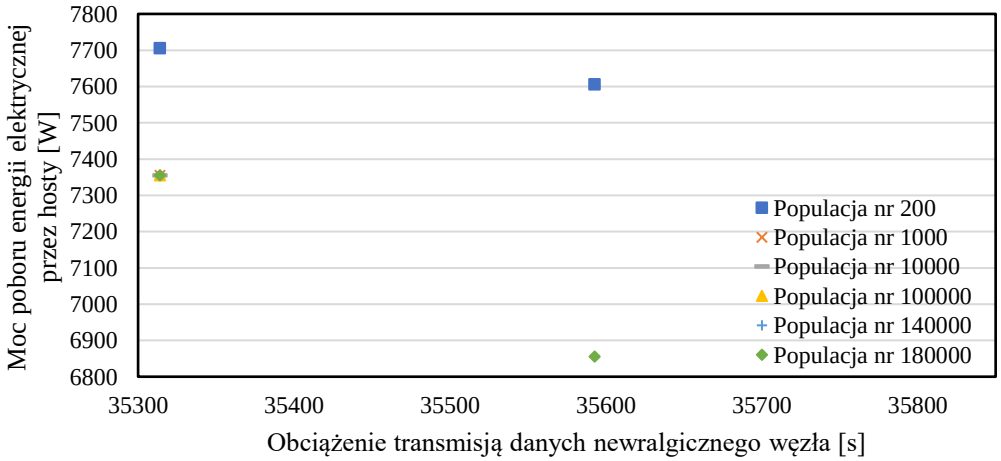
Rysunek 41 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia transmisją danych newralgicznego węzła i kosztu hostów.
Źródło: opracowanie własne.



Rysunek 42 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia CPU newralgicznego hosta i kosztu hostów.
Źródło: opracowanie własne.

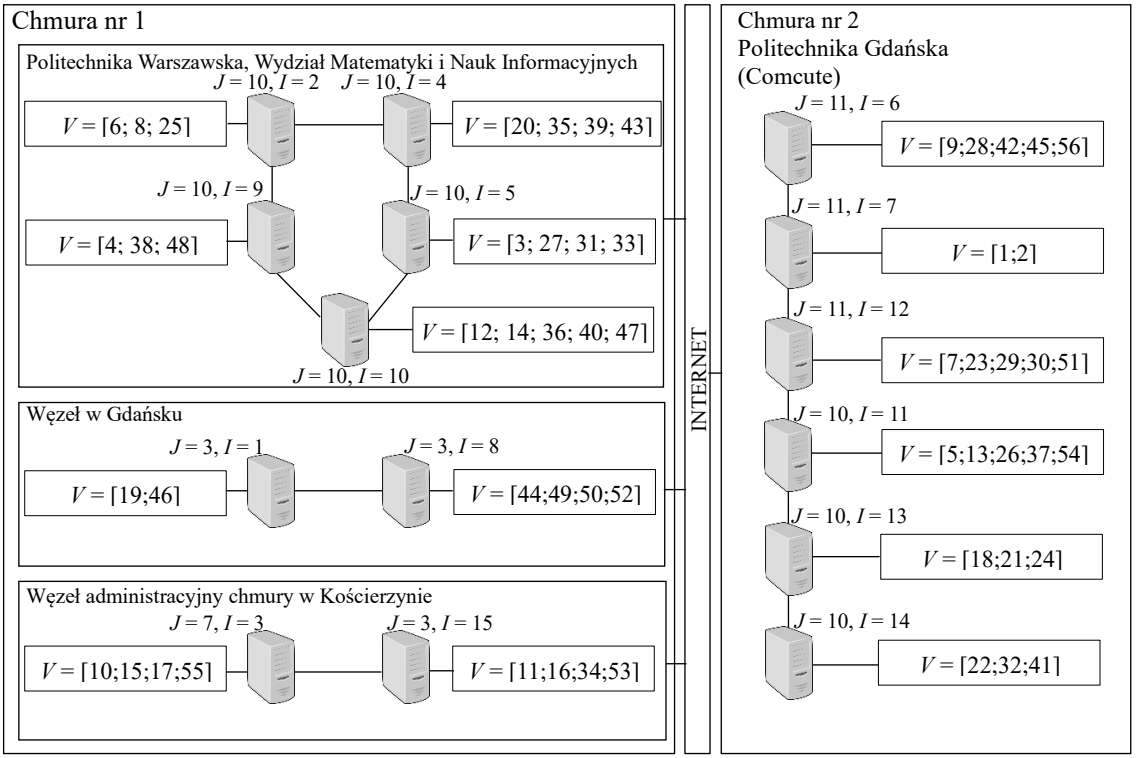


Rysunek 43 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia CPU newralgicznego hosta i zużycia energii hostów.
Źródło: opracowanie własne.



Rysunek 44 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia komunikacyjnego newralgicznego węzła i zużycia energii hostów.
Źródło: opracowanie własne.

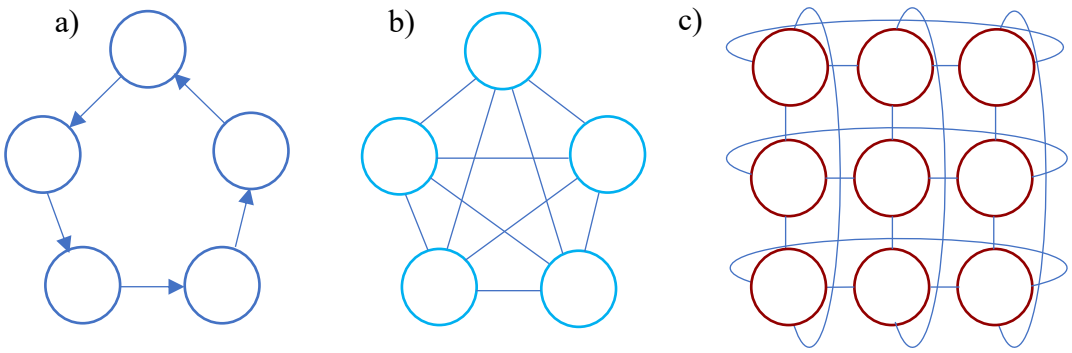
Na rysunku 45 zobrazowano rozwiązanie kompromisowe dla parametru p=2. Ze względu na rozmiar instancji obliczenia wykonano również na superkomputerze Prometheus, który eksploatowany jest w Akademii Górniczo-Hutniczej w Krakowie od 2015 roku. Moc obliczeniowa systemu 53 tysięcy rdzeni to 2,4 PFLOPS wg testu HPC Linpack. Prometheus korzysta z systemu operacyjnego CentOS w wersji 7. Komputer jest wyposażony w procesory Intel Xeon Haswell i Intel Xeon Gold, procesory graficzne NVIDIA K40 XL i GPGPU NVIDIA K80 oraz zestaw serwerów. Może wspomagać optymalizację migracji maszyn wirtualnych w chmurze edukacyjnej. W ramach grantu pt. *Wspomaganie prac dyplomowych za pomocą superkomputerów* powtórzono wyznaczenie rozwiązania kompromisowego za pomocą równoległej wersji algorytmu DEMCA.



Rysunek 45 Alokacja maszyn wirtualnych w hostach projektowanego rozszerzenia demonstratora chmury edukacyjnej dla wyznaczonego rozwiązania kompromisowego z parametrem p=2.
Źródło: opracowanie własne.

Bardzo efektywny okazał się model wyspowy o topologii typu torus, w którym niezależne algorytmy DEMCA (wyspy) wymieniają się interwałowo rozwiązania (rys. 46c). Rozważano różne sposoby „eksportu" rozwiązań, w tym wymianę najsłabszych w sensie funkcji sprawności, wymianę najlepszych oraz wybór losowy z możliwością powrotu na wyspę lub też nie. Istotną kwestia była również liczba emigrantów. Dla zadanej topologii modelowanej za pomocą grafu skierowanego, rozwiązania przesyłane są między wyspami (wierzchołkami grafu) w kierunku wskazanym przez łuk (rys. 46). Populacja

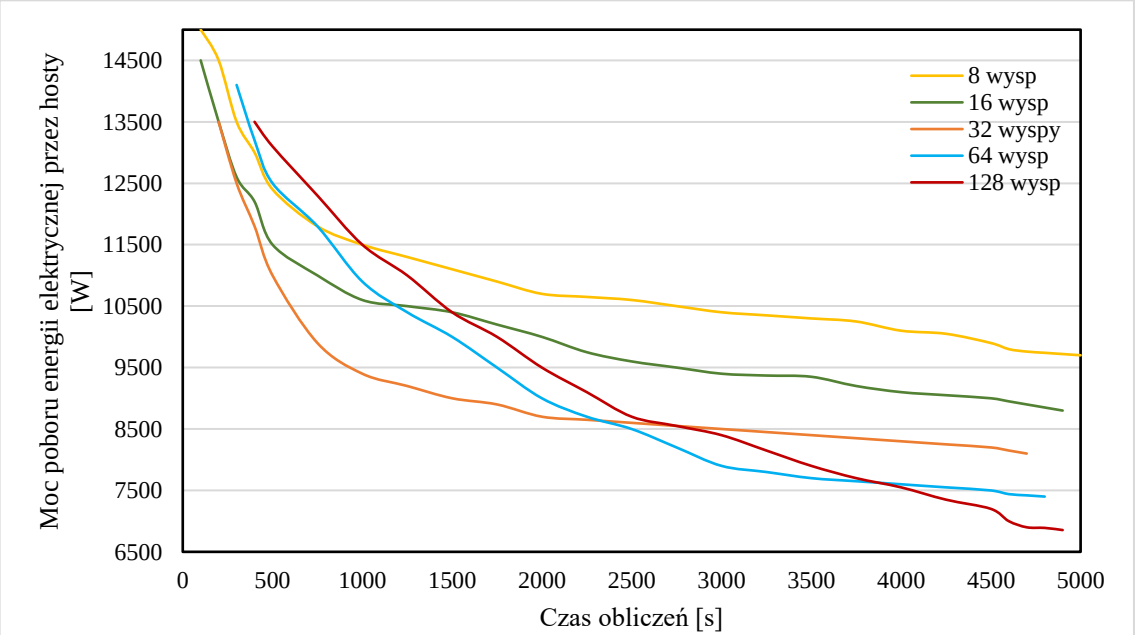
algorytmu, który „utknął” w optimum lokalnym może być wzbogacona za pomocą wysokiej jakości alternatyw z innej wyspy, co może poprawić zbieżności algorytmu.



Rysunek 46 Wybrane topologie równoległych wersji algorytm DEMCA.
Źródło: opracowanie własne.

System kolejkowania SLURM obsługuje aplikacje i zarządza dostępnymi zasobami superkomputera. Za pomocą polecenia *sbatch* określa się zapotrzebowanie na zasoby, w tym: nazwę zlecenia, liczbę węzłów obliczeniowych, wielkość pamięci, a także czas zlecenia. Węzeł ma 24 rdzenie, co pozwala na efektywne obliczenia współbieżne 24 procesów. Komunikacja między procesami na różnych węzłach odbywa się za pomocą interfejsu MPI (ang. *Message Passing Interface*), który jest implementowany za pomocą następujących bibliotek: MPICH, MPICH2, OpenMPI oraz Intel MPI.

Na rysunku 47 porównano minimalizację mocy poboru energii elektrycznej do 6855 watów za pomocą wybranych wersji DEMCA dla zadanych liczb wysp.



Rysunek 47 Minimalizacja łącznej mocy poboru energii elektrycznej dla równoległych wersji algorytmu DEMCA.
Źródło: opracowanie własne.

Dla 8 i 16 wysp uruchomiono jeden proces, zaś dla 32, 64 i 128 wysp uruchomiono odpowiednio 2, 4 i 8 procesów po 16 wątków. 15 wątków procesu zarządza 15 wyspami, które wymieniają się osobnikami w obrębie procesu. Natomiast szesnasty wątek procesu, przesyła rozwiązania ze swojej wyspy do sąsiedniego procesu. Każdy przebieg algorytmu DEMCA trwał nie dłużej niż godzinę i 20 minut. Algorytm za każdym razem wyznaczał rozwiązanie kompromisowe dla 128 wysp.

5.5 Wnioski i uwagi

Wysoką jakość rozwiązań wyznaczanych za pomocą wielokryterialnego algorytmu ewolucji różnicowej DEMCA potwierdzono za pomocą kilkudziesięciu eksperymentów numerycznych, z których omówiono cztery reprezentatywne przykłady. Podczas pierwszego eksperymentu z instancją B90 wyznaczono rekomendowane wartości metaparametrów: współczynnika skalowania mutacji q oraz progu akceptacji genów potomków w krzyżowaniu Cp.

W drugim eksperymencie z instancją B306 porównano jakość rozwiązań wyznaczonych za pomocą algorytmu DEMCA z alternatywami otrzymanymi za pomocą algorytmu poszukiwania harmonii MOHS oraz algorytmu programowania genetycznego MOGPA. Za pomocą algorytmu DEMCA wyznaczono warianty efektywne wyższej jakości niż wspomniane metaheurystyki przy szybszej zbieżności do brzegu Pareto.

Za pomocą eksperymentu trzeciego z instancją B855 zaprojektowano modernizację chmury obliczeniowej Comcute na Politechnice Gdańskiej. Porównano rozwiązania wyznaczone za pomocą wielokryterialnych algorytmów: MOHS, MOGPA, algorytmu ewolucyjnego NSGA-II i algorytmu optymalizacji zbioru cząstek OMOPSO. W tym wypadku za pomocą algorytmu DEMCA wyznaczono rozwiązania niezdominowane cechujące się wyższą jakością niż pozostałe algorytmy.

Czwarty eksperyment z instancji B1020 dotyczył projektu rozbudowy demonstratora chmury edukacyjnej do 15 hostów. Skonstruowano demonstrator chmury edukacyjnej GUT-WUT na 10 hostach, przy czym 20 maszyn wirtualnych mogło migrować między 9 hostami. Dokonano pomiarów czasów transmisji między modułami systemu Moodle oraz agentami pedagogicznymi. Zmierzono również czasy pracy agentów na serwerach. Oszacowano pobór mocy energii elektrycznej przez hosty oraz koszty ich zakupu. Za pomocą algorytmu DEMCA wyznaczono rozwiązanie kompromisowe dla parametru

p=2, uwzględniając zwiększone zapotrzebowanie na zaso-by przez 54 agenty pedagogiczne i 2 moduły Moodla.

Zweryfikowano wyspowy model równoległych obliczeń za pomocą algorytmu DEMCA na superkomputerze Prometheus w środowisku PL Grid. Eksperymentalnie potwierdzono, że zwiększenie liczby wątków obliczeniowych do 128 umożliwia intensywniejszą eksplorację przestrzeni przeszukiwań, a w efekcie szybsze wyznaczenie rozwiązań wysokiej jakości, co jest istotne w implementacji chmury edukacyjnej.

PODSUMOWANIE

Optymalizacja zasobów chmury obliczeniowej, w której wykorzystuje się inteligentne agenty do wspomagania zdalnego nauczania jest ważnym zagadnieniem badawczym w informatyce technicznej i telekomunikacji. Oryginalnym dorobkiem autora jest opracowanie koncepcji i zbudowanie demonstratora chmury edukacyjnej opartej o oprogramowanie platformy OpenStack, która może łączyć chmury wydziałowe i uczelniane, współdzieląc zasoby informatyczne i edukacyjne. Ponadto opracowano model migracji maszyn wirtualnych oraz doboru rodzajów hostów w chmurze. Omówiono kryteria i ograniczenia, na podstawie których sformułowano adekwatne zagadnienia wyznaczania reprezentacji rozwiązań optymalnych w sensie Pareto. Ze zbioru rozwiązań niezdominowanych wyselekcjonowano rozwiązania kompromisowe dla p=2.

Problem optymalizacji wielokryterialnej (4.2.1) – (4.2.5) dotyczy wyznaczania migracji maszyn wirtualnych w chmurze edukacyjnej i opiera się na minimalizacji czterech kryteriów: obciążenia procesorów w newralgicznym hoście, obciążenia komunikacyjnego w kluczowym węźle, łącznego poboru mocy energii elektrycznej przez hosty, a także kosztu zakupu hostów, przy uwzględnieniu ograniczeń górnych dla przyjętych kryteriów. Ponadto uwzględnia się postulat nieprzekroczenia wielkości dostępnej pamięci RAM oraz pamięci dyskowej. Warto podkreślić, że sformułowane zagadnienie optymalizacji może być rozszerzone o maksymalizację wielkości dostępnej pamięci RAM w newralgicznym serwerze. Można również rozważać maksymalizację stopnia rozproszenia modułów w chmurze.

Do wyznaczania rozwiązań sformułowanego problemu optymalizacji (4.2.1) – (4.2.5) zaproponowano wielokryterialny algorytm ewolucji różnicowej DEMCA. Implementacja algorytmu napisana w języku programowania Java umożliwia powtórzenie uzyskanych wyników badań i może być zastosowana do wyznaczania efektywnych migracji agentów pedagogicznych w najbardziej odpowiednich do tego hostach.

Przeprowadzone eksperymenty numeryczne z różnorodnymi instancjami dla zagadnienia (4.2.1) – (4.2.5) wykazały, że zazwyczaj ocena dominująca nie istnieje, co wynika z konfliktu między kryteriami cząstkowymi. Zamiast rozwiązań dominujących proponuje się zatem wyznaczanie rozwiązań niezdominowanych za pomocą algorytmu ewolucji różnicowej DEMCA. Warto podkreślić, że ze zbioru reprezentacji rozwiązań suboptymalnych w sensie Pareto można wyselekcjonować rozwiązania hierarchiczne lub warianty kompromisowe dla dowolnej wartości parametru p.

Analizując specyfikę problemu optymalizacji wielokryterialnej $(\boldsymbol{X}, F, \succ_{\leq})$ (4.2.1) – (4.2.5), udowodniono twierdzenie 4.1 o zawieraniu się zbioru rozwiązań optymalnych w sensie Pareto $X_P^{\leq}(I) \subseteq \boldsymbol{X}$ w zbiorze wariantów Pareto $\boldsymbol{X}_P^{\leq}(I+l) \subseteq \boldsymbol{X}$ dla I+l kryteriów, l=1,2,3,…, a także dla tego samego zbioru alternatyw dopuszczalnych $\boldsymbol{X}$ oraz relacji dominowania $\succ_{\leq}$ w $\boldsymbol{R}^{I+l}$.

Dokonano eksperymentalnej weryfikacji poprawności opracowanego wielokryterialnego algorytmu ewolucji różnicowej za pomocą kilkudziesięciu instancji testowych. Zbudowano demonstrator chmury edukacyjnej, a w celu jego modernizacji wyznaczono rozwiązanie kompromisowe zagadnienia optymalizacji wektorowej (4.2.1) – (4.2.5).

Pożądanym kierunkiem dalszych badań jest zastosowanie modeli uczenia głębokiego w edukacji. Rozwiązania kompromisowe wyznaczone za pomocą algorytmu DEMCA na superkomputerach mogą efektywnie wspomagać migrację maszyn wirtualnych w rozległych chmurach edukacyjnych. Natomiast wielokryterialny algorytm ewolucji różnicowej może podlegać dalszej poprawie za pomocą wykorzystania bramek kwantowych. Skonstruowany model demonstratora chmury edukacyjnej może być efektywną platformą w edukacji na poziomie wybranych wydziałów i uczelni. Zwłaszcza w sytuacjach intensywnej pandemii, zagrożeń militarnych oraz kryzysu energetycznego zaprezentowane podejście może efektywnie wspomagać edukację w uczelniach i szkołach, w których dostęp do materiałów dydaktycznych, laboratoriów oraz kadry dydaktycznej jest utrudniony.

*****

Na zakończenie chciałbym podziękować mojej rodzinie za wsparcie i zrozumienie. Żonie Magdalenie za cierpliwość, a następnemu pokoleniu: Kacprowi, Jakubowi i Lenie za słuchanie wieczorem opowiadań o chmurze edukacyjnej. Dziękuję moim Współpracownikom z Katedry Architektury Systemów Komputerowych Politechniki Gdańskiej za owocny czas zdobywania doświadczeń zawodowych podczas pobytu w katedrze jako doktorant i asystent w latach 2014-2021. Podziękowania składam również na ręce mojego Promotora prof. Jerzego Balickiego za bezcenne rady i uwagi. Książki, które rekomendował mi Promotor wywarły na mnie ogromne wrażenie. Dlatego niezmiernie ważne są podziękowania dla wybitnych Autorów tych monografii. Rozprawa nie powstałaby, gdyby nie znakomite prace panów Profesorów: Andrzeja Ameljańczyka, Józefa Korbicza, Henryka Krawczyka i Jana Węglarza.

BIBLIOGRAFIA

1. Abbass H. A., Sarker R. A.: The Pareto differential evolution algorithm. Int. J. Artif. Intell. Tools, vol. 11(4), 2002, pp. 531-552.
2. Abdelmaboud A., Jawawi D. N. A., Ghani I., Elsafi A., Kitchenham B. A.: Quality of service approaches in cloud computing: A systematic mapping study. J. Syst. Softw., vol. 101, 2015, pp. 159-179.
3. Abdykarimova S., Kerimbayev N., Nurym N., Akramova A.: Virtual educational environment: interactive communication using LMS Moodle. Educ. Inf. Technol., vol. 25(3), 2020, pp. 1965-1982.
4. Advanced Distributed Learning at NATO SCHOOL Oberammergau, http://www.natoschool.nato.int/Academics/eLearning, dostęp: listopad 2022.
5. Agarwal R., Deo A., Das S.: Intelligent agents in e-learning. ACM SIGSOFT Software Engineering Notes, vol. 29(2), 2004.
6. Alam M. A., Saiyeda A.: A cloud-based solution for smart education. Int. J. Smart Educ. Urban Soc., vol. 11(2), 2020, pp. 28-37.
7. Alberto I., Azcarate C., Mateo P. M.: Multi-objective evolutionary algorithms. Pareto rankings. Monografias del Seminario Matematico Garcia de Galdeano, vol. 27, 2003, pp. 27-35.
8. Ali J., Zafari F., Khan G. M., Mahmud S. A.: Future client's requests estimation for dynamic resource allocation in cloud data center using CGPANN. ICMLA, vol. 2, 2013, pp. 331-334.
9. Alkhatib A. A. A., Sawalha T., AlZu'bi S.: Load balancing techniques in software-defined cloud computing: an overview. SDS, 2020, pp. 240-244.
10. Ameljańczyk A.: Wprowadzenie do optymalizacji wielokryterialnej. WAT, Warszawa 2022.
11. Ameta D., Tiwari S., Singh P.: A preliminary study on case-based learning teaching pedagogy: Scope in SE education. Proc. of the 13th Innovations in Software Engineering Conf., February 2020, No. 11, pp. 11:1-11:12.
12. Ammar M. B., Neji M.: A multi-agent based system for affective peer-e-learning. Second Int. Mobile Multimedia Communications Conf., 2006, pp. 40-46.
13. Aneja Y.P., Nair K.P.K.: Bicriteria transportation problem. Management Science, vol. 25(1), 1979, pp. 73-78.
14. Ardagna D., Casale G., Ciavotta M., Pérez J. F., Wang W.: Quality-of-service in cloud computing: modeling techniques and their applications. J. Internet Serv. Appl., vol. 5(1), 2014, pp. 11:1-11:17.
15. Aslam S., Shah M. A.: Load balancing algorithms in cloud computing: a survey of modern techniques. National Software Engineering Conference, 2015, pp. 30-35.
16. Balicki J.: Algorytmy ewolucyjne oraz algorytmy przeszukiwania tabu do optymalizacji przydziałów modułów programów w rozproszonych systemach komputerowych. Wyd. AMW, Gdynia 2000.
17. Balouek-Thomert D., Bhattacharya A.K., Caron E., Gadireddy K., Lefèvre L.: Parallel differential evolution approach for cloud workflow placements under simultaneous optimization of multiple objectives. CEC, 2016, pp. 822-829.
18. Begam R., Wang W.: TIMER-Cloud: time-sensitive VM provisioning in resource-constrained clouds. IEEE Trans. Cloud Comput., vol. 8(1), 2020, pp. 297-311.
19. Beloglazov A., Buyya R.: Energy efficient resource management in virtualized cloud data centers. Proc. of the 2010 10th IEEE/ACM Int. Conf. on Cluster, Cloud and Grid Computing, 2010, pp. 826-831.

20. Benchmark procesorów, https://www.cpubenchmark.net/multi_cpu.html, dostęp: listopad 2022.
21. Błażewicz J.: Problemy optymalizacji kombinatorycznej - Złożoność obliczeniowa. Algorytmy aproksymacyjne. PWN, Warszawa 1986.
22. Belter B., Mika M., Węglarz J.: Scheduling of network tasks to minimize the consumed energy. International Transactions in Operational Research, 2021, vol. 28, iss. 1, s. 168-200.
23. Brest J., Greiner S., Boskovic B., Mernik M., Žumer V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput., vol. 10(6), 2006, pp. 646-657.
24. Brest J., Maucec M. S.: Population size reduction for the differential evolution algorithm. Appl. Intell., vol. 29(3), 2008, pp. 228-247.
25. Brummett T., Galloway M.: Towards providing resource management in a local IaaS cloud architecture. Inf. Technology: New Generations, 2016, pp. 413-423.
26. Buyya R., Beloglazov A., Abawajy J.: Energy-efficient management of data center resources for cloud computing. Proc. of the 2010 Int. Conf. on Parallel and Distributed Processing Techniques and Applications, 2010, pp. 6-20.
27. Chanaa A., El Faddouli N.-E.: Deep learning for a smart e-learning system. ICSDE, 2018, pp. 197-202.
28. Chen C.-M., Hong C.-M., Mei-Hui Chang M.-H.: Personalized learning path generation scheme utilizing genetic algorithm for web-based learning. WSEAS Transactions on Information Science and Applications, vol. 3(1), 2006, pp. 88-95.
29. Chen L., Chen P., Lin Z.: Artificial intelligence in education: A review. IEEE Access, vol. 8, 2020, pp. 75264-75278.
30. Chin-Bang Y.: Context-aware customization e-learning system with intelligent online examination mechanism. Int. Conf. on Information Systems, 2009, pp. 502-507.
31. Cisco Webex, https://cart.webex.com/, listopad 2022.
32. Cornelius R.: Improving prescripted agent behavior with neuroevolution. department of computer sciences. The University of Texas at Austin, Undergraduate Honors Thesis, 2005.
33. Das S., Mullick S. S., Suganthan P. N.: Recent advances in differentia evolution – An updated survey. Swarmand Evolutionary Computation, vol. 27, 2016, pp. 1-30.
34. Dash R.: DECPNN: A hybrid stock predictor model using differential evolution and Chebyshev polynomial neural network. Intell. Decis. Technol., vol 12(1), 2018, pp. 93-104.
35. Dashti S.E., Rahmani A.M.: Dynamic VMs placement for energy efficiency by PSO in cloud computing. J. Exp. Theor. Artif. Intell., vol. 28(1-2), 2016, pp. 97-112.
36. Dokeos, https://www.dokeos.com/, listopad 2022.
37. Dongarra J.J.: The LINPACK Benchmark: An Explanation. ICS, 1987, pp. 456-474.
38. Dragoi E.-N., Dafinescu V.: Parameter control and hybridization techniques in Differential evolution: a survey. Artif. Intell. Rev., vol 45 (4), 2016, pp. 447-470.
39. Dryja P., Balicki J., Balicka H., Tyszka M.: Big data and the Internet of Things in edge computing for smart city. IFIP Int. Conf. on Computer Information Systems and Industrial Management, 2019, pp. 99-109.
40. Dryja P., Balicki J., Balicka H., Tyszka M.: Multi-criteria differential evolution for optimization of virtual machine resources in smart city cloud. Computer Information Systems and Industrial Management, 2020, pp. 332-344.

41. Dryja P., Balicki J., Balicka H., Tyszka M.: Social media and efficient computer infrastructure in smart city. The 10th Jubilee Scientific Conf. – InfoGlob, 2018, pp. 1003:1-1003:10.
42. Dryja P., Balicki J., Balicka H., Tyszka M.: Social media for e-learning of citizens in smart city. The 10th Jubilee Scientific Conf. – InfoGlob, 2018, pp. 1002:1-1002:10.
43. Dryja P., Balicki J., Beringer M., Korłub W., Paluszak J., Przybyłek P., Tyszka M., Zadroga M., Zakidalski M.: Inteligentne systemy agentowe w systemach zdalnego nauczania. EduAkcja, vol. 1(9), 2015, pp. 51-64.
44. Dryja P., Balicki J., Korłub W., Tyszka M.: Metody i aplikacje zdalnego szkolenia mieszkańców inteligentnych miast. Studia i Materiały Instytutu Transportu i Handlu Morskiego, vol. 14, 2017, pp. 158-175.
45. Dryja P., Balicki J., Korłub W., Tyszka M.: Metody zwiększania dostępności i efektywności informatycznej infrastruktury w inteligentnym mieście. Współczesna Gospodarka, vol. 8(3), 2017, pp. 1-12.
46. Dryja P., Balicki J., Korłub W.: Harmony search for data mining with Big Data. IFIP Int. Conf. on Computer Information Systems and Industrial Management, 2016, pp. 553-565.
47. Dryja P., Balicki J., Zakidalski M.: Some artificial intelligence driven algorithms for mobile edge computing in smart city. IFIP Int. Conf. on Computer Information Systems and Industrial Management, 2019, pp. 110-119.
48. Dukhanov A., Karpova M., Bochenina K.: Design virtual learning labs for courses in computational science with use of cloud computing technologies. Procedia Computer Science, vol. 29, 2014, pp. 2472-2482.
49. El Mhouti A., Erradi M., Nasseh A.: Using cloud computing services in e-learning process: Benefits and challenges. Educ. Inf. Technol, vol. 23(2), 2018, pp. 893-909.
50. Endo P. T., de Almeida Palhares A. V., Pereira N. N., Goncalves G. E., Sadok D., Kelner J., Melander B., Mångs J. E.: Resource allocation for distributed cloud: concepts and research challenges. IEEE Netw., vol. 25(4), 2011, pp. 42-46.
51. Fernández-Manjón B., Sánchez-Pérez J. M., Gómez-Pulido J. A., Vega-Rodríguez M. A., Bravo-Rodríguez J.: Computers and education: e-learning, from theory to practice. Springer, 2007.
52. Fonseca C. M., Fleming P. J.: Genetic algorithms for multiobjective optimization: formulation, discussion, and generalization. Proc. of the Fifth Int. Conf. on Genetic Algorithms, 1993, pp. 416-423.
53. Franklin S., Graesser A. C.: Is it an agent, or just a program? A taxonomy for autonomous agents. ATAL, 1996, pp. 21-35.
54. Gaebel M., Kupriyanova V., Morais R., Colucci E.: E-learning in European higer education institutions. EUA Publications, 2014.
55. Gâlea, D., Leon, F., Zaharia M. H.: E-learning distributed framework using intelligent agents. Tech. University Asachi, Polirom Press, Iasi, 2003, pp. 219-223.
56. Garruzzo S., Rosaci D., Sarne G. M. L.: ISABELE: A multi agent e-learning system that supports multiple devices. IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, 2007, pp. 485-488.
57. Goldberg D.E., Genetic algorithms in search, optimization, and machine learning. Addison Wesley, 1989.
58. Gong W., Wang W.: Application of support vector machine in e-learning for personality. Cloud Computing and Intelligence Systems, 2011, pp. 638-624.
59. Google Meet, https://workspace.google.com/intl/en_ie/products/meet, dostęp: listopad 2022.

60. Google Trends, https://www.google.pl/trends/, dostęp: listopad 2022.
61. Gu J., Hu J., Zhao T., Sun G.: A new resource scheduling strategy based on genetic algorithm in cloud computing environment. Journal of Computers, vol. 7(1), 2012, pp. 42-52.
62. Hancer E.: A new multi-objective differential evolution approach for simultaneous clustering and feature selection. Eng. Appl. Artif. Intell., vol. 87, 2020, pp. 1-9.
63. HAProxy, http://cbonte.github.io/haproxy-dconv/1.9/intro.html, dostęp: listopad 2022.
64. Hu H.: An educational Arduino robot for visual deep learning experiments. IJIRA, vol. 4(1), 2020, pp. 73-81.
65. Hu W. X., Zheng J., Hua X. Y., Yang Y. O.: A computing capability allocation algorithm for cloud computing environment. Appl. Mech. Mater., 2013, pp. 347-350.
66. Huibin J., Mingguang L.: An improved differential evolution algorithm for optimization. IITA International Conference on Control, Automation and Systems Engineering, 2009, pp: 659-662.
67. Introduction to intelligent agents, http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php, dostęp: listopad 2022.
68. Islam S., Keung J., Lee K., Liu A.: Empirical prediction models for adaptive resource provisioning in the cloud. Future Gener. Comput. Syst., vol. 28(1), 2012, pp. 155-162.
69. Jagtap A., Bodkhe B., Gaikwad B., Kalyana S.: Homogenizing social networking with smart education by means of machine learning and Hadoop: A case study. IOTA, 2016, pp. 85-90.
70. Jamali A., Mallipeddi R., Salehpour M., Bagheri A.: Multi-objective differential evolution algorithm with fuzzy inference adaptive mutation factor for Pareto optimum design of suspension system. Swarm Evol. Comput., vol. 54, 2020, pp. 1-14.
71. Jara-Roa D., Valdiviezo-Díaz P., Agila-Palacios M., Sarango-Lapo C., Rodriguez-Artacho M.: An adaptive multi-agent based architecture for engineering education. IEEE EDUCON, 2010, pp. 217-222.
72. Jayanthi S.: Literature review: dynamic resource allocation mechanism in cloud computing environment. International Conference on Electronics, Communication and Computational Engineering, 2014, pp. 279-281.
73. Jennings B., Stadler R.: Resource management in clouds: survey and research challenges. J. Netw. Syst. Manag., vol. 23(3), 2015, pp. 567-619.
74. Jennings N. R., Sycara K., Wooldridge M.: A roadmap of agent research and development. Journal of Autonomous Agents and Multi-Agent Systems, vol. 1(1), 1998, pp. 7-38.
75. Jyoti A., Shrimali M.: Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing. Cluster Computing, vol. 23(1), 2020, pp. 377-395.
76. Kansal N.J., Chana I.: Artificial bee colony based energy-aware resource utilization technique for cloud computing. Concurr. Comput., vol. 27(5), 2015, pp. 1207-1225.
77. Katyal M., Mishra A.: A comparative study of load balancing algorithms in cloud computing environment. Int. Journal of Distributed and Cloud Computing, vol. 1(2), 2014, pp. 5-14.
78. Kaur R., Luthra P.: Load balancing in cloud computing. Association of Computer Electronics and Electrical Engineers, 2014, pp. 374-381.
79. Kisiel-Dorohinicki M.: Agentowe architektury populacyjnych systemów inteligencji obliczeniowej. Wyd. AGH, Kraków 2013.

80. Kloos C.D., Alario-Hoyos C., Muñoz-Merino P.J., Ibáñez-Espiga M.-B., Estévez-Ayres I., Panadero M. C. F.: Educational technology in the age of natural interfaces and deep learning. Rev. Iberoam. de Tecnol. del Aprendiz., 15(1), 2020, pp. 26-33.
81. Kolekar S. V., Sanjeevi S. G., Bormane D. S.: Learning style recognition using artificial neural networks for adaptive user interface in e-learning. Int. Conf. on Computational Intelligence and Computing Research, 2010, pp. 1-5.
82. Korbicz J., Maniewski R., Patan K., Kowal M. (Eds.): Current trends in biomedical engineering and bioimages analysis. Wyd, Springer, Berlin 2020.
83. Korłub W.: Optymalizacja strategii sieci inteligentnych agentów za pomocą programowania genetycznego w systemie rozproszonym realizującym paradygmat volunteer computing. Rozprawa doktorska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2016.
84. Kubiak M.: Wirtualna edukacja. Mikom, 2000.
85. Kumar N., Chilamkurti N., Zeadally S., Jeong Y-S.: Achieving quality of service (QoS) using resource allocation and adaptive scheduling in cloud computing with grid support. Comput. J., vol. 57(2), 2014, pp. 281-290.
86. Landowska A.: Wprowadzenie do zagadnień zdalnego nauczania. Raport Techniczny Wydziału ETI Politechniki Gdańskiej, vol. 2, 2007.
87. Leon M., Xiong N.: Investigation of mutation strategies in differential evolution for solving global optimization problems. Artificial Intelligence and Soft Computing, 2014, pp. 372-383.
88. Li Y., Wang S., Yang B.: An improved differential evolution algorithm with dual mutation strategies collaboration. Expert Syst. Appl., vol. 153, 2020, pp. 1-17.
89. Li Y., Wang S.: Differential evolution algorithm with elite archive and mutation strategies collaboration. Artif. Intell. Rev., vol. 53(6), 2020, pp. 4005-4050.
90. Liang Y., Rui Q. P., Xu J.: Computing resource allocation for enterprise information management based on cloud platform ant colony optimization algorithm. Adv. Mater. Res., vol. 791-793, 2013, pp. 1232-1237.
91. Lima S., Roch Á, Roque L.: An overview of OpenStack architecture: a message queuing services node. Cluster Computing, vol. 22, 2019, pp. 7087-7098.
92. Lin Y., Wang F., Hwang K.: A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problems. Congress on Evolutionary Computation, 1999, pp. 2159-2166.
93. Madni S. H. H., Latiff M.S. A., Coulibaly Y., Abdulhamid S. I. M.: An appraisal of meta-heuristic resource allocation techniques for IaaS cloud. Indian J. Sci. Technol., vol. 9(4), 2016, pp. 1-14.
94. Madni S. H. H., Latiff M. S. A., Coulibaly Y., Abdulhamid S. I. M.: Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. Cluster Comput, vol. 20(3), 2017, pp. 2489-2533.
95. Makasarwala H. A., Hazari P.: Using genetic algorithm for load balancing in cloud computing. The 8th Int. Conf. on Electronics, Computers and Artificial Intelligence, 2016, pp. 1-6.
96. Manvi S. S., Shyam G. K.: Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey. J. Netw. Comput. Appl., vol. 41, 2014, pp. 424-440.
97. Markovic L., Sofronijevic A.: Building a gamified system for caputring MOOC related data: Smart city learning community as its most precious source of intangible cultural heritage. Proc. of the Int. Conf. on Culture and Computing, Culture and Computing, 2015, pp. 175-182.

98. Martín A.C., Alario-Hoyos C., Kloos C.D.: Smart education: A review and future research directions. The 13th Int. Conf. on Ubiquitous Computing and Ambient Intelligence, 2019, pp. 1-10.
99. Microsoft Teams, https://www.microsoft.com/en/microsoft-365/microsoft-teams, dostęp: listopad 2022.
100. MNISW rekomendacja uczelniom – nauczanie w systemie zdalnym, https://www.gov.pl/web/nauka/mnisw-rekomenduje-uczelniom-nauczanie-w-systemie-zdalnym, dostęp: listopad 2022.
101. Mohana R. S.: A position balanced parallel particle swarm optimization method for resource allocation in cloud. Indian J. Sci. Technol., vol. 8(S3), 2015, pp. 1-7.
102. Moharkan Z. A., Choudhury T., Gupta S. C., Raj G.: Internet of Things and its applications in e-learning. The 3rd Int. Conf. on Computational Intelligence & Communication Technology, 2017, pp. 1-5.
103. Moodle dokumentacja, https://docs.Moodle.org, dostęp: listopad 2022.
104. Mpungose C. B.: Is Moodle or WhatsApp the preferred e-learning platform at a South African university? First-year students experiences. Educ. Inf. Technol., vol. 25(2), 2020, pp. 927-941.
105. Myicourse, http://myicourse.com/, dostęp: listopad 2022.
106. Nauka języka angielskiego przez komunikator Skype, http://www.angielski-skype.com/, dostęp: listopad 2022.
107. Nwana H. S., Ndumu D. T.: An introduction to agent technology. Software Agents and Soft Computing, 1997, pp. 3-26.
108. Opara K.R., Arabas J.: Differential evolution: A survey of theoretical analyses. Swarm Evol. Comput., vol. 44, 2019, pp. 546-558.
109. OpenStack dokumentacja, https://docs.openstack.org, dostęp: listopad 2022.
110. Page J., Naughton J.: Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. The 19th IEEE Int. Parallel and Distributed Processing Symposium, 2005, pp. 8-17.
111. Paluszak J.: Optymalizacja wykorzystania zasobów w systemach rozproszonych o architekturze typu grid. Rozprawa doktorska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2015.
112. Panda S. K., Jana P. K.: An efficient resource allocation algorithm for IaaS cloud. Distributed Computing and Internet Technology, 2015, pp. 351-355.
113. Pawar N., Sonkar S. K.: An approach towards e-learning using SVM classification technique and ranking technique in microblog supported classroom: A survey. Int. Journal of Emerging Technology and Advanced Engineering, vol. 3(7), 2013, pp. 315-322.
114. Pillai P. S., Rao S.: Resource allocation in cloud computing using the uncertainty principle of game theory. IEEE Syst. J., vol. 10(2), 2016, pp. 637-648.
115. Platforma ATutor, http://www.atutor.ca/, dostęp: listopad 2022.
116. Platforma Claroline, http://www.claroline.net/, dostęp: listopad 2022.
117. Platforma Fle3 Future Learning Env., http://fle3.uiah.fi/, dostęp: listopad 2022.
118. Platforma Ilias, http://www.ibm.com/developerworks/lotus/library/ls-elearning_evolution/, dostęp: listopad 2022.
119. Platforma Lotus, http://www.ibm.com/developerworks/lotus/library/lselearning_evolution/, dostęp: listopad 2022.
120. Platforma Manhattan Virtual Classroom, http://manhattan.sourceforge.net/, dostęp: listopad 2022.
121. Platforma Oracle iLearning, http://ilearning.oracle.com, dostęp: listopad 2022.
122. Platforma Pelp, http://www.pelp.net/, dostęp: listopad 2022.

123. Platforma WBTServer, http://polska.4system.com/, dostęp: listopad 2022.
124. Poradnik dla nauczycieli MEN, https://www.gov.pl/web/zdalnelekcje/poradnik-dla-nauczycieli, dostęp: listopad 2022.
125. Portal armii kanadyjskiej, http://www.armyelearning.ca/, dostęp: listopad 2022.
126. Portal e-learningowy armii amerykańskiej, http://www.army.mil/ako/, dostęp: listopad 2022.
127. Portal e-learningowy armii brytyjskiej, https://www.dlp.mod.uk/, dostęp: listopad 2022.
128. Portal e-learningowy armii francuskiej, http://www.itweb.co.za, dostęp: listopad 2022.
129. Portal firmy Benntec dotyczący zdalnego nauczania w armii niemieckiej, http://www.benntec.de/index.php/en/news/95-fernausbildungskongress-2010, dostęp: listopad 2022.
130. Portal ośrodka kształcenia na odległość OKNO Politechniki Warszawskiej, http://www.okno.pw.edu.pl/, dostęp: listopad 2022.
131. Portal platformy edukacyjnej Szkoły Głównej Handlowej, http://www.e-sgh.pl/, dostęp: listopad 2022.
132. Powszechna Licencja Publiczna GNU, http://gnu.org.pl, dostęp: listopad 2022.
133. Price K. V., Rönkkönen J.: Comparing the unimodal scaling performance of global and local selection in a mutation-only differential evolution algorithm. IEEE Congress on Evolutionary Computation 2006, 2006, pp. 2034-2041.
134. Project RDO, https://www.rdoproject.org, dostęp: listopad 2022.
135. Qian Z., Wang X., Liu X., Xie X., Song T.: An approach to dynamically assigning cloud resource considering user demand and benefit of cloud platform. Computing, vol 102(8), 2020, pp. 1817-1842.
136. Qin A., Suganthan P.: Self-adaptive differential evolution algorithm for numerical optimization. Congress on Evolutionary Computation, 2005, pp. 1785-1791.
137. Qin A., Huang V., Suganthan P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput., vol. 13(2), 2009, pp. 398-417.
138. Rackspace, https://www.rackspace.com, dostęp: listopad 2022.
139. Radhakrishnan A., Kavitha V.: Trusted virtual machine allocation in cloud computing IaaS service. Res. J. Appl. Sci. Eng. Technol., 7(14), 2014, pp. 2921-2928.
140. Rasiowa H.: Wstęp do matematyki współczesnej. PWN, Warszawa 2007.
141. Romero C., Ventura S., Hervás C., González P.: Rule discovery in web-based educational systems using grammar-based genetic programming. WIT Transactions on Information and Communication Technologies, 2005.
142. Sakailms, https://www.sakailms.org/, dostęp: listopad 2022.
143. Sayed M., Baker F.: E-learning optimization using supervised artificial neural-network. J. of Software Engineering and Applications, vol. 8, 2015, pp. 26-34.
144. Schaffer J. D.: Multiple objective optimization with vector evaluated genetic algorithm. In: Grefenstette J. J. (ed.): Genetic Algorithms and Their Applications: Proc. of the First Int. Conf. on Genetic Algorithms, 1985, pp. 93-100.
145. Schoology, https://www.schoology.com/, dostęp: listopad 2022.
146. Shaheen Q., Shiraz M., Khan S., Majeed R., Guizani M., Khan N., Aseere A. M.: Towards energy saving in computational clouds: taxonomy, review, and open challenges. IEEE Access, vol. 6, 2018, pp. 29407-29418.
147. Skype, https://www.skype.com/, dostęp: listopad 2022.

148. Soliman M., Guetl C.: Implementing intelligent pedagogical agents in virtual worlds: Tutoring natural science experiments in OpenWonderland. EDUCON, 2013, pp. 782-789.
149. Sridevi S., Uthariaraj R.: A survey of soft computing techniques applied in cloud load balancing. The Eighth Int. Conf. on Advanced Computing, 2017, pp. 131-137.
150. Srivastava A., Yammiyavar P.: Augmenting tutoring of students using tangible smart learning objects: An IOT based approach to assist student learning in laboratories. Proc. of the Int. Conf. on Internet of Things and Applications, 2016.
151. Stanley K. O., Bryant B. D., Miikkulainen R.: Real-time neuroevolution in the NERO video game. EEE Trans. Evol. Comput., vol. 9(6), 2005, pp. 653-668.
152. Storn R., Price K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, vol. 11(4), 1997, pp. 341-359.
153. Storn R.: Designing nonstandard filters with differential evolution. IEEE Signal Process. Mag., vol 22(1), 2005, pp. 103-106.
154. Strona poświęcona rozprawie, https://www.researchgate.net/project/Optimization-of-cloud-computing-resources-using-intelligent-agents-in-remote-teaching-2, dostęp: listopad 2022.
155. Sun A., Li Y.-J., Huang Y.-M., Li Q.: Using facial expression to detect emotion in e-learning system: A deep learning method. SETE@ICWL, 2017, pp. 446-455.
156. Susnea E.: Using artificial neural networks in e-learning systems. UPB Sci. Bull., Series C, vol. 72(4), 2010, pp. 91-100.
157. Szkolenia E-Pracownik, http://www.cisco.com/web/PL/seminaria/epracownik.html/, dostęp: listopad 2022.
158. Szymański, J., Krawczyk, H., Proficz, J., Orzechowski, P.: Categorization of Cloud Workload Types with Clustering. In Proc. of the Int. Conf. on Signal, Networks, Computing, and Systems. ICSNCS 2016, Volume 1, pp. 303-313.
159. Test CPU Mark, http://www.passmark.com/, dostęp: listopad 2022.
160. The Open University, http://www.openuniversity.edu/, dostęp: listopad 2022.
161. Tsai H. K., Yang J. M., Tsai Y. F., Kao C. Y.: An evolutionary algorithm for large Traveling Salesman Problems. IEEE Trans. Syst. Man Cybern. Part B, vol. 34(4), 2004, pp. 1718-1729.
162. Vasu R., Nehru E. I., Ramakrishnan G.: Load forecasting for optimal resource allocation in cloud computing using neural method. Middle East J. Sci. Res., vol. 24(6), 2016, pp. 1995-2002.
163. Wang C., Tsai C., Chiang M., Yang C.: An efficient local search for grid scheduling problem in learning system. The 9th Int. Conf. on Fuzzy Systems and Knowledge Discovery, 2012, pp. 2447-2451.
164. Wang F., Wang W., Yang H., Pan Q.: A novel discrete differential evolution algorithm for computer-aided test-sheet composition problems. Int. Conf. on Information Engineering and Computer Science, 2009, pp. 1-4.
165. Wang H., Wang F., Liu J., Wang D., Groen J.: Enabling customer-provided resources for cloud computing: potentials, challenges, and implementation. IEEE Trans. Parallel Distrib. Syst., vol. 26(7), 2015, pp. 1874-1886.
166. Wang T., Liu Z., Chen Y., Xu Y., Dai X.: Load balancing task scheduling based on genetic algorithm in cloud computing. DASC, 2014, pp. 146-152.
167. Wang W., Jiang Y., Wu W.: Multiagent-based resource allocation for energy minimization in cloud computing systems. IEEE Trans. Syst. Man Cybern. Syst., vol. 47(2), 2017, pp. 205-220.

168. Winkler R., Hobert S., Appius M.: Empowering educators to build their own conversational agents in online education. ECIS, 2020.
169. Wooldridge M., Jennings N. R.: Agent theories, architectures, and languages: A survey. ECAI Workshop on Agent Theories, Architectures, and Languages, 1994, pp. 1-39.
170. Wu J., Feng Q.: Recommendation system design for college network education based on deep learning and fuzzy uncertainty. J. Intell. Fuzzy Syst., vol. 38(6), 2020, pp. 7083-7094.
171. Xu D., Huang W. W., Wang H., Heales J.: Enhancing e-learning effectiveness using an intelligent agent-supported personalized virtual learning environment: An empirical investigation. Inf. Manag., vol. 51(4), 2014, pp. 430-440.
172. Xu R., Chen J., Han J., Tan L., Xu L.: Towards emotion-sensitive learning cognitive state analysis of big data in education: deep learning-based facial expression analysis using ordinal information. Computing, vol. 102(3), 2020, pp. 765-780.
173. Xue F., Sanderson A. C., Graves R. J.: Multi-objective differential evolution and its application to enterprise planning. ICRA, 2003, pp. 3535-3541.
174. Yang C.-T., Wan T.-Y.: Implementation of an energy saving cloud infrastructure with virtual machine power usage monitoring and live migration on OpenStack, Computing, vol. 102(6), 2020, pp. 1547-1566.
175. Yang Z., Liu M., Xiu J., Liu C.: Study on cloud resource allocation strategy based on particle swarm ant colony optimization algorithm. IEEE 2nd Int. Conf. on Cloud Computing and Intelligent Systems, 2012, pp. 488-491.
176. Yang Z., Tang K., Yao X.: Self-adaptive differential evolution with neighborhood search. IEEE Congress on Evolutionary Computation, 2008, pp. 1110-1116.
177. Ye J.: Modeling of performance evaluation of educational information based on big data deep learning and cloud platform. J. Intell. Fuzzy Syst., vol. 38(6), 2020.
178. Younge A. J., Von Laszewski G., Wang L., Lopez-Alarcon S., Carithers W.: Efficient resource management for cloud computing environments. Int. Conf. on Green Computing, 2010, pp. 357-364.
179. Zacniewski A., Optymalizacja alokacji modułów programistycznych w rozproszonym systemie szkolenia wojskowego, Rozprawa doktorska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska 2011.
180. Zafra A., Romero C., Ventura S.: Predicting academic achievement using multiple instance genetic programming. ISDA, 2009, pp. 1120-1125.
181. Zaiane O.E.: Building a recommender agent for e-learning systems. Proc. of the Int. Conf. on Computers in Education, 2002, pp. 55-59.
182. Zaydman O., Zhirin R.: Teleportation of VM disk images over WAN. Cloud, 2019.
183. Zhang H., Huang T., Liu S., Yin H., Li J., Yang H., Xia Y.: A learning style classification approach based on deep belief network for large-scale online education. J. Cloud Computing, vol. 9, 2020, pp. 1-17.
184. Zhang Q., Cheng L., Boutaba R.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl., vol. 1(1), 2010, pp. 7-18.
185. Zhang Y., Gong D.-W., Gao X.-G., Tian T.: Binary differential evolution with self-learning for multi-objective feature selection. Inf. Sci., vol. 507, 2020, pp. 67-85.
186. Zheng Z., Xie K., He S., Deng J.: A Multi-objective Optimization Scheduling Method Based on the Improved Differential Evolution Algorithm in Cloud Computing. ICCCS, vol. 1, 2017, pp. 226-238.
187. Zhu Z.T., Yu M.H., Riezebos P.: A research framework of smart education. Smart Learn. Environ. vol. 3, 2016, pp. 1-17.
188. Zoom, https://zoom.us/, dostęp: listopad 2022.

SPIS RYSUNKÓW

Rysunek 1 Rodzaje e-learningu w inteligentnej edukacji [179]. 12
Rysunek 2 Względna częstotliwość wyszukiwania fraz Distance Education oraz E-learning za pomocą wyszukiwarki Google w latach 2019 – 2022 [60]. 13
Rysunek 3 Zainteresowanie wybranymi platformami zdalnego nauczania [60]. 15
Rysunek 4 Architektura systemu agentowego HBSS [179]. 17
Rysunek 5 Agenty pedagogiczne w e-learningu [43]. 19
Rysunek 6 Graficzna interpretacja mutacji różnicowej w przestrzeni dwuwymiarowej [133]. 26
Rysunek 7 Diagram algorytmu ewolucji różnicowej do optymalizacji jednokryterialnej. 27
Rysunek 8 Względna odległość rozwiązań efektywnych od brzegu Pareto dla wybranych wartości metaparametrów ewolucji różnicowej. 28
Rysunek 9 Minimalizacja długości trasy w instancji problemu TSP za pomocą algorytmu ewolucji różnicowej. 30
Rysunek 10 Ilustracja wyznaczania rang za pomocą procedury Fonseci-Fleminga. 31
Rysunek 11 Diagram wielokryterialnego algorytmu ewolucji różnicowej Abbasa-Sarkera. 32
Rysunek 12 Komunikacja z podwodnym dronem za pomocą usługi Hardware as a Service. 35
Rysunek 13 Agenty pedagogiczne na platformie OpenStack. 36
Rysunek 14 Rodzaje hostów i węzły pomocnicze w środowisku obliczeniowym OpenStack. 36
Rysunek 15 Adresowanie komponentów chmury opartej na platformie OpenStack. 37
Rysunek 16 Diagram zarządzania usługami przez KeyStone. 38
Rysunek 17 Szacowanie wykorzystania zasobów za pomocą usługi Horizon. 39
Rysunek 18 Akwizycja danych i prognoza wyników w systemie zdalnego nauczania. 42
Rysunek 19 Wybrane elementy demonstratora chmury edukacyjnej. 43
Rysunek 20 Podział metod zarządzania zasobami w chmurze obliczeniowej [94]. 44
Rysunek 21 Wybrane oszacowania obciążenia CPU newralgicznego hosta oraz kosztów hostów dla instancji B90. 57
Rysunek 22 Wybrane oszacowania obciążenia CPU newralgicznego hosta oraz obciążenia transmisją danych newralgicznego węzła dla instancji B90. 58
Rysunek 23 Wybrane oszacowania obciążenia komunikacyjnego newralgicznego węzła oraz kosztu hostów dla instancji B90. 58
Rysunek 24 Wybrane oszacowania obciążenia CPU newralgicznego hosta oraz łącznej mocy poboru energii elektrycznej przez hosty chmury dla instancji B90. 58
Rysunek 25 Wybrane wartości kryteriów skalarnych $F3$ oraz $F4$ w wypadku instancji B90. 59
Rysunek 26 Wybrane wartości kryteriów skalarnych $F2$ oraz $F4$ w wypadku instancji B90. 59
Rysunek 27 Diagram mutacji tabu search w ewolucji różnicowej. 65
Rysunek 28 Diagram wielokryterialnego algorytmu ewolucji różnicowej DEMCA. 67
Rysunek 29 Migracja ocen lokalnie niezdominowanych w kierunku brzegu Pareto na wybranych etapach przebiegu algorytmu DEMCA dla q=1, Cp=0,1. 73
Rysunek 30 Zbieżność ocen rozwiązań niezdominowanych wyznaczonych za pomocą ewolucji różnicowej dla instancji B306. 76
Rysunek 31 Oceny rozwiązań wyznaczone przez wybrane metaheurystyki dla instancji B306. 76

Rysunek 32 Uśredniona zbieżność ocen dla wybranych wielokryterialnych metaheurystyk. ... 77
Rysunek 33 Oceny niezdominowane na wybranych etapach ewolucji różnicowej dla instancji B855. ... 78
Rysunek 34 Porównanie wyników wyznaczonych za pomocą algorytmu DEMCA z ocenami otrzymanymi za pomocą innych metaheurystyk. ... 79
Rysunek 35 Wybór rozwiązania kompromisowego ze zbioru Pareto po normalizacji względem punktu idealnego i punktu nadir. ... 80
Rysunek 36 Rozlokowanie hostów demonstratora chmury edukacyjnej oraz możliwe rozszerzenia. ... 81
Rysunek 37 Rozmieszczenie czterech maszyn wirtualnych na dwóch hostach w chmurze nr 1 demonstratora. ... 85
Rysunek 38 Rozmieszczenie szesnastu maszyn wirtualnych w chmurze nr 2 demonstratora. ... 86
Rysunek 39 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do kosztu hostów i mocy poboru energii elektrycznej przez hosty. ... 89
Rysunek 40 Migracja ocen rozwiązań sprawnych na wybranych etapach ewolucji w odniesieniu do obciążenia CPU newralgicznego hosta i obciążenia transmisją danych newralgicznego węzła. ... 89
Rysunek 41 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia transmisją danych newralgicznego węzła i kosztu hostów. ... 89
Rysunek 42 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia CPU newralgicznego hosta i kosztu hostów. ... 90
Rysunek 43 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia CPU newralgicznego hosta i zużycia energii hostów. ... 90
Rysunek 44 Migracja ocen rozwiązań niezdominowanych na wybranych etapach ewolucji różnicowej w odniesieniu do obciążenia komunikacyjnego newralgicznego węzła i zużycia energii hostów. ... 90
Rysunek 45 Alokacja maszyn wirtualnych w hostach projektowanego rozszerzenia demonstratora chmury edukacyjnej dla wyznaczonego rozwiązania kompromisowego z parametrem p=2. ... 91
Rysunek 46 Wybrane topologie równoległych wersji algorytm DEMCA. ... 92
Rysunek 47 Minimalizacja łącznej mocy poboru energii elektrycznej dla równoległych wersji algorytmu DEMCA. ... 92
Rysunek 48 Zawartość archiwum aplikacji DEMCA dla platformy a) Linux b) Windows. ... 109
Rysunek 49 Ekran startowy programu DEMCA. ... 110
Rysunek 50 Interfejs graficzny programu DEMCA. ... 110
Rysunek 51 Zakładki konfiguracyjne aplikacji: a) część górna b) część dolna. ... 111

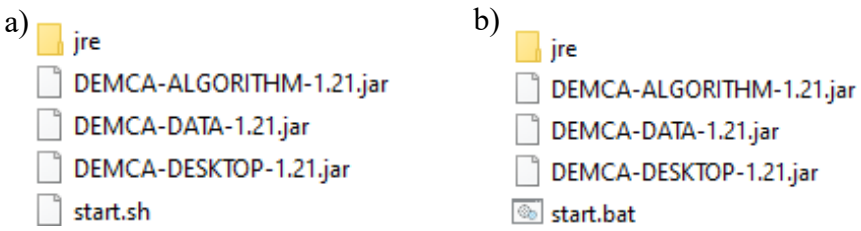
SPIS TABEL

Tabela 1 Długości tras wyznaczone za pomocą algorytmu ewolucji różnicowej dla instancji problemu TSP. 29
Tabela 2 Wersje baz danych wspierane przez system Moodle w wersji 3.11 [103]. 40
Tabela 3 Standardowe role dostępne w systemie Moodle [103]. 41
Tabela 4 Klasyfikacja algorytmów równoważących obciążenie [149]. 46
Tabela 5 Zbiór rozwiązań Pareto-optymalnych dla instancji B90. 73
Tabela 6 Uśredniona procentowa odległość rozwiązań wyznaczonych za pomocą algorytmu DEMCA od brzegu Pareto dla wybranych strategii zmian metaparametrów Cp i q. 74
Tabela 7 Specyfikacja rozwiązań i ocen sprawnych wyznaczonych na pomocą algorytmu DEMCA. 76
Tabela 8 Oceny rozwiązań sprawnych wyznaczonych za pomocą algorytmu DEMCA dla instancji B855. 78
Tabela 9 Oceny wyznaczone za pomocą algorytmów DEMCA, MOGPA, MOHS, NSGA-II i OMOPSO. 79
Tabela 10 Parametry wybranych rodzajów serwerów. 82
Tabela 11 Wymagane zasoby dla wybranych obrazów maszyn wirtualnych na platformie OpenStack. 83
Tabela 12 Uśrednione zapotrzebowanie na zasoby dla maszyn wirtualnych na hoście referencyjnym H_3. 84
Tabela 13 Rozmieszczenie dwudziestu maszyn wirtualnych na dziewięciu hostach w demonstratorze chmury edukacyjnej przed modernizacją. 85
Tabela 14 Specyfikacja rozwiązań Pareto-optymalnych dla instancji B1024 – część 1. 87
Tabela 15 Specyfikacja rozwiązań Pareto-optymalnych dla instancji B1024 – część 2. 88

DODATEK. Aplikacja algorytmu ewolucji różnicowej DEMCA

Aplikację wielokryterialnego algorytmu ewolucji różnicowej DEMCA napisano w języku Java w wersji 11 przy użyciu biblioteki graficznej JavaFX. Aplikacja typu desktop składa się z trzech modułów programistycznych. Moduł DEMCA-DATA parsuje parametry wejściowe, a moduł DEMCA-DESKTOP jest interfejsem graficznym użytkownika. Natomiast moduł DEMCA-ALGORYTM to implementacja algorytmu ewolucji różnicowej, algorytmu przeszukiwania losowego oraz algorytmu metody systematycznego przeglądu.

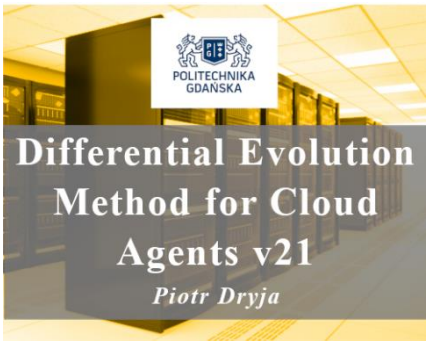
Przygotowano wersje aplikacji dla systemu Linux Fedora 32 i dla systemu Windows 10 i 11. Po „rozpakowaniu" archiwum w katalogu są cztery pliki (rys. 48). W wypadku Windowsa należy dwukrotnie kliknąć w ikonkę pliku *start.bat*, a w wypadku Linuxa należy ustawić plik *start.sh* jako plik wykonywalny i uruchomić w linii poleceń konsoli.



Rysunek 48 Zawartość archiwum aplikacji DEMCA dla platformy a) Linux b) Windows. Źródło: opracowanie własne.

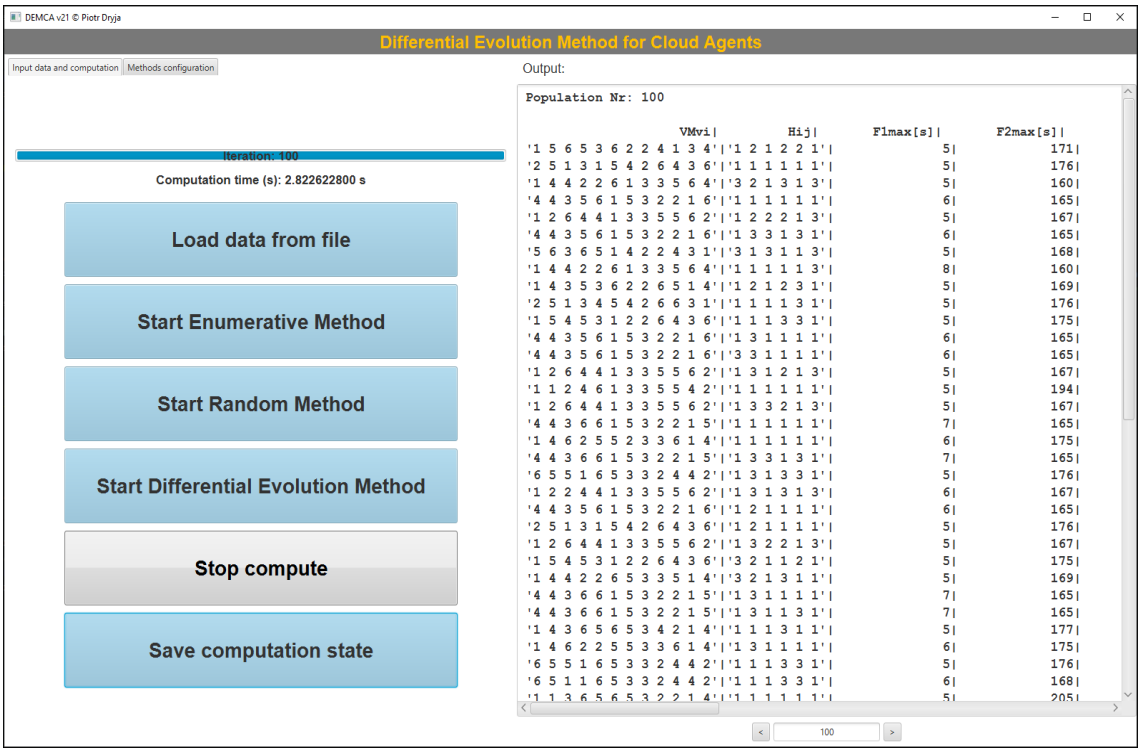
Po uruchomieniu aplikacji widoczny jest ekran startowy (rys. 49), a po załadowaniu programu do pamięci RAM uwidacznia się interfejs graficzny, który składa się z dwóch części (rys. 50). W lewej części interfejsu dostępne są zakładki, a część prawa zawiera wyniki obliczeń i umożliwia podgląd wyników dla wybranych populacji.

Zakładka *Input data and computation* umożliwia wczytanie danych za pomocą przycisku *Load data from file*. Dane są w formacie plików tekstowych z rozszerzeniem *txt*. Ponadto trzy inne przyciski pozwalają na uruchomienie wybranych algorytmów. Opcja *Start Enumertive Method* uruchamia algorytm systematycznie przeszukujący wszystkie możliwe rozwiązania. Opcja *Start Random Method* rozpoczyna działanie algorytmu przeszukiwania losowego, a przycisk *Start Differential Evolution Method* - algorytmu ewolucji różnicowej. Natomiast opcja *Stop compute* zatrzymuje obliczenia. Przycisk *Save computation state* zapisuje rozwiązania wyznaczone podczas ostatniej iteracji algorytmu.



Rysunek 49 Ekran startowy programu DEMCA.
Źródło: opracowanie własne

Zakładka *Methods configuration* pozwala na ustawienie wartości parametrów algorytmów oraz aplikacji (rys. 51a). Sekcja *Output* tej zakładki umożliwia konfigurację częstotliwości wyświetlanych rezultatów oraz ustawienie opcji zapisu wyników do pliku w formacie CSV. Sekcja umożliwia również ustawienie zapisu tylko poprawnych rozwiązań oraz uruchomienie trybu podglądu wyników pośrednich.



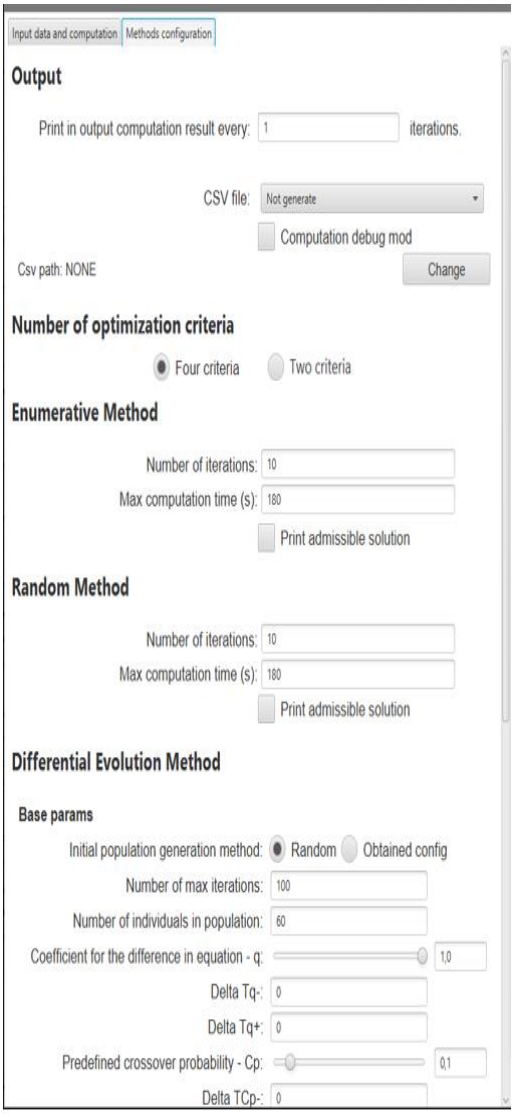
Rysunek 50 Interfejs graficzny programu DEMCA.
Źródło: opracowanie własne.

Następna sekcja umożliwia wybór kryteriów optymalizacji. Dwie kolejne sekcje pozwalają na ustawienie ograniczeń na maksymalną liczbę generacji i na maksymalny czas obliczeń. Ponadto ustawiana może być opcja pozwalająca na wyświetlanie dopuszczalnych rozwiązań podczas obliczeń. Ostatnia z sekcji pozwala na wybór parametrów algorytmu ewolucji różnicowej.

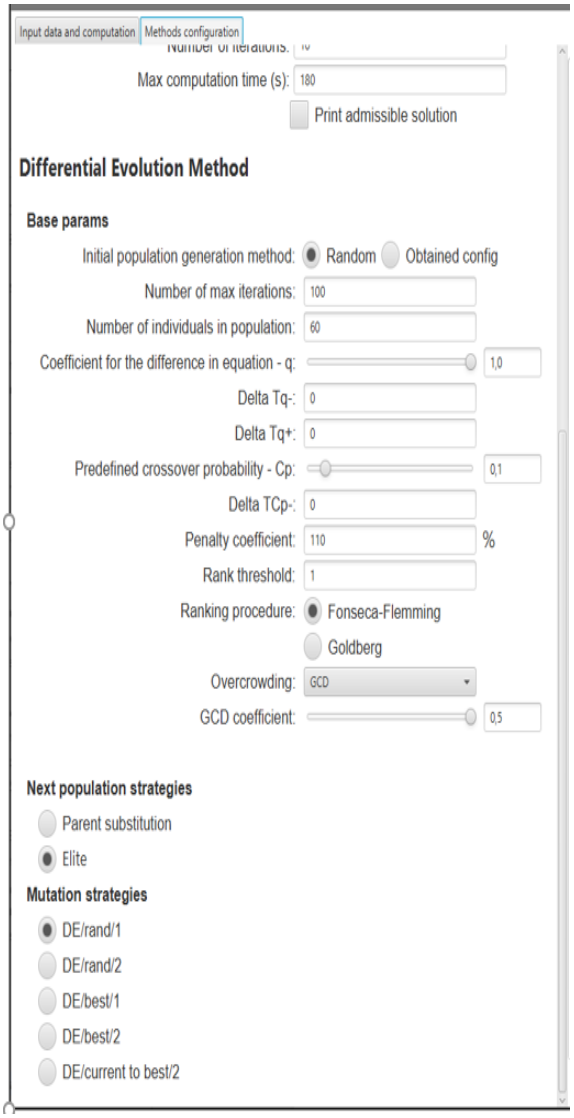
Pierwsza jej część umożliwia wybór procedury wyznaczania populacji zerowej, która może zostać wygenerowana losowo lub wprowadzona z pliku. Ponadto ustawione mogą być takie parametry, jak: liczba populacji, liczba osobników w populacji, współczynnik skalujący, próg preferowania genów potomków, współczynnik skalujący rangi rozwiązań niezdominowanych oraz możliwość zastosowania geometrycznej miary zagęszczenia.

Dostępne są również procedury modyfikacji współczynnika skalującego i współczynnika krzyżowania. Wyborowi podlegają również dwa sposoby selekcji do następnej populacji: elitarystyczny oraz podstawowy. Można również wybrać strategię mutacji.

a)



b)



Rysunek 51 Zakładki konfiguracyjne aplikacji: a) część górna b) część dolna.
Źródło: opracowanie własne.

Aplikację oraz dane do instancji, które rozwiązano za pomocą powyższej aplikacji umieszczono na stronie rozprawy [154]. Ponadto benchmarki umieszczono na płycie DVD dołączonej do rozprawy.