Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **informatyka stosowana (IST)**
Specjalność: **projektowanie systemów informatycznych (PSI)**

PRACA DYPLOMOWA MAGISTERSKA

Koewolucyjny algorytm ewolucyjny w rozwiązywaniu wybranego silnie ograniczonego wielokryterialnego problemu harmonogramowania

Aleksandra Stecka

Opiekun pracy
dr hab. inż. Paweł Myszkowski

Słowa kluczowe: metaheurystyki, koewolucyjny algorytm ewolucyjny, optymalizacja wielokryterialna, silnie ograniczony problem harmonogramowania MS-RCPSP

WROCŁAW 2024

Streszczenie

Praca dotyczy zastosowania inspiracji zjawiskiem koewolucji w rozwiązywaniu wybranego silnie ograniczonego wielokryterialnego problemu harmonogramowania za pomocą algorytmu ewolucyjnego. Jako konkretny problem harmonogramowania wybrano problem MS-RCPSP (ang. *Multi-Skill Resource-Constrained Project Scheduling Problem*), który dotyczy znajdowania harmonogramu dla projektu, w którym aby wykonać konkretne zadanie, zasób musi posiadać wymaganą przez to zadanie umiejętność co najmniej na wymaganym przez to zadanie poziomie. Przedstawiono formalną definicję wybranego problemu harmonogramowania, a w oparciu o przegląd literatury opracowano autorską metodę o nazwie CCoDE (ang. *Cooperative Coevolutionary Differential Evolution*) oraz dokonano selekcji metod wykorzystanych jako punkt odniesienia (w tym metoda NTGA2 oraz metoda B-NTGA). Autorską metodę porównano również z analogiczną metodą opartą na ewolucji różnicowej, która nie jest koewolucyjna - MODE (ang. *Multi-Objective Differential Evolution*). Badania nad skutecznością autorskiej metody w porównaniu do metod wybranych jako punkt odniesienia wykonano zgodnie z przedstawionymi założeniami i organizacją badań, a następnie dokonano analizy wyników badań. Niestety, metodzie CCoDE nie udało się osiągnąć większej skuteczności od metody MODE. Między zaproponowanymi wariantami metody CCoDE nie zaobserwowano znaczącej różnicy w skuteczności. Udało się uzyskać większą skuteczność metody CCoDE w stosunku do metody losowej. Metoda MODE osiągnęła nieznacznie większą skuteczność od metody B-NTGA co wskazuje na to, że ewolucja różnicowa może być skuteczna w rozwiązywaniu problemu MS-RCPSP.

Abstract

The paper concerns the application of ideas drawn from the phenomenon of coevolution in solving a selected strongly constrained multi-criteria scheduling problem using an evolutionary algorithm. The specific scheduling problem chosen is the MS-RCPSP (Multi-Skill Resource-Constrained Project Scheduling Problem), which involves finding a schedule for a project where, in order to perform a specific task, a resource must have the skill required by that task at least at the level required by the task. A formal definition of the selected scheduling problem was presented, and based on a literature review, an original method called CCoDE (Cooperative Coevolutionary Differential Evolution) was developed and methods used as a benchmark are selected (including the NTGA2 method and the B-NTGA method). The original method was also compared with a similar method based on differential evolution, which is not coevolutionary - the MODE method (Multi-Objective Differential Evolution). Research on the effectiveness of the CCoDE method compared to the benchmark methods was conducted according to the presented research assumptions and organization, followed by an analysis of the research results. Unfortunately, CCoDE did not achieve greater effectiveness than MODE. No significant difference in effectiveness was observed between the proposed versions of CCoDE. Greater effectiveness of CCoDE was achieved compared to the random method. MODE achieved slightly greater effectiveness than B-NTGA, indicating that differential evolution may be effective in solving the MS-RCPSP problem.

Spis treści

Wstęp

W potocznym rozumieniu harmonogramowanie (ang. *scheduling*) to „planowanie prac poprzez przydzielanie zasobów pracy i narzędzi do zadań oraz ustalenie rozkładu tych zadań w czasie" [1]. Harmonogramowanie, czyli tworzenie harmonogramu, jest więc ustaleniem, które z dostępnych zasobów i narzędzi powinno być przydzielone do którego z zadań oraz które z zadań powinno być wykonywane w którym momencie czasu. Razem z czynnościami takimi jak planowanie, realizacja, kontrola i rozliczenie zadań składających się na realizację, harmonogramowanie jest podstawowym elementem zarządzania projektem (ang. *project management*). Zarządzanie projektem można potocznie zdefiniować jako zbiór czynności umożliwiających osiągnięcie w skończonym czasie pewnych wyznaczonych celów. Jeżeli proces zarządzania projektem nie jest wykonywany wcale lub jest wykonywany błędnie, szansa na terminowe zakończenie projektu sukcesem jest znikoma. Z tego powodu zarządzanie projektem i wszystkie jego elementy, w tym harmonogramowanie, mają bardzo duże znaczenie dla biznesu.

Wiele problemów optymalizacyjnych ze świata rzeczywistego, które opłacałoby się rozwiązywać w sposób obliczeniowy, ma bardzo duży stopień skomplikowania, posiada ograniczenia, których złamanie jest niedopuszczalne, a także może być oceniane biorąc pod uwagę jednocześnie kilka różnych kryteriów. Dla takich problemów, w tym problemu harmonogramowania, zadaniem bardzo trudnym i często niemożliwym jest opracowanie algorytmu, który gwarantowałby znalezienie rozwiązania optymalnego. Oprócz tego, jeżeli rozwiązania problemu mogą być oceniane biorąc pod uwagę jednocześnie kilka kryteriów, sama definicja „rozwiązania optymalnego" nie jest trywialna. Dla takich problemów zastosowanie mają podejścia heurystyczne, które są metodami przeszukiwania przestrzeni rozwiązań, a nie algorytmami generowania rozwiązania. Podejścia heurystyczne (gr. *heuresis* - „odnaleźć, odkryć") nie gwarantują znalezienia rozwiązania optymalnego - operują na niepełnym zbiorze informacji i pozwalają jedynie na znalezienie akceptowalnego rozwiązania stanowiącego optimum lokalne w danym otoczeniu i znalezienie go w akceptowalnym czasie. Metaheurystyki są "heurystykami wyższego poziomu" - nie rozwiązują bezpośrednio żadnego problemu, a jedynie podają sposób budowy podejścia korzystającego z odpowiednich heurystyk. Jedną z najbardziej popularnych metaheurystyk jest algorytm ewolucyjny, który czerpie inspirację z darwinowskiej teorii ewolucji - rozwiązania problemu nazywane są osobnikami i są przetwarzane w populacjach stanowiących kolejne pokolenia. Populacje potomne tworzone są w wyniku krzyżowania i mutacji osobników. Algorytm ewolucyjny posiada wiele modyfikacji, na przykład algorytm genetyczny, strategie ewolucyjne, programowanie genetyczne, ewolucja różnicowa. Jedną z modyfikacji jest zastosowanie w procesie ewolucyjnym koewolucji, czyli wspólnej ewolucji kilku gatunków. Poprzez interakcje międzygatunkowe koewoluujące gatunki wywierają na siebie wzajemny wpływ, co kształtuje ich proces ewolucyjny. Algorytm ewolucyjny, w szczególności koewolucyjny algorytm ewolucyjny, może zostać zastosowany do rozwiązywania problemu harmonogramowania przy jednoczesnej optymalizacji wielu kryteriów.

Przedmiot pracy

Przedmiotem pracy jest zastosowanie koewolucyjnego algorytmu ewolucyjnego do rozwiązywania wybranego silnie ograniczonego i wielokryterialnego problemu harmonogramowania. Koewolucyjny algorytm ewolucyjny jest modyfikacją algorytmu ewolucyjnego, w której ewoluuje jednocześnie kilka populacji, wchodzących we wzajemne interakcje między sobą. Interakcje między populacjami mają najczęściej charakter kooperacyjny lub konkurencyjny. W koewolucji kooperacyjnej gatunki współpracują ze sobą, a korzyść dla jednego gatunku jest korzyścią dla obu, podczas gdy w koewolucji konkurencyjnej gatunki rywalizują ze sobą, czyli korzyść dla jednego gatunku jest niekorzyścią dla drugiego. Zastosowanie inspiracji koewolucją i metodami opartymi na algorytmie ewolucyjnym osiągającymi wysoką skuteczność przedstawionymi w literaturze może pozwolić na opracowanie metody osiągającej dobre wyniki w rozwiązywaniu wybranego silnie ograniczonego i wielokryterialnego problemu harmonogramowania.

Wybranym silnie ograniczonym i wielokryterialnym problemem harmonogramowania jest problem MS-RCPSP (ang. *Multi-Skill Resource-Constrained Project Scheduling Problem*) [47,68,75], którego podstawowym elementem jest oczywiście utworzenie harmonogramu, w którym każde zadanie posiada przypisany wykonujący je zasób oraz każde zadanie ułożone jest na linii czasu. Rozwiązaniem problemu MS-RCPSP jest poprawny harmonogram, czyli harmonogram spełniający wszystkie ograniczenia - problem MS-RCPSP jest silnie ograniczony, co oznacza, że wszystkie ograniczenia bezwzględnie muszą być spełnione aby rozwiązanie było poprawne. Ograniczenia występujące w problemie MS-RCPSP to: ograniczona liczba zasobów, ograniczenia pierwszeństwa (jeżeli zadanie posiada poprzedników, wszyscy poprzednicy muszą zostać zrealizowani przed rozpoczęciem tego zadania), brak współpracy przy wykonywaniu zadań, brak wywłaszczenia zadań (przerwanie wykonywania zadania jest niemożliwe) oraz posiadanie przez wykonujący zadanie zasób umiejętności wymaganych do wykonania tego zadania. Problem MS-RCPSP uzupełnia podstawową formalną definicję problemu harmonogramowania, czyli problemu RCPSP (ang. *Resource-Constrained Project Scheduling Problem*) [24,33,55], o umiejętności - do wykonania zadań wymagane jest posiadanie umiejętności pewnego rodzaju i na pewnym poziomie, a każdy zasób posiada zbiór pewnych umiejętności.

Problem MS-RCPSP może być rozwiązywany jako problem jedno- lub wielokryterialny [6,28,49,50]. W optymalizacji jednokryterialnej celem jest znalezienie rozwiązania optymalnego względem jednego kryterium - definicja „rozwiązania optymalnego” dla jednego kryterium nie jest trudna, a do oceny samych rozwiązań wystarczy jedynie ocena względem tego kryterium. W optymalizacji wielokryterialnej wiele kryteriów analizowane jest jednocześnie i możliwe jest przyjęcie różnych wag kryteriów (kryteria mogą mieć równe wagi lub niektóre kryteria mogą być ważniejsze od innych) [50]. Wtedy porównanie dwóch rozwiązań, w których jedno ma lepszą wartość w jednym kryterium, a drugie ma lepszą wartość w drugim kryterium, nie jest proste - które z tych dwóch rozwiązań jest „lepsze”? Celem optymalizacji wielokryterialnej nie jest znalezienie jednego rozwiązania optymalnego, ale całego zbioru rozwiązań, które są tak samo dobrej jakości, czyli z których nie da się wybrać najlepszego rozwiązania. W celu znalezienia zbioru równie dobrych rozwiązań wymagane jest zastosowanie nieco innego podejścia niż w optymalizacji jednokryterialnej. Rzeczywiste problemy mające zastosowanie w biznesie, w tym problem harmonogramowania, są często wielokryterialne - na przykład

dobry harmonogram powinien być jednocześnie jak najkrótszy i jak najtańszy. Rozwiązywanie takich problemów biorąc pod uwagę tylko jedno kryterium jest dużym uproszczeniem i może nie dawać zadowalających efektów z punktu widzenia biznesu.

Cel pracy

Celem pracy jest zbadanie, czy zastosowanie inspiracji koewolucją w metodach opartych na algorytmie ewolucyjnym zwiększa skuteczność w rozwiązywaniu wielokryterialnego problemu MS-RCPSP. Do osiągnięcia celu pracy konieczne jest dokonanie przeglądu literatury, opracowanie koewolucyjnej metody opartej na algorytmie ewolucyjnym i mającej zastosowanie do rozwiązywania wielokryterialnego problemu MS-RCPSP, implementacja narzędzi umożliwiających przeprowadzenie badań, wykonanie badań oraz analiza ich rezultatów włącznie z weryfikacją istotności statystycznej uzyskanych różnic i wyciągnięcie wniosków. W ramach pracy zrealizowano następujące punkty, będące celami pośrednimi składającymi się na realizację celu pracy:

1. Zaimplementowano przyjętą reprezentację problemu MS-RCPSP w oparciu o bibliotekę iMOPSE [3].

2. W oparciu o przegląd literatury opracowano autorską metodę do rozwiązywania wielokryterialnego problemu MS-RCPSP, opartą na algorytmie ewolucyjnym (w szczególności na ewolucji różnicowej) i wykorzystującą inspirację zjawiskiem koewolucji.

3. Zaimplementowano autorską metodę w formie umożliwiającej wybór sposobu wyboru osobników w momencie ich koewolucyjnej oceny oraz metody wybrane jako punkt odniesienia: metodę losową i metodę opartą na ewolucji różnicowej, w której wykorzystywany jest operator selekcji GAP. Implementacja metod również oparta jest o bibliotekę iMOPSE [3].

4. Dokonano strojenia, czyli doboru parametrów, metod dla badanych wzorcowych (ang. *benchmark*) instancji problemu MS-RCPSP [50]. Strojenia dokonano dla autorskiej metody we wszystkich wariantach oraz metod wybranych jako punkt odniesienia.

5. Przeprowadzono badania skuteczności metod w rozwiązywaniu wielokryterialnego problemu MS-RCPSP wykorzystując wzorcowe instancje problemu [50]. Badania przeprowadzono dla wszystkich wariantów autorskiej metody oraz metod wybranych jako punkt odniesienia.

6. Dokonano analizy uzyskanych wyników badań i porównano wyniki uzyskane dla autorskiej metody we wszystkich wariantach oraz dla metod wybranych jako punkt odniesienia. Porównano wyniki uzyskane dla autorskiej metody działającej we wszystkich trybach oraz dla metod z literatury. Zweryfikowano istotność statystyczną uzyskanych różnic w wynikach. Wyciągnięto wnioski.

Struktura pracy

Praca dzieli się na 7 rozdziałów:

Niniejszy rozdział stanowi wstęp do problematyki pracy. Zawiera opis przedmiotu pracy oraz stawianego celu i celów pośrednich. Rozdział kończy przedstawienie struktury pracy.

W rozdziale 1. opisano i formalnie zdefiniowano wybrany silnie ograniczony i wielokryterialny problem harmonogramowania - problem MS-RCPSP - oraz przedstawiono najczęściej stosowane kryteria oceny jego rozwiązań.

Rozdział 2 zawiera przegląd literatury dotyczącej problematyki pracy. Rozdział rozpoczyna omówienie podstaw optymalizacji wielokryterialnej, w tym definicje relacji dominacji Pareto oraz zbioru i frontu Pareto. Przedstawiono stan wiedzy na temat zastosowania algorytmu ewolucyjnego do rozwiązywania problemów wielokryterialnych, a także zasadę działania wybranych podejść. Rozdział kończy omówienie wykorzystania podejść koewolucyjnych do rozwiązywania problemów optymalizacyjnych zgodnie z zaproponowanym podziałem na metody kooperacyjne, konkurencyjne i mieszane.

W rozdziale 3. opisano działanie autorskiej metody koewolucyjnej do rozwiązywania wielokryterialnego problemu MS-RCPSP opracowanej na podstawie przeglądu literatury. Określono również różnice w różnych wariantach autorskiej metody.

W rozdziale 4. przedstawiono założenia metodologiczne badań oraz organizację badań. Omówiono wykorzystane narzędzia oraz charakterystykę instancji testowych wybranego silnie ograniczonego i wielokryterialnego problemu harmonogramowania. Zdefiniowano miary oceny jakości przybliżeń prawdziwego frontu Pareto uzyskanych w wyniku działania optymalizatora wielokryterialnego, które zastosowano do oceny skuteczności metod. Postawiono pytania badawcze. Omówiono proces strojenia metod oraz sformułowano plan badań.

Rozdział 5. zawiera szczegółowe wyniki badań oraz ich interpretację. Dokonano również weryfikacji istotności statystycznej dla uzyskanych różnic w wynikach. W dalszej części rozdziału 5. W sposób zbiorczy podsumowano wyniki badań, a także wyciągnięto wnioski i odpowiedziano na pytania badawcze.

Rozdział 6 stanowi zakończenie pracy. Zawiera podsumowanie wyników pracy i wnioski końcowe z pracy, a także weryfikację osiągnięcia celu pracy. W tym rozdziale przedstawiono również możliwe dalsze kierunki badań.

Podziękowania

Mojemu promotorowi panu dr hab. inż. Pawłowi Myszkowskiemu pragnę serdecznie podziękować za prowadzenie niniejszej pracy dyplomowej, wsparcie i poświęcony czas, a także za zrozumienie i wyrozumiałość. Mojej prowadzącej seminarium dyplomowego pani dr hab. inż. Adriannie Kozierkiewicz bardzo dziękuję za cenne wskazówki dotyczące weryfikacji istotności statystycznej uzyskanych różnic w wynikach. Panu mgr inż. Michałowi Antkiewiczowi serdecznie dziękuję za udostępnienie wyników uruchomienia metod NTGA2 oraz B-NTGA.

Panu mgr inż. Konradowi Gmyrkowi bardzo dziękuję za pomoc w weryfikacji poprawności kodu źródłowego. Wszystkim pracownikom dydaktycznym, którzy przez cały czas trwania trwania mojej edukacji wyższej prowadzili zajęcia, w których uczestniczyłam, pragnę podziękować za przekazaną wiedzę i poświęcony czas.

1. Wybrany silnie ograniczony wielokryterialny problem harmonogramowania

Harmonogramowanie zadań jest integralnym elementem zarządzania projektem i polega na planowaniu kolejności wykonywania zadań oraz określaniu, który zasób będzie odpowiedzialny za wykonanie którego zadania. Z uwagi na ograniczoną liczbę zasobów zadania konkurują ze sobą o dostęp do nich. Pomiędzy zadaniami mogą występować ograniczenia pierwszeństwa - aby wykonać jedno zadanie, należy wcześniej zakończyć wszystkich jego poprzedników. Znaną formalną definicją tego zagadnienia jest silnie ograniczony problem harmonogramowania, czyli problem RCPSP [24, 33, 55]. Ponieważ problem RCPSP jest problemem NP-trudnym [11] nie istnieje algorytm, który mógłby wyznaczyć optymalne rozwiązanie w czasie wielomianowym. Z tego powodu do jego rozwiązywania zastosowanie mają metaheurystyki, które pozwalają na znalezienie dobrego rozwiązania w akceptowalnym czasie, przy czym znalezione rozwiązanie może nie być rozwiązaniem optymalnym.

Istnieje wiele wariantów i rozszerzeń problemu RCPSP [22, 23] - możliwe są wywłaszczenie zadań, wykonywanie zadań w różnych trybach, definicja daty wydania zadania jako najwcześniejszego czasu rozpoczęcia zadania i terminu ostatecznego jako najpóźniejszego czasu zakończenia zadania, definicja czasu transportu dla zasobów wykonujących zadania w różnych lokalizacjach, ocena harmonogramów według różnych kryteriów i inne modyfikacje. Jednym z rozszerzeń problemu RCPSP jest silnie ograniczony problem harmonogramowania z wieloma umiejętnościami, czyli problem MS-RCPSP [47, 68, 75]. W tym wariancie istnieje dodatkowe ograniczenie związane z umiejętnościami wymaganymi do wykonania zadań. Każdy zasób posiada określony zbiór swoich umiejętności i aby możliwe było wykonanie zadania przez dany zasób, musi on posiadać umiejętność wymaganą do wykonania tego zadania.

Rozwiązaniem problemu MS-RCPSP jest harmonogram, w którym każde zadanie posiada określone czasy rozpoczęcia i zakończenia oraz do każdego zadania przypisany jest wykonujący je zasób. Każde zadanie wykonywane jest przez pewną określoną ilość czasu - jest to czas trwania zadania. Nie jest możliwe wywłaszczenie zadań, czyli jeżeli zadanie zostanie rozpoczęte, wykonujący je zasób jest zajęty przez cały czas jego trwania i nie może przerwać jego wykonywania. Dodatkowo, między zadaniami mogą występować ograniczenia pierwszeństwa, czyli zadanie może posiadać zbiór swoich poprzedników, których zakończenie jest wymagane, aby rozpocząć wykonywanie tego zadania. W problemie MS-RCPSP czas postrzegany jest jako zbiór dyskretnych momentów czasu i w jednym momencie czasu zasób może wykonywać tylko jedno zadanie. Zasoby nie mogą również współpracować przy wykonywaniu zadań - zadanie może być wykonywane przez tylko jeden zasób. Jedną z różnic w stosunku do problemu RCPSP jest fakt, że zasoby są zasobami ludzkimi i jako takie otrzymują pewne wynagrodzenie

za wykonywaną pracę. Różne zasoby mają różne godzinowe stawki wynagrodzenia, które odzwierciedlają różne stawki godzinowe osób o różnym stażu pracy lub zatrudnionych na różnych stanowiskach. Drugą różnicą w stosunku do klasycznego problemu RCPSP jest fakt, że do wykonania zadań wymagane jest posiadanie określonej umiejętności i z tego powodu nie każdy zasób może wykonać każde zadanie. Każdy zasób posiada pewne umiejętności i dla każdego typu umiejętności określony jest poziom znajomości tej umiejętności, na przykład pewien zasób posiada umiejętność „Programowanie w języku C++" na poziomie średnioza-awansowanym oraz umiejętność „Programowanie w języku Java" na poziomie zaawansowanym. Uwzględnienie umiejętności wymaganych do wykonania zadań oznacza, że zasób R może wykonać zadanie T tylko wtedy, gdy R posiada umiejętność wymaganą do wykonania T na poziomie wymaganym przez T lub wyższym. Przykład reprezentacji możliwości wykonania zadań przez zasoby w formie macierzy umiejętności zaprezentowano na rysunku 1.1.

Umiejętności zdefiniowane w projekcie: *S1, S2, S3*
Możliwe poziomy umiejętności: *Sx.1, Sx.2, Sx.3*

✓ Zasób może wykonać zadanie
✗ Zasób nie może wykonać zadania

		Zadania			
		T1	***T2***	***T3***	***T4***
		S2.2	*S3.1*	*S2.2*	*S1.1*
Zasoby	***R1*** *S1.3, S2.2*	✓	✗	✓	✓
	R2 *S2.1, S3.2*	✗	✓	✗	✗
	R3 *S1.2, S2.1*	✗	✗	✗	✓
	R4 *S2.2, S3.3*	✓	✓	✓	✗

Rysunek 1.1: Macierz umiejętności w problemie MS-RCPSP, zaadaptowano z [48].

W macierzy umiejętności zaprezentowanej na rysunku 1.1 umiejętności wymagane do wykonania zadania zapisane zostały pod identyfikatorem zadania, a umiejętności posiadane przez zasoby obok identyfikatora zasobu. Na przykład, do wykonania $T1$ wymagana jest umiejętność typu $S2$ na poziomie znajomości dwa, a $R2$ posiada umiejętność typu $S2$ na poziomie znajomości jeden oraz umiejętność typu $S3$ na poziomie znajomości dwa. Dla zadanej macierzy umiejętności $R1$ może wykonać $T1$, $T3$ oraz $T4$ - w szczególności może wykonać $T4$ ponieważ jego poziom znajomości dla umiejętności typu $S1$ jest wyższy niż poziom wymagany przez $T4$. Odwrotna sytuacja następuje dla $R2$ i $R3$ - żaden z nich nie może wykonać $T1$ ani $T3$, ponieważ mimo że posiadają oni umiejętność typu $S2$, mają zbyt niski poziom jej znajomości. $R2$ może wykonać tylko $T2$, a $R3$ tylko $T4$. $R4$ może wykonać $T1$, $T2$ oraz $T3$ i w szczególności może wykonać $T2$ ponieważ jego poziom znajomości dla umiejętności typu $S3$ jest wyższy niż poziom wymagany przez $T2$ - analogicznie do $R1$ i $T4$. Nie ma żadnego

zadania, dla którego nie byłoby zasobu, który może je wykonać. Zasoby mogą wykonywać różne liczby zadań, na przykład $R3$ może wykonać tylko jedno zadanie, a $R4$ aż trzy zadania.

Rozwiązaniem problemu MS-RCPSP jest harmonogram, który jest jednocześnie poprawny i najlepszy pod względem wybranych kryteriów. Poprawny harmonogram spełnia wszystkie ograniczenia związane z umiejętnościami, kolejnością zadań oraz dostępnością zasobów. Ponieważ dla zachowania poprawności rozwiązania wszystkie ograniczenia muszą być spełnione, problem MS-RCPSP jest silnie ograniczony. Formalną definicję problemu MS-RCPSP przedstawiono w podrozdziale 1.1. Stosowane kryteria oceny harmonogramów zdefiniowano w podrozdziale 1.2.

1.1. Formalna definicja problemu MS-RCPSP

Do sformułowania formalnej definicji problemu MS-RCPSP [34, 43, 47, 48, 56] wykorzystane zostaną następujące oznaczenia:

- Parametry:
  - d_t - czas trwania zadania t,
  - P_t - zbiór poprzedników zadania t,
  - s_t - umiejętność wymagana do wykonania zadania t,
  - r_{salary} - wynagrodzenie godzinowe zasobu r,
  - S_r - zbiór umiejętności posiadanych przez zasób r.
- Zmienne decyzyjne:
  - F_t - czas zakończenia zadania t,
  - $U_{t,r}^{ts}$ - przypisanie zasobu r do zadania t w momencie czasu ts.

Problem optymalizacyjny MS-RCPSP jest, jako rozszerzenie problemu RCPSP, kombinatorycznym problemem NP-trudnym [11]. Podstawowymi elementami problemu są zbiór zadań T i zbiór zasobów R. Zbiory zadań i zasobów są ze sobą powiązane zbiorem umiejętności S. Zadania ze zbioru T mogą być powiązane ze sobą nawzajem ograniczeniami pierwszeństwa - definiują one dla każdego zadania zbiór jego poprzedników P_t, czyli zbiór zadań, które muszą być zakończone przed rozpoczęciem wykonywania tego zadania. Każde zadanie posiada również pewien czas trwania d_t. W harmonogramie PS, stanowiącym rozwiązanie problemu, każde zadanie posiada czas zakończenia F_t oraz czas rozpoczęcia $F_t - d_t$. Do wykonania każdego zadania wymagana jest pewna umiejętność s_t, posiadająca określony typ h_{s_t} oraz poziom znajomości l_{s_t}. Każdy zasób r posiada określone wynagrodzenie godzinowe r_{salary}. Oprócz tego, do każdego zasobu przypisany jest zbiór posiadanych przez niego umiejętności S_r, który jest podzbiorem S. Każda umiejętność zasobu s_r posiada określony typ h_{s_r} oraz poziom znajomości l_{s_r}. Żeby zasób r mógł wykonać zadanie t musi posiadać umiejętność typu h_{s_t} na poziomie znajomości równym l_{s_t} lub od niego większym. W harmonogramie PS każdy zasób wykonuje pewien zbiór zadań T^r, będący podzbiorem T. Definiowana jest także

zmienna określająca, czy w momencie czasu ts zasób r wykonuje zadanie t: $U_{t,r}^{ts} \in \{0,1\}$. Wartość $U_{t,r}^{ts} = 1$ oznacza, że r wykonuje t w momencie czasu ts. Jeżeli jednak r nie wykonuje t w momencie czasu ts: $U_{t,r}^{ts} = 0$. Przypisanie zadaniom wykonujących je zasobów można zdefiniować jako funkcję $\alpha : T \to R$. Zapis $\alpha(t) = r$ oznacza, że zasób r wykonuje zadanie t bez przerwy od momentu $F_t - d_t$ do momentu F_t.

Formalną definicję problemu MS-RCPSP przedstawia równanie 1.1, gdzie Ω jest przestrzenią rozwiązań, a f to funkcja celu. Rozwiązaniem problemu jest poprawny harmonogram PS. Tworzą go zadania umieszczone na osi czasu, do których przypisane są wykonujące je zasoby. Harmonogram PS jest poprawny, jeżeli spełnia ograniczenia opisane równaniami 1.2 - 1.10.

$$f : \Omega \to \mathbb{R},\ min(f) \tag{1.1}$$

$$\forall_{r \in R}\, r_{salary} \geqslant 0 \tag{1.2}$$

$$\forall_{r \in R}\, S_r \neq \emptyset \tag{1.3}$$

$$\forall_{t \in T}\, d_t \geqslant 0 \tag{1.4}$$

$$\forall_{t \in T}\, F_t \geqslant 0 \tag{1.5}$$

$$\forall_{t \in T,\, t \neq 1}\, \forall_{p \in P_t}\, F_p \leqslant F_t - d_t \tag{1.6}$$

$$\forall_{r \in R}\, \forall_{t \in T^r}\, \exists_{s_r \in S^r}\, h_{s_r} = h_{s_t} \ \wedge\ l_{s_r} \geqslant l_{s_t} \tag{1.7}$$

$$\forall_{r \in R}\, \forall_{ts \in \tau}\, \sum_{t=1}^{n} U_{t,r}^{ts} \leqslant 1 \tag{1.8}$$

$$\forall_{t \in T}\, \exists!_{r \in R}\, \exists_{ts \in \tau}\, U_{t,r}^{ts} = 1 \tag{1.9}$$

$$\alpha(t) = r \ \Rightarrow\ \forall_{F_t - d_t \leqslant ts \leqslant F_t}\, U_{t,r}^{ts} = 1 \ \wedge\ \forall_{ts \notin [F_t - d_t,\, F_t]}\, U_{t,r}^{ts} = 0 \tag{1.10}$$

Ograniczenia 1.2 - 1.4 dotyczą poprawności instancji problemu. Ograniczenie 1.2 określa godzinowe stawki wynagrodzenia zasobów - stawki godzinowe nie mogą mieć wartości ujemnych. O wymaganiach dla umiejętności zasobów mówi ograniczenie 1.3 - każdy zasób musi posiadać co najmniej jedną umiejętność. Nieujemny czas trwania zadań gwarantuje ograniczenie 1.4. Ograniczenia 1.5 - 1.10 dotyczą poprawności harmonogramu. W harmonogramie każde zadanie jest opisywane nieujemnym czasem zakończenia - mówi o tym ograniczenie 1.5. Reprezentacją ograniczeń pierwszeństwa jest ograniczenie 1.6. Przed rozpoczęciem zadania każdy jego poprzednik musi być wykonany. Czas rozpoczęcia zadania t to $F_t - d_t$ i nie może być on mniejszy niż czas zakończenia F_p dla żadnego z poprzedników zadania należących do zbioru P^t. Posiadanie przez zasoby umiejętności wymaganych do wykonania zadań gwarantuje

ograniczenie 1.7. Dla każdego z zadań przypisanych do zasobu r, czyli zadań należących do zbioru T^r, wśród umiejętności zasobu S^r musi istnieć umiejętność takiego samego typu, jaki jest typ umiejętności wymaganej przez zadanie. Oprócz tego, zasób musi posiadać umiejętność tego typu na poziomie znajomości równym poziomowi znajomości wymaganemu przez zadanie lub od niego większym. W danym momencie czasu zasób może wykonywać co najwyżej jedno zadanie - mówi o tym ograniczenie 1.8. W danym momencie czasu zasób może także nie wykonywać żadnego zadania. Może nawet wystąpić sytuacja, w której zasób przez cały czas trwania harmonogramu nie wykonuje żadnego zadania. Jednym z założeń problemu MS-RCPSP jest brak współpracy zasobów przy wykonywaniu zadań. To założenie jest gwarantowane przez ograniczenie 1.9 jednocześnie z gwarancją wykonania wszystkich zadań w projekcie - dla każdego zadania musi istnieć dokładnie jeden przypisany do niego zasób. Ograniczenie 1.10 dotyczy braku wywłaszczenia zadań. Jeżeli zasób przypisany jest do zadania, wykonuje to zadanie przez cały jego czas trwania, a poza czasem wykonywania tego zadania nie pracuje nad nim.

Całkowita przestrzeń rozwiązań SS problemu MS-RCPSP zawiera i poprawne, i niepoprawne harmonogramy oraz może być wyznaczona jako:

$$SS(n, m) = n! \cdot m^n \tag{1.11}$$

Aby znaleźć liczbę wszystkich możliwych kolejności wykonywania zadań należy wyznaczyć liczbę permutacji n zadań, czyli $n!$. W tej liczbie zawarte są również takie kolejności wykonywania zadań, dla których złamane są ograniczenia pierwszeństwa zadań. Aby znaleźć liczbę wszystkich możliwych przypisań zasobów do zadań, należy wyznaczyć liczbę n-wyrazowych wariacji ze zbioru m zasobów, czyli m^n. Oczywiście w tej liczbie zawarte są także takie przypisania, dla których nie są zachowane ograniczenia związane z wymaganymi do wykonania zadań umiejętnościami. Dla „łatwego" projektu, w którym występuje 100 zadań i 20 zasobów istnieje $SS(100, 20) = 1,18 \cdot 10^{288}$ rozwiązań, z czego część to rozwiązania niepoprawne.

1.2. Kryteria oceny rozwiązań problemu MS-RCPSP

Według równania 1.1 problem MS-RCPSP jest problemem minimalizacyjnym. Harmonogram oceniany jest według pewnych ustalonych kryteriów. Najczęściej rozpatrywane kryteria to czas trwania harmonogramu i koszt harmonogramu [44]:

$$f_\tau(PS) = \max_{t \in T} F_t \tag{1.12}$$

$$f_C(PS) = \sum_{i=1}^{n} R_i^{salary} \cdot d_i \tag{1.13}$$

Równanie 1.12 definiuje całkowity czas trwania harmonogramu - jest on równy czasowi zakończenia F_t ostatniego zadania w harmonogramie. Całkowity koszt harmonogramu jest definiowany przez równanie 1.13, gdzie n jest liczbą przypisań zasobów do zadań i R_i^{salary} to godzinowa stawka wynagrodzenia zasobu przypisanego do zadania w i-tym przypisaniu zasób-zadanie. Całkowity koszt harmonogramu jest równy sumie wynagrodzeń wszystkich zasobów

za wykonywanie zadań, a za wykonanie jednego zadania zasobowi przysługuje wynagrodzenie równe iloczynowi jego stawki godzinowej i czasu trwania zadania.

Możliwe jest rozpatrywanie problemu MS-RCPSP jako problemu jednokryterialnego - najczęściej rozpatrywany jest całkowity czas trwania harmonogramu [28,39,49,56,74]. Formalną definicję problemu MS-RCPSP jako problemu jednokryterialnego opisuje równanie 1.14. W tym przypadku definicja „najlepszego rozwiązania" jest jednoznaczna - dla problemu MS-RCPSP z kryterium czasu najlepszy harmonogram to taki, który ma najkrótszy czas trwania. Porównanie dwóch rozwiązań w celu wyłonienia „lepszego" z nich jest trywialne, ponieważ „lepszym" rozwiązaniem jest zawsze ten harmonogram, który trwa krócej. Analiza tylko kryterium czasu jest jednak pewnym ograniczeniem - na przykład najkrótszy harmonogram może być jednocześnie bardzo drogi, co jest nieopłacalne z punktu widzenia firmy wykonującej projekt.

$$minf(PS) = min[f_\tau(PS)] \tag{1.14}$$

Często spotykanym podejściem jest rozpatrywanie problemu wielokryterialnego MS-RCPSP jako problemu dwukryterialnego z kryteriami czasu 1.12 i kosztu 1.13 [37, 68, 75]. Wtedy definicja problemu MS-RCPSP jako problemu wielokryterialnego jest następująca:

$$minf(PS) = min[f_\tau(PS), f_C(PS)] \tag{1.15}$$

Biorąc pod uwagę kilka kryteriów definicja „najlepszego rozwiązania" nie jest trywialna. Jeden harmonogram może być od drugiego dużo krótszy, jednocześnie będąc dużo droższym - który z nich jest wtedy „lepszym" rozwiązaniem? W takiej sytuacji, ocena rozwiązania jako „lepsze" lub „gorsze" leży w gestii użytkownika, czyli w tym przypadku osoby odpowiedzialnej za harmonogramowanie projektu. Podjęta przez użytkownika decyzja może zależeć od różnych, często trudno definiowalnych czynników. Możliwe jest również, że inny użytkownik podjąłby dla tej samej pary harmonogramów inną decyzję. Definicja „najlepszego rozwiązania" nie jest więc jednoznaczna dla kilku kryteriów optymalizacji. Jednym z podejść do rozwiązywania dwukryterialnego problemu MS-RCPSP jest wprowadzenie ważonej funkcji celu, która łączy ze sobą kryteria czasu trwania harmonogramu i kosztu harmonogramu. Ważona funkcja celu wykorzystuje normalizowane wartości $f'_\tau(PS)$ oraz $f'_C(PS)$, definiowane jako [45, 47, 49]:

$$f'_\tau(PS) = \frac{\tau}{\tau_{max}} \tag{1.16}$$

$$f'_C(PS) = \frac{\sum_{i=1}^{n} c_i}{c_{max} - c_{min}} \tag{1.17}$$

Równanie 1.16 definiuje normalizowaną wartość kryterium czasu trwania harmonogramu - jest ona równa czasowi trwania harmonogramu τ podzielonemu przez pesymistyczny (maksymalny) czas trwania harmonogramu τ_{max}. Maksymalny czas trwania harmonogramu τ_{max} uzyskiwany jest dla harmonogramu, w którym wszystkie zadania wykonywane są jedno po drugim. Normalizowana wartość kryterium kosztu harmonogramu jest definiowana przez równanie 1.17, gdzie n to liczba przypisań zasób-zadanie, c_i to koszt wykonania i-tego zadania

(koszt wykonania zadania jest równy czasowi trwania zadania pomnożonemu przez godzinowe wynagrodzenie zasobu wykonującego to zadanie), a c_{max} i c_{min} to odpowiednio największy i najmniejszy całkowity koszt harmonogramu. Wartości c_{max} i c_{min} wyznaczane są bez brania pod uwagę ograniczeń związanych z umiejętnościami, zatem c_{max} to koszt harmonogramu, w którym wszystkie zadania wykonywane są przez najdroższy zasób, a c_{min} to koszt harmonogramu, w którym wszystkie zadania wykonywane są przez najtańszy zasób. Wykorzystując ważoną funkcję celu można zdefiniować problem MS-RCPSP jako:

$$minf(PS) = \omega_\tau f'_\tau(PS) + (1 - \omega_\tau) f'_C(PS) \tag{1.18}$$

Parametr ω_τ oznacza wagę komponentu czasowego i przyjmuje wartości z przedziału $[0, 1]$. Wykorzystanie tego parametru pozwala określić, które z kryteriów jest ważniejsze w danym procesie optymalizacyjnym. Możliwe jest przeprowadzenie optymalizacji jednokryterialnej - dla wartości $\omega_\tau = 1$ optymalizowany jest tylko czas trwania harmonogramu, a dla wartości $\omega_\tau = 0$ optymalizowany jest tylko koszt harmonogramu. Dla wartości $\omega_\tau = 0,5$ oba kryteria są równoważne - jest to optymalizacja zbalansowana [49]. Pozostałe wartości ω_τ pozwalają ustalić bardziej szczegółowe proporcje wagi kryteriów w zależności od potrzeb użytkownika. Takie podejście pozwala na transformację problemu wielokryterialnego na problem jednokryterialny, gdzie rolę kryterium pełni ważona funkcja fitness. Dla takiego prostego rzutowania pominięte zostaje wiele rozwiązań, ale bardzo ułatwiona jest metoda rozwiązywania - podejście jest więc akceptowalne dla niektórych praktycznych zastosowań. Nie jest to jednak rozwiązywanie problemu jako problemu rzeczywiście wielokryterialnego oraz wymaga ustalenia wag kryteriów, co może nie być łatwym zadaniem. Alternatywą dla tego podejścia jest podejście oparte o definicję relacji dominacji Pareto, które przedstawiono w podrozdziale 2.1 i w którym problem rzeczywiście rozwiązywany jest jako problem wielokryterialny, czyli analizowane jest kilka kryteriów wyrażonych niezależnymi funkcjami.

2. Teoretyczne podstawy problematyki badań

W rozdziale 1 przedstawiono wybrany silnie ograniczony problem harmonogramowania - problem MS-RCPSP, którego rozwiązywanie jest pierwszym elementem przedmiotu pracy. Może być on rozwiązywany jako problem wielokryterialny, na przykład biorąc pod uwagę jednocześnie kryteria czasu i kosztu. Rozwiązywanie problemów optymalizacyjnych z wieloma kryteriami wymaga zastosowania innego podejścia niż dla problemów optymalizacyjnych z jednym kryterium. Drugim elementem przedmiotu pracy jest wykorzystanie inspiracji koewolucją w algorytmie ewolucyjnym. Koewolucja polega na występowaniu interakcji między różnymi gatunkami, które wpływają na adaptację tych warunków do aktualnych warunków środowiska. Algorytm ewolucyjny rozszerzony o inspirację koewolucją może operować na kilku powiązanych ze sobą populacjach. Przeglądu literatury dokonano w celu znalezienia odpowiedzi na następujące pytania:

1. Jakie są teoretyczne podstawy metod optymalizacji wielokryterialnej?
2. Jakie metody ewolucyjne stosowane są do rozwiązywania problemów wielokryterialnych?
3. Które z metod ewolucyjnych stosowane są do rozwiązywania problemu MS-RCPSP?
4. Na czym polegają podejścia koewolucyjne i jakie są ich rodzaje?
5. Które z rodzajów podejść koewolucyjnych stosowane są do rozwiązywania problemu MS-RCPSPS?

W podrozdziale 2.1 przedstawiono teoretyczne podstawy optymalizacji wielokryterialnej. W podrozdziale 2.2 omówiono stosowane w literaturze podejścia do rozwiązywania problemów wielokryterialnych. Zasadę działania i podział podejść koewolucyjnych przedstawiono w podrozdziale 2.3. Podrozdział 2.4 zawiera podsumowanie przeglądu literaturowego.

2.1. Optymalizacja wielokryterialna

W wielokryterialnych problemach optymalizacyjnych (ang. *Multi-Objective Optimisation Problem*, MOP) [70] rozwiązanie oceniane jest względem kilku kryteriów. Z tego powodu definicja „najlepszego rozwiązania" nie jest trywialna, ponieważ często występują sytuacje, w których rozwiązanie jest lepsze od drugiego rozwiązania względem jednego kryterium, ale gorsze względem innego. Obejściem tego problemu może być wykorzystanie zaprezentowanej w podrozdziale 1.2 funkcji ważonej - wtedy problem wielokryterialny zamieniany jest na problem jednokryterialny, gdzie kryterium jest funkcja ważona. Najczęściej jednak stosowane w literaturze podejście oparte jest na definicji relacji dominacji. W tym podejściu jako wynik

optymalizacji uzyskany zostaje zbiór równie dobrych rozwiązań, spośród których nie da się wyznaczyć „zwycięzcy".

Problem optymalizacyjny może być zdefiniowany jako krotka (X, Z, f, rel), gdzie X oznacza przestrzeń poszukiwań, inaczej nazywaną przestrzenią decyzyjną (ang. *decision space*), Z oznacza przestrzeń celu (ang. *objective space*), funkcja $f : X \to Z$ przypisuje dla każdego rozwiązania (wektora decyzyjnego) $x \in X$ odpowiadający mu wektor kryteriów $z \in Z$, a rel jest relacją częściowego porządku na Z [32]. Dla problemu jednokryterialnego najczęściej używaną relacją porządku rel jest relacja $\leqslant$, czyli „mniejsze lub równe". Dzięki temu zawsze istnieje jedna optymalna wartość $z^* \in f(X)$. Dla problemu wielokryterialnego definiowane są trzy relacje częściowego porządku rel: relacja ostrej dominacji (ang. *strict dominance relation*), relacja dominacji (ang. *dominance relation*) oraz relacja słabej dominacji (ang. *weak dominance relation*). Relacja słabej dominacji $\preceq$, gdzie $z^1 \preceq z^2 \Leftrightarrow \forall_{i\in\{1,2,\ldots,n\}}\, z_i^1 \leqslant z_i^2$, a n to liczba kryteriów, jest rozszerzeniem relacji nierówności $\leqslant$ na $\mathbb{R}^n$. Wektor z^1 słabo dominuje wektor z^2, jeżeli dla każdego kryterium jest nie gorszy niż z^2. Odpowiadającym relacji słabej dominacji ostrym częściowym porządkiem jest relacja dominacji $\prec$, gdzie $z^1 \prec z^2 \Leftrightarrow z^1 \preceq z^2 \wedge \neg z^2 \preceq z^1$. Wektor z^1 dominuje wektor z^2, jeżeli dla każdego kryterium jest nie gorszy od z^2 i jednocześnie dla przynajmniej jednego kryterium jest lepszy od z^2. Zdefiniowana jest także relacja ostrej dominacji $\prec\prec$, gdzie $z^1 \prec\prec z^2 \Leftrightarrow \forall_{i\in\{1,2,\ldots,n\}}\, z_i^1 < z_i^2$, a n to liczba kryteriów, czyli wektor z^1 dominuje wektor z^2, jeżeli jest od niego lepszy w każdym kryterium. W implementacjach optymalizatorów wielokryterialnych wykorzystywana jest relacja dominacji [16, 42, 43, 57, 64]. Definicje różnych relacji częściowego porządku podsumowano w tabeli 2.1.

Tabela 2.1: Definicje wybranych relacji częściowego porządku na przestrzeni kryteriów, zaadaptowano z [32].

Relacja	Symbol	Interpretacja w przestrzeni kryteriów
ostro dominuje	$z^1 \prec\prec z^2$	z^1 jest lepsze od z^2 w każdym kryterium
dominuje	$z^1 \prec z^2$	z^1 jest nie gorsze od z^2 w każdym kryterium i lepsze w przynajmniej jednym
słabo dominuje	$z^1 \preceq z^2$	z^1 jest nie gorsze niż z^2 w każdym kryterium

Różnicą między optymalizacją jednokryterialną a wielokryterialną jest fakt, że w optymalizacji wielokryterialnej zamiast jednego rozwiązania optymalnego najlepszego pod względem analizowanego kryterium możliwe jest istnienie wielu rozwiązań, które nie są zdominowane przez żadne inne rozwiązania. Takie rozwiązania nazywane są rozwiązaniami niezdominowanymi (ang. *non-dominated*). Zbiór Pareto (ang. *Pareto-optimal set*) jest więc definiowany jako $X^* = \{x \in X \mid \nexists_{y\in X}\, y \prec x\}$. Odpowiadające elementom zbioru Pareto wektory kryteriów tworzą front Pareto (ang. *Pareto-optimal front, Pareto front*) $F^* = \{f(x) \mid x \in X^*\}$. Ponieważ kilka wektorów decyzyjnych może być mapowanych na ten sam wektor kryteriów, front Pareto może posiadać mniej elementów niż zbiór Pareto [16, 32, 64].

W teorii celem optymalizacji wielokryterialnej jest znalezienie wszystkich globalnie niezdominowanych rozwiązań, które tworzą prawdziwy front Pareto (ang. *true Pareto front*). Niestety, przy rozwiązywaniu problemów rzeczywistych (ang. *real world*) bardzo często prawdziwy front Pareto nie jest znany i znalezienie go jest niewykonalne na przykład dlatego,

że rozwiązywany problem jest NP-trudny lub dlatego, że niezdominowanych rozwiązań jest zbyt dużo. Z tego powodu front Pareto uzyskiwany w wyniku procesu optymalizacyjnego jest tylko przybliżeniem prawdziwego frontu Pareto (ang. *Pareto front approximation*), a celem optymalizacji wielokryterialnej jest znalezienie najlepszego przybliżenia. Ponieważ znalezienie prawdziwego frontu Pareto jest niewykonalne, w dalszej części pracy sformułowania „front Pareto" i „przybliżenie prawdziwego frontu Pareto" będą używane wymiennie tam, gdzie nie będzie to wprowadzało niejasności.

W niektórych sytuacjach (na przykład do wyznaczenia niektórych miar oceny jakości przybliżeń prawdziwego frontu Pareto) potrzebne jest porównanie z prawdziwym frontem Pareto. Aby obejść konieczność estymowania go, można wykorzystać zdefiniowane punkty odniesienia (ang. *reference points*) [42]. Jeżeli prawdziwy front Pareto nie jest znany, jego przybliżenia mogą być porównywane do punktów, które posiadają określone cechy lub wartości kryteriów i jednocześnie są możliwe do wyznaczenia. Najczęściej wykorzystywane punkty to *Perfect Point* i *Nadir Point*. *Perfect Point*, inaczej *Utopia Point* lub *Ideal Point* (pl. punkt perfekcyjny, utopijny lub idealny), to punkt o najlepszych możliwych wartościach dla każdego z kryteriów. Jest to „najlepszy" możliwy punkt. Często jest punktem abstrakcyjnym, który nie może być osiągnięty przez żadne poprawne rozwiązanie. Dla problemu MS-RCPSP z kryteriami czasu i kosztu najlepszą wartością kryterium czasu jest czas trwania najkrótszego zadania pomnożony przez liczbę zadań i podzielony przez liczbę zasobów (jest to czas trwania hipotetycznego harmonogramu, w którym każde zadanie trwa tyle, co najkrótsze zadanie i wszystkie zasoby są równomiernie rozdysponowane do wykonywania zadań), a najlepszą wartością kryterium kosztu jest koszt harmonogramu, w którym wszystkie zadania wykonywane są przez najtańszy zasób. Przeciwieństwem *Perfect Point* jest *Nadir Point* (pl. najniższy lub najgorszy punkt), czyli punkt o najgorszych możliwych wartościach dla każdego z kryteriów - „najgorszy" możliwy punkt. *Nadir Point*, tak samo jak *Perfect Point*, często jest punktem abstrakcyjnym, niemożliwym do osiągnięcia przez żadne poprawne rozwiązanie. Często jako *Nadir Point* wybierany jest punkt jeszcze gorszy niż punkt o najgorszych możliwych wartościach każdego z kryteriów, ponieważ pozwala to na bardziej sprawiedliwe porównanie. Dla problemu MS-RCPSP z kryteriami czasu i kosztu do wyznaczenia punktu *Nadir Point* najgorsza wartość kryterium czasu wyznaczana jest jako czas trwania harmonogramu, w którym wszystkie zadania są wykonywane jedno po drugim przez tylko jeden zasób, a najgorsza wartość kryterium kosztu wyznaczana jest jako koszt harmonogramu, w którym wszystkie zadania wykonywane są przez najdroższy zasób.

2.2. Obliczenia ewolucyjne w optymalizacji jedno- i wielokryterialnej

Ewolucja w biologii to zmiany dziedziczonych cech organizmów z biegiem pokoleń. Biorąc pod uwagę środowisko, w którym może występować jednocześnie jedynie ograniczona liczba organizmów (na przykład z powodu ograniczonej ilości pożywienia), oraz instynkt jednostek do rozmnażania się nieunikniony staje się fakt, że nie wszystkie jednostki doczekają się potomstwa. Rozmnażają się te osobniki, których cechy wpływają na najlepsze ich przystosowanie do aktualnych warunków środowiska. To zjawisko nazywane jest selekcją lub doborem

naturalnym - z uwagi na ograniczone zasoby środowiska przetrwają i doczekają się potomstwa najlepiej przystosowane osobniki (ang. *survival of the fittest*). Ponieważ największą szansę na rozmnażanie mają najlepiej przystosowane osobniki i to one przekazują potomstwu swoje geny, ewolucyjne zmiany całego gatunku prowadzą do zwiększenia przeciętnej adaptacji organizmów do aktualnych warunków.

Wynikiem zastosowania inspiracji darwinowską teorią ewolucji do automatycznego rozwiązywania problemów są obliczenia ewolucyjne (ang. *evolutionary computing*) - obszar inteligencji obliczeniowej (ang. *computational intelligence*) rozwijający się od lat 40. XIX wieku [17]. W automatycznym rozwiązywaniu problemów pewnym rodzajem środowiska jest sam problem, rozwiązania problemu pełnią rolę organizmów (osobników), a jakość rozwiązań (bliskość rozwiązania do optimum globalnego) pełni rolę przystosowania organizmów do warunków środowiska. Opracowane zostały takie metody jak algorytm genetyczny (ang. *genetic algorithm*), programowanie ewolucyjne (ang. *evolutionary programming*), strategie ewolucyjne (ang. *evolutionary strategies*) oraz ewolucja różnicowa (ang. *differential evolution*) [17]. Często określa się je zbiorczo nazwą „algorytm ewolucyjny" (ang. *evolutionary algorithm*), gdzie poszczególne metody pełnią rolę wariantów algorytmu. Istotne jest, aby podkreślić, że nazwa „algorytm ewolucyjny" jest nazwą zwyczajową, ponieważ algorytm ewolucyjny jest w rzeczywistości metaheurystyką, a nie algorytmem - nie gwarantuje znalezienia optymalnego rozwiązania analizowanego problemu. Jako metaheurystyka ma zastosowanie do rozwiązywania problemów, które nie są efektywnie algorytmizowane lub dla których algorytm nie istnieje. Wykorzystanie algorytmu ewolucyjnego pozwala na znalezienie rozwiązania zbliżonego do rozwiązania optymalnego w akceptowalnym czasie.

Pseudokod 2.1: Pseudokod algorytmu genetycznego, zaadaptowano z [17, 54].

```
   Data: terminationCondition
   Result: bestIndividual
1  initialise population;
2  evaluate population;
3  while terminationCondition is not satisfied do
4  |  select parents from population;
5  |  perform crossover of parents to obtain children;
6  |  perform mutation on children;
7  |  population ← children;
8  |  evaluate population;
9  end while
10 return the fittest individual from population;
```

Algorytm genetyczny, stanowiący jeden z wariantów algorytmu ewolucyjnego, zaprezentowano w pseudokodzie 2.1. W pierwszej kolejności inicjalizowana i oceniana jest populacja początkowa osobników - najczęściej osobniki inicjalizowane są losowo. Każdy osobnik reprezentuje jedno rozwiązanie zakodowane w postaci genotypu (ang. *genotype*). Cechy osobnika uzyskiwane w wyniku odkodowania genotypu nazywane są fenotypem (ang. *phenotype*). Następnie w każdym kolejnym pokoleniu generowana jest nowa populacja za pomocą operatorów selekcji (ang. *selection*), krzyżowania (ang. *crossover*) i mutacji (ang. *mutation*). Operator

selekcji odpowiedzialny jest za wybór osobników rodzicielskich, które w następnym kroku będą się krzyżować. Każdy osobnik w populacji może być wybrany do krzyżowania z pewnym prawdopodobieństwem zależnym od jego wartości przystosowania (ang. *fitness*) - lepiej przystosowane osobniki mają większe prawdopodobieństwo selekcji. Za pomocą operatora krzyżowania następuje generowanie osobników potomnych w wyniku połączenia osobników rodzicielskich. Krzyżowanie osobników rodzicielskich następuje z pewnym prawdopodobieństwem krzyżowania, określonym parametrem metody. Uzyskane w wyniku krzyżowania osobniki potomne poddawane są operatorowi mutacji - z pewnym prawdopodobieństwem mutacji, również określanym parametrem metody, dokonywane są w nich losowe zmiany. Najczęściej wykorzystywanym warunkiem stopu jest określona liczba pokoleń. Jako wynik działania algorytmu zwracane jest rozwiązanie o najlepszej wartości przystosowania względem analizowanego kryterium.

Pseudokod 2.2: Pseudokod ewolucji różnicowej, zaadaptowano z [38].

```
Data: terminationCondition
Result: bestIndividual
initialise population;
evaluate population;
while terminationCondition is not satisfied do
    perform mutation to obtain mutants;
    perform crossover of individuals and mutants to obtain children;
    evaluate children;
    perform selection between individuals and children to obtain population;
end while
return the fittest individual from population;
```

Jednym z wariantów algorytmu ewolucyjnego, który zastosowano do rozwiązywania problemu MS-RCPSP z ważoną funckją celu jest metoda DEGR (*Differential Evolution + Greedy*) [48], która jest podejściem hybrydowym opartym na ewolucji różnicowej i algorytmie zachłannym. W tym podejściu ewolucja różnicowa operuje na osobnikach, których genotyp określa przypisania zasób-zadanie, a algorytm zachłanny buduje dla każdego osobnika ostateczny harmonogram stanowiący fenotyp osobnika poprzez znalezienie dla każdego zadania czasu rozpoczęcia. Algorytm zachłanny jest algorytmem deterministycznym, podejmującym w każdym kroku lokalnie najlepszą decyzję w nadziei, że doprowadzi to do osiągnięcia optimum globalnego [12]. W ewolucji różnicowej wykorzystywana jest reprezentacja genotypu w postaci wektora liczb rzeczywistych oraz odpowiednio zdefiniowane operatory mutacji, krzyżowania i selekcji. Ewolucja różnicowa różni się od algorytmu genetycznego w szczególności kolejnością zastosowania operatorów i zasadą działania operatorów. Ewolucję różnicową zaprezentowano w pseudokodzie 2.2. Operator mutacji tworzy nowego osobnika-mutanta na podstawie trzech różnych osobników z populacji. W trakcie badań metody DEGR przetestowano trzy operatory mutacji - w każdym z nich dwa z trzech osobników wybierane były losowo, a jako trzeci osobnik wybierano losowego osobnika (mutacja losowa, ang. *random mutation*), najlepszego osobnika (mutacja z najlepszym osobnikiem, ang. *best mutation*) lub aktualnego osobnika (mutacja z aktualnym osobnikiem, ang. *current mutation*). Najlepsze wyniki osiągnięto dla mutacji

losowej z uwagi na większą różnorodność rozwiązań w populacji. Operator krzyżowania służy do wygenerowania osobnika potomnego na podstawie osobnika-mutanta i aktualnie analizowanego osobnika. Najpopularniejsze operatory krzyżowania to krzyżowanie dwumianowe (ang. *binomial crossover*) i wykładnicze (ang. *exponential crossover*) [38, 72] - w metodzie DEGR uzyskano lepsze wyniki dla krzyżowania dwumianowego. W obu metodach krzyżowania osobnik potomny powstaje w wyniku wymieszania genów rodzica i osobnika-mutanta powstałego w wyniku zastosowania operatora mutacji na rodzicu. Po operacji krzyżowania następuje selekcja osobnika, który „przeżywa" do następnego pokolenia - osobnik potomny porównywany jest z aktualnym i wybierany jest osobnik o lepszej wartości przystosowania.

Pionierskimi metodami w dziedzinie optymalizacji wielokryterialnej, opartymi na algorytmie genetycznym i relacji dominacji Pareto, były VEGA (*Vector Evaluation Genetic Algorithm*), NPGA (*Niched Pareto Genetic Algorithm*), SPEA (*Strength Pareto Evolutionary Algorithm*) i NSGA (*Nondominated Sorting Genetic Algorithm*) [70]. Metoda VEGA [77] wykorzystuje podział populacji na podpopulacje w taki sposób, że każda podpopulacja odpowiada jednemu kryterium. Osobniki rodzicielskie wybierane są z prawdopodobieństwem proporcjonalnym do ich przystosowania biorąc pod uwagę kryterium przypisane do danej podpopulacji. Na wybranych osobnikach rodzicielskich dokonywane są następnie operatory krzyżowania i mutacji w celu utworzenia populacji potomnej, która przechodzi do następnej iteracji.

W metodzie NPGA [25] wykorzystano nowy operator selekcji - turniej oparty na dominacji Pareto (ang. *Pareto-dominance based tournament*). W turnieju opartym na dominacji Pareto biorą udział dwa losowe osobniki z populacji i oceniane są one wykorzystując zbiór losowych osobników z populacji zwany zbiorem porównawczym (ang. *comparison set*). Jeżeli jeden z wybranych osobników jest dominowany przez zbiór porównawczy, a drugi nie, do krzyżowania wybierany jest niezdominowany osobnik. Jeżeli oba osobniki są jednocześnie dominowane lub niezdominowane przez zbiór porównawczy porównywane są one pod względem współdzielonej wartości fitness (ang. *shared fitness*), która bierze pod uwagę liczbę nisz (ang. *niche count*), do których należą osobniki, w celu promowania nieeksplorowanych obszarów przestrzeni. Rozmiar niszy (ang. *niche radius*) określa parametr, którego wartość musi być strojona dla każdego rozwiązywanego problemu i wartości pozostałych parametrów. Uzupełnieniem metody NPGA jest NPGA2 [18], w której zamiast zbioru porównawczego wykorzystano rangę dominacji (ang. *domination rank*) [19]. Definicję rangi dominacji przestawiono w równaniu 2.1, gdzie x oznacza aktualnie analizowanego osobnika, a p liczbę dominujących go osobników. Niezdominowane osobniki mają rangę jeden, a im wyższa jest ranga osobnika, tym więcej osobników go dominuje. Selekcja turniejowa dokonywana jest biorąc pod uwagę rangę osobników - wygrywa osobnik o niższej randze, a jeżeli osobniki posiadają taką samą rangę wykorzystywana jest współdzielona wartość fitness.

$$rank(x) = 1 + p \tag{2.1}$$

Metoda SPEA [77] wykorzystuje archiwum niezdominowanych rozwiązań i ocenia osobniki z populacji tylko biorąc pod uwagę to, czy dominują one osobniki przechowywane w archiwum. Każde rozwiązanie z archiwum posiada siłę (ang. *strength*), która jest proporcjonalna do liczby osobników z populacji, które są dominowane przez to rozwiązanie - siła rozwiązania

z archiwum jest jednocześnie jego wartością fitness. Fitness osobnika jest liczony jako suma siły wszystkich rozwiązań z archiwum, które go dominują. Do uzyskanej wartości dodawana jest wartość jeden aby osobniki z populacji miały gorszą wartość przystosowania niż osobniki przechowywane w archiwum. Operatory selekcji, krzyżowania i mutacji działają na sumie populacji i archiwum. Jeżeli liczba rozwiązań w archiwum miałaby być większa od zadanego rozmiaru archiwum dokonywane jest grupowanie (ang. *clustering*) aby zmniejszyć rozmiar archiwum. Opracowana została także ulepszona wersja, SPEA2 [76], w której i dla osobników z populacji, i dla osobników przechowywanych w archiwum wyznaczana jest siła osobnika. Fitness osobnika zależy od sumy siły osobników, które go dominują oraz od gęstości (ang. *density*) rozwiązań. Aby wprowadzić rozróżnienie między osobnikami o takich samych surowych wartościach fitness (równych sumie siły rozwiązań, przez które są dominowane) analizowana jest odległość osobników od k-tego najbliższego sąsiada.

Metoda NSGA [62] działa bardzo podobnie do metody NPGA2, z tym że osobnikom zamiast rangi przypisywana jest zastępcza wartość fitness (ang. *dummy fitness value*) w taki sposób, że osobniki o niższej randze mają lepszą wartość fitness. Aby zachować różnorodność populacji wykorzystywana jest współdzielona wartość fitness. Proces nadawania osobnikom zastępczej wartości fitness poprzez ustalenie, do którego frontu one należą (które osobniki je dominują i które osobniki są przez nie dominowane), nazywany jest sortowaniem niezdominowanym (ang. *nondominated sorting*). NSGA posiada trzy zasadnicze wady: dużą złożoność obliczeniową sortowania niezdominowanego, brak elityzmu (zabezpieczania dobrych osobników przez usunięciem) i konieczność określenia rozmiaru niszy. Te wady wyeliminowano w kolejnej wersji algorytmu, czyli NSGA-II (*Elitist Nondominated Sorting Genetic Algorithm*) [15]. NSGA-II wykorzystuje szybką metodę sortowania niezdominowanego opartą na analizie rangi dominacji osobnika i zbioru osobników przez niego dominowanych. Zamiast liczby nisz do zapewnienia różnorodności populacji wykorzystywana jest odległość zatłoczenia (ang. *crowding distance*) - maksymalna objętość prostopadłościanu, w którym leży tylko analizowane rozwiązanie i żadne inne. Ranga dominacji osobnika i jego odległość zatłoczenia brane są pod uwagę w trakcie porównywania ze sobą osobników operatorem porównania biorącym pod uwagę zatłoczenie (ang. *crowded comparison operator*). Definicję operatora przedstawiono w równaniu 2.2, gdzie i oraz j to porównywane rozwiązania, i_{rank} oraz j_{rank} to ranga dominacji rozwiązań, a $i_{distance}$ oraz $j_{distance}$ to odległość zatłoczenia rozwiązań.

$$i \geqslant_n j \text{ if } (i_{rank} < j_{rank}) \ \vee \ ((i_{rank} = j_{rank}) \wedge (i_{distance} > j_{distance})) \tag{2.2}$$

Metoda NSGA-II została zaprojektowana dla optymalizacji dwóch kryteriów jednocześnie. Wersją metody NSGA-II przystosowaną do optymalizacji wielokryterialnej jest U-NSGA-III (*Unified NSGA-III*) [58], która działa dobrze dla trzech lub więcej kryteriów. U-NSGA-III wykorzystuje do wspierania ewolucji zbiór szeroko rozproszonych punktów odniesienia (ang. *reference points*). Metoda różni się od NSGA-II wyborem rozwiązań tworzących populację w następnym pokoleniu. Aby uzyskać populację do następnej iteracji w pierwszej kolejności dodawane są do niej kolejne fronty uzyskane w wyniku sortowania niezdominowanego. Jeżeli dodanie kolejnego frontu przekroczyłoby rozmiar populacji, wykorzystywana jest technika niszowania (ang. *niching*) aby wybrać część osobników z tego frontu. Następuje normalizacja wartości wszystkich kryteriów zgodnie z wartościami występującymi w populacji, analizowanym

froncie i zbiorze punktów odniesienia. W kolejnym kroku każdy osobnik z analizowanego frontu podwiązywany jest z najbliższym mu punktem odniesienia, a do populacji dodawane są osobniki powiązane z punktami referencyjnymi, którym przypisano najmniej osobników.

Obiecujące wyniki uzyskano stosując do rozwiązywania wielokryterialnego problemu MS-RCPSP metodę NTGA (*Nondominated Tournament Genetic Algorithm*) [34, 46] i jej drugą wersję, NTGA2 [43]. NTGA jest oparta na metodzie NSGA-II i wprowadza następujące modyfikacje: populacja rodzicielska nie jest łączona z populacją potomną, jako operator selekcji wykorzystywana jest selekcja turniejowa wykorzystująca operator porównania biorący pod uwagę tylko rangę (ang. *rank comparison operator*) oraz wykorzystywany jest mechanizm zapobiegania występowaniu klonów (ang. *clone prevention mechanism*). Operator porównania biorący pod uwagę tylko rangę wykorzystywany w selekcji turniejowej zdefiniowano w równaniu 2.3, gdzie i oraz j to porównywane rozwiązania, a i_{rank} oraz j_{rank} to ranga dominacji rozwiązań. W przeciwieństwie do NSGA-II nie jest brana pod uwagę odległość zatłoczenia osobników. Inaczej niż w metodzie NPGA, selekcja turniejowa polega na losowym wyborze pewnej liczby (określonej parametrem metody) osobników z populacji, które rywalizują ze sobą w „turnieju", który wygrywa (czyli wybierany jest do krzyżowania) tylko najlepszy z nich. Z uwagi na duże ciśnienie selekcyjne i związane z nim zmniejszenie różnorodności populacji, metoda NTGA wykorzystuje mechanizm zapobiegania występowaniu klonów. Osobnik jest klonem innego osobnika, jeżeli wszystkie ich geny mają takie same wartości. Aby zapobiec występowaniu klonów po wygenerowaniu każdego osobnika potomnego sprawdza się, czy nie jest on klonem innego osobnika z populacji. Jeżeli jest, na klonie kilkukrotnie wykonywany jest operator mutacji w celu osiągnięcia unikatowego osobnika. Jeżeli uzyskany zostanie unikatowy osobnik, proces mutacji jest przerywany. Jeżeli po kilku mutacjach dokonanych na osobniku wciąż stanowi on klona innego osobnika, jest on zastępowany przez losowego osobnika.

$$i \geqslant_r j \text{ if } (i_{rank} < j_{rank}) \tag{2.3}$$

Druga wersja, NTGA2, wprowadza operator selekcji GAP opartej na lukach (ang. *gap selection*), w którym aktywny udział bierze archiwum. Archiwum służy do przechowywania globalnie niezdominowanych osobników i jest jednocześnie wynikiem metody oraz przybliżeniem prawdziwego zbioru Pareto. Metodę NTGA2 zaprezentowano w pseudokodzie 2.3. W pierwszej kolejności następuje inicjalizacja i ocena populacji początkowej oraz zapisanie w archiwum niezdominowanych osobników z populacji początkowej. W kolejnych iteracjach za pomocą operatorów genetycznych budowana jest populacja potomna, wykorzystując mechanizm zapobiegania występowaniu klonów zaproponowany w metodzie NTGA w celu zapewnienia różnorodności. Metoda NTGA2 wykorzystuje naprzemiennie dwa operatory selekcji - selekcję turniejową wykorzystującą operator porównania rangi oraz selekcję GAP. Liczbę pokoleń, w których stosowany jest każdy z operatorów selekcji określa wartość parametru. Selekcja turniejowa wykorzystująca operator porównania rangi zaproponowana została w metodzie NTGA i polega na wyborze z losowej grupy osobników osobnika o najmniejszej randze dominacji. Operator selekcji GAP, przedstawiony w pseudokodzie 2.4, ma na celu promowanie tych osobników, które leżą blisko największych „luk" w przybliżeniu prawdziwego frontu Pareto. Aktualne przybliżenie prawdziwego frontu Pareto jest przechowywane w archiwum i „luki" w nim oznaczają istnienie nieeksplorowanych obszarów przestrzeni. Operator selekcji GAP wykorzystuje więc sumę aktualnej populacji i archiwum. W pierwszej kolejności metodą

Pseudokod 2.3: Pseudokod metody NTGA2, zaadaptowano z [43].

Data: *generationLimit*
gapSelection - number of generations created using each selection operator
Result: *archive* - approximation of Pareto set

```
archive ← empty;
initialise population;
evaluate population;
update archive with population;
i ← 0;
while i ≠ generationLimit do
    nextPopulation ← empty;
    while size(nextPopulation) < size(population) do
        if i % (2 * gapSelection) < gapSelection then
            perform tournament selection on population to obtain parents;
        else
            perform gap selection on population ∪ archive to obtain parents;
        end if
        perform crossover and mutation on parents to obtain children;
        while nextPopulation contains children do
            perform mutation on children;
        end while
        evaluate children;
        nextPopulation ← nextPopulation ∪ children;
        update archive with children;
    end while
    population ← nextPopulation;
    i ← i + 1;
end while
return archive;
```

Pseudokod 2.4: Pseudokod operatora selekcji GAP (*GapSelection*), zaadaptowano z [43].

Data: *population*
archive - current approximation of Pareto set
Result: *parents* - two parent individuals chosen for crossover

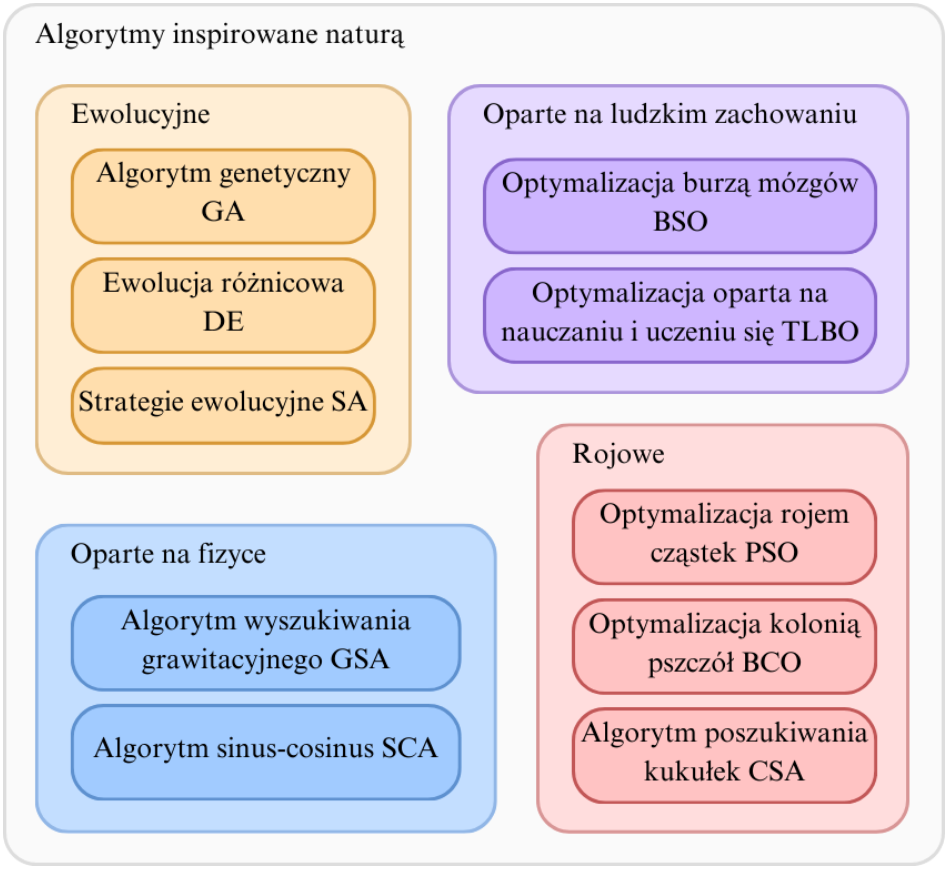
```
perform tournament selection considering gap sizes on population ∪ archive to obtain
  firstParent;
secondParent ← random neighbour of firstParent;
if secondParent is null then
    perform tournament selection considering gap sizes on population ∪ archive to
      obtain secondParent;
return firstParent ∪ secondParent;
```

selekcji turniejowej wykorzystującej rozmiar luk do oceny osobników wybierany jest pierwszy rodzic. Rozmiar luki dla pojedynczego osobnika obliczany jest biorąc pod uwagę tylko jedno kryterium - populacja podzielona jest na liczbę części równą liczbie kryteriów i w każdej części analizowane są tylko luki w danym kryterium. Rozmiar luki jest minimalną odległością euklidesową do dalszego z dwóch sąsiadów osobnika. Sąsiadami osobnika są dwaj najbliżsi osobnicy - jeden o gorszej wartości analizowanego kryterium i jeden o lepszej jego wartości. Jeżeli osobnik ma tylko jednego sąsiada (leży na „krawędzi" przybliżenia prawdziwego frontu Pareto) za rozmiar luki przyjmowana jest nieskończoność. W selekcji turniejowej wykorzystującej rozmiar luk do oceny osobników preferowane są osobniki o większej wartości rozmiaru luki. Jako drugi rodzic w operatorze selekcji GAP wybierany jest losowy sąsiad pierwszego rodzica. Dla osobników leżących na „krawędzi" przybliżenia prawdziwego frontu Pareto możliwe jest, że drugi rodzic nie zostanie wybrany - w takim przypadku drugi rodzic wybierany jest analogicznie do pierwszego rodzica.

2.3. Podejścia koewolucyjne

Algorytm ewolucyjny jest podejściem inspirowanym naturą i poza nim istnieje oczywiście wiele innych podejść inspirowanych innymi zjawiskami naturalnymi [27, 51]. Optymalizacja rojem cząstek (ang. *Particle Swarm Optimisation*, PSO), optymalizacja kolonią pszczół (ang. *Bee Colony Optimisation*, BCO) oraz algorytm poszukiwania kukułek (ang. *Cuckoo Search Algorithm*, CSA) reprezentują algorytmy inspirowane zachowaniami roju (stada). Podejścia takie jak optymalizacja burzą mózgów (ang. *Brain Storm Optimisation*, BSO) oraz optymalizacja oparta na nauczaniu i uczeniu się (ang. *Teaching-Learning Based Optimisation*, TLBO) są reprezentantami algorytmów inspirowanych ludzkim zachowaniem. Algorytm wyszukiwania grawitacyjnego (ang. *Gravitational Search Algorithm*, GSA) i algorytm sinus-cosinus (ang. *Sine Cosine Algorithm*, SCA) reprezentują algorytmy inspirowane zjawiskami fizycznymi. Podział algorytmów inspirowanych naturą przedstawiono na rysunku 2.1.

Inspiracje naturą są więc bardzo powszechne w metodach inteligencji obliczeniowej, a ponieważ zjawisko koewolucji jest ściśle związane z ewolucją, naturalnym jest, że przy rozwoju algorytmu ewolucyjnego pojawiła się inspiracja nim. Koewolucja w biologii występuje wtedy, gdy dwa lub więcej gatunków ewoluują w taki sposób, że ich interakcje międzygatunkowe wpływają na adaptację tych organizmów do aktualnych warunków środowiska. Innymi słowy, występuje pewne sprzężenie zwrotne i gatunki ewoluują, wpływając na siebie nawzajem. Między koewoluującymi gatunkami mogą występować różne rodzaje relacji. Przykładem koewolucji, w której oba gatunki osiągają pewną korzyść, jest relacja zapylaczy i zapylanych przez nich roślin - owady przenoszą pyłek, umożliwiając roślinom zapylenie, a w „nagrodę za ich pracę" uzyskują nektar lub nasiona, którymi się odżywiają. Koewoluujące gatunki mogą też konkurować lub walczyć ze sobą. Na przykład w relacji drapieżnik-ofiara (ang. *predator-prey*) między muchą a żabą: u żaby wykształcił się długi i klejący język, a u muchy organy odpowiadające za wykrywanie zagrożenia i techniki unikania go. Ofiara ewoluuje w taki sposób, żeby lepiej uciekać, a drapieżnik tak, by lepiej łapać ofiarę. Podstawą koewolucji jest więc interakcja między różnymi gatunkami, przy czym gatunki te się nie krzyżują, tylko wpływają na siebie nawzajem w kontekście przystosowania do aktualnych warunków środowiska.



Rysunek 2.1: Podział algorytmów inspirowanych naturą z przykładami algorytmów należących do każdej kategorii, zaadaptowano z [27, 51].

Wprowadzając inspirację koewolucją do algorytmów ewolucyjnych można uzyskać podejście koewolucyjne, w którym zamiast jednej populacji ewoluuje kilka osobnych populacji (gatunków), które wchodzą ze sobą w interakcje poprzez powiązane funkcje przystosowania [14]. Te populacje mogą ewoluować przy pomocy różnych algorytmów o różnych parametrach, ich ewolucja może być równoległa lub synchronizowana na różne sposoby, a także sam sposób powiązania funkcji przystosowania może być różny. Podstawową cechą podejść koewolucyjnych jest istnienie kilku powiązanych populacji, czyli reprezentanci jednej populacji wchodzą w interakcje z reprezentantami innych populacji. Interakcje między reprezentantami różnych populacji mogą mieć charakter kooperacyjny (ang. *cooperative*) lub konkurencyjny (ang. *competitive*). W koewolucji kooperacyjnej istnieje element współpracy między gatunkami - relacja „razem wygrywamy, razem przegrywamy", zaś w koewolucji konkurencyjnej gatunki walczą ze sobą na zasadzie „ja wygrywam, ty przegrywasz". Biologicznym przykładem koewolucji kooperacyjnej jest wspomniana wcześniej relacja zapylaczy i zapylanych przez nie roślin, a koewolucji konkurencyjnej relacja drapieżnik-ofiara. Konkurencyjne modele koewolucyjne sprawdzają się szczególnie dobrze dla problemów, w których trudno jest wprost zdefiniować obiektywną funkcję przystosowania, na przykład w określaniu strategii gry przez „sztucznego" gracza wykorzystującego metody sztucznej inteligencji. Kooperacyjne modele koewolucyjne są zaś bardziej odpowiednie dla problemów, w których między możliwymi do wyróżnienia

podproblemami występują nieliniowe zależności, przez które wchodzące w interakcje elementy są od siebie w dużym stopniu zależne [31]. Przykłady kooperacyjnych podejść koewolucyjnych zaprezentowano w podrozdziale 2.3.1, a konkurencyjnych podejść koewolucyjnych w podrozdziale 2.3.2. Oprócz podejść, które można zaklasyfikować do jednej z tych odmian koewolucji, istnieją również podejścia, które łączą w sobie cechy kooperacyjne i konkurencyjne. Przykłady takich podejść omówiono w podrozdziale 2.3.3.

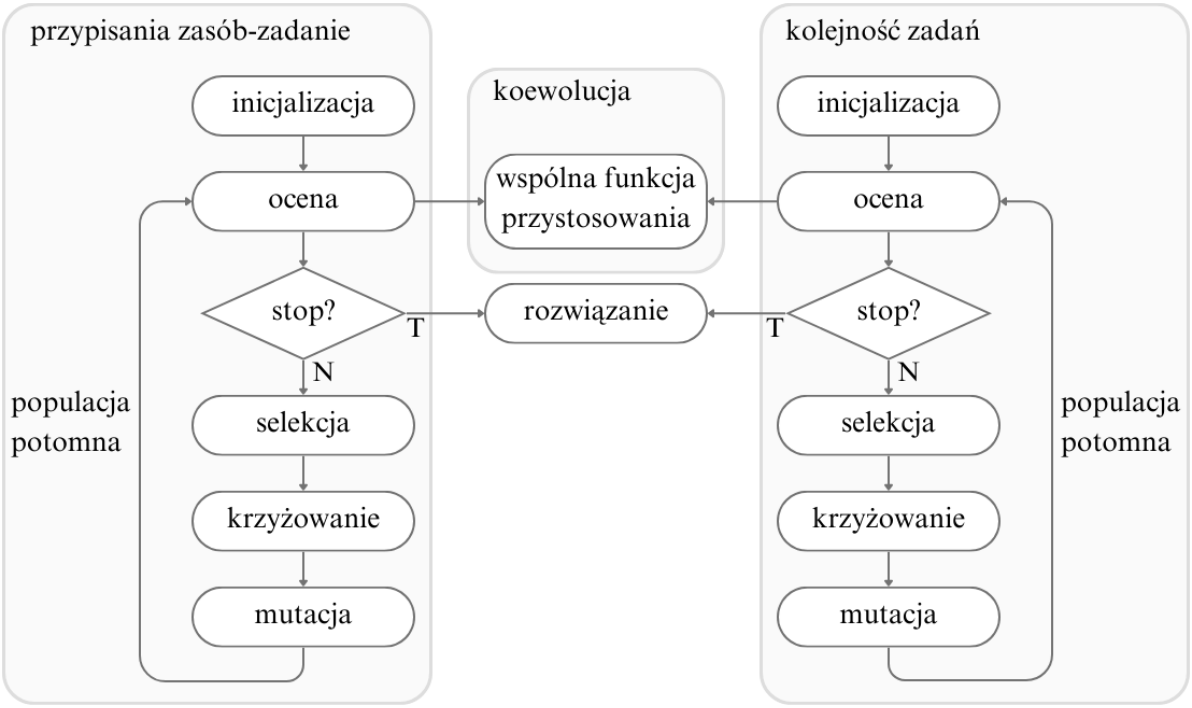
2.3.1. Koewolucja kooperacyjna

Tradycyjne metody optymalizacji, w tym algorytm ewolucyjny, działają wydajnie i skutecznie dla problemów z przestrzeniami przeszukiwania o małej liczbie wymiarów, ale zwykle nie skalują się dobrze na problemy o większej liczbie wymiarów. Jest to powszechnie znane jako „przekleństwo wymiarowości" (ang. *curse of dimensionality*) [20]. Z tego powodu często wykorzystywana jest dekompozycja problemu na podproblemy, która pozwala zmniejszyć liczbę wymiarów. Naturalnym rozwiązaniem „przekleństwa wymiarowości" jest paradygmat dziel i rządź (ang. *divide and conquer*), który polega na dekompozycji problemu na podproblemy, rozwiązaniu podproblemów, a następnie połączeniu uzyskanych rozwiązań w celu uzyskania rozwiązania pierwotnego problemu. Najczęściej stosowanym podejściem jest koewolucja kooperacyjna oparta na tym paradygmacie. Pierwotny problem dzielony jest na podproblemy, które następnie rozwiązywane są wspólnie.

Niektóre problemy mogą być w naturalny sposób podzielone na podproblemy - na przykład problem mobilnego złodzieja (ang. *Travelling Thief Problem*, TTP) [10], elastyczny problem sklepu z zadaniami (ang. *Flexible Job Shop Problem*, fJSP) [52] i jego wersja rozmyta (ang. *Fuzzy Flexible Job Shop Problem*, FfJSP) [52] oraz problem MS-RCPSP. Problem TTP jest połączeniem dwóch innych problemów: problemu komiwojażera (ang. *Travelling Salesman Problem*, TSP) oraz problemu plecakowego (ang. *Knapsack Problem*, KNP). Jako połączenie tych dwóch problemów polega na tym, że złodziej odwiedza w pewnej kolejności pewną liczbę miast i w każdym z miast decyduje, które przedmioty z danego miasta chce włożyć do swojego plecaka. Złodziej musi każde miasto odwiedzić dokładnie raz oraz jego trasa kończy się w tym samym mieście, w którym się zaczęła. Przedmioty mają określoną wagę, a plecak złodzieja posiada ograniczoną pojemność, więc niemożliwe jest zabranie wszystkich przedmiotów. Oprócz tego, im cięższy jest plecak złodzieja, tym dłużej zajmuje mu droga z miasta do miasta. Celem złodzieja jest zdobycie przedmiotów o jak największej wartości w jak najkrótszym czasie. Problem fJSP jest dosyć podobny do problemu MS-RCPSP, z tym że zadania zamiast przez zasoby ludzkie wykonywane są przez fizyczne maszyny. Występują ograniczenia pierwszeństwa i przepustowości dla zadań - ograniczenia pierwszeństwa dotyczą poprzedników zadań, a zgodnie z ograniczeniami przepustowości każde zadanie wymaga nieprzerwanego i wyłącznego korzystania z maszyny należącej do pewnego podzbioru maszyn. Oznacza to, że czas przetwarzania zadania zależy od maszyny, na której to zadanie jest przetwarzane. Rozwiązaniem problemu jest harmonogram, w którym każde zadanie ma przydzielony czas rozpoczęcia oraz wszystkie zadania są przypisane do wykonujących je maszyn i oczywiście wszystkie ograniczenia są spełnione. Harmonogram może być oceniany pod względem różnych kryteriów. W rozmytej wersji problemu, czyli w problemie FfJSP, liczby rozmyte są wykorzystywane do określenia czasów wykonywania zadań, które mogą być nie być

znane z góry lub trudne do określenia. Problem MS-RCPSP opisano szczegółowo w rozdziale 1. Dla problemu TTP w sposób naturalny nasuwa się model koewolucji kooperacyjnej, w którym jedna populacja (gatunek) odpowiada za zaproponowanie kolejności odwiedzanych miast, a druga za zaproponowanie przedmiotów schowanych do plecaka.

Zastosowana do rozwiązania problemu FfJSP metoda CELS (*Cooperative Coevolutionary Algorithm Hybridised with Local Search*) [52] łączy w sobie koewolucję kooperacyjną i lokalne poszukiwanie oraz wprowadza podział na dwie populacje, w której jedna populacja proponuje przypisania maszyna-zadanie, a druga kolejność wykonywania zadań. Interakcja między populacjami polega na tym, że dla każdego osobnika reprezentującego rozwiązanie częściowe pamiętani są trzej partnerzy do współpracy, którzy pochodzą z drugiej populacji (ang. *collaboration partners*). Rolę partnerów do współpracy pełnią losowy osobnik, najlepszy osobnik z poprzedniej iteracji i osobnik przechowywany na tym samym indeksie (dla przyjętego arbitralnie sortowania). W celu uzyskania rozwiązania problemu pierwotnego osobnik łączony jest z każdym partnerem do współpracy i wybierany jest taki partner, dla którego uzyskano najlepsze przystosowanie.



Rysunek 2.2: Poglądowy schemat działania kooperacyjnego modelu koewolucyjnego zastosowanego do rozwiązania problemu MS-RCPSP, zaadaptowano z [45].

Dla problemu MS-RCPSP zastosowano bardzo podobny podział jak w metodzie CELS - jedna populacja proponuje przypisania zasób-zadanie, a druga kolejność wykonywania zadań [45]. Schemat zaproponowanej do rozwiązywania problemu MS-RCPSP metody CoEA (*Coevolutionary Algorithm*) zaprezentowano na rysunku 2.2. W CoEA do oceny osobnika z jednej populacji wybierana jest pewna liczba osobników z drugiej populacji określona przez parametr metody. Dodatkowo, przechowywany jest osobnik z drugiej populacji, dla którego w poprzedniej iteracji uzyskano najlepsze przystosowanie. Tak jak w metodzie CELS, analizo-

wany osobnik łączony jest z każdym wybranym reprezentantem drugiej populacji i wybierany jest ten partner, dla którego uzyskano najlepsze przystosowanie. Jest on zapamiętywany do następnej iteracji. Do wygenerowania harmonogramu na podstawie rozwiązań częściowych wykorzystywany jest algorytm zachłanny.

Metody CELS i CoEA dokonują optymalizacji jednokryterialnej. Dla metod takich jak CELS i CoEA każda populacja generuje częściowe rozwiązanie problemu, a rozwiązanie problemu pierwotnego jest uzyskiwane dopiero w wyniku połączenia rozwiązań częściowych. W celu połączenia rozwiązań częściowych wykorzystywana jest wspólna funkcja przystosowania, łącząca wszystkie populacje - osobniki z jednej populacji nie są oceniane niezależnie. Między populacjami występuje interakcja w postaci relacji współpracy w celu uzyskania rozwiązania pierwotnego problemu, dzięki czemu te metody można nazwać metodami koewolucji kooperacyjnej.

Bardzo podobną metodę zastosowano do rozwiązywania wielokryterialnego problemu produkcji *seru* (ang. *seru production*) - MOCCMS-NSGA-II/III (*Multi-objective Cooperative Coevolution Algorithm with a Master–Slave Mechanism based on NSGA-II/III*) [36]. *Seru* to japońska nazwa na nazwę komórki linii produkcyjnej, czyli wydzielonego z całej linii produkcyjnej pewnego mniejszego fragmentu [60]. Problem produkcji *seru* składa się z dwóch problemów NP-trudnych, czyli problemu utworzenia *seru* (ang. *seru formation*) i problemu planowania *seru* (ang. *seru scheduling*). Problem utworzenia *seru* polega na zadecydowaniu, ile komórek powinno zostać utworzone i którzy pracownicy powinni być przypisani do której komórki. Problem planowania *seru* skupia się zaś na tym, które serie produkcyjne powinny być wytwarzane przez które komórki. Dla tego problemu naturalnym kooperacyjnym modelem koewolucyjnym jest model, w którym jeden gatunek odpowiada za rozwiązywanie problemu utworzenia *seru*, a drugi za rozwiązywanie problemu planowania *seru*. Dodatkową modyfikacją wprowadzoną przez metodę MOCCMS-NSGA-II/III jest wykorzystanie mechanizmu *„master-slave”*. Metoda wykorzystuje po trzy populacje dla każdego gatunku (trzy populacje rozwiązują problem utworzenia *seru* i trzy populacje rozwiązują problem planowania *seru*). Dla każdego gatunku wykorzystywana jest populacja *master*, która przechowuje niezdominowane rozwiązania problemu rozwiązywanego przez dany gatunek. Osobniki populacji *master* są wykorzystywane w momencie krzyżowania - zawsze w celu utworzenia osobnika potomnego wykorzystywany jest jeden rodzic z populacji typu *slave* i drugi rodzic z populacji typu *master*. Zgodnie z ideą koewolucji kooperacyjnej do oceny reprezentanta jednego gatunku wymagana jest jego współpraca z reprezentantem drugiego gatunku - wykorzystywane są niezdominowane rozwiązania z drugiego gatunku.

Klasyczną metodą optymalizacji wielokryterialnej opartą na koewolucji kooperacyjnej jest metoda CCEA (*Cooperative Coevolutionary Algorithm*), znana również pod nazwą MOCCA (*Multi-objective Cooperative Coevolutionary Algorithm*) [65]. W metodzie CCEA dla każdej zmiennej decyzyjnej tworzona jest oddzielna populacja i osobniki z różnych populacji wchodzą ze sobą w interakcje w celu zbudowania kompletnego rozwiązania. Każda podpopulacja optymalizuje jedną zmienną decyzyjną, a jej reprezentantem w celu współpracy z innymi podpopulacjami jest jej najlepszy osobnik. Ponieważ CCEA jest metodą optymalizacji wielokryterialnej, zapamiętywane jest archiwum niezdominowanych osobników, które stanowi również ostateczny rezultat działania metody. W przypadku znalezienia nowego niezdomino-

wanego osobnika jest on dodawany do archiwum, a wszystkie osobniki, które są przez niego dominowane, są z archiwum usuwane. Jeżeli archiwum jest pełne, dla każdego z jego elementów obliczana jest liczba nisz, do których element należy - jest ona tym większa, im więcej innych elementów archiwum znajduje się blisko analizowanego elementu. Nisza (ang. *niche*) oznacza pewien mały obszar przestrzeni, do którego należą sąsiadujące osobniki - rozmiar sąsiedztwa określa parametr metody. Osobnik o najmniejszej liczbie nisz, do których należy, jest następnie klonowany do podpopulacji, zastępując losowe osobniki. Dzięki temu algorytm skupia się na jeszcze nieodkrytych fragmentach przestrzeni. Liczba nisz jest też wykorzystywana w trakcie selekcji turniejowej. W przypadku, gdy dwa osobniki mają taką samą rangę dominacji (ranga dominacji reprezentuje liczbę dominujących dane rozwiązanie innych rozwiązań) wybierany jest ten osobnik, który należy do mniejszej liczby nisz. Modyfikacją metody CCEA jest metoda MOCCA-II [73], która wykorzystuje inny mechanizm oceny osobników i inny mechanizm wyznaczania niszy. W trakcie interakcji między podpopulacjami każda podpopulacja posiada dwóch reprezentantów - swojego najlepszego osobnika i pewnego losowego osobnika. Po zbudowaniu kompletnych rozwiązań dla najlepszego i losowego reprezentanta podpopulacji do dalszej analizy wybierane jest rozwiązanie zbudowane z najlepszego reprezentanta podpopulacji, chyba że jest ono zdominowane przez drugie rozwiązanie zbudowane z losowego reprezentanta podpopulacji. Zamiast liczby nisz wykorzystywana jest odległość zatłoczenia zaproponowana po raz pierwszy w metodzie NSGA-II. Preferowane są osobniki, których odległość zatłoczenia jest duża, czyli które znajdują się daleko od innych osobników.

Istnieje również wiele problemów, dla których podział na podproblemy nie jest oczywisty. Oprócz poszukiwania optymalnego rozwiązania (lub rozwiązania możliwie bliskiego optymalnemu) należy także znaleźć sposób dekompozycji o najlepszej jakości, ponieważ jakość dekompozycji określa stopień uproszczenia oryginalnego problemu. Dla dobrej dekompozycji zmienne zgrupowane jako jeden podproblem (komponent) wchodzą ze sobą w dużą liczbę interakcji, a jednocześnie zależności między podproblemami (komponentami) są małe [59]. Przykładem metody wykorzystującej grupowanie zmiennych decyzyjnych w oparciu o analizę interakcji jest CCMOGA-ICRA (*Cooperative Coevolutionary Genetic Algorithm with Improved Computational Resource Allocation*) [59]. W tej metodzie zmienne decyzyjne grupowane są w komponenty w oparciu o analizę interakcji między nimi. W każdej iteracji analizowany jest wkład komponentu do poprawy zbieżności frontu Pareto. W przypadku, gdy numer cyklu koewolucyjnego (iteracji) jest różny od wielokrotności liczby komponentów, ewoluuje tylko komponent, którego wkład do poprawy zbieżności frontu Pareto jest największy. W przeciwnym wypadku, ewoluują wszystkie komponenty. Innym podejściem wykorzystującym analizę wkładu do ewolucji jest BICCA (*Bi-space Interactive Cooperative Coevolutionary Algorithm*) [20]. BICCA analizuje przestrzeń wzorców (ang. *pattern space*) i przestrzeń poszukiwań (ang. *search space*). Wzorzec reprezentuje zachowanie współdziałających zmiennych i składa się z grupy zmiennych oraz czterech atrybutów opisujących charakterystykę wzorca. Atrybuty opisujące charakterystykę wzorca to: liczba udanych optymalizacji, liczba nieudanych optymalizacji, aktywność (ang. *activity*) i zwartość (ang. *compactness*). Jeżeli wzorzec jest aktywny (jego aktywność jest wysoka), jego zmienne są bardziej podatne na interakcje i jest bardziej prawdopodobne, że grupa zmiennych wniesie wkład do ewolucji. Zwartość opisuje połączenia między zmiennymi wewnątrz grupy. BICCA wykonuje ewolucję wzorców (ang. *pattern evolution*) i na podstawie znalezionych grup zmiennych decyzyjnych tworzone są komponenty, które

ewoluują współpracując ze sobą poprzez udostępnianie sobie nawzajem swojego reprezentanta (którego rolę pełni najlepszy osobnik). Metody takie jak CCMOGA-ICRA i BICCA mają zastosowanie dla problemów (częściowo) nieseparowalnych lub takich, dla których nie narzuca się naturalny sposób dekompozycji problemu na podproblemy (komponenty).

Całkowicie inne podejście do koewolucji kooperacyjnej reprezentuje metoda VPEGA (*Virus Coevolutionary Partheno-genetic Algorithm*) [29]. VPEGA jest partenogenetyczną odmianą algorytmu genetycznego, co oznacza, że reprodukcja dokonywana jest z udziałem tylko jednego osobnika. W biologii partenogeneza jest jednym z rodzajów bezpłciowego rozmnażania - zarodek rozwija się z niezapłodnionej komórki jajowej, zatem reprodukcja zachodzi bez udziału samca. Algorytm partenogenetyczny wykorzystuje trzy operatory: *swap* (pl. zamiana), *reverse* (pl. odwrócenie) oraz *insert* (pl. wstawienie). Operator *swap* powoduje zamianę pozycjami dla wartości dwóch losowych genów. W wyniku działania operatora *reverse* kolejność wartości wszystkich genów znajdujących się między dwoma losowymi pozycjami jest odwracana. Operator *insert* powoduje wstawienie wartości genu znajdującego się na jednej pozycji na inną pozycję i przesunięcie wartości wszystkich genów pomiędzy tymi pozycjami o jedną pozycję w prawo. Metoda VPEGA wykorzystuje operatory partenogenetyczne *swap*, *reverse* oraz *insert* podczas ewolucji populacji rozwiązań. Drugą populacją jest populacja wirusów (schematów), która posiada własne operatory: skopiowanie wirusa (ang. *virus copy*), przepisanie wirusa (ang. *virus transcription*) i skrócenie wirusa (ang. *virus cut*). Wirusy reprezentowane są za pomocą ciągów zer, jedynek i znaków *. W wyniku skopiowania wirusa z danego rozwiązania powstaje nowy wirus. Za pomocą przepisania wirusa populacja wirusów wpływa na populację rozwiązań - wirus „infekuje” rozwiązanie. „Infekcja” rozwiązania przez wirus polega na przepisaniu fragmentu chromosomu wirusa do rozwiązania, pomijając wszystkie znaki *. Operator skrócenia wirusa powoduje skrócenie jego chromosomu i wykorzystywany jest dla wirusów o słabej wartości przystosowania. Populacja wirusów powstaje z populacji rozwiązań w wyniku kopiowania fragmentów chromosomów rozwiązań, następnie każdy wirus przez swój czas życia może infekować rozwiązania - liczba infekcji jest ograniczona wskaźnikiem infekcji wirusów (ang. *virus infection rate*). Po „śmierci” wirusów operatorem kopiowania generowany jest nowy wirus. Metoda VPEGA operuje więc na dwóch zupełnie innych populacjach, z których każda wykorzystuje własną reprezentację osobnika, własną funkcję fitness i własne operatory. Pomimo wykorzystania nazwy „wirus”, nie budzącej pozytywnych skojarzeń, między wirusami a rozwiązaniami występuje relacja współpracy, ponieważ mają one ten sam ogólny cel - poprawę jakości rozwiązań. Przystosowanie wirusów oceniane jest jako sumaryczna wartość poprawy wszystkich zainfekowanych rozwiązań. Wirusy i rozwiązania współpracują więc ze sobą w celu wypracowania najlepszego rozwiązania.

2.3.2. Koewolucja konkurencyjna

Podczas gdy w kooperacyjnych modelach koewolucyjnych pomiędzy populacjami występuje relacja współpracy, w konkurencyjnych modelach koewolucyjnych kilka populacji konkuruje między sobą. Dla wielu problemów, na przykład wymienionych w podrozdziale 2.3.1, łatwo można zidentyfikować komponenty, które mogą ze sobą współpracować w celu uzyskania rozwiązania pierwotnego problemu. W innych dziedzinach istnieje jednak naturalny podział na dwa konkurujące ze sobą elementy. Przykładem takiej dziedziny jest teoria gier i gry o sumie

zerowej [9], w których gracz A i gracz B konkurują ze sobą i jeżeli jeden wygrywa, drugi przegrywa. Żeby wygrać, gracze nie mogą ze sobą współpracować. Istnieją oczywiście również gry koooperacyjne lub częściowo kooperacyjne. Nie są to gry o sumie zerowej i często mają o wyższy poziom skomplikowania. Dla gier kooperacyjnych i częściowo kooperacyjnych nie można w sposób jednoznaczny określić interakcji między graczami jako „konkurencja". Dla problemów, w których relacja konkurencji występuje w sposób jednoznaczny, opłacalne może być zastosowanie koewolucji konkurencyjnej.

Przykładem zastosowania konkurencyjnego modelu koewolucyjnego w teorii gier jest metoda CoEvoSG (*Coevolutionary Algorithm for solving SSG*) [78]. W grach SSG (*Stackelberg Security Games*) zadaniem gracza obrońcy (ang. *defender*) jest ciągła obrona pewnych celów ataku przy wykorzystaniu ograniczonych zasobów, a zadaniem gracza napastnika (ang. *attacker*) jest analiza strategii obrońcy i atak określonych celów [30]. Pojedyncza akcja obrońcy reprezentuje przypisanie pewnych zasobów do patroli lub punktów kontrolnych, co pozwala na obronę określonych celów ataku. Pojedynczą akcją napastnika jest atak. Czystą strategią gracza (ang. *pure strategy*) jest lista wykonywanych przez niego akcji w kolejnych momentach czasu. Strategie graczy są ustalone od początku gry, czyli nie jest możliwa zmiana zachowania w odpowiedzi na podjęte przez przeciwnika kroki. Zadaniem jest znalezienie stanu równowagi Stackelberga (ang. *Stackelberg equilibrium*), czyli takiej pary strategii obrońcy i napastnika, dla której obaj gracze uzyskują maksymalną liczbę punktów. Metoda CoEvoSG wykorzystuje koewolucję konkurencyjną, aby uniknąć konieczności iteracji po wszystkich możliwych czystych strategiach napastnika w celu oceny strategii obrońcy. Koewolucji poddawane są dwie populacje - populacja mieszanych strategii obrońców i populacja czystych strategii napastników, a interakcja między nimi ma postać powiązanej funkcji oceny. Zamiast oceniać strategię obrońcy w stosunku do wszystkich możliwych czystych strategii napastnika, oceniana jest ona tylko w odniesieniu do podzbioru strategii napastnika, czyli populacji napastników - analogicznie dla oceny strategii napastników. Populacje konkurują ze sobą, czyli populacja obrońców szuka najlepszej możliwej odpowiedzi na strategie napastników i odwrotnie. W metodzie CoEvoSG populacje ewoluują naprzemiennie - najpierw populacja napastników ewoluuje przez pewną liczbę pokoleń konkurując z niezmieniającą się populacją obrońców, a następnie przez taką samą liczbę pokoleń ewoluuje populacja obrońców konkurując z niezmieniającą się populacją napastników.

Innym przykładem problemu, dla którego istnieje podział na dwa elementy konkurujące ze sobą na zasadzie „ja wygrywam, ty przegrywasz", jest optymalizacja roju bezzałogowych statków powietrznych (ang. *unmanned aerial vehicles*, UAV), inaczej znanych jako drony. Rój dronów ma na celu monitorowanie pewnego obszaru i ochronę go przed intruzami. Jednocześnie, intruzi chcą wtargnąć na monitorowany obszar i nie dać się złapać dronom. Problem można traktować zgodnie z podejściem drapieżnik-ofiara, gdzie rolę drapieżnika pełni dron, a rolę ofiary intruz. Zadaniem dronów jest maksymalizacja wykrywania intruzów, podczas gdy zadaniem intruzów jest maksymalizacja udanych włamań. Do rozwiązania tego problemu zastosowano metodę CompCGA (*Competitive Coevolutionary Genetic Algorithm*) [63]. W metodzie CompCGA ewoluują dwie populacje - jedna populacja to populacja dronów, która ma za zadanie maksymalizować procent wykrytych intruzów, a druga populacja to populacja intruzów, która ten procent minimalizuje. Populacje wchodzą w interakcję podczas wyznaczania przystosowania osobników - do oceny populacji dronów wykorzystywany jest

pewien reprezentant z populacji intruzów i odwrotnie. Reprezentantem populacji jest jej najlepszy osobnik, który jest aktualizowany co kilka pokoleń. Dzięki temu konkurująca z reprezentantem populacja ma czas, by się do niego dostosować. Jednocześnie populacje stale rozwijają się, ponieważ reprezentant jest aktualizowany.

Niektóre koewolucyjne podejścia konkurencyjne wykorzystują relację kolonizacji - na przykład metoda CoMOGP (*Coevolutionary Multi-objective Genetic Programming*) [41]. Kolonizacja oparta jest na relacji drapieżnik-ofiara, gdzie rolę drapieżników pełnią osobniki o lepszej wartości przystosowania, zastępując (kolonizując) ofiary, czyli osobniki o gorszej wartości przystosowania. CoMOGP wykorzystuje dwie populacje, które oceniane są różnymi funkcjami przystosowania i z których tylko jedna pełni rolę kolonizatora. Populacja będąca kolonizatorem wysyła osobniki kolonizujące do drugiej populacji, przy czym proces kolonizacji odbywa się co pewną liczbę pokoleń. W pierwszej kolejności wybierana jest część najlepszych rozwiązań z populacji kolonizującej oraz równa jej pod względem liczebności grupa losowych rozwiązań z populacji kolonizowanej. Następnie kolonizatorzy walczą o przetrwanie z rozwiązaniami kolonizowanymi - rozwiązania oceniane są biorąc pod uwagę funkcję fitness populacji kolonizowanej. Z dwóch rozwiązań udaje się przetrwać temu rozwiązaniu, które dominuje swojego przeciwnika - nie wszyscy kolonizatorzy zastępują zatem rozwiązania kolonizowane. Rozwiązania, którym udało się przetrwać, przechodzą do następnej iteracji, często zastępując w populacji kolonizowanej rozwiązania, z którymi rywalizowały.

Konkurencyjny model koewolucyjny ma zastosowanie także dla optymalizacji wielokryterialnej. Jako inspirowaną koewolucją konkurencyjną można potraktować metodę MMPICEAg (*Preference-inspired Coevolutionary Algorithm with Active Diversity Strategy for Multi-objective Multi-modal Optimisation*) [69]. MMPICEAg koewoluje populację rozwiązań z populacją wektorów celu, reprezentujących preferencje użytkownika. Wektory celu są losowo rozmieszczone wewnątrz hipersześcianu określonego przez punkty *Perfect Point* i *Nadir Point*. Punkty *Perfect Point* i *Nadir Point* to odpowiednio punkt o najlepszych możliwych wartościach kryteriów i punkt o najgorszych możliwych wartościach kryteriów, zgodnie z definicją przedstawioną w podrozdziale 2.1. Rozwiązania i wektory celu konkurują ze sobą, ponieważ rozwiązania starają się zdominować jak najwięcej wektorów celu, a wektory celu starają się uniknąć zdominowania. Przystosowanie osobników każdego gatunku jest więc wyznaczane na podstawie interakcji z drugim gatunkiem. Dodatkowo, przystosowanie osobników jest obliczane w sposób uwzględniający różnorodność. Rozwiązania uzyskują tym większą wartość przystosowania, im większą liczbę wektorów celu dominują, z tym że wartość przystosowania jest dzielona między wszystkie rozwiązania, które dominują te wektory celu. Oznacza to, że wartość przystosowania rozwiązań jest proporcjonalna do ich rangi gęstości (ang. *density rank*) w przestrzeni celu. Wartość przystosowania wektorów celu jest tym mniejsza, im więcej rozwiązań dominuje ten wektor celu.

Całkowicie inne podejście do koewolucji konkurencyjnej zastosowane zostało w selekcji cech (ang. *feature selection*) w systemie diagnostyki wspomaganej komputerowo (ang. *computer-aided diagnosis system, CAD system*) dla diagnozowania rozedmy płuc na podstawie tomografii komputerowej klatki piersiowej [26]. W tym podejściu dwa algorytmy inspirowane naturą - algorytm czepiaków (ang. *spider monkey optimisation*, SMO) oraz algorytm pól ryżowych (ang. *paddy field algorithm*, PFA) - operują każdy na swoim zbiorze rozwiązań,

który można traktować jako odpowiednik populacji. Dzięki wykorzystaniu dwóch algorytmów przestrzeń przeszukiwana jest na różne sposoby, co zwykle pozwala uniknąć utknięcia w lokalnych optimach. W każdej iteracji koewolucji konkurencyjnej ta sama populacja początkowa podawana jest jako wejście dla metod SMO i PFA. Każda metoda niezależnie pracuje na swojej populacji, w wyniku czego uzyskiwane są dwie populacje, które mogą być traktowane jako dwa różne gatunki o różnych charakterystykach. Znalezione rozwiązania są oceniane i sortowane według swoich wartości przystosowania. Następnym krokiem jest interakcja obu populacji - najlepszy osobnik z populacji SMO walczy z najlepszych osobnikiem z populacji PFA, drugi najlepszy osobnik z populacji SMO walczy z drugim najlepszym osobnikiem z populacji PFA i tak dalej. Taki „pojedynek" wygrywa zawsze osobnik o lepszej wartości przystosowania. Zbiór wszystkich osobników wygrywających tworzy populację początkową do następnej iteracji koewolucji konkurencyjnej. Mimo, że to podejście różni się od innych przedstawionych w literaturze, pomiędzy wynikami działania metod SMO i PFA, które można traktować jako populacje, występuje relacja konkurencji. Metoda może być więc określona mianem koewolucji konkurencyjnej.

Zastosowaniem podobnej idei dla optymalizacji wielokryterialnej jest metoda DPPCP (*Dual-population Competitive Coevolutionary Approach*) [66]. W metodzie DPPCP konkurują ze sobą populacja ewoluowana mechanizmem Pareto (ang. *Pareto-based*) i populacja ewoluowana mechanizmem dekompozycji (ang. *decomposition-based*). Mechanizm Pareto polega na szukaniu rozwiązań niezdominowanych i wykorzystuje relację dominacji Pareto. Mechanizm dekompozycji polega na transformacji problemu wielokryterialnego w wiele problemów jednokryterialnych i uzyskaniu rozwiązania problemu wielokryterialnego poprzez połączenie rozwiązań problemów jednokryterialnych [35]. Populacja ewoluowana mechanizmem Pareto skupia się na osiągnięciu dobrej zbieżności do prawdziwego frontu Pareto, a populacja ewoluowana mechanizmem dekompozycji na zachowaniu dużej różnorodności rozwiązań. W każdym pokoleniu za pomocą mechanizmu selekcji opartego na sąsiadach (ang. *neighbour-based selection mechanism*) wybierani są trzej kandydujący rodzice z każdej populacji. Potomstwo dla każdego gatunku generowane jest za pomocą ewolucji różnicowej. Następnie zachodzi konkurencja obu gatunków - konkuruje ze sobą potomstwo wygenerowane dla każdego gatunku. Do populacji potomnej ewoluowanej mechanizmem Pareto przechodzi ten potomek, który wygrywa biorąc pod uwagę dominację Pareto. Analogicznie, do populacji potomnej ewoluowanej mechanizmem dekompozycji przechodzi ten potomek, który jest zwycięzcą konkurencji przy użyciu metryk opartych na dekompozycji. Na końcu procesu koewolucji populacja wynikowa jest połączeniem obu populacji, aby zachować ich dobre właściwości - różnorodność i zbieżność.

2.3.3. Podejścia mieszane

Poza podejściami zgodnymi z ideą koewolucji kooperacyjnej lub konkurencyjnej istnieją też metody koewolucyjne, które trudno jednoznacznie określić jako zgodne z tylko jednym z podejść. Występuje w nich podział na kilka populacji, które wchodzą ze sobą w interakcje, ale sam charakter tych interakcji jest trudny do określenia lub występuje kilka sposobów interakcji. Z tego powodu w tej pracy zdefiniowano podział koewolucyjnych podejść na kooperacyjne, konkurencyjne i mieszane.

Przykładem podejścia mieszanego jest MLGA (*Multilevel Cooperative Genetic Algori-*

thm) [5]. Jest ona oparta na idei wielopoziomowej selekcji (ang. *multilevel selection*). Wielopoziomowa selekcja jest inspirowana tworzeniem się kolektywów (stad, watah, plemion) w ramach jednego gatunku. Osobniki z jednego kolektywu współpracują ze sobą, co zwiększa ich szanse na przetrwanie, a jednocześnie różne kolektywy konkurują ze sobą. Istnieją dwa najczęstsze podejścia do wielopoziomowej selekcji - fitness kolektywu oceniany jest tą samą funkcją, co fitness osobników (MLS1) lub fitness kolektywu oceniany jest inną funkcją, niż fitness osobników (MLS2). W przypadku MLS1 przystosowanie grupy może być równe przystosowaniu najlepszego osobnika z tej grupy lub wartości przystosowania wszystkich osobników z grupy mogą być agregowane [61]. Metoda MLGA wykorzystuje arbitralny sposób dekompozycji pierwotnego problemu na podproblemy. Każdy podproblem jest optymalizowany przez jedną populację i reprezentanci wszystkich populacji łączą się w celu uzyskania rozwiązania pierwotnego problemu. Metoda MLGA jest więc zgodna z podejściem koewolucji kooperacyjnej. Każda populacja jest jednak dzielona na grupy i między grupami zachodzi relacja konkurencji, zatem MLGA wykazuje cechy pewne cechy kooperacyjne i pewne cechy konkurencyjne. W każdym pokoleniu na ewoluujących populacjach dokonywane są standardowe operatory ewolucyjne - selekcja, krzyżowanie oraz mutacja. Co pewną liczbę pokoleń następuje interakcja między grupami - grupa o najgorszej wartości fitness jest kolonizowana przez grupę o najlepszej wartości fitness. W wyniku kolonizacji grupa kolonizowana (o gorszym przystosowaniu) jest z pewnym prawdopodobieństwem zastępowana przez grupę kolonizującą (o lepszym przystosowaniu). Interakcje między grupami mają zatem charakter konkurencyjny, podczas gdy interakcje między populacjami mają charakter kooperacyjny.

Inną metodą, w której występują i interakcje o charakterze kooperacyjnym, i interakcje o charakterze konkurencyjnym jest metoda CCPSO (*Competitive and Cooperative Coevolutionary Multi-objective Particle Swarm Optimization Algorithm*) [21]. Metoda PSO jest reprezentantem grupy metod inteligencji rojowej lub rozproszonej (ang. *swarm intelligence*). W tych metodach ze współpracy pojedynczych jednostek, nie posiadających właściwie żadnej inteligencji i kierujących się prostymi regułami, wyłania się inteligencja zbiorowości, pozwalająca na zbliżanie się do optymalnego rozwiązania analizowanego problemu. W metodzie PSO każda cząsteczka (rozwiązanie) posiada pewne położenie i prędkość, przy czym jakość cząstki oceniana jest na podstawie jej położenia. Każdy członek roju w każdej iteracji zmienia swoje położenie biorąc pod uwagę swój wektor prędkości. W każdej iteracji zmieniana jest też prędkość cząstki, w taki sposób by dążyła ona do najlepszego rozwiązania, jakie dotychczas znalazła i jednocześnie do najlepszego rozwiązania znalezionego przez jej sąsiadów. Podczas modyfikacji prędkości brana jest również pod uwagę pewna bezwładność cząstki. Istnieją różne topologie sąsiedztwa dla PSO, w szczególności metoda CCPSO wykorzystuje jako zbiór sąsiadów archiwum globalnie niezdominowanych cząstek. Każda cząstka kierowana jest więc w stronę znalezionego frontu Pareto, aby uzyskać jak największą zbieżność do prawdziwego frontu Pareto. W metodzie CCPSO występuje wiele powiązanych rojów cząstek - pierwotny problem dekomponowany jest na zmienne decyzyjne i dla każdej zmiennej decyzyjnej może istnieć kilka optymalizujących ją rojów cząstek (rój cząstek może również optymalizować kilka zmiennych decyzyjnych). Pomiędzy rojami występuje relacja współpracy, ponieważ rozwiązanie pierwotnego problemu jest połączeniem rozwiązań znalezionych przez roje reprezentujące każdą ze zmiennych decyzyjnych. Do utworzenia rozwiązania problemu pierwotnego dla każdej zmiennej decyzyjnej można wybrać tylko jednego reprezentanta, a jednocześnie kilka rojów

może optymalizować tę samą zmienną decyzyjną. Z tego powodu między rojami występuje również relacja konkurencji. Każdy rój może reprezentować zmienną decyzyjną z pewnym prawdopodobieństwem, które jest aktualizowane zgodnie z wynikiem rywalizacji w taki sposób, że prawdopodobieństwo wyboru roju wygrywającego jest zwiększane. W momencie rywalizacji budowane jest rozwiązanie problemu pierwotnego dla aktualnie analizowanego roju i jednego konkurenta (ang. *competitor swarm*) i rozwiązania są ze sobą porównywane. Do budowy rozwiązania problemu pierwotnego wykorzystywana jest najlepsza cząstka z aktualnego roju i najlepsze cząstki z rojów reprezentujących pozostałe zmienne decyzyjne. Rój, dla którego uzyskano lepsze rozwiązanie, jest zwycięzcą i reprezentuje zmienną decyzyjną w tej iteracji.

2.4. Podsumowanie przeglądu literaturowego

W niniejszym rozdziale przedstawiono wybrany silnie ograniczony problem harmonogramowania (problem MS-RCPSP) oraz jego formalną definicję, a także kryteria oceny jego rozwiązań. Omówiono relację dominacji Pareto, która jest oparciem dla bardzo wielu metod optymalizacji wielokryterialnej. Jako wprowadzenie do przeglądu metod koewolucyjnych przedstawiono metody ewolucyjne, w szczególności klasyczne metody ewolucyjne dla optymalizacji wielokryterialnej. Główną część przeglądu literatury stanowi przegląd metod koewolucyjnych. Podstawową cechą podejść koewolucyjnych jest istnienie kilku populacji, które wpływają na siebie nawzajem za pomocą interakcji o różnym charakterze. Zależnie od charakteru interakcji między populacjami większość podejść koewolucyjnych można podzielić na dwa zasadnicze rodzaje: kooperacyjne i konkurencyjne. W podejściach kooperacyjnych między populacjami występuje relacja współpracy. Najczęściej podejścia kooperacyjne zgodne są z paradygmatem dziel i rządź, czyli pierwotny problem dekomponowany jest na podproblemy, które rozwiązywane są w osobnych populacjach. Interakcja między populacjami ma postać łączenia rozwiązań podproblemów w celu uzyskania rozwiązania problemu pierwotnego. W podejściach konkurencyjnych między populacjami występuje relacja konkurencji - na przykład kolonizacja lub gra na zasadzie „ja wygrywam, ty przegrywasz". Niektóre metody wykorzystują kilka populacji wchodzących w interakcje o charakterze trudnym do określenia lub w kilka różnych rodzajów interakcji - takie metody nazwano podejściami mieszanymi.

Przeglądu literatury dokonano ze szczególną uwagą na osiągające najlepsze wyniki metody oraz metody stosowane do rozwiązywania wybranego silnie ograniczonego problemu harmonogramowania. Obiecującą metodą ewolucyjną dla problemu MS-RCPSP jest ewolucja różnicowa, która zastosowana została do rozwiązywania problemu MS-RCPSP z ważoną funkcją celu [48]. Problem MS-RCPSP w naturalny sposób dzieli się na dwa podproblemy - przypisywanie zasobów do zadań oraz określanie kolejności wykonywania zadań. Nasuwa się więc wykorzystanie kooperacyjnego modelu koewolucyjnego. Taki model został zastosowany do rozwiązywania problemu MS-RCPSP, przyjmując optymalizację jednokryterialną z ważoną funkcją fitness [45]. Oprócz tego, żadna z przedstawionych kooperacyjnych metod koewolucyjnych nie wykorzystuje operatora selekcji GAP, zaprezentowanego po raz pierwszy w metodzie NTGA2, który promuje osobniki leżące blisko „luk" we froncie Pareto. Rozszerzając kooperacyjny model koewolucyjny zastosowany do rozwiązywania problemu MS-RCPSP jako problemu jednokryterialnego o relację dominacji Pareto i inspirując się klasycznymi metodami

optymalizacji wielokryterialnej, w szczególności wykorzystując operator selekcji GAP, istnieje możliwość otrzymania dobrych rezultatów również dla optymalizacji wielokryterialnej metodą koewolucyjną w problemie MS-RCPSP.

3. Opis proponowanego podejścia

Do rozwiązywania problemu MS-RCPSP zaproponowano autorską metodę opracowaną na podstawie przeglądu literatury - w szczególności wykorzystano inspiracje metodami CoEA dla MS-RCPSP [45] i NTGA2 [43] oraz zastosowaniem ewolucji różnicowej do rozwiązywania MS-RCPSP metodą DEGR [48], a także modyfikacją ewolucji różnicowej wykorzystującą operator selekcji GAP i zastosowaną w GaMeDE2 [8]. Autorska metoda jest kooperacyjnym podejściem koewolucyjnym do rozwiązywania problemu MS-RCPSP z kryterium czasu i kosztu, przy czym nie jest wykorzystywana ważona funkcji przystosowania. Wynikiem działania metody jest przybliżenie prawdziwego zbioru Pareto i odpowiadające mu przybliżenie prawdziwego frontu Pareto. Zastosowano kooperacyjny model koewolucyjny dla problemu MS-RCPSP przestawiony w CoEA: podział na dwie populacje, z których jedna proponuje przypisania zasób-zadanie, a druga kolejność wykonania zadań. Interakcja między populacjami polega na połączeniu ich reprezentantów w celu uzyskania harmonogramu stanowiącego rozwiązanie analizowanego problemu. Do wygenerowania harmonogramu na podstawie osobników populacji wykorzystano zachłanny schemat generowania harmonogramów (ang. *schedule generator schema*). Podstawę autorskiej metody stanowi ewolucja różnicowa wykorzystująca selekcję GAP, inspirowana GaMeDE2. Dwoma podstawowymi cechami GaMeDE2, z których czerpano inspiracje, są oparcie metody na ewolucji różnicowej oraz wykorzystanie operatora selekcji GAP zaproponowanego w NTGA2. Jako nazwę proponowanego podejścia wybrano CCoDE (ang. *Cooperative Coevolutionary Differential Evolution*).

Autorską metodę CCoDE przedstawiono w pseudokodzie 3.1. W pierwszej kolejności inicjalizowane i oceniane są populacje początkowe oraz wybierane są z nich osobniki niezdominowane, do których przechowywania służy archiwum. W archiwum osobników niezdominowanych przechowywane są tylko pełne rozwiązania pierwotnego problemu, czyli osobniki powstałe w wyniku koewolucji. Podstawą metody jest iteracyjne przetwarzanie populacji przez liczbę pokoleń (iteracji) określoną parametrem metody. W każdej iteracji generowana jest populacja potomna, która będzie pełniła rolę populacji rodzicielskiej w następnej iteracji. W pierwszej kolejności operatorem selekcji turniejowej opartej na dominacji Pareto lub operatorem selekcji GAP wybierana jest para rodziców. Dla przyjętego warunku wykorzystania każdego z operatorów selekcji (linijka 11 pseudokodu 3.1), najpierw przez pewną liczbę pokoleń wykorzystywana jest selekcja turniejowa oparta na dominacji Pareto, a następnie przez pewną liczbę pokoleń wykorzystywana jest selekcja GAP. Liczba pokoleń wykorzystujących każdą z metod selekcji określona jest parametrem metody jako procent pokoleń, w których wykorzystana ma być selekcja GAP - stąd w warunku wykorzystania każdego z operatorów selekcji występuje wartość 100. Wybrani odpowiednim operatorem selekcji rodzice następnie poddawani są operatorom mutacji i krzyżowania. Stosowane operatory mutacji i krzyżowania opisano szczegółowo w podrozdziałach 3.2 i 3.3. Po wygenerowaniu całej populacji potomnej jest ona oceniana wykorzystując drugą populację i archiwum osobników niezdominowanych. Po ocenie obu populacji następuje aktualizacja archiwum. Wszystkie elementy archiwum, które są dominowane

przez osobniki uzyskane w tej iteracji, są usuwane z archiwum, a niezdominowane osobniki uzyskane w tej iteracji są do niego dodawane. Jako wynik pojedynczego wykonania metody zwracane jest archiwum osobników niezdominowanych stanowiące przybliżenie prawdziwego zbioru Pareto.

Pseudokod 3.1: Pseudokod autorskiej metody CCoDE.

```
   Data: generationLimit
         gapSelection - percent of iterations in which the gap selection operator is used
   Result: archive - approximation of Pareto set
 1 archive ← empty;
 2 initialise populations;
 3 evaluate populations;
 4 update archive with populations;
 5 i ← 0;
 6 while i < generationLimit do
 7     nextPopulations ← empty;
 8     foreach population in populations do
 9         nextPopulation ← nextPopulations.get(population);
10         while size(nextPopulation) < size(population) do
11             if (i % gapSelection) < (100 − gapSelection) then
12                 perform Pareto-dominance based tournament selection on
                    population ∪ archive to obtain parents;
13             else
14                 perform gap selection on population ∪ archive to obtain parents;
15             end if
16             perform mutation and crossover on parents to obtain children;
17             while nextPopulation contains children do
18                 perform mutation on children;
19             end while
20             nextPopulation ← nextPopulation ∪ children;
21         end while
22     end foreach
23     choose collaborationPartners for every individual in nextPopulations;
24     evaluate nextPopulations against collaborationPartners;
25     populations ← nextPopulations;
26     update archive with populations;
27     i ← i + 1;
28 end while
29 return archive;
```

W podejściu koewolucyjnym w celu oceny współewoluujących populacji muszą zostać one połączone. W pseudokodzie 3.1 odpowiadają za to linijki 23-24. Zdefiniowano kilka wariantów metody. W każdym wariancie dla każdego osobnika z obu populacji wybierana jest pewna liczba kandydatów na partnera do współpracy, z którymi łączony jest dany osobnik - liczba

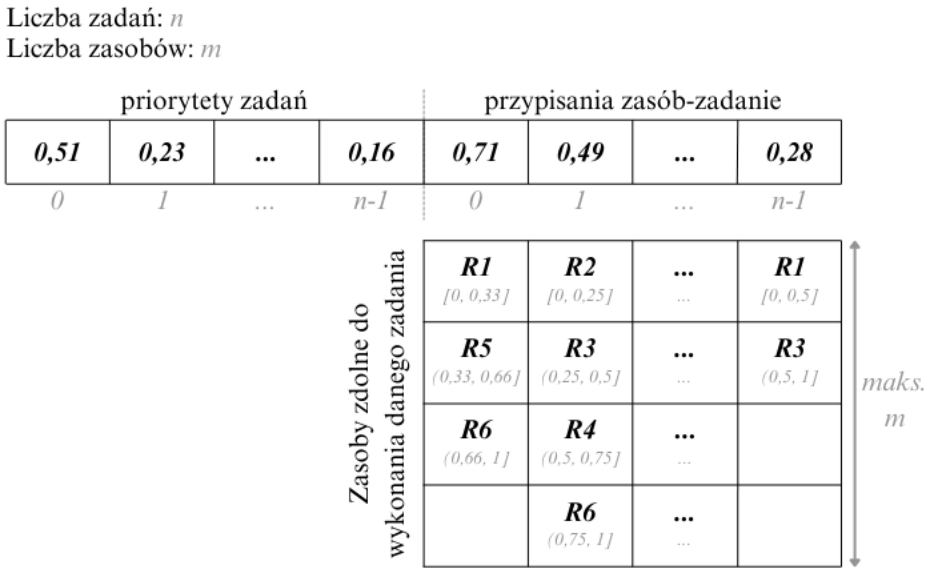
wybieranych osobników określona jest parametrem metody. Osobnik łączony jest z każdym kandydatem w taki sposób, że jedna połowa genotypu brana jest od osobnika, a druga połowa genotypu od kandydata. Dla każdej pary osobników powstaje więc nowy osobnik, posiadający wymieszane geny, i to na jego podstawie generowany jest harmonogram. Następnie każdy utworzony harmonogram oceniany jest względem wszystkich kryteriów. Jako „najlepszy" kandydat, czyli kandydat stanowiący ostatecznego partnera do współpracy wybieranego do wygenerowania harmonogramu, wybierany jest ten spośród kandydatów, dla którego wartość miary HV, równa objętości hiperześcianu wyznaczonego przez front Pareto i najgorszy możliwy punkt pod względem wszystkich kryteriów, jest najlepsza. Rozwiązania niezdominowane mają lepszą wartość HV od rozwiązań zdominowanych, dlatego nie jest konieczne sprawdzanie, który z uzyskanych harmonogramów jest niezdominowany. Szczegółową definicję miary HV przedstawiono w podrozdziale 4.3. Możliwe jest zastosowanie innej miary jakości w celu oceny „najlepszego" kandydata na partnera do współpracy - na przykład miary ED, równej średniej odległości euklidesowej między punktami frontu Pareto a najlepszym możliwym punktem pod względem wszystkich kryteriów. Rozwiązania niezdominowane znajdują się bliżej idealnego punktu od rozwiązań zdominowanych, mają zatem lepszą wartość ED i, tak jak w przypadku miary HV, nie jest konieczne sprawdzanie, który z uzyskanych harmonogramów jest niezdominowany. Kandydaci na partnera do współpracy wybierani są z drugiej populacji sposobem zależnym od wariantu metody CCoDE. Zależnie od wariantu metody, koewolucję może wspierać również archiwum osobników niezdominowanych - wtedy kandydaci na partnera do współpracy wybierani są zarówno z drugiej populacji, jak i z archiwum. Warianty metody CCoDE i różnice między nimi opisano szczegółowo w podrozdziale 3.5.

Parametry metody:

- liczba analizowanych pokoleń $GenerationLimit \in \mathbb{N}$ - warunek stopu,
- rozmiar populacji $PopulationSize \in \mathbb{N}$,
- liczba pokoleń wykorzystujących każdą z metod selekcji interpretowana jako wartość procentowa $GapSelectionPercent \in \{0, 1, 2, \ldots, 100\}$,
- rozmiar turnieju dla operatora selekcji GAP interpretowany jako procent rozmiaru populacji $GapTournamentSize \in \{0, 1, 2, \ldots, 100\}$,
- rozmiar turnieju dla operatora selekcji turniejowej interpretowany jako procent rozmiaru populacji $TournamentSize \in \{0, 1, 2, \ldots, 100\}$,
- liczba kandydatów na partnera do współpracy $PartnerCandidates \in \mathbb{N}$,
- dla schematu ewolucji różnicowej:
  - siła mutacji $F \in [0, 1]$,
  - współczynnik krzyżowania $CR \in [0, 1]$,
- dla schematu klasycznego algorytmu ewolucyjnego:
  - prawdopodobieństwo mutacji $p_m \in [0, 1]$,
  - prawdopodobieństwo krzyżowania $p_{cx} \in [0, 1]$.

3.1. Reprezentacja osobnika

Metoda CCoDE wykorzystuje reprezentację genotypu jako wektora liczb rzeczywistych o długości $2n$, gdzie n to liczba zadań. Reprezentacja genotypu jako wektora liczb rzeczywistych pozwala na zastosowanie operatorów krzyżowania i mutacji zdefiniowanych dla ewolucji różnicowej, przedstawionych w podrozdziale 3.2. Pierwsza połowa genotypu definiuje dla każdego zadania priorytet jego wykonania - zadania o wyższym priorytecie wykonywane są w pierwszej kolejności. Druga połowa genotypu wskazuje dla każdego zadania propozycję zasobu, który powinien je wykonywać w ostatecznym harmonogramie - proponowane są tylko zasoby, które mogą wykonać zadanie, czyli posiadają umiejętność wymaganą przez zadanie na tym samym lub wyższym poziomie. Dla każdego zadania zdefiniowany jest zbiór zasobów, które mogą je wykonać i wartość genu wskazuje dla danego zadania jeden zasób z tego zbioru. W przykładzie przedstawionym na rysunku 3.1 zadanie o indeksie równym zero może być wykonane przez trzy zasoby: $R1$, $R5$ oraz $R6$, więc każdemu z zasobów przypada wartość około $0,33$. Ponieważ wartość genu wynosi $0,71$ do wykonania tego zadania zostanie przypisany zasób $R6$ $(0,66 < 0,71 < 1)$. Zadanie o indeksie równym jeden może być wykonane przez cztery zasoby: $R2$, $R3$, $R4$ i $R6$, zatem każdemu z zasobów przypada wartość $0,25$. Ponieważ wartość genu wynosi $0,49$ do wykonania tego zadania zostanie przypisany zasób $R3$ $(0,25 < 0,49 < 0,5)$. Analogicznie można określić przypisany zasób dla każdego kolejnego zadania.



Rysunek 3.1: Reprezentacja osobnika.

Dla takiej reprezentacji w podejściu koewolucyjnym osobniki z obu populacji są reprezentowane w ten sam sposób, czyli w populacji odpowiedzialnej za kolejność zasobów występuje narzut informacyjny związany z przechowywaniem informacji o przypisaniach zasób-zadanie i analogicznie dla drugiej populacji odpowiedzialnej za przypisania zasób-zadanie. Taka reprezentacja pozwala jednak na łatwą implementację metod koewolucyjnej i ewolucyjnej (bez podziału na populacje). Możliwa jest również ocena osobnika niezależnie (bez interakcji

z reprezentantem drugiego gatunku), co umożliwia porównanie osobników ze sobą przed procesem koewolucyjnej oceny, na przykład w celu wykorzystania operatora selekcji do wyboru kandydatów na partnera do współpracy.

3.2. Wykorzystanie ewolucji różnicowej

Metoda CCoDE oparta jest na ewolucji różnicowej wykorzystującej selekcję GAP oraz inspirowanej metodami GaMeDE2, NTGA2 oraz DEGR. Podstawową różnicą jest brak wykorzystania lokalnego przeszukiwania oraz inny charakter archiwum - archiwum służy jedynie do przechowywania osobników globalnie niezdominowanych i w związku z tym stanowi przybliżenie prawdziwego zbioru Pareto.

Populacja początkowa tworzona jest jako pierwszy krok ewolucji różnicowej i to od niej zaczyna się proces optymalizacji - wszystkie kolejne populacje powstają w wyniku ewolucji poprzedniej populacji. Populacja początkowa może zostać wygenerowana za pomocą różnych metod heurystycznych [48]. Najpopularniejszą i najprostszą metodą inicjalizacji początkowej jest inicjalizacja losowa - populacja początkowa jest populacją osobników o losowych wartościach każdego genu - i to ta metoda inicjalizacji wykorzystywana jest w opracowanym podejściu.

Każda kolejna populacja generowana jest następująco: w pierwszej kolejności operatorem selekcji wybierane są dwa osobniki rodzicielskie, które są następnie poddawane operatorowi mutacji w celu uzyskania osobników-mutantów, a w ostatnim kroku osobnicy-mutanci poddawani są operatorowi krzyżowania, dzięki czemu generowane są osobniki potomne. Tak jak w metodzie NTGA2, wykorzystywane są dwa operatory selekcji: selekcja turniejowa oraz selekcja GAP, przy czym każdy z operatorów wykorzystywany jest przez pewną liczbę pokoleń. Operator selekcji turniejowej wybiera z połączenia populacji i archiwum pewną liczbę losowych osobników, które następnie porównuje ze sobą pod względem ich rangi dominacji. Turniej wygrywa osobnik o najmniejszej randze dominacji, czyli osobnik dominowany przez najmniejszą liczbę innych osobników. Operator selekcji GAP zaproponowany w metodzie NTGA2 wybiera z połączenia populacji i archiwum dwa osobniki, które leżą blisko największych luk w aktualnym przybliżeniu frontu Pareto. W tym celu analizowane są luki jako minimalna odległość euklidesowa między osobnikiem a jego sąsiadami. Operator selekcji GAP opisano szczegółowo w podrozdziale 2.2. Archiwum bierze aktywny udział w selekcji w celu kierowania procesu optymalizacyjnego w stronę aktualnego przybliżenia prawdziwego frontu Pareto. Za pomocą obu operatorów selekcji wybierane są dwa osobniki rodzicielskie, które poddawane są operatorom mutacji i krzyżowania. Zasadę działania operatora mutacji opisuje równanie 3.1, gdzie v jest genotypem osobnika-mutanta, *parent* to jeden z osobników wybranych odpowiednim operatorem selekcji, r_1 oraz r_2 to różne od siebie nawzajem i osobnika *parent* losowe osobniki z pokolenia rodzica, x to genotyp osobników niezmutowanych (*parent*, r_1 oraz r_2), a F to siła mutacji będąca parametrem metody [8, 48]. Operator mutacji dla obu osobników rodzicielskich generuje osobnika-mutanta.

$$v_{parent} = x_{parent} + F * (x_{r_1} - x_{r_2}) \tag{3.1}$$

Do utworzenia populacji potomnej wykorzystywany jest operator krzyżowania dwumianowego przedstawiony w równaniu 3.2, gdzie u_j to j-ty gen osobnika potomnego, $v_{j,\,parent}$ to j-ty gen osobnika-mutanta powstałego w wyniku zastosowania operatora mutacji na rodzicu, $x_{j,\,parent}$ to j-ty gen rodzica, a CR to wskaźnik krzyżowania będący parametrem metody [72]. Krzyżowanie dwumianowe polega więc na wyborze genu z rodzica lub powstałego z rodzica osobnika-mutanta z pewnym prawdopodobieństwem określonym przez parametr CR i wpisaniu jego wartości do genotypu osobnika potomnego.

$$u_j = \begin{cases} v_{j,\,parent} & \text{if } rand() < CR, \\ x_{j,\,parent} & \text{otherwise.} \end{cases} \tag{3.2}$$

3.3. Porównanie z klasycznym algorytmem ewolucyjnym

Operator mutacji zaprezentowany w równaniu 3.1 i operator krzyżowania zaprezentowany w równaniu 3.2 to klasyczne operatory stosowane w ewolucji różnicowej. W celu oceny skuteczności ewolucji różnicowej w rozwiązywaniu wielokryterialnego problemu MS-RCPSP dla przyjętej reprezentacji istotne jest porównanie z klasycznym algorytmem ewolucyjnym.

Najczęściej stosowanym operatorem mutacji w klasycznym algorytmie ewolucyjnym jest mutacja losowa (ang. *random bit mutation, random resetting mutation, bitwise mutation*) [17]. Operator zaadaptowany dla reprezentacji rzeczywistoliczbowej przedstawiono w równaniu 3.3, gdzie v_j to j-ty gen osobnika-mutanta, x_j to j-ty gen osobnika mutowanego, a p_m to prawdopodobieństwo mutacji będące parametrem metody. Każdy gen osobnika mutowanego jest więc z pewnym prawdopodobieństwem zastępowany przez losową wartość i w wyniku tej operacji powstaje osobnik-mutant.

$$v_j = \begin{cases} rand() & \text{if } rand() < p_m, \\ x_j & \text{otherwise.} \end{cases} \tag{3.3}$$

Powstałe w wyniku mutacji osobniki-mutanty są następnie krzyżowane ze sobą za pomocą operatora krzyżowania równomiernego (ang. *uniform crossover*) [17] przedstawionego w równaniu 3.4, gdzie u_j to j-ty gen osobnika potomnego, $v_{j,\,parent1}$ to j-ty gen osobnika-mutanta powstałego z pierwszego rodzica, a $v_{j,\,parent2}$ to j-ty gen osobnika-mutanta powstałego z drugiego rodzica. Każdy gen osobnika potomnego może być z równym prawdopodobieństwem przepisany z pierwszego lub drugiego osobnika-mutanta powstałego z osobników rodzicielskich. Krzyżowanie dla każdej pary rodziców następuje z pewnym prawdopodobieństwem określonym parametrem metody.

$$u_j = \begin{cases} v_{j,\,parent1} & \text{if } rand() < 0,5, \\ v_{j,\,parent2} & \text{otherwise.} \end{cases} \tag{3.4}$$

3.4. Generowanie fenotypu

Fenotypem osobnika w problemie MS-RCPSP jest reprezentowany przez niego harmonogram, w którym zadania wykonywane są przez określone przez genotyp zasoby oraz w określonej przez genotyp kolejności. Do tworzenia harmonogramów na podstawie genotypu każdego osobnika autorska metoda CCoDE wykorzystuje zachłanny schemat generowania harmonogramów zaprezentowany w metodzie NTGA2 i przedstawiony w pseudokodzie 3.2. Harmonogram generowany jest biorąc pod uwagę priorytety zadań (pierwszą część genotypu) oraz przypisania zasób-zadanie (drugą część genotypu). Jako aktualnie wykonywane zadanie wybierane jest zadanie, które może być wykonane w danym momencie (spełnione są jego ograniczenia kolejnościowe) i jednocześnie posiadające najwyższy priorytet. Jeżeli nie wszyscy poprzednicy kolejnego pod względem priorytetu zadania nie zostali jeszcze wykonani, następuje przypisanie zasobów i czasu rozpoczęcia do każdego z niezrealizowanych poprzedników. Do wykonania wybranego zadania przypisywany jest zasób zgodnie z wartością genu dla tego zadania - szczegółowo reprezentację przypisań zasób-zadanie omówiono w podrozdziale 3.1. Wybrane zadanie wpisywane jest do harmonogramu jako wykonywane przez wybrany zasób w najwcześniejszym możliwym momencie. Najwcześniejszy możliwy moment to najwcześniejszym momencie, w którym zasób jest „wolny", tj. nie wykonuje innego zadania, a jednocześnie wszyscy poprzednicy zadania zostali wykonani.

Pseudokod 3.2: Pseudokod schematu generowania harmonogramów (*Schedule Generator Scheme*), zaadaptowano do potrzeb przyjętej reprezentacji z [3].

```
Data: Tasks
      Resources
      taskPriorities - genotype part representing task priorities
      resourceAssignments - genotype part representing task-resource assignments
Result: schedule
1  schedule ← empty;
2  foreach task in Tasks do
3  |  resource ← resourceAssignments.get(task);
4  |  schedule.assign(task, resource);
5  end foreach
6  Tasks ← Tasks.sort(taskPriorities);
7  foreach task in Tasks do
8  |  if task.canBePerformed() then
9  |  |  assign all not yet performed task.getPredecessors;
10 |  |  resource ← schedule.getResource(task);
11 |  |  startTime ← max(GetEarliestStartTime(task), resource.finishTime);
12 |  |  task.startTime ← startTime;
13 |  |  resource.finishTime ← startTime + task.duration;
14 |  end if
15 end foreach
16 return schedule;
```

3.5. Warianty proponowanej metody

Proponowana metoda może działać w kilku różnych wariantach, różniących się sposobem wyboru osobników do koewolucyjnej oceny oraz wykorzystaniem archiwum niezdominowanych osobników do wspomagania koewolucji. Podstawowe warianty działania metody to:

- kooperacyjna koewolucja z selekcją losową (CCoDE_R, ang. *Cooperative Coevolutionary Differential Evolution, Random*),
- kooperacyjna koewolucja z selekcją GAP (CCoDE_S, ang. *Cooperative Coevolutionary Differential Evolution, Selection based*),
- kooperacyjna koewolucja z selekcją mieszaną (CCoDE_SR, ang. *Cooperative Coevolutionary Differential Evolution, Selection based and Random*) - wykorzystuje zarówno selekcję losową, jak i GAP.

Wszystkie podstawowe warianty metody wykorzystują archiwum niezdominowanych osobników i różnią się jedynie sposobem wyboru kandydatów na partnera do współpracy z drugiej populacji oraz archiwum. Z drugiej populacji i archiwum wybierana jest taka sama liczba osobników, określona parametrem metody. W wariancie z selekcją losową (CCoDE_R) osobniki wybierane są w sposób losowy - zasadę działania tego wariantu zaprezentowano w pseudokodzie 3.3.

Pseudokod 3.3: Pseudokod wyboru kandydatów na partnera do współpracy w wariancie z selekcją losową (CCoDE_R).

Data: *population* - population to which the current individual does not belong
archive - approximation of Pareto set
partnerCandidates - number of collaboration partner candidates
Result: *candidates* - set of collaboration partner candidates

```
1 candidates ← empty;
2 i ← 0;
3 while i < partnerCandidates do
4 |   newCandidate ← random individual from population ∪ archive;
5 |   candidates ← candidates ∪ newCandidate;
6 |   i ← i + 1;
7 end while
8 return candidates;
```

W wariancie z selekcją GAP (CCoDE_S) osobniki wybierane są za pomocą operatora selekcji GAP. W przypadku osobników z drugiej populacji, są one w pierwszej kolejności oceniane niezależnie, a następnie na ocenionych niezależnie osobnikach działa operator selekcji GAP. Niezależna ocena jest możliwa, ponieważ osobniki w obu populacjach posiadają pełen genotyp pozwalający na wygenerowanie harmonogramu. W przypadku osobników z archiwum osobników niezdominowanych, są one oceniane koewolucyjnie przed dodaniem do archiwum, dlatego przed zastosowaniem selekcji GAP nie trzeba wykonywać żadnych dodatkowych

czynności. Zasadę działania wariantu CCoDE_S przedstawiono w pseudokodzie 3.4. Selekcja GAP to tak naprawdę selekcja turniejowa, w której osobniki porównywane są względem rozmiaru „luk" we froncie Pareto między osobnikiem a jego najbliższymi sąsiadami - promowane są osobniki, dla których luki we froncie Pareto są jak największe. Kryterium, według którego wyznaczane są rozmiary luk, wybierane jest losowo. Osobniki, dla których wyznaczone mają być rozmiary luk, sortowane są według wybranego kryterium. Dla pierwszego i ostatniego osobnika rozmiar luki ustalany jest jako nieskończoność, ponieważ są to skrajne punkty należące do frontu Pareto. Dla każdego następnego osobnika wyznaczane są różnice wartości wybranego kryterium dla tego osobnika i jego dwóch sąsiadów, a następnie jako rozmiar luki dla tego osobnika ustalana jest maksymalna z tych dwóch wartości.

Pseudokod 3.4: Pseudokod wyboru kandydatów na partnera do współpracy w wariancie z selekcją GAP (CCoDE_S).

Data: *population* - population to which the current individual does not belong
archive - approximation of Pareto set
partnerCandidates - number of collaboration partner candidates
Result: *candidates* - set of collaboration partner candidates
1 *candidates* ← empty;
2 perform gap selection based on individual fitness values on *population* ∪ *archive* to obtain *candidates* of size *partnerCandidates*;
3 **return** *candidates*;

Ostatnim z wariantów podstawowych jest wariant z selekcją mieszaną (CCoDE_SR), w którym stosowany jest zarówno losowy wybór osobników, jak i selekcja GAP. W tym wariancie oceniany osobnik łączony jest z dwa razy większą liczbą kandydatów na partnera do współpracy.

Oprócz wariantów podstawowych zdefiniowano także kilka wariantów pomocniczych, w szczególności:

- kooperacyjna koewolucja z selekcją losową i bez archiwum (CCoDE_R_NA, ang. *Cooperative Coevolutionary Differential Evolution, Random, No Archive*),
- kooperacyjna koewolucja z selekcją GAP i bez archiwum (CCoDE_S_NA), ang. *Cooperative Coevolutionary Differential Evolution, Selection based, No Archive*),
- kooperacyjna koewolucja z selekcją mieszaną i bez archiwum (CCoDE_SR_NA, ang. *Cooperative Coevolutionary Differential Evolution, Selection based and Random, No Archive*) - wykorzystuje zarówno selekcję losową, jak i GAP.

Jedyne, czym te warianty różnią się od wariantów podstawowych, to brak wykorzystania archiwum w momencie wyboru kandydatów na partnera do współpracy. Kandydaci na partnera do współpracy wybierani są jedynie z drugiej populacji.

Istnieje również możliwość uruchomienia metody bez koewolucji. Jest to wtedy metoda oparta na ewolucji różnicowej i wykorzystująca operator selekcji GAP, ale niekoewolucyjna o nazwie MODE (ang. *Multi-Objective Differential Evolution*). Ponieważ metoda MODE oparta jest na tej samej implementacji, również wykorzystuje reprezentację opisaną w podrozdziale 3.1 oraz operator selekcji GAP na zmianę z operatorem selekcji turniejowej opartej na dominacji Pareto. Różnicą między metodą uruchomioną z lub bez koewolucji jest liczba i rozmiar analizowanych populacji. Dla metody koewolucyjnej w jednym z dozwolonych wariantów działania (CCoDE_R, CCoDE_S, CCoDE_SR, CCoDE_R_NA, CCoDE_S_NA, CCoDE_SR_NA) analizowane są dwie populacje, każda o rozmiarze równym parametrowi metody *PopulationSize*. Dla metody MODE analizowana jest tylko jedna populacja o rozmiarze dwa razy większym niż wartość parametru metody *PopulationSize*. Metodę MODE zaimplementowano jako metodę referencyjną i porównanie z nią ma na celu sprawdzenie, czy zastosowanie mechanizmu koewolucji pozwala na osiągnięcie lepszych wyników. Pseudokod metody MODE zaprezentowano w pseudokodzie 3.5.

Jako opcjonalny element metody wprowadzono również mechanizm zapobiegania występowaniu klonów, wykorzystany w metodzie NTGA2 do wspierania różnorodności. Każdy osobnik potomny poddawany jest operatorowi mutacji tak długo, jak nie jest unikatowy. W przeciwieństwie do metody NTGA2, porównanie osobników następuje pod względem ich wartości kryteriów - osobnik jest klonem innego osobnika, jeżeli ich wartości kryteriów są sobie równe. Osobniki o różnych genotypach mogą mieć takie same wartości kryteriów, więc różne genotypowo osobniki wciąż mogą zostać oznaczone jako klony. Takie podejście może powodować oznaczenie jako klonów (i w efekcie usunięcie) osobników, które mogłyby prowadzić w obszary przestrzeni dobrej jakości, ale porównywanie każdego genu osobników jest niestety zbyt kosztowne obliczeniowo - złożoność obliczeniowa średnia $O(n/2)$ i pesymistyczna $O(n)$, a każdy osobnik potomny musi być porównany z każdym innym osobnikiem potomnym. Mechanizm zapobiegania występowaniu klonów wprowadzono jako element opcjonalny metody w celu oceny jego wpływu na jej skuteczność.

Pseudokod 3.5: Pseudokod metody opartej na ewolucji różnicowej i nie wykorzystującej koewolucji (MODE).

```
Data: generationLimit
      gapSelection - percent of iterations in which the gap selection operator is used
Result: archive - approximation of Pareto set
archive ← empty;
initialise population;
evaluate population;
update archive with population;
i ← 0;
while i < generationLimit do
    nextPopulation ← empty;
    while size(nextPopulation) < size(population) do
        if (i % gapSelection) < (100 − gapSelection) then
            perform Pareto-dominance based tournament selection on
             population ∪ archive to obtain parents;
        else
            perform gap selection on population ∪ archive to obtain parents;
        end if
        perform mutation and crossover on parents to obtain children;
        while nextPopulation contains children do
            perform mutation on children;
        end while
        nextPopulation ← nextPopulation ∪ children;
    end while
    evaluate nextPopulation;
    population ← nextPopulation;
    update archive with population;
    i ← i + 1;
end while
return archive;
```

4. Założenia metodologiczne i organizacja badań

Celem pracy jest ocena skuteczności podejścia koewolucyjnego w rozwiązywaniu problemów wielokryterialnych na podstawie wybranego problemu wielokryterialnego z silnymi ograniczeniami - wybrano silnie ograniczony problem MS-RCPSP rozwiązywany jednocześnie z kryteriami kosztu i czasu. Do osiągnięcia celu pracy przeprowadzono strojenie proponowanego podejścia oraz badania z jego udziałem.

Aby umożliwić porównanie autorskiego podejścia z metodami przedstawionymi w literaturze badania przeprowadzono na wzorcowych instancjach wybranego problemu - charakterystykę instancji testowych omówiono w podrozdziale 4.1. Podrozdział 4.2 poświęcony jest organizacji badań - w szczególności, przedstawione zostały narzędzia wykorzystane do przeprowadzenia badań oraz ogólny plan badań. W podrozdziale 4.3 zdefiniowano miary oceny jakości rozwiązań metod optymalizacji wielokryterialnej, które następnie wykorzystane zostały do oceny skuteczności proponowanej metody. W podrozdziale 4.4 postawiono pytania badawcze. Ponieważ autorska metoda CCoDE oparta jest na algorytmie ewolucyjnym i jako taka jest metaheurystyką, zanim zostanie zastosowana do rozwiązywania problemu musi zostać dostrojona drogą empiryczną. Zagadnienie strojenia metod poruszono w podrozdziale 4.5.

4.1. Charakterystyka instancji testowych

Zbiór wzorcowych instancji problemu MS-RCPSP wykorzystany jako instancje testowe do badań opublikowano w ramach projektu iMOPSE (*Intelligent Multi Objective Project Scheduling Environment*) [2,47,50]. W projekcie iMOPSE opracowano dwa zbiory - łatwy zbiór *EDU* (od „edukacyjny"), składający się z 6 instancji, oraz trudniejszy zbiór wzorcowy *d36*. Zbiór *EDU* nie zostanie wykorzystany do badań w ramach niniejszej pracy z uwagi na niski poziom trudności jego instancji. Zbiór *d36* zawiera 36 instancji, które zostały wygenerowane na podstawie rzeczywistych projektów we współpracy z działem Volvo IT we Wrocławiu i które mają większy poziom trudności.

Każda z instancji problemu MS-RCPSP definiowana jest poprzez liczbę zadań, zasobów, rodzajów umiejętności oraz ograniczeń pierwszeństwa. Szczegółowa definicja instancji zawiera oprócz tego stawki godzinowe i posiadane umiejętności dla każdego z zasobów oraz czas trwania i poprzedników dla każdego z zadań. Instancje ze zbioru *d36* dzielą się na dwie grupy: instancje posiadające 100 zadań i instancje posiadające 200 zadań. W pierwszej grupie instancje posiadają liczbę zasobów równą 5, 10 lub 20, a w drugiej grupie - 10, 20 lub 40. W obu grupach liczba rodzajów umiejętności równa jest 9 lub 15, a liczba ograniczeń pierwszeństwa jest zróżnicowana. Każdą z instancji przeanalizowano pod kątem miar między innymi przynależność (ang. *affiliation*), obciążenie (ang. *load*), różnica czasowa (ang. *time difference*), różnorodność

(ang. *variety*). Przynależność opisuje poziom powiązania między zadaniami. Obciążenie dotyczy obciążenia zasobów przez zadania do wykonania. Różnica czasowa dotyczy różnorodności w czasie trwania zadań. Różnorodność opisuje poziom zróżnicowania zasobów pod kątem posiadanych przez nie umiejętności. Na podstawie wszystkich zdefiniowanych miar definiowana jest miara trudności (ang. *difficulty*) i instancje ze zbioru *d36* podzielono na najłatwiejsze, średniej trudności i najtrudniejsze na podstawie tej miary.

4.2. Organizacja badań i przyjęte założenia

Jako rozszerzenie algorytmu ewolucyjnego autorska metoda CCoDE zawiera element niedeterminizmu. Wykorzystuje probabilistyczną selekcję, operatory krzyżowania działające z pewnym prawdopodobieństwem i wykorzystujące generator liczb pseudolosowych oraz operatory mutacji działające z pewnym prawdopodobieństwem i wprowadzające losowe zmiany w osobnikach. Z tego powodu badania skuteczności metody CCoDE wymagają wielokrotnego uruchomienia metody dla każdej instancji testowej. Metodę dla każdej instancji należy uruchomić min. dziesięć razy. Duża (lub mała) skuteczność uzyskana w pojedynczym uruchomieniu może być wynikiem przypadku. Z tego powodu dla każdej instancji każda metoda uruchamiana jest dziesięć razy i wyniki z tych uruchomień są uśredniane, co pozwala na analizę wartości średnich, odchyleń standardowych oraz testy statystyczne dla weryfikacji istotności statystycznej uzyskanych różnic.

Do uruchomienia jakiejkolwiek metody inteligencji obliczeniowej niezbędna jest w szczególności implementacja reprezentacji rozwiązywanego problemu oraz samej metody. Implementację wykonano w języku C++/20 w oparciu o bibliotekę iMOPSE [3]. Bibliotekę uzupełniono o reprezentację problemu MS-RCPSP dla podejścia koewolucyjnego, w którym do oceny każdego osobnika wymagany jest wybór osobnika stanowiącego partnera do współpracy i połączenie osobników w celu uzyskania ostatecznego harmonogramu. Reprezentację genotypu osobnika zaimplementowano zgodnie z opisem w podrozdziale 3.1, czyli genotyp osobnika jest wektorem liczb rzeczywistych. W skład implementacji reprezentacji problemu MS-RCPSP wchodzą: kodowanie poszczególnych rozwiązań problemu (genotyp osobników) oraz ocena rozwiązania, na którą składają się generowanie harmonogramu na podstawie genotypu i wyznaczenie wartości kryteriów dla uzyskanego harmonogramu. Wartości kryteriów wyznaczone dla fenotypu osobnika (harmonogramu wygenerowanego na podstawie genotypu) stanowią wartości kryteriów dla tego osobnika. Na potrzeby metody CCoDE bibliotekę iMOPSE uzupełniono o implementację operatorów krzyżowania dwumianowego i mutacji dla ewolucji różnicowej oraz zaimplementowano samą metodę CCoDE zgodnie z pseudokodem zaprezentowanym i omówionym w rozdziale 3. Do wyznaczenia miar oceny jakości przybliżeń prawdziwego frontu Pareto powstających jako wynik metod optymalizacji wielokryterialnej zastosowano moduł *paretoAnalyzer* udostępniony w ramach biblioteki iMOPSE. Do wizualizacji uzyskanych przybliżeń prawdziwego frontu Pareto na wykresach zastosowano moduł *scripts* z biblioteki iMOPSE.

Z uwagi na dużą czasochłonność wykonywania jednego uruchomienia metody, konieczność wielokrotnego jej uruchamiania dla każdej kombinacji wartości parametrów oraz dużą liczbę testowanych konfiguracji badania przeprowadzono na dziesięciu wybranych instancjach ze zbioru

wzorcowego *d36*. Wybrano dwie instancje najłatwiejsze (200_20_145_15 i 200_40_91_15), trzy instancje najtrudniejsze (100_10_64_9, 100_5_22_15 i 100_5_64_9) oraz pięć instancji średniej trudności (100_10_26_15, 100_20_47_9, 100_20_65_9, 200_10_128_15 i 200_40_90_9), gdzie poziom trudności instancji mierzony był miarą trudności. Instancje wybrano w taki sposób, aby zapewnić wybór co najmniej jednej instancji mniejszej (posiadającej 100 zadań) i jednej instancji większej (posiadającej 200 zadań), a także różną liczbę ograniczeń pierwszeństwa (różną gęstość ograniczeń). Charakterystykę wybranych instancji przedstawiono w tabeli 4.1

Tabela 4.1: Skrót charakterystyki wybranych ze zbioru wzorcowego *d36* iMOPSE, zaadaptowano z [50].

	Liczba			
Nazwa instancji	Zadań	Zasobów	Ograniczeń pierwszeństwa	Rodzajów umiejętności
100_5_22_15	100	5	22	15
100_5_64_9	100	5	64	9
100_10_26_15	100	10	26	15
100_10_64_9	100	10	64	9
100_20_47_9	100	20	47	9
100_20_65_9	100	20	65	9
200_10_128_15	200	10	128	15
200_20_145_15	200	20	145	15
200_40_90_9	200	40	90	9
200_40_91_15	200	40	91	15

Przed przejściem do badania metody CCoDE wymagane jest empiryczne dostosowanie wartości jej parametrów, zwane strojeniem. Dzięki strojeniu metody możliwe jest znalezienie konfiguracji metody wymaganej do właściwego badania jej skuteczności. W samej procedurze badań można wydzielić dwa etapy: badania wstępne i badania właściwe. W ramach badań wstępnych początkowo ręcznie manipulowano parametrami metody działającej w różnych wariantach w celu weryfikacji poprawności metody oraz określenia, jakie wartości parametrów należy przeanalizować w momencie właściwego strojenia. W następnej kolejności wstępnie zbadano między innymi: działanie metody we wszystkich zdefiniowanych wariantach, włącznie z pomocniczymi, działanie metody z wykorzystaniem miary HV lub ED do wyboru partnera do współpracy z kilku możliwych kandydatów, działanie metody z lub bez zastosowania mechanizmu zapobiegania występowania klonom oraz działanie metody przy wykorzystaniu schematu ewolucji różnicowej (mutacja z ewolucji różnicowej oraz krzyżowanie dwumianowe) lub schematu klasycznego algorytmu ewolucyjnego (mutacja losowa oraz krzyżowanie równomierne). Badania wstępne wykonano w celu wypracowania intuicji dla strojenia metody i dalszych badań oraz znalezienia najbardziej obiecujących wariantów, miar i mechanizmów do właściwej części badawczej pracy. Właściwa część badawcza pracy ma za zadanie umożliwić ocenę skuteczności metody CCoDE w odniesieniu do metod referencyjnych. Autorską metodę CCoDE w różnych wariantach porównano z metodą losową, metodą MODE oraz

metodami z literatury - NTGA2 [43] oraz B-NTGA [6,7]. Metodę losową przyjęto jako metodę bazową, od której dla każdej badanej metody skuteczność powinna być większa. Jako metodę losową przyjęto losowe generowanie liczby osobników równej liczbie urodzeń (analizowanych rozwiązań) w pozostałych metodach aby zachować sprawiedliwość porównania. Przybliżenia prawdziwego frontu Pareto uzyskane metodami NTGA2 i B-NTGA otrzymano od pana mgr inż. M. Antkiewicza, jednego z autorów metody B-NTGA. Porównano również skuteczność metody CCoDE w różnych jej wariantach. Różnice w działaniu poszczególnych wariantów metody opisano w rozdziale 3.

Podsumowanie metod przyjętych jako metody odniesienia:

- Random - losowe generowanie genotypu,
- NTGA2 z parametrami $GenLimit = 1000$, $PopSize = 50$, $P_m = 0,01$, $P_x = 0,6$, $TourSize = 6$ lub $TourSize = 20$ (zależnie od instancji), $gsGen = 50$ [43],
- B-NTGA z parametrami $GenLimit = 1000$, $PopSize = 50$, $P_m = 0,01$, $P_x = 0,9$, $TourSize = 40$ [6,7].

4.3. Miary oceny jakości rozwiązań

W optymalizacji wielokryterialnej wynikiem działania metody jest zbiór Pareto i odpowiadający mu front Pareto, zatem zadanie oceny wyniku jest stosunkowo skomplikowane. Uzyskane przybliżenia prawdziwego frontu Pareto powinny być oceniane pod względem ich zbieżności (ang. *convergence*) do prawdziwego frontu Pareto oraz różnorodności (ang. *diversity*) rozwiązań w zbiorze Pareto [16,42]. Zbieżność do prawdziwego frontu Pareto zapewnia zwracanie użytkownikowi jak najlepszych rozwiązań, a duża różnorodność rozwiązań oznacza, że nie ma obszarów przestrzeni rozwiązań, które nie były eksplorowane. Do oceny zbieżności i różnorodności wykorzystywane są różne miary oceny jakości (ang. *quality measures*), w tym PFS, IGD, HV oraz *Purity* [42,43]. Nie istnieje pojedyncza miara jakości, która może być wykorzystana do oceny jakości przybliżenia frontu Pareto. Istotne jest wykorzystanie takich miar jakości, które uzupełniają się nawzajem i razem pozwalają na precyzyjną ocenę uzyskanych przybliżeń frontu Pareto.

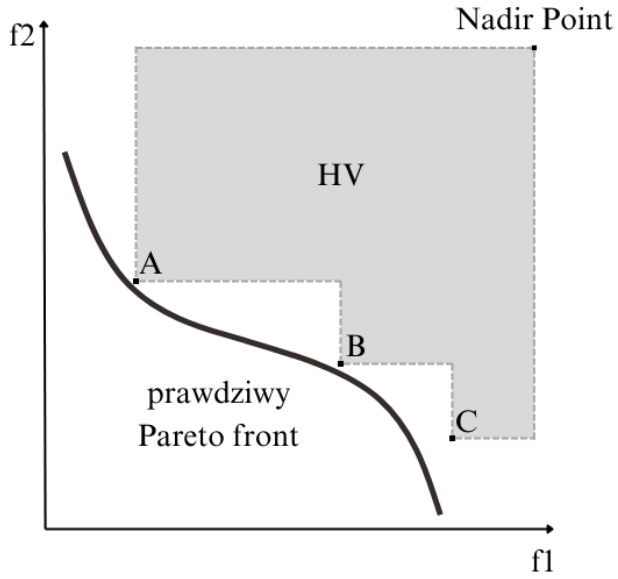
Do wyznaczenia niektórych miar oceny jakości wykorzystywany może być prawdziwy front Pareto. Dla wielu problemów, w tym problemu MS-RCPSP, prawdziwy front Pareto nie jest niestety znany. Dla miar oceny jakości wykorzystujących „prawdziwy" front Pareto jest on estymowany jako suma wszystkich znalezionych przybliżeń prawdziwego frontu Pareto, czyli zbiór wszystkich znalezionych globalnie niezdominowanych punktów. Punkty z jednego przybliżenia prawdziwego frontu Pareto, które są dominowane przez punkty z innego przybliżenia prawdziwego frontu Pareto nie należą do estymowanego „prawdziwego" frontu Pareto. Aby obejść konieczność estymacji prawdziwego frontu Pareto, niektóre miary oceny jakości korzystają ze zdefiniowanych punktów odniesienia, których definicja kończy podrozdział 2.1.

Najłatwiejszą miarą oceny jakości frontu Pareto jest miara PFS (ang. *Pareto Front Size*), czyli rozmiar frontu Pareto. Miara PFS jest równa liczbie unikalnych punktów frontu Pareto.

Front Pareto, który zawiera dużo unikalnych punktów, powstaje w wyniku dokładniejszego przeszukania przestrzeni. Dla dużego frontu Pareto użytkownik otrzymuje również więcej rozwiązań do wyboru według jego indywidualnych preferencji. Miara PFS powinna być więc maksymalizowana.

Do oceny zbieżności i różnorodności frontu Pareto można wykorzystać miarę odwróconej odległości pokoleniowej (ang. *Inverted Generational Distance*, IGD), wyznaczoną jako średnia odległość euklidesowa między punktami prawdziwego frontu Pareto a najbliższym punktem z przybliżenia prawdziwego frontu Pareto. Pojedynczy punkt prawdziwego frontu Pareto może być najbliższym punktem dla wielu punktów z przybliżenia prawdziwego frontu Pareto. Miarę IGD definiuje równanie 4.1, gdzie TPF to estymowany „prawdziwy" front Pareto (zbiór wszystkich niezdominowanych punktów), a d jest odległością między punktem s_i ze zbioru wszystkich niezdominowanych punktów TPF a najbliższym punktem przybliżenia prawdziwego frontu Pareto PF. Odległość między przybliżeniem prawdziwego frontu Pareto a prawdziwym frontem Pareto powinna być jak najmniejsza, zatem miara IGD powinna być minimalizowana.

$$IGD(PF) = \frac{\sqrt{\sum_{i=1}^{|TPF|} d(s_i, PF)^2}}{|TPF|} \tag{4.1}$$



Rysunek 4.1: Wizualizacja miary HV dla dwóch kryteriów, na której zaznaczone są niezdominowane punkty A, B, C należące do przybliżenia prawdziwego frontu Pareto oraz *Nadir Point*, zaadaptowano z [16, 32].

Punkt *Nadir Point* jest wykorzystywany do definicji miary HV (ang. *Hypervolume*), która mierzy zarówno zbieżność, jak i różnorodność. Miara HV jest równa objętości hiperześcianu wyznaczonego przez front Pareto i *Nadir Point*. Wizualizację miary HV przedstawiono na rysunku 4.1, a jej definicję zawarto w równaniu 4.2, gdzie s jest punktem należącym do frontu Pareto, s^{nadir} to *Nadir Point*, a Λ to miara Lebesgue'a zdefiniowana w równaniu

4.3 [53]. Niestety, złożoność obliczeniowa miary HV jest wykładnicza dla liczby kryteriów i wielomianowa dla liczby punktów, zatem dla więcej niż dwóch kryteriów obliczanie tej miary jest zbyt czasochłonne [32]. W badaniach w ramach pracy i w samej metodzie możliwe jest zastosowanie miary HV mimo jej wykładniczej złożoności obliczeniowej dla liczby kryteriów, ponieważ problem analizowany jest względem dwóch kryteriów - kosztu i czasu. Ponieważ odległość frontu Pareto od punktu *Nadir Point*, a zatem również objętość hiperześcianu wyznaczonego przez front Pareto i *Nadir Point* powinny być jak największe, miara HV powinna być maksymalizowana.

$$HV(PF) = \Lambda(\bigcup_{s \in PF} \{s' \mid s \prec s' \prec s^{nadir}\}) \tag{4.2}$$

$$\Lambda(A) = inf\{\sum_{j=0}^{\inf} \lambda^r(I_j) : (I_j)_{j \in \mathbb{N}} \text{ of half-open intervals with } A \subseteq \bigcup_{j \in \mathbb{N}} I_j\} \tag{4.3}$$

Miara *Purity* (pl. czystość) jest wykorzystywana jako dodatkowa metoda bezpośredniego porównania kilku frontów Pareto i definiowana jest równaniem 4.4, gdzie r_i^* to liczba punktów należących do i-tego przybliżenia prawdziwego frontu Pareto, a r to rozmiar estymowanego „prawdziwego" frontu Pareto TPF. Dzięki mierze *Purity* możliwe jest porównanie, które z przybliżeń prawdziwego frontu Pareto zawiera największą część niezdominowanych punktów i tym samym dominuje największą część przestrzeni, zatem miara *Purity* powinna być maksymalizowana. Suma wartości miary *Purity* uzyskanych dla różnych frontów Pareto nie musi być równa 1, ponieważ fronty te mogą zawierać te same punkty.

$$P_i = \frac{r_i^*}{r} \tag{4.4}$$

4.4. Pytania badawcze

1. Dla jakich parametrów skuteczność metody CCoDE jest największa?
2. Czy wykorzystanie dla przyjętej reprezentacji ewolucji różnicowej zwiększa skuteczność metody CCoDE ocenianą na podstawie wybranych miar oceny jakości w stosunku do metod opartych na algorytmie ewolucyjnym?
3. Czy między różnymi wariantami metody CCoDE występują istotne statystycznie różnice skuteczności ocenianej na podstawie wybranych miar oceny jakości?
4. Jaka jest skuteczność metody CCoDE oceniana na podstawie wybranych miar oceny jakości w porównaniu do metod referencyjnych? Czy uzyskane różnice skuteczności są istotne statystycznie?
5. Jak dobrze metoda CCoDE przybliża „prawdziwy" Pareto front estymowany jako suma przybliżeń prawdziwego frontu Pareto wygenerowanych przez wszystkie metody?

4.5. Strojenie metod

Dla metod metaheurystycznych nie istnieje jedyny skuteczny sposób doboru wartości parametrów dla konkretnej metody, konkretnego problemu i konkretnej instancji. Dlatego wymagane jest tak zwane „strojenie" metod, czyli dobór wartości parametrów na podstawie eksperymentów. W trakcie strojenia metody jest ona wielokrotnie (min. dziesięć razy) uruchamiana dla tej samej instancji, ale z różnymi kombinacjami wartości parametrów. Analizowana jest uzyskana skuteczność metody i dopiero na tej podstawie ustalane są wartości parametrów do dalszych badań. Dla różnych instancji problemu metoda może osiągać największą skuteczność dla konfiguracji. Oprócz tego, metody często mają wiele parametrów o szerokich zakresach wartości. Z tego powodu strojenie jest procesem bardzo czasochłonnym - na przykład dokonując pełnego eksperymentu czynnikowego (ang. *full factorial design*) dla metody o pięciu parametrach, gdzie każdy parametr może przyjąć dowolną z pięciu różnych wartości, należałoby przetestować $5^5 = 3125$ kombinacji wartości parametrów.

W celu ograniczenia liczby eksperymentów często stosowana jest metoda Taguchi (ang. *Taguchi's method*) oraz jej uzupełniona wersja dla wielu parametrów wynikowych [13,40,67,71]. Metoda Taguchi oparta jest na tablicach ortogonalnych (ang. *orthogonal arrays*), których zastosowanie pozwala na znalezienie wyniku optymalnego zbliżonego do tego uzyskanego z pełnego podejścia czynnikowego przy dużym zmniejszeniu liczby eksperymentów. Metoda Taguchi ma jednak zastosowanie dla eksperymentów, w których występuje jeden parametr wynikowy, a w przypadku optymalizacji wielokryterialnej parametrów wynikowych jest tyle, ile jest wykorzystywanych miar oceny jakości frontów Pareto. Z tego powodu stosuje się metodę Taguchi uzupełnioną o teorię szarych systemów (ang. *grey systems theory*, GST), a w szczególności o proces badania podobieństwa dwóch wektorów - *grey relational analysis*, GRA. Uzupełnienie metody Taguchi o GRA nazywane jest modelem GRA opartym na metodzie Taguchi (ang. *Taguchi-based GRA model*) i umożliwia ograniczenie liczby eksperymentów dla metod i problemów, w których występuje kilka parametrów wynikowych.

Macierz A o rozmiarze $N \times k$ i wartościach ze zbioru S o rozmiarze s jest tablicą (macierzą) ortogonalną o s poziomach i sile t, jeżeli w każdej jej podmacierzy o rozmiarze $N \times t$ każda krotka o rozmiarze t i wartościach ze zbioru S występuje jako wiersz dokładnie taką samą liczbę razy. Taka macierz ortogonalna oznaczana jest jako OA(N, k, s, t). Przykład macierzy ortogonalnej o rozmiarze 4×3, dwóch poziomach i sile równej dwa zaprezentowano w tabeli 4.2. Dla zamieszczonego przykładu OA(8, 5, 2, 2) każda wybrana z drugiego wiersza krotka o rozmiarze dwa - (1,0), (0,0), (0,1) i (1,1) - w podtablicy o rozmiarze 8×2 występuje dokładnie dwa razy. Istnieją archiwa tablic ortogonalnych, zawierające tablice o różnych rozmiarach, różnych liczbie poziomów i różnych wartościach siły - na przykład [4], które wykorzystano do wyboru tablicy ortogonalnej w niniejszej pracy.

W modelu GRA opartym na metodzie Taguchi w pierwszej kolejności należy wybrać odpowiednią tablicę ortogonalną dla zadanych liczby parametrów i liczby poziomów dla parametrów, a następnie należy przeprowadzić eksperymenty wykorzystując kombinacje wartości parametrów dyktowane przez tablicę ortogonalną. Kolejnym krokiem jest normalizacja danych, której sposób wykonania różni się w zależności od pożądanych charakterystyk parametrów wyjściowych. Najczęściej stosowane schematy normalizacji, czyli schematy normalizacji o na-

Tabela 4.2: Przykład tabeli ortogonalnej OA(8, 5, 2, 2), zaadaptowano z [4].

L. p.	Parametr 1	Parametr 2	Parametr 3	Parametr 4	Parametr 5
1	0	0	0	0	0
2	1	0	0	1	1
3	0	1	0	1	0
4	0	0	1	0	1
5	1	1	0	0	1
6	1	0	1	1	0
7	0	1	1	1	1
8	1	1	1	0	0

zwach „im większe tym lepsze" (ang. *„the larger the better"*) oraz „im mniejsze tym lepsze" (ang. *„the smaller the better"*) zaprezentowano w równaniach odpowiednio 4.5 i 4.6, gdzie $i = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$, a m i k to odpowiednio całkowita liczba analizowanych eksperymentów i całkowita liczba obserwacji. Wartości są więc normalizowane względem najmniejszej i największej wartości zadanego parametru wyjściowego spośród wyników dla wszystkich kombinacji wartości parametrów wyjściowych. Istnieją też inne schematy normalizacji, na przykład normalizacja do zadanej wartości celowej. Dla różnych parametrów wyjściowych możliwe jest wykorzystanie różnych schematów normalizacji, zależnie od tego czy wartość parametru powinna być maksymalizowana, czy minimalizowana.

$$x_i(k) = \frac{y_i(k) - min(y_i(k))}{max(y_i(k)) - min(y_i(k))} \tag{4.5}$$

$$x_i(k) = \frac{max(y_i(k)) - y_i(k)}{max(y_i(k)) - min(y_i(k))} \tag{4.6}$$

Po normalizacji danych należy wyznaczyć współczynnik GRC - *grey relational coefficient*. Współczynnik GRC reprezentuje korelację pomiędzy pożądanymi a rzeczywistymi wartościami parametrów wyjściowych. Do jego wyznaczenia wykorzystywana jest idealna (wzorcowa) sekwencja $x_0(k)$ (ang. *ideal, reference sequence*), czyli dla każdego parametru wyjściowego ustalana jest wartość najlepsza - dla wartości znormalizowanych wykorzystywana jest wartość jeden. Równanie 4.7 definiuje współczynnik GRC, gdzie $\Delta_{0i}(k) = |x_0(k) - x_i(k)|$, Δ_{max} oznacza maksymalną wartość $\Delta_{0i}(k)$, a Δ_{min} oznacza minimalną wartość $\Delta_{0i}(k)$. ψ jest parametrem przyjmującym wartości z przedziału [0, 1] i ustalany jest jako $0,5$ jeżeli wszystkie parametry mają równe wagi.

$$\xi_i(k) = \frac{\Delta_{min} + \psi\Delta_{max}}{\Delta_{0i}(k) + \psi\Delta_{max}} \tag{4.7}$$

Po wyznaczeniu współczynnika GRC dla każdej kombinacji wartości parametrów i każdego parametru wyjściowego należy wyznaczyć stopień GRG - *grey relational grade*. Jest on równy średniej arytmetycznej współczynników GRC dla każdego parametru wyjściowego, co przedstawiono w równaniu 4.8. Jeżeli parametry wyjściowe mają różne wagi zamiast średniej arytmetycznej stosowana jest średnia ważona [40]. Następnie w celu wyznaczenia optymalnej kombinacji wartości parametrów wyznaczany jest średni stopień GRG dla każdego poziomu każdego parametru wejściowego. Optymalnym poziomem dla każdego z parametrów jest ten, dla którego średni stopień GRG osiąga największą wartość, a jako optymalna kombinacja wartości parametrów przyjmowane są optymalne poziomy uzyskane dla każdego z nich. Model GRA oparty na metodzie Taguchi pozwala więc na konwersję problemu optymalizacyjnego z wieloma kryteriami (rolę kryteriów pełnią parametry wyjściowe) na problem optymalizacyjny z jednym kryterium (rolę kryteriów pełni stopień GRG).

$$\gamma_i = \frac{1}{n} \sum_{k=1}^{n} \xi_i(k) \tag{4.8}$$

5. Badania

Niniejszy rozdział zawiera wyniki badań, które wykonane zostały zgodnie z założeniami przedstawionymi w rozdziale 4. Celem przeprowadzonych badań jest ocena skuteczności metody CCoDE ocenianej za pomocą wybranych miar oceny jakości oraz w odniesieniu do metody losowej i wybranych metod z literatury - NTGA2 i B-NTGA. Podrozdział 5.1 stanowi ogólne podsumowanie wstępnych badań metody. Wnioski z badań wstępnych pozwoliły na wybór wartości parametrów wykorzystywanych w trakcie strojenia metody oraz jej wariantów, miar i mechanizmów badanych szczegółowo w trakcie właściwej części badawczej. Podsumowaniu strojenia metody poświęcony jest podrozdział 5.2. Strojenie metody pozwoliło na znalezienie takiego zestawu wartości parametrów, dla którego metoda osiąga największą skuteczność. Ten zestaw wartości parametrów wykorzystano w trakcie badań właściwych, których wyniki przedstawiono w podrozdziale 5.3. Podrozdział 5.3 zawiera również weryfikację istotności statystycznej uzyskanych wyników. W podrozdziale 5.4 podsumowano wyniki badań. Wnioskom z wykonanych badań poświęcony jest podrozdział 5.5. Niniejszy rozdział kończy weryfikacja osiągnięcia celu pracy, przedstawiona w podrozdziale 5.6.

5.1. Wyniki badań wstępnych

W pierwszej kolejności sprawdzono, dla jakich operatorów badana metoda osiąga największą skuteczność przy przyjętej reprezentacji. Porównano schemat ewolucji różnicowej (mutacja z ewolucji różnicowej oraz krzyżowanie dwumianowe) ze schematem klasycznego algorytmu ewolucyjnego (mutacja losowa oraz krzyżowanie równomierne). Porównanie wyników uruchomienia metody CCoDE z każdym ze schematów pozwoliło wysunąć wniosek, że dla przyjętej reprezentacji rzeczywistoliczbowej lepiej sprawdza się schemat ewolucji różnicowej.

Następnie zbadano wpływ zastosowania mechanizmu zapobiegania występowaniu klonów na działanie metody CCoDE. Mechanizm zapobiegania występowaniu klonów polega na wykonywaniu mutacji tak długo, jak osobnik nie jest unikatowy. Przyjęto operator mutacji z ewolucji różnicowej. Przy zastosowaniu mechanizmu zapobiegania występowaniu klonów osobniki będące klonami poddawane są stosunkowo dużym zmianom - każdy gen przesuwany jest w stronę innego, losowo wybranego osobnika. Dla bardzo dużej przestrzeni rozwiązań odległości między osobnikami są dużo większe, co dodatkowo zwiększa zmiany, którym poddawane są klony. Z tych powodów mechanizm zapobiegania występowaniu klonów wprowadza zbyt duże losowe zmiany w osobnikach, co powoduje uzyskanie średnio gorszych populacji i co za tym idzie, średnio gorszych frontów Pareto.

Porównano również skuteczność badanej metody przy wykorzystaniu miar HV lub ED do wyboru partnera do współpracy z kilku możliwych kandydatów. Wybrana miara wykorzystywana jest w momencie koewolucyjnej oceny osobników. Przypuszczano, że lepiej sprawdzi się wykorzystanie miary HV. Posiada ona jednak złożoność obliczeniową rosnącą wykładniczo

dla liczby kryteriów, zatem wykorzystanie jej przy większej liczbie kryteriów niż dwa jest zbyt czasochłonne obliczeniowo. Wyniki uruchomień metody CCoDE z miarami HV lub ED potwierdziły jednak przypuszczenie, że dla miary HV uzyskana zostanie większa skuteczność. Ponieważ w niniejszej pracy problem MS-RCPSP rozwiązywany jest z dwoma kryteriami, można zastosować miarę HV do wyboru partnera do współpracy z kilku możliwych kandydatów. Dla większej liczby kryteriów prawdopodobnie należałoby poświęcić większą skuteczność wynikającą z zastosowania miary HV z powodu jej zbyt dużej złożoności obliczeniowej.

W następnej kolejności zbadano wpływ wykorzystania archiwum niezdominowanych osobników w procesie koewolucyjnej oceny osobników. Archiwum stanowi przybliżenie zbioru Pareto, przechowuje więc najlepsze względem kryterium dominacji dotychczasowo znalezione osobniki. Przypuszczano, że wykorzystanie archiwum w momencie wyboru kandydatów na partnera do współpracy pozwoli na osiągnięcie większej skuteczności. W celu weryfikacji tego przypuszczenia porównano skuteczność metody CCoDE w podstawowych i pomocniczych wariantach. Podstawowe warianty wykorzystują archiwum, a pomocnicze warianty nie wykorzystują go. Zaobserwowano uzyskanie większej skuteczności dla metody CCoDE w wariantach podstawowych, czyli wykorzystujących archiwum (CCoDE_R, CCoDE_S i CCoDE_SR). Nie zaobserwowano jednoznacznych różnic między wariantami wchodzącymi w skład jednej grupy - na przykład, nie udało się jednoznacznie ocenić, że metoda CCoDE_R jest gorsza od metody metody CCoDE_S i analogicznie dla metody metody CCoDE_SR oraz wariantów pomocniczych.

Ostatnim badanym w etapie badań wstępnych mechanizmem było nadpisywanie nadmiarowej części genotypu osobników przez odpowiadający jej fragment genotypu partnera do współpracy. W przyjętej reprezentacji każdy osobnik posiada podwójny genotyp - dla osobników z populacji odpowiadającej za proponowanie kolejności zadań, nadmiarowym jest fragment proponujący przypisania zasób-zadanie, a dla osobników z populacji odpowiadającej za proponowanie przypisań zasób-zadanie, nadmiarowy fragment to fragment proponujący kolejność zadań. Nadpisanie u osobników nadmiarowego fragmentu genotypu nie powoduje błędu. Porównano wpływ nadpisywania nadmiarowej części genotypu osobników przez geny pochodzące od partnera do współpracy w stosunku do nie nadpisywania genotypu osobników wcale. Ponieważ wybierany jest najlepszy względem wybranej miary partner do współpracy, nadpisywanie nadmiarowej części genotypu osobników przez pochodzące od niego geny pozwala na poprawienie jakości osobników i w efekcie osiągnięcie większej skuteczności metody CCoDE.

Wnioski ogólne z badań wstępnych:

- Dla przyjętej reprezentacji większą skuteczność autorskiej metody CCoDE pozwala uzyskać zastosowanie mutacji z ewolucji różnicowej oraz krzyżowanie dwumianowe w stosunku do autorskiej metody CCoDE wykorzystującej mutację losową i krzyżowanie równomierne.

- Większą skuteczność autorskiej metody CCoDE pozwala uzyskać wyłączenie mechanizmu zapobiegania występowaniu klonów, który wprowadza zbyt duże losowe zmiany w genotypach osobników-klonów.

- Większą skuteczność autorskiej metody CCoDE pozwala uzyskać wykorzystanie miary HV do wyboru partnera do współpracy z kilku możliwych kandydatów.
- Większą skuteczność autorskiej metody CCoDE pozwala uzyskać wykorzystanie archiwum do wyboru kandydatów na partnera do współpracy, czyli uruchomienie badanej metody w jednym z wariantów podstawowych.
- Większą skuteczność autorskiej metody CCoDE pozwala uzyskać nadpisywanie nadmiarowego fragmentu genotypu osobników przez odpowiadającą mu część genotypu partnera do współpracy.

Podsumowanie metod badanych w trakcie badań właściwych:

- *Random* - losowe generowanie genotypu, uruchamiane liczbę razy równą liczbie urodzeń,
- NTGA2 z parametrami $GenLimit = 1000$, $PopSize = 50$, $P_m = 0,01$, $P_x = 0,6$, $TourSize = 6$ lub $TourSize = 20$ (zależnie od instancji), $gsGen = 50$ [43],
- B-NTGA z parametrami $GenLimit = 1000$, $PopSize = 50$, $P_m = 0,01$, $P_x = 0,9$, $TourSize = 40$ [6, 7],
- MODE - niekoewolucyjna ewolucja różnicowa dla MS-RCPSP inspirowana NTGA2 (wykorzystuje operator selekcji GAP na zmianę z operatorem selekcji turniejowej opartej na dominacji Pareto),
- CCoDE_R - autorska metoda w wariancie z selekcją losową,
- CCoDE_S - autorska metoda w wariancie z selekcją GAP,
- CCoDE_SR - autorska metoda w wariancie z selekcją mieszaną (losową i GAP).

5.2. Wyniki strojenia metod dla wybranych instancji

Strojenia metod dokonano dla pięciu wybranych instancji - 100_5_64_9, 100_10_26_15, 100_20_47_9, 200_40_90_9 i 200_40_91_15. Do badań nad pozostałymi wybranymi instancjami (100_5_22_15, 100_10_64_9, 100_20_65_9, 200_10_128_15 i 200_20_145_15) zastosowano wartości parametrów najczęściej występujące w najlepszych znalezionych konfiguracjach dla poszczególnych metod.

Wartości parametrów metody dostrojono za pomocą modelu GRA opartego na metodzie Taguchi, opisanego szczegółowo w podrozdziale 4.5. Aby zastosować model GRA oparty na metodzie Taguchi w pierwszej kolejności należy wybrać odpowiednią tablicę ortogonalną oraz ustalić poziomy dla każdego z parametrów. Autorska metoda CCoDE posiada osiem parametrów, zatem z biblioteki tablic ortogonalnych [4] wybrano tablicę ortogonalną OA(32, 9, 4, 2) z uwagi na jej stosunkowo małą liczbę wierszy oraz liczbę kolumn większą od liczby parametrów metody. Każdą tablicę ortogonalną OA(N, k, s, t) można zmniejszyć do tablicy OA(N, k', s, t') gdzie $t' = \min[k', t']$, która również jest tablicą ortogonalną [71]. Ta własność

oznacza, że z tablicy ortogonalnej można usunąć jedną lub więcej kolumn i powstająca tablica stanowi tablicę ortogonalną o mniejszej liczbie parametrów. Ponieważ metoda CCoDE posiada osiem parametrów, tablicę ortogonalną OA(32, 9, 4, 2) zmniejszono do OA(32, 8, 4, 2) usuwając ostatnią kolumnę. Uzyskaną tablicę ortogonalną wykorzystaną do strojenia metody przedstawiono w 5.1. Wartości komórek odpowiadają poziomom poszczególnych parametrów.

Tabela 5.1: Wybrana tablica ortogonalna OA(32, 8, 4, 2), zaadaptowano z [4].

L. p.	Parametr 1	Parametr 2	Parametr 3	Parametr 4	Parametr 5	Parametr 6	Parametr 7	Parametr 8
1	1	1	4	4	4	4	2	3
2	1	2	3	1	3	2	4	2
3	1	3	2	3	1	1	3	4
4	1	4	1	2	2	3	1	1
5	2	1	3	3	2	4	4	4
6	2	2	4	2	1	2	2	1
7	2	3	1	4	3	1	1	3
8	2	4	2	1	4	3	3	2
9	3	1	2	4	1	3	4	1
10	3	2	1	1	2	1	2	4
11	3	3	4	3	4	2	1	2
12	3	4	3	2	3	4	3	3
13	4	1	1	3	3	3	2	2
14	4	2	2	2	4	1	4	3
15	4	3	3	4	2	2	3	1
16	4	4	4	1	1	4	1	4
17	1	1	4	2	2	1	3	2
18	1	2	3	3	1	3	1	3
19	1	3	2	1	3	4	2	1
20	1	4	1	4	4	2	4	4
21	2	1	3	1	4	1	1	1
22	2	2	4	4	3	3	3	4
23	2	3	1	2	1	4	4	2
24	2	4	2	3	2	2	2	3
25	3	1	2	2	3	2	1	4
26	3	2	1	3	4	4	3	1
27	3	3	4	1	2	3	4	3
28	3	4	3	4	1	1	2	2
29	4	1	1	1	1	2	3	3
30	4	2	2	4	2	4	1	2
31	4	3	3	2	4	3	2	4
32	4	4	4	3	3	1	4	1

Wykorzystanie tablicy ortogonalnej OA(32, 8, 4, 2) narzuciło ustalenie dla każdego parametru czterech poziomów. Wartości parametrów ustalono na podstawie wyników badań wstępnych.

Głównym ograniczeniem wartości poziomów była szybkość wykonywania metody na procesorze Intel ® Core ™ i7-7700K CPU @ 4,20 GHz - większy rozmiar populacji *PopulationSize*, większa liczba pokoleń *GenerationLimit*, większa liczba kandydatów na partnera do współpracy *PartnerCandidates* oraz większe rozmiary turnieju *GapTournamentSize* i *TournamentSize* powodują dłuższe wykonywanie metody. Wartości poszczególnych poziomów dla poszczególnych parametrów przedstawiono w tabeli 5.2. Po wyborze tablicy ortogonalnej oraz wartości poszczególnych parametrów dla każdego poziomu ustalono 32 konfiguracje i dziesięciokrotnie uruchomiono metodę dla każdej konfiguracji. Metodę MODE uruchamiano za każdym razem z dwa razy większym rozmiarem populacji, aby zachować taką samą liczbę urodzeń i w związku z tym zapewnić sprawiedliwe porównanie. Metody metody CCoDE_R, CCoDE_S i CCoDE_SR wykorzystują dwie populacje o rozmiarze równym parametrowi *PopulationSize*. Jako parametry wyjściowe do modelu GRA opartego na metodzie Taguchi wybrano miary HV, IGD, PFS oraz *Purity*. Miary HV, PFS oraz *Purity* znormalizowano schematem „im większe tym lepsze", a miarę IGD - schematem „im mniejsze tym lepsze". Dla każdej miary i kombinacji wartości parametrów wyznaczono współczynnik GRC, a następnie dla każdego poziomu każdego parametru wejściowego wyznaczono stopień GRG, którego największa wartość wskazuje najlepszy znaleziony poziom dla każdego parametru wejściowego.

Tabela 5.2: Poziomy dla strojenia parametrów za pomocą modelu GRA opartego na metodzie Taguchi.

L. p.	Parametr	Poziom 1	Poziom 2	Poziom 3	Poziom 4
1	*PopulationSize*	200	300	400	500
2	*GenerationLimit*	2000	3000	4000	5000
3	*PartnerCandidates*	2	3	4	5
4	*CR*	0,05	0,1	0,2	0,4
5	*F*	0,4	0,5	0,6	0,7
6	*GapSelectionPercent*	10	25	50	75
7	*GapTournamentSize*	1	5	10	15
8	*TournamentSize*	1	5	10	15

Dla różnych instancji uzyskano różne najlepsze konfiguracje. Dla rozmiaru populacji *PopulationSize* oraz liczby pokoleń *GenerationLimit* najczęściej występującymi najlepszymi znalezionymi wartościami są odpowiednio 500 i 5000. Oznacza to, że nie znaleziono górnej granicy najlepszej znalezionej wartości tych parametrów i jednym z kierunków dalszych prac mogłaby być weryfikacja, czy dla większych wartości metody osiągają większą skuteczność. Co interesujące, nie występuje zależność, że im więcej kandydatów na partnera do współpracy, których liczbę określa parametr *PartnerCandidates*, tym większą skuteczność osiąga metoda. Co również ciekawe, zaobserwowano bardzo małą najlepszą znalezioną wartość współczynnika krzyżowania CR oraz bardzo dużą najlepszą znalezioną wartość siły mutacji F. Siła mutacji F odpowiada za wielkość różnic między niezmutowanymi osobnikami wejściowymi a osobnikiem-mutantem. Bardzo mały współczynnik krzyżowania CR może tłumaczyć fakt nadpisywania nadmiarowego fragmentu genotypu osobnika przez odpowiadającą mu część genotypu partnera do współpracy. Dla małego współczynnika krzyżowania istnieje małe prawdopodobieństwo

wyboru osobnika-mutanta zamiast osobnika rodzicielskiego, czyli promowane są osobniki, których nadmiarowy fragment pochodzi od kandydata na partnera do współpracy, dla którego w wyniku połączenia z oryginalnym osobnikiem wygenerowano harmonogram o największej wartości miary HV. Procent stosowania operatora selekcji GAP mieści się w granicach 50% - 75%. Rozmiar turnieju *TournamentSize* równy jeden oznacza losową selekcję osobnika, co powoduje zmniejszenie ciśnienia selekcyjnego i większą różnorodność rozwiązań. Najlepsze znalezione wartości parametrów dla uruchamianych metod podsumowano w tabeli 5.3.

Tabela 5.3: Najlepsze znalezione konfiguracje dla uruchamianych metod.

Instancja	Metoda	*PopulationSize*	*GenerationLimit*	*PartnerCandidates*	*CR*	*F*	*GapSelectionPercent*	*GapTournamentSize*	*TournamentSize*
100_5_64_9	MODE	400	5000	3	0,1	0,4	50	5	1
	CCoDE_R	400	5000	3	0,1	0,7	75	10	1
	CCoDE_S	500	5000	3	0,1	0,7	75	15	10
	CCoDE_SR	500	5000	4	0,1	0,7	50	10	1
100_10_26_15	MODE	400	5000	3	0,1	0,4	50	15	15
	CCoDE_R	500	5000	3	0,1	0,6	50	10	15
	CCoDE_S	500	5000	5	0,1	0,7	50	15	15
	CCoDE_SR	500	5000	3	0,1	0,7	50	5	1
100_20_40_9	MODE	500	4000	3	0,05	0,7	50	15	15
	CCoDE_R	500	5000	3	0,05	0,7	50	10	5
	CCoDE_S	500	5000	5	0,05	0,7	75	15	10
	CCoDE_SR	500	4000	5	0,05	0,7	50	15	10
200_40_90_9	MODE	300	5000	3	0,05	0,7	50	10	10
	CCoDE_R	500	4000	5	0,05	0,7	50	15	10
	CCoDE_S	500	5000	3	0,05	0,7	75	10	5
	CCoDE_SR	500	5000	3	0,05	0,7	50	15	5
200_40_91_15	MODE	400	5000	3	0,05	0,7	50	10	10
	CCoDE_R	500	5000	5	0,05	0,7	75	5	1
	CCoDE_S	500	5000	5	0,05	0,7	50	10	5
	CCoDE_SR	500	5000	5	0,05	0,6	75	15	15

5.3. Wyniki właściwej części badawczej

Niniejszy podrozdział dzieli się na trzy części. W pierwszej części porównano wartości miar oceny jakości uzyskanych dla przybliżeń prawdziwego frontu Pareto wygenerowanych za pomocą badanych metod - 5.3.1. Wizualizacji uzyskanych dla wybranych instancji przybliżeń prawdziwego frontu Pareto oraz ich porównaniu poświęcony jest podrozdział 5.3.2. W podrozdziale 5.3.3 dokonano weryfikacji istotności statystycznej różnic w wartościach poszczególnych miar oceny jakości, opisanych szczegółowo w podrozdziale 5.3.1.

5.3.1. Porównanie średnich wartości miar oceny jakości

Metody uruchomiono w najlepszych konfiguracjach znalezionych w wyniku strojenia. Uzyskane przybliżenia prawdziwego frontu Pareto oceniono względem „prawdziwego" frontu Pareto estymowanego jako suma wszystkich przybliżeń prawdziwego frontu Pareto wygenerowanych przez różne metody, czyli zbiór wszystkich globalnie niezdominowanych punktów. Porównanie najlepszych znalezionych średnich wartości miar HV, PFS oraz IGD z dziesięciu uruchomień zaprezentowano w tabelach 5.4 - 5.9. Symbolem $\overline{X}$ oznaczono średnią arytmetyczną ze wszystkich dziesięciu uruchomień, a symbolem σ - odchylenie standardowe od wartości średniej. Najlepszą wartość zaznaczono pogrubioną czcionką. Miarę *Purity* wyznaczono dla połączenia dziesięciu przybliżeń prawdziwego frontu Pareto wygenerowanych przez każdą z metod i przedstawiono w tabeli 5.10.

Tabela 5.4: Porównanie uzyskanych dla różnych metod średnich wartości miary HV.

	Random	B-NTGA	NTGA2	MODE	CCoDE_R	CCoDE_S	CCoDE_SR
Instancja	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$
100_5_22_15	0,10015	0,76951	0,7691	**0,77522**	0,73883	0,68939	0,77199
100_5_64_9	0,51479	0,88204	0,87888	0,88596	0,88652	0,87962	**0,88683**
100_10_26_15	0,38727	0,71875	0,71252	0,72904	0,73101	0,73529	**0,73653**
100_10_64_9	0,33236	0,75639	0,74842	**0,76573**	0,75335	0,73586	0,75621
100_20_47_9	0,33413	0,86656	0,85452	**0,87550**	0,78384	0,80822	0,77345
100_20_65_9	0,21079	**0,8201**	0,8021	0,81969	0,74709	0,67737	0,74116
200_10_128_15	0,2757	**0,8465**	0,83535	0,83755	0,78163	0,78119	0,80426
200_20_145_15	0,25948	**0,79514**	0,74784	0,7934	0,74313	0,74283	0,75559
200_40_90_9	0,28823	0,85718	0,78805	**0,8576**	0,78555	0,72235	0,78433
200_40_91_15	0,18787	**0,88224**	0,80164	0,87484	0,78092	0,80185	0,74333
$\overline{X}$	0,28908	0,81944	0,79384	**0,82145**	0,77319	0,7574	0,77537

Miara HV reprezentuje objętość hipersześcianu wyznaczonego przez przybliżenie prawdziwego frontu Pareto oraz *Nadir Point* i w związku z tym jej najlepsze wartości to wartości

największe. Tabela 5.4 zawiera podsumowanie uzyskanych średnich wartości miary HV ze wszystkich dziesięciu uruchomień. Metody MODE i B-NTGA wygenerowały fronty Pareto o najlepszej średniej wartości miary HV dla aż czterech instancji. Dla pozostałych dwóch instancji najlepszą średnią wartość miary HV osiągnęły wyniki działania metody CCoDE_SR. Dla każdej badanej instancji najgorsze średnie wartości miary HV uzyskały fronty Pareto wygenerowane metodą *Random* - dla tej metody również uśredniona dla wszystkich instancji wartość miary HV jest najmniejsza. Najlepszą uśrednioną dla wszystkich instancji wartość miary HV uzyskano dla metody MODE. Dla wszystkich trzech wariantów autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) uśredniona dla wszystkich instancji wartość miary HV jest mniejsza od uśrednionej wartości osiągniętej dla metod NTGA2 oraz B-NTGA. Najmniejsze uśrednione dla wszystkich instancji odchylenie standardowe od wartości średniej uzyskano dla metody B-NTGA, co oznacza większą stabilność tej metody. Wartości odchylenia standardowego zaprezentowano w tabeli 5.5.

Tabela 5.5: Porównanie uzyskanych dla różnych metod wartości odchylenia standardowego od średniej dla miary HV.

	Random	B-NTGA	NTGA2	MODE	CCoDE_R	CCoDE_S	CCoDE_SR
Instancja	σ	σ	σ	σ	σ	σ	σ
100_5_22_15	0,02605	0,00122	0,00335	**0,00079**	0,05776	0,07563	0,01153
100_5_64_9	0,00733	0,00107	0,00313	**0,00053**	0,00129	0,00916	0,00118
100_10_26_15	0,00467	0,0037	0,00552	0,00572	0,00611	**0,00094**	0,00119
100_10_64_9	0,00942	0,00359	0,00509	**0,00174**	0,01280	0,05068	0,01393
100_20_47_9	0,01459	**0,00383**	0,00465	0,00652	0,07618	0,07266	0,0885
100_20_65_9	0,00893	**0,00258**	0,00746	0,00715	0,05901	0,13012	0,05063
200_10_128_15	0,01236	**0,00633**	0,00799	0,00928	0,05935	0,08263	0,05169
200_20_145_15	0,00472	**0,00238**	0,00922	0,00775	0,03736	0,05747	0,04694
200_40_90_9	**0,00432**	0,00485	0,00995	0,01590	0,07825	0,07856	0,08578
200_40_91_15	0,00675	**0,00328**	0,00775	0,0091	0,05264	0,06375	0,14069
$\overline{X}$	0,00991	**0,00328**	0,00641	0,00645	0,04408	0,06216	0,04921

Miara PFS opisuje rozmiar wygenerowanego frontu Pareto, przez co powinna być maksymalizowana. Podsumowanie znalezionych średnich wartości miary PFS ze wszystkich dziesięciu uruchomień znajduje się w tabeli 5.6. Dla aż dziewięciu instancji najlepszą średnią wartość miary PFS uzyskały fronty Pareto wygenerowane metodą MODE. Dla tylko jednej instancji najlepszą średnią wartość miary PFS osiągnęły wyniki działania metody CCoDE_SR. Dla każdej instancji najgorsze średnie wartości miary PFS uzyskały wyniki działania metody *Random*, tak samo jak w przypadku miary HV. Również uśredniona wartość miary PFS dla wszystkich instancji jest najmniejsza dla metody *Random*. Ponownie najlepszą uśrednioną dla wszystkich instancji wartość analizowanej miary osiągnięto dla metody MODE. Inaczej niż dla miary

Tabela 5.6: Porównanie uzyskanych dla różnych metod średnich wartości miary PFS.

Instancja	*Random* $\overline{X}$	B-NTGA $\overline{X}$	NTGA2 $\overline{X}$	MODE $\overline{X}$	CCoDE_R $\overline{X}$	CCoDE_S $\overline{X}$	CCoDE_SR $\overline{X}$
100_5_22_15	10,2	152,7	152,8	**163**	133,1	108,6	152,8
100_5_64_9	19,7	753,4	705,8	**766,8**	763,4	751,4	740,2
100_10_26_15	21,3	241,1	222,4	325,5	307,2	**339,3**	353,5
100_10_64_9	22,6	355,4	170,3	**371,4**	331	294,8	319,3
100_20_47_9	19,5	329,2	136	**498,6**	298,1	224,9	327,6
100_20_65_9	16,3	291,2	131,8	**353,1**	209,7	165,1	233,5
200_10_128_15	14,8	299,8	167,6	**336,5**	231,1	250,8	284,8
200_20_145_15	23,7	271,1	138	**484,4**	228	261,2	313,3
200_40_90_9	16,9	185,9	102,8	**286**	175,9	187,6	199,7
200_40_91_15	17,3	145,3	96	**287,9**	163	158,7	218,9
$\overline{X}$	18,2	302,5	202,4	**387,3**	284,1	274,2	314,4

Tabela 5.7: Porównanie uzyskanych dla różnych metod wartości odchylenia standardowego od średniej dla miary PFS.

Instancja	*Random* σ	B-NTGA σ	NTGA2 σ	MODE σ	CCoDE_R σ	CCoDE_S σ	CCoDE_SR σ
100_5_22_15	**2,4**	4,4	6	8,6	38,8	60,9	25,6
100_5_64_9	**3,5**	13,7	18,6	44	121,5	95,9	132,6
100_10_26_15	**3,2**	9,2	13,9	6,7	32,1	8,8	9,2
100_10_64_9	**5**	25,7	11,7	28,3	60,5	109,5	52,4
100_20_47_9	**4,2**	26,1	15,6	36,3	107,3	29,6	112,3
100_20_65_9	**2,1**	18,6	8,5	50,8	61,8	45,6	61,8
200_10_128_15	**4,1**	10	7,3	23,6	67,7	93,2	92,2
200_20_145_15	**2,6**	30,4	8,9	45,1	13,9	32,4	51,9
200_40_90_9	**2,7**	14,4	6,2	13	23,4	23,3	28,1
200_40_91_15	**4,6**	9,2	7,7	45,2	18,4	26,9	110,6
$\overline{X}$	**3,4**	16,2	10,4	30,2	54,5	52,6	67,7

HV, dla wszystkich wariantów autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) uśredniona dla wszystkich instancji wartość miary PFS jest większa od uśrednionej wartości uzyskanej dla metody NTGA2. Dla metody CCoDE_SR uzyskano uśrednioną dla wszystkich instancji wartość miary PFS większą również od uśrednionej wartości dla metody B-NTGA. Wartości odchylenia standardowego zaprezentowano w tabeli 5.7. Najmniejszą różnorodność rozmiarów frontów Pareto uzyskano dla metody *Random*, niestety dla tej metody znajdowano fronty o średnio najmniejszym rozmiarze.

Tabela 5.8: Porównanie uzyskanych dla różnych metod średnich wartości miary IGD.

	Random	B-NTGA	NTGA2	MODE	CCoDE_R	CCoDE_S	CCoDE_SR
Instancja	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$
100_5_22_15	0,04428	0,00082	0,00083	**0,00058**	0,00446	0,00999	0,00112
100_5_64_9	0,01213	**0,00014**	0,0003	0,00031	0,00059	0,00066	0,00084
100_10_26_15	0,01759	0,00199	0,00258	0,00051	0,0005	0,00023	**0,00019**
100_10_64_9	0,01787	**0,00086**	0,00289	0,0009	0,00219	0,00324	0,0018
100_20_47_9	0,02604	0,00122	0,00628	**0,00069**	0,00716	0,00634	0,00702
100_20_65_9	0,03086	0,00117	0,00811	**0,00104**	0,00775	0,01085	0,00795
200_10_128_15	0,02275	**0,00055**	0,00276	0,00072	0,00504	0,00495	0,00389
200_20_145_15	0,02547	0,00105	0,01028	**0,00089**	0,00742	0,00457	0,00388
200_40_90_9	0,03399	0,00222	0,01458	**0,00158**	0,00864	0,01047	0,00809
200_40_91_15	0,03823	0,00238	0,00995	**0,00168**	0,00758	0,00640	0,00783
$\overline{X}$	0,02692	0,00124	0,00586	**0,00089**	0,00513	0,00577	0,00426

Miara IGD ocenia odległość między punktami przybliżenia prawdziwego frontu Pareto a najbliższymi punktami należącymi do „prawdziwego" frontu Pareto estymowanego jako suma wszystkich przybliżeń. Odległość między przybliżeniem prawdziwego frontu Pareto a prawdziwym frontem Pareto powinna być jak najmniejsza, więc najlepszymi wartościami miary IGD są wartości najmniejsze. Podsumowanie znalezionych średnich wartości miary IGD ze wszystkich dziesięciu uruchomień znajduje się w tabeli 5.8. Dla ponad połowy instancji najlepszą średnią wartość miary IGD uzyskały fronty Pareto wygenerowane metodą MODE. Dla trzech instancji najlepszą średnią wartość miary IGD osiągnęły wyniki uruchomienia metody B-NTGA, a dla tylko jednej instancji - wyniki uruchomienia metody CCoDE_SR. Wyniki działania metody *Random* ponownie osiągnęły najgorsze średnie wartości miary IGD, a także najgorsze uśrednione dla wszystkich instancji wartości miary IGD. Fronty Pareto wygenerowane metodą MODE ponownie uzyskały najlepsze uśrednione dla wszystkich instancji wartości analizowanej miary. Uśrednione dla wszystkich instancji wartości miary IGD są dla wszystkich wariantów autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) mniejsze od uśrednionej wartości znalezionej dla metody NTGA2, ale większe od uśrednionych wartości znalezionych dla metody B-NTGA. Fronty Pareto najmniej różniące się pod względem

wartości miary IGD wygenerowano metodą B-NTGA - wartości odchylenia standardowego zaprezentowano w tabeli 5.9.

Tabela 5.9: Porównanie uzyskanych dla różnych metod wartości odchylenia standardowego od średniej dla miary IGD.

	Random	B-NTGA	NTGA2	MODE	CCoDE_R	CCoDE_S	CCoDE_SR
Instancja	σ	σ	σ	σ	σ	σ	σ
100_5_22_15	0,00178	**0,00009**	0,0002	0,00011	0,00522	0,00823	0,00114
100_5_64_9	0,00066	**0,00003**	0,00016	0,00038	0,00079	0,00084	0,00088
100_10_26_15	0,00093	0,00056	0,00077	0,00028	0,00046	**0,00003**	**0,00003**
100_10_64_9	0,00093	**0,00029**	0,0011	0,00052	0,00158	0,00362	0,00094
100_20_47_9	0,00081	**0,00024**	0,00188	0,00064	0,00495	0,00425	0,00479
100_20_65_9	0,00076	**0,0003**	0,00146	0,00037	0,00328	0,00759	0,0037
200_10_128_15	0,00087	**0,00024**	0,00065	0,00026	0,00352	0,00478	0,00334
200_20_145_15	0,00076	**0,00024**	0,00114	0,00033	0,00247	0,00347	0,00303
200_40_90_9	**0,00055**	0,00076	0,00134	0,00104	0,00362	0,004	0,00446
200_40_91_15	0,0006	0,00063	0,00063	**0,00042**	0,00228	0,00344	0,00718
$\overline{X}$	0,00087	**0,00034**	0,00093	0,00044	0,00282	0,00403	0,00295

Miara *Purity* to stosunek liczby punktów należących do przybliżenia prawdziwego frontu Pareto wygenerowanego przez jedną metodę, które wchodzą w skład „prawdziwego" frontu Pareto, do rozmiaru „prawdziwego" frontu Pareto. Na potrzeby wyznaczenia wartości miary Purity „prawdziwy" front Pareto estymowany jest jako połączenie wszystkich wygenerowanych jego przybliżeń. Większa liczba punktów „prawdziwego" frontu Pareto, które pochodzą z jego przybliżenia wygenerowanego przez daną metodę, świadczy o dominacji większej części przestrzeni przez tę metodę. Najlepsze wartości miary *Purity* to zatem wartości największe. Miarę *Purity* wyznaczono dla połączonych dziesięciu frontów Pareto wygenerowanych przez każdą metodę. Analiza średnich wartości HV, PFS i IGD ze wszystkich uruchomień oraz miary *Purity* wyznaczonej z połączonych dla tych uruchomień przybliżeń prawdziwego frontu Pareto pozwala określić spodziewaną jakość wyników uruchomienia poszczególnych metod. Tabela 5.10 zawiera podsumowanie osiągniętych wartości miary *Purity* połączonych dziesięciu frontów Pareto wygenerowanych przez każdą z metod. Przede wszystkim, zerowe średnie wartości miary *Purity* uzyskane przez metodę *Random* dla każdej instancji oznaczają, że żaden punkt znaleziony przez tę metodę w wyniku jej działania nie znajduje się w ostatecznym „prawdziwym" froncie Pareto. Dla czterech instancji najlepszą średnią wartość miary *Purity* osiągnięto dla przybliżeń frontu Pareto wygenerowanych metodą B-NTGA. Zarówno dla metody MODE, jak i CCoDE_SR uzyskano najlepszą średnią wartość miary *Purity* dla trzech instancji. Najlepszą uśrednioną dla wszystkich instancji wartość analizowanej miary osiągnięto dla metody MODE. Uśrednione dla wszystkich instancji wartości miary *Purity*

wyznaczone dla wyników uruchomienia trzech wariantów autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) są mniejsze od uśrednionej wartości uzyskanej dla metody B-NTGA, ale większe od uśrednionej wartości wyznaczonej dla metody NTGA2.

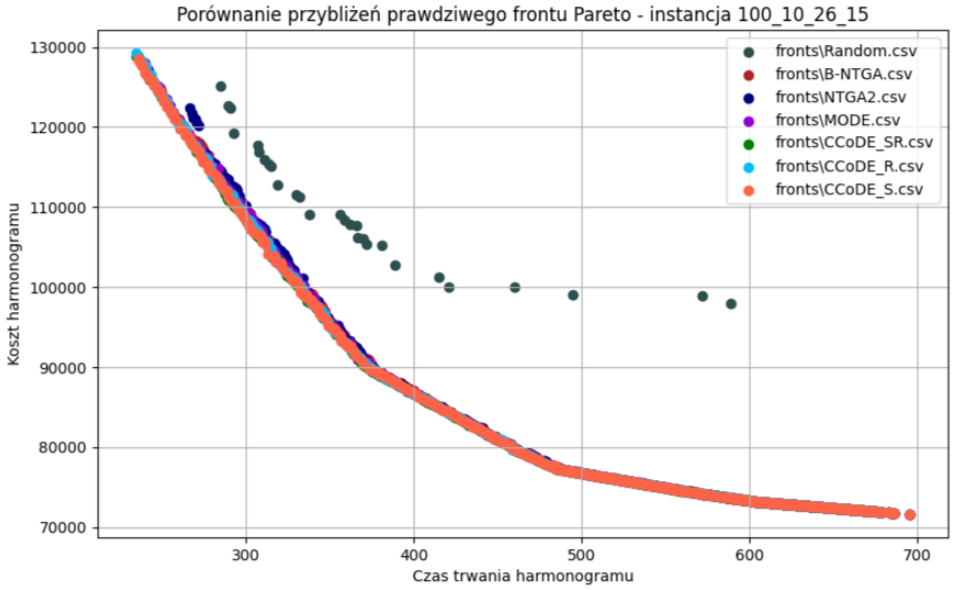
Tabela 5.10: Porównanie uzyskanych dla różnych metod średnich wartości miary *Purity* dla połączonych wyników z dziesięciu uruchomień.

	Random	B-NTGA	NTGA2	MODE	CCoDE_R	CCoDE_S	CCoDE_SR
Instancja	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$	$\overline{X}$
100_5_22_15	0	0,63684	0,65263	0,72632	0,72632	0,66842	**0,77895**
100_5_64_9	0	0,74179	0,74179	0,82446	0,88449	0,81314	**0,92979**
100_10_26_15	0	0,24485	0,25858	0,30206	0,39359	0,49886	**0,60870**
100_10_64_9	0	0,00646	0,01292	**0,52989**	0,15347	0,47173	0,43296
100_20_47_9	0	0,52134	0,02896	**0,55488**	0,04726	0,05335	0,05488
100_20_65_9	0	**0,58824**	0,04934	0,16129	0,16698	0,00380	0,03226
200_10_128_15	0	**0,39501**	0,11227	0,3264	0,02703	0,02287	0,29938
200_20_145_15	0	**0,45735**	0,03448	0,23594	0,08167	0,0363	0,14701
200_40_90_9	0	0,20772	0,11869	**0,59644**	0,0178	0,00297	0,05638
200_40_91_15	0	**0,36015**	0,11494	0,32184	0,08812	0,05364	0,0613
$\overline{X}$	0	0,41598	0,21246	**0,45795**	0,25867	0,26251	0,34016

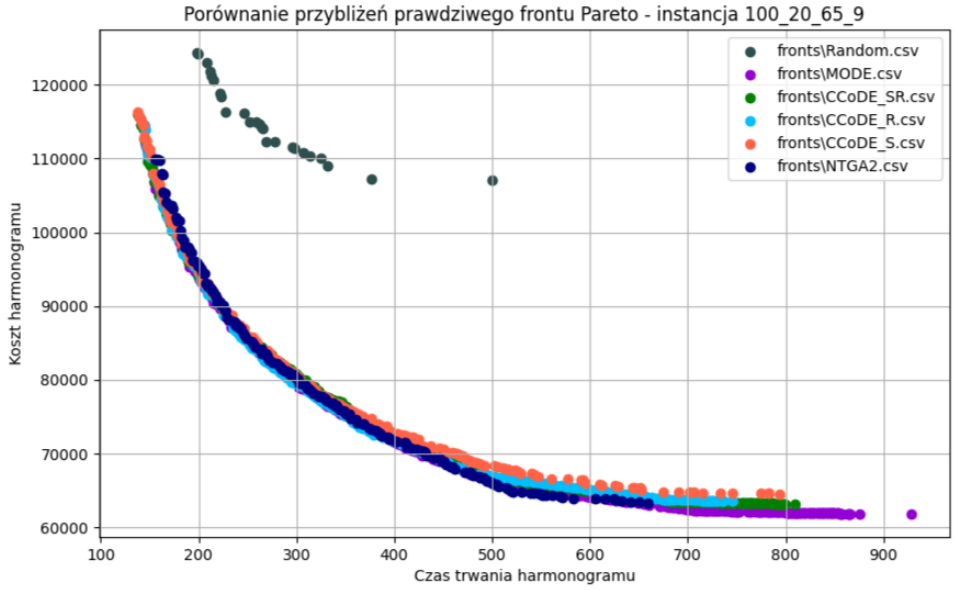
5.3.2. Wizualizacja najlepszych rozwiązań

Na rysunkach 5.1 - 5.5 zaprezentowano zsumowane przybliżenia prawdziwego frontu Pareto wygenerowane przez wybrane metody dla wybranych instancji. W celu porównania uzyskanych frontów Pareto zsumowano dla każdej z metod wszystkie wyniki jej uruchomienia. Wyniki uruchomienia metody *Random* zaprezentowano jedynie dla weryfikacji poprawności działania pozostałych metod. Losowy front Pareto powinien względem pozostałych znajdować się daleko „w tyle", czyli zawierać punkty o większych minimalnych wartościach czasu trwania oraz kosztu harmonogramu - tak właśnie jest dla każdej z badanych instancji.

Dla większości instancji nie zaobserwowano znaczących różnic - jako przykład wybrano instancję 100_10_26_15. Wygenerowane dla tej instancji fronty Pareto zaprezentowano na rysunku 5.1. Nieznaczne różnice można zaobserwować dla lewych „krańców" (minimalny czas trwania harmonogramu i maksymalny jego koszt) frontów Pareto, ale prawe „krańce" (minimalny koszt harmonogramu i maksymalny jego czas trwania) pokrywają się dla każdej metody.



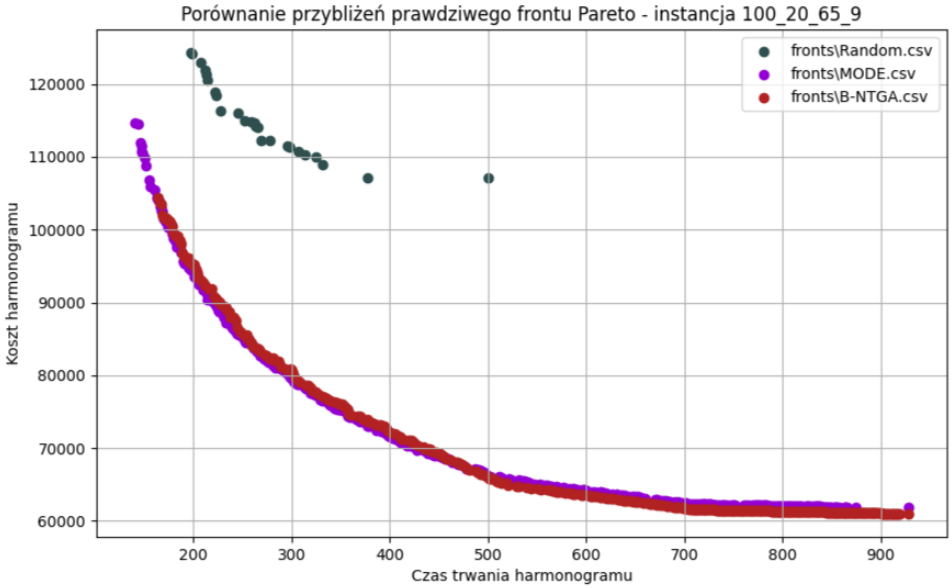
Rysunek 5.1: Przybliżenia prawdziwego frontu Pareto stanowiące wynik uruchomienia poszczególnych metod dla instancji 100_10_26_15.



Rysunek 5.2: Przybliżenia prawdziwego frontu Pareto stanowiące wynik uruchomienia metod CCoDE_R, CCoDE_S, CCoDE_SR, MODE i NTGA2 dla instancji 100_20_65_9.

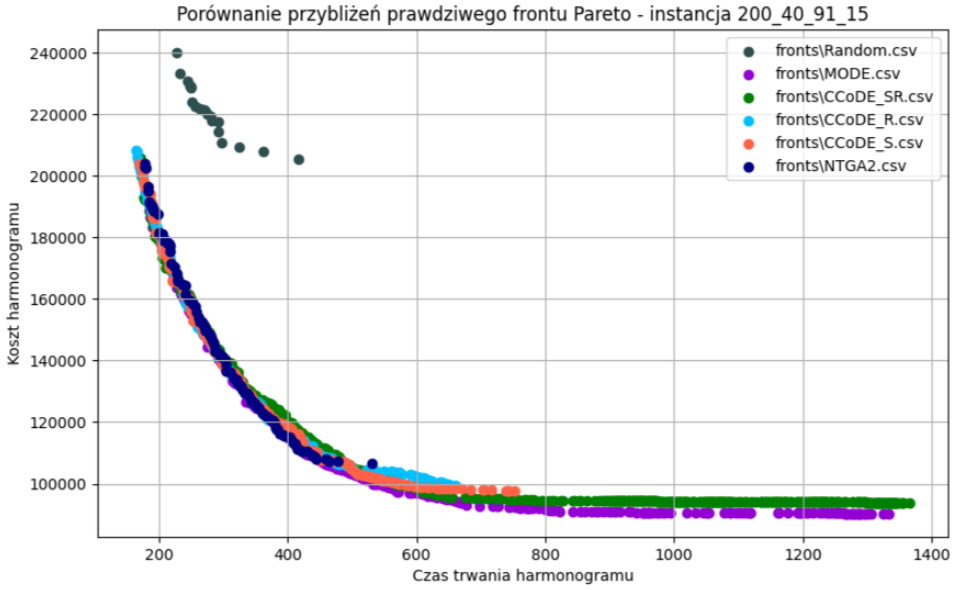
Znaczące różnice we frontach Pareto wygenerowanych przez poszczególne metody można zaobserwować dla instancji 100_20_65_9. Wyniki uruchomienia wszystkich trzech wariantów autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) oraz metod MODE i NTGA2 zaprezentowano na rysunku 5.2, a wyniki uruchomienia metod MODE i B-NTGA na rysunku 5.3. Wszystkie warianty autorskiej metody CCoDE oraz metoda MODE były w stanie znaleźć

harmonogramy o mniejszym czasie trwania niż metody NTGA2 i B-NTGA. Istnieją istotne różnice między frontami Pareto wygenerowanymi przez poszczególne warianty autorskiej metody CCoDE a frontem Pareto wygenerowanym za pomocą metody MODE z korzyścią dla metody MODE, której udało się znaleźć harmonogramy o mniejszym koszcie. Fronty Pareto wygenerowane przez poszczególne warianty autorskiej metody CCoDE są do siebie bardzo zbliżone, a także każdy z nich zawiera więcej punktów o niskim koszcie harmonogramu w porównaniu do wyniku uruchomienia metody NTGA2. Jeżeli chodzi o porównanie wyników uruchomienia metod MODE i B-NTGA, metoda MODE znalazła harmonogramy o mniejszym czasie trwania, podczas gry metoda B-NTGA harmonogramy o nieznacznie mniejszym koszcie.

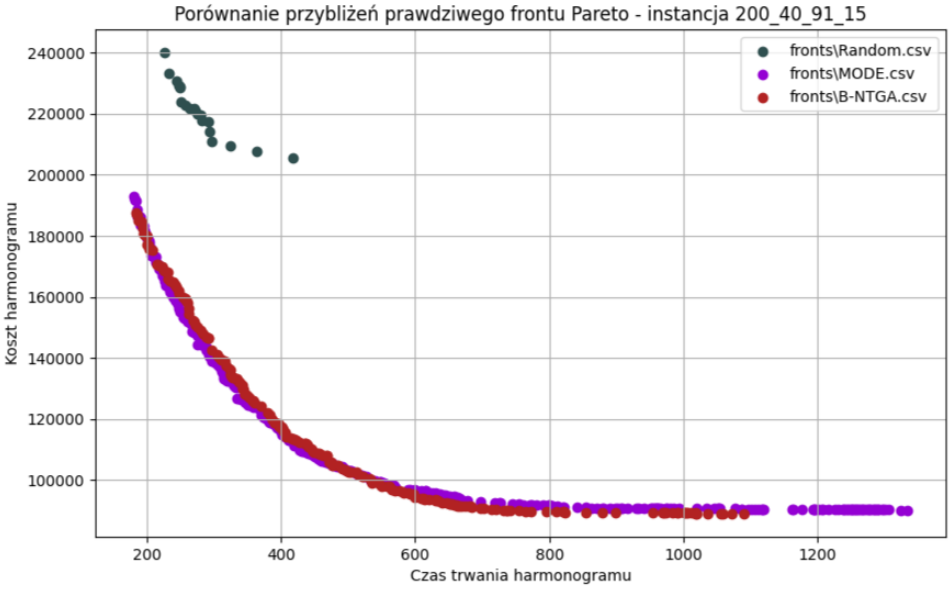


Rysunek 5.3: Przybliżenia prawdziwego frontu Pareto stanowiące wynik uruchomienia metod MODE i B-NTGA dla instancji 100_20_47_9.

Dla instancji 200_40_91_15 również uzyskano znaczne różnice między wynikami uruchomienia poszczególnych metod. Przybliżenia prawdziwego frontu Pareto wygenerowane za pomocą wszystkich trzech wariantów autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) oraz metod MODE i NTGA2 zaprezentowano na rysunku 5.4, a przybliżenia prawdziwego frontu Pareto wygenerowane za pomocą metod MODE i B-NTGA na rysunku 5.5. Tak jak dla instancji 100_20_65_9, wszystkim wariantom autorskiej metody CCoDE oraz metodzie MODE udało się znaleźć harmonogramy o mniejszym czasie trwania niż metodzie NTGA2. Inaczej niż dla instancji 100_20_65_9 jednak, fronty Pareto wygenerowane przez metody CCoDE_SR i MODE są stosunkowo zbliżone. Metodom CCoDE_SR i MODE udało się znaleźć tańsze harmonogramy niż metodom CCoDE_R i CCoDE_S. Fronty Pareto wygenerowane przez metody CCoDE_SR i MODE zawierają również znacznie więcej punktów o niskim koszcie harmonogramu. Metoda B-NTGA ponownie znalazła harmonogramy o nieznacznie mniejszym koszcie niż metoda MODE. Metoda MODE znalazła jednak znacznie większą liczbę punktów o niskim koszcie harmonogramu niż metoda B-NTGA. Lewe „krańce” wszystkich uzyskanych frontów Pareto są dla tej instancji bardzo zbliżone.



Rysunek 5.4: Przybliżenia prawdziwego frontu Pareto stanowiące wynik uruchomienia metod CCoDE_R, CCoDE_S, CCoDE_SR, MODE i NTGA2 dla instancji 200_40_91_15.



Rysunek 5.5: Przybliżenia prawdziwego frontu Pareto stanowiące wynik uruchomienia metod MODE i B-NTGA dla instancji 200_40_91_15.

5.3.3. Weryfikacja istotności statystycznej uzyskanych różnic

Uzyskane różnice między wartościami poszczególnych miar oceny jakości uzyskane dla wyników uruchomienia analizowanych metod zweryfikowano pod kątem istotności statystycznej. Uzyskano próbkę o rozmiarze dziesięć dla każdej z następujących siedmiu metod: *Random*, NTGA2, B-NTGA, MODE, CCoDE_R, CCoDE_S oraz CCoDE_SR. Taki rozmiar próbki jest stosunkowo mały i dla osiągnięcia pełnej wiarygodności wyników testów statystycznych warto wykonać badania na większą skalę, czyli na większej liczbie instancji.

Ponieważ występuje potrzeba zbadania jednocześnie siedmiu grup, zastosowano jeden z testów do porównywania wielu grup aby kontrolować błąd pierwszego rodzaju α (odrzucenie hipotezy zerowej H_0 gdy jest ona w rzeczywistości prawdziwa). Spośród testów statystycznych porównujących wiele grup jednocześnie wybrano nieparametryczny test ANOVA Friedmana z uwagi na zależne pomiary dla każdej z analizowanych miar oceny jakości. Pomiary każdej z analizowanych miar oceny jakości są zależne, ponieważ wyznaczano za każdym razem wartości miar dla tych samych dziesięciu instancji. Test ANOVA Friedmana jest testem oceniającym istotność statystyczną różnic między wieloma grupami jednocześnie. W tym teście hipotezy dotyczą równości średnich rang dla porównywanych populacji. Hipotezy mogą być również upraszczane do median, co zastosowano w definicji hipotez w niniejszej pracy - hipoteza zerowa ma postać $H_0 : \theta_1 = \theta_2 = \cdots = \theta_k$, a hipoteza alternatywna H_1 : nie wszystkie wartości θ_j są sobie równe, gdzie k to liczba grup, $j = 1, 2, \ldots, k$, a $\theta_1, \theta_2, \ldots, \theta_k$ to mediany badanej cechy w kolejnych pomiarach.

Test ANOVA Friedmana porównuje wiele grup jednocześnie, więc może jedynie wskazać, czy istotne statystycznie różnice między tymi grupami występują, czy nie. Z tego powodu wykorzystano nieparametryczny test *post-hoc* Conover-Inman, który może wskazać, między którymi grupami występują istotne statystycznie różnice. W tym teście grupy testowane są parami na zasadzie „każda z każdą". Hipotezy ponownie dotyczą równości średnich rang dla porównywanych populacji lub są upraszczane do median, co zastosowano w niniejszej pracy. W teście *post-hoc* Conover-Inman hipoteza zerowa ma postać $H_0 : \theta_j = \theta_{j+1}$, a hipoteza alternatywna $H_1 : \theta_j \neq \theta_{j+1}$, gdzie $j = 1, 2, \ldots, k-1$, a $\theta_j, \theta_{j+1}, \ldots, \theta_k$ to mediany badanej cechy w kolejnych pomiarach. Na podstawie wartości mediany dla każdej grupy można ocenić, w którym kierunku występuje różnica, jeżeli jest ona istotna statystycznie.

Do wykonania testów statystycznych wykorzystano oprogramowanie PQStat. Dla wszystkich testów statystycznych przyjęto poziom istotności $\alpha = 0,05$.

Miara HV

Wykonano test ANOVA Friedmana. Dla miary HV istnieją istotne statystycznie różnice między grupami, *p-value* $= 0,000001$. Wykonano zatem test *post-hoc* Conover-Inman. Podsumowanie uzyskanych wartości *p-value* dla wybranych par metod zaprezentowano w tabeli 5.11. Pogrubioną czcionką oznaczono wartości *p-value* mniejsze od przyjętego poziomu istotności α. W tabeli 5.12 zaprezentowano wartości mediany miary HV dla każdej z metod, na podstawie których oceniono, która metoda osiąga większe wartości miary HV.

Tabela 5.11: Podsumowanie wartości *p-value* uzyskanych dla wybranych par w teście *post-hoc* Conover-Inman dla miary HV.

	CCoDE_R	CCoDE_S	CCoDE_SR	MODE
Random	**0,000029**	**0,001911**	**0,000001**	**< 0,000001**
B-NTGA	**0,001911**	**0,000029**	**0,026298**	0,626445
NTGA2	0,870987	0,147741	0,418146	**0,000714**
CCoDE_R	-	0,197282	0,331911	**0,000429**
CCoDE_S	0,197282	-	**0,026298**	**0,000005**
CCoDE_SR	0,331911	**0,026298**	-	**0,007587**
MODE	**0,000429**	**0,000005**	**0,007587**	-

Tabela 5.12: Wartości mediany w teście w teście *post-hoc* Conover-Inman dla miary HV.

	θ
Random	0,28197
B-NTGA	0,8333
NTGA2	0,79485
MODE	0,82862
CCoDE_R	0,76714
CCoDE_S	0,73935
CCoDE_SR	0,7641

Dla każdego wariantu autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) oraz metody MODE uzyskano istotne statystycznie różnice w stosunku do metody *Random* z niekorzyścią dla metody *Random* w każdym przypadku. Dla metody MODE uzyskano istotne statystycznie różnice w stosunku do każdego wariantu autorskiej metody CCoDE oraz metody NTGA2 z korzyścią dla metody MODE w każdym przypadku. Uzyskano istotne statystycznie różnice dla metody CCoDE_SR w stosunku do metody CCoDE_S z korzyścią dla metody CCoDE_SR. Dla metody B-NTGA uzyskano istotnie statystyczne różnice w stosunku do każdego z wariantów autorskiej metody CCoDE niestety z korzyścią dla metody B-NTGA. Nie uzyskano istotnej statystycznie różnicy dla metod MODE i B-NTGA.

Miara PFS

Wykonano test ANOVA Friedmana. Dla miary PFS istnieją istotne statystycznie różnice między grupami, *p-value* $< 0,000001$. Wykonano test *post-hoc* Conover-Inman. Tabela 5.13 zawiera podsumowanie uzyskanych wartości *p-value* dla wybranych par metod. Pogrubioną czcionką oznaczono wartości *p-value* mniejsze od przyjętego poziomu istotności α. W tabeli 5.14 zaprezentowano wartości mediany miary PFS dla każdej z metod, na podstawie których oceniono, która metoda osiąga większe wartości miary PFS.

Tabela 5.13: Podsumowanie wartości *p-value* uzyskanych dla wybranych par w teście *post-hoc* Conover-Inman dla miary PFS.

	CCoDE_R	CCoDE_S	CCoDE_SR	MODE
Random	**< 0,000001**	**< 0,000001**	**< 0,000001**	**< 0,000001**
B-NTGA	0,09934	**0,040771**	0,349747	**0,000103**
NTGA2	**0,001067**	**0,00365**	**< 0,000001**	**< 0,000001**
CCoDE_R	-	0,676712	**0,011388**	**< 0,000001**
CCoDE_S	0,676712	-	**0,00365**	**< 0,000001**
CCoDE_SR	**0,011388**	**0,00365**	-	**0,001995**
MODE	**< 0,000001**	**< 0,000001**	**0,001995**	-

Tabela 5.14: Wartości mediany w teście w teście *post-hoc* Conover-Inman dla miary PFS.

	θ
Random	18,4
B-NTGA	281,2
NTGA2	145,4
MODE	344,8
CCoDE_R	229,6
CCoDE_S	237,9
CCoDE_SR	299,1

Dla każdego wariantu autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) oraz metody MODE uzyskano istotne statystycznie różnice w stosunku do metod *Random* i NTGA2 z niekorzyścią dla metod *Random* i NTGA2 w każdym przypadku. Tak jak w przypadku miary HV, dla metody MODE uzyskano istotne statystycznie różnice w stosunku do każdego wariantu autorskiej metody CCoDE oraz metody NTGA2 z korzyścią dla metody MODE w każdym przypadku. Dla metody CCoDE_SR uzyskano istotne statystycznie różnice w stosunku do metod CCoDE_R i CCoDE_S z korzyścią dla metody CCoDE_SR w obu przypadkach. Inaczej niż w przypadku miary HV, tylko dla metod CCoDE_S i B-NTGA uzyskano istotne statystycznie różnice z korzyścią dla metody B-NTGA. Co istotne, uzyskano istotne statystycznie różnice dla metod MODE i B-NTGA z korzyścią dla metody MODE.

Miara IGD

Wykonano test ANOVA Friedmana. Dla miary IGD istnieją istotne statystycznie różnice między grupami, *p-value* $< 0,000001$. Wykonano zatem test *post-hoc* Conover-Inman. Tabela 5.15 przedstawia podsumowanie uzyskanych wartości *p-value* dla wybranych par metod. Pogrubioną czcionką oznaczono wartości *p-value* mniejsze od przyjętego poziomu istotności α. W tabeli 5.16 zaprezentowano wartości mediany miary IGD dla każdej z metod, na podstawie których oceniono, która metoda osiąga mniejsze wartości miary IGD.

Tabela 5.15: Podsumowanie wartości *p-value* uzyskanych dla wybranych par w teście *post-hoc* Conover-Inman dla miary IGD.

	CCoDE_R	CCoDE_S	CCoDE_SR	MODE
Random	**0,000033**	**0,000107**	**0,000001**	**< 0,000001**
B-NTGA	**0,000107**	**0,000033**	**0,002777**	0,603441
NTGA2	0,862379	0,862379	0,228057	**0,00001**
CCoDE_R	-	0,728931	0,300654	**0,000018**
CCoDE_S	0,728931	-	0,169207	**0,000005**
CCoDE_SR	0,300654	0,169207	-	**0,000578**
MODE	**0,000018**	**0,000005**	**0,000578**	-

Tabela 5.16: Wartości mediany w teście w teście *post-hoc* Conover-Inman dla miary IGD.

	θ
Random	0,02576
B-NTGA	0,00111
NTGA2	0,00459
MODE	0,00081
CCoDE_R	0,0061
CCoDE_S	0,00565
CCoDE_SR	0,00389

Dla każdego wariantu autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) oraz metody MODE uzyskano istotne statystycznie różnice w stosunku do metody *Random* z niekorzyścią dla metody *Random* w każdym przypadku. Dla metody MODE uzyskano istotne statystycznie różnice w stosunku do każdego wariantu autorskiej metody CCoDE oraz metody NTGA2 z korzyścią dla metody MODE w każdym przypadku. Dla metody B-NTGA uzyskano istotnie statystyczne różnice w stosunku do każdego z wariantów autorskiej metody CCoDE niestety z korzyścią dla metody B-NTGA. Nie uzyskano istotnej statystycznie różnicy dla metod MODE i B-NTGA.

Miara *Purity*

Wykonano test ANOVA Friedmana. Dla miary *Purity* istotne statystycznie różnice między grupami, *p-value* $= 0,000013$. Wykonano test *post-hoc* Conover-Inman. Podsumowanie uzyskanych wartości *p-value* dla wybranych par metod zaprezentowano w tabeli 5.17. Pogrubioną czcionką oznaczono wartości *p-value* mniejsze od przyjętego poziomu istotności α. W tabeli 5.18 zaprezentowano wartości mediany miary *Purity* dla każdej z metod, na podstawie których oceniono, która metoda osiąga większe wartości miary *Purity*.

Tabela 5.17: Podsumowanie wartości *p-value* uzyskanych dla wybranych par w teście *post-hoc* Conover-Inman dla miary *Purity*.

	CCoDE_R	CCoDE_S	CCoDE_SR	MODE
Random	**0,00001**	**0,000615**	**< 0,000001**	**< 0,000001**
B-NTGA	0,469997	0,054622	0,717425	0,151399
NTGA2	0,151399	0,828032	**0,013761**	**0,000615**
CCoDE_R	-	0,221465	0,279938	**0,033416**
CCoDE_S	0,221465	-	**0,023668**	**0,0012**
CCoDE_SR	0,279938	**0,023668**	-	0,279938
MODE	**0,033416**	**0,0012**	0,279938	-

Tabela 5.18: Wartości mediany w teście w teście *post-hoc* Conover-Inman dla miary *Purity*.

	θ
Random	0
B-NTGA	0,42618
NTGA2	0,11361
MODE	0,42815
CCoDE_R	0,1208
CCoDE_S	0,0535
CCoDE_SR	0,2232

Dla każdego wariantu autorskiej metody (CCoDE_R, CCoDE_S i CCoDE_SR) oraz metody MODE uzyskano istotne statystycznie różnice w stosunku do metody *Random* z niekorzyścią dla metody *Random* w każdym przypadku. Jedynie dla metod CCoDE_SR i MODE uzyskano istotne statystycznie różnice w stosunku do metody NTGA2 z niekorzyścią dla metody NTGA2 w każdym przypadku. Dla metod CCoDE_SR i MODE uzyskano istotne statystycznie różnice w stosunku do metody CCoDE_S z niekorzyścią dla metody CCoDE_S w każdym przypadku. Dla metody MODE uzyskano również istotne statystycznie różnice w stosunku do metody CCoDE_R z niekorzyścią dla CCoDE_R. Nie uzyskano istotnej statystycznie różnicy między żadnym z wariantów autorskiej metody CCoDE oraz metodą MODE a metodą B-NTGA.

5.4. Podsumowanie wyników badań

Dla badanych metod CCoDE oraz MODE uzyskano istotne statystycznie różnice w średnich wartościach wybranych miar oceny jakości w stosunku do metody *Random*. Uzyskane różnice są niekorzystne dla metody losowej, czyli wyniki jej uruchomienia uzyskały istotnie mniejsze średnie wartości miar HV, PFS i *Purity* oraz istotnie większą średnią wartość miary IGD.

Za najlepszą z badanych metod uznana została metoda MODE, która wygenerowała fronty Pareto o najlepszych wartościach średnich dla każdej z wybranych miar oceny jakości -

największe średnie wartości miar HV, PFS i *Purity* oraz najmniejszą średnią wartość miary IGD. Dla każdej z wybranych miar jakości różnice między metodą MODE oraz pozostałymi metodami są istotne statystycznie z korzyścią dla metody MODE, za wyjątkiem metody CCoDE_SR dla miary *Purity* oraz metody B-NTGA dla miar HV, IGD oraz *Purity*. Dla tych miar między metodą MODE a metodami CCoDE_SR i B-NTGA nie wykryto istotnej statystycznie różnicy.

Różne warianty autorskiej metody (CCoDE_R, CCoDE_S oraz CCoDE_SR) osiągnęły bardzo podobną skuteczność ocenianą na podstawie wybranych miar oceny jakości. Przy weryfikacji istotności statystycznej uzyskanych różnic metoda CCoDE_SR okazała się jednak osiągać istotnie większe wartości miar HV i Purity w stosunku do metody CCoDE_S oraz istotnie większe wartości miary PFS w stosunku do metod CCoDE_S i CCoDE_R. Nie zaobserwowano znaczącej różnicy w skuteczności metod CCoDE_R oraz CCoDE_S ocenianej na podstawie wybranych miar oceny jakości. Również zbliżone wartości wybranych miar oceny jakości uzyskały wszystkie warianty autorskiej metody (CCoDE_R, CCoDE_S oraz CCoDE_SR) oraz metoda NTGA2. Dla miar HV oraz IGD nie zaobserwowano istotnej różnicy między uzyskanymi dla tych metod wynikami. Wszystkim wariantom autorskiej metody CCoDE udało się jednak uzyskać istotnie większe wartości miary PFS w stosunku do metody NTGA2. Metodzie CCoDE_SR udało się również uzyskać istotnie większą wartość miary *Purity* w stosunku do metody NTGA2. Przy porównaniu różnych wariantów autorskiej metody CCoDE z metodą B-NTGA, autorska metoda osiąga mniejszą skuteczność. Metoda B-NTGA osiągnęła istotnie większe wartości miary HV oraz istotnie mniejsze wartości miary IGD od każdego z wariantów autorskiej metody. Dla miary *Purity* nie uzyskano istotnych statystycznie różnic, a dla miary PFS istotnie większą wartość również uzyskała metoda B-NTGA.

Dla wielu z badanych instancji przedstawione na wykresach przybliżenia prawdziwego frontu Pareto wygenerowane przez każdą z metod pokrywają się lub nie różnią się znacznie. Dla instancji, w których występowały różnice, największe różnice zaobserwowano na „krańcach" frontów Pareto, czyli dla harmonogramów o najkrótszym czasie trwania i jednocześnie największym koszcie oraz dla harmonogramów o najdłuższym czasie trwania i jednocześnie najmniejszym koszcie. Zaobserwowano bardzo zbliżone fronty Pareto wygenerowane przez metody MODE i B-NTGA. Skuteczność każdego z wariantów autorskiej metody CCoDE mocno zależy od rozwiązywanej instancji problemu.

5.5. Wnioski

Autorska metoda CCoDE jest metodą koewolucyjną, czyli operuje jednocześnie na dwóch populacjach. Opracowano kilka wariantów metody, które różnią się sposobem wyboru osobników z obu populacji, które wchodzą ze sobą w interakcje. Spośród badanych wariantów nie udało się wyłonić jednoznacznie najbardziej skutecznego - dla większości wybranych miar oceny jakości nie uzyskano istotnych różnic. Osiągnięto istotnie większą skuteczność każdego wariantu metody CCoDE w stosunku do metody losowej (*Random*). Zaobserwowano również istotnie większe wartości miary PFS dla każdego wariantu metody CCoDE w stosunku do metody NTGA2, a także istotnie większą wartość miary *Purity* dla metody

CCoDE_SR w stosunku do metody NTGA2. Zastosowanie inspiracji koewolucją pozwala więc na skuteczne rozwiązywanie problemu MS-RCPSP z kryterium czasu i kosztu. Nie zaobserwowano jednak większej skuteczności metody koewolucyjnej w porównaniu do metody niekoewolucyjnej MODE - przeciwnie, dla większości wybranych miar jakości istotnie lepsze wartości osiąga metoda MODE. Może to wynikać z przyjętego schematu koewolucyjnego, w szczególności wykorzystania miary HV do wyboru „najlepszego" spośród kandydatów na partnera do współpracy. Wstępnie przetestowano wykorzystanie do tego celu również miary ED, dla której uzyskano jeszcze mniejszą skuteczność. Przy rozwiązywaniu problemów wielokryterialnych, próba transformacji problemu na problem jednokryterialny powoduje pominięcie wielu rozwiązań, co może tłumaczyć mniejszą skuteczność metody koewolucyjnej w stosunku do metody niekoewolucyjnej. Z uwagi na dużą liczbę osobników i dużą liczbę interakcji między nimi, kolejną wadą metody koewolucyjnej CCoDE jest również bardzo duża czasochłonność wykonywania.

Zaobserwowano wzrost skuteczności autorskiej metody CCoDE przy wykorzystaniu reprezentantów archiwum osobników niezdominowanych w momencie koewolucyjnej oceny. W przypadku metody CCoDE archiwum jest wykorzystywane w momencie oceny osobnika - część kandydatów na partnera do współpracy to osobniki pochodzące z archiwum. Archiwum jest aktualnym przybliżeniem zbioru Pareto i jako takie przechowuje najlepsze znalezione do tej pory rozwiązania. Aktywne włączenie archiwum osobników niezdominowanych może więc powodować wzrost skuteczności metody.

Zaobserwowano również większą skuteczność autorskiej metody CCoDE gdy, oprócz interakcji między populacjami w postaci współpracy w momencie oceny, występował także pewien rodzaj krzyżowania między populacjami. Dla podwójnej reprezentacji wykorzystanej w autorskiej metodzie nadpisanie nadmiarowego fragmentu genotypu osobnika nie powoduje zakłócenia procesu ewolucyjnego poprzez utratę genów. Ponieważ autorska metoda CCoDE wykorzystuje momentami niezależną ocenę osobnika bez wykorzystania reprezentanta drugiego gatunku, nadpisanie nadmiarowego fragmentu genotypu osobnika może powodować zwiększenie jego wartości przystosowania wyznaczanej niezależnie, tj. poprawę jego jakości. Dopuszczenie mieszania się genów między populacjami przyczyniło się w przypadku metody CCoDE do poprawy jej skuteczności.

Niekoewolucyjna ewolucja różnicowa inspirowana metodami NTGA2, DEGR i GaMeDE2 o nazwie MODE okazała się skuteczną metodą rozwiązywania problemu MS-RCPSP z kryterium czasu i kosztu. Zaobserwowano istotnie większą skuteczność względem metody losowej (*Random*) oraz metody NTGA2. Metoda MODE znajdowała również istotnie większe przybliżenia frontu Pareto od nowatorskiej metody B-NTGA oraz nie wykazała istotnie mniejszej skuteczności ocenianej na podstawie żadnej z pozostałych wybranych miar oceny jakości. Metoda MODE była także w stanie znaleźć punkty o lepszych wartościach jednego z kryteriów, jak również znaleźć większą liczbę niezdominowanych punktów w stosunku do metody B-NTGA. Duża skuteczność metody MODE wskazuje na duży potencjał metod opartych na ewolucji różnicowej w rozwiązywaniu problemu MS-RCPSP w porównaniu do metod, które nie stosują rzeczywistoliczbowej reprezentacji i schematu mutacji i krzyżowania właściwego dla ewolucji różnicowej.

Istotnym jest jednak aby podkreślić, że badane metody CCoDE i MODE otrzymały znacznie większy budżet od wybranych metod z literatury (200000 - 2500000 urodzeń dla metod CCoDE i MODE, 50000 urodzeń dla metod NTGA2 i B-NTGA). Możliwe jest, że metody NTGA2 i B-NTGA uzyskałyby większą skuteczność dla większego budżetu.

5.6. Weryfikacja osiągnięcia celu pracy

Celem pracy było zbadanie, czy zastosowanie inspiracji koewolucją w metodach opartych na algorytmie ewolucyjnym zwiększa skuteczność w rozwiązywaniu wielokryterialnego problemu MS-RCPSP. Zdefiniowano również sześć celów pośrednich, których realizacja składała się na realizację celu pracy. Cele pośrednie to między innymi implementacja przyjętej reprezentacji problemu MS-RCPSP, opracowanie i implementacja autorskiej metody wykorzystującej inspirację zjawiskiem koewolucji, strojenie metod oraz wykonanie badań i analiza uzyskanych wyników.

Opracowano autorską metodę o nazwie CCoDE w kilku wariantach. Dla każdej z wybranych instancji wzorcowych wielokrotnie uruchomiono każdą metodę, oceniono skuteczność metod na podstawie średnich wartości czterech wybranych miar oceny jakości wygenerowanych przez nie wyników oraz zweryfikowano istotność statystyczną uzyskanych różnic. Zidentyfikowano najbardziej skuteczną metodę - jest to niekoewolucyjna metoda oparta na ewolucji różnicowej o nazwie MODE. Zastosowanie inspiracji koewolucją w metodzie CCoDE opartej na ewolucji różnicowej pozwoliło uzyskać większą skuteczność ocenianą na podstawie wybranych miar oceny jakości w stosunku do metody losowej, ale nie pozwoliło uzyskać większej skuteczności od analogicznej metodzie opartej na ewolucji różnicowej, w której koewolucja nie została zastosowana. Stanowi to uzasadnienie osiągnięcia celu pracy.

6. Zakończenie

W niniejszej pracy dokonano badań nad skutecznością opracowanej na podstawie przeglądu literatury koewolucyjnej metody dla rozwiązywania wybranego silnie ograniczonego i wielokryterialnego problemu harmonogramowania. Dokonano próby uogólnienia metody osiągającej dużą skuteczność w rozwiązywaniu wybranego problemu harmonogramowania z jednym kryterium (ważoną funkcją fitness) na wiele kryteriów. Praca zawiera aspekty innowacyjne - po raz pierwszy zastosowano metodę koewolucyjną do rozwiązywania problemu MS-RCPSP z wieloma kryteriami, stosując podejście oparte na relacji dominacji Pareto. We wcześniejszych pracach problem MS-RCPSP rozwiązywano dla wielu kryteriów metodami ewolucyjnymi, nie wykorzystującymi inspiracji koewolucją, lub metodami koewolucyjnymi, ale dla tylko jednego kryterium. Elementem nie spotykanym wcześniej w literaturze jest również wprowadzanie mieszania się genów pomiędzy populacjami, które okazało się zwiększać skuteczność metody ocenianej na podstawie wybranych miar oceny jakości. Opracowana metoda CCoDE korzysta także w sposób aktywny z archiwum osobników niezdominowanych, co pozwala na zwiększenie skuteczności związane z wykorzystaniem w procesie ewolucyjnym najlepszych dotychczas znalezionych rozwiązań. Metoda CCoDE ma charakter uniwersalny i może zostać zastosowana dla każdego problemu, który można rozwiązywać modelem koewolucji kooperacyjnej. Jednym z przyszłych kierunków badań jest ocena skuteczności metody CCoDE na innych problemach, na przykład problemie TTP.

Z zastosowaniem koewolucyjnego modelu do rozwiązywania problemów obliczeniowych wiąże się jednak wiele wyzwań. Przede wszystkim, między koewoluującymi gatunkami występuje ogromna liczba interakcji, co znacznie zwiększa czasochłonność wykonywania metody. Kolejnym wyzwaniem jest określenie sposobu wyboru reprezentanta drugiego gatunku w momencie oceny osobnika należącego do pierwszego gatunku. W niniejszej pracy zastosowano do tego celu miarę HV. Taki schemat koewolucyjny jednak nie sprawdził się, ponieważ nie udało się osiągnąć skuteczności większej od analogicznej metody nie wykorzystującej koewolucji (MODE). Pierwszym kierunkiem rozwoju metody jest więc opracowanie alternatywnego sposobu wyboru partnera do współpracy spośród kilku kandydatów. Autorska metoda koewolucyjna CCoDE i jej siostrzana metoda MODE są oparte na ewolucji różnicowej. Spośród wszystkich badanych metod, włącznie z metodami z literatury, największą skuteczność ocenianą na podstawie wybranych miar jakości osiągnęła metoda MODE. Wskazuje to duży potencjał ewolucji różnicowej w rozwiązywaniu problemów optymalizacyjnych. Warta zbadania jest na przykład skuteczność metod NTGA2 i B-NTGA wykorzystujących reprezentację rzeczywistoliczbową i schemat mutacji i krzyżowania właściwy dla ewolucji różnicowej.

Wybrany problem MS-RCPSP, ale też inne warianty problemu harmonogramowania, mają dużą wagę z biznesowego punktu widzenia z uwagi na bardzo ważną rolę harmonogramowania w efektywnym zarządzaniu projektami. Problemy świata rzeczywistego są często zbyt złożone, by były efektywnie algorytmizowane. Dla rozwiązywania takich problemów mają zastosowanie metaheurystyki, takie jak zaprezentowane w niniejszej pracy metody CCoDE i MODE.

Bibliografia

[1] Definicja terminu harmonogramowanie w Słowniku Języka Polskiego. https://sjp.pl/harmonogramowanie. Dostęp 20.04.2024.

[2] Strona główna portalu iMOPSE. http://imopse.ii.pwr.wroc.pl/. Dostęp 16.04.2024.

[3] Repozytorium biblioteki iMOPSE na portalu GitHub. https://github.com/imopse/iMOPSE. Dostęp 07.02.2024.

[4] Strona główna biblioteki tablic ortogonalnych. http://neilsloane.com/oadir/. Dostęp 19.04.2024.

[5] R. Akbari, V. Zeighami, K. Ziarati. MLGA: A Multilevel Cooperative Genetic Algorithm. *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, strony 271–277, 2010.

[6] M. Antkiewicz, P. B. Myszkowski. Balancing Pareto Front exploration of Non-dominated Tournament Genetic Algorithm (B-NTGA) in solving multi-objective NP-hard problems with constraints. *Information Sciences*, 667:120400, 2024.

[7] M. Antkiewicz, P. B. Myszkowski. Corrigendum to "Balancing Pareto Front exploration of Non-dominated Tournament Genetic Algorithm (B-NTGA) in solving multi-objective NP-hard problems with constraints" [Inform. Sci., 667 (2024) 120400]. *Information Sciences*, 670:120634, 2024.

[8] M. Antkiewicz, P. B. Myszkowski, M. Laszczyk. GaMeDE2 — improved Gap–based Memetic Differential Evolution applied to Multimodal Optimization. *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, strony 291–300, 2022.

[9] M. Bacharach. *Zero-sum Games*, strony 727–731. Palgrave Macmillan UK, 1991.

[10] M. R. Bonyadi, Z. Michalewicz, L. Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. *2013 IEEE Congress on Evolutionary Computation*, strony 1037–1044, 2013.

[11] J. Błażewicz, J. K. Lenstra, A. H. G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. *Introduction To Algorithms*. MIT Press, wydanie 2, 2001.

[13] S. Datta, A. Bandyopadhyay, P. K. Pal. Grey-based taguchi method for optimization of bead geometry in submerged arc bead-on-plate welding. *The International Journal of Advanced Manufacturing Technology*, 39(11):1136–1143, 2008.

[14] K. De Jong. Co-Evolutionary Algorithms: A Useful Computational Abstraction? *Search-Based Software Engineering*, strony 3–11, 2015.

[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[16] J. J. Durillo, A. J. Nebro. jMetal: A Java framework for multi-objective optimisation. *Advances in Engineering Software*, 42(10):760–771, 2011.

[17] A. E. Eiben, J. Smith. *Introduction To Evolutionary Computing*, wolumen 45. Springer Berlin, Heidelberg, 2003.

[18] M. Erickson, A. Mayer, J. Horn. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. *Evolutionary Multi-Criterion Optimization*, strony 681–695. Springer Berlin Heidelberg, 2001.

[19] C. Fonseca, P. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. *Proceedings of the ICGA-93: Fifth International Conference on Genetic Algorithms*, wolumen 93, strony 416–423, 1993.

[20] H. Ge, M. Zhao, Y. Hou, Z. Kai, L. Sun, G. Tan, Q. Zhang, C. L. P. Chen. Bi-space Interactive Cooperative Coevolutionary algorithm for large scale black-box optimisation. *Applied Soft Computing*, 97:106798, 2020.

[21] C. Goh, K. C. Tan, D. Liu, S. C. Chiam. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, 202(1):42–54, 2010.

[22] S. Hartmann, D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

[23] S. Hartmann, D. Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297(1):1–14, 2022.

[24] S. Hartmann, R. Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407, 2000.

[25] J. Horn, N. Nafpliotis, D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, wolumen 1, strony 82 – 87, 1994.

[26] A. Isaac, H. K. Nehemiah, S. D. Dunston, V. R. E. Christo, A. C. Kannan. Feature selection using competitive coevolution of bio-inspired algorithms for the diagnosis of pulmonary emphysema. *Biomedical Signal Processing and Control*, 72:103340, 2022.

[27] B. Ismail, N. I. Abdul Wahab, M. L. Othman, M. A. M. Radzi, K. Naidu Vijyakumar, M. N. Mat Naain. A Comprehensive Review on Optimal Location and Sizing of Reactive Power Compensation Using Hybrid-Based Approaches for Power Loss Reduction, Voltage Stability Improvement, Voltage Profile Enhancement and Loadability Enhancement. *IEEE Access*, 8:222733–222765, 2020.

[28] P. Jędrzejowicz, E. Ratajczak-Ropel. A-Team Solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Procedia Computer Science*, 207:3300–3309, 2022.

[29] F. Kang, J. Li, Q. Xu. Virus coevolution partheno-genetic algorithms for optimal sensor placement. *Advanced Engineering Informatics*, 22(3):362–370, 2008.

[30] D. Kar, T. H. Nguyen, F. Fang, M. Brown, A. Sinha, M. Tambe, A. X. Jiang. *Trends and Applications in Stackelberg Security Games*. Springer International Publishing, 2018.

[31] R. Kicinger, T. Arciszewski, K. De Jong. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23):1943–1978, 2005.

[32] J. D. Knowles, L. Thiele, E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Raport instytutowy, ETH Zurich, 2006.

[33] R. Kolisch, S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006.

[34] M. Laszczyk, P. B. Myszkowski. Improved selection in evolutionary multi–objective optimisation of multi–skill resource–constrained project scheduling problem. *Information Sciences*, 481:412–431, 2019.

[35] K. Li, S. Kwong, K. Deb. A dual-population paradigm for evolutionary multiobjective optimisation. *Information Sciences*, 309:50–72, 2015.

[36] X. Li, Y. Yu, M. Huang. Multi-objective cooperative coevolution algorithm with a Master–Slave mechanism for Seru Production. *Applied Soft Computing*, 119:108593, 2022.

[37] Y. Y. Li, J. Lin, Z. J. Wang. Multi-skill resource constrained project scheduling using a multi-objective discrete Jaya algorithm. *Applied Intelligence*, 52(5):5718–5738, 2022.

[38] C. Lin, A. Qing, Q. Feng. A comparative study of crossover in differential evolution. *Journal of Heuristics*, 17(6):675–703, 2011.

[39] J. Lin, L. Zhu, K. Gao. A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications*, 140:112915, 2020.

[40] S. S. Lin, M. T. Chuang, J. L. Wen, Y. K. Yang. Optimization of 6061T6 CNC Boring Process Using the Taguchi Method and Grey Relational Analysis. *The Open Industrial & Manufacturing Engineering Journal*, 2:14–20, 04 2009.

[41] G. Mauša, T. Galinac Grbac. Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study. *Applied Soft Computing*, 55:331–351, 2017.

[42] P. B. Myszkowski, M. Laszczyk. Survey of quality measures for multi-objective optimisation: Construction of complementary set of multi-objective quality measures. *Swarm and Evolutionary Computation*, 48:109133, 2019.

[43] P. B. Myszkowski, M. Laszczyk. Diversity based selection for many-objective evolutionary optimisation problems with constraints. *Information Sciences*, 546:665–700, 2021.

[44] P. B. Myszkowski, M. Laszczyk. Investigation of benchmark dataset for many-objective Multi-Skill Resource Constrained Project Scheduling Problem. *Applied Soft Computing*, 127:109253, 2022.

[45] P. B. Myszkowski, M. Laszczyk, D. Kalinowski. Co-Evolutionary Algorithm solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems*, wolumen 11, strona 75–82, 2017.

[46] P. B. Myszkowski, M. Laszczyk, J. Lichodij. Efficient selection operators in NSGA-II for solving bi-objective multi-skill resource-constrained project scheduling problem. *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, strony 83–86, 2017.

[47] P. B. Myszkowski, M. Laszczyk, I. Nikulin, M. Skowroński. iMOPSE: a library for bicriteria optimization in Multi-Skill Resource-Constrained Project Scheduling Problem. *Soft Computing*, 23(10):3397–3410, 2019.

[48] P. B. Myszkowski, Ł. P. Olech, M. Laszczyk, M. E. Skowroński. Hybrid Differential Evolution and Greedy Algorithm (DEGR) for solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Applied Soft Computing*, 62:1–14, 2018.

[49] P. B. Myszkowski, M. E. Skowroński, Ł. P. Olech, K. Oślizło. Hybrid ant colony optimisation in solving multi-skill resource-constrained project scheduling problem. *Soft Computing*, 19(12):3599–3619, 2015.

[50] P. B. Myszkowski, M. E. Skowroński, K. Sikora. A new benchmark dataset for multi-skill resource-constrained project scheduling problem. *Proceedings of the Federated Conference on Computer Science and Information Systems*, wolumen 5, strona 129–138, 2015.

[51] S. Nama, S. Sharma, A. K. Saha, A. H. Gandomi. A quantum mutation-based backtracking search algorithm. *Artificial Intelligence Review*, 55(4):3019–3073, 2022.

[52] J. J. Palacios, I. González-Rodríguez, C. R. Vela, J. Puente. Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Sets and Systems*, 278:81–97, 2015. Special Issue on uncertainty and imprecision modelling in decision making (EUROFUSE 2013).

[53] E. Pap. *Handbook of Measure Theory*. North-Holland, Amsterdam, 2002.

[54] M. Peeyee, S. Rahman, N. Abdul Hamid, Z. Zakaria. Heuristic based model for groceries shopping navigator. *Indonesian Journal of Electrical Engineering and Computer Science*, 16:932, 2019.

[55] R. Pellerin, N. Perrier, F. Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2):395–416, 2020.

[56] H. D. Quoc. MEMINV: A hybrid efficient approximation method solving the multi skill-resource constrained project scheduling problem. *Mathematical Biosciences and Engineering*, 20(8):15407–15430, 2023.

[57] A. S. Sayyad, T. Menzies, H. Ammar. On the value of user preferences in search-based software engineering: A case study in software product lines. *2013 35th International Conference on Software Engineering (ICSE)*, strony 492–501, 2013.

[58] H. Seada, K. Deb. U-NSGA-III: A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives: Proof-of-Principle Results. *Evolutionary Multi-Criterion Optimization*, strony 34–49, 2015.

[59] X. Shen, Y. Guo, A. Li. Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling. *Applied Soft Computing*, 88:106059, 2020.

[60] Y. C. Shih. Seru Production System: A Review and Projections for Future Research. *Management and Production Engineering Review*, vol. 12(No 4), 2021.

[61] A. J. Sobey, P. A. Grudniewski. Re-inspiring the genetic algorithm with multi-level selection theory: multi-level selection genetic algorithm. *Bioinspiration & Biomimetics*, 13(5):056007, 2018.

[62] N. Srinivas, K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[63] D. H. Stolfi, M. R. Brust, G. Danoy, P. Bouvry. A competitive Predator–Prey approach to enhance surveillance by UAV swarms. *Applied Soft Computing*, 111:107701, 2021.

[64] T. Takagi, K. Takadama, H. Sato. Directional Pareto Front and Its Estimation to Encourage Multi-Objective Decision-Making. *IEEE Access*, 11:20619–20634, 2023.

[65] K. Tan, Y. Chew, T. Lee, Y. Yang. A cooperative coevolutionary algorithm for multiobjective optimisation. *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, wolumen 1, strony 390–395, 2003.

[66] V. Truong Vu, L. T. Bui, T. T. Nguyen. A Competitive Co-Evolutionary Approach for the Multi-Objective Evolutionary Algorithms. *IEEE Access*, 8:56927–56947, 2020.

[67] C. J. Tzeng, Y. H. Lin, Y. K. Yang, M. C. Jeng. Optimization of turning operations with multiple performance characteristics using the Taguchi method and Grey relational analysis. *Journal of Materials Processing Technology*, 209(6):2753–2759, 2009.

[68] L. Wang, X. Zheng. A knowledge-guided multi-objective fruit fly optimisation algorithm for the multi-skill resource constrained project scheduling problem. *Swarm and Evolutionary Computation*, 38:54–63, 2018.

[69] R. Wang, W. Ma, M. Tan, G. Wu, L. Wang, D. Gong, J. Xiong. Preference-inspired coevolutionary algorithm with active diversity strategy for multi-objective multi-modal optimisation. *Information Sciences*, 546:1148–1165, 2021.

[70] Z. Wang, Y. Pei, L. Jianqiang. A Survey on Search Strategy of Evolutionary Multi-Objective Optimization Algorithms. *Applied Sciences*, 13(7):4643, 2023.

[71] W. C. Weng, F. Yang, A. Z. Elsherbeni. Linear Antenna Array Synthesis Using Taguchi's Method: A Novel Optimization Technique in Electromagnetics. *IEEE Transactions on Antennas and Propagation*, 55(3):723–730, 2007.

[72] Y. Zhang, G. Chen, L. Cheng, Q. Wang, Q. Li. Methods to balance the exploration and exploitation in Differential Evolution from different scales: A survey. *Neurocomputing*, 561:126899, 2023.

[73] W. Zhao, S. Alam, H. A. Abbass. MOCCA-II: A multi-objective co-operative co-evolutionary algorithm. *Applied Soft Computing*, 23:407–416, 2014.

[74] L. Zhu, J. Lin, , Z. J. Wang. A discrete oppositional multi-verse optimisation algorithm for multi-skill resource constrained project scheduling problem. *Applied Soft Computing*, 85:105805, 2019.

[75] L. Zhu, J. Lin, Y. Y. Li, Z. J. Wang. A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Knowledge-Based Systems*, 225:107099, 2021.

[76] E. Zitzler, M. Laumanns, L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Raport instytutowy, ETH Zurich, 2001.

[77] E. Zitzler, L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[78] A. Żychowski, J. Mańdziuk. Coevolution of players strategies in security games. *Journal of Computational Science*, 68:101980, 2023.

Spis rysunków

Spis tabel

Spis algorytmów