Politechnika Wrocławska

Rozprawa Doktorska

Wykorzystanie idei „linkage learning" w rozwiązywaniu ciągłych problemów optymalizacyjnych

Marcin Michał Komarnicki

Promotor: dr hab. inż. Michał Przewoźniczek, prof. uczelni

Wrocław, 2023

Rodzicom i wszystkim cierpliwym osobom

Spis treści

Streszczenie

Skuteczne i efektywne rozwiązywanie problemów optymalizacyjnych jest ważne z punktu widzenia praktyki. W przypadku trudnych problemów optymalizacyjnych, często stosuje się w tym celu algorytmy ewolucyjne. W ostatnich latach rozwój algorytmów ewolucyjnych skupia się m.in. na wykorzystaniu idei „linkage learning" podczas ich działania. Pojedynczym rozwiązaniem zadanego problemu optymalizacyjnego może być wektor zmiennych decyzyjnych. Techniki typu „linkage learning" starają się wskazać, które zmienne decyzyjne warto przetwarzać wspólnie, a które niekoniecznie ze względu na ich niezależność. Głównym zastosowaniem tego typu technik w rozwiązywaniu ciągłych problemów optymalizacyjnych są ich wielowymiarowe instancje, które posiadają co najmniej 500 zmiennych decyzyjnych. Poprawna identyfikacja zależnych zmiennych decyzyjnych przez strategie dekompozycji, które są przykładem technik typu „linkage learning", pozwala na dekompozycję zadanego problemu optymalizacyjnego na mniejsze podproblemy, które mogą być następnie niezależnie optymalizowane zmniejszając tym samym wymiarowość rozważanego problemu.

Niniejsza praca skupia się głównie na strategiach dekompozycji wielowymiarowych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań oraz na wykorzystaniu informacji dostarczonych przez te strategie w dalszym procesie optymalizacji. Na podstawie przeprowadzonego przeglądu literatury wskazano wady aktualnie wiodących strategii dekompozycji, które bazują na grupowaniu różnicowym lub sprawdzaniu monotoniczności, m.in. niepoprawna klasyfikacja zmiennych decyzyjnych jako zależnych i niewykrywanie wielu z istniejących zależności pomiędzy zmiennymi decyzyjnymi. Celem niniejszej pracy było zatem zaproponowanie nowej strategii dekompozycji, która będzie eliminować wskazane wady. Wyniki eksperymentów, które zostały przeprowadzone na typowym zbiorze problemów testowych, jego dwóch zaproponowanych w ramach niniejszej pracy modyfikacjach oraz problemie praktycznym z dziedziny sieci komputerowych i komunikacyjnych wskazują, że cel pracy został osiągnięty. Nowo zaproponowana strategia dekompozycji dekomponuje rozważane problemy optymalizacyjne przeważnie z lepszą dokładnością niż ak-

tualnie wiodące strategie dekompozycji. Dokładniejsza dekompozycja wiązała się ze zwiększonym kosztem jej uzyskania. Był on jednak ciągle niewielkim odsetkiem całego budżetu obliczeniowego, dlatego głównym czynnikiem wpływającym na jakość otrzymanych wyników optymalizacji była prawie zawsze dokładność dekompozycji.

Abstract

Solving optimization problems effectively and efficiently is important in terms of real-world applications. To this end, evolutionary algorithms are commonly applied when hard optimization problems are considered. Some recent advances in evolutionary algorithms focuse on linkage learning techniques. A single problem solution may be a vector of decision variables. Linkage learning techniques try to mark those decision variables that are worth processing together and those that are not due to their independence. In continuous optimization, these techniques are mainly used to solve large-scale instances that consist of at least 500 decision variables. An accurate discovery of dependent decision variables by decomposition strategies, which are examples of linkage learning techniques, allows of the problem decomposition into smaller subproblems that can be optimized separately afterward. Thus, the problem size is reduced.

This paper focuses mainly on decomposition strategies for large-scale continuous optimization problems and how to utilize information provided by them in the subsequent optimization process. Based on a literature review made by the author, flaws of state-of-the-art decomposition strategies, which are derived from differential grouping or monotonicity checking, are pointed out. For instance, reporting independent variables as dependent and missing many of the existing interactions between variables. Thus, the main objective of this paper was to propose a new decomposition strategy that should not suffer from any of these flaws. According to experimental results for a typical set of test problems, its two modifications proposed in this paper, and a real-world optimization problem related to computer and communication networks, the objective of this paper was achieved. The new decomposition strategy can mostly decompose the considered optimization problems more accurately when compared with state-of-the-art decomposition strategies. The more accurate decomposition resulted in a higher cost. However, it was only a small part of the overall computational budget. Therefore, the main factor that influenced the optimization results was almost always the accuracy of decomposition.

Wstęp

Algorytmy ewolucyjne (ang. *evolutionary algorithms*) stanowią szeroką grupę metod optymalizacji, których inspiracją jest proces ewolucji występujący w przyrodzie [3, 50, 63, 87, 118]. Do algorytmów ewolucyjnych należą m.in. algorytmy genetyczne (ang. *genetic algorithms*) [40, 52], strategie ewolucyjne (ang. *evolution strategies*) [2, 115] oraz programowanie genetyczne (ang. *genetic programming*) [58, 76]. Jednym z głównych zastosowań algorytmów ewolucyjnych jest poszukiwanie jak najlepszego rozwiązania (najlepiej globalnego optimum) danego problemu optymalizacyjnego (ang. *global optimization*). Typowym zastosowaniem metod ewolucyjnych jest optymalizacja problemów NP-trudnych i NP-zupełnych. W przypadku optymalizacji takich problemów, do których należy wiele problemów praktycznych [69, 81], algorytmy ewolucyjne często osiągają wyniki o znacznie lepszej jakości niż klasyczne metody optymalizacji [95, 112].

Pojedyncze rozwiązanie każdego problemu optymalizacyjnego reprezentowane jest w algorytmach ewolucyjnych poprzez osobnika (ang. *individual*). Każdy osobnik posiada genotyp (ang. *genotype*), który definiuje jego fenotyp (ang. *phenotype*). Fenotyp otrzymujemy po zdekodowaniu genotypu (ang. *decoding*) [87]. Osobniki, które tworzą populację (ang. *population*), rywalizują ze sobą w ramach wirtualnego środowiska określonego za pomocą funkcji przystosowania (ang. *fitness function*). Funkcja przystosowania może być tożsama z optymalizowaną funkcją zwaną funkcją celu (ang. *objective function*) [23, 42] lub być jej modyfikacją poprzez skalowanie, przesunięcie lub inne przekształcenie [23, 60, 140]. Dodatkowo, w przypadku problemów z ograniczeniami, funkcja przystosowania może uwzględniać karę, która nakładana jest na osobnika gdy reprezentowane przez niego rozwiązanie nie mieści się w zbiorze dopuszczalnym (ang. *feasible region*). Wartość kary wyliczana jest zgodnie z funkcją kary (ang. *penalty function*) [3, 19, 51, 67]. Za przeglądanie przestrzeni przeszukiwań odpowiadają operatory ewolucyjne, które modyfikują osobniki. Najbardziej typowe to krzyżowanie (ang. *crossover*) i mutacja (ang. *mutation*). Przystosowanie osobnika wyliczane jest na podstawie jego fenotypu, natomiast operatory ewolucyjne zwykle koncentrują się na przekształcaniu genotypu [3,87]. Treści przedstawione w niniejszej

pracy nie wymagają rozróżnienia pomiędzy genotypem, a fenotypem. Autor przyjął zatem założenie, że genotyp jest tożsamy z fenotypem i pojęcie genotypu opisywać będzie zarówno genotyp, jak i fenotyp. W zależności od przestrzeni przeszukiwań (ang. *search space*), osobnik może posiadać geny o różnej postaci np. wartości binarne, liczby całkowite lub zmiennoprzecinkowe, które kodują jedno z rozwiązań. Ponadto, istnieje możliwość stosowania kodowania mieszanego, gdzie różne rodzaje wartości genów występują wspólnie w jednym genotypie [138, 139].

Skuteczność operatorów ewolucyjnych może zależeć od wewnętrznej struktury problemu [36, 105, 108]. Rozpatrzmy optymalizację dwustuwymiarowej funkcji $\ddot{f}_b$, której dziedziną jest zbiór $\{0,1\}^{200}$. Gdy wyliczymy wartość funkcji $\ddot{f}_b$ dla wszystkich 2^{200} możliwych rozwiązań to mamy pewność, że znajdziemy rozwiązanie optymalne. Jeżeli założymy, że funkcja $\ddot{f}_b$ składa się z dwudziestu dziesięciowymiarowych podproblemów i, zgodnie z tym założeniem, jest określona wzorem $\ddot{f}_b([x_1, ..., x_{200}]) = \ddot{f}_{b,1}([x_1, ..., x_{10}]) + \ddot{f}_{b,2}([x_{11}, ..., x_{20}]) + ... + \ddot{f}_{b,20}([x_{191}, ..., x_{200}])$ to wprowadzamy wiedzę na temat zależności i niezależności zmiennych decyzyjnych. Dwie zmienne, które są wspólnie argumentami przynajmniej jednego podproblemu uznawane są za zależne. W przeciwnym przypadku, nie oddziałują one na siebie wzajemnie. Na podstawie tak rozumianej wiedzy o strukturze problemu, wystarczy policzyć wartość funkcji $\ddot{f}_b$ dla $20 \cdot 2^{10}$ rozwiązań by znaleźć rozwiązanie optymalne, ponieważ funkcja $\ddot{f}_b$ składa się z dwudziestu addytywnie separowalnych podproblemów o rozmiarze 10.

Wiele algorytmów ewolucyjnych nie zakłada żadnych cech optymalizowanego problemu (ang. *black-box optimization*), w tym jego struktury. Niemniej jednak, jeżeli problem składa się z podproblemów to dostarczając metodzie ewolucyjnej wiedzę *a priori* na temat liczby podproblemów oraz ich zmiennych (ang. *gray-box optimization*), możemy znacząco zwiększyć skuteczność i efektywność algorytmu ewolucyjnego [88, 131]. Taka wiedza może zostać wykorzystana na przynajmniej dwa różne sposoby, które w wielu przypadkach prowadzą do uzyskania znacząco lepszych wyników. Po pierwsze, można znacząco zmniejszyć koszt wyliczenia pojedynczego przystosowania, przeliczając jedynie te podproblemy, dla których zaszły jakiekolwiek zmiany [10, 11, 102, 132]. Tego typu optymalizacja może być szczególnie przydatna dla problemów, które cechują się ogromną złożonością funkcji przystosowania np. problemów opartych na symulacji [80, 114]. Zakładając, że znamy wewnętrzną strukturę optymalizowanego problemu, możemy zaproponować nowe, lepiej ukierunkowane operatory ewolucyjne lub metody lokalnej optymalizacji, które z tej wiedzy korzystają usprawniając proces przeszukiwania przestrzeni rozwiązań [17, 32, 132, 133]. Należy jednak zauważyć, że wiele prac dotyczących optymalizacji z wykorzystaniem wiedzy o wewnętrznej strukturze problemu, skupia się na

rozwiązywaniu problemów, których podproblemy nakładają się na siebie (ang. *overlapping subproblems*) [16, 131, 132]. Takich problemów nie da się podzielić na niezależne podproblemy, ponieważ dla każdego podproblemu istnieje co najmniej jeden inny podproblem, z którym współdzieli on część swoich zmiennych decyzyjnych (ang. *shared decision variables*) [47,48,125]. Dobór problemów optymalizacyjnych w pracach dotyczących optymalizacji z wykorzystaniem wiedzy o wewnętrznej strukturze problemu wydaje się zatem uzasadniony, ponieważ w przypadku nie w pełni separowalnych problemów, wiedza o wewnętrznej strukturze problemu pozwala na niezależną optymalizację każdego z podproblemów, zmniejszając tym samym nakład obliczeniowy potrzebny na znalezienie jak najlepszego rozwiązania całego problemu.

W sytuacji gdy nie posiadamy żadnej wiedzy dotyczącej struktury optymalizowanego problemu, użyteczne mogą okazać się techniki, które pozwalają na wykrycie zależności pomiędzy genami (ang. *linkage learning*). Zazwyczaj budują one modele na podstawie wartości funkcji przystosowania próbek, które uzyskujemy próbkując przestrzeń przeszukiwań. Model może zostać zbudowany przed uruchomieniem metody optymalizacji, wtedy pozostaje on niezmienny podczas jej działania (ang. *predetermined linkage models*) [89, 123, 125, 129, 141]. Takie podejście wymaga często dodatkowych wyliczeń funkcji przystosowania. Z drugiej strony istnieją techniki, które uczą model w trakcie procesu optymalizacji (ang. *learned linkage models*) [11,43,105,129,130]. Mogą one bazować na ocenionych dotychczas osobnikach, zatem dodatkowe wyliczenia funkcji przystosowania nie są konieczne. Modele powinny jak najlepiej odwzorowywać rzeczywistą strukturę problemu, jednakże nie wszystkie modele są doskonałe. Autorzy [105] wskazują dwa typy niedokładności modeli: brak powiązań pomiędzy zależnymi genami (ang. *missing linkage*) oraz oznaczenie niezależnych genów jako zależnych (ang. *false linkage*). Należy pamiętać, że techniki typu „linkage learning" wspomagają jedynie proces optymalizacji i ich użyteczność powinno się oceniać z perspektywy wyników, które osiągają wykorzystujące je metody optymalizacji. W wielu przypadkach skuteczność metody jest wprost proporcjonalna do jakości modelu [94,104,123]. Analizując wykorzystanie technik typu „linkage learning" w rozwiązywaniu ciągłych problemów optymalizacyjnych, możemy zauważyć ich powszechne zastosowanie podczas globalnej optymalizacji wielowymiarowych problemów (ang. *large-scale global optimization*) [73,89,93]. Przyjmuje się, że tego typu problemy, posiadają przynajmniej 500 zmiennych decyzyjnych. Najczęściej mają jednak one co najmniej 1000 wymiarów [49,153]. Są to ważne zadania optymalizacji z punktu widzenia praktyki, ponieważ wiele praktycznych problemów jest wielowymiarowych [46,93,147]. Optymalizacja wielowymiarowych, ciągłych problemów optymalizacyjnych została ponadto zaklasyfikowana jako jedno z bardziej

skomplikowanych zadań optymalizacji gdy rozpatrujemy ciągłą przestrzeń przeszukiwań [153]. Powodem jest wykładniczy wzrost rozmiaru przestrzeni przeszukiwań wraz ze wzrostem liczby wymiarów optymalizowanego problemu [153], który skutkuje m.in. wykładniczym wzrostem liczby optimów lokalnych [49], ogromnym kosztem uruchomienia algorytmów estymowania rozkładu (ang. *estimation of distribution algorithms*) [24] lub częstym zwiększeniem złożoności problemu [144].

Ze względu na trudności w rozwiązywaniu wielowymiarowych problemów przez tradycyjne algorytmy ewolucyjne, badacze zaczęli szukać alternatywnych podejść. W ostatnich latach dwa podejścia zyskały szczególną popularność [93]. Pierwszym są metody optymalizacji, które działają zgodnie z zasadą dziel i zwyciężaj (ang. *divide and conquer*) [77]. W tym celu, rozwiązywany problem optymalizacyjny jest na początku dekomponowany na mniejsze podproblemy by następnie optymalizować każdy podproblem oddzielnie, wykorzystując zazwyczaj koewolucję kooperatywną (ang. *cooperative co-evolution*) [73, 89, 92, 123, 125, 141, 149]. Drugim podejściem, które nie wymaga dekompozycji problemu, są hybrydowe metody optymalizacji, czyli łączenie różnych metod optymalizacji, często o różnych cechach, podczas jednego procesu przeszukiwania [33, 65, 66, 82]. Przykładem takiej metody jest SHADE-ILS (ang. *Success-History Based Parameter Adaptation for Differential Evolution with Iterative Local Search*) [82], która w 2018 roku wygrała konkurs, w ramach konferencji *IEEE Congress on Evolutionary Computation* (CEC), na najlepszą metodę optymalizacji zaprojektowaną do rozwiązywania wielowymiarowych problemów o ciągłej przestrzeni przeszukiwań[1].

Za strategie dekompozycji, które najdokładniej dekomponują ciągłe problemy optymalizacyjne, uważa się strategie bazujące na grupowaniu różnicowym (ang. *differential grouping*) [89, 94, 121, 123, 125, 126]. Nie są one jednak wolne od wad, ponieważ zakładają, że dekomponowane problemy posiadają addytywnie separowalne podproblemy. W przeciwnym przypadku, mogą zostać wykryte zależności, które w rzeczywistości nie istnieją. To założenie nie jest jednak spełnione dla wielu praktycznych problemów optymalizacyjnych np. problemów, które są częścią współczesnych aplikacji przetwarzających ogromne wolumeny danych (ang. *big data*) [25]. Ogólniejsze strategie, tzn. niezakładające konkretnego rodzaju separowalności, wykorzystują m.in. sprawdzanie monotoniczności (ang. *monotonicity checking*) [14, 29, 120, 143]. Strategie bazujące na sprawdzaniu monotoniczności mają tendencje do niewykrywania zależności, które w rzeczywistości istnieją. Pomimo większej ogólności, wada ta przeważa prowadząc do uzyskiwania dekompozycji o gorszej dokładności niż grupowanie różnicowe i wywodzące się z niego kolejne strategie [29,89,94,123]. Jedna z naj-

[1] `https://www.tflsgo.org/download/comp2018_slides.pdf`

nowszych propozycji opartych na grupowaniu różnicowym, RDG3 (ang. *Recursive Differential Grouping 3*) [125], po osadzeniu jej w architekturze koewolucji kooperatywnej zwanej CBCC (ang. *Contribution-based Cooperative Co-evolution*) [70, 92], wygrała w 2019 roku kolejną edycję konkursu w ramach konferencji CEC i została uznana za konkurencyjną wobec SHADE-ILS[2].

W ramach niniejszej pracy poruszona zostanie tematyka związana z rozwiązywaniem wielowymiarowych ciągłych problemów optymalizacyjnych przez metody, które nie zakładają żadnych cech optymalizowanego problemu.

Strategie bazujące na grupowaniu różnicowym lub sprawdzaniu monotoniczności nie są wolne od wad, co prowadzi do sformułowania następującej tezy pracy.

Teza główna pracy

Możliwe jest zaprojektowanie nowej strategii dekompozycji, bazującej na idei sprawdzania monotoniczności, która jest techniką typu „linkage learning", o typowej złożoności liniowo-logarytmicznej, równie dokładnej jak strategie bazujące na grupowaniu różnicowym dla addytywnie separowalnych problemów optymalizacyjnych i dokładniejszej dla nieaddytywnie separowalnych problemów. Dokładność dekompozycji zostanie zmierzona miarami powszechnie stosowanymi w literaturze.

W celu uprawdopodobnienia tezy pracy sformułowano następujący cel rozprawy.

Cel pracy

Celem niniejszej pracy jest opracowanie nowej strategii dekompozycji, przeznaczonej dla ciągłych przestrzeni przeszukiwań, która dekomponowałaby z wysoką dokładnością problemy składające się z zarówno addytywnie, jak i nieaddytywnie separowalnych podproblemów. Koewolucja kooperatywna wykorzystująca proponowaną strategię dekompozycji będzie rozwiązywać wielowymiarowe problemy optymalizacyjne równie skutecznie w przypadku addytywnej separowalności oraz skuteczniej dla problemów z nieaddytywnie separowalnymi podproblemami, niż inne strategie dekompozycji osadzone w tych samych architekturach koewolucji kooperatywnej.

Do osiągnięcia celu rozprawy zdefiniowano następujące zadania badawcze.

Zadania badawcze

[2] `https://www.tflsgo.org/assets/cec2019/comp2019_slides.pdf`

1. Opracowanie zbioru wielowymiarowych problemów testowych o różnych cechach, tj. problemy o ciągłej przestrzeni przeszukiwań, składające się z addytywnie separowalnych podproblemów, nieaddytywanie separowalnych podproblemów oraz problemy w pełni nieseparowalne.

2. Opracowanie, w przypadku braku odpowiednich problemów w literaturze, nowych problemów testowych o zadanych cechach w celu uzupełnienia zbioru opracowanego w ramach zadania 1.

3. Opracowanie nowej strategii dekompozycji, która dekomponuje z wysoką dokładnością problemy składające się z zarówno addytywnie, jak i nieaddytywnie separowalnych podproblemów.

4. Opracowanie analizy porównawczej jakości i kosztu dekompozycji proponowanej strategii z inną, aktualnie wiodącą strategią. Podstawą porównania będzie zbiór problemów testowych opracowany w ramach zadań 1 i 2.

5. Opracowanie analizy porównawczej skuteczności aktualnych wersji koewolucji kooperatywnej wykorzystujących wiodącą strategię wybraną w zadaniu 4, oraz autorską strategię opracowaną w ramach zadania 3. Podstawą analizy porównawczej będzie zbiór problemów testowych opracowany w ramach zadań 1 i 2.

6. Opracowanie zbioru instancji problemu praktycznego, który składa się z nieaddytywnie separowalnych podproblemów.

7. Opracowanie analizy porównawczej wyników optymalizacji problemu praktycznego wybranego w punkcie 6 uzyskanych przez połączenie proponowanej strategii dekompozycji oraz strategi wybranej w punkcie 4 z powszechnie stosowanymi architekturami koewolucji kooperatywnej.

Struktura pracy jest następująca:

1. Rozdział 1 zawiera definicję wielowymiarowych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań. W tym rozdziale przedstawione zostały również aktualne podejścia do rozwiązywania tego typu problemów.

2. W rozdziale 2 zostały opisane strategie dekompozycji ciągłych problemów optymalizacyjnych oraz idea „linkage learning", która jest ściśle z nimi związana.

3. Rozdział 3 przedstawia zaproponowane w ramach niniejszej pracy inkrementacyjne i rekursywne grupowanie rankingowe (ang. *Incremental Recursive Ranking Grouping*, IRRG). Dla strategii IRRG zaprezentowano motywacje stojące za jej sformułowaniem, jej budowę, porównanie z innymi strategiami bazującymi na sprawdzaniu monotoniczności oraz jej złożoność obliczeniową. Wszystkie te elementy są autorskimi osiągnięciami autora niniejszej pracy.

4. Rozdział 4 prezentuje wybrane problemy testowe wraz z wyszczególnieniem ich cech. W tym rozdziale opisany został także przebieg i wyniki eksperymentów, na podstawie których wyciągnięte zostały następnie wnioski. Opracowanie dwóch nowych zbiorów problemów testowych oraz dokładna analiza uzyskanych wyników są autorskimi osiągnięciami autora niniejszej pracy.

5. Rozdział 5 opisuje wybrany problem praktyczny, którym jest projektowanie przepływu z rozgałęzieniami w sieciach komputerowych i komunikacyjnych. Rozdział 5 zawiera także konfigurację i wyniki eksperymentów, które są podstawą porównania strategii IRRG z inną, aktualnie wiodącą strategią dekompozycji. Pokazanie, że wybrany problem praktyczny jest problemem składającym się z nieaddytywnie separowalnych podproblemów oraz dokładna analiza uzyskanych wyników są autorskimi osiągnięciami autora niniejszej pracy.

Dotychczasowe publikacje autora niniejszej pracy można podzielić na następujące trzy kategorie.

– Publikacje bezpośrednio związane z tematem niniejszej pracy, które poruszają tematykę wykorzystania idei „linkage learning” w rozwiązywaniu ciągłych problemów optymalizacyjnych.

– Publikacje pośrednio związane z tematem niniejszej pracy, które także dotyczą wykorzystania idei „linkage learning” w rozwiązywaniu problemów optymalizacyjnych, lecz o innej niż ciągła przestrzeni przeszukiwań.

– Inne publikacje, których tematyka nie dotyka w ogóle wykorzystania idei „linkage learning” w procesie optymalizacji.

Lista publikacji autora niniejszej pracy jest następująca.

Publikacja bezpośrednio związana z tematem niniejszej pracy opublikowana w czasopiśmie z listy filadelfijskiej

Marcin Komarnicki, Michał Przewoźniczek, Halina Kwaśnicka, Krzysztof Walkowiak. Incremental recursive ranking grouping for large scale global optimization. IEEE Transactions on Evolutionary Computation. 2022.

Publikacje pośrednio związane z tematem niniejszej pracy opublikowane w czasopismach z listy filadelfijskiej

Michał Przewoźniczek, **Marcin Komarnicki**. Empirical problem decomposition — the key to the evolutionary effectiveness in solving a large-scale non-binary discrete real-world problem. Applied Soft Computing. 2021, vol. 113, art. 107864, s. 1-17.

Michał Przewoźniczek, **Marcin Komarnicki**. Empirical linkage learning. IEEE Transactions on Evolutionary Computation. 2020, vol. 24, nr 6, s. 1097-1111.

Michał Przewoźniczek, Rituparna Datta, Krzysztof Walkowiak, **Marcin Komarnicki**. Splitting the fitness and penalty factor for temporal diversity increase in practical problem solving. Expert Systems with Applications. 2020, vol. 145, art. 113126, s. 1-11.

Michał Przewoźniczek, Krzysztof Walkowiak, Arunabha Sen, **Marcin Komarnicki**, Piotr Lechowicz. The transformation of the k-Shortest Steiner Trees search problem into binary dynamic problem for effective evolutionary methods application. Information Sciences. 2019, vol. 479, s. 1-19.

Publikacje pośrednio związane z tematem niniejszej pracy opublikowane jako referaty konferencyjne

Michał Przewoźniczek, Renato Tinós, Bartosz Frej, **Marcin Komarnicki**. On turning black - into dark gray-optimization with the direct empirical linkage discovery and partition crossover. W: GECCO '22 : Proceedings of the 2022 Genetic and Evolutionary Computation Conference, July 9-13, 2022, Boston, Massachusetts / ed. Jonathan E. Fieldsend. New York, NY : ACM, cop. 2022. s. 269-277.

Michał Przewoźniczek, **Marcin Komarnicki**, Bartosz Frej. Direct linkage discovery with empirical linkage learning. W: GECCO'21 Companion : Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion, July 10-14, 2021 Lille, France / eds. Francisco Chicano, Krzysztof Krawiec. New York, NY

: ACM, cop. 2021. s. 609-617.

Michał Przewoźniczek, **Marcin Komarnicki**. Fitness caching - from a minor mechanism to major consequences in modern Evolutionary Computation. W: 2021 IEEE Congress on Evolutionary Computation (CEC), 28.06-1.07.2021, Kraków, Poland : proceedings / eds. Jacek Mańdziuk and Hussein Abbass. Danvers : IEEE, cop. 2021. s. 1785-1791.

Michał Przewoźniczek, **Marcin Komarnicki**, Peter Bosman, Dirk Thierens, Bartosz Frej, Ngoc Hoang Luong. Hybrid linkage learning for permutation optimization with Gene-pool optimal mixing evolutionary algorithms. W: GECCO'21 Companion : Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion, July 10-14, 2021 Lille, France / eds. Francisco Chicano, Krzysztof Krawiec. New York, NY : ACM, cop. 2021. s. 1442-1450.

Michał Przewoźniczek, Bartosz Frej, **Marcin Komarnicki**. On measuring and improving the quality of linkage learning in modern evolutionary algorithms applied to solve partially additively separable problems. W: GECCO '20 : Proceedings of the Genetic and Evolutionary Computation Conference, Cancún, Mexico, July 08-12, 2020 / ed. Jose A. Lozano. New York, NY : ACM, cop. 2020. s. 742-750.

Marcin Komarnicki, Michał Przewoźniczek, Tomasz Durda. Comparative mixing for DSMGA-II. W: GECCO '20 : Proceedings of the Genetic and Evolutionary Computation Conference, Cancún, Mexico, July 08-12, 2020 / ed. Jose A. Lozano. New York, NY : ACM, cop. 2020. s. 708-716.

Szymon Woźniak, Michał Przewoźniczek, **Marcin Komarnicki**. Parameter-less Population Pyramid for Permutation-based problems. W: Parallel Problem Solving-from Nature-PPSN XVI : 16th International Conference, PPSN 2020, Leiden, the Netherlands, September 5-9, 2020 : proceedings. Pt 2 / eds. Thomas Bäck [i in.]. Cham : Springer, cop. 2020. s. 418-430.

Adam Zieliński, **Marcin Komarnicki**, Michał Przewoźniczek. Parameter-less population pyramid with automatic feedback. W: GECCO '19 : Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, July 13-17, 2019. New York, NY : ACM, cop. 2019. s. 312-313.

Marcin Komarnicki, Michał Przewoźniczek. Parameter-less, population-sizing DSMGA-II. W: GECCO '19 : Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, July 13-17, 2019. New York, NY : ACM, cop. 2019. s. 289-290.

Michał Przewoźniczek, **Marcin Komarnicki**. The influence of fitness caching on modern evolutionary methods and fair computation load measurement. W: GECCO '18 : Proceedings of the Genetic and Evolutionary Computation Conference Companion, Kyoto, Japan, July 15-19, 2018. New York, NY : ACM, cop. 2018. s. 241-242.

Michał Przewoźniczek, **Marcin Komarnicki**. The practical use of problem encoding allowing cheap fitness computation of mutated individuals. W: Proceedings of the 2018 Federated Conference on Computer Science and Information Systems : September 9-12, 2018, Poznań, Poland / eds. Maria Ganzha, Leszek Maciaszek, Marcin Paprzycki. Warszawa : Polskie Towarzystwo Informatyczne ; Los Alamitos, Ca. : IEEE, cop. 2018. s. 57-65.

Marcin Komarnicki, Michał Przewoźniczek. Parameter-less population pyramid with feedback. W: GECCO '17 : Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, July 15-19, 2017. New York, NY : ACM, cop. 2017. s. 109-110.

Marcin Komarnicki, Michał Przewoźniczek. Linked genes migration in Island Models. W: Proceedings of the 8th International Joint Conference on Computational Intelligence, IJCCI 2016 : Porto, Portugal, November 09-11, 2016. Vol. 3, ECTA / Eds. Juan Julian Merelo [i in.]. [B.m] : SciTePress, cop. 2016. s. 30-40.

Inna publikacja opublikowana w czasopiśmie z listy filadelfijskiej

Piotr Dziurzanski, Shuai Zhao, Michał Przewoźniczek, **Marcin Komarnicki**, Leandro Soares. Indrusiak. Scalable distributed evolutionary algorithm orchestration using Docker containers. Journal of Computational Science. 2020, vol. 40, art. 101069, s. 1-14.

Inna publikacja opublikowana jako referat konferencyjny

Shuai Zhao, Piotr Dziurzanski, Michał Przewoźniczek, **Marcin Komarnicki**, Lean-

dro Soares. Indrusiak. Cloud-based dynamic distributed optimisation of integrated process planning and scheduling in smart factories. W: GECCO'19 : Proceedings of the 2019 : Genetic and Evolutionary Computation Conference, July 13-17, 2019, Prague, Czech Republic / ed. Manuel López-Ibáñez. New York, NY : Association for Computing Machinery, cop. 2019. s. 1381-1389.

Rozdział 1

Optymalizacja wielowymiarowych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań

Każdy problem optymalizacyjny możemy zdefiniować za pomocą co najmniej jednej funkcji celu, zwanej także kryterium optymalizacji. W przypadku globalnej optymalizacji, celem jest znalezienie minimum lub maksimum globalnego jednej z nich. W ramach niniejszej pracy, rozpatrujemy jedynie problemy jednokryterialne. Ponadto, bez utraty ogólności, rozpatrujemy zadanie minimalizacji. Niech $f : \mathcal{U} \to \mathbb{R}$ oznacza funkcję celu, gdzie $\mathcal{U}$ jest przestrzenią przeszukiwań. Najlepszym rozwiązaniem problemu optymalizacyjnego jest wtedy $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$, gdzie $\mathcal{D} \subseteq \mathcal{U}$ jest zbiorem dopuszczalnym. Jeżeli $\mathcal{D} = \mathcal{U}$ to rozwiązujemy problem bez ograniczeń. W przeciwnym przypadku, rozpatrywany problem posiada jest problemem z ograniczeniami.

Przestrzenią przeszukiwań n-wymiarowych ciągłych problemów optymalizacyjnych jest podzbiór n-krotnego iloczynu kartezjańskiego zbioru liczb rzeczywistych, $\mathcal{U} = \mathbb{R}^n$. W niniejszej pracy, skupimy się na ciągłych problemach z ograniczeniami kostkowymi (ang. *bounding-box*) [3, 8]. Oznaczając jako $\mathbf{x} = [x_1, ..., x_n]$ pojedyncze rozwiązanie takiego problemu, to dla każdej zmiennej decyzyjnej (ang. *decision variable*) x_i możemy określić jej minimalną (ang. *lower bound*) i maksymalną (ang. *upper bound*) dopuszczalną wartość, $\forall_{i\in\{1,...,n\}} l_i \leq x_i \leq u_i$. Zbiór dopuszczalny możemy zdefiniować wtedy jako $\mathcal{D} = [l_1, u_1] \times ... \times [l_n, u_n] \subset \mathbb{R}^n$ [3].

W niniejszym rozdziale zaprezentowane zostaną dwa główne, aktualne w literaturze problemu [66, 73, 82, 93, 125], podejścia stosowane do optymalizacji wielowymiarowych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań. Tego typu problemy charakteryzują się liczbą wymiarów większą lub równą 500, chociaż

zwykle jest ich co najmniej 1000 [49, 153].

1.1 Zasada dziel i zwyciężaj

Metody działające zgodnie z zasadą dziel i zwyciężaj, zazwyczaj rozwiązują problemy optymalizacyjne dwuetapowo [73, 77, 89, 123, 141]. Pierwszą fazą metody jest wówczas dekompozycja danego problemu na mniejsze podproblemy, które następnie są oddzielnie optymalizowane. W tym celu powszechnie stosuje się koewolucję kooperatywną [92, 124, 149]. Istnieje również podejście, w którym problem dekomponowany jest w trakcie działania koewolucji kooperatywnej. Po wykryciu nowych zależności, następuje wtedy reorganizacja jej komponentów. Ze względu na gorszą skuteczność takich rozwiązań i zakres niniejszej pracy, nie zostaną one szczegółowo opisane [44, 91, 151].

1.1.1 Dekompozycja ciągłych problemów optymalizacyjnych

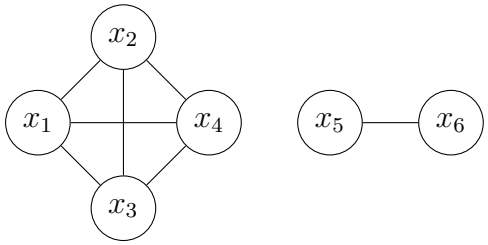
Strategie dekompozycji problemów optymalizacyjnych dzielą dany problem na mniejsze części — podproblemy. Wynikiem dekompozycji ciągłych problemów jest zwykle $1 \leq k \leq n$ grup, które są rozłącznymi podzbiorami zbioru wszystkich zmiennych decyzyjnych, tj. $X = X_1 \cup ... \cup X_k = \{x_1, ..., x_n\}$ [29, 47, 89, 94, 123]. Dokładność otrzymanej dekompozycji jest ściśle związana z separowalnością funkcji celu [94]. Funkcja $f : \mathcal{D} \to \mathbb{R}$ jest nie w pełni separowalna i składa się z $2 \leq m < n$ niezależnych składowych jeżeli

$$\arg\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) = \left[\arg\min_{\mathbf{x_1}\in\mathcal{D}_1} f(\mathbf{x_1}, ...), ..., \arg\min_{\mathbf{x_m}\in\mathcal{D}_m} f(..., \mathbf{x_m}) \right] \tag{1.1}$$

gdzie $\mathcal{D} = \mathcal{D}_1 \times ... \times \mathcal{D}_m$. Gdy $m = n$ to każda zmienna jest niezależna, a funkcja jest w pełni separowalna. Innym brzegowym przypadkiem jest funkcja w pełni nieseparowalna, której wszystkie zmienne są zależne od pozostałych [90].

Zmienne decyzyjne mogą oddziaływać na siebie wzajemnie w sposób bezpośredni (ang. *direct interaction*) lub pośredni (ang. *indirect interaction*) [109, 121, 122]. W literaturze można znaleźć ich alternatywne określenia, odpowiednio jawne (ang. *explicit interaction*) i niejawne (ang. *implicit interaction*) oddziaływania [47]. W celu wyjaśnienia tego zjawiska, wprowadźmy macierz interakcji Θ, która opisuje wewnętrzną strukturę problemu [77]. Macierz interakcji jest macierzą symetryczną o wymiarach $n \times n$. Wartość na przecięciu i-tego wiersza z j-tą kolumną definiuje, czy zmienne x_i oraz x_j są bezpośrednio zależne. Wartość 1 oznacza bezpośrednią zależność, natomiast 0 oznacza brak bezpośredniej zależności. Należy zauwa-

żyć, że zmienne, które nie są bezpośrednio zależne, mogą być niezależne od siebie lub pośrednio zależne (zostanie to wyjaśnione w dalszej części rozdziału). Współczynniki leżące na głównej przekątnej są nieokreślone, ponieważ macierz interakcji służy do przechowywania informacji na temat zależności pomiędzy różnymi zmiennymi decyzyjnymi. Rysunek 1.1 przedstawia wewnętrzna strukturę przykładowego sześciowymiarowego problemu optymalizacyjnego. Należy zauważyć, że każdą macierz interakcji można przedstawić za pomocą nieskierowanego grafu interakcji, gdzie wierzchołkami są zmienne decyzyjne, a krawędź pomiędzy dwoma wierzchołkami istnieje wtedy i tylko wtedy, gdy odpowiadający im element w macierzy interakcji ma wartość równą 1. Krawędzie łączą zatem jedynie te zmienne, które bezpośrednio oddziałują na siebie wzajemnie. Przeglądając graf interakcji możemy wyznaczyć pary zmiennych decyzyjnych, które są pośrednio zależne. Zmienne x_i i x_j pośrednio oddziałują na siebie wzajemnie wtedy i tylko wtedy, gdy nie są bezpośrednio zależne i istnieje pomiędzy nimi ścieżka o długości większej niż 1. Na rysunku Rysunek 1.2, przerywaną linią zaznaczono dwie pary zmiennych, (x_1, x_4) oraz (x_2, x_4), które są od siebie pośrednio zależne. Przedstawiono także dwie zmienne decyzyjne, mianowicie x_5 i x_6, które są w pełni separowalne — nie zależą bezpośrednio ani pośrednio od żadnej innej zmiennej decyzyjnej. Przykładowym problemem optymalizacyjnym, o wewnętrznej strukturze takiej, jak przedstawiono na rysunku Rysunek 1.2, jest zadanie minimalizacji funkcji $\ddot{f}_{c,1} : [-5, 5]^6 \to [0, 475]$ określonej wzorem $\ddot{f}_{c,1}(\mathbf{x}) = (x_1+x_2)^2+(x_1+x_3)^2+(x_2+x_3)^2+(x_3+x_4)^2+x_4^2+x_5^2+x_6^2$. Najlepszym rozwiązaniem problemu jest $\mathbf{x}^* = [0, 0, 0, 0, 0, 0]$. Zmienne x_5 i x_6 są w pełni separowalne, zatem niezależnie od wartości innych zmiennych, ich najlepszą wartością zawsze jest 0. Formalny zapis powyższego stwierdzenia, na przykładzie zmiennej x_6, jest następujący $\forall_{a_1,a_2,a_3,a_4,a_5\in[-5,5]} \arg\min_{x_6\in[-5,5]} \ddot{f}_{c,1}([a_1, a_2, a_3, a_4, a_5, x_6]) = 0$. Zmienne x_1, x_2 i x_3 bezpośrednio oddziałują na siebie wzajemnie, tzn. przeszukując zbiór dopuszczalny tylko jednej z nich, znajdziemy jej globalnie najlepszą wartość (wartość, która jest częścią rozwiązania optymalnego) wtedy i tylko wtedy, gdy wartości pozostałych dwóch zmiennych będą odpowiednie np. równe 0. Bezpośrednią zależność pomiędzy zmiennymi x_1 i x_2 możemy zapisać jako $\exists_{a_2\in[-5,5]} \arg\min_{x_1\in[-5,5]} \ddot{f}_{c,1}([x_1, a_2, 0, 0, 0, 0]) \neq 0$. Istnieje zatem przynajmniej jedna wartość a_2, dla której najlepszą wartością zmiennej x_1 nie jest 0, mimo że wartości pozostałych zmiennych należą do rozwiązania optymalnego. Pośrednia zależność istnieje natomiast pomiędzy zmienną x_4, a x_1 i x_2. Oznacza to, że zmieniając wartość x_4 nie wpływamy na lokalnie najlepszą wartość x_1 i x_2, np. $\forall_{a_4\in[-5,5]} \arg\min_{x_1\in[-5,5]} \ddot{f}_{c,1}([x_1, 0, 0, a_4, 0, 0]) = 0$. Z drugiej strony, przeszukiwanie zbiorów dopuszczalnych zmiennych x_1 i x_2 może nie zakończyć się znalezieniem ich globalnie najlepszych wartości ze względu na bez-

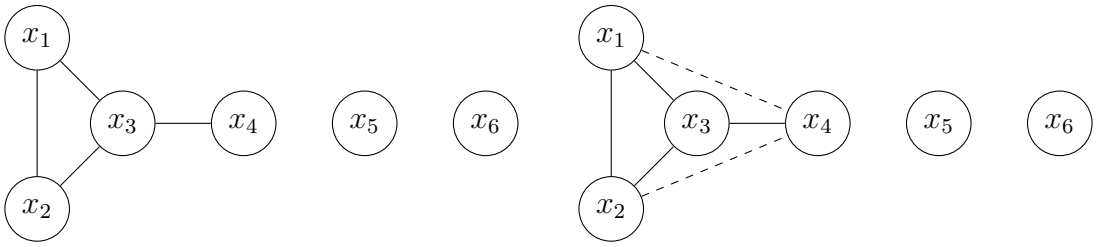


(a) Graf interakcji

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	-	1	1	1	0	0
x_2	1	-	1	1	0	0
x_3	1	1	-	1	0	0
x_4	1	1	1	-	0	0
x_5	0	0	0	0	-	1
x_6	0	0	0	0	1	-

(b) Macierz interakcji

Rysunek 1.1: Wewnętrzna struktura przykładowego problemu optymalizacyjnego



(a) Przed zaznaczeniem pośrednich zależności (b) Po zaznaczeniu pośrednich zależności

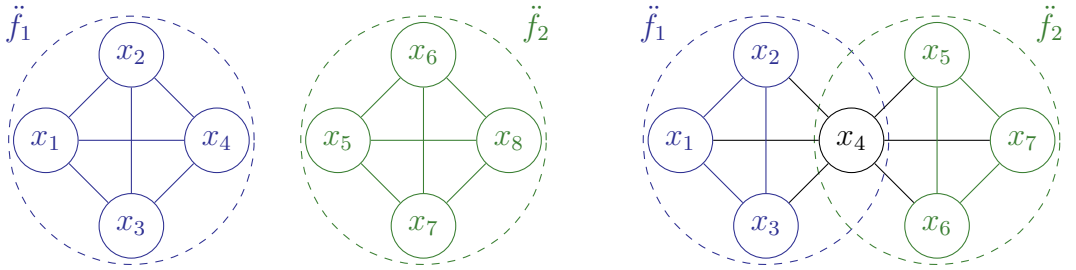
Rysunek 1.2: Graf interakcji problemu z pośrednio zależnymi zmiennymi decyzyjnymi

pośrednie oddziaływanie zmiennej x_4 na zmienną x_3. Rozważmy następujące rozwiązanie początkowe $[0, 0, 0, 3, 0, 0]$, dla którego chcemy znaleźć najlepszą lokalnie wartość dla zmiennej x_3. Wynikiem takiego procesu optymalizacji jest rozwiązanie $[0, 0, -1, 3, 0, 0]$, ponieważ $\arg\min_{x_3 \in [-5,5]} \ddot{f}_{c,1}([0, 0, x_3, 3, 0, 0]) = -1$. Przypisanie wartości -1 do zmiennej x_3 uniemożliwia znalezienie globalnie najlepszych wartości dla zmiennych x_1 i x_2 jeżeli chcielibyśmy je optymalizować pojedynczo, ponieważ $\arg\min_{x_1 \in [-5,5]} \ddot{f}_{c,1}([x_1, 0, -1, 3, 0, 0]) = 0,5$ i $\arg\min_{x_2 \in [-5,5]} \ddot{f}_{c,1}([0, x_2, -1, 3, 0, 0]) = 0,5$. Możemy zatem stwierdzić, że zmienna x_4 bezpośrednio oddziałuje na zmienną x_3 i pośrednio na zmienne x_1 i x_2.

Zgodnie ze wzorem (1.1) i w odniesieniu do grafu interakcji, problem optymalizacyjny jest zbudowany z m niezależnych podproblemów gdy jego graf interakcji składa się z dokładnie m spójnych składowych. Jeżeli spójna składowa nie jest grafem pełnym to reprezentowany przez nią niezależny podproblem posiada zmienne, które są pośrednio od siebie zależne. Pośrednia zależność pomiędzy zmiennymi x_p i x_q istnieje wtedy, gdy należą one do tej samej spójnej składowej lecz ich wierzchołki nie są połączone krawędzią. Na rysunku Rysunek 1.2b przerywaną linią zaznaczone zostały dwie pary zmiennych pośrednio zależnych. Jeżeli niezależny podproblem posiada zmienne, które pośrednio od siebie zależą to można z niego wyodrębnić kolejne podproblemy, które nakładają się na siebie wzajemnie (ang. *overlapping subproblems*). Rysunek 1.3 przedstawia dwa problemy, które składają się z dwóch

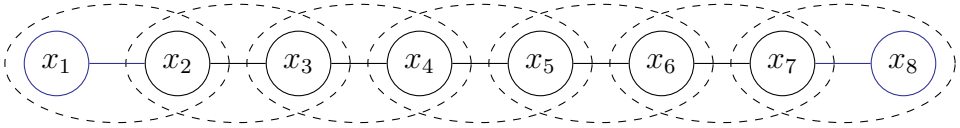
czterowymiarowych podproblemów $\ddot{f}_1$ i $\ddot{f}_2$. Podproblemy na rysunku Rysunek 1.3a są od siebie niezależne, natomiast Rysunek 1.3b przedstawia dwa nakładające się na siebie podproblemy. W przypadku nakładających się na siebie podproblemów możemy znaleźć pary zmiennych, które pośrednio od siebie zależą (Rysunek 1.3b, np. zmienne x_3 i x_5) oraz zmienne decyzyjne, które są częścią więcej niż jednego podproblemu (Rysunek 1.3b, zmienna x_4). Takie zmienne nazywane są zmiennymi współdzielonymi (ang. *shared variables*) [47, 48, 125]. Przykładem problemu, który składa się z nakładających się na siebie podproblemów jest funkcja Rosenbrocka, która określona jest wzorem $f_{rosenbrock}(\mathbf{x}) = \sum_{i=1}^{n-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$. Rysunek 1.4 przedstawia graf interakcji jej ośmiowymiarowej wersji. W zależności od najlepszych wartości zmiennych współdzielonych w kontekście każdego z podproblemów, do których należą, możemy sklasyfikować nakładające się na siebie podproblemy jako pasujące do siebie (ang. *conforming*) lub niepasujące do siebie (ang. *conflicting*) [47, 70, 125]. W przypadku dwóch pasujących do siebie podproblemów, najlepsze wartości ich współdzielonych zmiennych decyzyjnych są takie same dla obu. Jeżeli dwa nakładające się na siebie podproblemy nie pasują do siebie to ich optymalne wartości dla zmiennych współdzielonych mogą się od siebie różnić. Przykładem funkcji z nakładającymi się na siebie podproblemami, które do siebie pasują jest funkcja $\ddot{f}_{c,2}$ określona następującym wzorem: $\ddot{f}_{c,2}(\mathbf{x}) = \ddot{f}_{c,2,1}([x_1, x_2]) + \ddot{f}_{c,2,2}([x_2, x_3]) = [-x_1 \cdot (x_2 + 1)] + [-(x_2 + 2) \cdot x_3]$, której zbiorem dopuszczalnym jest $[-5, 5]^3$. Dla tak zdefiniowanego zbioru dopuszczalnego, $\arg\min_{x_1,x_2\in[-5,5]} \ddot{f}_{c,2,1}([x_1, x_2]) = [5, 5]$ i $\arg\min_{x_2,x_3\in[-5,5]} \ddot{f}_{c,2,2}([x_2, x_3]) = [5, 5]$. Możemy zatem zauważyć, że dla obu podproblemów, ich najlepsza wartość osiągana jest dla $x_2 = 5$. Zmienna x_2 jest współdzieloną zmienną decyzyjną. Za przykład funkcji z nakładającymi się na siebie podproblemami, które nie pasują do siebie niech posłuży funkcja $\ddot{f}_{c,3}$. Przyjmijmy, że jej zbiorem dopuszczalnym jest także $[-5, 5]^3$. Jej wzór to $\ddot{f}_{c,3}(\mathbf{x}) = \ddot{f}_{c,3,1}([x_1, x_2]) + \ddot{f}_{c,3,2}([x_2, x_3]) = [-x_1 \cdot (x_2 + 1)] + [-(x_2 - 1) \cdot x_3]$. Analizując argumenty, dla których podproblemy $\ddot{f}_{c,3,1}$ i $\ddot{f}_{c,3,2}$ osiągają ich minimalne wartości, tj. $\arg\min_{x_1,x_2\in[-5,5]} \ddot{f}_{c,3,1}([x_1, x_2]) = [5, 5]$ i $\arg\min_{x_2,x_3\in[-5,5]} \ddot{f}_{c,3,2}([x_2, x_3]) = [-5, -5]$, możemy zauważyć, że najlepsza wartość współdzielonej zmiennej decyzyjnej x_2 różni się pomiędzy podproblemami ($x_2 = 5$ dla $\ddot{f}_{c,3,1}$ i $x_2 = -5$ dla $\ddot{f}_{c,3,2}$).

Dokładność dekompozycji problemu można zweryfikować np. za pomocą trzech miar zaproponowanych w [77]. Pierwsza miara (ρ_1) odnosi się do znalezionych wzajemnych oddziaływań pomiędzy zmiennymi i jej wartość jest maksymalna, gdy żadna z istniejących zależności nie zostanie pominięta. Druga miara (ρ_2) mierzy zależności, które zostały błędnie wykryte. Im mniej wykrytych interakcji pomiędzy zmiennymi, które w rzeczywistości nie istnieją, tym wyższa jej wartość. Trzecia miara (ρ_3) jest



(a) Separowalne podproblemy (b) Nakładające się na siebie podproblemy

Rysunek 1.3: Grafy interakcji dwóch przykładowych problemów optymalizacyjnych składających się z dwóch podproblemów (na podstawie [125])



Rysunek 1.4: Graf interakcji dla ośmiowymiarowej funkcji Rosenbrocka

połączeniem dwóch poprzednich miar i bierze pod uwagę zarówno niewykryte, jak i błędnie wykryte interakcje pomiędzy zmiennymi. Jeżeli wszystkie istniejące interakcje zostały wykryte i dodatkowo żadna para zmiennych nie została błędnie oznaczona jako zależna, to ρ_3 osiąga wartość maksymalną. Powyższe trzy miary można zapisać następującymi wzorami

$$\rho_1 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta \circ \Theta^*)_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta^*)_{i,j}} \cdot 100\% \tag{1.2}$$

$$\rho_2 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}((\mathbf{1}-\Theta) \circ (\mathbf{1}-\Theta^*))_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}-\Theta^*)_{i,j}} \cdot 100\% \tag{1.3}$$

$$\rho_3 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}-|\Theta-\Theta^*|)_{i,j}}{n\cdot(n-1)/2} \cdot 100\% \tag{1.4}$$

gdzie n jest długością rozważanego problemu optymalizacyjnego (liczbą optymalizowanych zmiennych), Θ^* to idealna macierz interakcji, $\mathbf{1}$ oznacza macierz, która składa się z samych jedynek, natomiast $\circ$ jest iloczynem Hadamarda dwóch macierzy. Pojedynczy element, będący przecięciem i-tego wiersza i j-tej kolumny, macierzy $\boldsymbol{M}$ oznaczony został jako $(\boldsymbol{M})_{i,j}$. Wynikiem operacji $|\Theta - \Theta^*|$ jest macierz, której elementami są $|(\Theta)_{i,j} - (\Theta^*)_{i,j}|$. Należy zauważyć, że sposób, w którym wzory (1.2), (1.3) i (1.4) porównują macierze interakcji, uwzględnia jedynie bezpośrednie interakcje pomiędzy zmiennymi decyzyjnymi.

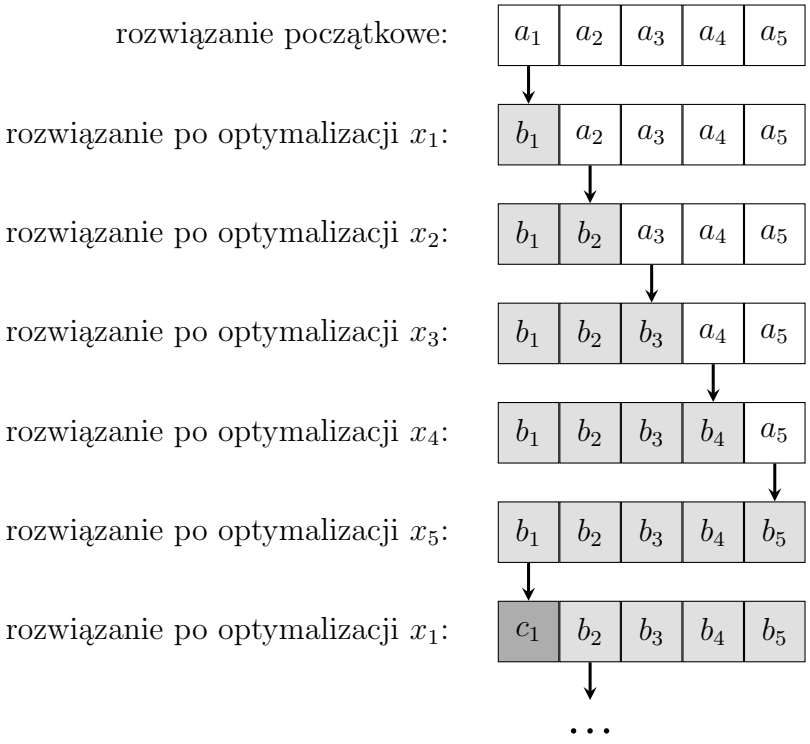
Zaproponowano wiele różnych strategii do tej pory w literaturze m.in. jednostajna dekompozycja (ang. *uniformed decomposition*) [136], losowe grupowanie (ang.

random grouping) [151], grupowanie delta (ang. *delta grouping*) [91], metamodelowanie (ang. *meta modelling*) [75], grupowanie na podstawie wzoru funkcji (ang. *formula-based grouping*) [141], grupowanie różnicowe (ang. *differential grouping*) [89, 94,121,123,125,126] i sprawdzanie monotoniczności (ang. *monotonicity checking*) [14, 29, 120, 143]. Szczegółowe omówienie najbardziej aktualnych w literaturze strategii dekompozycji dedykowanych dla przestrzeni ciągłych znajduje się w rozdziale 2.

1.1.2 Koewolucja kooperatywna

Główną zasadą działania oryginalnej wersji architektury koewolucji kooperatywnej jest oddzielna optymalizacja każdej zmiennej decyzyjnej [101]. Zakładając, że problem składa się z n zmiennych, tworzonych jest n gatunków. W ramach jednego gatunku, osobniki rywalizują ze sobą tylko pod względem jednej zmiennej. W praktyce oznacza to, że utrzymujemy n algorytmów ewolucyjnych, które działają w standardowy sposób z wyjątkiem fazy ewaluacji osobników. W celu ocenienia osobnika potrzebujemy wektor składający się ze wszystkich zmiennych decyzyjnych rozpatrywanego problemu optymalizacyjnego. Skoro każda z użytych metod optymalizacji skupia się tylko na jednej zmiennej decyzyjnej, to rozsądnym wydaje się przyjęcie stałych wartości dla pozostałych zmiennych podczas tej fazy. W przypadku problemów w pełni separowalnych, naturalnym sposobem jest wzięcie tych wartości z dotychczas najlepszego osobnika, którego oznaczmy $\hat{\mathbf{x}}_b = [\hat{x}_{b,1}, ..., \hat{x}_{b,n}]$. Wyliczenie przystosowania osobnika należącego do i-tej populacji odbywa się wtedy na podstawie wektora $\hat{\mathbf{x}} = [\hat{x}_{b,1}, ..., x_i, ..., \hat{x}_{b,n}]$. Podczas rozwiązywania problemów nie w pełni separowalnych i w pełni nieseparowalnych, wybór stałych wartości nie jest już tak oczywisty. W przypadku rozważanej w niniejszej pracy optymalizacji bez jakiejkolwiek wiedzy na temat cech optymalizowanego problemu, koewolucja kooperatywna nie wie jakiego typu problem rozwiązuje, stąd często bierze się wartości z najlepszego znalezionego dotychczas osobnika [47, 74, 92, 101, 149].

Istotnym zagadnieniem, w kontekście architektur kooperacji koewolucyjnej, jest kolejność uruchamiania poszczególnych metod optymalizacji. Pierwsze zaproponowane architektury używają, w tym celu, algorytmu karuzelowego (ang. *round-robin*) [74,101]. Rysunek 1.5 przedstawia ogólną zasadę jego działania na przykładzie omawianego powyżej wariantu kooperacji koewolucyjnej i pięciowymiarowego problemu optymalizacyjnego. Zgodnie z zamieszczonym rysunkiem, zmienne są kolejno optymalizowane, aktualizując tym samym najlepsze znalezione dotychczas rozwiązanie. Zakładając, że jeden cykl koewolucji kooperatywnej to optymalizacja kolejno wszystkich zmiennych decyzyjnych, należy pamiętać, że takich cykli może być wiele. Architektury koewolucji kooperatywnej, podobnie jak algorytmy ewolucyjne, wymagają



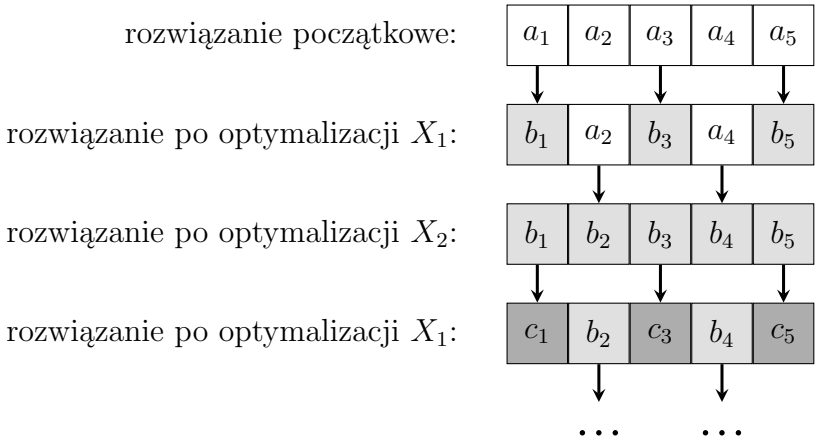
Rysunek 1.5: Ogólny schemat działania oryginalnej wersji koewolucji kooperatywnej

zdefiniowania warunku zatrzymania np. liczba cykli (iteracji) architektury, liczba wyliczeń funkcji przystosowania lub czas obliczeń [125, 149]. W przypadku algorytmu karuzelowego i podstawowej wersji kooperacji koewolucyjnej, budżet obliczeniowy przeznaczony do optymalizacji każdej ze zmiennej powinien być podobny. Jednostka budżetu obliczeniowego jest ściśle związana z warunkiem zatrzymania. Na przykład, gdy warunkiem zatrzymania jest liczba iteracji, to podstawową jednostka budżetu obliczeniowego jest pojedyncza iteracja. Zgodnie z tym założeniem, pojedyncze uruchomienie koewolucji kooperacyjnej mogłoby trwać jeden cykl. Każda zmienna optymalizowana byłaby wtedy tylko raz zużywając od razu cały, przeznaczony dla niej, budżet obliczeniowy. Takie podejście nie jest jednak powszechnie stosowane. Lepszym wyborem wydaje się rozbicie budżetu na kilka cykli. Powodem są, przede wszystkim, możliwe zależności pomiędzy zmiennymi decyzyjnymi. Niektóre zmienne mogą wymagać odpowiednich wartości u części z pozostałych zmiennych by było możliwe znalezienie ich globalnie najlepszych wartości [74, 137]. Kilka pomniejszych procesów optymalizacji tej samej zmiennej może skutkować zatem osiągnięciem lepszego rozwiązania niż pojedyncza optymalizacja z budżetem równym sumie ich budżetów.

Teoretycznie, rozpatrywanie każdej zmiennej osobno nie powinno prowadzić do otrzymania wyników o gorszej jakości. W praktyce, nie tylko skuteczność metody jest

ważna, lecz również jej efektywność. Metody, które znajdują tak samo dobre rozwiązania szybciej, są uznawane za lepsze [56,88,105,109]. W wielu przypadkach, bardziej opłaca się modyfikować kilka zmiennych jednocześnie pomiędzy kolejnymi wyliczeniami funkcji przystosowania [125, 130]. W klasycznym algorytmach ewolucyjnych, stosowany jest w tym celu m.in. operator krzyżowania [3, 135]. Zachowanie odpowiedniej równowagi pomiędzy eksploracją (ang. *exploration*), a eksploatacją (ang. *exploitation*) jest cechą najlepszych metod optymalizacji, które efektywnie i skutecznie rozwiązują problemy optymalizacyjne gdy liczba wymiarów jest stosunkowo niska [21,35,45,116,127]. Często nie jest wtedy dla nich istotne czy optymalizowany problem jest w pełni separowalny, nie w pełni separowalny, czy też w pełni nieseparowalny [5,37]. Rozsądne wydaje się zatem przerzucenie części odpowiedzialności z koewolucji kooperatywnej na używane przez nie metody optymalizacji. Zamiast optymalizować każdą zmienną oddzielnie, można podzielić problem na m rozłącznych komponentów $X_1, ..., X_m$ mogących się składać z więcej niż jednej zmiennej decyzyjnej, które będą także niezależnie optymalizowane [74, 92, 137, 149]. Wszystkie komponenty muszą razem zawierać wszystkie zmienne decyzyjne. Rysunek 1.6 przedstawia tę koncepcję dla dwóch komponentów $X_1 = \{x_1, x_3, x_5\}$ i $X_2 = \{x_2, x_4\}$, gdzie kolejność optymalizacji wyznaczana, tak jak poprzednio, za pomocą algorytmu karuzelowego, gdyż ciągle mogą występować zależności, tym razem pomiędzy komponentami. Dwa komponenty X_p i X_q możemy uznać za zależne, gdy istnieją dwie oddziałujące na siebie wzajemnie zmienne x_p i x_q takie, że $x_p \in X_p$ i $x_q \in X_q$. Skuteczność i efektywność tego typu architektur koewolucji kooperatywnej może być ściśle związana z podziałem problemu optymalizacyjnego na komponenty. Różne podziały tego samego problemu mogą prowadzić do zupełnie innych wyników, szczególnie przy mocno ograniczonym budżecie obliczeniowym [94]. Podział problemu na komponenty może się odbywać na podstawie jego dekompozycji dostarczonej przez jedną z licznych strategii dekompozycji [73,89,93]. Jakość otrzymanej dekompozycji nie zawsze jest wprost proporcjonalna do jakości znalezionego następnie rozwiązania przez koewolucję kooperatywną. W niektórych przypadkach, dobrej jakości dekompozycja może prowadzić do gorszych wyników optymalizacji w porównaniu z dekompozycją o gorszej dokładności [125]. Ten paradoks (lepszej jakości informacja może prowadzić dany optymalizator do gorszych wyników) może prowadzić do wniosku, że potencjał naukowy poszukiwań nowych metod optymalizacji o wyższej skuteczności dedykowanych dla przestrzeni ciągłych pozostaje znaczący.

Sterowanie kolejnością optymalizacji za pomocą algorytmu karuzelowego traktuje równo każdy z komponentów, ponieważ każdy z nich zostanie tyle samo razy optymalizowany w trakcie działania koewolucji kooperatywnej. Należy jednak zauważyć,



Rysunek 1.6: Ogólny schemat działania koewolucji kooperatywnej wykorzystującej algorytm karuzelowy dla problemu optymalizacyjnego podzielonego wcześniej na komponenty

że wpływ procesów optymalizacji na poprawę jakości aktualnie najlepszego rozwiązania może być inny dla różnych komponentów [92]. Optymalizacja części z nich może prowadzić do większej i częstszej poprawy najlepszego rozwiązania niż optymalizacja pozostałych komponentów. Zachłanny wybór komponentu do optymalizacji, na podstawie jego wpływu na optymalizację całego problemu, może znacząco zwiększyć efektywność koewolucji kooperatywnej [92, 124, 149]. Przykładami takich architektur są CBCC (ang. *Contribution-based Cooperative Co-evolution*) [92, 124] i CCFR2 [148], które dla każdego komponentu wyznaczają jego wkład w poprawę jakości najlepszego znalezionego dotychczas rozwiązania w sposób skumulowany. Pod uwagę brane są wszystkie dotychczasowe ulepszenia z zastrzeżeniem, że późniejsze poprawy mają większy wpływ na wartość skumulowanego wkładu (ang. *accumulated contribution*) i-tego komponentu oznaczonego symbolem ΔF_i. Zakładając zadanie minimalizacji funkcji f, w CBCC wartość ΔF_i wyliczana jest w następujący sposób

$$\Delta F_i = w \cdot \Delta \hat{F}_i + (1 - w) \cdot \frac{f(\hat{\mathbf{x}}_b) - f(\mathbf{x}_b)}{f(\hat{\mathbf{x}}_b)} \tag{1.5}$$

gdzie $\Delta \hat{F}_i$ jest poprzednią wartością skumulowanego wkładu, w to parametr podawany przez użytkownika będący współczynnikiem wygładzania (ang. *smoothing factor*), $\hat{\mathbf{x}}_b$ oznacza poprzednie najlepsze znalezione rozwiązanie, natomiast $\mathbf{x}_b$ jest nowym najlepszym rozwiązaniem otrzymanym po optymalizacji i-tego komponentu. Podczas pierwszego cyklu CBCC, wykorzystywany jest algorytm karuzelowy. Wszystkie komponenty są kolejno optymalizowane z budżetem, który jest zadaną liczbą wyliczeń funkcji przystosowania. Na tej podstawie, dla każdego komponentu wyznaczana jest początkową wartość ΔF_i. W kolejnych cyklach CBCC, optymalizowany

jest tylko jeden komponent, także z budżetem obliczeniowym mierzonym zadaną liczbą wyliczeń funkcji przystosowania, który osiągnął najwyższą wartość skumulowanego wkładu. Wzór (1.5) bierze pod uwagę względną poprawę jakości najlepszego rozwiązania [68, 154], żeby wyeliminować potencjalny wpływ wartości przystosowania rozwiązania $\hat{\mathbf{x}}_b$ na wyliczony skumulowany wkład. Zasadność takiego podejścia można pokazać analizując jak zmienia się jakość najlepszego znalezionego dotychczas rozwiązania w różnych etapach procesu optymalizacji. Na samym początku procesu optymalizacji interesuje nas zazwyczaj eksploracja przestrzeni przeszukiwań, natomiast pod koniec skupiamy się często jedynie na eksploatacji. Należy zauważyć, że podczas eksploracji łatwiej o większą poprawę jakości najlepszego znalezionego dotychczas rozwiązania niż podczas eksploatacji.

Alokacja zasobów obliczeniowych w architekturze CBCC nie rozróżnia wymiarów poszczególnych komponentów. Optymalizacja każdego komponentu, niezależnie od jego wielkości, zużywa tyle samo wyliczeń funkcji przystosowania. Istnieją algorytmy ewolucyjne, w szczególności strategie ewolucyjne, które wyznaczają oczekiwany rozmiar populacji na podstawie długości optymalizowanego problemu zakładając, że mniejsze problemy wymagają mniejszych populacji do efektywnego przeszukiwania ich zbioru dopuszczalnego [11, 35]. Zgodnie z tym założeniem, w architekturze CCFR2 [149] wprowadzono alokację zasobów obliczeniowych na podstawie długości każdego z komponentów. Sterowanie alokacją liczby wyliczeń funkcji przystosowania odbywa się za pomocą uruchamiania każdej zarządzanej metody obliczeniowej przez taką samą liczbę iteracji. W przypadku użycia jednakowej metody optymalizacji dla wszystkich komponentów, której rozmiar populacji jest wprost proporcjonalny do długości optymalizowanego problemu, dłuższe komponenty będą miały do dyspozycji większą liczbę wyliczeń funkcji przystosowania. Skumulowany wkład w architekturze CCFR2, którego wartość jest następnie podstawą do podjęcia decyzji o kolejnym komponencie do optymalizacji, bierze pod uwagę, że różne komponenty mogą mieć do dyspozycji różną liczbę wyliczeń funkcji przystosowania. Wartość skumulowanego wkładu bazuje na bezwzględnej poprawie jakości najlepszego znalezionego dotychczas rozwiązania na pojedyncze wyliczenie funkcji przystosowania i jest wyliczana używając następującego wzoru

$$\Delta F_i = w \cdot \Delta \hat{F}_i + \frac{f(\hat{\mathbf{x}}_b) - f(\mathbf{x}_b)}{n_i} \tag{1.6}$$

gdzie n_i jest liczbą wyliczeń funkcji przystosowania, która została wykorzystana do optymalizacji i-tego komponentu. Wzór (1.6) zakłada zadanie minimalizacji funkcji f. Pozostałe mechanizmy w CCFR2 są identyczne jak w CBCC, z wyjątkiem

warunku zatrzymania optymalizacji każdego z komponentów. Komponenty optymalizowane są przez zadaną liczbę iteracji, a nie do momentu osiągnięcia zadanej liczby wyliczeń funkcji przystosowania.

Architektury koewolucji kooperatywnej nie narzucają użycia konkretnych metod do optymalizacji poszczególnych komponentów. Istnieje podejście, które pozwala na użycie kilku metod do optymalizacji tego samego komponentu, gdzie częstość wyboru konkretnej metody zależy np. od odpowiadającej jej wartości skumulowanego wkładu [124].

1.2 Hybrydowe metody optymalizacji

Drugim popularnym podejściem do rozwiązywania wielowymiarowych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań, które nie wymaga wcześniejszej dekompozycji problemu, jest wykorzystanie hybrydowych metod optymalizacji. Przykładem takiej metody jest MOS (ang. *Multiple Offspring Sampling*) [65, 66], która w latach 2013–2017 była niepokonana w konkursie na najlepszą metodę optymalizacji zaprojektowaną do rozwiązywania wielowymiarowych problemów o ciągłej przestrzeni przeszukiwań, organizowanego w ramach konferencji CEC[1]. Została ona pokonana w 2018 roku przez metodę SHADE-ILS (ang. *Success-History Based Parameter Adaptation for Differential Evolution with Iterative Local Search*) [82], która także reprezentuje tego typu podejście.

Hybrydowe metody optymalizacji należy traktować jako wysokopoziomowe architektury, które zarządzają kilkoma różnymi metodami w ramach tego samego procesu optymalizacji [33, 65, 66, 82]. Poziom udziału każdej z zarządzanych metod optymalizacji, w całym procesie optymalizacji, jest zazwyczaj wprost proporcjonalny do ich wkładu w poprawę najlepszego znalezionego dotychczas rozwiązania. Skoro używane są różne metody, można przypuszczać, że będą one odpowiednie do problemów o różnej charakterystyce. Hybrydowe metody optymalizacji mają zatem na celu takie dostrojenie udziału każdej z zarządzanych metod, aby promowane były te metody, które lepiej sobie radzą z charakterystyką aktualnie rozwiązywanego problemu optymalizacyjnego. Poziomem udziału można sterować na wiele różnych sposobów m.in.:

– różną liczbą osobników tworzonych przez metody optymalizacji, które następnie zasilają globalną populację [65, 66],

– różnymi prawdopodobieństwami uruchomienia każdej z metod [82],

[1] `https://www.tflsgo.org/download/comp2018_slides.pdf`

– różnym budżetem obliczeniowym przeznaczonym na optymalizację przy użyciu każdej z metod [33].

Zasada działania hybrydowych metod optymalizacji może przypominać idee stojące za zachłannym wyborem komponentu do optymalizacji przez niektóre architektury koewolucji kooperatywnej np. CBCC [92, 124] lub CCFR2 [149], które zostały opisane w rozdziale 1.1.2. Zarówno w CBCC, jak i w CCFR2, jest także kilka metod optymalizacji, które uruchamiane są częściej, gdy metoda ma większy udział w poprawie najlepszego znalezionego dotychczas rozwiązania. Z drugiej strony, różnią się one od hybrydowych metod optymalizacji przynajmniej jednym istotnym szczegółem, który zostanie omówiony na przykładzie problemu składającego się z separowalnych podproblemów o różnej charakterystyce. W przypadku dostarczenia do CBCC lub CCFR2 idealnej dekompozycji problemu, każda z zarządzanych przez nie metod optymalizacji skupi się na rozwiązywaniu komponentu, który jest niezależny od innych. Przetwarzając jedynie te zmienne, które oddziałują na siebie wzajemnie, metoda będzie mogła dostroić swoje parametry do cech konkretnego podproblemu. Metody zarządzane przez hybrydowe metody optymalizacji przetwarzają natomiast wszystkie zmienne decyzyjne jednocześnie lub ich losowy podzbiór. W przypadku gdy podproblemy mają różny wpływ na jakość rozwiązania całego problemu, udział zarządzanych metod optymalizacji będzie najprawdopodobniej preferował te metody, które lepiej radzą sobie z charakterystyką bardziej wpływowych podproblemów.

Ze względu na fakt, że SHADE-ILS osiąga najlepsze wyniki spośród hybrydowych metod optymalizacji [82], zostanie ona, jako jedyna, bardziej szczegółowo opisana. SHADE-ILS rozszerza metodę SHADE (ang. *Success-History Based Parameter Adaptation for Differential Evolution*) [127] o wykorzystanie dwóch metod lokalnego przeszukiwania. SHADE, której podstawą jest ewolucja różnicowa (ang. *differential evolution*) [119], w ramach SHADE-ILS odpowiada za globalną eksplorację w celu znalezienia obiecujących regionów w zbiorze dopuszczalnym. Autorzy SHADE-ILS wskazują, że nie użyli ulepszonej wersji SHADE zwanej L-SHADE (ang. *Success-history based parameter adaptation of Differential Evolution using linear population size reduction algorithm*) [100], ponieważ wprowadza ona mechanizmy, które zwiększają zbieżność metody. W SHADE-ILS funkcję tę pełnią natomiast dwie metody lokalnego przeszukiwania, którymi są MTS-L1 [72] i L-BFGS-B [83]. Wybór ich nie jest przypadkowy, ponieważ uzupełniają się one wzajemnie z powodu różnych mocnych i słabych stron. MTS-L1 jest bardzo efektywną metodą, która skutecznie rozwiązuje w pełni separowalne problemy, lecz jest mało odporna na zależności pomiędzy zmiennymi. Z kolei L-BFGS-B jest klasyczną metodą optymalizacji, która aproksy-

muje gradient podczas przeszukiwania zbioru dopuszczalnego. Jest ona zatem mniej czuła na zależności pomiędzy zmiennymi, ale także mniej efektywna, szczególnie gdy żadne zależności pomiędzy zmiennymi nie występują. Pojedyncza iteracja SHADE-ILS składa się z dwóch części. W pierwszej fazie, metoda SHADE jest za każdym razem uruchamiana na określoną liczbę wyliczeń funkcji przystosowania. Następnie, najlepsze znalezione dotychczas rozwiązanie jest ulepszane za pomocą MTS-L1 lub L-BFGS-B, także przez określoną liczbę wyliczeń funkcji przystosowania. Wybór metody lokalnego przeszukiwania odbywa się na podstawie prawdopodobieństwa przypisanego do każdej z nich. Im większy wpływ na dotychczasowe polepszanie jakości najlepszego znalezionego rozwiązania, tym większe prawdopodobieństwo wyboru konkretnej metody lokalnego przeszukiwania w późniejszych iteracjach SHADE-ILS. Kolejnym mechanizmem, który został wprowadzony do SHADE-ILS i sprawdza się podczas rozwiązywania wielowymiarowych problemów optymalizacyjnych, jest restartowanie stanu wszystkich trzech metod optymalizacji i rozpoczęcie przeszukiwania zbioru dopuszczalnego od początku. Decyzja o przeprowadzeniu restartu jest podejmowana na podstawie względnej poprawy jakości najlepszego znalezionego dotychczas rozwiązania. Dla zadania minimalizacji funkcji f, współczynnik ten wyliczany jest następująco

$$\frac{f(\hat{\mathbf{x}}_b) - f(\mathbf{x}_b)}{f(\hat{\mathbf{x}}_b)} \tag{1.7}$$

gdzie $\hat{\mathbf{x}}_b$ i $\mathbf{x}_b$ oznaczają kolejno najlepsze rozwiązanie przed i po optymalizacji. W przypadku gdy wartość względnej poprawy, wyliczona na podstawie powyższego wzoru, jest mniejsza niż 5% dla trzech kolejnych iteracji metody SHADE-ILS.

Rozdział 2

Strategie dekompozycji ciągłych problemów optymalizacyjnych

W poprzednim rozdziale omówione zostało pojęcie dekompozycji ciągłych problemów optymalizacyjnych, a ponadto przedstawiony został przykładowy sposób wykorzystania dekompozycji do optymalizacji tego typu problemów. W niniejszym rozdziale zostaną natomiast opisane strategie, które dekomponują dany problem na mniejsze podproblemy. Szczegółowo zostaną opisane strategie dekompozycji wywodzące się z grupowania różnicowego oraz bazujące na sprawdzaniu monotoniczności. Ich wady były motywacją dla prac nad nową strategią dekompozycji. Pod koniec niniejszego rozdziału, poruszona zostanie tematyka związana z ideą „linkage learning". Idea ta jest podstawą prawie każdej strategii dekompozycji oraz jest istotnym elementem wielu aktualnie wiodących metod optymalizacji, które są dedykowane przestrzeniom przeszukiwań o innym typie m.in. binarnym.

2.1 Grupowanie różnicowe

Modularność, która jest cechą wielu praktycznych problemów optymalizacyjnych [134], można zamodelować za pomocą addytywnej separowalności [89]. Funkcja $f : \mathcal{D} \to \mathbb{R}$ jest addytywnie separowalna jeżeli jest sumą $m \geq 2$ niezależnych składowych

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x_i}) \tag{2.1}$$

gdzie $\forall_{i\in\{1,\ldots,m\}}\mathbf{x_i} \in \mathcal{D}_i$ i $\mathcal{D} = \mathcal{D}_1 \times ... \times \mathcal{D}_m$. Należy zauważyć, że każda funkcja addytywnie separowalna jest także w pełni lub nie w pełni separowalna, ponieważ dla każdej takiej funkcji spełniony jest warunek zdefiniowany we wzorze (1.1).

Grupowanie różnicowe (ang. *Differential Grouping*, DG) [89] zostało zaprojektowane do dekomponowania problemów optymalizacyjnych, które składają się z addytywnie separowalnych podproblemów. Autorzy grupowania różnicowego udowodnili formalnie, że jeżeli funkcja f jest addytywnie separowalna to dwie zmienne decyzyjne x_p i x_q uznaje się za bezpośrednio zależne wtedy i tylko wtedy, gdy $\exists a, b_1 \neq b_2, \delta > 0, \ddot{\mathbf{x}} \in \mathcal{D}$ następujący warunek jest spełniony

$$\Delta_{\delta,x_p}[f](\ddot{\mathbf{x}})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\ddot{\mathbf{x}})|_{x_p=a,x_q=b_2} \tag{2.2}$$

gdzie x_p i x_q oznaczają odpowiednio p-tą i q-tą zmienną dekomponowanego problemu, podczas gdy wartość funkcji $\Delta_{\delta,x_p}[f]$ wyliczana jest zgodnie ze wzorem

$$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(..., x_p + \delta, ...) - f(..., x_p, ...) \tag{2.3}$$

Każda perturbacja p-tej zmiennej decyzyjnej, tj. $x_p + \delta$, powinna zwrócić wartość ze zbioru dopuszczalnego. Użyta we wzorze (2.2) notacja $f(\mathbf{x})|_{x_p=a,x_q=b}$ oznacza wartość funkcji f wyliczonej dla wektora $\mathbf{x}$ po wcześniejszej zmianie wartości p-tej zmiennej na wartość a i analogicznie q-tej zmiennej na wartość b. Wartości a i b powinny pochodzić ze zbioru dopuszczalnego odpowiednio p-tej i q-tej zmiennej. Zastosowanie grupowania różnicowego to głównie dekompozycja ciągłych problemów optymalizacyjnych, zatem wartości a, b_1, b_2 i δ jest nieskończenie wiele. Z tego powodu przyjmuje się, że do podjęcia decyzji o istnieniu interakcji pomiędzy dwoma zmiennymi decyzyjnymi wystarczy sprawdzenie tylko jednej czwórki wartości.

Pomimo formalnego dowodu, mogą zdarzyć się przypadki, dla których grupowanie różnicowe podejmie błędną decyzję o istnieniu interakcji pomiędzy zmiennymi funkcji addytywnie separowalnej. Powodem jest niedoskonałość reprezentacji liczb zmiennoprzecinkowych. Z tego względu, wzór (2.2) należy przeformułować na nierówność. Oznaczmy lewą stronę wzoru (2.2) jako Δ_1 i analogicznie prawą jako Δ_2. wtedy zamiast $\Delta_1 \neq \Delta_2$ możemy sprawdzać prawdziwość nierówności $|\Delta_1 - \Delta_2| > \epsilon$, gdzie ϵ jest parametrem podawanym przez użytkownika i kontroluje czułość wykrywania zależności przez grupowanie różnicowe. Odpowiednia wartość parametru ϵ może być inna dla różnych problemów optymalizacyjnych [94].

Powyższy opis przedstawia zasadę wykrywania bezpośrednich interakcji pomiędzy parą zmiennych. Celem dekompozycji jest natomiast znalezienie grup zmiennych, które wzajemnie od siebie zależą. Z tego powodu, w grupowaniu różnicowym, wiele różnych par zmiennych jest sprawdzanych pod kątem możliwej zależności. Pojedyncze sprawdzenie wymaga czterech wyliczeń funkcji przystosowania. W celu zmniejszenia wymaganej liczby wyliczeń funkcji przystosowania, Autorzy grupowa-

nia wykorzystują następującą obserwację odnośnie przechodniości interakcji. Jeżeli zmienne x_p i x_q oddziałują na siebie wzajemnie i istnieje zależność pomiędzy x_p i x_r to wnioskujemy, że zmienne x_q i x_r także są od siebie zależne. Rozpatrzmy problem optymalizacyjny, dla którego da się wyodrębnić podproblem, którego zmienne $\{x_1, x_2, x_3, x_4\}$ oddziałują na siebie wzajemnie. Jeżeli wykryjemy interakcję pomiędzy następującymi parami zmiennych (x_1, x_2), (x_1, x_3) oraz (x_1, x_4) to nie musimy sprawdzać pozostałych par by stwierdzić, że zmienne od x_1 do x_4 są od siebie wzajemnie zależne. Niemniej jednak, nie wiemy czy zależność jest bezpośrednia, czy pośrednia. Grupowanie różnicowe grupuje zależne zmienne zgodnie z powyższą zasadą. Na samym początku, wykrywane są wszystkie bezpośrednie interakcje z pierwszą zmienną. W tym celu sprawdzana jest bezpośrednia zależność pomiędzy każdą parą zmiennych (x_1, x_q), gdzie $q \in 2, ..., n$. Wynikiem tego kroku jest grupa X_1, która składa się ze zmiennej x_1 oraz wszystkich innych zmiennych decyzyjnych, dla których zachodzi bezpośrednia interakcja z x_1. W kolejnych krokach grupowania różnicowego, następna zmienna decyzyjna, która nie należy do żadnej z utworzonych wcześniej grup, jest w podobny sposób przetwarzana. Jedyna różnica jest następująca. Pary zmiennych, które następnie będą sprawdzane pod kątem bezpośredniej zależności, uwzględniają także jedynie te zmienne, które nie należą do żadnej z utworzonych wcześniej grup.

Najlepszą złożoność obliczeniową, mierzoną liczbą wyliczeń funkcji przystosowania, grupowanie różnicowe osiąga w przypadku problemów w pełni nieseparowanych. Grupowanie różnicowe zużywa wtedy $4(n-1)$ wyliczeń funkcji przystosowania. Najgorszym przypadkiem są natomiast problemy w pełni separowalne, ponieważ grupowanie różnicowe potrzebuje wtedy aż $2n(n-1)$ wyliczeń funkcji przystosowania. Na tej podstawie możemy stwierdzić, że złożoność obliczeniowa grupowania różnicowego to $\mathcal{O}(n^2)$.

Jedną z dostrzeżonych wad grupowania różnicowego jest konieczność podawania przez użytkownika wartości parametru ϵ [94]. Dodatkowo, wartość ϵ jest stała dla wszystkich sprawdzeń, czy istnieje bezpośrednia zależność pomiędzy parą zmiennych. DG2 (ang. *Differential Grouping 2*) [94] jest propozycją, która eliminuje m.in. tę wadę grupowania różnicowego. W DG2, odpowiednia wartość ϵ jest automatycznie wyliczana, oddzielnie dla każdego sprawdzenia bezpośredniej interakcji. Niech $f(\ddot{\mathbf{x}}_\mathbf{1})$, $f(\ddot{\mathbf{x}}_\mathbf{2})$, $f(\ddot{\mathbf{x}}_\mathbf{3})$ i $f(\ddot{\mathbf{x}}_\mathbf{4})$ oznaczają wartość funkcji przystosowania odpowiednio w punktach $\ddot{\mathbf{x}}_\mathbf{1}$, $\ddot{\mathbf{x}}_\mathbf{2}$, $\ddot{\mathbf{x}}_\mathbf{3}$ i $\ddot{\mathbf{x}}_\mathbf{4}$. Gdy powyższe cztery wartości biorą udział w wyliczeniu wartości Δ_1 i Δ_2 to wzór na ϵ z nierówności $|\Delta_1 - \Delta_2| > \epsilon$ jest następujący

$$\epsilon = f_\gamma(\sqrt{n} + 2) \cdot (|f(\ddot{\mathbf{x}}_\mathbf{1})| + |f(\ddot{\mathbf{x}}_\mathbf{2})| + |f(\ddot{\mathbf{x}}_\mathbf{3})| + |f(\ddot{\mathbf{x}}_\mathbf{4})|) \tag{2.4}$$

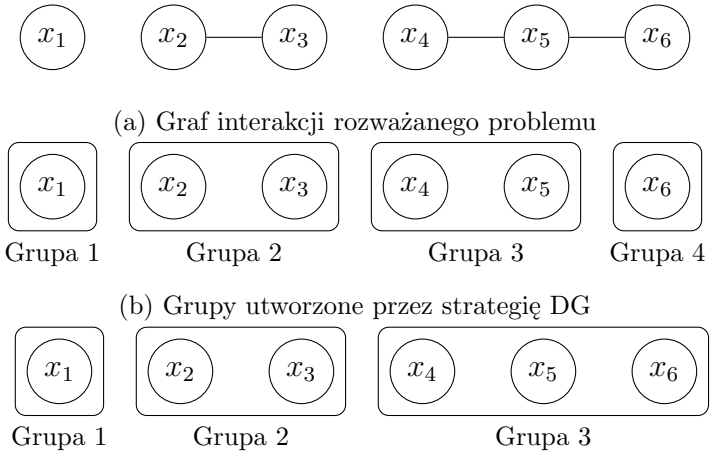
gdzie n to rozmiar rozpatrywanego problemu optymalizacyjnego, a f_γ to funkcja, która wykorzystuje stałą zależną od urządzenia μ_M[1] [1] i jest zdefiniowana jako

$$f_\gamma(k) = \frac{k\mu_M}{1 - k\mu_M} \tag{2.5}$$

W przypadku problemów z nakładającymi się na siebie podproblemami, nie istnieje ich idealna i zarazem unikalna dekompozycja [94, 105]. Rysunek 2.1a przedstawia graf interakcji problemu, w którym zmienne x_4 i x_6 pośrednio od siebie zależą. Możemy zatem wyodrębnić dwa podproblemy, które nakładają się na siebie: $\{x_4, x_5\}$ i $\{x_5, x_6\}$. Gdy użyjemy grupowania różnicowego do dekompozycji tego problemu i założymy, że podczas jego działania zmienne będą przetwarzane w kolejności od x_1 do x_6 to zmienne decyzyjne należące do tych dwóch nakładających się podproblemów zostaną podzielone na dwie grupy: $\{x_4, x_5\}$ oraz $\{x_6\}$. Sytuacja ta została zaprezentowana na rysunku Rysunek 2.1b. Przyczyną takiego grupowania zmiennych jest fakt, że zmienne x_4 i x_6 oddziałują na siebie wzajemnie w sposób pośredni, a nie bezpośredni. Grupowanie różnicowe nie posiada mechanizmów, które pozwoliłyby stwierdzić, że istnieje pośrednia interakcja pomiędzy daną parą zmiennych. Skoro nie istnieje idealna dekompozycja gdy problem składa się m.in. z podproblemów, które nakładają się na siebie to DG2 łączy wszystkie zmienne, które bezpośrednio lub pośrednio oddziałują na siebie wzajemnie w jedną grupę. W DG2, w tym celu, każda para zmiennych decyzyjnych jest sprawdzana pod kątem bezpośredniej zależności. Mając do dyspozycji macierz interakcji, w której oznaczone są wszystkie możliwe bezpośrednie interakcje pomiędzy zmiennymi, możemy wyznaczyć wszystkie spójne składowe odpowiadającego jej grafowi interakcji. W DG2, wykorzystywany jest algorytm, który wyznacza wszystkie spójne składowe w czasie liniowym w zależności od n, czyli od długości rozważanego problemu optymalizacyjnego [41]. Graf interakcji przedstawiony na rysunku Rysunek 2.1a posiada trzy spójne składowe. Strategia DG2 znajduje zatem trzy odpowiadające im grupy zmiennych decyzyjnych (Rysunek 2.1c).

Ze względu na fakt, że w DG2 sprawdzane są zawsze wszystkie możliwe pary zmiennych pod kątem bezpośredniej interakcji, liczba wyliczeń funkcji przystosowania w trakcie działanie tej strategii dekompozycji zależy jedynie od liczby wymiarów rozpatrywanego problemu. W przypadku grupowania różnicowego, istotna była także wewnętrzna struktura problemu. Wszystkich par zmiennych w n-wymiarowym problemie jest $\binom{n}{2} = \frac{n(n-1)}{2}$. Skoro każde sprawdzenie bezpośredniej interakcji wymaga czterech wartości funkcji przystosowania to należy oczekiwać, że cała proce-

[1] W środowisku MATLAB, wartość μ_M jest równa 2^{-53} [126].



(a) Graf interakcji rozważanego problemu

(b) Grupy utworzone przez strategię DG

(c) Grupy utworzone przez strategię DG2

Rysunek 2.1: Różnice w grupowaniu problemów zawierających nakładające się na siebie podproblemy

dura grupowania zużyje $2n(n-1)$ wyliczeń funkcji przystosowania. Autorzy DG2 zauważyli jednak, że część wyliczonych już wartości funkcji przystosowania można użyć ponownie do sprawdzenia innej pary zmiennych pod kątem bezpośredniej zależności. Na przykład, niech $\ddot{f}_{c,4}$ będzie dowolną ciągłą funkcją trzech zmiennych, natomiast wartości a, b i c niech będą elementami zbiorów dopuszczalnych zmiennych x_1, x_2 i x_3. Dodatkowo, niech perturbacje zmiennych x_1, x_2 i x_3 prowadzą do otrzymania wartości odpowiednio $a' = a + \delta$, $b' = b + \delta$ i $c' = c + \delta$. Przy takich założeniach, wartości Δ_1 i Δ_2 dla każdej możliwej pary zmiennych wyliczane są w następujący sposób

– (x_1, x_2): $\Delta_1 = \ddot{f}_{c,4}([a', b, c]) - \ddot{f}_{c,4}([a, b, c])$, $\Delta_2 = \ddot{f}_{c,4}([a', b', c]) - \ddot{f}_{c,4}([a, b', c])$

– (x_1, x_3): $\Delta_1 = \ddot{f}_{c,4}([a', b, c]) - \ddot{f}_{c,4}([a, b, c])$, $\Delta_2 = \ddot{f}_{c,4}([a', b, c']) - \ddot{f}_{c,4}([a, b, c'])$

– (x_2, x_3): $\Delta_1 = \ddot{f}_{c,4}([a, b', c]) - \ddot{f}_{c,4}([a, b, c])$, $\Delta_2 = \ddot{f}_{c,4}([a, b', c']) - \ddot{f}_{c,4}([a, b, c'])$

Dobierając odpowiednio wektory, dla których będzie wyliczana wartość funkcji przystosowania, zamiast dwunastu wyliczeń można ich wykonać jedynie siedem. Dzięki tej obserwacji, strategia DG2 potrzebuje $\frac{n(n+1)}{2} + 1$ wyliczeń funkcji przystosowania by zdekomponować n-wymiarowy problem optymalizacyjny. Jeżeli rozpatrywany problem jest w pełni separowalny to wyliczeń będzie około dwa razy mniej niż w przypadku użycia strategii DG.

2.2 Rekursywne grupowanie różnicowe

Złożoność obliczeniowa strategii DG i DG2 to $\mathcal{O}(n^2)$. Taka złożoność może być nieakceptowalna w przypadku wielowymiarowych problemów optymalizacyjnych. Z tego powodu, jedną z kolejnych propozycji strategii dekompozycji bazującej na grupowaniu różnicowym jest RDG (ang. *Recursive Differential Grouping*) [123]. Jej złożoność jest mniejsza niż złożoność obliczeniowa strategii DG i DG2, ponieważ wynosi $\mathcal{O}(n\log(n))$. Jest to możliwe dzięki sprawdzaniu ewentualnej interakcji pomiędzy dwoma zbiorami zmiennych, zamiast rozważania jedynie dwóch zmiennych. Oznaczmy jako X_1 i X_2 dwa rozłączne podzbiory zbioru wszystkich zmiennych decyzyjnych $X = \{x_1, ..., x_n\}$. Podzbiory X_1 i X_2 oddziałują na siebie wzajemnie gdy istnieje przynajmniej jedna para bezpośrednio zależnych od siebie zmiennych (x_p, x_q), takich że $x_p \in X_1$ i $x_q \in X_2$. Zakładając, że funkcja f jest addytywnie separowalna oraz zachodzi interakcja pomiędzy X_1 i X_2 to istnieją takie wartości $\delta_1 > 0$ i $\delta_2 > 0$, wektory jednostkowe $\mathbf{u_1} \in U_{X_1}$ i $\mathbf{u_2} \in U_{X_2}$ oraz wektor zmiennych decyzyjnych $\ddot{\mathbf{x}} \in \mathcal{D}$, które spełniają następujący warunek

$$f(\ddot{\mathbf{x}} + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2}) - f(\ddot{\mathbf{x}} + \delta_2\mathbf{u_2}) \neq f(\ddot{\mathbf{x}} + \delta_1\mathbf{u_1}) - f(\ddot{\mathbf{x}}) \tag{2.6}$$

gdzie $\mathbf{u_j} = [u_1, ..., u_n] \in U_{X_j}$ jest wektorem jednostkowym, takim że

$$\forall_{i\in\{1,...,n\}} u_i = 0 \iff x_i \notin X_j \tag{2.7}$$

Dowód powyższego twierdzenia można znaleźć w [123].

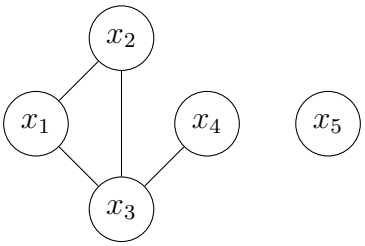
Rysunek 2.2 przedstawia ogólną koncepcję strategii RDG. Formowanie grup zależnych zmiennych odbywa się, w niej, w sposób rekursywny. Na samym początku sprawdzamy, czy pierwsza zmienna decyzyjna jest bezpośrednio zależna od jakiejkolwiek innej zmiennej. W rozważanym, w tym przykładzie, problemie jest to sprawdzenie, czy dwie grupy zmiennych $X_1 = \{x_1\}$ i $X_2 = \{x_2, x_3, x_4, x_5\}$ oddziałują na siebie wzajemnie. Jeżeli istnieje pomiędzy nimi interakcja to ciągle nie wiemy, które konkretnie zmienne ze zbioru X_2 zależą bezpośrednio od zmiennej x_1. W tym celu dzielimy zbiór X_2 na dwa równe podzbiory $X_2^1 = \{x_2, x_3\}$ i $X_2^2 = \{x_4, x_5\}$, a następnie wykonujemy dwa dodatkowe sprawdzenia pomiędzy X_1 i X_2^1 oraz X_1 i X_2^2 pod kątem ich ewentualnej interakcji. Dzielenie zbioru na pół powtarzamy, aż do momentu gdy nie wykryjemy jego interakcji ze zbiorem X_1 lub jest on zbiorem jednoelementowym. Następnie, wszystkie zmienne z jednoelementowych zbiorów, dla których wykryta została interakcja ze zmienną x_1, zostają dodane do X_1 i zarazem usunięte z X_2. W prezentowanym przykładzie, na koniec pierwszej iteracji RDG, oba

zbiory będą posiadały następujące elementy $X_1 = \{x_1, x_2, x_3\}$ oraz $X_2 = \{x_4, x_5\}$. W RDG, problemy z nakładającymi się na siebie podproblemami są traktowane w taki sam sposób, jak w przypadku strategii DG2. Oczekujemy, że wszystkie zmienne, które oddziałują na siebie wzajemnie utworzą jedną grupę bez względu na charakter zależności — pośrednia lub bezpośrednia. Z tego względu, w kolejnej iteracji RDG (Rysunek 2.2c), ponawiamy rekursywną procedurę dla powiększonego zbioru X_1 i pomniejszonego X_2. Podczas jej trwania wykrywamy, że zmienna x_4 jest bezpośrednio zależna przynajmniej od jednej zmiennej ze zbioru X_2, najprawdopodobniej od x_2 lub x_3. W kontekście zwracanych grup przez strategię RDG, nie musimy dociekać, od której konkretnie zmiennej, zmienna x_4 jest zależna. Jeżeli w pierwszym kroku dowolnej iteracji RDG nie wykryjemy interakcji pomiędzy X_1 i X_2 to ze zmiennych będących elementami zbioru X_1 tworzymy grupę zmiennych zależnych, a następnie usuwamy wszystkie zmienne ze zbioru X_1 i dodajemy do niego pierwszą zmienną ze zbioru X_2 usuwając ją zarazem z X_2. Następnie, ponownie uruchamiana jest rekursywna procedura. Strategia RDG kończy swoje działanie, gdy ze zbioru X_2 zostanie usunięty ostatni element. Dla problemu, którego graf interakcji został zaprezentowany na rysunku Rysunek 2.1a, strategia RDG utworzy dokładnie takie same grupy jak strategia DG2 (Rysunek 2.1c), jednakże wykorzysta w tym celu mniejszą liczbę wyliczeń funkcji przystosowania.

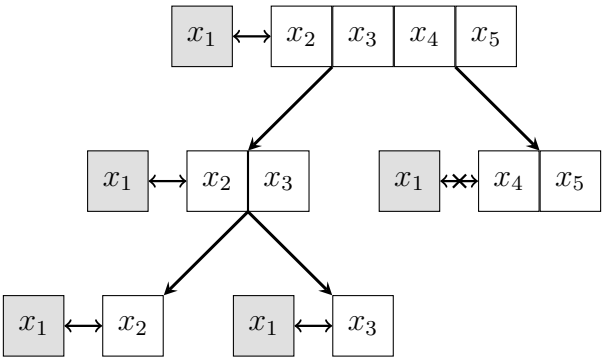
W przypadku strategii RDG, liczba potrzebnych wyliczeń funkcji przystosowania by zdekomponować podany problem optymalizacyjny zależy od jego wewnętrznej struktury. W [123] przeanalizowane zostały następujące rodzaje problemów, gdzie wszystkie zaprezentowane analizy zakładają, że RDG nie wykryje żadnych nieistniejących zależności oraz nie pominie żadnych interakcji, które faktycznie istnieją.

1. Problemy w pełni separowalne: dla każdej zmiennej decyzyjnej przeprowadzone zostanie dokładnie jedno sprawdzenie, które wykryje, że zmienna ta nie jest zależna od żadnej innej zmiennej. Pojedyncze sprawdzenie ewentualnej interakcji w RDG to koszt trzech wyliczeń funkcji przystosowania. Wzór (2.6) sugeruje, że wymagane są cztery wyliczenia, jednakże używając tego samego $\ddot{\mathbf{x}}$ we wszystkich sprawdzeniach, możemy wyliczyć wartość $f(\ddot{\mathbf{x}})$ jedynie raz podczas całego działania strategii RDG. Skoro problem składa się z n zmiennych decyzyjnych to, w przypadku problemów w pełni separowalnych, RDG potrzebuje około $3n$ wyliczeń wartości funkcji przystosowania.

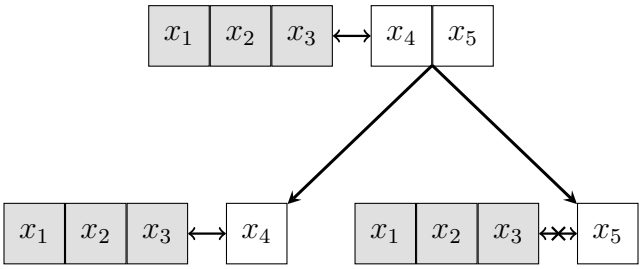
2. Problemy w pełni nieseparowalne: rekursywne dzielenie zbioru X_2 na pół powoduje, że sprawdzenie ewentualnej interakcji odbywa się około $\sum_{i=0}^{k} \frac{n}{2^i}$ razy,



(a) Graf interakcji rozważanego problemu



(b) Pierwsza iteracja strategii RDG



(c) Druga iteracja strategii RDG

Rysunek 2.2: Schemat podstawowej zasady działania strategii RDG

gdzie $k = \log_2(n)$. Wartość $\sum_{i=0}^{k}$ możemy ograniczyć z góry

$$\sum_{i=0}^{k} \frac{n}{2^i} = n\left(2 - \left(\frac{1}{2}\right)^k\right) < 2n \tag{2.8}$$

Możemy zatem stwierdzić, ze dekompozycja problemów w pełni nieseparowalnych wymaga około $6n$ wyliczeń funkcji przystosowania.

3. Problemy nie w pełni separowalne składające się z m, w pełni nieseparowalnych, podproblemów o rozmiarze $l = \frac{n}{m}$: grupowanie l nieseparowalnych zmiennych decyzyjnych w jedną grupę wymaga mniej niż $2l\log_2(n)$ sprawdzeń ewentualnej interakcji. Skoro oczekujemy, że strategia RDG zwróci m grup to wszystkich sprawdzeń będzie mniej niż $2l\log_2(n) \cdot m = 2n\log_2(n)$. Całkowity koszt jest zatem mniejszy niż $6n\log_2(n)$ wyliczeń funkcji przystosowania.

4. Problemy nie w pełni separowalne z jednym, w pełni nieseparowalnym podproblemem o rozmiarze l: do identyfikacji $n-l$ separowalnych zmiennych decyzyjnych, strategia RDG potrzebuje około $3(n-l)$ wyliczeń funkcji przystosowania. Wykrycie pojedynczej grupy składającej się z l oddziałujących na siebie wzajemnie zmiennych wymaga mniej niż $6l\log_2(n)$ wyliczeń funkcji przystosowania. Całkowity koszt jest zatem mniejszy niż $3(n-l) + 6l\log_2(n)$ wyliczeń funkcji przystosowania.

5. Funkcja Rosenbrocka [70]: jest to problem z nakładającymi się na siebie podproblemami o rozmiarze 3. Graf interakcji ośmiowymiarowej funkcji Rosenbrocka został zaprezentowany na rysunku Rysunek 1.4. Zaczynając od zmiennej x_i potrzebujemy około $3 \cdot 2 \cdot 2\log_2(n) = 12\log_2(n)$ wyliczeń funkcji przystosowania by wykryć, że zachodzi interakcja pomiędzy x_i, a x_{i-1} oraz x_{i+1}. Kolejne $12\log_2(n)$ wyliczeń jest potrzebne by znaleźć dwie zmienne x_{i-2} i x_{i+2}, które oddziałują na siebie wzajemnie ze zmiennymi ze zbioru $\{x_{i-1}, x_i, x_{i+1}\}$. W ten sposób dobieramy do zbioru zależnych zmiennych kolejne dwie skrajne zmienne, aż do uzyskania grupy składającej się ze wszystkich zmiennych decyzyjnych rozważanego n-wymiarowego problemu optymalizacyjnego. Całkowity koszt dekompozycji funkcji Rosenbrocka jest zatem zbliżony do $\frac{n}{2} \cdot 12\log_2(n) = 6\log_2(n)$ wyliczeń funkcji przystosowania.

Powyższa analiza potwierdza, że złożoność obliczeniowa strategii RDG mierzona liczbą wyliczeń funkcji przystosowania to $\mathcal{O}(n\log(n))$.

Ze względów praktycznych, dokładnie z powodu niedoskonałości reprezentacji liczb zmiennoprzecinkowych, wzór (2.6) należy przeformułować na nierówność wy-

korzystującą ϵ. Strategia RDG jest strategią dekompozycji, która automatycznie wyznacza wartość ϵ w zależności od rozpatrywanego problemu optymalizacyjnego. W tym celu, losowanych jest k rozwiązań problemu i na ich podstawie wyliczana jest wartość ϵ

$$\epsilon = \alpha \cdot \min\{|f(\ddot{\mathbf{x}}_\mathbf{1})|, ..., |f(\ddot{\mathbf{x}}_\mathbf{k})|\} \tag{2.9}$$

gdzie α jest współczynnikiem, którego odpowiednia wartość nie jest łatwa do określenia [78, 126] i wymaga przeprowadzenia procedury dostrajania na jak najbardziej reprezentatywnym zbiorze problemów optymalizacyjnych. Kolejną wersją RDG jest strategia RDG2 (ang. *Recursive Differential Grouping 2*) [126], która jest strategią bezparametrową i nie wymaga podawania wartości parametrów α i k przez użytkownika. W RDG2 wykorzystano mechanizm automatycznego wyliczania wartości ϵ zaproponowany w [94]. Przewagą tego mechanizmu jest fakt, że działa on lokalnie. Dla każdego sprawdzenia, czy dwa zbiory zmiennych oddziałują na siebie wzajemnie, może on zwrócić inna wartość ϵ. W przypadku strategii RDG, wartość ϵ jest wartością globalną. Raz wyliczona, tuż po uruchomieniu RDG, obowiązuje dla wszystkich sprawdzeń ewentualnej interakcji.

Strategia RDG2 została następnie rozwinięta poprzez dodanie dwóch parametrów, ϵ_s i ϵ_n, tworząc strategię RDG3 (ang. *Recursive Differential Grouping 3*) [125]. Oba parametry zostały zaproponowane, żeby zwiększyć skuteczność i efektywność architektur koewolucji kooperatywnej, które następnie korzystają z dekompozycji dostarczonej przez strategię RDG3. Dzięki parametrowi ϵ_s, zmienne, dla których nie zostały znalezione żadne interakcje, łączą się w grupy o rozmiarze ϵ_s. W przypadku gdy liczba takich zmiennych nie jest podzielna przez ϵ_s, ostatnia grupa będzie mniej liczna. Powodem takiego grupowania jest obserwacja, że wiele metod optymalizacji potrafi efektywniej i równie skutecznie rozwiązywać problemy w pełni separowalne niż optymalizacja każdej zmiennej oddzielnie, gdy liczba zmiennych optymalizowanego problemu jest odpowiednio mała. CMA-ES (ang. *Covariance Matrix Adaptation Evolution Strategy*) [35] jest przykładem skutecznej i efektywnej metody optymalizacji gdy liczba separowalnych zmiennych decyzyjnych nie przekracza 100 [116].

Drugim wprowadzonym w strategii RDG3 parametrem jest ϵ_n. Parametr ten kontroluje wielkość grup w przypadku dekomponowania problemów składających się z nakładających się na siebie podproblemów. Autorzy RDG3 w swoich badaniach pokazali, że dla wielu problemów tego typu, grupowanie zmiennych w mniejsze grupy daje lepsze wyniki optymalizacji, w przypadku użycia koewolucji kooperatywnej, niż rozpatrywanie tylko jednej grupy. W tym celu, na początku każdej iteracji strategii RDG3, sprawdzana jest liczność zbioru X_1. Jeżeli $|X_1| \geq \epsilon_n$ to ze zmiennych będących elementami X_1 tworzymy grupę zmiennych zależnych i w następnej iteracji

RDG3 będziemy szukać zależności dla kolejnej zmiennej decyzyjnej. Rysunek 2.3 przedstawia, jak trzy różne strategie dekompozycji dzielą ten sam problem optymalizacyjny na różne grupy ze względu na istnienie podproblemów, które nakładają się na siebie. Warto zauważyć, że zarówno strategia DG, jak i RDG3 dzielą nakładające się podproblemy na mniejsze grupy. Co więcej, strategia DG pogrupuje tak samo zmienne należące do nakładających się na siebie podproblemów jak RDG3 z $\epsilon_n = 0$.

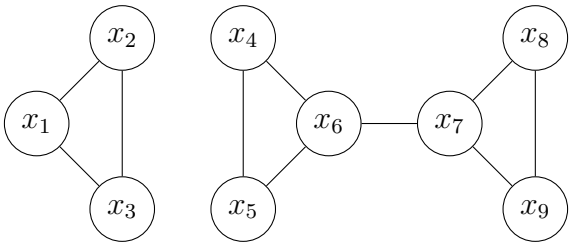
2.3 Sprawdzanie monotoniczności

Strategie bazujące na grupowaniu różnicowym wymagają problemów, które składają się z addytywnie separowalnych podproblemów. W przeciwnym wypadku, mogą wykryć zależności, które w rzeczywistości nie istnieją. Inną rodzinę strategii dekompozycji, dla których to założenie nie musi być spełnione, stanowią strategie bazujące na sprawdzaniu monotoniczności. W tych strategiach, dwie zmienne decyzyjne x_p i x_q zostaną uznane za bezpośrednio oddziałujące na siebie jeżeli $\exists a_1 \neq a_2, b_1 \neq b_2, \ddot{\mathbf{x}} \in \mathcal{D}$ takie, że

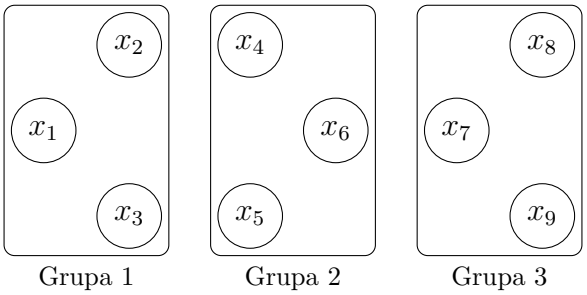
$$f(\ddot{\mathbf{x}})|_{x_p=a_1,x_q=b_1} \leq f(\ddot{\mathbf{x}})|_{x_p=a_2,x_q=b_1} \wedge f(\ddot{\mathbf{x}})|_{x_p=a_1,x_q=b_2} > f(\ddot{\mathbf{x}})|_{x_p=a_2,x_q=b_2} \tag{2.10}$$

Takie wykrywanie bezpośrednich zależności pomiędzy parą zmiennych można znaleźć m.in. w pracach [14, 120, 143]. Wsród nich znajduje się strategia CCVIL (ang. *Cooperative Co-evolution with Variable Interaction Learning*) [14], której Autorzy zainspirowali się pracą [143], ponieważ wykrywanie zależności pomiędzy parą zmiennych osadzili w swojej strategii w podobny sposób. Ich ideą jest optymalizacja każdej zmiennej decyzyjnej oddzielnie używając dowolnej metody optymalizacji, która w trakcie swojego działania utrzymuje populację rozwiązań kandydujących. Zmienne optymalizowane są w losowej kolejności, natomiast jedynie te zmienne, które były optymalizowane jedna po drugiej są sprawdzane pod kątem bezpośredniej interakcji. W CCVIL, używana jest klasyczna architektura koewolucji kooperatywnej, która została opisana w rozdziale 1.1.2. Oznaczmy jako x_p zmienną decyzyjną, której optymalizacja właśnie się zakończyła, natomiast niech x_q będzie poprzednio optymalizowaną zmienną. Zgodnie z oznaczeniami użytymi we wzorze (2.10), niech

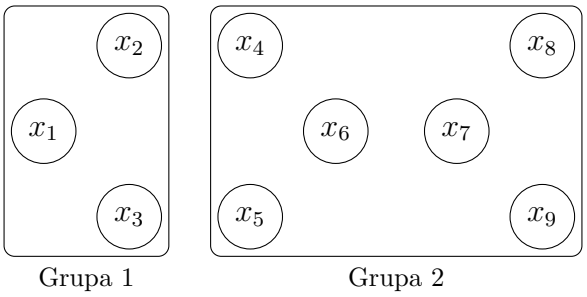
- $\ddot{\mathbf{x}}$: najlepsze znalezione dotychczas rozwiązanie,
- a_1: najlepsza wartość p-tej zmiennej po właśnie zakończonej optymalizacji x_p,
- a_2: najlepsza wartość p-tej zmiennej przed rozpoczęciem właśnie zakończonej optymalizacji x_p,



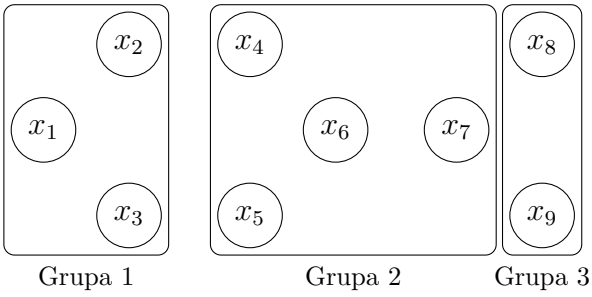
(a) Graf interakcji rozważanego problemu



(b) Grupy utworzone przez strategię DG



(c) Grupy utworzone przez strategię RDG



(d) Grupy utworzone przez strategię RDG3 dla $\epsilon_n = 4$

Rysunek 2.3: Różnice w grupowaniu problemów zawierających nakładające się na siebie podproblemy

– b_1: najlepsza wartość q-tej zmiennej przed rozpoczęciem właśnie zakończonej optymalizacji x_p,

– b_2: wartość q-tej zmiennej wzięta z losowego rozwiązania z populacji przypisanej do q-tej zmiennej.

Jeżeli wartość wyrażenia logicznego ze wzoru (2.10) dla powyższych wartości jest prawdziwa to CCVIL umieści obie zmienne decyzyjne w jednej grupie. Pojedyncza iteracja CCVIL składa się z optymalizacji wszystkich zmiennych w losowej kolejności. CCVIL kończy swoje działanie gdy jeden z poniższych warunków zostanie spełniony.

– Wszystkie zmienne zostały umieszczone w jednej grupie. Dotyczy to problemów w pełni nieseparowalnych.

– Nie wykryto żadnych bezpośrednich zależności pomiędzy zmiennymi i liczba wykonanych przez CCVIL iteracji jest większa niż wartość parametru podanego przez użytkownika $\check{K}$.

– Liczba wykonanych przez CCVIL iteracji jest większa niż wartość parametru podanego przez użytkownika $\hat{K}$.

Innym przykładem strategii dekompozycji, która także analizuje monotoniczność jest SVIL (ang. *Statistical Variable Interdependence Learning*) [120]. W tej strategii, dla każdej pary zmiennych decyzyjnych, generowane jest wiele krotek $(\ddot{\mathbf{x}}, a_1, a_2, b_1, b_2)$. Dla każdej pary zmiennych następnie liczymy ile krotek spełniło wzór (2.10). Jeżeli ta liczba jest większa od zadanego progu to uznajemy, że rozważana para zmiennych jest bezpośrednio od siebie zależna.

Złożoność obliczeniowa dwóch omówionych powyżej strategii to $\mathcal{O}(n^2)$. Podobnie, jak w przypadku strategii bazujących na grupowaniu różnicowym, zaproponowana została strategia FVIL (ang. *Fast Variable Interdependence Learning*) [29], która działa w sposób rekursywny zmniejszając złożoność obliczeniową do $\mathcal{O}(n\log(n))$. W tym celu dwie rozłączne grupy zmiennych X_1 i X_2 są sprawdzane, czy zachodzi pomiędzy nimi interakcja. Stwierdzamy interakcję wtedy i tylko wtedy, gdy $\exists\delta_1 > 0$, $\delta_2 > 0$, $\mathbf{u_1} \in U_{X_1}$, $\mathbf{u_2} \in U_{X_2}$, $\ddot{\mathbf{x}} \in \mathcal{D}$ takie, że

$$f(\ddot{\mathbf{x}}) \leq f(\ddot{\mathbf{x}} + \delta_1\mathbf{u_1}) \wedge f(\ddot{\mathbf{x}} + \delta_2\mathbf{u_2}) > f(\ddot{\mathbf{x}} + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2}) \tag{2.11}$$

gdzie $\mathbf{u_j} \in U_{X_j}$ jest wektorem jednostkowym zgodnym ze wzorem (2.7). Pojedyncza iteracja FVIL jest bardzo podobna do do pojedynczej iteracji RDG, gdy zbiór X_1 jest jednoelementowy (Rysunek 2.2c). W FVIL, drugi zbiór zmiennych jest dzielony

na pół rekursywnie, aż do momentu gdy nie wykryjemy jego interakcji z pojedynczą zmienną będącą elementem X_1 lub gdy jest on już zbiorem jednoelementowym. Po zakończeniu każdej iteracji FVIL, wszystkie zmienne, dla których została wykryta interakcja z X_1 tworzą, wraz z pojedynczą zmienną z X_1, grupę zmiennych zależnych. Są one także usuwane ze zbioru X_2. Następnie, w przeciwieństwie do RDG, w następnej iteracji FVIL, rozważana będzie kolejna pojedyncza zmienna z X_2 zamiast utworzonej przed chwilą grupy. Różnica ta będzie widoczna jedynie w tych grupach, które zostały utworzone dla problemów posiadających nakładające się na siebie podproblemy. Strategia FVIL nie połączy wszystkich zmiennych należących do takich podproblemów w jedną grupę, lecz utworzy grupy zbliżone do grup uzyskanych dzięki strategii DG (Rysunek 2.3b).

2.4 Techniki typu „linkage learning”

W literaturze, strategie dekompozycji kojarzone są głównie z problemami o ciągłej przestrzeni przeszukiwań, natomiast pierwsze prace dotyczące szukania zależności pomiędzy zmiennymi (genami) dotyczyły algorytmów genetycznych i binarnych problemów optymalizacyjnych. Proces wyszukiwania zależności pomiędzy genami w binarnej optymalizacji nazywany jest w literaturze jako „linkage learning”. Pomimo różnych nazw, strategie dekompozycji i techniki typu „linkage learning” są w zasadzie tym samym [89]. Dodatkowo, wzory (2.2) i (2.10), które są podstawą każdej strategii dekompozycji bazującej odpowiednio na grupowaniu różnicowym i sprawdzanie monotoniczności, zostały wcześniej zaproponowane w pracach dotyczących problemów binarnych. Grupowanie różnicowe ma podobne podstawy jak LINC (ang. *Linkage Identification by Nonlinearity Check*) [85], natomiast sprawdzanie monotoniczności opiera się na podobnych zasadach jak LIMD (ang. *Linkage Identification by Non-Monotonicity Detection*) [86].

Do tej pory zaproponowanych zostało wiele różnych klasyfikacji technik typu „linkage learning” [89, 98, 105, 152]. W kontekście niniejszej pracy, najistotniejsza jest klasyfikacja na predykcyjne i empiryczne techniki typu „linkage learning” [105]. Podział ten zostanie wykorzystany w rozdziale 3.1 do klasyfikacji strategii dekompozycji zaprojektowanych dla ciągłych problemów optymalizacyjnych.

2.4.1 Predykcyjne techniki typu „linkage learning”

Większość dotychczas zaproponowanych technik typu „linkage learning” to techniki predykcyjne [105]. Przewidują one, które pary genów są od siebie zależne nie mając jednak pewności, że wykryte zależności w rzeczywistości istnieją. Wiele róż-

nych mechanizmów do predykcji zależności zostało do tej pory zaproponowanych m.in. budowanie modeli probabilistycznych [96, 97], porównywanie genotypów różnych osobników [54,64] lub analiza statystyczna genotypów aktualnej populacji osobników [43, 130].

W ostatnich latach za najlepsze predykcyjne techniki uważa się te techniki, które wyliczają pewne statystyki na podstawie osobników, które aktualnie znajdują się w populacji [9, 30, 43, 130, 145]. W przypadku binarnych przestrzeni przeszukiwań, dla każdej pary genów można policzyć wystąpienia konkretnych par wartości genów, a następnie policzyć wartość miary bazującej na entropii. Przykładem takiej miary jest informacja wzajemna (ang. *mutual information*) [30, 43, 61, 130], określona wzorem

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{2.12}$$

gdzie X i Y są zmiennymi losowymi. Na podstawie wartości wybranej miary, predykcyjne techniki wysnuwają przypuszczenie, która para genów jest bardziej zależna od innych. Wartości miary wyliczonej dla każdej pary genów przechowywane są w macierzy zależności (ang. *dependency structure matrix*). W przeciwieństwie do macierzy interakcji, wartości w macierzy zależności, przynajmniej dla technik predykcyjnych, są wartościami ze zbioru liczb rzeczywistych. Z tego względu, reprezentowany przez macierz zależności graf, jest traktowany zazwyczaj jako graf pełny z wagami, gdzie wagami są wartości z macierzy zależności. W celu otrzymania grup składających się z genów, które prawdopodobnie są zależne, nie szuka się zatem spójnych składowych grafu zależności. W literaturze popularne są natomiast dwa inne mechanizmy.

– Macierz zależności, lub jej drobną modyfikację, można potraktować jako macierz odległości by następnie przeprowadzić hierarchiczną klasteryzację. Wynikiem jest wtedy tzw. drzewo powiązań między genami (ang. *linkage tree*), którego węzłami są grupy genów uznawane za zależne. Takie podejście stosowane jest m.in. w metodach LT-GOMEA (ang. *Linkage Tree Gene-pool Optimal Mixing Evolutionary Algorithm*) [26, 130] i P3 (ang. *Parameter-less Population Pyramid*) [30, 31, 55, 156].

– Macierz zależności można przedstawić za pomocą nieskierowanego grafu z wagami, w którym węzłami są geny, natomiast wartości wag przypisane do każdej krawędzi brane są z macierzy zależności. Odpowiednio przeszukując taki graf, możemy znaleźć grupy genów, które powinny być od siebie zależne. W tym celu, dla dowolnego wierzchołka początkowego szukamy innego wierzchołka, który połączony jest z wierzchołkiem początkowym krawędzią o największej

wadze. Następnie szukamy kolejnego wierzchołka, dla którego suma wag krawędzi łączących go z poprzednimi dwoma wierzchołkami jest największa spośród pozostałych wierzchołków. Procedurę tę powtarzamy, tj. szukanie wierzchołka o maksymalnej sumie wag krawędzi łączących go z poprzednio odwiedzonymi wierzchołkami, aż do momentu gdy odwiedzone zostaną już wszystkie wierzchołki grafu. Dobierając kolejne wierzchołki, inkrementacyjnie tworzymy zbiór składający się z grup genów, które prawdopodobnie są ze sobą powiązane (ang. *incremental linkage set*). Metoda DSMGA-II (ang. *Dependency Structure Matrix Genetic Algorithm II*) [43] oraz jej późniejsze wersje [12, 56] wykorzystują powyżej opisany sposób, lub jego drobne modyfikacje, do szukania grup genów prawdopodobnie od siebie zależnych.

Grupy genów, które zostały uznane za zależne wykorzystywane są następnie podczas optymalnego mieszania (ang. *optimal mixing*) [43, 56, 128, 130]. Operator optymalnego mieszania zmienia wartości genów danego osobnika, a następnie sprawdza czy zmiana wartości genów nie pogarsza wartości funkcji przystosowania dla rozważanego osobnika. Jeżeli wartość funkcji przystosowania uległa pogorszeniu to wszystkie zmiany wartości genów są cofane. Tylko geny należące do tej samej grupy mogą zostać zmienione podczas pojedynczego wykonania operacji optymalnego mieszania. Jeżeli grupa składa się z genów, które faktycznie od siebie zależą i są niezależne od pozostałej części genotypu to dzięki cofaniu zmian odrzucamy wartości genów, które na pewno nie są częścią rozwiązania optymalnego [11, 132]. Należy zauważyć, że jest to cecha pożądana dla operatorów selekcji. Możemy zatem stwierdzić, że operator selekcji jest integraclną częścią operatora optymalnego mieszania.

Koncepcja szukania grup genów, które prawdopodobnie są od siebie zależne, i ich późniejsze wykorzystanie podczas optymalnego mieszania była podstawową do zaproponowania nowych metod optymalizacji również dla permutacyjnych [9, 108, 145] i ciągłych przestrzeni przeszukiwań [10, 11, 88]. Przykładem takiej metody optymalizacji, która rozwiązuje ciągłe problemy optymalizacyjne, jest RV-GOMEA (ang. *Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm*) [11]. W metodzie RV-GOMEA, miarą zależności dwóch genów (zmiennych) x_i i x_j jest wartość estymacji informacji wzajemnej dla ciągłych zmiennych [59], która wyliczana jest zgodnie z następującym wzorem

$$\mathrm{MI}_{i,j} = \log_2 \sqrt{\frac{1}{1 - r_{i,j}^2}} \tag{2.13}$$

gdzie $r_{i,j} \in [-1, 1]$ jest współczynnikiem korealcji liniowej Pearsona, którego wartość w przypadku metody RV-GOMEA estymowana jest na podstawie osobników, które zostały wybrane z aktualnej populacji. Do estymacji wartości $r_{i,j}$ potrzebu-

jemy wyznaczyć estymację macierzy kowariancji ($\hat{\Sigma}$) przy użyciu metody największej wiarygodności. Po wyznaczeniu $\hat{\Sigma}$, wartość $r_{i,j}$ wyliczamy w następujący sposób

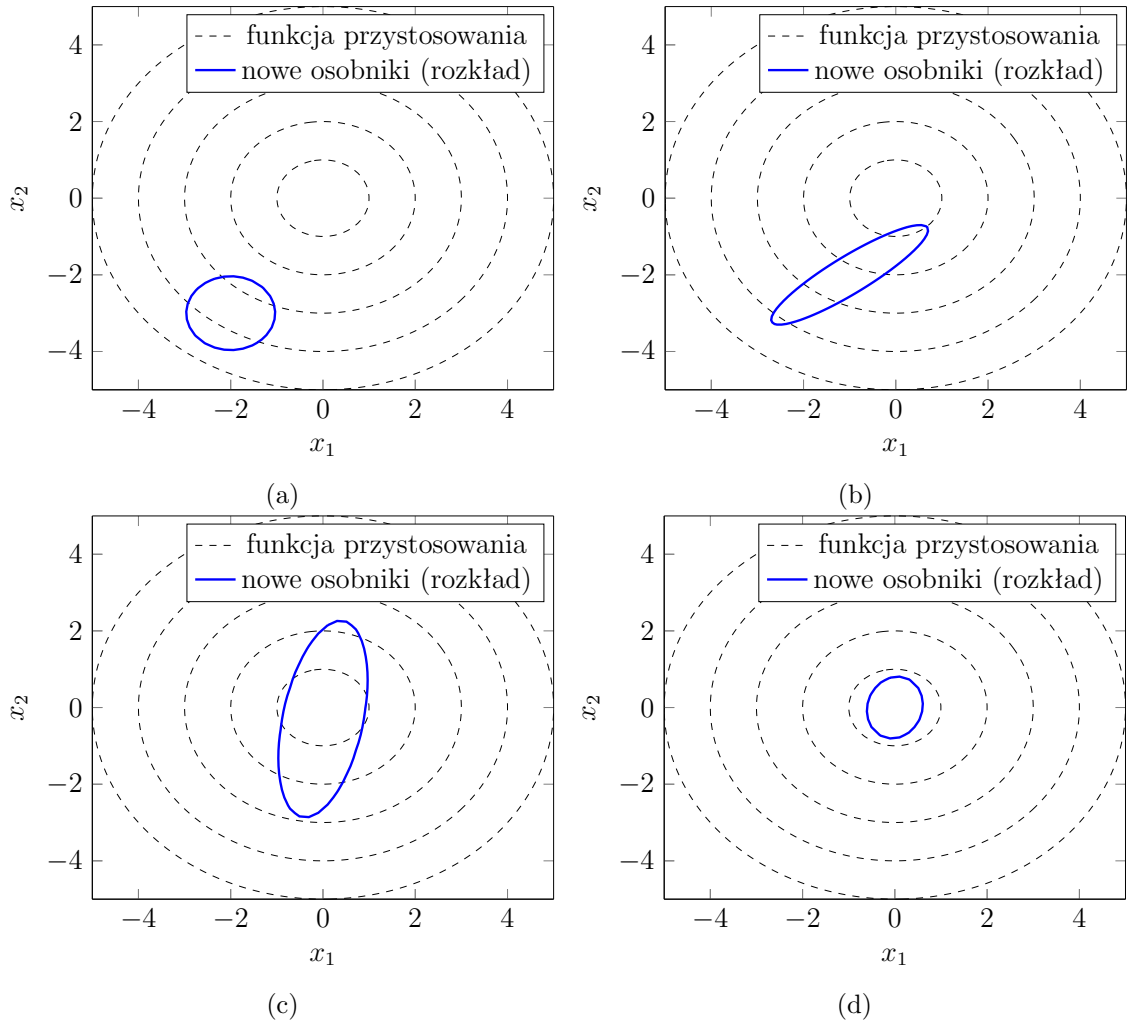
$$r_{i,j} = \frac{\hat{\Sigma}_{i,j}}{\hat{\sigma}_i \hat{\sigma}_j} \tag{2.14}$$

gdzie $\hat{\Sigma}$ jest macierzą o wymiarach $n \times n$ i przyjmuje następującą postać

$$\begin{bmatrix} \hat{\sigma}_1^2 & \hat{\Sigma}_{1,2} & \cdots & \hat{\Sigma}_{1,n-1} & \hat{\Sigma}_{1,n} \\ \hat{\Sigma}_{2,1} & \hat{\sigma}_2^2 & \cdots & \hat{\Sigma}_{2,n-1} & \hat{\Sigma}_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{\Sigma}_{n-1,1} & \hat{\Sigma}_{n-1,2} & \cdots & \hat{\sigma}_{n-1}^2 & \hat{\Sigma}_{n-1,n} \\ \hat{\Sigma}_{n,1} & \hat{\Sigma}_{n,2} & \cdots & \hat{\Sigma}_{n,n-1} & \hat{\sigma}_n^2 \end{bmatrix} \tag{2.15}$$

Metoda RV-GOMEA należy do metod optymalizacji, które estymują rozkład prawdopodobieństwa by na jego podstawie generować nowe, lepsze osobniki. W tym celu, w metodzie RV-GOEMA, wykorzystywany jest wielowymiarowy rozkład normalny. Ideą estymacji parametrów wielowymiarowego rozkładu normalnego jest uzyskanie kształtu, który będzie ukierunkowywał przeszukiwanie w stronę lokalnego optimum. Należy zauważyć, że w podobny sposób działają gradientowe metody optymalizacji [22, 83]. Rozpatrzmy zastosowanie metody RV-GOMEA do optymalizacji dwuwymiarowej funkcji sferycznej, określonej wzorem $x_1^2 + x_2^2$ i mającej optimum globalne w punkcie $[0, 0]$. Rysunek 2.4 przedstawia rozkłady prawdopodobieństwa, z których generowane są nowe osobniki w różnych fazach procesu optymalizacji. Podczas pierwszej iteracji (Rysunek 2.4a), metoda RV-GOMEA nie posiada żadnych informacji na temat optymalizowanego problemu, dlatego estymowana macierz kowariancji jest macierzą jednostkową. W późniejszych iteracjach (Rysunek 2.4b i Rysunek 2.4c), estymacja macierzy kowariancji odbywa się na podstawie wyselekcjonowanych osobników z populacji. Wynikiem takiej estymacji jest kształt rozkładu, który pozwala na generację nowych osobników w kierunku optimum globalnego. Podczas ostatnich iteracji (Rysunek 2.4d), metoda RV-GOMEA generuje osobniki wokół optimum globalnego. Estymowana macierz kowariancji zbliżona jest zatem do macierzy diagonalnej.

Wartość miary $\mathrm{MI}_{i,j}$, dla każdej pary zmiennych x_i i x_j, liczona jest na podstawie wyselekcjonowanych osobników z aktualnej populacji. Jej wartość uwarunkowana jest zatem kształtem rozkładu, z którego generowane są nowe osobniki. Funkcja sferyczna jest funkcją w pełni separowalną. Pomimo, że obie zmienne nie oddziałują na siebie wzajemnie, wartość miary $\mathrm{MI}_{1,2}$ wyliczonej dla osobników wygenerowanych z



Rysunek 2.4: Rozkłady prawdopodobieństwa, z których generowane są nowe osobniki w różnych fazach procesu optymalizacji dwuwymiarowej funkcji sferycznej przez metodę RV-GOMEA

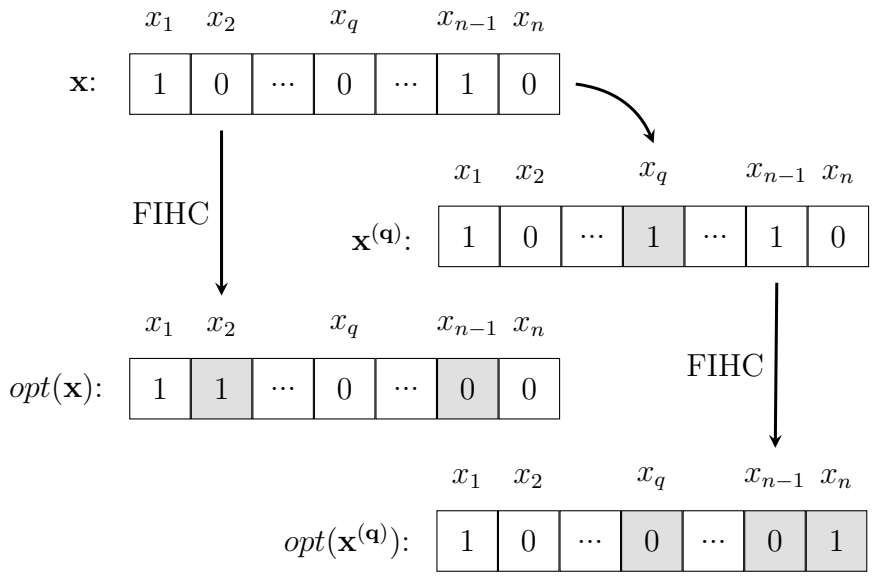
rozkładów przedstawionych na rysunkach Rysunek 2.4b i Rysunek 2.4c sugerować będzie ich zależność, ponieważ podniesiona do kwadratu wartość współczynnika liniowej korelacji Pearsona $r^2_{1,2}$ jest istotnie większa od 0. Brak pewności czy dwie zmienne rzeczywiście od siebie zależą spowodował, że najnowsze wersje metody RV-GOMEA, które osiągają lepsze wyniki niż ich poprzednicy, wykrywają zależności pomiędzy zmiennymi na podstawie wzoru (2.2) znanego ze strategii DG [88, 89].

2.4.2 Empiryczne techniki typu „linkage learning”

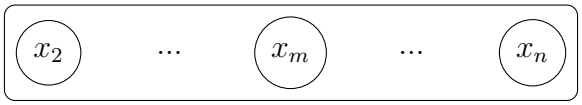
W empirycznych technikach typu „linkage learning”, predykcja, że dwa geny są od siebie zależne zostaje zastąpiona pewnością [105, 106, 108, 109]. Empiryczne techniki nigdy nie wykryją zależności, które w rzeczywistości nie istnieją. Z drugiej strony, mogą one pominąć niektóre z istniejących zależności.

Pierwszą empiryczną techniką zaproponowaną dla binarnych przestrzeniach przeszukiwań jest 3LO (ang. *Linkage Learning based on Local Optimization*) [105]. Rysunek 2.5 przedstawia ogólną koncepcję wykrywania zależnych genów przez tę technikę. W 3LO, zależności wykrywane są na podstawie kodowanego binarnie genotypu $\mathbf{x} = [x_1, ..., x_n]$. Genotyp ten zaburzamy, wybierając inną wartość dla q-tego genu tworząc tym samym genotyp $\mathbf{x^{(q)}} = [x_1, ..., x'_q, ..., x_n]$. W przypadku binarnych problemów optymalizacyjnych, inną wartością q-tego genu, oznaczoną jako x'_q jest po prostu zanegowana wartość genu x_q. Oba genotypy są następnie optymalizowane za pomocą lokalnej metody optymalizacji FIHC (ang. *First Improvement Hill Climber*) [30]. Zasada działania metody FIHC jest następująca. Dla zadanego genotypu i wylosowanej kolejności przetwarzania genów, negowane są po kolei wszystkie geny. Jeżeli zmiana wartości genu spowoduje polepszenie wartości funkcji przystosowania to zmiana jest zachowywana, w przeciwnym wypadku jest ona cofana. FIHC kończy swoje działanie gdy po ostatniej udanej zmianie wartości, zmiany dla wszystkich pozostałych genów zostaną kolejno cofnięte. Innymi słowy, jeżeli n kolejnych negacji nie przyniesie poprawy wartości funkcji przystosowania to metoda FIHC nie wykona już więcej iteracji. Geny, których wartości różnią się między genotypami $\mathbf{x}$ i $\mathbf{x^{(q)}}$ oraz gen, którego wartość została na samym początku zaburzona są od siebie zależne. Zostało to udowodnione w [105].

3LO wykrywa zarówno bezpośrednie, jak i pośrednie zależności pomiędzy genami. Wiąże się to z wysoką złożonością techniki 3LO, która uniemożliwia jej wykorzystanie w ramach takich metod obliczeniowych jak LT-GOMEA, P3, czy DSMGA-II. Z tego powodu, zaproponowano nową metodę ewolucyjną dedykowaną do osadzenia w niej 3LO. Zmodyfikowaną wersją techniki 3LO jest DLED (ang. *Direct Linkage Empirical Discovery*) [109]. DLED zachowuje podstawową cechę techniki 3LO



(a) Utworzenie genotypów $opt(\mathbf{x})$ i $opt(\mathbf{x}^{(\mathbf{q})})$ na podstawie genotypu $\mathbf{x}$ i indeksu q



(b) Grupa zależnych genów otrzymana po porównaniu genotypów $opt(\mathbf{x})$ i $opt(\mathbf{x}^{(\mathbf{q})})$

Rysunek 2.5: Ogólny schemat koncepcji wykrywania zależnych genów przez 3LO (na podstawie [105])

i znajduje jedynie istniejące w rzeczywistości zależności. DLED wymaga mniejszej liczby wyliczeń funkcji przystosowania niż 3LO i wykrywa tylko bezpośrednie zależności pomiędzy genami. Dzięki temu, technikę DLED można osadzić w wiodących aktualnie metodach ewolucyjnych, które korzystają z predykcyjnych technik typu „linkage learning". Ponadto wykrywanie jedynie bezpośrednich zależności może być korzystne podczas rozwiązywania problemów z nakładającymi się na siebie podproblemami [110].

Koncepcja techniki 3LO, która została pierwotnie zaproponowana dla przestrzeni binarnych, została ostatnio rozszerzona do przestrzeni dyskretnych wyłączając przestrzenie permutacyjne [106]. Dla przestrzeni permutacyjnych, zaproponowana została inna empiryczna technika typu „linkage learning", która różni się od 3LO, lecz także dostarczony został dowód, że wykrywa ona jedynie prawdziwe zależności [108].

Rozdział 3

Inkrementacyjne i rekursywne grupowanie rankingowe — propozycja nowej strategii dekompozycji

W niniejszym rozdziale przedstawiona zostanie nowa strategia dekompozycji, zaproponowana przez autora niniejszej pracy. Opis nowej propozycji zostanie poprzedzony analizą mocnych i słabych stron dwóch najpopularniejszych w ostatnich latach podejść, na których bazują aktualnie wiodące strategie dekompozycji ciągłych problemów optymalizacyjnych. Na podstawie tej analizy, wyciągnięte zostały wnioski, które były zarazem motywacją dla prac nad nową strategią dekompozycji dla problemów o ciągłej przestrzeni przeszukiwań.

3.1 Motywacje za zaproponowaniem nowej strategii dekompozycji

Dwa najpopularniejsze, w ostatnich latach, podejścia wykorzystywane do dekompozycji ciągłych problemów optymalizacyjnych wywodzą się z grupowania różnicowego i sprawdzania monotoniczności. Nie są one jednak wolne od wad. Wady dostrzeżone przez autora niniejszej pracy zostaną opisane w tym podrozdziale na podstawie przykładów pomijania istniejących zależności lub raportowania nieistniejących. Z tego powodu, dyskusja dotycząca niezależności zmiennych uwzględniając definicję podaną w [105], zostanie przeprowadzona na samym początku tego podrozdziału. Wszystkie spostrzeżenia i wnioski autora niniejszej pracy zostaną wysunięte w oparciu o analizę funkcji dwóch zmiennych $\tilde{f}$ i dwóch funkcji jednej zmiennej $\tilde{g}$ i $\tilde{h}$. Funkcje $\tilde{g}$ i $\tilde{h}$ upraszczają funkcję $\tilde{f}$ poprzez zastąpienie drugiej zmiennej de-

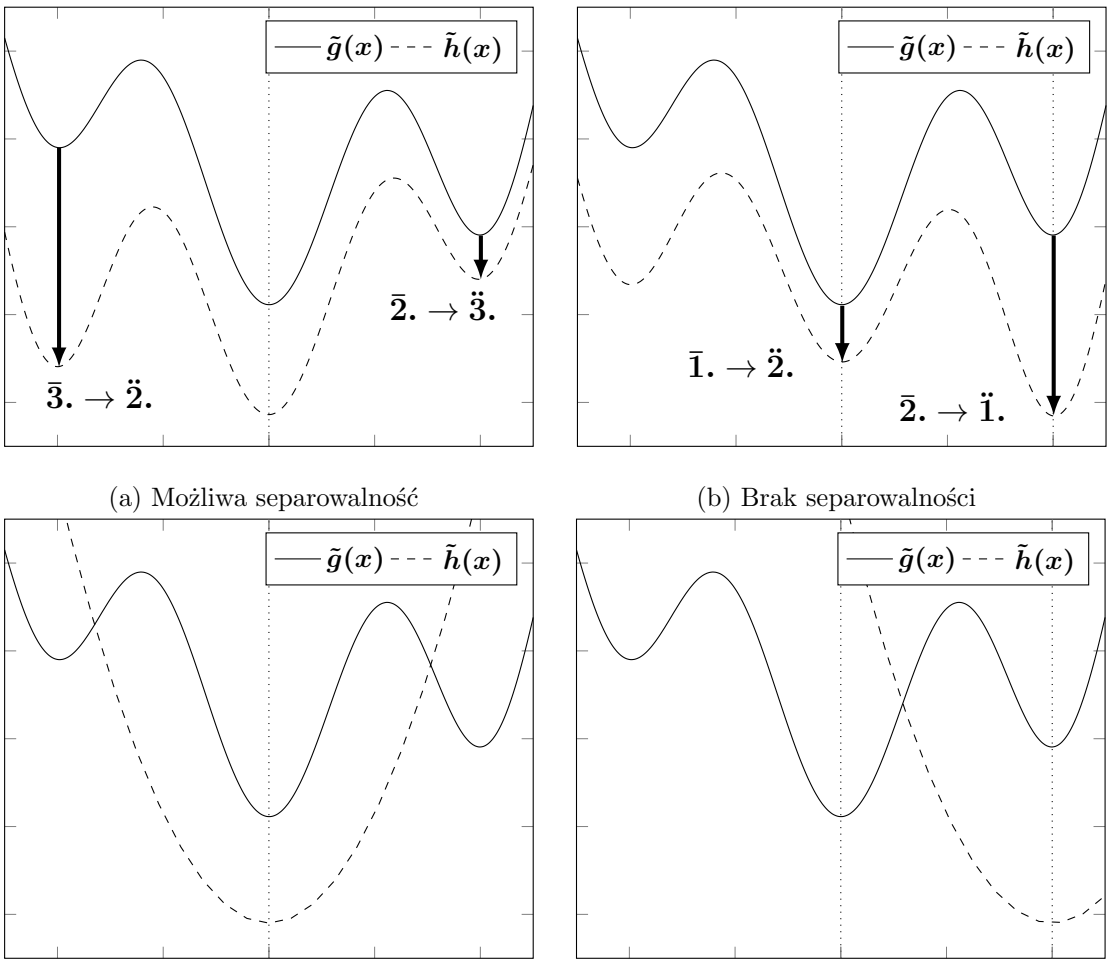
cyzyjnej stałą wartością. Możemy je zatem zdefiniować jako $\tilde{g}(x) = \tilde{f}(\mathbf{x})|_{x_1=x,x_2=b_1}$ i $\tilde{h}(x) = \tilde{f}(\mathbf{x})|_{x_1=x,x_2=b_2}$, gdzie b_1 i b_2 to stałe o dwóch różnych wartościach. Należy zauważyć, że jedynym argumentem funkcji $\tilde{g}$ i $\tilde{h}$ jest pierwszy argument funkcji $\tilde{f}$, czyli x_1. Wszystkie wnioski wyciągnięte dla funkcji dwóch zmiennych mogą być następnie łatwo uogólnione do funkcji o większej liczbie argumentów.

3.1.1 Sprawdzanie monotoniczności jako empiryczna technika typu „linkage learning"

Zgodnie ze wzorem (1.1) i uwzględniając funkcje $\tilde{g}$ oraz $\tilde{h}$, zmienne decyzyjne x_1 i x_2 będące argumentami funkcji $\tilde{f}$ oddziałują na siebie wzajemnie jeżeli $\arg\min_x \tilde{g}(x) \neq \arg\min_x \tilde{h}(x)$. W przeciwnym wypadku, nie możemy jednoznacznie stwierdzić, czy zachodzi interakcja pomiędzy x_1 i x_2, czy też nie. Jest to spowodowane faktem, że porównując globalne optima funkcji $\tilde{g}$ i $\tilde{h}$ uwzględniliśmy jedynie dwie różne wartości x_2 tj. b_1 i b_2. Jeżeli w definicjach funkcji $\tilde{g}$ i $\tilde{h}$ jako wartość x_2 przyjęlibyśmy b_3 i b_4 takie, że $b_3 \neq b_4 \wedge b_3 \neq b_1 \wedge b_4 \neq b_2$ to warunek $\arg\min_x \tilde{g}(x) \neq \arg\min_x \tilde{h}(x)$ mógłby być wtedy spełniony.

Podejmowanie decyzji o ewentualnej interakcji tylko na podstawie globalnych optimów może pomijać zmiany w wykresach funkcji (ang. *fitness landscape characteristics*) [79]. Zakładając, że dwie funkcje przystosowania mają to samo globalne optimum, lecz ich wykresy różnią się między sobą to ta sama metoda optymalizacji może się zupełnie inaczej zachować w obu przypadkach [27, 53]. Tego typu różnice mogą wpływać zarówno na skuteczność, jak i efektywność metody. Na rysunkach Rysunek 3.1a i Rysunek 3.1b przedstawione zostały dwie sytuacje, w których zmiana wartości x_2 z b_1 na b_2 powoduje, że wartość funkcji w jednym z optimów staje się lepsza niż wartość funkcji w jednym z pozostałych optimów. Jeżeli ta zmiana dotyczy globalnego optimum, oznaczonego przerywaną pionową kreską, to wykryta zostanie zależność pomiędzy zmiennymi. W przeciwnym wypadku, nie stwierdzamy interakcji dla tych konkretnych wartości b_1 i b_2. Podobnie wygląda sytuacja gdy zmiana wartości x_2 z b_1 na b_2 przekształca funkcję wielomodalną w unimodalną (Rysunek 3.1c i Rysunek 3.1d). Jeżeli globalne optimum się nie zmieniło to zależność pomiędzy x_1 i x_2 nie zostanie wykryta dla tych konkretnych wartości b_1 i b_2.

Przedstawione powyżej przykłady pokazują, że mimo możliwej separowalności zmiennych x_1 i x_2 biorąc pod uwagę wzór (1.1), metody optymalizacji mogą się różnie zachować w zależności od tego, która z funkcji, $\tilde{g}$ lub $\tilde{h}$, będzie optymalizowana. Wydaje się zatem, że warunek bazujący na obserwacji globalnych optimów jest niewystarczający do rozstrzygnięcia, czy zmienne oddziałują na siebie wzajemnie. W kolejnym przykładzie, zaprezentowanym na rysunku Rysunek 3.2, rozważamy za-



(a) Możliwa separowalność

(b) Brak separowalności

(c) Możliwa separowalność

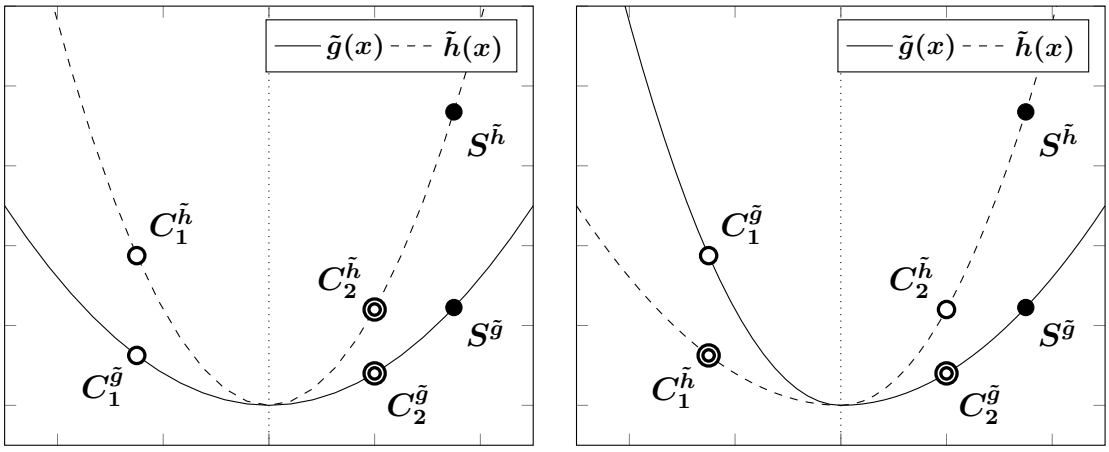
(d) Brak separowalności

Rysunek 3.1: Separowalność na podstawie wzoru (1.1)

chłanną metodę optymalizacji, która rozpoczyna poszukiwanie najlepszego rozwiązania w punkcie $S^{\tilde{g}}$ dla funkcji $\tilde{g}$ oraz w punkcie $S^{\tilde{h}}$ dla funkcji $\tilde{h}$. Należy zauważyć, że obie funkcje, $\tilde{g}$ i $\tilde{h}$, mają to samo optimum globalne, zatem zmienne decyzyjne x_1 i x_2 mogą być niezależne od siebie zgodnie ze wzorem (1.1). Niech zachłanna metoda optymalizacji ma do wyboru po dwa rozwiązania kandydujące dla każdej funkcji. Oznaczmy je jako $C_1^{\tilde{g}}$ i $C_2^{\tilde{g}}$ dla funkcji $\tilde{g}$ oraz $C_1^{\tilde{h}}$ i $C_2^{\tilde{h}}$ dla funkcji $\tilde{h}$. Podwójnymi okręgami zaznaczone jest lepsze rozwiązanie kandydujące dla każdej z rozważanych funkcji. W przypadku sytuacji przedstawionej na rysunku Rysunek 3.2a, lepszym rozwiązaniem kandydującym dla obu rozważanych funkcji jest drugie rozwiązanie ($C_2^{\tilde{g}}$ i $C_2^{\tilde{h}}$), zatem zmienne x_1 i x_2 mogą być separowalne. Przeciwne wnioski można wyciągnąć analizując Rysunek 3.2b. Lepszym rozwiązaniem kandydującym rozważając funkcję $\tilde{g}$ jest drugie rozwiązanie, natomiast dla funkcji $\tilde{h}$ lepszym wyborem jest $C_2^{\tilde{h}}$. Proces optymalizacji może zakończyć się zatem w różnych punktach zbioru dopuszczalnego w zależności od funkcji, którą weźmiemy pod uwagę. Z tego powodu, wydaje się rozsądne by uważać zmienne x_1 i x_2 jednak jako oddziałujące na siebie wzajemnie. Przedstawione w powyższym przykładzie funkcje są unimodalne. W przypadku bardziej złożonych funkcji, zachłanna metoda optymalizacji mogłaby zostać zwiedziona w stronę optimum lokalnego np. dla funkcji $\tilde{g}$ i zarazem znaleźć globalne optimum optymalizując funkcję $\tilde{h}$, mimo że obie funkcje mają to samo najlepsze rozwiązanie. W metodach populacyjnych, które utrzymują i przetwarzają więcej niż jedno rozwiązanie, operatorem odpowiedzialnym za ukierunkowanie przeszukiwania jest zazwyczaj selekcja. Operatory selekcji wybierają częściej te rozwiązanie, dla których wartość funkcji przystosowania jest lepsza i nawet jedna zmiana w wyniku porównania dwóch rozwiązań może wpłynąć na kierunek dalszych przeszukiwań [13].

Powyższe spostrzeżenia są spójne z ogólną koncepcją techniki 3LO [105], opisanej w rozdziale 2.4.2. W celu pokazania podobieństwa, rozważmy dyskretną funkcję $\ddot{f}_d : \{A, B, C, D\} \times \{E, F\} \rightarrow \{0, 1, 2, 3\}$, której wartości znajdują się w tabeli Tabela 3.1. Niech rozwiązaniem, dla którego uruchomiona zostanie technika 3LO będzie $\mathbf{x} = [D, E]$. Integralną częścią tej techniki jest metoda lokalnego przeszukiwania FIHC [30]. Niech FIHC optymalizuje zmienne w kolejności (x_1, x_2), natomiast kolejność rozważanych wartości dla zmiennej x_1 niech będzie zgodna z kolejnością wierszy w tabeli Tabela 3.1. FIHC przechodzi do optymalizacji kolejnej zmiennej gdy jeden z poniższych warunków jest spełniony

– FIHC napotka pierwszą poprawę wartości funkcji przystosowania po zmianie wartości rozważanej aktualnie zmiennej bez względu na fakt, że kolejne wartości mogą dać większą poprawę,



(a) Możliwa separowalność (b) Brak separowalności

Rysunek 3.2: Zachłanna optymalizacja, a separowalność

Tabela 3.1: Wartości funkcji $\ddot{f}_d$

$\mathbf{x}$	$\ddot{f}_d(\mathbf{x})$	$\mathbf{x}$	$\ddot{f}_d(\mathbf{x})$
$[D, E]$	3	$[D, F]$	2
$[A, E]$	2	$[A, F]$	3
$[B, E]$	0	$[B, F]$	0
$[C, E]$	1	$[C, F]$	1

– po podmianie wszystkich możliwych wartości rozważanej aktualnie zmiennej, żadna z nich nie przyniosła poprawy wartości funkcji przystosowania.

W celu sprawdzenia, czy zmienne x_1 i x_2 są zależne, zaburzmy wartość drugiej zmiennej w rozwiązaniu $\mathbf{x}$ otrzymując $\mathbf{x^{(2)}} = [D, F]$. Optymalizacja obu rozwiązań przy użyciu metody FIHC prowadzi do znalezienia rozwiązań $opt(\mathbf{x}) = [A, E]$ i $opt(\mathbf{x^{(2)}}) = [B, F]$. Wartości pierwszej zmiennej różnią się między tymi rozwiązaniami, zatem 3LO uzna, że zmienne x_1 i x_2 oddziałują na siebie wzajemnie. Technika 3LO pomija fakt, że $\forall_{b\in\{E,F\}} \arg\min_{x_1} \ddot{f}_d([x_1, b]) = B$, a jej decyzja o zależności zmiennych x_1 i x_2 wynika jedynie z różnych wyników metody FIHC. Różne wyniki są natomiast skutkiem co najmniej jednego różnego wyniku porównania dwóch wartości funkcji przystosowania, mianowicie $\ddot{f}_d([A, E]) < \ddot{f}_d([D, E]) \wedge \ddot{f}_d([A, F]) > \ddot{f}_d([D, F])$. Należy zauważyć, że wykrywanie tego typu różnic w wartościach funkcji przystosowania jest podstawą strategii dekompozycji, które bazują na sprawdzaniu monotoniczności (wzory (2.10) i (2.11)).

Dowód, że technika 3LO nigdy nie wykryje zależności, które w rzeczywistości nie istnieją opiera się na następującej definicji. Dwa rozłączne zbiory zmiennych X_1 i X_2 są niezależne wtedy i tylko wtedy, gdy $\forall\delta_1 > 0$, $\delta_2 > 0$, $\mathbf{u_1} \in U_{X_1}$, $\mathbf{u_2} \in U_{X_2}$,

$\ddot{\mathbf{x}} \in \mathcal{D}$ spełnione są poniższe dwa warunki

$$f(\ddot{\mathbf{x}}) < f(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1}) \iff f(\ddot{\mathbf{x}} + \delta_2 \mathbf{u_2}) < f(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1} + \delta_2 \mathbf{u_2}) \tag{3.1}$$

i

$$f(\ddot{\mathbf{x}}) = f(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1}) \iff f(\ddot{\mathbf{x}} + \delta_2 \mathbf{u_2}) = f(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1} + \delta_2 \mathbf{u_2}) \tag{3.2}$$

przy założeniu, że $(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1})$, $(\ddot{\mathbf{x}} + \delta_2 \mathbf{u_2})$, $(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1} + \delta_2 \mathbf{u_2}) \in \mathcal{D}$. Jeżeli zatem uda nam się znaleźć takie wartości $\delta_1 > 0$, $\delta_2 > 0$, wektory jednostkowe $\mathbf{u_1} \in U_{X_1}$, $\mathbf{u_2} \in U_{X_2}$ i wektor zmiennych decyzyjnych $\ddot{\mathbf{x}} \in \mathcal{D}$, że $f(\ddot{\mathbf{x}}) \leq f(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1})$ i $f(\ddot{\mathbf{x}} + \delta_2 \mathbf{u_2}) > f(\ddot{\mathbf{x}} + \delta_1 \mathbf{u_1} + \delta_2 \mathbf{u_2})$ to stwierdzimy, że zbiory X_1 i X_2 nie mogą być niezależne. To wnioskowanie jest tożsame z warunkiem interakcji pomiędzy dwoma rozłącznymi zbiorami przedstawionym we wzorze (2.11). Możemy zatem uznać, że strategie dekompozycji bazujące na sprawdzaniu monotoniczności, których ten wzór dotyczy, mogą zostać zaklasyfikowane jako empiryczne techniki typu „linkage learning", które nigdy nie wykryją nieistniejących zależności.

3.1.2 Pomijanie istniejących zależności lub raportowanie nieistniejących

Strategie dekompozycji bazujące na grupowaniu różnicowym nie wykryją żadnych zależności, które w rzeczywistości nie istnieją wtedy i tylko wtedy, gdy rozważana funkcja jest addytywnie separowalna [89, 123]. Funkcja jest addytywnie separowalna jeżeli każda para jej składowych jest również addytywnie separowalna. Rozważmy dwie separowalne składowe f_i i f_j. Są one addytywnie separowalne jeżeli następujący warunek jest spełniony

$$\frac{\partial^2 f}{\partial x_q \partial x_p} = 0 \tag{3.3}$$

gdzie x_p i x_q są dowolnymi argumentami odpowiednio f_i i f_j [123]. W przeciwnym wypadku, składowe f_i i f_j są nieaddytywnie separowalne. Jeżeli wszystkie pary składowych rozważanej funkcji są nieaddytywnie separowalne to jest ona również nieaddytywnie separowalna. W takim przypadku, strategie bazujące na grupowaniu różnicowym mogą zaraportować zależności, które w rzeczywistości nie istnieją. Za przykład niech posłuży zadanie minimalizacji funkcji $\ddot{f}_{c,5} : [0, 5]^2 \to [0, 100]$ o wzorze $\ddot{f}_{c,5}(\mathbf{x}) = (x_1 + x_2)^2$. Jeżeli użyjemy dowolnej zachłannej metody optymalizacji, która jest zarazem deterministyczna to podczas optymalizacji zmiennej x_1 zbiegnie ona do tego samego punktu ze zbioru dopuszczalnego dla każdej możliwej wartości

zmiennej x_2. Gdy rozważymy odwrotną sytuację tj. optymalizację zmiennej x_2 przy stałej wartości zmiennej x_1 to wyciągniemy dokładnie te same wnioski. Zakładając nieskończoną dokładność użytej metody optymalizacji, w obu przypadkach zbiegnie ona do 0. Zmienne x_1 i x_2 są zatem niezależne. Strategie dekompozycji, które bazują na grupowaniu różnicowym mogą jednak błędnie wykryć zależność między nimi, ponieważ po podstawieniu do wzoru (2.2) wartości $(a, a+\delta, b_1, b_2) = (1, 2, 1, 2)$ otrzymamy $\Delta_1 = 5$ i $\Delta_2 = 7$. Różne wartości delt są podstawą do stwierdzenia zależności pomiędzy zmiennymi x_1 i x_2 przez strategie bazujące na grupowaniu różnicowym.

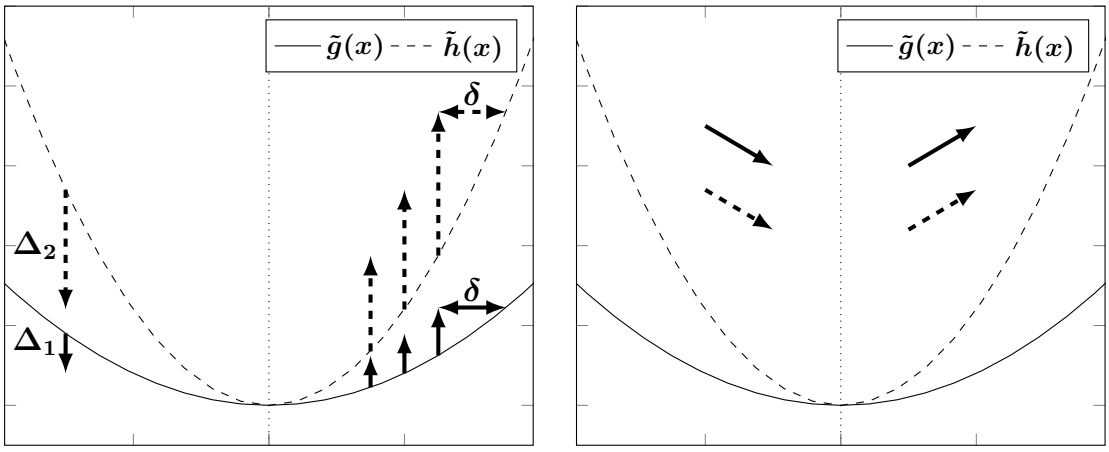
Zgodnie z rozdziałem 3.1.1, strategie bazujące na sprawdzaniu monotoniczności nigdy nie wykryją zależności, które w rzeczywistości nie istnieją. W praktyce, ze względu na niedokładność reprezentacji liczb zmiennoprzecinkowych, odpowiednie wartości ϵ muszą zostać dostarczone by rozstrzygnąć prawdziwość nierówności we wzorach (2.10) i (2.11). Zakładając, że wartość ϵ nigdy nie będzie zbyt niska to takie strategie wykryją tylko istniejące zależności dla problemów zarówno z addytywnie, jak i nieaddytywnie separowalnymi podproblemami. Tej cechy nie posiadają strategie bazujące na grupowaniu różnicowym, dla których odpowiednia wartość ϵ we wszystkich sprawdzeniach, czy dwie zmienne lub dwie grupy zmiennych decyzyjnych oddziałują na siebie wzajemnie, nie gwarantuje wykrycia jedynie istniejących interakcji dla problemów składających się z nieaddytywnie separowalnych podproblemów. Mimo tej wady, strategie bazujące na grupowaniu różnicowym dekomponują problemy optymalizacyjne dokładniej niż strategie bazujące na sprawdzaniu monotoniczności [29, 89, 94, 123]. Przyczyną jest podatność dotychczas zaproponowanych strategii bazujących na sprawdzaniu monotoniczności do pomijania istniejących zależności.

Na początku niniejszego rozdziału, podane zostały definicje funkcji $\tilde{f}$, $\tilde{g}$ i $\tilde{h}$. Na ich podstawie, zostaną zaprezentowane kolejne przykłady by dokładniej pokazać różnice pomiędzy grupowaniem różnicowym, a sprawdzaniem monotoniczności. W tym celu wzory, na których bazują strategie wywodzące się z tych dwóch podejść, zostaną przeformułowane by uwzględniały funkcje $\tilde{g}$ i $\tilde{h}$. Porównanie wartości Δ_1 i Δ_2 by sprawdzić zależność między x_1 i x_2, wzory (2.2) i (2.3), można zapisać jako

$$\tilde{g}(a+\delta) - \tilde{g}(a) \neq \tilde{h}(a+\delta) - \tilde{h}(a) \tag{3.4}$$

natomiast warunek interakcji między zmiennymi x_1 i x_2 dla sprawdzania monotoniczności, wzór (2.10), przyjmuje następującą postać

$$\tilde{g}(a_1) \leq \tilde{g}(a_2) \wedge \tilde{h}(a_1) > \tilde{h}(a_2) \tag{3.5}$$

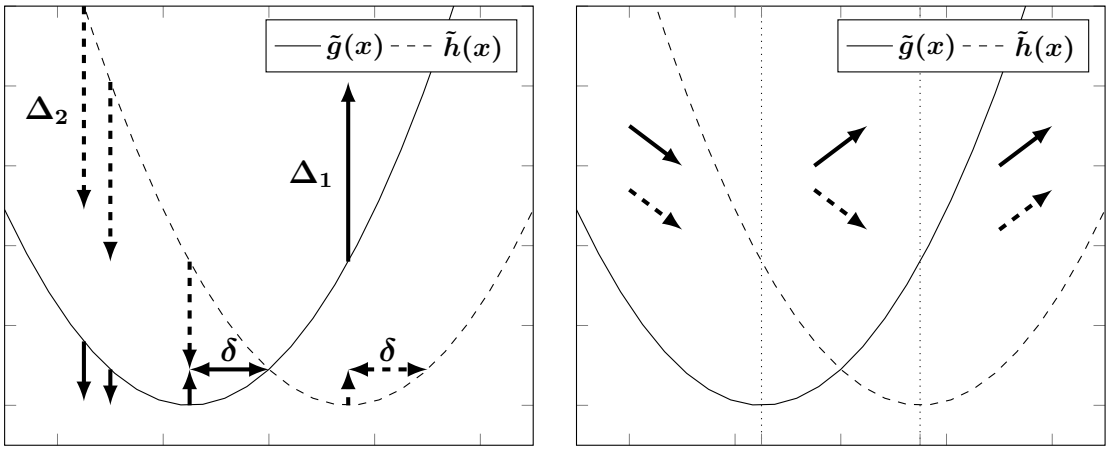


(a) Grupowanie różnicowe (b) Sprawdzanie monotoniczności

Rysunek 3.3: Możliwość wykrycia nieistniejących zależności po przeskalowaniu funkcji

Rysunek 3.3 przedstawia kolejny przykład, w którym strategie bazujące na grupowaniu różnicowym wykryją zależność między zmiennymi x_1 i x_2, która w rzeczywistości nie istnieje. Skalowanie funkcji przez dodatnią liczbę nie zmienia żadnych wyników porównań dowolnych wartości funkcji przed i po skalowaniu. Jeżeli $\tilde{h}(x) = \alpha \cdot \tilde{g}(x)$, gdzie $\alpha > 0$, to $\forall_{a_1.a_2} \tilde{h}(a_1) < \tilde{h}(a_2) \iff \tilde{g}(a_1) < \tilde{g}(a_2)$ i $\forall_{a_1.a_2} \tilde{h}(a_1) = \tilde{h}(a_2) \iff \tilde{g}(a_1) = \tilde{g}(a_2)$. Możemy zatem stwierdzić, że w tym konkretnym przypadku, funkcja $\tilde{f}$ jest w pełni separowalna. Na rysunku Rysunek 3.3a, delty Δ_1 i Δ_2 reprezentowane są przez wektory, których długość i zwrot zostały wyznaczone na podstawie różnych wartości a rozproszonych po osi poziomej i tej samej wartości δ. Należy zauważyć, że przynajmniej dla jednej wartości a, oba wektory różnią się. Na tej podstawie, strategie dekompozycji bazujące na grupowaniu różnicowym mogą wykryć zależność pomiędzy zmiennymi x_1 i x_2, mimo że w rzeczywistości ona nie istnieje. Sprawdzanie monotoniczności (Rysunek 3.3b) rozważa m.in. przedziały monotoniczności, które są takie same dla funkcji $\tilde{g}$ i $\tilde{h}$. Jest to jednak warunek konieczny, lecz niewystarczający by tego typu strategie nie wykryły żadnych zależności. W przypadku skalowania funkcji przez dodatnią wartość α możemy zapisać, że $\forall_{\alpha>0} \tilde{h}(a_1) \leq \tilde{h}(a_2) \iff \tilde{g}(a_1) \leq \tilde{g}(a_2)$. Wykorzystując ten fakt we wzorze (3.5) dochodzimy do sprzeczności $\tilde{h}(a_1) \leq \tilde{h}(a_2) \wedge \tilde{h}(a_1) > \tilde{h}(a_2)$. Możemy zatem uznać, że strategie bazujące na sprawdzaniu monotoniczności nigdy nie wykryją nieistniejących zależności w przypadku skalowania funkcji.

Jedną z niewątpliwych zalet grupowania różnicowego jest jego wysoka czułość podczas szukania interakcji pomiędzy zmiennymi lub grupami zmiennych. Na rysunku Rysunek 3.4a można zauważyć, że istnieje wiele różnych wartości a, dla których oddziaływanie pomiędzy x_1 i x_2 zostanie prawidłowo wykryte. Z drugiej strony,



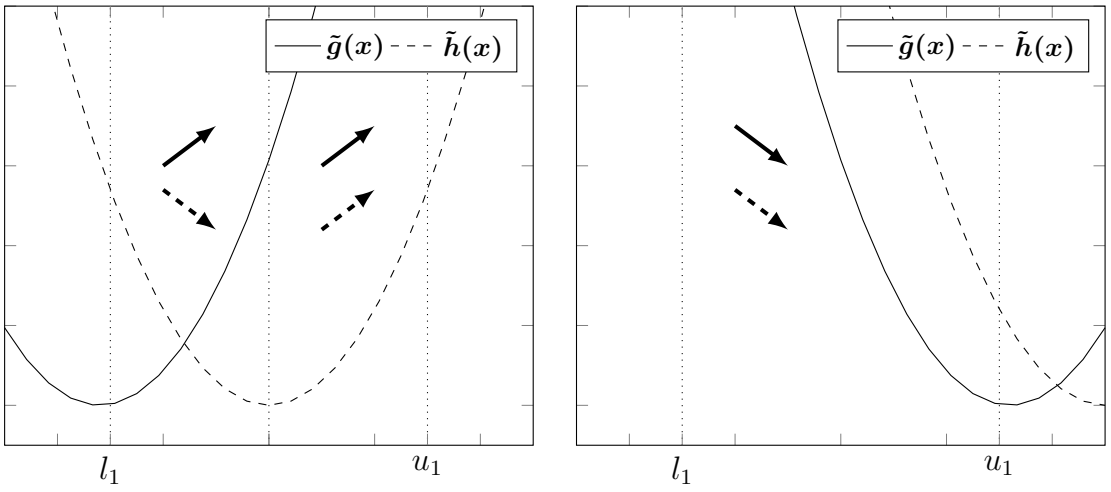
(a) Grupowanie różnicowe (b) Sprawdzanie monotoniczności

Rysunek 3.4: Czułość wykrywania interakcji

w przypadku sprawdzania monotoniczności (Rysunek 3.4b), tylko rozpatrzenie wartości a_1 i a_2 ze środkowego przedziału daje nam gwarancję wykrycia zależności, ponieważ funkcja $\tilde{g}$ jest rosnąca na tym przedziale, natomiast funkcja $\tilde{h}$ jest malejąca. Przeciwny wniosek możemy wyciągnąć gry obie wartości a_1 i a_2 zostaną wzięte z lewego lub prawego przedziału. Strategie bazujące na sprawdzaniu monotoniczności nie wykryją wtedy istniejącej zależności pomiędzy zmiennymi x_1 i x_2, ponieważ obie funkcje odpowiednio maleją i rosną na tych przedziałach. W pozostałych przypadkach np. a_1 pochodzące z lewego przedziału oraz a_2 ze środkowego, tego typu strategie mogą, ale nie muszą wykryć interakcji.

3.1.3 Problemy optymalizacyjne z ograniczeniami, a wykrywanie zależności pomiędzy zmiennymi

Ograniczenia nakładane na zmienne mogą także wpływać na jakość dekompozycji zwracanej przez strategie bazujące na sprawdzaniu monotoniczności — konkretnie mogą prowadzić do pomijania istniejących zależności. Za przykład niech posłuży zadanie minimalizacji funkcji $\ddot{f}_{c,6} : [-2,2] \times [-8,8] \to [0,100]$ o wzorze $\ddot{f}_{c,6}(\mathbf{x}) = (x_1 + x_2)^2$. Funkcja $\ddot{f}_{c,6}$ jest w pełni nieseparowalna, ponieważ jej minimalną wartość uzyskujemy gdy $x_1 = -x_2$. Jeżeli do wzoru (2.10) podstawimy następujące wartości $(a_1, a_2, b_1, b_2) = (-2, -1, 2, 1)$ to takie sprawdzenie monotoniczności rzeczywiście wykryje zależność pomiędzy x_1 i x_2. Z drugiej strony, jeżeli weźmiemy pod uwagę $b_1 = -6$ i $b_2 = -5$ to nie znajdziemy żadnych wartości a_1 i a_2, należących do zbioru $[-2, 2]$, takich, że wykryta zostanie interakcja między tymi zmiennymi. W celu wyjaśnienia tego zjawiska, rozpatrzmy podobną sytuację, która



(a) Możliwe (b) Niemożliwe

Rysunek 3.5: Wykrywanie zależności przez sprawdzanie monotoniczności w problemach z ograniczeniami

została zaprezentowana na rysunku Rysunek 3.5b. Wartości b_1 i b_2, podobnie jak w poprzednim przykładzie, zostały tak dobrane by strategie bazujące na sprawdzaniu monotoniczności nigdy nie wykryły, że zmienne x_1 i x_2 oddziałują na siebie wzajemnie. Jest to spowodowane faktem, że różne wartości a_1 i a_2 mogą zostać wybrane jedynie ze zbioru dopuszczalnego pierwszej zmiennej, czyli $[l_1, u_1]$, na którym obie funkcje $\tilde{g}$ i $\tilde{h}$ są malejące.

3.1.4 Wnioski — motywacje dla dalszych prac

Na podstawie analizy literatury, możemy założyć, że istnieje skuteczny mechanizm, który dostarczy nam odpowiednią wartość ϵ dla wszystkich sprawdzeń, czy zachodzi interakcja pomiędzy dwoma grupami zmiennych [126]. W przypadku grupowania różnicowego ciągle nie będziemy mieć jednak pewności, czy wykryte zależności w rzeczywistości istnieją. Z drugiej strony, strategie bazujące na sprawdzaniu monotoniczności mają tendencje do niewykrywania istniejących zależności. Istnieje zatem potrzeba zaproponowania nowej strategii dekompozycji, która będzie wolna od obu tych wad.

Należy również pamiętać, że dekompozycja jest jedynie częścią całego procesu optymalizacji. Nowo powstała strategia powinna zatem zużywać możliwie małą liczbę wyliczeń funkcji przystosowania w trakcie dekompozycji rozważanego problemu optymalizacyjnego.

W ostatnich latach, badacze skupiali się głównie na zmniejszaniu kosztu dekompozycji przez strategie bazujące na grupowaniu różnicowym [73, 150]. Typowy zbiór

problemów testowych składa się w większości z problemów, które zbudowane są z addytywnie separowalnych podproblemów [70, 73, 82, 123, 149, 150]. Z tego powodu, dokładność dekompozycji nieaddytywnie separowalnych problemów optymalizacyjnych była często pomijana w analizie porównawczej różnych strategii dekompozycji.

3.2 Wprowadzenie

Skoro teoria stojąca za sprawdzaniem monotoniczności gwarantuje, że wszystkie wykryte oddziaływania pomiędzy zmiennymi decyzyjnymi istnieją w rzeczywistości to należy zastanowić się w jaki sposób zmniejszyć prawdopodobieństwo, że przynajmniej jedna istniejąca zależność zostanie pominięta. Najprostszym sposobem wydaje się sprawdzenie ogromnej liczby próbek pod kątem możliwej interakcji pomiędzy zmiennymi lub grupami zmiennych. Pojedyncza próbka dla wzoru (2.10) to a_1, a_2, b_1, b_2 i $\ddot{\mathbf{x}}$. W przypadku rekursywnej wersji sprawdzania monotoniczności (wzór (2.11)) pojedynczą próbką jest δ_1, δ_2, $\mathbf{u_1}$, $\mathbf{u_2}$ i $\ddot{\mathbf{x}}$. Takie podejście wiąże się jednak z wysokim kosztem dekompozycji mierzonym liczbą wyliczeń funkcji przystosowania. Koszt ten może być nieadekwatny w kontekście całego procesu optymalizacji.

Inkrementacyjne i rekursywne grupowanie rankingowe (ang. *Incremental Recursive Ranking Grouping*, IRRG) jest nową strategią dekompozycji bazującą na sprawdzaniu monotoniczności. Nowa strategia została zaproponowana przez autora niniejszej pracy i jej koncepcja opiera się na porównywaniu dwóch rankingów próbek. W celu sprawdzenia zależności pomiędzy zmiennymi tworzymy dwa rankingi, które zawierają n_s próbek, gdzie n_s jest parametrem podawanym przez użytkownika. Rankingi są tworzone by zwiększyć czułość wykrywania istniejących zależności pomiędzy zmiennymi decyzyjnymi. W celu sprawdzenia, czy zmienne x_p i x_q bezpośrednio oddziałują na siebie wzajemnie wykorzystamy

- wartości b_1 i b_2 ($b_1 \neq b_2$) dla zmiennej x_q,
- rozwiązanie $\ddot{\mathbf{x}}$,
- n_s równomiernie wygenerowanych wartości ze zbioru dopuszczalnego zmiennej x_p.

Oznaczmy n_s wartości dla zmiennej x_p jako $\tilde{a}_1$, ..., $\tilde{a}_{n_s}$. Korzystając ze wszystkich wyżej wymienionych wartości, możemy utworzyć dwa rankingi $\mathbf{r_1} = [r_{1,1}, ..., r_{1,n_s}]$ i $\mathbf{r_2} = [r_{2,1}, ..., r_{2,n_s}]$. Wartość j-tego rankingu dla i-tego elementu, oznaczonego jako $r_{j,i}$ zależy od wartości $f(\ddot{\mathbf{x}})|_{x_p=\tilde{a}_i, x_q=b_j}$. Jeżeli istnieje takie $i \in \{1, ..., n_s\}$, że $r_{1,i} \neq r_{2,i}$ to uznajemy zmienne x_p i x_q za oddziałujące na siebie wzajemnie. Za przykład niech

posłuży sprawdzenie, czy zmienne x_1 i x_2 są od siebie zależne rozpatrując funkcję $\ddot{f}_{c,7} : [-3,3]^4 \to [-111,111]$ o wzorze $\ddot{f}_{c,7}(\mathbf{x}) = (x_1 + x_2)^2 \cdot x_3 + x_4$. Przyjmujemy: $\ddot{\mathbf{x}} = [1,0,3,2]$, $b_1 = 1$, $b_2 = 2$ i $n_s = 3$ wartości równomiernie wygenerowanych ze zbioru $[-3,3]$. Otrzymujemy zatem $\tilde{a}_1 = -3$, $\tilde{a}_2 = 0$ i $\tilde{a}_3 = 3$. W kolejnym kroku wyliczamy następujące wartości na podstawie funkcji przystosowania

- $\ddot{f}_{c,7}(\ddot{\mathbf{x}})|_{x_1=-3,x_2=1} = 14$, $\ddot{f}_{c,7}(\ddot{\mathbf{x}})|_{x_1=0,x_2=1} = 5$ i $\ddot{f}_{c,7}(\ddot{\mathbf{x}})|_{x_1=3,x_2=1} = 50$ do utworzenia pierwszego rankingu,
- $\ddot{f}_{c,7}(\ddot{\mathbf{x}})|_{x_1=-3,x_2=2} = 5$, $\ddot{f}_{c,7}(\ddot{\mathbf{x}})|_{x_1=0,x_2=2} = 14$ i $\ddot{f}_{c,7}(\ddot{\mathbf{x}})|_{x_1=3,x_2=2} = 77$ na potrzeby drugiego rankingu.

Rankingi utworzone na podstawie powyższych wartości funkcji przystosowania mają następującą postać

- $\mathbf{r_1} = [2,1,3]$,
- $\mathbf{r_2} = [1,2,3]$.

Zmienne decyzyjne x_1 i x_2 uznajemy za bezpośrednio zależne, ponieważ $r_{1,1} \neq r_{2,1}$.

Stosując jedynie sprawdzanie bezpośredniej interakcji pomiędzy parami zmiennych decydujemy się na złożoność obliczeniową $\mathcal{O}(n^2)$. Współczesne strategie interakcji mają złożoność $\mathcal{O}(n\log(n))$ [29, 123, 125, 126]. Uogólnijmy zatem koncepcję tworzenia i porównywania rankingów do wykrywania zależności pomiędzy dwoma zbiorami zmiennych, X_1 i X_2, zgodnie ze wzorem (2.11). Do utworzenia dwóch rankingów potrzebujemy n_s wartości, tym razem, dla każdej zmiennej x_j należącej do zbioru X_1. Są one również równomiernie generowane ze zbioru dopuszczalnego j-tej zmiennej decyzyjnej. Oznaczmy te wartości jako $\tilde{a}_{1,j}$, ..., $\tilde{a}_{n_s,j}$. Na podstawie tych wartości oraz danego wektora zmiennych decyzyjnych $\ddot{\mathbf{x}} = [\ddot{x}_1, ..., \ddot{x}_n]$, można zdefiniować n_s wektorów $\tilde{\mathbf{x}}_\mathbf{i} = [\tilde{x}_{i,1}, ..., \tilde{x}_{i,n}]$ o następujących elementach

$$\tilde{x}_{i,j} = \begin{cases} \tilde{a}_{\pi_j(i),j} & , x_j \in X_1 \\ \ddot{x}_j & , x_j \notin X_1 \end{cases} \tag{3.6}$$

gdzie π_j jest losową permutacją zbioru $\{1, ..., n_s\}$ wygenerowaną dla j-tej zmiennej decyzyjnej. Wektory te są następnie podstawą do utworzenia dwóch rankingów $\mathbf{r_1}$ i $\mathbf{r_2}$, gdzie $r_{1,i}$ bazuje na wartości $f(\tilde{\mathbf{x}}_\mathbf{i})$, natomiast $r_{2,i}$ bierze pod uwagę wartość $f(\tilde{\mathbf{x}}_\mathbf{i} + \delta_2\mathbf{u_2})$. Należy zauważyć, że uwzględnienie wszystkich możliwych kombinacji wartości $\tilde{a}_{i,j}$ wymagałoby utworzenia aż $n_s^{|X_1|}$ wektorów zmiennych decyzyjnych. Nie jest to jednak rekomendowane ze względów praktycznych. Niemniej jednak należy uwzględnić, że zmienne decyzyjne będące elementami zbioru X_1 mogą oddziaływać

na siebie wzajemnie. Z tego względu, użyto losowych permutacji by wyeliminować potencjalny wpływ specyficznego ułożenia wartości $\tilde{a}_{i,j}$ na decyzję o zależności zbiorów X_1 i X_2. Analogicznie do sprawdzania ewentualnej zależności pomiędzy zmiennymi x_p i x_q, jeżeli istnieje takie $i^* \in \{1, ..., n_s\}$, że $r_{1,i^*} \neq r_{2,i^*}$ to zbiory X_1 i X_2 uznajemy za oddziałujące na siebie wzajemnie. W takiej sytuacji, istnieje przynajmniej jedno $i^* \in \{1, ..., n_s\}$, że poniższy warunek jest spełniony

$$f\left(\tilde{\mathbf{x}}_{\mathbf{r_{1,i^*}}}\right) \leq f\left(\tilde{\mathbf{x}}_{\mathbf{r_{2,i^*}}}\right) \wedge f\left(\tilde{\mathbf{x}}_{\mathbf{r_{1,i^*}}} + \delta_2 \mathbf{u_2}\right) > f\left(\tilde{\mathbf{x}}_{\mathbf{r_{2,i^*}}} + \delta_2 \mathbf{u_2}\right) \tag{3.7}$$

Należy zauważyć, że powyższy wzór jest specyficznym przypadkiem wzoru (2.11). Możemy zatem stwierdzić, że podstawowe założenia strategii IRRG bazują na sprawdzaniu monotoniczności. Na przykład, sprawdźmy czy dwa zbiory zmiennych decyzyjnych $X_1 = \{x_1, x_4\}$ i $X_2 = \{x_2, x_3\}$ oddziałują na siebie wzajemnie w kontekście zdefiniowanej wcześniej funkcji $\ddot{f}_{c,7}$. Przyjmujemy: $\ddot{\mathbf{x}} = [1, 1, 2, 2]$, $\delta_2 = \sqrt{2}$, $\mathbf{u_2} = \left[0, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0\right]$, $\pi_1(1) = 1$, $\pi_1(2) = 2$, $\pi_1(3) = 3$, $\pi_4(1) = 3$, $\pi_4(2) = 2$, $\pi_4(3) = 1$ i $n_s = 3$ wartości równomiernie wygenerowanych ze zbioru $[-3, 3]$ dla zmiennych x_1 i x_4, które należą do zbioru X_1. Otrzymujemy zatem $\tilde{a}_{1,1} = -3$, $\tilde{a}_{2,1} = 0$, $\tilde{a}_{3,1} = 3$, $\tilde{a}_{1,4} = -3$, $\tilde{a}_{2,4} = 0$, $\tilde{a}_{3,4} = 3$, $\tilde{\mathbf{x}}_\mathbf{1} = [-3, 1, 2, 3]$, $\tilde{\mathbf{x}}_\mathbf{2} = [0, 1, 2, 0]$, $\tilde{\mathbf{x}}_\mathbf{3} = [3, 1, 2, -3]$, $\tilde{\mathbf{x}}_\mathbf{1} + \delta_2 \mathbf{u_2} = [-3, 2, 3, 3]$, $\tilde{\mathbf{x}}_\mathbf{2} + \delta_2 \mathbf{u_2} = [0, 2, 3, 0]$ i $\tilde{\mathbf{x}}_\mathbf{3} + \delta_2 \mathbf{u_2} = [3, 2, 3, -3]$. W kolejnym kroku wyliczamy następujące wartości na podstawie funkcji przystosowania

- $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{1}) = 11$, $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{2}) = 2$ i $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{3}) = 29$ do utworzenia pierwszego rankingu,
- $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{1} + \delta_2 \mathbf{u_2}) = 6$, $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{2} + \delta_2 \mathbf{u_2}) = 12$ i $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{3} + \delta_2 \mathbf{u_2}) = 72$ na potrzeby drugiego rankingu.

Rankingi utworzone na podstawie powyższych wartości funkcji przystosowania mają następującą postać

- $\mathbf{r_1} = [2, 1, 3]$,
- $\mathbf{r_2} = [1, 2, 3]$.

Uznajemy, że dwa zbiory zmiennych decyzyjnych $X_1 = \{x_1, x_4\}$ i $X_2 = \{x_2, x_3\}$ oddziałują na siebie wzajemnie, ponieważ $r_{1,1} \neq r_{2,1}$, a w konsekwencji $\ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{2}) \leq \ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{1}) \wedge \ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{2} + \delta_2 \mathbf{u_2}) > \ddot{f}_{c,7}(\tilde{\mathbf{x}}_\mathbf{1} + \delta_2 \mathbf{u_2})$.

W kolejnym podrozdziale opisane zostanie rekursywne grupowanie rankingowe, które jest najważniejszą częścią strategii IRRG. Następnie zaprezentowany zostanie ogólny sposób działania strategii IRRG. W ostatnim podrozdziale tego rozdziału, omówiona zostanie jej złożoność obliczeniowa.

3.3 Rekursywne grupowanie rankingowe

Rekursywne grupowanie różnicowe (ang. *Recursive Ranking Grouping*, RRG), jako główna część strategii IRRG, odpowiada za wykrywanie zależności pomiędzy dwoma rozłącznymi zbiorami zmiennych decyzyjnych X_1 i X_2. Zanim przejdziemy do przedstawienia rekursywnego grupowania różnicowego, omówione zostaną wszystkie funkcje pomocnicze, z których RRG korzysta. Następujące oznaczenia, zostaną w nich użyte. Zbiór grup składających się ze zmiennych decyzyjnych oznaczonych jako oddziałujące na siebie wzajemnie będzie reprezentowany przez symbol $\boldsymbol{G}$, natomiast V będzie zbiorem zmiennych decyzyjnych, dla których do tej pory nie znaleziono żadnych interakcji. Macierz $\bar{\boldsymbol{X}}_{\mathbf{1}}$ przechowywać będzie n_s równomiernie wygenerowanych wartości dla każdej zmiennej decyzyjnej. Jej wymiary to $n_s \times n$, gdzie wartości w j-tej kolumnie należą do j-tej zmiennej decyzyjnej.

Funkcja UwzględnijPojedynczeZmienne (Pseudokod 1) służy do sprawdzenia, czy zmienne będące elementami zbioru V są warte uwzględnienia podczas szukania zależności. Decyzja będzie pozytywna, jeżeli przynajmniej jeden z poniższych warunków będzie spełniony.

1. Zbiór $\boldsymbol{G}$ jest pusty, zatem żadna interakcja nie została do tej pory wykryta (linia 2).

2. Zbiór V jest jednoelementowy (linia 2). Istnieje zatem spora szansa, że jeszcze nie wykryta została przynajmniej jedna zależność, która w rzeczywistości istnieje. Dodatkowo, uwzględnienie pojedynczej zmiennej nie zwiększy znacząco kosztu pozostałych funkcji pomocniczych, mierzonego liczbą wyliczeń funkcji przystosowania.

3. Zbiór V jest losowo dzielony na dwa rozłączne i możliwie równoliczne podzbiory V_1 i V_2. Wyjątkiem gdy oba podzbiory nie mają równej liczby elementów jest przypadek, w którym liczba elementów w zbiorze V jest nieparzysta. Jeżeli zostanie wykryta interakcja pomiędzy V_1 i V_2 to wszystkie zmienne decyzyjne ze zbioru V zostaną uwzględnione podczas szukania zależności (linie 6–14).

4. Jeżeli pomiędzy V i dowolną grupą z $\boldsymbol{G}$ zachodzi interakcja to również wszystkie zmienne decyzyjne ze zbioru V zostaną uwzględnione podczas szukania zależności (linie 15–21).

W ramach funkcji UwzględnijPojedynczeZmienne kilkukrotnie tworzony jest ranking $\mathbf{r_1}$. W tym celu wywoływana jest funkcja UtwórzPierwszyRanking

(Pseudokod 2), która tworzy ranking zgodnie z opisem przedstawionym w rozdziale 3.2. Funkcja UtwórzPierwszyRanking zwraca również wektor wartości funkcji f, dla których powstał ranking. Wektor ten, oznaczony jako $\bar{\mathbf{y}}_\mathbf{1}$, wykorzystywany jest przez inne funkcje pomocnicze.

Pseudokod 1 Sprawdzenie, czy warto uwzględniać zmienne, dla których nie znaleziono dotychczas żadnych interakcji, w poszukiwaniu zależności

wejście: V: pojedyncze zmienne do sprawdzenia, $\boldsymbol{G}$: grupy zależnych zmiennych, $\mathbf{x_{hq}}$: wysokiej jakości wektor zmiennych decyzyjnych, $\bar{\boldsymbol{X}}_\mathbf{1}$: macierz o wymiarach $n_s \times n$ zawierająca równomiernie wygenerowane wartości dla każdej zmiennej decyzyjnej, $\bar{\mathbf{x}}_\mathbf{2}$: wektor zmiennych decyzyjnych różny od $\mathbf{x_{hq}}$, f: problem optymalizacyjny, n_s: liczba próbek
wyjście: decyzja, czy przynajmniej jedna zmienna ze zbioru V może być nieseparowalna

1: **funkcja** UwzględnijPojedynczeZmienne(V, $\boldsymbol{G}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, f, n_s$)
2: **jeżeli** $|\boldsymbol{G}| = 0$ **lub** $|V| = 1$ **to**
3: **zwróć** *tak*
4: **jeżeli** $|V| = 0$ **to**
5: **zwróć** *nie*
6: $V \leftarrow$ przetasuj zbiór V
7: $V_1 \leftarrow$ pierwsza połowa zbioru V
8: $V_2 \leftarrow$ druga połowa zbioru V
9: $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}) \leftarrow$ UtwórzPierwszyRanking($V_1, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
10: **jeżeli** ZachodziInterakcja($V_1, V_2, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$) **to**
11: **zwróć** *tak*
12: $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}) \leftarrow$ UtwórzPierwszyRanking($V_2, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
13: **jeżeli** ZachodziInterakcja($V_2, V_1, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$) **to**
14: **zwróć** *tak*
15: **dla** $i \leftarrow 1$ **do** $|\boldsymbol{G}|$ **wykonaj**
16: $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}) \leftarrow$ UtwórzPierwszyRanking($V, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
17: **jeżeli** ZachodziInterakcja($V, \boldsymbol{G}[i], \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$) **to**
18: **zwróć** *tak*
19: $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}) \leftarrow$ UtwórzPierwszyRanking($\boldsymbol{G}[i], \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
20: **jeżeli** ZachodziInterakcja($\boldsymbol{G}[i], V, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$) **to**
21: **zwróć** *tak*
22: **zwróć** *nie*

Kolejną funkcją pomocniczą jest funkcja ZachodziInterakcja (Pseudokod 3). Jej celem jest sprawdzenie, czy dwa rozłączne zbiory zmiennych decyzyjnych X_1 i X_2 oddziałują na siebie wzajemnie. Teoretycznie, funkcja ZachodziInterakcja powinna utworzyć ranking $\mathbf{r_2}$ i zweryfikować, czy różni się on od rankingu $\mathbf{r_1}$. W praktyce, żeby zaoszczędzić jak najwięcej niepotrzebnych wyliczeń funkcji przystosowania, ewentualna zmiana kolejności w $\mathbf{r_2}$ sprawdzana jest na bieżąco. Próbki przetwarzane są w kolejności wyznaczonej przez ranking $\mathbf{r_1}$ aż do momentu, gdy późniejsza próbka będzie lepsza od swojej poprzedniczki w kontekście rankingu $\mathbf{r_2}$.

Pseudokod 2 Tworzenie pierwszego rankingu

wejście: X_1: zbiór zmiennych decyzyjnych, $\mathbf{x}$: wektor zmiennych decyzyjnych, $\bar{\boldsymbol{X}}_\mathbf{1}$: macierz o wymiarach $n_s \times n$ zawierająca równomiernie wygenerowane wartości dla każdej zmiennej decyzyjnej, f: problem optymalizacyjny, n_s: liczba próbek
wyjście: $\bar{\mathbf{y}}_\mathbf{1}$: wartości funkcji f wyliczone na podstawie $\mathbf{x}$ i $\bar{\boldsymbol{X}}_\mathbf{1}$, $\mathbf{r_1}$: ranking wartości będących elementami $\bar{\mathbf{y}}_\mathbf{1}$

1: **funkcja** UtwórzPierwszyRanking($X_1, \mathbf{x}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
2: $\quad\bar{\mathbf{x}} \leftarrow \mathbf{x}$
3: $\quad$**dla** $i \leftarrow 1$ **to** n_s **wykonaj**
4: $\quad\quad\bar{\mathbf{x}}[X_1] \leftarrow \bar{\boldsymbol{X}}_\mathbf{1}[i][X_1]$
5: $\quad\quad\bar{\mathbf{y}}_\mathbf{1}[i] \leftarrow f(\bar{\mathbf{x}})$ ▷ wartości funkcji f wyliczone na podstawie $\mathbf{x}$ i $\bar{\boldsymbol{X}}_\mathbf{1}$
6: $\quad\mathbf{r_1} \leftarrow$ indeksy i od 1 do n_s posortowane względem $\bar{\mathbf{y}}_\mathbf{1}[i]$
7: $\quad$**zwróć** $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1})$

Pseudokod 3 Sprawdzenie, czy zachodzi interakcja pomiędzy dwoma rozłącznymi zbiorami zmiennych decyzyjnych

wejście: X_1, X_2: rozłączne zbiory zmiennych decyzyjnych, $\mathbf{x_{hq}}$: wysokiej jakości wektor zmiennych decyzyjnych, $\bar{\boldsymbol{X}}_\mathbf{1}$: macierz o wymiarach $n_s \times n$ zawierająca równomiernie wygenerowane wartości dla każdej zmiennej decyzyjnej, $\bar{\mathbf{x}}_\mathbf{2}$: wektor zmiennych decyzyjnych różny od $\mathbf{x_{hq}}$, $\bar{\mathbf{y}}_\mathbf{1}$: wartości funkcji f wyliczone na podstawie $\mathbf{x_{hq}}$ i $\bar{\boldsymbol{X}}_\mathbf{1}$, $\mathbf{r_1}$: ranking wartości będących elementami $\bar{\mathbf{y}}_\mathbf{1}$, f: problem optymalizacyjny, n_s: liczba próbek
wyjście: decyzja, czy zbiory X_1 i X_2 oddziałują na siebie wzajemnie

1: **funkcja** ZachodziInterakcja($X_1, X_2, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
2: $\quad\bar{\mathbf{x}} \leftarrow \mathbf{x_{hq}}$
3: $\quad\bar{\mathbf{x}}[X_2] \leftarrow \bar{\mathbf{x}}_\mathbf{2}[X_2]$
4: $\quad\bar{\mathbf{x}}[X_1] \leftarrow \bar{\boldsymbol{X}}_\mathbf{1}[\mathbf{r_1}[1]][X_1]$
5: $\quad\bar{\mathbf{y}}_\mathbf{2}[1] \leftarrow f(\bar{\mathbf{x}})$ ▷ wartości funkcji f wyliczone na podstawie $\mathbf{x_{hq}}$, $\bar{\mathbf{x}}_\mathbf{2}$ i $\bar{\boldsymbol{X}}_\mathbf{1}$
6: $\quad$**dla** $i \leftarrow 2$ **to** n_s **wykonaj**
7: $\quad\quad\epsilon_1 \leftarrow f_\gamma(\sqrt{n}+1) \cdot (|\bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i]]| + |\bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i-1]]|)$ ▷ wzór (2.5)
8: $\quad\quad$**jeżeli** $\operatorname{sgn}(\bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i]] - \bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i-1]], \epsilon_1) = 0$ **to** ▷ wzór (3.20)
9: $\quad\quad\quad$**kontynuuj**
10: $\quad\quad\bar{\mathbf{x}}[X_1] \leftarrow \bar{\boldsymbol{X}}_\mathbf{1}[\mathbf{r_1}[i]][X_1]$
11: $\quad\quad\bar{\mathbf{y}}_\mathbf{2}[i] \leftarrow f(\bar{\mathbf{x}})$
12: $\quad\quad\epsilon_2 \leftarrow f_\gamma(\sqrt{n}+1) \cdot (|\bar{\mathbf{y}}_\mathbf{2}[i]| + |\bar{\mathbf{y}}_\mathbf{2}[i-1]|)$ ▷ wzór (2.5)
13: $\quad\quad$**jeżeli** $\operatorname{sgn}(\bar{\mathbf{y}}_\mathbf{2}[i] - \bar{\mathbf{y}}_\mathbf{2}[i-1], \epsilon_2) < 0$ **to** ▷ wzór (3.20)
14: $\quad\quad\quad$**zwróć** *tak*
15: $\quad$**zwróć** *nie*

Niedokładność reprezentacji liczb zmiennoprzecinkowych, podobnie jak w przypadku strategii bazujących na grupowaniu różnicowym [89, 94, 121, 123, 125, 126], może wpłynąć na decyzję o zależności lub niezależności dwóch grup składających się ze zmiennych decyzyjnych podejmowaną przez funkcję ZachodziInterakcja — dokładnie na wynik porównywania zmian pozycji próbek w rankingach $\mathbf{r_1}$ i $\mathbf{r_2}$. Z tego względu, w celu porównania dwóch wartości funkcji przystosowania wymagane jest podanie wartości ϵ.

W strategii IRRG, odpowiednia wartość ϵ wyznaczana jest oddzielnie dla każdego porównania dwóch wartości funkcji przystosowania na podstawie mechanizmu opisanego w [126], który jest następujący. Rozważmy dwa wektory zmiennych decyzyjnych $\ddot{\mathbf{x}}_\mathbf{1}, \ddot{\mathbf{x}}_\mathbf{2} \in \mathcal{D}$ oraz funkcję przystosowania f. Niech λ będzie teoretyczną wartością modułu różnicy wartości $f(\ddot{\mathbf{x}}_\mathbf{1})$ i $f(\ddot{\mathbf{x}}_\mathbf{2})$, tj. $\lambda = |f(\ddot{\mathbf{x}}_\mathbf{1}) - f(\ddot{\mathbf{x}}_\mathbf{2})|$. Jako $\hat{\lambda}$ oznaczmy natomiast wynik tego samego działania matematycznego w reprezentacji zmiennoprzecinkowej używanej przez komputery, która obarczona jest błędem zaokrąglenia. Niech f_{fl} będzie funkcją, która przekształca liczby rzeczywiste w odpowiadające im wartości w reprezentacji zmiennoprzecinkowej. Zbiór liczb w reprezentacji zmiennoprzecinkowej oznaczmy jako $\mathbb{F}$, wtedy $f_{fl} : \mathbb{R} \to \mathbb{F}$. Relatywny błąd reprezentacji zmiennoprzecinkowej dla $x \in \mathbb{R}$ zdefiniowany jest następująco

$$\delta_{fl} = \frac{f_{fl}(x) - x}{x} \tag{3.8}$$

Na różnicę w wyniku $|\lambda - \hat{\lambda}|$, gdzie $\hat{\lambda} = |\hat{f}(\ddot{\mathbf{x}}_\mathbf{1}) \ominus \hat{f}(\ddot{\mathbf{x}}_\mathbf{2})|$, wpływają dwa czynniki.

– Skumulowany błąd zaokrąglenia wszystkich operacji arytmetycznych podczas wyliczania wartości funkcji f. Niech $\hat{f}$ będzie funkcją, która jest odpowiednikiem funkcji f w reprezentacji zmiennoprzecinkowej.

– Błąd zaokrąglenia związany z operatorem odejmowania zdefiniowanego dla liczb w reprezentacji zmiennoprzecinkowej. Operator ten został oznaczony symbolem $\ominus$.

Zgodnie ze standardem IEEE 754 [1], dla $x_1, x_2 \in \mathbb{R}$ zachodzi równość $x_1 \ominus x_2 = f_{fl}(x_1 - x_2)$. Wykorzystując wzór (3.8), możemy zapisać $\hat{\lambda}$ jako

$$\hat{\lambda} = |\hat{f}(\ddot{\mathbf{x}}_\mathbf{1}) \ominus \hat{f}(\ddot{\mathbf{x}}_\mathbf{2})| = |(\hat{f}(\ddot{\mathbf{x}}_\mathbf{1}) - \hat{f}(\ddot{\mathbf{x}}_\mathbf{2}))(1 + \delta_{fl,1})| \tag{3.9}$$

gdzie $\delta_{fl,1}$ jest relatywnym błędem reprezentacji zmiennoprzecinkowej dla wyniku działania $\hat{f}(\ddot{\mathbf{x}}_\mathbf{1}) - \hat{f}(\ddot{\mathbf{x}}_\mathbf{2})$. Na podstawie teorii opisanej w [20] możemy stwierdzić, że wyrażenie $1 + \delta_{fl,1}$ jest równoważne z wyrażeniem $1 + \theta_1$, gdzie $|\theta_k| \leq f_\gamma(k)$. Funkcja f_γ została opisana w rozdziale 2.1, a jej definicja została podana we wzorze (2.5).

Wartość k zależy natomiast od liczby przeprowadzonych operacji arytmetycznych. W tym przypadku jest to jedno odejmowanie, stąd $k = 1$. W ramach niniejszej pracy rozpatrujemy optymalizację problemów, na temat których nie posiadamy żadnej wiedzy dotyczącej ich wewnętrznej struktury. Z tego powodu możemy jedynie oszacować liczbę przeprowadzanych operacji arytmetycznych podczas wyliczania wartości funkcji przystosowania [94,126]. Autorzy prac [94,126] uważają, że rozsądnym założeniem jest liniowa zależność pomiędzy liczbą przeprowadzanych operacji arytmetycznych, a liczbą zmiennych rozważanego problemu optymalizacyjnego. Dodatkowo, Autor pracy [39] zakłada, że błąd zaokrąglenia rośnie proporcjonalnie do liczby przeprowadzonych operacji arytmetycznych zgodnie z funkcją pierwiastkową. Na podstawie tych dwóch założeń możemy zapisać, że $k \approx \sqrt{n}$, a w konsekwencji zdefiniować funkcję $\hat{f}$ jako

$$\hat{f}(\mathbf{x}) = f(\mathbf{x})(1 + \theta_{\sqrt{n}}) \tag{3.10}$$

Wzór (3.9) możemy następnie przekształcić, uwzględniając wzór (3.10), do postaci

$$\hat{\lambda} = |f(\ddot{\mathbf{x}}_\mathbf{1})(1 + \theta_{\sqrt{n}})(1 + \theta_1) - f(\ddot{\mathbf{x}}_\mathbf{2})(1 + \theta_{\sqrt{n}})(1 + \theta_1)| \tag{3.11}$$

gdzie zgodnie z teorią opisaną w [20], wyrażenie $(1+\theta_i)(1+\theta_j)$ jest tożsame wyrażeniu $1 + \theta_{i+j}$. Na tej podstawie otrzymujemy

$$\hat{\lambda} = |f(\ddot{\mathbf{x}}_\mathbf{1})(1 + \theta_{\sqrt{n}+1}) - f(\ddot{\mathbf{x}}_\mathbf{2})(1 + \theta_{\sqrt{n}+1})| \tag{3.12}$$

Następnie wykorzystujemy fakt, że $\forall_{x_1,x_2\in\mathbb{R}}|x_1 + x_2| \leq |x_1| + |x_2|$

$$\hat{\lambda} \leq |f(\ddot{\mathbf{x}}_\mathbf{1}) - f(\ddot{\mathbf{x}}_\mathbf{2})| + |f(\ddot{\mathbf{x}}_\mathbf{1})\theta_{\sqrt{n}+1} - f(\ddot{\mathbf{x}}_\mathbf{2})\theta_{\sqrt{n}+1}| \tag{3.13}$$

gdzie $|f(\ddot{\mathbf{x}}_\mathbf{1}) - f(\ddot{\mathbf{x}}_\mathbf{2})| = \lambda$, zatem

$$\hat{\lambda} \leq \lambda + |f(\ddot{\mathbf{x}}_\mathbf{1})\theta_{\sqrt{n}+1} - f(\ddot{\mathbf{x}}_\mathbf{2})\theta_{\sqrt{n}+1}| \tag{3.14}$$

W kolejnym kroku korzystamy z własności, że $|\sum_i^m x_i| \leq \sum_i^m |x_i|$ dla każdej dodatniej liczby całkowitej m, otrzymując

$$\hat{\lambda} \leq \lambda + \theta_{\sqrt{n}+1}(|f(\ddot{\mathbf{x}}_\mathbf{1})| + |f(\ddot{\mathbf{x}}_\mathbf{2})|) \tag{3.15}$$

Wartość $\theta_{\sqrt{n}+1}$ można ograniczyć z góry przez wartość $f_\gamma(\sqrt{n} + 1)$

$$\hat{\lambda} \leq \lambda + f_\gamma(\sqrt{n} + 1) \cdot (|f(\ddot{\mathbf{x}}_\mathbf{1})| + |f(\ddot{\mathbf{x}}_\mathbf{2})|) \tag{3.16}$$

Po przeniesieniu λ z prawej na lewą stronę nierówności otrzymujemy

$$\hat{\lambda} - \lambda \leq f_\gamma(\sqrt{n}+1) \cdot (|f(\ddot{\mathbf{x}}_\mathbf{1})| + |f(\ddot{\mathbf{x}}_\mathbf{2})|) \tag{3.17}$$

Z drugiej strony, przekształcając wzór (3.12) można wykorzystać fakt, że $\forall_{x_1,x_2\in\mathbb{R}}|x_1+x_2| \geq |x_1|-|x_2|$. Stosując rozumowanie podobne do powyższego, dochodzimy wtedy do nierówności

$$\lambda - \hat{\lambda} \leq f_\gamma(\sqrt{n}+1) \cdot (|f(\ddot{\mathbf{x}}_\mathbf{1})| + |f(\ddot{\mathbf{x}}_\mathbf{2})|) \tag{3.18}$$

Łącząc wzory (3.17) i (3.18) w jeden wzór otrzymujemy

$$|\lambda - \hat{\lambda}| \leq f_\gamma(\sqrt{n}+1) \cdot (|f(\ddot{\mathbf{x}}_\mathbf{1})| + |f(\ddot{\mathbf{x}}_\mathbf{2})|) \tag{3.19}$$

Wartość górnego ograniczenia wyrażenia $|\lambda-\hat{\lambda}|$ wykorzystywana jest następnie jako wartość ϵ (Pseudokod 3, linie 7 i 12). W celu ustalenia porządku między wartościami $f(\ddot{\mathbf{x}}_\mathbf{1})$ i $f(\ddot{\mathbf{x}}_\mathbf{2})$, używana jest zmodyfikowana wersja funkcji signum (linie 8 i 13), która uwzględnia wartość ϵ i jest zdefiniowana w następujący sposób

$$\operatorname{sgn}(x,\epsilon) = \begin{cases} -1, & \text{jeżeli } x < -\epsilon \\ 0, & \text{jeżeli } |x| \leq \epsilon \\ 1, & \text{jeżeli } x > \epsilon \end{cases} \tag{3.20}$$

Dobrej jakości rozwiązania rozważanego problemu optymalizacyjnego są zazwyczaj jego lokalnymi optimami lub znajdują się w ich pobliżu [15, 142]. W przypadku gdy problem nie jest w pełni separowalny, obserwacja dobrej jakości rozwiązań powinna prowadzić do wykrycia zależności pomiędzy zmiennymi częściej niż podczas analizy rozwiązań o gorszej jakości [105].W celu wykorzystania tego wniosku, analiza różnic rankingów $\mathbf{r_1}$ i $\mathbf{r_2}$ rozpoczyna się od próbek zajmujących najwyższe pozycje w rankingu $\mathbf{r_1}$. Jeżeli zachodzi interakcja pomiędzy rozpatrywanymi grupami zmiennych decyzyjnych to liczymy, że najlepsze próbki z rankingu $\mathbf{r_1}$ będą powiązane z innymi optimami lokalnymi niż najlepsze próbki z $\mathbf{r_2}$.

Strategia IRRG posiada mechanizmy, które zmniejszają prawdopodobieństwo pominięcia jakiejkolwiek z istniejących interakcji pomiędzy zmiennymi. Z drugiej strony, mechanizmy te mogą przynieść odwrotny skutek — zmniejszyć jakość zwracanej dekompozycji, gdyż zwiększają prawdopodobieństwo wykrycia nieistniejących zależności ze względu na niedoskonałość reprezentacji liczb zmiennoprzecinkowych. W celu zminimalizowania tego efektu ubocznego, decyzja o ewentualnej interakcji podejmowana jest tylko wtedy, gdy obie wartości zmodyfikowanej funkcji signum (wzór (3.20)) są różne od 0 (linie 8 i 13).

Podczas sprawdzania, czy rozłączne zbiory zmiennych decyzyjnych X_1 i X_2 oddziałują na siebie wzajemnie, rozpatrywanych jest n_s różnych wartości dla zmiennych ze zbioru X_1 i tylko dwie wartości dla zmiennych należących do zbioru X_2. Funkcja ZachodziInterakcja jest zatem asymetryczna, dlatego w ramach funkcji UwzględnijPojedynczeZmienne, dla każdej rozważanej pary zbiorów zmiennych decyzyjnych może ona zostać wywołana dwukrotnie (linie 10, 13, 17 i 20).

W rozdziale 3.1.3 opisany został wpływ ograniczeń będących częścią problemów optymalizacyjnych z ograniczeniami, np. ograniczenia kostkowe, na jakość dekompozycji zwracanej przez strategie bazujące na sprawdzaniu monotoniczności. Istnienie ograniczeń może mieć negatywny wpływ na tego typu strategie, ponieważ może uniemożliwić wykrycie istniejących zależności pomiędzy zmiennymi decyzyjnymi. W celu wyeliminowania tego zjawiska, jednym z wektorów zmiennych decyzyjnych używanym w trakcie wyszukiwania zależności jest $\mathbf{x_{hq}}$, który powinien być rozwiązaniem o dobrej jakości. Jeżeli zbiory X_1 i X_2 oddziałują na siebie wzajemnie, a $\mathbf{x_{hq}}$ znajduje się blisko lokalnego optimum to możliwe jest wystąpienie jednej z poniższych sytuacji.

1. Wektor $\mathbf{x_{hq}} + \delta_2 \mathbf{u_2}$, otrzymany po zmianie wartości zmiennych należących do zbioru X_2, nie znajduje się już dłużej blisko lokalnego optimum. Oznacza to, że zmieniły się przedziały monotoniczności w sąsiedztwie $\mathbf{x_{hq}}$ i rekursywne grupowanie różnicowe powinno wykryć zależność pomiędzy zbiorami X_1 i X_2.

2. Przedziały monotoniczności w sąsiedztwie $\mathbf{x_{hq}}$ nie zmieniły się, ale porządek wartości funkcji przystosowania wokół $\mathbf{x_{hq}}$ uległ zmianie. Rysunek 3.2b przedstawia przykład takiej sytuacji, w której rekursywne grupowanie różnicowe prawdopodobnie także wykryje interakcję pomiędzy dwoma rozważanymi zbiorami zmiennych decyzyjnych.

3. Zarówno przedziały monotoniczności, jak i porządek wartości funkcji przystosowania w sąsiedztwie $\mathbf{x_{hq}}$ pozostały bez zmian. W takim przypadku, rekursywne grupowanie różnicowe nie wykryje zależności, lecz taka sytuacja jest mało prawdopodobna gdy zbiory X_1 i X_2 oddziałują na siebie wzajemnie.

Liczba optimów lokalnych jest kolejnym czynnikiem, który wpływa na czułość rekursywnego grupowania różnicowego. Im większa ich liczba, tym więcej punktów o podobnych predyspozycjach do wykrywania interakcji jak $\mathbf{x_{hq}}$, które znajdują się w obszarach przyciągania różnych optimów lokalnych. Liczba optimów lokalnych rozważanego problemu optymalizacyjnego jest zatem odwrotnie proporcjonalna do prawdopodobieństwa pominięcia jakiejkolwiek z istniejących zależności.

Pseudokod 4 przedstawia kolejną funkcję pomocniczą. Jest nią funkcja SzukajInterakcji, której argumentami są m.in. dwa rozłączne zbiory, $\boldsymbol{G_1}$ i $\boldsymbol{G_2}$, składające się z grup, których elementami są zmienne oddziałujące na siebie wzajemnie. Głównym zadaniem funkcji SzukajInterakcji jest zidentyfikowanie wszystkich grup ze zbioru $\boldsymbol{G_2}$, dla których zachodzi bezpośrednia interakcja z przynajmniej jedną grupą będącą elementem zbioru $\boldsymbol{G_1}$. Wynikiem funkcji SzukajInterakcji jest zbiór $\boldsymbol{G_1^*}$, taki że $\boldsymbol{G_1} \subseteq \boldsymbol{G_1^*}$. Na samym początku, zbiór $\boldsymbol{G_1^*}$ jest tożsamy zbiorowi $\boldsymbol{G_1}$ (linia 2). Do sprawdzenia, czy istnieje bezpośrednia zależność pomiędzy przynajmniej jedną grupą z $\boldsymbol{G_1}$ i co najmniej jedną grupą należącą do $\boldsymbol{G_2}$, wykorzystywana jest funkcja ZachodziInterakcja. Funkcja ZachodziInterakcja operuje na zbiorach zmiennych decyzyjnych, a nie na zbiorach grup zmiennych decyzyjnych. Z tego powodu, zbiory $\boldsymbol{G_1}$ i $\boldsymbol{G_2}$ muszą zostać spłaszczone do zbiorów $X_1 = \bigcup \boldsymbol{G_1}$ i $X_2 = \bigcup \boldsymbol{G_2}$ (linie 3 i 4), dla których następnie wywoływana jest funkcja ZachodziInterakcja (linia 5). Jeżeli wykryta zostanie zależność to zbiór $\boldsymbol{G_2}$ dzielony jest na dwa równoliczne podzbiory $\boldsymbol{G_2^1}$ i $\boldsymbol{G_2^2}$ (linie 9 i 10). Wyjątkiem jest sytuacja, gdy zbiór $\boldsymbol{G_2}$ ma nieparzystą liczbę elementów. W takim wypadku, liczność zbiorów $\boldsymbol{G_2^1}$ i $\boldsymbol{G_2^2}$ różni się o 1. Funkcja ZachodziInterakcja jest następnie rekursywnie wywoływana dla obu podzbiorów (linie 11 i 12). Jeżeli po wykryciu interakcji pomiędzy X_1 i X_2 okaże się, że zbiór $\boldsymbol{G_2}$ jest jednoelementowy to ta pojedyncza grupa należąca do zbioru $\boldsymbol{G_2}$ zostaje dodana do zbioru $\boldsymbol{G_1^*}$ (linia 7). Podsumowując, funkcja SzukajInterakcji stara się połączyć zbiór $\boldsymbol{G_1}$ z wszystkimi grupami należącymi do $\boldsymbol{G_2}$, które bezpośrednio zależą od przynajmniej jednej grupy z $\boldsymbol{G_1}$. W wyniku tej operacji otrzymujemy jeden zbiór $\boldsymbol{G_1^*}$, który po spłaszczeniu $(\bigcup \boldsymbol{G_1^*})$ zawiera zmienne decyzyjne oddziałujące na siebie wzajemnie.

Wszystkie opisane w tym podrozdziale funkcje pomocnicze są wykorzystywane w ramach rekursywnego grupowania rankingowego. Pseudokod 5 przestawia jego ogólną procedurę. Przeprowadzenie skutecznego rekursywnego grupowania rankingowego mierzonego skutecznością otrzymanej dekompozycji wymaga by jeden z wejściowych wektorów zmiennych decyzyjnych był dobrej jakości. Powody tego wymagania zostały opisane podczas omawiania funkcji ZachodziInterakcja. W pierwszym kroku rekursywnego grupowania rankingowego, tworzone są grupy, które zawierają zmienne decyzyjne oddziałujące na siebie wzajemnie. Są one tworzone na podstawie macierzy interakcji Θ, która gromadzi wszystkie wykryte dotychczas zależności pomiędzy zmiennymi. Kolejność zmiennych w każdej grupie jest losowa by wyeliminować ewentualny wpływ konkretnego ciągu zmiennych na jakość uzyskanej dekompozycji (linie 5–11). Następnie budowana jest macierz $\bar{\boldsymbol{X}}_{\boldsymbol{1}}$, która zawiera n_s równomiernie wygenerowanych wartości dla każdej zmiennej decyzyjnej rozważa-

Pseudokod 4 Szukanie grup zmiennych decyzyjnych, które oddziałują na siebie wzajemnie, lecz nie zostały jeszcze do tej pory wykryte

wejście: $\boldsymbol{G_1}, \boldsymbol{G_2}$: rozłączne zbiory grup zmiennych zależnych, $\mathbf{x_{hq}}$: wysokiej jakości wektor zmiennych decyzyjnych, $\bar{\boldsymbol{X}}_\mathbf{1}$: macierz o wymiarach $n_s \times n$ zawierająca równomiernie wygenerowane wartości dla każdej zmiennej decyzyjnej, $\bar{\mathbf{x}}_\mathbf{2}$: wektor zmiennych decyzyjnych różny od $\mathbf{x_{hq}}$, $\bar{\mathbf{y}}_\mathbf{1}$: wartości funkcji f wyliczone na podstawie $\mathbf{x_{hq}}$ i $\bar{\boldsymbol{X}}_\mathbf{1}$, $\mathbf{r_1}$: ranking wartości będących elementami $\bar{\mathbf{y}}_\mathbf{1}$, f: problem optymalizacyjny, n_s: liczba próbek

wyjście: $\boldsymbol{G}_\mathbf{1}^*$: suma zbioru $\boldsymbol{G_1}$ i tych grup z $\boldsymbol{G_2}$, dla których wykryta została bezpośrednia interakcja z przynajmniej jedną grupą z $\boldsymbol{G_1}$

1: **funkcja** SzukajInterakcji($\boldsymbol{G_1}, \boldsymbol{G_2}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
2: $\quad\boldsymbol{G}_\mathbf{1}^* \leftarrow \boldsymbol{G_1}$
3: $\quad X_1 \leftarrow$ spłaszcz zbiór $\boldsymbol{G_1}$
4: $\quad X_2 \leftarrow$ spłaszcz zbiór $\boldsymbol{G_2}$
5: $\quad$**jeżeli** ZachodziInterakcja($X_1, X_2, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$) **to**
6: $\quad\quad$**jeżeli** $|\boldsymbol{G_2}| = 1$ **to**
7: $\quad\quad\quad$Dodaj pojedynczą grupę z $\boldsymbol{G_2}$ do $\boldsymbol{G}_\mathbf{1}^*$
8: $\quad\quad$**w przeciwnym wypadku**
9: $\quad\quad\quad\boldsymbol{G}_\mathbf{2}^\mathbf{1} \leftarrow$ pierwsza połowa zbioru $\boldsymbol{G_2}$
10: $\quad\quad\quad\boldsymbol{G}_\mathbf{2}^\mathbf{2} \leftarrow$ druga połowa zbioru $\boldsymbol{G_2}$
11: $\quad\quad\quad\boldsymbol{G}_\mathbf{1}^{\mathbf{1}*} \leftarrow$ SzukajInterakcji($\boldsymbol{G_1}, \boldsymbol{G}_\mathbf{1}^\mathbf{1}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
12: $\quad\quad\quad\boldsymbol{G}_\mathbf{1}^{\mathbf{2}*} \leftarrow$ SzukajInterakcji($\boldsymbol{G_1}, \boldsymbol{G}_\mathbf{2}^\mathbf{2}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
13: $\quad\quad\quad$Dodaj wszystkie nowo odkryte grupy z $\boldsymbol{G}_\mathbf{1}^{\mathbf{1}*}$ do $\boldsymbol{G}_\mathbf{1}^*$
14: $\quad\quad\quad$Dodaj wszystkie nowo odkryte grupy z $\boldsymbol{G}_\mathbf{1}^{\mathbf{2}*}$ do $\boldsymbol{G}_\mathbf{1}^*$
15: $\quad$**zwróć** $\boldsymbol{G}_\mathbf{1}^*$

nego problemu optymalizacyjnego (linie 12–15). Budowanie macierzy $\bar{\boldsymbol{X}}_{\mathbf{1}}$ odbywa się zgodnie z opisem przedstawionym w rozdziale 3.2. W kolejnym kroku, wywoływana jest funkcja UwzględnijPojedynczeZmienne w celu ustalenia, czy zmienne, dla których nie wykryto do tej pory żadnych zależności, powinny brać udział w kolejnych etapach rekursywnego grupowania rankingowego (linia 17). Po dodaniu wszystkich elementów do zbioru $\boldsymbol{G}$, zapewniamy jego losową kolejność by także konkretna sekwencja grup nie wpłynęła na jakość wynikowej dekompozycji. Zbiór $\boldsymbol{G}$ jest następnie dzielony na dwa podzbiory $\boldsymbol{G}_{\mathbf{1}}$ i $\boldsymbol{G}_{\mathbf{2}}$ (linie 18–20).

W dalszej części rekursywnego grupowania rankingowego, szukane są zależności pomiędzy grupami zależnych zmiennych należących do zbioru $\boldsymbol{G}_{\mathbf{1}}$, a grupami ze zbioru $\boldsymbol{G}_{\mathbf{2}}$ (linie 21–41). W tym celu tworzony jest ranking $\mathbf{r}_{\mathbf{1}}$ i wywoływana jest funkcja SzukajInterakcji. Jeżeli co najmniej jedna nowa interakcja zostanie wykryta to wykonywane są następujące operacje:

- dodanie nowo wykrytych grup, które bezpośrednio zależą od przynajmniej jednej grupy z $\boldsymbol{G}_{\mathbf{1}}$, do zbioru $\boldsymbol{G}_{\mathbf{1}}$ (linia 38),
- usunięcie tych grup ze zbioru $\boldsymbol{G}_{\mathbf{2}}$ (linia 39),
- ponowne wywołanie funkcji SzukajInterakcji, tym razem dla powiększonego zbioru $\boldsymbol{G}_{\mathbf{1}}$.

W przypadku, gdy żadne nowe zależności nie zostaną wykryte, ale w poprzedniej iteracji rekursywnego grupowania rankingowego zbiór $\boldsymbol{G}_{\mathbf{1}}$ został rozszerzony o co najmniej jedną nową grupę to:

- spłaszczony zbiór $\boldsymbol{G}_{\mathbf{1}}$ zostaje dodany jako pojedyncza grupa do wynikowego zbioru grup składających się ze zmiennych oddziałujących na siebie wzajemnie (linia 34),
- usuwane są wszystkie elementy ze zbioru $\boldsymbol{G}_{\mathbf{1}}$ i dodawana jest do niego pierwsza grupa będąca elementem zbioru $\boldsymbol{G}_{\mathbf{2}}$ (linia 35),
- ze zbioru $\boldsymbol{G}_{\mathbf{2}}$ usuwana jest grupa, która aktualnie stanowi pojedynczy element zbioru $\boldsymbol{G}_{\mathbf{1}}$ (linia 36).

Jeżeli jednak zbiór $\boldsymbol{G}_{\mathbf{1}}$ nie został nigdy rozszerzony o żadną nową grupę i jest jednoelementowy to zachodzi operacja losowego usuwania połowy zmiennych decyzyjnych z pojedynczej grupy należącej do zbioru $\boldsymbol{G}_{\mathbf{1}}$ (linie 27–29). Intuicja, która stoi za wykonywaniem tego kroku jest następująca. W przypadku gdy pojedyncza grupa należąca do zbioru $\boldsymbol{G}_{\mathbf{1}}$ składa się z wielu zmiennych decyzyjnych to pewna jej podgrupa

Pseudokod 5 Rekursywne grupowanie rankingowe

wejście: $\mathbf{x_{hq}}$: wysokiej jakości wektor zmiennych decyzyjnych, $\bar{\mathbf{x}}_\mathbf{2}$: wektor zmiennych decyzyjnych różny od $\mathbf{x_{hq}}$, Θ: macierz interakcji, f: problem optymalizacyjny, n: liczba wymiarów problemu f, **lb**: minimalne wartości dla każdej zmiennej problemu f, **ub**: maksymalne wartości dla każdej zmiennej problemu f, n_s: liczba próbek
wyjście: ***Niesep***: grupy składające się ze zmiennych decyzyjnych, które oddziałują na siebie wzajemnie

funkcja RRG($\mathbf{x_{hq}}, \bar{\mathbf{x}}_\mathbf{2}, \Theta, f, n, \mathbf{lb}, \mathbf{ub}, n_s$)
  Niesep $\leftarrow \varnothing$
  $\boldsymbol{G} \leftarrow \varnothing$ ▷ grupy zależnych zmiennych służące do wykrycia nowych interakcji
  $V \leftarrow$ zbiór zmiennych decyzyjnych od x_1 do x_n
  dla $v \in V$ **wykonaj**
    $Inter \leftarrow$ znajdź zmienne, które zależą od zmiennej v na podstawie Θ
    Dodaj zmienną v do zbioru $Inter$
    jeżeli $|Inters| > 1$ **to**
      $V \leftarrow V - Inter$ ▷ różnica zbiorów
      $Inter \leftarrow$ przetasuj $Inter$
      Dodaj $Inter$ do $\boldsymbol{G}$ jako grupę zależnych od siebie zmiennych
  dla $i \leftarrow 1$ **do** n **wykonaj** ▷ budowanie macierzy $n \times n_s$ zawierającej próbki
    $\bar{\boldsymbol{X}}_\mathbf{1}^\intercal[i] \leftarrow n_s$ równomiernie wygenerowanych wartości ze zbioru $[\mathbf{lb}[i], \mathbf{ub}[i]]$
    $\bar{\boldsymbol{X}}_\mathbf{1}^\intercal[i] \leftarrow$ przetasuj $\bar{\boldsymbol{X}}_\mathbf{1}^\intercal[i]$
  $\bar{\boldsymbol{X}}_\mathbf{1} \leftarrow$ transponuj macierz $\bar{\boldsymbol{X}}_\mathbf{1}^\intercal$ ▷ macierz $n_s \times n$ zawierająca próbki
  jeżeli UwzględnijPojedynczeZmienne($V, \boldsymbol{G}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, f, n_s$) **to**
    Dodaj wszystkie zmienne ze zbioru V do zbioru $\boldsymbol{G}$ jako pojedyncze grupy
  $\boldsymbol{G} \leftarrow$ przetasuj kolejność grup w $\boldsymbol{G}$
  $\boldsymbol{G_1} \leftarrow$ weź pierwszą grupę z $\boldsymbol{G}$
  $\boldsymbol{G_2} \leftarrow$ weź pozostałe grupy z $\boldsymbol{G}$
  dopóki $|\boldsymbol{G_2}| > 0$ **wykonaj**
    $X_1 \leftarrow$ spłaszcz zbiór $\boldsymbol{G_1}$
    $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}) \leftarrow$ UtwórzPierwszyRanking($X_1, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
    $\boldsymbol{G}_\mathbf{1}^* \leftarrow$ SzukajInterakcji($\boldsymbol{G_1}, \boldsymbol{G_2}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
    jeżeli $|\boldsymbol{G_1}| = |\boldsymbol{G}_\mathbf{1}^*|$ **to**
      jeżeli $|\boldsymbol{G_1}| = 1$ **to**
        $minRozmiar = \min_i |\boldsymbol{G_2}[i]|$
        jeżeli $|\boldsymbol{G_1}[1]| \geq \max\{minRozmiar, 2\}$ **to**
          Usuń połowę zmiennych z $\boldsymbol{G_1}[1]$
        w przeciwnym wypadku
          $\boldsymbol{G_1} \leftarrow$ weź pierwszą grupę z $\boldsymbol{G_2}$
          $\boldsymbol{G_2} \leftarrow \boldsymbol{G_2}$ wyłączając pierwszą grupę
      w przeciwnym wypadku
        Dodaj spłaszczony zbiór $\boldsymbol{G_1}$ do zbioru ***Niesep***
        $\boldsymbol{G_1} \leftarrow$ weź pierwszą grupę z $\boldsymbol{G_2}$
        $\boldsymbol{G_2} \leftarrow \boldsymbol{G_2}$ wyłączając pierwszą grupę
    w przeciwnym wypadku
      $\boldsymbol{G_1} \leftarrow \boldsymbol{G}_\mathbf{1}^*$
      $\boldsymbol{G_2} \leftarrow \boldsymbol{G_2}$ wyłączając grupy ze zbioru $\boldsymbol{G_1}$
  jeżeli $|\boldsymbol{G_1}| > 1$ **to**
    Dodaj spłaszczony zbiór $\boldsymbol{G_1}$ do zbioru ***Niesep***
  zwróć ***Niesep***

może na tyle mocno wpływać na kolejność w rankingach $\mathbf{r_1}$ i $\mathbf{r_2}$, że wpływ zmiennych należących do zbioru $\bigcup \boldsymbol{G_2}$ stanie się pomijalny, przy założeniu, że istnieje przynajmniej jedna zmienna w zbiorze $\bigcup \boldsymbol{G_1}$, dla której zachodzi interakcja z co najmniej jedną zmienną ze zbioru $\bigcup \boldsymbol{G_2}$. Jeżeli usuniemy losowo część zmiennych z pojedynczej grupy należącej do $\boldsymbol{G_1}$ to zwiększamy szansę, że wzrośnie wpływ części ze zmiennych ze zbioru $\bigcup \boldsymbol{G_2}$ na kolejność w rankingach $\mathbf{r_1}$ i $\mathbf{r_2}$, co może skutkować wykryciem interakcji pomiędzy zmiennymi, które wcześniej zostały pominięte. Opisana powyżej sytuacja często ma miejsce gdy większość zależności została już wykryta. Rozsądnym wydaje się zatem by zmniejszać rozmiar pojedynczej grupy należącej do zbioru $\boldsymbol{G_1}$ tylko wtedy, gdy jej rozmiar nie jest mniejszy niż rozmiar najmniejszej grupy należącej do zbioru $\boldsymbol{G_2}$.

3.4 Inkrementacyjne grupowanie

Strategia IRRG, podczas swojego działania, inkrementacyjnie buduje macierz interakcji Θ [77], której własności zostały szczegółowo opisane w rozdziale 1.1.1. Pseudokod 6 przedstawia ogólną zasadę działania strategii IRRG, której pierwszym krokiem jest znalezienie wysokiej jakości rozwiązania $\mathbf{x_{hq}}$ za pomocą metody optymalizacji podanej przez użytkownika (linia 2). Po utworzeniu macierzy interakcji Θ, która początkowo nie zawiera żadnych informacji na temat zależności pomiędzy zmiennymi (linia 3), uruchamiane jest po raz pierwszy rekursywne grupowanie rankingowe (funkcja RRG), które zwraca zbiór grup składających się ze zmiennych decyzyjnych oddziałujących na siebie wzajemnie o nazwie ***NiesepTym*** (linia 9). Zbiór ten używany jest następnie do aktualizacji macierzy interakcji (linia 10). Podczas aktualizacji macierzy Θ, każda para zmiennych należąca do tej samej grupy oznaczana jest jako oddziałująca na siebie wzajemnie. Dodatkowo, każda zmiana wartości pojedynczego elementu macierzy interakcji z 0 na 1 powoduje zmiany wartości innych elementów macierzy Θ, takich które zapewnią jej symetrię i tranzytywność. W przypadku problemów optymalizacyjnych, które nie są zbudowane z nakładających się na siebie podproblemów, zapewnienie tranzytywności macierzy interakcji jest efektem porządanym, gdyż prowadzi do oznaczenia nowych bezpośrednich zależności pomiędzy zmiennymi decyzyjnymi bez zużycia żadnych dodatkowych wyliczeń funkcji przystosowania. Z drugiej strony, gdy istnieją podproblemy, które nakładają się na siebie to analizując macierz interakcji nie będziemy w stanie stwierdzić, czy oznaczona interakcja pomiędzy zmiennymi jest bezpośrednia czy pośrednia. Strategia IRRG, podobnie jak strategia RDG (Rysunek 2.3c), ma tendencję do grupowania wszystkich zmiennych, które należą do nakładających się na siebie podproblemów

w jedną grupę.

Wywołanie funkcji RRG oraz aktualizacja macierzy interakcji są powtarzane, aż do momentu spełnienia warunku zatrzymania strategii IRRG. Działanie strategii IRRG jest przerywane gdy jeden z dwóch warunków zostanie spełniony.

– Kolejne ϵ_{sti} wywołania funkcji RRG nie prowadziły do wykrycia nowych zależności pomiędzy zmiennymi decyzyjnymi. Parametr ϵ_{sti} podawany jest przez użytkownika.

– Pierwsze wywołanie funkcji RRG nie zwróciło żadnej grupy składającej się ze zmiennych, które oddziałują na siebie wzajemnie. Warunek ten został wprowadzony by zaoszczędzić jak najwięcej wyliczeń funkcji przystosowania. Skoro największa szansa na znalezienie nowych interakcji pomiędzy zmiennymi decyzyjnymi jest wtedy, gdy macierz Θ jest macierzą jednostkową to niepowtarzanie wywołania funkcji RRG ani razu więcej wydaje się rozsądne, gdy żadne zależności pomiędzy zmiennymi nie zostały wykryte podczas pierwszego uruchomienia rekursywnego grupowania rankingowego.

Wynikiem działania strategii IRRG są dwa zbiory składające się z grup, których elementami są zmienne decyzyjne. Zbiory te przechowują grupy o różnym charakterze. W zbiorze ***Niesep*** znajdują się grupy, których zmienne oddziałują na siebie wzajemnie, natomiast zbiór ***Sep*** składa się z grup, których elementami są separowalne zmienne decyzyjne. Grupowanie separowalnych zmiennych w grupy o rozmiarze sterowanym przez podawany przez użytkownika parametr ϵ_s (linie 27–33) zostało zapożyczone ze strategii RDG3 [125], która opisana została w rozdziale 2.2.

3.5 Parametry strategii IRRG

W opisie działania strategii IRRG wyszczególnione zostały cztery parametry, których wartości muszą zostać podane przez użytkownika. Należą do nich:

– metoda optymalizacji używana w trakcie początkowego procesu optymalizacji,

– liczba próbek n_s,

– maksymalna liczba kolejnych iteracji strategii IRRG bez wykrycia nowych interakcji pomiędzy zmiennymi decyzyjnymi ϵ_{sti},

– preferowany rozmiar grup zawierających separowalne zmienne ϵ_s.

Pseudokod 6 Strategia IRRG

wejście: f: problem optymalizacyjny, n: liczba wymiarów problemu f, **lb**: minimalne wartości dla każdej zmiennej problemu f, **ub**: maksymalne wartości dla każdej zmiennej problemu f, *opt*: metoda optymalizacji używana w trakcie początkowego procesu optymalizacji, n_s: liczba próbek, ϵ_s: preferowany rozmiar grup zawierających separowalne zmienne, ϵ_{sti}: maksymalna liczba kolejnych iteracji strategii IRRG bez wykrycia nowych interakcji pomiędzy zmiennymi decyzyjnymi
wyjście: ***Sep***: grupy separowalnych zmiennych decyzyjnych, ***Niesep***: grupy składające się ze zmiennych decyzyjnych, które oddziałują na siebie wzajemnie

1: $\boldsymbol{Sep}, \boldsymbol{Niesep} \leftarrow \varnothing$
2: $\mathbf{x_{hq}} \leftarrow$ optymalizuj problem f używając *opt*
3: $\Theta \leftarrow$ macierz jednostkowa o wymiarach $n \times n$ ▷ macierz interakcji
4: $pierwszaIteracja \leftarrow tak$
5: $cnt_{sti} \leftarrow 0$ ▷ licznik kolejnych iteracji bez wykrycia żadnych nowych interakcji
6: $przerwij \leftarrow nie$
7: **dopóki nie** $przerwij$ **wykonaj**
8: $\quad \mathbf{x_{lq}} \leftarrow$ utwórz losowe rozwiązanie
9: $\quad \boldsymbol{NiesepTym} \leftarrow \mathrm{RRG}(\mathbf{x_{hq}}, \mathbf{x_{lq}}, \Theta, f, n, \mathbf{lb}, \mathbf{ub}, n_s)$
10: $\quad$ Zaktualizuj macierz Θ uwzględniając grupy ze zbioru ***NiesepTym***
11: $\quad$ **jeżeli** żadna nowa interakcja nie została wykryta **to** ▷ zobacz linię 10
12: $\quad\quad cnt_{sti} \leftarrow cnt_{sti} + 1$
13: $\quad\quad przerwij \leftarrow pierwszaIteracja$ **lub** $cnt_{sti} = \epsilon_{sti}$
14: $\quad$ **w przeciwnym wypadku**
15: $\quad\quad cnt_{sti} \leftarrow 0$
16: $\quad pierwszaIteracja \leftarrow nie$
17: $V \leftarrow$ zbiór zmiennych decyzyjnych od x_1 do x_n
18: $S \leftarrow \varnothing$ ▷ zbiór separowalnych zmiennych decyzyjnych przed ich grupowaniem
19: **dla** $v \in V$ **wykonaj**
20: $\quad Inter \leftarrow$ znajdź zmienne, które zależą od zmiennej v na podstawie Θ
21: $\quad$ Dodaj zmienną v do zbioru $Inter$
22: $\quad$ **jeżeli** $|Inter| = 1$ **to**
23: $\quad\quad$ Dodaj zmienną v do zbioru S
24: $\quad$ **w przeciwnym wypadku**
25: $\quad\quad$ Dodaj zbiór $Inter$ do zbioru ***Niesep*** jako grupę zmiennych od siebie zależnych
26: $\quad V \leftarrow V - Inter$ ▷ różnica zbiorów
27: **dopóki** $|S| > 0$ **wykonaj**
28: $\quad$ **jeżeli** $|S| < \epsilon_s$ **to**
29: $\quad\quad$ Dodaj zbiór S do zbioru ***Sep*** jako grupę separowalnych zmiennych
30: $\quad\quad S \leftarrow \varnothing$
31: $\quad$ **w przeciwnym wypadku**
32: $\quad\quad$ Dodaj ϵ_s zmiennych ze zbioru S do ***Sep*** jako grupę separowalnych zmiennych
33: $\quad\quad$ Usuń ϵ_s zmiennych ze zbioru S
34: **zwróć** $(\boldsymbol{Sep}, \boldsymbol{Niesep})$

Celem początkowego procesu optymalizacji jest znalezienie rozwiązania o wysokiej jakości $\mathbf{x_{hq}}$ (znajdującego się w pobliżu jednego z optimów lokalnych). W przypadku wielowymiarowych problemów optymalizacyjnych, w szczególności w gdy spodziewamy się problemu nie w pełni separowalnego, autor niniejszej rozprawy rekomenduje podzielenie początkowego procesu optymalizacji na dwa etapy. W pierwszym z nich należy skupić się na eksploracji przestrzeni przeszukiwań, natomiast w drugim na eksploracji znalezionego w pierwszym etapie obiecującego regionu. W przeciwnym wypadku, wysokiej jakości rozwiązania mogą nie zostać znalezione dla niektórych z podproblemów. Należy również wspomnieć, że początkowy proces optymalizacji nie musi zawsze prowadzić do osiągnięcia przez strategię IRRG dekompozycji o lepszej jakości. Z tego powodu, koszt początkowego procesu optymalizacji nie powinien być zbyt wysoki. W niniejszej pracy proponujemy aby budżet ten stanowił wartość bliską $2 \cdot n_s \cdot n$ wyliczeń funkcji przystosowania, co odpowiada najmniejszej możliwej złożoności obliczeniowej strategii IRRG [57].

Wpływ wartości parametrów n_s i ϵ_{sti} na jakość dekompozycji jest następujący. Im większa ich wartość, tym większa szansa na otrzymanie dekompozycji o wyższej jakości. Z drugiej strony, większe wartości tych dwóch parametrów powodują większą liczbę wyliczeń funkcji przystosowania. Dodatkowo, najprawdopodobniej dla każdego problemu optymalizacyjnego istnieją takie wartości parametrów n_s i ϵ_{sti}, że ich zwiększenie nie powoduje już polepszenia jakości wynikowej dekompozycji. Rośnie natomiast koszt uruchomienia strategii IRRG.

Ostatnim parametrem strategii IRRG jest preferowany rozmiar grup zawierających separowalne zmienne (ϵ_s). Parametr ten został zapożyczony ze strategii RDG3 [125]. Dyskusję na jego temat można znaleźć w rozdziale 2.2.

3.6 Porównanie z innymi strategiami bazującymi na sprawdzaniu monotoniczności

Najbardziej zbliżoną, spośród omawianych w niniejszej pracy strategii, strategią bazującą na sprawdzaniu monotoniczności do strategii IRRG jest strategia FVIL. Obie strategie należą do grupy rekursywnych strategii dekompozycji i decyzję o ewentualnej bezpośredniej interakcji pomiędzy dwoma grupami zmiennych decyzyjnych, X_1 i X_2, podejmują na podstawie wzoru (2.11). Jeżeli interakcja zostanie wykryta to grupa X_2, w obu strategiach, jest dzielona na dwie podgrupy X_2^1 i X_2^2, których rozmiar powinien być taki sam. Wyjątkiem jest sytuacja, w której grupa X_2 posiada nieparzystą liczbę elementów. Wówczas jedna z podgrup będzie posiadać o jeden element więcej. Każda z podgrup jest następnie sprawdzana pod kątem

istnienia bezpośredniej zależności od X_1. W przypadku strategii FVIL, podczas każdej próby wykrycia zależności pomiędzy dwiema grupami zmiennych decyzyjnych, próbka (δ_1, δ_2, $\mathbf{u_1}$, $\mathbf{u_2}$, $\ddot{\mathbf{x}}$) tworzona jest w losowy sposób. Załóżmy, że grupy X_1 i X_2 oddziałują na siebie wzajemnie, ponieważ zmienne $x_p \in X_1$ i $x_q \in X_2$ są od siebie zależne i strategia FVIL wykryła tę zależność poprawnie. Po podziale grupy X_2 na X_2^1 i X_2^2, niech zmienna x_q będzie elementem podgrupy X_2^1. Należy zauważyć, że tym razem strategia FVIL może nie wykryć interakcji pomiędzy X_1 i X_2^1, ponieważ inne próbki mogą zostać wzięte pod uwagę ze względu na ich losową generację. Z tego powodu, w strategii IRRG zrezygnowano z losowej generacji próbek za każdym razem. W przypadku wykrycia interakcji pomiędzy X_1 i X_2 przez strategię IRRG, analizowane wartości zmiennych należących do X_1, X_2^1 i X_2^2 będą takie same jak w poprzednim kroku, tj. macierz $\bar{\boldsymbol{X}}_\mathbf{1}$, wektor $\mathbf{x_{hq}}$ oraz wektor $\bar{\mathbf{x}}_\mathbf{2}$ pozostaną bez zmian. Dzięki powtórnemu przypisywaniu tych samych wartości do części ze zmiennych, wyliczone do tej pory wartości funkcji przystosowania mogą zostać użyte w późniejszych, rekursywnych, krokach.

Kolejną różnicą w sposobie działania strategii FVIL i IRRG, która wpływa na jakość zwracanej przez obie strategie dekompozycji, jest liczba elementów grupy X_1 w procesie szukania interakcji. W strategii FVIL, X_1 jest zawsze grupą jednoelementową, natomiast strategia IRRG dopuszcza by grupa ta zawierała więcej niż jeden element. Jeżeli podczas przetwarzania grupy X_1 przez strategię IRRG wykryte zostaną zmienne, które bezpośrednio zależą od przynajmniej jednej zmiennej będącej elementem grupy X_1 to dalszy proces szukania interakcji zostanie przeprowadzony po dodaniu tych zmiennych do X_1. We wszystkich poza pierwszą iteracją strategii IRRG, rozważane mogą być wielowymiarowe grupy X_1, ponieważ uwzględniają one wszystkie dotychczas wykryte interakcje pomiędzy zmiennymi. Szukanie niewykrytych dotychczas interakcji dla wieloelementowych grup X_1 zmniejsza szansę na niewykrycie zależności pomiędzy zmiennymi, które w rzeczywistości istnieją.

Pierwszą operacją wykonywaną w ramach strategii IRRG jest uruchomienie podanej przez użytkownika metody optymalizacji w celu znalezienia wysokiej jakości rozwiązania ($\mathbf{x_{hq}}$) rozważanego problemu optymalizacyjnego. W rozdziale 3.3 wskazano, że mogą istnieć problemy optymalizacyjne, dla których wykorzystanie takiego rozwiązania w szukaniu interakcji pomiędzy grupami zmiennych decyzyjnych zwiększa szanse na wykrycie wszystkich istniejących zależności. Istnieją inne strategie dekompozycji, które także podczas swojego działania korzystają z metody optymalizacji [14,143]. Przykładem takiej strategii jest strategia CCVIL [14]. Podczas pojedynczej iteracji strategii CCVIL, wszystkie zmienne są pojedynczo optymalizowane (w losowej kolejności). W tym celu wykorzystywane są populacyjne metody opty-

malizacji. Uruchamiane są one jednak tylko dla trzech osobników i tylko na jedną iterację. Możemy zatem oczekiwać, że wynikiem takiego procesu optymalizacji będzie wektor, którego jakość będzie znacząco gorsza od rozwiązania $\mathbf{x_{hq}}$ zwracanego przez pierwszą operację strategii IRRG.

3.7 Złożoność obliczeniowa

Podobnie jak RDG [123], RDG2 [126], RDG3 [125] i FVIL [29], strategia IRRG jest rekursywną strategią dekompozycji. Typowa złożoność obliczeniowa dla tej klasy strategii, mierzona liczbą wyliczeń funkcji przystosowania, to $\mathcal{O}(n\log(n))$, gdzie n jest liczbą wymiarów rozważanego problemu optymalizacyjnego. W niniejszym podrozdziale, przedstawiona zostanie szczegółowa analiza złożoności obliczeniowej poszczególnych etapów strategii IRRG.

W początkowej fazie strategii IRRG uruchamiany jest proces optymalizacji w celu znalezienia rozwiązania o wysokiej jakości ($\mathbf{x_{hq}}$). Zakładając, że rozpatrywany problem optymalizacyjny jest wielowymiarowy to liczymy, że posiada on wiele optimów lokalnych i znalezienie jedno z nich nie powinno być kosztownym procesem. Dodatkowo, użytkownik może wybrać dowolną metodę optymalizacji wraz z dowolnym warunkiem zatrzymania. Autor niniejszej pracy rekomenduje by na początkową optymalizację przeznaczyć stosunkowo niską liczbę wyliczeń funkcji przystosowania, która nie będzie znaczącym czynnikiem w kontekście całościowego kosztu strategii IRRG. Z powyższych powodów, w analizie złożoności obliczeniowej strategii IRRG proces szukania rozwiązania $\mathbf{x_{hq}}$ zostanie pominięty.

Analiza złożoności obliczeniowej strategii IRRG bazuje na analizie złożoności strategii RDG [123], która została przedstawiona w rozdziale 2.2. Koszt strategii IRRG, mierzony liczbą wyliczeń funkcji przystosowania, związany jest z dwoma operacjami. Utworzenie rankingu $\mathbf{r_1}$ wymaga n_s wyliczeń funkcji przystosowania, natomiast koszt funkcji ZachodziInterakcja to maksymalnie n_s wyliczeń funkcji przystosowania. Każde wywołanie funkcji SzukajInterakcji składa się m.in. z jednego wywołania funkcji ZachodziInterakcja, zatem koszt funkcji SzukajInterakcji to również maksymalnie n_s wyliczeń funkcji przystosowania. W zależności od rodzaju rozważanego problemu optymalizacyjnego, analiza pierwszej iteracji strategii IRRG jest następująca.

1. Problemy w pełni separowalne: dla każdej zmiennej decyzyjnej zostanie utworzony ranking $\mathbf{r_1}$ oraz dokładnie raz wywołana zostanie funkcja SzukajInterakcji. Całkowity koszt pierwszej iteracji strategii IRRG to zatem $n_s{\cdot}n{+}n_s{\cdot}n$.

2. Problemy w pełni nieseparowalne: rekursywne dzielenie zbioru $\boldsymbol{G_2}$ na pół powoduje, że funkcja SzukajInterakcji wywoływana jest około $\sum_{i=0}^{k} \frac{n}{2^i}$ razy, gdzie $k = \log_2(n)$. Zgodnie ze wzorem (2.8), wartość $\sum_{i=0}^{k}$ można ograniczyć z góry przez $2n$. Koszt wszystkich wywołań funkcji SzukajInterakcji jest zatem mniejszy niż $2 \cdot n_s \cdot n$. Dodatkowo, jeden raz musi zostać utworzony ranking $\mathbf{r_1}$. Możemy zatem stwierdzić, że koszt pierwszej iteracji strategii IRRG dla problemów w pełni nieseparowalnych to około $2 \cdot n_s \cdot n + n_s$ wyliczeń funkcji przystosowania.

3. Problemy nie w pełni separowalne składające się z m, w pełni nieseparowalnych, podproblemów o rozmiarze $l = \frac{n}{m}$: pogrupowanie l nieseparowalnych zmiennych decyzyjnych wymaga mniej niż $2 \cdot l \cdot \log_2(n)$ wywołań funkcji SzukajInterakcji i jednokrotnego utworzenia rankingu $\mathbf{r_1}$. Skoro takich grup jest m to całkowity koszt pierwszej iteracji strategii IRRG to maksymalnie $2 \cdot n_s \cdot n \cdot \log_2(n) + n_s \cdot m$ wyliczeń funkcji przystosowania.

4. Problemy nie w pełni separowalne z jednym, w pełni nieseparowalnym podproblemem o rozmiarze l: przetworzenie $n - l$ separowalnych zmiennych decyzyjnych to koszt $n_s \cdot (n - l) + n_s \cdot (n - l)$ wyliczeń funkcji przystosowania. Pogrupowanie pozostałych l zmiennych decyzyjnych wymaga natomiast maksymalnie $2 \cdot n_s \cdot l \cdot \log_2(n) + n_s$ wyliczeń funkcji przystosowania. Możemy zatem stwierdzić, że maksymalny koszt pierwszej iteracji strategii IRRG dla takich problemów to $n_s \cdot (n - l) + 2 \cdot n_s \cdot l \cdot \log_2(n) + n_s \cdot (n - l + 1)$ wyliczeń funkcji przystosowania.

5. Funkcja Rosenbrocka [70]: jest to problem z nakładającymi się na siebie podproblemami o rozmiarze 3. Graf interakcji ośmiowymiarowej funkcji Rosenbrocka został zaprezentowany na rysunku Rysunek 1.4. Zaczynając od zmiennej x_i potrzebujemy około $2 \cdot 2 \cdot \log_2(n)$ wywołań funkcji SzukajInterakcji i utworzenia jednego rankingu $\mathbf{r_1}$ by wykryć zależność pomiędzy x_i, a x_{i-1} oraz x_{i+1}. Daje to razem około $n_s \cdot 2 \cdot 2 \cdot \log_2(n) + n_s$ wyliczeń funkcji przystosowania. Do pogrupowania wszystkich zmiennych w jedną grupę, potrzebnych jest $\frac{n}{2}$ takich kroków. Prowadzi to zatem to całkowitego kosztu zbliżonego do $2 \cdot n_s \cdot n \log_2(n) + n_s \cdot \frac{n}{2}$ wyliczeń funkcji przystosowania.

Podobna analiza może zostać przeprowadzona dla kolejnych iteracji strategii IRRG. W tym celu wprowadźmy następujące oznaczenia. Niech n_V oznacza liczbę zmiennych, dla których nie znaleziono dotychczas żadnych zmiennych zależnych. Liczbę wykrytych do tej pory grup zmiennych zależnych oznaczmy natomiast symbolem n_G. Druga iteracja strategii IRRG wykona się jedynie wtedy, gdy rozważany

problem optymalizacyjny nie jest problemem w pełni separowalnym. Możemy zatem stwierdzić, że $n_V + n_G < n$. Zakładając, że w pierwszej iteracji strategii IRRG wszystkie interakcje zostały wykryte to koszt związany z utworzeniem rankingu $\mathbf{r_1}$ i wywołaniem funkcji SzukajInterakcji wynosi około $n_s \cdot (n_V + n_G) + n_s \cdot (n_V + n_G)$, ponieważ żadna nowa interakcja nie zostanie już wykryta. Z tego powodu potrzebujemy raz utworzyć ranking $\mathbf{r_1}$ i raz wywołać funkcję SzukajInterakcji dla każdej separowalnej zmiennej i każdej grupy zawierającej zmienne, które oddziałują na siebie wzajemnie. Jeżeli jednak nie zostaną wykryte wszystkie zależności to koszt związany z utworzeniem rankingu $\mathbf{r_1}$ i wywołaniem funkcji SzukajInterakcji będzie i tak niższy niż w przypadku pierwszej iteracji strategii IRRG ze względu na fakt, że $n_V + n_G < n$.

Z drugiej strony, podczas każdej kolejnej iteracji strategii IRRG dochodzi koszt związany z usuwaniem połowy zmiennych z pojedynczej grupy należącej do $\boldsymbol{G}_{\mathbb{1}}$ (Pseudokod 5, linia 29) oraz koszt wywołania funkcji UwzględnijPojedynczeZmienne. Najgorszy przypadek, pod względem kosztu mierzonego liczbą wyliczeń funkcji przystosowania, dla usuwania połowy zmiennych jest wtedy, gdy $|\bigcup \boldsymbol{G_1}| = n-1 \wedge |\bigcup \boldsymbol{G_2}| = 1$. Maksymalna liczba takich operacji usuwania to około $\log_2(n-1)$. Zakładając, że każda pojedyncza operacja nie przyniesie wykrycia nowych interakcji to jej koszt nie przekroczy $n_s + n_s$ wyliczeń funkcji przystosowania. Utworzenie rankingu $\mathbf{r_1}$ wymaga n_s wyliczeń funkcji przystosowania. Taki sam koszt związany jest także z wywołaniem funkcji SzukajInterakcji, dla której nie będzie żadnych wywołań rekursywnych.

Najgorszym przypadkiem dla funkcji UwzględnijPojedynczeZmienne jest sytuacja, w której podwójnie tworzony jest ranking $\mathbf{r_1}$ oraz podwójnie wywoływana jest funkcja ZachodziInterakcja dla pojedynczych zmiennych i dla każdej grupy składającej się ze zmiennych, które oddziałują na siebie wzajemnie. Możemy zatem stwierdzić, że maksymalny koszt wywołania funkcji UwzględnijPojedynczeZmienne to $2 \cdot n_s + 2 \cdot n_s + 2 \cdot n_s \cdot n_G + 2 \cdot n_s \cdot n_G$ wyliczeń funkcji przystosowania.

Podsumowując powyższą analizę złożoności obliczeniowej strategii IRRG, jeżeli $n_s \ll n$ i liczba iteracji strategii IRRG jest dużo mniejsza od n to złożoność obliczeniowa strategii IRRG wynosi $\mathcal{O}(n \log(n))$. Mogą się zdarzyć jednak przypadki, dla których założenie, że liczba iteracji strategii IRRG jest dużo mniejsza od n jest nieprawdziwe. Jedną z sytuacji, w której strategia IRRG wykonuje sporo iteracji jest następujący proces dekompozycji n-wymiarowego problemu optymalizacyjnego, który składa się z $\frac{n}{2}$ separowalnych podproblemów, gdzie każdy podproblem posiada dwie zmienne decyzyjne. Podczas pierwszej iteracji, strategia IRRG wykrywa jedną interakcję, a następnie w każdej co ϵ_{sti}-tej iteracji wykrywane są kolejne pojedyncze

interakcje. Całkowita liczba iteracji strategii IRRG wyniesie wówczas $1+\epsilon_{sti} \cdot \frac{n}{2}$. Złożoność obliczeniowa wykrycia pojedynczej interakcji wynosi $\mathcal{O}(n)$, zatem złożoność obliczeniowa najgorszego przypadku dla strategii IRRG to $\mathcal{O}(n^2)$.

Rozdział 4

Problemy testowe

W niniejszym rozdziale porównane zostaną ze sobą strategie IRRG i RDG3 pod względem jakości otrzymanej dekompozycji oraz kosztu, który został poniesiony podczas dekompozycji zadanego problemu optymalizacyjnego. Proces dekompozycji problemu to jedynie element całego procesu optymalizacji. Z tego powodu porównane zostaną także metody optymalizacji, w tym architektury koewolucji kooperatywnej po osadzeniu w nich obu rozpatrywanych strategii dekompozycji.

Wszystkie przedstawione, w niniejszym rozdziale, porównania bazują na wynikach eksperymentów, które zostały przeprowadzone dla testowych problemy optymalizacyjnych. Testowe problemu są istotnym elementem badań naukowych, ponieważ znane są wszystkie ich cechy i badacze mogą sprawdzić wpływ istnienia poszczególnych cech na skuteczność i efektywność swojej metody optymalizacji. Pożądane jest by testowe problemy optymalizacyjne posiadały cechy, które są przynajmniej zbliżone do cech widocznych w praktycznych problemach optymalizacyjnych.

4.1 Zbiór problemów CEC'2013

Powszechnie stosowanym zbiorem problemów testowych dla prac obejmujących tematykę wielowymiarowej optymalizacji problemów o ciągłej przestrzeni przeszukiwań jest zbiór zaproponowany na potrzeby konkursu organizowanego w ramach konferencji CEC w 2013 roku [70, 82, 94, 123, 125, 126, 149]. Niniejszy podrozdział przedstawia definicje wszystkich piętnastu problemów optymalizacyjnych, które wchodzą w skład powyższego zbioru, którego nazywać będziemy zbiorem problemów CEC'2013.

4.1.1 Funkcje bazowe

Wszystkie problemy optymalizacyjne ze zbioru CEC'2013 zbudowane są funkcji, które od wielu lat są podstawą badań znacznej części prac dotykających tematyki optymalizacji problemów o ciągłej przestrzeni przeszukiwań [62, 71, 146]. Ich n-wymiarowe definicje przedstawione są poniżej.

1. Funkcja sferyczna
$$f_{sferyczna}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{4.1}$$

2. Funkcja eliptyczna
$$f_{eliptyczna}(\mathbf{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} x_i^2 \tag{4.2}$$

3. Funkcja Rastrigina
$$f_{rastrigin}(\mathbf{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10] \tag{4.3}$$

4. Funkcja Ackleya
$$f_{ackley}(\mathbf{x}) = -20\exp\left(-0,2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \tag{4.4}$$

5. Funkcja Schwefela w wersji 1.2
$$f_{schwefel}(\mathbf{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{4.5}$$

6. Funkcja Rosenbrocka
$$f_{rosenbrock}(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \tag{4.6}$$

4.1.2 Funkcje testowe

Zbiór funkcji testowych należących do zbioru CEC'2013 to odpowiednio zmodyfikowane funkcje bazowe. W celu przedstawienia definicji funkcji testowych zacznijmy od prezentacji symboli i funkcji, które poza funkcjami bazowymi także są częścią ich definicji.

– S: zbiór zawierający liczby wymiarów wszystkich podproblemów, na przykład $S = \{s_1, s_2, s_3\} = \{50, 25, 100\}$ oznacza problem składający się z trzech podproblemów odpowiednio o 50, 25 i 100 wymiarach.

– C_i: suma i pierwszych elementów ze zbioru S, tj. $C_i = \sum_{j=1}^{i} s_i$. Dla ułatwienia zdefiniowania pozostałych symboli, przyjmuje się, że $C_0 = 0$.

– $\mathcal{P}$: losowa permutacja zbioru $\{1, ..., n\}$.

– $\mathbf{x}(\mathcal{P}_i : \mathcal{P}_j)$: wektor, którego elementami są wartości wektora $\mathbf{x}$ oznaczone indeksami wskazanymi przez permutację $\mathcal{P}$ począwszy od jej i-tego elementu, a kończąc na jej j-tym elemencie. Na przykład, dla $\mathbf{x} = [x_1, ..., x_6]$ i $\mathcal{P} = \{3, 1, 5, 4, 2, 6\}$, wektor $\mathbf{x}(\mathcal{P}_2 : \mathcal{P}_5)$ ma postać $[x_1, x_5, x_4, x_2]$.

– w_i: losowo wygenerowana waga dla i-tego podproblemu. Generowanie w_i odbywa się zgodnie z następującym wzorem

$$w_i = 10^{3\mathcal{N}(0,1)} \tag{4.7}$$

gdzie $\mathcal{N}(0, 1)$ oznacza wartość wygenerowaną przy użyciu rozkładu normalnego o zerowej wartości oczekiwanej i wariancji równej 1.

– T_{osz}: funkcja transformująca punkty w przestrzeni $\mathbb{R}^n$, która wprowadza gładkie lokalne nieregularności [34]. Wynikiem funkcji $T_{osz} : \mathbb{R}^n \to \mathbb{R}^n$ jest n-elementowy wektor $[z_{osz,1}, ..., z_{osz,n}]$, którego i-ty element jest następujący

$$z_{osz,i} = \text{sgn}(x_i) \cdot e^{\hat{x}_i + 0{,}049 \cdot [\sin(c_{1,i} \cdot \hat{x}_i) + \sin(c_{2,i} \cdot \hat{x}_i)]} \tag{4.8}$$

gdzie

$$\hat{x}_i = \begin{cases} \log(|x_i|), & \text{jeżeli } x_i \neq 0 \\ 0, & \text{jeżeli } x_i = 0 \end{cases} \tag{4.9}$$

$$\text{sgn}(x) = \begin{cases} -1, & \text{jeżeli } x < 0 \\ 0, & \text{jeżeli } x = 0 \\ 1, & \text{jeżeli } x > 0 \end{cases} \tag{4.10}$$

$$c_{1,i} = \begin{cases} 10, & \text{jeżeli } x_i > 0 \\ 5{,}5, & \text{jeżeli } x_i \leq 0 \end{cases} \tag{4.11}$$

$$c_{2,i} = \begin{cases} 7{,}9, & \text{jeżeli } x_i > 0 \\ 3{,}1, & \text{jeżeli } x_i \leq 0 \end{cases} \tag{4.12}$$

– T_{asy}^{β}: funkcja transformująca punkty w przestrzeni $\mathbb{R}^n$, która w przypadku funkcji symetrycznych powoduje, że nie są już one dłużej symetryczne [34]. Wynikiem funkcji $T_{asy}^{\beta} : \mathbb{R}^n \to \mathbb{R}^n$ jest n-elementowy wektor $[z_{asy,1}^{\beta}, ..., z_{asy,n}^{\beta}]$, którego i-ty element zdefiniowany jest następująco

$$z_{asy,i}^{\beta} = \begin{cases} x_i^{1+\beta\frac{i-1}{n-1}\sqrt{x_i}}, & \text{jeżeli } x_i > 0 \\ x_i, & \text{jeżeli } x_i \leq 0 \end{cases} \tag{4.13}$$

– Λ^{α} macierz diagonalna o wymiarach $n \times n$. Wartości jej głównej diagonali $\lambda_{i,i}$ wynoszą $\alpha^{0,5\frac{i-1}{n-1}}$. Przy pomocy macierzy Λ^{α} można sterować uwarunkowaniem funkcji, gdzie α jest wskaźnikiem uwarunkowania [34].

– $\boldsymbol{R}$: ortogonalna macierz obrotu (ang. *rotation matrix*) wygenerowana w sposób losowy by zapobiec sytuacji, w której optima lokalne położone są wzdłuż osi [117].

– o: liczba współdzielonych zmiennych decyzyjnych pomiędzy dwoma nakładającymi się na siebie podproblemami.

Zbiór problemów CEC'2013 składa się z następujących piętnastu funkcji testowych.

1. Funkcja f_1: przesunięta funkcja eliptyczna

$$f_1(\mathbf{z}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} z_i^2 \tag{4.14}$$

gdzie:

– $\mathbf{z} = T_{osz}(\mathbf{x} - \mathbf{x}^*)$,

– $\mathbf{x} \in [-100, 100]^n$,

– $n = 1000$.

Wybrane własności funkcji f_1:

– unimodalna,

– w pełni separowalna,

– $f_1(\mathbf{x}^*) = 0$.

2. Funkcja f_2: przesunięta funkcja Rastrigina

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} [z_i^2 - 10\cos(2\pi z_i) + 10] \tag{4.15}$$

gdzie:

- $\mathbf{z} = \Lambda^{10} T^{0,2}_{asy}(T_{osz}(\mathbf{x} - \mathbf{x}^*))$,
- $\mathbf{x} \in [-5, 5]^n$,
- $n = 1000$.

Wybrane własności funkcji f_2:

- wielomodalna,
- w pełni separowalna,
- $f_2(\mathbf{x}^*) = 0$.

3. Funkcja f_3: przesunięta funkcja Ackleya

$$f_3(\mathbf{z}) = -20 \exp\left(-0,2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)\right) + 20 + e \quad (4.16)$$

gdzie:

- $\mathbf{z} = \Lambda^{10} T^{0,2}_{asy}(T_{osz}(\mathbf{x} - \mathbf{x}^*))$,
- $\mathbf{x} \in [-32, 32]^n$,
- $n = 1000$.

Wybrane własności funkcji f_3:

- wielomodalna,
- w pełni separowalna,
- $f_3(\mathbf{x}^*) = 0$.

4. Funkcja f_4:

$$f_4(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{eliptyczna}(\mathbf{z_i}) + f_{eliptyczna}(\mathbf{z_{|S|}}) \quad (4.17)$$

gdzie:

- $\mathbf{z_i} = T_{osz}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S| - 1\}$,
- $\mathbf{z_{|S|}} = T_{osz}(\mathbf{y_{|S|}})$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,

- $\mathbf{x} \in [-100, 100]^n$,
- $S = \{50, 25, 25, 100, 50, 25, 25, 700\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S| - 1\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_4:

- unimodalna,
- nie w pełni separowalna,
- $f_4(\mathbf{x}^*) = 0$.

5. Funkcja f_5:

$$f_5(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{rastrigin}(\mathbf{z_i}) + f_{rastrigin}(\mathbf{z_{|S|}}) \tag{4.18}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T^{0,2}_{asy}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S| - 1\}$,
- $\mathbf{z_{|S|}} = \Lambda^{10} T^{0,2}_{asy}(\mathbf{y_{|S|}})$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-5, 5]^n$,
- $S = \{50, 25, 25, 100, 50, 25, 25, 700\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S| - 1\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_5:

- wielomodalna,
- nie w pełni separowalna,
- $f_5(\mathbf{x}^*) = 0$.

6. Funkcja f_6:

$$f_6(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{ackley}(\mathbf{z_i}) + f_{ackley}(\mathbf{z_{|S|}}) \tag{4.19}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T^{0,2}_{asy}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S| - 1\}$,

- $\mathbf{z_{|S|}} = \Lambda^{10} T^{0,2}_{asy}(\mathbf{y_{|S|}})$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-32, 32]^n$,
- $S = \{50, 25, 25, 100, 50, 25, 25, 700\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S| - 1\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_6:

- wielomodalna,
- nie w pełni separowalna,
- $f_6(\mathbf{x}^*) = 0$.

7. Funkcja f_7:

$$f_7(\mathbf{z}) = \sum_{i=1}^{|S|-1} w_i f_{schwefel}(\mathbf{z_i}) + f_{sferyczna}(\mathbf{z_{|S|}}) \tag{4.20}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T^{0,2}_{asy}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S| - 1\}$,
- $\mathbf{z_{|S|}} = \Lambda^{10} T^{0,2}_{asy}(\mathbf{y_{|S|}})$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-100, 100]^n$,
- $S = \{50, 25, 25, 100, 50, 25, 25, 700\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S| - 1\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_7:

- unimodalna,
- nie w pełni separowalna,
- $f_7(\mathbf{x}^*) = 0$.

8. Funkcja f_8:

$$f_8(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{eliptyczna}(\mathbf{z_i}) \tag{4.21}$$

gdzie:

- $\mathbf{z_i} = T_{osz}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-100, 100]^n$,
- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S|\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_8:

- unimodalna,
- nie w pełni separowalna,
- $f_8(\mathbf{x}^*) = 0$.

9. Funkcja f_9:

$$f_9(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{rastrigin}(\mathbf{z_i}) \tag{4.22}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T^{0,2}_{asy}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-5, 5]^n$,
- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S|\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_9:

- wielomodalna,

- nie w pełni separowalna,
- $f_9(\mathbf{x}^*) = 0$.

10. Funkcja f_{10}:

$$f_{10}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{ackley}(\mathbf{z_i}) \tag{4.23}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T_{asy}^{0,2}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-32, 32]^n$,
- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S|\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_{10}:

- wielomodalna,
- nie w pełni separowalna,
- $f_{10}(\mathbf{x}^*) = 0$.

11. Funkcja f_{11}:

$$f_{11}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{schwefel}(\mathbf{z_i}) \tag{4.24}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T_{asy}^{0,2}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1} : \mathcal{P}_{C_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-100, 100]^n$,
- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S|\}$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i = 1000$.

Wybrane własności funkcji f_{11}:

- unimodalna,
- nie w pełni separowalna,
- $f_{11}(\mathbf{x}^*) = 0$.

12. Funkcja f_{12}: przesunięta funkcja Rosenbrocka

$$f_{12}(\mathbf{z}) = \sum_{i=1}^{n-1}[100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2] \tag{4.25}$$

gdzie:

- $\mathbf{z} = \mathbf{x} - \mathbf{x}^* + [1, ..., 1]$,
- $\mathbf{x} \in [-100, 100]^n$,
- $n = 1000$.

Wybrane własności funkcji f_{12}:

- wielomodalna,
- funkcja z nakładającymi się na siebie podproblemami,
- $f_{12}(\mathbf{x}^*) = 0$.

13. Funkcja f_{13}:

$$f_{13}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{schwefel}(\mathbf{z_i}) \tag{4.26}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T_{asy}^{0,2}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y_i} = \mathbf{y}(\mathcal{P}_{C_{i-1}+1-o\cdot(i-1)} : \mathcal{P}_{C_i - o\cdot(i-1)})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$,
- $\mathbf{x} \in [-100, 100]^n$,
- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S|\}$,
- $o = 5$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i - o \cdot (|S| - 1) = 905$.

Wybrane własności funkcji f_{13}:

- unimodalna,
- funkcja z nakładającymi się na siebie podproblemami, które do siebie pasują,
- $f_{13}(\mathbf{x}^*) = 0$.

14. Funkcja f_{14}:

$$f_{14}(\mathbf{z}) = \sum_{i=1}^{|S|} w_i f_{schwefel}(\mathbf{z_i}) \tag{4.27}$$

gdzie:

- $\mathbf{z_i} = \Lambda^{10} T_{asy}^{0,2}(\boldsymbol{R_i}\mathbf{y_i})$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{y_i} = \mathbf{x}(\mathcal{P}_{C_{i-1}+1-o\cdot(i-1)} : \mathcal{P}_{C_i-o\cdot(i-1)}) - \mathbf{x_i^*}$ dla $i \in \{1, ..., |S|\}$,
- $\mathbf{x_i^*}$ to najlepsze rozwiązanie i-tego podproblemu dla $i \in \{1, ..., |S|\}$,
- $\mathbf{x} \in [-100, 100]^n$,
- $S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$,
- $\boldsymbol{R_i}$ to i-ta macierz obrotu o wymiarach $s_i \times s_i$ dla $i \in \{1, ..., |S|\}$,
- $o = 5$,
- $n = \sum_{i \in \{1,...,|S|\}} s_i - o \cdot (|S| - 1) = 905$.

Wybrane własności funkcji f_{14}:

- unimodalna,
- funkcja z nakładającymi się na siebie podproblemami, które nie pasują do siebie,
- $f_{14}(\mathbf{x}^*) \geq 0$.

15. Funkcja f_{15}: przesunięta funkcja Schwefela w wersji 1.2

$$f_{15}(\mathbf{z}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j \right)^2 \tag{4.28}$$

gdzie:

- $\mathbf{z} = T_{asy}^{0,2}(T_{osz}(\mathbf{x} - \mathbf{x}^*))$,
- $\mathbf{x} \in [-100, 100]^n$,

– $n = 1000$.

Wybrane własności funkcji f_{15}:

– unimodalna,

– w pełni nieseparowalna,

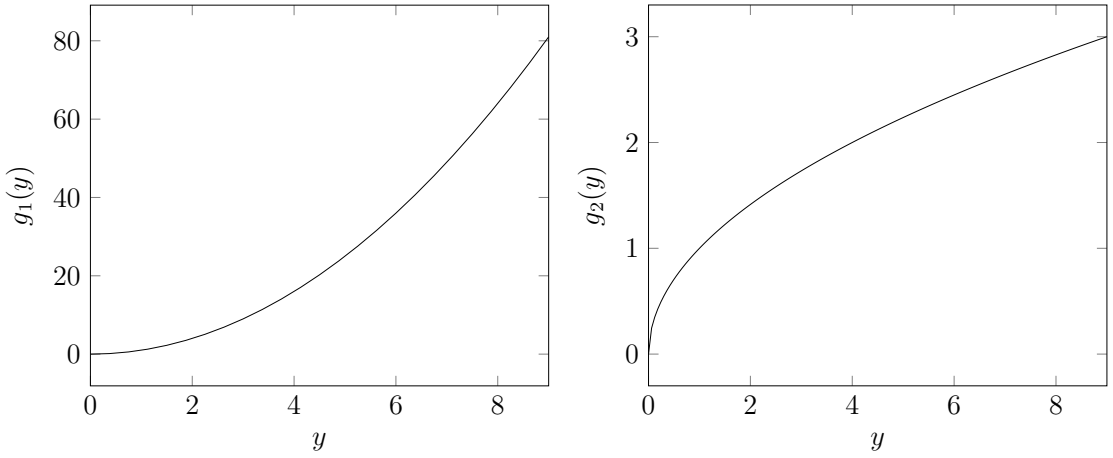
– $f_{15}(\mathbf{x}^*) = 0$.

4.2 Modyfikacje zbioru problemów CEC'2013

Zbiór problemów testowych CEC'2013 składa się z funkcji, które w większości zbudowane są z addytywnie separowalnych podproblemów. Wyjątkami są funkcje f_3 i f_{15}. Funkcja f_3 jest w pełni separowalna i składa się z nieaddytywnie separowalnych podproblemów, które są jednowymiarowe, natomiast funkcja f_{15} jest w pełni nieseparowalna. Możemy zatem stwierdzić, że zbiór problemów testowych CEC'2013 jest odpowiedni dla strategii bazujących na grupowaniu różnicowym, ponieważ prawie wszystkie funkcje należące do tego zbioru spełniają założenia grupowania różnicowego. Z tego powodu, autor niniejszej pracy proponuje dwie modyfikacje zbioru CEC'2013, które przekształcają addytywną separowalność w nieaddytywną.

Zbiór wartości wszystkich piętnastu funkcji zbioru CEC'2013 jest podzbiorem zbioru nieujemnych liczb rzeczywistych. Rozważmy funkcję f, której zbiór wartości to Y oraz funkcję g, która jest ściśle rosnąca w zbiorze Y. Przy takim założeniu możemy zapisać, że $\forall_{\ddot{\mathbf{x}}_1,\ddot{\mathbf{x}}_2\in\mathcal{D}} f(\ddot{\mathbf{x}}_1) < f(\ddot{\mathbf{x}}_2) \rightarrow h(\ddot{\mathbf{x}}_1) < h(\ddot{\mathbf{x}}_2)$ i $\forall_{\ddot{\mathbf{x}}_1,\ddot{\mathbf{x}}_2\in\mathcal{D}} f(\ddot{\mathbf{x}}_1) = f(\ddot{\mathbf{x}}_2) \rightarrow h(\ddot{\mathbf{x}}_1) = h(\ddot{\mathbf{x}}_2)$, gdzie $h = g \circ f$. Należy zauważyć, że dla każdego tego rodzaju złożenia funkcji, wynik porównania jakości dwóch dowolnych rozwiązań będzie dokładnie taki sam, gdy weźmiemy pod uwagę zarówno funkcję f, jak i funkcję h. Z drugiej strony, jeżeli funkcja f jest addytywnie separowalna to funkcja h może już być nieaddytywnie separowalna np. gdy funkcja g jest nielinowa. Niemniej jednak, ich grafy interakcji będą dokładnie takie same.

Skoro zbiorem wartości wszystkich funkcji ze zbioru CEC'2013 jest zbiór nieujemnych liczb rzeczywistych to do złożenia potrzebujemy funkcji g, która jest rosnąca dla nieujemnych argumentów. Autor niniejszej pracy wybrał w tym celu dwie funkcje. Są nimi funkcja kwadratowa $g_1(y) = y^2$ oraz funkcja pierwiastkowa $g_2(y) = \sqrt{y}$. Zostały one wybrane, ponieważ są ściśle rosnące w zbiorze nieujemnych liczb rzeczywistych w różny sposób. Przyrost funkcji kwadratowej jest większy dla większych argumentów $\forall_{\ddot{y}_2>\ddot{y}_1\geq 0,\delta>0} g_1(\ddot{y}_1+\delta) - g_1(\ddot{y}_1) < g_1(\ddot{y}_2+\delta) - g_1(\ddot{y}_2)$, natomiast przyrost funkcji pierwiastkowej jest mniejszy dla większych argumentów $\forall_{\ddot{y}_2>\ddot{y}_1\geq 0,\delta>0} g_2(\ddot{y}_1+\delta) - g_2(\ddot{y}_1) > g_2(\ddot{y}_2+\delta) - g_2(\ddot{y}_2)$. Innymi słowy, pochodna funkcji



(a) (b)

Rysunek 4.1: Wykres funkcji kwadratowej (a) i funkcji pierwiastkowej (b) dla nieujemnych argumentów

kwadratowej jest funkcją rosnącą, natomiast pochodna funkcji pierwiastkowej jest funkcją malejącą. Rysunek 4.1 przedstawia wykresy obu funkcji. Funkcje otrzymane w ramach modyfikacji zbioru CEC'2013 zostaną oznaczone jako $(f_i)^2$ dla złożenia funkcji f_i z funkcja kwadratowa oraz jako $\sqrt{f_i}$, gdy funkcja f_i zostanie złożona z funkcją pierwiastkową. Innymi słowy, $(f_i)^2 = g_1 \circ f_i$ oraz $\sqrt{f_i} = g_2 \circ f_i$.

4.3 Klasyfikacja problemów testowych

Dla każdego $i \in \{1, ..., 15\}$ funkcje f_i, $(f_i)^2$ i $\sqrt{f_i}$ reprezentowane są przez ten sam graf interakcji. Dodatkowo posiadają one także te same przedziały monotoniczności. Możemy zatem oczekiwać, że analizując wyniki eksperymentów, będziemy mogli wyciągnąć dla nich podobne wnioski. Z tego powodu, zdefiniujmy sobie zbiór $F_i = \{f_i, (f_i)^2, \sqrt{f_i}\}$. Jeżeli mowa będzie o przedziale F_i–F_j to oznaczać on będzie następującą sumę zbiorów $F_i \cup F_{i+1} \cup ... \cup F_{j-1} \cup F_j$.

Wszystkie rozważane problemy testowe możemy podzielić na pięć kategorii ze względu na ich wewnętrzną strukturę reprezentowaną przez graf interakcji [70, 149].

1. Funkcje w pełni separowalne (F_1–F_3).
2. Funkcje nie w pełni separowalne składające się m.in. z w pełni separowalnych zmiennych (F_4–F_7).
3. Funkcje nie w pełni separowalne, które nie posiadają żadnych w pełni separowalnych zmiennych (F_8–F_{11}).
4. Funkcje z nakładającymi się na siebie podproblemami (F_{12}–F_{14}).

5. Funkcje w pełni nieseparowalne (F_{15}).

Powyższa klasyfikacja nie uwzględnia typu separowalności (addytywna lub nieaddytywna) pomiędzy zmiennymi decyzyjnymi, dlatego podzielmy wszystkie problemy testowe jeszcze na dwie inne kategorie.

1. Funkcje zawierające addytywnie separowalne podproblemy (f_1, f_2 i f_4–f_{14}).

2. Funkcje zawierające nieaddytywnie separowalne podproblemy (f_3, $(f_1)^2$–$(f_{14})^2$ i $\sqrt{f_1}$–$\sqrt{f_{14}}$).

4.4 Konfiguracja i wyniki eksperymentów

Autor niniejszej pracy przeprowadził szereg eksperymentów na zbiorze problemów testowych CEC'2013 oraz ich dwóch modyfikacji. Ich celem było znalezienie odpowiedzi na następujące pytania badawcze.

1. Czy zaproponowana w ramach niniejszej pracy strategia dekompozycji zwraca dekompozycję o jakości nie gorszej niż w przypadku rekursywnych strategii dekompozycji bazujących na grupowaniu różnicowym dla problemów składających się z addytywnie separowalnych podproblemów i istotnie lepszej dla problemów z nieaddytywnie separowalnymi podproblemami?

2. Czy strategia IRRG, pomimo zwiększonego kosztu dekompozycji względem rekursywnych strategii dekompozycji bazujących na grupowaniu różnicowym, może zostać skutecznie osadzona w architekturach koewolucji kooperatywnej?

Wyniki eksperymentów zaprezentowane w podrozdziałach 4.4.3, 4.4.4 i 4.4.5 zostały opublikowane przez autora niniejszej pracy w czasopiśmie IEEE Transactions on Evolutionary Computation [57].

4.4.1 Metody optymalizacji wykorzystujące strategię IRRG i metody konkurujące

W celu zweryfikowania drugiego z postawionych pytań badawczych, strategię dekompozycji IRRG osadzono w dwóch różnych architekturach koewolucji kooperatywnej, CBCC [92, 124] i CCFR2 [149]. Obie architektury po osadzeniu w nich istniejących strategii dekompozycji osiągały dobre wyniki optymalizacji wielowymiarowych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań. Metody optymalizacji, które powstały w wyniku osadzenia strategii IRRG w architekturach

CBCC i CCFR2 oznaczone zostały jako CBCC-IRRG i CCFR2-IRRG. Analogiczne nazewnictwo zostało zastosowane do metod CBCC-RDG3 [92, 124, 125] i CCFR2-RDG3 [125, 149], które powstały dzięki osadzeniu strategii RDG3 w architekturach CRCC i CCFR2. Należy wspomnieć, że w przeciwieństwie do CBCC-RDG3, metoda CCFR2-RDG3 nie jest znana w literaturze. W pracy [149], w której zaproponowana została architektura CCFR2, połączono ją ze strategią DG2 [94]. Rozsądnym wydaje się zatem by w architekturze CCFR2 osadzić jednak nowszą wersję strategii bazującej na grupowaniu różnicowym, która pozwala na osiągnięcie lepszych wyników optymalizacji wykorzystując architektury koewolucji kooperatywnej z zachłannym wyborem komponentu do optymalizacji w porównaniu z poprzednikami strategii RDG3 [125]. Zbiór metod konkurujących uzupełnia metoda SHADE-ILS [82], która jest przedstawicielem innej rodziny metod optymalizacji, które są dedykowane do rozwiązywania wielowymiarowych problemów optymalizacyjnych, mianowicie hybrydowych metod optymalizacji. Metoda SHADE-ILS uznawana jest obecnie za jedną z najlepszych tego typu metod [82].

Architektury koewolucji kooperatywnej wymagają zdefiniowania metod optymalizacji, które będą odpowiedzialne za szukanie najlepszego rozwiązania w ramach komponentów. Autor niniejszej pracy wybrał w tym celu metodę CMA-ES [35] dla wszystkich komponentów, ponieważ wyniki przedstawione w pracach [123,149] sugerują, że jest to najlepszy wybór dla architektur CBCC i CCFR2. Metoda CMA-ES została zaimplementowana w wielu językach programowania. Różne implementacje metody CMA-ES mogą zwracać różne wyniki optymalizacji dla tych samych instancji problemów testowych [7]. Implementacja metody CMA-ES w języku programowania Python[1] osiąga najlepsze rezultaty [7], dlatego została ona użyta w przeprowadzonych w ramach niniejszej pracy eksperymentach. Autorzy metody CBCC-RDG3 zaimplementowali ją w języku Matlab[2]. W celu użycia implementacji metody CMA-ES w języku Python w ramach architektury CBCC, dostarczony przez Autorów kod źródłowy architektury CBCC został przepisany do języka Python. Kod źródłowy odpowiedzialny za działanie architektury CCFR2 został również zaimplementowany, przez autora niniejszej pracy, w języku Python na podstawie pseudokodów zawartych w pracy, w której architektura CCFR2 została zaproponowana [149]. Dekompozycja wszystkich rozważanych problemów testowych strategią RDG3 została przeprowadzona przy użyciu oryginalnych kodów źródłowych napisanych w języku Matlab. Wynik tak przeprowadzonej dekompozycji jest następnie przekazywany jako argument wejściowy do programu odpowiedzialnego za optymalizację zdekompono-

[1] `https://github.com/CMA-ES/pycma`
[2] `https://bitbucket.org/yuans/rdg3`

wanego problemu optymalizacyjnego za pomocą architektury CBCC lub CCFR2, zaimplementowanego w języku Python. W ten sam sposób zintegrowane zostały ze sobą programy, których zadaniem jest odpowiednio zdekomponowanie danego problemu używając strategii IRRG oraz uruchomienie architektury CBCC lub CCFR2. Strategia IRRG została zaimplementowana w języku programowania C++. Ostatnią z rozważanych metod optymalizacji stanowi metoda SHADE-ILS. Wszystkie eksperymenty dotyczące metody SHADE-ILS zostały przeprowadzone przy użyciu kodów źródłowych dostarczonych przez jej Autorów[3], które zostały napisane w języku Python.

Wszystkie rozważane w niniejszej pracy metody optymalizacji posiadają parametry, których wartości muszą zostać podane przez użytkownika. W przypadku strategii RDG3 (Tabela 4.1), architektur CBCC (Tabela 4.2) i CCFR2 (Tabela 4.3) oraz metody SHADE-ILS (Tabela 4.4), wartości ich parametrów zostały wzięte z prac, w których zostały one zaproponowane lub niedawno użyte i zostały oznaczone jako zalecane.

Wartości parametrów zaproponowanej przez autora niniejszej pracy strategii IRRG (Tabela 4.5) zostały dobrane na podstawie wyników wstępnych eksperymentów. Do otrzymania wysokiej jakości rozwiązania ($\mathbf{x_{hq}}$) wykorzystane zostaną dwie metody optymalizacji, których wartości parametrów zostaną ustawione zgodnie z rekomendowanymi wartościami. Pierwszą metodą optymalizacji jest metoda SHADE [127], której zadaniem jest eksploracja przestrzeni przeszukiwań. Najlepsze rozwiązania znalezione przez metodę SHADE będzie następnie optymalizowane lokalnie przez metodę MTS-LS1 [72]. Wstępne eksperymenty zostały przeprowadzone na standardowym zbiorze problemów testowych CEC'2013 i polegały na znalezieniu minimalnych wartości warunku zatrzymania bazującego na liczbie wyliczeń funkcji przystosowania dla obu użytych metod optymalizacji oraz minimalnych wartości parametrów ϵ_{sti} i n_s, dla których strategia IRRG ciągle zwraca idealną dekompozycję dla problemów bez nakładających się na siebie podproblemów.

Wszystkie rozważane, w niniejszej pracy, metody optymalizacji są niedeterministyczne. Z tego powodu każdy eksperyment został powtórzony 25 razy. Wartość ta jest powszechnie stosowana w pracach, które w swoich wynikach uwzględniają zbiór problemów CEC'2013 [73, 82, 94, 125, 149].

4.4.2 Warunek zatrzymania

W ramach niniejszej pracy, warunkiem zatrzymania jest maksymalna liczba wyliczeń funkcji przystosowania. Został on wybrany, ponieważ jest on powszechnie

[3] `https://github.com/dmolina/shadeils`

Tabela 4.1: Wartości parametrów strategii RDG3 [125]

Nazwa parametru	Wartość
Preferowany rozmiar grup zawierających separowalne zmienne (ϵ_s)	100
Próg wielkości grup dla problemów z nakładającymi się na siebie podproblemami (ϵ_n)	50

Tabela 4.2: Wartości parametrów architektury CBCC [124, 125]

Nazwa parametru	Wartość
Metoda optymalizacji pojedynczego komponentu	CMA-ES
Maksymalna liczba wyliczeń funkcji przystosowania podczas każdej optymalizacji pojedynczego komponentu	1000
Współczynnik wygładzania (w)	0,5

Tabela 4.3: Wartości parametrów architektury CCFR2 [149]

Nazwa parametru	Wartość
Metoda optymalizacji pojedynczego komponentu	CMA-ES
Maksymalna liczba iteracji metody CMA-ES podczas każdej optymalizacji pojedynczego komponentu	100
Współczynnik wygładzania (w)	0,1

Tabela 4.4: Wartości parametrów metody SHADE-ILS [82]

Nazwa parametru	Wartość
Rozmiar populacji metody SHADE	100
Początkowy krok metody MTS-LS1	20
Minimalna wartość współczynnika względnej poprawy jakości najlepszego znalezionego dotychczas rozwiązania	5%
Liczba iteracji bez osiągnięcia minimalnej wartości współczynnika względnej poprawy jakości najlepszego znalezionego dotychczas rozwiązania, po których następuje restart metody SHADE-ILS	3

Tabela 4.5: Wartości parametrów strategii IRRG

Nazwa parametru	Wartość
Metoda optymalizacji używana w trakcie początkowego procesu optymalizacji	SHADE i MTS-LS1
Rozmiar populacji metody SHADE [127]	100
Wielkość archiwum metody SHADE [127]	200
Procent najlepszych znalezionych dotychczas rozwiązań, które mogą zostać wybrane do mutacji przez metodę SHADE [127]	10%
Wielkość pamięci, w której przechowywane są wartości współczynników wykorzystywanych w mutacji i krzyżowaniu przez metodę SHADE [127]	10^3
Maksymalna liczba wyliczeń funkcji przystosowania przez metodę SHADE	$5 \cdot 10^3$
Początkowy zakres przeszukiwań metody MTS-LS1 określony jako procent różnicy pomiędzy maksymalną i minimalną dopuszczalną wartością [72]	20%
Maksymalna liczba wyliczeń funkcji przystosowania przez metodę MTS-LS1	$1,5 \cdot 10^4$
Liczba próbek (n_s)	10
Maksymalna liczba kolejnych iteracji strategii IRRG bez wykrycia nowych interakcji pomiędzy zmiennymi decyzyjnymi (ϵ_{sti})	15
Preferowany rozmiar grup zawierających separowalne zmienne (ϵ_s) [125]	100

stosowany w literaturze, gdy badania dotyczą wielowymiarowych problemów optymalizacyjnych ze zbioru CEC'2013 [73, 82, 94, 125, 149]. Maksymalna liczba wyliczeń funkcji przystosowania jest miarodajnym warunkiem zatrzymania, gdy chcemy porównać między sobą różne metody optymalizacji, o ile nie korzystają one z żadnych mechanizmów przechowujących dotychczas wyliczone wartości funkcji przystosowania [107] oraz sumaryczny nakład obliczeniowy poświęcany na wyliczenie przystosowania jest znacząco większy niż nakład obliczeniowy poświęcony na pozostałe czynności metody [102]. Metody SHADE-ILS i CMA-ES nie posiadają mechanizmów zapamiętujących dotychczas wyliczone wartości funkcji przystosowania, natomiast w przypadku wielowymiarowych problemów optymalizacyjnych istnieje uzasadnione podejrzenie, że czas pojedynczego wyliczenia funkcji przystosowania będzie odpowiednio wysoki.

Dwie wartości maksymalnej liczby wyliczeń funkcji przystosowania rozważane są w niniejszej pracy. Pierwszą wartością, oznaczoną w dalszej części pracy jako standardowy warunek zatrzymania, jest $3 \cdot 10^6$ wyliczeń funkcji przystosowania. Wartość ta jest powszechnie stosowana w literaturze dla zbioru CEC'2013. Drugą rozważaną wartością jest $6 \cdot 10^6$ wyliczeń funkcji przystosowania. Ten wydłużony warunek zatrzymania został uwzględniony w celu zweryfikowania, czy wpływ zwiększonego kosztu dekompozycji strategii IRRG w porównaniu do strategii RDG3 na wyniki optymalizacji będzie malał wraz ze zwiększeniem nakładu obliczeniowego.

4.4.3 Dokładność dekompozycji

Strategia IRRG, w przeciwieństwie do strategii bazujących na grupowaniu różnicowych, nigdy nie wykryje zależności, które w rzeczywistości nie istnieją, a ponadto powinna eliminować typowy problem strategii bazujących na sprawdzaniu monotoniczności, czyli zbyt wiele pominiętych zależności. Powyższe założenia zostały zweryfikowane mierząc dokładność dekompozycji za pomocą trzech miar (ρ_1, ρ_2 i ρ_3) opisanych w rozdziale 1.1.1 dla wszystkich rozważanych problemów testowych.

Tabela 4.6 przedstawia dokładność dekompozycji strategii IRRG i RDG3. Strategia IRRG, w przeciwieństwie do strategii RDG3, jest strategią niedeterministyczną, dlatego raportujemy dla niej medianę, wartość średnią oraz odchylenie standardowe z 50 niezależnych przebiegów. Do policzenia dokładności wykorzystane zostały dekompozycje, które następnie zostały użyte podczas optymalizacji metodami CBCC-IRRG i CCFR2-IRRG przy standardowym warunku zatrzymania. Dla funkcji, które nie zawierają nakładających się na siebie podproblemów (wszystkie poza F_{12}–F_{14}), strategia IRRG jest wysoce powtarzalna, ponieważ wartości odchyleń standardowych dla wszystkich miar wynoszą 0%. Należy jednak wspomnieć, że wartości dla

miar ρ_2 i ρ_3 w przypadku funkcji, których przynajmniej dwie zmienne są w pełni separowalne, zostały wyliczone tuż przed ich podzieleniem na grupy o rozmiarze ϵ_s. W przeciwnym wypadku, miary te mylnie pokazywałyby, że obie strategie wykryły zależności, które w rzeczywistości nie istnieją. Wartości miary ρ_2 dla funkcji, które składają się z nakładających się na siebie podproblemów, są poniżej 100% dla strategii IRRG, mimo że nie powinna ona wykrywać nieistniejących interakcji pomiędzy zmiennymi decyzyjnymi. Jest to spowodowane faktem, że wszystkie trzy miary biorą pod uwagę jedynie bezpośrednie zależności pomiędzy zmiennymi, natomiast strategie IRRG i RDG3 wykrywają również pośrednie interakcje [109, 121].

Dla funkcji w pełni separowalnych i nie w pełni separowalnych strategia IRRG znajduje idealną dekompozycję we wszystkich swoich przebiegach niezależnie czy separowalność jest addytywna czy też nie. Tabela 4.6 przedstawia optymalną wartość wszystkich trzech miar dla funkcji ze zbioru F_1–F_{11}. Strategia RDG3 nie wykrywa żadnych nieistniejących zależności dla prawie wszystkich funkcji ze standardowego zbioru CEC'2013. Wyjątkami są funkcje f_3 i f_6. Dla f_3, strategia RDG3 tworzy jedną grupę składającą się ze wszystkich zmiennych decyzyjnych ($\rho_2 = \rho_3 = 0\%$). Jest to oczekiwany wynik, ponieważ f_3 jest jedyną funkcją w standardowym zbiorze CEC'2013, która jest nieaddytywnie separowalna. Z drugiej strony, pomimo, że, funkcja f_6 jest addytywnie separowalna, wartość $\rho_2 = 71,93\%$ jest zaskakująco niska. Analiza wyników wskazuje, że prawdopodobną przyczyną takiego wyniku jest niedokładność automatycznego mechanizmu do estymacji wartości ϵ, który w niektórych przypadkach mógł zwracać zbyt niską wartość. Strategii RDG3 nie udało się wykryć wszystkich zależności dla funkcji f_5, f_8 i f_{10}. Odpowiadające im wartości miary ρ_1 są poniżej 100%. Przyczyną może być tym razem zbyt wysoka wartość ϵ dla niektórych sprawdzeń czy dwie grupy zmiennych decyzyjnych oddziałują na siebie wzajemnie. Strategia IRRG stara się minimalizować ryzyko, że niektóre z istniejących zależności zostaną pominięte ze względu na zbyt wysoką wartość ϵ poprzez wielokrotne wywoływanie RRG. Z drugiej strony, taki zabieg zwiększa szansę na wykrycie interakcji, które w rzeczywistości nie istnieją. Na podstawie wyników przedstawionych w tabeli Tabela 4.6 możemy stwierdzić, że ten niepożądany efekt nie zdarza się nigdy lub zdarza się niezwykle rzadko. Prawdopodobnie wynika to z faktu, że w strategii IRRG pomijane są próbki, dla których różnice w odpowiadających im wartościach funkcji przystosowania są mniejsze od wyliczonej wartości ϵ (Pseudokod 3, linie 8 i 13).

Zmodyfikowane wersje funkcji od f_1 do f_3 są nieaddytywnie separowalne, gdzie każdy podproblem jest jednowymiarowy. Z tego powodu, oczekiwanym wynikiem dla miary ρ_2 jest wartość 0% gdy rozważamy dekomponowanie problemu strategią

Tabela 4.6: Dokładność dekompozycji

Fun.	IRRG									RDG3		
	ρ_1[%]			ρ_2[%]			ρ_3[%]			ρ_1[%]	ρ_2[%]	ρ_3[%]
	Mdn.	Śr.	Odch. st.	Mdn.	Śr.	Odch. st.	Mdn.	Śr.	Odch. st.			
f_1	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	**100**	**100**
$(f_1)^2$	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	39,97	39,97
$\sqrt{f_1}$	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	50,40	50,40
f_2	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	**100**	**100**
$(f_2)^2$	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	0	0
$\sqrt{f_2}$	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	0	0
f_3	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	0	0
$(f_3)^2$	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	0	0
$\sqrt{f_3}$	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	**100**	**0**	Nd.	0	0
f_4	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**100**
$(f_4)^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	89,26	56,57	57,13
$\sqrt{f_4}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	70,70	73,09	73,05
f_5	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	98,85	**100**	99,98
$(f_5)^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	89,22	51,78	52,42
$\sqrt{f_5}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	89,83	83,17	83,29
f_6	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	71,93	72,42
$(f_6)^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	70,13	67,36	67,41
$\sqrt{f_6}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	84,86	58,02	58,48
f_7	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**100**
$(f_7)^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	92,62	92,74
$\sqrt{f_7}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	92,62	92,74
f_8	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	70,20	**100**	97,98
$(f_8)^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	63,88	99,72	97,28
f_8	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	69,57	38,80	40,89
f_9	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**100**
$(f_9)^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	84,28	44,36	47,07
$\sqrt{f_9}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	92,47	63,37	65,35
f_{10}	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	93,28	**100**	99,54
$(f_{10})^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	93,21	98,82	98,44
$\sqrt{f_{10}}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	89,85	98,84	98,23
f_{11}	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**100**
$(f_{11})^2$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	98,60	43,58	47,32
$\sqrt{f_{11}}$	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	95,09	49,35	52,45
f_{12}	**99,20**	**99,07**	**0,32**	84,86	83,94	4,80	84,89	83,97	4,79	98,10	**95,28**	**95,29**
$(f_{12})^2$	99,20	99,14	0,29	**84,56**	**81,80**	**9,14**	**84,59**	**81,83**	**9,12**	**100**	0	0,20
$\sqrt{f_{12}}$	99,20	99,14	0,36	**83,54**	**81,77**	**7,51**	**83,57**	**81,80**	**7,49**	**100**	0	0,20
f_{13}	**99,93**	**99,84**	**0,22**	0,23	18,04	23,88	8,44	24,78	21,90	92,65	**98,61**	**98,12**
$(f_{13})^2$	99,93	99,83	0,18	**0,23**	**21,33**	**24,36**	**8,44**	**27,79**	**22,34**	**100**	0	8,23
$\sqrt{f_{13}}$	99,85	99,81	0,20	**0,23**	**21,77**	**26,30**	**8,44**	**28,20**	**24,12**	**100**	0	8,23
f_{14}	**99,70**	**99,70**	**0,24**	37,54	33,07	28,28	42,66	38,55	25,93	94,55	**96,88**	**96,68**
$(f_{14})^2$	**99,78**	**99,78**	**0,22**	13,35	22,66	25,41	20,47	29,01	23,30	87,83	**53,26**	**56,11**
$\sqrt{f_{14}}$	**99,89**	**99,78**	**0,24**	0,35	24,52	27,28	8,55	30,72	25,01	73,85	**56,69**	**58,11**
f_{15}	**100**	**100**	**0**	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	Nd.	**100**
$(f_{15})^2$	**100**	**100**	**0**	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	Nd.	**100**
$\sqrt{f_{15}}$	**100**	**100**	**0**	Nd.	Nd.	Nd.	**100**	**100**	**0**	**100**	Nd.	**100**

RDG3. Niemniej jednak, wartości dla funkcji $(f_1)^2$ i $\sqrt{f_1}$ są większe od 0, chociaż ciągle dużo mniejsze od 100%. Wytłumaczeniem tego fenomenu może być również zbyt wysoka wartość ϵ, która zapobiega wykrywaniu nieistniejących zależności przez strategię RDG3. Kolejnym czynnikiem może być także próg wielkości grup ϵ_n stosowany przez strategię RDG3, który w przypadku osiągnięcia nie pozwala na dalsze dodawanie zmiennych decyzyjnych do aktualnie przetwarzanej grupy. Powyższe przypuszczenia są zgodne z obserwacją, że wartość miary ρ_2 jest różna gdy porównamy wartości dla $(f_1)^2$ i $\sqrt{f_1}$. Dla zmodyfikowanych funkcji, które nie są w pełni separowalne ($(f_4)^2$–$(f_{11})^2$ i $\sqrt{f_4}$–$\sqrt{f_{11}}$) oczekiwaliśmy również połączenia wszystkich zmiennych decyzyjnych w jedną grupę. Zgodnie z otrzymanymi wynikami, tj. $\rho_1 < 100\%$ lub $\rho_2 > 0\%$, stwierdzamy, że utworzonych zostało więcej grup. Przyczyny niewykrycia wszystkich nieistniejących interakcji przez strategię RDG3 są zapewne takie same jak w przypadku funkcji $(f_1)^2$–$(f_3)^2$ i $\sqrt{f_1}$–$\sqrt{f_3}$.

W przypadku funkcji z nakładającymi się na siebie podproblemami, np. F_{12}–F_{14}, ich idealna dekompozycja nie istnieje [94,105]. Dodatkowo różne dekompozycje mogą być użyteczne w różnych fazach procesu optymalizacji [56,105]. Dla funkcji ze zbioru F_{12}–F_{14}, każda para zmiennych jest od siebie zależna bezpośrednio lub pośrednio. W takim przypadku, wartość miary ρ_1 jest maksymalna gdy wszystkie zmienne utworzą pojedynczą grupę, natomiast miara ρ_2 osiągnie wartość 100% tylko wtedy, gdy w każdej grupie znajdować się będą jedynie zmienne, które bezpośrednie oddziałują na siebie wzajemnie. Ostatnia miara, ρ_3, jest kompromisem pomiędzy tymi dwiema miarami, dlatego nigdy nie osiągnie ona wartości 100%. Strategia IRRG może zwróci inną dekompozycję dla funkcji z nakładającymi się na siebie podproblemami podczas każdego jej uruchomienia. Z tego powodu, wartości odchyleń standardowych są większe niż 0%.

Strategia IRRG powinna być niewrażliwa na rodzaj separowalności pomiędzy podproblemami (addytywna separowalność lub nieaddytywna separowalność). Z tego powodu, różnica w dokładności dekompozycji powinna być nieznacząca gdy porównamy funkcje z nakładającymi się na siebie podproblemami, które należą do standardowego zbioru CEC'2013 z ich zmodyfikowanymi wersjami. Na podstawie otrzymanych wyników możemy stwierdzić, że dla większości funkcji wartości trzech rozważanych miar są podobne dla ich standardowej, podniesionej do kwadratu oraz spierwiastkowanej wersji. Hipoteza ta została zweryfikowana testem Wilcoxona z uwzględnieniem korekty Holma-Bonferroniego na poziomie istotności równym 5%. Weryfikowane były wartości każdej z trzech miar dla par $(f_i, (f_i)^2)$, $(f_i, \sqrt{f_i})$, i $((f_i)^2, \sqrt{f_i})$, gdzie $i \in \{12, 13, 14\}$ (dla pozostałych funkcji, strategia IRRG uzyskała te same wartości miar ρ_1–ρ_3 dla wszystkich przebiegów). Gdy przyjrzymy się

wartościom mediany dla ρ_1 dla funkcji f_{14}, $(f_{14})^2$ i $\sqrt{f_{14}}$ to widzimy duże różnice w wartościach. Nie zostały one sklasyfikowane jako istotne przez testy statystyczne ze względu na wysoką wartość odchylenia standardowego.

Zupełnie inne wnioski można wyciągnąć analizując dokładność dekompozycji dla funkcji ze zbioru F_{12}–F_{14} uzyskaną przez strategię RDG3. Dla funkcji f_{12}–f_{14}, strategia RDG3 potrafi utworzyć takie grupy zmiennych decyzyjnych, które nie pomijają wielu istniejących i bezpośrednich zależności pomiędzy zmiennymi oraz grupy składają się głównie z samych zmiennych, które bezpośrednio oddziałują na siebie wzajemnie, ponieważ wartości wszystkich trzech miar są powyżej 90%. Gdy weźmiemy pod uwagę funkcje $(f_{12})^2$, $\sqrt{f_{12}}$, $(f_{13})^2$ i $\sqrt{f_{13}}$ to odbiegają one mocno od swoich standardowych wersji dla miar ρ_2 i ρ_3. Niemniej jednak dla par $((f_{12})^2, \sqrt{f_{12}})$ i $((f_{13})^2, \sqrt{f_{13}})$ uzyskana została identyczna dekompozycja. Porównując wartości miar ρ_1, ρ_2 i ρ_3 dla funkcji ze zbioru F_{14} to możemy zauważyć, że różnią się miedzy sobą. Dodatkowo, różne modyfikacje funkcji f_{14} skutkują otrzymaniem innej dekompozycji przez strategię RDG3. Prawdopodobnie przyczyną jest również niedoskonałość mechanizmu, który automatycznie estymuje wartość ϵ oraz próg wielkości grup ϵ_n.

Ostatnią rozważaną grupą problemów testowych są funkcje w pełni nieseparowalne, tj. f_{15}, $(f_{15})^2$ i $\sqrt{f_{15}}$. Obie strategie wykrywają dla nich poprawnie wszystkie istniejące zależności. Modyfikacje funkcji f_{15} nie wpływają na dokładność ich dekompozycji, ponieważ nie wszystkie funkcje ze zbioru F_{15} nie posiadają zmiennych decyzyjnych, które są w pełni separowalne.

4.4.4 Koszt dekompozycji

Strategia IRRG osiąga lepszą dokładność dekompozycji niż strategia RDG3 dla w pełni i nie w pełni separowanych funkcji. W przypadku funkcji w pełni nieseparowalnych obie strategie wykryły wszystkie istniejące zależności pomiędzy zmiennymi, natomiast sytuacja dla funkcji z nakładającymi się na siebie podproblemami nie jest jednoznaczna ze względu na brak idealnej dekompozycji dla tego typu problemów. Możemy zatem stwierdzić, że w ogólności strategia IRRG dostarcza dekompozycje problemów optymalizacyjnych o wyższej jakości niż strategia RDG3. Niemniej jednak, wyższej jakości dekompozycja wiąże się także z wyższym kosztem jej uzyskania. Koszt dekompozycji mierzony jest liczbą wyliczeń funkcji przystosowania.

Tabela 4.7 przedstawia ile razy koszt dekompozycji strategią IRRG jest średnio wyższy od liczby wyliczeń funkcji przystosowania wymaganych przez strategię RDG3. Dla funkcji ze standardowego zbioru CEC'2013, strategia IRRG zużywa średnio od $3,6$ do $14,6$ razy więcej wyliczeń funkcji przystosowania niż strategia RDG3. Podobne wartości względnego kosztu dekompozycji strategią IRRG możemy

Tabela 4.7: Względny koszt dekompozycji strategią IRRG w porównaniu do strategii RDG3, gdzie bezwzględny koszt dekompozycji mierzony jest liczbą wyliczeń funkcji przystosowania

Funkcja	IRRG (średnia) / RDG3				
	Minimum	Maksimum	Mediana	Średnia	Odch. st.
f_1–f_3	6,7	13,3	13,3	11,1	3,8
f_4–f_7	4,8	8,8	7,8	7,3	1,8
f_8–f_{11}	**3,6**	5,4	4,2	4,3	0,8
f_{12}–f_{14}	4,2	**14,6**	4,4	7,7	5,9
f_{15}	4,7	4,7	4,7	4,7	Nd.
$(f_1)^2$–$(f_3)^2$	5,8	6,7	6,7	6,4	0,5
$(f_4)^2$–$(f_7)^2$	5,6	12,7	10,0	9,6	3,1
$(f_8)^2$–$(f_{11})^2$	**3,8**	9,3	6,8	6,7	2,6
$(f_{12})^2$–$(f_{14})^2$	8,3	**124,2**	12,7	48,4	65,7
$(f_{15})^2$	4,8	4,8	4,8	4,8	Nd.
$\sqrt{f_1}$–$\sqrt{f_3}$	5,4	6,7	6,7	6,3	0,7
$\sqrt{f_4}$–$\sqrt{f_7}$	5,4	12,8	7,2	8,1	3,5
$\sqrt{f_8}$–$\sqrt{f_{11}}$	**3,8**	13,5	8,2	8,4	4,1
$\sqrt{f_{12}}$–$\sqrt{f_{14}}$	8,9	**123,0**	12,8	48,2	64,8
$\sqrt{f_{15}}$	4,9	4,9	4,9	4,9	Nd.

zaobserwować dla funkcji należących do zmodyfikowanych zbiorów CEC'2013 jeżeli wykluczmy z nich funkcje z nakładającymi się na siebie podproblemami. Powodem, dla którego strategia IRRG zużywa średnio aż 124, 2 i 123 razy więcej wyliczeń funkcji przystosowania niż RDG3 odpowiednio dla funkcji $(f_{12})^2$–$(f_{14})^2$ i $\sqrt{f_{12}}$–$\sqrt{f_{14}}$, jest koszt dekompozycji zmodyfikowanych wersji funkcji f_{12} przez strategię RDG3. Zgodnie z tabelą Tabela 4.8, strategia RDG3 potrzebuje ponad ośmiokrotnie więcej wyliczeń funkcji przystosowania by zdekomponować funkcję f_{12} niż w przypadku dekomponowania funkcji $(f_{12})^2$ i $\sqrt{f_{12}}$. Koszt dekompozycji używając strategii RDG3 jest znacząco niższy w porównaniu do strategii IRRG. Potwierdzają to wyniki przeprowadzonych testów statystycznych, mianowicie testów t-Studenta z uwzględnieniem korekty Holma-Bonferroniego na poziomie istotności równym 5%.

Podobnie jak w przypadku dokładności dekompozycji, sprawdziliśmy czy po wprowadzeniu modyfikacji funkcji ze standardowego zbioru CEC'2013, koszt dekompozycji używając strategii IRRG będzie się istotnie różnić. Istotność różnic została zweryfikowana testem Wilcoxona z uwzględnieniem korekty Holma-Bonferroniego na poziomie istotności równym 5%. Jedynie dla funkcji ze zbiorów F_8 i F_{10}, różnice okazały się znaczące. Prawdopodobnie ze względu na automatyczną procedurą do estymacji wartości ϵ, która zależy od przedziału wartości dekomponowanych problemów.

Analizując wyniki przedstawione w tabeli Tabela 4.8 możemy zauważyć, że najbardziej kosztowną operacją jest dekomponowanie funkcji ze zbioru F_{12} przez strategię IRRG. Funkcje ze zbioru F_{12} są funkcjami z nakładającymi się na siebie pod-

Tabela 4.8: Koszt dekompozycji mierzony liczbą wyliczeń funkcji przystosowania

Funkcja	IRRG			RDG3
	Mediana	Średnia	Odch. st.	
f_1	4,00E+04	4,00E+04	5,03E-01	**3,00E+03**
$(f_1)^2$	4,00E+04	4,00E+04	4,97E-01	**6,92E+03**
$\sqrt{f_1}$	4,00E+04	4,00E+04	4,97E-01	**7,36E+03**
f_2	4,00E+04	4,00E+04	5,05E-01	**3,00E+03**
$(f_2)^2$	4,00E+04	4,00E+04	5,05E-01	**5,99E+03**
$\sqrt{f_2}$	4,00E+04	4,00E+04	5,04E-01	**5,99E+03**
f_3	4,00E+04	4,00E+04	5,00E-01	**5,99E+03**
$(f_3)^2$	4,00E+04	4,00E+04	4,97E-01	**5,99E+03**
$\sqrt{f_3}$	4,00E+04	4,00E+04	4,92E-01	**5,99E+03**
f_4	9,27E+04	8,69E+04	7,92E+03	**9,82E+03**
$(f_4)^2$	7,96E+04	8,41E+04	7,46E+03	**7,45E+03**
$\sqrt{f_4}$	7,97E+04	8,27E+04	6,66E+03	**9,37E+03**
f_5	7,47E+04	7,48E+04	2,96E+03	**9,81E+03**
$(f_5)^2$	7,46E+04	7,55E+04	4,79E+03	**8,58E+03**
$\sqrt{f_5}$	7,46E+04	7,57E+04	5,11E+03	**1,37E+04**
f_6	5,79E+04	5,79E+04	2,71E+02	**1,20E+04**
$(f_6)^2$	5,79E+04	5,79E+04	2,77E+02	**1,03E+04**
$\sqrt{f_6}$	5,78E+04	5,78E+04	2,90E+02	**1,08E+04**
f_7	7,65E+04	7,87E+04	5,69E+03	**9,81E+03**
$(f_7)^2$	7,69E+04	7,85E+04	4,86E+03	**6,18E+03**
$\sqrt{f_7}$	7,67E+04	7,96E+04	6,45E+03	**6,20E+03**
f_8	1,03E+05	1,04E+05	4,68E+03	**1,93E+04**
$(f_8)^2$	1,00E+05	1,01E+05	4,39E+03	**1,89E+04**
$\sqrt{f_8}$	1,05E+05	1,06E+05	6,71E+03	**7,86E+03**
f_9	7,72E+04	7,73E+04	1,26E+03	**1,91E+04**
$(f_9)^2$	7,71E+04	7,72E+04	1,32E+03	**9,33E+03**
$\sqrt{f_9}$	7,70E+04	7,72E+04	1,32E+03	**1,08E+04**
f_{10}	7,02E+04	7,03E+04	6,15E+02	**1,97E+04**
$(f_{10})^2$	6,97E+04	6,97E+04	5,01E+02	**1,83E+04**
$\sqrt{f_{10}}$	7,17E+04	7,17E+04	9,92E+02	**1,91E+04**
f_{11}	8,50E+04	8,50E+04	2,15E+03	**1,94E+04**
$(f_{11})^2$	8,53E+04	8,54E+04	1,96E+03	**9,17E+03**
$\sqrt{f_{11}}$	8,48E+04	8,53E+04	2,98E+03	**9,21E+03**
f_{12}	7,17E+05	7,26E+05	5,41E+04	**4,99E+04**
$(f_{12})^2$	7,42E+05	7,44E+05	3,98E+04	**5,99E+03**
$\sqrt{f_{12}}$	7,33E+05	7,37E+05	4,46E+04	**5,99E+03**
f_{13}	6,85E+04	6,97E+04	4,43E+03	**1,60E+04**
$(f_{13})^2$	6,83E+04	6,88E+04	3,62E+03	**5,42E+03**
$\sqrt{f_{13}}$	6,87E+04	6,92E+04	3,12E+03	**5,42E+03**
f_{14}	6,81E+04	6,82E+04	3,23E+03	**1,63E+04**
$(f_{14})^2$	6,79E+04	6,78E+04	3,00E+03	**8,18E+03**
$\sqrt{f_{14}}$	6,70E+04	6,76E+04	2,82E+03	**7,62E+03**
f_{15}	2,80E+04	2,84E+04	1,75E+03	**5,99E+03**
$(f_{15})^2$	2,82E+04	2,87E+04	2,24E+03	**5,99E+03**
$\sqrt{f_{15}}$	2,85E+04	2,93E+04	2,64E+03	**5,99E+03**

problemami. Podproblemy są dwuwymiarowe, a dwa nakładające się na siebie podproblemy współdzielą dokładnie jedną zmienną decyzyjną. Rysunek 1.4 przedstawia graf interakcji ośmiowymiarowej wersji takiego problemu. Strategia IRRG dąży do połączenia wszystkich zmiennych w jedną grupę gdy problem składa się z nakładających się na siebie podproblemów. W celu dodania zmiennych dwóch nakładających się podproblemów do jednej grupy istnieje tylko jedna możliwa interakcja pomiędzy zmiennymi, która musi zostać wykryta. Wykrycie pojedynczej interakcji dla funkcji ze zbioru F_{12} jest trudne dla strategii IRRG. Dodatkowo funkcje te składają się z wielu podproblemów. Te dwa czynniki powodują, że strategia IRRG potrzebuje wielu iteracji by przeprowadzić proces dekompozycji funkcji f_{12}, $(f_{12})^2$ i $\sqrt{f_{12}}$.

Spośród wszystkich przeprowadzonych eksperymentów, największa liczba iteracji strategii IRRG została zaraportowana podczas dekomponowania funkcji $\sqrt{f_{12}}$. Strategia IRRG potrzebowała w tym celu 169 iteracji. Z drugiej strony, strategia IRRG idealnie dekomponuje funkcje ze zbioru F_1–F_3 wykonując tylko jedną iterację. Strategia IRRG wykonała średnio $22,74 \pm 29,47$ iteracji. Niemniej jednak, gdy nie uwzględnimy skrajnych przypadków, tj. w pełni separowalnych funkcji i F_{12}, to średnia liczba iteracji maleje do wartości $19,22 \pm 4,44$. Obie wartości średnie są znacznie mniejsze niż długość rozpatrywanych problemów optymalizacyjnych. Możemy zatem stwierdzić, że typowa złożoność obliczeniowa strategii IRRG to $\mathcal{O}(n \log(n))$.

4.4.5 Wyniki optymalizacji dla standardowego warunku zatrzymania

Głównym celem dekompozycji danego problemu optymalizacyjnego jest jej późniejsze użycie podczas procesu optymalizacji. Nawet dobrej jakości dekompozycja może nie przekładać się następnie na dobre wyniki optymalizacji. Proces dekompozycji mógł kosztować zbyt dużo w kontekście całego budżetu przeznaczonego na optymalizację [94] lub wykorzystana metoda optymalizacji mogła nie być dostosowana do sposobu zbierania informacji o zależnościach lub do efektywnego wykorzystania otrzymanych danych [105]. Na przykład, architektury CBCC i CCFR2 zakładają, że dostarczone im grupy zmiennych decyzyjnych są od siebie niezależne. Dzięki takiemu założeniu, mogą one zachłannie wybierać komponent do optymalizacji co w przypadku spełnionego założenia może znacząco przyspieszyć zbieżność metody bez zwiększenia szansy utknięcia w optimum lokalnym [92, 124, 149]. Innym przykładem jest dostosowanie otrzymanej struktury problemu do używanej metody optymalizacji w ramach architektury koewolucji kooperatywnej. Wiedząc, że metoda CMA-ES potrafi skutecznie optymalizować w pełni separowalne problemy składające się do 100 zmiennych decyzyjnych [116], możemy połączyć separowalne zmienne

Tabela 4.9: Liczba zwycięstw, remisów i porażek strategii IRRG ze strategią RDG3 gdy zostaną one osadzone w tej samej architekturze koewolucji kooperatywnej (CBCC lub CCFR2) dla standardowego warunku zatrzymania

i	CBCC-IRRG vs. CBCC-RDG3			CCFR2-IRRG vs. CCFR2-RDG3		
	f_i z/r/p	$(f_i)^2$ z/r/p	$\sqrt{f_i}$ z/r/p	f_i z/r/p	$(f_i)^2$ z/r/p	$\sqrt{f_i}$ z/r/p
1–3	1/**2**/0	**3**/0/0	**3**/0/0	1/**2**/0	**3**/0/0	**3**/0/0
4–7	0/**4**/0	**3**/0/1	**3**/1/0	0/**4**/0	**3**/0/1	**3**/1/0
8–11	0/**2**/**2**	**4**/0/0	**4**/0/0	1/**3**/0	**3**/1/0	**3**/1/0
12–14	1/0/**2**	**2**/1/0	1/**2**/0	1/0/**2**	**2**/1/0	**3**/0/0
15	0/**1**/0	0/**1**/0	0/**1**/0	0/**1**/0	0/**1**/0	0/**1**/0
Suma	2/**9**/4	**12**/2/1	**11**/4/0	3/**10**/2	**11**/3/1	**12**/3/0

w większe grupy zamiast przetwarzać je oddzielnie [125]. Skoro strategia dekompozycji powinna być dopasowana do metody optymalizacji, która następnie korzysta z wyniku dekompozycji to w przeprowadzonych badaniach sprawdziliśmy skuteczność strategii IRRG i RDG3 po osadzeniu ich w dwóch różnych architekturach koewolucji kooperatywnej, mianowicie CBCC i CCFR2. W celu weryfikacji czy dekomponowanie problemu jest w ogóle przydatne podczas optymalizacji wielowymiarowych problemów optymalizacyjnych, badania wykonano także dla metody SHADE-ILS. Wszystkie różnice w otrzymanych wynikach optymalizacji zostały zweryfikowane pod kątem istotności wykonując test Wilcoxona z uwzględnieniem korekty Holma-Bonferroniego na poziomie istotności równym 5%.

Tabela 4.9 przedstawia zbiorcze porównanie metody CBCC-IRRG z metodą CBCC-RDG3 oraz metod CCFR2-IRRG i CCFR2-RDG3. W przypadku funkcji ze standardowego zbioru CEC'2013, oba porównania wskazują na podobną skuteczność rozważanych metod optymalizacji. Inaczej wygląda sytuacja dla funkcji $(f_1)^2$–$(f_{15})^2$ oraz $\sqrt{f_1}$–$\sqrt{f_{15}}$. Propozycje CBCC-IRRG i CCFR2-IRRG są znaczące skuteczniejsze od ich odpowiedników korzystających ze strategii RDG3. Możemy zatem stwierdzić, że osadzenie strategii IRRG w najnowszych architekturach koewolucji kooperatywnej jest ogólniejszą propozycją w porównaniu do osadzenia w nich strategii RDG3. Obie strategie różnią się znacząco kosztem ich uruchomienia. Niemniej jednak, wyższy koszt dekompozycji strategią IRRG pozostaje zwykle niewielkim odsetkiem całości budżetu obliczeniowego. Prowadzi to do otrzymania podobnych wyników optymalizacji dla funkcji, dla których obie strategie zwróciły podobnej dokładności dekompozycję.

Na podstawie porównania przedstawionego w tabeli Tabela 4.10 możemy stwierdzić, że skuteczność metod CBCC-IRRG i CCFR2-IRRG jest porównywalna z niewielką przewagą metody CCFR2-IRRG, która następnie została wybrana do porównania z metodą SHADE-ILS. Metoda SHADE-ILS nie wykorzystuje żadnej wie-

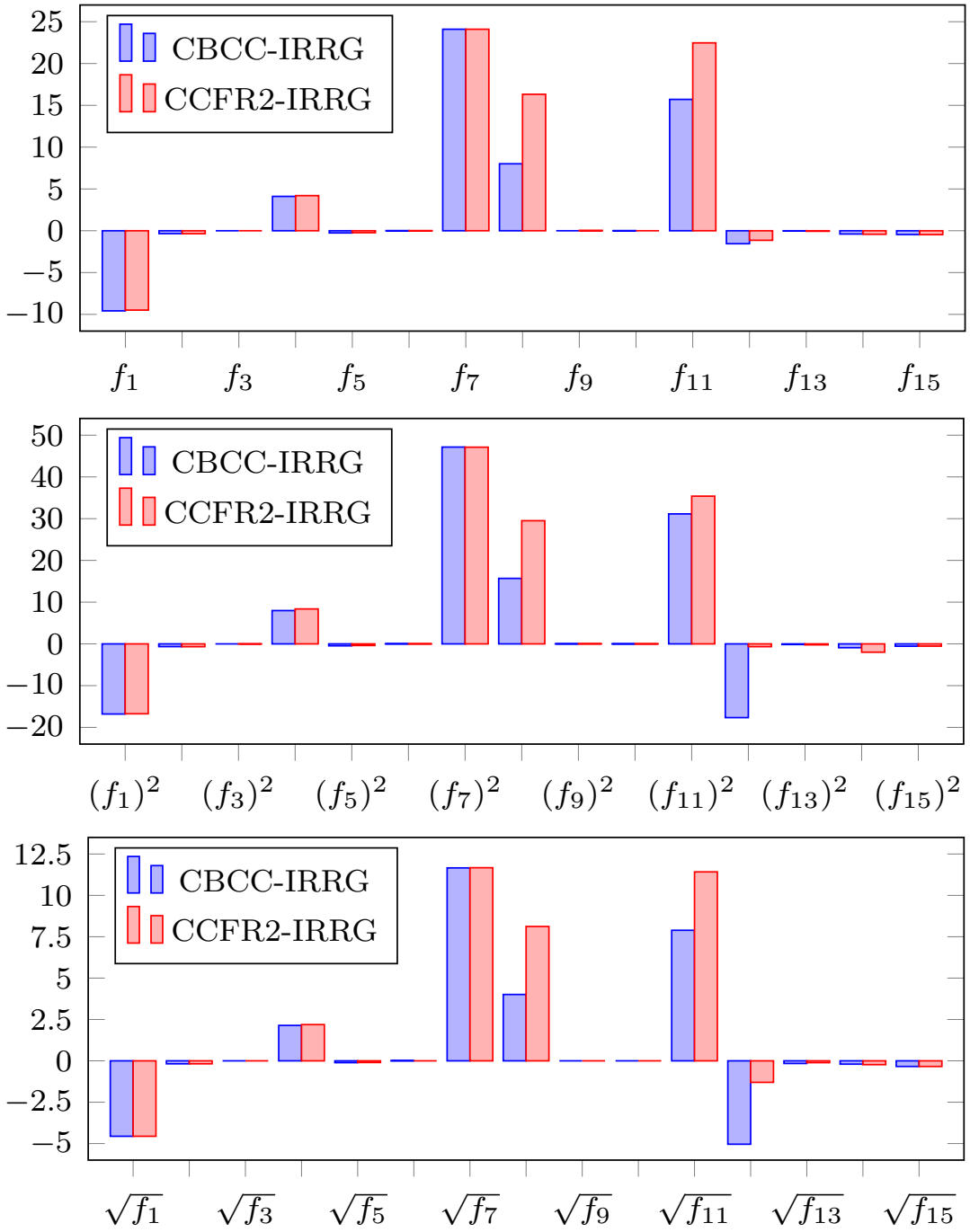
Tabela 4.10: Liczba zwycięstw, remisów i porażek metody CCFR2-IRRG z metodami CBCC-IRRG i SHADE-ILS dla standardowego warunku zatrzymania

i	CCFR2-IRRG vs. CBCC-IRRG			CCFR2-IRRG vs. SHADE-ILS		
	f_i z/r/p	$(f_i)^2$ z/r/p	$\sqrt{f_i}$ z/r/p	f_i z/r/p	$(f_i)^2$ z/r/p	$\sqrt{f_i}$ z/r/p
1–3	0/**3**/0	0/**3**/0	0/**3**/0	0/0/**3**	0/0/**3**	0/1/**2**
4–7	0/**4**/0	0/**4**/0	0/**4**/0	**3**/0/1	**3**/0/1	**3**/0/1
8–11	**2**/**2**/0	1/**3**/0	**2**/**2**/0	**3**/1/0	**3**/1/0	**2**/**2**/0
12–14	0/**3**/0	0/**3**/0	0/**3**/0	0/1/**2**	0/0/**3**	0/1/**2**
15	0/**1**/0	0/**1**/0	0/**1**/0	0/0/**1**	0/0/**1**	0/0/**1**
Suma	2/**13**/0	1/**14**/0	2/**13**/0	6/2/**7**	6/1/**8**	5/4/**6**

dzy na temat wewnętrznej struktury optymalizowanego problemu, dlatego metoda CCFR2-IRRG osiągnęła istotnie lepsze wyniki optymalizacji dla nie w pełni separowalnych funkcji ze wszystkich rozważanych zbiorów testowych (F_4–F_{11}). Z drugiej strony, dla funkcji w pełni separowalnych (F_1–F_3) i nie w pełni separowalnych (F_{15}) wyniki są przeciwne i metoda SHADE-ILS jest znacząco lepsza od CCFR2-IRRG. Dwa czynniki mogą być powodem osiągnięcia znacząco lepszych wyników przez metodę SHADE-ILS dla funkcji w pełni separowalnych. Po pierwsze, SHADE-ILS wywołuje metodę lokalnej optymalizacji MTS-L1, która została zaprojektowana do rozwiązywania w pełni separowalnych problemów [72, 82]. Po drugie, metoda SHADE-ILS posiada mechanizm restartowania swojego stanu, który jest szczególnie przydatny dla wielomodalnych problemów optymalizacyjnych gdy metoda utknie w lokalnym optimum. Każdy komponent w metodzie CCFR2-IRRG optymalizowany jest używając standardowej wersji metody CMA-ES, która nie posiada mechanizmu restartowania swojego stanu oraz potrafi skutecznie rozwiązywać problemy optymalizacyjne jeżeli posiadają one maksymalnie 100 zmiennych decyzyjnych [116]. Funkcje ze zbioru F_{15} są w pełni nieseparowalne, dlatego powstanie tylko jeden komponent do optymalizacji, którego rozmiar wynosi 1000.

Metoda SHADE-ILS, w ogólności, jest nieznacznie lepszą metodą niż CBCC-IRRG i CCFR2-IRRG. Niemniej jednak należy podkreślić fakt, że dla niektórych funkcji metody korzystające z informacji o wewnętrznej strukturze problemu dostarczonej przez strategię IRRG osiągają wyniki lepsze o wiele rzędów wielkości (Rysunek 4.2). Dodatkowo, liczba takich funkcji jest większa dla CBCC-IRRG i CCFR2-IRRG w porównaniu z przeciwną sytuacją gdy metoda SHADE-ILS osiąga wyniki lepsze o wiele rzędów wielkości.

Szczegółowe wyniki optymalizacji dla funkcji ze standardowego zbioru CEC'2013 zostały zaprezentowane w tabeli Tabela 4.11. Ich analiza pozwala na wyciągnięcie wniosków dla kilu rozważanych funkcji. Strategia IRRG, w przeciwieństwie do



Rysunek 4.2: Rzędy wielkości różnic w wynikach optymalizacji pomiędzy metodami CBCC-IRRG i CCFR2-IRRG, a metodą SHADE-ILS dla standardowego warunku zatrzymania. Dodatnie liczby wskazują na osiągnięcie lepszych wyników przez CBCC-IRRG lub CCFR2-IRRG, natomiast ujemne wartości oznaczają przewagę metody SHADE-ILS.

strategii RDG3, potrafiła idealnie zdekomponować funkcję f_8 (Tabela 4.6). W konsekwencji, metoda CCFR2-IRRG osiąga około 10^8 razy lepsze wyniki optymalizacji niż CBCC-RDG3 i CCFR2-RDG3 oraz jest około 10^{16}-krotnie lepsza od metody SHADE-ILS. Z drugiej strony, metoda CBCC-IRRG osiąga wyniki zbliżone do CBCC-RDG3 i CCFR2-RDG3, mimo że korzysta z idealnej dekompozycji funkcji f_8. Przyczyną jest najprawdopodobniej mechanizm wyboru kolejnego komponentu do optymalizacji, który różni się w architekturach CBCC i CCFR2. W architekturze CCFR2, komponenty, które początkowo miały mały wpływ na poprawę jakości najlepszego znalezionego dotychczas rozwiązania, mogą szybciej zostać wybrane do optymalizacji w kolejnym cyklu niż w przypadku architektury CBCC. Wyniki optymalizacji funkcji f_8 uzyskane przez metodę CBCC-IRRG są idealnym przykładem, że posiadanie idealnej wiedzy o wewnętrznej strukturze problemu nie musi prowadzić do uzyskania wyników optymalizacji o wysokiej jakości ze względu na ograniczenia samej metody optymalizacji. Różnice w architekturach CBCC i CCFR2 są również widoczne w wynikach optymalizacji funkcji f_{11}. Metody optymalizacji działające zgodnie z architekturą CCFR2 są znaczące lepsze.

Idealne przedstawienie zależności pomiędzy zmiennymi za pomocą rozłącznych zbiorów zmiennych jest niemożliwe dla problemów zbudowanych z nakładających się na siebie podproblemów [94, 105]. Dlatego podział zmiennych decyzyjnych na grupy dostarczony do architektur CBCC i CCFR2 wpływa znacząco na wyniki optymalizacji dla tego typu problemów, ponieważ raz utworzona struktura komponentów przez CBCC i CCFR2 pozostaje niezmienna aż do końca procesu optymalizacji. Funkcja f_{13} posiada nakładające się na siebie podproblemy, które do siebie pasują. W przypadku funkcji f_{14}, nakładające się na siebie podproblemy nie pasują do siebie. Obie rozważane w niniejszej pracy architektury koewolucji kooperatywnej zakładają, że ich komponenty są rozłączne, więc żadna zmienna decyzyjna nie może należeć do kilku komponentów jednocześnie. W przypadku pasujących do siebie podproblemów, podzielenie problemu na komponenty, które są zbliżone do podproblemów powinno być opłacalne. Z drugiej strony, taka redukcja wymiarowości może nie być przydatna gdy podproblemy nie pasują do siebie, ponieważ komponenty mogą się wzajemnie blokować w trakcie procesu optymalizacji. Ze względu na próg wielkości grup dla problemów z nakładającymi się na siebie podproblemami (ϵ_n), strategia RDG3 tworzy zazwyczaj więcej grup, które zawierają mniej zmiennych decyzyjnych, niż strategia IRRG. Z tego powodu, metody CBCC-RDG3 i CCFR2-RDG3 osiągają lepsze wyniki optymalizacji dla funkcji f_{13} niż CBCC-IRRG i CCFR2-IRRG, lecz są znacząco gorsze dla funkcji f_{14}. Funkcja f_{12} posiada także nakładające się na siebie podproblemy. Jej nakładające się podproblemy nie zostały jednak sklasyfikowane

Tabela 4.11: Wyniki optymalizacji dla funkcji ze standardowego zbioru CEC'2013 dla standardowego warunku zatrzymania

Funkcja	Statystyka	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
f_1	Mediana	7,87E-19	6,78E-19	8,14E-19	7,64E-19	**0,00E+00**
	Średnia	9,16E-19	7,41E-19	8,57E-19	8,89E-19	**2,40E-28**
	Odch. st.	3,35E-19	1,89E-19	2,77E-19	3,07E-19	**5,10E-28**
f_2	Mediana	2,35E+03	2,35E+03	2,36E+03	2,33E+03	**1,03E+03**
	Średnia	2,34E+03	2,36E+03	2,34E+03	2,33E+03	**1,06E+03**
	Odch. st.	9,64E+01	1,24E+02	9,56E+01	9,57E+01	**1,37E+02**
f_3	Mediana	2,02E+01	2,02E+01	2,04E+01	2,04E+01	**2,01E+01**
	Średnia	2,02E+01	2,02E+01	2,04E+01	2,04E+01	**2,01E+01**
	Odch. st.	1,07E-01	1,07E-01	7,48E-02	6,89E-02	**1,13E-02**
f_4	Mediana	**1,58E+04**	**2,08E+03**	**5,68E+03**	**8,74E+02**	2,52E+08
	Średnia	**1,97E+04**	**1,62E+04**	**1,45E+04**	**1,77E+04**	2,54E+08
	Odch. st.	**2,00E+04**	**3,03E+04**	**2,54E+04**	**3,46E+04**	9,98E+07
f_5	Mediana	2,42E+06	2,22E+06	2,23E+06	2,09E+06	**1,31E+06**
	Średnia	2,32E+06	2,17E+06	2,31E+06	2,06E+06	**1,29E+06**
	Odch. st.	4,97E+05	3,23E+05	3,64E+05	3,28E+05	**2,04E+05**
f_6	Mediana	**9,96E+05**	9,96E+05	**9,96E+05**	**9,96E+05**	1,04E+06
	Średnia	**1,00E+06**	1,01E+06	**9,99E+05**	**1,00E+06**	1,04E+06
	Odch. st.	**2,27E+04**	2,27E+04	**1,34E+04**	**2,25E+04**	5,64E+03
f_7	Mediana	**9,43E-22**	**9,25E-22**	**9,18E-22**	**9,19E-22**	1,07E+02
	Średnia	**9,33E-22**	**9,27E-22**	**9,38E-22**	**9,25E-22**	1,18E+03
	Odch. st.	**9,62E-23**	**7,24E-23**	**8,06E-23**	**6,11E-23**	4,94E+03
f_8	Mediana	9.13E+03	**3,50E-05**	5,91E+03	3,48E+03	8,65E+11
	Średnia	9,12E+03	**4,47E-05**	7,58E+03	3,41E+03	9,32E+11
	Odch. st.	1,37E+03	**2,99E-05**	5,23E+03	1,40E+03	5,80E+11
f_9	Mediana	**1,59E+08**	**1,50E+08**	**1,51E+08**	**1,65E+08**	**1,68E+08**
	Średnia	**1,64E+08**	**1,49E+08**	**1,56E+08**	**1,67E+08**	**1,64E+08**
	Odch. st.	**2,76E+07**	**3,19E+07**	**3,05E+07**	**3,23E+07**	**2,31E+07**
f_{10}	Mediana	**9,05E+07**	**9,05E+07**	**9,05E+07**	**9,05E+07**	9,28E+07
	Średnia	**9,10E+07**	**9,17E+07**	**9,12E+07**	**9,23E+07**	9,27E+07
	Odch. st.	**1,27E+06**	**1,88E+06**	**1,54E+06**	**2,08E+06**	4,27E+05
f_{11}	Mediana	5,49E-12	**6,70E-18**	9,64E-15	**7,11E-18**	4,99E+05
	Średnia	1,01E-10	**1,73E-17**	2,47E-12	**7,13E-18**	5,11E+05
	Odch. st.	1,56E-10	**3,36E-17**	9,11E-12	**3,47E-18**	1,37E+05
f_{12}	Mediana	8,12E+02	9,04E+02	7,24E+02	7,88E+02	**6,46E-01**
	Średnia	2,32E+03	9,06E+02	7,10E+02	7,98E+02	**6,60E+01**
	Odch. st.	7,17E+03	7,18E+01	9,01E+01	7,76E+01	**2,38E+02**
f_{13}	Mediana	1,34E+06	1,50E+06	4,65E+04	**1,75E+04**	1,04E+06
	Średnia	1,14E+06	1,30E+06	5,02E+04	**2,43E+04**	1,13E+06
	Odch. st.	5,61E+05	5,80E+05	2,40E+04	**1,66E+04**	1,00E+06
f_{14}	Mediana	1,60E+07	1,53E+07	1,47E+09	2,15E+09	**7,55E+06**
	Średnia	1,89E+07	2,01E+07	1,82E+09	2,38E+09	**7,69E+06**
	Odch. st.	8,75E+06	1,14E+07	1,52E+09	1,70E+09	**1,13E+06**
f_{15}	Mediana	2,22E+06	2,27E+06	2,20E+06	2,18E+06	**6,10E+05**
	Średnia	2,21E+06	2,29E+06	2,19E+06	2,27E+06	**7,88E+05**
	Odch. st.	2,35E+05	1,91E+05	1,72E+05	2,52E+05	**7,62E+05**

jako pasujące, ani jako niepasujące [70]. Na podstawie przeprowadzonych eksperymentów możemy stwierdzić, że wykorzystanie obu strategii dekompozycji prowadzi do uzyskania podobnych wyników optymalizacji dla funkcji f_{12}.

Wyniki optymalizacji dla zmodyfikowanych wersji funkcji ze standardowego zbioru CEC'2013 przedstawiają tabele Tabela 4.12 i Tabela 4.13. Metody CBCC-RDG3 i CCFR2-RDG3 osiągają dla nich znacząco gorsze wyniki w porównaniu do CBCC-IRRG i CCFR2-IRRG ze względu na niedokładną dekompozycję dostarczaną przez strategię RDG3. Ze względu na wykrywanie przez strategię RDG3 bezpośrednich interakcji pomiędzy zmiennymi, które w rzeczywistości nie istnieją, nie udaje się zredukować wymiarowość funkcji $(f_{13})^2$ i $\sqrt{f_{13}}$, których nadkładające się na siebie podproblemy pasują do siebie. W konsekwencji architektury CBCC i CCFR2 osiągają podobne wyniki optymalizacji po osadzeniu w nich obu rozważanych w niniejszej pracy strategii dekompozycji. Z drugiej strony, zmniejszenie liczby grup zwracanych przez strategię RDG3 powinno poprawić wyniki optymalizacji dla funkcji $(f_{14})^2$ i $\sqrt{f_{14}}$ w kontekście strategii IRRG, ponieważ zbudowane są one z nadkładających się na siebie podproblemów, które nie pasują do siebie. Wyniki się jednak nie poprawiły, ponieważ jest rzeczywiście więcej, lecz zmienne w nich są wymieszane i pochodzą z różnych podproblemów. Innymi słowy, zmienne należące do tego samego podproblemu zostały rozbite na kilka różnych grup.

Pomimo uzyskania dekompozycji o gorszej dokładności przez strategię RDG3 dla funkcji $(f_5)^2$ i $\sqrt{f_5}$, metody CBCC-RDG3 i CCFR2-RDG3 nie osiągają gorszych wyników optymalizacji niż CBCC-IRRG i CCFR2-IRRG. Co więcej, najlepsze wyniki dla funkcji zbioru $F_5 = \{f_5, (f_5)^2, \sqrt{f_5}\}$ osiąga zawsze metoda SHADE-ILS, która nie korzysta z żadnych informacji dotyczących wewnętrznej struktury optymalizowanego problemu. Funkcje ze zbioru F_5 są nie w pełni separowalne i składają się m.in. ze zmiennych, które są w pełni separowalne. Prawdopodobnie wpływ separowalnych zmiennych na jakość otrzymanego rozwiązania jest większy niż w przypadku innych funkcji tego typu (F_4, F_6 i F_7). Potwierdza to również fakt, że najlepszą metodą jest SHADE-ILS, która także osiągnęła najlepsze wyniki dla funkcji, które są w pełni separowalne (F_1–F_3).

4.4.6 Wyniki optymalizacji dla wydłużonego warunku zatrzymania

Większość rozważanych w niniejszej pracy metod optymalizacji nie posiada żadnych mechanizmów do restartu swojego stanu by zacząć przeszukiwanie przestrzeni rozwiązań od początku i tym samym umożliwić im zbiegnięcie do lokalnego optimum o lepszej jakości lub nawet w okolice globalnego optimum. Wyjątkiem jest metoda

Tabela 4.12: Wyniki optymalizacji dla funkcji ze standardowego zbioru CEC'2013, które zostały następnie podniesione do kwadratu dla standardowego warunku zatrzymania

Funkcja	Statystyka	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
$(f_1)^2$	Mediana	5,75E-37	4,90E-37	1,11E+06	1,41E+06	**0,00E+00**
	Średnia	9,18E-37	8,10E-37	7,76E+06	7,69E+06	**1,46E-53**
	Odch. st.	6,67E-37	6,99E-37	1,15E+07	1,38E+07	**6,13E-53**
$(f_2)^2$	Mediana	5,25E+06	5,53E+06	2,23E+07	2,29E+07	**1,12E+06**
	Średnia	5,34E+06	5,53E+06	2,22E+07	2,28E+07	**1,19E+06**
	Odch. st.	5,25E+05	5,09E+05	1,81E+06	2,12E+06	**2,57E+05**
$(f_3)^2$	Mediana	**4,07E+02**	4,07E+02	4,15E+02	4,16E+02	**4,02E+02**
	Średnia	**4,06E+02**	4,08E+02	4,15E+02	4,16E+02	**4,02E+02**
	Odch. st.	**5,12E+00**	3,74E+00	3,07E+00	2,13E+00	**5,73E-01**
$(f_4)^2$	Mediana	**1,53E+07**	**3,72E+07**	2,99E+17	3,46E+17	6,55E+16
	Średnia	**9,45E+08**	**3,88E+08**	2,98E+17	3,55E+17	9,06E+16
	Odch. st.	**3,52E+09**	**5,12E+08**	5,99E+16	7,44E+16	7,37E+16
$(f_5)^2$	Mediana	5,18E+12	4,24E+12	3,07E+12	2,90E+12	**1,66E+12**
	Średnia	5,44E+12	4,46E+12	3,23E+12	2,99E+12	**1,88E+12**
	Odch. st.	1,84E+12	1,46E+12	1,05E+12	8,07E+11	**5,84E+11**
$(f_6)^2$	Mediana	**9,92E+11**	**9,92E+11**	1,01E+12	1,01E+12	1,07E+12
	Średnia	**1,00E+12**	**1,01E+12**	1,01E+12	1,01E+12	1,07E+12
	Odch. st.	**3,83E+10**	**4,59E+10**	5,31E+09	6,37E+09	1,49E+10
$(f_7)^2$	Mediana	**8,08E-43**	**8,59E-43**	5,71E+10	1,16E+01	2,82E+04
	Średnia	**8,15E-43**	**8,89E-43**	5,85E+10	1,14E+01	1,16E+05
	Odch. st.	**1,54E-43**	**1,46E-43**	7,10E+09	5,94E+00	1,93E+05
$(f_8)^2$	Mediana	7,66E+07	**4,85E-07**	1,39E+19	2,38E+19	1,80E+23
	Średnia	7,88E+07	**1,20E-06**	9,90E+19	1,45E+21	3,90E+23
	Odch. st.	2,63E+07	**2,16E-06**	2,35E+20	4,60E+21	7,28E+23
$(f_9)^2$	Mediana	**2,81E+16**	**2,46E+16**	4,17E+16	3,48E+16	**2,73E+16**
	Średnia	**2,65E+16**	**2,56E+16**	4,13E+16	3,84E+16	**2,76E+16**
	Odch. st.	**9,74E+15**	**8,92E+15**	1,22E+16	1,01E+16	**5,18E+15**
$(f_{10})^2$	Mediana	**8,20E+15**	**8,20E+15**	8,27E+15	8,28E+15	8,60E+15
	Średnia	**8,32E+15**	**8,36E+15**	8,26E+15	8,31E+15	8,59E+15
	Odch. st.	**2,75E+14**	**3,19E+14**	5,90E+13	1,91E+14	7,88E+13
$(f_{11})^2$	Mediana	**2,50E-27**	**2,51E-26**	9,78E+13	1,07E+14	2,80E+11
	Średnia	**2,35E-20**	**1,29E-24**	1,04E+14	1,06E+14	3,21E+11
	Odch. st.	**7,68E-20**	**4,28E-24**	2,50E+13	2,10E+13	1,85E+11
$(f_{12})^2$	Mediana	8,29E+05	7,97E+05	9,95E+05	1,01E+06	**6,16E+04**
	Średnia	7,95E+22	8,06E+05	9,29E+05	9,32E+05	**1,75E+05**
	Odch. st.	2,27E+23	1,01E+05	1,40E+05	1,63E+05	**2,19E+05**
$(f_{13})^2$	Mediana	7,70E+11	1,84E+12	2,13E+12	2,04E+12	**9,39E+10**
	Średnia	1,31E+12	1,57E+12	2,25E+12	2,07E+12	**9,32E+11**
	Odch. st.	1,09E+12	1,01E+12	6,00E+11	4,81E+11	**1,60E+12**
$(f_{14})^2$	Mediana	3,64E+14	7,11E+14	1,71E+19	1,47E+18	**5,33E+13**
	Średnia	4,83E+14	5,85E+15	1,82E+19	3,85E+18	**5,98E+13**
	Odch. st.	3,08E+14	2,65E+16	1,33E+19	5,88E+18	**1,81E+13**
$(f_{15})^2$	Mediana	5,28E+12	5,12E+12	4,94E+12	4,99E+12	**3,25E+11**
	Średnia	5,36E+12	5,23E+12	4,87E+12	5,10E+12	**1,49E+12**
	Odch. st.	1,10E+12	1,03E+12	8,43E+11	9,27E+11	**2,99E+12**

Tabela 4.13: Wyniki optymalizacji dla funkcji ze standardowego zbioru CEC'2013, które zostały następnie spierwiastkowane dla standardowego warunku zatrzymania

Funkcja	Statystyka	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
$\sqrt{f_1}$	Mediana	8,86E-10	8,78E-10	2,71E-02	1,72E-03	**0,00E+00**
	Średnia	9,20E-10	9,16E-10	2,83E-02	1,88E-03	**2,52E-14**
	Odch. st.	1,75E-10	1,35E-10	1,67E-02	1,01E-03	**5,91E-14**
$\sqrt{f_2}$	Mediana	4,83E+01	4,83E+01	6,92E+01	6,84E+01	**3,19E+01**
	Średnia	4,84E+01	4,85E+01	6,90E+01	6,83E+01	**3,21E+01**
	Odch. st.	9,51E-01	1,00E+00	1,55E+00	1,31E+00	**1,32E+00**
$\sqrt{f_3}$	Mediana	4,49E+00	**4,49E+00**	4,52E+00	4,52E+00	**4,48E+00**
	Średnia	4,49E+00	**4,49E+00**	4,52E+00	4,52E+00	**4,48E+00**
	Odch. st.	1,26E-02	**1,20E-02**	5,65E-03	5,28E-03	**1,43E-03**
$\sqrt{f_4}$	Mediana	**1,07E+02**	**9,68E+01**	1,77E+04	6,68E+03	1,58E+04
	Średnia	**1,18E+02**	**1,05E+02**	1,51E+04	6,82E+03	1,63E+04
	Odch. st.	**6,10E+01**	**9,66E+01**	4,15E+03	5,65E+02	2,90E+03
$\sqrt{f_5}$	Mediana	1,51E+03	1,47E+03	1,43E+03	1,45E+03	**1,18E+03**
	Średnia	1,49E+03	1,45E+03	1,43E+03	1,42E+03	**1,16E+03**
	Odch. st.	1,25E+02	1,47E+02	1,21E+02	1,21E+02	**9,35E+01**
$\sqrt{f_6}$	Mediana	**9,98E+02**	**9,98E+02**	1,00E+03	1,02E+03	1,02E+03
	Średnia	**1,00E+03**	**1,00E+03**	1,01E+03	1,01E+03	1,02E+03
	Odch. st.	**9,22E+00**	**1,36E+01**	1,09E+01	1,21E+01	3,42E+00
$\sqrt{f_7}$	Mediana	**3,00E-11**	**2,99E-11**	4,83E+02	7,67E-01	1,04E+01
	Średnia	**3,01E-11**	**2,99E-11**	4,82E+02	8,07E-01	1,39E+01
	Odch. st.	**1,11E-12**	**1,48E-12**	1,10E+01	1,75E-01	9,68E+00
$\sqrt{f_8}$	Mediana	9,95E+01	**5,46E-03**	3,84E+06	4,40E+06	1,10E+06
	Średnia	9,97E+01	**7,77E-03**	4,56E+06	4,45E+06	1,02E+06
	Odch. st.	9,71E+00	**7,70E-03**	1,89E+06	7,34E+05	3,66E+05
$\sqrt{f_9}$	Mediana	**1,28E+04**	**1,25E+04**	1,48E+04	1,48E+04	**1,25E+04**
	Średnia	**1,25E+04**	**1,24E+04**	1,51E+04	1,47E+04	**1,25E+04**
	Odch. st.	**1,03E+03**	**1,22E+03**	2,44E+03	7,60E+02	**6,39E+02**
$\sqrt{f_{10}}$	Mediana	**9,52E+03**	**9,52E+03**	9,53E+03	9,53E+03	9,63E+03
	Średnia	**9,55E+03**	**9,59E+03**	9,53E+03	9,55E+03	9,62E+03
	Odch. st.	**7,85E+01**	**1,03E+02**	1,32E+01	6,65E+01	2,32E+01
$\sqrt{f_{11}}$	Mediana	4,81E-07	**2,74E-09**	2,82E+03	2,79E+03	6,76E+02
	Średnia	8,93E-06	**2,68E-09**	2,83E+03	2,77E+03	7,00E+02
	Odch. st.	1,27E-05	**7,49E-10**	1,43E+02	1,66E+02	1,31E+02
$\sqrt{f_{12}}$	Mediana	3,05E+01	3,00E+01	3,16E+01	3,17E+01	**1,12E-04**
	Średnia	1,66E+05	3,02E+01	3,09E+01	3,12E+01	**1,50E+00**
	Odch. st.	3,39E+05	1,07E+00	1,38E+00	1,16E+00	**3,35E+00**
$\sqrt{f_{13}}$	Mediana	1,23E+03	**8,38E+02**	1,18E+03	1,25E+03	**7,65E+02**
	Średnia	1,08E+03	**9,56E+02**	1,20E+03	1,23E+03	**7,46E+02**
	Odch. st.	2,63E+02	**3,07E+02**	6,48E+01	8,42E+01	**2,31E+02**
$\sqrt{f_{14}}$	Mediana	4,20E+03	5,13E+03	4,14E+04	6,10E+04	**2,67E+03**
	Średnia	4,34E+03	4,64E+03	4,20E+04	5,93E+04	**2,70E+03**
	Odch. st.	9,63E+02	9,36E+02	1,58E+04	2,97E+04	**1,77E+02**
$\sqrt{f_{15}}$	Mediana	1,51E+03	1,49E+03	1,51E+03	1,49E+03	**6,66E+02**
	Średnia	1,52E+03	1,49E+03	1,49E+03	1,50E+03	**6,87E+02**
	Odch. st.	7,99E+01	6,46E+01	7,31E+01	7,98E+01	**2,85E+02**

Tabela 4.14: Liczba zwycięstw, remisów i porażek dla wybranych par metod optymalizacji dla standardowego warunku zatrzymania

Funkcja	CBCC-IRRG vs. CBCC-RDG3	CCFR2-IRRG vs. CCFR2-RDG3	CCFR2-IRRG vs. CBCC-IRRG	CCFR2-IRRG vs. SHADE-ILS
	z/r/p	z/r/p	z/r/p	z/r/p
f_1–f_3	1/**2**/0	1/**2**/0	0/**3**/0	1/0/**2**
f_4–f_7	0/**4**/0	0/**4**/0	0/**4**/0	**3**/0/1
f_8–f_{11}	1/**3**/0	1/**3**/0	0/**4**/0	**3**/1/0
f_{12}–f_{14}	1/0/**2**	**1**/**1**/**1**	0/**3**/0	0/0/**3**
f_{15}	0/**1**/0	0/**1**/0	0/**1**/0	0/0/**1**
Suma	3/**10**/2	3/**11**/1	0/**15**/0	**7**/1/**7**

SHADE-ILS. Analiza wyników optymalizacji dla wydłużonego warunku zatrzymania pozwoli nam odpowiedzieć na pytanie czy znaczące różnice w wynikach pomiędzy metodami optymalizacji bazujących na architekturach koewolucji kooperacyjnej wynikają z różnicy w kosztach przeprowadzenia procesu dekompozycji, czy z różnej dokładności dekompozycji dostarczanej przez strategie IRRG i RDG3. Eksperymenty zostały przeprowadzone jedynie dla funkcji ze standardowego zbioru CEC'2013, ponieważ dla pozostałych problemów testowych, strategia RDG3 dostarcza dekompozycje w większości o niskiej jakości.

Na podstawie wyników zawartych w tabelach Tabela 4.9 i Tabela 4.14 możemy stwierdzić, że wynik porównania metod CBCC-IRRG i CCFR2-IRRG odpowiednio z CBCC-RDG3 i CCFR2-RDG3 zmienił się na korzyść metod korzystających z informacji dostarczanych przez strategię IRRG. Istotności różnic w wynikach optymalizacji zostały zweryfikowane testem Wilcoxona z uwzględnieniem korekty Holma-Bonferroniego na poziomie istotności równym 5%. Tabela 4.15 przedstawia wyniki optymalizacji dla wszystkich funkcji ze standardowego zbioru CEC'2013 bez ich pogrupowania względem typu problemu. Możemy z niej wyczytać, w porównaniu do tabeli Tabela 4.11, że metoda CBCC-RDG3 nie jest już dłużej znacząco lepsza od CBCC-IRRG dla funkcji f_8 i f_{11}. Podobnie, metoda CCFR2-IRRG nie jest znacząco gorsza od CCFR2-RDG3 dla funkcji f_{12} gdy rozpatrujemy wydłużony warunek zatrzymania.

Konsekwencją wydłużenia warunku zatrzymania z $3{\cdot}10^6$ do $6{\cdot}10^6$ wyliczeń funkcji przystosowania jest dwukrotne zmniejszenie udziału kosztu dekompozycji w całkowitym budżecie obliczeniowym, które jest wystarczające by metoda CBCC-IRRG nie była już ani razu znacząco gorsza od CBCC-RDG3 dla funkcji ze standardowego zbioru CEC'2013, które nie posiadają nakładających się na siebie podproblemów. Dla wydłużonego warunku zatrzymania nie ma również znaczenia, w której architekturze koewolucji kooperatywnej, CBCC lub CCFR2, zostanie osadzona strategia IRRG (Tabela 4.14, porównanie CBCC-IRRG z CCFR2-IRRG).

Tabela 4.15: Wyniki optymalizacji dla funkcji ze standardowego zbioru CEC'2013 dla wydłużonego warunku zatrzymania

Funkcja	Statystyka	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
f_1	Mediana	6,80E-19	7,88E-19	7,83E-19	1,10E-18	**0,00E+00**
	Średnia	8,07E-19	9,35E-19	9,02E-19	1,03E-18	**1,58E-29**
	Odch. st.	2,46E-19	3,43E-19	3,30E-19	3,76E-19	**5,34E-29**
f_2	Mediana	2,36E+03	2,38E+03	2,36E+03	2,41E+03	**8,63E+02**
	Średnia	2,36E+03	2,39E+03	2,37E+03	2,36E+03	**8,68E+02**
	Odch. st.	1,26E+02	1,11E+02	1,00E+02	1,19E+02	**4,40E+01**
f_3	Mediana	**2,00E+01**	**2,00E+01**	2,02E+01	2,02E+01	2,00E+01
	Średnia	**2,00E+01**	**2,00E+01**	2,02E+01	2,02E+01	2,01E+01
	Odch. st.	**8,12E-02**	**7,28E-02**	5,67E-02	5,15E-02	6,49E-03
f_4	Mediana	**1,63E-18**	**1,57E-18**	**1,54E-18**	**1,55E-18**	1,17E+08
	Średnia	**1,61E-18**	**1,57E-18**	**5,54E-12**	**1,61E-18**	1,39E+08
	Odch. st.	**1,48E-19**	**1,84E-19**	**2,76E-11**	**1,72E-19**	6,48E+07
f_5	Mediana	2,39E+06	2,33E+06	2,07E+06	2,30E+06	**1,25E+06**
	Średnia	2,37E+06	2,34E+06	2,11E+06	2,35E+06	**1,25E+06**
	Odch. st.	3,11E+05	3,79E+05	3,46E+05	3,92E+05	**1,51E+05**
f_6	Mediana	**9,96E+05**	**9,96E+05**	**9,96E+05**	9,96E+05	1,03E+06
	Średnia	**9,96E+05**	**9,96E+05**	**9,96E+05**	9,99E+05	1,03E+06
	Odch. st.	**8,76E+01**	**8,49E+01**	**1,20E+02**	1,36E+04	7,29E+03
f_7	Mediana	**8,04E-22**	**8,21E-22**	**7,90E-22**	**8,04E-22**	3,72E-04
	Średnia	**8,08E-22**	**8,23E-22**	**7,86E-22**	**8,11E-22**	1,07E-03
	Odch. st.	**7,78E-23**	**8,34E-23**	**4,73E-23**	**7,18E-23**	1,73E-03
f_8	Mediana	**4,68E-14**	**4,30E-14**	3,46E+03	3,39E+03	1,09E+11
	Średnia	**4,59E-14**	**4,39E-14**	4,20E+03	4,04E+03	2,11E+11
	Odch. st.	**5,77E-15**	**6,60E-15**	3,03E+03	2,18E+03	2,99E+11
f_9	Mediana	**1,60E+08**	**1,74E+08**	**1,62E+08**	1,63E+08	**1,44E+08**
	Średnia	**1,67E+08**	**1,63E+08**	**1,61E+08**	1,62E+08	**1,45E+08**
	Odch. st.	**3,13E+07**	**3,55E+07**	**2,81E+07**	2,65E+07	**1,42E+07**
f_{10}	Mediana	**9,05E+07**	**9,05E+07**	**9,05E+07**	**9,05E+07**	9,25E+07
	Średnia	**9,08E+07**	**9,09E+07**	**9,05E+07**	**9,07E+07**	9,24E+07
	Odch. st.	**1,05E+06**	**1,11E+06**	**1,08E+04**	**8,77E+05**	4,06E+05
f_{11}	Mediana	**5,11E-20**	5,54E-20	**5,34E-20**	**4,99E-20**	2,28E+04
	Średnia	**5,04E-20**	5,50E-20	**5,44E-20**	**4,89E-20**	2,79E+04
	Odch. st.	**7,23E-21**	7,74E-21	**1,11E-20**	**7,31E-21**	1,53E+04
f_{12}	Mediana	8,84E+02	8,62E+02	5,74E+02	8,65E+02	**9,77E-21**
	Średnia	2,12E+11	8,42E+02	5,79E+02	8,23E+02	**1,06E+00**
	Odch. st.	3,69E+11	1,01E+02	1,38E+02	9,77E+01	**1,75E+00**
f_{13}	Mediana	2,13E+05	2,12E+05	**1,86E+04**	**1,67E+04**	**1,54E+04**
	Średnia	1,65E+05	1,76E+05	**2,40E+04**	**2,08E+04**	**6,96E+04**
	Odch. st.	9,54E+04	8,32E+04	**1,51E+04**	**1,27E+04**	**8,02E+04**
f_{14}	Mediana	7,40E+06	6,10E+06	1,22E+09	2,17E+09	**4,98E+06**
	Średnia	7,43E+06	6,93E+06	1,68E+09	2,33E+09	**4,98E+06**
	Odch. st.	1,74E+06	1,75E+06	1,40E+09	1,28E+09	**1,26E+05**
f_{15}	Mediana	5,51E+05	5,51E+05	5,70E+05	5,13E+05	**1,22E+05**
	Średnia	5,62E+05	5,57E+05	5,90E+05	5,17E+05	**2,03E+05**
	Odch. st.	7,36E+04	8,12E+04	9,34E+04	5,73E+04	**2,07E+05**

Wydłużenie warunku zatrzymania nie wpłynęło mocno na porównanie metod CCFR2-IRRG i SHADE-ILS. Zmieniły się jedynie wyniki dla dwóch funkcji. Powiększyła się przewaga metody SHADE-ILS dla funkcji f_{12}, która jest teraz znacząca. Z drugiej strony, metoda CCFR2-IRRG jest teraz znacząco lepsza dla funkcji f_3 mimo, że dla standardowego warunku zatrzymania była ona znacząco gorsza od metody SHADE-ILS. Funkcja f_3 posiada wiele optimum lokalnych. Bardzo możliwe, że dla wydłużonego warunku zatrzymania, idealna dekompozycja dostarczana przez strategię IRRG ma większy wpływ na jakość otrzymanych wyników optymalizacji niż mechanizm restartowania własnego stanu będący częścią metody SHADE-ILS.

Rozdział 5

Problem praktyczny

Problemy optymalizacyjne, które składają się z nieaddytywnie separowalnych podproblemów występują również w praktyce [25]. Przykładem takiego problemu jest projektowanie przepływu z rozgałęzieniami w sieciach komputerowych i komunikacyjnych (ang. *multi-path routing problem in computer and communication networks*) [6, 18, 57, 99]. Problem projektowania przepływu z rozgałęzieniami ma swoje zastosowania m.in. w sieciach typu Internet [38], mobilne sieci ad hoc (ang. *mobile ad hoc networks*, MANETs) [84]. bezprzewodowe sieci czujnikowe (ang. *wireless sensor networks*, WSNs) [113] i elastyczne sieci optyczne (ang. *elastic optical networks*) [155]. Wyniki eksperymentów zaprezentowane pod koniec niniejszego rozdziału zostały opublikowane przez autora niniejszej pracy w czasopiśmie IEEE Transactions on Evolutionary Computation [57].

5.1 Opis problemu

W tradycyjnym podejściu do projektowania przepływu w sieciach komputerowych i komunikacyjnych rozpatruje się przepływ bez rozgałęzień, w którym każdy z jego składników przepływa tylko wzdłuż jednej trasy. Przeciwieństwem takiego podejścia jest przepływ z rozgałęzieniami, w którym składniki mogą przepływać wzdłuż wielu tras [57, 111]. Dzięki możliwości rozdzielenia każdego ze składników przepływu na kilka tras, zwiększa się przeżywalność (ang. *survivability*) rozważanej sieci, ponieważ w przypadku awarii dowolnego jej łuku (ang. *link*), część tras realizowanych dla tego samego składnika przepływu może go nie uwzględniać w swojej ścieżce. W konsekwencji, część składnika może ciągle przepływać od swojego źródła do ujścia pomimo awarii. Należy zauważyć, że awarie łuków są najczęściej występującymi awariami w sieciach [57]. Przepływ z rozgałęzieniami poprawia również jakość uwzględnianych przez sieć usług (ang. *Quality of Service*, QoS) m.in. przeciążenia

(ang. *congestion*) i przepustowość (ang. *throughput*) [6, 18, 38, 99].

Sieć możemy zamodelować za pomocą grafu $G = (V, E)$, gdzie V jest zbiorem wierzchołków (węzłów sieci), natomiast E jest zbiorem krawędzi (łuków sieci). Przepływ w tak zdefiniowanej sieci określamy za pomocą tras, które zbudowane są z łuków sieci. W ramach niniejszej pracy, rozpatrujemy przepływ wieloskładnikowy, który jest powszechnym podejściem stosowanym w modelowaniu przepływu w sieciach komputerowych i komunikacyjnych [4, 99]. Pojedynczy składnik określony jest za pomocą węzła początkowego, węzła końcowego oraz zapotrzebowania. Niech D będzie zbiorem wszystkich składników, które w całości muszą zostać uwzględnione w przepływie. Dla każdego składnika $d \in D$ jako $P(d)$ oznaczmy zbiór wszystkich dostępnych tras, wzdłuż których może on przepływać. Należy zauważyć, że zbiór $P(d)$ jest podzbiorem wszystkich możliwych tras dla d-tego składnika, ponieważ może zawierać jedynie część wszystkich możliwych tras. W ramach niniejszej pracy, zakładamy, że nie rozpatrujemy wszystkich możliwych tras dla każdego składnika, tylko ich wybraną część. Do zdefiniowania każdej trasy użyjemy binarnej stałej $\delta_{e,d,p}$, której wartość określa, czy trasa $p \in P(d)$ realizująca przepływ d-tego składnika zbudowana jest z łuku $e \in E$. W przypadku projektowania przepływu z rozgałęzieniami, rozpatrywanego w niniejszej pracy, zapotrzebowanie h_d składnika d może zostać podzielone na dowolną liczbę tras ze zbioru $P(d)$. Do określenia jaka część (procent) d-tego składnika będzie przepływać wzdłuż trasy $p \in P(d)$ służy zmienna decyzyjna $x_{d,p}$. Rozważany problem optymalizacyjny to zadanie minimalizacji średniego opóźnienia (ang. *average delay*). Funkcja celu takiego zadania minimalizacji jest powszechnie stosowana jako miara wydajności (ang. *performance metric*) sieci komputerowych [28, 99].

Zgodnie z przedstawionymi powyżej oznaczeniami, projektowanie przepływu z rozgałęzieniami można zdefiniować następująco.

zbiory

E	łuki
D	składniki
$P(d)$	dostępne trasy dla przepływów realizujących składnik d

stałe

$\delta_{e,d,p}$	=1, jeżeli trasa p realizująca przepływ d-tego składnika zbudowana jest z łuku e
	=0, w przeciwnym wypadku
h_d	zapotrzebowanie d-tego składnika
c_e	przepustowość łuku e

zmienne

$x_{d,p}$ część d-tego składnika, która przepływa wzdłuż trasy p (wartość z przedziału $[0, 1]$)

f_e sumaryczny przepływ w łuku e (nieujemna wartość rzeczywista)

cel

$$\text{minimalizacja} \quad F = \sum_{e \in E} \frac{f_e}{c_e - f_e} \tag{5.1a}$$

ograniczenia

$$f_e = \sum_{d \in D} \sum_{p \in P(d)} \delta_{e,d,p} x_{d,p} h_d, \quad e \in E \tag{5.1b}$$

$$\sum_{p \in P(d)} x_{d,p} = 1, \quad d \in D \tag{5.1c}$$

$$f_e \leq c_e, \quad e \in E \tag{5.1d}$$

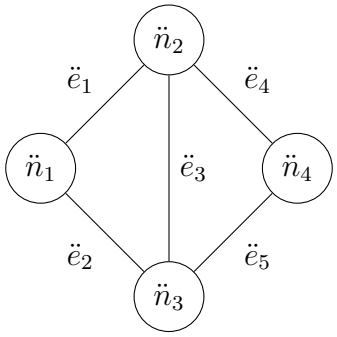
Celem (5.1a) rozważanego praktycznego problemu optymalizacyjnego jest minimalizacja średniego opóźnienia dla podanej w zadaniu sieci. Należy zauważyć, że funkcja celu jest funkcją wypukłą [57, 99]. Wartość f_e wyliczana jest na podstawie wzoru (5.1b) i jest ona sumą wszystkich części składników, które przepływają przez łuk e. Ograniczenie (5.1c) zapewnia, że całe zapotrzebowanie d-tego składnika zostanie w całości podzielone na wszystkie lub pewien podzbiór możliwych tras realizujących składnik d, natomiast ograniczenie (5.1d) ma na celu zapobiegnięcie sytuacji, w której sumaryczny przepływ w dowolnym łuku e przekroczy jego przepustowość. Pomimo, że funkcja celu jest funkcją wypukłą, tak zdefiniowany problem optymalizacyjny jest problemem NP-zupełnym [99].

5.2 Nieaddytywna separowalność

Dla problemu projektowania przepływu z rozgałęzieniami można wygenerować instancje, które są nieaddytywnie separowalne. Przykładem takiej instancji jest projektowanie przepływu z rozgałęzieniami dla sieci komputerowej przedstawionej na rysunku Rysunek 5.1, który uwzględnia przepływ dwóch składników $\ddot{d}_1$ i $\ddot{d}_2$. Węzłem początkowym składnika $\ddot{d}_1$ jest $\ddot{n}_1$, natomiast jego węzeł końcowy to $\ddot{n}_3$. Drugi składnik ($\ddot{d}_2$) ma swoje źródło w $\ddot{n}_2$ i ujście w $\ddot{n}_4$. Dla każdego składnika wybieramy po dwie trasy, wzdłuż których mogą one przepływać. Wybrane trasy są następujące.

– Pierwsza trasa dla $\ddot{d}_1$: $\ddot{e}_2$.

– Druga trasa dla $\ddot{d}_1$: $\ddot{e}_1 \rightarrow \ddot{e}_3$.



Rysunek 5.1: Przykładowa sieć komputerowa składająca się z czterech węzłów i pięciu łuków

– Pierwsza trasa dla $\ddot{d}_2$: $\ddot{e}_4$.

– Druga trasa dla $\ddot{d}_2$: $\ddot{e}_3 \rightarrow \ddot{e}_5$.

Dla tak zdefiniowanych tras, wartości sumarycznych przepływów w każdym łuku rozważanej sieci komputerowej można wyliczyć używając poniższych wzorów.

$$f_{\ddot{e}_1} = x_{\ddot{d}_1,2} h_{\ddot{d}_1} \tag{5.2a}$$

$$f_{\ddot{e}_2} = x_{\ddot{d}_1,1} h_{\ddot{d}_1} \tag{5.2b}$$

$$f_{\ddot{e}_3} = x_{\ddot{d}_1,2} h_{\ddot{d}_1} + x_{\ddot{d}_2,2} h_{\ddot{d}_2} \tag{5.2c}$$

$$f_{\ddot{e}_4} = x_{\ddot{d}_2,1} h_{\ddot{d}_2} \tag{5.2d}$$

$$f_{\ddot{e}_5} = x_{\ddot{d}_2,2} h_{\ddot{d}_2} \tag{5.2e}$$

Funkcja celu przyjmuje zatem następującą postać

$$\begin{aligned} F = & \frac{x_{\ddot{d}_1,2} h_{\ddot{d}_1}}{c_{\ddot{e}_1} - x_{\ddot{d}_1,2} h_{\ddot{d}_1}} + \frac{x_{\ddot{d}_1,1} h_{\ddot{d}_1}}{c_{\ddot{e}_2} - x_{\ddot{d}_1,1} h_{\ddot{d}_1}} \\ & + \frac{x_{\ddot{d}_1,2} h_{\ddot{d}_1} + x_{\ddot{d}_2,2} h_{\ddot{d}_2}}{c_{\ddot{e}_3} - x_{\ddot{d}_1,2} h_{\ddot{d}_1} - x_{\ddot{d}_2,2} h_{\ddot{d}_2}} \\ & + \frac{x_{\ddot{d}_2,1} h_{\ddot{d}_2}}{c_{\ddot{e}_4} - x_{\ddot{d}_2,1} h_{\ddot{d}_2}} + \frac{x_{\ddot{d}_2,2} h_{\ddot{d}_2}}{c_{\ddot{e}_5} - x_{\ddot{d}_2,2} h_{\ddot{d}_2}} \end{aligned} \tag{5.3}$$

Na podstawie ograniczenia (5.1c) możemy stwierdzić, że pary zmiennych $x_{\ddot{d}_1,1}$ i

$x_{\ddot{d}_1,2}$ oraz $x_{\ddot{d}_2,1}$ i $x_{\ddot{d}_2,2}$ oddziałują na siebie wzajemnie, ponieważ ich suma musi się równać 1. Ograniczenie (5.1d) nie uwzględnia natomiast pozostałych czterech par zmiennych. Pary $x_{\ddot{d}_1,1}$ i $x_{\ddot{d}_2,2}$ oraz $x_{\ddot{d}_1,2}$ i $x_{\ddot{d}_2,1}$ są parami zmiennych decyzyjnych, które nie oddziałują na siebie wzajemnie.

5.3 Kodowanie pojedynczego rozwiązania

Projektowanie przepływu z rozgałęzieniami jest problemem z ograniczeniami. Ograniczenie (5.1d) można wyeliminować odpowiednio modyfikując funkcję celu (5.1a). W przypadku gdy $f_e \geq c_e$ dla dowolnego łuku e to całe rozwiązanie oceniane jest w najgorszy możliwy sposób, czyli odpowiadająca mu wartość funkcji przystosowania będzie się równać dodatniej nieskończoności. Należy zauważyć, że taka modyfikacja przypomina funkcję kary, która jest powszechnym podejściem do eliminacji ograniczeń [19, 51, 67].

Ograniczenie (5.1c) można natomiast wyeliminować stosując takie kodowanie pojedynczego rozwiązania, które uniemożliwi zakodowanie rozwiązania, które nie spełnia tego ograniczenia. W tym celu, dla każdego d-tego składnika, zdefiniujmy sobie $|P(d)|-1$ ciągłych zmiennych decyzyjnych $z_{d,1}$, ..., $z_{d,|P(d)|-1}$. Niech te zmienne przyjmują wartości z zakresu $[0,1]$ (tj. $\forall_{j\in\{1,...,|P(d)|-1\}} z_{d,j} \in [0,1]$) i niech reprezentują podział przepływu d-tego składnika na trasy ze zbioru $P(d)$. Kolejny symbol π_d niech oznacza permutację zbioru $\{1, ..., |P(d)|-1\}$, taką że wskazuje ona rosnący porządek wartości $\{z_{d,1}, ..., z_{d,|P(d)|-1}\}$. Zgodnie z powyższymi oznaczeniami, wartość $x_{d,p}$ może zostać wyliczona w następujący sposób

$$x_{d,p} = \begin{cases} z_{d,p} & , p < |P(d)| \wedge \pi_d(p) = 1 \\ (1 - \sum_{j\in\{1,...,\pi_d(p)-1\}} x_{d,\pi_d^{-1}(j)}) \cdot z_{d,p} & , p < |P(d)| \wedge \pi_d(p) > 1 \\ 1 - \sum_{j\in\{1,...,|P(d)|-1\}} x_{d,j} & , p = |P(d)| \end{cases} \tag{5.4}$$

gdzie π_d^{-1} jest funkcją odwrotną do π_d. Na przykład, dla $z_{\ddot{d},1} = 0,6$, $z_{\ddot{d},2} = 0,3$ i $z_{\ddot{d},3} = 0,9$, otrzymamy $\pi_{\ddot{d}}(1) = 2$, $\pi_{\ddot{d}}(2) = 1$ i $\pi_{\ddot{d}}(3) = 3$ oraz na ich podstawie możemy wyliczyć, że $x_{\ddot{d},1} = 0,42$, $x_{\ddot{d},2} = 0,3$, $x_{\ddot{d},3} = 0,252$ i $x_{\ddot{d},4} = 0,028$. Rosnące uporządkowanie wartości $\{z_{d,1}, ..., z_{d,|P(d)|-1}\}$ ma na celu wyeliminowanie płaskich obszarów (ang. *plateaus*) z funkcji przystosowania. Stosując takie kodowanie pojedynczego rozwiązania, wszystkie rozwiązania rozważanego praktycznego problemu optymalizacyjnego da się przedstawić za pomocą $\sum_{d\in D}(|P(d)| - 1)$ zmiennych decyzyjnych.

W rozdziale 5.2 zaprezentowana została przykładowa instancja problemu skła-

dająca się z dwóch podproblemów, które mogę być nieaddytywnie separowalne lub mogą nakładać się na siebie. Używając zaproponowanego kodowania pojedynczego rozwiązania, możemy pokazać, że wewnętrzna struktura tej przykładowej instancji zależy od wartości przepustowości łuków oraz zapotrzebowań składników przepływu. Pojedyncze rozwiązanie rozważanej instancji możemy zakodować za pomocą dwóch zmiennych decyzyjnych, tj. $z_{\ddot{d}_1,1}$ i $z_{\ddot{d}_2,1}$. Skoro mamy do dyspozycji tylko dwie zmienne decyzyjne to reprezentacja rozważanej instancji może być w pełni nieaddytywnie separowalna lub w pełni nieseparowalna. W rzeczywistości, pełna nieseparowalność oznacza, że rozważana przykładowa instancja składa się z dwóch nakładających się na siebie podproblemów, w których zmienne $x_{\ddot{d}_1,2}$ i $x_{\ddot{d}_2,2}$ są zmiennymi współdzielonymi przez te dwa podproblemy.

W celu pokazania nieaddytywnie separowalnej instancji, przyjmijmy następujące wartości przepustowości łuków oraz zapotrzebowań składników przepływu:

- $c_{\ddot{e}_1} = 115$,
- $c_{\ddot{e}_2} = 120$,
- $c_{\ddot{e}_3} = 5 \cdot 10^3$,
- $c_{\ddot{e}_4} = 130$,
- $c_{\ddot{e}_5} = 125$,
- $h_{\ddot{d}_1} = 100$,
- $h_{\ddot{d}_2} = 110$.

Rozważmy następnie cztery różne scenariusze.

1. Drugi składnik przepływa w całości wzdłuż drugiej trasy ($z_{\ddot{d}_2,1} = 0$). Analizujemy podział realizacji pierwszego składnika na dwie trasy w poszukiwaniu jego optymalnej wartości (Rysunek 5.2).

2. Drugi składnik przepływa w całości wzdłuż pierwszej trasy ($z_{\ddot{d}_2,1} = 1$). Analizujemy podział realizacji pierwszego składnika na dwie trasy w poszukiwaniu jego optymalnej wartości (Rysunek 5.2).

3. Pierwszy składnik przepływa w całości wzdłuż drugiej trasy ($z_{\ddot{d}_1,1} = 0$). Analizujemy podział realizacji drugiego składnika na dwie trasy w poszukiwaniu jego optymalnej wartości (Rysunek 5.3).

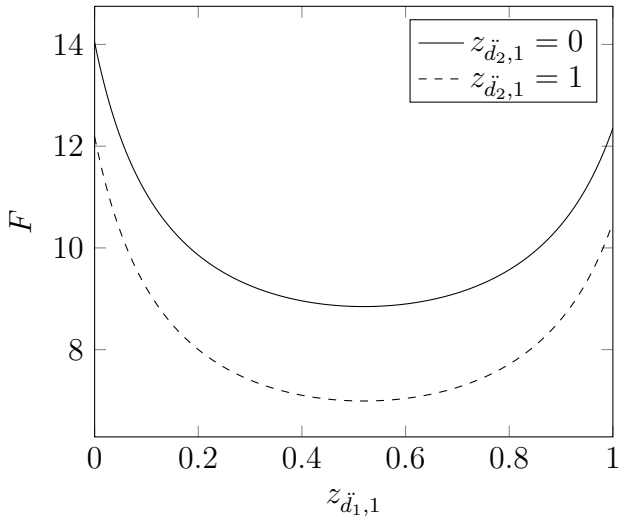
4. Pierwszy składnik przepływa w całości wzdłuż pierwszej trasy ($z_{\ddot{d}_1,1} = 1$). Analizujemy podział realizacji drugiego składnika na dwie trasy w poszukiwaniu jego optymalnej wartości (Rysunek 5.3).

Analizując rysunek Rysunek 5.2 możemy zauważyć, że optymalne rozwiązanie dla scenariusza 1 jest dokładnie takie samo jak dla scenariusza 2, tj. wartość $z_{\ddot{d}_1,1}$, dla której wartość funkcji F jest minimalna. Możemy zatem stwierdzić, że nie ma to znaczenia, czy drugi składnik przepływa w całości wzdłuż trasy składającej się ze wspólnego łuku $\ddot{e}_3$, czy przepływa w całości wzdłuż pierwszej trasy, która nie składa się z żadnego wspólnego łuku. Analogiczna sytuacja została zaprezentowana na rysunku Rysunek 5.3 dla scenariuszy 3 i 4, gdzie najlepsza wartość $z_{\ddot{d}_2,1}$ w kontekście minimalizacji F jest taka sama niezależnie od podziału pierwszego składnika. Pierwszy składnik może przepływać w całości wzdłuż trasy składającej się ze wspólnego łuku lub może zostać wybrana tylko alternatywna trasa. Podjęta decyzja nie wpływa w żaden sposób na optymalną wartość $z_{\ddot{d}_2,1}$. Powyższa analiza czterech różnych scenariuszy prowadzi do wniosku, że zmienne $z_{\ddot{d}_1,1}$ i $z_{\ddot{d}_2,1}$ są niezależne i rozważana instancja wybranego praktycznego problemu optymalizacyjnego jest nieaddytywnie separowalna.

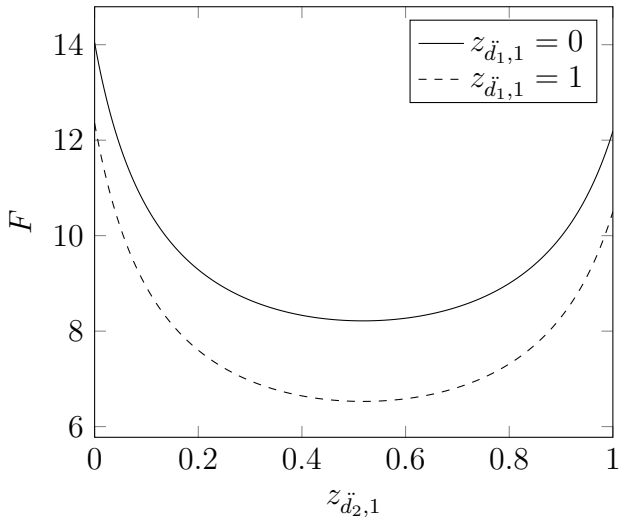
Z drugiej strony, zmieniając wartość przepustowości wspólnego łuku ($c_{\ddot{e}_3}$) z $5 \cdot 10^3$ na 200, zmienimy także wewnętrzną strukturę rozważanej instancji problemu. Zgodnie z rysunkiem Rysunek 5.4, dla tak ustalonej wartości stałej $c_{\ddot{e}_3}$, optymalna wartość zmiennej $z_{\ddot{d}_1,1}$ różni się w zależności, która z dwóch skrajnych wartości zmiennej $z_{\ddot{d}_2,1}$ zostanie wzięta pod uwagę. Na tej podstawie możemy wnioskować, że rozważana instancja problemu składa się w tym przypadku z dwóch podproblemów, które nakładają się na siebie wzajemnie.

5.4 Konfiguracja i wyniki eksperymentów

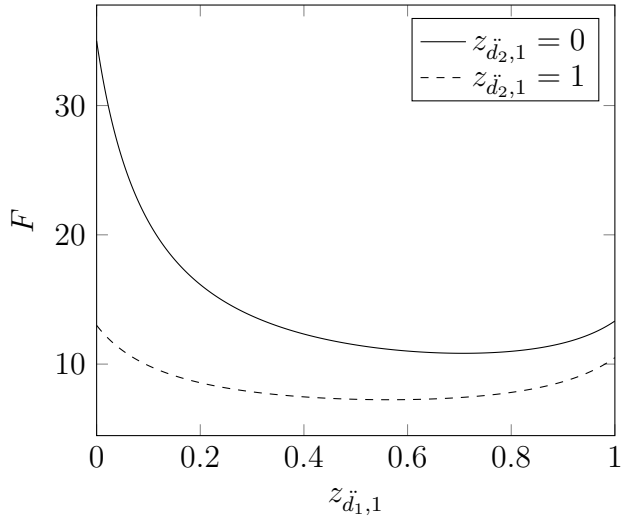
Eksperymenty dotyczące problemu praktycznego zostały przeprowadzone używając tych samych metod optymalizacji jak w przypadku problemów testowych, tj. CBCC-IRRG, CCFR2-IRRG, CBCC-RDG3, CCFR2-RDG3 i SHADE-ILS. Wszystkie wymienione metody zostały zaprojektowane do rozwiązywania wielowymiarowych problemów optymalizacyjnych i nie są one metodami bezparametrowymi. Wartości ich parametrów zostały dostrojone przez ich Autorów do problemów składających się z około 1000 zmiennych decyzyjnych. Z tego powodu, rozmiar wszystkich wygenerowanych instancji rozważanego problemu praktycznego, które następnie zostały wykorzystane w badaniach, jest zbliżony do 1000. Budżet obliczeniowy odpowiada także standardowemu warunkowi zatrzymania dla eksperymentów na



Rysunek 5.2: Średnie opóźnienie w rozważanej sieci komputerowej dla różnych wartości $z_{\ddot{d}_1,1}$ i dwóch stałych wartości $z_{\ddot{d}_2,1}$ (możliwa separowalność)



Rysunek 5.3: Średnie opóźnienie w rozważanej sieci komputerowej dla różnych wartości $z_{\ddot{d}_2,1}$ i dwóch stałych wartości $z_{\ddot{d}_1,1}$ (możliwa separowalność)



Rysunek 5.4: Średnie opóźnienie w rozważanej sieci komputerowej dla różnych wartości $z_{\ddot{d}_1,1}$ i dwóch stałych wartości $z_{\ddot{d}_2,1}$ (brak separowalności)

standardowym zbiorze CEC'2013 i wynosi $3 \cdot 10^6$ wyliczeń funkcji przystosowania [73, 82, 94, 125, 149].

5.4.1 Instancje problemu praktycznego użyte w eksperymentach

Wszystkie eksperymenty związane z problemem praktycznym zostały przeprowadzone na zbiorze 120 losowo wygenerowanych instancji problemu projektowania przepływu z rozgałęzieniami w sieciach komputerowych i komunikacyjnych. Wszystkie instancje są wielowymiarowe i przepływy, które są w ramach nich projektowane uwzględniają 63 składniki. Dla każdego składnika, utworzonych zostało 17 różnych tras, wzdłuż których każdy ze składników może przepływać. Uwzględniając powyższe założenia we wzorze $\sum_{d \in D}(|P(d)|-1)$ otrzymamy, że każda instancja może zostać zakodowana za pomocą 1008 zmiennych decyzyjnych, tj. $[z_{1,1}, z_{1,2}, ..., z_{63,15}, z_{63,16}]$.

W ramach niniejszej pracy, skupimy się na projektowaniu przepływu z rozgałęzieniami w następujących sieciach komputerowych: 104, 114, 128, 144, 162 i g120 [103]. Wartości przepustowości ich łuków zostały losowo wygenerowane z przedziału od 10^6 do $5 \cdot 10^8$. W celu zróżnicowania i pogrupowania instancji pod względem ich wewnętrznej struktury, wartości zapotrzebowań składników zostały losowo wygenerowane z czterech różnych przedziałów wartości: $[1, 50]$, $[10, 500]$, $[100, 5 \cdot 10^3]$ i $[10^3, 5 \cdot 10^4]$. Po pięć instancji zostało wygenerowanych dla każdej pary rozważanej sieci komputerowej i przedziału wartości zapotrzebowań składników. Wygenerowane instancje można podzielić na cztery grupy ze względu na przedział wartości, z któ-

Tabela 5.1: Ranking rozważanych metod optymalizacji dla problemu projektowania przepływu z rozgałęzieniami

Kategoria	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
K1	**1**	**1**	5	4	3
K2	**1**	**1**	5	4	3
K3	2	**1**	4	4	3
K4	4	2	5	3	**1**
Średnia	2.00	**1.25**	4.75	3.75	2.50

rego zostały wylosowane wartości zapotrzebowań składników.

1. K1: Wartości zapotrzebowań składników z przedziału od 1 do 50.
2. K2: Wartości zapotrzebowań składników z przedziału od 10 do 500.
3. K3: Wartości zapotrzebowań składników z przedziału od 100 do $5 \cdot 10^3$.
4. K4: Wartości zapotrzebowań składników z przedziału od 10^3 do $5 \cdot 10^4$.

Należy wspomnieć, że powyższe kategorie nie muszą reprezentować różnych struktur wewnętrznych. Niemniej jednak, intencją autora niniejszej pracy było uzyskanie różnorodnych kategorii, gdzie każda kolejna kategoria powinna zawierać instancje z większą liczbą podproblemów, które nakładają się na siebie wzajemnie.

5.4.2 Wyniki optymalizacji

Ze względu na brak wiedzy na temat idealnej macierzy interakcji każdej z rozważanych instancji, nie jest możliwe przeprowadzenie dokładnej analizy jakości dekompozycji. Możemy jedynie przypuszczać jaka była jakości dekompozycji na podstawie otrzymanych wyników optymalizacji, wierząc że lepsze wyniki optymalizacji związane są z dekompozycją o wyższej jakości.

Dla każdej kategorii sporządzony został oddzielny ranking (Tabela 5.1), który uwzględnia najlepsze rozwiązania otrzymane przez każdą z rozważanych metod optymalizacji. W celu rozstrzygnięcia, czy różnice w wynikach optymalizacji w ramach danej kategorii są znaczące przeprowadzone zostały testy znaków (ang. *sign tests*) z uwzględnieniem korekty Holma-Bonferroniego. Analiza wyników wszystkich przeprowadzonych testów statystycznych zakładała poziom istotności równy 5%. Pełne wyniki optymalizacji zostały zaprezentowane w tabelach Tabela 5.2, Tabela 5.3, Tabela 5.4 i Tabela 5.5.

Problem projektowania przepływu z rozgałęzieniami w sieciach komputerowych i komunikacyjnych w celu minimalizacji średniego opóźnienia w rozważanej sieci jest problemem, który zawiera nieaddytywnie separowalne podproblemy. Wraz ze

Tabela 5.2: Wyniki optymalizacji problemu projektowania przepływu z rozgałęzieniami dla instancji, których wartości zapotrzebowań składników są z przedziału od 1 do 50 (K1)

Instancja	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
104_1_50_0	**2,50E-05**	**2,50E-05**	5,11E-05	4,92E-05	2,71E-05
104_1_50_1	**2,53E-05**	**2,53E-05**	4,52E-05	4,51E-05	2,72E-05
104_1_50_2	**2,21E-05**	**2,21E-05**	3,00E-05	3,06E-05	2,36E-05
104_1_50_3	**2,42E-05**	**2,42E-05**	6,01E-05	3,18E-05	2,52E-05
104_1_50_4	**2,46E-05**	**2,46E-05**	3,11E-05	3,45E-05	2,60E-05
114_1_50_0	**2,26E-05**	**2,26E-05**	4,98E-05	3,03E-05	2,43E-05
114_1_50_1	**2,10E-05**	**2,10E-05**	4,29E-05	2,91E-05	2,14E-05
114_1_50_2	**1,91E-05**	**1,91E-05**	3,80E-05	2,65E-05	1,99E-05
114_1_50_3	**2,23E-05**	**2,23E-05**	3,36E-05	3,33E-05	2,28E-05
114_1_50_4	**2,13E-05**	**2,13E-05**	3,94E-05	3,92E-05	2,20E-05
128_1_50_0	**1,87E-05**	**1,87E-05**	3,77E-05	2,98E-05	1,91E-05
128_1_50_1	**1,85E-05**	**1,85E-05**	3,49E-05	2,65E-05	1,90E-05
128_1_50_2	**1,64E-05**	**1,64E-05**	3,11E-05	3,11E-05	1,69E-05
128_1_50_3	1,81E-05	**1,78E-05**	3,13E-05	3,14E-05	1,81E-05
128_1_50_4	**1,60E-05**	**1,60E-05**	2,79E-05	2,12E-05	1,63E-05
144_1_50_0	**1,84E-05**	**1,84E-05**	3,06E-05	2,47E-05	1,89E-05
144_1_50_1	**1,78E-05**	**1,78E-05**	2,90E-05	2,86E-05	1,80E-05
144_1_50_2	**1,54E-05**	**1,54E-05**	2,63E-05	2,60E-05	1,55E-05
144_1_50_3	**1,76E-05**	**1,76E-05**	2,82E-05	2,52E-05	1,83E-05
144_1_50_4	**1,56E-05**	**1,56E-05**	2,41E-05	2,35E-05	1,65E-05
162_1_50_0	**1,64E-05**	**1,64E-05**	2,09E-05	2,12E-05	1,71E-05
162_1_50_1	**1,51E-05**	**1,51E-05**	2,26E-05	2,25E-05	1,57E-05
162_1_50_2	**1,38E-05**	**1,38E-05**	2,19E-05	2,18E-05	1,42E-05
162_1_50_3	**1,52E-05**	**1,52E-05**	2,11E-05	2,15E-05	1,55E-05
162_1_50_4	**1,47E-05**	**1,47E-05**	2,36E-05	2,19E-05	1,59E-05
g120_1_50_0	**2,47E-05**	**2,47E-05**	3,25E-05	3,25E-05	2,51E-05
g120_1_50_1	**2,82E-05**	**2,82E-05**	3,49E-05	3,48E-05	2,87E-05
g120_1_50_2	**2,21E-05**	**2,21E-05**	3,22E-05	3,22E-05	2,25E-05
g120_1_50_3	**2,57E-05**	**2,57E-05**	3,27E-05	3,33E-05	2,63E-05
g120_1_50_4	**2,63E-05**	**2,63E-05**	3,25E-05	3,21E-05	2,70E-05

Tabela 5.3: Wyniki optymalizacji problemu projektowania przepływu z rozgałęzieniami dla instancji, których wartości zapotrzebowań składników są z przedziału od 10 do 500 (K2)

Instancja	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
104_10_500_0	**2,59E-04**	**2,59E-04**	4,99E-04	5,06E-04	2,70E-04
104_10_500_1	3,97E-04	**2,38E-04**	4,31E-04	4,30E-04	2,43E-04
104_10_500_2	**2,71E-04**	**2,71E-04**	3,70E-04	3,68E-04	2,86E-04
104_10_500_3	**2,50E-04**	**2,50E-04**	6,26E-04	3,26E-04	2,56E-04
104_10_500_4	3,96E-04	**2,30E-04**	2,84E-04	5,40E-04	2,34E-04
114_10_500_0	**2,34E-04**	**2,34E-04**	3,20E-04	3,10E-04	2,40E-04
114_10_500_1	**1,96E-04**	**1,96E-04**	3,87E-04	2,62E-04	2,10E-04
114_10_500_2	**2,26E-04**	**2,26E-04**	4,67E-04	3,10E-04	2,41E-04
114_10_500_3	**2,17E-04**	**2,17E-04**	4,09E-04	3,24E-04	2,21E-04
114_10_500_4	**1,99E-04**	**1,99E-04**	3,78E-04	3,78E-04	2,05E-04
128_10_500_0	**1,94E-04**	**1,94E-04**	3,85E-04	3,08E-04	2,00E-04
128_10_500_1	**1,72E-04**	**1,72E-04**	2,57E-04	2,51E-04	1,85E-04
128_10_500_2	**1,95E-04**	**1,95E-04**	3,92E-04	3,77E-04	2,03E-04
128_10_500_3	1,77E-04	**1,73E-04**	3,22E-04	3,19E-04	1,80E-04
128_10_500_4	1,45E-04	**1,44E-04**	2,91E-04	2,04E-04	1,48E-04
144_10_500_0	**1,88E-04**	**1,88E-04**	2,99E-04	2,96E-04	1,97E-04
144_10_500_1	**1,62E-04**	**1,62E-04**	2,80E-04	2,29E-04	1,70E-04
144_10_500_2	1,84E-04	**1,82E-04**	2,99E-04	2,95E-04	1,91E-04
144_10_500_3	**1,73E-04**	**1,73E-04**	2,86E-04	2,32E-04	1,76E-04
144_10_500_4	**1,43E-04**	**1,43E-04**	2,23E-04	2,27E-04	1,46E-04
162_10_500_0	**1,64E-04**	**1,64E-04**	2,12E-04	2,10E-04	1,67E-04
162_10_500_1	**1,42E-04**	**1,42E-04**	2,48E-04	2,46E-04	1,51E-04
162_10_500_2	**1,64E-04**	1,68E-04	2,59E-04	2,15E-04	1,69E-04
162_10_500_3	**1,48E-04**	**1,48E-04**	2,34E-04	2,13E-04	1,50E-04
162_10_500_4	**1,38E-04**	**1,38E-04**	2,18E-04	2,03E-04	1,41E-04
g120_10_500_0	**2,50E-04**	**2,50E-04**	3,23E-04	3,04E-04	2,55E-04
g120_10_500_1	**2,64E-04**	**2,64E-04**	3,32E-04	3,32E-04	2,70E-04
g120_10_500_2	**2,65E-04**	**2,65E-04**	3,64E-04	3,66E-04	2,70E-04
g120_10_500_3	**2,41E-04**	**2,41E-04**	3,37E-04	2,82E-04	2,46E-04
g120_10_500_4	2,49E-04	**2,44E-04**	3,10E-04	3,09E-04	2,52E-04

Tabela 5.4: Wyniki optymalizacji problemu projektowania przepływu z rozgałęzieniami dla instancji, których wartości zapotrzebowań składników są z przedziału od 100 do $5 \cdot 10^3$ (K3)

Instancja	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
104_100_5000_0	4,03E-03	4,54E-03	4,20E-03	4,18E-03	**2,29E-03**
104_100_5000_1	4,16E-03	**1,89E-03**	3,51E-03	3,51E-03	1,97E-03
104_100_5000_2	**2,25E-03**	5,11E-03	2,98E-03	2,98E-03	2,35E-03
104_100_5000_3	5,33E-03	**2,26E-03**	5,45E-03	3,09E-03	2,35E-03
104_100_5000_4	2,98E-03	5,90E-03	3,05E-03	3,02E-03	**2,39E-03**
114_100_5000_0	**2,05E-03**	2,06E-03	4,31E-03	3,11E-03	2,15E-03
114_100_5000_1	**1,74E-03**	3,75E-03	2,28E-03	2,32E-03	1,82E-03
114_100_5000_2	**1,94E-03**	3,08E-03	2,62E-03	3,67E-03	1,99E-03
114_100_5000_3	3,15E-03	**2,03E-03**	3,75E-03	2,69E-03	2,09E-03
114_100_5000_4	**1,91E-03**	1,95E-03	3,78E-03	3,72E-03	2,00E-03
128_100_5000_0	3,56E-03	**1,67E-03**	2,66E-03	2,62E-03	1,76E-03
128_100_5000_1	**1,48E-03**	1,49E-03	2,57E-03	2,58E-03	1,50E-03
128_100_5000_2	1,70E-03	**1,57E-03**	3,00E-03	2,96E-03	1,61E-03
128_100_5000_3	1,82E-03	**1,64E-03**	2,50E-03	2,73E-03	1,66E-03
128_100_5000_4	**1,43E-03**	**1,43E-03**	2,72E-03	2,27E-03	1,48E-03
144_100_5000_0	**1,66E-03**	1,67E-03	2,05E-03	2,47E-03	1,70E-03
144_100_5000_1	**1,44E-03**	**1,44E-03**	2,36E-03	1,87E-03	1,49E-03
144_100_5000_2	1,49E-03	**1,47E-03**	2,43E-03	2,34E-03	1,48E-03
144_100_5000_3	**1,64E-03**	**1,64E-03**	2,67E-03	2,05E-03	1,68E-03
144_100_5000_4	**1,42E-03**	**1,42E-03**	2,26E-03	2,11E-03	1,46E-03
162_100_5000_0	1,85E-03	**1,47E-03**	2,23E-03	1,89E-03	1,51E-03
162_100_5000_1	**1,24E-03**	**1,24E-03**	2,18E-03	1,78E-03	1,27E-03
162_100_5000_2	1,62E-03	**1,37E-03**	2,22E-03	2,13E-03	1,41E-03
162_100_5000_3	**1,46E-03**	**1,46E-03**	1,94E-03	1,92E-03	1,52E-03
162_100_5000_4	1,34E-03	**1,33E-03**	2,24E-03	1,98E-03	1,38E-03
g120_100_5000_0	2,13E-03	**2,10E-03**	2,94E-03	2,91E-03	2,19E-03
g120_100_5000_1	**2,28E-03**	**2,28E-03**	2,81E-03	2,80E-03	2,32E-03
g120_100_5000_2	**2,24E-03**	**2,24E-03**	3,17E-03	3,18E-03	2,33E-03
g120_100_5000_3	**2,39E-03**	2,40E-03	2,74E-03	2,78E-03	2,43E-03
g120_100_5000_4	**2,54E-03**	**2,54E-03**	3,21E-03	3,13E-03	2,61E-03

Tabela 5.5: Wyniki optymalizacji problemu projektowania przepływu z rozgałęzieniami dla instancji, których wartości zapotrzebowań składników są z przedziału od 10^3 do $5 \cdot 10^4$ (K4)

Instancja	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
104_1000_50000_0	5,35E-02	2,88E-02	5,22E-02	4,24E-02	**2,37E-02**
104_1000_50000_1	3,10E-02	2,90E-02	4,13E-02	4,01E-02	**2,53E-02**
104_1000_50000_2	3,43E-02	3,31E-02	3,48E-02	3,51E-02	**2,75E-02**
104_1000_50000_3	3,65E-02	3,66E-02	7,43E-02	3,82E-02	**2,95E-02**
104_1000_50000_4	2,71E-02	2,77E-02	3,05E-02	3,08E-02	**2,31E-02**
114_1000_50000_0	2,69E-02	2,77E-02	3,97E-02	2,88E-02	**2,22E-02**
114_1000_50000_1	2,63E-02	2,59E-02	4,01E-02	2,75E-02	**2,18E-02**
114_1000_50000_2	2,86E-02	2,83E-02	2,99E-02	3,03E-02	**2,37E-02**
114_1000_50000_3	3,31E-02	3,26E-02	3,95E-02	3,91E-02	**2,75E-02**
114_1000_50000_4	2,35E-02	2,31E-02	3,48E-02	3,44E-02	**1,94E-02**
128_1000_50000_0	2,24E-02	2,23E-02	3,32E-02	2,51E-02	**1,87E-02**
128_1000_50000_1	3,50E-02	2,13E-02	3,24E-02	2,98E-02	**1,80E-02**
128_1000_50000_2	2,26E-02	2,24E-02	3,50E-02	3,26E-02	**1,89E-02**
128_1000_50000_3	2,59E-02	2,67E-02	4,35E-02	4,08E-02	**2,28E-02**
128_1000_50000_4	1,76E-02	1,76E-02	2,45E-02	2,43E-02	**1,40E-02**
144_1000_50000_0	2,10E-02	3,03E-02	2,45E-02	2,51E-02	**1,82E-02**
144_1000_50000_1	1,93E-02	1,97E-02	2,64E-02	2,28E-02	**1,68E-02**
144_1000_50000_2	3,41E-02	2,02E-02	2,83E-02	2,11E-02	**1,83E-02**
144_1000_50000_3	2,52E-02	2,45E-02	3,72E-02	3,15E-02	**2,19E-02**
144_1000_50000_4	2,77E-02	1,77E-02	2,13E-02	2,09E-02	**1,44E-02**
162_1000_50000_0	2,60E-02	3,15E-02	2,22E-02	2,03E-02	**1,58E-02**
162_1000_50000_1	1,72E-02	1,70E-02	2,44E-02	2,31E-02	**1,49E-02**
162_1000_50000_2	2,75E-02	1,86E-02	2,44E-02	2,39E-02	**1,60E-02**
162_1000_50000_3	3,22E-02	2,10E-02	3,07E-02	2,56E-02	**1,98E-02**
162_1000_50000_4	1,59E-02	1,51E-02	2,09E-02	1,87E-02	**1,37E-02**
g120_1000_50000_0	3,76E-02	3,57E-02	3,32E-02	3,34E-02	**2,31E-02**
g120_1000_50000_1	3,01E-02	3,01E-02	3,36E-02	3,36E-02	**2,79E-02**
g120_1000_50000_2	3,87E-02	3,88E-02	3,30E-02	3,28E-02	**2,44E-02**
g120_1000_50000_3	4,63E-02	3,73E-02	4,35E-02	3,61E-02	**3,07E-02**
g120_1000_50000_4	3,84E-02	3,76E-02	2,87E-02	2,92E-02	**2,37E-02**

wzrostem wartości zapotrzebowań składników, coraz więcej podproblemów zaczyna się prawdopodobnie nakładać na siebie wzajemnie. Z tego powodu dekompozycja poszczególnych instancji dostarczana przez strategię IRRG coraz bardziej przestaje redukować ich wymiarowości. W konsekwencji metoda CCFR2-IRRG, która zajęła pierwsze miejsce dla pierwszych trzech kategorii, osiągnęła znacząco gorsze wyniki niż metoda SHADE-ILS dla instancji z czwartej kategorii. W przypadku rozważanego w niniejszej pracy problemu praktycznego, gdy podproblemy nakładają się na siebie wzajemnie to zazwyczaj nie pasują one do siebie. Należy zauważyć, że metoda SHADE-ILS była znacząco lepsza od pozostałych rozważanych metod optymalizacji dla testowych funkcji ze zbioru F_{14}, które także składają się nakładających się na siebie popdroblemów, które do siebie nie pasują (rozdziały 4.4.5 i 4.4.6). CBCC-IRRG jest drugą rozważaną metodą optymalizacji, która korzysta z informacji dostarczanych przez strategię IRRG. Jej średnia pozycja w rankingu jest również wysoka. Najgorsze miejsca w rankingu zajmowały głównie metody CBCC-RDG3 i CCFR2-RDG3, ponieważ osadzona w nich strategia RDG prawdopodobnie wykrywa wiele interakcji pomiędzy zmiennymi decyzyjnymi, które w rzeczywistości nie istnieją ze względu na nieaddytywny charakter separowalności.

Podsumowanie

Wielowymiarowość jest cechą wielu praktycznych problemów optymalizacyjnych o ciągłej przestrzeni przeszukiwań [46, 93, 147]. Dokładna dekompozycja tych trudnych problemów [57,89,125] może prowadzić do znalezienia ich dobrej jakości rozwiązań w sposób powtarzalny i efektywny. Aktualnie najlepsze strategie dekompozycji bazują na grupowaniu różnicowym [73, 89, 123, 125], które posiada jedną istotną wadę. W przypadku problemów, które zbudowane są z nieaddytywnie separowalnych podproblemów, grupowanie różnicowe może wykryć wiele interakcji pomiędzy zmiennymi decyzyjnymi, które w rzeczywistości nie istnieją. Głównym celem niniejszej pracy było zatem zaproponowanie nowej strategii dekompozycji, przeznaczonej dla ciągłych problemów optymalizacyjnych, dla której dokładność zwracanej dekompozycji jest wysoka zarówno dla problemów zawierających zarówno addytywnie, jak i problemów składających się z nieaddytywnie separowalnych podproblemów. Koszt dekompozycji przez nowo powstałą strategię musiał być na tyle niski by osadzenie jej w architekturze koewolucji kooperatywnej prowadziło do uzyskania porównywalnych wyników optymalizacji dla problemów zawierających addytywnie separowalne podproblemy i znacząco lepszych wyników dla problemów z nieaddytywnie separowalnymi podproblemami.

Teza pracy została uprawdopodobniona, a cel pracy został osiągnięty, czego dowodem są wyniki przeprowadzonych eksperymentów. Zaproponowana przez autora niniejszej pracy strategia IRRG idealnie zdekomponowała wszystkie rozważane problemy testowe z wyjątkiem problem z nakładającymi się na siebie podproblemami, dla których taka dekompozycja nie istnieje (rozdział 4.4.3). Rozważane problemy testowe uwzględniały oba rodzaje separowalności, addytywną i nieaddytywną. Typowa złożoność obliczeniowa strategii IRRG to $\mathcal{O}(n \log(n))$. Pomimo, że strategia RDG3 cechuje się taką samą złożonością obliczeniową, wymaga ona znacząco mniejszej liczby wyliczeń funkcji przystosowania podczas procesu dekomponowania danego problemu optymalizacyjnego (rozdział 4.4.4). Zwiększony koszt dekompozycji strategią IRRG jest jednak ciągle małą częścią całkowitego budżetu obliczeniowego. Po osadzeniu strategii IRRG w dwóch architekturach koewolucji kooperatyw-

nej, CBCC i CCFR2, wyniki optymalizacji uzyskane dla funkcji ze standardowego zbioru CEC'2013 są zbliżone do wyników osiągniętych przez metody CBCC-RDG3 i CCFR2-RDG3 (rozdziały 4.4.5 i 4.4.6). Po zmodyfikowaniu tych funkcji w celu zastąpienia addytywnej separowalności nieaddytywną, skuteczność metod CBCC-IRRG i CCFR2-IRRG pozostała bez zmian, natomiast skuteczność metod CBCC-RDG3 i CCFR2-RDG3 znacząco się obniżyła (rozdział 4.4.5). Problem projektowania przepływu z rozgałęzieniami w sieciach komputerowych i komunikacyjnych w celu minimalizacji średniego opóźnienia w rozważanej sieci jest przykładem praktycznego problemu optymalizacyjnego, który zawiera nieaddytywnie separowalne podproblemy (rozdział 5). Dla tego problemu, rozważane w niniejszej pracy architektury koewolucji kooperatywnej osiągnęły znacząco lepsze wyniki po osadzeniu w nich strategii IRRG niż w przypadku osadzenia w nich strategii RDG3 (rozdział 5.4.2).

Najważniejsze osiągnięcia rozprawy doktorskiej są następujące:

1. Opracowano i zbadano mechanizm tworzenia dwóch rankingów w autorskiej strategii IRRG, które następnie są porównywane w celu wykrycia zależności pomiędzy dwiema grupami zmiennych decyzyjnych (rozdział 3.2). Raz utworzony ranking $\mathbf{r_1}$ pozostaje niezmienny dla wszystkich rekurencyjnych wywołań funkcji SzukajInterakcji (rozdział 3.3). Jeżeli zbiory zmiennych X_1 i X_2 zostaną uznane za oddziaływające na siebie wzajemnie i zbiór X_2 podzielimy na dwa podzbiory X_2^1 i X_2^2 to dzięki temu samemu rankingowi $\mathbf{r_1}$ istnieje duża szansa, że zbiory X_1 i X_2^1 lub X_1 i X_2^2 zostaną także uznane za zależne. Tworzenie rankingu $\mathbf{r_2}$ na bieżąco pozwala natomiast na zmniejszenie całkowitego kosztu strategii IRRG (rozdział 3.3).

2. Opracowano i zbadano mechanizm uruchamiania procesu wstępnej optymalizacji w celu uzyskania wysokiej jakości rozwiązania $\mathbf{x_{hq}}$, które jest następnie wykorzystywane w każdym sprawdzeniu interakcji pomiędzy dwiema grupami zmiennych decyzyjnych (rozdział 3.4). Użycie rozwiązania $\mathbf{x_{hq}}$ może zwiększyć dokładność wynikowej dekompozycji, ponieważ zmniejsza szansę na pominięcie jakiejkolwiek z istniejących zależności pomiędzy dwoma zmiennymi (rozdział 3.3).

3. Opracowano i zbadano mechanizm inkrementacyjnego budowania macierzy zależności Θ poprzez wielokrotne wywoływanie rekursywnego grupowania rankingowego (rozdział 3.4). Każde kolejne wywołanie RRG zwiększa prawdopodobieństwo, że istniejąca zależność pomiędzy dwoma zmiennymi decyzyjnymi zostanie wreszcie wykryta. Należy również zauważyć, że kolejne wywołania RRG powinny zużywać coraz mniej wyliczeń funkcji przystosowania, ponie-

waż korzystają one ze wszystkich dotychczas wykrytych interakcji pomiędzy zmiennymi.

4. Opracowano analizę, która pokazuje, że sprawdzanie monotoniczności jest empiryczną techniką typu „linkage learning", która w teorii nigdy nie wykryje bezpośredniej zależności pomiędzy zmiennymi decyzyjnymi, które w rzeczywistości nie istnieją oraz wskazuje, że powszechna definicja separowalności zdefiniowana wzorem (1.1) może nie być adekwatna dla algorytmów ewolucyjnych (rozdział 3.1.1).

5. Opracowano dwie modyfikacje dla funkcji ze standardowego zbioru problemów CEC'2013, które przekształcają addytywną separowalność w nieaddytywną (rozdział 4.2).

6. Opracowano analizę, która wskazuje, że istnieją instancje problemu optymalizacji średniego opóźnienia w sieciach komputerowych i komunikacyjnych gdy rozpatrywany jest przepływ z rozgałęzieniami, które są nieaddytywnie separowalne (rozdział 5.2).

7. Zaimplementowano opracowaną strategię dekompozycji oraz dwa nowe zbiory problemów testowych w środowisku programistycznym. Kody źródłowe są publicznie dostępne pod adresem `https://github.com/kommar/IRRG`.

8. Przeprowadzono obszerne eksperymenty obliczeniowe opracowanej strategii dekompozycji i dokładną analizę uzyskanych wyników (rozdziały 4.4 i 5.4).

Analizując wyniki przeprowadzonych eksperymentów w odniesieniu do sposobu działania strategii IRRG, autor niniejszej pracy sugeruje następujące kierunki badań, które powinny pozwolić na rozwój wyżej wymienionej strategii dekompozycji.

1. Parametr ϵ_{sti} (maksymalna liczba kolejnych iteracji strategii IRRG bez wykrycia nowych interakcji pomiędzy zmiennymi decyzyjnymi) posiada stałą wartość przez cały przebieg strategii IRRG. Rozsądnym wydaje się zatem zaproponowanie mechanizmu zmiany wartości parametru ϵ_{sti} w zależności od stanu strategii IRRG w celu zmniejszenia jej kosztu. Na przykład, im późniejsza iteracja startegii IRRG, tym mniejsza wartość parametru ϵ_{sti}.

2. W przypadku problemów z nakładającymi się na siebie podproblemami, istotne może być znalezienie jedynie bezpośrednich zależności pomiędzy zmiennymi [47]. Strategia IRRG aktualnie grupuje razem zmienne, które także pośrednio oddziałują na siebie wzajemnie. Odpowiednia modyfikacja strategii IRRG jest

zatem niezbędna, która powinna zachować aktualną złożoność obliczeniową strategii IRRG wynoszącą $\mathcal{O}(n \log(n))$.

3. Zaproponowanie nowego mechanizmu wyznaczania ϵ podczas porównywania rankingów. Na przykładzie strategii RDG3 możemy zauważyć, że obecnie używany mechanizm jest niedoskonały i potrafi zwrócić zarówno zbyt niską, jak i zbyt wysoką wartość ϵ.

Wykorzystanie informacji dostarczanej przez strategie dekompozycji do architektur koewolucji kooperatywnej także, zdaniem autora niniejszej pracy, można ulepszyć poprzez świadomy dobór metody optymalizacji na podstawie wielkości komponentu oraz liczby w pełni separowalnych zmiennych decyzyjnych, które są jego składnikami.

Bibliografia

[1] Ieee standard for floating-point arithmetic. *IEEE Std 754-2008* (2008), s. 1–70.

[2] Akimoto, Y., Hansen, N. Cma-es and advanced adaptation mechanisms. W *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2022), GECCO '22, Association for Computing Machinery, s. 1243–1268.

[3] Arabas, J. *Wykład z algorytmów ewolucyjnych*. Wydawnictwo WNT, 2004.

[4] Assad, A. Multicommodity network flows-a survey. *Networks 8*, 1 (1978), s. 37–91.

[5] Awad, N. H., Ali, M. Z., Suganthan, P. N., Reynolds, R. G. Cade: A hybridization of cultural algorithm and differential evolution for numerical optimization. *Information Sciences 378* (2017), s. 215–241.

[6] Banner, R., Orda, A. Multipath routing algorithms for congestion minimization. *IEEE/ACM Trans. Networking 15*, 2 (2007), s. 413–424.

[7] Biedrzycki, R. On equivalence of algorithm's implementations: The cma-es algorithm and its five implementations. W *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2019), GECCO '19, s. 247–248.

[8] Biedrzycki, R. Handling bound constraints in cma-es: An experimental study. *Swarm and Evolutionary Computation 52* (2020), s. 100627.

[9] Bosman, P. A., Luong, N. H., Thierens, D. Expanding from discrete cartesian to permutation gene-pool optimal mixing evolutionary algorithms. W *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (New York, NY, USA, 2016), GECCO '16, Association for Computing Machinery, s. 637–644.

[10] Bouter, A., Alderliesten, T., Bosman, P. A. Achieving highly scalable evolutionary real-valued optimization by exploiting partial evaluations. *Evolutionary Computation 29*, 1 (2021), s. 129–155.

[11] Bouter, A., Alderliesten, T., Witteveen, C., Bosman, P. A. N. Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), GECCO '17, s. 705–712.

[12] Chen, P.-L., Peng, C.-J., Lu, C.-Y., Yu, T.-L. Two-edge graphical linkage model for DSMGA-II. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), GECCO '17, ACM, s. 745–752.

[13] Chen, S., Bolufé-Röhler, A., Montgomery, J., Hendtlass, T. An analysis on the effect of selection on exploration in particle swarm optimization and differential evolution. W *Proc. IEEE Congr. Evol. Comput. (CEC)* (2019), s. 3037–3044.

[14] Chen, W., Weise, T., Yang, Z., Tang, K. Large-scale global optimization using cooperative coevolution with variable interaction learning. W *Parallel Problem Solving from Nature, PPSN XI* (Berlin, Heidelberg, 2010), R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, Eds., Springer Berlin Heidelberg, s. 300–309.

[15] Chen, Z.-G., Zhan, Z.-H., Wang, H., Zhang, J. Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems. *IEEE Transactions on Evolutionary Computation 24*, 4 (2020), s. 708–719.

[16] Chicano, F., Ochoa, G., Whitley, L. D., Tinós, R. Dynastic Potential Crossover Operator. *Evolutionary Computation 30*, 3 (2022), s. 409–446.

[17] Chicano, F., Whitley, D., Ochoa, G., Tinós, R. Optimizing one million variable nk landscapes by hybridizing deterministic recombination and local search. W *Proceedings of the Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2017), GECCO '17, Association for Computing Machinery, s. 753–760.

[18] Cidon, I., Rom, R., Shavitt, Y. Analysis of multi-path routing. *IEEE/ACM Trans. Networking 7*, 6 (1999), s. 885–896.

[19] Coit, D. W., Smith, A. E., Tate, D. M. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS J. Comput. 8*, 2 (1996), s. 173–182.

[20] Corless, R. M., Fillion, N. *A Graduate Introduction to Numerical Methods*. Springer, 2013.

[21] Črepinšek, M., Liu, S.-H., Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR) 45*, 3 (2013), s. 1–33.

[22] Daoud, M. S., Shehab, M., Al-Mimi, H. M., Abualigah, L., Zitar, R. A., Shambour, M. K. Y. Gradient-based optimizer (gbo): A review, theory, variants, and applications. *Archives of Computational Methods in Engineering* (2022).

[23] Das, S., Suganthan, P. N. Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems.

[24] Dong, W., Chen, T., Tiňo, P., Yao, X. Scaling up estimation of distribution algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation 17*, 6 (2013), s. 797–822.

[25] Duan, Q., Shao, C., Qu, L., Shi, Y., Niu, B. When cooperative coevolution meets coordinate descent: Theoretically deeper understandings and practically better implementations. W *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), s. 721–730.

[26] Dushatskiy, A., Virgolin, M., Bouter, A., Thierens, D., Bosman, P. A. N. Parameterless gene-pool optimal mixing evolutionary algorithms. *arXiv preprint arXiv:2109.05259* (2021).

[27] Engelbrecht, A., Bosman, P., Malan, K. The influence of fitness landscape characteristics on particle swarm optimisers. *Natural Computing* (2021).

[28] Fratta, L., Gerla, M., Kleinrock, L. The flow deviation method: An approach to store-and-forward communication network design. *Networks 3*, 2 (1973), s. 97–133.

[29] Ge, H., Sun, L., Yang, X., Yoshida, S., Liang, Y. Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation. *Applied Soft Computing 36* (2015), s. 300–314.

[30] Goldman, B. W., Punch, W. F. Parameter-less population pyramid. W *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2014), GECCO '14, Association for Computing Machinery, s. 785–792.

[31] Goldman, B. W., Punch, W. F. Fast and Efficient Black Box Optimization Using the Parameter-less Population Pyramid. *Evolutionary Computation 23*, 3 (09 2015), s. 451–479.

[32] Goldman, B. W., Punch, W. F. Gray-box optimization using the parameter-less population pyramid. W *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2015), GECCO '15, Association for Computing Machinery, s. 855–862.

[33] Hadi, A., Wagdy, A., Jambi, K. Lshade-spa memetic framework for solving large-scale optimization problems. *Complex & Intelligent Systems 5* (12 2018).

[34] Hansen, N., Finck, S., Ros, R., Auger, A. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions.

[35] Hansen, N., Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput. 9*, 2 (2001), s. 159–195.

[36] Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A. Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems. *Applied Soft Computing 11*, 8 (2011), s. 5755–5769.

[37] Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A. Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems. *Applied Soft Computing 11*, 8 (2011), s. 5755–5769.

[38] He, J., Rexford, J. Toward internet-wide multipath routing. *IEEE Network 22*, 2 (2008), s. 16–21.

[39] Higham, N. J. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[40] Holland, J. H. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[41] Hopcroft, J., Tarjan, R. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM 16*, 6 (1973), s. 372–378.

[42] Horst, R., Pardalos, P., Van Thoai, N. *Introduction to Global Optimization*. Springer US, 2000.

[43] Hsu, S.-H., Yu, T.-L. Optimization by pairwise linkage detection, incremental linkage set, and restricted / back mixing: DSMGA-II. W *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), GECCO '15, ACM, s. 519–526.

[44] Hu, X.-M., He, F.-L., Chen, W.-N., Zhang, J. Cooperation coevolution with fast interdependency identification for large scale optimization. *Information Sciences 381* (2017), s. 142–160.

[45] Hussain, A., Muhammad, Y. S. Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. *Complex & Intelligent Systems 6*, 1 (2020), s. 1–14.

[46] Jia, Y.-H., Mei, Y., Zhang, M. A memetic level-based learning swarm optimizer for large-scale water distribution network optimization. W *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (2020), GECCO '20, s. 1107–1115.

[47] Jia, Y.-H., Mei, Y., Zhang, M. Contribution-based cooperative coevolution for nonseparable large-scale problems with overlapping subcomponents. *IEEE Transactions on Cybernetics 52*, 6 (2022), s. 4246–4259.

[48] Jia, Y.-H., Zhou, Y.-R., Lin, Y., Yu, W.-J., Gao, Y., Lu, L. A distributed cooperative co-evolutionary cma evolution strategy for global optimization of large-scale overlapping problems. *IEEE Access 7* (2019), s. 19821–19834.

[49] Jian, J.-R., Zhan, Z.-H., Zhang, J. Large-scale evolutionary optimization: a survey and experimental comparative study. *International Journal of Machine Learning and Cybernetics 11* (03 2020).

[50] Jong, K. A. D. Evolutionary computation: Where we are and where we're headed. *Fundam. Informaticae 35*, 1-4 (1998), s. 247–259.

[51] Jordehi, A. R. A review on constraint handling strategies in particle swarm optimisation. *Neural Computing and Applications 26*, 6 (2015), s. 1265–1275.

[52] Katoch, S., Chauhan, S. S., Kumar, V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications 80*, 5 (2021), s. 8091–8126.

[53] Kerschke, P., Trautmann, H. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evol. Comput. 27*, 1 (2019), s. 99–127.

[54] Komarnicki., M., Przewozniczek., M. Linked genes migration in island models. W *Proceedings of the 8th International Joint Conference on Computational Intelligence - ECTA, (IJCCI 2016)* (2016), INSTICC, SciTePress, s. 30–40.

[55] Komarnicki, M. M., Przewozniczek, M. W. Parameter-less population pyramid with feedback. W *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2017), GECCO '17, ACM, s. 109–110.

[56] Komarnicki, M. M., Przewozniczek, M. W., Durda, T. M. Comparative mixing for dsmga-ii. W *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2020), GECCO '20, Association for Computing Machinery, s. 708–716.

[57] Komarnicki, M. M., Przewozniczek, M. W., Kwasnicka, H., Walkowiak, K. Incremental recursive ranking grouping for large scale global optimization. *IEEE Transactions on Evolutionary Computation* (2022).

[58] Koza, J. *Genetic programming: on the programming of computers by means of natural selection*. University of Michigan Press, 1992.

[59] Kraskov, A., Stögbauer, H., Grassberger, P. Estimating mutual information. *Phys. Rev. E 69* (2004), s. 066138.

[60] Kreinovich, V., Quintana, C., Fuentes, O. Genetic algorithms: What fitness scaling is optimal? *Cybernetics and Systems 24*, 1 (1993), s. 9–26.

[61] Kullback, S., Leibler, R. A. On information and sufficiency. *Annals of Mathematical Statistics 22* (1951), s. 79–86.

[62] Kumar, A., Price, K., Mohamed, A., Hadi, A., Suganthan, P. Problem definitions and evaluation criteria for the cec 2022 special session and competition on single objective bound constrained numerical optimization, 2021.

[63] Kwasnicka, H. *Obliczenia ewolucyjne w sztucznej inteligencji*. Wrocław : Oficyna Wydaw. Politechniki Wrocławskiej, 1999.

[64] Kwasnicka, H., Przewozniczek, M. Multi population pattern searching algorithm: A new evolutionary method based on the idea of messy genetic algorithm. *IEEE Transactions on Evolutionary Computation 15*, 5 (2011), s. 715–734.

[65] LaTorre, A., Muelas, S., Peña, J. Large scale global optimization: Experimental results with mos-based hybrid algorithms. W *2013 IEEE Congress on Evolutionary Computation* (2013), s. 2742–2749.

[66] LaTorre, A., Muelas, S., Peña, J.-M. A comprehensive comparison of large scale global optimizers. *Information Sciences 316* (2015), s. 517–549. Nature-Inspired Algorithms for Large Scale Global Optimization.

[67] Li, G., Zhang, Q. Multiple penalties and multiple local surrogates for expensive constrained optimization. *IEEE Trans. Evol. Comput. 25*, 4 (2021), s. 769–778.

[68] Li, K., Fialho, A., Kwong, S., Zhang, Q. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation 18*, 1 (2014), s. 114–130.

[69] Li, W., Ding, Y., Yang, Y., Sherratt, R. S., Park, J. H., Wang, J. Parameterized algorithms of fundamental np-hard problems: a survey. *Human-centric Computing and Information Sciences 10*, 1 (2020), s. 29.

[70] Li, X., Tang, K., Omidvar, M. N., Yang, Z., Qin, K. Benchmark functions for the cec'2013 special session and competition on large-scale global optimization.

[71] Liang, J., Qu, B., Suganthan, P. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, 2013.

[72] Lin-Yu Tseng, Chun Chen. Multiple trajectory search for large scale global optimization. W *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (2008), s. 3052–3059.

[73] Ma, X., Huang, Z., Li, X., Wang, L., Qi, Y., Zhu, Z. Merged differential grouping for large-scale global optimization. *IEEE Transactions on Evolutionary Computation 26*, 6 (2022), s. 1439–1451.

[74] Ma, X., Li, X., Zhang, Q., Tang, K., Liang, Z., Xie, W., Zhu, Z. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 23*, 3 (2019), s. 421–441.

[75] Mahdavi, S., Shiri, M. E., Rahnamayan, S. Cooperative co-evolution with a new decomposition method for large-scale optimization. W *2014 IEEE Congress on Evolutionary Computation (CEC)* (2014), s. 1285–1292.

[76] Mei, Y., Chen, Q., Lensen, A., Xue, B., Zhang, M. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation* (2022).

[77] Mei, Y., Omidvar, M. N., Li, X., Yao, X. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Trans. Math. Softw. 42*, 2 (June 2016).

[78] Mei, Y., Omidvar, M. N., Li, X., Yao, X. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Trans. Math. Softw. 42*, 2 (2016).

[79] Mersmann, O., Preuss, M., Trautmann, H. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. W *Parallel Problem Solving Nat.* (2010), s. 73–82.

[80] Michalak, K. Evolutionary algorithms with machine learning models for multiobjective optimization in epidemics control. W *Evolutionary Multi-Criterion Optimization* (Cham, 2023), Springer Nature Switzerland, s. 435–448.

[81] Mohseni, N., McMahon, P. L., Byrnes, T. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 6 (2022), s. 363–379.

[82] Molina, D., LaTorre, A., Herrera, F. Shade with iterative local search for large-scale global optimization. W *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), s. 1–8.

[83] Morales, J. L., Nocedal, J. Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization".

[84] Mueller, S., Tsang, R. P., Ghosal, D. Multipath routing in mobile ad hoc networks: Issues and challenges. W *Performance Tools and Applications*

to Networked Systems (Berlin, Heidelberg, 2004), Springer Berlin Heidelberg, s. 209–234.

[85] Munetomo, M., Goldberg, D. A genetic algorithm using linkage identification by nonlinearity check. W *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)* (1999), vol. 1, s. 595–600 vol.1.

[86] Munetomo, M., Goldberg, D. E. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation 7*, 4 (1999), s. 377–398.

[87] Obuchowicz, A. *Algorytmy ewolucyjne z mutacją stabilną*. Akademicka Oficyna Wydawnicza EXIT Andrzej Lang, 2013.

[88] Olieman, C., Bouter, A., Bosman, P. A. N. Fitness-based linkage learning in the real-valued gene-pool optimal mixing evolutionary algorithm. *IEEE Transactions on Evolutionary Computation 25*, 2 (2021), s. 358–370.

[89] Omidvar, M. N., Li, X., Mei, Y., Yao, X. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation 18*, 3 (2014), s. 378–393.

[90] Omidvar, M. N., Li, X., Tang, K. Designing benchmark problems for large-scale continuous optimization. *Information Sciences 316* (2015), s. 419 – 436. Nature-Inspired Algorithms for Large Scale Global Optimization.

[91] Omidvar, M. N., Li, X., Yao, X. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. W *IEEE Congress on Evolutionary Computation* (2010), s. 1–8.

[92] Omidvar, M. N., Li, X., Yao, X. Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. W *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (2011), GECCO '11, s. 1115–1122.

[93] Omidvar, M. N., Li, X., Yao, X. A review of population-based metaheuristics for large-scale black-box global optimization—part i. *IEEE Transactions on Evolutionary Computation 26*, 5 (2022), s. 802–822.

[94] Omidvar, M. N., Yang, M., Mei, Y., Li, X., Yao, X. Dg2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation 21*, 6 (2017), s. 929–942.

[95] Palod, N., Prasad, V., Khare, R. Redefining the application of an evolutionary algorithm for the optimal pipe sizing problem. *Journal of Water and Climate Change 12*, 6 (2021), s. 2299–2313.

[96] Pelikan, M., Goldberg, D. E., Cantú-Paz, E. Boa: The bayesian optimization algorithm. W *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1* (San Francisco, CA, USA, 1999), GECCO'99, Morgan Kaufmann Publishers Inc., s. 525–532.

[97] Pelikan, M., Lin, T.-K. Parameter-less hierarchical boa. W *Genetic and Evolutionary Computation – GECCO 2004* (Berlin, Heidelberg, 2004), Springer Berlin Heidelberg, s. 24–35.

[98] ping Chen, Y., Yu, T.-L., Sastry, K., Goldberg, D. E. A survey of linkage learning techniques in genetic and evolutionary algorithms.

[99] Pioro, M., Medhi, D. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, 2004.

[100] Poláková, R., Tvrdík, J., Bujok, P. L-shade with competing strategies applied to cec2015 learning-based test suite. W *2016 IEEE Congress on Evolutionary Computation (CEC)* (2016), s. 4790–4796.

[101] Potter, M. A., De Jong, K. A. A cooperative coevolutionary approach to function optimization. W *Parallel Problem Solving from Nature — PPSN III* (Berlin, Heidelberg, 1994), Springer Berlin Heidelberg, s. 249–257.

[102] Przewozniczek, M. W. Problem encoding allowing cheap fitness computation of mutated individuals. W *2017 IEEE Congress on Evolutionary Computation (CEC)* (2017), s. 308–316.

[103] Przewozniczek, M. W., Datta, R., Walkowiak, K., Komarnicki, M. M. Splitting the fitness and penalty factor for temporal diversity increase in practical problem solving. *Expert Syst. Appl. 145* (2020), s. 113126.

[104] Przewozniczek, M. W., Frej, B., Komarnicki, M. M. On measuring and improving the quality of linkage learning in modern evolutionary algorithms applied to solve partially additively separable problems. W *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2020), GECCO '20, Association for Computing Machinery, s. 742–750.

[105] Przewozniczek, M. W., Komarnicki, M. M. Empirical linkage learning. *IEEE Transactions on Evolutionary Computation 24*, 6 (2020), s. 1097–1111.

[106] Przewozniczek, M. W., Komarnicki, M. M. Empirical problem decomposition — the key to the evolutionary effectiveness in solving a large-scale non-binary discrete real-world problem. *Applied Soft Computing 113* (2021), s. 107864.

[107] Przewozniczek, M. W., Komarnicki, M. M. Fitness caching - from a minor mechanism to major consequences in modern evolutionary computation. W *2021 IEEE Congress on Evolutionary Computation (CEC)* (2021), s. 1785–1791.

[108] Przewozniczek, M. W., Komarnicki, M. M., Bosman, P. A. N., Thierens, D., Frej, B., Luong, N. H. Hybrid linkage learning for permutation optimization with gene-pool optimal mixing evolutionary algorithms. W *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2021), GECCO '21, Association for Computing Machinery, s. 1442–1450.

[109] Przewozniczek, M. W., Komarnicki, M. M., Frej, B. Direct linkage discovery with empirical linkage learning. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2021), GECCO '21, s. 609–617.

[110] Przewozniczek, M. W., Tinós, R., Frej, B., Komarnicki, M. M. On turning black- into dark gray-optimization with the direct empirical linkage discovery and partition crossover. W *Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2022), GECCO '22, Association for Computing Machinery.

[111] Przewoźniczek, M. Nowy szablon z kodowaniem nieporządnym jako remedium na typowe wady algorytmu genetycznego, 2012.

[112] Przewoźniczek, M. W., Walkowiak, K., Sen, A., Komarnicki, M., Lechowicz, P. The transformation of the k-shortest steiner trees search problem into binary dynamic problem for effective evolutionary methods application. *Information Sciences 479* (2019), s. 1–19.

[113] Radi, M., Dezfouli, B., Bakar, K. A., Lee, M. Multipath routing in wireless sensor networks: Survey and research challenges. *Sens. 12*, 1 (2012), s. 650–685.

[114] Rakhshani, H., Idoumghar, L., Lepagnot, J., Brévilliers, M. Speed up differential evolution for computationally expensive protein structure prediction problems. *Swarm and Evolutionary Computation 50* (2019), s. 100493.

[115] Rechenberg, I. *Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, 1973.

[116] Ros, R., Hansen, N. A simple modification in cma-es achieving linear time and space complexity. W *Parallel Problem Solving from Nature – PPSN X* (2008), Springer Berlin Heidelberg, s. 296–305.

[117] Salomon, R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *Biosystems 39*, 3 (1996), s. 263–278.

[118] Slowik, A., Kwasnicka, H. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications 32*, 16 (2020), s. 12363–12379.

[119] Storn, R., Price, K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 4 (Dec 1997), s. 341–359.

[120] Sun, L., Yoshida, S., Cheng, X., Liang, Y. A cooperative particle swarm optimizer with statistical variable interdependence learning. *Information Sciences 186*, 1 (2012), s. 20–39.

[121] Sun, Y., Kirley, M., Halgamuge, S. K. Extended differential grouping for large scale global optimization with direct and indirect variable interactions. W *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), GECCO '15, s. 313–320.

[122] Sun, Y., Kirley, M., Halgamuge, S. K. Quantifying variable interactions in continuous optimization problems. *IEEE Transactions on Evolutionary Computation 21*, 2 (2017), s. 249–264.

[123] Sun, Y., Kirley, M., Halgamuge, S. K. A recursive decomposition method for large scale continuous optimization. *IEEE Transactions on Evolutionary Computation 22*, 5 (2018), s. 647–661.

[124] Sun, Y., Kirley, M., Li, X. Cooperative co-evolution with online optimizer selection for large-scale optimization. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2018), GECCO '18, s. 1079–1086.

[125] Sun, Y., Li, X., Ernst, A., Omidvar, M. N. Decomposition for large-scale optimization problems with overlapping components. W *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), s. 326–333.

[126] Sun, Y., Omidvar, M. N., Kirley, M., Li, X. Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2018), GECCO '18, s. 889–896.

[127] Tanabe, R., Fukunaga, A. Success-history based parameter adaptation for differential evolution. W *2013 IEEE Congress on Evolutionary Computation* (2013), s. 71–78.

[128] Thierens, D., Bosman, P. Optimal mixing evolutionary algorithms. W *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011* (2011), N. Krasnogor and P. Lanzi, Eds., ACM, s. 617–624.

[129] Thierens, D., Bosman, P. Predetermined versus learned linkage models. W *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2012), GECCO '12, Association for Computing Machinery, s. 289–296.

[130] Thierens, D., Bosman, P. A. Hierarchical problem solving with the linkage tree genetic algorithm. W *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation* (2013), GECCO '13, ACM, s. 877–884.

[131] Tinós, R., Przewozniczek, M. W., Whitley, D. Iterated local search with perturbation based on variables interaction for pseudo-boolean optimization. W *Proceedings of the Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2022), GECCO '22, Association for Computing Machinery, s. 296–304.

[132] Tinós, R., Whitley, D., Chicano, F. Partition crossover for pseudo-boolean optimization. W *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII* (New York, NY, USA, 2015), FOGA '15, Association for Computing Machinery, s. 137–149.

[133] Tinós, R., Whitley, D., Chicano, F., Ochoa, G. Partition crossover for continuous optimization: Epx. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2021), s. 627–635.

[134] Toint, P. L. Test problems for partially separable optimization and results for the routine pspmin. *The University of Namur, Department of Mathematics, Belgium, Tech. Rep* (1983).

[135] Umbarkar, A. J., Sheth, P. D. Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing 6*, 1 (2015).

[136] van den Bergh, F., Engelbrecht, A. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation 8*, 3 (2004), s. 225–239.

[137] van den Bergh, F., Engelbrecht, A. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation 8*, 3 (2004), s. 225–239.

[138] Wang, F., Li, Y., Zhou, A., Tang, K. An estimation of distribution algorithm for mixed-variable newsvendor problems. *IEEE Transactions on Evolutionary Computation 24*, 3 (2020), s. 479–493.

[139] Wang, F., Zhang, H., Zhou, A. A particle swarm optimization algorithm for mixed-variable optimization problems. *Swarm and Evolutionary Computation 60* (2021), s. 100808.

[140] Wang, S., Lu, Z., Wei, L., Ji, G., Yang, J. Fitness-scaling adaptive genetic algorithm with local search for solving the multiple depot vehicle routing problem. *Simulation 92*, 7 (2016), s. 601–616.

[141] Wang, Y., Liu, H., Wei, F., Zong, T., Li, X. Cooperative coevolution with formula-based variable grouping for large-scale global optimization. *Evolutionary Computation 26*, 4 (2018), s. 569–596.

[142] Wang, Z.-J., Zhan, Z.-H., Lin, Y., Yu, W.-J., Wang, H., Kwong, S., Zhang, J. Automatic niching differential evolution with contour prediction approach for multimodal optimization problems. *IEEE Transactions on Evolutionary Computation 24*, 1 (2020), s. 114–128.

[143] Weicker, K., Weicker, N. On the improvement of coevolutionary optimizers by learning variable interdependencies. W *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* (1999), vol. 3, s. 1627–1632 Vol. 3.

[144] Weise, T., Chiong, R., Tang, K. Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology 27* (09 2012).

[145] Wozniak, S., Przewozniczek, M. W., Komarnicki, M. M. Parameterless population pyramid for permutation-based problems. In *Parallel Problem*

Solving from Nature – PPSN XVI. Springer International Publishing, 2020, s. 418–430.

[146] Wu, G., Mallipeddi, R., Suganthan, P. Problem definitions and evaluation criteria for the cec 2017 competition and special session on constrained single objective real-parameter optimization, 2016.

[147] Yaman, A., Mocanu, D. C., Iacca, G., Fletcher, G., Pechenizkiy, M. Limited evaluation cooperative co-evolutionary differential evolution for large-scale neuroevolution. W *Proceedings of the Genetic and Evolutionary Computation Conference* (2018), GECCO '18, s. 569–576.

[148] Yang, M., Omidvar, M. N., Li, C., Li, X., Cai, Z., Kazimipour, B., Yao, X. Efficient resource allocation in cooperative co-evolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation 21*, 4 (2017), s. 493–505.

[149] Yang, M., Zhou, A., Li, C., Guan, J., Yan, X. Ccfr2: A more efficient cooperative co-evolutionary framework for large-scale global optimization. *Information Sciences 512* (2020), s. 64 –79.

[150] Yang, M., Zhou, A., Li, C., Yao, X. An efficient recursive differential grouping for large-scale continuous problems. *IEEE Transactions on Evolutionary Computation 25*, 1 (2021), s. 159–171.

[151] Yang, Z., Tang, K., Yao, X. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences 178*, 15 (2008), s. 2985–2999. Nature Inspired Problem-Solving.

[152] Yu, T.-L., Goldberg, D. E., Sastry, K., Lima, C. F., Pelikan, M. Dependency structure matrix, genetic algorithms, and effective recombination. *Evol. Comput. 17*, 4 (2009), s. 595–626.

[153] Zhan, Z.-H., Shi, L., Tan, K., Zhang, J. A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review* (2021).

[154] Zhou, A., Zhang, Q. Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 20*, 1 (2016), s. 52–64.

[155] Zhu, Z., Lu, W., Zhang, L., Ansari, N. Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing. *J. Lightwave Technol. 31*, 1 (2013), s. 15–22.

[156] Zielinski, A. M., Komarnicki, M. M., Przewozniczek, M. W. Parameter-less population pyramid with automatic feedback. W *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2019), GECCO '19, ACM, s. 312–313.