POLITECHNIKA GDAŃSKA
Wydział Elektroniki, Telekomunikacji i Informatyki

Artur Zacniewski

Optymalizacja alokacji modułów programistycznych w rozproszonym systemie szkolenia wojskowego

Rozprawa doktorska

Promotor:
dr hab. inż. Jerzy Balicki, prof. nadzw. PG
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Gdańsk, 2010

SPIS TREŚCI

WPROWADZENIE

W doktrynie militarnych działań sieciocentrycznych, która w ostatnich latach zdobywa uznanie wśród strategów i dowódców wiodących armii świata, zakłada się wysoki poziom wyszkolenia żołnierzy pod względem umiejętności wykorzystania nowoczesnych technologii informatycznych. Ze względu na rozproszoną dyslokację wojsk oraz intensywne zmiany w technologii komputerowej, koniecznym jest prowadzenie systematycznych szkoleń wojskowych.

Na tym tle, perspektywicznym rozwiązaniem jest zastosowanie platform zdalnego szkolenia w odniesieniu do żołnierzy przebywających w różnych częściach świata. W związku z tym, w wojskach zarówno należących do NATO, jak i tych spoza układu, zdalne szkolenie wojskowe zaczyna odgrywać priorytetową rolę. Rządy państw, często przy pomocy profesjonalnych firm informatycznych oraz uniwersyteckich zespołów badawczych, inwestują znaczące fundusze w rozwój informatycznych systemów zdalnego szkolenia. Dąży się do tego, aby żołnierz miał dostęp do zasobów edukacyjnych z każdego miejsca na świecie.

Nauczanie na odległość jest metodą prowadzenia procesu dydaktycznego w warunkach, gdy nauczyciele i słuchacze nie przebywają w tym samym miejscu, przy czym stosują do przekazywania wiedzy – oprócz tradycyjnych sposobów komunikowania się – nowoczesne technologie internetowe. Umożliwia to również bezpośredni kontakt w czasie rzeczywistym między nauczycielem a słuchaczem za pomocą audio- lub wideokonferencji, niezależnie od odległości, jaka ich dzieli [94].

Przedmiotem poznania w rozprawie jest wielokryterialna optymalizacja systemów zdalnego szkolenia wojskowego z wykorzystaniem algorytmów ewolucyjnych. Algorytmy ewolucyjne intensywnie rozwijają się od około dziesięciu lat w obszarze wyznaczania reprezentacji rozwiązań optymalnych w sensie Pareto. Jednakże nie zastosowano ich dotychczas do optymalizacji systemów zdalnego szkolenia wojskowego i tę właśnie „lukę” zamierza się wypełnić za pomocą niniejszej dysertacji.

Proces kształcenia na odległość zapoczątkowano około 1700 roku w Ameryce Północnej. Nauka polegała wówczas na przesyłaniu materiałów i zadań za pomocą listów. Pierwszy uniwersytet stosujący tę formę kształcenia założono w 1883 roku w Nowym Jorku pod nazwą Uniwersytet Nauki Korespondencyjnej. Natomiast Międzynarodowa Szkoła Korespondencyjna rozpoczęła działalność w 1890 roku.

Ważnym etapem w rozwoju kształcenia na odległość było wykorzystywanie edukacyjnych programów radiowych. Pierwszy taki program wyemitowano na Uniwersytecie Stanu Iowa w 1925 roku. Wykłady radiowe były także bardzo popularne na słabo zaludnionych obszarach Australii. W latach siedemdziesiątych dwudziestego wieku doskonalono edukacyjne programy radiowe, a wykorzystywano je głównie do kształcenia mieszkańców z obszarów wiejskich.

Telewizja edukacyjna rozpoczęła funkcjonowanie w 1945 roku na Uniwersytecie Stanu Iowa. Od 1952 roku dzięki Radzie Telewizji Edukacyjnej w USA powstało wiele stacji edukacyjnych, a sale lekcyjne wyposażono w odbiorniki TV.

Kiedy w 1962 roku pierwszy amerykański satelita telekomunikacyjny Telstar 1 został umieszczony na orbicie okołoziemskiej, zaczęto stosować w nauczaniu na odległość tele- i wideokonferencje. Prekursorami w tym zakresie byli wykładowcy z Uniwersytetu Alaska, którzy udostępniali kursy uczelniom w Stanach Zjednoczonych.

W Polsce nauczanie na odległość realizowane jest już od około 200 lat. Na Uniwersytecie Jagiellońskim wprowadzono wykłady korespondencyjne dla studentów przebywających poza uczelnią. Na przełomie XIX i XX wieku założono Towarzystwo Kursów Akademickich dla Kobiet oraz organizację Powszechne Wykłady Uniwersyteckie. W latach 1966-1971 funkcjonowała Politechnika Telewizyjna [71].

W latach dziewięćdziesiątych XX wieku coraz większy wpływ na proces nauczania miał rozwój telekomunikacji cyfrowej, łączności satelitarnej oraz technik komputerowych i multimedialnych. Przełom nastąpił wraz z rozwojem Internetu. Edukacja przez Internet wykorzystuje zaawansowane technologie pozwalające na powszechny i niedrogi dostęp do multimedialnych materiałów edukacyjnych [165].

Zaletą kształcenia na odległość jest likwidowanie barier w dostępie do edukacji, uwzględnianie indywidualnych możliwości, potrzeb i oczekiwań osób uczących się. Istotnym jest także fakt, że wprowadzenie takiego modelu nauczania pozwala ograniczyć wydatki na szkolenie. Na stronach WWW polskich uczelni coraz częściej pojawiają się oferty nauczania na odległość. Warto także podkreślić, że internetowe technologie wspierające zdalne nauczanie stanowią najszybciej rozwijającą się gałąź rynku IT (ang. *Internet Technology*). *E-learning* będzie się intensywnie rozwijał ze względu na rozwój: szerokopasmowego Internetu, bezprzewodowej technologii przesyłu danych, a także urządzeń przenośnych [41].

Ten sposób edukacji został również wprowadzony do szkolnictwa wojskowego. W armii amerykańskiej w 1999 roku uruchomiono portal edukacyjny AKO (ang. *Army*

Knowledge Online). Rząd Stanów Zjednoczonych od wielu lat kładzie duży nacisk na zdobywanie wiedzy przez kadrę i pracowników wojska poprzez Internet. Utworzono nawet specjalny program zdalnego nauczania dla małżonków żołnierzy, którzy z racji charakteru pracy partnera nie mogą poszerzać wiedzy w tradycyjny sposób.

W polskiej armii, jak do tej pory, nauczanie przez Internet praktycznie realizowane jest tylko w uczelniach wojskowych. Informatyczne systemy szkolenia wojskowego zawierają wiele cech wspólnych z systemami nauczania stosowanymi na uczelniach cywilnych, dlatego w dalszej części pracy nie będzie wyraźnego rozgraniczenia między zdalnym nauczaniem w szkolnictwie wojskowym a zdalnym nauczaniem w szkolnictwie cywilnym.

W rozprawie zakłada się, że kluczowa jest jakość systemu informatycznego wspomagającego nauczanie. Jakość tej klasy systemu można poprawić w odniesieniu do jego dostępności oraz skuteczności terminowego wykonywania zadań, za pomocą dekompozycji oprogramowania na moduły, a następnie dokonania odpowiedniej ich alokacji.

Celem pracy jest *optymalizacja przydziałów modułów programistycznych systemu MOODLE, wykorzystywanego w zdalnym szkoleniu wojskowym, uwzględniając maksymalizację dostępności systemu, a także maksymalizację skuteczności realizacji zadań w terminach.*

Na podstawie tak określonego celu sformułowano następujące zagadnienia badawcze:

Należy opracować wielokryterialny algorytm ewolucyjny oraz wykonać jego implementację w aspekcie optymalizacji systemu zdalnego szkolenia wojskowego, który funkcjonuje z wykorzystaniem oprogramowania MOODLE.

Zdeterminowanie głównego celu pracy oraz zagadnienia badawczego pozwoliło na sprecyzowanie następującej hipotezy roboczej:

Optymalizacja alokacji modułów programistycznych, przy wykorzystaniu wielokryterialnego algorytmu ewolucyjnego, umożliwia znaczące zwiększenie dostępności systemu zdalnego szkolenia wojskowego oraz zwiększenie skuteczności realizacji zadań w porównaniu do systemu, który nie jest optymalizowany.

Przy rozwiązywaniu problemów badawczych i weryfikacji przyjętej hipotezy roboczej korzystano z dorobku teorii naukowych dotyczących przydziałów i szeregowania zadań, optymalizacji wielokryterialnej, algorytmów ewolucyjnych oraz

systemów informatycznych. Wybór metod badawczych i zakres ich wykorzystania wynikał z charakteru rozpatrywanego problemu.

Stosunkowo dużą rolę w procesie badawczym odegrała krytyczna analiza literatury przedmiotu. Modelowanie stosowane było zarówno na poziomie rozważań abstrakcyjnych, jak też do wspomagania eksperymentów numerycznych. Definiując pojęcia z zakresu przydziałów modułów oraz przesyłania danych między modułami, wspierano się teorią grafów. Do implementacji opracowanych algorytmów, wyznaczających rozwiązania sformułowanych zadań optymalizacji, zastosowano języki programowania obiektowego MATLAB i C++.

System metodologiczny wykorzystania algorytmów ewolucyjnych do optymalizacji przydziałów modułów programów w systemach zdalnego szkolenia wojskowego powinien spełniać trzy warunki: spójności logicznej, zwięzłości i otwartości. Brak takiego systemu metodologicznego powoduje, że badacze mogą usiłować rozwiązywać problemy, co do których już wcześniej podano odpowiednie metody. Luka powstała wskutek nieopracowania systemu metodologicznego hamuje rozwój teorii i zastosowań optymalizacji przydziałów zadań w systemach zdalnego szkolenia wojskowego.

Praca składa się z czterech rozdziałów.

Rozdział pierwszy zawiera charakterystykę zdalnego nauczania w wybranych uczelniach i organizacjach. Omówiono wykorzystanie systemów i portali zdalnego nauczania w szkolnictwie wojskowym w armiach amerykańskiej, niemieckiej, francuskiej, kanadyjskiej oraz armii polskiej. Przedstawiono charakterystykę informatycznych systemów rozproszonych. Opisane zostały wybrane platformy zdalnego nauczania. Scharakteryzowano szczegółowo system *MOODLE*, którego struktura została wykorzystana w rozprawie do eksperymentów numerycznych.

W rozdziale drugim odniesiono się do wybranych modeli przydziału modułów programistycznych. Przedstawiono podstawowe założenia modelu szkolenia wojskowego rozpatrywanego w rozprawie. Określono zbiór dopuszczalnych przydziałów modułów do komputerów oraz kryteria oceny jakości tychże przydziałów. Omówiono wykorzystanie agentów programistycznych w systemach zdalnego nauczania opartych o platformę *MOODLE*.

W rozdziale trzecim zawarto zasady pozwalające na maksymalizację dostępności systemu i miary skutecznej realizacji modułów w terminach. Omówiono sposoby wyznaczania wartości kryteriów ocen jakości przydziałów modułów do

komputerów. Pokazano jak wyznaczyć przydziały Pareto-optymalne, rozwiązania leksykograficzne oraz rozwiązania kompromisowe.

W rozdziale czwartym dokonano przeglądu metod optymalizacji wektorowej. Przedstawiono wybrane algorytmy ewolucyjne, szczegółowo omówiono algorytmy NSGA-II i AMEA, a także znacząco zmodyfikowany algorytm NSGA-III. Scharakteryzowano implementację wielokryterialnego algorytmu ewolucyjnego w środowisku MATLAB. Opisano wyniki eksperymentów numerycznych oraz wskazano zastosowania rozpatrywanego systemu. Zaprezentowano także koncepcję oryginalnego algorytmu ewolucyjnego działającego na populacji sieci neuronowych.

W dodatkach opisano oprogramowanie do wspomagania pracy administratora systemu *MOODLE* oraz narzędzie do wyznaczania przydziału modułów systemu *MOODLE*.

Wykaz ważniejszych oznaczeń

α_n – czas rozpoczęcia wykonywania programu rozproszonego $\boldsymbol{P_n}$;

$\boldsymbol{A}=\{A_1,\ldots, A_v,\ldots, A_V\}$ – zbiór maksymalnych liczb wykonań modułu w programie;

β_n – czas ukończenia wykonywania programu rozproszonego $\boldsymbol{P_n}$;

$\boldsymbol{C}=\{C_1,\ldots, C_v,\ldots, C_V\}$ – zbiór czasów ukończenia procesów;

$C_j^{\max}$ – najpóźniejszy moment ukończenia procesu na komputerze typu j;

$C_{\max}$ – najpóźniejszy moment ukończenia procesu w systemie;

$\boldsymbol{d}=\{d_1,\ldots,d_v,\ldots,d_V\}$ – zbiór zadanych najpóźniejszych dopuszczalnych momentów ukończenia procesów, które to zakończenie nie spowoduje przekroczenia czasu założonego na wykonanie programu;

$\boldsymbol{E}=\{e_1,\ldots,e_v,\ldots,e_V\}$ – zbiór czasów rozpoczęcia wykonywania modułów niezależnych czasowo od innych modułów w systemie;

$\boldsymbol{G_F}$ – diagram przepływu systemu rozproszonego;

$\boldsymbol{\Theta}=\{\Theta_1,\ldots, \Theta_v,\ldots, \Theta_V\}$ – zbiór określający, w jakim typie struktury diagramu reprezentującego instancję programu znajduje się zadany moduł;

IW – wejściowa macierz wag w sztucznej dwuwarstwowej sieci neuronowej;

L – rozmiar populacji rozwiązań w algorytmie ewolucyjnym;

L_c – parametr określający, jaka część populacji może być wymieniana między osobnikami podczas krzyżowania w algorytmie ewolucyjnym;

L_{INST} – liczba instancji diagramu przepływu $\boldsymbol{G_F}$;

LW – macierz wag warstwy ukrytej w sztucznej dwuwarstwowej sieci neuronowej;

$\boldsymbol{\Lambda}=\{\lambda_1,\ldots,\lambda_j,\ldots,\lambda_J\}$ – zbiór parametrów rozkładów wykładniczych dla rodzajów komputerów;

$\boldsymbol{M}=\{m_1,\ldots,m_v,\ldots,m_V\}$ – zbiór modułów programistycznych;

$\overline{\boldsymbol{M}}=\{M_1,\ldots,M_m,\ldots,M_M\}$ – zbiór zadań wyznaczony w wyniku przetwarzania rozproszonego;

$\boldsymbol{P_n}$ – program rozproszony;

P_{time} – kryterium skuteczności realizacji zadań w terminach;

$\boldsymbol{\Pi}=\{\pi_1,\ldots,\pi_j,\ldots,\pi_J\}$ – zbiór typów komputerów;

R – dostępność systemu informatycznego wspomagającego zdalne nauczanie;

S_i – stan spełnienia ograniczeń czasowych w odniesieniu do i-tej instancji diagramu przepływu;

$\boldsymbol{T}$ – macierz szacowanych czasów realizacji modułów programu na komputerach;

$\overline{\boldsymbol{T}}$ – macierz szacowanych kosztów przetwarzania danych za pomocą modułów na komputerach;

T_{gr} – największe dopuszczalne obciążenie węzła w systemie;

$\boldsymbol{\tau}$ – macierz szacowanych czasów komunikacji danych między parami modułów;

$\overline{\boldsymbol{\tau}}$ – macierz szacowanych kosztów interakcji między parami modułów;

$\boldsymbol{W}=\{w_1,\ldots,w_i,\ldots,w_I\}$ – zbiór węzłów z komputerami;

$\boldsymbol{X}$ – zbiór rozwiązań dopuszczalnych;

x^m – binarny wektor przydziału modułów programistycznych do węzłów;

X^m – całkowitoliczbowy wektor przydziału modułów programistycznych do węzłów;

x^π – binarny wektor przydziału komputerów do węzłów;

X^π – całkowitoliczbowy wektor przydziału komputerów do węzłów;

$\boldsymbol{Y}$ – obraz zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$;

$\boldsymbol{\Omega}$ – macierz przepływu, określająca kolejność wykonywania się modułów;

■ – koniec dowodu.

1. PRZEGLĄD INFORMATYCZNYCH SYSTEMÓW ROZPROSZONYCH W NAUCZANIU CYWILNYM I SZKOLENIU WOJSKOWYM

Intensywny rozwój technologii informatycznych powoduje ewoluowanie form kształcenia na odległość. Wiele organizacji edukacyjnych, uczelni, szkół i instytucji wojskowych zmaga się z zapotrzebowaniem na szkolenia, kursy, a także dydaktyczne materiały w formie elektronicznej. Dostępnych jest wiele platform zdalnego nauczania, zarówno komercyjnych, jak i bezpłatnych, a wybór tej najodpowiedniejszej jest złożonym zagadnieniem.

1.1. Zdalne nauczanie i szkolenie w kraju i na świecie

Kształcenie na odległość umożliwia uczenie się studentów (lub innych osób szkolonych, w tym żołnierzy) od nauczyciela, bez bezpośredniego z nim kontaktu. Umożliwia kierowanie treści dydaktycznych do rozproszonych w sensie geograficznym grup słuchaczy.

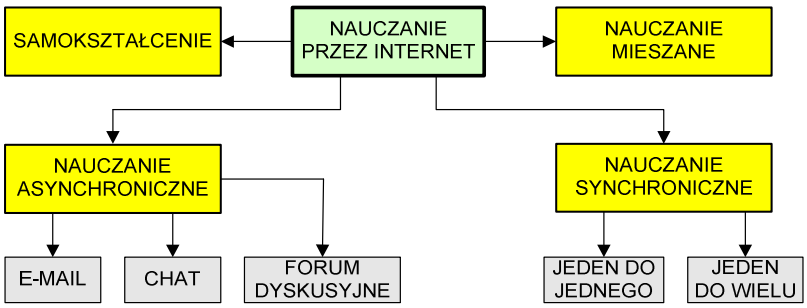
Formą stanowiącą przełom w nauczaniu na odległość jest *e-learning*. Nie należy utożsamiać pojęcia *e-learningu* z kształceniem przez Internet. *E-learning* jest terminem szerszym i obejmuje formy kształcenia na odległość, w których proces edukacji realizowany jest przy użyciu nowoczesnych technologii elektronicznych. Natomiast kształcenie przez Internet to kluczowa forma *e-learningu*, która umożliwia przekazywanie treści dydaktycznych oraz komunikowanie się ze studentami za pomocą Internetu bądź mniej otwartych sieci, takich jak intranet lub ekstranet [109].

Wyróżnia się cztery sposoby nauczania przez Internet: samokształcenie, nauczanie asynchroniczne, nauczanie synchroniczne i nauczanie mieszane (rys. 1.1).

Samokształcenie charakteryzujące się brakiem kontaktu studenta z prowadzącym zajęcia. *Nauczanie asynchroniczne* zachodzi wówczas, gdy studenci i prowadzący nie muszą jednocześnie być w tym samym miejscu i czasie w trakcie procesu edukacyjnego. Jest to popularny sposób nauczania przez Internet. Charakteryzuje się częściowym brakiem bezpośredniego kontaktu z nauczycielem, który zazwyczaj ograniczony jest do rozmów w wirtualnych pokojach rozmów (ang. *chat room*). Formy kontaktu pośredniego to: forum dyskusyjne, poczta elektroniczna oraz *chat*.

Zaletami takiego rozwiązania są: możliwość nauczania z wybranego miejsca, dostęp do materiałów w dogodnym czasie i w dowolnym miejscu, czas na analizę

materiału uwarunkowany od percepcji studenta oraz niskie koszty prowadzenia zajęć. Koszt przygotowania materiałów zależy od: stopnia zaawansowania, skali rozbudowy treści oraz platformy informatycznej.



Rys. 1.1. Sposoby nauczania przez Internet [109]

Nauczanie synchroniczne występuje wówczas, gdy studenci i prowadzący muszą być w tym samym czasie, a w przypadku nauczania tradycyjnego - także w tym samym miejscu. Model internetowego nauczania synchronicznego jest bliższy systemowi tradycyjnemu niż model nauczania asynchronicznego Nauczanie synchroniczne ma wiele zalet: interakcje w czasie rzeczywistym, praca grupowa, prezentowania materiałów w czasie dyskusji oraz możliwość bezpośredniego nadzoru nad studentami.

Wyróżnić można dwie formy zdalnego nauczania synchronicznego. Pierwszą formą jest *jeden do wielu* (słuchacze usytuowani są w różnych miejscach), a drugą – *jeden do jednego* (zajęcia prowadzone przez nauczyciela znajdującego się w innym miejscu niż grupa osób zgromadzonych w pewnym miejscu).

Formy te różnią się zastosowanymi rozwiązaniami oraz kosztami ich wdrożenia. Preferowanym modelem jest model *jeden do jednego*, który wymaga odpowiedniego oprogramowania i sprzętu jedynie w dwu centrach. Jest to rekomendowana forma prowadzenia zajęć w systemie szkoleń poligonowych, gdy szkoleni znajdują się na poligonie, a wykładowcy – w uczelni.

Nauczanie mieszane (ang. *blended learning*) cechuje się stosowaniem rozwiązań *e-learningu*, które wspierają proces kształcenia prowadzony w sposób tradycyjny. Jest to model bardzo popularny. Jest także uznawany przez wielu ekspertów akademickich za najlepszy. Stosowany jest w Szkole Głównej Handlowej w Warszawie, gdzie wykłady stacjonarne uzupełniane są zajęciami wirtualnymi przez internetową platformę nauczania [203].

Obok szkoleń na odległość, stosowane są systemy zarządzania procesem nauczania LMS (ang. *Learning Management System*), LCMS (ang. *Learning Content Management System*), a także systemy VCS (ang. *Virtual Classroom System*).

System informatyczny LMS wspomaga proces zarządzania, administracji, nadzoru i raportowania działań związanych ze szkoleniami. Do *funkcji zarządzających* systemu zalicza się: projektowanie harmonogramów zajęć, budowę katalogu dostępnych zasobów, import i udostępnianie kursów, zarządzanie zasobami dydaktycznymi oraz zarządzania opłatami za kursy.

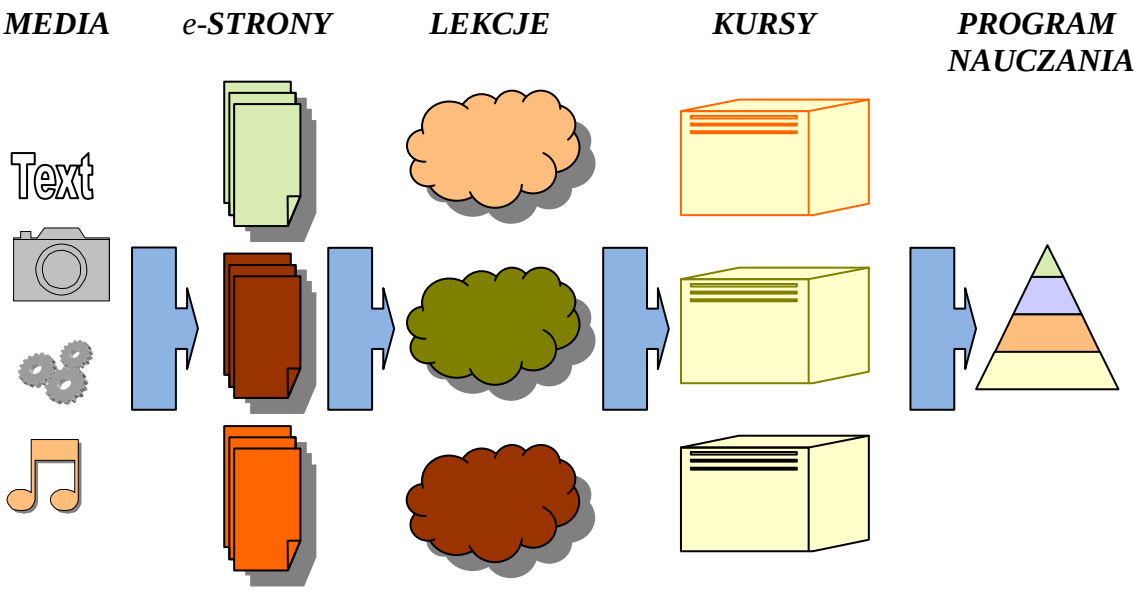
Funkcje monitorujące proces kształcenia to: zapisywanie danych dotyczących procesu edukacyjnego dla każdej osoby szkolonej, monitorowanie działania nauczycieli, śledzenie zachowania studentów oraz ewidencjonowanie kosztów.

Z kolei *funkcje raportujące* dostarczają zestawienia dotyczące rezultatów kształcenia, satysfakcji studentów, ich zachowań w procesie edukacyjnym oraz raporty o charakterze administracyjnym.

System LCMS jest używany do projektowania, składowania i udostępniania spersonalizowanych materiałów szkoleniowych w postaci tzw. obiektów szkoleniowych. System cechuje się także funkcjonalnościami systemu LMS. Pozwala na wspomaganie zarządzania pracy zespołowej nad przygotowaniem treści szkoleniowych [81].

VCS umożliwia zarządzanie i prowadzenie nauki na odległość w trybie synchronicznym. Rozwiązania tej klasy pozwalają na: wykorzystanie wirtualnej tablicy, współdzielenie ekranu, współdzielenie aplikacji, przekaz audio lub wideo, a także *chat*. Ponadto wspomagają zarządzanie procesem nauczania synchronicznego oraz opracowanie materiałów szkoleniowych do szkoleń "na żywo" [72, 152].

Stosowanie standardów jest warunkiem poprawnego współdziałania produktów programistycznych w systemach zdalnego nauczania. Gdy standardy są opublikowane, a następnie zaimplementowane, są „niewidoczne” dla internautów [115]. W przypadku nauczania na odległość standaryzacja jest konieczna. Studenci mają problem z wyszukaniem odpowiedniego kursu, autorzy kursów - w łączeniu treści pochodzących z różnych źródeł, a administratorzy - odnośnie przenoszenia materiałów złożonych z powiązanych ze sobą plików. Standardy pomagają w upowszechnieniu modułowej, wielokrotnego użytku struktury materiałów edukacyjnych (rys. 1.2) oraz zmniejszają zależność od indywidualnych rozwiązań.



Rys. 1.2. Modułowa struktura materiałów edukacyjnych [75]

Istotnym jest osiągnięcie sytuacji, gdy do powtórnego wykorzystania nadają się elementy na wszystkich poziomach zależności: kursy, lekcje, strony i media (tekst, grafika, dźwięk, wideo). Jednostka wiedzy z możliwością ponownego wykorzystania nazywana jest obiektem wiedzy wielokrotnego użytku (ang. *reusable learning object*) i może być przez autora kursu wykorzystywana w różnych projektach [75].

Analogicznie przy projektowaniu systemów informatycznych stosuje się łączenie komponentów, z których każdy oferuje odpowiednie usługi i funkcjonalności. Przykładowo Java EE jest standardem do projektowania aplikacji w języku Java, bazującym na wielowarstwowej architekturze składającej się z komponentów [93, 170].

Organizacje standaryzujące działają w czterech obszarach. Pierwszy obszar dotyczy *technologii nauczania na odległość*. Najstarszą w tej grupie organizacją jest AICC (ang. *Aviation Industry CBT Committete*) [181]. Pozostałe organizacje to: *IEEE Learning Technology Standards Committee* [211], *IMS Global Consortium* [112, 213] oraz ADL (ang. *Advanced Distributed Learning*) [179]. Strategią ADL jest opracowanie modelu referencyjnego SCORM (ang. *Sharable Content Object Reference Model*).

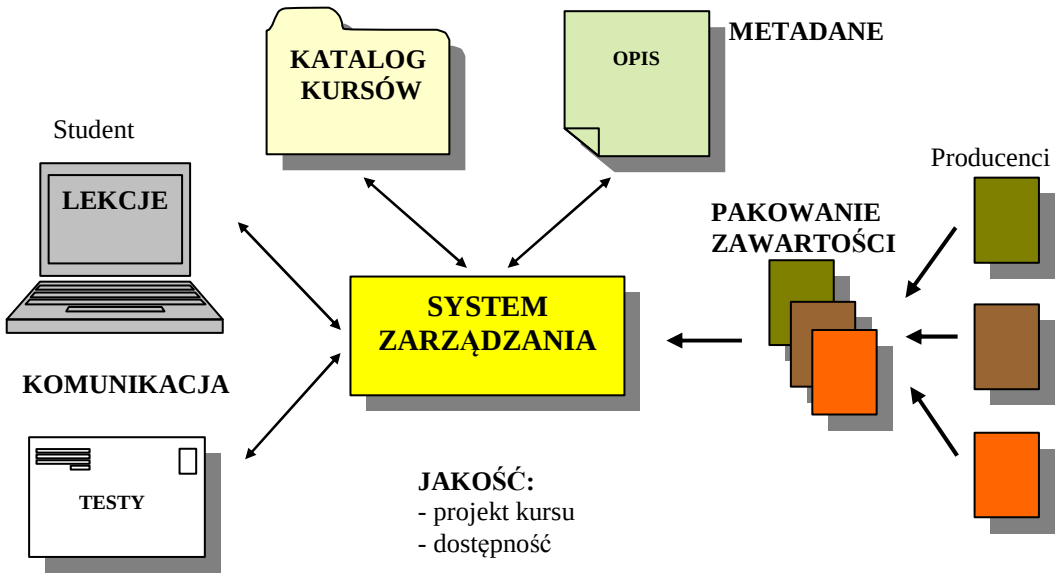
Drugi obszar obejmuje *jakość nauczania na odległość*. *ASTD Certification Institute* zajmuje się certyfikacją kursów zdalnego nauczania, bardziej z punktu widzenia zawartości merytorycznej niż technologii [187].

Trzeci obszar odnosi się do *technologii cząstkowych*, takich jak XML. Kluczową pozycję ma organizacja W3C (ang. *World Wide Web Consortium*) [245].

Czwarty obszar związany jest z *akredytacją standardów*. Pierwszym etapem dla większości standardów jest akceptacja IEEE [210]. Następnie standard może stać się standardem ISO [216].

Wyróżnia się cztery główne typy standardów (rys. 1.3). Standardy *pakowania zawartości* (ang. *packaging standards*) pozwalają na złożenie wielu kursów w jeden. Pozwalają na import i organizowanie przez LMS komponentów kursu. Zapewniają plikowi dotarcie we właściwe miejsce platformy dydaktycznej [75]. Stosuje się standardy AICC, IMS oraz SCORM. Do narzędzi ułatwiających stosowanie standardów pakowania należy *Manifest Maker for ADL SCORM* [199].

Druga grupa to *standardy komunikacyjne* (ang. *communications standards*). Określają one, w jaki sposób moduły wymieniają dane. W tej grupie wykorzystuje się standardy AICC i SCORM. Do sprawdzenia zgodności systemu zarządzania ze standardem komunikacyjnym można wykorzystać narzędzia: *AICC/CMI Test Suite* lub *SCORM Conformance Test Suite* [180, 182].



Rys. 1.3. Typy standardów w *e-learningu* [75]

Trzecia grupa standardów określa, w jaki sposób producenci powinni przygotować opisy kursów i innych modułów, tak by system zarządzania mógł wygenerować listę dostępnych materiałów nauczania. Są to *standardy metadanych*. Do najważniejszych standardów należą: *IEEE 1484.12 Learning Object Metadata Standard* [171], *IMS Learning Resources Meta-data Specification* [214] oraz *SCORM Meta-data*

standard [240]. Metadane są przechowywane i przetwarzane w formie dokumentu języka XML. Zaawansowanym narzędziem do implementacji metadanych jest *Simple SCORM Packager* [219].

Czwartą grupę standardów stanowią *standardy jakości*. Dotyczą one projektowania modułów i kursów, jak również sposobu dostępności przez osoby niepełnosprawne. Zapewniają, że *obiekty wiedzy* nie tylko nadają się do ponownego użycia, ale także są użyteczne do opracowania kursów i programów nauczania. Główne standardy odnośnie projektowania opisane są w dokumentach organizacji ASTD [188]. W przypadku standardów dostępności stosowane są *Section508 of the U.S. Rehabilitation Act* oraz *Web Accessibility Initiative* [241].

Zgodność ze standardem nie gwarantuje osiągnięcia celu, zapewnia za to wiarygodną informację, na której można bazować, podejmując różnorakie decyzje [75].

Studia przez Internet nie są jeszcze w Polsce popularne. Brakuje możliwości uzupełniania (poza stacjonarną uczelnią) wiedzy na poziomie akademickim. Tymczasem w Kanadzie, która liczy 28 mln mieszkańców, jest ponad 1600 *college-ów* oferujących około dwóch tysięcy kursów przez Internet [144].

Zazwyczaj *wirtualne uczelnie* funkcjonują jako oddziały lub filie przy uczelniach. Polski Uniwersytet Wirtualny (PUW, ang. *Polish Open University*) oraz Ośrodek Kształcenia na Odległość „OKNO" Politechniki Warszawskiej bazują na zasobach uczelni, przy których powstały. Takie rozwiązanie ma na celu wykorzystanie zaplecza dydaktycznego. Wdrożenie wirtualnego oddziału jest szansą rozwoju i poszerzenia oferty dydaktycznej przez uczelnię [230, 237].

Struktura szkoły wyższej prowadzącej wyłącznie zdalne nauczanie różni się od tradycyjnej uczelni tym, że istnieje konieczność budowy wydajnego systemu informatycznego. Istotne znaczenie ma zatrudnienie odpowiedniej kadry i inwestycje w nowoczesny sprzęt komputerowy. Atutem uczelni wirtualnej jest mniejsze obciążenie infrastruktury - student odwiedza uczelnię tylko w czasie egzaminów, zaliczeń czy dodatkowych zjazdów.

Tematyka *e-learningu* jest przedmiotem wydawnictwa Szkoły Głównej Handlowej „E-mentor," dostępnego zarówno *online*, jak również w wersji PDF [200]. Znajdują się tam informacje na temat zdalnej edukacji w Polsce i na świecie, metod i programów kształcenia, e-biznesu, konferencji dotyczących *e-learningu* oraz programów i projektów dotyczących kształcenia na odległość.

Ośrodek Kształcenia Na Odległość „OKNO” Politechniki Warszawskiej powstał w 2001 roku. System SPRINT (od *Studia PRez INTernet*) oferuje odczyt materiałów dydaktycznych nagranych na dyskach CD, wysyłanie i odbiór poczty elektronicznej e-mail, rozwiązywanie zadań, wykonywanie raportów, projektów, konsultacje oraz dyskusje między wykładowcami a studentami.

Materiał dydaktyczny przedmiotu podzielony jest na *jednostki lekcyjne*. Wydawany jest na płycie CD w formie podręcznika multimedialnego. Podręcznik ten dostępny jest także w Internecie. Studenci otrzymują podręczniki do przedmiotów, mogą je odczytać na komputerze, nie muszą pobierać materiału dydaktycznego z serwera Uczelni. Dostępna jest również *e-biblioteka,* gdzie po zalogowaniu się, można korzystać z podręczników w wersji PDF. System nauczania oparty jest o komercyjną platformę *Lotus Learning Space* firmy IBM [230].

Polski Uniwersytet Wirtualny jest wspólnym przedsięwzięciem Wyższej Szkoły Humanistyczno-Ekonomicznej w Łodzi i Uniwersytetu Marii Curie-Skłodowskiej w Lublinie. Studia *online* zostały uruchomione w 2002 r. na kierunkach: *Zarządzanie i marketing* oraz *Informatyka*. Rok później rozpoczęły się zajęcia na *Politologii* oraz *Pielęgniarstwie*. W 2005 r. pierwsi absolwenci odebrali dyplomy. W PUW korzysta się z kilku platform zdalnego nauczania, takich jak: *WebCT*, *Lotus Learning Space*, *R5 Generation* oraz *NETg* [237].

Szkoła Główna Handlowa rozpoczęła działalność wspomagania zajęć materiałami *online* w 2001 r. Poprzez autorską platformę prowadzone są zajęcia dydaktyczne. System służy również do prowadzenia szkoleń i kursów. Stosowane są cztery metody kontaktu studentów i wykładowców: czat, forum, SMS oraz e-mail. System został wyposażony w moduł umożliwiający wysyłanie SMS-ów do jednego, kilku lub wszystkich studentów danej grupy. Dla nauczyciela dostępny jest *wirtualny dziennik,* dzięki któremu może on oceniać zaangażowanie uczestników oraz określać postępy w nauce. Zaimplementowano opcje przeglądania statystyk aktywności studenta, wyników testów, ocen oraz komunikacji ze studentem [202].

W 2008 r. Minister Nauki i Szkolnictwa Wyższego podpisał rozporządzenie w sprawie warunków, jakie muszą być spełnione, aby zajęcia dydaktyczne na studiach mogły być prowadzone z wykorzystaniem metod i technik kształcenia na odległość. Ustalono, że liczba godzin zajęć dydaktycznych prowadzonych zdalnie, nie może być większa niż 60% ogólnej liczby godzin zajęć dydaktycznych określonych

w standardach kształcenia dla poszczególnych kierunków studiów oraz poziomów kształcenia, z wyłączeniem zajęć praktycznych i laboratoryjnych [201].

Zdalne kształcenie to nie tylko domena uczelni wyższych. Również szkoły średnie oferują nauczanie przez Internet. Można tu wymienić Liceum Zaoczne dla Dorosłych w Warszawie lub Zespół Szkół Prywatnych w Bydgoszczy [222, 243].

ATVN (akronim od Akademicka Telewizja Naukowa) to projekt realizowany w Interdyscyplinarnym Centrum Modelowania Matematycznego i Komputerowego Uniwersytetu Warszawskiego od 2002 r. do 2008 r.. ATVN jest kanałem telewizyjnym o profilu naukowym, emitowanym przez Internet. Bloki programowe trwające około 30 minut, są powtarzane przez całą dobę, obejmując wszystkie strefy czasowe. Programy przygotowywano w polskiej wersji językowej, a niektóre z nich w lektorskiej wersji angielskiej. Polskie stacje telewizyjne i firmy produkcyjne: TVP S.A., Media Corporation, Adi Art, PFUN czy TMT, udostępniły wybrane produkcje tej platformie, jak również wyemitowały programy produkcji ATVN [190].

Centrum Wiedzy to pierwszy polski niekomercyjny portal wiedzy o ekonomii i zarządzaniu, który powstał w SGH w 2001 roku [228]. Natomiast *Świętokrzyskie Centrum Edukacji na Odległość* to źródło informacji o zdalnej edukacji [239].

Kształcenie na odległość jest stosowane w firmach. Wymienić należy projekt *E-Pracownik* dla pracowników i kadry zarządzającej małych i średnich przedsiębiorstw. Partnerzy regionalni odpowiedzialni za realizację szkoleń to: Polsko-Japońska Wyższa Szkoła Technik Komputerowych w Warszawie, Uniwersytet Mikołaja Kopernika w Toruniu, Uniwersytet Gdański oraz Wyższa Szkoła Informatyki i Zarządzania w Rzeszowie. Nadzór merytoryczny nad realizacją szkoleń pełni *Cisco Systems*. Oferta obejmuje prawie 30 kursów informatycznych zróżnicowanych pod względem tematyki i poziomu zaawansowania. Są to zarówno szkolenia z obsługi komputera, aplikacji, programów biurowych, a także szkolenia biznesowe i kursy Cisco przeznaczone dla administratorów systemów [193].

W firmie *Kaspersky Lab*, która projektuje oprogramowanie antywirusowe, uruchomiono program szkoleniowy dla użytkowników oprogramowania. Szkolący się mogą pogłębiać wiedzę na bezpłatnych kursach *online*. Celem jest umożliwienie zrozumienia zasad działania produktów, pozwalającego na ich efektywne użytkowanie. Interesujące kursy dostępne są w portalu poświęconym pomocy technicznej. Materiał edukacyjny jest przedstawiony w formie prezentacji i ćwiczeń, które użytkownicy

wykonują w symulatorach imitujących działanie produktów *Kaspersky Lab*. Te same ćwiczenia mogą być także przeglądane w postaci nieinteraktywnej demonstracji [177].

Firma *WiedzaNET* we współpracy ze *SkillSoft* oferuje bogaty zbiór gotowych szkoleń. Kursy są budowane z użyciem standardów AICC i SCORM, co pozwala na ich wykorzystanie nie tylko na platformie LMS *SkillPort*, ale także w innych systemach zarządzania nauczaniem, np. *Oracle iLearning*. Interesującym produktem jest *Books24x7* zawierający zbiór publikacji dostępnych *online*, których wykorzystanie wspiera organizacje uczące się. *Books24x7* pozwala wyszukać rozwiązanie problemu za pomocą wydajnego, wielopoziomowego mechanizmu przeszukiwania zbioru, który zawiera ponad 19 500 książek biznesowych i raportów ponad 330 wydawców.

Spośród innych produktów e-learningowych warto wymienić *LDC* zawierający nagrania ekspertów biznesowych „na życzenie", usługę *CLT* do nauki języka angielskiego *online* na różnych poziomach zaawansowania oraz *Caspian learning*, czyli produkt z pogranicza rozwiązań szkoleniowych i gier. Technologia 3D pozwala lepiej przedstawić środowisko, którego dotyczy szkolenie, a technika szkolenia poprzez zadania wykonywane w wirtualnym świecie przy pomocy awatara umożliwia przyswojenie wiedzy [246].

Do nauczania na odległość wykorzystywane są również komunikatory internetowe, takie jak Gadu-Gadu lub Skype. Można dzięki nim nauczyć się grać na instrumencie lub nauczyć się języka angielskiego [183].

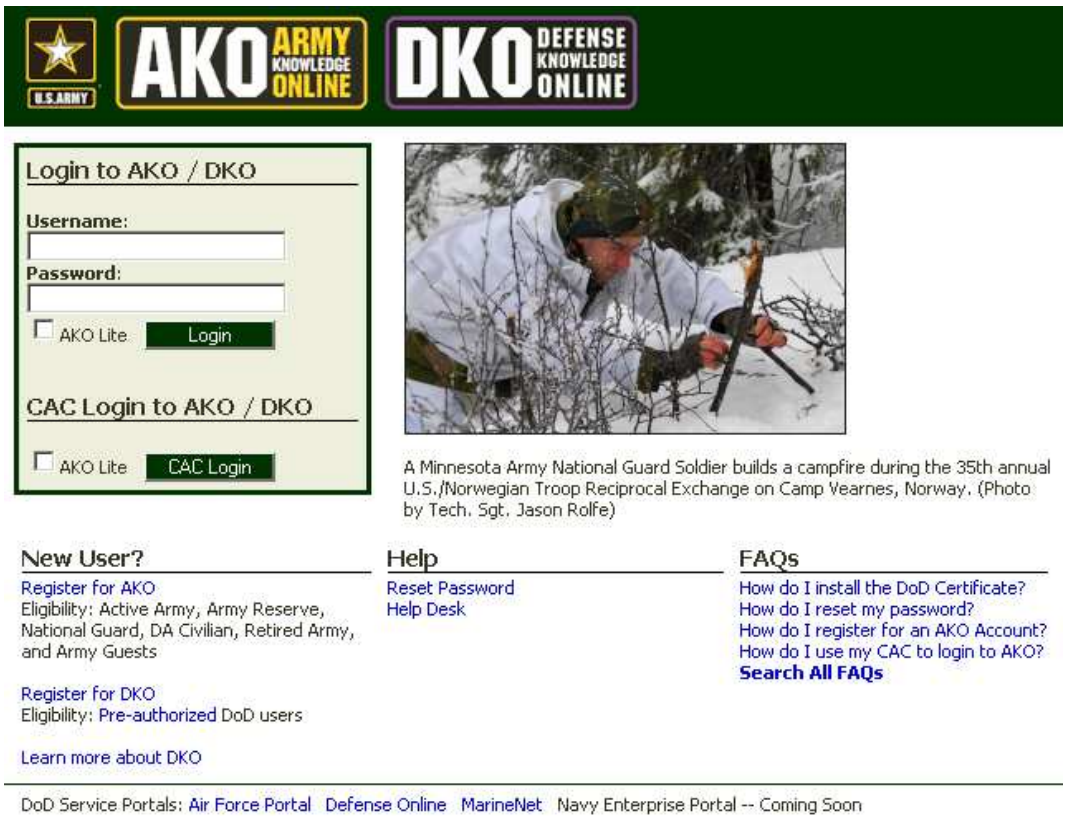
Zdalne szkolenie jest także bardzo intensywnie prowadzone w nowoczesnych armiach. Portal armii amerykańskiej *Army Knowledge Online* (akronim *AKO*) jest głównym elementem strategii zarządzania wiedzą w armii (ang. *AKM - Army Knowledge Management*) [185]. AKO jest dostępne dla żołnierzy w czynnej służbie i w stanie spoczynku, rezerwistów, wojskowej straży narodowej, departamentu pracowników wojska oraz dla użytkowników wspomaganych przez armię (rys. 1.4).

AKO ma około 3 milionów użytkowników, w tym około 1,5 miliona personelu wojskowego. AKO jest portalem dla ponad 40 serwisów WWW *US Army* oraz „bramą" do 30 centrów wiedzy Departamentu Obrony. W bazie wiedzy portalu AKO indeksowanych jest ok. 800 tys. dokumentów i 2,5 mln adresów WWW. AKO składa się z dwóch portali: NIPRNet (ogólny dostęp dla personelu US Army, informacje jawne) oraz SIPRNet (dostęp tylko dla uprawnionych, informacje do klauzuli *tajne*). Obok nich działa publiczna strona *Army Home Page* [184].

Bazując na akcjach podejmowanych przez użytkownika, jego profilu oraz stopniu wojskowym, oprogramowanie portalu AKO przesyła odnośniki do odpowiednich zasobów. Uprawnieni internauci mogą wymieniać się informacjami w zaszyfrowanym środowisku za pomocą 128-bitowego klucza. Użytkownik może projektować strony WWW, przy czym istnieje możliwość personalizacji każdej z nich tak, aby wyświetlane były tylko wybrane informacje i kursy. Zapewniona jest obsługa interakcji w czasie rzeczywistym [185].

W ramach programu *Army e-Learning* możliwy jest dostęp poprzez portal AKO do ponad 5 400 kursów dydaktycznych z dowodzenia, informatyki, języków obcych lub biznesu. Dostępne są również materiały *Books24x7*. Od lipca 2010 używana jest platforma *SkillPort* w wersji 7.0 [198]. W styczniu 2010 roku w życie weszło zarządzenie szefa sztabu armii amerykańskiej nakazujące wdrożyć projekt *Army e-Learning Program* jako główny element zdalnego nauczania w wojsku.

W ramach specjalnego programu zdalnego nauczania dla małżonków żołnierzy dostępnych jest około 17 000 książek i 2 800 kursów *online*, m.in. ze ścieżek certyfikacyjnych Cisco i Microsoft. Koszt rocznego dostępu do zasobów wiedzy wynosi 500 USD i jest pokrywany przez rząd USA.



Rys. 1.4. Portal AKO armii amerykańskiej [185]

Firma Rosetta Stone szkoli żołnierzy armii amerykańskiej z zakresu nauki kilkudziesięciu języków obcych, w tym języka polskiego. Zajęcia prowadzone są z wykorzystaniem rozwiązań multimedialnych, a po zakończeniu kursu uczestnik otrzymuje certyfikat poprzez e-mail.

Od 2007 roku francuska armia zaangażowała się w wspieranie ruchu *open source*. We współpracy z firmą Mozilla powstała wtyczka bezpieczeństwa *TrustedBird*, która jest wykorzystywana w kliencie poczty *Thunderbird* na około 100 tysiącach komputerów francuskiego resortu obrony. System Windows sukcesywnie jest zastępowany systemem Linux, a dominującą pozycję wśród pakietów biurowych zdobywa *Open Office* [218].

Zdalne nauczanie przez Internet realizowane jest przy pomocy oprogramowania *SAP Business Objects Knowledge Accelerator*, zintegrowanego z portalem francuskiej armii. Należy dodać, że rozwiązania firmy SAP wykorzystywane są w zdalnym szkoleniu sił powietrznych USA, szwedzkich siłach zbrojnych oraz hiszpańskiej straży cywilnej [238].

W armii brytyjskiej w 2005 roku uruchomiono portal DLP (ang. *Defence Learning Portal*), który w październiku 2010 posiadał ponad 200 tysięcy użytkowników, zarówno wojskowych jak i cywilnych. Dostępnych jest około 1200 kursów, które posiadają klauzule jawne i zastrzeżone. Poprzez portal zarządzanych jest 16 forów *online*, na których omawiane są kwestie e-learningu i szkoleń. DLP zarządza 14 zbiorami *wiki*, a dostęp do nich jest zawężony do wybranych użytkowników [197].

Wyzwania mobilnej armii oraz paradygmaty społeczeństwa informacyjnego zmotywowały Bundeswerę do silnego wsparcia zdalnego nauczania, celem jakościowej i ekonomicznej optymalizacji jej struktur szkolenia. W 2009 roku zostało utworzone na Uniwersytecie Bundeswery, Centrum Edukacji i Szkolenia Opartego Na Technologii, które zajmuje się tematyką zdalnego nauczania w armii [227].

Od 1995 roku firma *Benntec Systemtechnik GmbH* jest partnerem Szkoły Lotniczej armii niemieckiej w Bückeburg w dziedzinie zdalnego nauczania. Wdrażany jest zintegrowany system zdalnego szkolenia dla pilotów śmigłowców ILT HGA. W planach jest zintegrowanie najnowszych technik zdalnego nauczania z trójwymiarowymi, wirtualnymi symulatorami akcji na potrzeby szkoleń [192].

Poprzez portal niemieckiej *Nato School* w Oberammergau dostępnych jest ponad 30 kursów z różnych dziedzin wojskowości. Kursy te realizowane są w ramach programu ADL (ang. *Advanced Distributed Learning*), zapoczątkowanego przez rząd

USA. Portal posiada ponad 6000 zarejestrowanych użytkowników, a każdego miesiąca liczba ta zwiększa się średnio o 260 osób. Transfer danych w ramach portalu wzrósł ze 105 GB w 2008 roku do 150 GB w 2009 roku [227].

W armii kanadyjskiej za tworzenie i dostarczanie materiałów zdalnego szkolenia oraz wsparcie szkolących odpowiedzialne jest *Centrum Wsparcia Nauczania w Armii*. Współpracująca z nimi komórka Ministerstwa Obrony Narodowej *DNDLearn* składa się z ponad 2000 instruktorów, deweloperów i administratorów, którzy są odpowiedzialni za portal zdalnego szkolenia. Wykorzystując możliwości technologii Web 2.0, możliwa jest współpraca społeczności żołnierskich w celu poszerzania wiedzy z różnych dziedzin wojskowości [186].

W polskiej armii nie ma jeszcze odpowiednika portalu AKO. Wdrożony w 2005 roku przez studentów cybernetyki WAT i systematycznie ulepszany system RSW miał duże szanse zastosowania w szkoleniu wojskowym na odległość. Warto także wspomnieć o systemie zdalnego nauczania w Akademii Marynarki Wojennej, przy opracowaniu którego współuczestniczył Autor niniejszej dysertacji [173]. W obu uczelniach wojskowych podstawowym oprogramowaniem do zdalnego nauczania jest *MOODLE*, a kursy dydaktyczne zawierają informacje jawne. Natomiast w wypadku zdalnego nauczania kursów niejawnych wskazana jest adaptacja sieci MIL-WAN [215].

Na wyższych szczeblach dowodzenia MON wykorzystuje się zestawy typu „mały pokój konferencyjny” pozwalające na uczestnictwo w wideo-konferencji. Zestawy tego typu nadają się do wykorzystania w zdalnym nauczaniu [234].

1.2. Informatyczne systemy rozproszone

System rozproszony (ang. *distributed system*) to zbiór niezależnych komputerów połączonych siecią komunikacyjną i wyposażonych w oprogramowanie, zaprojektowane po to, by implementować zintegrowane i spójne środowisko obliczeniowe [48]. Systemy rozproszone pozwalają użytkownikom na skuteczną współpracę i koordynację działań. Kluczowe cechy systemów rozproszonych to: współdzielenie zasobów (ang. *resource sharing*), otwartość (ang. *openness*), współbieżność (ang. *concurrency*), skalowalność (ang. *scalability*), odporność na błędy i uszkodzenia (ang. *fault-tolerance*) oraz przezroczystość (ang. *transparency*) [48].

Systemy rozproszone ułatwiają użytkownikom dostęp do zdalnych zasobów, ukrywając fakt ich rozproszenia i umożliwiając *współdzielenie*. Pojęcie *zasób* jest

zazwyczaj rozumiane szeroko: począwszy od urządzeń (laserowe drukarki kolorowe, macierz dyskowa), poprzez oprogramowanie. a skończywszy na plikach i stronach WWW [147].

System rozproszony sprawia na użytkownikach wrażenie logicznie spójnego, a rozproszenie zasobów i procesów na maszynach jest maskowane. Mówimy o takim systemie, że jest *przezroczysty*. Przezroczystość może być postrzegana na różnych poziomach, a poziomy te mogą być mniej lub bardziej istotne dla użytkownika końcowego [178].

Przezroczystość dostępu (ang. *access transparency*) to ujednolicenie metod dostępu do danych i ukrywanie różnic w reprezentacji danych. Użytkownik korzysta ze standardowego interfejsu dostępu do danych. Różnice w reprezentacji danych mogą wynikać z zastosowania różnych architektur komputerowych [33].

Z kolei *przezroczystość położenia* (ang. *location transparency*) charakteryzuje się tym, że użytkownicy nie mogą określić fizycznego położenia zasobu na podstawie jego nazwy czy identyfikatora. Warunkiem koniecznym jest stosowanie nazewnictwa zasobów, które abstrahuje od ich położenia.

Przezroczystość wędrówki (ang. *migration transparency*) pozwala na przenoszenie zasobów między serwerami bez potrzeby zmiany sposobu odwoływania się do nich. Uzyskanie przezroczystości wędrówki wymaga zrealizowania przezroczystości położenia.

Przezroczystość przemieszczania (ang. *relocation transparency*) oznacza, że zasoby mogą być przenoszone nawet wtedy, gdy są używane przez użytkowników i nie wymaga to informowania użytkowników o zmianie położenia.

Przezroczystość zwielokrotniania (ang. *replication transparency*) charakteryzuje się tym, że mimo zwielokrotniania zasobów użytkownicy nie zauważają tego faktu, gdyż korzystają z zasobów dokładnie w taki sam sposób, jak w systemie niestosującym zwielokrotniania.

Przezroczystość współbieżności (ang. *concurrency transparency*) gwarantuje, że współbieżne odwoływanie się do tego samego zasobu realizowane przez wielu użytkowników nie będzie prowadziło do powstania stanu niespójnego w systemie.

Przezroczystość awarii (ang. *failure transparency*) cechuje się tym, że użytkownik nie zauważa faktu uszkodzenia pojedynczych węzłów. Trudność polega na odróżnieniu awarii węzła od awarii łączy komunikacyjnych.

Przezroczystość trwałości (ang. *persistence transparency*) ma za zadanie ukrywać mechanizmy zarządzania i przechowywania zdalnych zasobów. Osiągnięcie wysokiego poziomu przezroczystości wiąże się z wysokimi kosztami. W praktyce dąży się do osiągnięcia racjonalnego kompromisu między obserwowaną przez użytkownika złożonością systemu a efektywnością jego pracy [70].

Systemy rozproszone, aby mogły być rozbudowywane, muszą być *otwarte*. Implikuje to konieczność standaryzacji protokołów i interfejsów komunikacyjnych. Specyfikacja interfejsu, aby mogła być implementowana niezależnie przez wielu dostawców oprogramowania, musi być *kompletna* i *neutralna*. *Kompletność* oznacza, że opis jest wystarczający do wykonania implementacji, a *neutralność* cechuje się tym, że specyfikacja nie narzuca szczegółów dotyczących implementacji [178].

Rozwój sieci komputerowych i liczba komputerów przyłączanych do Internetu w systemach *e-learningu* rośnie tak intensywnie, że jednym z najważniejszych wymagań projektowych jest zapewnienie *skalowalności*. Skalowalność może być rozważana w trzech aspektach. Po pierwsze, skalowalność pod względem *rozmiaru* oznacza możliwość dodawania do systemu nowych użytkowników i zasobów. Po drugie, skalowalność *geograficzna* umożliwia geograficzne rozproszenie użytkowników. Po trzecie, skalowalność pod względem *administracyjnym* odnosi się do tego, że zarządzanie systemem pozostaje równie nieskomplikowane, mimo zwiększania jego rozmiaru i dystrybucji odpowiedzialności na jednostki administracyjne [116].

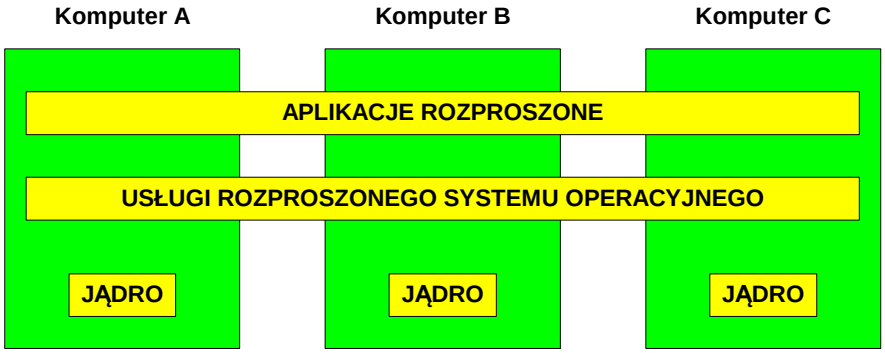
Istotne są trzy metody osiągnięcia wysokiej skalowalności: ukrywanie opóźnień komunikacyjnych, rozpraszanie i zwielokrotnianie. *Ukrywanie opóźnień* cechuje się stosowaniem komunikacji asynchronicznej, w której program zlecający operację nie czeka na jej wynik, lecz wykonuje dalsze przetwarzanie niewymagające wyniku ostatniej operacji. Niestety komunikację asynchroniczną można stosować efektywnie jedynie w systemach nieinteraktywnych. Z kolei *rozpraszanie* polega na podziale zadań komponentu programowego na wiele jednostek i rozlokowanie tych jednostek w sieci. Przykładem może być system DNS (ang. *Domain Name System*). *Zwielokrotnianie* (ang. *replication*) umożliwia poprawienie dostępności zasobów oraz równoważenie obciążenia. Ponadto zwielokrotniony serwer może być zastąpiony innym w przypadku awarii. Szczególną formę zwielokrotniania stanowi stosowanie pamięci podręcznych (ang. *cache*), które zawierają kopie oryginalnych danych [86].

Zadania systemu operacyjnego (akronim SO) w systemie rozproszonym są podobne do zadań scentralizowanego SO w zakresie zarządzania zasobami, takimi jak:

procesory, pamięci, urządzenia zewnętrzne i sieci komunikacyjne. Ponadto, rozproszone SO mogą globalnie zarządzać zasobami systemu rozproszonego - są to systemy ściśle powiązane (ang. *tightly-coupled*). Systemy operacyjne tej klasy projektowane są dla systemów wieloprocesorowych i multikomputerów homogenicznych. W wieloprocesorach wszystkie procesory mają dostęp do zasobów za pomocą wspólnej przestrzeni adresowej. W multikomputerach każda jednostka ma swoją lokalną pamięć. Systemy luźno powiązane (ang. *loosely-coupled*) to zbiór komputerów z lokalnymi systemami operacyjnymi, które ze sobą współpracują.

System operacyjny dla multikomputera składa się z lokalnych systemów operacyjnych, z których każdy ma własne jądro zawierające moduły zarządzania lokalnymi zasobami (rys. 1.5). W węźle uruchamiany jest moduł komunikacyjny umożliwiający wysyłanie i odbiór komunikatów do innych węzłów [137].

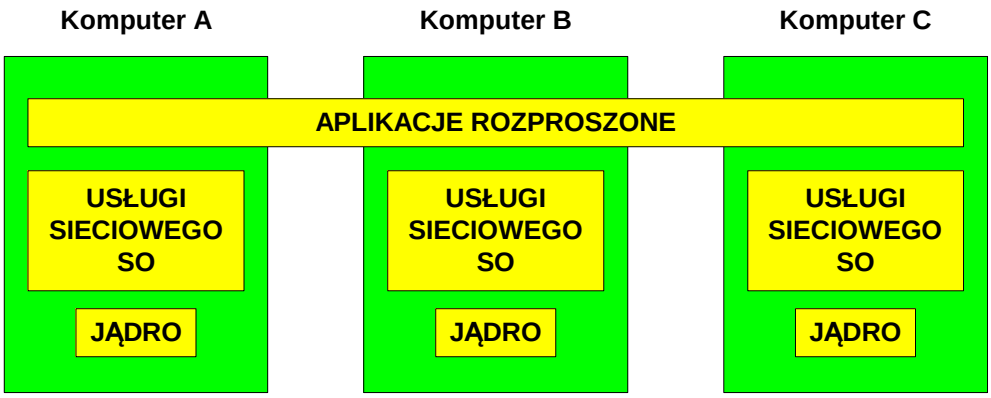
Powyżej jądra znajduje się warstwa oprogramowania, która dostarcza usługi dla *maszyny wirtualnej* umożliwiającej równoległe i współbieżne wykonywanie zadań. Warstwa ta może również udostępniać pamięć dzieloną. Ważne zadania tej warstwy to: szeregowanie zadań, maskowanie awarii oraz zapewnianie przezroczystości pamięci.



Rys. 1.5. Warstwowa struktura wielokomputerowego systemu operacyjnego [178]

W systemach wielokomputerowych, w których nie zastosowano pamięci dzielonej do komunikacji, stosuje się wymianę komunikatów. Istnieje jednak możliwość emulowania pamięci dzielonej, która to emulacja umożliwia działanie aplikacji analogicznie jak dla wieloprocesorów [147].

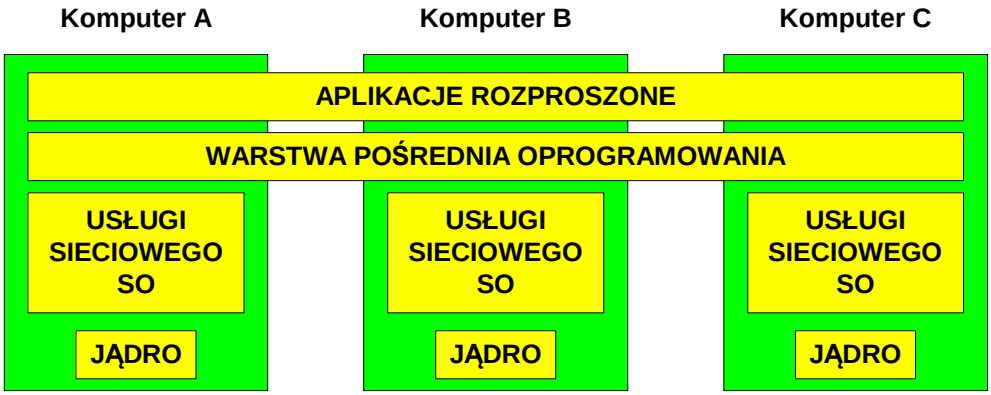
Systemy operacyjne dla heterogenicznych systemów wielokomputerowych określane są jako sieciowe systemy operacyjne (ang. *network operating system*) (rys. 1.6). Usługi sieciowych SO to: zdalne logowanie, kopiowanie plików między systemami oraz sieciowe systemy plików (ang. *network file system*).



Rys. 1.6. Warstwowa struktura sieciowego systemu operacyjnego [178]

Rozbudowa sieciowego SO w celu osiągnięcia przezroczystości zasobów prowadzi do implementacji oprogramowania warstwy pośredniej (ang. *middleware*).

Zarządzanie zasobami lokalnymi jest powierzone lokalnemu systemowi operacyjnemu, a warstwa pośrednia nadbudowuje na tej bazie nowe usługi i ukrywa fakt heterogeniczności systemów składowych (rys. 1.7). W warstwie pośredniej definiuje się interfejsy komunikacyjne niezależne od interfejsów lokalnych systemów operacyjnych. Aplikacje rozproszone posługują się tylko tymi interfejsami i nie korzystają bezpośrednio z usług lokalnego systemu operacyjnego [146].



Rys. 1.7. Oprogramowanie warstwy pośredniej w modelu warstwowym [178]

W tabeli 1 przedstawiono klasyfikację systemów operacyjnych dla systemów rozproszonych.

Do paradygmatów projektowania oprogramowania warstwy pośredniej zaliczyć należy traktowanie zasobów systemu jako pliki (podobnie jak w systemie Unix) oraz zdalne wywołania procedur, co wiąże się z ukryciem faktu komunikacji sieciowej.

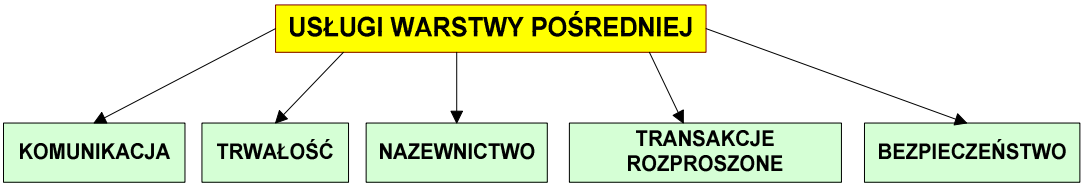
Ponadto stosuje się obiekty rozproszone (obiekt na jednej maszynie, interfejs do niej na wielu) oraz model dokumentów rozproszonych, np. strona WWW.

Tabela 1. Klasyfikacja systemów operacyjnych dla systemów rozproszonych [178]

SYSTEM	OPIS	GŁÓWNY CEL
Rozproszony system operacyjny	Ściśle powiązany system operacyjny dla wieloprocesorów i multikomputerów homogenicznych	Ukrywanie zasobów sprzętowych i zarządzanie nimi
Sieciowy system operacyjny	Luźno powiązany system operacyjny dla multikomputerów heterogenicznych (sieci LAN i WAN)	Oferowanie lokalnych usług klientom zdalnym
Warstwa pośrednia	Dodatkowa warstwa na szczycie sieciowego systemu operacyjnego, realizująca usługi ogólnego przeznaczenia	Zapewnianie przezroczystości rozproszenia

Jedną z podstawowych usług warstwy pośredniej jest *komunikacja* ukrywająca fakt rozproszenia poszczególnych maszyn (rys. 1.8). Może być realizowana z wykorzystaniem zdalnych wywołań procedur, obiektów rozproszonych lub poprzez transparentny dostęp do danych (plików czy baz danych).

Usługą warstwy pośredniej jest *nazewnictwo* umożliwiające publikowanie i wyszukiwanie informacji. Problem w implementacji wiąże się z koniecznością zapewnienia skalowalności. Czas dostępu do informacji, w tym czas odwzorowania identyfikatora na poszukiwany obiekt, powinien być niezależny od wielkości systemu.



Rys. 1.8. Usługi warstwy pośredniej
Źródło: opracowanie własne.

Z punktu widzenia aplikacji biznesowych istotną usługą jest możliwość zapewniania *trwałości* generowanym przez aplikację danych. Usługa ta jest oferowana w przypadku rozproszonych systemów plików czy rozproszonych baz danych, ale może też występować jako element innych usług, np. rozproszonej pamięci dzielonej.

Przetwarzanie danych w systemach rozproszonych może wymagać stosowania *rozproszonych transakcji*. Jedną z własności transakcji jest *atomowość* gwarantująca wykonanie operacji składowych transakcji w sposób niepodzielny. Niepodzielna

realizacja zbioru operacji na różnych serwerach, które dodatkowo mogą ulegać awarii, jest zadaniem bardzo złożonym, a często wręcz niemożliwym do zrealizowania. Transakcje rozproszone generują problemy związane ze skalowalnością.

Problem *bezpieczeństwa* jest pomijany w prototypowych implementacjach, ale jest istotny z praktycznego punktu widzenia. W systemie rozproszonym nie można wykorzystywać jedynie mechanizmów bezpieczeństwa udostępnianych i zarządzanych przez lokalne systemy operacyjne. Współpraca tych systemów wymaga, aby mechanizmy zapewniające bezpieczeństwo były realizowane także w warstwie pośredniej [178].

W praktyce nie ma systemów, które jednocześnie cechują się: dużą przezroczystością, otwartością, wysoką wydajnością i skalowalność (tabela 2). Systemy oparte na warstwie pośredniej oferują dużą przezroczystość, ale z ograniczoną skalowalnością.

Do najbardziej znanych rozproszonych systemów operacyjnych należą Amoeba, Chorus, Cronus oraz Inferno [138]. Przykładami sieciowego systemu operacyjnego są: Novell NetWare, LANtastic i Microsoft Windows Server 2008 R2.

Tabela 2. Porównanie systemów operacyjnych [178]

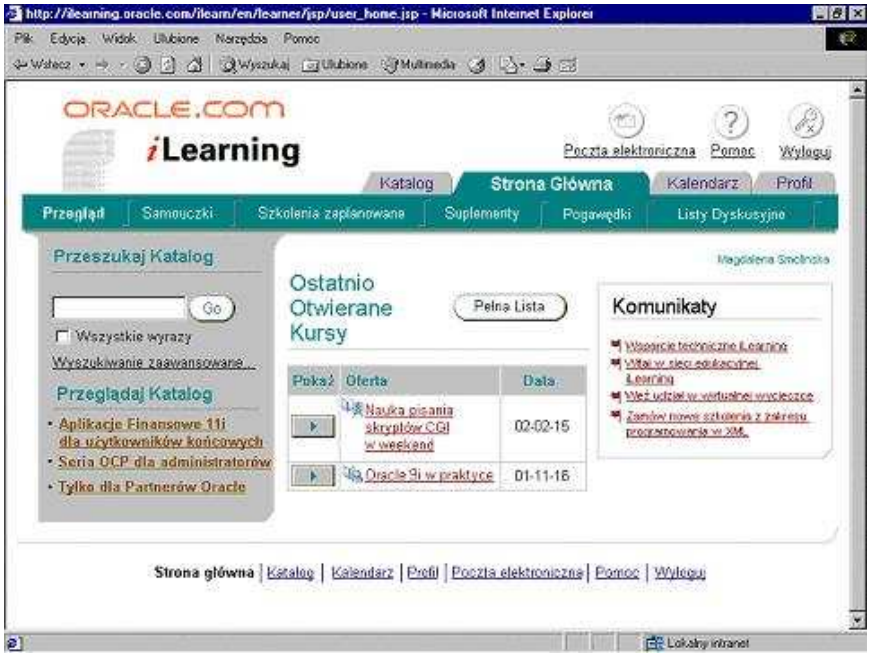
KRYTERIUM PORÓWNANIA	**ROZPROSZONY SO**		**SIECIOWY SO**	***MIDDLEWARE***
	wielo-procesorowy	wielo-komputerowy		
Przezroczystość	bardzo duża	duża	mała	duża
Jeden SO	tak	tak	nie	nie
Liczba kopii SO	1	N	N	N
Komunikacja	pamięć dzielona	komunikaty	pliki	zależna od modelu
Zarządzanie zasobami	globalne, centralne	globalne, rozproszone	lokalne	lokalne
Skalowalność	nie	umiarkowana	tak	zmienna
Otwartość	zamknięty	zamknięty	otwarty	otwarty

Wśród oprogramowania warstwy pośredniej wyróżnia się oprogramowanie związane ze zdalnym wywoływaniem procedur, systemami bazodanowymi, monitorami transakcyjnymi oraz portalami internetowymi [224]. Do najbardziej znanych architektur należą CORBA (ang. *Common Object Request Broker Architecture*), DCOM (ang. *Distributed Component Object Model*), RPC (ang. *Remote Procedure Calls*) oraz Java RMI (ang. *Remote Method Invocations*).

1.3. Platformy zdalnego nauczania i szkolenia

Wybór platformy *MOODLE* nie jest wyborem łatwym w kontekście licznego zbioru dostępnych platform zdalnego nauczania. Warto zatem dokonać analizy implementowanych platform pod kątem ich potencjalnego zastosowania w szkolnictwie wojskowym.

Platforma *Oracle iLearning* to komercyjny system informatyczny firmy Oracle implementujący środowisko do zarządzania procesami e-szkolenia w instytucjach oraz do zarządzania kursami (rys. 1.9). Oprócz podstawowych funkcji takich jak: zarządzanie użytkownikami, zarządzanie kursami, importowanie i odtwarzanie kursów, monitorowanie i weryfikacja postępów słuchaczy, czy raportowanie wyników, *Oracle iLearning* posiada również funkcje dodatkowe [169].



Rys. 1.9. Strona główna platformy *Oracle iLearning* [169]

Do realizacji komunikacji między użytkownikami służy poczta elektroniczna, forum dyskusyjne oraz chat. Ponadto możliwe jest planowanie ścieżek indywidualnego nauczania i prowadzenie szkoleń synchronicznych. Istotną funkcjonalnością jest personalizacja interfejsu użytkownika. Możliwa jest sprzedaż szkoleń i pobieranie opłat *online*. Przydatna jest integracja z systemami planowania zasobów ERP (ang. *Enterprise Resource Planning*) i serwisami B2B (ang. *Business to Business*). Platforma

implementuje standardy AICC i SCORM. Wykorzystywana jest przez Polsko-Amerykańskie Centrum Zarządzania w Łodzi oraz przez portal Wirtualnej Polski [231].

WBTServer to platforma firmy *4system*, która oparta jest na technologii J2EE (ang. *Java Platform, Enterprise Edition*). Oprogramowanie umożliwia rejestrację nieograniczonej liczby użytkowników przy jednej licencji. Nie wymaga się dodatkowych instalacji na komputerze szkolonym, gdyż komunikacja odbywa się przy wykorzystaniu języków HTML i JavaScript. Platforma współpracuje z aplikacjami opartymi na rozwiązaniach IBM, Oracle i Microsoft.

Platforma *WBTServer* współpracuje z systemami zarządzania bazami danych: MySQL, MS SQL Server oraz Oracle. Może być zainstalowana w systemach operacyjnych: Windows, Linux, Solaris, Mac OSX lub AIX. *WBTServer* zawiera rozbudowany moduł raportów z możliwością eksportu do plików w wybranych formatach oraz moduł egzaminacyjny. Jako element dodatkowy występuje moduł oparty na sztucznej inteligencji „wirtualny doradca".

Istnieje możliwość dopasowania platformy do korporacyjnych standardów i wewnętrznych procedur bezpieczeństwa firmy. Platforma instalowana jest na wielu serwerach (moduł Multiserver) [176]. Wówczas zachodzi konieczność optymalizacji przydziałów modułów programistycznych platformy. Dla firm zatrudniających ponad 5000 pracowników oferowana jest wersja Enterprise. Platforma wykorzystywana jest przez Politechnikę Koszalińską oraz BOT Elektrownię Bełchatów SA [66, 242].

E-SGH to platforma opracowana w Centrum Rozwoju Edukacji Niestacjonarnej SGH w oparciu o język XML, język PHP i system zarządzania bazą danych MySQL. Wykłady są projektowane i przechowywane w formacie XML. Możliwa są także inne formy publikacji, np. w formie informatora na płycie CD-ROM.

System posiada funkcje wspomagające proces nauczania. Należą do nich: audio, chat, forum oraz system wiadomości SMS. Oprócz tego dochodzą tzw. funkcje ukryte, dostępne jedynie użytkownikom o odpowiednim statusie - są to: *wirtualny dziennik* i *panel sterowania* przeznaczony dla administratorów [202].

Do zaawansowanych rozwiązań zaliczyć można platformę *E-edusystems* wdrożoną w 2002 roku w Polskim Uniwersytecie Wirtualnym [235] oraz Wydawnictwie Pret [236]. Platforma została uhonorowana godłem "Teraz Polska" przyznawanym najlepszym polskim produktom. *E-edusystems* uzyskał również ochronę patentową w Polsce, Europie i USA.

Netstudier jest zamkniętą, zabezpieczoną hasłowo, platformą do zdalnego nauczania. Menu jest dostępne w siedmiu językach. Z systemu korzysta ponad 60 000 użytkowników w około 70 instytucjach edukacyjnych. Średnia liczba logowań wynosi około 4 tys. dziennie. W Danii *Netstudier* uznany został za najlepszy system zdalnego nauczania. System można rozszerzać, więc liczba podłączonych instytucji może być znaczna. Zarządzanie platformą bazuje na standardzie TLM (ang. *Total Loss Management*) opracowanym na *University of Calgary*. Oprogramowanie zawiera bibliotekę zasobów cyfrowych, w której można umieszczać lub modyfikować zawartość. Utworzone materiały można udostępniać grupom lub tylko wybranym szkolonym. W opracowanie zawartości zaangażowanych jest kilkuset nauczycieli. W zamian za pracę nad opracowaniem zawartości systemu, oferuje się im kursy i szkolenia. Tworzone materiały stają się własnością *Netstudier-a*, a jednocześnie są bezpłatnie dostępne dla nauczycieli. Jest to kolejny, po Akademii Sieci CISCO, pilotażowy projekt wprowadzony przez Regionalne Studium Edukacji Informatycznej Uniwersytetu Mikołaja Kopernika w Toruniu [229].

RSW to system opracowany w 2005 roku i rozwijany przez studentów Wydziału Cybernetyki Wojskowej Akademii Technicznej. Alternatywnie w WAT stosowany jest także system *MOODLE*. *RSW* składa się z trzech modułów. *Moduł sesji* jest odpowiedzialny za bezpośrednie kontakty uczestników procesu nauczania. Moduł zapewnia przesyłanie obrazu konwersujących osób oraz ich głosu. Ponadto moduł realizuje dodatkowe funkcje, takie jak komunikator tekstowy czy współdzielony katalog.

Ponadto możliwe jest synchroniczne prezentowanie slajdów studentom. Wspólny pulpit umożliwia zespołowe wykonywanie niektórych operacji. Nadzór nad słuchaczami realizowany jest za pomocą zarządzania środowiskiem pracy przez wykładowcę. Moduł umożliwia trzypoziomowe uwierzytelnienie użytkowników (nośnik identyfikacyjny, hasło do nośnika oraz konto).

Moduł *portal internetowy* pracuje w środowisku, na które składają się: serwer Apache, interpreter języka PHP 5 oraz baza danych MySQL. URL (ang. *Uniform Resource Locator*) portalu zabezpieczony jest 256-bitowym szyfrem *Rijndael*, co zapobiega sytuacji, że student wpisze tę samą ścieżkę URL, aby jeszcze raz rozpocząć test. Wykorzystywany jest protokół SSL (ang. *Security Socket Layer*), który zapewnia: poufność i integralność transmisji danych oraz uwierzytelnienie. Opiera się on na szyfrach asymetrycznych oraz certyfikatach standardu X.509.

Moduł c*entralna baza danych* zapewnia automatyzację i koordynację zadań wewnątrz instytucji oraz przechowuje aktualne dane wykorzystywane w systemie. Przechowywane dane można podzielić na cztery funkcjonalne grupy, co ułatwia rozproszenie bazy danych. Pierwsza grupa zawiera dane osobowe, kod zajmowanego stanowiska oraz tytuł lub stopień naukowy osób zarejestrowanych w systemie. Drugą grupę stanowią informacje, które są wykorzystywane przy opracowywaniu planów. Są to: lista przedmiotów wraz z formą zajęć (ćwiczenia, wykłady), a także informacja o terminie sesji (godzina rozpoczęcia i zakończenia bloku). Trzecia grupa informacji to oceny studentów w skali (0-100), co umożliwia standaryzację ocen do punktacji stosowanej na uczelni. Ostatnia grupa obejmuje dane wykorzystywane w finansach oraz księgowości [145].

LEO (ang. *Learning Environment Online*) to platforma zaprojektowana przez firmę *Young Digital Planet* umożliwiająca implementację portalu szkoleniowego. System pozwala na administrowanie wieloma wirtualnymi ośrodkami szkoleniowymi. Aplikacje instalowane są na serwerach webowych. W platformie stosowany jest serwer *Login Server* do uwierzytelniania użytkowników oraz *CMS Server* do implementacji rozproszonego dostępu, edycji i prezentacji danych. Natomiast *LMS Server* umożliwia zarządzanie strukturą szkoleń, nadawanie uprawnień dostępu do zasobów systemu, dostęp do dodatkowych serwisów wspomagających nauczanie oraz przechowywanie informacji statystycznych. Serwer *Web Application Server* implementuje dodatkowe usługi rozszerzające możliwości platformy, np.: wideo-konferencje lub symulatory. Platforma jest zgodna ze standardami AICC i SCORM [247].

Platformą komercyjną jest *R5 Generation*, która wykorzystywana jest przez Polski Uniwersytet Wirtualny [237]. Tej klasy oprogramowaniem jest również *Microsoft Class Server* stosowany w Politechnice Wrocławskiej. W 2007 roku *Microsoft* zaprzestał jego rozwoju na rzecz *SharePoint Learning Kit* [223]. W Ośrodku Kształcenia Na Odległość OKNO Politechniki Warszawskiej wykorzystuje się *Lotus Learning Space* [230]. Z kolei firma IBM wycofała się z rozwijania tej platformy na rzecz *Lotus Learning Management System* i *Lotus Virtual Classroom* [209].

WebCT to pierwsze wirtualne środowisko nauczania, które odniosło komercyjny sukces. Wykorzystywane jest przez Polsko-Amerykańskie Centrum Zarządzania w Łodzi [231]. Z kolei platforma *Blackboard* wykorzystywana jest przez AGH i Centrum Zdalnego Nauczania Uniwersytetu Jagiellońskiego [195, 196]. Należy dodać, że firma Blackboard roku przejęła firmę WebCT w 2005.

Oprogramowanie do kształcenia przez Internet może być oparte na licencji GNU GPL (ang. *GNU General Public Licence*) jako produkt *open source*. Ważną zaletą dla instytucji edukacyjnych jest to, iż mogą je swobodnie modyfikować i wykorzystywać w swojej *e-learningowej* działalności.

MOODLE to oprogramowanie typu LCMS udostępniane jako *open source*. System zaimplementowano na około 50000 witrynach internetowych w 210 krajach (październik 2010 r.). System jest dostępny w 75 wersjach językowych, w tym polskiej. Platforma rozwijana jest w oparciu o język programowania PHP, systemy zarządzania bazą danych MySQL i PostgreSQL. Platforma wykorzystywana jest przez Politechnikę Gdańską, Akademię Górniczo-Hutniczą, Uniwersytet Marii Curie-Skłodowskiej, a także Akademię Marynarki Wojennej i Wojskową Akademię Techniczną [225].

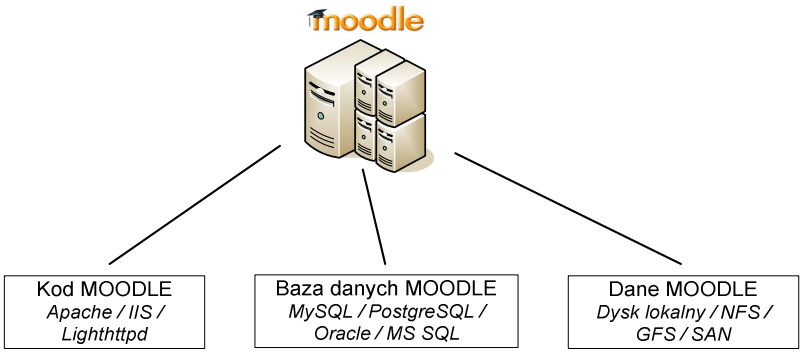
Rozwój platformy *ILIAS* koordynowany jest przez zespół z *University of Cologne* [212]. System wykorzystywany jest w 140 ośrodkach akademickich i instytucjach edukacyjnych w 16 krajach. Dostępny jest w 8 wersjach językowych, w tym polskiej. W Polsce system *ILIAS* został zaimplementowany przez Akademię Obrony Narodowej, Akademię Morską w Gdyni oraz Uniwersytet im. Adama Mickiewicza w Poznaniu, a w NATO platforma ta wykorzystywana jest w *NATO School* w Oberammergau [227].

Platforma *Claroline* posiada 28 wersji językowych, w tym polską i wykorzystywany jest przez setki organizacji ze świata. W Polsce platforma stosowana jest przez Uniwersytet Łódzki w programie *e-Campus* [194].

Do mniej znanych platform dostępnych w polskiej wersji językowej należy *Manhattan Virtual Classroom* oferowany przez *Western New England College* w 7 językach [172]. Natomiast *Fle3 Future Learning Environment* to fińska platforma dostępna w 15 wersjach językowych [168], a platforma *ATutor* dostępna jest w 41 wersjach językowych [189]. Stosowane jest także oprogramowanie *Pelp.net* [232]. Liczba platform, zarówno komercyjnych jak i bezpłatnych jest o wiele większa, a wybór odpowiedniej nie należy do najłatwiejszych zadań [204, 205, 206].

1.4. Charakterystyka systemu *MOODLE*

Termin *MOODLE* jest akronimem od *Modular Object-Oriented Dynamic Learning Environment*, co odzwierciedla kluczowe cechy tego środowiska nauczania: modularność, obiektowość oraz dynamiczność. System jest oparty na licencji GNU GPL. Możliwa jest implementacja aplikacji w systemach operacyjnych: Linux, Windows i Mac OS X. Środowisko, w którym instalowane będzie *MOODLE*, powinno wspierać język PHP i systemy zarządzania bazami danych (MySQL lub PostgreSQL), przy czym wymagana jest tylko jeden system zarządzania bazą. *MOODLE* może być podzielone na podsystemy przedstawione na rysunku 1.10 [114].



Rys. 1.10. Główne podsystemy systemu *MOODLE*
Źródło: opracowanie własne.

Pliki z kodem źródłowym umieszczone są na serwerze webowym. Umożliwia to administratorowi udostępnianie kursów za pomocą protokołów *Appletalk*, SMB (ang. *Server Message Block*), NFS (ang. *Network File System*), FTP (ang. *File Transfer Protocol*) lub WebDAV (ang. *Web-based Distributed Authoring and Versioning*).

Dla pojedynczego serwera szacowana maksymalna zalogowana liczba użytkowników biorących udział w kursach posiada następującą wartość [114]:

$$\mu=\mu_1 \mu_2 , \tag{1.1}$$

gdzie

μ_1 – wielkość pamięci RAM dostępna dla systemu *MOODLE* [*GB*],

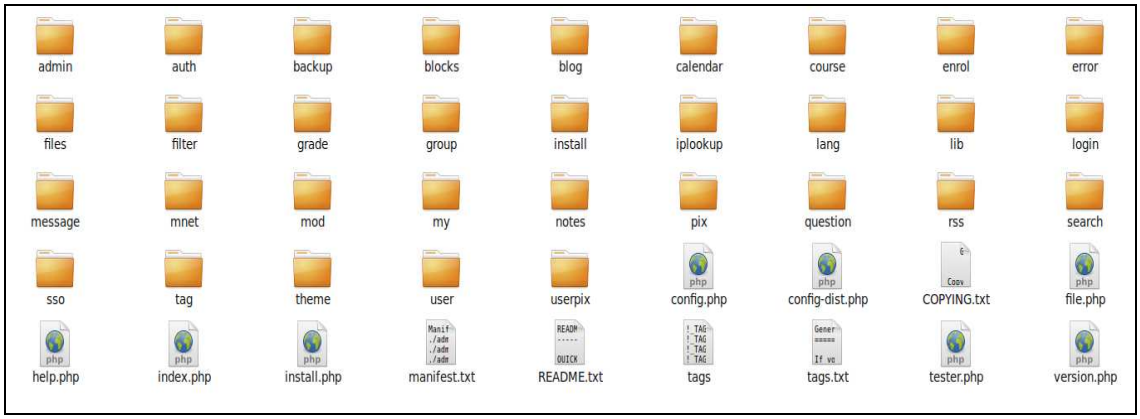
μ_2 – liczba zalogowanych użytkowników dla 1 GB RAM [GB^{-1}], dla *MOODLE* μ_2=50.

Przykładowo, serwer wyposażony w 2 GB RAM może obsłużyć jednocześnie 100 użytkowników *MOODLE*.

Struktura katalogów *MOODLE* odzwierciedla jego modularność. Folder w głównym katalogu reprezentuje ważny komponent systemu (rys. 1.11). Katalog

‘*admin*’ zawiera pliki wspomagające administrację interfejsem użytkownika i utrzymanie systemu. Katalog ‘*auth*’ zawiera moduły uwierzytelniania. Katalog ‘*backup*’ obejmuje funkcje do wykonania kopii zapasowych, importu i przywracania kursów. Katalog ‘*lib*’ mieści biblioteki rdzenia aplikacji. Projektując moduły, wykorzystuje się klasy i funkcje zdefiniowane w tym katalogu.

Większość z głównych komponentów *MOODLE* pozwala na dodawanie *modułów-wtyczek* (ang. *plugin*). Z punktu widzenia użytkownika moduły są instalowane poprzez skopiowanie ich do odpowiedniej lokalizacji na serwerze. Po kolejnym zalogowaniu administratora moduł jest identyfikowany przez system [114].



Rys. 1.11. Główny katalog systemu *MOODLE* w wersji 1.9.7
Źródło: opracowanie własne.

Moduły uwierzytelniania pozwalają korzystać z LDAP (ang. *Lightweight Directory Access Protocol*), IMAP (ang. *Internet Message Access Protocol*), POP3, NNTP (ang. *Network News Transport Protocol*) lub też z baz danych jako ze źródła informacji o użytkownikach. Wspierane są także certyfikaty: SSL oraz TLS (ang. *Transport Layer Security*).

Relacyjna baza danych *MOODLE* zawiera około 200 powiązanych tabel. Domyślna nazwa tabeli zaczyna się od przedrostka *mdl_*. Danymi mogą być pliki związane z kursami oraz dane sesji zalogowanego użytkownika. Każdy kurs umieszczony jest w osobnym folderze ocechowanym liczbą całkowitą. W bazie danych *MOODLE* może się znajdować pakiet językowy.

W ramach procedur bezpieczeństwa prowadzący kursy mogą generować "klucze dostępu” do kursu. Klucz może być przekazywany za pomocą poczty elektronicznej. Istnieje także możliwość importowania i eksportowania danych za pomocą formatów opartych na XML (ang. *eXtensible Markup Language*).

MOODLE zaprojektowano po to, by wspierać prowadzenie procesu dydaktycznego, a także jak uważa projektant oprogramowania *Martin Dougiamas* - społeczną edukację polegającą na współpracy uczących się osób, co jest podstawą społecznego konstrukcjonizmu. Społeczny konstrukcjonizm (lub konstruktywizm) to teoria socjologiczna rozwinięta przez Petera L. Bergera i Thomasa Luckmanna w monografii *The Social Construction of Reality* [26].

Podstawowym założeniem konstruktywizmu jest to, że świat postrzegany jest w sposób subiektywny [175]. Zakłada się także, że student jest aktywnym podmiotem, który samodzielnie rozwija własny system wiedzy, korzystając z dostępnych źródeł naukowych. Rola nauczyciela, obok motywowania i wspierania podopiecznych, polega na przydzielaniu zadań i formułowaniu pytań stanowiących dla uczniów problemy do rozwiązania. Przyjmuje się, że wiedza przyswajana jest najefektywniej, kiedy mamy do czynienia z uczeniem się przez współpracę. Jest to możliwe, kiedy słuchacz pracuje w grupie, dzieląc się własnymi doświadczeniami i opiniami, będąc otwarty na doświadczenia i opinie innych [207].

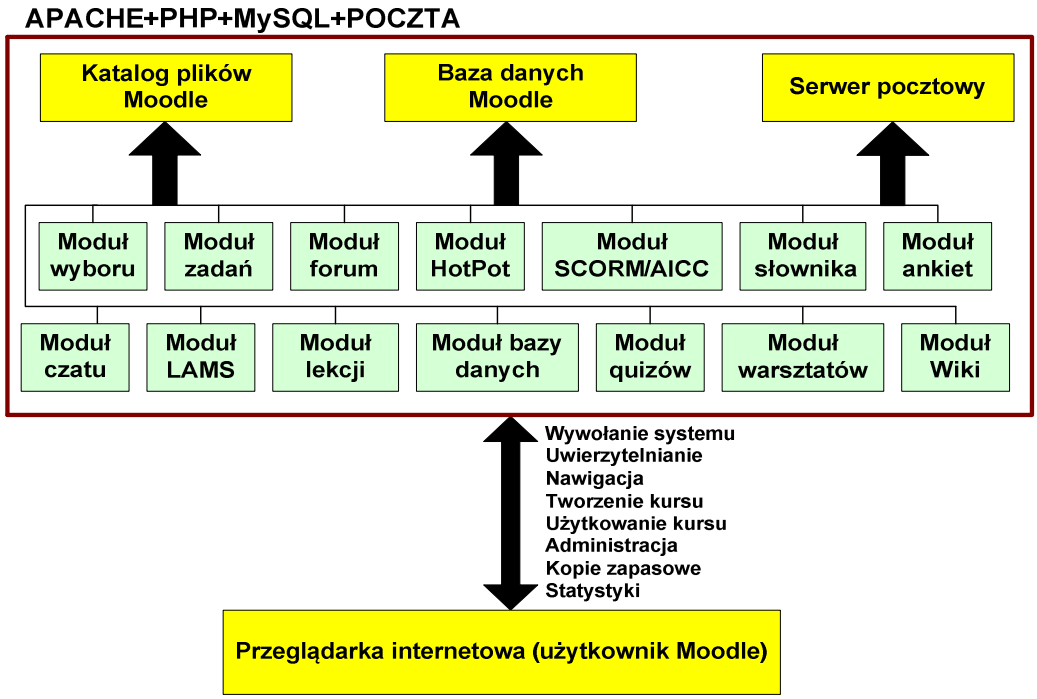
Zadania nauczyciela polegają na wprowadzaniu zasobów i ich edycji. Zadania dotyczą konsultacji, odpowiadania na pytania na forum dyskusyjnym oraz prowadzenie dyskusji w czasie rzeczywistym poprzez czat. Dodatkowo nauczyciel określa termin wykonania zadań. Ponadto w trakcie kursu realizowane są czynności kontrolne [128].

Istnieje kilka ról, które może pełnić użytkownik. *Student* to najniższa rola w hierarchii w sensie uprawnień do zasobów. Może przeglądać i zapisywać się na kursy, a także przeglądać swoje oceny i korzystać z różnych aktywności w ramach kursu. *Nauczyciel bez prawa edycji* pełni rolę administratora kursu. Może sprawdzać aktywność studentów i ich oceniać. Nauczyciel może także dodawać lub usuwać aktywności w kursie, zmieniać kryteria oceniania, zapisywać kopie zapasowe kursu oraz dołączać pliki do kursu. *Twórca kursów* może dodawać lub usuwać kursy. *Administrator* może generować raporty aktywności, dodawać moduły usług i moduły interfejsu, dodawać i usuwać użytkowników oraz zmieniać globalne ustawienia. Każda z ról dziedziczy uprawnienia z niższego poziomu w hierarchii.

W *MOODLE* występuje szeroki wachlarz modułów aktywności (ang. *activity modules*), zarówno standardowych, jak i niestandardowych, które mogą być wykorzystane do implementacji kursu oraz pozwalają na kontrolę realizacji założonego procesu dydaktycznego (rys. 1.12). Moduły te zlokalizowane są w katalogu '*mod*'.

Moduł *wyboru* używany jest do głosowań nad kwestiami merytorycznymi, po to, by zebrać opinie lub uzyskać zgodę na prowadzenie badań. Instruktor przygotowuje pytanie i pewną liczbę odpowiedzi do wyboru, zaś wyniki są prezentowane słuchaczom.

Zadanie polega na wyznaczeniu słuchaczowi pracy do wykonania, daty jej ukończenia i maksymalnej oceny. Następnie studenci przesyłają plik z rozwiązaniem. Data przesłania pliku zostaje zapisana, a na stronie kursu można przeglądać plik wraz z informacją o tym, z jakim opóźnieniem lub wyprzedzeniem został przesłany. Ponadto można zapisać ocenę i komentarz. Pół godziny po wystawieniu studentowi oceny, system automatycznie wysyła do niego zawiadomienie pocztą elektroniczną. Funkcjonalność ta wykorzystywana jest do bieżącej kontroli procesu dydaktycznego.



Rys. 1.12. Standardowe moduły aktywności w *MOODLE* w wersji 1.9.7
Źródło: opracowanie własne.

Dyskusje odbywają się za pomocą modułu *forum*. Fora dyskusyjne mogą posiadać różną strukturę i umożliwiać ocenę każdego postu przez współuczestników. Posty mogą być przeglądane w wybranych formatach z załącznikami. Uczestnicy, którzy są subskrybentami forum otrzymują pocztą elektroniczną kopię nowego postu. Prowadzący może narzucić subskrypcję uczestnikom.

Moduł *HotPot* pozwala nauczycielom zarządzać tzw. quizami *Hot Potatoes* [208]. W ich skład wchodzą: krzyżówki, pytania z wielokrotnym wyborem,

uzupełnianie brakujących wyrazów, pytania prawda/fałsz, układanie wyrazów we właściwej kolejności oraz dopasowywanie odpowiedzi.

Moduł *SCORM/AICC* pozwala przesłać dowolny pakiet zgodny ze standardami SCORM lub AICC, a także opracować część kursu.

Moduł *słownika* umożliwia uczestnikom kursu założenie i wykorzystanie zbioru definicji. Definicje mogą być wyszukiwane lub przeglądane na wiele sposobów. Słownik umożliwia nauczycielom eksportowanie zawartości z jednego słownika pojęć do drugiego w ramach tego samego kursu. Moduł umożliwia również wygenerowanie odnośników do zawartych definicji w ramach kursu.

Oferowane są dwa rodzaje *ankiet*. COLLES (ang. *Constructivist On-Line Learning Environment Survey*) to ankieta na temat konstruktywistycznego środowiska uczenia się *online.* ATTLS (ang. *Attitudes to Thinking and Learning Survey*) jest ankietą na temat podejścia do procesów myślenia i uczenia się. Ankiety umożliwiają badanie znaczenia dostępności modułów *online* oraz ich pomocy w uczeniu się. Ponadto, można prowadzić analizę poziomu myślenia refleksyjnego studenta. Ankiety zwiększają także stopień interaktywności. Wyniki ankiet pozwalają identyfikować trendy, które pojawiają się wśród uczestników w odniesieniu do zainteresowania bądź zniechęcenia do formy kształcenia na odległość. Mogą także sygnalizować zbyt mały nakład pracy nad materiałem.

Moduł *czatu* pozwala uczestnikom na przeprowadzanie dyskusji w czasie rzeczywistym. Jest to sposób na lepsze zrozumienie studiowanych kwestii. Sposób używania *czatu* zasadniczo różni się od dyskusji na *forum*, która jest asynchroniczna. Moduł zawiera opcje pozwalające na zarządzanie i przeglądanie dyskusji.

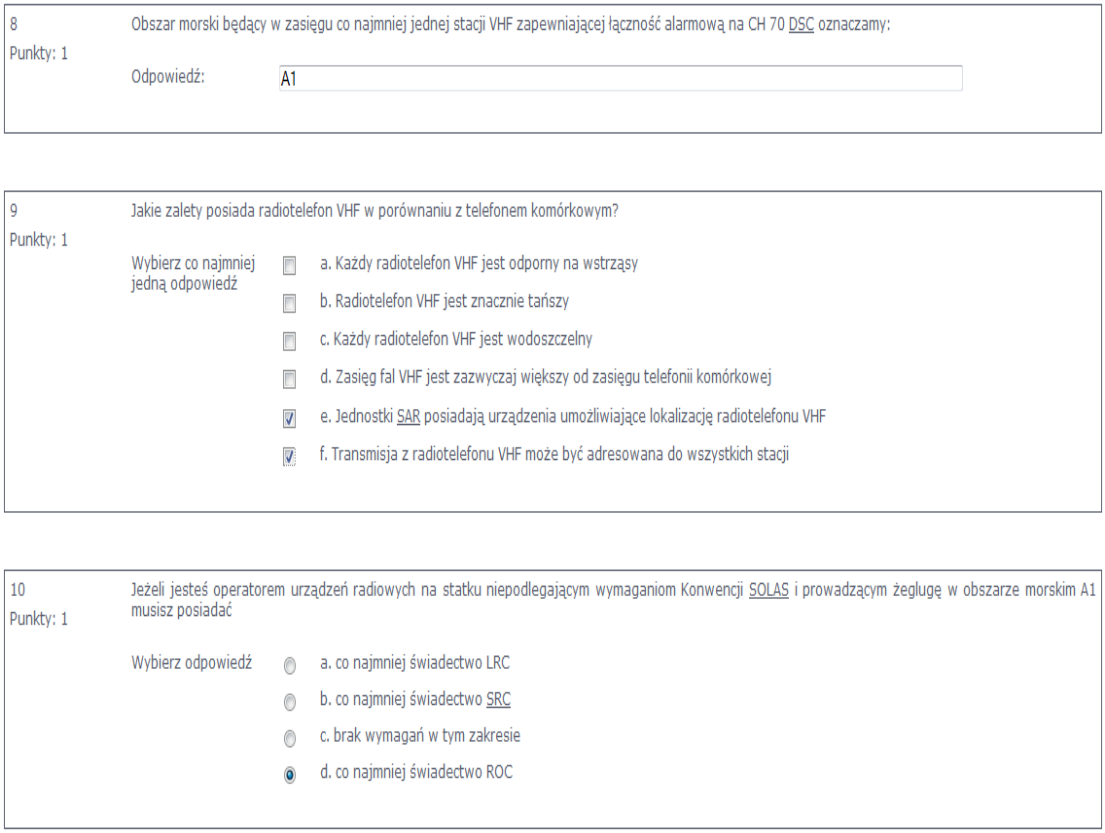
W wersji *MOODLE* 1.6.3 w 2006 roku zintegrowano moduł LAMS (ang. *Learning Activity Management System*), który powstał na Uniwersytecie *Macquarie* w Sydney [142]. LAMS umożliwia współdzielenie treści dydaktycznych oraz korzystanie z innych platform. Implementacja powyższych rozwiązań zwiększa funkcjonalność systemu, możliwości projektowania kursów, a także efektywność zarządzania *online* treściami dydaktycznymi z wykorzystaniem standardów *e-learningu*.

Lekcja pozwala na przedstawienie treści w interesujący i elastyczny sposób. Składa się z pewnej liczby stron. Strona zazwyczaj kończy się pytaniem i kilkoma odpowiedziami. Jeśli student opanuje materiał, przechodzi do następnej strony.

Moduł *bazy danych* pozwala na budowanie, wyświetlanie i przeszukiwanie odnośników do materiałów dotyczących wybranego tematu [164]. Format i struktura

odnośników jest bardzo urozmaicona. Mogą to być obrazy, pliki, linki do stron WWW, tekst lub liczby. Nie należy jednak utożsamiać tego modułu z bazą danych SQL, która przechowuje wszystkie informacje wykorzystywane w kursach i jest zarządzana wyłącznie przez administratorów *MOODLE*.

Quiz jest funkcjonalnością pozwalającą na kontrolowanie procesu zdobywania wiedzy. Opcja ta pozwala przygotować formy zapytań, takie jak: pytania wielokrotnego wyboru, pytania typu prawda/fałsz, pytania otwarte, pytania „zaznacz poprawną odpowiedź" lub pytania wymagających krótkich odpowiedzi. Pytania, uporządkowane wg kategorii, przechowywane są w bazie danych i mogą być ponownie wykorzystywane w ramach danego kursu lub przeniesione do innego (rys. 1.13). *Quizy* mogą dopuszczać wielokrotne próby wyznaczenia rozwiązania. Próba jest sprawdzana i prowadzący może decydować, czy przedstawić swój komentarz czy też udostępnić poprawne odpowiedzi. Moduł *Quiz* posiada narzędzia umożliwiające wystawianie ocen.



Rys. 1.13. Przykład quizu w systemie *MOODLE*
Źródło: opracowanie własne.

Warsztaty to forma aktywności umożliwiająca wzajemną ocenę przez uczestników projektów i dokumentów w formacie plików graficznych lub prezentacji.

Moduł zajmuje się także koordynacją i dystrybucją tychże ocen. Nauczyciel dokonuje końcowej oceny, a także kontroluje czas i etapy pracy.

Moduł *Wiki* pozwala na zespołowe opracowanie dokumentów webowych przy użyciu przeglądarki internetowej. Praca z Wiki zaczyna się od głównej strony będącej swego rodzaju spisem treści. Autor może dodawać kolejne strony do Wiki za pomocą linku do projektowanej strony.

Oprócz ww. standardowych modułów aktywności, istnieją moduły niestandardowe [164]. Dodatkowe moduły zawierające składowe kursów mogą być dodawane do zainstalowanej wersji *MOODLE* [120]. Wykaz wybranych niestandardowych modułów przedstawiono w tabeli 3.

Należy zauważyć, że pojęcia moduł aktywności i moduł programistyczny nie są tożsame. Katalogi, w których znajdują się pliki PHP modułów aktywności, mogą być modułami programistycznymi lub też stanowić ich część.

Tabela 3. Wykaz wybranych niestandardowych modułów systemu *MOODLE*

NAZWA MODUŁU	**ZADANIE MODUŁU**
Moduł dialogu	Prywatna rozmowa ze studentem *online*
Moduł certyfikatu	Opracowanie dla studentów w pełni edytowalnych certyfikatów w formacie PDF
Moduł poczty elektronicznej	Pozwala wysłać wiadomość do wielu uczestników jednocześnie
Moduł śledzący	Wykrywanie błędów w środowisku *MOODLE*
Moduł projektu	Ułatwia nauczanie podczas realizacji projektu
Moduł rejestracji	Pozwala na rejestrację studentów na egzaminy w instytucjach niemających centralnego systemu rejestracji
Moduł folderu zwrotnego	Pozwala studentom przesyłać pliki do folderu nauczyciela, który może wybrane pliki udostępniać innym studentom

Źródło: opracowanie własne.

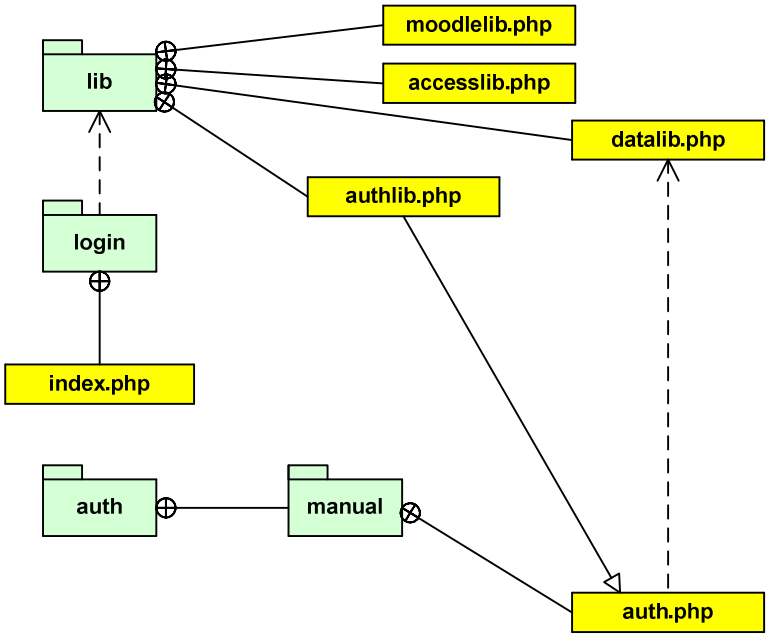
Struktura katalogów systemu *MOODLE* oraz tematykę wyboru modułów programistycznych omówiono w dodatku B.

Obok powyższych komponentów służących do pomiaru skuteczności uczenia się i kontroli postępów uczestników szkolenia, system posiada mechanizm raportowania i statystyki. Raport aktywności studenta, który znajduje się przy nazwisku na liście studentów lub na stronie profilu dowolnego użytkownika, jest dobrym sposobem sprawdzenia postępów dydaktycznych słuchacza.

Analiza aktywności użytkowników i rejestrowanie prac, odbywa się za pomocą narzędzi raportujących. Prezentują one zarejestrowane operacje, jakie wykonał użytkownik po uwierzytelnieniu się w systemie. Logi mają formę przejrzystego, mniej szczegółowego opisu lub szczegółowej listy przedstawiającej zadany okres czasu. Ponadto po zalogowaniu się, można zaobserwować wykonane akcje.

Wiele z komponentów oprogramowania nie posiada zabezpieczeń. Mogą być jednak implementowane odpowiednie usługi bezpieczeństwa, takie jak uwierzytelnienie, kontrola dostępu lub niezaprzeczalność (ang. *non-repudiation*).

Aby opracować model diagramu klas dla *MOODLE*, zakłada się, że pliki PHP są klasami, a skrypty posiadają metodę *main*() [73]. Na rysunku 1.14 przedstawiono diagram klas w zakresie usług bezpieczeństwa. Klasa *accesslib.php* posiada metody potrzebne do zapisywania i odzyskiwania zezwoleń dla każdej z ról pełnionych przez użytkownika. Zezwolenia te są przechowywane w bazie danych *MOODLE*. Klasa *datalib.php* definiuje metody dla usługi niezaprzeczalności. Klasa *moodlib.php* zawiera metody dla usług w *MOODLE*, w tym usług bezpieczeństwa.



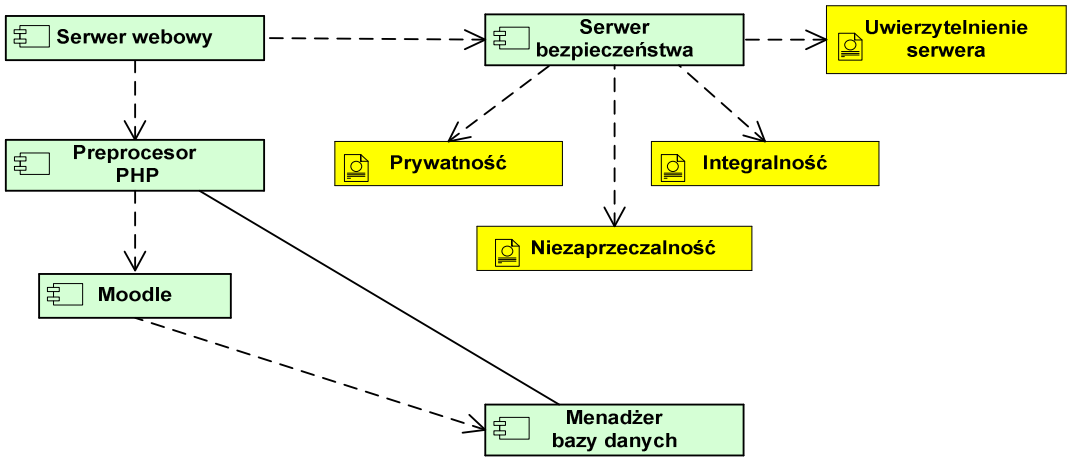
Rys. 1.14. Diagram klas dla usług bezpieczeństwa w systemie *MOODLE*
Źródło: opracowanie własne.

Usługa uwierzytelniania może być realizowana przez uwierzytelnianie użytkownika lub sesji. Uwierzytelnianie użytkownika implementowane jest za pomocą porównania nazwy użytkownika i hasła z danymi znajdującymi się w bazie danych.

Moduły wspomagające uwierzytelnianie pochodzą od klasy *auth_plugin_database*, która znajduje się wewnątrz klasy *authlib.php*. Dane konfiguracji strony internetowej *MOODLE* przechowywane są w klasie *config.php* w katalogu głównym.

Niektóre usługi bezpieczeństwa nie są bezpośrednio implementowane, ale bazują na konfiguracji serwera webowego. Aby określić te usługi, należy wcześniej zidentyfikować komponenty implementacji systemu *MOODLE*. Serwer webowy przedstawiony na rysunku 1.15 oferuje usługi stron internetowych.

Serwer webowy wykorzystuje usługi serwera bezpieczeństwa, który z kolei oferuje niezaprzeczalność, uwierzytelnianie serwera, integralność oraz poufność poprzez protokół SSL, gdy serwer webowy zgłasza takie żądanie. Do identyfikacji aktywnej sesji służą dwie wartości *MoodleSession* i *MoodleSessionTest* przechowywane w pliku *cookie*. Uzyskanie danych z żądania HTTP z *cookie* może być zrealizowane, ponieważ *MOODLE* używa SSL tylko w wybranych usługach.



Rys. 1.15. Diagram komponentów systemu *MOODLE*
Źródło: opracowanie własne.

Efektywne ataki na sesje w *MOODLE* to: przechwytywanie sesji (ang. *session hijacking*) oraz dołączenie użytkownika do sesji hakera (ang. *session fixation*). Pierwszy rodzaj ataku odnosi się do sytuacji, gdy haker próbuje uzyskać dostęp do istniejącej sesji użytkownika (identyfikator został już przydzielony). Podczas ataku drugiego rodzaju haker nie dąży do zdobycia identyfikatora sesji, a raczej do skłonienia ofiary do użycia identyfikatora wskazanego przez hakera, np. przez spreparowany odnośnik z dołączonym identyfikatorem. W wersji 1.9.8 i wyższych zapobieganie temu atakowi jest włączone domyślnie.

Oprogramowanie może być wrażliwe na atak polegający na przewidywaniu haseł. Realizuje się go poprzez wysyłanie do serwera żądań z pustym plikiem *cookie*. Wówczas licznik błędnych logowań jest zerowany, co pozwala na zastosowanie metody siłowej do wykrycia hasła.

W przypadku ataku polegającego na przewidywaniu nazw użytkownika stosuje się przechwytywanie *cookie* lub metodę siłową. Wykorzystuje się fakt, że odpowiedź serwera w przypadku poprawnej nazwy użytkownika jest dłuższa niż ta z niepoprawną nazwą [73].

Kolejnym problemem związanym z bezpieczeństwem jest atak XSS (ang. *cross-site scripting*) polegający na osadzeniu w treści atakowanej strony kodu, który wyświetlony innym użytkownikom może doprowadzić do wykonania przez nich niepożądanych akcji. Sposób walki z tym zagrożeniem przedstawiony został w [166].

Można uniknąć wad w zakresie bezpieczeństwa systemu *MOODLE* za pomocą modyfikacji kodu i dodania nowych funkcji. Jednym z rozwiązań jest spowodowanie, by serwer tworzył połączenia SSL ze swoimi klientami. Można to osiągnąć, dodając skrypty PHP zmieniające wartości odpowiednich zmiennych i flag.

Kolejnym sposobem jest generacja nowego identyfikatora sesji w przypadku, gdy użytkownik próbuje się uwierzytelnić, wykorzystując protokół HTTP [73]. Na oficjalnej stronie *MOODLE* przedstawiono przegląd zagadnień związanych z bezpieczeństwem. Dotyczy on wersji 1.8.9 i wcześniejszych [166] .

Platforma *MOODLE* znacząco przyczynia się do rozwoju *e-learningu* w polskiej edukacji, a przez to do rozwoju społeczeństwa opartego na wiedzy. Świadczy o tym rosnąca liczba wdrożeń systemu *MOODLE* w Polsce, która to liczba zwiększyła się z 400 serwisów na początku roku 2007 do ponad 1100 w październiku 2010 roku [226].

Przykładami funkcjonowania bezpłatnych rozwiązań *e-learning-u* opartymi o platformę *MOODLE* (w formie szkoleń i kursów) jest Centrum Edukacyjne Świętokrzyskiego Centrum Edukacji na Odległość oraz Portal E-learningu Wyższej Szkoły Handlowej w Kielcach [174, 239].

1.5. Wnioski i uwagi

E-learning jest efektywną metodą globalnej edukacji. W najbardziej rozwiniętych państwach świata wykorzystuje się ten sposób nauczania na potrzeby długofalowych strategii rozwoju gospodarki, w których główną rolę odgrywa edukacja

wspomagana za pomocą technologii informatycznych. Zdalne nauczanie jest jedną z najszybciej rozwijających się gałęzi rynku IT. W konsekwencji tego trendu, w nowoczesnych siłach zbrojnych wdrażane są systemy wspomagające zdalne szkolenie żołnierzy.

W wyniku powszechnej dostępności szerokopasmowego Internetu, rozwoju informatycznych systemów rozproszonych oraz doskonalenia urządzeń przenośnych, zdalne szkolenie wojskowe odgrywać będzie coraz bardziej znaczącą rolę w tworzeniu nowoczesnego systemu edukacji militarnej oraz w uzyskaniu przewagi w wyszkoleniu żołnierzy i personelu wojskowego.

Nauczanie przez Internet w Polsce w zakresie obronnym jest w fazie początkowego rozwoju. Koszt dostępu do Internetu maleje, zwiększane są przepustowości łączy, a także pojawiają się korzystne uwarunkowania prawne, co powoduje, że coraz więcej słuchaczy szkół i akademii wojskowych może wykorzystywać zdalną formę nauczania. Ponadto możliwe jest systematyczne kształcenie żołnierzy przebywających na długotrwałych zwolnieniach lekarskich, żołnierzy pełniących służbę w odległych od uczelni miejscach, a także żołnierzy-rodziców małych dzieci.

Coraz większa ilość danych przechowywanych, przetwarzanych i przesyłanych podczas zdalnego szkolenia wojskowego powoduje, że zachodzi konieczność rozproszenia modułów wchodzących w skład wykorzystywanego oprogramowania.

Optymalne przydzielenie modułów programistycznych pozwala na bardziej efektywną pracę. Platforma *MOODLE* posiada możliwość rozproszenia modułów wchodzących w jej skład. Jest to platforma klasy *open source*, która jest systematycznie rozwijana. Jej możliwości są porównywalne z rozwiązaniami komercyjnymi, a dostępność kodu pozwala na modyfikację oprogramowania pod względem zwiększenia poziomu bezpieczeństwa, a także optymalizację alokacji modułów. Moduły niestandardowe zwiększają funkcjonalność oprogramowania.

Wzorcowym wykorzystaniem zdalnego nauczania w szkolnictwie wojskowym jest portal armii amerykańskiej AKO. W Polsce jeszcze nie ma odpowiednika AKO, aczkolwiek prowadzone są prace nad zmianą tego stanu, m.in. przy wykorzystaniu platformy *MOODLE*.

Rozwiązania w zakresie zdalnego nauczania stosowane w Wojskowej Akademii Technicznej, Akademii Obrony Narodowej oraz Akademii Marynarki Wojennej powinny być rozwinięte, a następnie zastosowane w innych uczelniach wojskowych

i ośrodkach szkolenia. Istotna jest także współpraca naukowo-badawcza z wiodącymi ośrodkami e-learningu w Polsce: Politechniką Gdańską, Politechniką Poznańską lub Politechniką Warszawską. Należy oczekiwać, że korzyści z e-kształcenia żołnierzy zgrupowanych na ćwiczeniach lub w ramach szkoleń Narodowych Sił Rezerwowych będą także znaczące.

Zdalne szkolenie wspiera proces profesjonalizacji armii. Niewątpliwie ułatwi także proces rekonwersji żołnierzy zawodowych. W wypadku wprowadzania nowoczesnego uzbrojenia należy oczekiwać akceleracji procesu szkolenia przynajmniej w zakresie teoretycznym. Również istotnych korzyści należy oczekiwać przy trenowaniu interoperacyjności w ramach sił NATO. Docelowo systemy cechujące się wysoką jakością w zakresie poziomu bezpieczeństwa powinny być zastosowane do wspierania szkolenia żołnierzy na misjach zagranicznych.

System *MOODLE*, który byłby zoptymalizowany pod kątem dostępności i skutecznej realizacji zadań w terminach oraz odpowiednio skonfigurowany pod względem bezpieczeństwa, mógłby zostać wykorzystany do realizacji polskiego odpowiednika AKO i pozwoliłby na osiągnięcie wymiernych korzyści w szkoleniu wojskowym, co jest szczególnie istotne w odniesieniu do szkolenia żołnierzy i personelu służącego na misjach zagranicznych.

2. MODELE PRZYDZIAŁÓW MODUŁÓW PROGRAMISTYCZNYCH

Ocenia się, że koszt modyfikacji programu podczas użytkowania przekracza 50% kosztów związanych z oprogramowaniem [102]. Koszt ten można zredukować, dzieląc program na moduły, co umożliwia wprowadzanie zmian jedynie w podzbiorze modułów, a nie w całym programie.

Modułem (segmentem, jednostką) programistycznym (ang. *software module*) nazywa się sekwencję instrukcji (rozkazów) programu wyodrębnioną za pomocą ograniczników i mającą identyfikator umożliwiający odwołanie się do modułu [158]. Pojęcia *moduł programistyczny* i *moduł programowy* są zbliżone pod względem semantycznym, przy czym częściej używany jest termin *moduł programistyczny*.

Moduł programowy (ang. *program module*) to logicznie zamknięta część większego programu, np. procedura (ang. *subroutine*). Może to również być plik, który zawiera instrukcje w formie kodu źródłowego [122]. Słowo *moduł* oznacza także pojedynczy plik wykonywalny, który jest częścią aplikacji.

Modułem nazywa się także segment, który zawiera sekwencję instrukcji i dane lokalne [20]. Dotyczy to modułu w postaci wykonywalnej, a nie źródłowej. Nazywa się nim także segment, zawierający dane globalne dostępne dla innych modułów. Zazwyczaj ze sobą powiązanych tematycznie podprogramów tworzy się moduły będące bibliotekami podprogramów. Przykładem może być plik DLL (ang. *Dynamic Loaded Library*).

Zadanie (proces) to wykonanie modułu. Moduł jest obiektem pasywnym, podobnie jak zawartość pliku przechowywanego na dysku. Zadanie jest obiektem aktywnym ze stosem przechowującym dane tymczasowe (parametry procedur, adresy powrotne, zmienne tymczasowe), sekcją zmiennych globalnych oraz licznikiem rozkazów określającym następny rozkaz do wykonania [107, 135].

Moduł może być wielokrotnie wywoływany do wykonania. Przy pierwszym wywołaniu system operacyjny rejestruje zadanie odpowiadające temu modułowi. Proces ten może zostać zakończony, a po pewnym czasie rozważany moduł może być wywołany po raz drugi. Wówczas zachodzi sekwencyjne wykonywanie modułu za pomocą sekwencji procesów. W tym wypadku czas wykonania modułu jest sumą czasów realizacji wszystkich zadań będących wywołaniami tego modułu. Ponieważ czasy te można oszacować na podstawie kodu źródłowego modułu programu lub zmierzyć dla różnych rodzajów komputerów, przyjmuje się, że są one znane *a priori*.

Nieco inna sytuacja jest wówczas, kiedy moduł jest wywoływany do wykonania i podczas jego wykonywania następuje kolejne wywołanie rozważanego modułu. Odpowiada to sytuacji, gdy użytkownik dwukrotnie, w krótkim odstępie czasu, uruchomił ten sam program. Istnieje zatem moment czasu, podczas którego dwa procesy odpowiadające temu segmentowi programu są realizowane. Wówczas zachodzi równoległe (jednoczesne) wykonywanie modułu za pomocą równoległych procesów. W tym wypadku czas wykonania jednostki programu nie jest sumą czasów realizacji wszystkich zadań będących wywołaniami tego modułu. Natomiast łączny czas realizacji poszczególnych zadań generowanych z modułu nazywa się *obciążeniem w sensie czasu* modułu programistycznego.

Rozpatrując rozdział procesów w systemie operacyjnym, można uzyskać przydział, w którym zadania generowane z zadanego modułu będą wykonywane na różnych komputerach. Jeżeli każde zadanie jest wykonywane na tym samym komputerze, co stowarzyszony z nim moduł, to przydział segmentów programu można utożsamiać z przydziałem zadań [20].

2.1. Przegląd wybranych modeli przydziałów

Dzielenie programu na moduły ułatwia zarządzanie projektem programistycznym realizowanym przez zespół programistów, co skutkuje podniesieniem jakości programu [119]. W ten sposób można poprawić przejrzystość pisanych programów, a także obniżyć ryzyko popełnienia błędu. Jeżeli programista dokładnie przetestuje moduł, to może wielokrotnie wykorzystywać raz napisany kod [10]. W oprogramowaniu klasy *open source* programiści mogą dopisywać nowe moduły do już istniejącego systemu. Taka sytuacja występuje przy rozszerzaniu funkcjonalności Linuksa czy też *MOODLE* pod kątem zwiększenia bezpieczeństwa.

Możliwość przetwarzania rozproszonego istnieje, gdy komputery są wzajemnie połączone tak, aby instrukcja z jednego modułu mogła przekazywać sterowanie i dane do segmentu, który jest wykonywany na drugim komputerze. Program nazywamy *rozproszonym*, jeżeli jest podzielony, na co najmniej dwa moduły, które mogą być wykonywane na dwóch różnych komputerach oraz mogą się wzajemnie komunikować. Problemy wyznaczania przydziałów modułów programów należą do fundamentalnych aspektów obliczeń rozproszonych, realizowanych w rozproszonych systemach

komputerowych, gdyż możliwe pełniejsze jest wykorzystanie specyficznych właściwości komputerów w zależności od cech programów.

MOODLE napisany jest w języku PHP, a komunikacja między modułami odbywa się zgodnie z modelem *klient-serwer* [80]. Za pomocą metod protokołu HTTP *get* i *post* z modułu klienta można przekazać dane z formularza na serwer. Ponadto procedura *dircopy* pozwala na skopiowanie katalogu do innego katalogu łącznie z podkatalogami. Natomiast funkcja *move_uploaded_file* przenosi plik do wskazanej lokalizacji, jeśli był on wcześniej przesłany za pomocą metody *post*. Z kolei funkcją *imagecopy* można przekopiować fragment obrazu o zadanych współrzędnych początkowych i wymiarach [225].

Dla systemu *MOODLE* zaprojektowano procedurę *backup_copy_user_files*, która kopiuje niezbędne pliki z katalogu użytkowników do katalogu z plikami danego użytkownika, tworząc ich kopie zapasową. Funkcja *PclZipUtilCopyBlock* pozwala na kopiowanie pliku z ustalonym trybem kompresji. Wszystkie procedury dostępne są wraz z kodem źródłowym na stronie platformy [225].

Istotnym aspektem jest odpowiedni podział programu na moduły [14]. Każdy duży program powinien być podzielony na mniejsze, łatwiejsze do analizy składowe. Podział programu na moduły powinien być taki, aby powiązania między modułami były jak najmniejsze oraz aby jak najmniej elementów jednego modułu miało wpływ na budowę innego.

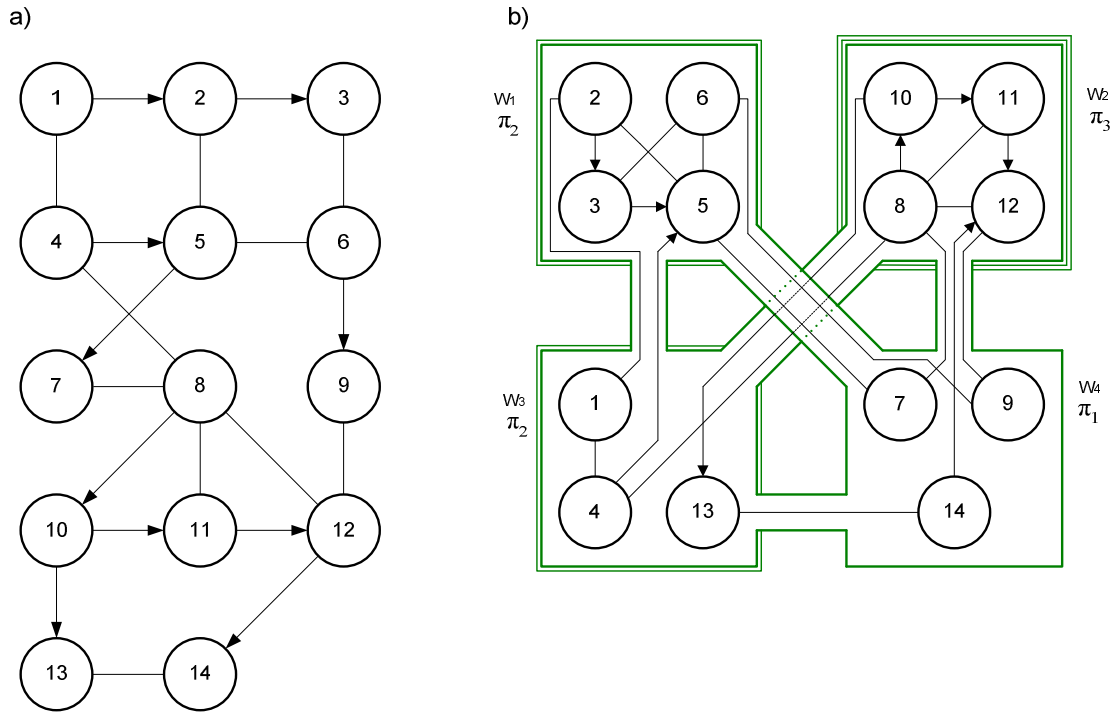
W module powinna być implementowana jedna kwestia projektowa. Nie należy łączyć niezwiązanych ze sobą zagadnień projektowych w jednym module zgodnie z tzw. zasadą separacji koncepcji (ang. *separation of concerns*). Użytkownicy modułów powinni koncentrować się na tym, co jest określone w interfejsie i specyfikacji modułu, a nie na sposobie implementacji modułu [2].

Program rozproszony nazywamy *sekwencyjnym* (ang. *serial*) w sensie wykonywania modułów, jeżeli w dowolnej chwili jest wykonywany co najwyżej jeden jego moduł. Program rozproszony nazywamy *równoległym* (ang. *parallel*) w sensie wykonywania modułów, jeżeli podczas jego wykonywania istnieje moment czasu, w którym co najmniej dwa moduły tego programu są wykonywane na dwóch komputerach. Program rozproszony nazywamy *współbieżnym* (ang. *concurrent*) w sensie wykonywania modułów, jeżeli zawiera taką parę modułów, że moment rozpoczęcia wykonywania jednego modułu zachodzi przed zakończeniem realizacji drugiego. Program równoległy jest współbieżny, ale nie na odwrót.

Przykładową strukturę programu rozproszonego przedstawiono na rysunku 2.1. Wierzchołki odnoszą się do modułów. Łuk (*A*, *B*) na rysunku 2.1a), gdzie wierzchołki *A* i *B* należą do zbioru {1, 2,..., 8}, reprezentuje przesyłanie danych z modułu *A* do *B* lub fakt, że moduł *B* wykona się po zakończeniu realizacji modułu *A*. Krawędzie odnoszą się do przesyłania danych między parą modułów w obie strony co najmniej raz podczas wykonywania programu. Kwadraty na rysunku 2.1b) reprezentują komputery połączone kanałami komunikacyjnymi. Nie wszystkie pary komputerów powinny być połączone. Istotne jest, aby między dowolną parą istniała możliwość przesyłania danych.

Jeżeli w rozproszonym programie sekwencyjnym moduł *A* z komputera w węźle w_1 wywołuje moduł *B* na komputerze przydzielonym do węzła w_2, to stan aktywności programu sekwencyjnego przemieszcza się z węzła w_1 do węzła w_2. Moduł *B* zwraca sterowanie do modułu *A* lub może wywołać inny moduł.

Klasyczny problem wyznaczania przydziału modułów programu rozproszonego polega na przydzieleniu modułów do komputerów tak, aby zminimalizować koszt wykonania programu [31, 143]. Modyfikacja tego podejścia polega na sformułowaniu zagadnienia minimalizacji łącznego czasu przetwarzania danych.



Rys. 2.1. Przykład programu rozproszonego wzorowanego na aplikacji *MOODLE*:
a) diagram programu rozproszonego zawierającego czternaście modułów
b) rozdział modułów w systemie z czterema komputerami
Źródło: opracowanie własne.

Aby zredukować czas komunikacji między maszynami w rozproszonym systemie komputerowym (RSK), moduły współpracujące powinny rezydować na tym samym komputerze. Z drugiej strony, aby skrócić czas obliczeń, moduły należy przydzielić tam, gdzie wykonują się najszybciej. Jeżeli dostępny jest komputer, na którym w najkrótszym czasie realizowane są wszystkie moduły z osobna, to umożliwi on także minimalizację czasu obliczeń zbioru modułów.

W systemach heterogenicznych istotną przyczyną rozproszenia modułów jest dążenie do wykorzystania specyficznych cech różnych typów komputerów. Jeżeli zatem istnieje para modułów o najkrótszych czasach realizacji na różnych komputerach, to korzyści czasowe, wynikające z przydziału minimalizującego łączny czas wykonania modułów, mogą być zniwelowane jedynie przez opóźnienia komunikacyjne. Występuje zatem konflikt między minimalizacją czasu komunikacji a minimalizacją czasu obliczeń na komputerach podczas przetwarzania danych w systemie rozproszonym [20].

Pierwszą odnotowaną w literaturze aplikacją, wykorzystującą optymalizację rozdziału modułów programu, był system graficzny w Brown University, w którym moduły programu podzielono między komputer IBM 360 a mikrokomputer graficzny [60]. Zastosowanie wówczas metody optymalizacji rozdziału modułów, którą zaproponował Stone [143], znacząco zmniejszyło koszt obliczeń. Model przydziału modułów programistycznych opracowano, opierając się na grafie z odpowiednią funkcją kosztu określoną na łukach. Metoda rozwiązania problemu polegała na skonstruowaniu tzw. grafu przydziału, a następnie na wyznaczeniu maksymalnego przepływu. Złożoność zaproponowanej metody była $O(V^3)$, gdzie V oznacza liczbę modułów.

Model Stone'a nie uwzględniał istotnych ograniczeń wynikających z wielkości pamięci operacyjnej. Rao, Stone i Hu udowodnili, że zadanie minimalizacji kosztu współbieżnego przetwarzania danych w systemie dwukomputerowym z ograniczoną wielkością pamięci operacyjnej należy do klasy problemów NP-trudnych [126].

Aby wyznaczyć optymalny rozdział modułów programów w rozległej sieci komputerowej ARPANET, Stone usiłował rozszerzyć swoją metodę minimalizacji kosztu użytkowania programu rozproszonego, na co najmniej trzy komputery [143]. Jednakże w tym wypadku metoda Stone'a nie gwarantowała wyznaczania rozwiązań optymalnych. Dla większej liczby komputerów w pracy [143] nie podano metody, gdyż algorytmu maksymalnego przepływu nie można było zaadaptować w odniesieniu do instancji o większej liczby komputerów [31].

Bokhari opracował efektywną metodę optymalizacji rozdziału modułów programu hierarchicznego [31]. Specyfika programu hierarchicznego polega na tym, że moduł nadrzędny wywołuje moduły poziomu niższego, które z kolei mogą wywoływać jednostki należące do jeszcze niższego poziomu hierarchii. Nie dopuszcza się do wywołania modułu, przez co najmniej dwa segmenty z poziomu nadrzędnego. Moduł może wchodzić w interakcje jedynie z jednostkami programowymi niższego poziomu oraz z co najwyżej jednym modułem stopnia nadrzędnego. W metodzie wykorzystuje się procedurę wyznaczania najkrótszej ścieżki w odpowiednio skonstruowanym grafie przydziału, a jej cechą charakterystyczną jest złożoność obliczeniowa rzędu $O(n^3)$, gdzie $n = \max\{V, k\}$, przy czym k reprezentuje liczbę poziomów hierarchii [32].

Koherentną syntezę (ang. *co-synthesis*) modułów w hierarchicznych systemach rozproszonych pokazano w [27]. Bazując na hierarchicznych, acyklicznych grafach zadań (ang. *task graphs*), generowana jest wydajna, hierarchiczna architektura oprogramowania, która spełnia ograniczenia dla systemów czasu rzeczywistego.

Zagadnienia dotyczące niezawodności w systemach z hierarchicznym oprogramowaniem, składających się z systemu, podsystemów i modułów zostały przedstawione w [130]. Dla wymagań niezawodnościowych dotyczących oprogramowania, można oszacować niezawodność każdego modułu oraz relacje niezawodnościowe między modułami, podsystemami i systemem. Wykorzystano moduły wchodzące w skład oprogramowania użytkowanego przez NASA.

Efektywna metoda optymalizacji przydziałów modułów w systemie rozproszonym została opracowana przez Towsleya dla programu o strukturze sekwencyjno-równoległej [149]. Programy te cechują się strukturą interakcji między modułami reprezentowaną grafem sekwencyjno-równoległym. W metodzie stosuje się procedurę wyznaczania najkrótszej ścieżki w skonstruowanym grafie przydziału, a złożoność obliczeniowa metody jest rzędu $O(n^4)$, gdzie $n=\max\{V, k\}$, przy czym k reprezentuje liczbę poziomów modułów w programie sekwencyjno-równoległym.

Powyższe metody minimalizują łączny koszt użytkowania programu rozproszonego. Szczególną sytuację przetwarzania danych za pomocą programu potokowego w RSK o strukturze typu łańcuch analizowano w [31]. Czas trwania ramki to czas trwania cyklu wykonywania modułów podczas przetwarzania potokowego na najbardziej obciążonym komputerze [31]. Efektywną metodę minimalizacji czasu trwania ramki cechuje złożoność obliczeniowa rzędu $O(V^4)$. W metodzie korzysta się także z procedury wyznaczania najkrótszej ścieżki.

Do optymalizacji przydziału modułów zbioru programów potokowych w RSK o strukturze gwiazdy opracowano metodę minimalizacji czasu trwania ramki o złożoności obliczeniowej $O(V^3 \log V)$ [30].

Interesujący nurt badań ukierunkowano na opracowanie modeli i metod optymalizacji rozdziału modułów programów w macierzy procesorów, co jest określane mianem zagadnień *mappingu* [31,132]. W metodzie zaproponowanej przez Liu i Soffa graf przydziału dzielony jest na podgrafy, w odniesieniu do których równolegle stosowane są algorytmy optymalizacji [104]. Zaletami tego podejścia są: możliwość użycia algorytmów o większej złożoności oraz zastosowanie różnorodnych technik *mappingu* jednocześnie. Zmiana w programie nie pociąga modyfikacji grafu przydziału.

Chaudhary i Aggarwal zaproponowali alternatywną heurystyczną strategię *mappingu*, która bazuje na teorii grafów i programowaniu matematycznym [37].

Efektywną metodę dystrybucji danych w macierzy procesorów przedstawili Kalns i Ni [89]. Pozwalała ona osiągnąć wyniki lepsze o około 40% w porównaniu z innymi metodami. Metodę udoskonalono w [77].

W algorytmie Legranda dane są dzielone między procesory umieszczone w tzw. wirtualnym pierścieniu [103]. Celem jest minimalizacja czasu wykonania zadań. Podano także zasadę skonstruowania wirtualnego pierścienia w sieci, w której nie wszystkie pary węzłów komunikacyjnych mają bezpośrednie połączenia.

Interesujący problem dotyczy optymalizacji rozdziału modułów przy występowaniu terminów zakończenia zadań. Nieprzekraczalny termin (ang. *deadline*) wykonania zadania nazywany jest *twardym*, jeśli jego przekroczenie może prowadzić do nieprawidłowego działania systemu [92]. W ograniczeniu *miękkim* dopuszcza się przekroczenie terminu wykonania zadania, przy czym ważne jest jego ukończenie. Ponadto wyróżnia się ograniczenia *umiarkowanie twarde* (ang. *weakly-hard*). W tym przypadku dopuszcza się przekroczenie terminów wykonania zadań, aczkolwiek w przewidywalny sposób i przy założeniu, że liczba ich jest ograniczona [108].

Moduły mogą być wykonywane cyklicznie lub acyklicznie [110]. W [76] przyjęto, że moduły są wykonywane cyklicznie i dodatkowo podlegają ograniczeniom kolejnościowym. Do maksymalizacji miary skutecznej realizacji zadań w terminach zastosowano metodę podziału i ograniczeń. Liniowy problem całkowitoliczbowy z funkcję celu, która minimalizuje maksimum z wszystkich opóźnień cyklicznych zadań sformułowano w [148]. Opóźnienie jest różnicą między czasem ukończenia (ang. *completion time*) a nieprzekraczalnym terminem wykonania modułu.

Xie i Qin [157] opracowali schemat alokacji aplikacji równoległych w klastrach, który, oprócz miary skutecznej realizacji zadań w terminach, maksymalizuje również jakość bezpieczeństwa podczas alokacji. Przegląd algorytmów, w których cyklicznie wykonywane moduły mogą nie zostać wykonane w terminach, został zaprezentowany w [108]. Przyjęto, że moduły nie są poddane ograniczeniom kolejnościowym i mogą zostać wywłaszczone w dowolnym momencie.

Pokrewnymi problemami są zagadnienia wyznaczania przydziału oraz liczby kopii zbiorów danych [117]. Nie będą one jednak w rozprawie poruszane.

Niektóre modele i metody dotyczące przydziału modułów programistycznych opierają się na pojęciach i procedurach teorii grafów. Modele te cechują się wówczas wrażliwością na wystąpienie interakcji między dowolną parą modułów. Wybrane interakcje mogą zakłócić „regularność" grafu reprezentującego komunikację danych między modułami. Jest to zasadnicza wada podejść wywodzących się z teorii grafów.

Bokhari [31] i Sinclair [136] podali, że problem optymalizacji przydziału modułów polegający na minimalizacji kosztów wykonania programu współbieżnego o dowolnej strukturze interakcji między modułami, dla co najmniej czterech komputerów, należy do klasy problemów NP-trudnych, nawet przy braku ograniczeń wynikających z limitowanych wielkości zasobów. Złożoność obliczeniowa problemu, w którym rozpatruje się trzy komputery, jest zagadnieniem otwartym [30].

Interesujący nurt badań zapoczątkowała praca Price'a i Poocha, w której autorzy zaproponowali oryginalne podejście do formułowania problemów optymalizacji przydziału modułów programów w systemach rozproszonych [124]. Istotnym elementem ich podejścia było przyjęcie nieliniowej funkcji kosztów. Sposób rozwiązania problemu bazował na heurystycznej metodzie lokalnego przeszukiwania. Rozważano także metodę programowania dynamicznego do optymalizacji przydziału modułów [54]. Następnie Iqbal opracował metodę przeszukiwania charakteryzującą się złożonością obliczeniową $O(n^2 \log n)$ [82]. Powyższe metody nie uwzględniały ograniczeń wynikających z wielkości zasobów systemu.

Konakia i Tobagi opracowali wariant metody podziału i ograniczeń do optymalizacji przydziału modułów programów w systemie z limitowanymi zasobami lokalnymi [91]. Metody wykorzystujące sieci neuronowe w systemie z ograniczonymi zasobami lokalnymi przedstawiono w [23]. Optymalizację przydziału modułów w przypadku ograniczonej zarówno pamięci lokalnej, jak i wydajności obliczeniowej przedstawiono w [59].

W zagadnieniach wyznaczania optymalnego przydziału modułów, obok kosztu wykonania programu, stosuje się inne funkcje celu, takie jak łączne obciążenie systemu [31], obciążenie najbardziej eksploatowanego komputera [40], równomierność obciążenia komputerów [105] czy niezawodność wykonania programu rozproszonego [90]. Rozpatruje się także prawdopodobieństwo ukończenia zadań w terminach [18, 76] lub miarę podobieństwa grafów ważonych [11, 118]. W programach potokowych często stosowanym kryterium jest czas trwania ramki [31].

O metodach rozdziału modułów programów w sieci komputerowej z dzielonym kanałem transmisji danych dyskutowano w [136], gdzie zaproponowano, aby dla zadanych wartości obciążenia komunikacyjnego sieci wyznaczać przydziały modułów programów za pomocą scharakteryzowanych wcześniej metod. Lo zaproponowała karanie przydziału wszystkich modułów do jednego komputera, opracowała algorytm optymalizacji rozdziału modułów programów, a następnie porównała zalety skonstruowanego algorytmu *A* z algorytmem przydziału zadań w rozproszonym systemie operacyjnym Micros [105]. Ajith i Murthy wykorzystali algorytm *A* w celu optymalnej alokacji modułów do procesorów w RSK [4]. Oprócz kosztu wykonania modułu rozważano koszt komunikacji między modułami zlokalizowanymi w wybranych procesorach. Zmodyfikowany algorytmu *A* rozpatrywano także w [88].

Chu i Lan zaprojektowali przydział 23 modułów do 3 komputerów w ramach projektu DPAP dla systemu obrony powietrznej [40]. Była to pierwsza aplikacja, w której minimalizowano obciążenie najbardziej wykorzystywanego komputera.

Kartik i Murthy opracowali wariant metody podziału i ograniczeń do maksymalizacji niezawodności wykonania programu rozproszonego [90]. Ponadto dokonali przeglądu stosowanych metod w odniesieniu do tak sformułowanego NP-trudnego zadania optymalizacji.

Problemom konstrukcji systemów rozproszonych tolerujących uszkodzenia poświęcono prace [34, 79]. Metodę alokacji modułów w tej klasy systemach przedstawiono w [38]. Po wstępnej nie nadmiarowej alokacji modułów, wyznaczana jest ponowna alokacja, tym razem nadmiarowa. W przypadku awarii procesora, moduł do niego przydzielony jest realizowany na jednym z pozostałych procesorów.

Aby osiągnąć wysokie prawdopodobieństwo wykonania zadań, niektóre z modułów mogą być replikowane, a następnie wykonywane bez względu na fakt wystąpienia uszkodzenia [151]. Zadania poddane replikacji są dodawane do zbioru wykonywanych zadań, jeśli zwiększają prawdopodobieństwo ich wykonania.

Algorytm wykorzystujący diagramy decyzji binarnych BDD (ang. *Binary Decision Diagrams*) zaprezentowano w [150]. Jest to pierwsza praca, w której BDD zostały użyte do optymalizacji niezawodności. W algorytmie wykorzystano rozwiązania zaproponowane w [3].

Metodę stochastycznej optymalizacji zastosowano do alokacji modułów w [155]. Problem optymalizacji przetransformowano w problem stochastyczny, a do jego rozwiązania użyto modyfikację metody Monte Carlo.

Na tym tle niezmiernie interesujące jest rozważenie innego rodzaju decyzji podejmowanych na etapie projektowania lub modyfikacji RSK, które mają wpływ na skrócenie czasu rozproszonego przetwarzania danych. Kluczową rolę odgrywa dobór odpowiednich typów komputerów, gdyż koszt systemu zależy w dużym stopniu od cen maszyn, a wydajność systemu jest związana z wydajnością obliczeniową komputerów wchodzących w jego skład.

Dobierając zestaw komputerów oraz przydzielając do nich moduły, można uzyskać rozwiązania minimalizujące koszt wykonania programu oraz maksymalizujące wydajność systemu. W wypadku programu sekwencyjnego w systemie dwukomputerowym dogodnie jest minimalizować czas wykonania programu lub koszt komputerów. Wymagane jest wówczas wprowadzenie dodatkowych założeń do modeli znanych z literatury przedmiotu [20].

Zasadnicza niedogodność dążenia do integracji decyzji projektowych w RSK polega na istnieniu sprzeczności między celami projektowania. Jeżeli zostaną zastosowane komputery cechujące się niższymi cenami zakupu, to czasy wykonania modułów programu sekwencyjnego zazwyczaj będą dłuższe. Z kolei, dążąc do minimalizacji czasu sekwencyjnego przetwarzania danych, należałoby wykorzystać najbardziej wydajne, a co za tym idzie – najdroższe komputery.

Podobna sytuacja konfliktowa występuje w innym modelu podczas minimalizacji czasu trwania ramki systemu przetwarzającego potokowo dane oraz minimalizacji kosztu komputerów.

Z kolei w wypadku przetwarzania współbieżnego minimalizacja kosztu przetwarzania jest związana z doborem tańszych komputerów o mniejszej mocy obliczeniowej. W rezultacie może to prowadzić do wydłużenia czasu wykonania programu.

Modelowanie zaistniałych sytuacji konfliktowych oparte może być w ramach teorii optymalizacji wielokryterialnej na sformułowaniu odpowiednich zagadnień

optymalizacyjnych z uwzględnieniem wybranych preferencji decyzyjnych. Dlatego w rozprawie położono nacisk na formułowanie zagadnień polioptymalizacji, w których poszukuje się reprezentacji rozwiązań optymalnych w sensie Pareto [6].

Za pomocą algorytmów ewolucyjnych wyznaczono najlepsze rozwiązania, jakie uzyskano do tej pory dla szeregu instancji zagadnień optymalizacji [125]. W pracy [39] opracowano model podziału i alokacji obiektów do odwzorowania aplikacji zorientowanych obiektowo na środowisko heterogeniczne. Wykorzystano algorytm genetyczny NPGA (ang. *Niched-Pareto Genetic Algorithm*). Wielokryterialne algorytmy ewolucyjne wykorzystuje się do wyznaczania reprezentacji efektywnych przydziałów modułów [19].

Alokację modułów w *MOODLE* można przeprowadzić w środowisku linuksowym przy użyciu sieciowego systemu plików. Na każdym komputerze powinna być założona identyczna struktura katalogów systemu *MOODLE*. Zawartość katalogu znajduje się na komputerze z zainstalowanym serwerem NFS, a na pozostałych maszynach będących klientami NFS, katalog ten należy zamontować w sensie Linuxa. Każdy moduł *MOODLE* znajduje się w odrębnym katalogu lub katalogach. Specyfikację opisującą, które moduły mogą zostać poddane alokacji określa się w plikach konfiguracyjnych systemu Linux.

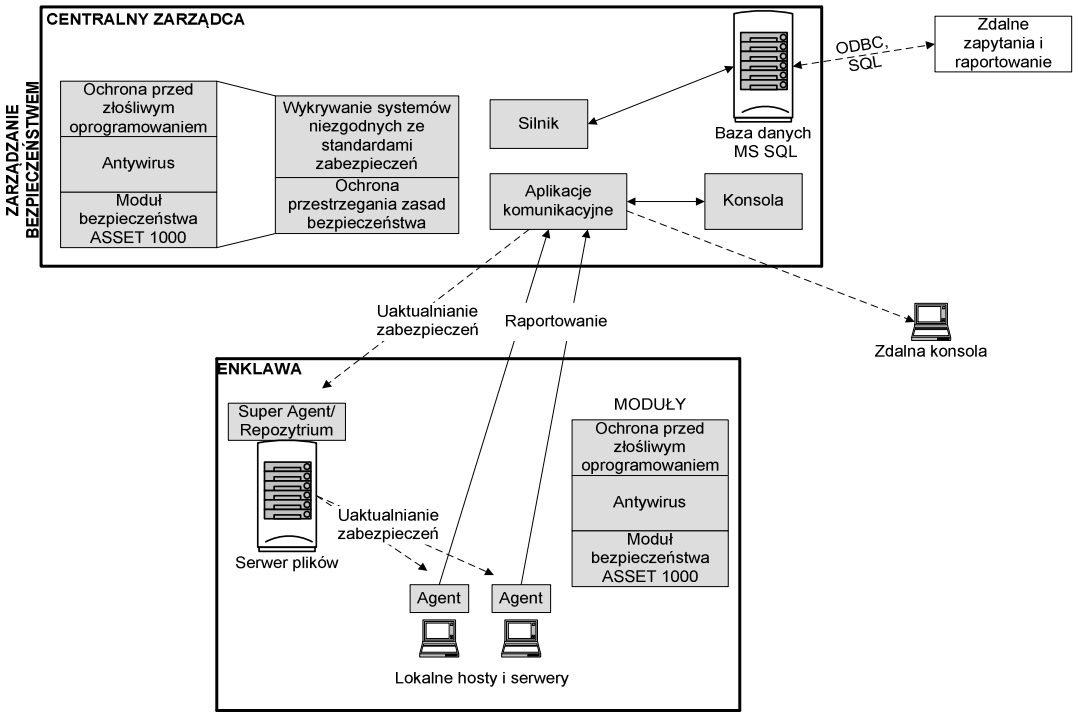
Jest wysoce prawdopodobne, że wcześniej nie były podejmowane próby rozproszenia systemu *MOODLE* w celu poprawy dostępności systemu i miary skutecznej realizacji zadań w terminach. Udany eksperyment z alokacją modułów w systemie *MOODLE* opisano w dodatku B.

2.2. Podstawowe założenia informatyczne modelu szkolenia wojskowego

Systemy szkolenia na odległość odgrywają kluczową rolę w szkoleniu wojskowym nowoczesnych armii. Istotną decyzją jest dobór platformy programistycznej. W armii amerykańskiej od 2007 roku następuje transformacja AKO w DKO (ang. *Defence Knowledge Online*), tak aby DKO było zgodne ze standardami SOA (ang. *Service Oriented Architecture*).

Planowana architektura ma być oparta o modularne podkatalogi, mające zapewnić większą skalowalność. Aby zapewnić wymagany poziom mobilności i modularności, wykorzystuje się EDS (ang. *Enterprise Directory Service*) w powiązaniu z NCES (ang. *Net-Centric Enterprise Services*).

Narzędziami mającymi podnieść bezpieczeństwo są Cisco NAC (ang. *Network Admission Control*) oraz przedstawiony na rysunku 2.2 HBSS (ang. *Host-Based Security System*) bazujący na rozwiązaniach firmy McAfee [139].



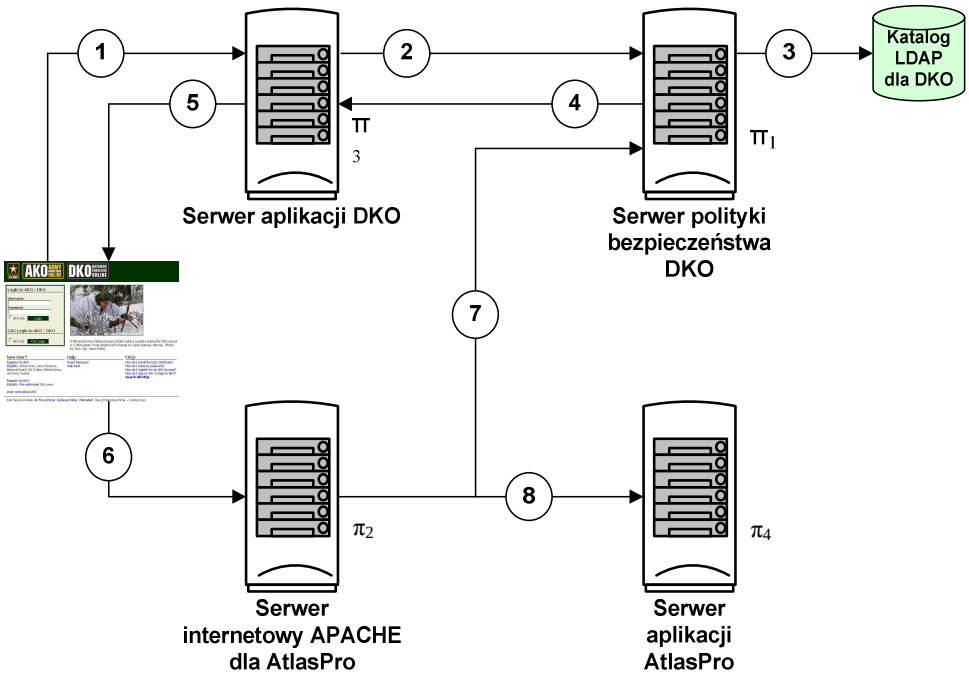
Rys. 2.2. System bezpieczeństwa HBSS stosowany w US Army [139]

Oprócz powyższego rozwiązania, jako platforma *e-learningu* wykorzystany może być *WBTExpress*, którego liczba serwerów zależy od liczby użytkowników. W architekturze *MultiServer* system składa się z 22 serwerów (w tym 20 serwerów pomocniczych), co pozwala na realizację szeregu wariantów alokacji modułów. Wadami są: wysoka cena oraz możliwość aktualizacji i pomocy technicznej tylko przez okres 12 miesięcy. Należy wspomnieć o *WBTExpress Free MOODLE Edition* – bezpłatnym narzędziu do projektowania kursów kompatybilnych z *MOODLE* [176].

Niech $\boldsymbol{M} = \{m_1,...,m_v,...,m_V\}$ oznacza zbiór modułów, który jest wykonywany w rozproszonym systemie *MOODLE* proponowanym do wykorzystania w szkolnictwie wojskowym. W wyniku przetwarzania rozproszonego można wyznaczyć także zbiór zadań $\overline{\boldsymbol{M}} = \{M_1,...,M_m,...,M_M\}$. Wykonaniu modułu odpowiadać może kilka zadań, które są przydzielone do tego samego komputera, co stowarzyszony z nim moduł.

Zakłada się, że moduł m_v może być wykonywany na komputerach, których rodzaje należą do zbioru $\boldsymbol{\Pi} = \{\pi_1,...,\pi_j,...,\pi_J\}$. Przykładowo, w celu uzyskania dostępu

do systemu *AtlasPro* klasy LMS poprzez portal DKO armii amerykańskiej, mogą być użyte komputery różnego rodzaju do implementacji szeregu serwerów (rys. 2.3). Przykładowo *serwer Apache* z rysunku 2.3 może być zainstalowany na komputerze klasy IBM System z10 Enterprise Class (π_1), IBM System x3500 M3 (π_2) lub innym (π_3). Komputery tej klasy mogą być zastosowane do implementacji *serwera aplikacji* DKO w innym węźle.



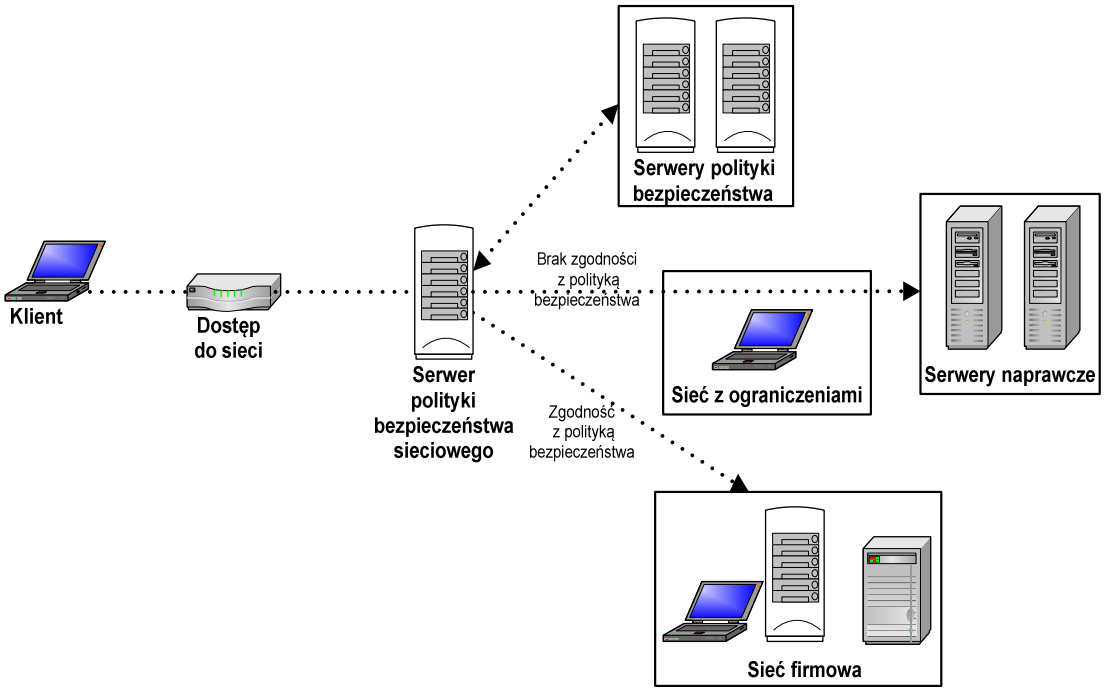
Rys. 2.3. Dostęp do systemu *AtlasPro* poprzez portal DKO [139]

Poprzez portal DKO dostęp do systemu *AtlasPro* jest możliwy przy wykorzystaniu pojedynczego logowania SSO (ang. *Single Sign On*). W procesie identyfikacji użytkownika wykorzystywany jest serwer polityki bezpieczeństwa (ang. *Policy Server*), który znacząco różni się parametrami technicznymi od serwerów, takich jak serwer internetowy z oprogramowaniem Apache czy serwer aplikacji DKO. Rodzaje komputerów do implementacji wybranego serwera są zazwyczaj różne. Może jednakże wystąpić sytuacja, gdy w systemie funkcjonują komputery tej samej klasy, realizujące funkcje różnych serwerów.

W systemie rozproszonym *AtlasPro* wdrażany jest serwer pocztowy MS Exchange 2007, zastępujący wersje 5.5 i 2003. Przyczyną jest wspieranie 64-bitowej architektury oraz uproszczenie *routingu*.

Systematyczna replikacja klastrów CCR (ang. *Cluster Continous Replication*) pozwala zaoszczędzić kilka milionów dolarów rocznie za pomocą niższych kosztów wymaganego sprzętu. Ponadto CCR cechuje się tym, że nie stosuje się kosztownych urządzeń do wykonywania kopii zapasowych (ang. *tape backups*). Replikacja klastrów umożliwia także uniknięcia występowania pojedynczych punktów uszkodzeń [139].

Zabezpieczenie dostępu do sieci realizowane jest w oparciu o serwery polityki bezpieczeństwa sieciowego (ang. *Network Policy Server*) oraz serwery naprawcze (ang. *Fix Up Server*) (rys. 2.4).



Rys. 2.4. Zasada dostępu do sieci w DKO [139]

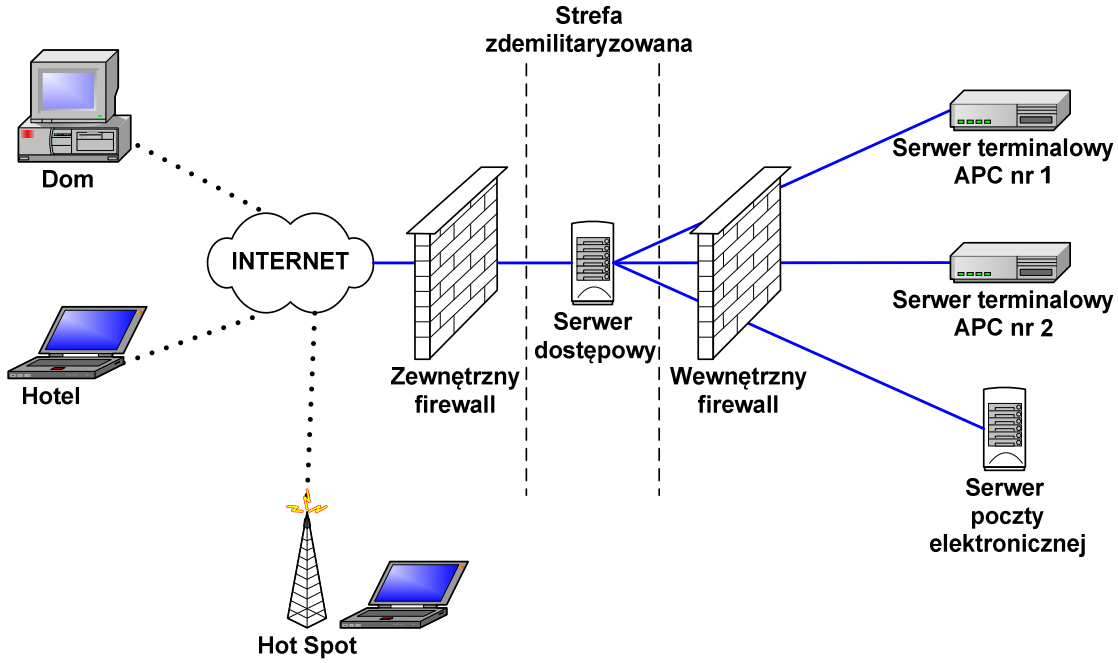
Zdalny dostęp do zasobów informacyjnych, sprzętowych lub programowych, umieszczonych na serwerach w strefie APC (ang. *Area Processing Center*) realizuje się przy wykorzystaniu serwera dostępowego (ang. *Gateway Server*) znajdującego się w strefie zdemilitaryzowanej DMZ (rys. 2.5).

W zależności od statusu i architektury węzła sieciowego, w jego składzie może się znaleźć serwer zarządzania systemem, serwer strumieniowego wideo, serwer wirtualnych konferencji lub serwer SQL [46].

Węzły obliczeniowe mogą być zlokalizowane w centrach obliczeniowych: uczelni wojskowych, sztabów, zespołów informatycznych i innych jednostek wojskowych. Przyjmuje się założenie, że komputery mogą być usytuowane tylko

w ustalonych węzłach należących do zbioru $\boldsymbol{W} = \{w_1,...,w_i,...,w_I\}$. Jeśli w systemie szkolenia jest I węzłów, to przykładowo w jednym z nich alokowana może być baza danych, a na pozostałych inne moduły systemu *MOODLE*.

Pary węzłów połączone są kanałami umożliwiającymi realizację interakcji między modułami programu, np. synchronizację wykonywania modułów, komunikację między modułami lub rywalizację o zasoby. Przyjmuje się, że w węźle przetwarzania danych jest usytuowany tylko jeden komputer, wybrany ze zbioru możliwych typów komputerów. Niech także zadana jest macierz szacowanych czasów realizacji modułów programu na komputerach $\boldsymbol{T}=[t_{vj}]_{VxJ}$, gdzie t_{vj} oznacza szacowany czas wykonania modułu m_v na komputerze typu π_j.



Rys. 2.5. Zdalny dostęp do serwerów w APC [139]

Ponieważ system *MOODLE* jest napisany w języku PHP do oszacowania czasów realizacji modułów na komputerach można wykorzystać odpowiednie narzędzia do testowania i mierzenia wydajności kodu programu.

Serwer *Apache* posiada narzędzie o nazwie *ab* służące do pomiaru wydajności serwera. Dostępny jest wybór opcji, z którymi może być przeprowadzony pomiar.

Natomiast narzędzie *Siege* umożliwia zmierzyć wydajność kodu „pod oblężeniem” (ang. *under siege*) zadanej liczby współbieżnych użytkowników. Użytkownik może zażądać dostępu do serwera ustaloną liczbę razy.

Xdebug's Profiler pozwala na: analizę kodu PHP, określenie „wąskich gardeł” oraz wskazanie części programu, które mogą zostać zmodyfikowane pod kątem wzrostu wydajności.

NuSpherePhpED posiada narzędzie *dbg* do lokalnej lub zdalnej analizy programów. Zintegrowany z nim *PHP Profiler* pozwala oszacować czasy wykonywania plików napisanych w języku PHP [233].

Przyjęto, że zadana jest macierz szacowanych czasów komunikacji danych między parami modułów $\boldsymbol{\tau} = [\tau_{vu}]_{V \times V}$, gdzie τ_{vu} oznacza szacowany czas transmisji danych do segmentu m_u z modułu m_v. W modelu podstawowym przyjmuje się, że czas komunikacji między parą zadań jest znany *a priori*, stały i zależy od komunikującej się pary [143]. Wprowadzenie tego założenia wynika z faktu, że czas komunikacji między parą zadań zależy od objętości przesyłanych danych oraz od przepustowości połączenia wirtualnego między nimi. Zakłada się, że przepustowość połączenia wirtualnego między zadaniami jest jednakowa i dlatego czas komunikacji między procesami nie zależy ani od komputerów, ani od tego, do jakich węzłów zostały one przydzielone. Jeśli moduły są realizowane w tym samym węźle, to przyjmuje się, że nie występują opóźnienia komunikacyjne między modułami. Zakłada się, że $\tau_{vv} = 0$ dla $v = \overline{1,V}$.

Warto podkreślić różnicę między sposobami wyznaczania charakterystyk czasowych w zależności od sposobu przetwarzania sekwencyjnego. Jeżeli nie występują interakcje między modułami, to czas realizacji modułów jest ciągłym przedziałem od momentu rozpoczęcia wykonywania modułu do momentu jego zakończenia. Opóźnienie między modułami związane może być z przesyłaniem danych do kolejnego modułu, przekazaniem sterowania do komputera wykonującego kolejny moduł lub z rywalizacją o zasoby.

W niektórych sekwencyjnych programach rozproszonych interakcje między modułami mogą występować wielokrotnie. Moduł rozpoczynający interakcję zawiesza swoje wykonywanie na komputerze i oczekuje na powrót sterowania z wywołanego modułu, po którym wznawia wykonywanie. Moduł odbierający interakcję rozpoczyna realizację na komputerze, a po zakończeniu zwraca możliwość wykonywania do modułu wywołującego.

Czasem cząstkowym realizacji modułu nazywamy czas wykonywania instrukcji tego modułu od momentu jego rozpoczęcia (lub wznowienia) do momentu zakończenia (lub wywołania innego modułu). Szacowany czas przetwarzania dla modułu jest sumą

czasów cząstkowych wykonywania tego modułu. Czas interakcji τ_{vu} jest sumą czasów przekazywania danych i sterowania z modułu m_v do modułu m_u.

Stosowanych jest kilka sposobów komunikacji między modułami w *MOODLE*. Pierwszy sposób polega na użyciu zdarzeń (ang. *events*). Przeprowadzane jest asynchroniczne przesyłanie komunikatów. Moduł źródłowy wysyła komunikat do kolejki zdarzeń wybranego segmentu.

Kolejny sposób komunikacji między modułami polega na tym, że niektóre moduły bezpośrednio używają lokalnych danych innego modułu. Jest to forma silnego powiązania komunikacyjnego modułów (ang. *strong coupling*). Interesująca strategia, będąca rozszerzeniem poprzedniej, polega na użyciu API (ang. *Application Programming Interface*) innego modułu, w jednym z plików bibliotek danego modułu.

Rozwijana jest także technika komunikacji między modułami polegająca na zastosowaniu dedykowanego pliku bibliotek. Pozwala to na implementację wybranych funkcji do interakcji między modułami (ang. *cross-module function entries*), a w rezultacie na zminimalizowanie powiązań modułów [225].

Do oszacowania czasu komunikacji między modułami *MOODLE* można wykorzystać narzędzia takie jak *Xdebug* w niektórych wersjach jądra Linuksa oraz zintegrowany z *NuSpherePhpED* moduł *PHP Profiler* w systemach Windows.

W przetwarzaniu sekwencyjnym bez przerywania wykonywania modułów każde zadanie komunikuje się z innym zadaniem, co najwyżej raz. W tym modelu prawdziwa jest następująca implikacja:

$$[\tau_{vu} \neq 0] \Rightarrow [\tau_{uv} = 0],\ v = \overline{1,V},\ u = \overline{1,V},\ v \neq u. \tag{2.1}$$

W wypadku przerywania wykonywania modułu i realizacji interakcji między modułami dla $\tau_{vu} \neq 0$ może zachodzić $\tau_{uv} \neq 0$. Załóżmy, że $\tau_{vu} := \tau_{vu} + \tau_{uv}$ i τ_{uv}=0 dla $v, u = \overline{1,V}, v < u$. Symbol := oznacza operację podstawiania. Zaletą tego założenia jest zachowanie konwencji, w której jeżeli $\tau_{vu} \neq 0$ dla $v, u = \overline{1,V}, v < u$, to τ_{vu} reprezentuje łączny czas interakcji między modułami m_v oraz m_u, przy czym nie jest istotne, jaki moduł inicjuje interakcje. Aby, sumując czasy interakcji, nie dodawać dwukrotnie uprzednio niezerowej wartości τ_{uv}, należy dodatkowo przyjąć, że τ_{uv}=0.

Rozważa się model minimalizacji sumarycznego obciążenia wykonania programu współbieżnego. Niech dana jest macierz szacowanych obciążeń związanych z przetwarzaniem $\overline{\boldsymbol{T}} = [\bar{t}_{vj}]_{V \times J}$, gdzie $\bar{t}_{vj}$ reprezentuje szacowane obciążenie związane

z wykonaniem modułu m_v na komputerze typu π_j [s]. Jeżeli interakcje generowane z m_v do m_u szacowane są na $\tilde{\tau}_{vu}$ [s], a interakcje generowane z m_u do m_v trwają łącznie $\tilde{\tau}_{uv}$ [s], to przyjmuje się, że $\bar{\tau}_{vu} := \tilde{\tau}_{vu} + \tilde{\tau}_{uv}, \bar{\tau}_{uv} = 0$ dla $v, u = \overline{1,V}, v < u$.

Przyjmuje się również, że zadana jest macierz szacowanych obciążeń związanych z interakcjami między parami modułów $\bar{\tau} = [\bar{\tau}_{vu}]_{V \times V}$. W rozproszonym programie współbieżnym szacowane czasy wykonywania modułów oraz szacowane czasy interakcji wyznacza się podobnie jak dla rozproszonego programu sekwencyjnego z przerywaniem wykonywania modułów. Zasadnicza różnica polega na tym, że moduł generujący interakcję może kontynuować wykonywanie. Wówczas moduł jest realizowany jednocześnie wraz z przekazywaniem danych do wybranego segmentu.

Przyjmuje się, że podstawowy model programu dotyczy rozproszonego programu sekwencyjnego i jest opisany za pomocą uporządkowanej piątki (***M, Π, W***, ***T***, ***τ***). Natomiast model programu współbieżnego jest opisany za pomocą piątki (***M***, ***Π***, ***W***, $\bar{\boldsymbol{T}}$, $\bar{\boldsymbol{\tau}}$).

2.3. Zbiór przydziałów dopuszczalnych

Do modelowania przydziałów dopuszczalnych wykorzystuje się podstawy teoretyczne z prac [15, 20]. Rozproszony program sekwencyjny jest scharakteryzowany za pomocą struktury przetwarzania. W analizie systemowej pod pojęciem struktury systemu rozumie się zbiór relacji między elementami systemu, co oznacza, że *system* może być rozumiany jako para, którą tworzą zbiór elementów (konfiguracja) oraz struktura [134].

Definicja 2.1

Strukturą przetwarzania rozproszonego programu sekwencyjnego nazywamy relację dwuczłonową R_M określoną na zbiorze modułów programu, która uwzględnia uwarunkowania kolejnościowe między modułami podczas wykonywania programu.

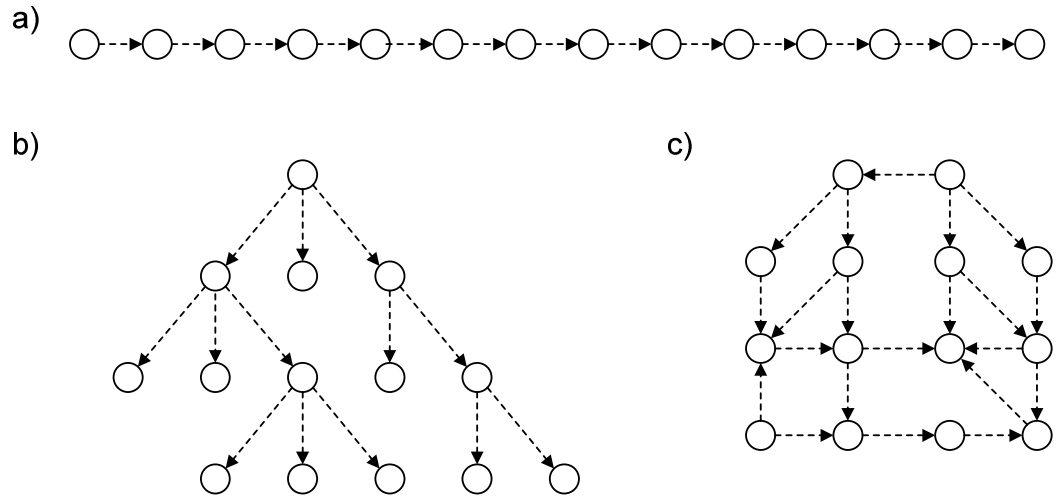
Przyjmuje się, że uwarunkowanie kolejnościowe między m_v *i* m_u polega na tym, że rozpoczęcie wykonywania m_u może wystąpić najwcześniej dopiero po zakończeniu realizacji modułu m_v. Uwarunkowanie kolejnościowe wynika z przekazania danych lub sterowania. Jeżeli m_u odbiera dane od modułów, to odbywa się to sekwencyjnie.

Moment rozpoczęcia wykonywania modułu m_u odpowiada momentowi zakończenia odbioru danych od ostatniego z modułów wysyłających dane do tego segmentu.

Strukturę przetwarzania rozproszonego programu sekwencyjnego można zdefiniować następująco:

$$R_M = \{(m_v, m_u) \in \boldsymbol{M} \times \boldsymbol{M} \mid m_v \text{ przekazuje dane lub sterowanie do } m_u,\ m_v \neq m_u\} \quad (2.2)$$

Relacja R_M jest przeciwzwrotna [127]. System wykonywania rozproszonego programu sekwencyjnego $G_M = (\boldsymbol{M}, R_M)$ można przedstawić jako graf skierowany (digraf), w którym wierzchołki reprezentują moduły, a łuki – uwarunkowania kolejnościowe [52, 57]. Na rysunku 2.6 zobrazowano wybrane struktury systemów rozproszonych realizujące przetwarzanie sekwencyjne.



Rys. 2.6. Wybrane struktury przetwarzania rozproszonego programu sekwencyjnego składającego się z czternastu modułów:
a) sekwencja, b) drzewo, c) struktura mieszana
Źródło: opracowanie własne.

W rozproszonym programie współbieżnym występują interakcje polegające na wzajemnym przesyłaniu danych między modułami. Jeżeli między parą modułów występują interakcje, to taka sytuacja jest oznaczana krawędzią. Przyjmuje się, że koszt komunikacji między modułami jest sumą kosztów przesyłania danych między tą parą. Między parą modułów występuje ograniczenie kolejnościowe (oznaczone łukiem), zachodzą interakcje (oznaczone łukiem lub krawędzią) lub relacja nie występuje.

Przydział modułów do węzłów można określić za pomocą funkcji przydziału modułów $\Phi_m : \boldsymbol{M} \rightarrow \boldsymbol{W}$. Zapis $\Phi_m(m_v) = w_i$ oznacza, że m_v przydzielono do węzła w_i.

Zbiór modułów przydzielonych do w_i określono jako $M_i = \{m_v \in \boldsymbol{M} \mid \Phi_m(m_v) = w_i\}$, przy czym $M_i \subset \boldsymbol{M}$. Rodzinę zbiorów segmentów przydzielonych do węzłów oznaczono jako $M_R = \{M_1,...,M_i,...,M_I\}$. Zbiory modułów przydzielone do węzłów są rozłączne $M_i \cap M_j = \varnothing$ dla $i, j = \overline{1,I}$, $i \neq j$. Ich sumą mnogościową jest zbiór $\boldsymbol{M} = \bigcup_{i=1}^{I} M_i$.

Przydział modułów programu do węzłów można również określić za pomocą binarnego wektora w następujący sposób:

$$x^m = [x_{11}^m,..., x_{1i}^m,..., x_{1I}^m,..., x_{vi}^m,..., x_{VI}^m]^T , \tag{2.3}$$

gdzie $x_{vi}^m = \begin{cases} 1 \text{ gdy } m_v \text{ usytuowano w } w_i, \\ 0 \text{ w przeciwnym razie,} \end{cases}$ dla $v = \overline{1,V}$, $i = \overline{1,I}$.

Przykładowy przydział modułów systemu *MOODLE* z rysunku 2.1 można wyspecyfikować następująco:

$$x^m = [0,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,]^T .$$

Przydział modułów do węzłów można scharakteryzować także za pomocą binarnej macierzy $\overline{X}^m = [x_{vi}^m]_{V \times I}$.

Ponieważ każdy moduł powinien zostać przydzielony tylko do jednego węzła, to formalnie powyższy postulat można zapisać następująco:

$$\sum_{i=1}^{I} x_{vi}^m = 1 \text{ dla } v = \overline{1,V}. \tag{2.4}$$

Aby umożliwić spełnienie ograniczenia (2.4), można przyjąć reprezentację przydziału modułów do węzłów za pomocą całkowitoliczbowego wektora przydziału $X^m = (X_1^m,...,X_v^m,...,X_V^m)^T$, gdzie X_v^m oznacza numer węzła, do którego przydzielono moduł m_v. Jeżeli $x_{vi}^m = 1$, to $X_v^m = i$, przy czym $1 \le X_v^m \le I$. Przykładowy przydział modułów systemu *MOODLE* z rysunku 2.1 można zapisać następująco za pomocą wektora $X^m = (3, 1, 1, 3, 1, 1, 4, 2, 4, 2, 2, 2, 3, 4)^T$.

Funkcja przydziału Φ_m jest przydatna w rozważaniach teoretycznych [21, 36]. Binarny wektor przydziału x^m umożliwia formułowanie zagadnień optymalizacji [22]. Całkowitoliczbowy wektor X^m jest stosowany w metodach optymalizacji [111].

Sposób kodowania przydziałów ma znaczący wpływ na przyśpieszenie obliczeń prowadzących do wyznaczenia optymalnego rozwiązania. Liczba możliwych (dopuszczalnych lub niedopuszczalnych) wariantów binarnego wektora przydziału

modułów programu x^m jest równa 2^{VI} i rośnie szybciej niż liczba możliwych wariantów całkowitoliczbowego wektora przydziału X^m, która wynosi I^V.

Niech porównanie pary wariantów zachodzi na komputerze, w którym czas ich porównania wynosi 1 pikosekundę (10^{-12} s). Zgodnie z tabelą 4 w wypadku wektora binarnego x^m czas obliczeń za pomocą metody przeglądu na tym komputerze, wynosi aż $4{,}02\times10^8$ stuleci dla 10 modułów i 10 węzłów. Jednakże powyższa metoda wyznacza rozwiązanie zaledwie w ciągu 0,01 sekundy w wypadku zastosowania wektora X^m.

Metoda przeglądu nie może być zastosowana dla instancji o znacząco większej liczbie modułów, nawet przy wykorzystaniu superkomputerów. Przy wzroście liczby modułów z 10 do 100 potrzeba aż $3{,}17\times10^{80}$ stuleci obliczeń za pomocą tej metody dla „efektywniejszego" wektora X^m. W ciągu godziny obliczeń rozważanego komputera mogą być porównane wszystkie warianty dla 7 modułów i 7 węzłów, przy reprezentacji binarnej x^m, a także dla 13 segmentów programów i 14 węzłów, przy reprezentacji całkowitoliczbowej X^m.

Tabela 4. Liczba możliwych wariantów przydziałów modułów do węzłów w zależności od liczby modułów oraz liczby węzłów

V	Liczba węzłów I							
	2		5		10		100	
	A	B	A	B	A	B	A	B
2	16	4	1024	25	$1{,}049 \times 10^6$	10^2	$1{,}607 \times 10^{60}$	10^4
10	$1{,}049 \times 10^6$	1024	$1{,}126 \times 10^{15}$	$9{,}766 \times 10^6$	$1{,}268 \times 10^{30}$	10^{10}	$1{,}072 \times 10^{301}$	10^{20}
14	$2{,}680 \times 10^8$	$1{,}638 \times 10^4$	$1{,}180 \times 10^{21}$	$6{,}100 \times 10^9$	$1{,}390 \times 10^{42}$	10^{14}	$2{,}767 \times 10^{421}$	10^{28}
10^2	$1{,}607 \times 10^{60}$	$1{,}268 \times 10^{30}$	$3{,}273 \times 10^{150}$	$7{,}889 \times 10^{69}$	$1{,}072 \times 10^{301}$	10^{100}	$1{,}436 \times 10^{3004}$	10^{200}
10^3	$1{,}148 \times 10^{602}$	$1{,}072 \times 10^{301}$	$1{,}486 \times 10^{1502}$	$4{,}223 \times 10^{684}$	$1{,}436 \times 10^{3004}$	10^{1000}	$1{,}325 \times 10^{30005}$	10^{2000}

A - przydział opisany za pomocą binarnego wektora x^m,

B - przydział opisany za pomocą całkowitoliczbowego wektora X^m.

Źródło: opracowanie własne.

Wzrost 1000-krotny mocy obliczeniowej komputera - co zajmuje około 15 lat przy założeniu, że moc obliczeniowa podwaja się co 1,5 roku - powoduje niewielki przyrost rozmiaru rozwiązywanego problemu. Wówczas w wypadku wektora x^m jest

możliwe rozważenie 7 modułów programu i 8 węzłów, a dla wektora X^m – 15 modułów i 15 węzłów. Nawet istotny przyrost mocy obliczeniowej komputera nie gwarantuje zatem wyznaczenia rozwiązania optymalnego za pomocą metody pełnego przeglądu dla obu reprezentacji przydziałów. Przy ustalonej liczbie węzłów oraz liczbie modułów reprezentacja całkowitoliczbowa znacząco redukuje liczbę wariantów.

Zbiór możliwych rodzajów komputerów jest scharakteryzowany za pomocą wektora kosztów $\kappa = [\kappa_1,...,\kappa_j,...,\kappa_J]^T$, gdzie κ_j jest kosztem komputera typu π_j. Jeżeli przyjąć, że w węźle może być usytuowany co najwyżej jeden komputer, to przydział rodzajów maszyn do węzłów określa się za pomocą następującej funkcji.

Definicja 2.2

Funkcją przydziału komputerów do węzłów nazywamy odwzorowanie $\Phi_\pi : \boldsymbol{W} \to \boldsymbol{\Pi}$, takie że $\Phi_\pi(w_i) = \pi_j$ oznacza przydzielenie do węzła w_i komputera typu π_j, przy czym jeżeli do w_i nie przydzielono żadnego komputera, to $\Phi_\pi(w_i) = \pi_0$.

Komputer typu π_0 nazywamy *pozornym* [36]. Do sformułowania zagadnień optymalizacji wygodniejsza jest reprezentacja przydziału typów komputerów do węzłów w postaci binarnego *wektora przydziału typów komputerów do węzłów*, który zapisuje się, jak niżej:

$$x^\pi = [x_{11}^\pi,...,x_{1j}^\pi,...,x_{1J}^\pi,...,x_{ij}^\pi,...,x_{IJ}^\pi]^T, \tag{2.5}$$

gdzie

$$x_{ij}^\pi = \begin{cases} 1 \text{ gdy do } w_i \text{ przydzielono komputer typu } \pi_j \neq \pi_0, \\ 0 \text{ w przeciwnym razie,} \end{cases} \quad \text{dla } i = \overline{1,I},\ j = \overline{1,J}.$$

Uwaga 2.1

Jeżeli $\Phi_\pi(w_i) = \pi_0$, to $x_{ij}^\pi = 0$ dla $j = \overline{1,J}$.

Możliwych jest 2^{IJ} przydziałów komputerów do węzłów, które są reprezentowane za pomocą wektora x^π. Przykładowy przydział typów komputerów do węzłów z rys. 2.1 dla trzech rodzajów komputerów, można zapisać w postaci następującego binarnego wektora $x^\pi = [0,1,0,0,0,1,0,1,0,1,0,0]^T$

Binarny wektor przydziału komputerów do węzłów umożliwia wyznaczenie przydziałów, których nie można określić za pomocą funkcji Φ_π. Są to przydziały, dla

których w węźle można usytuować co najmniej dwa komputery. Powyższe przydziały uwzględnia funkcja rozszerzonych przydziałów do komputerów.

Definicja 2.3

Funkcją rozszerzonego przydziału komputerów do węzłów nazywamy odwzorowanie $\Phi_{\pi E} : \boldsymbol{W} \to 2^{\boldsymbol{\Pi}}$, takie że $\Phi_{\pi E}(w_i) = \Pi_j$ oznacza przydzielenie do węzła w_i komputerów, których rodzaje należą do zbioru $\Pi_i = \{\pi_j \in \boldsymbol{\Pi} \mid x_{ij}^{\pi} = 1 \text{ dla } j = \overline{1, J}\}$.

Binarny wektor przydziałów komputerów do węzłów x^{π} oraz funkcja rozszerzonych przydziałów $\Phi_{\pi E}$ nie uwzględniają sytuacji, w których do węzła mogą być przydzielone co najmniej dwa komputery jednakowego typu. Przydziały tej klasy można określić za pomocą pełnej funkcji przydziałów komputerów do węzłów, przy dodatkowym założeniu, że istnieje nieograniczona liczba komputerów każdego typu.

Niech $\boldsymbol{N}_0 = \boldsymbol{N} \cup \{0\}$, przy czym $\boldsymbol{N}$ – zbiór liczb naturalnych.

Definicja 2.4

Pełną funkcją przydziału typów komputerów do węzłów nazywamy funkcję $\Phi_{\pi F} : W \to \boldsymbol{N}_0^{\,J}$, taką że $\Phi_{\pi F}(w_i) = [x_{i1}^{\pi F}, ... x_{ij}^{\pi F}, ..., x_{iJ}^{\pi F}]^T$ oznacza przydzielenie do węzła w_i po $x_{ij}^{\pi F}$ komputerów typu π_j, gdzie $x_{ij}^{\pi F} \in \boldsymbol{N}_0$.

*Uwaga **2.2***

Jeżeli do węzła w_i nie przydzielono komputera, to $x_{ij}^{\pi F} = 0$ dla $j = \overline{1, J}$.

Wektorem pełnego przydziału komputerów do węzłów nazywamy następujący wektor $x^{\pi F} = [x_{11}^{\pi F}, ... x_{ij}^{\pi F}, ..., x_{IJ}^{\pi F}]^T$, a *macierzą pełnego przydziału typów komputerów do węzłów* – macierz $X^{\pi F} = [x_{ij}^{\pi F}]_{I \times J}$.

W podstawowym modelu przydziału modułów przyjmuje się założenie, że w węźle usytuowany jest tylko jeden komputer, co implikuje następujące ograniczenie:

$$\sum_{j=1}^{J} x_{ij}^{\pi} = 1 \text{ dla } i = \overline{1, I}. \tag{2.6}$$

Ograniczenie (2.6) może być spełnione, jeżeli przydział rodzajów komputerów do węzłów jest przedstawiony w postaci całkowitoliczbowego wektora przydziału komputerów do węzłów, jak niżej:

$$X^{\pi} = [X_1^{\pi}, ..., X_i^{\pi}, ..., X_I^{\pi}]^T, \tag{2.7}$$

gdzie X_i^{π} reprezentuje numer typu komputera przydzielonego do węzła w_i.

Jeżeli $x_{ij}^{\pi} = 1$, to $X_i^{\pi} = j$. Ponadto $1 \le X_i^{\pi} \le J$ dla $i = \overline{1,I}$. Dla przydziału z rysunku 2.1 można zapisać wektor przydziału następująco $X^{\pi} = [2,3,2,1]^T$.

Definicja 2.5

Przydział modułów do węzłów oraz przydział komputerów do węzłów nazywamy *przydziałem modułów do komputerów.*

Binarny wektor przydziału modułów do komputerów oznaczamy następująco:

$$x = [x_{11}^m,...,x_{1i}^m,...,x_{1I}^m,...,x_{vi}^m,...,x_{VI}^m, x_{11}^{\pi},...,x_{1j}^{\pi},...,x_{1J}^{\pi},...,x_{ij}^{\pi},...,x_{IJ}^{\pi}]^T. \qquad (2.8)$$

Przyjmuje się, że przydział dopuszczalny powinien spełniać postulaty alokacji każdego modułu do jednego z węzłów oraz warunki przydziału tylko jednego komputera do węzła.

W niektórych modelach optymalizacji modułów zakłada się, że powinna istnieć co najmniej jedna para modułów przydzielonych do różnych węzłów, co umożliwia odrzucenie lub karanie rozwiązań, w których wszystkie moduły są przydzielone do tego samego węzła [105]. Powyższy warunek zapisano w następujący sposób:

$$\sum_{v=1}^{V}\sum_{\substack{u=1\\u>v}}^{V}\sum_{i_1=1}^{I}\sum_{\substack{i_2=1\\i_1\neq i_2}}^{I} x_{vi_1}^m x_{vi_2}^m \ge 1. \qquad (2.9)$$

Zbiór przydziałów dopuszczalnych $\boldsymbol{X}$ jest określony w następujący sposób:

$$\boldsymbol{X} = \{x \in \boldsymbol{B}^{I(V+J)} |\ \sum_{i=1}^{I} x_{vi}^m = 1 \text{ dla } v = \overline{1,V},\ \sum_{j=1}^{J} x_{ij}^{\pi} = 1 \text{ dla } i = \overline{1,I}\ \}, \qquad (2.10)$$

gdzie $\boldsymbol{B} = \{0,1\}$

Twierdzenie 2.1 [20]

Jeżeli $V \ge 2$ oraz $J \ge 1$, to zbiór rozwiązań dopuszczalnych $\boldsymbol{X}$ określony za pomocą zależności (2.10), nie jest zbiorem pustym i zawiera $I^V J^I$ przydziałów modułów do komputerów.

Liczbę wariantów przydziałów modułów do komputerów dla binarnego wektora alokacji oraz całkowitoliczbowego wektora przydziału wyznaczono w [20], gdzie przedstawiono także przykładowy zbiór rozwiązań dopuszczalnych $\boldsymbol{X}$.

Definicja 2.6

Niech I=2. Przydział dopuszczalny modułów do komputerów $s \in \boldsymbol{X}$

$$s=[x_{12}^m, x_{11}^m,...,x_{v2}^m, x_{v1}^m,...,x_{V2}^m, x_{V1}^m, x_{11}^\pi,...,x_{1j_2}^\pi,...,x_{2j_1}^\pi,...,x_{2J}^\pi]^T,\ x_{1j_2}^\pi=1,\ x_{2j_1}^\pi=1 \tag{2.11}$$

nazywamy *przydziałem symetrycznym do przydziału dopuszczalnego* $x \in \boldsymbol{X}$, gdzie

$$x=[x_{11}^m, x_{12}^m,...,x_{v1}^m, x_{v2}^m,...,x_{V1}^m, x_{V2}^m, x_{11}^\pi,...,x_{1j_1}^\pi,...,x_{2j_2}^\pi,...,x_{2J}^\pi]^T,\ x_{1j_1}^\pi=1,\ x_{2j_2}^\pi=1.$$

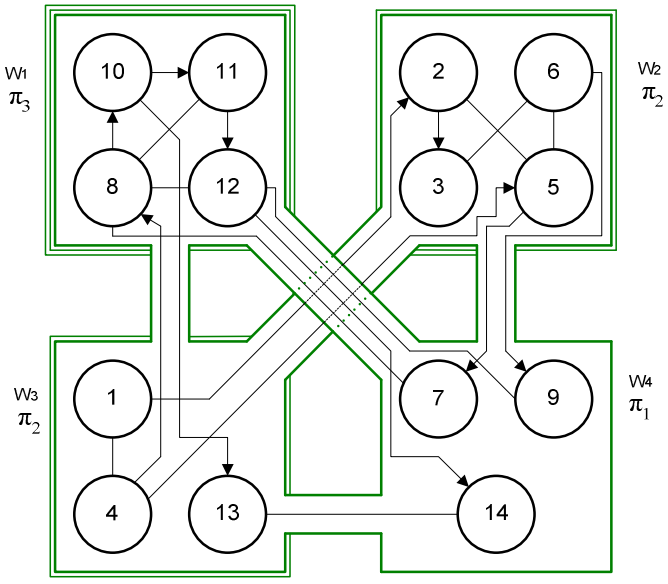
W rozwiązaniu *symetrycznym* *s* do przydziału *x* moduły oraz komputer przydzielone w wariancie *x* do węzła nr 1 są zaalokowane do węzła nr 2, a moduły oraz komputer przydzielone w wariancie *x* do węzła nr 2 znajdują się w węźle nr 1. Jeżeli alternatywa dopuszczalna *s* jest symetryczna do rozwiązania dopuszczalnego *x*, to przydział *x* jest symetryczny do wariantu *s*.

Ograniczenia $\sum_{i=1}^{I} x_{vi}^m = 1,\ v=\overline{1,V}$ oraz $\sum_{j=1}^{J} x_{ij}^\pi = 1,\ i=\overline{1,I}$ stanowią układ $V+I$ równań. Każde z rozważanych równań można przedstawić w postaci ogólnej, jak niżej:

$$\sum_{m=1}^{M} x_m = k, \tag{2.12}$$

gdzie: $k \le M$, $k=\overline{0,M}$, $x_m \in \{0,1\}$, $m=\overline{1,M}$.

Powyższe równanie nazywane jest *podstawowym ograniczeniem przydziału*, gdyż reprezentuje ograniczenia $\sum_{i=1}^{I} x_{vi}^m = 1,\ v=\overline{1,V}$, $\sum_{j=1}^{J} x_{ij}^\pi = 1,\ i=\overline{1,I}$, które pojawiają się w problemach optymalizacji przydziału modułów. Na rysunku 2.7 przedstawiono możliwe rozwiązanie symetryczne do przydziału modułów z rysunku 2.1.



Rys. 2.7. Przydział symetryczny do przydziału z rysunku 2.1
Źródło: opracowanie własne.

Podstawowe ograniczenie przydziału występuje w szeregu dyskutowanych problemach optymalizacji kombinatorycznej [28, 63], takich jak problem komiwojażera [56, 64, 121] czy zagadnienie przydziału układów elektronicznych do slotów [1].

2.4. Kryteria oceny jakości przydziałów

Postać funkcji czasu wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym zależy od założenia, czy są ustalone typy komputerów w węzłach, czy też należy wyznaczyć przydział komputerów.

Definicja 2.7

Funkcję $F_1 : \boldsymbol{B}^M \to \boldsymbol{R}$, gdzie $F_1(x)$ wyznacza się za pomocą zależności:

$$F_1(x) = \sum_{j=1}^{J} \sum_{v=1}^{V} \sum_{i=1}^{2} t_{vj} x_{vi}^m x_{ij}^\pi + \sum_{v=1}^{V} \sum_{u=1}^{V} \sum_{i=1}^{2} \tau_{vu} x_{vi}^m x_{u,3-i}^m, \quad x \in \boldsymbol{B}^M, \tag{2.13}$$

nazywamy funkcją modelu *podstawowego*.

Twierdzenie 2.2 [20]

Jeżeli zadany jest przydział modułów programów do węzłów spełniający ograniczenie (2.4) oraz przydział komputerów do węzłów spełniający ograniczenie (2.6), to wartość funkcji F_1 modelu podstawowego (2.13) jest równa szacowanemu czasowi wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym.

Wartości liczbowe funkcji F_1 można również interpretować jako koszt wykonania rozproszonego programu sekwencyjnego przy założeniu, że t_{vj} oznacza szacowany koszt realizacji modułu m_v na komputerze π_j, a τ_{vu} – szacowany koszt przesyłania danych do modułu m_u od modułu m_v. Za pomocą zależności (2.13) można także wyznaczyć koszt użytkowania rozproszonego programu współbieżnego w systemie dwukomputerowym, przy czym τ_{vu} reprezentuje szacowany koszt interakcji zachodzących między modułami m_v oraz m_u.

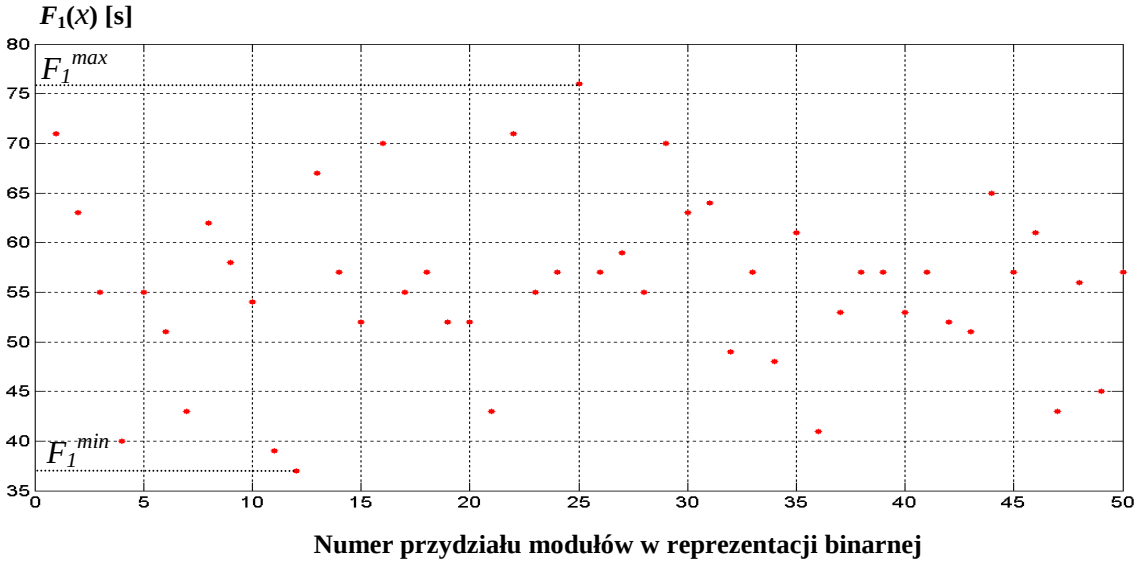
Na rysunku 2.8 przedstawiono zobrazowanie wartości czasów wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym dla 50 losowo wygenerowanych przydziałów modułów do komputerów. Przyjęto, że program składa się z czternastu modułów, a macierz czasów przetwarzania modułów dla dwóch rodzajów komputerów ma postać, jak niżej:

$$\boldsymbol{T} = \begin{bmatrix} 3 & 2 & 1 & 3 & 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 3 & 1 \end{bmatrix}^{\mathrm{T}} \text{ [s]}$$

Założono, że czasy komunikacji między parami modułów przyjmują następujące wartości wyrażone w sekundach: $\tau_{12}=\tau_{23}=\tau_{45}=\tau_{48}=\tau_{56}=\tau_{78}=\tau_{10,11}=\tau_{11,12}=\tau_{13,14}=1$, $\tau_{14}=\tau_{25}=\tau_{36}=\tau_{57}=\tau_{69}=\tau_{8,10}=\tau_{8,11}=\tau_{8,12}=\tau_{9,12}=2$ oraz $\tau_{10,13}=\tau_{12,14}=3$. Pozostałe wartości macierzy komunikacji przyjmują wartości równe zeru, co oznacza, że na rysunku 2.1a) między odpowiednimi parami wierzchołków nie występują łuki lub krawędzie. Do węzła może być przydzielony jeden z dwu typów komputerów. Należy zauważyć, że optymalny przydział (wariant nr 12) pozwala na ponad dwukrotnie krótsze wykonanie programu niż przy realizacji programu z „najwolniejszym” przydziałem modułów do komputerów (wariant nr 25).

Na rysunku 2.9 zobrazowano wartości czasów wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym dla przypadku, gdy $J=V=4$. Założono, że macierz czasów przetwarzania modułów ma postać, jak poniżej:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 2 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 1 & 1 & 2 & 3 \end{bmatrix} \text{ [s]}$$



Rys. 2.8. Wartości czasu wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym dla pięćdziesięciu wybranych przydziałów modułów
Źródło: opracowanie własne.

Przyjęto, że czasy komunikacji między parami modułów przyjmują następujące wartości wyrażone w sekundach: $\tau_{14}= \tau_{23}=1$, $\tau_{12}= \tau_{34}=2$, $\tau_{13}= \tau_{24}=3$. Należy zauważyć, że optymalny przydział pozwala na znacząco krótsze wykonanie programu niż dla przypadku z „najgorszym" przydziałem.

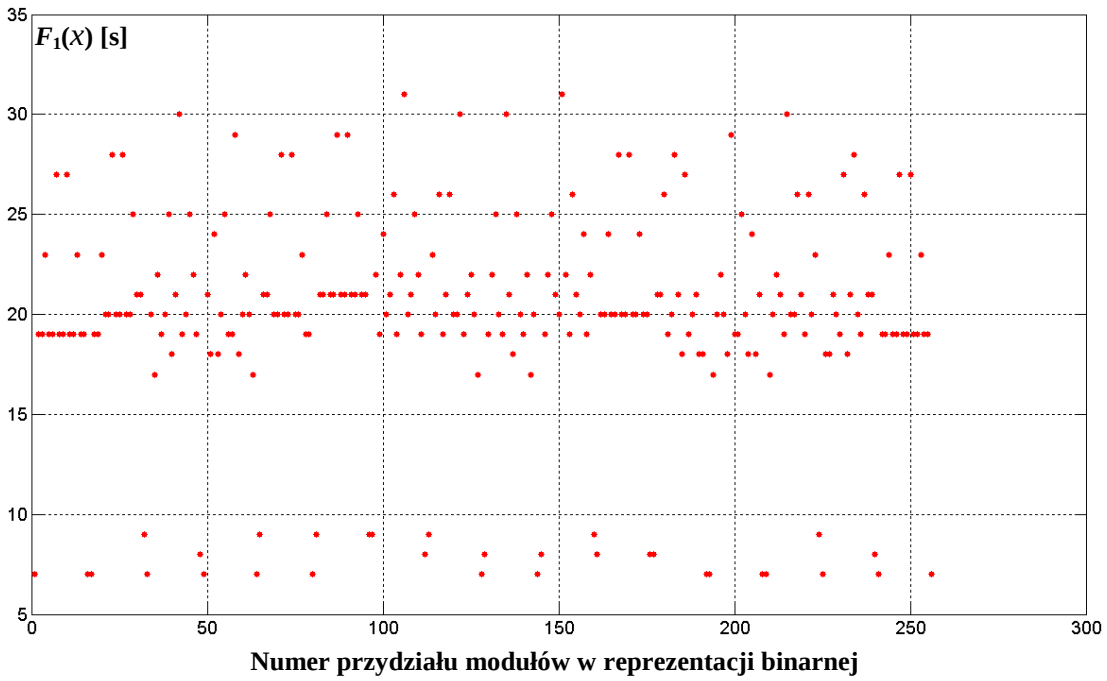
Ważnym kryterium oceny przydziału typów komputerów do węzłów jest także łączny koszt przydzielonych komputerów, który jest wyznaczony za pomocą następującej zależności:

$$F_2(x^{\pi})=\sum_{i=1}^{I}\sum_{j=1}^{J}\kappa_j x_{ij}^{\pi}. \tag{2.14}$$

Jeżeli do pary węzłów (w_1, w_2) zostaną przydzielone dwa rodzaje komputerów (π_{j_1},π_{j_2}), $j_1, j_2 \in \overline{1,J}$, to nałożone warunki na przydział pary komputerów można zapisać następująco:

$$x_{ij_1}^{\pi}=1,\ x_{ij_2}^{\pi}=1,\ x_{ij}^{\pi}=0 \ \text{ dla } \ i=\overline{1,2},\ j=\overline{1,J}\setminus\{j_1, j_2\}. \tag{2.15}$$

Niech wartości funkcji F_1 odpowiadają szacowanemu czasowi wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym, a wartości funkcji F_2 łącznemu kosztowi przydzielonych komputerów.



Rys. 2.9. Wartości czasów wykonania rozproszonego programu sekwencyjnego w systemie dwukomputerowym
Źródło: opracowanie własne.

Twierdzenie 2.3 [20]

Jeżeli s jest rozwiązaniem symetrycznym do rozwiązania dopuszczalnego $x \in \boldsymbol{X}$, to

$$\begin{aligned} F_1(x) &= F_1(s), \\ F_2(x) &= F_2(s). \end{aligned} \tag{2.16}$$

W modelu z zadanym przydziałem pary komputerów wartość czasu wykonania sekwencyjnego programu rozproszonego $F_1^{'}$ zależy jedynie od przydziału modułów do węzłów x^m. Wartość $F_1^{'}(x^m)$ może być wyznaczona za pomocą poniższej zależności:

$$F_1^{'}(x^m) = \sum_{v=1}^{V} \sum_{i=1}^{2} t_{vi}^{'} x_{vi}^m + \sum_{v=1}^{V} \sum_{u=1}^{V} \sum_{i=1}^{2} \tau_{vu} x_{vi}^m x_{u,3-i}^m , \tag{2.17}$$

gdzie $t_{vi}^{'}$ oznacza szacowany czas wykonania modułu m_v na komputerze π_{j_i} w węźle w_i, $v = \overline{1,V}$, $i = \overline{1,2}$.

Macierz czasów realizacji modułów na komputerach $\boldsymbol{T} = [t_{vj}]_{V \times J}$ redukuje się do macierzy dwukolumnowej $\boldsymbol{T}^{'} = [t_{vi}^{'}]_{V \times 2}$, która zawiera kolumnę nr j_1 oraz kolumnę nr j_2 z macierzy $\boldsymbol{T}$.

Problem minimalizacji czasu wykonania rozproszonego programu sekwencyjnego, którego moduły należy przydzielić do dwóch zadanych komputerów, można sformułować, jak niżej.

Dla danych: $\boldsymbol{T}^{'} = [t_{vi}^{'}]_{V \times 2}$, $\tau = [\tau_{vu}]_{V \times V}$ należy wyznaczyć x^{m*}, takie że

$$F_1^{'}(x^{m*}) = \min_{x' \in \boldsymbol{X}^{'}} F_1^{'}(x^m), \tag{2.18}$$

gdzie:

$\boldsymbol{X}^{'} = \{ x^m \in \boldsymbol{B}^{2V} \mid x^m = [x_{11}^m, ..., x_{vi}^m, ..., x_{V2}^m]^T , \sum_{i=1}^{2} x_{vi}^m = 1 \text{ dla v} = \overline{1,\text{V}} \}$.

Zadanie (2.18) należy do klasy problemów optymalizacji kombinatorycznej [78, 95, 101]. Możliwych jest 2^{2V} wariantów przydziałów modułów do zadanej pary komputerów. Do rozwiązania problemu (2.18) zastosowanie metody pełnego przeglądu jest nieefektywne, gdyż liczba wariantów rośnie wykładniczo wraz ze wzrostem liczby przydzielonych modułów V. Omawiane zagadnienie należy jednak do klasy problemów P-trudnych [29, 121], gdyż można pokazać, że daje się rozwiązać za pomocą wielomianowej metody Stone'a [143], w której stosuje się algorytm maksymalnego przepływu [62, 96].

W systemach, w których liczba komputerów jest większa od dwóch, stosuje się jako miarę jakości przydziału *łączne obciążenie komputerów* nazywane także *kosztem obliczeniowym*. Koszt wykonania programu rozproszonego w systemie wielokomputerowym jest określony w następujący sposób [20]:

$$K'(x) = \sum_{j=1}^{J}\sum_{v=1}^{V}\sum_{i=1}^{I} \bar{t}_{vj} x_{vi}^{m} x_{ij}^{\pi} + \sum_{v=1}^{V}\sum_{u=1}^{V}\sum_{i_1=1}^{I}\sum_{\substack{i_2=1 \\ i_2 \neq i_1}}^{I} \bar{\tau}_{vu} x_{vi_1}^{m} x_{ui_2}^{m}. \tag{2.19}$$

Dla zależności (2.19) przyjęto, że koszt komunikacji między modułami nie zależy od miejsca alokacji modułów. Takie założenie jest słuszne w systemach zamkniętych o niewielkiej liczbie węzłów, w których to systemach przepustowości wirtualnych połączeń komunikacyjnych między parami węzłów są równe.

W systemach otwartych, które cechują się zazwyczaj dużą liczbą węzłów, koszt komunikacji między modułami zależy od miejsca alokacji modułów. Wraz ze wzrostem liczby węzłów tranzytowych rośnie opóźnienie przesyłanego komunikatu między modułami. W tej sytuacji nie występują przydziały symetryczne. Istotny wpływ na wielkość opóźnienia wywierają także przepustowości kanałów komunikacyjnych, przez które przesyłany jest komunikat. Zatem przy tej samej wielkości przesyłanych danych, czas transmisji komunikatu zależy wówczas od pary węzłów, do których przydzielono moduły. Przyjmując, że koszt transmisji jest proporcjonalny do czasu transmisji, dostaje się następujące wyrażenie na koszt wykonania programu rozproszonego:

$$K(x) = \sum_{j=1}^{J}\sum_{v=1}^{V}\sum_{i=1}^{I} \bar{t}_{vj} x_{vi}^{m} x_{ij}^{\pi} + \sum_{v=1}^{V}\sum_{u=1}^{V}\sum_{i_1=1}^{I}\sum_{\substack{i_2=1 \\ i_2 \neq i_1}}^{I} \tilde{\tau}_{vui_1i_2} x_{vi_1}^{m} x_{ui_2}^{m}, \tag{2.20}$$

gdzie $\tilde{\tau}_{vui_1i_2}$ oznacza koszt przesyłania danych od *v*-tego modułu usytuowanego w węźle nr i_1 do *u*-tego modułu usytuowanego w węźle nr i_2 [JM – jednostka monetarna].

Koszt komunikacji między modułami nazywamy *niezależnym od pary węzłów*, do których przydzielono moduły, jeżeli zachodzi następująca relacja:

$$\bar{\tau}_{vu} = \tilde{\tau}_{vui_1i_2} \text{ dla } i_1, i_2 = \overline{1,I},\ v,u = \overline{1,V}. \tag{2.21}$$

Ważnym kryterium oceny przydziałów modułów programistycznych jest *łączna moc obliczeniowa* rozproszonego systemu komputerowego:

$$\tilde{F}_2(x) = \sum_{i=1}^{I}\sum_{j=1}^{J} \vartheta_j x_{ij}^{\pi}, \tag{2.22}$$

gdzie ϑ_j oznacza wydajność komputera typu π_j [MFlops].

Wydajność komputerów określonych klas może być wyznaczona za pomocą testów wydajności (*benchmarków*). Zestawienie wydajności najpopularniejszych komputerów wg wybranych testów zamieszczono w [191]. W rozproszonym systemie komputerowym wydajność komputerów powinna być wyznaczona dla przyjętego reprezentatywnego testu w odniesieniu do realizowanych zadań. Wydajność realizacji transakcji systemu nie powinna być mniejsza niż założone przez projektanta minimum wydajności ϑ_{min}, co zapisujemy jako kolejne ograniczenie na przydział dopuszczalny:

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\vartheta_j\, x_{ij}^{\pi} \geq \vartheta_{min}\,, \tag{2.23}$$

Jeżeli projektant systemu nie jest zadowolony z wartości kosztu wykonania programu, to może w projekcie wymienić komputery przydzielone do węzłów, aby obniżyć koszty przetwarzania danych. Istnieje jednak konflikt między obniżeniem kosztu wykonania programu a wzrostem mocy obliczeniowej RSK. Im intensywniej proces jest realizowany, tym na ogół wyższy jest koszt eksploatowanego komputera.

Jeśli koszt komunikacji między modułami nie zależy od węzłów ich alokacji, to można wykazać, że dla każdego rozwiązania dopuszczalnego *x* istnieje *I*!–1 permutacji tego przydziału o ocenach równych $[K(x), \widetilde{F}_2(x)]$. Te permutacje tworzy się z *x* przez zmianę alokacji komputerów. Jeżeli komputer jest przydzielony do *i*-tego węzła, to bez zmiany wartości mocy systemu oraz kosztu realizacji modułów można „przesunąć" ten komputer do innego węzła wraz z przydzielonymi do tego komputera modułami. Takie przydziały nazywamy *przesuniętymi*.

Dla dwóch komputerów rozwiązanie przesunięte jest alternatywą symetryczną. Liczba przydziałów przesuniętych względem *x* znacząco maleje w stosunku do liczności zbioru przydziałów dopuszczalnych wraz ze wzrostem liczby węzłów, modułów oraz typów komputerów. Dla $I = 3$ *współczynnik gęstości przydziałów przesuniętych* względem *x* wynosi 0,82% , dla $I = 4$ – 0,037% , a dla $I = 5$ – 0,0012% , przy czym przyjęto, że $I = V$ oraz $I = J$ [20].

Zapewnienie równomiernego obciążenia komputerów należy do istotnych postulatów dotyczących projektowania przydziałów modułów programistycznych [16]. Jeżeli przydział modułów minimalizuje koszt przetwarzania danych, to może wystąpić sytuacja skupienia modułów na komputerze, dla którego wartość funkcji kosztów będzie najmniejsza. Ten komputer będzie jednak najbardziej obciążony pod względem liczby przydzielonych modułów, a pozostałe komputery mogą być nieobciążone [119].

Nierównomierność obciążenia poszczególnych komputerów może zatem wystąpić w przydziałach minimalizujących wartość funkcji kosztu przetwarzania rozproszonego. Nierównomierność obciążenia komputerów może także charakteryzować przydziały minimalizujące wartość funkcji czasu sekwencyjnego przetwarzania danych.

W wypadku komputerów różniących się pod względem specyfiki przetwarzania danych na tyle, że macierz kosztów wykonywania modułów na komputerach cechuje się istnieniem minimalnych elementów w wierszach dla różnych kolumn, może wystąpić rozproszenie modułów w przydziale minimalizującym koszt wykonania programu. Jednakże ze zbioru dostępnych komputerów dwa lub trzy mogą być najbardziej obciążone pod względem liczby przydzielonych modułów, a pozostałe – w znacznie mniejszym stopniu. Zatem istotne jest wyznaczanie przydziałów modułów do komputerów, w których moduły są równomiernie rozmieszczone.

Stosowaną miarą obciążenia systemu jest suma czasu przetwarzania danych i czasu zewnętrznej komunikacji dla najbardziej eksploatowanego komputera, które to kryterium oznaczono jako $Z_{\max}$. W wyniku przydzielenia modułów do i-tego węzła komputer jest obciążony realizacją tych modułów w czasie $\hat{Z}_i(x)$, który nazywamy czasem użytkowania komputera przydzielonego do i-tego węzła. Wartość czasu $\hat{Z}_i(x)$ wyznacza się w następujący sposób:

$$\hat{Z}_i(x) = \sum_{j=1}^{J} \sum_{v=1}^{V} t_{vj} x_{vi}^{m} x_{ij}^{\pi}, \; i = \overline{1, I}. \tag{2.24}$$

Istotnym obciążeniem podsystemu komunikacyjnego i-tego węzła jest wysyłanie i odbiór danych w celu zapewnienia poprawnej realizacji modułów na komputerze. Łączny czas wysyłania i odbioru danych zależy od objętości przesyłanych danych, co powoduje, że może być traktowany jako miara obciążenia podsystemu komunikacyjnego.

Czas $\tilde{Z}_i(x)$ nadawania i odbioru danych w i-tym węźle wyznacza się, jak niżej:

$$\tilde{Z}_i(x) = \sum_{v=1}^{V} \sum_{\substack{u=1 \\ u \neq v}}^{V} \sum_{\substack{i_2=1 \\ i_2 \neq i}}^{I} \tau_{vuii_2} x_{vi}^{m} x_{ui_2}^{m}, \;\; i = \overline{1, I}.$$

Obciążeniem i-tego węzła nazywamy sumę czasu użytkowania komputera przydzielonego do i-tego węzła oraz czasu nadawania i odbioru danych w i-tym węźle. Obciążenie i-tego węzła $Z_i^+(x)$ dla zadanego przydziału x wyznaczamy następująco:

$$Z_i^+(x) = \rho_1 \sum_{j=1}^{J} \sum_{v=1}^{V} t_{vj} x_{vi}^m x_{ij}^\pi + \rho_2 \sum_{v=1}^{V} \sum_{\substack{u=1 \\ u \neq v}}^{V} \sum_{\substack{i_2=1 \\ i_2 \neq i}}^{I} \tau_{vuii_2} x_{vi}^m x_{ui_2}^m, \quad i = \overline{1, I}. \tag{2.25}$$

gdzie

ρ_1 - waga dla obciążenia podsystemu przetwarzania,

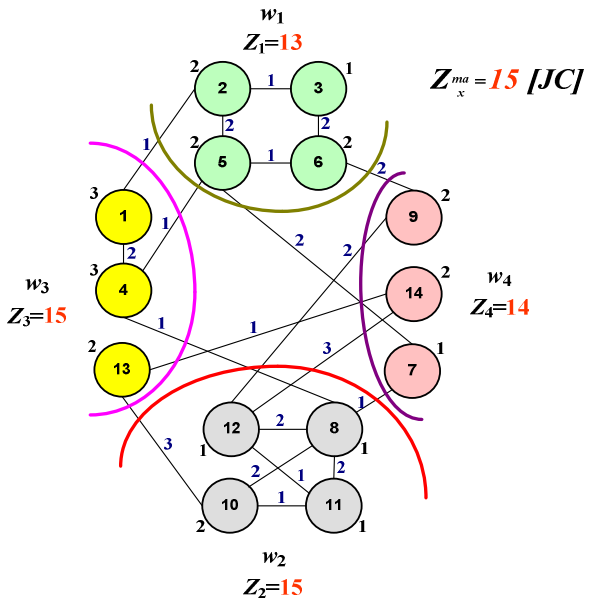
ρ_2 - waga dla obciążenia podsystemu komunikacyjnego, $0 \leq \rho_1, \rho_2 \leq 1$.

Węzeł o największym obciążeniu $Z_i^+(x)$ stanowi „wąskie gardło" systemu [156, 163]. Im mniejsze jest jego obciążenie, tym większe zdecentralizowanie modułów. Ponadto mniej obciążony węzeł może generować mniejszy ruch danych w sieci komputerowej.

Maksymalne obciążenie węzła dla zadanego przydziału modułów do komputerów x wyraża się następującą formułą:

$$Z_{max}(x) = \max_{i \in \overline{1,I}} \left\{ Z_i^+(x) \right\} \tag{2.26}$$

Na rysunku 2.10 przedstawiono obciążenia węzłów dla przydziału modułów z rysunku 2.1. Założono, że do węzłów przydzielono komputery tego samego typu.



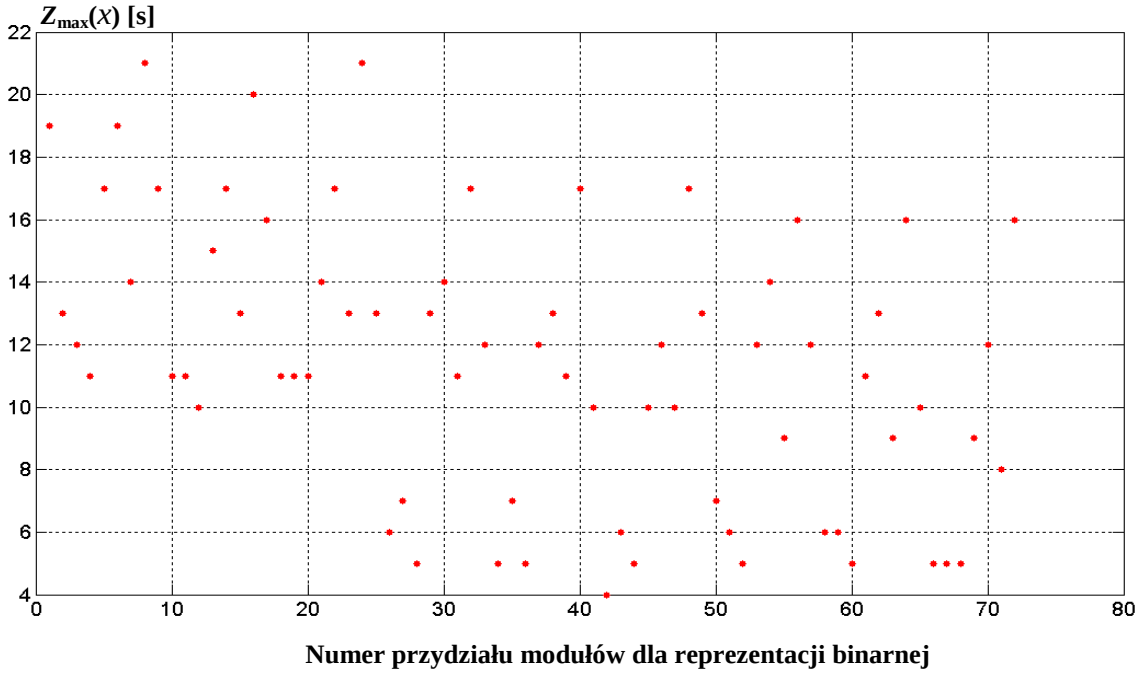
Rys. 2.10. Obciążenia węzłów dla przydziału czternastu modułów programistycznych minimalizującego obciążenie newralgicznego komputera
Źródło: opracowanie własne.

Jeżeli moduły będą przesunięte z komputera w najbardziej użytkowanym węźle na komputery nieobciążone lub obciążone w znacznie mniejszym stopniu, to będzie zmniejszone obciążenie newralgicznego komputera, co skutkuje wyrównywaniem obciążenia wszystkich węzłów w systemie.

Na rysunku 2.11 zobrazowano obciążenie newralgicznego komputera dla przypadku, gdy J=V=3 oraz I=2. Założono, że macierz czasów przetwarzania modułów ma postać, jak poniżej:

$$\boldsymbol{T} = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} [\mathrm{s}]$$

Przyjęto, że czasy komunikacji między parami modułów przyjmują następujące wartości wyrażone w sekundach: $\tau_{23}=1$, $\tau_{12}=2$, $\tau_{13}=3$. Wagi dla obciążenia podsystemu przetwarzania i komunikacyjnego wynoszą odpowiednio $\rho_1 = \rho_2 = 1$.



Rys. 2.11. Obciążenie newralgicznego komputera dla przypadku, gdy I=V=3 oraz J=2
Źródło: opracowanie własne.

Postulat równoważenia obciążenia w węzłach może być uwzględniony poprzez wprowadzenie dodatkowego ograniczenia, takiego aby $Z_{max}(x) \le T_{gr}$, gdzie T_{gr} reprezentuje założone największe dopuszczalne obciążenie węzła w systemie.

Ograniczenia mogą dotyczyć również wielkości pamięci komputerów. Niech dostępne będą zasoby lokalne $z_1,...,z_r,...,z_R$, a $\tilde{d}_{jr}$ oznacza wielkość zasobu z_r w komputerze *j*-tego typu [JZ_r – jednostka miary *r* - tego zasobu]. Przyjmuje się, że zadanie m_v rezerwuje c_{vr} jednostek zasobu z_r [JZ_r], $c_{vr} \geq 0$. Zakłada się także, że zapotrzebowanie na zasób jest stałe podczas realizacji zadań. W przydzielonym komputerze powinna być dostępna wystarczająca wielkość zasobów *addytywnych*, czyli takich, dla których łączne zapotrzebowanie przez pewien podzbiór modułów jest sumą zapotrzebowań przez poszczególne moduły:

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \leq \sum_{j=1}^{J} \tilde{d}_{jr} x_{ij}^{\pi}, \quad i = \overline{1,I}, \ r = \overline{1,R}. \tag{2.27}$$

Zasobem addytywnym może być pamięć operacyjna, pamięć zewnętrzna, czas pracy procesora lub przepustowość układów I/O [48].

Ponadto stosowane są mogą być kryteria, takie jak dostępność systemu lub miara skutecznej realizacji modułów w terminach [15].

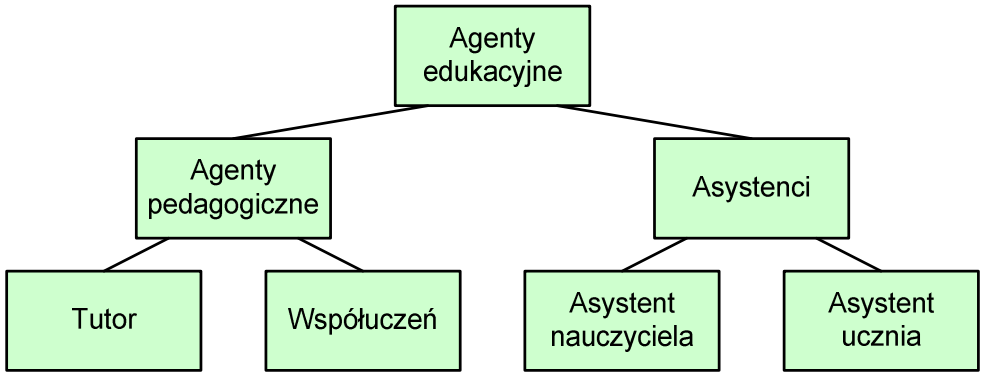
2.5. Systemy wieloagentowe związane z platformą MOODLE

Wykorzystanie inteligentnych agentów jest nowością w systemach nauczania klasy *MOODLE*. Agent edukacyjny jest to autonomiczny i cechujący się sztuczną inteligencją program komputerowy, którego zadaniem jest pomoc studentom i nauczycielom w osiąganiu celów kształcenia. Cechą charakterystyczną agenta jest działanie w imieniu wykładowcy, a także aktywność i samodzielność w wykonywaniu powierzonych zadań.

W [85, 129] jako inteligentny agent uważane jest oprogramowanie, które może być zdolne do naśladowania racjonalnych zachowań z wykorzystaniem sensorów w środowisku rozproszonym. Inteligentny moduł powinien autonomicznie reagować na zmiany w środowisku, zamierzając osiągnąć cel, do którego został zaprojektowany. W węższym sensie agenty tej klasy rozumiane są jako autonomiczne obiekty programowe z kognitywistycznymi modelami, które zarządzają zdarzeniami w eksplorowanym środowisku [100]. Klasyfikację tej klasy agentów przedstawiono na rysunku 2.12.

Pomoc słuchaczowi może obejmować czynności, takie jak poszukiwanie materiałów, a także udostępnianie zasobów i programów edukacyjnych. Zaawansowane

agenty przejmują na siebie częściowo rolę nauczycieli. W tych rozwiązaniach zazwyczaj komunikacja z agentem odbywa się za pomocą języka naturalnego.



Rys. 2. 12. Rodzaje agentów edukacyjnych [99]

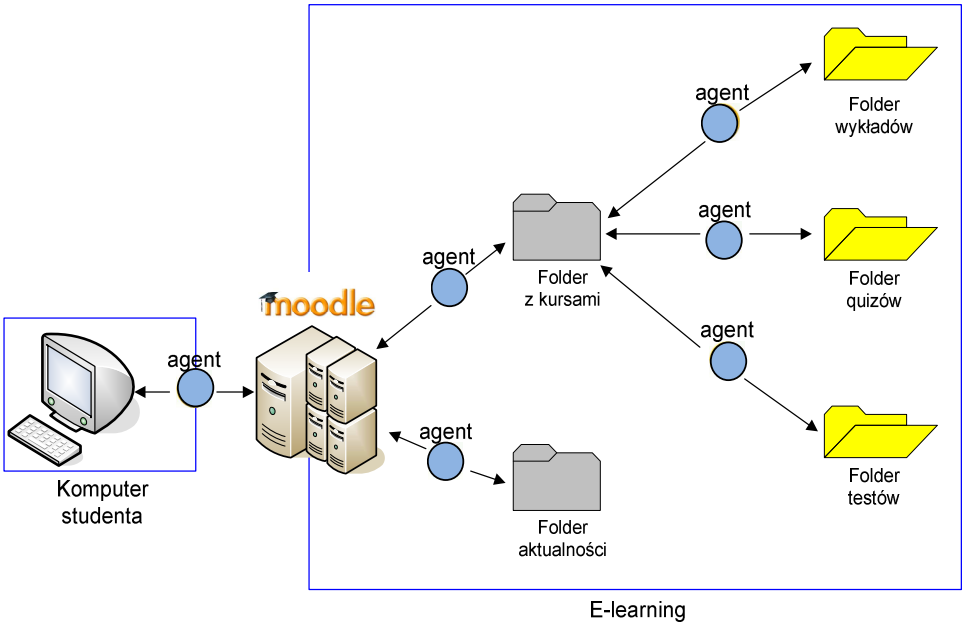
Agent edukacyjny potrafi odpowiadać na pytania, oceniać wypowiedzi ucznia czy podpowiadać. Implementowane są algorytmy sztucznej inteligencji, które stosuje się do: analizy języka naturalnego, tworzenia modelu kompetencji słuchacza oraz oceniania słuchaczy [99].

W e-learningu zazwyczaj zezwala się studentom na pobieranie materiałów kursowych na komputery i sprawdzanie aktualności dotyczących kursów. Prowadzi to do częstego odwiedzania witryny z kursem. W [100] przedstawiono rozwiązanie, w którym działanie agentów oparte jest na mechanizmach nazwanych *push and pull*. Wykorzystuje się kilka rodzajów agentów odpowiedzialnych za przeszukiwanie, pobieranie i informowanie o nowościach w kursie. Najnowsza wersja *MOODLE* nie zawiera jeszcze opcji automatycznego pobierania materiałów lub powiadamiania.

Na rysunku 2.13 przedstawiono system wieloagentowy w środowisku *MOODLE*. Na komputerze studenta instalowane jest oprogramowanie, pozwalające na uruchomienie agentów. Zachowanie agentów jest określone preferencjami studentów.

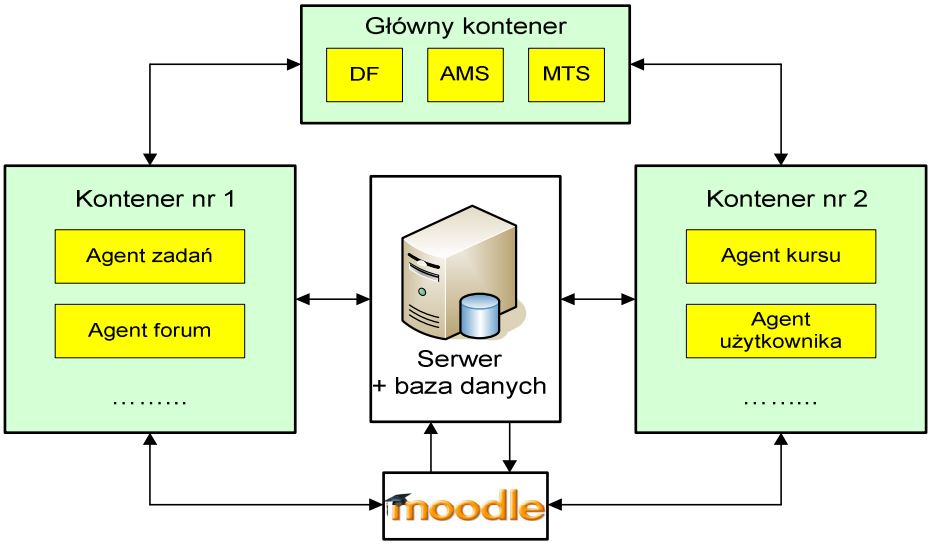
Wychodząc naprzeciw problemowi braku personalizacji w zdalnym nauczaniu, zintegrowano platformę JADE (ang. *Java Agent Development Framework*) z systemem *MOODLE*, tworząc spersonalizowaną wirtualną platformę nauczania, w której skład wchodzi system wieloagentowy wykorzystujący algorytm drzewa decyzyjnego [87].

Architektura systemu, przedstawiona na rysunku 2.14 składa się z platformy *MOODLE*, bazy danych z serwerem oraz platformy wieloagentowej JADE, zawierającej trzy kontenery.



Rys. 2. 13. System wieloagentowy w środowisku *MOODLE* [87]

W kontenerach nr 1 i 2 aktywność agentów jest ograniczona i może mieć miejsce w JADE w tym samym czasie, przy czym kontener główny jest niezbędny do działania systemu. W jego skład wchodzi agent DF (ang. *Directory Facilitator*) odpowiedzialny za dostarczanie aktualnej informacji o agentach.



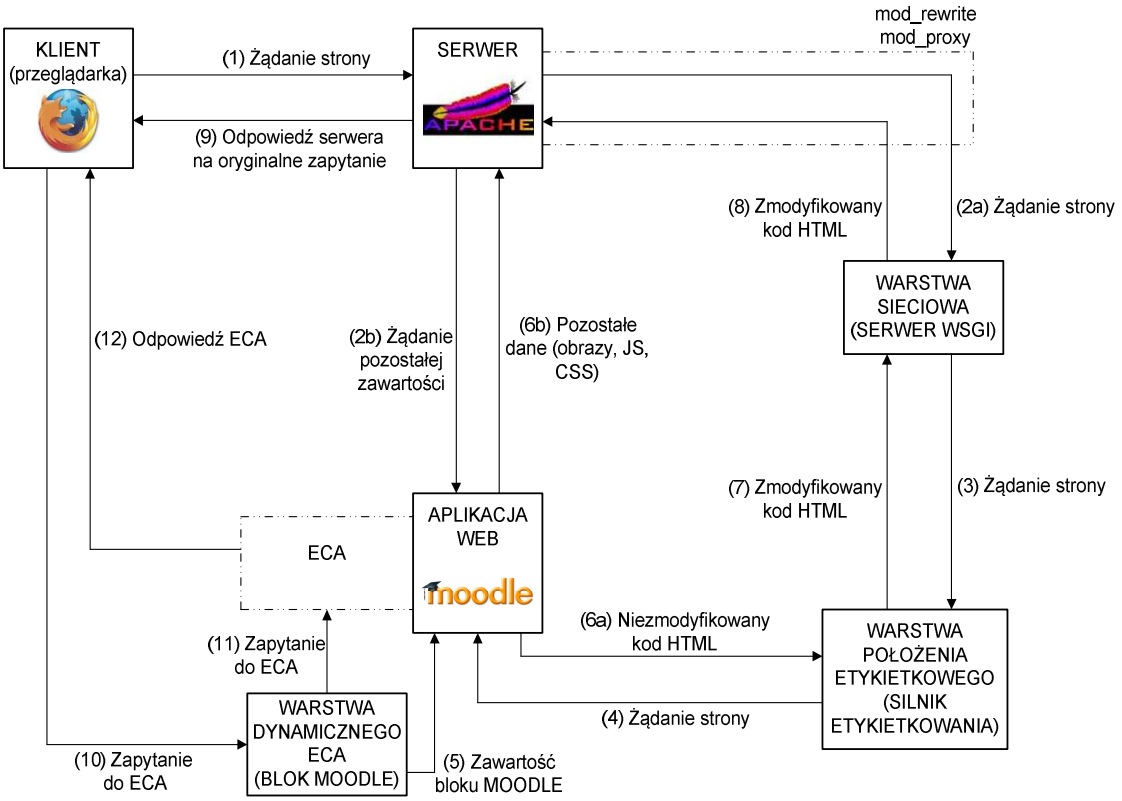
Rys. 2.14. Integracja systemu *MOODLE* z systemem JADE [87]

Agent AMS (ang. *Agent Management System*) zarządza operacjami na platformie agentów. Może wymusić wykonanie przez agenta określonego zadania, a w przypadku zignorowania żądania sam może przeprowadzić odpowiednią operację.

Agent MTS (ang. *Message Transport Service*) realizuje usługę pozwalającą na wymianę komunikatów między agentami na platformie lub między agentami z różnych platform.

Interesującym rozwiązaniem jest zastosowanie agenta ECA (ang. *Embodied Conversational Agent*) do poprawy jakości interakcji między portalem e-learningowym a użytkownikiem. Agent ECA pełni funkcję spersonifikowanego asystenta, zwiększając możliwości platformy [35]. Na rysunku 2.15 przedstawiono architekturę, w której wykorzystuje się agenta ECA do ułatwienia dostępu do informacji z platformy *MOODLE*.

Architektura składa się z trzech warstw i jest przezroczysta dla użytkownika. Warstwa sieciowa opiera się na architekturze klasy SOA (ang. *Server Oriented Architecture*). SOA bazuje na warstwie pośredniej, która zasadniczo jest warstwą komunikacyjną, pozwalającą aplikacjom na współpracę w heterogenicznych systemach operacyjnych i środowiskach sieciowych. Standard WSGI (ang. *Web Server Gateway Interface*) jest specyfikacją interfejsu między serwerami webowymi a aplikacjami webowymi. Modularna struktura serwera Apache zapewnia połączenie między serwerem webowym a warstwą za pomocą przekierowywania do niej żądań.



Rys. 2.15. Wykorzystanie agenta ECA w systemie *MOODLE* [35]

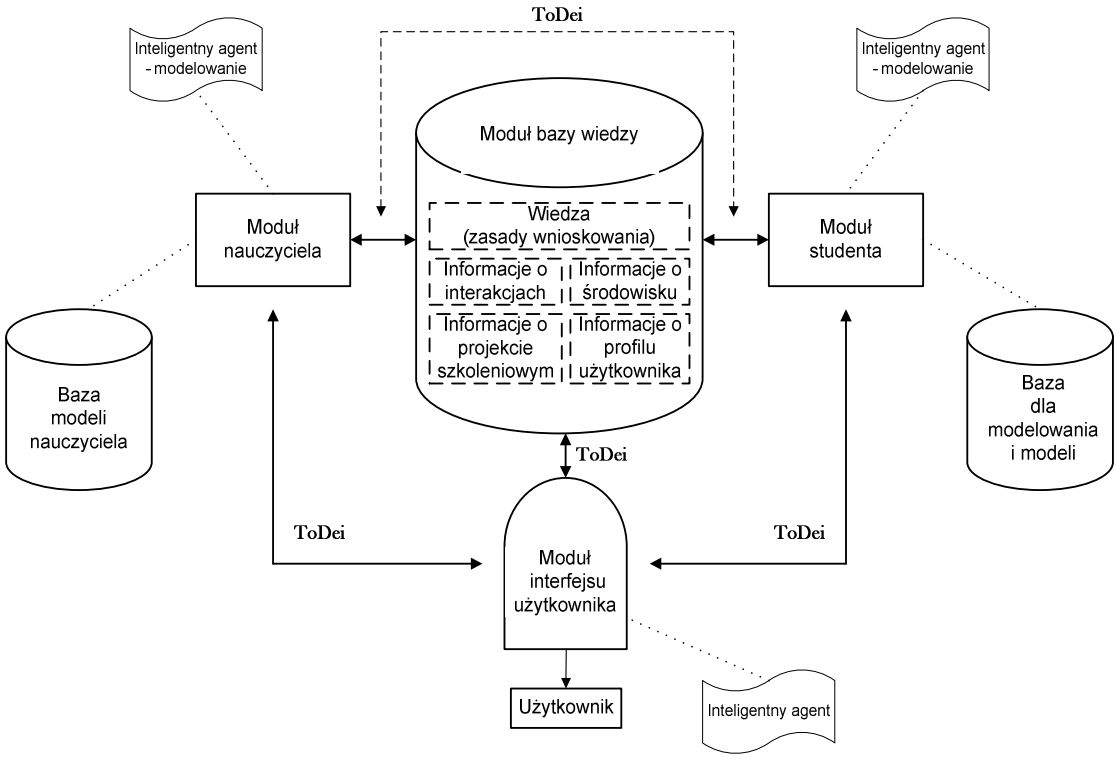
Etykietowanie (ang. *tagging*) polega na wyszukaniu określonego wzorca w zawartości strony, a następnie wynikowi tego poszukiwania przypisuje się etykietę, co pozwala na lokalizację pożądanej treści. Warstwa położenia etykiet wstawia dodatkową zawartość na stronach wysyłanych do przeglądarki w celu jej lokalizacji. Etykieta posiada typ i numer identyfikacyjny, które są unikalne na danej stronie. Decyzja etykietowania oparta jest na rozpoznawaniu wzorca wewnątrz kodu stron platformy e-learningowej. Zaproponowano umieszczenie w architekturze systemu silnika etykietowania (ang. *tagging engine*), który jest związany z aplikacją webową i warstwą pośrednią.

Warstwa dynamicznego ECA jest projektowana zgodnie z architekturą zorientowaną na klienta COA (ang. *Client Oriented Architecture*) i zapewnia właściwe przedstawienie treści na stronie internetowej. Powyższe uzyskuje się poprzez wyróżnienie informacji wykorzystywanych przez oprogramowanie klienckie za pomocą etykiet lokalizacji obszarów na stronie i pozyskanie ich współrzędnych w celu zmiany położenia ECA. COA integruje ECA z treścią strony poprzez zaprojektowanie API oferującego możliwości, takie jak: przemieszczanie ECA, jego powrót do pierwotnej lokalizacji lub zapewnienie reakcji w formie tekstowej. Architekturą wielokrotnego wykorzystania ECA przeznaczoną do celów e-learningowych jest *Cameleon* [35].

W pracy [84] podjęto próbę dostosowania systemu *MOODLE* do potrzeb studentów, tak by każdy z nich mógł zdobywać wiedzę w najbardziej efektywny dla siebie sposób. Adaptacyjna architektura przedstawiona jest na rysunku 2.16.

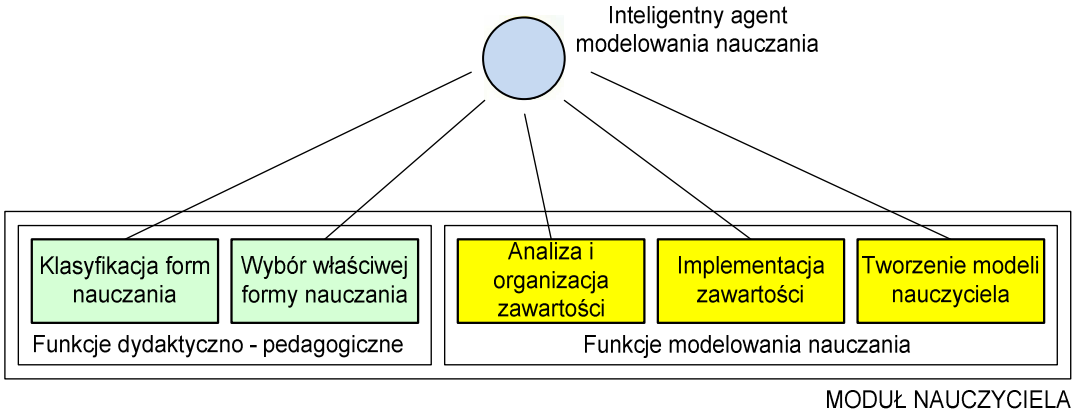
Składa się ona z czterech głównych modułów: modułu nauczyciela, modułu studenta, modułu interfejsu użytkownika oraz bazy wiedzy. Każdy z pierwszych trzech modułów posiada agenta, który wykonuje zadania modułów. Dla komunikacji i interakcji między agentami komponentów zdefiniowano agenta *ToDei*. Jego zadaniem jest zbieranie informacji od każdego modułu i wybór najlepszej metody działania dla użytkownika.

Dla modułu nauczyciela dostępne są funkcje dydaktyczno-pedagogiczne i modelowania nauczania. W przypadku modułu studenta istnieje możliwość tworzenia modeli studenta oraz aktualizacja informacji o studencie. W module interfejsu użytkownika identyfikowane jest oprogramowanie i rodzaj wykorzystywanego urządzenia oraz sposób połączenia z systemem.



Rys. 2.16. Adaptacyjna architektura dla systemu *MOODLE* [84]

Na rysunku 2.17 przedstawiono funkcje dostępne dla modułu nauczyciela.



Rys. 2.17. Funkcje dostępne w module nauczyciela [84]

Agent pełniący funkcję instruktora jest komponentem w tego typu architekturze, w której zidentyfikowany jest najlepszy sposób na udostępnienie informacji studentom.

ToDei to jedyny agent wchodzący w skład architektury, który przesyła informacje w systemie. Pełni główną rolę w zapewnieniu efektu adaptacyjności oraz decyduje o najlepszym sposobie przedstawiania informacji użytkownikowi [84].

2.6. Wnioski i uwagi

W systemie szkolenia wojskowego aplikacje modułowe cechują się znaczącymi zaletami. Dekompozycja programu na moduły podnosi jego jakość i ułatwia zarządzanie projektem programistycznym realizowanym przez zespół programistów. Możliwe jest także wykorzystanie specyficznych cech różnych typów komputerów.

Alokacja modułów *MOODLE* będących bazą wojskowego systemu szkolenia, w celu poprawy dostępności systemu i miary skutecznej realizacji zadań w terminach może zostać przeprowadzona w środowisku Linux, przy użyciu sieciowego systemu plików. Przykładowo, jeśli system szkolenia zawiera I węzłów, to w jednym z nich alokowana może być baza danych, a w pozostałych inne moduły systemu *MOODLE*.

W przypadku optymalizacji przydziału modułów do węzłów w systemie szkolenia z wykorzystaniem wektora X^m automatycznie spełnione jest założenie, że każdy moduł jest przydzielony tylko do jednego z węzłów. Dla całkowitoliczbowego wektora liczba wariantów J^I rośnie wolniej niż liczba wariantów wektora binarnego 2^{IJ}, przy rosnącej liczbie modułów i węzłów.

Odpowiedni przydział modułów programu do komputerów pozwala na skrócenie wykonywania czasu programu sekwencyjnego, obniża koszt realizacji rozproszonego programu oraz zwiększa moc obliczeniową systemu. Redukuje także obciążenie newralgicznego komputera, co prowadzi do równomiernego obciążenia serwerów.

Jeśli istnieje co najmniej jedna para modułów o minimalnych kosztach wykonania na dwóch rodzajach komputerów, to przydział optymalny może być zdecentralizowany, tylko gdy przyrost łącznego kosztu komunikacji między tymi modułami jest niższy niż zysk wynikający z przydzielenia tej pary do „najtańszych” komputerów.

Zbiór rozwiązań dopuszczalnych zawiera co najwyżej $I^V J^I$ przydziałów modułów do komputerów. Rozpatrywanie minimalizacji kosztu wykonania programu rozproszonego, przy liczbie węzłów większej od trzech, powoduje, że to zadanie optymalizacji jest NP-trudne.

W systemach otwartych z dużą liczbą węzłów, koszt komunikacji między modułami zależy od miejsca alokacji modułów. Przy tej samej wielkości przesyłanych danych, czas transmisji komunikatu zależy od pary węzłów, do której przydzielono moduły.

W systemach, w których liczba komputerów jest większa od dwóch, stosuje się jako miarę jakości przydziału *łączne obciążenie komputerów* nazywane także *kosztem obliczeniowym* (2.19). Stosowane są również inne kryteria, takie jak dostępność systemu czy też miara skutecznej realizacji modułów w terminach.

Inteligentne agenty programistyczne ISAs (ang. *Intelligent Software Agents*) coraz częściej stają się elementami rozproszonych systemów informatycznych, systemów operacyjnych, a także usług sieciowych [68, 159]. ISAs w połączeniu z systemem *MOODLE* mogą być wykorzystywane do optymalizacji procesu szkolenia wojskowego.

3. SFORMUŁOWANIE PROBLEMU OPTYMALIZACJI PRZYDZIAŁÓW MODUŁÓW PROGRAMISTYCZNYCH W SYSTEMIE ZDALNEGO SZKOLENIA WOJSKOWEGO

W modelu systemu informatycznego wspomagającego szkolenie wojskowe przyjmuje się, że podczas wykonywania procesów możliwe jest uszkodzenie komputera lub kanału komunikacyjnego. Aby system cechował się tolerowaniem uszkodzeń, powinien być zdolny do autonomicznego wykrycia awarii i określenia stopnia uszkodzenia. W szczególności do diagnozowania stanu systemu można użyć oprogramowania skanującego sieć [48], w tym programu implementującego sztuczną sieć neuronową do określenia stanu systemu [55].

Po ustaleniu stopnia uszkodzenia następuje zamknięcie systemu w trybie awaryjnym lub realizowane jest maskowanie awarii przed aplikacją użytkownika. Wystąpienie awarii powoduje zagrożenie w odniesieniu do wyznaczenia poprawnych wyników, dlatego w zagadnienia optymalizacji przydziałów przyjęto założenie o maksymalizacji *miary dostępności systemu* po to, aby minimalizować ryzyko awarii komputerów podczas wykonywania zadanego zbioru zadań.

W systemach zdalnego szkolenia wojskowego wymagany jest gwarantowany czas odpowiedzi na zdarzenia. W tym wypadku istotną rolę odgrywają zależności czasowe między współbieżnymi procesami. Aby uwzględnić te złożone ograniczenia czasowe, maksymalizuje się *miarę skutecznego ukończenia zadań w terminach*. Przyjmuje się, że rozpoczęcie procesu, powinno nastąpić nie wcześniej niż od zadanego momentu czasu, od którego możliwe jest jego wykonanie. Ponadto koniec zadania nie może nastąpić później niż założony moment najpóźniejszej realizacji procesu. Uszeregowanie procesów umożliwia spełnienie nie tylko formalnych ograniczeń nałożonych na rozpoczęcie i zakończenie zadań, ale również umożliwia optymalizację rozpatrywanego kryterium oceny – miary skutecznego ukończenia zadań w terminach.

Optymalizacja alokacji zadań wykonywanych w rozproszonym systemie informatycznym umożliwia znaczące zmniejszenie obciążenia newralgicznego komputera. Redukcja obciążenia może przekroczyć nawet 50% wartości obciążenia cechującego system, w którym procesy wygenerowano losowo. Nakłada się zatem ograniczenie, aby newralgiczny komputer był obciążony w stopniu nie większym niż założony poziom wymagany ważnością realizowanych zadań szkoleniowych.

Za pomocą doboru rodzajów komputerów można także zmniejszyć koszt systemu rozproszonego tak, aby przydzielony limit kosztów nie został przekroczony. Ponadto żąda się, aby wydajność systemu nie była mniejsza niż założona wielkość wg przyjętego benchmarku.

W pracy problem alokacji modułów programistycznych rozważanego systemu szkolenia wojskowego sformułowano w postaci zagadnienia optymalizacji dwukryterialnej. Do wyznaczania reprezentacji rozwiązań optymalnych w sensie Pareto można zastosować zmodyfikowany algorytm genetyczny NSGA-III, który opracowano na bazie algorytmu Deba. Uzyskane wyniki porównano z rezultatami otrzymanymi za pomocą adaptacyjnego algorytmu ewolucyjnego AMEA [20] oraz opracowanego w pracy oryginalnego algorytmu genetycznego operującego na populacji sztucznych dwuwarstwowych sieci neuronowych.

3.1. Maksymalizacja dostępności systemu

Podczas wykonywania procesów możliwe jest uszkodzenie komputera. Wystąpienie awarii powoduje zagrożenie w odniesieniu do poprawności uzyskanych wyników. Dlatego spośród kryteriów niezawodnościowych wybrano miarę dostępności (ang. *reliability*) systemu.

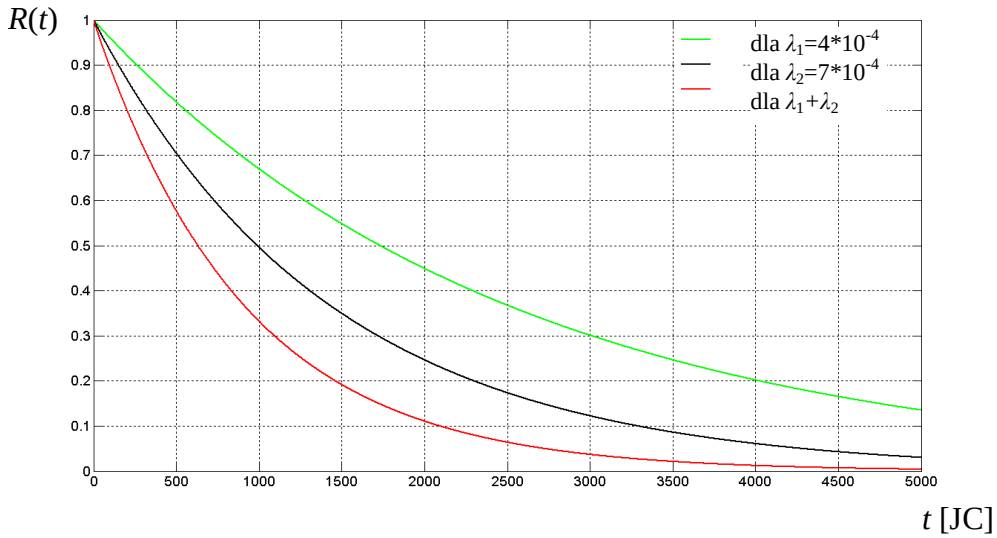
Jeżeli w jednostce czasu ma zajść $1/\lambda$ niezależnych zdarzeń prowadzących do uszkodzenia komputera w systemie, to rozkład wykładniczy z parametrem λ opisuje odstępy czasu pomiędzy kolejnymi zdarzeniami. Rozkład wykładniczy w literaturze nazywany jest rozkładem „bez pamięci” [123]. Przyjmuje się, że komputer *j*-tego rodzaju ulega awarii, przy czym dostępność komputera maleje w czasie wykładniczo wraz z parametrem λ_j. Dostępność systemu *R* wyznacza się w następujący sposób [133]:

$$R(\Lambda,T,x)=\prod_{v=1}^{V}\prod_{i=1}^{I}\prod_{j=1}^{J} exp(-\lambda_j t_{vj} x_{vi}^{m} x_{ij}^{\pi}) \tag{3.1}$$

gdzie $\boldsymbol{\Lambda}=[\lambda_1,...\lambda_j,...\lambda_J]$- wektor parametrów rozkładów wykładniczych uszkodzeń komputerów poszczególnych rodzajów.

Dostępność systemu nie posiada jednostek miary oraz przyjmuje wartości z przedziału domkniętego [0; 1]. Jeśli rozważa się system, w którym występują dwa rodzaje komputerów, to dostępność systemu maleje szybciej niż dostępność każdej z maszyn.

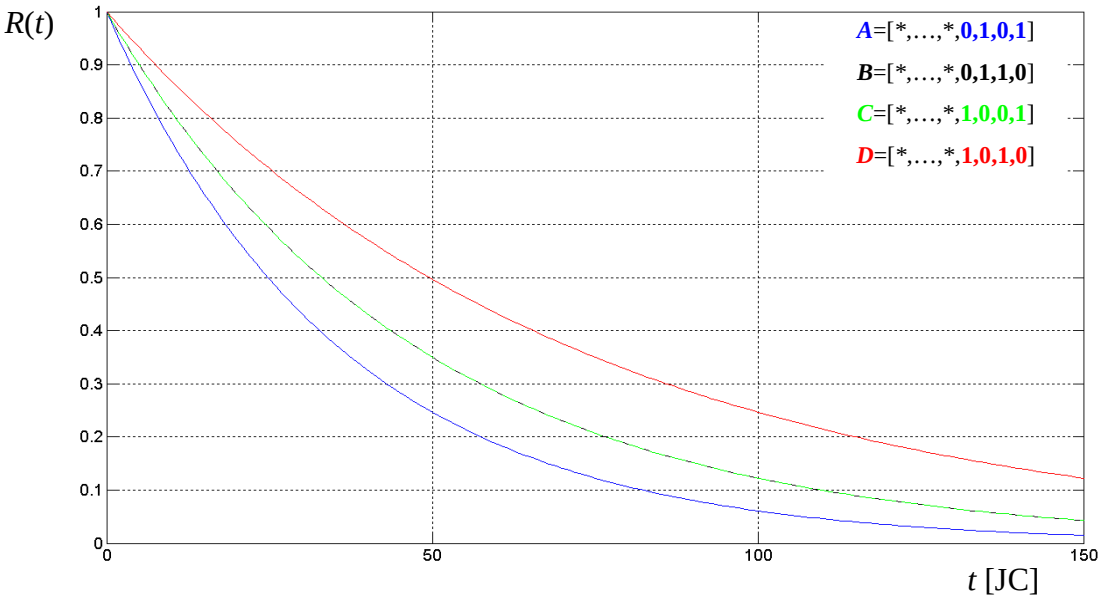
Niech w systemie dwukomputerowym zachodzi $\lambda_1 > \lambda_2$. Ponadto zadany jest wektor $\boldsymbol{\Lambda}$ o wartościach niezmiennych w czasie. Zakłada się, że system pracuje w czasie t zmieniającym się w sposób ciągły od chwili t_0=0 do chwili t_{stop}. Rysunek 3.1 przedstawia zależność miary dostępności systemu R od czasu dla rozważanego systemu. Przyjęto, że dostępność systemu zależy od czasu pracy systemu. Jeśli wszystkie elementy macierzy $\boldsymbol{T}$ będą równe t oraz jeśli t będzie ulegało zmianom w przedziale ciągłym [0; t_{stop}] dla różnych macierzy $\boldsymbol{T}$, to zależność dostępności systemu R dla ustalonego przydziału modułów do komputerów x oraz zadanego wektora niezawodności $\boldsymbol{\Lambda}$ zostanie zredukowana do zależności tylko od czasu t.



Rys. 3.1. Charakterystyka dostępności systemu dwukomputerowego w porównaniu do dostępności pojedynczych elementów systemu
Źródło: opracowanie własne.

Na rysunku 3.2 przedstawiono zależność miary dostępności systemu R od czasu w systemie dwukomputerowym dla czterech wybranych, wyszczególnionych na wykresie, przydziałów rodzajów komputerów do węzłów.

Pogrubiona część wektorów *A, B, C, D* odpowiada przydziałowi x^{π}. W skład systemu wchodzi 14 modułów, które mogą być alokowane na dwóch typach komputerów. Przyjęto wartości parametrów rozkładu wykładniczego awarii λ_1=0,0005 [JC^{-1}], λ_2=0,001 [JC^{-1}] oraz założono, że czas realizacji zadań wynosi 150 [JC] dla każdego komputera. W tym wypadku różnice w dostępności systemu zależą od wartości parametrów λ_1 i λ_2, a także od przydziału x^{π} rodzajów komputerów. Wariant *A* cechuje się najniższą dostępnością, gdyż do obu węzłów przydzielono komputery charakteryzujące się współczynnikiem λ_2 o większej wartości niż λ_1.



Rys. 3.2. Dynamika zmniejszania dostępności systemu dla wybranych przydziałów
Źródło: opracowanie własne.

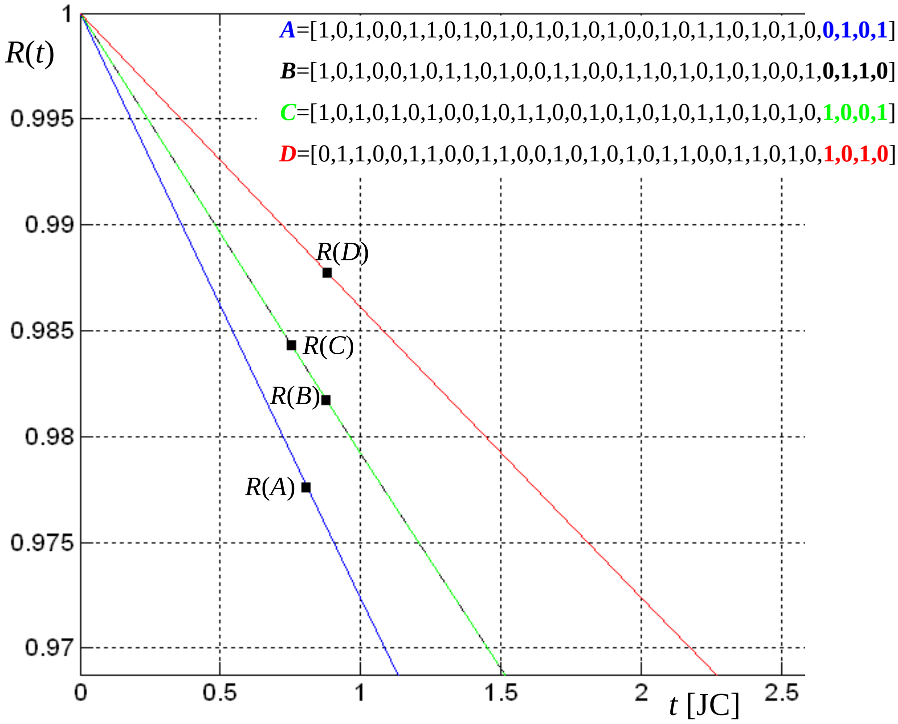
W drugim etapie analizy, zakłada się, że macierz czasów realizacji zadań odpowiada macierzy z podrozdziału 2.4. Warto podkreślić, że niskie wartości współczynników λ i czasów realizacji zadań powodują, że wartość dostępności systemu jest relatywnie wysoka. Wyznaczono i naniesiono na rysunku 3.3 następujące wartości dostępności: $R(A)$=0,9773, $R(B)$=0,9817, $R(C)$=0,9846 oraz $R(D)$=0,9876.

Warianty B i C nie charakteryzują się identyczną dostępnością, mimo że do węzłów przydzielono komputery, dla których sumy współczynników λ są równe. Wprawdzie wykresy dostępności na rysunku 3.2 dla wariantów B i C nakładają się, to jednak ocena wariantu zależy także od czasów realizacji zadań oraz od przydziału zadań.

Należy zauważyć, że dla przydziałów B i C, wartości dostępności są różne, gdyż czasy realizacji zadań cechują się innymi wartościami. Ustalając większe wartości współczynników λ_1=0,05 [JC^{-1}] i λ_2=0,1 [JC^{-1}], otrzymuje się mniejsze wartości dostępności systemu $R(A)$=0,1003, $R(B)$=0,1572, $R(C)$=0,2122 oraz $R(D)$=0,2865.

W kolejnym przypadku przyjęto, że program rozproszony składa się z pięciu modułów rozmieszczonych w dwóch węzłach, a macierz czasów przetwarzania modułów dla dwóch rodzajów komputerów ma postać, jak niżej:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 3 & 2 & 3 & 4 \\ 2 & 4 & 10 & 7 & 3 \end{bmatrix}^T \text{[JC]}$$



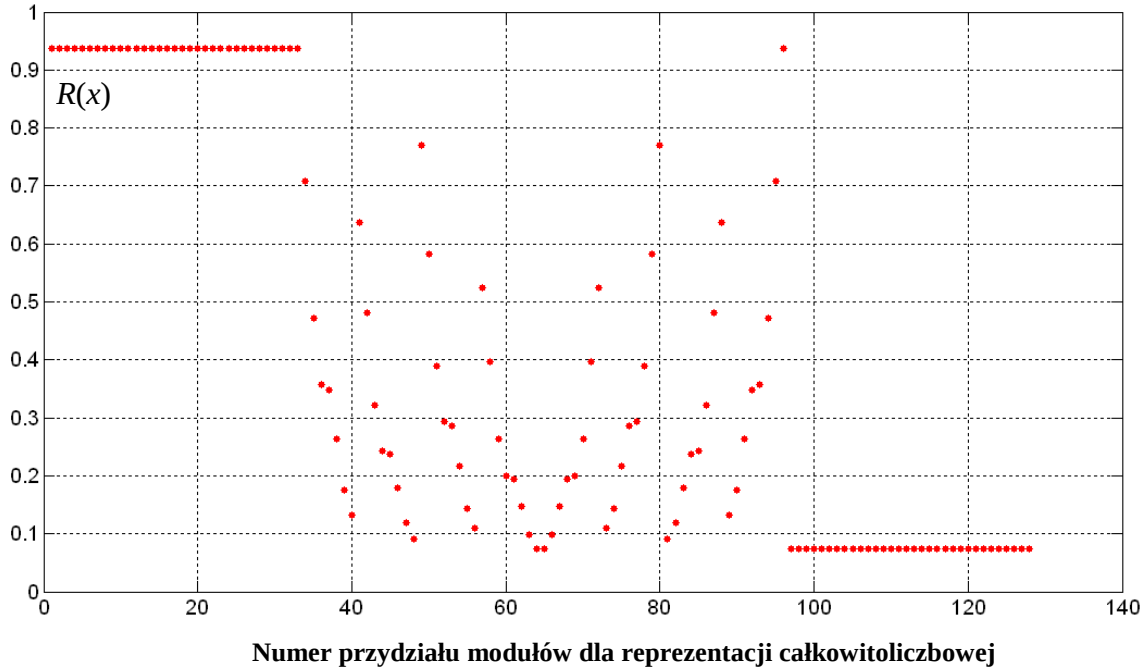
Rys. 3.3. Wartości dostępności systemu dla wybranych przydziałów
Źródło: opracowanie własne.

Wektor niezawodności komputerów przedstawiony jest poniżej:

$$\boldsymbol{\Lambda} = [5 \times 10^{-3}, \quad 10^{-1}]$$

Przy ustalonej macierzy $\boldsymbol{T}$ oraz zadanym wektorze $\boldsymbol{\Lambda}$ uzyskano, przedstawione na rysunku 3.4, wartości dostępności systemu między 0,0743 a 0,9371 w odniesieniu do rozpatrywanych przydziałów.

Warianty o numerach początkowych (rys. 3.4) cechują się taką samą dostępnością, co wynika ze sposobu obliczania $R(x)$ dla przydziałów całkowitoliczbowych. Do wszystkich węzłów jest przydzielony komputer tego samego typu nr j, przez co uwzględniane są tylko j-te czynniki w zależności (3.1). Liczba mogących mieć wpływ na dostępność czynników redukuje się z VIJ do VI. Ze względu na fakt, że moduł może zostać przydzielony tylko do jednego węzła, wpływ na wartość $R(x)$ z pozostałych VI czynników mają tylko te, dla których $x_{vi}^{m} = 1$. Zatem liczba istotnych czynników redukuje się do V. Dla przypadku z rysunku 3.4 dla pierwszych I^V=32 przydziałów w węzłach występują komputery tego samego typu ze współczynnikiem $\lambda_1 > \lambda_2$. Na wartość $R(x)$ mają wpływ: współczynnik λ_1 oraz czasy t_{v1}, co dla zadanej wartości j wyznacza wartość wynoszącą $R(x)=e^{(-\lambda_1(t_{11}+t_{21}+t_{31}+t_{41}+t_{51}))}$. Podobnie można analizować warianty o numerach największych na rysunku 3.4.



Rys. 3.4. Dostępność wybranego systemu dwukomputerowego
Źródło: opracowanie własne.

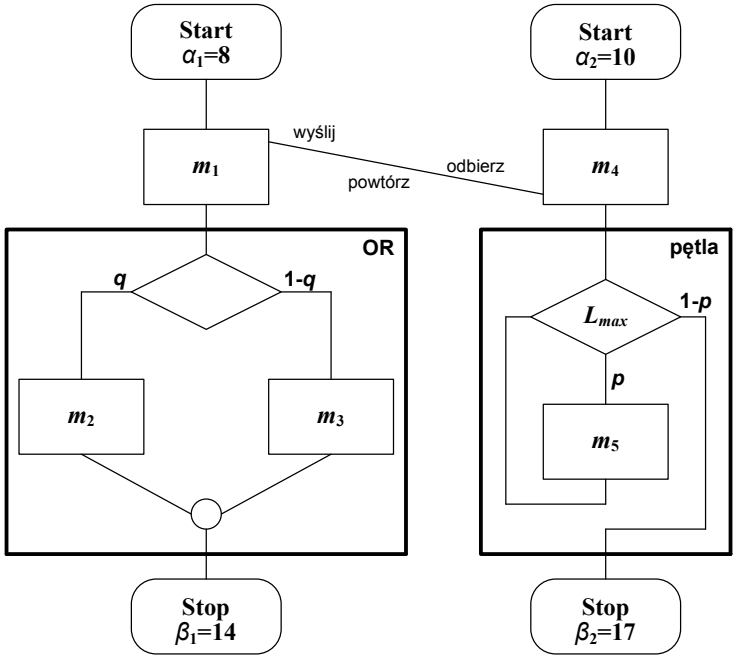
3.2. Maksymalizacja miary skutecznej realizacji modułów w terminach

Wyznaczenie przydziału modułów do komputerów, będącego parą (x^m, x^π), umożliwia oszacowanie łącznego kosztu wykonania procesów, obciążenia newralgicznego komputera w systemie, kosztu komputerów lub wydajności systemu [18]. Przydział procesów jest także podstawą do uszeregowania procesów w każdym komputerze.

Niech wykonywanie programu rozproszonego $\boldsymbol{P_d}$, gdzie d zmienia się od 1 do D, rozpoczyna się najwcześniej od momentu α_d i kończy się najpóźniej po upływie czasu β_d. Niektóre procesy mogą być wywoływane wielokrotnie podczas wykonania programu, a inne – co najwyżej raz.

Do modelowania logicznej struktury programów rozproszonych, w których wyodrębniono procesy, stosuje się diagram przepływu $\boldsymbol{G_F}$ systemu rozproszonego [76]. Zaletą takiego modelu jest uwzględnienie ograniczeń kolejnościowych oraz interakcji komunikacyjnych między procesami. Do podstawowych typów struktur diagramu przepływu oprócz sekwencji procesów należą struktury diagramu typu: *OR* i *AND*. Procedury komunikacyjne *wyślij*, *odbierz* i *powtórz* pozwalają na realizację komunikatów.

Na rysunku 3.5 przedstawiono przykładowy diagram przepływu dla dwóch programów rozproszonych. Wyróżniono strukturę diagramu typu *OR*, w którym proces m_2 realizowany jest z prawdopodobieństwem *q*, a proces m_3 – z (1-*q*).



Rys. 3.5. Przykładowy diagram przepływu dla dwóch programów rozproszonych i pięciu procesów
Źródło: opracowanie własne.

Niech w strukturze diagramu typu *pętla* (*AND*) proces m_5 wykonywany może być co najwyżej $L_{\max}$ razy, a każda ponowna realizacja m_5 zachodzi z częstością $p \in [0;1]$. Zakłada się, że warunek sprawdzany jest na początku instrukcji pętli, tak jak w instrukcjach *while* lub *for* w języku C++ lub Java. Wartości częstości *p* wyznacza się za pomocą wielokrotnego przetwarzania różnorodnych zestawów danych wejściowych.

Diagram przepływu $\boldsymbol{G_F}$ nie może być bezpośrednio wykorzystany do szeregowania procesów w wypadku, gdy występuje struktura diagramu typu pętla lub *OR*. W wypadku struktury typu *OR* z rysunku 3.5 konstruowane są dwie instancje diagramu przepływu. Pierwsza instancja zawiera proces m_2 zamiast struktury *OR* i jest realizowana z częstością *q*. Druga instancja zawiera proces m_3 zamiast struktury OR i jest realizowana z częstością (1-*q*).

W wypadku struktury typu *pętla* o maksymalnej liczbie $L_{\max}$ powtórzeń procesu konstruuje się $L_{\max}$ instancji diagramu przepływu, a w każdej instancji proces

wykonywany jest k razy (k=0,1,2,...$L_{\max}$). Niech instancja z k-krotnym wykonaniem procesu, np. m_5 jest realizowana z prawdopodobieństwem $(1-p)p^k$ zgodnie z rozkładem Bernoulliego [76]. W odniesieniu do diagramu przepływu z rysunku 3.5 możliwe jest wystąpienie $2(1+L_{\max})$ instancji. Instancja, w której realizowany jest proces m_2 oraz k-krotnie m_5, zachodzi z prawdopodobieństwem, jak niżej:

$$p_i = q(1-p)p^k \tag{3.2}$$

Do szeregowania zadań stosuje się techniki optymalizacji kombinatorycznej, programowania liniowego, programowania dynamicznego, a także teorii grafów [65]. Przydział procesów do komputerów $x=(x^m, x^\pi)$ umożliwia uszeregowanie procesów w każdym komputerze systemu. Niech uszeregowania zadań będą określone za pomocą następującego wektora liczb naturalnych:

$$N^m = [N_1, ..., N_v, ..., N_V]^T, \tag{3.3}$$

gdzie N_v – numer v-tego procesu w kolejce do wykonania na komputerze, do którego ten proces przydzielono.

Można wprowadzić ograniczenia dotyczące kolejności wykonania modułów. Pierwsze ograniczenie dotyczy przypadku, gdy moduły są przydzielone do tego samego węzła i powinny mieć różną kolejność wykonania. Dla każdej pary modułów v, u przydzielonych do węzła, jeśli $x_{vi}^m = x_{ui}^m = 1$, to zachodzi $N_v \neq N_u$ dla $i = \overline{1, I}$.

Drugie ograniczenie związane jest z kolejnością wykonania pary modułów na różnych komputerach. Jeśli dla każdej pary modułów v, u przydzielonych do różnych węzłów zachodzi $N_v = N_u$ to wtedy $x_{vi}^m \neq x_{ui}^m$ dla $i = \overline{1, I}$. Kolejne ograniczenie kolejnościowe związane jest z diagramem przepływu. Zapis $m_v \to m_l$ oznacza, że wykonanie modułu l może nastąpić dopiero po wykonaniu modułu v.

Na podstawie uszeregowanego przydziału procesów do komputerów $x=(x^m, x^\pi)$ można wyznaczyć czasy ukończenia procesów (C_1,...,C_v,...,C_V) [16]. Niech czas rozpoczęcia wykonywania procesu określony będzie, jak niżej [76]:

$$r_v = \max\left\{ r_v, \max_l \left\{ r_l + t_{vj} + \tau_{vl}(x) : m_v \to m_l \right\} \right\}, \; 2 \le v \le V \tag{3.4}$$

Niech d_v reprezentuje najpóźniejszy moment ukończenia v-tego procesu, które to zakończenie nie spowoduje przekroczenia czasu założonego na wykonanie programu:

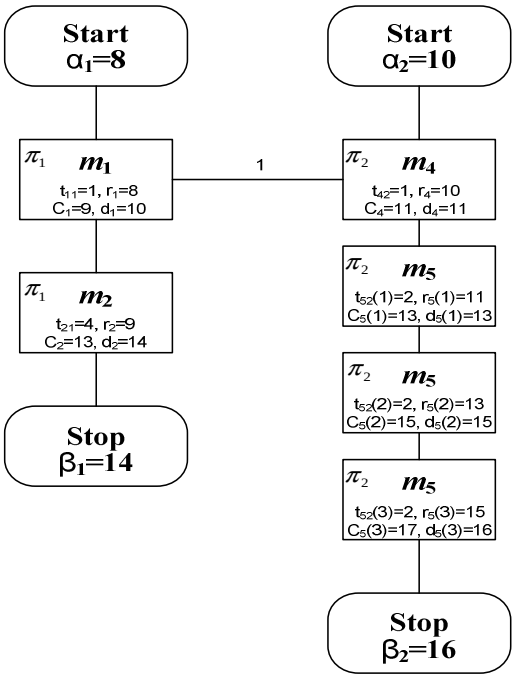
$$d_v = \min\left\{ d_v, \min_l \left\{ d_l - t_{vj} - \tau_{vl}(x) : m_v \to m_l \right\} \right\},\ v=V\text{-}1,\ V\text{-}2,\ldots,1 \qquad (3.5)$$

Początkowo d_v, przyjmuje się wartość nieprzekraczalnego terminu ukończenia programu β_n. Następnie wartości obliczane są wg zależności (3.5).

Czas ukończenia procesu C_v obliczany jest jako suma czasu rozpoczęcia wykonywania procesu r_v i czasu wykonania procesu t_{vj}:

$$C_v = r_v + t_{vj}, \quad v = \overline{1,V}. \qquad (3.6)$$

Rysunek 3.6 przedstawia instancję diagramu, dla której trzykrotnie (L_{max}=3) realizowany jest proces m_5 z prawdopodobieństwem p=0,5 oraz proces m_2 z prawdopodobieństwem q=0,7.



Rys. 3.6. Wybrana instancja diagramu przepływu z rysunku 3.3, w której realizowany jest proces m_2 oraz trzykrotnie proces m_5
Źródło: opracowanie własne.

Na rysunku 3.6 przedstawiono parametry czasowe poszczególnych procesów. Prawdopodobieństwo wystąpienia rozważanej instancji to $0{,}7(1-0{,}5)0{,}5^3 = 0{,}0437$. Czas komunikacji między procesem m_1 a procesem m_4 wynosi 1 [JC]. W przykładzie

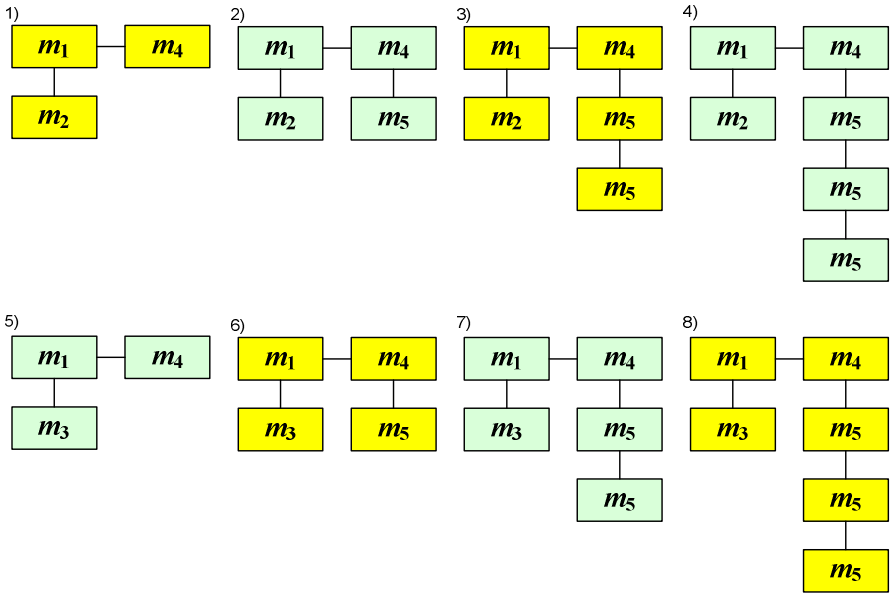
nie uwzględniono wywłaszczania (ang. *preemptive*) procesów przez inne procesy [76, 154]. Założono, że procesy m_1, m_2 realizowane są na komputerze typu π_1, a procesy m_4 i m_5 na komputerach typu π_2.

Na rysunku 3.7 przedstawiono dopuszczalne instancje diagramu przepływu z rysunku 3.5, przy założeniu maksymalnie potrójnego wykonania modułu m_5. W ogólnym przypadku liczba instancji L_{INST} wynosi:

$$L_{INST} = 2^A \prod_{p=1}^{P} (L_{\max}^p + 1),$$

gdzie A określa liczbę struktur diagramu typu OR w diagramie przepływu, a P oznacza liczbę struktur typu AND.

W odniesieniu do diagramu przepływu z rysunku 3.5, gdzie występuje struktura typu OR i struktura typu AND z parametrem L_{max}=3, otrzymuje się $L_{INST}=2^1(3+1)=8$ instancji.



Rys. 3.7. Instancje diagramu przepływu z rysunku 3.3, przy założeniu L_{max}=3
Źródło: opracowanie własne.

Jeśli $C_v \le d_v$, to spełnienie tego ograniczenia zapisuje się jako $\xi(d_v - C_v) = 1$. W wypadku przekroczenia czasu dopuszczalnego zachodzi $\xi(d_v - C_v) = 0$.

Wprowadza się dwukolumnową macierz przepływu $\boldsymbol{\Omega}$, w której wiersze zawierają numery par modułów. W pierwszej kolumnie podany jest numer modułu,

który wykonuje się jako pierwszy w odniesieniu do modułu, którego numer zawiera druga kolumna. W strukturze interakcji „każdy z każdym”, w wypadku gdy nie występują ograniczenia kolejnościowe, maksymalnie może wystąpić $V(V\text{-}1)$ wierszy macierzy $\boldsymbol{\Omega}$. Liczba wariantów zmniejsza się dwukrotnie, gdy występują ograniczenia kolejnościowe $m_v \rightarrow m_l$. Liczba wierszy macierzy $\boldsymbol{\Omega}$ zależy od diagramu przepływu. Dla diagramu z rysunku 3.5 macierz $\boldsymbol{\Omega}$ przedstawiona jest poniżej.

$$\boldsymbol{\Omega}=\begin{bmatrix} 1 & 1 & 1 & 4 & 5 \\ 2 & 3 & 4 & 5 & 5 \end{bmatrix}^T$$

Jeśli wiersz macierzy $\boldsymbol{\Omega}$ zawiera te same numery modułów, oznacza to, że w diagramie przepływu występuje pętla. Jeżeli kolejny wiersz pierwszej kolumny zawiera ten sam numer modułu oznacza to, że dany moduł poprzedza wykonanie co najmniej dwóch modułów. Dla przedstawionej powyżej macierzy $\boldsymbol{\Omega}$ moduł nr 1 poprzedza wykonanie zarówno modułu nr 2, jak i modułu nr 3 oraz modułu nr 4. Dla k-tej instancji diagramu przepływu generowana jest macierz $\boldsymbol{\Omega}_k$, która zawiera tylko pary modułów występujące w danej instancji. Przykładowo, dla instancji nr 4 z rysunku 3.7, przedstawionej na rysunku 3.6, macierz $\boldsymbol{\Omega}_3$ ma postać, jak niżej.

$$\boldsymbol{\Omega}_3=\begin{bmatrix} 1 & 1 & 4 & 5 & 5 \\ 2 & 4 & 5 & 5 & 5 \end{bmatrix}^T$$

Konstruuje się także wektor $\boldsymbol{A}$, którego elementy określają maksymalną liczbę wykonań danego modułu w programie.

Ponadto określa się wektor $\boldsymbol{E}$, którego elementy niezerowe zawierają zadane czasy rozpoczęcia wykonywania modułów. Należy zauważyć, że modułowi m_4 odpowiada zerowa wartość elementu wektora $\boldsymbol{E}$, mimo że jest on pierwszym modułem w drugim programie rozproszonym, ponieważ jego wywołanie następuje z modułu m_1, wchodzącego w skład pierwszego programu rozproszonego i nie ma zadanego czasu rozpoczęcia wykonania.

Ponadto konstruuje się wektor $\boldsymbol{\Theta}$, którego elementy określają, w jakiej strukturze diagramu znajduje się zadany moduł. Jeśli wartość elementu wektora $\boldsymbol{\Theta}$ przyjmuje wartość 2, oznacza to, że moduł odpowiadający temu elementowi wchodzi w skład struktury typu OR. Wartość 3 oznacza przynależność do struktury typu AND.

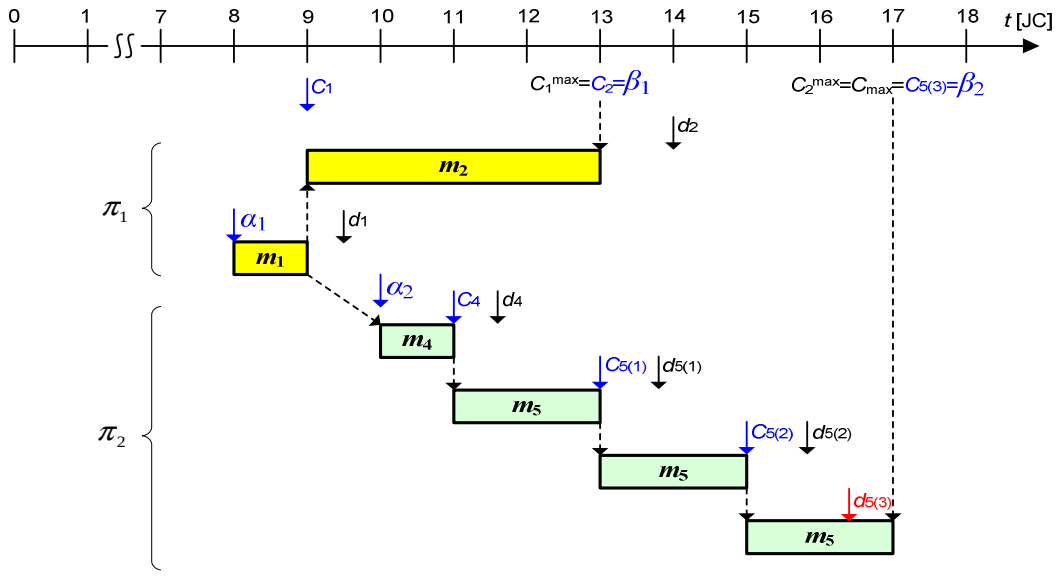
Dla diagramu przepływu z rysunku 3.5 wektory $\boldsymbol{A}$, $\boldsymbol{E}$ oraz $\boldsymbol{\Theta}$ to:

$\boldsymbol{A}$=[1,1,1,1,3], $\boldsymbol{E}$=[8,0,0,0,0], $\boldsymbol{\Theta}$ =[1,2,2,1,3].

Stan spełnienia ograniczeń czasowych w odniesieniu do *i*-tej instancji diagramu przepływu, w którym zbiór procesów oznaczono jako M_i, wyznacza się następująco:

$$S_i = \prod_{m_v \in M_i} \xi(d_v - C_v(x)) \tag{3.7}$$

Niech $C_j^{\max}$ oznacza najpóźniejszy czas ukończenia procesu na komputerze typu *j*. Natomiast najpóźniejszy czas ukończenia procesu w systemie oznaczony jest za pomocą $C_{\max}$. Diagram na rysunku 3.8 przedstawia zależności czasowe dla rozważanej instancji diagramu przepływu. Da trzeciej realizacji procesu m_5 zachodzi $\xi(d_v - C_v) = 0$.



Rys. 3.8. Diagram obrazujący zależności czasowe w diagramie przepływu dla wariantu nr 4 z rysunku 3.7
Źródło: opracowanie własne.

Miara skutecznej realizacji zbioru procesów dla L_{INST} możliwych instancji diagramu przepływu jest określona następująco:

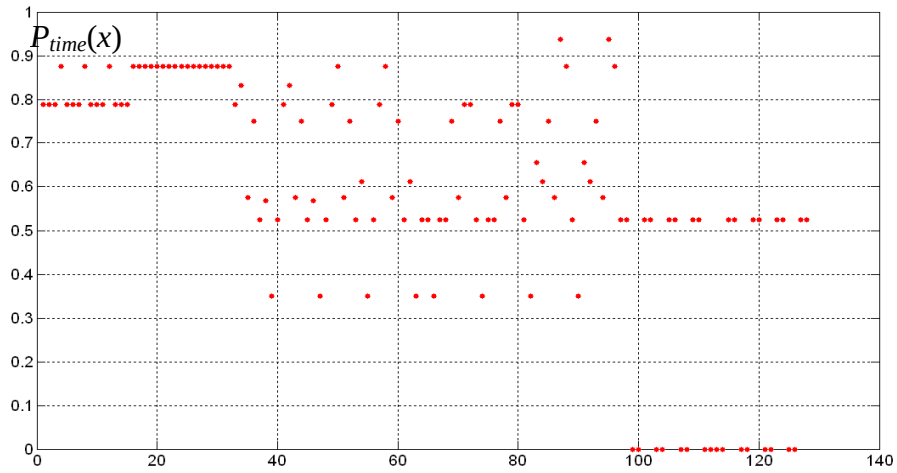
$$P_{time}(x) = \sum_{i=1}^{L_{INST}} p_i \prod_{m_v \in M_i} \xi(d_v - C_v(x)) \tag{3.8}$$

Instancja, dla której ograniczenie czasowe nie jest spełnione, nie powoduje zwiększenia $P_{time}(x)$. Jeśli dla diagramu przepływu z rysunku 3.5 ograniczenia czasowe są spełnione dla wszystkich instancji, otrzymuje się: p_1=0,35; p_2=0,175; p_3=0,0875; p_4=0,0437; p_5=0,15; p_6=0,075; p_7=0,0375; p_8=0,0188 oraz $P_{time}(x)$=0,9375.

Jeśli ograniczenia czasowe są spełnione dla *k* instancji, to liczba możliwych wartości miary $P_{time}(x)$ jest *k*-elementową kombinacją bez powtórzeń dla zbioru

zawierającego L_{INST} elementów. Niech L_{INST}=3 oraz k=2. Wówczas otrzymuje się zbiór {(1,2), (1,3), (2, 3)}, a dla k=3 - zbiór {1, 2, 3}. W nawiasach znajdują się numery instancji, dla których spełnione są ograniczenia czasowe. Liczba tych instancji obliczana jest przy pomocy zależności na liczbę kombinacji bez powtórzeń $\binom{L_{INST}}{k}$. Należy również uwzględnić przypadek, gdy żadne z ograniczeń czasowych nie będzie spełnione. Zatem liczba możliwych wartości $P_{time}(x)$, które można uzyskać wynosi $1+\sum_{k=1}^{L_{INST}}\binom{L_{INST}}{k}$, przy założeniu, że prawdopodobieństwa wystąpienia każdej instancji różnią się od siebie. W analizowanym przypadku dla ośmiu instancji diagramu przepływu otrzymuje się 256 możliwych wartości $P_{time}(x)$.

Niech czas komunikacji między procesami rozmieszczonymi w różnych węzłach wynosi 1 [JC]. Na rysunku 3.9 przedstawiono wartości miary skutecznej realizacji zadań w terminach dla V=5 oraz I=J=2, przy założeniu rozkładu Bernoulliego dla prawdopodobieństw wykonania modułu w strukturze diagramu typu AND.



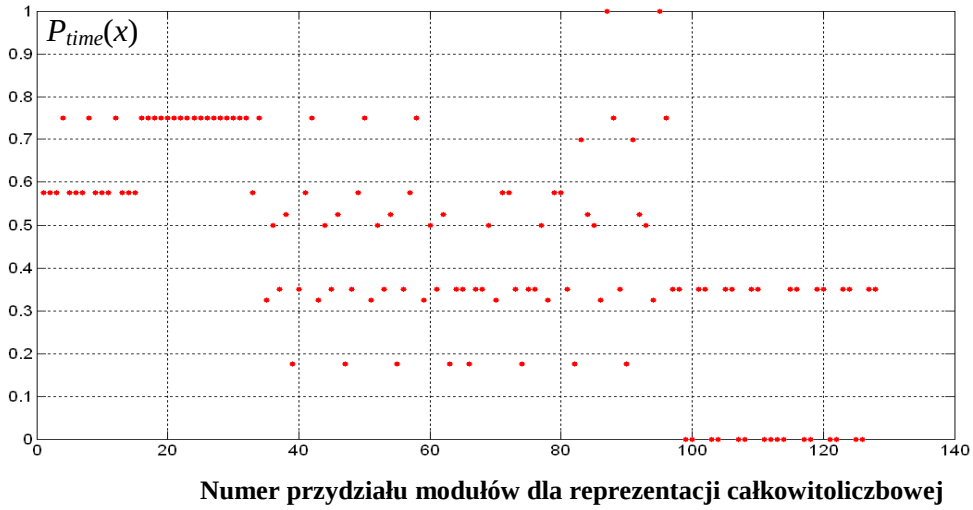
Rys. 3.9. Przegląd wartości miary skutecznej realizacji zadań w terminach przy założeniu rozkładu Bernoulliego dla prawdopodobieństw wykonania modułu w strukturze diagramu typu AND
Źródło: opracowanie własne.

Wartość prawdopodobieństwa wystąpienia instancji diagramu przepływu nie zależy od przydziału modułów do komputerów, ale zależy od wartości p, q oraz od wartości parametru L_{max}. Stan spełnienia ograniczeń czasowych S_i w odniesieniu do wybranej instancji diagramu przepływu przyjmuje wartość 0 lub 1 w zależności od rozmieszczenia procesów w węzłach.

Ponieważ w odniesieniu do struktury typu AND instancja może zawierać k wykonań zadanego modułu dla $k = 0,1,\ldots,L_{\max}$, to wskazane jest przyjęcie założenia, że prawdopodobieństwo k-krotnego wykonania modułu wynosi $1/(L_{\max}+1)$ zgodnie z rozkładem równomiernym. Wówczas w odniesieniu do diagramu z rysunku 3.5 zależność do wyznaczania prawdopodobieństwa wystąpienia instancji nr 3 ma następującą postać:

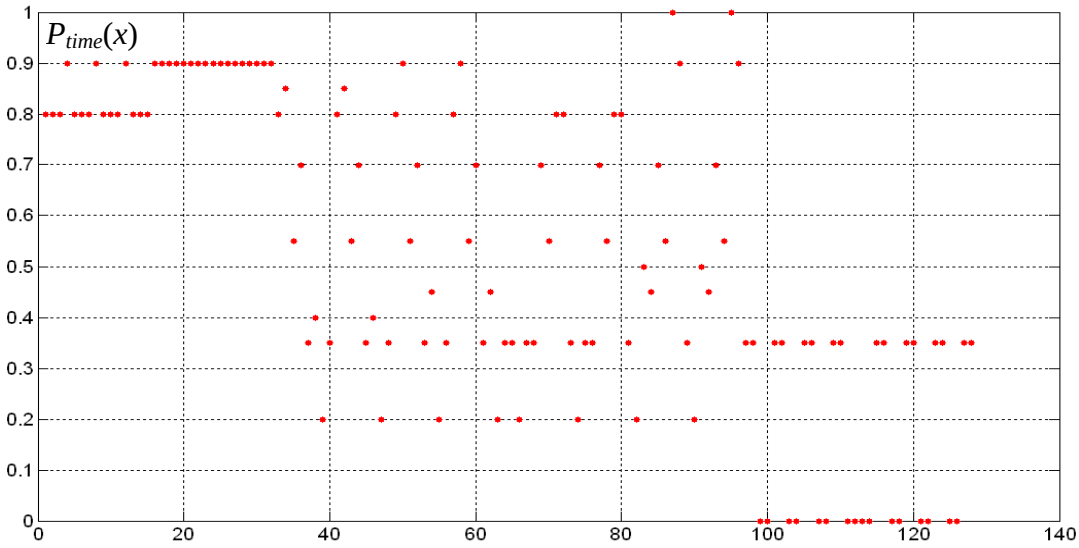
$$p_i=q/(L_{\max}+1) \tag{3.9}$$

Na rysunku 3.10 zobrazowano wartości miary skutecznej realizacji zadań w terminach przy założeniu rozkładu równomiernego dla prawdopodobieństw wykonania modułu w strukturze typu AND.



Rys. 3.10. Przegląd wartości miary skutecznej realizacji zadań w terminach przy założeniu rozkładu równomiernego dla prawdopodobieństw wykonania modułu w strukturze diagramu typu AND
Źródło: opracowanie własne.

Istnieje również możliwość empirycznego wyznaczenia wartości częstości wystąpienia instancji dla reprezentatywnego zbioru zestawów danych wejściowych. Na rysunku 3.11 przedstawiono oznaczenia wartości częstości ukończenia zadań w terminach dla ustalonych częstości wykonania modułu w strukturze diagramu typu AND w odniesieniu do reprezentatywnego zbioru zestawów danych wejściowych. Założono, że: $p_1=p_5=0,2$; $p_2=p_6=0,15$; $p_3=p_7=0,1$; $p_4=p_8=0,05$. W tym wypadku uzyskano dla wybranych przydziałów $P_{time}(x)=1$. Poszczególne prawdopodobieństwa wystąpienia instancji p_i nie zależą od parametrów p, q oraz L_{max}.



Rys. 3.11. Przegląd wartości miary skutecznej realizacji zadań w terminach dla wyznaczonych prawdopodobieństw wykonania modułu w strukturze diagramu typu AND w odniesieniu do reprezentatywnego zestawu danych
Źródło: opracowanie własne.

W tabeli 5 zamieszczono wartości $P_{time}(x)$ dla rozkładu Bernoulliego, rozkładu równomiernego oraz ustalonych prawdopodobieństw w odniesieniu do powtarzania się wykonywania modułu w pętli. Założono spełnienie ograniczeń czasowych.

Tabela 5. Wartości miary $P_{time}(x)$ dla wybranych rozkładów prawdopodobieństw wykonania modułu w strukturze diagramu typu *AND*

p=0,3											
q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$P_{time}(x)$ dla rozkładu empirycznego	1										
$P_{time}(x)$ dla rozkładu równomiernego	1										
$P_{time}(x)$ dla rozkładu Bernoulliego	0,992										
q=0,7											
p	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$P_{time}(x)$ dla rozkładu empirycznego	1										
$P_{time}(x)$ dla rozkładu równomiernego	1										
$P_{time}(x)$ dla rozkładu Bernoulliego	1	0,999	0,998	0,992	0,974	0,937	0,87	0,76	0,59	0,34	0

Źródło: opracowanie własne.

Dla rozkładów: równomiernego i empirycznego wartości parametrów p i q nie wpływają na wartość miary $P_{time}(x)$. W przypadku rozkładu Bernoulliego dla ustalonego parametru p, wartość $P_{time}(x)$ nie ulega zmianie, aczkolwiek jest mniejsza od jedności.

Z kolei dla ustalonego parametru q wartość $P_{time}(x)$ przyjmuje wartość 1, ale jest to przypadek, w którym p=1, co oznacza zerowe prawdopodobieństwo wykonania się modułu m_5 w pętli.

3.3. Wyznaczanie przydziałów optymalnych w sensie Pareto

Miara skutecznego ukończenia zadań w terminach oraz miara dostępności systemu podczas wykonywania zadań to założone kryteria cząstkowe oceny jakości realizacji zadań w rozważanym systemie rozproszonym.

Istnieje konflikt między miarą skutecznej realizacji zadań w terminach a dostępnością systemu. Systemy cechujące się większą miarą skutecznej realizacji zadań w terminach zazwyczaj cechuje mniejsza dostępność. Powoduje to, że zwiększenie wartości kryterium $P_{time}(x)$ jest związane ze spadkiem wartości $R(x)$.

Do wyznaczania reprezentacji Pareto-optymalnych przydziałów procesów do komputerów $x=(x^m, x^\pi)$, dla danych: ***M, Π, W, T, τ, Λ***, diagram przepływu ze zbiorem par ograniczeń kolejnościowych, ***d,*** p, oraz q, sformułowano zagadnienie optymalizacji wielokryterialnej w postaci uporządkowanej trójki:

$$(\boldsymbol{X}, F, P), \tag{3.10}$$

1) $\boldsymbol{X}$ – zbiór rozwiązań dopuszczalnych

$$\boldsymbol{X}=\Big\{ x\in \boldsymbol{B}^{I(V+J)} \;\Big|\; \sum_{i=1}^{I} x_{vi}^{m}=1; v=\overline{1,V}; \sum_{j=1}^{J} x_{ij}^{\pi}=1,\ i=\overline{1,I};\ j=\overline{1,J};\ v=\overline{1,V};$$

$$Z_{\max}(x)\le Z_{\max}^{\text{limit}};\ P_{time}(x)\ge P_{time}^{\text{limit}};\ R(x)\ge R_{limit};\ C_{\max}(x)\ge C_{\max}^{\text{limit}};\ K(x)<K_{\max};$$

$$\sum_{v=1}^{V} c_{vr}x_{vi}^{m}\le \sum_{j=1}^{J}\tilde{d}_{jr}x_{ij}^{\pi};\ \sum_{i=1}^{I}\sum_{j=1}^{J}\vartheta_j x_{ij}^{\pi}\ge \vartheta_{\min};\ r=\overline{1,R};$$

$m_v \to m_l$ dla $(v, l)\in K\Big\}$, gdzie $\boldsymbol{B}=\{0,1\}$

2) F – kryterium wektorowe

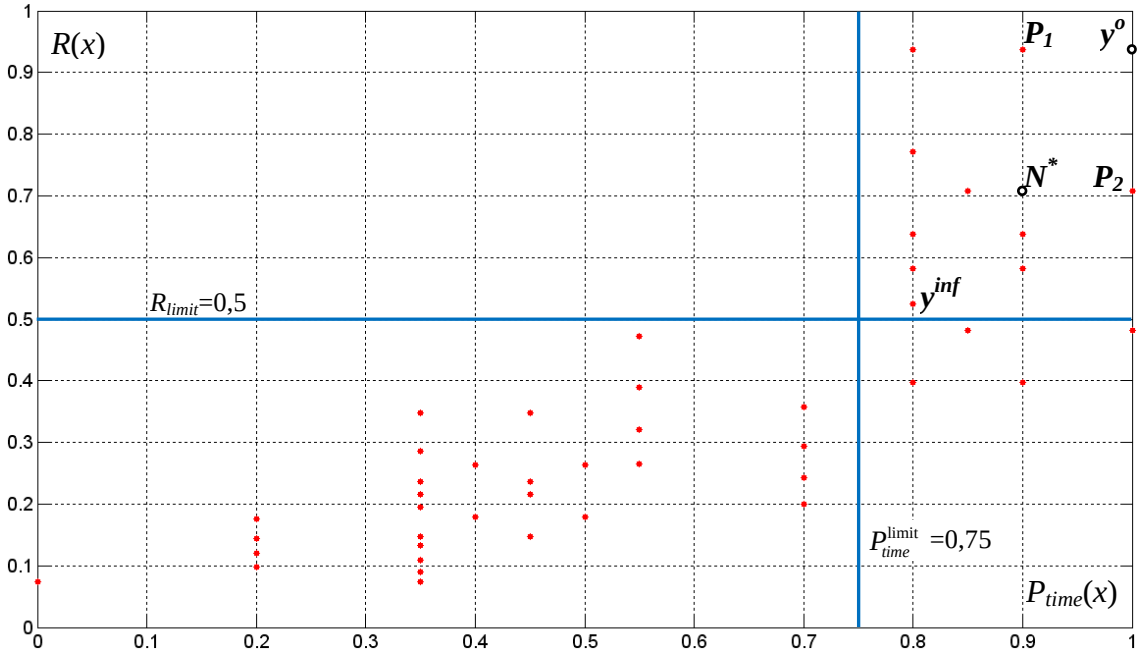
$$F: X \to R^2$$

gdzie $F(x) = [P_{time}(x), R(x)]^T$ dla $x\in \boldsymbol{X}$

3) P – relacja klasy „$\ge$” w R^2

Do modelowania powyższej sytuacji konfliktowej zastosowano teorię optymalizacji wielokryterialnej [7]. W niniejszym opracowaniu przyjęto system pojęć i oznaczeń z prac [6, 153]. Formuła zagadnienia optymalizacji wektorowej zależy w dużym stopniu od preferencji decydenta. Na ogół jest on zainteresowany podjęciem decyzji najlepszej z punktu widzenia wszystkich kryteriów cząstkowych. W sytuacji konfliktowej jest to zazwyczaj niemożliwe, dlatego rozwiązanie zaistniałej sytuacji decyzyjnej wymaga rozważenia zbioru wyników, w którym można porównywać oceny przydziałów modułów do komputerów.

Na rysunku 3.12 przedstawiono zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ oraz $P_{time}(x)$ w odniesieniu do przypadku analizowanego w podrozdziale 3.2. Wartości $P_{time}(x)$ obliczono dla rozkładu empirycznego. Dla $I^V J^I$=128 możliwych przydziałów całkowitoliczbowych uzyskano 45 ocen, co świadczy o tym, że dla różnych przydziałów uzyskano identyczne oceny. Natomiast przydziałów binarnych jest $2^{VI}2^{IJ}$ =16 384. Rozważając naniesione na rysunku ograniczenia odnośnie $R(x)$ i $P_{time}(x)$, otrzymuje się zbiór 10 ocen spełniających warunki zagadnienia (3.10). Oznaczono punkt idealny y^o, punkt antyidealny y^{inf}, punkt nadir N^* oraz poszukiwane oceny niezdominowane P_1, P_2.



Rys. 3.12. Zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ oraz $P_{time}(x)$ w odniesieniu do instancji V=5, I=2, J=2 w przypadku przydziałów kodowanych całkowitoliczbowo
Źródło: opracowanie własne.

Niech F jest funkcją kryterium (wskaźnikiem jakości, kryterium wektorowym), taką że $F : \boldsymbol{A} \to \boldsymbol{P}$, gdzie:

$\boldsymbol{A}$ – przestrzeń rozwiązań, $\boldsymbol{X} \subset \boldsymbol{A}$,

$\boldsymbol{P}$ – przestrzeń ocen wariantów, dalej $\boldsymbol{P} = \boldsymbol{R}^N$,

$F(x)$ – liczbowa ocena rozwiązania x,

$F(x) = [F_1(x),...,F_n(x),...,F_N(x)]^T \in \boldsymbol{R}^N$,

$F_n(x) \in \boldsymbol{R}$ dla $n = \overline{1, N}$.

Zbiorem wyników $\boldsymbol{Y}$ nazywa się obraz zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$ dla funkcji *F*, co zapisuje się w następującej postaci:

$$\boldsymbol{Y} = F(\boldsymbol{X}) = \{ y \in \boldsymbol{R}^{\mathbf{N}} \mid y = F(x),\ x \in \boldsymbol{X} \} \tag{3.11}$$

Twierdzenie 3.1

Jeżeli zbiór dopuszczalnych przydziałów modułów programu do komputerów $\boldsymbol{X}$ jest określony za pomocą zależności (2.10) oraz wartości funkcji kryterium są wyznaczone za pomocą następującej zależności:

$$F(x) = [P_{time}(x), R(x)]^T,\ F(x) \in \boldsymbol{R}^2 \text{ dla } x \in \boldsymbol{X}, \tag{3.12}$$

to zbiór wyników $\boldsymbol{Y}$ jest zbiorem skończonym i ograniczonym.

Dowód: Zbiór dopuszczalnych przydziałów modułów do komputerów $\boldsymbol{X}$, określony za pomocą zależności (2.10), nie jest zbiorem pustym, gdyż na mocy twierdzenia 2.1 zawiera $I^V J^I$ elementów (*V* i *J* są liczbami ograniczonymi). Obraz zbioru rozwiązań dopuszczalnych także zawiera skończoną liczbę elementów.

Miara skutecznej realizacji zadań w terminach posiada ograniczenia dolne i górne ze względu na zależność umożliwiającą wyznaczanie jej wartości.

Dostępność systemu, ze względu na wykładniczy charakter zależności ją opisujący, również posiada ograniczenia dolne i górne.

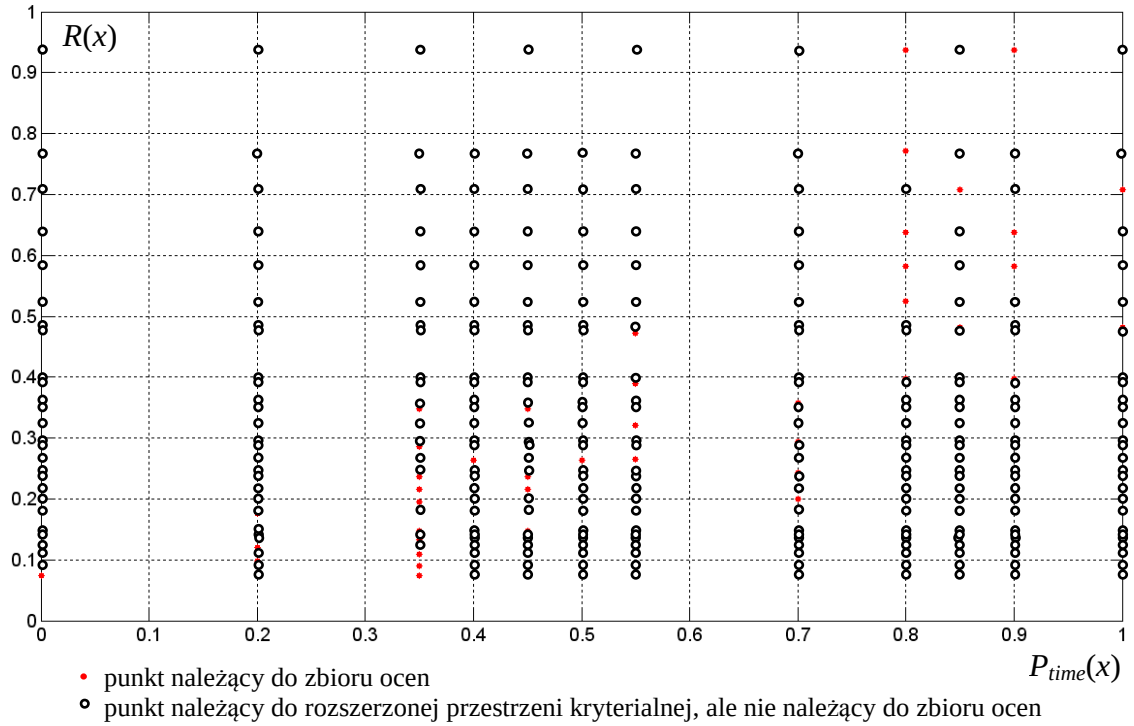
Zbiór wyników $\boldsymbol{Y}$ jest więc zbiorem skończonym i ograniczonym, co kończy dowód. ■

Niech $\boldsymbol{Y_n} = F_n(\boldsymbol{X}) = \{ y_n \in \boldsymbol{R} \mid y_n = F_n(x),\ x \in \boldsymbol{X} \}$ definiuje zbiór wartości *n*-tego kryterium cząstkowego. *Domknięciem zbioru* $\boldsymbol{Y_n}$ jest najmniejszy zbiór domknięty $\tilde{\boldsymbol{Y}}_n$ zawierający zbiór $\boldsymbol{Y_n}$. Zazwyczaj zachodzi $\tilde{\boldsymbol{Y}}_{\boldsymbol{n}} = \boldsymbol{Y_n}$.

Rozszerzoną przestrzenią kryterialną nazywamy iloczyn kartezjański domknięć zbiorów wartości kryteriów cząstkowych:

$$\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}_1 \times ... \times \tilde{\boldsymbol{Y}}_n \times ... \times \tilde{\boldsymbol{Y}}_N .$$

Na rysunku 3.13 przedstawiono przykładowy zbiór ocen i rozszerzoną przestrzeń kryterialną dla zbioru przydziałów dopuszczalnych, określonego za pomocą zależności (2.10). Funkcję kryterium zdefiniowano za pomocą formuły (3.12). Rozszerzona przestrzeń kryterialna jest również zbiorem skończonym i ograniczonym, gdyż zbiory $\tilde{\boldsymbol{Y}}_n = \boldsymbol{Y}_n$ dla $n = \overline{1,2}$ są zbiorami skończonymi i ograniczonymi.



Rys. 3.13. Przykładowy zbiór ocen dopuszczalnych przydziałów modułów do komputerów
Źródło: opracowanie własne.

Wielokryterialna optymalizacja przydziałów w sformułowanym zagadnieniu (3.10) jest związana z porządkowaniem przestrzeni $\boldsymbol{R}^2$ za pomocą relacji dominowania $P \subset \boldsymbol{R}^2 \times \boldsymbol{R}^2$. Dla zbioru ocen $\boldsymbol{Y} \subset \boldsymbol{R}^2$ można analizować zbiór zawierający wszystkie relacje dominowania $D(\boldsymbol{Y}) = 2^{Y \times Y}$ i wybrać z nich tę, która najbardziej odpowiada rozważanej sytuacji decyzyjnej.

Przyjmuje się, że decydent jest zainteresowany wyznaczeniem takiego przydziału modułów do komputerów, który maksymalizuje miarę ukończenia zadań w terminach oraz maksymalizuje dostępność systemu podczas wykonywania zadań.

Relacją Pareto $R_{\geq}$ typu „$\geq$” w zbiorze ocen $\boldsymbol{Y} \subset \boldsymbol{R}^N$ nazywa się zbiór wszystkich par elementów zbioru ocen (*y*, *z*), takich że $y_n \geq z_n$ dla $n = \overline{1, N}$, co zapisuje się jako:

$$R_{\geq} = \{(y, z) \in \boldsymbol{Y} \times \boldsymbol{Y} \mid y_n \geq z_n \text{ dla } n = \overline{1, N}\},$$

gdzie:

$y = [y_1, ..., y_n, ..., y_N]^T \in \boldsymbol{Y}$, $z = [z_1, ..., z_n, ..., z_N]^T \in \boldsymbol{Y}$.

Relacja Pareto $R_{\geq}$ jest generowana przez stożek $\boldsymbol{\Lambda}_{\geq} \subset \boldsymbol{R}^N$, który jest określony w następujący sposób [7]:

$$\boldsymbol{\Lambda}_{\geq} = \{\lambda = [\lambda_1, ..., \lambda_n, ..., \lambda_N]^T \in \boldsymbol{R}^N \mid \lambda_n \leq 0, n = \overline{1, N}\}.$$

Ponieważ minimalizację dowolnego kryterium cząstkowego można zamienić na maksymalizację tego kryterium ze zmienionym znakiem, to w dalszych rozważaniach, bez odstępowania od ogólności analizy, przyjmuje się relację Pareto $R_{\geq}$.

Rozwiązaniem dominującym x^d zadania optymalizacji wielokryterialnej $(\boldsymbol{X}, F, R_{\geq})$ nazywamy rozwiązanie dopuszczalne $x^d \in \boldsymbol{X}$, takie że:

$$F_n(x^d) = \sup_{x \in \mathrm{X}} F_n(x) \text{ dla } n = \overline{1, N}. \tag{3.13}$$

Dla ograniczonego zbioru ocen warunek (3.13) można zmienić następująco:

$$F_n(x^d) = \max_{x \in \mathrm{X}} F_n(x) \text{ dla } n = \overline{1, N}. \tag{3.14}$$

Wynikiem dominującym y^d nazywamy obraz rozwiązania dominującego. Zbiór alternatyw dominujących oznaczamy jako $\boldsymbol{X}_D^R$, a zbiór wyników dominujących – jako $\boldsymbol{Y}_D^R$. Punkt y^d na rysunku 3.13 to wynik dominujący w zbiorze ocen przydziałów. Własnością wyniku y^d jest dominowanie pozostałych ocen w sensie przyjętej relacji porządku. Odpowiada to sytuacji decyzyjnej, w której, ze względu na znane oceny, decydent wybiera tylko jedną ocenę.

Zbiór wyników dominujących $\boldsymbol{Y}_{\boldsymbol{D}}^{\geq}$ w sensie relacji Pareto $R_{\geq}$ w zbiorze ocen oraz zbiór elementów największych $\boldsymbol{Y}_{\sup}^{\geq}$ w uporządkowanym zbiorze ocen $(\boldsymbol{Y}, R_{\geq})$ zawierają identyczne elementy, co zapisujemy, jak niżej:

$$\boldsymbol{Y}_{\boldsymbol{D}}^{\geq} = \boldsymbol{Y}_{\boldsymbol{sup}}^{\geq}$$

Twierdzenie 3.2

Istnieje co najwyżej jeden wynik dominujący w sensie relacji Pareto w zbiorze ocen $\boldsymbol{Y}$ dla funkcji kryterium F, określonej za pomocą zależności (3.12), oraz dla zbioru rozwiązań dopuszczalnych $\boldsymbol{X}$, wyznaczonego za pomocą zależności (2.10).

Dowód: Zbiór uporządkowany może zawierać co najwyżej jeden element największy [127]. Zatem dla każdego zagadnienia polioptymalizacji $(\boldsymbol{X}, F, R)$ istnieje co najwyżej jeden wynik dominujący w sensie relacji porządku R. Relacja Pareto jest relacją porządku, dlatego istnieje co najmniej jeden wynik dominujący w sensie tej relacji. Natomiast przydziałów dominujących może być więcej, co kończy dowód. ■

Zadanie wyznaczania dominujących przydziałów zbioru modułów do komputerów dogodnie jest sformułować w postaci przedstawionej w (3.15).

Wadą przydziałów dominujących w sensie Pareto jest fakt, że w zadaniu optymalizacji wektorowej (3.15) dla niektórych zestawów danych wejściowych, warianty te mogą nie istnieć [20]. Proponuje się zatem tak zmodyfikować zadanie optymalizacji wielokryterialnej $(\boldsymbol{X}, F, R_{\geq})$ opisane za pomocą zależności (3.15), aby poszukiwać reprezentacji rozwiązań optymalnych w sensie Pareto.

Element $x^P \in \boldsymbol{X}$ nazywamy *rozwiązaniem optymalnym w sensie Pareto* wtedy i tylko wtedy, gdy w zbiorze rozwiązań dopuszczalnych $\boldsymbol{X}$ nie istnieje rozwiązanie x, takie że $F_n(x) \geq F_n(x^P)$ dla $n = \overline{1, N}$ oraz istnieje $l \in \overline{1, N}$, takie że $F_l(x) > F_l(x^P)$.

Dla danych: $M = \{m_1, ..., m_v, ..., m_V\}$, $\Pi = \{\pi_1, ..., \pi_j, ..., \pi_J\}$, $W = \{w_1, ..., w_i, ..., w_I\}$, $T = [t_{vj}]_{V \times J}$, $\boldsymbol{\tau} = [\tau_{\boldsymbol{vu}}]_{\boldsymbol{V} \times \boldsymbol{V}}$, $\boldsymbol{\Lambda} = \{\lambda_1, ..., \lambda_{\boldsymbol{j}}, ..., \lambda_{\boldsymbol{J}}\}$, $d = \{d_1, ..., d_v, ..., d_V\}$, diagram przepływu danych $\boldsymbol{G_F}$

należy wyznaczyć reprezentację przydziałów $\widetilde{\boldsymbol{X}}_D^{\geq}$ zbioru rozwiązań dominujących $\boldsymbol{X}_D^{\geq}$ zagadnienia optymalizacji wielokryterialnej

$$(\boldsymbol{X}, F, R_{\geq}), \tag{3.15}$$

gdzie:

$\boldsymbol{X}$ – zbiór rozwiązań dopuszczalnych określony za pomocą zależności (2.10),

F – kryterium jakości określone za pomocą zależności (3.12),

$R_{\geq}$ – relacja Pareto w zbiorze ocen $\boldsymbol{Y} \subset \boldsymbol{R}^2$.

Element $y^N \in \boldsymbol{Y}$ nazywamy wynikiem niezdominowanym w sensie dowolnej relacji $R \in D(\boldsymbol{Y})$ w zbiorze $\boldsymbol{Y}$, jeżeli nie istnieje $y \in \boldsymbol{Y}$, $y^N \neq y$, takie że $(y, y^N) \in R$ [8].

W literaturze warianty optymalne w sensie Pareto również nazywane są Pareto--optymalnymi, polioptymalnymi, efektywnymi lub sprawnymi. Wynik efektywny $y^P = F(x^P)$ jest szczególnym przypadkiem wyniku niezdominowanego dla dowolnej relacji. Dla relacji Pareto element niezdominowany jest wynikiem sprawnym [20].

Niech zbiór alternatyw polioptymalnych oznaczony będzie jako $\boldsymbol{X}_N^{\geq}$, a zbiór wyników efektywnych – jako $\boldsymbol{Y}_N^{\geq} = F(\boldsymbol{X}_N^{\geq})$. Zagadnienie wyznaczania sprawnych przydziałów dogodnie jest sformułować w postaci przedstawionej, jak niżej.

Dla danych: $\boldsymbol{M} = \{\boldsymbol{m}_1, ..., \boldsymbol{m}_v, ..., \boldsymbol{m}_V\}$, $\boldsymbol{\Pi} = \{\pi_1, ..., \pi_j, ..., \pi_J\}$, $\boldsymbol{W} = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_i, ..., \boldsymbol{w}_I\}$,
$\boldsymbol{T} = [\boldsymbol{t}_{vj}]_{V \times J}$, $\boldsymbol{\tau} = [\tau_{vu}]_{V \times V}$, $\boldsymbol{\Lambda} = \{\lambda_1, ..., \lambda_j, ..., \lambda_J\}$, $\boldsymbol{d} = \{\boldsymbol{d}_1, ..., \boldsymbol{d}_v, ..., \boldsymbol{d}_V\}$, zadanego
diagramu przepływu danych $\boldsymbol{G_F}$
należy wyznaczyć reprezentację przydziałów $\tilde{\boldsymbol{X}}_N^{\geq}$ zbioru rozwiązań
niezdominowanych $\boldsymbol{X}_N^{\geq}$ zadania

$$(X, F, R_{\geq}), \tag{3.16}$$

gdzie:
X – zbiór rozwiązań dopuszczalnych określony za pomocą zależności (2.10),
F – kryterium jakości określone za pomocą zależności (3.12),
$R_{\geq}$ – relacja Pareto w zbiorze ocen $\boldsymbol{Y} \subset \boldsymbol{R}^2$.

Warianty optymalne w sensie Pareto, w odróżnieniu od wariantów dominujących, istnieją zazwyczaj w zagadnieniach optymalizacji wielokryterialnej. Dla zagadnieniu (3.16) można pokazać, że jeżeli $V \geq 2$ oraz $J \geq 1$, to istnieją co najmniej dwa rozwiązania optymalne w sensie Pareto [20].

3.4. Wyznaczanie rozwiązań leksykograficznych

Istotną klasę rozwiązań optymalnych w sensie Pareto stanowią alternatywy leksykograficzne. System preferencji leksykograficznych odpowiada sytuacji, w której N wskaźników ustawiono według pewnej zasady hierarchii kryteriów od najważniejszego F_1 do najmniej ważnego F_N. Dlatego warianty leksykograficzne są także nazywane rozwiązaniami hierarchicznymi.

Relacją leksykograficzną $\boldsymbol{L}$ nazywamy zbiór par wyników $(y,z)\in \boldsymbol{Y}\times\boldsymbol{Y}$, takich że $y=z$, lub takich że istnieje pewien numer l skalarnego kryterium cząstkowego, dla którego jest spełniona ostra nierówność $y_l < z_l$, a jeżeli $l>1$, to dla $k<l$ zachodzi $y_k = z_k$.

Relację leksykograficzną można opisać zatem następująco:

$$\boldsymbol{L} = \left\{ (y,z)\in \boldsymbol{Y}\times\boldsymbol{Y} \mid y=z \quad \text{lub} \quad \left(\underset{l\in\overline{1,N}}{\exists} y_l < z_l \text{ oraz } y_k = z_k \quad \text{dla} \quad k<l\right) \right\} \tag{3.15}$$

Kluczowy związek między relacją Pareto a relacją leksykograficzną jest taki, że ta pierwsza jest podzbiorem drugiej [9]. W [20] wykazano, że dla relacji leksykograficznej wynik niezdominowany jest oceną dominującą, a także że wynik dominujący jest oceną niezdominowaną, co prowadzi do zależności:

$$\boldsymbol{Y}_D^L = \boldsymbol{Y}_N^L$$

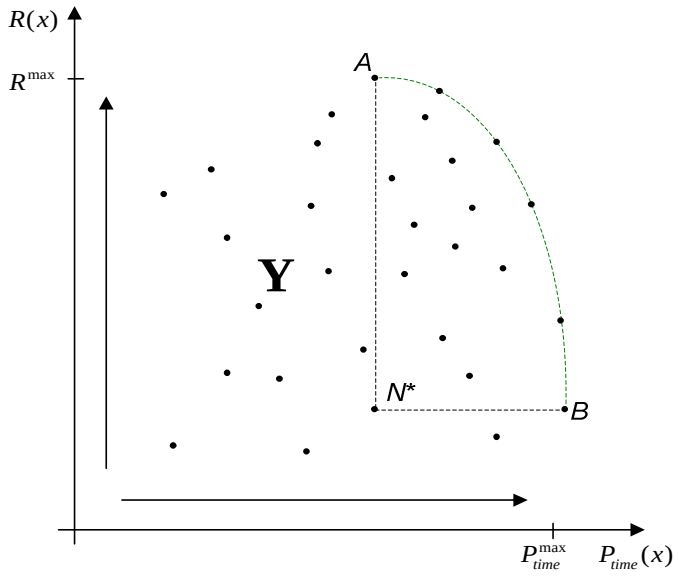
Rozwiązaniem leksykograficznym nazywamy wariant niezdominowany (dominujący) w sensie relacji leksykograficznej [20].

Niech $\boldsymbol{Y}^L$ oznacza zbiór wyników hierarchicznych, a $\boldsymbol{X}^L$ - zbiór rozwiązań leksykograficznych. Prawdziwe są następujące zależności:

$$\boldsymbol{Y}^L = \boldsymbol{Y}_D^L = \boldsymbol{Y}_N^L \tag{3.16}$$

$$\boldsymbol{X}^L = \boldsymbol{X}_D^L = \boldsymbol{X}_N^L \tag{3.17}$$

Na rysunku 3.14 przedstawiono przykładowy zbiór wyników ***Y***, w którym są poszukiwane oceny leksykograficzne.



Rys. 3.14. Wyniki leksykograficzne w zbiorze ocen ***Y***
Źródło: opracowanie własne.

Jeżeli preferowane jest kryterium $P_{time}(x)$ przed $R(x)$, to wynikiem hierarchicznym jest punkt *B*. W przeciwnym przypadku, ocenę leksykograficzną stanowi punkt *A*. Punkty hierarchiczne, są charakterystycznymi, „skrajnymi" punktami zbioru Pareto dla różnych preferencji leksykograficznych.

Oceny hierarchiczne dla różnych preferencji leksykograficznych między kryteriami cząstkowymi wyznaczają *punkt nadir* N^*. W tym przypadku oceny, które dominują punkt N^* w sensie relacji $R_{\geq}$, nie są zdominowane przez oceny leksykograficzne. Nazywa się je ocenami *zadowalającymi* w odniesieniu do punktu *nadir*. Z kolei wynik zdominowany przez punkt *nadir* nie jest optymalny w sensie Pareto i jest zdominowany przez oceny hierarchiczne.

Zagadnienie wyznaczania leksykograficznych przydziałów modułów do komputerów można sformułować następująco:

Dla danych: $\boldsymbol{M}=\{m_1,...,m_v,...,m_V\}$, $\boldsymbol{\Pi}=\{\pi_1,...,\pi_j,...,\pi_J\}$, $\boldsymbol{W}=\{w_1,...,w_i,...,w_I\}$,

$\boldsymbol{T}=[t_{vj}]_{V\times J}$, $\boldsymbol{\tau}=[\tau_{vu}]_{V\times V}$, $\boldsymbol{\Lambda}=\{\lambda_1,...,\lambda_j,...,\lambda_J\}$, $\boldsymbol{d}=\{d_1,...,d_v,...,d_V\}$, zadany diagram przepływu danych $\boldsymbol{G_F}$

należy wyznaczyć reprezentację przydziałów $\tilde{\boldsymbol{X}}^L$ zbioru rozwiązań leksykograficznych $\boldsymbol{X}^L$ zadania

$$(\boldsymbol{X}, F, L), \tag{3.18}$$

gdzie:

$\boldsymbol{X}$ – zbiór rozwiązań dopuszczalnych określony za pomocą zależności (2.10),

F – kryterium jakości określone za pomocą zależności (3.10),

L – relacja leksykograficzna w zbiorze ocen $\boldsymbol{Y}\subset\boldsymbol{R}^2$.

Twierdzenie 3.3

Istnieje co najmniej jeden wynik leksykograficzny oraz istnieją co najmniej dwa rozwiązania hierarchiczne dla zagadnienia optymalizacji wektorowej (3.18).

Dowód: Zbiór wariantów dopuszczalnych $\boldsymbol{X}$ w zadaniu (3.18) nie jest zbiorem pustym na mocy twierdzenia 2.1. Ponadto, na podstawie twierdzenia 3.1, zbiór ocen $\boldsymbol{Y}$ w zagadnieniu (3.18) jest zbiorem ograniczonym. Zatem istnieje co najmniej jeden wynik leksykograficzny oraz istnieją co najmniej dwa symetryczne rozwiązania hierarchiczne, co kończy dowód. ■

3.5. Wyznaczanie rozwiązań kompromisowych

Alternatywy niezdominowane, rozwiązania dominujące oraz warianty leksykograficzne cechuje szereg wad z praktycznego punktu widzenia. Zbiór wariantów dominujących często jest zbiorem pustym, a z kolei zbiór rozwiązań niezdominowanych na ogół jest bardzo liczny. Poszukiwania alternatyw hierarchicznych prowadzą do wyboru wariantów, dla których preferowane jest tylko jedno kryterium cząstkowe.

Na tym tle ciekawą ideą jest poszukiwanie rozwiązania kompromisowego. „Kompromis" między preferencjami dotyczącymi kryteriów cząstkowych $F_1,...,F_n,...,F_N$ osiąga się przez minimalizację odległości od punktu dopuszczalnego do punktu idealnego (utopijnego) y^o, który uważany jest za wynik najbardziej pożądany.

Punktem idealnym y^o dla zadania optymalizacji wielokryterialnej $(\boldsymbol{X},\boldsymbol{Y},R_{\geq})$ nazywamy punkt:

$$y^o = \left[y_1^o,...,y_n^o,...,y_N^o\right]^T,$$

taki, że:

$$y_n^o = \sup_{x\in\mathbf{X}} F_n(x) \text{ dla } n=\overline{1,N} \tag{3.19}$$

Punkt idealny jest punktem charakterystycznym zadania optymalizacji wektorowej. Należy do rozszerzonej przestrzeni kryterialnej, ale może nie należeć do zbioru wyników. Dlatego nazywany jest „utopijnym", gdyż jest swego rodzaju celem, do którego metody poszukujące rozwiązań powinny dążyć, aczkolwiek nie zawsze są w stanie go osiągnąć. Jeśli punkt idealny należy do zbioru ocen, to jest on jednocześnie wynikiem dominującym w sensie relacji $R_{\geq}$.

Punkt idealny na rysunku 3.12 nie należy do zbioru wyników i w tym przypadku może być nazywany utopijnym, gdyż nie istnieje rozwiązanie umożliwiające osiągnięcie celu. W zagadnieniach optymalizacji przydziałów występują górne i dolne dodatnie ograniczenia zbioru wartości funkcji $P_{time}(x)$ i $R(x)$, które umożliwiają wyznaczenie współrzędnych punktu idealnego, co można zapisać następująco:

$$y_n^o = \max_{x\in\mathbf{X}} F_n(x) \text{ dla } n=\overline{1,N}. \tag{3.20}$$

Zakłada się, że znane są idealne wartości poszczególnych kryteriów składowych, tzn. że w przestrzeni kryteriów określony jest *punkt idealny* $y^o = [y_1^o, y_2^o,..., y_N^o]^T$, do

którego należy dążyć poszukując rozwiązania problemu. Ocenom należącym do zbioru dopuszczalnych wartości kryteriów można przypisać liczby, które w sensie przyjętej metryki, będą wyrażać ich odległości od punktu y^o. Wówczas zadanie programowania wielokryterialnego zastępuje się zadaniem jednokryterialnym, w którym minimalizuje się odległość między punktami $y=[y_1, y_2,..., y_N]^T$ oraz y^o. Przykładowe postaci metryk, za pomocą których mierzy się odległość między punktami y i y^o określone są za pomocą wzoru [7]:

$$Q_p(y-y^o)=\left\|y-y^o\right\|_p=\left[\sum_{n=1}^{N}\left|y_n-y_n^o\right|^p\right]^{\frac{1}{p}}, \tag{3.21}$$

gdzie $1\le p\le\infty$.

Przyjmując wartości liczbowe, otrzymuje się szczególne postacie metryk. Dla $p=1$ wyznacza się metrykę prostokątną [141], jak niżej:

$$Q_1(y-y^o)=\sum_{n=1}^{N}\left|y_n-y_n^o\right| \tag{3.22}$$

Metrykę Euklidesa uzyskuje się dla $p=2$ następująco:

$$Q_2(y-y^o)=\sqrt{\sum_{n=1}^{N}\left|y-y^o\right|^2} \tag{3.23}$$

Dla $p=\infty$ otrzymuje się metrykę Czebyszewa:

$$Q_\infty(y-y^o)=\max_n\left|y_n-y_n^o\right| \tag{3.24}$$

Aby uniknąć dylematu uwzględniania różnych jednostek miar, podczas wyznaczania wartości normy $\left\|y-y^o\right\|$ dokonuje się normalizacji kryteriów cząstkowych. *Klasyczny* sposób normalizacji, polega na podzieleniu oceny cząstkowej $F_n(x)$ przez wartość składowej punktu idealnego y_n^o, zgodnie z następującą formułą:

$$\overline{F}_n(x)=\frac{F_n(x)}{y_n^o},\quad y_n^o>0,\ n=\overline{1,N}. \tag{3.25}$$

Punkt idealny $\bar{y}^o$, po zastosowaniu normalizacji klasycznej, ma jednostkowe współrzędne $\bar{y}^o=[1,...,1]^T\in R^N$. Jeśli y jest punktem idealnym, to współrzędne znormalizowanego wektora (y^o-y), które nazywamy *znormalizowanymi odchyłkami* $\Delta\bar{y}_n=\bar{y}_n^o-\bar{y}_n$ dla $n=\overline{1,N}$, są równe zeru.

Interesującą alternatywą dla normalizacji klasycznej jest normalizacja przedziałowa. W *normalizacji przedziałowej* przy maksymalizacji kryteriów cząstkowych wykorzystywana jest następująca zależność:

$$\overline{\overline{F}}_n(x) = \frac{F_n(x)}{y_n^o - y_n^{\min}}, \quad y_n^o \neq y_n^{\min},\ n = \overline{1, N}, \tag{3.26}$$

przy czym $y_n^{\min} = \inf_{x \in \mathbf{X}} F_n(x)$ dla $n = \overline{1, N}$.

Punkt idealny $\bar{y}^o$, po zastosowaniu normalizacji przedziałowej, ma jednostkowe współrzędne $\bar{y}^o = [\bar{y}_1^o, ..., \bar{y}_n^o, ..., \bar{y}_N^o]^T \in R^N$, gdzie $\bar{y}_n^o = \frac{y_n^o}{y_n^o - y_n^{\min}}$. Należy zauważyć, że wartość względnych odchyłek poszczególnych kryteriów jest znormalizowana w [0,1]. Normalizację przedziałową można przeprowadzić w odniesieniu do stosunkowo szerokiej klasy zadań optymalizacji przydziału modułów do komputerów. Sposoby normalizacji przestrzeni kryterialnej dla zadań optymalizacji wektorowej, w których minimalizuje się kryteria cząstkowe, przedstawiono w [20].

Poszukiwania rozwiązań kompromisowych zachodzą na ogół w znormalizowanym klasycznie zbiorze ocen, a nie w zbiorze ocen. Ponieważ wartość funkcji kompromisowej zależy od wariantu idealnego x, to problem wyznaczania przydziału kompromisowego $x^*(p)$ dla normalizacji klasycznej można sformułować w postaci następującego zagadnienia optymalizacji jednokryterialnej:

Dla danych: $\boldsymbol{M} = \{m_1, ..., m_v, ..., m_V\}$, $\boldsymbol{\Pi} = \{\pi_1, ..., \pi_j, ..., \pi_J\}$, $\boldsymbol{W} = \{w_1, ..., w_i, ..., w_I\}$, $\boldsymbol{T} = [t_{vj}]_{V \times J}$, $\boldsymbol{\tau} = [\tau_{vu}]_{V \times V}$, $\boldsymbol{\Lambda} = \{\lambda_1, ..., \lambda_j, ..., \lambda_J\}$, $\boldsymbol{d} = \{d_1, ..., d_v, ..., d_V\}$, zadany diagram przepływu danych $\boldsymbol{G_F}$ do zadania optymalizacji wielokryterialnej $(X, Y, R_{\geq})$
należy wyznaczyć $x^*(p) \in$ X, takie że

$$Q_p(x^*(p)) = \min_{x \in \mathbf{X}} Q_p(x). \tag{3.27}$$

Metoda wyznaczania wariantu kompromisowego zagadnienia optymalizacji wielokryterialnej zalicza się do metod skalaryzacji kryterium wektorowego, gdyż polega na poszukiwaniu rozwiązania pomocniczego zadania optymalizacji ze skalarną funkcją celu za pomocą metod optymalizacji jednokryterialnej.

3.6. Wnioski i uwagi

Do oceny rozwiązań zastosowano dwa kryteria. Pierwszym kryterium jest miara dostępności systemu, a drugim – miara skutecznej realizacji zadań w terminach. Ponieważ istnieje konflikt podczas maksymalizacji powyższych kryteriów, problem alokacji modułów programistycznych systemu zdalnego szkolenia wojskowego sformułowano w postaci zagadnienia optymalizacji dwukryterialnej.

Optymalizacja alokacji zadań umożliwia znaczące zmniejszenie obciążenia newralgicznego komputera poniżej założonego progu. Redukcja obciążenia może przekroczyć nawet 50% wartości obciążenia cechującego system, w którym procesy przydzielono losowo.

Do modelowania logicznej struktury programów rozproszonych, w których wyodrębniono procesy, stosuje się diagram przepływu $\boldsymbol{G_F}$ systemu rozproszonego. Do podstawowych typów struktur diagramu, oprócz sekwencji, należą struktury typu *OR* i *AND*. Wartości miary $P_{time}(x)$ zależą od przyjętego rozkładu prawdopodobieństwa wykonania modułu w strukturze diagramu typu *AND*.

Ponieważ alternatywy niezdominowane, rozwiązania dominujące oraz warianty leksykograficzne cechuje szereg wad z praktycznego punktu widzenia, często zamiast nich poszukuje się rozwiązania kompromisowego. Stosuje się minimalizację odległości między punktami należącymi do zbioru dopuszczalnych wartości kryteriów a punktem idealnym (utopijnym) y^o, który uważany jest za wynik najbardziej pożądany. Punkt ten może jednak nie istnieć w zbiorze ocen.

Wyznaczając przydział kompromisowy, zadanie programowania wielokryterialnego zastępuje się zadaniem jednokryterialnym, w którym minimalizuje się odległość między punktami dopuszczalnymi a y^o. Przykładowe metryki, za pomocą których mierzy się odległość między punktami y i y^o określone są za pomocą (3.21).

Aby uniknąć dylematu uwzględniania różnych jednostek miar, podczas wyznaczania (3.21) dokonuje się klasycznej lub przedziałowej normalizacji kryteriów cząstkowych.

4. METODY WYZNACZANIA ROZWIĄZAŃ OPTYMALNYCH W SENSIE PARETO

Podejście do procesu podejmowania decyzji oparte na poszukiwaniu ekstremum zgodnie ze skalarnym kryterium jakości w wielu zagadnieniach decyzyjnych nie spełnia oczekiwań decydentów. Trudności związane są ze zdefiniowaniem funkcji, która wyrażałaby wymagania stawiane rozwiązaniom „najlepszym".

Rozważając dylemat zakupu sprzętu komputerowego dla systemu zdalnego szkolenia wojskowego, bierze się pod uwagę szereg kryteriów, takich jak cena, wydajność, wielkość zasobów, dostępność na rynku, jakość gwarancji lub serwisu. Rozpatrując każdy z tych wskaźników oddzielnie, można podjąć decyzję, który komputer jest najbardziej wydajny za względu na wybrany test, a który najtańszy. Podejście to nie rozwiązuje jednak problemu wyznaczenia konfiguracji uwzględniającej preferowanie wszystkich kryteriów jednocześnie. Trudno jest ustalić jedną miarę agregującą takie wielkości jak: koszty, wydajność czy jakość serwisu. Jeśli nawet takie kryterium istnieje w świadomości decydenta, to bardzo trudno jest wyprowadzić jego zapis matematyczny.

Optymalizacja wielokryterialna wspomaga podejmowanie złożonych decyzji. Stosowana jest tam, gdzie ocena zależy od wielu kryteriów i polega na znalezieniu kompromisu łączącego „interesy" tychże kryteriów. Wybrane wielkości charakteryzujące jakość potencjalnych decyzji mogą nie być ze sobą ilościowo porównywalne z punktu widzenia podejmującego decyzję.

Zagadnienie optymalizacji wielokryterialnej zazwyczaj zastępuje się zadaniem lub sekwencją zadań jednokryterialnych. W konstruowaniu zastępczego zagadnienia jednokryterialnego wykorzystuje się różnorodne koncepcje. Zadanie wielokryterialne można sprowadzić do skalarnego problemu poprzez wprowadzenie dodatkowego kryterium porządkującego punkty niezdominowane. Alternatywną metodą jest wprowadzenie funkcji agregującej, której argumentami są wartości poszczególnych kryteriów wektorowego wskaźnika jakości [12].

Do interesujących metod polioptymalizacji należą algorytmy ewolucyjne, które realizują operacje na zbiorze rozwiązań, a nie na pojedynczych wariantach. W algorytmach ewolucyjnych stosuje się operacje genetyczne, takie jak selekcja naturalna, krzyżowanie i mutacja. Chromosomy reprezentowane są zazwyczaj za

pomocą złożonych struktur danych. Dopuszcza się stosowanie zaawansowanych technik optymalizacyjnych w roli procedury mutacji, takich jak mutacja tabu [18, 67].

4.1. Przegląd metod optymalizacji wektorowej

W modelu wielokryterialnej optymalizacji dąży się do maksymalizacji lub minimalizacji funkcji wektorowej $F(x)=[F_1(x), F_2(x),\ldots, F_N(x)]$. Ponadto zazwyczaj nałożone są ograniczenia wektorowe, zarówno równościowe jak i nierównościowe, co zapisujemy odpowiednio jako równanie wektorowe $\vec{\mathbf{H}}(x)=0$ oraz $\vec{\mathbf{G}}(x)\geq 0$, gdzie symbol → oznacza wektor.

Metoda sumy ważonej (ang. *weighted sum*) polega na wykorzystaniu funkcji jednokryterialnej utworzonej za pomocą przypisania wagi do każdego kryterium cząstkowego, jak niżej [53]:

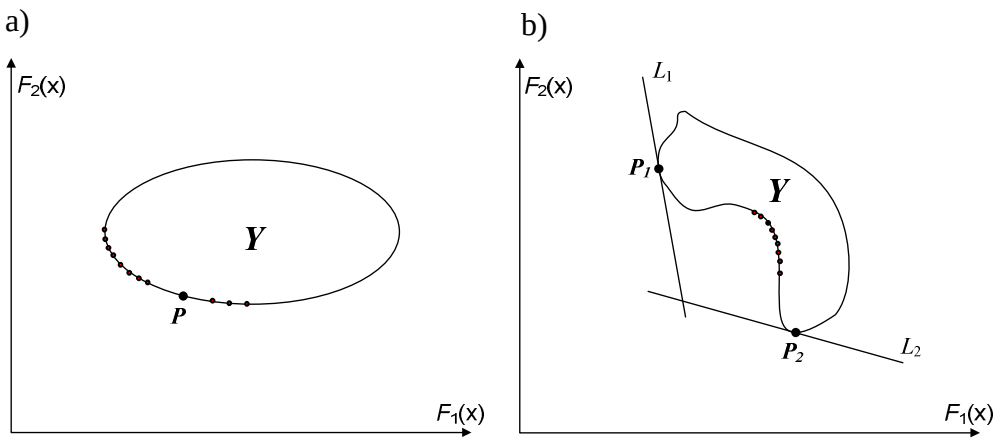
$$F'(x)=\sum_{i=1}^{N} r_i F_i(x), \tag{4.1}$$

przy ograniczeniach: $\vec{\mathbf{H}}(x)=0$, $\vec{\mathbf{G}}(x)\geq 0$, $0\leq r_i\leq 1$, $i=\overline{1,N}$, $\sum_{i=1}^{N} r_i=1$.

Funkcja F' jest liniową kombinacją kryteriów cząstkowych $F_i(x)$. W przypadku optymalizacji dwukryterialnej otrzymuje się $F'(x)=r_1F_1(x)+ r_2F_2(x)$. Wartości wag powinny być takie, aby w układzie współrzędnych F_1, F_2 wyznaczyć dwie przecinające się styczne do zbioru ocen $\boldsymbol{Y}$.

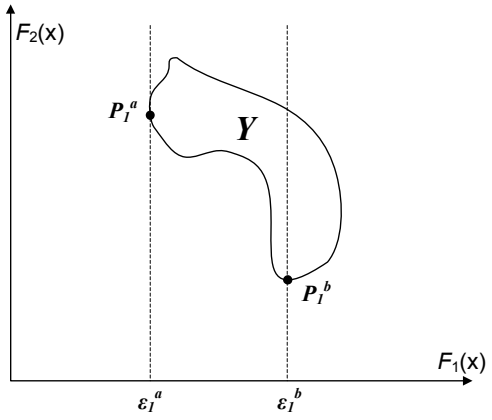
Na rysunku 4.1a rozpatrywana jest minimalizacja kryteriów cząstkowych F_1 i F_2, przy czym zbiór ocen jest zbiorem wypukłym (ang. *convex*). W wyniku optymalizacji zadania (4.1) zostanie wyznaczony punkt P. Punkt ten jest obrazem optymalnego rozwiązania w sensie Pareto dla wag r_1 i r_2. Po zmianie wartości wag r_1 i r_2 można wyznaczyć kolejny punkt, który będzie należał do frontu Pareto. Jeśli zbiór ocen $\boldsymbol{Y}$ jest ciągły i wypukły, to każdy element frontu Pareto może zostać wyznaczony za pomocą zmiany wartości wag r_1 i r_2.

W przypadku przedstawionym na rysunku 4.1b, gdy ciągły zbiór ocen $\boldsymbol{Y}$ nie jest zbiorem wypukłym, zaznaczono punkty, które nie zostaną osiągnięte za pomocą tej metody w odróżnieniu od punktów P_1 i P_2. Inną wadą jest fakt, że w przypadku dużych różnic między wartościami kryteriów cząstkowych konieczne jest zastosowanie normalizacji, co wiąże się ze zwiększeniem kosztu obliczeniowego [53].



Rys. 4.1. Rozwiązania uzyskane za pomocą metody sumy ważonej:
a) zbiór ocen jest wypukły
b) zbiór ocen nie jest wypukły
Źródło: opracowanie własne.

W *metodzie ε-ograniczeń* formułuje się zagadnienie, w którym optymalizacji podlega tylko jedno kryterium cząstkowe, a pozostałe włączone są do ograniczeń [53]. Wyznaczane są górne ograniczenia wartości kryteriów składowych niepodlegających optymalizacji. Optymalizuje się funkcję $F_i(x)$ przy ograniczeniach $F_k(x) \leq \varepsilon_k$, $k=\overline{1,N}$, $k \neq i$ oraz $\vec{\mathbf{H}}(x) = 0$, $\vec{\mathbf{G}}(x) \geq 0$. Na rysunku 4.2 przedstawiono wyniki uzyskane dla dwukrotnej minimalizacji funkcji F_2, przy czym na F_1 nałożone jest ograniczenie górne.



Rys. 4.2. Rozwiązania uzyskane w metodzie ε-ograniczeń
Źródło: opracowanie własne.

Jeśli górnym ograniczeniem funkcji F_1 jest ε_1^a, to rozwiązanie wyznaczone za pomocą tej metody cechuje się oceną będącą punktem P_1^a. W przypadku zmiany górnej wartości ograniczenia funkcji F_1 na ε_1^b, ocena rozwiązania odpowiada punktowi P_1^b.

Zaletą metody jest możliwość stosowania do zbioru ocen wypukłego lub wklęsłego. Wadą natomiast jest konieczność posiadania wiedzy o odpowiednich wartościach ε.

W metodzie *odległości do celu odniesienia* (ang. *distance to reference objective*) dąży się do optymalizacji funkcji [53]:

$$F^{'}(x) = \left[\sum_{n=1}^{N} \left| Z_i - F_i(x) \right|^p \right]^{\frac{1}{p}}, \tag{4.2}$$

przy założeniach: $\vec{\mathbf{H}}(x) = 0$, $\vec{\mathbf{G}}(x) \geq 0$, $1 \leq p < \infty$.

Należy wyznaczyć wartości referencyjne Z_i dla każdego kryterium składowego. Wartości te służą jako współrzędne punktu odniesienia do ustalenia punktów Pareto za pomocą funkcji odległości (4.2). Za pomocą zmiany wartości parametru p wybiera się odpowiednią funkcję odległości. Dla parametru p=1 uzyskane rozwiązania są ekwiwalentne do rozwiązań uzyskanych za pomocą metody sumy ważonej, ale bez normalizacji wartości funkcji kryteriów. Dla p=2 otrzymuje się odległość Euklidesa, a dla $p = \infty$ rozwiązuje się problem Czebyszewa [53].

W przypadku, gdy zbiór ocen nie jest wypukły, a tak jest w rozważanych zagadnieniach polioptymalizacji przydziałów, rozwiązania optymalne mogą nie zostać wyznaczone dla pewnych wartości parametru p. Należy również zwrócić uwagę na wybór punktu odniesienia. Niewłaściwie dobrane współrzędne tego punktu mogą nie prowadzić do zbieżności w kierunku frontu Pareto w wyniku optymalizacji.

Metoda *ważonych metryk* jest podobna do metody *programowania celowego*. Różnica polega na tym, że każde kryterium skalarne jest normalizowane z uwzględnieniem wagi [53]:

$$F^{'}(x) = \left[\sum_{n=1}^{N} w_i \left| Z_i - F_i(x) \right|^p \right]^{\frac{1}{p}}, \tag{4.3}$$

przy ograniczeniach: $\vec{\mathbf{H}}(x) = 0$, $\vec{\mathbf{G}}(x) \geq 0$, $1 \leq p < \infty$, $0 \leq w_i \leq 1$, $i = \overline{1, N}$, $\sum_{i=1}^{N} w_i = 1$.

Metoda Bensona podobna jest do metody metryk ważonych. Różnica polega na tym, że punkt referencyjny musi należeć do zbioru rozwiązań dopuszczalnych. Dąży się do maksymalizacji funkcji:

$$F^{''}(x) = \sum_{i=1}^{N} \max(0, Z_i - F_i(x)), \tag{4.4}$$

przy ograniczeniach: $\vec{\mathbf{H}}(x) = 0$, $\vec{\mathbf{G}}(x) \geq 0$, $F_i(x) \leq Z_i$, $i = \overline{1, N}$.

Możliwe jest wyznaczenie frontu Pareto dla niewypukłego zbioru rozwiązań. Wadą jest konieczność oszacowania odpowiednich wartości $\boldsymbol{Z}$ i konieczność posiadania dodatkowych informacji o przestrzeni rozwiązań dopuszczalnych.

Metodę programowania celowego, zwaną również *metodą punktu idealnego*, stosuje się, gdy w przestrzeni kryteriów określony jest *punkt* (*wektor*) *idealny* $y^o = [y_1^o, y_2^o, ..., y_N^o]^T$, do którego należy dążyć, poszukując rozwiązania [53]. Punktom należącym do zbioru ocen można przypisać ich odległości od punktu y^o w sensie przyjętej metryki. Wówczas zadanie programowania wielokryterialnego zastępuje się zadaniem jednokryterialnym, w którym minimalizuje się odległość między $y = [y_1, y_2, ..., y_N]^T$ a y^o.

Warto podkreślić, ze punkt idealny, którego współrzędne są najbardziej pożądanymi wartościami kryteriów składowych, nie musi należeć do zbioru ocen. Jeśli nie można wyznaczyć punktu idealnego, to zazwyczaj rozwiązuje się niezależnie sekwencję N zadań jednokryterialnych, minimalizując kryterium składowe, a otrzymane rozwiązania przyjmuje się za współrzędne punktu idealnego.

Odmienny wariant programowania celowego polega na wyborze w przestrzeni kryteriów punktu antyidealnego, dla którego wartości kryteriów składowych są „najgorsze” z możliwych, w sensie relacji dominowania. Poszukiwanie rozwiązania polega na maksymalizacji odległości od tego punktu, w sensie przyjętej metryki.

W *metodzie leksykograficznej* porządkowane są kryteria składowe wg ich ważności. Następnie w pierwszym etapie minimalizuje się (lub maksymalizuje) wartość najważniejszego kryterium cząstkowego przy ograniczeniach nałożonych na wartości pozostałych kryteriów. W kolejnych etapach minimalizuje się (lub maksymalizuje) wartości coraz to mniej ważnych kryteriów składowych, przy ograniczeniach odnoszących się do pozostałych kryteriów skalarnych. Za rozwiązanie zagadnienia wielokryterialnego przyjmuje się wariant optymalny wyznaczony dla ostatniego zadania jednokryterialnego. Metoda wymaga arbitralnego określenia ważności kryteriów. Zmiana priorytetu kryteriów prowadzi zazwyczaj do odmiennych wyników.

Zadanie optymalizacji wielokryterialnej jest jednym z trudniejszych zagadnień optymalizacji. W literaturze przedmiotu omówiono także metody iteracyjne, metody bazujące na logice rozmytej, a także metody meta-heurystyczne [53, 113, 160, 161]. Szczególnie intensywny rozwój cechuje metody ewolucyjne.

4.2. Wielokryterialne algorytmy ewolucyjne

Koncepcja opracowanego w 1985 przez Schaffera pierwszego wielokryterialnego algorytmu genetycznego o nazwie VEGA (ang. *Vector Evaluated Genetic Algorithm*) polega na podziale populacji na *N* podpopulacji o jednakowej liczbie osobników, gdzie *N* odpowiada liczbie kryteriów cząstkowych. Dla każdej podpopulacji definiowana jest funkcja sprawności w oparciu o przypisane do tej podpopulacji kryterium skalarne. Kojarzenie i krzyżowanie chromosomów przekracza granice podpopulacji i obejmuje całą populacje. Wadą algorytmu jest tendencja do pomijania rozwiązań „pośrednich", które są nieźle oceniane ze względu na każde kryterium składowe, ale nienajlepiej ze względu na każde z nich z osobna.

Wprowadzenie procedury nadawania rang osobnikom w populacji zasugerował Goldberg w 1989 roku [69]. Procedury nadawania rang stanowiły znaczący postęp w rozwoju algorytmów ewolucyjnych, gdyż umożliwiają porównywanie rozwiązań w populacji zgodnie z definicją rozwiązań optymalnych w sensie Pareto. W procedurze tej warianty niezdominowane w populacji otrzymują rangę najniższą.

Fonseca i Fleming skonstruowali MOGA (ang. *Multi-Objective Genetic Algorithm*), w którym zaproponowali alternatywną procedurę nadawania rang [61]. Ranga chromosomu odpowiada liczbie dominujących je osobników w populacji.

Następnie Srinivas i Deb skonstruowali NSGA (ang. *Non-dominated Sorting Genetic Algorithm*), w którym zastosowano wersję procedury nadawania rang oraz graficzną metodę porównania wyników [140]. Kilka lat później Deb dokonał znaczących modyfikacji w algorytmie NSGA-II [50], który obecnie oferowany jest jako toolbox Matlaba.

Opracowanie procedury niszowania przez Deba i Goldberga umożliwiło dywersyfikację rozwiązań w populacji, a w konsekwencji uniknięcie zbieżności przeszukiwań do jednego tylko punktu ze zbioru Pareto [51]. Horn i Nafpliotis opracowali NPGA (ang. *Niched Pareto Genetic Algorithm*), w którym zaproponowano alternatywną procedurę niszowania [74].

Wielokryterialny algorytm ewolucyjny, który operował na populacji sztucznych sieci neuronowych opracowano w 1995 roku. Sieci zostały zaprojektowane do wyznaczania rozwiązań Pareto-optymalnych w badanym zagadnieniach przydziału zadań [24, 25].

W 1999 roku Zitzler i Thiele opracowali algorytm SPEA (ang. *Strength Pareto Evolutionary Algorithm*). Cechą charakterystyczną algorytmu jest fakt, że osobniki reprezentujące lokalne rozwiązania niezdominowane (wśród dotychczas rozważonych rozwiązań), są przechowywane w archiwalnym zbiorze zewnętrznym. Ponadto, wartość przystosowania osobnika należącego do populacji zależy od rangi wyznaczonej na podstawie porównania z osobnikami ze zbioru zewnętrznego, a nie z bieżącej populacji. Alternatywy ze zbioru zewnętrznego biorą udział w selekcji.

Zaletą algorytmu jest to, że wyznacza się relatywnie regularną reprezentację frontu Pareto. Zasadniczą wadą jest większa złożoność obliczeniowa. Szczególnie czasochłonna jest procedura wyznaczania dopasowania osobnika, a także konieczny jest przegląd zupełny archiwum zewnętrznego [162]. SPEA 2 został opracowany w 2001 roku, a trzy lata później - jego rozszerzona wersja SPEA 2+ [97].

Corne i Knowles opracowali w 1999 roku algorytm PAES (ang. *Pareto Archived Evolution Strategy*), a w 2000 roku PESA (ang. *Pareto Envelope-based Selection Algorithm*) [47]. PAES jest kombinacją strategii ewolucyjnej (1+1) oraz zbioru archiwalnego, w którym przechowuje się lokalne rozwiązania niezdominowane. Osobnik po mutacji jest porównywany z osobnikami znajdującymi się w archiwum.

W PAES zaprezentowano nowatorskie podejście pod względem utrzymania różnorodności, polegające na zastosowaniu procedury zagęszczenia (ang. *crowding procedure*). Każda ocena rozwiązania należy do odpowiedniego obszaru hiper-siatki (ang. *hyper-grid*), która powstała z układu współrzędnych. Wyznaczana jest informacja o liczbie rozwiązań w zadanym obszarze siatki. Zapewnia się zróżnicowanie rozwiązań przez preferowanie rozwiązań, których oceny należą do mniej zagęszczonych obszarów. Rozmiar obszarów może być kontrolowany przez wartość miary zagęszczenia (ang. *crowding distance*). Niewłaściwie dobrane wartości miary zagęszczenia powodują, że obszary mogą być zbyt duże i nie osiągnie się zróżnicowania. Jeśli obszarów jest zbyt wiele, to złożoność obliczeniowa może niekorzystnie wzrosnąć [44].

W podejściu stosowanym w PESA, podobnie jak w SPEA, wyróżnia się populację wewnętrzną i większą od niej populację zewnętrzną. W populacji archiwalnej przechowuje się lokalne rozwiązania niezdominowane, a populacja wewnętrzna zawiera rozwiązania rywalizujące o włączenie do archiwum. Różnorodność jest utrzymywana podobnie jak w PAES, przy czym selekcja w PESA jest realizowana w bardziej złożony sposób. Zarówno w PESA, PAES, jak i w SPEA decyzja, który osobnik powinien opuścić archiwum, oparta jest na mierze zagęszczenia.

Teoria wielokryterialnej optymalizacji w odniesieniu do algorytmów ewolucyjnych została przedstawione w monografii Deba z 2007 roku [49]. Monografią związaną z powyższym tematem jest praca [44, 45]. Interesujące opracowania dotyczą także aproksymacji frontu Pareto przy pomocy metod stochastycznych [42] oraz sposobów nadawania rang [43].

Pierwsza polskojęzyczna monografia na temat wykorzystania algorytmów ewolucyjnych do rozwiązywania problemów optymalizacji wielokryterialnej ukazała się w 2000 roku [20]. Analizę rozwoju wielokryterialnych algorytmów ewolucyjnych MOEA (ang. *Multi Objective Evolutionary Algorithm*) można znaleźć w [97] i [98].

Do poszukiwania przydziałów procesów w zagadnieniach dotyczących optymalizacji przydziału zadań wykorzystuje się adaptacyjny wielokryterialny algorytm ewolucyjny AMEA (ang. *adaptive multicriterion evolutionary algorithm*) [18, 19].

4.3. Implementacja wielokryterialnego algorytmu ewolucyjnego

Algorytm NSGA-II Deba uważany jest za jeden z najbardziej efektywnych algorytmów ewolucyjnych, który stosuje się do wyznaczania rozwiązań różnorodnych problemów optymalizacji wielokryterialnej [13, 49]. Wariant algorytmu NSGA-II dostępny jest jako *toolbox* w Matlabie [106]. Tę właśnie wersję znacząco zmodyfikowano pod kątem zwiększenia jakości wyznaczanych rozwiązań, a nową wersję nazwano NSGA-III.

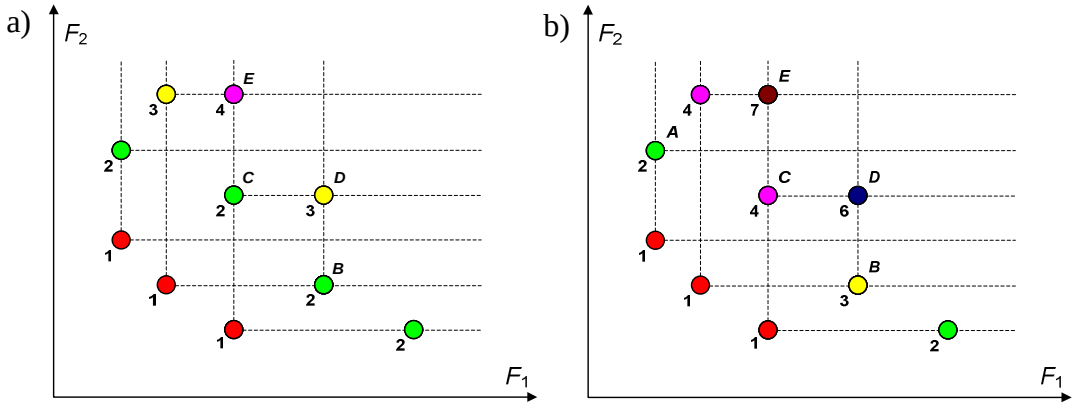
W algorytmie NSGA-III populacja początkowa generowana jest losowo. Ustalane są: rozmiar i typ populacji oraz zakres wartości zmiennych. Domyślny rozmiar populacji to *15* liczba zmiennych*, co może być niewystarczające dla problemów z dużą liczbą zmiennych. Złożoność obliczeniowa algorytmu jest rzędu $O(n^3)$, gdzie $n=\max\{L, k\}$, przy czym k określa liczbę kryteriów, a L rozmiar populacji [50].

Funkcja programu w języku Matlab *gamultiobj* wywołuje algorytm genetyczny. W domyślnym wariancie *elitarystycznym* algorytmu preferowane są osobniki z większą wartością funkcji przystosowania. W wariancie *kontrolowanym* dodatkowo preferuje się osobniki, które mogą zwiększyć różnorodność populacji, nawet jeśli mają mniejszą wartość przystosowania. Ma to na celu zapewnienie reprezentatywnej reprezentacji rozwiązań niezdominowanych w sensie Pareto.

W procedurze nadawania rang opartej na liczbie poziomów niezdominowanych rozwiązań przyjmuje się, że dwa osobniki mają równe rangi, jeśli są wzajemnie

niezdominowane (rys. 4.3.b). Implikuje to identyczne wartości funkcji przystosowania osobników z równymi rangami. Jeśli jednak wybrany osobnik dominuje nad drugim, to ma niższą rangę. Jeśli w populacji istnieją dopuszczalne rozwiązania, wtedy niezdominowane osobniki otrzymują rangę 1. Następnie są one tymczasowo eliminowane z populacji i kolejne Pareto-optymalne alternatywy otrzymują rangę 2. Procedura jest powtarzana aż do wyczerpania zbioru rozwiązań dopuszczalnych [50].

Podejście zaproponowane w [61] zakłada, że ranga osobnika w populacji zależy od liczby osobników, które nad nim dominują (rys. 4.3.b).



Rys. 4.3. Procedury nadawania rang wykorzystujące liczbę:
a) poziomów niezdominowanych rozwiązań
b) osobników dominujących
Źródło: opracowanie własne.

Punkty *E* i *D* są zdominowane przez większą liczbę ocen (rys. 4.3.b) niż liczba poziomów niezdominowanych wyników dla populacji (rys. 4.3.a). Punkty te cechują się niższym prawdopodobieństwem selekcji. Może to prowadzić do przedwczesnej zbieżności. Co więcej, w przypadku procedury wykorzystującej liczbę osobników dominujących, punkty *B* i *C* mają różne rangi, mimo że obie oceny nie dominują się wzajemnie. Oprócz dwóch wymienionych metod nadawania rang osobnikom stosuje się metody przedstawione w pracach [5] i [83].

Liczba elitarnych osobników z tą samą rangą w zbiorze Pareto może być zredukowana poprzez faworyzowanie alternatyw, które są bardziej odległe od siebie (opcja *'DistanceFcn'*). Dla każdego osobnika z populacji wyznacza się wartość miary zagęszczenia, bazując na usytuowaniu sąsiednich ocen rozwiązań w przestrzeni kryterialnej lub rozwiązań w przestrzeni rozwiązań. Odpowiednia liczba alternatyw tej samej rangi, posiadająca największą wartość miary zagęszczenia, może być usuwana z populacji, jeśli jest to konieczne. Wykorzystuje się w tym celu operator miary

zagęszczenia $\prec_n$, za pomocą którego dla rozwiązań o tej samej randze, preferuje się oceny rozwiązań zlokalizowane w mniej „zatłoczonych" obszarach [50].

W pierwszym kroku algorytmu NSGA-III generowana jest losowa populacja $P(0)$ o rozmiarze L, która jest sortowana według relacji dominowania (rys. 4.4).

```
1. BEGIN
2. t:=0, ustaw liczebność populacji L, prawdopodobieństwo krzyżowania p_c, tempo
   mutacji p_m
3. wygeneruj populację początkową P(t) o rozmiarze L
4. oblicz rangi r(x) i przystosowanie f(x), x∈P(t)
5. wygeneruj populację Q(t) o rozmiarze L używając selekcji, mutacji i krzyżowania
6. finish:=FALSE
7. WHILE NOT finish DO
8.       BEGIN /* nowa populacja */
9.       wygeneruj populację R(t)=P(t)∪Q(t) o rozmiarze 2L
10.      oblicz rangi r(x) dla R(t)
11.      t:=t+1, P(t):=∅
12.      WHILE rozmiar P(t)<L DO
13.            dodaj kolejny zbiór rozwiązań (front) o najniższej randze do P(t)
14.            IF (liczba rozwiązań z wykorzystanych zbiorów) > L THEN
15.                  sortuj w malejącym porządku, używając operatora miary
                     zagęszczenia ≺_n
16.            END
17.      END
18.        wygeneruj populację Q(t) o rozmiarze L używając selekcji, mutacji
           i krzyżowania dla adaptacyjnie zmieniających się wartości p_c, p_m
19. finish:=TRUE
20. END
```

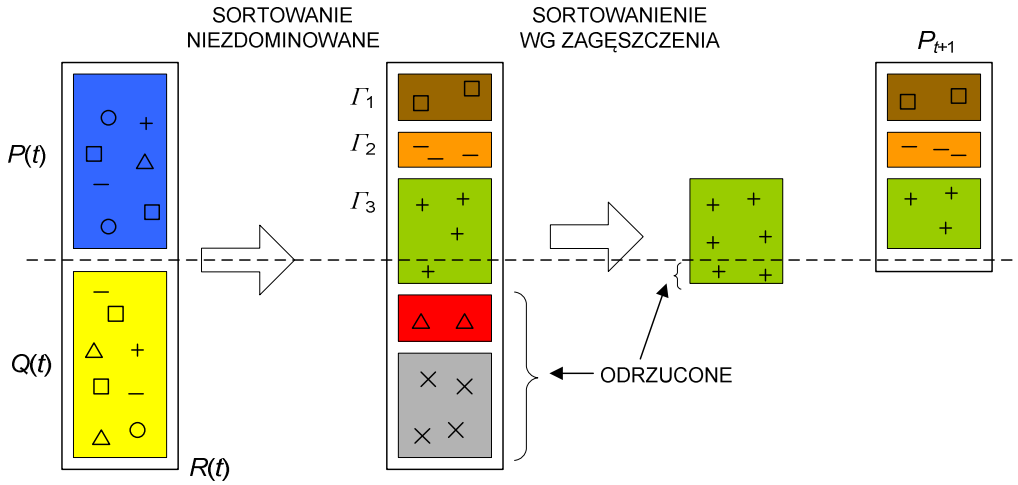
Rys. 4.4. Pseudokod dla algorytmu NSGA-III
Opracowanie własne.

Rozwiązaniu w populacji przydzielana jest ranga w zależności od wartości przystosowania. Następnie selekcja, krzyżowanie i mutacja są wykorzystywane do wygenerowania populacji potomnej $Q(0)$ o rozmiarze L. Powstaje populacja $R(0)$ o rozmiarze $2L$. Procedura obliczeń po wygenerowaniu populacji początkowej jest inna, ponieważ zasada elitaryzmu polega na porównaniu bieżącej populacji z poprzednio wyznaczonymi najlepszymi rozwiązaniami niezdominowanymi.

Generowana jest zatem populacja $R(t)=P(t)\cup Q(t)$ o rozmiarze $2L$, która jest sortowana według relacji dominowania. Rozwiązania należące do niezdominowanego zbioru Γ_1 z najmniejszymi rangami, muszą być preferowane bardziej niż inne. Jeśli liczność zbioru Γ_1 jest mniejsza od L, wtedy elementy zbioru Γ_1 są wybierane do nowej

populacji $P(t+1)$. W następnej fazie populacja $P(t+1)$ jest kompletowana poprzez wybieranie osobników z wyznaczonych zbiorów lokalnie niezdominowanych.

W przypadku przedstawionym na rysunku 4.5 rozwiązania wybierane są ze zbioru Γ_2, a następnie z Γ_3. Gdy liczba rozwiązań ze wszystkich rozważanych zbiorów jest większa od L, wtedy w ostatnim ze zbiorów stosuje się sortowanie, wykorzystując operator miary zagęszczenia $\prec_n$ i wybiera się najlepsze rozwiązania niezbędne do wypełnienia populacji.



Rys. 4.5. Zasada działania algorytmu NSGA-III
Źródło: opracowanie własne.

W algorytmie można określić liczbę osobników, które zostaną skopiowane do następnego pokolenia (opcja '*EliteCount*'), co oznacza się jako L_{elite}.

Selekcja przeprowadzana jest za pomocą selekcji turniejowej. Domyślna liczba graczy turniejowych wynosi 4. Z losowo generowanych grup osobników o ustalonym rozmiarze (opcja '*tournamentSize*) do następnego pokolenia wybierany jest osobnik z największą wartością funkcji przystosowania.

Stosowane są także opcje związane z krzyżowaniem i mutacją. Krzyżowanie może być jednopunktowe, dwupunktowe lub arytmetyczne. Ponadto krzyżowanie pośrednie (ang. *intermediate*) oraz krzyżowanie heurystyczne dostępne są dla współrzędnych rozwiązania będących liczbami rzeczywistymi. Punkty krzyżowania wyznaczane są losowo. Ponieważ tempo krzyżowania jest stałe podczas przeszukiwania algorytmu, wprowadzono tempo krzyżowania, które maleje wraz ze wzrostem numeru populacji t. Wykorzystano zależność $p_c = e^{-t/T_{\max}}$.

W mutacji adaptacyjnej, będącej domyślną dla programu, mutowane geny powinny spełniać ograniczenia liniowe. Dostępna jest również mutacja z dystrybucją gaussowską. Natomiast domyślna wartość prawdopodobieństwa równomiernej mutacji wynosi p_m=0,01.

Istnieje możliwość zadeklarowania, jaki procent populacji może być skopiowany do archiwizowanego zbioru Pareto. Opcję '*ParetoFraction*' oznaczamy jako L_{Pareto}. Zachodzi $0 \leq L_{Pareto} \leq 1$.

Za pomocą parametru L_c określa się, jaka część populacji podlega krzyżowaniu (opcja '*CrossoverFraction*'). Zachodzi $0 \leq L_c \leq 1$. Warto podkreślić, że parametr L_c nie jest prawdopodobieństwem krzyżowania p_c. Niech populacja składa się z L=20 osobników, z których dwa są osobnikami elitarnymi (L_{elite}=2). Niech także współczynnik krzyżowania L_c wynosi 0,8. W następnym pokoleniu oprócz dwóch elitarnych osobników, zostanie wyznaczonych $L_c*(L- L_{elite}) \approx 14$ osobników w wyniku operacji krzyżowania, a pozostałe 4 alternatywy w wyniku mutacji.

Aby wyznaczyć wartość tempa mutacji p_m korzysta się z zależności, jak niżej:

$$p_m = \frac{\lceil (L - L_{elite})(1 - L_c) \rceil}{M} \tag{4.2}$$

gdzie

$\lceil x \rceil$ - zaokrąglenie górne liczby rzeczywistej x do liczby całkowitej,

M - liczba zmiennych decyzyjnych, obliczane jako $M=(V+J)I$ przy kodowaniu binarnym chromosomu w zagadnieniach przydziału modułów.

Ponieważ wartość tempa mutacji jest stała, co wpływa niekorzystnie na jakość przeszukiwań, wprowadzono systematyczne zwiększanie tej wartości. Wyniki o wyższej jakości uzyskano przy wzroście tempa mutacji wraz ze wzrostem numeru populacji, zgodnie z zależnością $p_m = \frac{1 - e^{-t/T_{\max}}}{LM}$.

Generowane chromosomy w populacji początkowej mogą być liczbami binarnymi lub rzeczywistymi. Mogą być także obiektami zdefiniowanymi przez użytkownika (ang. *custom*). Aby badać przydziały całkowitoliczbowe należy zdefiniować typ populacji jako *custom*. Liniowe ograniczenia są dostępne tylko dla domyślnego typu chromosomów w postaci liczb rzeczywistych.

Działanie algorytmu można zakończyć na trzy sposoby. Pierwszy to określenie maksymalnej liczby pokoleń, które powinny zostać wygenerowane. Domyślna wartość

to 200 pokoleń. W drugim przypadku program zatrzymuje się, gdy zmiana średniej wartości funkcji przystosowania przez zadaną liczbę pokoleń (opcja '*StallGenLimit*') jest mniejsza niż wartość określona w parametrach programu (opcja '*TolFun*'). Trzecie kryterium stopu pozwala na zakończenie działania po upływie określonego czasu.

Funkcja *gamultiobj* pozwala na osiągnięcie obszaru w pobliżu frontu Pareto. Następnie koniecznych jest zazwyczaj wiele obliczeń do osiągnięcia optimum. Stosuje się zatem podejście polegające na uruchomieniu funkcji *gamultiobj* dla pewnej liczby pokoleń, po to by wyznaczyć rozwiązania blisko optymalnego frontu. Rozwiązanie uzyskane za pomocą funkcji *gamultiobj* stanowi punkt startowy dla funkcji *fgoalattain* z pakietu *Optimization Toolbox*, która jest skuteczniejsza w poszukiwaniu lokalnych optimów.

Użycie funkcji *gamultiobj* i *fgoalattain* pozwala na uzyskanie optymalnego zbioru Pareto, ale może prowadzić do utraty różnorodności rozwiązania. W celu zapobieżenia temu zjawisku, finalna populacja uzyskana za pomocą *fgoalattain* zostaje populacją wyjściową, dla uruchomionej jeszcze raz funkcji *gamultiobj* [220].

Niech $F(x)$ jest dwukryterialną funkcją, gdzie x jest zmienną dwuwymiarową, $F_1 = x_1^4 - 10x_1^2x_2 + x_2^4 - x_1^2x_2^2$, a $F_2 = x_2^4 - x_1^2x_2^2 + x_1^4 + x_1x_2$. Założono również, że $-5 \le x_1, x_2 \le 5$.

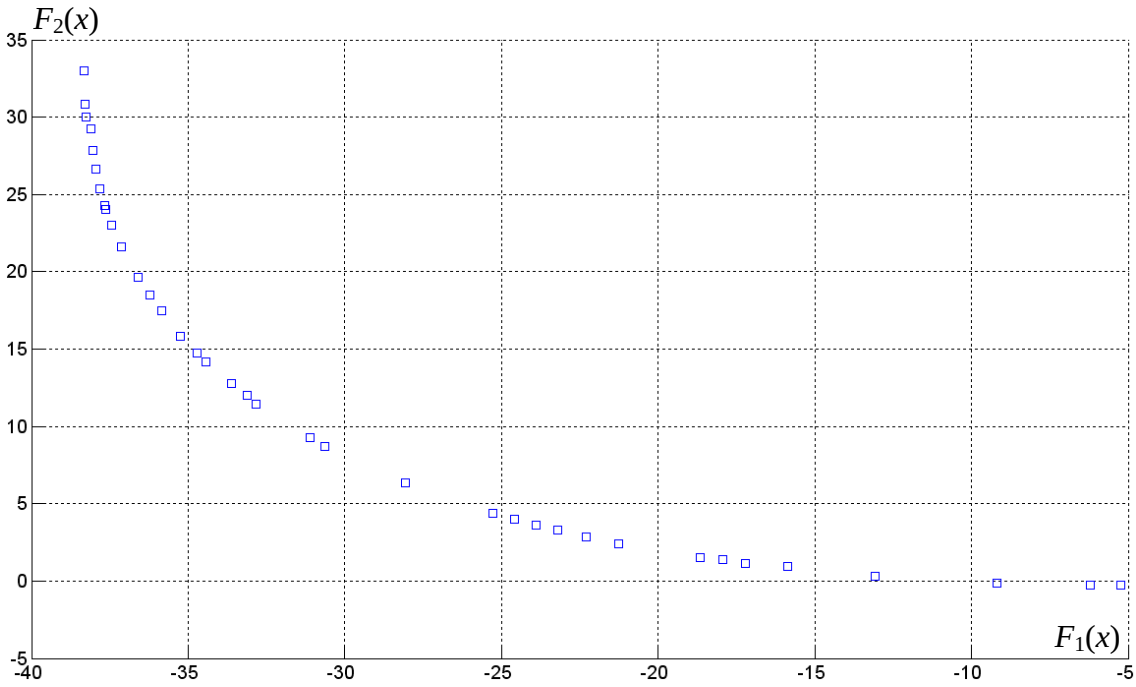
Przeprowadzono eksperymenty polegające na empirycznym doborze wybranych opcji programu. Rozmiar populacji ustalono na L=30, co stanowi piętnastokrotność liczby zmiennych. Przyjęto, że maksymalnie 35% osobników z populacji może być elementami ze zbioru Pareto (L_{Pareto}=0,35). Wykonywanie programu zakończono, gdy zmiana średniej wartości funkcji przystosowania przez zadaną liczbę pokoleń (opcja '*StallGenLimit*' =100) była mniejsza niż wartość określona w parametrze '*TolFun*' = 10^{-4}. 11 rozwiązań zakwalifikowano jako elementy zbioru Pareto. Do momentu zakończenia działania programu, przez 14,65 sekund, wygenerowano 377 pokoleń, a funkcję sprawności wyznaczono 11 341 razy.

W kolejnym przebiegu algorytmu ustalono rozmiar populacji L=75, a także dopuszczono występowanie 50% populacji osobników z populacji może być elementami ze zbioru Pareto. Wyznaczono 38 punktów ze zbioru Pareto po wygenerowaniu 171 pokoleń. Zwiększenie liczebności populacji pozwoliło na szybsze osiągnięcie rozwiązania optymalnego w sensie Pareto.

W następnym eksperymencie zwiększono dwa parametry *StallGenLimit* =150 oraz *TolFun* = 10^{-3}, co pozwoliło zmniejszyć liczbę generacji do 152. Uzyskano średnią miarę odległości ocen w zbiorze Pareto rzędu 0,021. Mniejsze wartości tej miary wskazują na większą równomierność reprezentacji rozwiązań w zbiorze Pareto.

W następnym eksperymencie zastosowano funkcję hybrydową wykorzystującą procedurę *fgoalattain*. Mimo tego, że wyznaczone elementy należały do zbioru Pareto, uzyskano średnią miarę odległości tych rozwiązań w zbiorze ocen rzędu 0,037 (rys. 4.6). Wskazuje to na pogorszenie różnorodności rozwiązań.

Aby wzmóc poszukiwania w kierunku słabo reprezentowanym, stosuje się metody zapewniające zróżnicowanie punktów frontu Pareto [58, 131]. Wyróżnia się metodę zwaną *fitness sharing*/*niching*. Rozmiar (promień) sąsiedztwa (niszy) jest kontrolowany za pomocą wartości parametru (promień niszy) σ_{share}. Po wyznaczeniu liczby rozwiązań znajdujących się w tej samej niszy, wartość przystosowania jest zmniejszana proporcjonalnie do liczby osobników należących do tego samego sąsiedztwa.



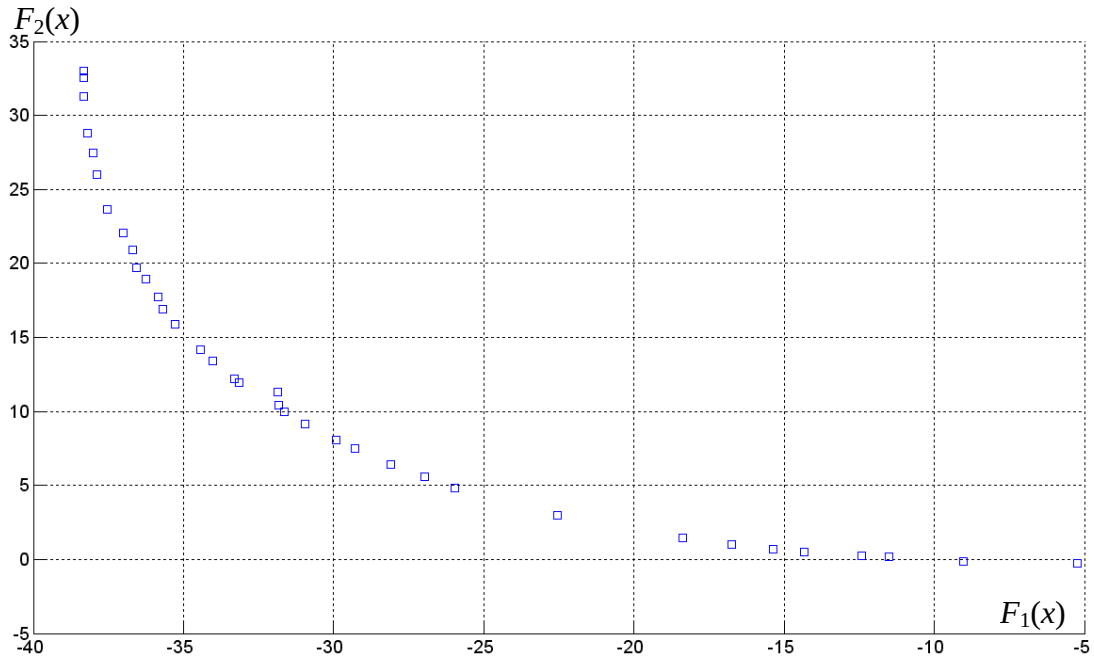
Rys. 4.6. Reprezentacja zbioru Pareto uzyskana za pomocą funkcji hybrydowej
Źródło: opracowanie własne.

Alternatywną metodą stosowaną w NSGA-II, a w konsekwencji i w NSGA-III, do zwiększenia różnorodności jest *crowding*/*clustering*, w której rozwiązania wybiera

się na podstawie miary zagęszczenia w przestrzeni kryteriów. Zasada działania jest podobna do *fitness sharing*.

Wykorzystuje się również „łagodne” formy dominowania (ang. *relaxed form of dominance*), w których dopuszcza się wybór rozwiązania gorszego niż inne rozwiązanie w sensie konkretnego kryterium. Złagodzenie może być zrekompensowane poprawą w stosunku do innych kryteriów [58].

Po użyciu wyznaczonej populacji jako populacji początkowej dla funkcji *gamultiobj*, średnia miara odległości rozwiązań w zbiorze wyników wyniosła 0,014 (rys. 4.7).



Rys. 4.7. Front Pareto uzyskany bez stosowania funkcji hybrydowej, z populacją początkową uzyskaną za pomocą funkcji *fgoalattain*
Źródło: opracowanie własne.

W pracach [15] i [16] do minimalizacji obciążenia najbardziej obciążonego komputera wykorzystano adaptacyjny wielokryterialny algorytm ewolucyjny AMEA (rys. 4.8). Określenie „adaptacyjny” związane jest ze zmianą wartości parametrów, takich jak: prawdopodobieństwo krzyżowania, tempo mutacji lub rozmiar populacji, w trakcie poszukiwania rozwiązań.

Algorytm ten pozwala na osiągnięcie przydziałów zadań wyższej jakości niż inne wielokryterialne algorytmy ewolucyjne w wielu zagadnieniach optymalizacji wielokryterialnej [17]. Wygenerowane we wstępnej populacji osobniki spełniają ograniczenia (2.4) i (2.6) dzięki całkowitoliczbowej reprezentacji chromosomów.

Niech czas obliczeń jest proporcjonalny do μ - liczby obliczeń funkcji przystosowania. Parzysta liczebność populacji L zależy od μ. Jeśli nie nastąpi przedwczesna zbieżność populacji, to $\mu = LT_{\max}$ gdzie $T_{\max}$ jest zadaną maksymalną liczbą populacji. Wzrost liczebności populacji powinien powodować zmniejszenie liczby populacji, które zostaną wygenerowane, dla założonej wartości μ.

Analiza jakości uzyskanych rozwiązań w odniesieniu do relacji między liczebnością populacji a liczbą generacji wykazała, że warianty o najwyższej jakości uzyskuje się w wypadku, gdy liczebność populacji jest znacząco mniejsza niż liczba generacji. Zaleca się zależność L=0,05$T_{\max}$. W tym przypadku, operatory genetyczne są preferowane przed losowym przeszukiwaniem [19].

1. BEGIN
2. t:=0, ustaw liczebność populacji L, prawdopodobieństwo mutacji p_m:=1/M, M – liczba współrzędnych rozwiązania
3. wygeneruj populację początkową $P(t)$,
4. oblicz rangi $r(x)$ i przystosowanie $f(x), x \in P(t)$
5. *finish*:=FALSE
6. WHILE NOT *finish* DO
7. BEGIN /* nowa populacja */
8. $t := t+1,\ P(t) := \varnothing$
9. wyznacz prawdopodobieństwo selekcji $p_s(x), x \in P(t-1)$
10. FOR L/2 DO
11. BEGIN /* cykl reprodukcyjny */
12. selekcja turniejowa w dwu kategoriach pary (a,b) z $P(t-1)$
13. krzyżowanie (a, b) z tempem $p_c = e^{-t/T_{\max}}$
14. mutacja potomków (a',b') z tempem p_m oraz mutacja tabu z tempem $3/LT_{\max}$
15. $P(t) = P(t) \cup (a',b')$
16. END
17. wyznacz $r(x)$ i $f(x), x \in P(t)$
18. IF ($P(t)$ jest zbieżna OR $t \geq T_{\max}$) THEN finish:=TRUE
19. END
20. END

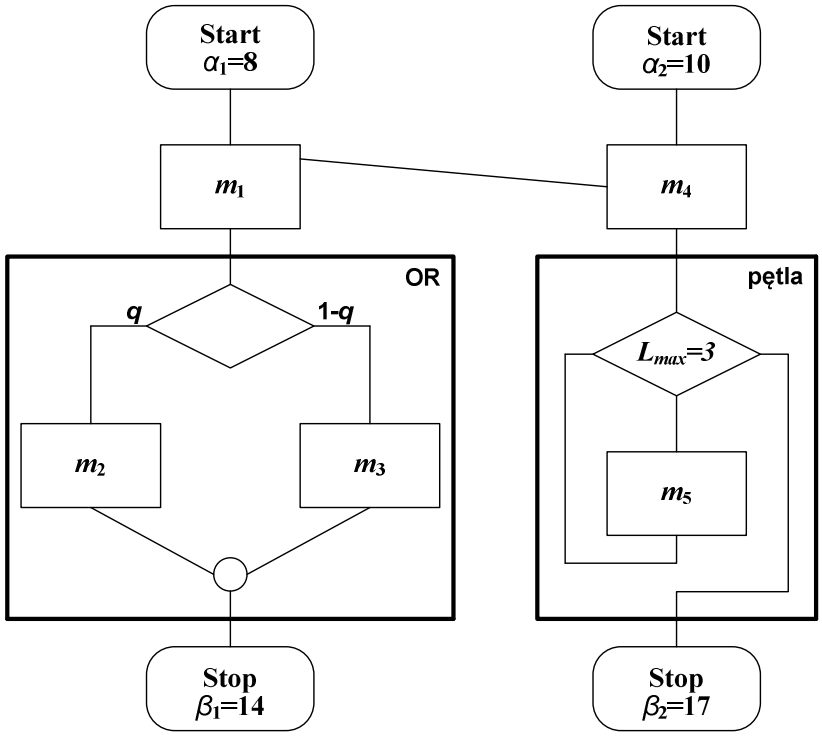
Rys. 4.8. Pseudokod algorytmu AMEA [16]

4.4. Eksperymenty numeryczne

Przed przystąpieniem do eksperymentów numerycznych konieczna była istotna modyfikacja zaimplementowanego w Matlabie algorytmu NSGA-II do wersji

NSGA-III. W zależności od wartości V, I, J, parametrów L_c, L_{elite} oraz L_{Pareto} należało wyznaczyć tempo mutacji p_m wg zależności (4.2) i zmodyfikować plik odpowiadający za mutację, co opisano w dodatku A. Wartość domyślnie ustawiona w Matlabie wynosząca p_m=0,01 była nieodpowiedniego rzędu i nie dawała poprawnych rezultatów. W wyniku analizy uzyskanych wyników wprowadzono adaptacyjne tempo mutacji. Konieczne były także zmiany parametrów L_c, L_{elite} oraz L_{Pareto} oraz wprowadzenie adaptacyjnego tempa krzyżowania.

W eksperymencie nr 1 rozważa się instancję problemu optymalizacji przydziału modułów z funkcją dwukryterialną $F(x)=[P_{time}(x),\ R(x)]^T$ określoną za pomocą zależności (3.12). Dla systemu dwukomputerowego, w którym V=5 oraz J=2, występuje 16 384 możliwe przydziały binarne. Diagram przepływu przedstawiono na rysunku 4.9.



Rys. 4.9. Diagram przepływu dla dwóch programów rozproszonych i pięciu procesów
Źródło: opracowanie własne.

Przyjęto następujące dane wejściowe do problemu: I=2, q=0,3 , $\boldsymbol{\Lambda}=\{5\text{x}10^{-3},10^{-1}\}$ oraz $\boldsymbol{d}=\{10, 18, 18, 17, 21\}$. Wyznaczono L_{INST}=8 instancji diagramu przepływu. Instancje przedstawiono na rysunku 4.10.

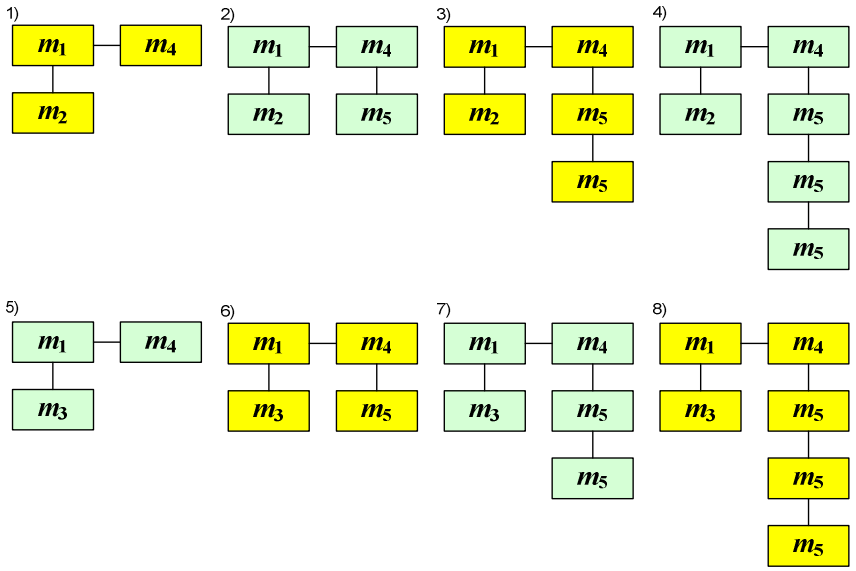
Założono rozkład równomierny dla prawdopodobieństw wykonania modułu w strukturze diagramu typu AND. Ustalono, że czas komunikacji między modułami

usytuowanymi w różnych węzłach wynosi 1 JC, a macierz czasów przetwarzania modułów dla dwóch rodzajów komputerów ma postać, jak niżej:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 3 & 2 & 3 & 4 \\ 2 & 4 & 10 & 7 & 3 \end{bmatrix}^{T} \text{ [JC]}$$

Wyznaczono 128 przydziałów dopuszczalnych, a gęstość rozwiązań dopuszczalnych w przestrzeni przeszukiwań wynosi 0,0078. Minimalna wartość dostępności wynosi 0,0743 i została uzyskana dla 34 przydziałów, a maksymalna - 0,9371 i została uzyskana dla 34 przydziałów.

Minimalna wartość miary realizacji zadań w terminach wynosi 0 i została uzyskana dla 16 przydziałów, a maksymalną wartość 1 uzyskano dla 4 przydziałów.

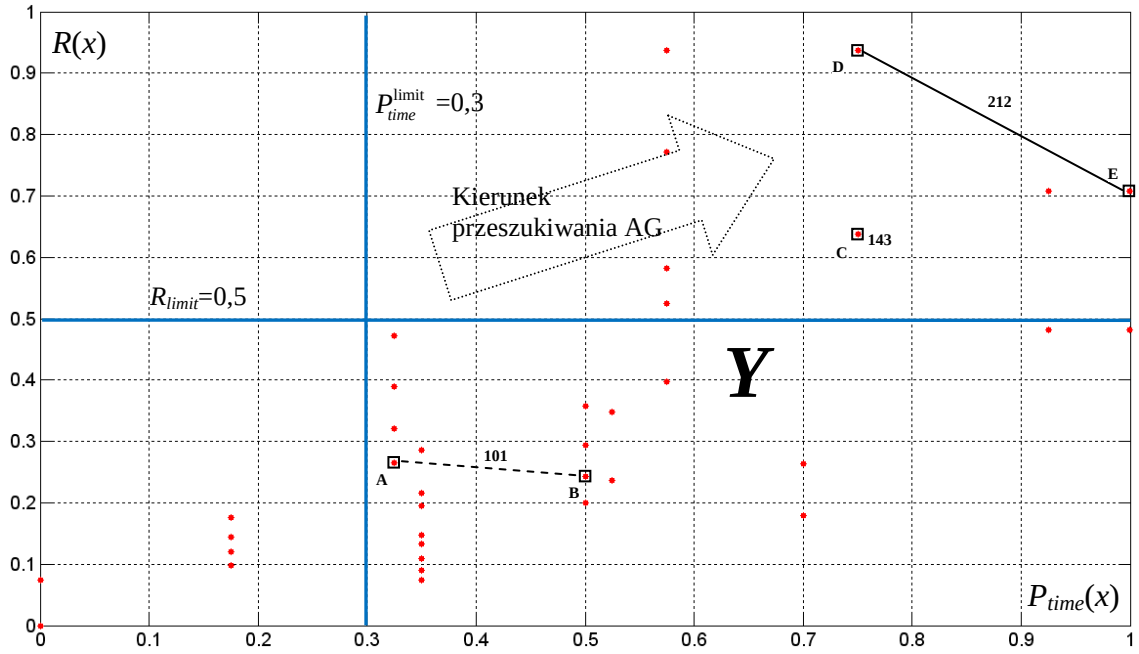


Rys. 4.10. Instancje diagramu przepływu z rysunku 4.9 dla L_{max}=3
Źródło: opracowanie własne.

Na rysunku 4.11 przedstawiono zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ i $P_{time}(x)$ w odniesieniu do analizowanego przypadku. Przy wybranych rozwiązaniach naniesiono numery pokoleń. Punkty połączone przerywanym odcinkiem reprezentują oceny rozwiązań dopuszczalnych. Linia ciągła świadczy o zakończeniu pracy programu z więcej niż jedną oceną optymalną w sensie Pareto. Wyznaczono dwie oceny optymalne.

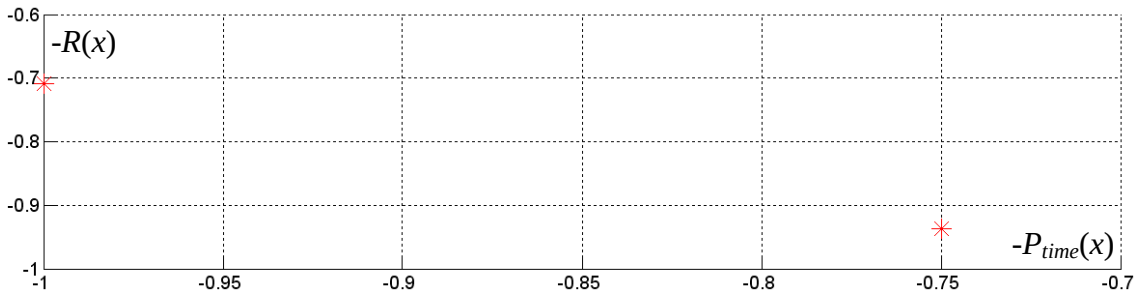
W generacji nr 51 wyznaczono dopuszczalne przydziały dominujące inne rozwiązania z populacji, cechujące się oceną *A*. Następnie skonstruowano przydziały generujące punkty *A* i *B* w populacji nr 101. Podczas generowania rozwiązań dla populacji nr 143 wyznaczono rozwiązanie lokalnie dominujące cechujące się oceną *C*.

W generacji nr 182 wyznaczone zostały dopuszczalne przydziały lokalnie dominujące, cechujące się oceną *D*. W wyniku przeprowadzenia obliczeń skonstruowano przydziały generujące punkty *D* i *E* w populacji nr 212.



Rys. 4.11. Zbiór ocen ***Y*** dla kryteriów $R(x)$ oraz $P_{time}(x)$ dla instancji V=5, I=2, J=2 przy zastosowaniu kodowania binarnego
Źródło: opracowanie własne.

Na rysunku 4.12 przedstawiono wyniki optymalne w sensie Pareto uzyskane dla binarnego kodowania chromosomów. Zobrazowanie wartości ujemnych wynika z pierwotnego wykorzystania funkcji *gamultiobj* w Matlabie do minimalizacji wartości zadanych kryteriów. W pracy wartości kryteriów są maksymalizowane, stąd zmiana znaku na wykresach. Liczba obliczeń funkcji sprawności wyniosła 5 113 przy 212 wygenerowanych pokoleniach.

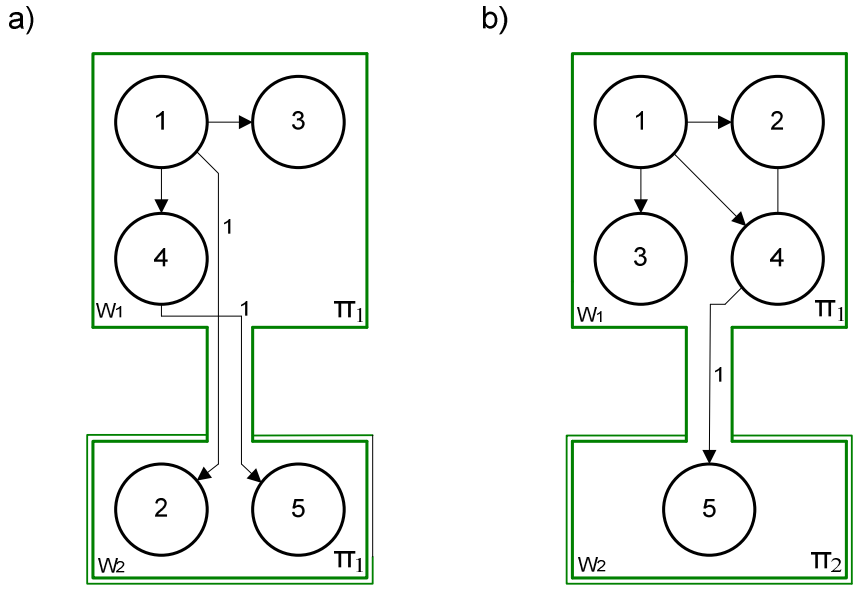


Rys. 4.12. Prezentacja wyników Pareto-optymalnych uzyskanych za pomocą algorytmu NSGA-III zaimplementowanego w Matlabie dla instancji V=5, I=2, J=2
Źródło: opracowanie własne.

Na rysunku 4.13 przedstawiono wyznaczone rozwiązania optymalne w sensie Pareto. Dla przydziału $x^1=[1,0,0,1,1,0,1,0,0,1,1,0,1,0]^T$ (rys. 4.13a) uzyskano wartość $P_{time}(x^1)=0,75$ oraz $R(x^1)=0,9371$. Z kolei dla przydziału z rysunku 4.13b), zapisanego jako $x^2=[1,0,1,0,1,0,1,0,0,1,1,0,0,1]^T$, wyznaczono $P_{time}(x^2)=1$ oraz $R(x^2)=0,7082$.

Przydział x^1 cechuje się większą dostępnością niż x^2 ze względu na większą dostępność komputerów klasy π_1 przydzielonych do obu węzłów w przydziale x^1 niż dostępność komputera klasy π_2 przydzielonego do węzła w_2 w przydziale x^2. Istotną rolę odgrywa parametr $\lambda_1=5x10^{-3}$, który powoduje wolniejsze zmniejszanie dostępności wraz z upływem czasu niż parametr $\lambda_2=10^{-1}$.

Natomiast wartość miary skutecznej realizacji zadań w terminach jest większa dla przydziału x^2. Wartości czasów ukończenia procesów w każdej instancji są mniejsze od nieprzekraczalnych terminów ukończenia tychże procesów, które to terminy nie powodują przekroczenia czasu założonego na wykonanie programu. Dla przydziału x^1, czasy ukończenia wybranych procesów cechują się większymi wartościami, co prowadzi do przekroczenia założonego czasu na wykonanie programu.



Rys. 4.13. Przydziały modułów do komputerów, dla których uzyskano rozwiązania optymalne w sensie Pareto:
a) $F(x)=[0,75; 0,9371]$,
b) $F(x)=[1; 0,7082]$.
Źródło: opracowanie własne.

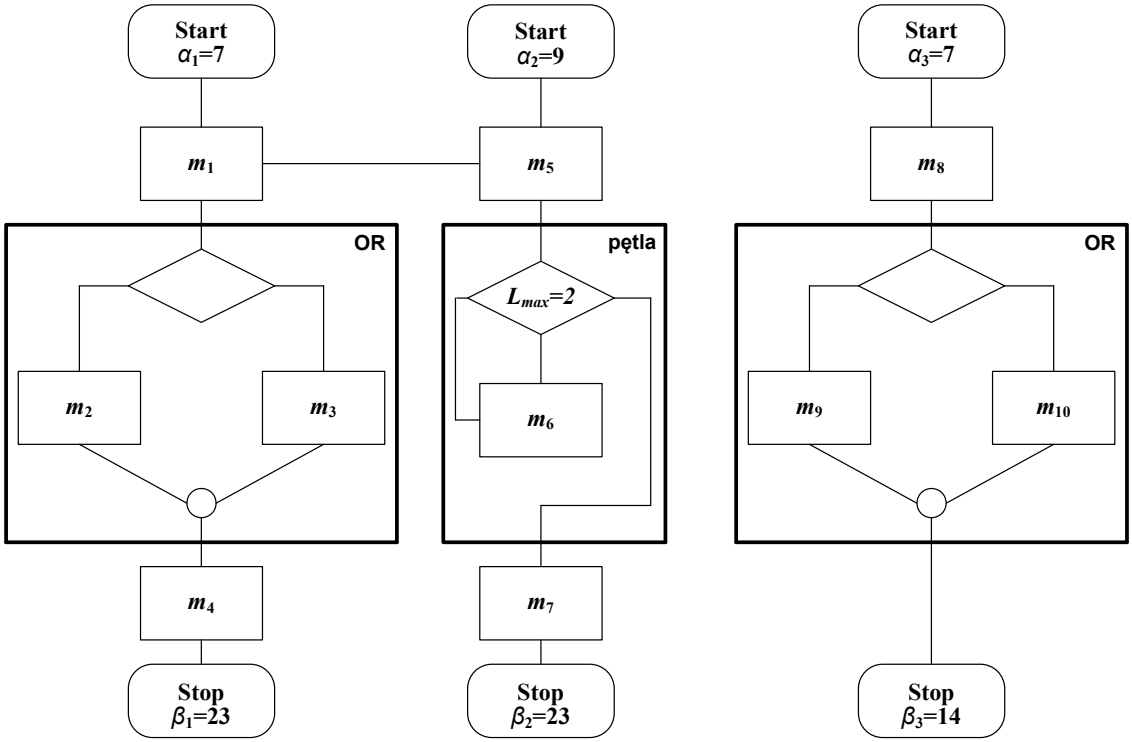
Na komputerze klasy PC z procesorem Intel 1,6 GHz dla pojedynczego przydziału czas wyznaczania $P_{time}(x)$ wyniósł 150 µs, a dla $R(x)$ - 14 µs. Metoda

pełnego przeglądu wyznacza zbiór Pareto za pomocą 16 384 oszacowań funkcji sprawności w 12 sekund.

W wyniku dostrojenia algorytmu genetycznego przyjęto następujące wartości parametrów: $L=22$, $L_{elite}=2$, $L_c=0{,}8$ oraz początkowo $p_m=2/7$.

Jeśli $P_{time}(x)=0$, to rozwiązanie jest niedopuszczalne ze względu na przekroczenie co najmniej jednego terminu realizacji zadania. Wówczas zbiór rozwiązań dopuszczalnych w rozważanym zagadnieniu ulega redukcji ze 128 do 112 przydziałów. Natomiast zakładając zaznaczone na rysunku 4.9 ograniczenia na wartości kryteriów, odpowiednio $P_{time}(x) \geq 0{,}3$ i $R(x) \geq 0{,}5$ otrzymuje się 44 dopuszczalne przydziały.

W eksperymencie nr 2 również rozważa się funkcję dwukryterialną $F(x)=[P_{time}(x),\ R(x)]^T$ określoną za pomocą zależności (3.12). Dla rozpatrywanego systemu dwukomputerowego, w którym $V=10$ oraz $J=2$ otrzymuje się 16 777 216 możliwe przydziały binarne. Diagram przepływu przedstawiono na rysunku 4.14.



Rys. 4.14. Diagram przepływu dla trzech programów rozproszonych i dziesięciu procesów
Źródło: opracowanie własne.

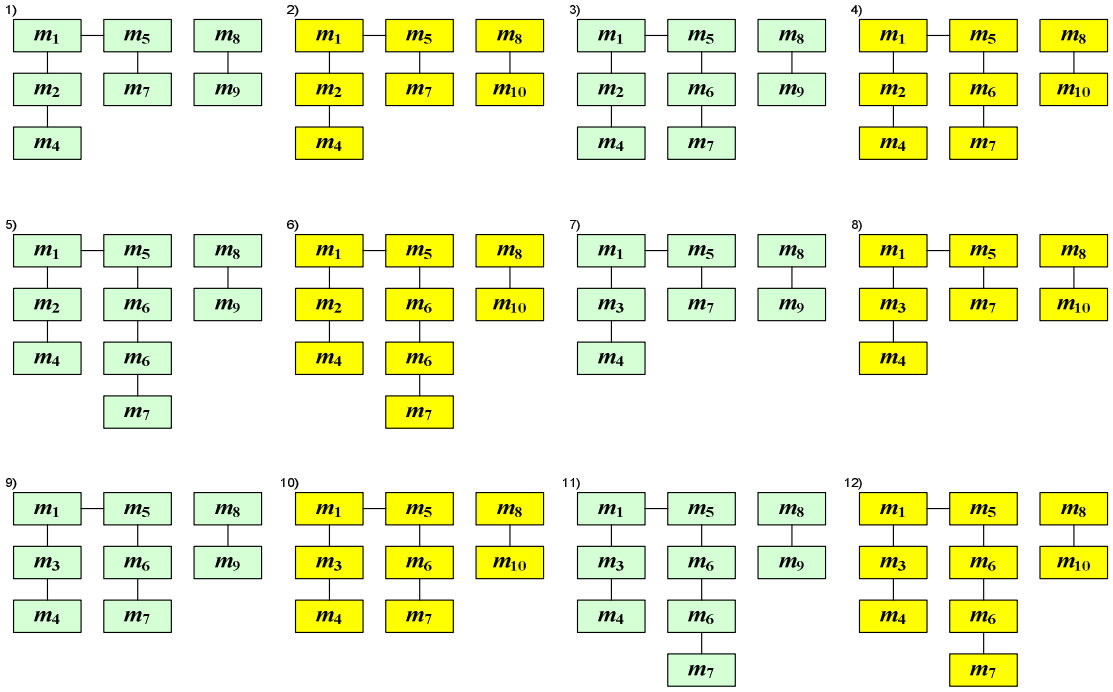
Przyjęto następujące dane wejściowe: $V=10$, $I=2$, $J=2$, $\boldsymbol{\Lambda}=\{10^{-1}, 10^{-2}\}$, $q=0.5$, $\boldsymbol{d}=\{12, 14, 19, 23, 14, 18, 23, 11, 14, 13\}$. Wyznaczono $L_{INST}=12$ instancji diagramu

przepływu (rys. 4.15). Założono rozkład równomierny dla prawdopodobieństw wykonania modułu w strukturze diagramu typu AND. Czas komunikacji między modułami usytuowanymi w różnych węzłach wynosi 1 JC. Macierz czasów przetwarzania modułów dla dwóch rodzajów komputerów ma postać, jak niżej:

$$\boldsymbol{T} = \begin{bmatrix} 2 & 5 & 2 & 3 & 5 & 1 & 2 & 5 & 2 & 4 \\ 5 & 3 & 10 & 7 & 1 & 4 & 5 & 2 & 6 & 1 \end{bmatrix}^T \text{ [JC]}$$

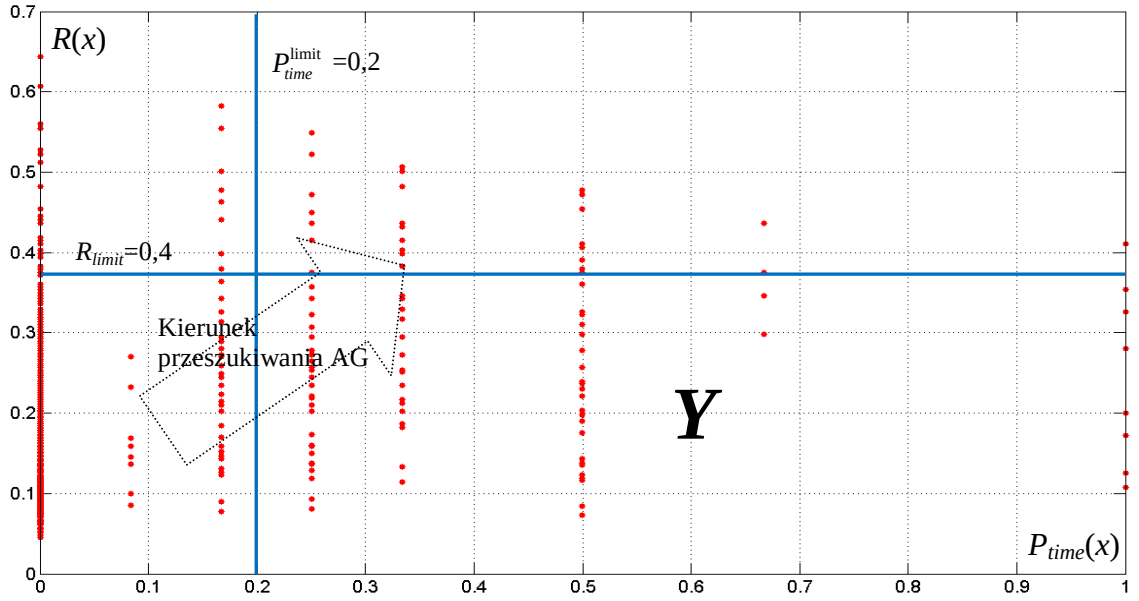
Dla diagramu przepływu z rysunku 4.14 wektory $\boldsymbol{A}$, $\boldsymbol{E}$ oraz $\boldsymbol{\Theta}$ przedstawione są poniżej.

$\boldsymbol{A}$=[1,1,1,1,1,3,1,1,1,1]; $\boldsymbol{E}$=[7,0,0,0,0,0,0,7,0,0]; $\boldsymbol{\Theta}$ =[1,2,2,1,1,3,1,1,2,2].



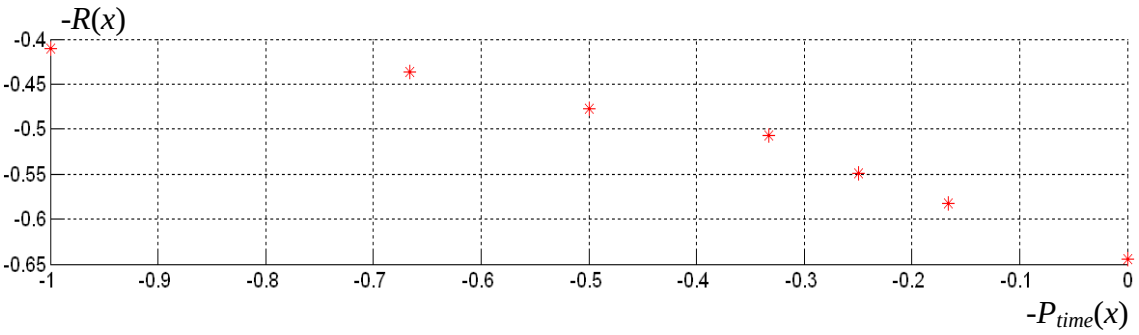
Rys. 4.15. Instancje diagramu przepływu z rysunku 4.14, przy założeniu L_{max}=2
Źródło: opracowanie własne.

Na rysunku 4.16 przedstawiono wyznaczony zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ oraz $P_{time}(x)$. Otrzymano 4 096 przydziałów dopuszczalnych oraz siedem punktów Pareto-optymalnych. Minimalną wartość dostępności 0,045 uzyskano dla 1026 przydziałów, podobnie jak maksymalną wartość dostępności 0,644 - także dla 1026 przydziałów. Gęstość rozwiązań dopuszczalnych w przestrzeni przeszukiwań wynosi 0,000244. Minimalna wartość miary skutecznej realizacji zadań została uzyskana dla 3760 przydziałów, a maksymalna - dla 16.



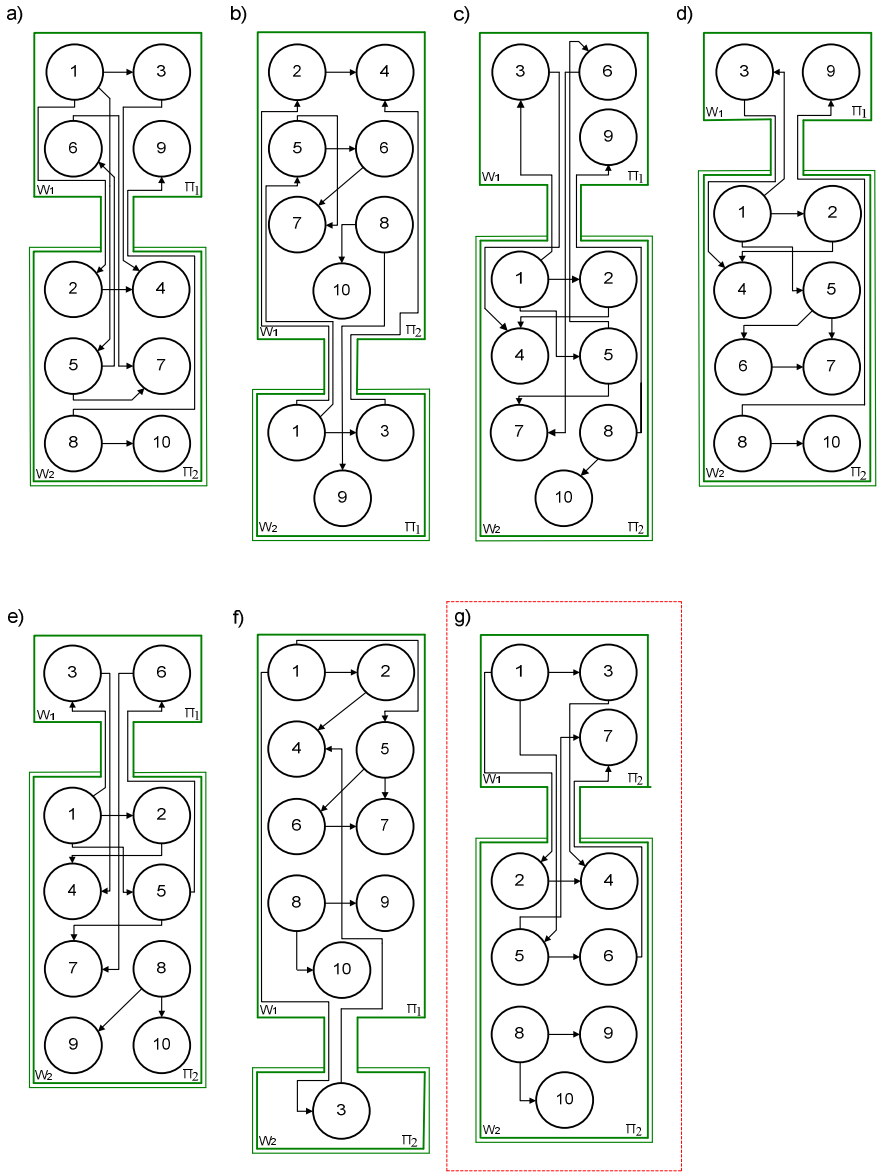
Rys. 4.16. Zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ oraz $P_{time}(x)$ dla instancji V=10, I=2, J=2 w przypadku przydziału binarnego
Źródło: opracowanie własne.

Na rysunku 4.17 zaprezentowano wyniki, które uzyskano na komputerze klasy PC z procesorem Intel 1,6 GHz za pomocą algorytmu NSGA-III po upływie 55 minut po wykonaniu 130 791 obliczeń funkcji sprawności. Czas wyznaczenia wartości $P_{time}(x)$ dla pojedynczego przydziału wyniósł 180 µs oraz 18 µs dla $R(x)$. Metoda pełnego przeglądu wyznacza zbiór Pareto za pomocą 16 777 216 oszacowań funkcji sprawności w czasie około trzech godzin.



Rys. 4.17. Prezentacja wyników optymalnych uzyskanych za pomocą algorytmu NSGA-III zaimplementowanego w Matlabie dla instancji V=10, I=2, J=2
Źródło: opracowanie własne.

W wyniku dostrojenia algorytmu przyjęto następujące wartości parametrów: $L=10$, $L_{elite}=2$, L_c=0,8 oraz początkowo p_m=1/6. Na rysunku 4.18 przedstawiono przydziały modułów do komputerów, dla których uzyskano oceny optymalne w sensie Pareto dla analizowanego przypadku.



Rys. 4.18. Przydziały dziesięciu modułów do komputerów, będące rozwiązaniami optymalnymi w sensie Pareto:
a) $F(x)=[1; 0,4107]$, b) $F(x)=[0,6667; 0,436]$, c) $F(x)=[0,5; 0,4771]$,
d) $F(x)=[0,333; 0,5066]$, e) $F(x)=[0,25; 0,5488]$,
f) $F(x)=[0,1667; 0,5827]$, g) $F(x)=[0; 0,644]$.
Źródło: opracowanie własne.

Dla przydziału $x^1=[1,0,0,1,1,0,0,1,0,1,1,0,0,1,0,1,1,0,0,1,1,0,0,1]^T$ z rysunku 4.18a) uzyskano wartość $P_{time}(x^1)=1$ oraz $R(x^1)=0,4107$. Z kolei dla przydziału z rysunku 4.18b), zapisanego jako $x^2=[0,1,1,0,0,1,1,0,1,0,1,0,1,0,1,0,0,1,1,0,0,1,1,0]^T$, uzyskano wartość $P_{time}(x^2)=0,6667$ oraz $R(x^2)=0,436$.

Następnie dla przydziału $x^3=[0,1,0,1,1,0,0,1,0,1,1,0,0,1,0,1,1,0,0,1,1,0,0,1]^T$ z rysunku 4.18c) uzyskano wartość $P_{time}(x^3)=0,5$ oraz $R(x^3)=0,4771$. Dla przydziału

z rysunku 4.18d), zapisanego jako x^4=[0,1,0,1,1,0,0,1,0,1,0,1,0,1,0,1,1,0,0,1,1,0,0,1]^T, uzyskano wartość $P_{time}(x^4)$=0,333 oraz $R(x^4)$=0,5066.

Dla x^5=[0,1,0,1,1,0,0,1,0,1,1,0,0,1,0,1,0,1,0,1,1,0,0,1]^T (rys. 4.18e) uzyskano $P_{time}(x^5)$=0,25 oraz $R(x^5)$=0,5488. Dla x^6=[1,0,1,0,0,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,1,0]^T (rys. 4.18f) uzyskano $P_{time}(x^6)$=0,1667 oraz $R(x^6)$=0,5827. Natomiast dla przydziału x^7=[1,0,0,1,1,0,0,1,0,1,0,1,1,0,0,1,0,1,0,1,0,1,0,1]^T (rys. 4.18g) wyznaczono $P_{time}(x^7)$=0 oraz $R(x^7)$=0,644.

Przydział (rys. 4.18g) wyznaczono za pomocą metody pełnego przeglądu. Ponieważ $P_{time}(x^7)$=0, to przydział jest niedopuszczalny z powodu przekroczenia co najmniej jednego terminu realizacji zadania. Dlatego też przydział ten wyróżniono przerywaną linią.

Jeśli $P_{time}(x)$=0, to zbiór rozwiązań dopuszczalnych w rozważanym zagadnieniu ulega redukcji z 4096 do 336 przydziałów. Zakładając zaznaczone na rysunku 4.14 ograniczenia na wartości kryteriów $P_{time}(x)$>0,2 i $R(x)$>0,4, otrzymuje się 40 dopuszczalnych przydziałów. Powoduje to zmniejszenie liczby punktów Pareto-optymalnych z siedmiu do pięciu.

W eksperymencie nr 3 rozpatruje się model systemu *MOODLE*, dla którego optymalizuje się funkcję dwukryterialną $F(x)=[P_{time}(x),\ R(x)]^T$. Dla systemu dwukomputerowego, w którym V=14 oraz J=2 otrzymuje się 2^{32}= 4 294 967 296 możliwe przydziały binarne. Diagram przepływu przedstawiono na rysunku 4.19.

Przyjęto następujące dane wejściowe: V=14, I=2, J=2, q=0.3, $\boldsymbol{\Lambda}=\{10^{-1}, 10^{-2}\}$, $\boldsymbol{d}$={9, 13, 19, 23, 21, 25, 29, 32, 37, 43, 45, 50, 54, 55}. Wyznaczono L_{INST}=4 instancje diagramu przepływu. Instancje przedstawiono na rysunku 4.20. Założono rozkład równomierny dla prawdopodobieństw wykonania modułu w strukturze diagramu typu AND.

Czas komunikacji między modułami znajdującymi się w różnych węzłach wynosi 1 JC. Macierz czasów przetwarzania modułów dla dwóch rodzajów komputerów ma postać, jak niżej:

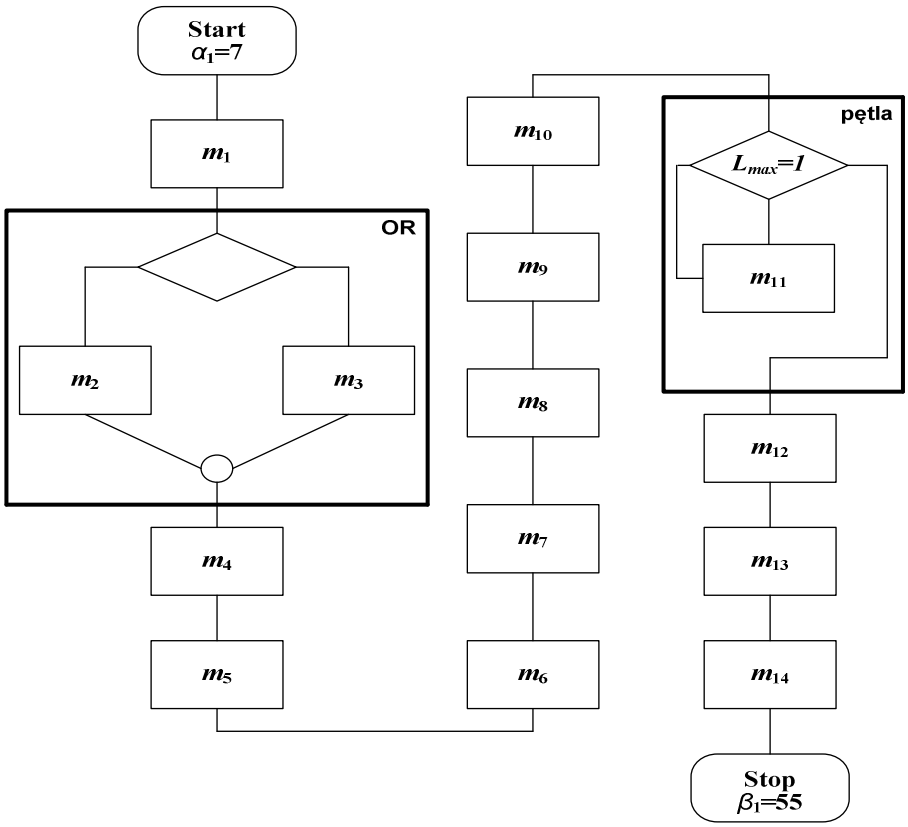
$$\mathbf{T}=\begin{bmatrix}1 & 3 & 2 & 3 & 4 & 3 & 3 & 4 & 1 & 5 & 2 & 7 & 4 & 1\\ 2 & 4 & 10 & 7 & 3 & 4 & 3 & 5 & 6 & 2 & 2 & 1 & 2 & 1\end{bmatrix}^T \text{ [JC]}$$

Dla diagramu przepływu z rysunku 4.19 wektory $\boldsymbol{A}$, $\boldsymbol{E}$ oraz $\boldsymbol{\Theta}$ są, jak niżej:

$\boldsymbol{A}$=[1,1,1,1,1,1,1,1,1,1,1,1,1,1],

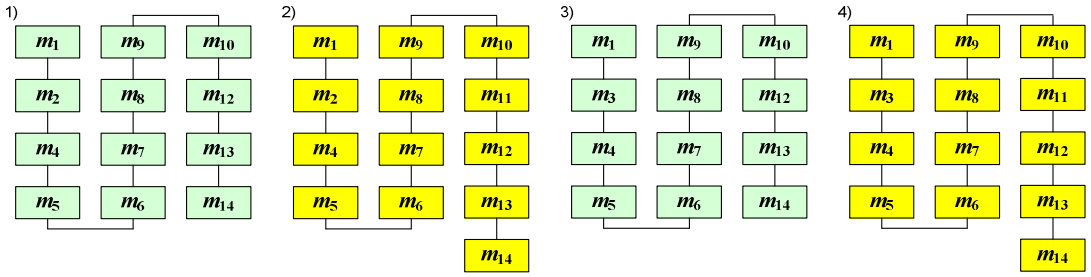
$\boldsymbol{E}$=[7,0,0,0,0,0,0,0,0,0,0,0,0,0],

$\boldsymbol{\Theta}$ =[1,2,2,1,1,1,1,1,1,1,3,1,1,1].



Rys. 4.19. Diagram przepływu dla programu rozproszonego składającego się z czternastu procesów
Źródło: opracowanie własne.

Na rysunku 4.21 przedstawiono zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ oraz $P_{time}(x)$ w odniesieniu do analizowanego przypadku.

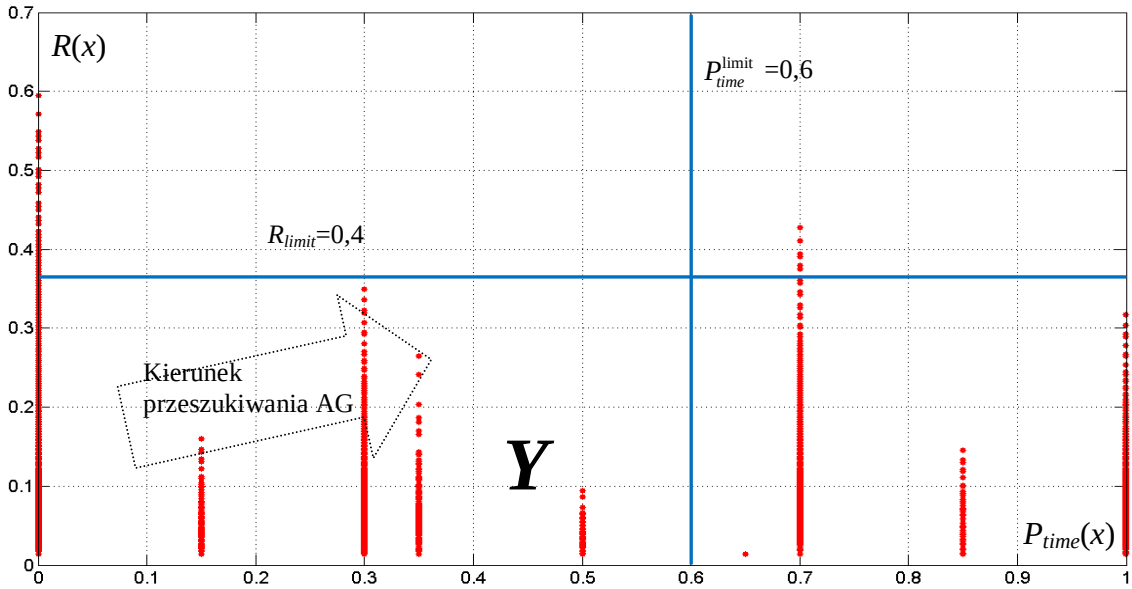


Rys. 4.20. Instancje diagramu przepływu z rysunku 4.19, przy założeniu L_{max}=1
Źródło: opracowanie własne.

Wyznaczono 65 536 przydziałów dopuszczalnych, których gęstość w przestrzeni przeszukiwań wyniosła 0,00001526. Wyznaczono trzy punkty P-optymalne. Minimalną

wartość dostępności 0,0136 uzyskano dla 16386 przydziałów, a maksymalną wartość 0,5945 dla 16386 rozwiązań. Minimalną wartość miary sprawnej realizacji zadań otrzymano dla 40 348 przydziałów, a maksymalną - dla 14 614.

Na rysunku 4.22 zaprezentowano wyniki, które wyznaczono na komputerze klasy PC z procesorem Intel 1,6 GHz za pomocą algorytmu NSGA-III po upływie około 15 godzin w rezultacie wykonania 5 550 011 obliczeń funkcji sprawności. W wyniku dostrojenia algorytmu genetycznego otrzymano następujące wartości parametrów: $L=10$, $L_{elite}=2$, L_c=0,8 i p_m=1/8. Czas wyznaczenia wartości $P_{time}(x)$ dla przydziału wyniósł 146 µs oraz 84 µs dla $R(x)$.

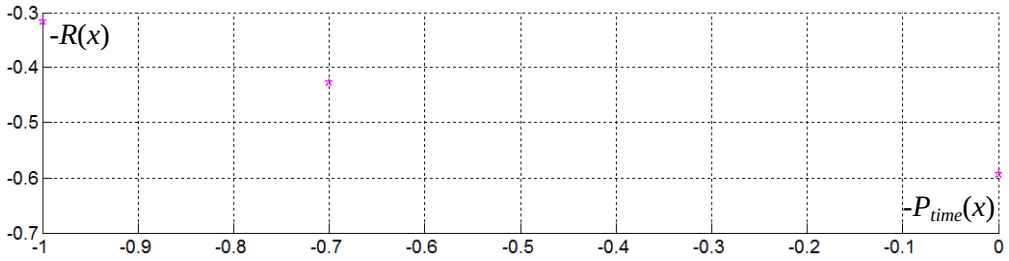


Rys. 4.21. Zbiór ocen $\boldsymbol{Y}$ dla kryteriów $R(x)$ oraz $P_{time}(x)$ dla instancji V=14, I=2, J=2 w przypadku przydziału binarnego
Źródło: opracowanie własne.

Metoda pełnego przeglądu wyznacza zbiór Pareto za pomocą 4 294 967 296 oszacowań funkcji sprawności w czasie około 300 godzin.

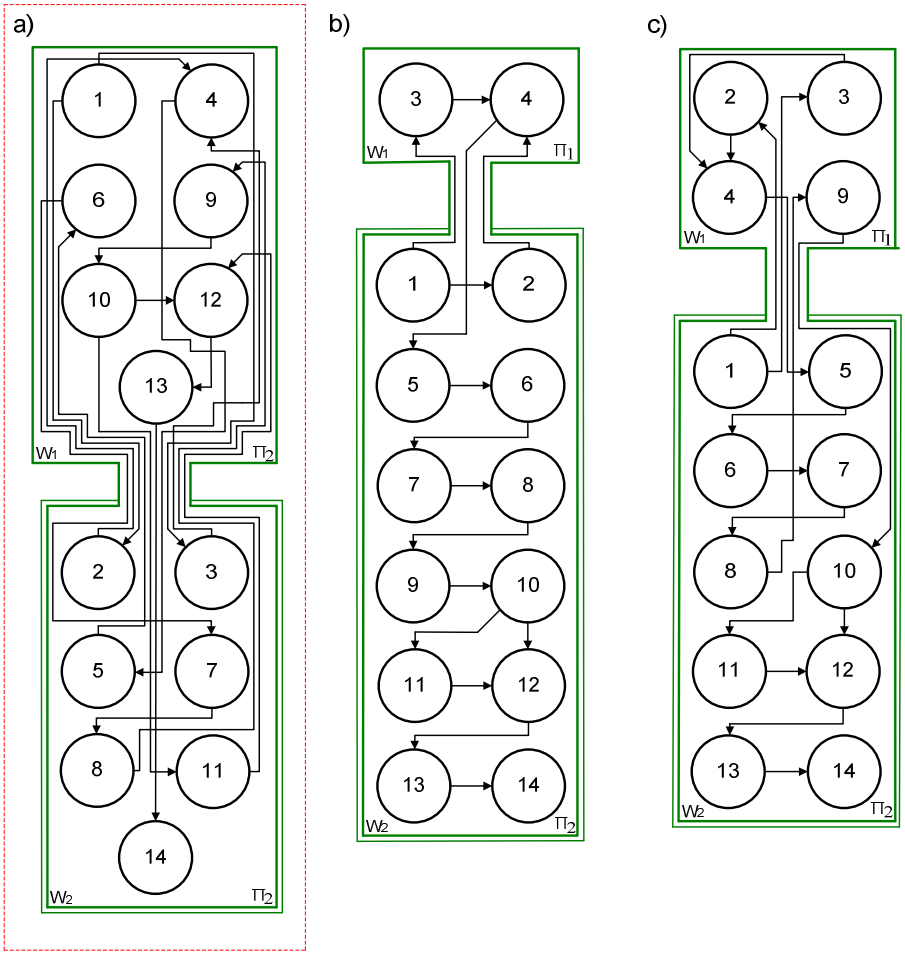
Na rysunku 4.23 przedstawiono przydziały optymalne w sensie Pareto. Dla przydziału x^1=[1,0,0,1,0,1,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,1,1,0,1,0,0,1,0,1,0,1]^T z rysunku 4.23a) wyznaczono wartość $P_{time}(x^1)$=0 oraz $R(x^1)$=0,5945. Z kolei dla przydziału z rysunku 4.23b), zapisanego jako x^2=[0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,0,0,1]^T, uzyskano: $P_{time}(x^2)$=0,7 oraz $R(x^2)$=0,4274. Następnie dla x^3=[0,1,1,0,1,0,1,0,0,1,0,1,0,1,0,1,1,0,0,1,0,1,0,1,0,1,0,1,1,0,0,1]^T z rysunku 4.23c) otrzymano wartość $P_{time}(x^3)$=1 oraz $R(x^3)$=0,3166.

Wyznaczono przydział, który mógłby być optymalny w sensie Pareto, gdyby nie fakt, że $P_{time}(x^1)$=0.



Rys. 4.22. Prezentacja wyników optymalnych uzyskanych za pomocą algorytmu NSGA-III dla instancji V=14, I=2, J=2
Źródło: opracowanie własne.

Jednakże rozwiązanie to należy uznać za niedopuszczalne z powodu przekroczenia co najmniej jednego terminu realizacji zadania. Na rysunku 4.23a) przydział ten został oznaczony czerwoną, przerywaną linią.



Rys. 4.23. Przydziały czternastu modułów do komputerów, dla których uzyskano rozwiązania optymalne w sensie Pareto:
a) $F(x)$=[0; 0,5945], b) $F(x)$=[0,7; 0,4274], c) $F(x)$=[1; 0,3166]
Źródło: opracowanie własne.

Jeśli $P_{time}(x)=0$, to rozwiązanie jest niedopuszczalne. Wówczas zbiór rozwiązań dopuszczalnych ulega redukcji z 65 536 do 25188 przydziałów. Zakładając zaznaczone na rysunku 4.19 ograniczenia na wartości kryteriów $P_{time}(x)>0,6$ i $R(x)>0,4$, otrzymuje się 4 dopuszczalne przydziały. Nałożone ograniczenia powodują również zmniejszenie liczby z trzech punktów Pareto-optymalnych do jednego rozwiązania dominującego.

Wykorzystując wyniki z przeprowadzonych eksperymentów oraz wykonane oprogramowanie, opracowano pakiet do wspomagania administratora systemu *MOODLE* pod kątem optymalizacji rozmieszczenia modułów (dodatek A).

4.5. Wielokryterialny algorytm ewolucyjny działający na populacji sieci neuronowych

Interesującym i perspektywicznym podejściem do optymalizacji przydziałów modułów jest wielokryterialny algorytm ewolucyjny operujący na populacji sztucznych sieci neuronowych o nazwie MEA^2N^2 (ang. *Multi-criteria Evolutionary Algorithm on Artificial Neural Networks*).

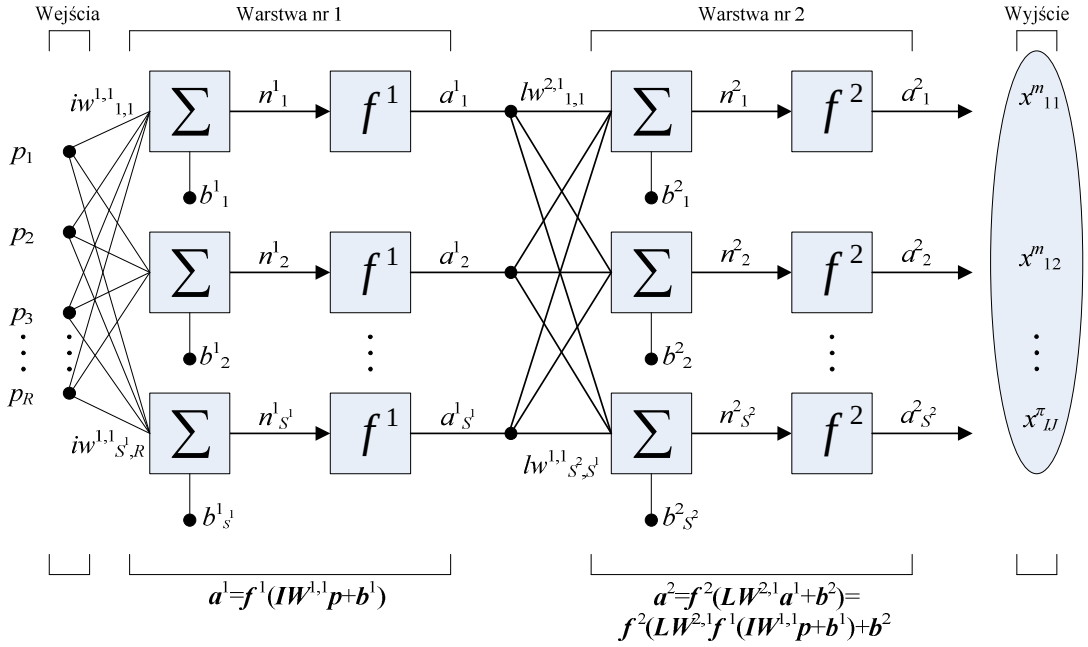
Rozważa się dwuwarstwową sztuczną sieć neuronową przedstawioną na rysunku 4.24. Warstwa pierwsza składa się z S^1 neuronów. Warstwa druga, w tym przypadku nazywana wyjściową, zawiera S^2 neuronów. Każdy element R-elementowego wektora wejściowego $\boldsymbol{p}$ jest połączony z każdym neuronem wejściowym, co charakteryzuje wejściowa macierz wag $\boldsymbol{IW}$ (ang. *input weights*) o rozmiarze $S^1 \times R$.

W i-tym neuronie po zsumowaniu skalarnych wartości z ważonych wejść i wejścia własnego generującego wartość przesunięcia (ang. *bias*), wyznacza się wartość poziomu aktywacji $n(i)$. Konstruowany jest S-elementowy wektor $\boldsymbol{n}$, który po zastosowaniu funkcji przejścia f formuje wektor wyjściowy z warstwy neuronów $\boldsymbol{a}$. Warstwowa macierz wag $\boldsymbol{LW}$ (ang. *layer weights*) charakteryzuje sieć połączeń z wyjść warstwy nr 1 do wejść warstwy nr 2. Rozmiar tej macierzy to $S^1 \times S^2$. W neuronach pierwszej warstwy zastosowano sigmoidalną funkcję przejścia, a w neuronach drugiej warstwy - liniową funkcję przejścia. Sieć tej klasy może być trenowana do implementacji aproksymacji wybranej funkcji ze skończoną liczbą nieciągłości [220]. Zakłada się, że sieć neuronowa jest statyczna, czyli nie posiada sprzężeń i opóźnień.

Koncepcja zastosowania sieci neuronowej do optymalizacji przydziałów modułów polega na wykorzystaniu danych wejściowych do problemu, a w szczególności elementów macierzy $\boldsymbol{T}$, $\boldsymbol{\Omega}$ oraz elementów wektorów $\boldsymbol{d}$, $\boldsymbol{\Lambda}$, $\boldsymbol{B}$, $\boldsymbol{E}$, $\boldsymbol{\Theta}$

jako elementów jednokolumnowego, wejściowego wektora ***p*** o liczbie wierszy zależnej od rozmiarów wymienionych macierzy i wektorów. Na wyjściu sieci znajduje się przydział *x* odpowiadający bieżącym wartościom elementów wejściowego wektora ***p***.

Uczenie sieci polega na dobraniu takich wartości wag, przy których sieć będzie wyznaczała wzorcowe rozwiązania zagadnienia optymalizacji przydziałów. Zastosowano uczenie z nauczycielem. Dla każdego zestawu wzorców wejściowych z ciągu treningowego, znane jest wzorcowe wyjście reprezentujące rozwiązanie Pareto-optymalne.



Rys. 4.24. Dwuwarstwowa sieć neuronowa w wersji graficznej
Źródło: opracowanie własne.

Za pomocą algorytmu uczenia wyznacza się korekty wag po każdorazowym ustaleniu odpowiedzi przez sieć na podany wzorzec wejściowy. Do treningu sieci zastosowano algorytm Levenberga-Marquardta [63]. Trening ten jest realizowany w założonym czasie, którego wartość jest parametrem algorytmu ewolucyjnego operującego na populacji sztucznych sieci neuronowych.

Na rysunku 4.25 przedstawiono pseudokod algorytmu ewolucyjnego operującego na populacji dwuwarstwowych sieci neuronowych. Chromosom w tym przypadku składa się z elementów macierzy ***IW*** i ***LW***, wektora ***b*** oraz parametrów funkcji przejścia *f*. Każdy z tych elementów cechujących sieć neuronowych może ulec zmianie w wyniku działania operatorów algorytmu ewolucyjnego.

Mutacja może być realizowana za pomocą: zmiany funkcji przejścia w neuronie na inną, zmiany parametrów funkcji przejścia w neuronie, zwiększenia lub zmniejszenia wartości wylosowanej wagi synaptycznej lub biasa, a także zmiany liczby neuronów w warstwie. Nie przewiduje się natomiast zmiany liczby warstw. Ponadto stosowana jest mutacja adaptacyjna polegająca na wykorzystaniu metody Levenberga-Marquardta do lokalnej poprawy parametrów sieci, przy czym wzorcami są rozwiązania niezdominowane w populacji. Powyższe podejście jest wzorowane na wykorzystaniu antygenów w systemie immunologicznym do algorytmu optymalizacji wielokryterialnej przedstawionego w [17].

1. BEGIN
2. t:=0, ustaw liczebność populacji L, czas treningu pojedynczej sieci w populacji t_{net}, prawdopodobieństwo mutacji p_m:=1/ML, M – liczba współrzędnych rozwiązania.
3. ustaw rozmiar wektora wejściowego $\boldsymbol{p}$, wczytaj treningowe dane wejściowe do sieci i wzorce wyjściowe
4. wygeneruj losowo populację początkową sieci neuronowych $P(t)$ za pomocą wyznaczenia wartości macierzy wag $\boldsymbol{LW}$, $\boldsymbol{IW}$ i wektora przesunięcia $\boldsymbol{b}$
5. podaj dane wejściowe na wejścia sieci w populacji, odczytaj wyjścia $x \in P(t)$, oblicz rangi $r(x)$ i przystosowanie $f(x), x \in P(t)$
6. *finish*:=FALSE
7. WHILE NOT *finish* DO
8. BEGIN /* nowa populacja */
9. t:=t+1, $P(t) := \varnothing$
10. wyznacz prawdopodobieństwo selekcji $p_s(x), x \in P(t-1)$
11. FOR L/2 DO
12. BEGIN /* cykl reprodukcyjny */
13. selekcja turniejowa w dwu kategoriach pary (a,b) z $P(t-1)$
14. krzyżowanie (a,b) z tempem $p_c := e^{-t/T_{\max}}$
15. mutacja potomków (a',b') z tempem p_m
16. mutacja metodą Levenberga-Marquardta w odniesieniu do sieci o parametrach (a',b') w czasie t_{net} z tempem 0,1 p_m
16. $P(t) = P(t) \cup (a',b')$
17. END
18. podaj dane wejściowe na wejścia sieci w populacji, odczytaj wyjścia $x \in P(t)$, wyznacz $r(x)$ i $f(x), x \in P(t)$
19. IF ($P(t)$ jest zbieżna OR $t \geq T_{\max}$) THEN finish:=TRUE
20. END

Rys. 4.25. Pseudokod algorytmu genetycznego MEA^{2}N^2 działającego na populacji sieci neuronowych
Źródło: opracowanie własne.

Krzyżowanie przeprowadzane jest za pomocą wymiany odpowiednich części chromosomów, które modelują elementy sieci neuronowej. Punkt krzyżowania wybiera się „w pionie” między warstwami sieci lub „w poziomie” między neuronami.

Porównując omówione algorytmy, należy zauważyć, że algorytm neuronowo-genetyczny MEA^2N^2 cechuje się największą złożonością czasową, a zaimplementowany na komputerze jednoprocesorowym nie przynosi satysfakcjonujących wyników, głównie za sprawą zbyt niskiej wydajności, a także znaczącej liczby wejść w sieciach neuronowych w wypadku rozwiązywania instancji o większych rozmiarach. Jednakże wykorzystanie tego algorytmu w systemie rozproszonym przyniosłoby znaczącą poprawę rezultatów.

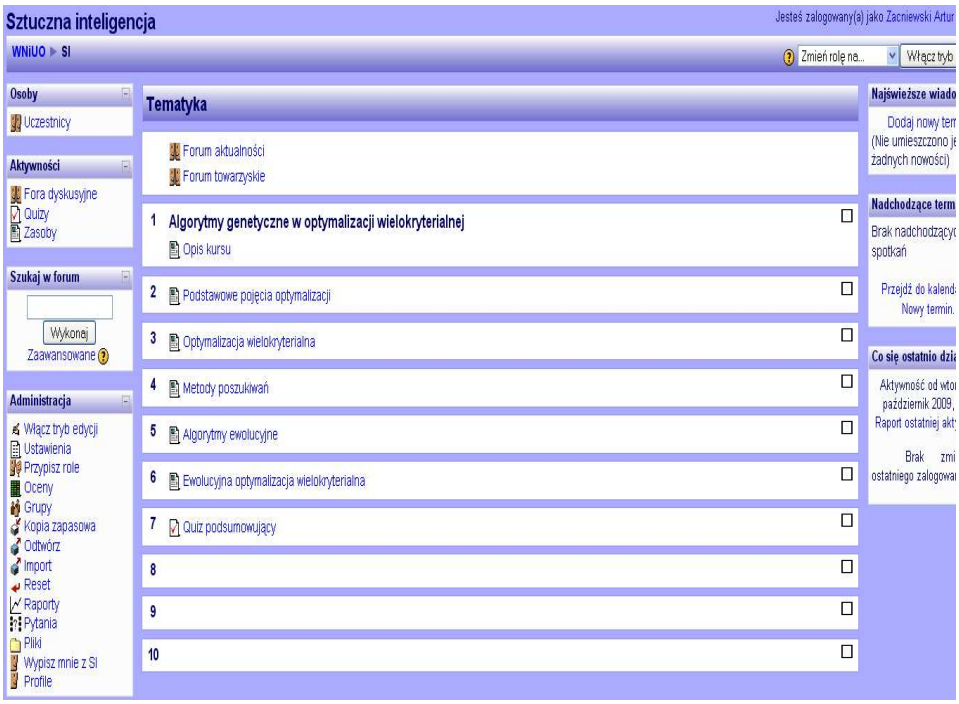
Algorytm NSGA-III pozwala na osiągnięcie wyników o wyższej jakości przy założonej liczbie obliczeń funkcji sprawności niż NSGA-II. Wyniki uzyskane za pomocą NSGA-III są także wyższej jakości niż wyniki otrzymane z algorytmu AMEA w wersji bez mutacji tabu oraz w wypadku binarnego kodowania chromosomu. Wyniki z algorytmu AMEA mogą być nieco wyższej jakości niż w odniesieniu do algorytmu NSGA-III, ale jest to związane z dodatkowym kosztem obliczeniowym będącym konsekwencją zastosowania mutacji tabu. W wypadku rozważanych algorytmów istotne przyśpieszenie można uzyskać stosując zrównoleglenie obliczeń.

4.6. Zastosowania systemu *MOODLE* do szkolenia studentów z tematu *Algorytmy genetyczne w optymalizacji wielokryterialnej* w ramach przedmiotu *Sztuczna Inteligencja*

Pod adresem internetowym [173] dostępny jest kurs omawiający algorytmy genetyczne w optymalizacji wielokryterialnej (rys. 4.26).

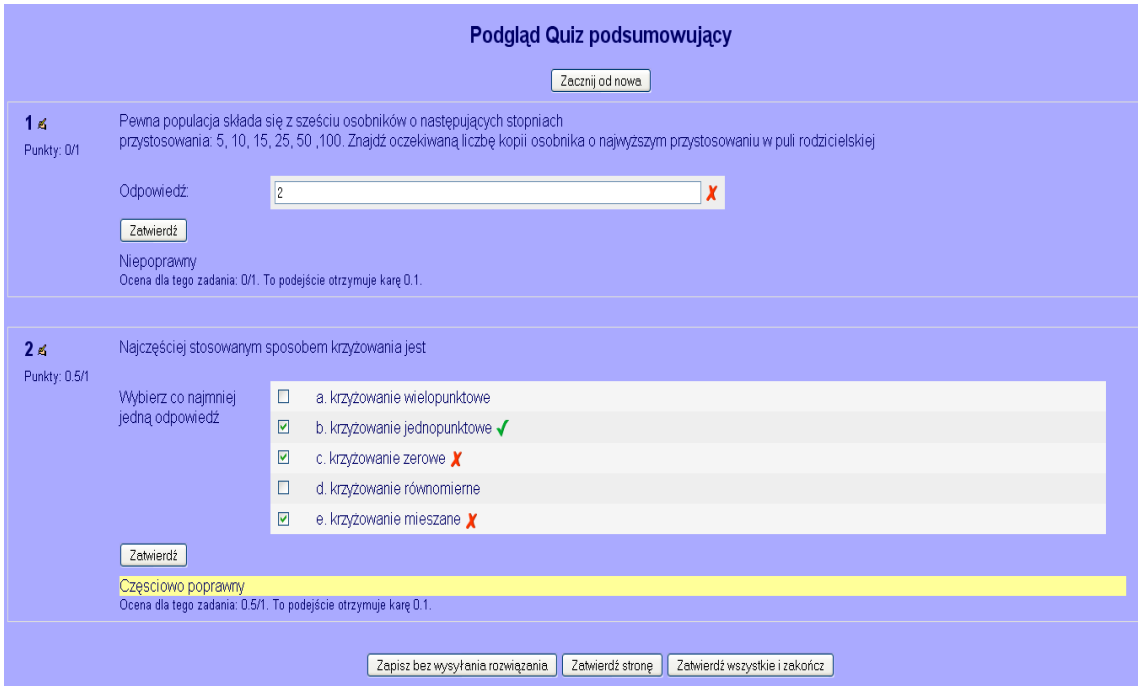
Kurs wykorzystywany był do wspomagania przedmiotu *Sztuczna Inteligencja* na kierunku *Informatyka* studiów I stopnia w AMW. W ramach kursu omówiono podstawowe pojęcia optymalizacji, optymalizację wielokryterialną, metody poszukiwań rozwiązań optymalnych, a także algorytmy ewolucyjne i ich wykorzystanie w optymalizacji wielokryterialnej.

Dostęp do kursu możliwy jest dla użytkowników posiadających odpowiedni klucz dostępu.



Rys. 4.26. Interfejs graficzny kursu *Sztuczna Inteligencja* zrealizowanego w AMW za pomocą systemu *MOODLE*
Źródło: opracowanie własne.

Opracowano quiz podsumowujący kurs, a przykładowe pytania przedstawiono na rysunku 4.27.



Rys. 4.27. Quiz do kursu zrealizowanego w AMW za pomocą systemu *MOODLE*
Źródło: opracowanie własne.

4.7. Wnioski i uwagi

Algorytmy ewolucyjne mogą skutecznie wyznaczać reprezentację rozwiązań optymalnych w sensie Pareto. Realizują one operacje na zbiorze rozwiązań, a nie na pojedynczych wariantach, przy wykorzystaniu operacji genetycznych, takich jak selekcja, krzyżowanie czy mutacja. Chromosomy mogą być reprezentowane za pomocą złożonych struktur danych, np. modelujących sztuczne sieci neuronowe, a nie tylko za pomocą jednowymiarowej tablicy z binarnymi elementami.

Do wyznaczania rozwiązań problemów optymalizacji wielokryterialnej opracowano algorytm NSGA-III. Koniecznym było zmodyfikowanie algorytmu NSGA-II, gdyż wartość tempa mutacji i innych ustawionych domyślnie parametrów nie umożliwiały wyznaczania reprezentacji rozwiązań optymalnych w sensie Pareto nawet dla relatywnie prostych problemów w założonym czasie.

W szczególności opracowano funkcję *instances_generator*, która na podstawie macierzy przepływu $\boldsymbol{\Omega}$, a także wektorów $\boldsymbol{B}$, $\boldsymbol{E}$ i $\boldsymbol{\Theta}$ generuje wszystkie instancje danego diagramu przepływu.

W celu przyspieszenia przeglądu przydziałów opracowano funkcję *preview_accelerator*, która wybiera do analizy tylko przydziały spełniające założenia (2.4) i (2.6). Zamiast analizy $2^{(V+J)I}$ przydziałów w przypadku pełnego przeglądu, gdy nie używa się funkcji *preview_accelerator*, do analizy wybieranych jest tylko $I^V J^I$ przydziałów.

Czas wyznaczania reprezentacji rozwiązań optymalnych przy wykorzystaniu wielokryterialnych algorytmów ewolucyjnych został znacząco zredukowany w porównaniu do metody pełnego przeglądu.

W procesie optymalizacji istnieje możliwość wykorzystania algorytmu genetycznego działającego na populacji sieci neuronowych $\mathrm{MEA}^2\mathrm{N}^2$, a także algorytmu adaptacyjnego AMEA. Wyniki najwyższej jakości uzyskano jednak za pomocą algorytmu NSGA-III przy ustalonej liczbie oszacowań funkcji sprawności.

Ze zbioru rozwiązań optymalnych w sensie Pareto wyznaczonego za pomocą algorytmu ewolucyjnego można otrzymać rozwiązanie kompromisowe dla ustalonej wartości parametru p. Szczególne atrakcyjne jest wykorzystanie normy Euklidesa dla p=2. Można także zastosować metodę interaktywną z decydentem w celu zawężenia zbioru alternatyw. Alternatywnie stosowane jest wykorzystanie dodatkowego kryterium w celu dokonania ostatecznej selekcji.

PODSUMOWANIE

Oryginalnym dorobkiem autora jest przedstawiony w rozprawie system metodologiczny do wyznaczania i oceny przydziałów modułów programów w rozproszonym systemie informatycznym w odniesieniu do systemu *MOODLE*, który może wspomagać zdalne nauczanie i szkolenie wojskowe. Opracowano także modele matematyczne funkcjonowania rozproszonych systemów komputerowych, na podstawie których sformułowano zadania optymalizacji wektorowej. W zakresie metod rozwiązania sformułowanych problemów, główny nacisk położono na zastosowanie algorytmów ewolucyjnych. Aplikacje napisane w języku programowania Matlab umożliwiają powtórzenie załączonych wyników badań i mogą być zastosowane do wyznaczania rozwiązań zagadnień wielokryterialnej optymalizacji przydziałów modułów do komputerów.

Warto również podkreślić, że sformułowane zagadnienia optymalizacji wektorowej oraz opracowane metody ich rozwiązania za pomocą wielokryterialnego algorytmu ewolucyjnego NSGA-III oraz hybrydowego algorytmu neuronowo-ewolucyjnego z populacją sieci neuronowych MEA^2N^2 stanowią oryginalny dorobek autora. Umożliwiają, obok adaptacyjnego wielokryterialnego algorytmu ewolucyjnego AMEA, poszukiwanie rozwiązań optymalnych w sensie Pareto, na podstawie których można wyznaczyć warianty leksykograficzne lub alternatywy kompromisowe dla wybranych wartości parametru *p*.

Stosując procedurę nadawania rang, wykorzystano algorytmy ewolucyjne do wyznaczania rozwiązań suboptymalnych w sensie Pareto dla szerokiej klasy zadań optymalizacji przydziałów modułów programistycznych. Dokonano weryfikacji opracowanych modeli za pomocą wyznaczania alternatyw dla szeregu instancji testowych. Uzyskane wyniki potwierdziły słuszność przeprowadzonych rozważań teoretycznych w odniesieniu do zbudowanych modeli, sformułowanych zagadnień optymalizacji i zaproponowanych metod. Wielowariantowe eksperymenty numeryczne wykazały poprawność logiczną modeli i zadań optymalizacji, ich informacyjną spójność i adekwatność do modelowanych sytuacji decyzyjnych.

Skonstruowane modele i metody optymalizacji, które bazują na algorytmach ewolucyjnych, uzyskane wyniki oraz wyciągnięte na ich podstawie wnioski stanowią przesłankę do zasadnego stwierdzenia, że sformułowany na wstępie problem badawczy został pomyślnie rozwiązany. Wnioskować można stąd, że postawiona hipoteza robocza

została naukowo zweryfikowana z pozytywnym skutkiem. Tak więc założony cel pracy został osiągnięty zarówno w aspekcie ogólnym, jak też w węższym aspekcie metodologicznym.

Rozważane modele, sformułowane zadania optymalizacji oraz opracowane metody stanowią wkład autora do teorii optymalizacji przydziałów, a także mają istotne znaczenie praktyczne. Mogą być stosowane do projektowania systemów rozproszonych wspomagających szkolenie wojskowe, w których rozdział modułów programowych w systemach klasy *MOODLE* decyduje zwiększeniu jego dostępności, a także miary skutecznej realizacji zadań w wyznaczonych terminach. Istotne ograniczenia mogą być nałożone na równomierność obciążenia komputerów, koszt budowy systemu lub jego wydajność. Opracowane metody są przeznaczone do wyznaczania rozwiązań dla szerokiej klasy zagadnień optymalizacji wektorowej.

Niniejsza praca nie wyczerpuje bogatej tematyki dotyczącej optymalizacji przydziałów modułów w rozproszonych systemach szkolenia wojskowego na platformie *MOODLE* za pomocą algorytmów ewolucyjnych. Nie zamyka również procesu badawczego w rozważanym zakresie. Zaprezentowane modele, sformułowane zadania optymalizacji, opracowane metody i uzyskane wyniki stanowią podstawę do dalszych ukierunkowanych badań nad tymi zagadnieniami. Podjęte rozważania stworzyły nowe możliwości kontynuacji dalszych naukowych poszukiwań z uwzględnieniem problemów w pracy pominiętych lub modelowo uproszczonych.

Zbudowane modele i związane z nimi rozważania nie są pozbawione ograniczeń i mankamentów. Ograniczenia scharakteryzowano w trakcie konstruowania poszczególnych modeli przetwarzania rozproszonego, na etapie formułowania założeń. Jednym z mankamentów jest pominięcie interesującego problemu optymalnego rozmieszczenia modułów w macierzy komputerów, w celu minimalizacji liczby obciążonych połączeń między komputerami. W powyższym wypadku istnieje jednak możliwość wykorzystania opracowanego algorytmu pod warunkiem skonstruowania adekwatnych modeli matematycznych i sformułowania odpowiednich zadań optymalizacji.

Zdeterminowane cele badawcze i ograniczone ramy pracy nie pozwoliły na rozwinięcie innych ciekawych planów badawczych z zakresu szeroko rozumianej teorii optymalizacji przydziałów modułów w rozproszonych systemach komputerowych wspierających szkolnictwo wojskowe. Mogą być one jednak kontynuowane na podstawie niniejszego opracowania.

Zaprezentowane modele, metody i towarzyszące im implementacje komputerowe były dotychczas stosowane przez autora w ramach prac naukowo-badawczych prowadzonych w Akademii Marynarki Wojennej, szkoleń wojskowych w Centrum Wsparcia Teleinformatycznego i Dowodzenia Marynarki Wojennej RP. Wybrane modele i metody zaprezentowano na konferencjach naukowych. W prowadzonej działalności szkoleniowej wykorzystano modele sieci neuronowych oraz algorytmy ewolucyjne.

Atrakcyjnym kierunkiem badawczym w teorii i zastosowaniach optymalizacji jest rozwijanie hybrydowych metod neuronowo-ewolucyjnych. Ta dziedzina zastosowań wymaga opracowania odpowiednich modeli, sformułowania zadań, dokonania modyfikacji skonstruowanych metod oraz przeprowadzenia czasochłonnych eksperymentów numerycznych, co znacznie wykracza poza założone w pracy ramy i cele badawcze.

Zasadne jest zatem stwierdzenie, że rozwój problematyki dotyczącej teorii i zastosowań algorytmów ewolucyjnych w zagadnieniach projektowania systemów informatycznych wspomagających szkolnictwo wojskowe jest istotnym zadaniem badawczym w zakresie wzrostu obronności kraju.

BIBLIOGRAFIA

1. Abe S.: *Convergence acceleration of the Hopfield neural network by optimizing integration step sizes*. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26, No. 1, February 1996, pp. 194-201.
2. Abelson H., Sussman G. & J.: *Struktura i interpretacja programów komputerowych*, WNT, Warszawa 2002.
3. Ajith P.T., Murthy C. S. R.: *Algorithms for reliability-oriented module allocation in distributed computing systems*, Journal of System Software, Vol. 40, No. 2, 1998, pp. 125-138.
4. Ajith A. P., Murthy C. S. R.: *An improved algorithm for module allocation in distributed computing systems*, Journal of Parallel and Distributed Computing, Vol. 42, No. 1, November 1997, pp. 82-90.
5. Alberto I., Azcarate C., Mateo P.M.: *Multiobjective evolutionary algorithms. Pareto rankings.* Monografias del Seminario Matematico Garcia de Galdeano, Vol. 27, 2003, pp. 27-35.
6. Ameljańczyk A.: *Optymalizacja wielokryterialna*. WAT, Warszawa, 1986.
7. Ameljańczyk A.: Op*tymalizacja wielokryterialna w problemach sterowania i zarządzania*. PAN, Warszawa, 1984.
8. Ameljańczyk A.: *Teoria gier i optymalizacja wektorowa*. WAT, Warszawa, 1980.
9. Ameljańczyk A.: *Elementy optymalizacji wielokryterialnej*, WAT, Warszawa, 1979.
10. Andrews G. R.: *Concurrent programming. Principles and practice.* The Benjamin/Cummings Publishing Company, Redwood City 1991.
11. Antkiewicz R., Najgebauer A., Tarapata Z., Rulka J., Pierzchała D., Kulas W., Rekowski R.: *Case-based C2 modelling and effective development, implementation and experimentation for simulation based operational training support system.* In: Proceedings of the Int. Conf. on the Effectiveness of Modelling and Simulation – From Anecdotal to Substantive Evidence, Neuilly-sur-Seine, France, 2005, pp. 6-1-6-16.
12. Arabas J.: *Wykłady z algorytmów ewolucyjnych*. WNT, Warszawa 2004.
13. Babbar M., Lakshmikantha A., Goldberg D.: *A modified NSGA-II to solve noisy multiobjective problems*. In: Foster J. et al. (eds.): Genetic and Evolutionary Computation Conference (GECCO'2003), Chicago, Illinois, USA, Lecture Notes in Computer Science, Vol. 2723, 2003, pp. 21–27.

14. Bacon J: *Concurrent systems.* Addison-Wesley, Wokingham 1993.
15. Balicki J.: *Multi-criterion decision making by artificial intelligence techniques*, Proceedings of the 8th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, Cambridge 20–25 February 2009, WSEAS Press, pp. 319-325.
16. Balicki J.: *Some numerical experiments on multi-criterion tabu programming for finding Pareto-optimal solutions*, WSEAS Transactions on Systems, Vol. 8, 2009, pp. 241-250.
17. Balicki J.: *Multicriterion evolutionary algorithm for workload balancing of the web bank servers*, International Journal of Computer Science and Network Security, Vol. 6, No.10, October 2006.
18. Balicki J.: *Immune systems in multi-criterion evolutionary algorithm for task assignments in distributed computer system*, Lectures Notes in Computer Science, Vol. 3528, 2005, pp. 51-56.
19. Balicki J.: *Multicriterion evolutionary algorithm with model of the immune system to handle constraints for task assignments*, Lecture Notes in Computer Science. Subserie: Lecture Notes in Artificial Intelligence, Vol. 3070, 2004, pp. 394-399.
20. Balicki J.: *Multi-criterion optimisation of distributed system performance by evolutionary task assignments*, Journal of Research and Practice in Information Technology, Vol. 33, No. 3, pp. 173-185, May 2001.
21. Balicki J.: *Algorytmy ewolucyjne oraz algorytmy przeszukiwania tabu do optymalizacji przydziałów modułów programów w rozproszonych systemach komputerowych*, Wyd. AMW, Gdynia 2000.
22. Balicki J., Kitowski Z.: *Genetic and evolutionary algorithms for finding efficient program module allocations*. In: Mastorakis N. E.: *Advances in intelligent systems and computer science*, World Scientific and Engineering Society Press, Danvers, USA, 1999, pp. 151-156.
23. Balicki J.: *Evolutionary algorithms using artificial neural networks for solving multiobjective optimization problems.* Proceedings of the 9th International Symposium on System-Modelling-Control, Technical University of Łódź, Polish Cybernetical Society, Polish Society of Computer Simulation, Polish Society of Medical Informatics, Zakopane, April 1998, CD-ROM.
24. Balicki J., Kitowski Z.: *Multicriteria optimization of computer resource allocations with using genetic algorithms and artificial neural networks*. Proceedings of the 12th

International Conference on Systems Science, Vol. III, September 1995, Wrocław, Poland, pp. 11-18.

25. Balicki J.: *Artificial neural networks for multicriteria optimization problems of program module allocations*. Proceedings of 8th International Symposium on System-Modelling-Control, Technical University of Łódź, Polish Cybernetical Society, Polish Society of Computer Simulation, Polish Society of Medical Informatics, Vol.3, Zakopane, May 1995, pp. 1-6.
26. Berger P. L., Luckmann T.: *The social construction of reality: A treatise in the sociology of knowledge*. Anchor Books, Garden City, New York 1966.
27. Bharat D., Niraj K.: *Hardware-software co-synthesis of hierarchical heterogeneous distributed embedded systems,* United States Patent, No. 6289488, September 2001.
28. Błażewicz J.: *Problemy optymalizacji kombinatorycznej - Złożoność obliczeniowa. Algorytmy aproksymacyjne.* PWN, Warszawa 1986.
29. Błażewicz J., Cellary W., Słowiński R., Węglarz J.: *Badania operacyjne dla informatyków*. WNT, Warszawa 1983.
30. Bokhari S. H.: *Partitioning problems in parallel, pipelined, and distributed computing.* IEEE Transactions on Computers, Vol. C-37, No. 1, January 1988, pp. 48-57.
31. Bokhari S. H.: *Assignment problems in parallel and distributed computing.* Kluwer Academic Publishers, Boston 1987.
32. Bokhari S. H.: *A shortest tree algorithm for optimal assignments across space and time in a distributed processor system.* IEEE Transactions on Software Engineering, Vol. SE-7, No. 6, November 1981, pp. 583-589.
33. Brzeziński J., Sobaniec C.: *Systemy rozproszone*, Wyd. Politechniki Poznańskiej, Poznań 2006.
34. Brzeziński J., Palak M., Szychowiak M.: *Problemy konstrukcji systemów rozproszonych tolerujących uszkodzenia.* Raport RB-018/98, Instytut Informatyki Politechniki Poznańskiej, Poznań 1998.
35. Carlier F., Renault V.: *An epiphyte architecture to enrich content in an existing educational website*. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, Australia, December 2008, pp. 135-138.
36. Chang S. K.: *A model for distributed computer system design*. IEEE Transactions on Systems, Man, and Cybernetics. Vol. smc-5, No. 6, May 1976, pp. 344-359.

37. Chaudhary V., Aggarwal J.K.: *A generalized scheme for mapping parallel algorithms*. IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 3, March 1993, pp. 328-346.
38. Chen C., Cherkassky V.: *Task allocation and reallocation for fault tolerance in multicomputer systems*, IEEE Transaction on Aerospace and Electronic Systems, Volume 30, Issue 4, October 1994, pp. 1094 – 1104.
39. Choi S., Chisu W.: P*artitioning and allocation of objects in heterogeneous distributed environments using niched Pareto genetic-algorithm*, In *Proc. of 1998 Asia Pacific Software Engineering Conference (APSEC 98)*, Taipei, Taiwan, December 1998, pp. 322-329,
40. Chu W. W., Lan L. M. T.: *Task allocation and precedence relations for distributed real-time systems.* IEEE Transactions on Computers, Vol. C-36, No. 6, June 1987, pp. 667-679.
41. Clarke A.: *E-learning - nauka na odległość*. WKiŁ, Warszawa 2007.
42. Coello Coello C.A., Schuetze O., Laumanns M., Tantar E.: *Computing gap-free Pareto front approximations with stochastic search algorithms*, Evolutionary Computation, Vol. 18, 2010, pp. 65-96.
43. Coello Coello C.A., Jaimes A.L., Quintero L.V.S.: *Ranking methods in many-objective evolutionary algorithms*. In: Raymond Chiong (editor), *Nature-inspired algorithms for optimisation*, Springer, New York 2009, pp. 413-434.
44. Coello Coello C.A., Lamont G., Veldhuizen D.: *Evolutionary algorithms for solving multi-objective problems*, Springer, New York 2007 (second edition).
45. Coello Coello C.A., Lamont G., Veldhuizen D.: *Evolutionary algorithms for solving multi-objective problems*, Springer, New York 2002 (first edition).
46. Comer D. E., *Sieci komputerowe i intersieci*, Wydawnictwa Naukowo-Techniczne, Warszawa 2003.
47. Corne D. W., Knowles J. D., Oates M. J.: *The Pareto envelope-based selection algorithm for multiobjective optimization*. In: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, 14. E. Lutton, J. J. Merelo, and H.-P. Schwefel (editors), Proceedings of the VI Conference on Parallel Problem Solving from Nature, Paris, September 2000, Lecture Notes in Computer Science, Vol. 1917, 2000, pp. 839–848.
48. Coulouris G., Dollimore J., Kindberg T.: *Distributed systems: Concepts and design*. Addison-Wesley, Harlow 2005.

49. Deb K.: *Multi-objective optimization using evolutionary algorithms*, John Wiley & Sons, New York 2001.
50. Deb K., Agrawal S., Pratap A. and T. Meyarivan: *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. In: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, (Eds.), Proceedings of the VI Conference on Parallel Problem Solving from Nature, Paris, September 2000, Lecture Notes in Computer Science, Vol. 1917, 2000, pp. 849–858.
51. Deb K., Goldberg D. E.: *An investigation of niche and species formation in genetic function optimization*. In: J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, June 1989, pp. 42–50.
52. Deo N.: *Teoria grafów i jej zastosowania w technice i informatyce*. WNT, Warszawa 1981.
53. Donoso Y., Fabregat R.: *Multi-objective optimization in computer networks using metaheuristics*, Auerbach Publications, Boca Raton 2007.
54. Doty K. W., McEntire P. L., O'Reilly J. G.: *Task allocation in a distributed computer system.* Proceedings of the IEEE Infocom, Las Vegas, Nevada, USA, 1982, pp. 33-38.
55. Duch W., Korbicz J., Rutkowski L., Tadeusiewicz R.: *Sieci neuronowe*. AOW Exit, Warszawa, 2000.
56. Edmonds J.: *Paths, trees and flowers.* Canadian Journal of Mathematics, Vol. 17, 1965, pp. 449-467.
57. Even S.: *Graphs algorithms*. Computer Science Press, Potomac 1979.
58. Farina M., Amato P.: *Fuzzy optimality and evolutionary multiobjective optimization*. Proceeding of the Second International Conference on Evolutionary Multi-Criterion Optimization, Faro, Portugal, April 2003, Lecture Notes in Computer Science. Vol. 2632, pp. 58–72,
59. Fisher N., Anderson J., Baruah S.: *Task partitioning upon memory-constrained multiprocessors*, Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05), Hong Kong, August 2005, pp. 416-421.
60. Foley J. D., Van Dam A.: *Fundamentals of interactive computer graphics*. Addison-Wesley, Massachusetts 1982.
61. Fonseca C. M. and Fleming P. J.: *Genetic algorithms for multiobjective optimization: formulation, discussion and generalization*, In: S. Forrest (Ed.), Proceedings of the

Fifth International Conference on Genetic Algorithms, San Mateo, University of Illinois at Urbana-Champaign, California, June 1993, pp. 416–423.

62. Ford L. R. jr, Fulkerson D. R.: *Przepływy w sieciach*. PWN, Warszawa 1969.
63. Funabiki N., Kitamichi J., Nishikawa S.: *An evolutionary neural network algorithm for max cut problems*. Proceedings of the 1997 International Conference on Neural Networks, Vol. 2, Houston, USA, June 1997, pp. 1260-1265.
64. Garey M. R., Johnson D. S, Stockmayer L.: *Some simplified NP-complete problems*. Theoretical Computer Science, No. 1, 1971, pp. 237-267.
65. Giaro K.: *Wybrane zastosowania niestandardowych modeli kolorowania w szeregowaniu dwuprocesorowych zadań jednostkowych*, Automatyka, 2003, tom 7, ss. 105-111.
66. Gliszczyński P.: S*tudium przypadku: e-learning w BOT Elektrowni Bełchatów S.A*. „e-mentor", Nr 3, Vol. 15, czerwiec 2006, s. 24.
67. Glover F., Laguna M.: *Tabu search.* Kluwer Academic Publishers, Boston 1997.
68. Goczyła K., Ciarkowski J.: *Wydobywanie wiedzy i klasyfikacja danych w systemie typu Web Forming*, W: *Electronic commerce: teoria i zastosowanie*, Wyd. Politechniki Gdańskiej, Gdańsk 2005, ss. 69-79.
69. Goldberg D. E.: *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
70. Gościński A.: *Distributed operating systems. The logical design*. Addison Wesley, Boston 1991.
71. Górnikiewicz J.: *Szkoła przez Internet wyzwaniem dla polskiej tradycji oświatowej: nadzieje i niepokoje*. W: A. W. Mitas (red.): Media i edukacja w aspekcie globalizacji, Wyd. Uniwersytetu Śląskiego, Cieszyn 2003.
72. Gulińska H., Bartoszewicz M.: *Tablica interaktywna środkiem wspomagającym nauczania*. „e-mentor", nr 1, Vol. 18, luty 2007, s. 27.
73. Hernandez J.C. Chavez M. A.: *Moodle security vulnerabilities,* Proceeding of the 5th Int. Conference on Electrical Engineering, Computing Science and Automatic Control (CCE 2008), Mexico City, November 2008, pp. 352–357.
74. Horn J. and Nafpliotis N.: *Multiobjective optimization using the niched Pareto genetic algorithm*, Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.
75. Horton W. & K.: *E-learning tools and technologies*, Wiley Publishing, New York 2003.

76. Hou C. J., Shin K. G.: *Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems.* IEEE Transactions on Computers, Vol. 45, No. 12, December 1997, pp. 1338-1356.
77. Hsu C., Chung Y., Yang D., Dow C.: *A generalized processor mapping technique for array redistribution.* IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 7, July 2001, pp. 743-757.
78. Hu T. C.: *Combinatorial algorithms.* Addison-Wesley, Reading, Massachusetts, 1982.
79. Huang Y., Tripathi S.K.*: Resource allocation for primary-site fault-tolerant systems.* IEEE Transactions on Software Engineering, Vol.19, No.2, February 1993, pp. 108-119.
80. Hura G. S., Singhal M.: *Data and computer communications. Networking and internetworking*. CRC Press LLC, Boca Raton, Florida, 2001.
81. Hyla M.: *Przewodnik po e-learningu*. Wolters Kluwer, Łódź 2007.
82. Iqbal M. A.: *Approximate algorithms for partitioning and assignment problems.* ICASE Report 86-40 NASA Contractor Report 178130, June 1986.
83. Jaimes A. L., Quintero L. V. S. and Coello Coello C. A.: *Ranking methods in many-objective evolutionary algorithms*. In: Raymond Chiong (ed.), *Nature-Inspired Algorithms for Optimisation*, Springer, New York 2009.
84. Jara-Roa D. *et al*.: *An adaptive multi-agent based architecture for engineering education*, Proceeding of the IEEE EDUCON Education Engineering Conference on The Future of Global Learning Engineering Education, Madrid, April 2010, pp. 217-222.
85. Jennings, N. R., Wooldridge, M.: *Applications of intelligent agents.* In: Jennings N. R., Wooldridge M.: *Intelligent agents*, Springer-Verlag New York, Inc., Secaucus, 1998, pp. 3-28.
86. Jia W., Zhou W.: *Distributed network systems – from concepts to implementation*. Springer Science, New York 2005.
87. Jian L., Pengkun L.: *A personalized learning platform based on multi-agent and MOODLE*. Proceedings of International Conference on Computational Intelligence and Software Engineering, Wuhan, December 2009, pp. 1-4.
88. Kafil M., Ahmad I.: *Optimal task assignment in heterogeneous distributed computing systems.* IEEE Concurrency, Vol. 6, No. 3, 1998, pp. 42-51.

89. Kalns E.T., Ni L.M.: *Processor mapping techniques toward efficient data redistribution*. IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 12, December 1995, pp. 1234-1247.
90. Kartik S., Murthy C. S. R.: *Task allocation algorithms for maximizing reliability of distributed computing systems.* IEEE Transactions on Computers, Vol. C-37, No. 1, January 1988, pp. 48-57.
91. Konakia H. R., Tobagi F. A.: *On distributed computations with limited resources.* IEEE Transactions on Computers, Vol. 48, No. 6, June 1997, pp. 719-724.
92. Kopetz H.: *Design principles for distributed embedded applications,* Kluwer Academic Publishers, Boston 1997.
93. Krawczyk H., Rek A.: *Methodology for developing Web-based applications from reusable components using open-source tools*, Proceedings of the 2nd International Conference on Information Technology, ICIT 2010, June 2010, Gdansk, pp. 117-120.
94. Kubiak M.: *Wirtualna edukacja*. Mikom, Warszawa 2000.
95. Kuhn H. W.: *The Hungarian method for the assignment problem,* Naval Research Logistic Quarterly, Vol. 2, 1955, pp. 83-97.
96. Kulikowski J. L.: *Zarys teorii grafów*. PWN, Warszawa 1986.
97. Kunkle D.: *A summary and comparison of MOEA algorithms,* Northeastern University Report, Boston, Massachusetts, 2005.
98. Lamont G. B.: *Multi-objective evolutionary algorithms: what, why and where. A tutorial,* Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization, Guanajuato, Mexico, March 2005, pp. 23-35.
99. Landowska A.: *Rola agentów edukacyjnych w środowiskach zdalnego nauczania,* Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, nr 25, 2008, str. 83-86.
100. Latif N. A.: *Automated notification and document downloading in e-learning – Development of an agent-based framework utilizing the push-pull technology interaction policy*, Proceedings of the International Symposium on Information Technology, Kuala Lumpur, September 2008, pp. 1-7.
101. Lawler E.: *Combinatorial optimization: networks and matroids.* Holt, Rinehart & Wiston Pub., New York 1981.
102. Ledgard H. F.: *Mała księga programowania obiektowego.* WNT, Warszawa 1998.

103. Legrand A., Renard R., Robert Y., Vivien V.: *Mapping and load-balancing iterative computations.* IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 6, June 2004, pp. 546-558.
104. Liu S.F, Soffa M.L.: *Parallel task assignment by graph partitioning*. Lecture Notes in Computer Science, Vol. 605, 1992, pp. 965-966.
105. Lo V.M.: *Heuristic algorithms for task assignment in distributed systems.* IEEE Transactions on Computers, Vol. C-37, No. 11, November 1988, pp. 1384-1397.
106. Lobo F.G., Lima C.F., Michalewicz Z.: *Parameter setting in evolutionary algorithms*, Studies in Computational Intelligence, Vol. 54, 2007, pp. 1-18.
107. Malina W., Szwoch M.: Metodologia i techniki programowania. PWN, Warszawa 2008.
108. Marchand A., Silly-Chetto M.: *Dynamic scheduling of skippable periodic tasks: issues and proposals*, Journal of Software, Vol. 2, No. 5, November 2007, pp. 44-51.
109. Marciniak J.: *E-learning: technologia, zastosowania, korzyści*. Materiały Krajowej Konferencji *E-learning: Analiza Rozwiązań i Wdrożeń*, Poznań, grudzień 2002, ss. 17-24.
110. Marwedel P.: *Embedded system design*. Springer, Dortmund 2006.
111. Michalewicz Z.: *Algorytmy genetyczne + struktury danych = programy ewolucyjne.* WNT, Warszawa 1996.
112. Mierzejewska B.: *Ciekawa inicjatywa IMS Global Learning Consortium.* „e-mentor", Nr 3, Vol. 20, kwiecień 2007, s. 29.
113. Montusiewicz J.: *Ewolucyjna analiza wielokryterialna w zagadnieniach technicznych*, Wyd. Instytut Podstawowych Problemów Techniki PAN, Warszawa 2004.
114. Moore J., Churchward M.: *Moodle 1.9. Extensions Development*, Packt Publishing, Birmingham 2010.
115. Morrison D.: *E-learning strategies*. Wiley & Sons, Chichester 2003.
116. Mucha M., *Sieci komputerowe. Budowa i działanie*, Helion, Gliwice 2003.
117. Murthy L., Seo P. K.: *A dual-descent procedure for the file allocation and join site selection problem on a telecommunications network*. An International Journal Networks, Vol. 33, No. 2, 1999, pp. 109-123.
118. Najgebauer A., Antkiewicz R., Pierzchała D., Tarapata Z.: *The automation of combat decision processes in the simulation based operational training support system,*

Proceedings of the IEEE Symposium on Computational Intelligence in Security and Defense Applications, CISDA 2007, Honolulu, USA, April 2007, pp. 145-152.

119. Nawrocki J. R., Łukasik E., Błażewicz J., Complak W., Pawlak G.: *Towards quality oriented education engineering.* Proceedings of the 4th Annual Conference on the Teaching of Computing, Dublin, 1999, pp. 96-99.
120. Nowicki K., Hasse M.: *Analiza możliwości rozszerzenia systemu zarządzającego nauczaniem opartego na licencji Open Source o implementację SCORM.* Wyd. Wyższa Szkoła Humanistyczno-Ekonomiczna, Łódź 2005.
121. Papadimitriou C. D., Steiglitz K.: *Combinatorial optimization: algorithms and complexity.* Prentice Hall, Englewood Cliffs, New York 1982.
122. Parker S.: *Dictionary of scientific and technical terms*, McGraw-Hill, New York 2002.
123. Plucińska A., Pluciński E.: *Elementy probabilistyki*, PWN, Warszawa 1990.
124. Price C. C., Pooch U. W.: *Search techniques for a nonlinear multiprocessor scheduling problem.* Naval Research Logistics Quarterly, Vol.29, No.2, June 1982, pp. 213-233.
125. Raidl G.: *An improved genetic algorithm for the multiconstrained 0-1 Knapsack Problem.* Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, May 1998, pp. I-207 - I-211.
126. Rao G. S., Stone H. S., Hu T. C.: *Assignment of tasks in a distributed processor system with limited memory.* IEEE Transactions on Computers, Vol. C-28, No. 4, April 1979, pp. 291-291.
127. Rasiowa H.: *Wstęp do matematyki współczesnej*. PWN, Warszawa 2007 (wyd. czternaste).
128. Rice W.H.: *From technologies to solutions*: *Moodle e-learning course development. A complete guide to successful learning using Moodle*, Packt Publishing, Birmingham 2006.
129. Russell S. J., Norvig P.: *Artificial intelligence: A modern approach*, Prentice Hall, New Jersey 2002.
130. Schneidewind N.: *Allocation and analysis of reliability: multiple levels: system, subsystem, and module*, Innovations in System and Software Engineering, Springer London, Vol. 2, No. 3-4, December 2006, pp. 121-136.
131. Schuetze O., Laumanns M., Tantar E., Coello Coello C. A.: *Computing gap-free Pareto front approximations with stochastic search algorithms*, Evolutionary Computation, Vol. 18, No. 1, 2010, pp. 65-96.

132. Shang W., Fortes J.A.B.: *On time mapping of uniform dependence algorithms into lower dimensional processor arrays*. IEEE Transactions on Parallel and Distributed Systems, Vol. 3, No. 3, May 1992, pp. 350-363.
133. Shatz S.M., Wang J.P.: *Models & algorithms for reliability-oriented task-allocation in redundant distributed-computer systems*, IEEE Transactions On Reliability, Vol. 38, No. 1, April 1989, pp. 16-27.
134. Sienkiewicz P.: *Analiza systemowa. Podstawy i zastosowania.* Bellona, Warszawa 1994.
135. Silberschatz A., Peterson J. L., Galvin P. B.: *Podstawy systemów operacyjnych.* WNT, Warszawa 1993.
136. Sinclair J. B.: *Optimal assignments in broadcast networks.* IEEE Transactions on Computers, No. 5, Vol. C-37, May 1988, pp. 521-531.
137. Singhal M., Shivaratri N. G.: *Advanced concepts in operating systems – distributed, database, and multiprocessor operating systems*. The McGraw-Hill Companies, Columbus 1994.
138. Sinha P. K.: *Distributed operating systems – concepts and design*. IEEE Press, San Francisco 1997.
139. Spragg M., Bacher R.: *LandWarNet network operations* (*NetOps*), US Army Chief Information Office G-6 Report, Fort Lauderdale, August 2007.
140. Srinivas N., Deb K.: *Multiobjective optimization using nondominated sorting in genetic algorithms*, Evolutionary Computation, No. 2, Vol. 3, 1994, pp. 221–248.
141. Stadnicki J.: *Teoria i praktyka rozwiązywania zadań optymalizacji*. WNT, Warszawa 2006.
142. Stecyk A.: *Charakterystyka funkcjonowania systemu LAMS*. „e-mentor", Vol. 14, nr 2, kwiecień 2006, s. 77.
143. Stone H. S.: *Critical load factors in two-processor distributed systems.* IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, May 1978, pp. 254-258.
144. Szymonowicz M. K., *Wykład na monitorze*. Ekoprofit, Vol. 79, nr 2, luty 2006, ss. 23-26
145. Świerczyński Z., Strzelecki Ł., Renczewski K., Walkusz M., Pancerow J.: *RSW – system zdalnego nauczania*, Biuletyn Instytutu Automatyki i Robotyki Wojskowej Akademii Technicznej, nr 22, 2005, ss. 55-70.
146. Tanenbaum A. S.: *Computer networks*. Prentice-Hall, New Jersey 2003.

147. Tanenbaum A. S., van Steen M.: *Distributed systems: Principles and paradigms*. Prentice-Hall, New Jersey 2006.
148. Thanikesavan S., Killat U. : *Global scheduling of periodic tasks in a decentralized real-time control system*, Proceedings of the IEEE International Workshop on Factory Communication Systems, September 2004, pp. 307-310.
149. Towsley D. P.: *Allocating programs containing branches and loops within a multiple processor system*. IEEE Transactions on Software Engineering, Vol. SE-12, October 1986, pp. 1018-1024.
150. Tsuchiya T, Kikuno T.: *A BDD-based approach to reliability-optimal module allocation in networks*, Technical report of IEICE, Osaka University, November 2008, pp. 25-30.
151. Varvarigou T., Trotter J.: *Module replication for fault-tolerant real-time distributed systems***,** IEEE Transaction on Reliability, Vol. 47, No. 1, 1998, pp. 8-18.
152. Waćkowski K., Chmielewski J.: *Rola standaryzacji platform w e-learningu*, „e-mentor", Vol. 19, nr 2, kwiecień 2007, s. 26.
153. Węglarz J., Słowiński R., Soniewicki B.: *MPS – komputerowy system wspomagania decyzji dla wielokryterialnego rozdziału zasobów*, Archiwum Automatyki i Robotyki, tom XXXVI, Zeszyt I, 1991, ss. 4-17.
154. Węglarz J., Błażewicz J., Kovalyov M.: *Preemptable malleable task scheduling problem*, IEEE Transactions On Computers, Vol. 55, No. 4, April 2006, pp. 486-490.
155. Widell N., Nyberg C.: *Cross entropy based module allocation for distributed systems*, Proceedings of the 16th IASTED International Conference on Parallel and Distributed Computing and Systems, Cambridge, Massachusetts, USA, November 2004, pp. 223-231.
156. Winograd S.: *How fast can computer add?*, Scientific American, October 1968, pp. 93-100.
157. Xie T., Qin X.,: *A new allocation scheme for parallel applications with deadline and security constraints on clusters*. Proceedings of the IEEE International Conference on Cluster Computing, Boston, Massachusetts, USA, September 2005, pp. 1-10.
158. Yourdon E., Constantine L.: *Structural design. Fundamentals of a discipline of computer programming and design.* Prentice-Hall, New York 1979.
159. Zacniewski A., Balicki J., Masiejczyk J.: *Reguły efektywnego projektowania semantycznych usług WWW*. W: J. Górski, C. Orłowski (red.): Inżynieria

oprogramowania w procesach integracji systemów informatycznych. Pomorskie Wydawnictwo Naukowo Techniczne, Gdańsk 2010, ss.25-33.

160. Zacniewski A., Balicki J., Balicka H., Masiejczyk J.: *Multi-criterion decision making in distributed systems by quantum evolutionary algorithms.* Proceedings of The 2nd European Conference on Computer Science, Puerto de la Cruz, Spain, November 30 – December 2, 2010, WSEAS Press, pp. 328-333.
161. Zacniewski A., Balicki J., Gloza I.: *Neural meta-heuristics for interpolation of underwater object features.* In: J. Balicki, A. Cichocki (Eds.) Advances in Computer Science – Network Centric Warfare, CTM Press, Gdynia 2009, ss. 60-73.
162. Zitzler E. and Thiele L.: *Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach*. IEEE Transactions on Evolutionary Computation, No. 3, Vol. 4, November 1999, pp. 257–271.
163. Żymierin D.G., Maksymienko W.I. (red.): *Podstawy projektowania wielkich sieci teleinformatycznych*, WNT, Warszawa 1981.
164. http://docs.moodle.org/en/Category:Modules, strona internetowa opisująca moduły aktywności systemu *MOODLE*, październik 2010.
165. http://docs.moodle.org/en/Online_Learning_History, strona internetowa opisująca historię nauczania online, październik 2010.
166. http://docs.moodle.org/en/Security_overview, strona opisująca problemy bezpieczeństwa w *MOODLE*, październik 2010.
167. http://docs.moodle.org/en/Step-by-step_Install_Guide_for_Ubuntu, strona opisująca instalację *MOODLE* w systemie Linux, październik 2010.
168. http://fle3.uiah.fi/, strona internetowa platformy Fle3 Future Learning Environment, październik 2010.
169. http://ilearning.oracle.com/ilearn/en/learner/jsp/login.jsp, strona internetowa platformy Oracle iLearning, październik 2010.
170. http://java.sun.com/javaee, strona opisująca technologię Java EE, październik 2010.
171. http://ltsc.ieee.org/wg12/, strona internetowa opisująca standard metadanych LOM, październik 2010.
172. http://manhattan.sourceforge.net/, strona internetowa platformy Manhattan Virtual Classroom, październik 2010.
173. http://moodle.amw.gdynia.pl/, strona internetowa Wydziału Nawigacji i Uzbrojenia Akademii Marynarki Wojennej wykorzystująca system MOODLE, październik 2010.

174. http://moodle.wsh-kielce.edu.pl/, portal e-learning Wyższej Szkoły Handlowej w Kielcach, październik 2010.
175. http://pl.wikipedia.org/wiki/Konstrukcjonizm, strona internetowa dotycząca pojęcia konstrukcjonizmu, październik 2010.
176. http://polska.4system.com/, strona internetowa producenta oprogramowania WBTServer, październik 2010.
177. http://support.kaspersky.com/learning/courses/, strona internetowa producenta oprogramowania antywirusowego oferującego darmowe szkolenia, październik 2010.
178. http://wazniak.mimuw.edu.pl/index.php?title=Systemy_rozproszone, UCZELNIA ONLINE, wspólne przedsięwzięcie Politechniki Poznańskiej, UW, PW i UJ, październik 2010.
179. http://www.adlnet.gov/, strona internetowa organizacji ADL, październik 2010.
180. http://www.adlnet.gov/adlnews/Documents/News%20Archive/TheSCORMConformanceTestSuiteVersion127SelfTestisAvailableforDownload.aspx, strona internetowa opisująca narzędzie do weryfikacji zgodności ze standardem komunikacyjnym, październik 2010.
181. http://www.aicc.org/, strona internetowa organizacji AICC, październik 2010.
182. http://www.aicc.org/pages/aicc_ts.htm, strona internetowa opisująca narzędzie do sprawdzania zgodności standardu komunikacyjnego z zaleceniami AICC, październik 2010.
183. http://www.angielski-skype.com/, strona internetowa firmy oferującej naukę języka angielskiego przez komunikator Skype, październik 2010.
184. http://www.army.mil/, oficjalna strona internetowa armii amerykańskiej, październik 2010.
185. http://www.army.mil/ako/, portal armii amerykańskiej, październik 2010.
186. http://www.armyelearning.ca/, portal internetowy armii kanadyjskiej.
187. http://www.astd.org/, strona internetowa organizacji ASTD, październik 2010.
188. http://www.astd.org/content/education/certificatePrograms/, strona internetowa opisująca programy certyfikacji organizacji ASTD, październik 2010.
189. http://www.atutor.ca/, strona internetowa platformy ATutor, październik 2010.
190. http://www.atvn.pl/, strona internetowa Akademickiej Telewizji Naukowej, październik 2010.

191. http://www.benchmark.pl/, strona internetowa z testami wydajności komputerów, październik 2010.
192. http://www.benntec.de/index.php/en/news/95-fernausbildungskongress-2010, strona internetowa firmy *benntec* dotycząca zdalnego nauczania w armii niemieckiej, październik 2010.
193. http://www.cisco.com/web/PL/seminaria/epracownik.html/, strona internetowa projektu E-Pracownik, październik 2010.
194. http://www.claroline.net/, strona internetowa platformy Claroline, październik 2010.
195. http://www.cyf-kr.edu.pl/enauczanie/, strona internetowa Akademickiego Centrum Komputerowego Akademii Górniczo - Hutniczej, październik 2010.
196. http://www.czn.uj.edu.pl/, strona internetowa Centrum Zdalnego Nauczania Uniwersytetu Jagiellońskiego, październik 2010.
197. https://www.dlp.mod.uk/, portal internetowy armii brytyjskiej.
198. https://www.dls.army.mil/pdf/Army_e-Learner.pdf, newsletter dotyczący e-learningu w armii amerykańskiej, Issue 18, May 2010.
199. http://www.e-learningconsulting.com/products/manifest-maker.html, strona internetowa opisująca narzędzie Manifest Maker, październik 2010.
200. http://www.e-mentor.edu.pl/, strona internetowa wydawnictwa „e-mentor", październik 2010.
201. http://www.e-mentor.edu.pl/artykul_v2.php?numer=24&id=526/, artykuł na temat nowelizacji regulacji dotyczących kształcenia na odległość, październik 2010.
202. http://www.e-sgh.pl/, strona internetowa platformy edukacyjnej Szkoły Głównej Handlowej, październik 2010.
203. http://www.e-sgh.pl/abc.php, strona internetowa opisująca podstawy e-learningu, październik 2010.
204. http://www.etechnow.blogspot.com/2010/03/lms-satisfaction-features-and-barriers.html, strona internetowa dotycząca porównania platform LMS, październik 2010.
205. http://www.google.com/Top/Computers/Software/Online_Training/Delivery_and_Management_Systems/, strona internetowa dotycząca oprogramowania e-learningowego, październik 2010.

206. http://www.google.com/Top/Reference/Education/Instructional_Technology/Course_Website_Software/, strona internetowa dotycząca oprogramowania e-learningowego, październik 2010.
207. http://www.gu.us.edu.pl/node/229681, artykuł o systemie MOODLE w „Gazecie Uniwersyteckej", miesięcznik Uniwersytetu Śląskiego w Katowicach, nr 6/2006.
208. http://www.hotpot.uvic.ca, strona internetowa opisująca quizy *HotPotato*, październik 2010.
209. http://www.ibm.com/developerworks/lotus/library/ls-elearning_evolution/, strona internetowa opisująca ewolucję systemów Lotus, październik 2010.
210. http://www.ieee.org/, strona internetowa organizacji IEEE, październik 2010.
211. http://www.ieeeltsc.org/, strona internetowa organizacji IEEE LTSC, październik 2010.
212. http://www.ilias.de/, strona internetowa platformy Ilias, październik 2010.
213. http://www.imsglobal.org/, strona internetowa organizacji IMS, październik 2010.
214. http://www.imsglobal.org/metadata/, strona internetowa opisująca standard metadanych IMS, październik 2010.
215. http://www.iniejawna.pl/przepisy/mon/instrukcja_zpb.html, strona internetowa zawierająca instrukcję o zasadach pracy biurowej w resorcie Obrony Narodowej, październik 2010.
216. http://www.iso.org/, strona internetowa organizacji ISO, październik 2010.
217. http://www.ITforAll.pl/, strona internetowa autora rozprawy zawierająca pliki wykorzystywane w rozprawie, październik 2010.
218. http://www.itweb.co.za/index.php?option=com_content&task=view&id=28850&Itemid=70, strona internetowa dotycząca e-learningu w armii francuskiej, październik 2010.
219. http://www.jcasolutions.com/index.php/ssp, strona internetowa opisująca narzędzie do tworzenia metadanych, październik 2010.
220. http://www.mathworks.com/help/, strona internetowa firmy MathWorks zawierająca dokumentację Matlaba, październik 2010.
221. http://www.mathworks.com/support/solutions/en/data/1-6M21RQ/?solution=1-6M21RQ, strona internetowa z rozwiązaniami technicznymi firmy MathWorks dotyczącymi pakietu MCR, październik 2010.
222. http://www.menedzer.pl/, strona internetowa Zespołu Szkół Prywatnych w Bydgoszczy, październik 2010.

223. http://www.microsoft.com/education/slk.mspx, strona internetowa opisująca narzędzie *SharePoint Learning Kit* firmy Microsoft, październik 2010.
224. http://www.middleware.org, strona internetowa opisująca rodzaje oprogramowania warstwy pośredniej, październik 2010
225. http://www.moodle.org/, strona internetowa platformy Moodle, październik 2010.
226. http://www.moodle.org/sites/index.php?country=PL, strona internetowa opisująca wdrożenia systemu *MOODLE* w Polsce, październik 2010.
227. https://www.natoschool.nato.int/index.asp, strona NATO School w Oberammergau, październik 2010.
228. http://www.nbportal.pl/pl/cw/, strona internetowa darmowego portalu wiedzy o ekonomii i zarządzaniu, październik 2010.
229. http://www.netstudier.borytucholskie.pl/netstudier/, strona internetowa o systemie Netstudier, październik 2010.
230. http://www.okno.pw.edu.pl/, strona internetowa Ośrodka Kształcenia Na Odległość OKNO Politechniki Warszawskiej, październik 2010.
231. http://www.pamctr.uni.lodz.pl/, strona Polsko-Amerykańskiego Centrum Zarządzania w Łodzi, październik 2010.
232. http://www.pelp.net/, strona internetowa platformy Pelp, październik 2010.
233. http://www.php-debugger.com/dbg/, strona internetowa opisująca oprogramowanie DBG | PHP Debugger and Profiler, październik 2010.
234. http://www.polycom.eu/products/telepresence_video/video_conference_systems, strona internetowa opisująca zestawy wideo-konferencyjne, październik 2010.
235. http://www.pou.pl/, strona internetowa Wyższej Szkoły Zarządzania (Polish Open University), październik 2010.
236. http://www.pret.com.pl/, strona internetowa wydawnictwa PRET S.A., październik 2010.
237. http://www.puw.pl/, strona internetowa Polskiego Uniwersytetu Wirtualnego, październik 2010.
238. http://www.sap.com/industries/defense-security/customers/index.epx, strona internetowa firmy SAP dotycząca wdrożeń ich produktów w armiach NATO, październik 2010.
239. http://www.sceno.edu.pl/, strona internetowa Świętokrzyskiego Centrum Edukacji Na Odległość, październik 2010.

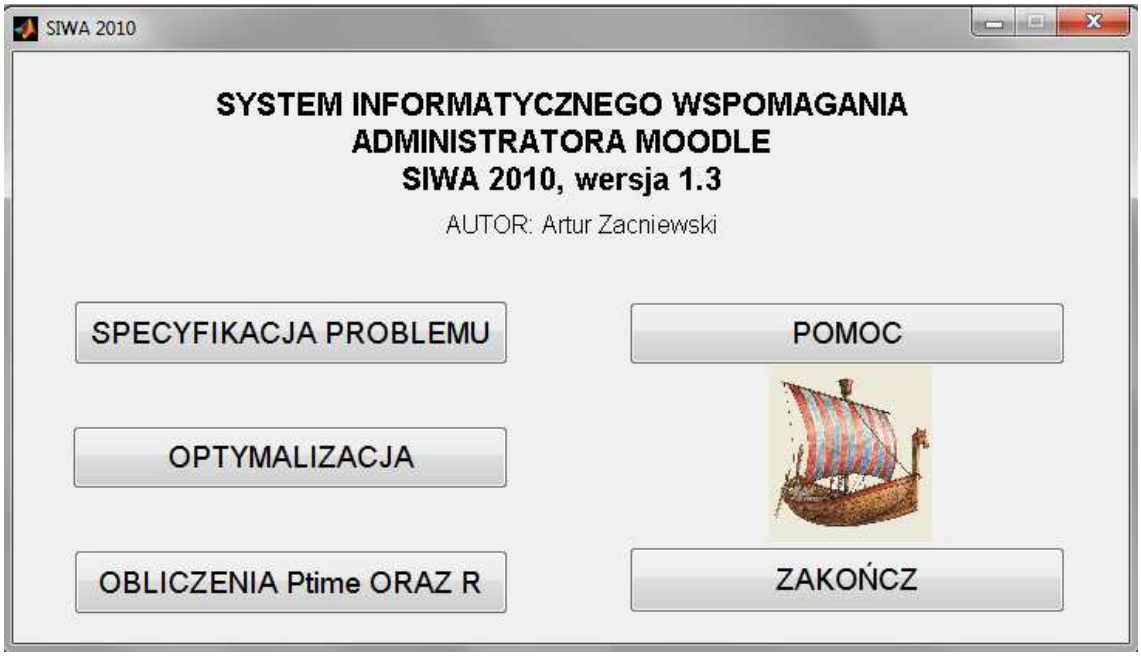
240. http://www.scormsoft.com/scorm/cam/metadata, strona internetowa opisująca standard meta danych SCORM, październik 2010.
241. http://www.section508.gov/, strona internetowa opisująca standard jakości, październik 2010.
242. http://www.studianet.pl/, strona internetowa zdalnego nauczania w Politechnice Koszalińskiej, październik 2010.
243. http://www.szkola-online.pl/, strona internetowa Liceum Zaocznego dla Dorosłych w Warszawie, październik 2010.
244. http://www.ubuntugeek.com/nfs-server-and-client-configuration-in-ubuntu.html, strona opisująca instalację serwera i klienta NFS w systemie Linux, październik 2010.
245. http://www.w3.org/, strona internetowa organizacji W3C, październik 2010.
246. http://www.wiedzanet.pl/, strona internetowa spółki zajmującej się produkcją i dystrybucją szkoleń elektronicznych, październik 2010.
247. http://www.ydp.com.pl/, strona internetowa firmy Young Digital Planet, październik 2010.

WYKAZ RYSUNKÓW

WYKAZ TABEL

DODATEK A. OPROGRAMOWANIE DO WSPOMAGANIA PRACY ADMINISTRATORA SYSTEMU *MOODLE*

Do usprawnienia procesu zarządzania zasobami systemu szkolenia wojskowego na platformie *MOODLE* zaprojektowano aplikację *System Informatycznego Wspomagania Administratora MOODLE* o nazwie SIWA 2010. Po uruchomieniu z linii poleceń skryptu inicjującego system, prezentowane są opcje (rys. A.1).



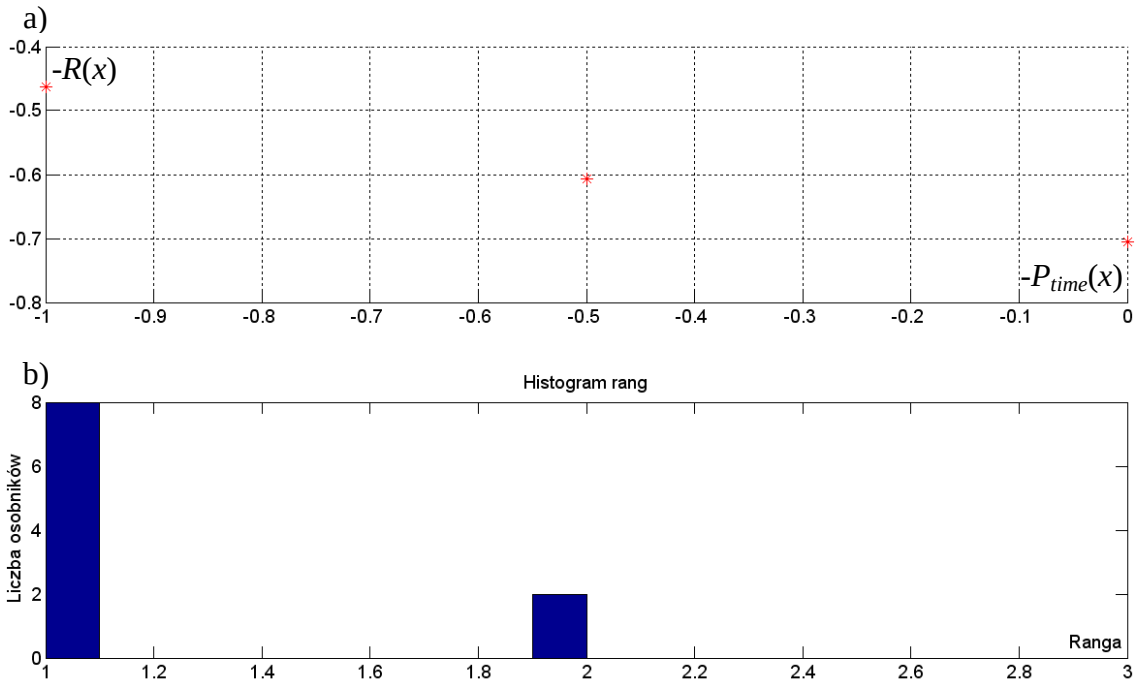
Rys. A.1. Interfejs graficzny systemu SIWA 2010
Źródło: opracowanie własne.

Wybór opcji *Specyfikacja problemu* pozwala na zapoznanie się administratora ze sformułowaniem problemu optymalizacji przydziału modułów programistycznych oraz metodą jego rozwiązania dla platformy *MOODLE*. Wykorzystano wyniki uzyskane podczas przeprowadzonych eksperymentów numerycznych. Plik z opisem otwierany jest za pomocą domyślnego czytnika plików PDF.

Wybór opcji *Optymalizacja* umożliwia wyznaczanie optymalnych przydziałów modułów do komputerów dla ustalonych wartości parametrów i danych wejściowych do problemu. Zastosowano algorytm ewolucyjny NSGA-III. Na rysunku A.2 przedstawiono rozwiązania optymalne w sensie Pareto dla przykładowej instancji V=8, I=3, J=2. Rangę równą 1 otrzymało 8 osobników, a rangę 2 nadano 2 osobnikom. W głównym oknie Matlab-a wyświetlana jest informacja o liczbie generacji, liczbie

elementów w zbiorze ocen Pareto-optymalnych oraz o liczbie oszacowań funkcji sprawności. Pliki związane z instancją umieszczono w `/MATLAB/dodatek A`.

Opcja *Wyznaczanie Ptime oraz R* pozwala na obliczenie wartości wskazanych kryteriów dla instancji w odniesieniu do V=14, I=2, J=2. Administrator systemu *MOODLE* może dokonać zmian parametrów, tak aby uzyskać optymalne przydziały dla alternatywnego zestawu danych.



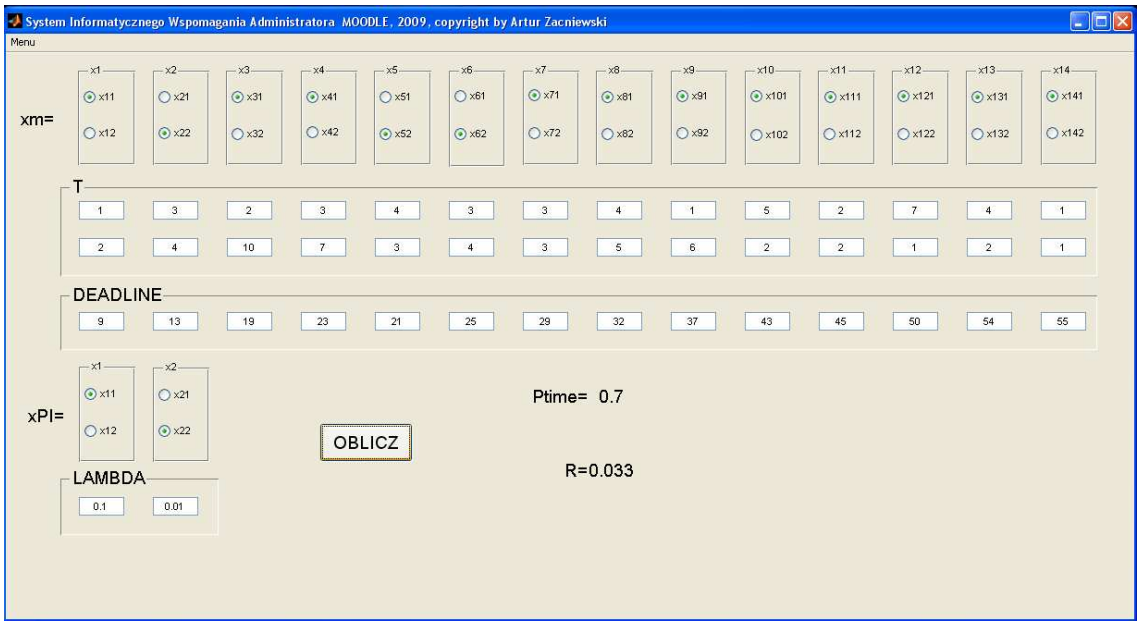
Rys. A.2. Optymalne rozwiązania uzyskane dla instancji V=8, I=3, J=2:
a) oceny w zbiorze wyników
b) histogram rang rozwiązań w wyznaczonej populacji
Źródło: opracowanie własne.

Dane do obliczeń dla wariantu z czternastoma modułami zapisane są w pliku *DaneV14I2J2K4_bin.m*. Do modyfikacji przydziałów modułów do węzłów i przydziałów komputerów do węzłów zastosowano przyciski typu *radio button*, co zapewnia spełnienie ograniczeń (2.4) i (2.6). Istnieje możliwość edycji elementów macierzy $\boldsymbol{T}$ oraz wektorów $\boldsymbol{d}$, $\boldsymbol{\Lambda}$, a także zmiany przydziałów modułów do komputerów (rys. A.3).

W wypadku konieczności uruchomienia aplikacji do obliczania P_{time} oraz R na komputerze bez zainstalowanego oprogramowania Matlab, niezbędna jest instalacja pakietu MCR (ang. *MATLAB Component Runtime*). Wówczas aplikacja, której interfejs graficzny przedstawiono na rysunku A.3 uruchamia się z opóźnieniem. Przyczyną jest

to, że MCR nie jest szybkim środowiskiem uruchomieniowym w przypadku samodzielnych aplikacji (ang. *standalone application*) [221].

Opcja *Pomoc* (rys. A1) pozwala na uzyskanie obszernych informacji związanych z oprogramowaniem.



Rys. A.3. Graficzny interfejs do wyznaczania wartości kryteriów cząstkowych przy wspomaganiu administratora systemu *MOODLE*
Źródło: opracowanie własne.

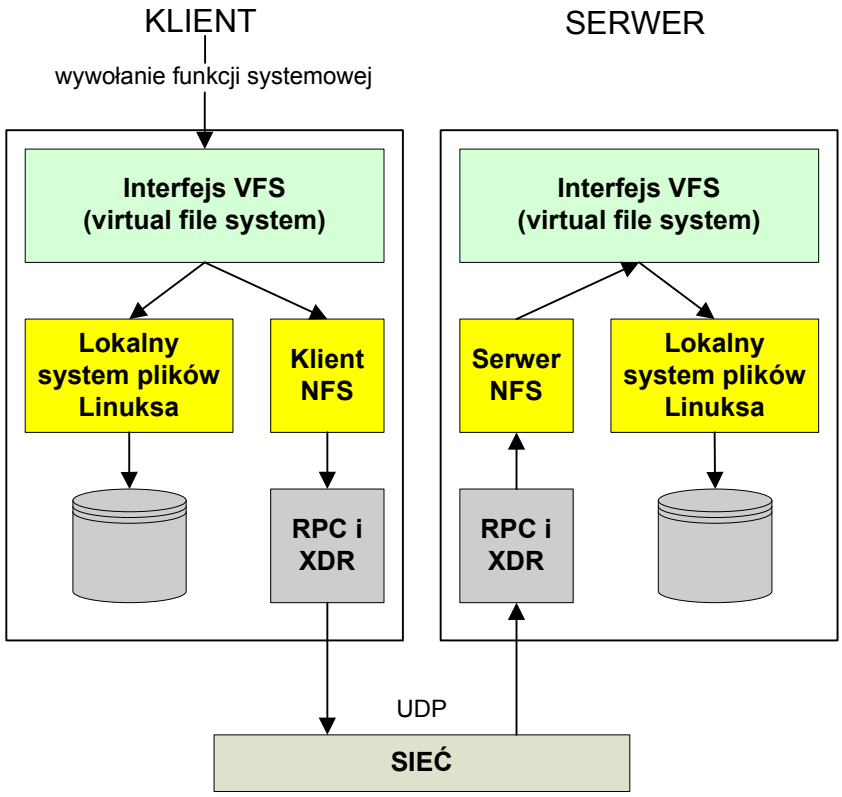
Oprogramowanie wykonano przy pomocy środowiska MATLAB w wersji 2008a zainstalowanego w systemie Windows Vista na komputerze z architekturą x86. Na dołączonej płycie CD w katalogu `\MATLAB` znajdują się pliki użyte do generacji odpowiednich rysunków umieszczonych w rozprawie.

Ze względu na rozwój środowiska Matlab, najnowsze wersje plików znajdują się na stronie WWW związanej z tematyką rozprawy [217]. Pliki z dołączonej płyty należy skopiować do wybranego folderu na dysku twardym, po czym w zakładce *Current Directory* w Matlabie należy wskazać wybrany folder z plikami, a następnie uruchomić skrypt '*siwa*'.

DODATEK B. NARZĘDZIE DO PRZYDZIELANIA MODUŁÓW SYSTEMU MOODLE DO DWÓCH KOMPUTERÓW

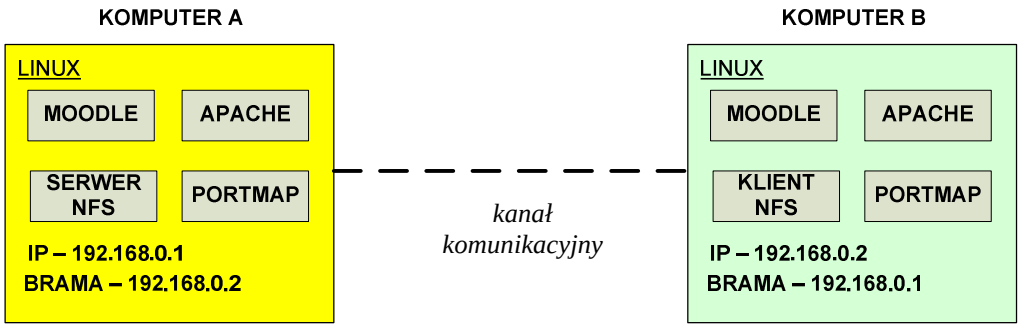
Przeprowadzono eksperyment polegający na rozproszeniu modułów systemu *MOODLE* między dwa komputery. Każdy z komputerów posiadał zainstalowaną linuksową dystrybucję *Ubuntu Karmic Koala* 9.10 oraz system *MOODLE* w wersji 1.9.7+. Wymagana była instalacja serwera Apache oraz obsługa plików PHP [167]. Zainstalowano również klienta i serwera NFS oraz usługi *portmap*, która jest niezbędna do obsługi wywołań zdalnych procedur RPC (ang. *Remote Procedure Call*) [244].

NFS odwzorowuje abstrakcyjny model systemu plików w system lokalny, zależny od systemu operacyjnego zainstalowanego na komputerze. NFS jest usługą bezstanową, która oprócz RPC wykorzystuje standard XDR (ang. *eXternal Data Representation*) oraz protokół UDP, co zilustrowano na rysunku B.1.



Rys. B.1. Mechanizmy NFS w systemie Linux
Źródło: opracowanie własne.

Na rysunku B.2 przedstawiono schemat systemu przeznaczonego do rozproszenia modułów platformy *MOODLE*. Adres IP pierwszego komputera jest adresem bramy drugiego i na odwrót. Jest to sposób połączenia dwóch komputerów bez dodatkowych urządzeń.

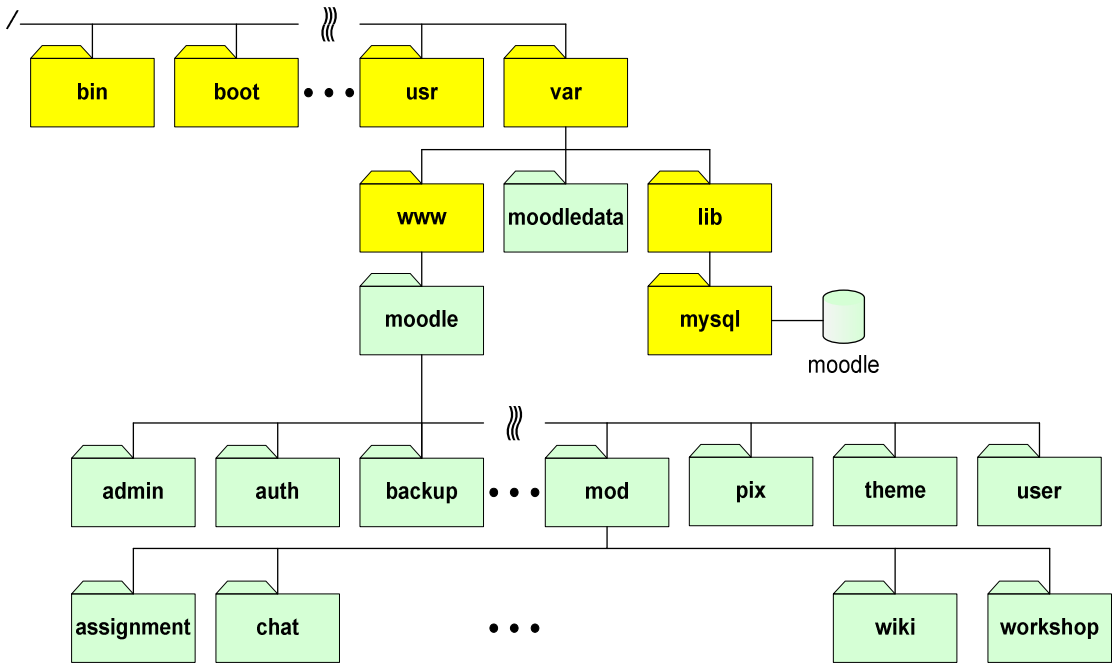


Rys. B.2. Połączenie dwóch komputerów w celu rozproszenia systemu *MOODLE*
Źródło: opracowanie własne.

Wersja systemu *MOODLE* oraz struktura jego katalogów powinny być takie same na obu komputerach. Najwygodniejszą opcją jest opracowanie kursu na jednym z komputerów, utworzeniu kopii zapasowej kursu i wygenerowanie go na drugim komputerze za pomocą opcji odtwarzania kursu. Na rysunku B.3 zaznaczono kolorem zielonym katalogi systemu *MOODLE* powstałe po instalacji.

Po uruchomieniu systemu za pomocą komendy *ping* można sprawdzić stan połączenia. W pliku `/etc/exports` serwera NFS definiuje się udostępniane katalogi. W plikach `/etc/hosts.allow` i `/etc/hosts.deny` specyfikuje się komputery mogące mieć dostęp do zasobów w przypadku serwera NFS oraz lokalizację udostępnianych zasoby w przypadku klienta NFS. Aby połączyć się z zasobem NFS, należy go zamontować na komputerze klienta. Wskazanie jest montowanie zasobu przy starcie systemu przez odpowiednie wpisy w pliku `/etc/fstab` klienta. Aby zapewnić zamontowanie zasobów w przypadku braku działania systemu DNS, w pliku `/etc/hosts` klienta dodaje się adres i nazwę serwera NFS. Zamiast NFS istnieje możliwość wykorzystania innego systemu plików, np. globalnego systemu plików GFS (ang. *Global File System*).

Czasy wykonywania się modułów oszacowano za pomocą narzędzia *NuSphere PhpED*. Jest to rozbudowane środowisko IDE wspomagające projektowanie i zarządzanie kodem PHP. Aplikacja zintegrowano z klientem baz danych MySQL. Oferuje również *profiler* kodu i wsparcie dla protokołów WebDAV/HTTPS.



Rys. B.3. Struktura katalogów oprogramowania *MOODLE* w systemie Linux
Źródło: opracowanie własne.

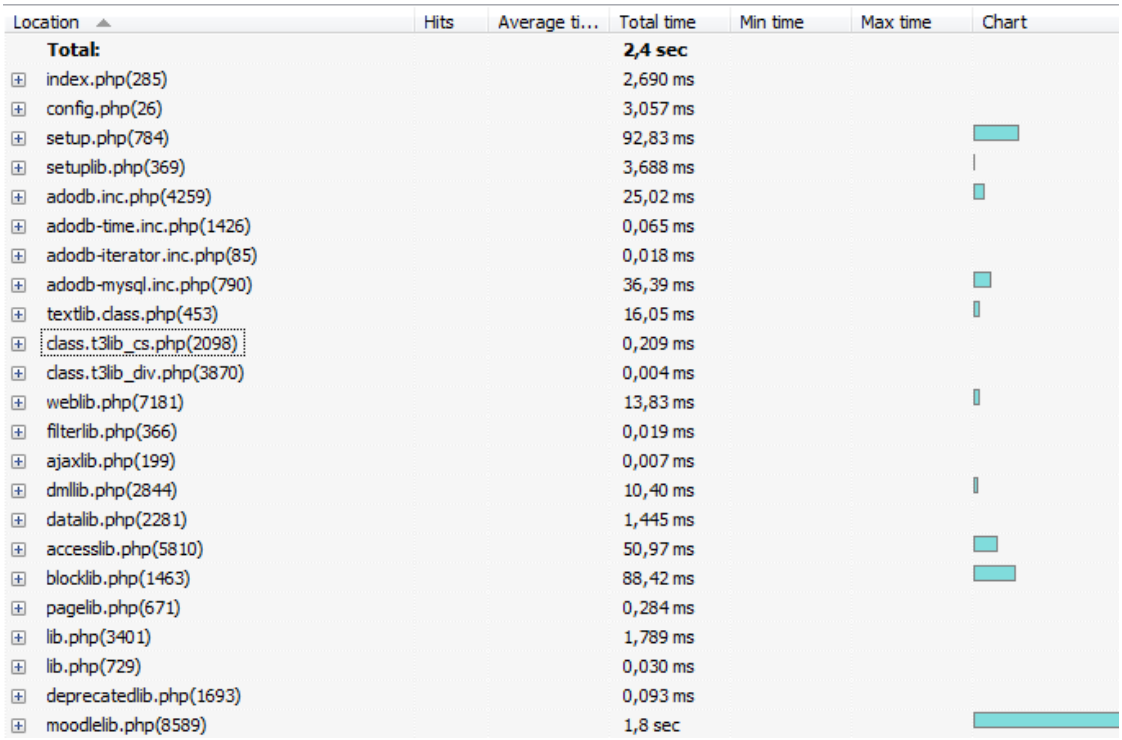
W systemie zainstalowano pakiet oprogramowania XAMPP zawierający serwer Apache i serwer MySQL. Aby wykorzystać narzędzie *NuSphere*, należy kurs *MOODLE* otworzyć w systemie Windows.

Wykorzystując wbudowany *profiler*, można oszacować parametry czasowe modułów programistycznych systemu *MOODLE*. Na rysunku B.4 przedstawiono interfejs *profilera*, w którym obok nazwy pliku dostępny jest łączny czas wykonań pliku w systemie. Ponadto dostępny jest wykres, na którym dane w kolumnie '*chart*', w postaci niebieskiego paska, odnoszą się do najdłużej wykonującego się modułu.

Wybierając znak '+' z lewej strony nazwy pliku, otrzymuje się informację o parametrach czasowych poszczególnych wywołań danego pliku. Na rysunku B.5 przedstawiono parametry czasowe dla pliku *setuplib.php*. Liczba w nawiasie oznacza numer linii w pliku, a liczba w drugiej kolumnie opisuje, ile było odwołań do zadanego wiersza w czasie pracy systemu.

Można również uzyskać informację o czasie pracy całego systemu *MOODLE* oraz minimalnych i maksymalnych wartościach czasów wykonywania się wybranego pliku, gdy jest on wykonywany więcej niż raz.

Korzystając z opcji '*run profiler with debugger*', otrzymuje się rezultaty profilowania w czasie trwania sesji *debuggera*.

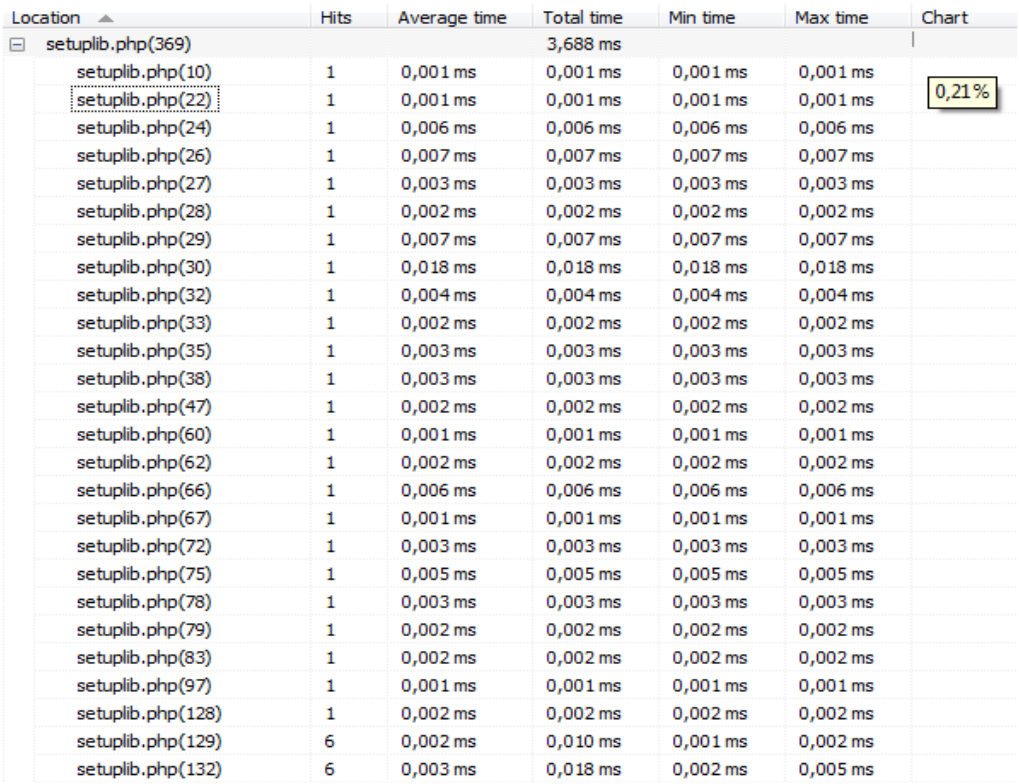


Location	Hits	Average ti...	Total time	Min time	Max time	Chart
Total:			**2,4 sec**			
index.php(285)			2,690 ms			
config.php(26)			3,057 ms			
setup.php(784)			92,83 ms			
setuplib.php(369)			3,688 ms			
adodb.inc.php(4259)			25,02 ms			
adodb-time.inc.php(1426)			0,065 ms			
adodb-iterator.inc.php(85)			0,018 ms			
adodb-mysql.inc.php(790)			36,39 ms			
textlib.class.php(453)			16,05 ms			
class.t3lib_cs.php(2098)			0,209 ms			
class.t3lib_div.php(3870)			0,004 ms			
weblib.php(7181)			13,83 ms			
filterlib.php(366)			0,019 ms			
ajaxlib.php(199)			0,007 ms			
dmllib.php(2844)			10,40 ms			
datalib.php(2281)			1,445 ms			
accesslib.php(5810)			50,97 ms			
blocklib.php(1463)			88,42 ms			
pagelib.php(671)			0,284 ms			
lib.php(3401)			1,789 ms			
lib.php(729)			0,030 ms			
deprecatedlib.php(1693)			0,093 ms			
moodlelib.php(8589)			1,8 sec			

Rys. B.4. Łączne czasy wykonywania się modułów programistycznych *MOODLE* na testowanym komputerze
Źródło: opracowanie własne.

Dane w systemie mogą zostać pogrupowane według kolejnych procedur wykonywanych w systemie lub według kolejnych uruchamianych plików PHP wchodzących w skład systemu. Przyjęto, że moduły zawierają katalogi, w których znajdują się wspomniane pliki.

Dla kursu omówionego w podrozdziale 4.6 wykonanych zostało 199 plików PHP. Łącznie miało miejsce 17 547 odwołań do tychże plików. Pliki zlokalizowano w 76 katalogach. Należy zauważyć, że pliki podlegające wykonaniu oraz liczba ich wykonań zmienia się w zależności od składowych kursu *MOODLE*. Przed rozpoczęciem alokacji modułów należy zaprojektować, z jakich katalogów będą się one składać. Przykładowo pliki, do których ścieżka zaczyna się od `var/www/moodle/blocks/` lub `var/moodledata/` mogłyby stanowić jeden moduł.



Location	Hits	Average time	Total time	Min time	Max time	Chart
setuplib.php(369)			3,688 ms			
setuplib.php(10)	1	0,001 ms	0,001 ms	0,001 ms	0,001 ms	
setuplib.php(22)	1	0,001 ms	0,001 ms	0,001 ms	0,001 ms	0,21%
setuplib.php(24)	1	0,006 ms	0,006 ms	0,006 ms	0,006 ms	
setuplib.php(26)	1	0,007 ms	0,007 ms	0,007 ms	0,007 ms	
setuplib.php(27)	1	0,003 ms	0,003 ms	0,003 ms	0,003 ms	
setuplib.php(28)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(29)	1	0,007 ms	0,007 ms	0,007 ms	0,007 ms	
setuplib.php(30)	1	0,018 ms	0,018 ms	0,018 ms	0,018 ms	
setuplib.php(32)	1	0,004 ms	0,004 ms	0,004 ms	0,004 ms	
setuplib.php(33)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(35)	1	0,003 ms	0,003 ms	0,003 ms	0,003 ms	
setuplib.php(38)	1	0,003 ms	0,003 ms	0,003 ms	0,003 ms	
setuplib.php(47)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(60)	1	0,001 ms	0,001 ms	0,001 ms	0,001 ms	
setuplib.php(62)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(66)	1	0,006 ms	0,006 ms	0,006 ms	0,006 ms	
setuplib.php(67)	1	0,001 ms	0,001 ms	0,001 ms	0,001 ms	
setuplib.php(72)	1	0,003 ms	0,003 ms	0,003 ms	0,003 ms	
setuplib.php(75)	1	0,005 ms	0,005 ms	0,005 ms	0,005 ms	
setuplib.php(78)	1	0,003 ms	0,003 ms	0,003 ms	0,003 ms	
setuplib.php(79)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(83)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(97)	1	0,001 ms	0,001 ms	0,001 ms	0,001 ms	
setuplib.php(128)	1	0,002 ms	0,002 ms	0,002 ms	0,002 ms	
setuplib.php(129)	6	0,002 ms	0,010 ms	0,001 ms	0,002 ms	
setuplib.php(132)	6	0,003 ms	0,018 ms	0,002 ms	0,005 ms	

Rys. B.5. Szczegółowe parametry czasowe dla wybranego pliku systemu *MOODLE*
Źródło: opracowanie własne.

W ten sposób wyodrębniono 14 modułów, które mogą zostać optymalnie alokowane. W tabeli 6 przedstawiono liczby plików wykorzystywane przez moduły.

Tabela 6. Podział plików na moduły *MOODLE* i ich alokacja

Numer modułu	Moduł	Liczba plików	Numer komputera
1.	`var/www/moodle/`	4	1
2.	`var/www/moodle/blocks/`	32	1
3.	`var/www/moodle/mod/`	17	1
4.	`var/www/moodle/lib/`	28	1
5.	`var/moodledata/`	54	2
6.	`var/www/moodle/theme/`	4	1
7.	`var/www/moodle/lang/en_utf8`	28	2
8.	`var/www/moodle/admin/`	22	1
9.	`var/www/moodle/blog/`	3	2
10.	`var/www/moodle/tag/`	2	1
11.	`var/www/moodle/enrol/`	2	1
12.	`var/www/moodle/course/`	1	2
13.	`var/www/moodle/group/`	1	1
14.	`var/www/moodle/my/`	1	1
RAZEM:		199	

Źródło: opracowanie własne.

Ścieżka dostępu określa katalog, w którym znajdują się pliki PHP lub inne katalogi. Założono, że w wyniku procesu optymalizacji przydziału modułów do dwóch komputerów moduły nr 1, 2, 3, 4, 6, 8, 10, 11, 13, 14 przydzielono do komputera nr 1, a moduły nr 5, 7, 9, 12 przydzielono do komputera nr 2. Należy dodać odpowiednie wpisy w plikach konfiguracyjnych systemu Linuks. Kody źródłowe plików `/etc/hosts.deny`, `/etc/hosts.deny/`, `etc/exports/` oraz `etc/fstab` przedstawiono w tabelach 7, 8, 9 i 10.

Tabela 7. Kod źródłowy pliku `/etc/hosts.deny`

SERWER
`portmap mountd nfsd statd lockd rquotad : ALL`
KLIENT
`portmap : ALL`
UWAGI: wstępna blokada wszystkich klientów

Źródło: opracowanie własne.

Tabela 8. Kod źródłowy pliku `/etc/hosts.allow`

SERWER
`portmap mountd nfsd statd lockd rquotad : 192.168.0.1, 192.168.0.2`
KLIENT
`portmap: 192.168.0.1`
UWAGI: dozwolone adresy IP serwera i klientów, nie jest wskazane używanie nazw hostów

Źródło: opracowanie własne.

Tabela 9. Kod źródłowy pliku `/etc/exports`

SERWER

`/var/moodledata 192.168.0.2(rw,sync,no_subtree_check)`
`/var/www/moodle/lang/en_utf8`
`192.168.0.2(rw,sync,no_subtree_check)`
`/var/www/moodle/blog/`
`192.168.0.2(rw,sync,no_subtree_check)`
`/var/www/moodle/course/`
`192.168.0.2(rw,sync,no_subtree_check)`

UWAGI: Po lokalizacji zasobu podawane są adresy IP klientów. Po każdej zmianie zasobów należy wydać komendę: `sudo exportfs –ra` Restart serwera NFS: `sudo /etc/init.d/nfs-kernel-server restart`

Źródło: opracowanie własne.

Tabela 10. Kod źródłowy pliku `/etc/fstab`

KLIENT

`sudo mount 192.168.0.1:/var/moodledata  /var/`
`sudo mount 192.168.0.1:/var/www/moodle/lang/en_utf8 /var/www/moodle/lang/`
`sudo mount 192.168.0.1: /var/www/moodle/blog/ /var/www/moodle/`
`sudo mount 192.168.0.1: /var/www/moodle/course/ /var/www/moodle/`

UWAGI: po IP serwera określany jest folder przeznaczony do udostępnienia, a następnie folder lokalny, w którym będzie zamontowany zasób

Źródło: opracowanie własne.

DODATEK C. PŁYTA CD Z OPROGRAMOWANIEM

Na załączonej do rozprawy płycie CD umieszczono oprogramowanie wytworzone na potrzeby pracy. Folder `MATLAB` jest głównym folderem znajdującym się na płycie. W tym folderze znajdują się cztery foldery: `do rozdziału 2`, `do rozdziału 3`, `do rozdziału 4` oraz `dodatek A`.

W każdym z tych czterech folderów znajduje się plik tekstowy `!readme.txt`, w którym omówiona jest rola poszczególnych plików, znajdujących się w danym folderze. Wskazany jest również sposób uruchamiania odpowiednich skryptów.

Najnowsze wersje umieszczonych plików znajdują się na stronie autora [217]. Aby uzyskać do nich dostęp, należy po kliknięciu zakładki '*Kursy i projekty*' zalogować się używając loginu: *phd* i hasła: *Zacniewski_2010*.