Uniwersytet Śląski
Wydział Informatyki i Nauki o Materiałach
Instytut Informatyki

Rozprawa doktorska

Przemysław Juszczuk

Adaptacyjny algorytm ewolucji różnicowej w rozwiązywaniu problemów teorii gier

Promotor: dr hab. Urszula Boryczka, prof UŚ

Sosnowiec, 2013 r.

"Stworzenie czegoś nowego, a jednocześnie zgodnego z tym,
co wiadomo dotychczas jest niezwykle trudne." (Richard P. Feynman)

Rodzicom, za wsparcie przez wszystkie lata nauki, dedykuję i dziękuję.

Dziękuję za pomoc i cenne uwagi Pani prof. dr hab. Urszuli Boryczce. Dziękuję również współpracownikom z Instytutu Informatyki za wsparcie i dyskusje nie tylko o nauce.

Spis treści

ROZDZIAŁ 1

Wstęp

Metaheurystyki są obliczeniowymi metodami, które umożliwiają optymalizację problemu poprzez iteracyjne poprawianie danego rozwiązania na podstawie pewnej miary jakości. Należy tutaj zaznaczyć, iż samo pojęcie algorytmów metaheurystycznych jest bardzo ogólne. Do tej kategorii zaliczane są nie tylko metody inspirowane naturą, ale także techniki takie jak przeszukiwanie lokalne. W przypadku technik zainspirowanych naturą, w algorytmie najczęściej operuje się terminami zapożyczonymi ze świata biologii. Obiekty, określane także jako osobniki, lub też agenty naśladują pewne określone zachowania ze świata zwierząt poruszając się po obszarze określanym jako przestrzeń rozwiązań. Pojedynczy osobnik jest prostą strukturą, która posiada informację o aktualnym położeniu w przestrzeni rozwiązań, określaną jako genotyp. Z kolei jakość tego rozwiązania nosi nazwę fenotypu i jest odległością danego elementu od optimum globalnego

Temat poruszany w niniejszej rozprawie dotyczy podejścia związanego z jedną z metod zaliczanych do algorytmów ewolucyjnych, a mianowicie ewolucji różnicowej. Autor stara się niejako równocześnie rozwiązać dwa problemy. Po pierwsze, opracować skuteczny i zarazem ogólny algorytm bazujący na adaptacji parametrów. Ponadto przekształca zagadnienie dotyczące wyszukiwania równowag Nasha w grach n-osobowych w problem optymalizacji funkcji ciągłej. Podstawowy algorytm ewolucji różnicowej wymaga wcześniejszego ustalenia wartości kilku parametrów. Często zdarza się, że samo dobranie odpowiednich wartości współczynników jest bardzo czasochłonne, ponieważ ich wartości zależą od rozpatrywanego problemu. Liczne modyfikacje podstawowej wersji algorytmu dotyczą najczęściej tylko jednego wybranego zagadnienia, a ich skuteczność w przypadku innych problemów jest wyraźnie niższa. Istnieje również szereg metod hybrydowych, które w założeniu łączą najlepsze cechy poszczególnych rozwiązań. Każdy dodatkowy element algorytmu prowadzi najczęściej do konieczności ustalenia kolejnych wartości parametrów. Powyższy problem prowadzi do specjalizacji danej metody, co przeczy założeniu metaheurystyk, które powinny być algorytmami ogólnymi.

Teoria gier jest ściśle związana z teorią decyzji, gdzie istotny jest problem zarządzania doborem strategii obejmujący analizę i wspomaganie procesu podejmowania decyzji. Tematyka ta znajduje szerokie zastosowanie w skomplikowanych sytuacjach decyzyjnych, gdzie występuje konflikt dwóch lub więcej stron. Przykłady zastosowania teorii gier można znaleźć m. in. w ekonomii [7], naukach społecznych [108] i biologii [64]. Rozwiązania zapożyczone z teorii gier stosowane są w różnych dziedzinach informatyki takich jak sztuczna inteligencja [133], handel elektroniczny [62], sieci komputerowe [31]. Ponadto, teoria gier związana jest ściśle z sieciami transportowymi, gdzie pojęcie równowagi Nasha może zostać użyte do określenia przepływu w sieciach [41].

Jednym z kluczowych elementów niniejszej rozprawy jest przekształcenie zagadnień związanych z teorią gier w klasyczny problem optymalizacji funkcji. W tym kontekście możliwe jest wypracowanie pewnego ogólnego podejścia, bazującego na optymalizacji ciągłej, skutecznego w całej klasie problemów. Takie przedstawienie do problemu z jednoczesnym naciskiem na opracowanie mechanizmów odciążających użytkownika od problemu doboru parametrów pozwala na zaprojektowanie ogólnej metody skutecznej w wielu zagadnieniach teorii gier. Główny cel pracy jak i cele pośrednie opisane zostały poniżej.

Teza pracy

Zaproponowane modyfikacje algorytmu ewolucji różnicowej poprawiają jakość rozwiązań w problemie wyszukiwania ϵ-równowag Nasha w grach o sumie niezerowej. Przekształcenie powyższego zagadnienia do problemu optymalizacji funkcji ciągłej pozwala na wprowadzenie ogólnej funkcji oceny umożliwiającej jego rozwiązywanie bez wyraźnego zwiększenia kosztu obliczeniowego algorytmu.

Cel pracy

Głównym celem pracy jest opracowanie i przeanalizowanie adaptacyjnej wersji algorytmu ewolucji różnicowej do generowania równowag Nasha w grach losowych, niekooperacyjnych, w postaci strategicznej o sumie niezerowej dla n graczy. Tak opracowany algorytm porównany zostanie z algorytmami prostego podziału oraz z globalną metodą Newtona. Kolejne etapy realizacji głównego celu pracy obejmują:

- Przygotowanie podstawowej wersji algorytmu ewolucji różnicowej dostosowanej do omawianego problemu, ze szczególnym uwzględnieniem opracowania reprezentacji pojedynczego osobnika.
- Przegląd literatury związanej istniejącymi algorytmami stosowanymi do wyznaczania równowag Nasha w grach w postaci strategicznej. Konieczne jest tutaj wyraźne oddzielenie algorytmów dostosowanych dla gier 2-osobowych oraz dla gier n-osobowych (gdzie $n > 2$). Jednocześnie wskazane zostaną

dwa najskuteczniejsze algorytmy dla gier n-osobowych. Jednym z podstawowych kryteriów wyboru będzie szeroki zakres problemów, dla których możliwe będzie stosowanie algorytmów. Założeniem niniejszej rozprawy jest opracowanie algorytmu ogólnego, który będzie skuteczny dla zdecydowanej większości typów gier, stąd też istotne jest porównanie metody z możliwie jak najbardziej odmiennymi podejściami.

- Opracowanie adaptacyjnego algorytmu ewolucji różnicowej wraz z modyfikacją ukierunkowaną na generowanie przybliżonych równowag Nasha z minimalną liczbą stosowanych przez graczy strategii aktywnych. Modyfikacje obejmują przede wszystkim operator mutacji, w którym wprowadzono dodatkowy czynnik skalujący. Oprócz modyfikacji obejmujących operator mutacji, zmiany wprowadzone zostaną także w mechanizmach odpowiedzialnych za wielkość populacji w poszczególnych iteracjach algorytmu.
- Badanie skuteczności proponowanego w rozprawie adaptacyjnego algorytmu ewolucji różnicowej oraz porównanie z algorytmami prostego podziału, globalną metodą Newtona oraz podstawową wersją ewolucji różnicowej. Badania eksperymentalne obejmować będą kilka aspektów. Po pierwsze, powinny zostać wskazane słabe strony istniejących algorytmów oraz wykazana przewaga proponowanego algorytmu. Kolejne zagadnienie związane będzie z powtarzalnością wyników uzyskiwanych przez poszczególne algorytmy, a także liczbą strategii aktywnych dostępnych w generowanych rozwiązaniach.

- Ostatni z celów nadrzędnych obejmował będzie wybranie bardziej złożonych problemów związanych z wyszukiwaniem punktów równowag w grach a następnie wykazanie skuteczności proponowanego algorytmu w innych zagadnieniach z teorii gier.

Rozprawa składa się z 9 rozdziałów i podzielona została na dwie części. Pierwsza część teoretyczna to rozdziały 2-5. Opisane w nich zostały podstawy teoretyczne zaproponowanego w rozprawie algorytmu, a także jego najpopularniejsze modyfikacje. Rozdział 5 stanowi przegląd istniejących rozwiązań dla gier 2 oraz n osobowych. Część badawcza pracy poprzedzona jest szczegółowym opisem algorytmu zaproponowanego w rozprawie. Przedstawione rozwiązanie jest porównane z istniejącymi algorytami a opracowane wyniki badań mają na celu potwierdzenie skuteczności wprowadzonych modyfikacji.

W rozdziale 2 opisany został podstawowy algorytm ewolucji różnicowej. Przedstawione zostały mechanizmy, na jakich opiera się sam algorytm. Położono także szczególny nacisk na operator mutacji i potencjalne możliwości jego modyfikacji. W rozdziale tym przedstawiono także najpopularniejsze modyfikacje ewolucji różnicowej związane z optymalizacją dyskretną oraz dynamiczną.

Rozdział 3 zawiera zestawienie hybrydowych wariantów ewolucji różnicowej. Warianty związane ze zmianami wartości parametrów podzielone zostały na dwie

kategorie. Pierwsza z nich dotyczy dynamicznej zmiany parametrów, w której to współczynniki modyfikowane są w kolejnych iteracjach algorytmów zgodnie z ustalonym wcześniej wzorem. Z kolei druga grupa modyfikacji dotyczy adaptacyjności parametrów, w której to zmiany następują na podstawie cech charakterystycznych populacji w poprzednich iteracjach algorytmu.

W rozdziale 4 zaprezentowane zostały podstawowe definicje związane z teorią gier. Podana została definicja gry oraz macierzy wypłat. Poruszany jest także aspekt punktu równowagi i jego wpływ na wypłatę graczy podaną w jednostkach użyteczności. Ze względu na bardzo dużą liczbę zagadnień i problemów w teorii gier, w rozdziale tym wyraźnie wskazane zostaną problemy jakich dotyczy niniejsza rozprawa. Mnogość typów równowag wymaga ograniczenia do jednego tylko problemu określanego jako wyszukiwanie mieszanej równowagi Nasha. Zagadnienie to stanowi również pewien punkt wyjściowy do rozważań przedstawionych w rozdziale 9.

Rozdział 5 zawiera opisy najpopularniejszych algorytmów związanych z zagadnieniem wyszukiwania równowag Nasha. Ta część pracy podzielona została na dwa podrozdziały, w których metody stosowane w grach 2-osobowych oddzielone zostały od reszty zagadnień.

Rozdział 6 to szczegółowy opis autorskiego algorytmu stosowanego w omawianym problemie. Zagadnienia poruszane w tej części rozprawy dotyczą szczegółowego opisu parametrów operatora selektywnej mutacji. Przybliżona zostaje także metoda wyboru podzbioru strategii aktywnych ze zbioru strategii poszczególnych graczy określana jako próbkowanie przestrzeni rozwiązań. Istotną częścią rozdziału jest także opis ogólnej funkcji oceny stosowanej w algorytmie.

Rozdział 7 w całości poświęcony jest opracowaniu zbiorów testowych a następnie badaniom eksperymentalnym proponowanego algorytmu. W sumie zestawienie obejmuje 4 algorytmy a przebadane zostanie szerokie spektrum zagadnień. Problem wyszukiwania równowag Nasha jest tematem bardzo złożonym, dlatego eksperymenty oraz ich wyniki opisywane są sekwencyjnie. Obejmują one czas generowania rozwiązania, liczbę znalezionych rozwiązań, liczbę strategii aktywnych w jednym rozwiązaniu, a także dokładność rozwiązań. W celu potwierdzenia wniosków przedstawionych w rozprawie badania obejmują kilka typów gier (ze względu na objętość, część wyników przedstawiona zostanie w dodatku A).

W rozdziale 8 poruszane są zagadnienia związane z wyszukiwaniem równowag Nasha z dodatkowymi właściwościami. Przedstawiona została modyfikacja proponowanej funkcji oceny. Następnie ukazana jest możliwość zastosowania adaptacyjnego algorytmu ewolucji różnicowej w dwóch kolejnych problemach opartych na punktach równowagi. Ważnym elementem tego rozdziału jest wykazanie, iż pomimo wyraźnie większej trudności omawianych problemów, metoda przedstawiona w rozprawie umożliwia znalezienie rozwiązań bez istotnego zwiększenia narzutu obliczeniowego.

Rozprawa zakończona jest krótkim podsumowaniem obejmującym komentarz do poszczególnych etapów badań oraz celów pracy. Uwagi ze strony autora wskazują także dalsze kierunki rozwoju badań. Do rozprawy dołączony jest dodatek A zawierający szczegółowe wyniki badań dotyczące dwóch typów gier nieujętych w rozprawie.

ROZDZIAŁ 2

Ewolucja różnicowa jako technika populacyjna

W teorii obliczeń problem optymalizacyjny jest to problem obliczeniowy, którego rozwiązanie polega na znalezieniu największej bądź najmniejszej wartości pewnego parametru, który spełnia określoną własność. Poszukiwany parametr najczęściej określany jest jako funkcja kosztu. Zwykłe metody optymalizacji często zawodzą w przypadku funkcji wielowymiarowych i nieregularnych. Większość naturalnych problemów optymalizacyjnych, w tym problemów o dużym znaczeniu praktycznym, to problemy NP-trudne. Przy założeniu, że hipoteza $P \neq NP$ jest prawdziwa, dokładne rozwiązanie tych problemów jest nieakceptowalne czasowo. W kontekście teorii algorytmów aproksymacyjnych, problemy NP-trudne są ogromnie zróżnicowane. Rozwiązania niektórych z nich można przybliżać z pewną dokładnością, ale istnieją też takie, dla których takie przybliżenie nie jest możliwe. Dla powyższych zadań często istnieje skuteczny algorytm, który nie daje jednak gwarancji znalezienia rozwiązania optymalnego, a tylko w przybliżeniu optymalnego.

Pierwsze metody określane jako metaheurystyki związane były z aproksymacją stochastyczną i opisane zostały już w roku 1951 [100]. Aproksymacja stochastyczna należy do stochastycznych algorytmów aproksymacyjnych i był to jeden z pierwszych algorytmów opartych na pewnych zmiennych losowych. Rozwinięciem powyższej metody był algorytm Kiefer-Wolfowitza opisany rok później w [81]. Jedna z najważniejszych metod opartych na adaptacji oraz ewolucji to strategie ewolucyjne [124]. Podejście to opiera się na zastosowaniu operatora mutacji oraz selekcji w celu przeszukiwania obszaru potencjalnych rozwiązań.

Kolejnym ważnym etapem rozwoju metaheurystyk inspirowanych naturą było podejście opisane jako programowanie ewolucyjne [49] oraz programowanie genetyczne [128] związane z automatycznym tworzeniem programów komputerowych. Pojawienie się programowania genetycznego poprzedzone zostało opracowaniem

algorytmu genetycznego w 1975 roku [72]. Można uznać, iż algorytm genetyczny stał się podstawą do tworzenia bardziej złożonych systemów naśladujących naturę. Pomimo prawie 40 letniej historii, metoda ta wciąż uznawana jest za jedną z najlepszych a liczba jej modyfikacji stale rośnie. Jednym z ostatnich trendów w tej dziedzinie jest rozwijanie koncepcji kwantowych algorytmów genetycznych i możliwości ich zastosowań [5]. W kontekście niniejszej rozprawy należy również wspomnieć o modyfikacji algorytmu genetycznego uwzględniającej elementy teorii gier, gdzie pary osobników wewnątrz populacji grają w jednoetapową grę w postaci strategicznej, a wygrany przeniesiony zostaje do kolejnej generacji [35].

Niezwykle istotnym algorytmem z punktu widzenia podstaw matematycznych jest algorytm symulowanego wyżarzania [53] bazujący na pewnych procesach fizycznych dotyczących schładzania. Jest to rozwinięcie metody Monte Carlo opublikowanej po raz pierwszy w roku 1953 [97]. Na przestrzeni kolejnych lat wspomniane techniki były intensywnie rozwijane, a równocześnie wprowadzono szereg dalszych metod opartych na biomimetyźmie, czyli inspirowanych procesami biologicznymi. Wśród nich warto wymienić algorytmy mrowiskowe [40], algorytmy memetyczne [102], czy też optymalizację stadną cząsteczek [42].

Podejście związane z metaheurystykami rozwija się niejako jednocześnie w dwóch kierunkach. Pierwszy z nich, to wprowadzanie coraz to nowszych i bardziej złożonych algorytmów odzwierciedlających zachowanie organizmów ze świata zwierząt, czy też roślin. Do tych zagadnień można zaliczyć systemy pszczele [79], algorytm robaczków świętojańskich [147], algorytm nietoperzy [148]. Zdecydowana większość tych algorytmów opiera się na kopiowaniu określonych interakcji pomiędzy zwierzętami lub roślinami a następnie na przekształceniu ich w elementy działania algorytmu.

Wszystkie powyższe techniki opierają się prostym schemacie algorytmu ewolucyjnego, w którym to populacja pewnych osobników podlega zmianom w kolejnych chwilach czasu. Zmiany te dotyczą modyfikacji genotypu osobnika oraz przekształcenia go w fenotyp na podstawie pewnej funkcji oceny. Ogólny schemat algorytmu ewolucyjnego zaproponowany w pracy [98] przedstawiony został w (alg. 1)

Algorytm 1: Algorytm ewolucyjny

begin

1. $t = 0$;
2. Utwórz populację początkową $P(t)$;
3. Oceń $P(t)$;
4. **while** *not warunek zakończenia* **do**
5. $t = t + 1$;
6. Wybierz $P(t)$ z $P(t-1)$;
7. Zmień $P(t)$;
8. Oceń $P(t)$;

W klasycznym algorytmie ewolucyjnym stosowana jest jedna populacja P, która modyfikowana jest w kolejnych iteracjach t. W każdej iteracji algorytmu stosowana jest ocena osobników, a następnie najlepiej przystosowane jednostki wybierane są do kolejnego pokolenia. Cały proces powtarzany jest określoną liczbę razy. Powyższy schemat dotyczy właściwie wszystkich technik opartych na ewolucji osobników.

Ewolucja różnicowa (ang. *Differential Evolution* DE) to technika ewolucyjna opracowana przez K. Price'a i R. Storna, opisana szczegółowo w [117]. Mimo iż ewolucja różnicowa jest heurystyką ewolucyjną bazującą na populacji osobników, to jednak dość znacznie rożni się od innych metod zaliczanych do tej grupy. Pierwotnym zastosowaniem ewolucji różnicowej były zadania o ciągłej dziedzinie funkcji przystosowania. Stąd też wywodzi się najczęściej stosowana reprezentacja osobników dla tej techniki, czyli wektor liczb zmiennoprzecinkowych (liczba elementów w wektorze jest równa liczbie wymiarów przestrzeni poszukiwań). Ewolucję różnicową zastosowano z dużą skutecznością do wielu problemów rzeczywistych, wśród których wymienić można np. projektowanie filtrów [129], analizę obrazu [46, 68], grupowanie [85], projektowanie systemów [39, 86], szeregowanie zadań [94] czy uczenie rozmytych sieci kognitywnych [51]. Pomimo dużej skuteczności algorytmu w aplikacjach rzeczywistych, istnieje niewiele artykułów dotyczących analizy teoretycznej opisywanego algorytmu [59, 146, 150].

W ewolucji różnicowej w każdej iteracji algorytmu stosowane są trzy jednakowo liczne populacje osobników: populacja rodziców, osobników próbnych oraz potomków. Losowo rozmieszczony w przestrzeni poszukiwań zbiór obiektów reprezentujących osobniki w populacji poddawany jest w kolejnych iteracjach działaniu operatorów genetycznych: mutacji i krzyżowaniu. Ponadto, mutacja pełni rolę nadrzędną i jest wykonywana zawsze przed krzyżowaniem. Schemat mutacji ewolucji różnicowej, w przeciwieństwie do mutacji w algorytmie genetycznym, nie jest procesem trywialnym. Prosty schemat selekcji gwarantuje, że do kolejnej iteracji przechodzą zawsze osobniki najlepiej przystosowane. Taki model selekcji w połączeniu ze skomplikowanym schematem mutacji pozwala istotnie wpłynąć na szybkość zbieżności algorytmu do optimum, a także na poprawę dokładności uzyskiwanych rozwiązań.

2.1 Ogólny algorytm ewolucji różnicowej

Ewolucja różnicowa jest algorytmem opracowanym przede wszystkim do optymalizacji funkcji ciągłych [130]. Każde potencjalne rozwiązanie określane jako osobnik, przedstawiane jest w postaci wektora liczb z zadanego przedziału. Podobnie, jak w algorytmach ewolucyjnych, na początku działania algorytmu tworzona jest populacja osobników S losowo rozmieszczonych w przestrzeni potencjalnych rozwiązań. Liczba elementów w populacji nie jest odgórnie ustalona, jednak po-

winna równomiernie pokrywać całą przestrzeń rozwiązań. Każdy osobnik opisany jest jako wektor liczb rzeczywistych reprezentujących parametry rozwiązania:

$$S = \{\vec{s}_1, \vec{s}_2, \ldots \vec{s}_i\}, i = 1, 2, \ldots, NP, \tag{2.1}$$

gdzie: NP - liczba osobników w populacji.
Każdy osobnik w populacji może zostać określony jako n-elementowy wektor, gdzie n jest wymiarem problemu:

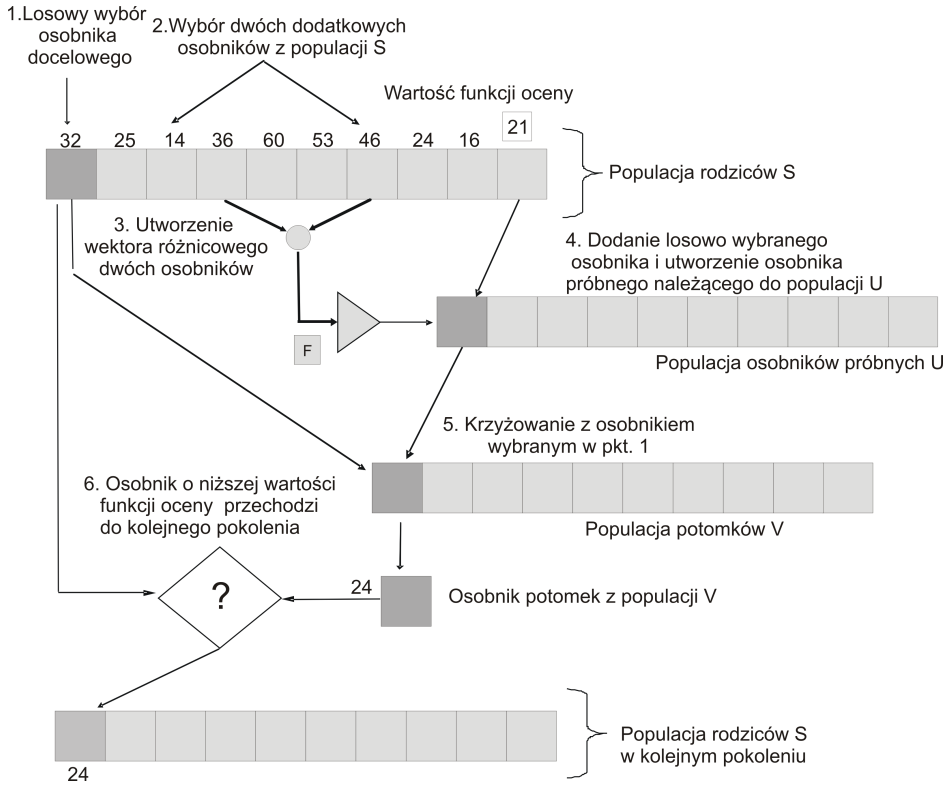
$$\vec{s}_i = \{s_{i,1}, s_{i,2}, \ldots s_{i,j}\}, j = 1, 2, \ldots, wymiar \tag{2.2}$$

gdzie: j - jest częścią wektora s_i.
Wartość j ustalana jest najczęściej zgodnie z rozkładem jednostajnym:

$$\forall_j \in wymiar\ x_j^{min} \leqslant s_{i,j} \leqslant x_j^{max},$$

gdzie:

min - dolna granica wartości w danym wymiarze,
max - górna granica wartości w danym wymiarze.



Rysunek 2.1: Schemat ewolucji różnicowej

W ewolucji różnicowej stosowane są dwa podstawowe operatory genetyczne: krzyżowanie oraz mutacja. W odróżnieniu od klasycznych algorytmów ewolucyj-

nych wskazać należy dwa istotne elementy ukazujące skuteczność ewolucji różnicowej:

- mutacja w ewolucji różnicowej wykonywana jest przed krzyżowaniem;
- krok mutacji w każdej iteracji nie zależy od rozkładu prawdopodobieństwa.

Podstawowy schemat działania ewolucji różnicowej przedstawiony został na rys. 2.1. Proces mutacji polega na utworzeniu w danej generacji t populacji wektorów próbnych U. Dla każdego osobnika z populacji S (określanej także jako populacja rodziców) tworzony jest dokładnie jeden osobnik z populacji próbnej U. Każdy osobnik tworzony jest zgodnie z następującym wzorem:

$$\forall_i, \forall_j \; U_{i,j} = S_{i,j} + F \cdot (S_{r_1,j} - S_{r_2,j}), \tag{2.3}$$

gdzie:

$S_{i,j}$ - j-ty element wektora celu,
F - współczynnik mutacji,
$(S_{r_2,j} - S_{r_3,j})$ - j-ty element wektora różnicowego.

Wektor różnicowy tworzony jest z dwóch osobników o losowych indeksach r_2 oraz r_3. W podstawowej wersji algorytmu wartość współczynnika mutacji określona była jako $F \in \langle 0, 1\rangle$ [117], jednak najczęściej przyjmuje się wartość $F \in \langle 0.5, 1\rangle$. Wartość powyższego parametru wpływa na szybkość przemieszczania się osobników w przestrzeni rozwiązań - duża wartość oznacza ukierunkowanie na eksplorację. Natomiast niewielka wartość F jest pożądana zwłaszcza w końcowej fazie algorytmu, gdzie konieczna jest eksploracja przestrzeni rozwiązań. W podstawowej wersji ewolucji różnicowej współczynnik mutacji jest wartością stałą, ustalaną odgórnie na początku działania algorytmu i jego wartość nie zależy od położenia osobnika czy iteracji algorytmu.

Populacja U uczestniczy następnie w procesie krzyżowania. Pozwala ona na utworzenie populacji potomków V. Część genotypu wektora $\vec{V_i}$ pochodzi od rodzica $\vec{S_i}$, natomiast pozostałe elementy pochodzą z wektora próbnego $\vec{U_i}$. Proces ten może zostać wyrażony następująco:

$$\forall_i, \forall_j \; V_{i,j} = \begin{cases} U_{i,j} & \text{jeżeli } rand_j\langle 0, 1) < CR, \\ S_{i,j} & \text{w przeciwnym razie,} \end{cases}$$

gdzie:

$CR \in (0, 1\rangle$ - współczynnik krzyżowania,
$rand_j\langle 0, 1)$ - liczba losowa z zadanego przedziału.

Powyższy wzór często zastępowany jest podejściem bardziej ogólnym, gdzie wyznaczany jest pewien zbiór J. Zbiór ten zawiera indeksy genów wybranego osobnika. Poprzez gen rozumiany jest tutaj konkretny wymiar rozpatrywanego problemu. Przykładowo dla problemu 10-wymiarowego genotyp osobnika ma 10 ele-

mentów, a wartość każdego elementu należy do pewnego przedziału zależnego od zagadnienia. Wyznaczenie zbioru J to wyznaczenie tych genów z genotypu osobnika próbnego, które przeniesione zostaną do potomka. Pozostałe geny kopiowane są z genotypu rodzica. Do wyznaczania powyższego zbioru najczęściej stosowany jest algorytm krzyżowania dwumianowego alg. 2, lub rzadziej algorytm krzyżowania wykładniczego alg. 3.

Algorytm 2: Pseudokod krzyżowania dwumianowego

```
1 j* ~ U(1, n_F);
2 J = J ∪ {j*};
3 foreach j ∈ {1, 2, ..., n_F} do
4 |   if Uniform(0, 1) < CR and j ≠ j* then
5 |   |   J = J ∪ {j};
```

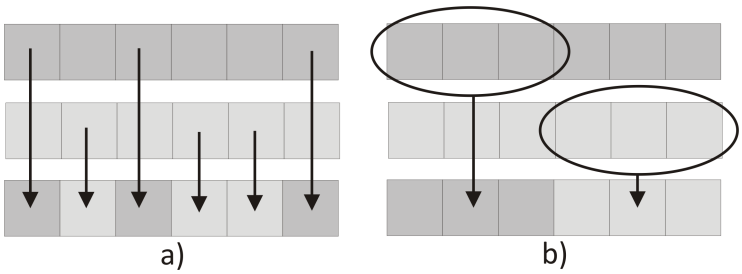
Oznaczenie j^* w algorytmie krzyżowania dwumianowego oznacza indeks genu wybrany losowo na podstawie rozkładu jednostajnego z genotypu osobnika próbnego - często stosuje się założenie, iż przynajmniej jeden jego element powinien zostać przepisany do potomka. Z kolei n_F w obydwu algorytmach oznacza liczbę genów w genotypie, czyli inaczej rozmiar problemu. Zasadnicza różnica pomiędzy przedstawionymi schematami krzyżowania polega na tym, iż w przypadku krzyżowania wykładniczego, z osobnika próbnego kopiowany jest pewien nieprzerwany podciąg genotypu. W przeciwieństwie do mutacji, wybór schematu krzyżowania ma niewielki wpływ na skuteczność algorytmu. Opisane schematy zostały porównane w pracy [151].

Algorytm 3: Pseudokod krzyżowania wykładniczego

```
1 J = {};
2 j ~ Uniform(0, n_F − 1);
3 repeat
4 |   J = J ∪ {j + 1};
5 |   j = (j + 1) mod n_F;
  until Uniform(0, 1) ⩾ CR or |J| = n_F;
```

Proces krzyżowania w ewolucji różnicowej prowadzi do utworzenia populacji potomków V. Liczba elementów populacji odpowiada liczbie osobników w populacji rodziców oraz osobników próbnych, a ogólna zasada krzyżowania przedstawiona została na rys. 2.2.

Po utworzeniu populacji potomków V następuje ocena ich przystosowania oraz jej porównanie z wartością funkcji przystosowania rodziców (z populacji S). Budowa funkcji oceny jest najistotniejszym elementem całego algorytmu, a jej poprawne zdefiniowanie gwarantuje dobrą jakość uzyskiwanych wyników. Sam me-



Rysunek 2.2: Uproszczony schemat krzyżowania: a) dwumianowego, gdzie poszczególne geny są dobierane losowo b) wykładniczego, gdzie kopiuje się kilka genów sąsiadujących

chanizm funkcji oceny jest analogiczny do funkcji oceny w algorytmach genetycznych. Genotyp osobnika, stanowiący jedno z możliwych rozwiązań, przekształcany jest na fenotyp. Fenotyp stanowi liczbowe określenie przystosowania osobnika, gdzie dla problemu minimalizacji niższa wartość fenotypu stanowi lepsze rozwiązanie.

Selekcja stosowana w ewolucji różnicowej gwarantuje, że do następnej iteracji przeniesione zostaną osobniki o najwyższej wartości funkcji przystosowania:

$$\forall_i, \vec{S}_i^{t+1} = \begin{cases} \vec{U}_i^t & \text{gdy } f(\vec{U}_i^t) \geqslant f(\vec{S}_i^t), \\ \vec{S}_i^t & \text{w przeciwnym razie,} \end{cases}$$

gdzie:

$\vec{S}_i^{t+1}$ - i-ty osobnik z generacji $t+1$,

$f(\cdot)$ - funkcja przystosowania.

Po etapie selekcji, w populacji S znajdują się osobniki o przystosowaniu nie gorszym niż w poprzedniej iteracji. Cały proces mutacji, krzyżowania i selekcji powtarzany jest aż do spełnienia warunku zakończenia algorytmu. Samo zakończenie działania algorytmu zależy w dużym stopniu od problemu. Nierzadko w algorytmie stosuje się kilka kryteriów stopu, gdzie do najpopularniejszych należą: maksymalna liczba iteracji i brak poprawy wartości funkcji oceny w n kolejnych iteracjach.

2.2 Schematy mutacji w ewolucji różnicowej

Podobnie, jak inne algorytmy ewolucyjne, tak i ewolucja różnicowa posiada szereg modyfikacji, w założeniu poprawiających jakość uzyskiwanych rozwiązań oraz przyspieszających zbieżność algorytmu do optimum. Najczęściej modyfikowane elementy w DE to:

- metoda dotycząca wyboru wektora celu (oznaczana jako x),

- liczba stosowanych wektorów różnicowych (y),
- schemat krzyżowania (z).

Zgodnie z powyższymi oznaczeniami, każda modyfikacja algorytmu ewolucji różnicowej może zostać opisana jako: DE/x/y/z. Najpopularniejsze modyfikacje schematów DE dotyczą operatora mutacji, jako że ma on najistotniejszy wpływ na jakość uzyskiwanych wyników. Do opisu wszystkich schematów zastosowano następujące oznaczenia:

- i - indeks osobnika w populacji rodziców S,
- j - indeks genu w genotypie danego osobnika,
- r_1, r_2, r_3 - losowe indeksy osobników z populacji S,
- $S_{r_1,j}$ - j-ty gen osobnika o indeksie r_1,
- F - współczynnik mutacji,
- $S_{best,j}$ - j-ty gen osobnika o najlepszym przystosowaniu.

Poniżej przedstawiono najpopularniejsze modyfikacje spotykane w literaturze [17, 28, 44]:

- Strategia I: DE/rand/1/z

  Jest ona podstawową strategią w ewolucji różnicowej. Indeks wektora celu jest tutaj identyczny, jak indeks osobnika próbnego. Oznacza to, iż osobnik próbny tworzony jest na podstawie losowego osobnika z populacji rodziców oraz na podstawie wektora różnicowego utworzonego z dwóch losowo wybranych osobników z populacji odpowiednio przeskalowanych parametrem F. Podobnie, jak w przypadku innych schematów, tak tutaj schemat krzyżowania może zostać wybrany dowolnie (krzyżowanie dwumianowe lub wykładnicze opisane wcześniej w tym rozdziale). Proces mutacji dla tego schematu może zostać opisany następującym wzorem:

  $$\forall_i, \forall_j \; U_{i,j} = S_{r_1,j} + F \cdot (S_{r_2,j} - S_{r_3,j}). \tag{2.4}$$

- Strategia II: DE/best/1/z

  W tym schemacie wymagane jest dodatkowe wyznaczenie najlepszego osobnika w populacji, tj. osobnika o najwyższej wartości funkcji przystosowania. W przypadku skomplikowanych problemów i dużej populacji, operacja taka może być czasochłonna, gdyż wyznaczenie najlepszego osobnika konieczne

jest w każdej iteracji algorytmu. Drugim elementem schematu jest pojedynczy wektor różnicowy, odpowiednio przeskalowany parametrem F. Schemat ten może zostać opisany następującym wzorem:

$$\forall_i, \forall_j \; U_{i,j} = S_{best,j} + F \cdot (S_{r_1,j} - S_{r_2,j}). \tag{2.5}$$

- Strategia III: DE/x/n_v/z

  W powyższych strategiach stosowany był tylko jeden wektor różnicowy obliczany na podstawie dwóch losowo wybranych osobników z populacji rodziców. Dla tego schematu wprowadzony jest dodatkowy parametr n_v określający liczbę wektorów różnicowych stosowanych w mutacji. Suma wektorów różnicowych skalowana jest następnie parametrem F. Dodatkowo, stosowany jest także jeden losowy osobnik z populacji rodziców. Należy zwrócić uwagę, iż duża wartość parametru n_v może prowadzić do sytuacji, kiedy modyfikacja genotypu osobnika nawet po przeskalowaniu parametrem F jest znaczna i prowadzi do umieszczenia osobnika na granicach obszaru przeszukiwań. Zwłaszcza w końcowej fazie algorytmu, gdzie oczekiwane jest zwiększenie nacisku na eksploatację przestrzeni rozwiązań, może prowadzić to do istotnego pogorszenia wyników. Wzór powyższego schematu przedstawiony jest poniżej:

  $$\forall_i, \forall_j \; U_{i,j} = S_{r_1,j} + F \cdot \sum_{k=1}^{n_s} (S_{r_{k.2},j} - S_{r_{k.3},j}). \tag{2.6}$$

- Strategia IV: DE/rand to best/n_v/z

  W strategii tej konieczne jest znalezienie najlepszego osobnika w populacji. Osobnik ten skalowany jest dodatkowym parametrem $\gamma \in \langle 0, 1 \rangle$. Jednocześnie, losowo wybrany osobnik z populacji skalowany jest parametrem $1 - \gamma$. Podejście takie pozwala na ukierunkowanie przeszukiwania przestrzeni rozwiązań poprzez położenie nacisku na osobnika losowego - większa eksploracja, lub też na najlepszego osobnika w populacji - większa eksploatacja. Ponadto, istotnym elementem jest też n_v dodatkowych wektorów różnicowych, tworzonych na podstawie losowych osobników z populacji:

  $$\forall_i, \forall_j \; U_{i,j} = \gamma S_{best,j} + (1 - \gamma) S_{r_1,j} + F \cdot \sum_{k=1}^{n_s} (S_{r_{k.2},j} - S_{r_{k.3},j}). \tag{2.7}$$

- Strategia V: DE/current to best/$1 + n_v$/z

  Ta strategia pozwala ukierunkować algorytm na szybką zbieżność do optimum. Podobnie jak w strategii II, w tym przypadku konieczne jest wskazanie osobnika o najwyższej wartości funkcji oceny z populacji rodziców. Kolejnym krokiem jest utworzenie wektora różnicowego z najlepszego osobnika oraz z

drugiego, losowo wybranego z populacji. Drugim elementem opisywanego schematu jest kilka (minimum jeden) wektorów różnicowych tworzonych z losowo wybranych osobników. Schemat ten opisany jest następującym wzorem:

$$\forall_i, \forall_j \; U_{i,j} = S_{i,j} + F \cdot (S_{best,j} - S_{r_1,j}) + F \cdot \sum_{k=1}^{n_s} (S_{r_{k.2},j} - S_{r_{k.3},j}). \qquad (2.8)$$

Parametr F pełni rolę skalującą wektory różnicowe. W końcowym etapie działania algorytmu, kiedy następuje przede wszystkim eksploatacja przestrzeni rozwiązań, wartość F powinna być niewielka, co prowadzi do niewielkich zmian położenia osobników w przestrzeni rozwiązań. W każdym z powyżej opisanych schematów mutacji, współczynnik mutacji nie wpływa w żaden sposób na wektor celu. Oznacza to, że wpływ tego osobnika na krok w mutacji jest taki sam niezależnie od iteracji algorytmu. Redukcja tej wartości poprzez wprowadzenie dodatkowego czynnika skalującego $\lambda \in \langle 0, 1 \rangle$, zmieniającego się wraz z kolejnymi iteracjami algorytmu może istotnie wpłynąć na poprawę zbieżności algorytmu w końcowej jego fazie. Powyższa modyfikacja określona została jako λ–modyfikacja, a jej skuteczność w optymalizacji funkcji ciągłych wykazana została w pracy [17]. Dla każdego z opisanych dotychczas schematów mutacji możliwe jest wprowadzenie następujących zmian:

- Strategia I: $\forall_i, \forall_j \; U_{i,j} = \lambda \cdot S_{r_1,j} + F \cdot (S_{r_2,j} - S_{r_3,j})$,
- Strategia II: $\forall_i, \forall_j \; U_{i,j} = \lambda \cdot S_{best,j} + F \cdot (S_{r_1,j} - S_{r_2,j})$,
- Strategia III: $\forall_i, \forall_j \; U_{i,j} = \lambda \cdot S_{r_1,j} + F \cdot \sum_{k=1}^{n_s} (S_{r_{k.2},j} - S_{r_{k.3},j})$,
- Strategia IV: $\forall_i, \forall_j \; U_{i,j} = \lambda \cdot (\gamma S_{best,j} + (1-\gamma) S_{r_1,j}) + F \cdot \sum_{k=1}^{n_s} (S_{r_{k.2},j} - S_{r_{k.3},j})$,
- Strategia V: $\forall_i, \forall_j \; U_{i,j} = \lambda \cdot S_{i,j} + F \cdot (S_{best,j} - S_{r_1,j}) + F \cdot \sum_{k=1}^{n_s} (S_{r_{k.2},j} - S_{r_{k.3},j})$.

Ta modyfikacja niesie ze sobą zadziwiająco dobrą poprawę wyników i pozwala jednocześnie na istotne ograniczenie maksymalnej liczby iteracji koniecznych do osiągnięcia optimum globalnego. Odwołania do powyższego podejścia znajdują się w rozdziale dotyczącym adaptacyjnej wersji ewolucji różnicowej, natomiast szczegóły tego podejścia podane zostaną w rozdziale dotyczącym autorskich modyfikacji.

2.3 Modyfikacje ewolucji różnicowej

Duża skuteczność opisywanego algorytmu doprowadziła do powstania licznych modyfikacji ukierunkowanych na konkretne zadania. Modyfikacje ewolucji różnicowej podzielić można ze względu na obszar potencjalnych rozwiązań, a co za tym idzie na genotyp osobnika. Poniżej przedstawiono najistotniejsze modyfikacje:

- ewolucja różnicowa dla wartości dyskretnych i binarnych;
- ewolucja różnicowa w optymalizacji wielokryterialnej i dynamicznej;
- ewolucja różnicowa w optymalizacji ciągłej, jednotablicowa oraz z losowym krokiem.

Powyższe zestawienie pozwala wskazać najważniejsze elementy poszczególnych modyfikacji. W szczególności istotny wydaje się podział uwzględniający reprezentację genotypu osobników. Jedną z najpopularniejszych modyfikacji jest binarna ewolucja różnicowa (ang. *binary DE*). Podejście zapożyczone z koncepcji J. Kennedy'ego i R. Eberharta [43] opisane zostało w [45]. W przedstawionym algorytmie każdy osobnik w DE reprezentowany jest przez wektor liczb rzeczywistych, jednak odmienna jest interpretacja takiego wektora. Każdy element określa prawdopodobieństwo wystąpienia wartości 1 oraz 0 na zadanej pozycji. Przed właściwą oceną, każdy genotyp dekodowany jest na wektor binarny. Wreszcie taki wektor binarny stanowi właściwy genotyp osobnika, który następnie przekształcany jest na wartość funkcji oceny. Sposób przekształcania wektora wartości rzeczywistych na wektor binarny przedstawiony został poniżej:

$$\forall_i, \forall_j \ \hat{S}_{i,j} = \begin{cases} 0 & \text{jeżeli } f(S_{i,j}) \geqslant 0.5, \\ 1 & \text{jeżeli } f(S_{i,j}) \leqslant 0.5, \end{cases}$$

gdzie f jest funkcją sigmoidalną:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.9}$$

W artykule [36] wykazano skuteczność binarnej ewolucji różnicowej w problemie plecakowym, gdzie w operatorze mutacji pominięty został współczynnik mutacji F. Z kolei w artykule [48] wykazano dużą skuteczność podejścia, w którym wektor wartości binarnych nie jest tworzony na podstawie wektora liczb rzeczywistych. Wartości binarne modyfikowane są bezpośrednio w kolejnych iteracjach algorytmu. Podjęto też pewne próby w celu wykazania skuteczności binarnej wersji algorytmu w problemach dynamicznych, gdzie odległość osobnika od optimum globalnego określana była na podstawie prostej odległości Hamminga [78].

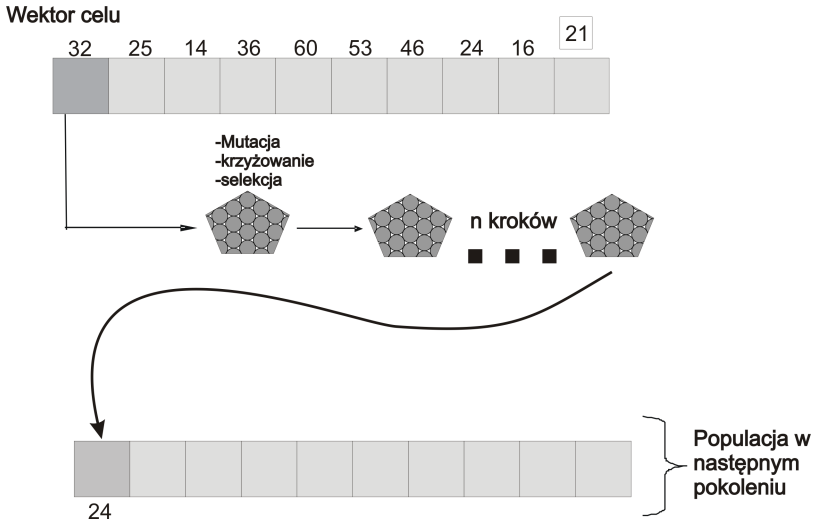
Algorytm binarnej ewolucji różnicowej może być rozpatrywany jako podejście ukierunkowane na rozwiązywanie problemów dyskretnych. Także w tym zakresie podjęte zostały liczne próby modyfikacji algorytmu, a rozpatrywane zagadnienia dotyczyły takich problemów jak szeregowanie zadań, czy problem komiwojażera [65, 115]. W problemie komiwojażera udało się wykazać skuteczność algorytmu tylko dla niewielkich zbiorów testowych. Stąd liczne modyfikacje pozwalające zwiększyć skuteczność algorytmu. Jedną z najpopularniejszych modyfikacji było dodanie przeszukiwania lokalnego [121]. Interesującym wydaje się też być zastoso-

wanie ewolucji różnicowej dla problemu rozwiązywania i generowania Sudoku [16]. Niestety konieczność zastosowania genotypu o długości 81 elementów (macierz 9 x 9 elementów) prowadzi do konkluzji, iż jest to jeden z problemów, w których algorytmy dedykowane wykazują znacznie większą skuteczność.

Z punktu widzenia niniejszej rozprawy, interesujące wydaje się być wprowadzenie modyfikacji pozwalającej na optymalizację wielokryterialną. Przy problemie optymalizacji wielokryterialnej należy tutaj zaznaczyć, że ewolucja różnicowa sprawdza się również bardzo dobrze jako algorytm niszowania [47]. Jednym z podejść do optymalizacji wielokryterialnej w ewolucji różnicowej jest zastosowanie metody Min-Max, gdzie na początku poszukiwane jest pewne rozwiązanie przybliżone. Algorytm ten zbliżony jest do metody kryterium globalnego, gdzie rozwiązaniem może być np. rozwiązanie stanowiące ekstremum dla poszczególnych kryteriów rozpatrywanych osobno. Metodę Min-Max zastosowano w pracach [3, 73]. Z kolei w pracy [9] zastosowano jedną z najpopularniejszych metod: metodę ważonych kryteriów, gdzie optymalizacja wielokryterialna sprowadzona zostaje do optymalizacji jednokryterialnej poprzez wprowadzenie kryterium zastępczego, będącego sumą ważoną poszczególnych kryteriów.

Jednym z pierwszych artykułów, w których nie zastosowano modyfikacji funkcji oceny był [1]. W proponowanym przez autorów podejściu, w każdej iteracji algorytmu, nowe osobniki tworzone są tylko ze zbioru rozwiązań niezdominowanych. Oznacza to konieczność przeglądu całej populacji w każdej iteracji algorytmu. Następnie dla tak powstałego zbioru stosowany jest operator mutacji i krzyżowania. W artykule [61] oprócz mechanizmu usuwania osobników zdominowanych wprowadzona została adaptacja współczynnika mutacji i krzyżowania. Problem adaptacji parametrów przedstawiony zostanie szerzej w kolejnym rozdziale rozprawy.

Do ostatniej grupy modyfikacji algorytmu ewolucji różnicowej należy m.in. sekwencyjny algorytm ewolucji różnicowej [136]. Podstawowa wersja DE, zwana także równoległym DE opiera się na zastosowaniu 3 populacji osobników o jednakowej liczności. W każdej iteracji, na podstawie populacji rodziców tworzona jest populacja osobników próbnych a następnie potomków. Z kolei w sekwencyjnej ewolucji różnicowej, w danej iteracji stosowana jest tylko jedna populacja, w której potomek porównywany jest od razu ze swoim rodzicem - oczywiście osobnik o wyższej wartości funkcji przystosowania zastępuje swojego rodzica. W algorytmie tym w każdej iteracji, w trakcie działania operatorów mutacji, krzyżowania i selekcji, do populacji trafiają nowe osobniki, co może prowadzić do przedwczesnej zbieżności do optimum. Zwłaszcza w złożonych problemach optymalizacyjnych może to być kłopotliwe. Zbliżoną modyfikację, o jednak znacznie większej skuteczności stanowi tzw. krzyżowa ewolucja różnicowa opisana w [136], w której dla każdego osobnika z populacji możliwe jest zastosowanie n kroków, gdzie w każdym kroku wykonywana jest mutacja, krzyżowanie i selekcja. Schemat tego podejścia przedstawiony został na rys. 2.3.



Rysunek 2.3: Krzyżowa ewolucja różnicowa

Należy zaznaczyć, iż istnieją modyfikacje dotyczące zastosowania ewolucji różnicowej w środowiskach dynamicznych, jednak z punktu widzenia niniejszej rozprawy podejścia takie nie są istotne. Środowiska dynamiczne wymagają zastosowania odmiennej oceny rozwiązań i nie mogą być zaadaptowane do proponowanego algorytmu.

Powyższy przegląd literaturowy dotyczący modyfikacji podstawowej wersji algorytmu ewolucji różnicowej pozwala zwrócić uwagę na istotny fakt, iż jednym z najważniejszych elementów mających wpływ na działanie metody ma odpowiedni wybór reprezentacji osobnika, który jest z kolei powiązany z operatorem mutacji. Operator mutacji stanowi najistotniejszy element ewolucji różnicowej oraz umożliwia ukierunkować przeszukiwania na szybkie znalezienie optimum. Pozwala to wysnuć wniosek, iż odpowiednia modyfikacja operatora mutacji powinna istotnie wpłynąć na jakość uzyskiwanych rozwiązań. Można też zauważyć, iż wprowadzenie reprezentacji binarnej często prowadzi do sytuacji, kiedy działanie algorytmu zostaje ograniczone tylko dla wybranego zagadnienia. Skutecznym rozwiązaniem może być tutaj opracowanie reprezentacji genotypu umożliwiającej rozwiązywanie szerokiej klasy problemów. Będzie to stanowiło jeden z istotnych elementów niniejszej rozprawy. W kolejnym rozdziale natomiast przestawiony zostanie przegląd istniejących metod hybrydowych, a także modyfikacji pozwalających na samoczynną zmianę istotnych parametrów algorytmu, takich jak wartość współczynnika mutacji czy krzyżowania.

ROZDZIAŁ 3

Hybrydowe i adaptacyjne warianty ewolucji różnicowej

Opisane w poprzednim rozdziale liczne modyfikacje podstawowej wersji DE dotyczą w większości adaptacji algorytmu do konkretnych problemów. Wskazano ponadto, iż najistotniejszym fragmentem pozwalającym na poprawę wyników i przyspieszenie zbieżności algorytmu jest operator mutacji. Niestety mnogość schematów mutacji nie pozwala jednoznacznie wskazać metody pozwalającej uzyskać najlepsze rezultaty. Dodatkowo, pewnym ograniczeniem wydają się być statyczne wartości wszystkich kluczowych parametrów algorytmu, takich jak: współczynnik mutacji F, współczynnik krzyżowania CR, czy wielkość populacji NP.

Jedną z modyfikacji algorytmu ewolucji różnicowej jest wprowadzenie metod hybrydowych. Nierzadko, w takich przypadkach działanie metody hybrydowej opiera się na przeszukiwaniu lokalnym wprowadzonym do podstawowej wersji algorytmu. Pozwala to w pewnym stopniu zrezygnować z dynamicznej zmiany parametrów algorytmu, a jednocześnie prowadzi do poprawy wyników. Niestety najczęściej podejścia takie związane są z dość znacznym nakładem obliczeniowym.

Interesującym podejściem jest dynamiczna zmiana parametrów (dotyczy to przede wszystkim operatora mutacji, rzadziej krzyżowania), gdzie w kolejnych iteracjach ich wartości są modyfikowane na podstawie pewnych z góry określonych wzorów. Pozwala to wprowadzić pewną losowość, a także w bardziej złożonych schematach uwzględnić zależność pomiędzy eksploracją a eksploatacją rozwiązań.

Głównym założeniem metod adaptacyjnych jest odciążenie użytkownika od konieczności doboru początkowych wartości parametrów. Jednym z rozwiązań w tych podejściach jest wybór wartości parametrów zgodnie z pewnym rozkładem, a następnie ich modyfikacja w kolejnych iteracjach w zależności od skuteczności poszukiwań. Metody takie pozwalają wyraźnie rozgraniczyć eksplorację i eksploatację przestrzeni rozwiązań i szczególnie w końcowej fazie działania algorytmu mogą poprawić szybkość zbiegania do optimum. Interesujące wydaje się być także

wprowadzenie zmiany wielkości populacji, zależnie od postępów algorytmu.

3.1 Metody hybrydowe

Pomimo dużej skuteczności algorytmu ewolucji różnicowej, nierzadko zdarza się, iż lokalizacja zadowalającego rozwiązania prowadzi do istotnego wydłużenia czasu działania algorytmu. Bardzo częste są modyfikacje pozwalające połączyć ten algorytm z przeszukiwaniem lokalnym. Jednym z pierwszych algorytmów hybrydowych było zaproponowane przez J. Chiou i F. Wang w pracy [25] podejście zawierające dwa nowe operatory: operator przyspieszenia i migracji. Pierwszy z nich oparty na metodach gradientowych, stosowany jest w momencie, gdy tradycyjne operatory mutacji i krzyżowania w danej iteracji nie pozwolą na poprawę jakości osobnika. Przy czym operator ten używany jest tylko dla osobnika o najwyższej wartości funkcji przystosowania w populacji:

$$\forall_i, \vec{S}_i^{t+1} = \begin{cases} \vec{U}_i^t & \text{gdy } f(\vec{U}_i^t) \geqslant f(\vec{S}_i^t), \\ \vec{S}_i^t - \eta(t) \cdot \nabla f & \text{w przeciwnym razie,} \end{cases}$$

gdzie:

$\eta \in (0, 1\rangle$ - wielkość kroku,

∇f - gradient funkcji celu.

Wartość η dobierana jest losowo w każdym kroku. W celu zapewnienia odpowiedniej różnorodności populacji, wprowadzony został także operator migracji. W przypadku, kiedy różnorodność populacji jest niewielka, część osobników zastępowana jest przez zmodyfikowane wersje najlepszego osobnika w populacji.

Kolejne, interesujące podejście opisane zostało przez T. Changa i H. Changa w pracy [23], gdzie oprócz standardowego schematu mutacji DE, zastosowany został operator mutacji, dobrze znany z algorytmów genetycznych pozwalający na wprowadzenie niewielkiego szumu do istniejących wektorów różnicowych:

$$\forall_i, \forall_j \; U_{i,j} = U_{i,j} + Uniform(min, max), \tag{3.1}$$

gdzie: $Uniform(min, max)$ - rozkład jednostajny w zadanym przedziale.

Z kolei podejście opisane przez H. Sarimveisa i A. Nikolakopoulosa w pracy [106] dotyczyło wprowadzenia do procesu mutacji selekcji rankingowej, gdzie przed właściwą mutacją, osobniki z populacji szeregowane są według wartości funkcji przystosowania. Osobniki lepiej przystosowane (z wyższą wartością funkcji przystosowania) mają większą szansę na wylosowanie. Podobne podejście zostało później opisane w pracy [120], jako część adaptacyjnej wersji algorytmu ewolucji różnicowej.

Jednym z najczęściej stosowanych połączeń algorytmów jest hybryda ewolucji różnicowej i optymalizacji stadnej cząsteczek (z ang. *Particle Swartm Optimiza-*

tion, PSO). Podejście to opisywane było w szeregu artykułów. T. Hendtlass [69] zaproponował proces reprodukcji ewolucji różnicowej w PSO - w określonych przedziałach czasowych populacja cząstek traktowana jest jako populacja w ewolucji różnicowej. Z kolei W-J. Zhang i X-F. Xie [145] oraz H. Talbi i M. Batouche [11] zastosowali podejście, w którym najlepsza lokalna pozycja osobnika w PSO aktualizowana jest na podstawie pewnej modyfikacji wektora różnicowego. Nowe położenie obliczane jest na podstawie wartości średniej dwóch losowych pozycji lokalnych.

Bardzo popularnym algorytmem hybrydowym jest połączenie ewolucji różnicowej i symulowanego wyżarzania (z ang. *Simulated Annealing* SA) jako metody przeszukiwania lokalnego. Podejście takie niestety jest kosztowne pod względem obliczeniowym - najczęściej po iteracji DE następuje proces przeszukiwania lokalnego z zastosowaniem symulowanego wyżarzania. Przykłady tego typu algorytmów można spotkać m.in. w [77, 80].

Algorytm określany jako DEACO (z ang. *Differential Evolution Ant Colony Optimization*) przedstawiony został w pracy [144] i dotyczy ewolucji różnicowej oraz optymalizacji mrowiskowej. Podejście to dotyczy właściwie usprawnienia odkładania śladu feromonowego w algorytmie mrowiskowym ACO (z ang. *Ant Colony System*), zastosowanego do problemu komiwojażera. Z kolei w pracy [4], gdzie rozpatrywany jest problem dyskretny zbliżony do problemu komiwojażera, ACO zastosowane zostało tylko do dostarczenia populacji początkowej dla algorytmu DE. Pomimo dużej skuteczności powyższych hybryd opisanych szczegółowo w artykułach, można mieć pewne zastrzeżenia do czasu działania algorytmu i jego złożoności obliczeniowej.

Wreszcie najnowszym algorytmem hybrydowym, w którym ewolucja różnicowa pełni istotną rolę jest podejście opisane w pracy [110]. Połączenie DE i systemu rojowego szczegółowo opisane zostało również w [76], gdzie nowy algorytm określono jako HDABCA (z ang. *Hybrid Differential Artificial Bee Colony Algorithm*). W artykule tym przedstawiono klasyczne podejście do algorytmów hybrydowych, gdzie po etapie działania algorytmu ABC (z ang. *Artificial Bee Colony*) wybrane zostaje n wektorów, a następnie dla takiego podzbioru realizowany jest algorytm ewolucji różnicowej. Skuteczność powyższego podejścia wykazana została na przykładzie problemu optymalizacji klasycznych, 30 - wymiarowych funkcji takich jak funkcja Rosenbrocka, czy Griewanka. Problem czasu działania algorytmu został w tym przypadku również pominięty.

Niestety pomimo wyraźnych zalet, jedną z najistotniejszych wad algorytmów hybrydowych jest konieczność odpowiedniego doboru parametrów. Nierzadko algorytmy hybrydowe wymagają dostrojenia większej liczby parametrów, niż ma to miejsce w tradycyjnych algorytmach. Stąd nasuwa się pytanie, czy dany problem wymaga zastosowania algorytmu hybrydowego, czy też odpowiednia metoda doboru wartości współczynników również korzystnie wpłynie na jakość uzy-

skiwanych rozwiązań bez wydłużenia czasu działania algorytmu, a jednocześnie odciąży użytkownika od konieczności doboru parametrów. Podsumowując, modyfikacje pozwalające sterować zachowaniem algorytmu w końcowej fazie działania algorytmu, zwiększające nacisk na eksplorację przestrzeni rozwiązań (przykładem może być tutaj podejście opisane w poprzednim rozdziale) mogą z dużym prawdopodobieństwem zastąpić skomplikowane metody hybrydowe, pozwalając przy tym poprawić jakość uzyskiwanych rozwiązań.

3.2 Metody oparte na dynamicznej zmianie parametrów

Jak wspomniano w poprzednim podrozdziale, konieczność podania pewnej liczby parametrów (zwłaszcza w przypadku algorytmów hybrydowych) często prowadzi do trudnego etapu strojenia algorytmu. W literaturze [87, 149] podane są optymalne wartości współczynników mutacji czy krzyżowania, należy jednak pamiętać, że najczęściej zbiorem testowym jest standardowy zestaw funkcji. W przypadku bardziej złożonych problemów najczęściej wymagane jest ustalenie innych wartości parametrów. W takich przypadkach dobrym rozwiązaniem jest zastosowanie modyfikacji opartych na dynamicznej zmianie parametrów algorytmu, takich jak współczynnik mutacji F, czy współczynnik krzyżowania CR.

Jednym z pierwszych podejść opartych na dynamicznej zmianie parametrów jest metoda zaproponowana w pracy [22], gdzie wartość współczynnika mutacji F w kolejnych iteracjach ustalana jest na podstawie wzoru:

$$F_{t+1} = F_t + (0.5 + F_0)/g, \tag{3.2}$$

gdzie:

F_{t+1} - wartość parametru F w kolejnej iteracji,

g - liczba iteracji algorytmu,

F_0 - początkowa wartość parametru.

Autorzy założyli, iż początkowa wartość F powinna zostać ustalona na poziomie 0.3 a następnie być stopniowo zwiększana. Z kolei wartość współczynnika krzyżowania CR w kolejnych iteracjach powinna być aktualizowana zgodnie ze wzorem:

$$CR_{t+1} = CR_t - (CR_0 - 0.7)/g, \tag{3.3}$$

gdzie:

CR_{t+1} - wartość parametru CR w kolejnej iteracji,

CR_0 - początkowa wartość parametru.

D. Zaharie zaproponowała podejście oparte na losowaniu wartości parametru F w każdej iteracji algorytmu [149], gdzie wartości współczynnika mutacji są nieza-

leżne i losowane zgodnie z rozkładem normalnym w każdej iteracji algorytmu. Taka modyfikacja pozwala na istotne zwiększenie różnorodności populacji w kolejnych iteracjach, poza tym zapobiega wystąpieniu sytuacji zwanej stagnacją populacji [88]. Problem ten dotyczy funkcji o małej liczbie wymiarów oraz o niewielkiej liczności populacji. Przyczyną wprowadzenia przez D. Zaharie wspomnianej modyfikacji była pozycja [55] dotycząca analizy zbieżności w algorytmach ewolucyjnych. Wprowadzenie zmiennej wartości było jednym z dwóch warunków zbieżności algorytmu (dodatkowym warunkiem spełnionym oczywiście przez algorytm ewolucji różnicowej była selekcja elitarna, gdzie do kolejnej iteracji przechodzi osobnik o najlepszej wartości funkcji przystosowania).

3.3 Adaptacja parametrów w ewolucji różnicowej

W tej części rozdziału przedstawione zostaną najpopularniejsze koncepcje dotyczące problemu adaptacji i samoadaptacji parametrów w algorytmie ewolucji różnicowej. Należy zauważyć, iż nierzadko pojęcia adaptacji oraz samoadaptacji są w wielu publikacjach stosowane zamiennie. Ciężko również wytyczyć jednoznaczną granicę pomiędzy dynamiczną a adaptacyjną zmianą parametrów w algorytmie ewolucji różnicowej. Na potrzeby niniejszej pracy autor przyjął założenie, iż wszelkie zmiany parametrów opierające się na aktualizacji parametrów na podstawie bieżącej lub poprzedniej populacji, a także zakładające stosowanie dowolnego rozkładu prawdopodobieństwa, określane są jako metody adaptacyjne. Większość istniejących modyfikacji zakłada niezmienną wielkość populacji w trakcie działania algorytmu.

Jedną z pierwszych prac, w których założono zmienną wielkość populacji jest [134]. W pracy zaproponowany został algorytm DESAP(z ang. *Differential Evolution with Self-Adapting Populations*), który oprócz dynamicznej zmiany współczynnika mutacji oraz krzyżowania oferuje także zmianę wielkości populacji w trakcie działania algorytmu. W algorytmie wprowadzony został dodatkowy współczynnik określany jako parametr populacji π. Wyszczególniono ponadto dwie wersje algorytmu. W wersji określonej jako DESAP-Abs początkowa wartość parametru π ustalana jest jako:

$$\pi = Round(NP + Normal(0, 1)), \tag{3.4}$$

gdzie:

$Round(\cdot)$ - zaokrąglenie wartości,

NP - początkowa wielkość populacji,

$Normal(0, 1)$ - wartość losowa generowana na podstawie rozkładu normalnego o wartości oczekiwanej 0 i wariancji równej 1.

W tym wypadku wartość π jest powiązana bezpośrednio z początkową wielkością

populacji NP. Z kolei w drugim podejściu określonym jako DESAP-Rel wartość π losowana jest na początku zgodnie z rozkładem jednostajnym w przedziale $\langle -0.5, 0.5 \rangle$. Trudno jednoznacznie stwierdzić, czy charakter zmian wielkości populacji jest adaptacyjny, czy też dynamiczny. Wartość parametru π ustalana jest w kolejnych iteracjach na podstawie rozkładu normalnego, jednak wielkość populacji zależy też od wartości NP w poprzedniej generacji. Sam proces określany jest przez autora jednoznacznie jako podejście adaptacyjne. Warto zauważyć, że ich adaptacyjny charakter zmian wielkości populacji powinien wynikać z informacji pozyskiwanych z całej populacji (takich jak wartości funkcji przystosowania osobników).

Na szczególną uwagę zasługuje adaptacyjna zmiana wielkości populacji, gdzie możliwe jest zmniejszenie wielkości populacji, jak i powiększenie jej rozmiaru. Algorytm DESAP w porównaniu z klasycznym schematem DE radzi sobie lepiej tylko w przypadku 1 z 5 klasycznych funkcji De Jonga. Autor wskazuje jednak, iż DESAP jest niejako punktem wyjściowym dla algorytmów oferujących możliwość redukcji parametrów.

W pracy [18] J. Brest stosował wzory na modyfikację współczynników F i CR. Wartości te w każdej iteracji modyfikowane były w pewnym stopniu losowo. Wielkość populacji ustalona została na stałym poziomie i nie była modyfikowana. Z kolei w pracy [19] tego samego autora zastosowana została (oprócz modyfikacji wartości F i CR) także redukcja wielkości populacji w trakcie działania algorytmu. Należy zauważyć, że proces redukcji populacji odbywał się stosunkowo rzadko, co g iteracji, natomiast redukcji ulega w tym przypadku połowa populacji. Drugą, istotną modyfikacją algorytmu jest mechanizm zmiany znaku współczynnika F. Proces ten odbywa się z pewnym określonym prawdopodobieństwem. Ten sam autor w pracy [21] założył dodatkowo stosowanie w algorytmie trzech różnych schematów mutacji. Ponadto odniesiono się do artykułu D. Zaharie [149], gdzie wartości parametrów powinny spełniać następującą zależność:

$$2 \cdot F^2 - \frac{2}{NP} + \frac{CR}{NP} = 0, \tag{3.5}$$

gdzie:

F - współczynnik mutacji,

CR - współczynnik krzyżowania,

NP - wielkość populacji.

Powyższe podejście nie określa jednak w żaden sposób nawet przybliżonych wartości współczynników. Należy również pamiętać, iż wzór ten jest skuteczny tylko dla parametrów statycznych. W przypadku adaptacji wartości nie jest możliwe zachowanie zależności pomiędzy nimi w kolejnych iteracjach algorytmu.

Podejście związane z losowym wyborem aktywnej strategii mutacji zaproponowane zostało w pracy [74], gdzie autorzy założyli, że w każdej iteracji wszystkie

cztery strategie mają pewne prawdopodobieństwo wyboru. Podejście to stanowiło rozwinięcie koncepcji z dwoma aktywnymi strategiami zaproponowanymi w pracy [118]. Wielkość populacji jest parametrem definiowanym przez użytkownika, z kolei wartości parametrów F oraz CR określane są na podstawie rozkładu normalnego. Metoda zakładająca stosowanie więcej niż jednego schematu mutacji w trakcie działania algorytmu przedstawiona została również w pracy [132]. Autorzy zaproponowali algorytm określany jako SaNSDE (z ang. *Self-Adaptive Neighborhood Search Differential Evolution*), w którym dla każdego z dwóch schematów mutacji wartość współczynnika F losowana jest na podstawie rozkładu normalnego bądź rozkładu Cauchy'ego:

$$\forall_i, \forall_j \; V_{i,j} = \begin{cases} rand_n(0.5, 0.3) & \text{jeżeli } rand\langle 0,1) < fp, \\ rand_c(0.0, 1.0) & \text{w przeciwnym razie,} \end{cases}$$

gdzie:

$rand_n(0.5, 0.3)$ - wartość z rozkładu normalnego o wartości oczekiwanej 0.5 oraz wariancji 0.3,

$rand_c(0.0, 0.1)$ - wartość z rozkładu Cauchy'ego o położeniu w punkcie 0.0 oraz skali 1.0,

$rand\langle 0, 1)$ - wartość losowa z zadanego przedziału,

fp - ustalona wartość z przedziału $(0, 1\rangle$.

Proces określania wartości fp jest taki sam, jak w przypadku algorytmu SaDE (z ang. *Self-adaptive Differential Evolution*) a szczegółowo został on opisany w pracy [118]. Istotny jest natomiast fakt, iż z czasem autorzy do oryginalnego algorytmu wprowadzili podejście związane z przeszukiwaniem lokalnym, gdzie po określonej liczbie iteracji następuje eksploatacja przestrzeni rozwiązań.

Odmienne podejście zastosowane zostało w algorytmie DE-AEC (z ang. *Differential Evolution with Adaptive Evolution Control*), gdzie opisany został model surogacki [120]. Metoda ta dotyczy problemów optymalizacyjnych z bardzo złożoną funkcją przystosowania. Główna idea tego podejścia opiera się na zastąpieniu oryginalnej funkcji oceny (kosztownej pod kątem obliczeniowym) pewnym modelem przybliżonym. Przykład modelu surogackiego opartego na radialnej sieci bazowej opisany został m.in. w pracy [116].

Jednym z najskuteczniejszych podejść związanych z adaptacją parametrów jest algorytm określany jako JADE [120]. Problem adaptacji parametrów F oraz CR oparty jest (podobnie jak w większości rozwiązań adaptacyjnych) na rozkładach: normalnym oraz rozkładzie Cauchy'ego. Istotny jest natomiast nowy schemat mutacji, w którym konieczne jest wyznaczenie podzbioru osobników o najwyższej funkcji przystosowania. Sam schemat mutacji przedstawiony został następująco:

$$\forall_i, \forall_j \; U_{i,j} = S_{i,j} + F_i(S^p_{best,j} - S_{i,j}) + F_i(S_{r_1,j} - S_{r_2,j}). \tag{3.6}$$

gdzie:

i-ty indeks osobnika w populacji rodziców S,
j-ty indeks genu w genotypie danego osobnika,
r_1, r_2 - losowe indeksy osobników z populacji S,
$S_{r_1,j}$ - j-ty gen osobnika o indeksie r_1,
F_i - współczynnik mutacji dla i-tego osobnika,
$S^p_{best,j}$ - losowy osobnik z p% najlepszych osobników w populacji.

Podejście określane jako JADE porównane zostało m. in. z algorytmami jDE, SaDE oraz PSO [120]. Przewaga tej metody widoczna była dla większości funkcji testowych takich jak funkcja Branina, Goldsteina-Price'a, czy Shekela.

Wszystkie opisane w powyższym podrozdziale modyfikacje przedstawiane jako adaptacyjna ewolucja różnicowa dotyczą przede wszystkim modyfikacji współczynnika mutacji F oraz krzyżowania CR, w mniejszym stopniu natomiast zmiany liczności populacji w czasie. Należy jednak zaznaczyć, iż tylko w jednej podanej pracy przedstawiono podejście umożliwiające zwiększenie liczności populacji w kolejnych iteracjach algorytmu. Istotnym kryterium adaptacyjnej zmiany wielkości populacji w kolejnych iteracjach powinna być różnorodność populacji. Jednym z możliwych podejść jest wprowadzenie miary entropii populacji w ewolucji różnicowej. Dotychczasowe prace uwzględniające zagadnienie entropii w algorytmach metaheurystycznych dotyczyły głównie algorytmów genetycznych [60]. Bardzo ciekawym spostrzeżeniem jest fakt, iż większość technik adaptacyjnych związanych jest z rozkładami statystycznymi. Mimo tych podobieństw przewaga niektórych metod (algorytm JADE) jest widoczna. Podejścia oparte na dobieraniu wartości parametru zgodnie z wybranym rozkładem prawdopodobieństwa wydają się być niewystarczające, gdyż nie uwzględniają zmian nacisku na eksplorację i eksploatację w trakcie działania algorytmu. W autorskim rozwiązaniu [17] wprowadzono tzw. λ-modyfikację, która niezależnie od współczynnika mutacji F wpływa na zmianę położenia osobnika. Rozwiązanie to opisane zostanie szerzej w rozdziale dotyczącym autorskich modyfikacji algorytmu.

ROZDZIAŁ 4

Elementy teorii gier

Teoria gier zajmuje się badaniem optymalnego zachowania w przypadku przeciwstawnych interesów i określana jest jako matematyczna teoria konfliktu. Przez konflikt (określany jako gra) rozumiana jest tutaj pewna sytuacja, w której uczestnicy dążą do maksymalizacji własnych zysków przedstawianych w pewnych jednostkach użyteczności. Zysk danego gracza zależy od jego sposobu postępowania (strategii) oraz od decyzji innych graczy. W typowym problemie teorii gier każdy uczestnik w pewien określony sposób wpływa na końcową wartość gry. Ponadto żaden z uczestników gry nie jest w stanie sam ustalić końcowego wyniku. W tym kontekście, najistotniejszym problemem teorii gier jest taki dobór sposobu postępowania uczestników gry, aby zmaksymalizować końcowy wynik każdego z nich.

Dowolna sytuacja w teorii podejmowania decyzji, określana jako gra, powinna zawierać kilka podstawowych elementów:

- minimum dwóch uczestników gry
- zbiór zasad, według których postępują uczestnicy gry
- wynik gry, który określony jest przez kombinację sposobów postępowania uczestników gry.

Ponadto umownym założeniem jest to, iż uczestnicy konfliktu to gracze racjonalni, dążący do maksymalizacji własnych zysków.

Za datę powstania teorii gier powszechnie uznaje się rok 1944, w którym to została opublikowana monografia J. von Neumanna oraz O. Morgensterna [101]. Przy czym należy zauważyć, że fundamentalne twierdzenie dotyczące dwuosobowych gier o sumie zerowej przedstawione zostało przez J. von Neumanna już w roku 1928 [140]. Bardzo często pomijaną osobą jest tutaj E. Borel, który już w 1924 roku wskazał możliwość zastosowania mechanizmów teorii gier w ekonomii [12].

4.1 Ogólne pojęcia dotyczące gier w postaci normalnej

Pod pojęciem gry rozumiana jest dowolna sytuacja konfliktowa, w której uczestnicy starają sie wybrać najlepszą dla nich strategię postępowania. Należy tutaj zauważyć, iż w kontekście teorii gier wszystkie loterie to również gry, gdzie jeden z uczestników określany jest jako „natura" (inaczej gry przeciwko naturze). Modele takie traktowane są jednak odrębnie, gdyż gracz określany jako „natura" nierzadko podejmuje decyzje opierając się tylko i wyłącznie na prawdopodobieństwie i nie jest racjonalny. Wspomniano już na początku rozdziału, iż umownym założeniem zdecydowanej większości rozpatrywanych gier jest właśnie racjonalność graczy, czyli ich zdolność do podejmowania decyzji umożliwiających maksymalizację wypłaty jednostek użyteczności.

W pracy rozpatrywane są gry jednoetapowe n-osobowe, niekooperacyjne, w postaci strategicznej, gdzie każdy z graczy ma do wyboru pewien skończony zbiór strategii. Gra strategiczna określana jest jako macierz wypłat dla każdego z graczy. Definicją strategii jest decyzja dająca wybranej stronie pewną wypłatę określaną w jednostkach użyteczności. Formalnie, jednoetapowa gra w postaci strategicznej dla n graczy definiowana jest jako:

$$\Gamma = \langle N, \{A_i\}, C, M \rangle, \ i = 1, 2, ..., n,$$

gdzie:

- $N = \{1, 2, ..., n\}$ jest zbiorem graczy;
- $\{A_i\}$ jest skończonym zbiorem strategii dla gracza i o m strategiach;
- $C = \{c_1, c_2, ..., c_n\}$ to zbiór macierzy wypłat graczy;
- $M = \{\mu_1, \mu_2, ..., \mu_n\}$ to zbiór funkcji wypłat dla graczy.

Na rys. 4.1 pokazano przykład gry dwuosobowej, gdzie każdy z graczy ma do wyboru 3 strategie. Pierwsze 3 wartości to prawdopodobieństwa wyboru strategii pierwszego gracza, natomiast kolejne - drugiego gracza.

Należy zauważyć, iż każdy z graczy posiada własną funkcję wypłat, która dla wybranej strategii daje mu pewną liczbę jednostek użyteczności. Ponadto, wypłata i gracza jest w pewnym stopniu zależna od strategii stosowanych przez innych graczy. Niekooperacyjność graczy oznacza sytuację, gdzie każdy z nich stara się indywidualnie maksymalizować własną funkcję wypłaty. W takich warunkach gra jest rozumiana jako równoczesny wybór strategii przez wszystkich graczy. W odróżnieniu od gier z niepełną informacją, każdy z uczestników dokładnie zna wszystkie możliwości ruchu swoich przeciwników - jest to gra z pełną informacją. W najprostszym przypadku, strategia danego gracza określana jest jako czysta i oznacza wybór strategii z prawdopodobieństwem równym 1. Bardziej złożonym zagadnieniem jest strategia mieszana gracza. Nierzadko zdarza się, że rozpatrywana

	Y1	Y2	Y3
X1	0 ; 0	0.625 ; 1	0.125 ; 0.25
X2	1 ; 0.625	0 ; 0	0.125 ; 0.375
X3	0.25 ; 0.125	0.375 ; 0.125	0.25 ; 0.25

Rysunek 4.1: Przykład gry dwuosobowej

gra nie ma rozwiązań w zbiorze strategii czystych i konieczne jest zastosowanie rozkładu prawdopodobieństwa nad zbiorem strategii A'_i, gdzie $A'_i \subset A_i$. W niektórych przypadkach zbiór strategii A'_i może pokrywać się ze zbiorem A_i. Oznacza to, iż każda strategia gracza i ma niezerowe prawdopodobieństwo wyboru. Powyższa sytuacja może zostać opisana następującym wzorem:

$$\vec{a_i} = (P(a_{i_1}), P(a_{i_2}), ..., P(a_{i_m})), \tag{4.1}$$

gdzie:

$\vec{a_i}$ - rozkład prawdopodobieństwa nad zbiorem strategii gracza i,

$P(a_{i_1})$ - prawdopodobieństwo wyboru strategii 1 przez gracza i.

Ponadto, strategia czysta może zostać przedstawiona jako szczególny przypadek strategii mieszanej, gdzie prawdopodobieństwo wyboru danej strategii i gracza jest równe 1. Mając to na uwadze, możliwe jest zdefiniowanie równoczesnego wyboru strategii mieszanych wszystkich graczy. Jest to profil strategii mieszanych wszystkich graczy i może zostać określony jako uporządkowany zbiór rozkładów prawdopodobieństwa nad zbiorem wszystkich graczy. Wymagane jest tutaj uporządkowanie, ponieważ pierwsza część profilu odnosi się do gracza pierwszego, kolejna część do gracza drugiego itd. Zgodnie z wcześniejszymi definicjami, każda część profilu oznaczana jest jako $\vec{a_i}$. Matematyczny zapis profilu wygląda następująco:

$$a = (\vec{a_1}, ..., \vec{a_n}), \tag{4.2}$$

gdzie: n jest liczbą graczy.

Profil strategii utożsamiany jest także z pojęciem układu strategii, które to związane jest bezpośrednio z koncepcją równowag. W dalszej części rozprawy stosowane będzie pojęcie układu strategii. Jednocześnie należy zaznaczyć, iż przy bardziej szczegółowych opisach równowag konieczne jest także operowanie pojęciem wykluczenia gracza. Sytuacja taka dotyczy układu strategii, gdzie jeden z graczy nie stosuje strategii mieszanej, tylko strategię czystą. Zależność taka dana

jest wzorem:

$$a_{-i} = (\vec{a_1}, ..., \vec{a_{i-1}}, \vec{a_{i+1}}, ..., \vec{a_n}), \tag{4.3}$$

gdzie: i - indeks gracza stosującego strategię czystą.
Najczęściej w rozwiązaniach stosowana jest część strategii poszczególnych graczy. Pozostałe strategie mają zerowe prawdopodobieństwo wyboru. Z kolei podzbiór strategii danego gracza mających niezerowe prawdopodobieństwo wyboru określany jest jako wsparcie:

$$\forall_j, a_{i_j} \in M_{a_i}, P(a_{i_j}) > 0, \tag{4.4}$$

gdzie:

j - indeks strategii gracza,
M_{a_i} - wsparcie i-tego gracza,
a_{i_j} - j-ta strategia i-tego gracza.

Bardzo często wsparcie danego gracza określane jest też jako zbiór strategii aktywnych. Każdy z graczy dysponuje własną macierzą wypłat, chociaż najczęściej przedstawiane są one jako jedna, wielowymiarowa macierz. W każdej komórce macierzy jest n wartości, gdzie n jest liczbą graczy. Na podstawie macierzy wypłat i funkcji wypłaty przypisanej do każdego gracza możliwe jest określenie jednostek użyteczności przyznanych graczowi:

$$\forall_i, \mu_i : c_i \to \mathbb{R} \tag{4.5}$$

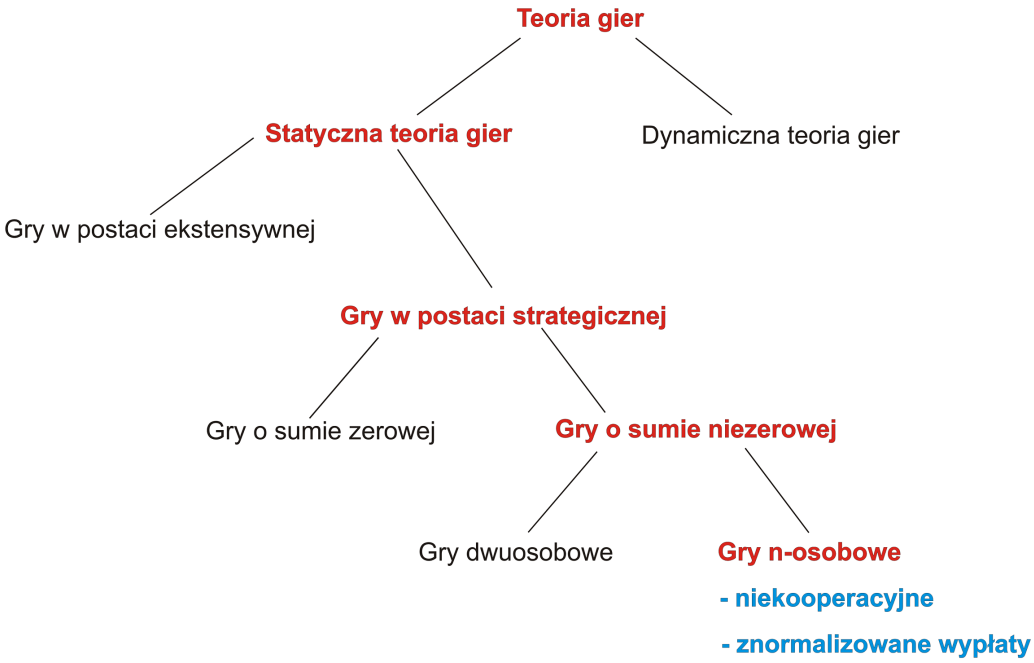
gdzie:

i - indeks gracza,
c_i - macierz wypłat i-tego gracza,
μ_i - funkcja wypłaty i-tego gracza.

Istotny jest tutaj fakt, że układ strategii każdemu graczowi przypisuje inną wartość funkcji wypłaty.

4.2 Klasyfikacja gier

Teoria gier jest dziedziną bardzo rozległą i nawet zawężenie tego pola do wybranych typów gier nie ułatwia analizy istniejących problemów. W podrozdziale tym przedstawiony zostanie szereg różnych rodzajów gier podzielonych w zależności od pewnych ich własności. Sam podział wynika bezpośrednio z tematu rozprawy i pozwala ją zawęzić. Na rys. 4.2 wskazano ogólne zależności pomiędzy typami gier. Jest to rysunek poglądowy i nie oddaje wszystkich zależności pomiędzy typami gier. W szczególności nie ujęto tutaj problemu związanego z grami jedno- oraz wieloetapowymi. Zdecydowana większość gier w postaci ekstensywnej to gry wieloetapowe.



Rysunek 4.2: Ogólny podział gier

W dalszej części podrozdziału, o ile autor nie określił inaczej, pojęcie równowagi dotyczy równowagi Nasha i jest równoznaczne ze znalezieniem rozwiązania w grze. Równowaga Nasha jest układem strategii, w którym żaden z graczy znając sposób postępowania swoich przeciwników nie może zyskać więcej jednostek użyteczności odstępując od swojej wybranej strategii. Definicja ta szczegółowo zostanie przedstawiona w podrozdziale 4.4. Podział gier zaproponowany w dalszej części rozdziału w pewnym stopniu bazuje na podziale z pozycji [113]. Samą klasyfikację gier można podzielić następująco:

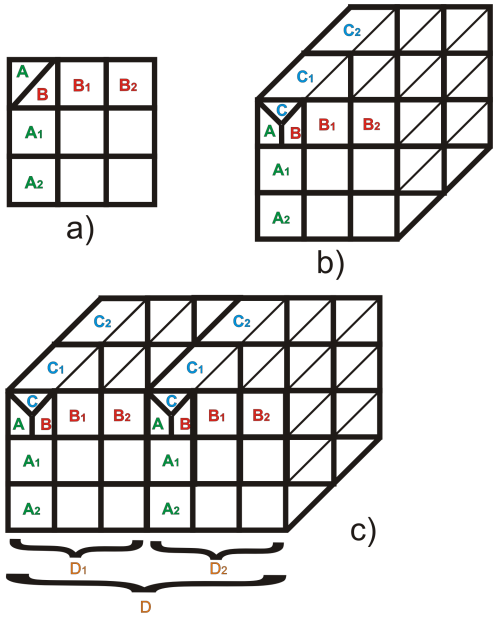
- ze względu na kolejność podejmowania decyzji:
  ▲ Gry w postaci strategicznej (normalnej) Opisują sytuacje, w których gracze podejmują decyzje jednocześnie, bez wiedzy o decyzjach innych uczestników gry. Często stosowane jest też określenie: gry macierzowe, gdyż oczekiwany zysk każdego z graczy określany jest na podstawie macierzy zawierającej jednostki użyteczności. Wielkość macierzy wypłat bezpośrednio zależy od liczby strategii stosowanych przez graczy oraz od liczby graczy. W szczególności drugi element wpływa wykładniczo na rozmiar macierzy. Klasyczne algorytmy wyszukujące punkty równowagi w grach macierzowych opierają się na przeglądzie macierzy, stąd już znalezienie rozwiązania w grach 3-osobowych jest wyjątkowo trudne. Problem wyszukiwania równowag w grach macierzowych jest jednym z kluczowych zagadnień niniejszej roz-

prawy i dokładniej opisany zostanie w dalszych rozdziałach. W poprzednim podrozdziale natomiast przedstawione zostały najważniejsze pojęcia związane z tym typem gier.
▲ Drugim typem są gry w postaci ekstensywnej (inaczej rozwiniętej), w których gracze podejmują decyzje sekwencyjnie, w kolejnych chwilach czasu, mając określone informacje o decyzjach innych graczy (i swoich) w poprzednich chwilach czasu. Określane są najczęściej jako drzewo gry i stosowane przy 2-osobowych grach takich jak szachy, warcaby, czy go. Ruchy graczy wykonywane są naprzemiennie, a każdy z nich w danej chwili zna wszystkie swoje możliwości ruchu. Każda gra w postaci ekstensywnej może zostać przekształcona w grę w postaci strategicznej.

- ze względu na posiadaną wiedzę:
  ▲ Gry z kompletną informacją, gdzie zakłada się, że każdy z graczy zna w pełni zasady gry, funkcje wypłat oraz zbiory możliwych strategii wszystkich pozostałych graczy. Nierzadko z pojęciem gry z kompletną informacją pojawia się wspomniane wcześniej założenie o racjonalności graczy, gdzie każdy z uczestników dąży do maksymalizacji zysku. W przypadku gier ekstensywnych, gracze w każdej chwili posiadają pełną informację o poprzednich decyzjach innych graczy i o ewentualnych ich posunięciach losowych.
  ▲ Gry z niekompletną informacją określane także jako gry bayesowskie. W zależności od typu (strategiczne lub ekstensywne) są definiowane nieco inaczej. Dla gier strategicznych L. Kockesen definiuje jawnie dodatkową funkcję prawdopodobieństwa, która w zależności od wartości określa końcową wypłatę graczy [82]. Dla uproszczenia można przyjąć, że gry strategiczne z niepełną informacją posiadają więcej niż jedną macierz wypłat. Dana macierz wypłat graczy losowana jest w zależności od funkcji prawdopodobieństwa.

- ze względu na liczbę graczy (liczba uczestników gry ma tutaj kluczowe znaczenie ze względu na możliwości stosowania algorytmów przybliżonych):
  ▲ W przypadku gier dwuosobowych w postaci normalnej macierz wypłat jest dwuwymiarowa, a gracze najczęściej określani są jako gracz kolumnowy oraz gracz wierszowy (każdy z nich posiada szereg strategii określanych jako kolumny oraz wiersze). Ponadto, w pewnych szczególnych przypadkach możliwe jest znalezienie rozwiązania dokonując przeglądu zupełnego wszystkich strategii obydwu graczy. Nierzadko stosuje się tutaj określenie dwuosobowych gier o sumie ogólnej, które opisują gry o sumie niezerowej (określane jako gry bimacierzowe), a także ich szczególny przypadek - gry o sumie zerowej. Z kolei gra dwuosobowa w postaci ekstensywnej może np. odnosić się do partii szachów lub dowolnej innej gry dla dwóch osób. T. Tyszka w swojej książce [135] stosuje tutaj określenie gier antagonistycznych, gdzie wygrana jednego gracza stanowi stratę jednostek użyteczności jego przeciwnika.

▲ W niniejszej rozprawie autor skupia się na grach n-osobowych w postaci normalnej, gdzie liczba graczy jest większa od 2. Problem wyszukiwania równowag jest już znany od dawna, a jedną z pierwszych prób określenia rozwiązania przedstawiono w pracy [143] i bazowała na rozszerzeniu klasycznego algorytmu dla gier 2-osobowych. Sam problem n-osobowych gier w postaci normalnej jest często marginalizowany, a proponowane rozwiązania nierzadko mają charakter czysto matematyczny. Dobre źródło informacji na temat n-osobowych gier w postaci ekstensywnej stanowi pozycja [30].



Rysunek 4.3: Przykład gier w postaci normalnej: a) gry 2-osobowej, b) gry 3-osobowej, c) gry 4-osobowej, gdzie każdy z graczy ma do dyspozycji dwie strategie.

Istniejące algorytmy dotyczące wyszukiwania równowag w grach n-osobowych wyraźnie rozgraniczają problem wzrostu stopnia trudności gry poprzez powiększenie zbioru graczy, a powiększenie liczby strategii czystych dla jednego gracza. Proponowany w pracy algorytm pozwala traktować te dwa elementy równorzędnie, co przekłada się na możliwość zwiększenia zbioru graczy bez wykładniczego wzrostu złożoności.

- ze względu na zbiory dostępnych akcji;
  ▲ Gry nieskończone, określane też jako gry z kontinuum akcji. Stosowane najczęściej do opisu sytuacji ekonomicznych.
  ▲ Z punktu widzenia niniejszej rozprawy, istotniejszy jest drugi typ gier - gry o skończonym zbiorze strategii. Częstym założeniem jest w tym przypadku to, iż każdy z graczy posiada równoliczny zbiór strategii. Jak już zostało

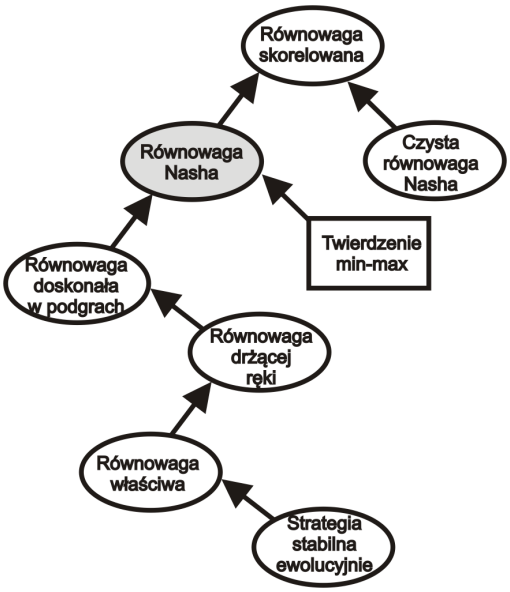
wspomniane, tego typu gra, w szczególności dla n graczy tworzy n wymiarową macierz, gdzie liczba jednostek użyteczności danego gracza zależy od wybranych strategii wszystkich graczy.

- ze względu na możliwość tworzenia koalicji;
  ▲ Gry kooperacyjne (koalicyjne) - gdy akcje przypisywane są grupom (koalicjom) graczy. W przeważającej większości, wszystkie rozpatrywane problemy dotyczą gier niekooperacyjnych. Z kolei w przypadku gier n-osobowych możliwe jest określenie gier kooperacyjnych, gdzie w skończonym zbiorze graczy możliwe jest wydzielenie podzbiorów graczy, którzy współpracują ze sobą.
  ▲ W przeciwieństwie do gier koalicyjnych (kooperacyjnych), gry niekooperacyjne zakładają zawsze, że interesy graczy są przeciwne. Istotnym założeniem przy grach niekooperacyjnych jest tutaj samo zachowanie poszczególnych graczy (zwłaszcza w przypadku gier o sumie niezerowej, gdzie interesy graczy nie muszą być przeciwstawne). Najczęściej, w takiej sytuacji zakłada się racjonalność graczy. Problem zachowania graczy opisany został w [114]. Istnieje też pojęcie superracjonalności graczy, które jednak nie jest przedmiotem szczegółowych badań. Ciekawym przypadkiem superracjonalnych graczy może być problem określany jako "Dylemat więźnia", gdzie obydwu skazanych w tej sytuacji stosuje strategię "współpracuj". Koncepcja superracjonalności graczy opisana została w [71]. Założenie o racjonalności graczy bardzo często jest pomijane, należy jednak zaznaczyć, że dla nieracjonalnego gracza każda ze strategii czystych ma takie samo prawdopodobieństwo wyboru. Najpopularniejszym przykładem tego gracza jest, najczęściej w grach wieloosobowych, los.

- ze względu na powtarzalność;
  ▲ W przypadku gier iterowanych można wyróżnić gry ze skończonym i nieskończonym horyzontem czasowym.
  ▲ Mnogość problemów w przypadku gier iterowanych stanowi istotny element teorii gier. Powyższy typ dotyczy gier powtarzanych wielokrotnie, gdzie końcowa wartość wypłaty dla poszczególnych graczy zależy w pewnym stopniu od ich działań podejmowanych na kolejnych etapach gry. Powyższe definicje podane zostały, aby zachować pewną ciągłość rozprawy, jednak podstawowym zamierzeniem autora jest przedstawienie sposobu rozwiązania gier jednoetapowych, które stanowią przeciwieństwo gier powtarzalnych. Każda gra jednoetapowa jest sytuacją, w której każdy z uczestników podejmuje decyzję równocześnie, a końcowa wartość jednostek użyteczności zależy tylko i wyłącznie od niej. Najistotniejszym problemem w przypadku gier jednoetapowych jest ograniczony rozmiar możliwych do rozwiązania gier, a także trudność w wyznaczeniu elementów gry świadczących o jej złożoności.

4.3 Typy równowag i ich właściwości

W poprzednim podrozdziale omówiona została szczegółowo klasyfikacja typów gier. Pojęcie równowagi jest jednym z najistotniejszych elementów teorii gier, a znalezienie układu strategii będących w równowadze stanowi często poważny problem. W zależności od typu gry, istniejący punkt równowagi powinien posiadać pewne określone cechy, przy czym znalezienie układu strategii spełniających dokładnie te założenia jest często prawie niemożliwe. W tym podrozdziale przedstawiona zostanie szczegółowa klasyfikacja i analiza istniejących równowag dla konkretnych typów gier. Możliwe jest podanie typów równowag stanowiących nadklasy dla bardziej szczegółowych koncepcji. Ogólna zależność przedstawiona została na rys. 4.4. Strzałki wskazują kierunek zawierania się poszczególnych elementów. Przykładowo, koncepcja równowagi Nasha pochodzi od bardziej ogólnego pojęcia równowagi skorelowanej.



Rysunek 4.4: Przykład gry dwuosobowej

Pierwszą grupę stanowią: równowaga skorelowana (ang. *correalted equilibrium*) oraz równowagi racjonalizowane (ang. *rationalizable equilibrium/ rationalizability*). Kolejną ważną grupę stanowią elementy pochodne od równowagi Nasha, takie jak równowaga „drżącej ręki" (ang. *Trembling Hand equilibrium*). Ich znalezienie często wymaga przyjęcia dodatkowych założeń jednak zasadniczo związane jest również z typami gier, w których występuje klasyczna równowaga Nasha. W 1989 roku I. Gilboa w artykule [54] rozwinął problem wyszukiwania równowag Nasha z dodatkowymi właściwościami, takimi jak minimalna liczba aktywnych strategii. Problemy te szczegółowo opisane zostaną rozdziale 8 niniejszej rozprawy.

Jednym z najważniejszych nadzbiorów równowagi Nasha jest równowaga skorelowana, wprowadzona przez R. Aumanna w 1974 roku [8]. Podstawowym założeniem równowagi skorelowanej jest wprowadzenie dodatkowego, zewnętrznego informatora, przesyłającego publiczny sygnał, który wpływa na decyzje poszczególnych graczy. Najpopularniejszym zabiegiem jest dodanie zmiennej losowej, której wartość wpływa na decyzję graczy. W [67] podana została również definicja dla równowagi skorelowanej w grach w postaci ekstensywnej, a także szczegółowa analiza gry macierzowej, zaproponowanej pierwotnie przez R. Aumanna. Jedną z zalet równowag skorelowanych jest ich niewielka złożoność obliczeniowa, ponieważ można ją wyznaczyć rozwiązując układ równań liniowych. Więcej szczegółów na temat złożoności opisywanej równowagi można znaleźć w [56]. Kolejnym podejściem zaproponowanym przy wyznaczeniu równowag skorelowanych była gra wieloetapowa, w której gracze stosują strategie zależne od poprzednich posunięć przeciwnika. Takie podejście skutkuje zbieżnością do wspomnianej równowagi [50].

Na rys. 4.4 wskazano także zależność pomiędzy równowagą Nasha oraz twierdzeniem von Neumanna o minmaksie [140]. Twierdzenie to dotyczy 2-osobowych gier o sumie zerowej. Każda gra macierzowa ma rozwiązanie [142]. Zgodnie z nim istnieje dokładnie jedna liczba nazywana wartością gry oraz optymalne strategie (czyste lub mieszane) obu graczy oraz takie, że:

- Jeżeli pierwszy gracz stosuje swoją optymalną strategię, to jego oczekiwana wygrana będzie większa lub równa niezależnie od strategii przeciwnika.
- Jeżeli drugi gracz stosuje swoją optymalną strategię, to oczekiwana wygrana będzie mniejsza lub równa niezależnie od strategii pierwszego gracza.

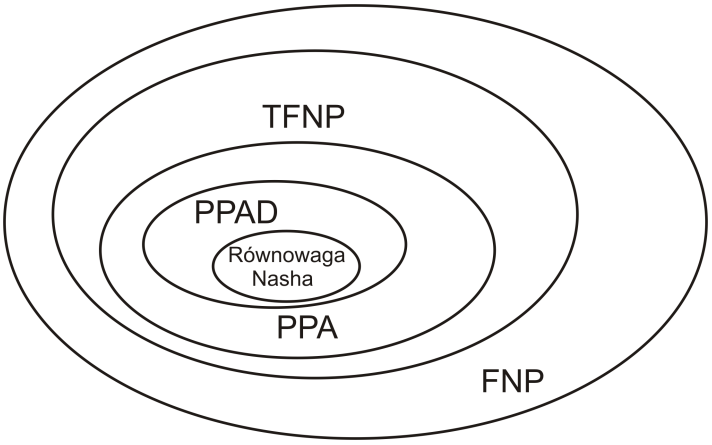
W przypadku gier 2-osobowych o sumie zerowej może również istnieć rozwiązanie w strategiach czystych (gdzie żaden z graczy nie stosuje strategii mieszanej). W takiej sytuacji rozwiązanie określane jest jako punkt siodłowy, a jego znalezienie w dowolnej grze ma liniową złożoność obliczeniową. W takim przypadku braku rozwiązania w strategiach czystych konieczne jest wyznaczenie strategii mieszanych. Podejście do tego problemu oparte na ewolucji różnicowej przedstawione zostało w [14].

Dwie opisane powyżej koncepcje bezpośrednio związane są z pojęciem równowagi Nasha i stanowiły dla niej niejako nadklasę. Pozostałe typy równowag przedstawione na rys. 4.4 nie są bezpośrednio związane z tematem rozprawy i zostały przedstawione tylko w celach informacyjnych.

4.4 Równowagi Nasha w grach n-osobowych

Przedstawiony w rozprawie problem wyszukiwania równowag Nasha w grach o sumie niezerowej jest istotnym zagadnieniem w teorii gier. J. Nash wykazał,

że równowaga istnieje w każdej grze [104]. Problem znalezienia równowagi należy do klasy PPAD (ang. *Partial Parity Argument Directed*) [32] i określany jest jako jeden z najistotniejszych w teorii gier [111]. PPAD Jest klasą zawierającą problemy podejrzane o przynależność do klasy NP, a wprowadzona została po raz pierwszy w artykule C. Papadimitriou [112]. Zależność pomiędzy problemem wyszukiwania równowag Nasha a klasami złożoności przedstawia rys. 4.5.



Rysunek 4.5: Zależności pomiędzy klasami złożności

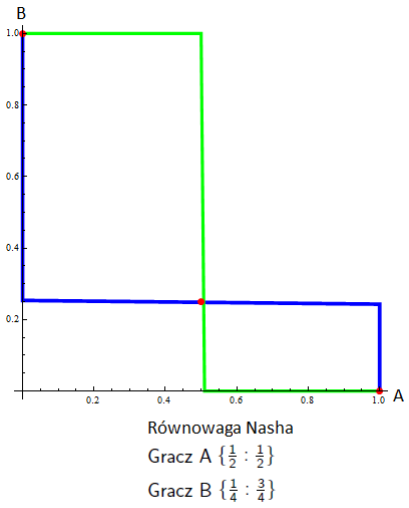
Klasa PPAD jest podklasą PPA (ang. *Polynomial Parity Argument*), do której należy inny problem związany z teorią gier: równowaga Arrow-Debreu [6]. Odwołania do powyższych klas złożoności wydają się dość istotne, gdyż zarówno klasa PPAD, jak i PPA należą do klasy TFNP [96]. Jest to odpowiednik klasy NP dla problemów funkcyjnych, gdzie mając daną wartość x oraz predykat postaci $F(x, y)$, znaleźć y spełniający $F(x, y)$.

Równowaga Nasha w grze n-osobowej jest układem strategii, w którym żaden z graczy znając strategię przeciwników nie zyskuje odstępując od wybranej strategii. W tabeli 4.1 przedstawiono przykład gry 2-osobowej. Wizulizacja równowagi Nasha dla tej gry przedstawiona została na rys. 4.6.

Tabela 4.1: Prosta gra 2-osobowa, gdzie każdy z graczy posiada 2 strategie

strategie	B1	B2
A1	1:4	0:6
A2	4:1	-1,-1

Na osi odciętych zaznaczone jest prawdopodobieństwo wyboru pierwszej strategii gracza B. Prawdopodobieństwo drugiej strategii obliczone jest jako $1 - P(B1)$. Analogicznie obliczane są wartości prawdopodobieństw dla gracza A. For-



Rysunek 4.6: Graficzna reprezentacja równowagi Nasha w grze 2-osobowej

malnie mieszana równowaga Nasha definiowana jest następująco:

$$\forall_i, \forall_j, \mu_i(a) \geqslant \mu_i(a_{i_j}, a_{-i}), \tag{4.6}$$

gdzie:

i - oznacza i-tego gracza,
j - jest numerem strategii danego gracza,
$\mu_i(a)$ - wypłata i-tego gracza dla profilu strategii a,
$\mu_i(a_{i_j}, a_{-i})$ - wypłata i gracza stosującego strategię j przeciwko a_{-i}.

Powyższy wzór odnosi się do równowagi Nasha określanej też czasami jako słaba równowaga. W sytuacji, gdy zamiast słabej, obecna jest ostra nierówność. Równowaga taka określana jest jako silna. Do tej pory wykazano, że dla opisywanego problemu nie istnieje algorytm o złożoności wielomianowej (nawet w przypadku gier 2-osobowych). Dowolna para macierzy nie zmienia się podczas przemnożenia ich przez stałą, dlatego powszechnym założeniem jest to, iż elementy macierzy wypłat graczy normalizowane są do przedziału $\langle 0, 1\rangle$ [91]. Przy powyższych założeniach, dla dowolnego ϵ, gdzie $\epsilon > 0$, przybliżona równowaga Nasha (ϵ-równowaga lub też równowaga aproksymacyjna) dla n graczy definiowana jest następująco:

$$\forall_i, \forall_j, \mu_i(a) \geqslant \mu_i(a_{i_j}, a_{-i}) - \epsilon, \tag{4.7}$$

gdzie: ϵ - jest wartością przybliżenia.
Taka równowaga przybliżona posiada szereg właściwości. Przede wszystkim każda

równowaga Nasha otoczona jest przez ϵ-równowagi. Zależność ta jednak nie jest spełniona w drugą stronę, to znaczy dana ϵ-równowaga nie musi znajdować się w pobliżu równowagi Nasha [90].

Z punktu widzenia niniejszej rozprawy jednym z najistotniejszych wniosków jest to, iż nawet znalezienie przybliżonej równowagi Nasha należy do klasy PPAD [37]. Ponadto przekształcenie problemu do gry typu Win-Lose nie upraszcza problemu [2]. Gry typu Win-Lose, inaczej określane także jako gry typu 0-1, to gry w postaci strategicznej, gdzie wszystkie elementy macierzy wypłat zastąpione zostały tylko dwoma wartościami: 0 oraz 1. W powyższym artykule przedstawiono wielomianowy algorytm umożliwiający przekształcenie dowolnej gry do postaci $0-1$. Z kolei rozważania na temat złożoności problemu dotyczącego gier typu $0-1$ przedstawione zostały również w [27].

Znalezienie czystej równowagi Nasha jest trywialne (w przeciwieństwie do równowag mieszanych). Gry o sumie zerowej mogą być rozpatrywane jako pewna klasa gier o sumie niezerowej. Taki typ gier z powodzeniem może zostać rozwiązany przez metodę sympleksu. Obecnie wiadomo, że nawet gry dla dwóch graczy należą do klasy problemów PPAD-kompletnych [32]. Analogiczna zależność wykazana została wcześniej również dla gier 3-osobowych [24].

Istnieje zależność pomiędzy wsparciem danego gracza oraz oczekiwanymi wypłatami graczy, I. Milchtaich w artykule [99] podał sposób wyznaczania oczekiwanych wypłat przy założeniu, że wsparcie obydwu graczy jest dane równaniem:

$$\forall_i \frac{\delta - a}{\delta - b} = \frac{|C_i - a \cdot E|}{|C_i - b \cdot E|}, \tag{4.8}$$

gdzie:

E - macierz w całości wypełniona jedynkami,
a, b - dowolne liczby, $a \neq b$,
R - macierz wypłat gracza X,
$|R|$ - wyznacznik macierzy R,
δ - oczekiwana wypłata danego gracza.

W artykule [15] autor wskazał na istnienie zależności pomiędzy zbiorem strategii czystych gracza a równowagą Nasha dla 2 graczy.

Interesującym typem równowagi jest równowaga Nasha z pełnym wsparciem (ang. *Well supported Nash equilibrium*), gdzie podstawowym założeniem jest to, iż każda strategia czysta danego gracza ma niezerowe prawdopodobieństwo wyboru. Problem określenia maksymalnego możliwego przybliżenia, czyli określenia wartości ϵ, rozwiązany został w [84].

W niniejszej rozprawie kluczowym elementem jest problem wyszukiwania mieszanych równowag Nasha, a także przybliżonych równowag Nasha (ϵ-równowag). Wyszukiwanie czystych równowag oraz równowag z danym wsparciem nie jest kosztowne obliczeniowo, a istniejące algorytmy (np. przeglądu zupełnego strategii

w przypadku czystych równowag Nasha) dobrze spełniają swoje zadanie. Istotne z naukowego punktu widzenia jest zaproponowanie metody dającej rozwiązanie w strategiach mieszanych dla n graczy, gdzie $n \geqslant 3$. Złożoność problemu rośnie wykładniczo ze względu na liczbę graczy i już dla $n = 4$ zdecydowana większość istniejących algorytmów daje rozwiązania nieakceptowalne ze względu na czas obliczeń.

ROZDZIAŁ 5

Algorytmy dla problemu wyszukiwania równowag Nasha

Kluczowym problemem teorii gier jest wyznaczanie równowag Nasha. Istnieje wiele algorytmów, które spełniają to zadanie. Pomimo faktu, iż wyszukiwanie równowag w grach 2-osobowych jest dużo łatwiejsze, to stanowi poważny problem. Istnieje wyraźne rozgraniczenie dla algorytmów dostosowanych do obydwu rodzajów gier. Chociaż gry 2-osobowe stanowią tylko szczególny przypadek gier n-osobowych, to sposoby ich rozwiązywania są odmienne. Pierwsze algorytmy dla gier 2-osobowych to przede wszystkim algorytm Lemke-Howsona [89] i jego modyfikacje. Ponadto, istnieje szereg metod opartych na enumeracji wsparć, w tym także podejścia równoległe. Jednym z najskuteczniejszych, lecz jednocześnie najbardziej złożonych obliczeniowo jest algorytm oparty na przeglądzie zupełnym zbiorów strategii graczy. Wreszcie istnieje też bardzo duża grupa algorytmów przybliżonych, które gwarantują określoną dokładność rozwiązania. Pewną podgrupę stanowią tutaj algorytmy oparte na metodach matematycznych pozwalające sprowadzić grę bimacierzową do jednej macierzy, która stanowi grę o sumie zero, rozwiązywalną w czasie wielomianowym.

Gry n-osobowe są zdecydowanie bardziej problematyczne. Najtrudniejszym elementem wydaje się w tym wypadku błyskawicznie rosnąca ze względu na liczbę graczy złożoność problemu. Już 4-osobowe gry nierzadko uniemożliwiają wygenerowanie jakiegokolwiek rozwiązania przez istniejące algorytmy w rozsądnym czasie. Niestety stopień trudności poszczególnych gier o takiej samej liczbie graczy i strategii aktywnych jest bardzo różny. Dwie wygenerowane macierze o identycznej wielkości mogą mieć zupełnie inny stopień skomplikowania. Istotne wydaje się być przedstawienie algorytmu umożliwiającego generowanie powtarzalnych wyników.

Algorytmy oparte na metaheurystykach wydają się być w tym wypadku marginalizowane. Jedną ze stosowanych w teorii gier technik jest przeszukiwanie tabu (ang. *tabu search*) stosowana do generowania czystych równowag Nasha. Należy

przy tym zauważyć, że rozwiązanie zawierające tylko równowagi czyste może zostać uzyskane poprzez przegląd wszystkich strategii czystych poszczególnych graczy (co z kolei nie jest kosztowne obliczeniowo).

W rozdziale tym przedstawione zostaną najistotniejsze algorytmy dotyczące wyszukiwania równowag Nasha z uwzględnieniem rozwiązań zawierających równowagi czyste. Ponadto jednym z założeń autora niniejszej rozprawy było opracowanie algorytmu ukierunkowanego na wyszukiwanie tylko i wyłącznie równowag mieszanych, co jest zagadnieniem dalece bardziej złożonym.

5.1 Algorytmy dla gier 2-osobowych

Wyszukiwanie równowag Nasha przeważnie określane jest jako jedno z najistotniejszych zagadnień z teorii gier. Koncepcja ta dotyczy gier o sumie niezerowej, należy jednak pamiętać, iż pierwsze opracowane algorytmy dotyczyły w tym przypadku gier o sumie zerowej. Te z kolei określane są jako podklasa gier o sumie niezerowej. Zgodne z tą definicją, pierwszym algorytmem dotyczącym gier o sumie zerowej był wspomniany wcześniej minmax [140].

W tej części rozdziału wymienione zostaną najważniejsze metody dotyczące gier 2-osobowych. Podane zostaną również szczegóły dwóch najistotniejszych algorytmów. Pierwszy to, wspomniany we wstępie, algorytm Lemke-Howsona uważany za podstawowy algorytm stosowany do rozwiązywania gier dla dwóch graczy. Ważną grupę metod stanowią również algorytmy przybliżone. Należy jednak wspomnieć, iż pierwsze podejście tego typu stosowane było również w grach o sumie zero.

Dla gier o sumie zerowej dopiero kilkadziesiąt lat później wykazano interesującą zależność, iż każdy gracz posiada strategię bliską optimum stosując liczbę strategii równą $\ln n$, gdzie n jest liczbą strategii czystych gracza [92]. Powyższe podejście zakłada arbitralne wybranie określonego zbioru strategii czystych dla danego gracza. W [142] przedstawiony został algorytm przybliżony oparty na częstości wyboru strategii. Pseudokod algorytmu przedstawiony został w alg. 4. Jego niewątpliwą zaletą jest prostota oraz szybkość działania. Już po kilku iteracjach otrzymane rozwiązanie jest zbliżone do optymalnego.

Omówiony powyżej algorytm dla gier o sumie zerowej jest bardzo szybki a sam problem wyznaczania równowag nie jest kosztowny obliczeniowo. Najistotniejszym problemem jest tutaj znalezienie rozwiązania dla gier o sumie niezerowej. W dalszej części rozdziału przedstawione zostaną algorytmy dotyczące tego zagadnienia.

Najpopularniejszym algorytmem jest nieco przestarzały już algorytm Lemke-Howsona (określany także czasami jako algorytm Lemke'go-Howsona) [89]. Algorytm ten związany jest z problemem komplementarności liniowej a sama jego idea może zostać przedstawiona w kilku prostych krokach:

Algorytm 4: Algorytm przybliżony

1 Wybierz dowolną strategię dla gracza wierszowego N_{1i} z macierzy c;
2 Skopiuj wybrany wiersz poniżej macierzy i zaznacz najmniejszą wartość;
3 Wybierz kolumnę dla gracza kolumnowego N_{2j}, która to znajduje się nad najmniejszą wartością wybraną w poprzednim kroku;
4 Skopiuj strategię N_{2j} na prawo od macierzy i zaznacz największą wartość w przepisanej kolumnie;
5 **while** *dopóki kryterium końca nie jest spełnione* **do**
6 Do wierszy pod macierzą dodaj wiersz N_{1k} (wskazany przez największą wartość z kolumny N_{2j});
7 Do kolumn obok macierzy dodaj kolumnę znajdującą się nad najmniejszą wartością w wierszu N_{1k};
8 powrót do kroku 6;

- ze zbioru strategii graczy należy usunąć wszystkie strategie zdominowane
- dla pozostałych strategii należy rozważyć wszystkie możliwe podzbiory, i dla tych podzbiorów:
  - rozwiązać układy równań oraz na ich podstawie określić, które strategie mogą być w równowadze
  - sprawdzić, czy wskazane strategie są rzeczywiście w równowadze.

Algorytm Lemke-Howsona posiada wykładniczą złożoność obliczeniową, a swoim działaniem przypomina algorytm sympleks. W pierwszej iteracji algorytmu danymi wejściowymi jest sztuczny punkt równowagi o wartości 0. Po znalezieniu równowagi Nasha możliwe jest powtórne wywołanie metody, gdzie punktem startu jest nowa równowaga. W ten sposób możliwe jest wyznaczenie kilku rozwiązań, jednak nie ma gwarancji, że wyznaczone zostaną wszystkie. W tabeli 5.1 przedstawiono zestawienie najpopularniejszych koncepcji i modyfikacji algorytmów dla gier 2-osobowych.

Najnowsze algorytmy przybliżone dla gier dwuosobowych opisane zostały w pracy [20]. Pierwszy z nich stanowi rozwinięcie metody opisanej powyżej, jednak określany jest przez autorów jako znacznie prostszy. Drugi z algorytmów pozwala uzyskać przybliżenie $\epsilon \approxeq 0.364$. W przypadku obydwu algorytmów podstawową koncepcją jest przekształcenie gry bimacierzowej w grę o sumie zero:

$$A = R - C, \tag{5.1}$$

gdzie: A - nowa macierz,
R i C - macierze wypłat poszczególnych graczy.
Przekształcenie gry bimiacierzowej w grę macierzową pozwala na zastosowanie

programowania liniowego do wyznaczenia strategii optymalnych graczy. W przypadku, gdy uzyskany wynik jest mniejszy od założonego przybliżenia 0.364, rozwiązaniem jest powyższy układ strategii. W przeciwnym wypadku dla jednego z graczy konieczne jest wyznaczenie najlepszej odpowiedzi na obliczoną w poprzednim kroku strategię gracza drugiego. Strategia drugiego gracza wyznaczana jest na drodze kilku dodatkowych przekształceń.

Tabela 5.1: Tabela przedstawiająca najpopularniejsze algorytmy i ich modyfikacje dla problemu wyszukiwania równowag w grach 2-osobowych. Zestawienie podzielone zostało na algorytmy dla gier o sumie zero, gier o sumie niezerowej oraz algorytmy przybliżone

Algorytmy dla gier o sumie zero
1945 r. Twierdzenie o minmaksie dla funkcji liniowych [141]
1958 r. Uogólnienie dla twierdzenia o minmaksie [126]
1964 r. Programowanie liniowe i algorytm sympleks do wyznaczania rozwiązań [139, 109]
1966 r. Wyszukiwanie punktu siodłowego (stosowanie przez graczy strategii czystych) [142]
1966 r. Graficzna metoda eliminacji strategii zdominowanych i dominujących [142]
Algorytmy dla gier o sumie niezerowej
1964 r. Modyfikacja Lemke-Howsona do wyszukiwania wszystkich rozwiązań w danej grze [93]
1974 r. Rozwinięcie algorytmu Lemke-Howsona dla wybranych typów gier [125]
1991 r. Przegląd zupełny możliwych wsparć - algorytm siłowy oparty na metodzie Lemke-Howsona [38]
1991 r. Enumeracja wsparć [103]
2004 r. Algorytm bazujący na przeszukiwaniu wsparć o niewielkiej liczności - skuteczny tylko dla wybranych typów gier [107]
2005 r. Algorytm losowy bazujący na metodzie Las Vegas [10]
2005 r. Metoda bazująca na programowaniu całkowitoliczbowym [29]
2006 r. Wyznaczenie wypłat graczy przy danym wsparciu [99]
2008 r. Ustalenie punktu startowego dla algorytmu Lemke-Howsona [26]
2008 r. Równoległy przegląd zbioru wsparć [63]
Algorytmy przybliżone
1991 r. Koncepcja, w której równowaga Nasha jest najlepszą odpowiedzią każdego z graczy. [52]
2006 r. Algorytm przybliżony $\epsilon = 0.5$, gracze stosują 2 strategie. Ich wartości obliczane są na podstawie najlepszej odpowiedzi [33]
2007 r. Algorytm przybliżony $\epsilon = 0.38$ umożliwiający generowanie rozwiązań z dużym wsparciem [34]
2009 r. Algorytm przybliżony $\epsilon = 0.75$, gdzie każdy z graczy stosuje tylko 2 strategie z równym prawdopodobieństwem [83]
2010 r. Porównanie algorytmu przybliżonego dla gier 2-osobowych z podejściem opartym na ewolucji różnicowej [13]
2010 r. Wyszukiwanie strategii optymalnych w grach o sumie zero - podejście oparte na ewolucji różnicowej [14]

5.2 Algorytmy dla gier n-osobowych

Dotychczas wspomniane algorytmy dotyczyły gier dwuosobowych. Liczba istniejących i skutecznych algorytmów dla gier n-osobowych, gdzie liczba graczy jest większa od 2 jest znacznie mniejsza. W grupie tej istnieją zarówno algorytmy przybliżone oparte na metaheurystykach, jak i metody dokładne. Niestety złożoność metod dokładnych nierzadko uniemożliwia zastosowanie ich w rzeczywistych problemach.

Algorytmem, który wykazuje bardzo dużą skuteczność (w stosunku do metod opartych na zmodyfikowanym algorytmie Lemke-Howsona) oraz stosunkowo szybkim jest algorytm podziału (ang. *Simplicial Subdivision*) [137]. Główna idea algorytmu dotyczy nieliniowego problemu komplementarności [75, 138], w którym należy znaleźć n-wymiarowy wektor liczb rzeczywistych spełniający pewne założenia. Podejście do tego zagadnienia opiera się na utworzeniu, przy pomocy triangulacji, siatki nad przestrzenią strategii mieszanych graczy. Kolejnym etapem algorytmu jest procedura przejścia ścieżkami tak utworzonej siatki (analogicznie, jak w przypadku algorytmu Lemke-Howsona) w celu wyznaczenia przybliżonego rozwiązania. Takie rozwiązanie może zostać następnie użyte jako punkt startu kolejnej iteracji algorytmu. Najpopularniejszą implementację algorytmu prostego podziału można znaleźć w oprogramowaniu Gambit [95]. W dalszej części rozprawy porównana zostanie z metodą zaproponowaną przez autora.

Jedną z najskuteczniejszych metod wyszukiwania równowag Nasha dla gier n-osobowych jest stosunkowo nowa globalna metoda Newtona (ang. *Global Newton Method*). Algorytm związany jest z metodą homotopii reprezentowanej jako system dynamiczny co pozwala na zastosowanie globalnej metody Newtona opisanej przez Smale'a [127]. W dużym uproszczeniu, działanie metody polega na przekształceniu układu równań w przypadek z jednym, prostym rozwiązaniem. Kolejnym krokiem jest odwrócenie całego procesu z jednoczesnym śledzeniem pośrednich rozwiązań w celu uzyskania rozwiązania oryginalnego układu. Przedstawione przez autorów rozwiązanie zastosowane zostało odpowiednio dla gier n-osobowych (2-6 graczy), przy czym maksymalna wielkość gry 3-osobowej, to 16 strategii, gry 4-osobowej 10 strategii, a 6-osobowej tylko 4 strategie. Podejście to daje wyniki lepsze, niż wspomniane wcześniej w rozdziale w pracy [70]. Ponadto, autorzy wspominają także o innym skutecznym algorytmie [57], który opisywany jest jako użyteczny algorytm startowy dla globalnej metody Newtona.

Tabela 5.2: Tabela przedstawiająca najpopularniejsze algorytmy i ich modyfikacje dla problemu wyszukiwania równowag w grach n-osobowych.

Algorytmy dla gier o sumie zero
1967 r. Algorytm Scarfa o złożoności wykładniczej [122]
1971 r. Uogólnienie algorytmu Lemke-Howsona dla gier n-osobowych [143]
1975 r. Ogólna idea wyszukiwania opartego na homotopii [66]
1979 r. Rozwinięcie algorytmu Scarfa [138]
1979 r. Wyszukiwanie z restartami [138]
1987 r. **Algorytm prostego podziału oparty na algorytmie Scarfa** [137]
1993 r. Logarytmiczna procedura śledząca [123]
2001 r. Podejście oparte na homotopii [70]
2003 r. **Globalna metoda Newtona** [58]
2003 r. Algorytm będący dopełnieniem globalnej metody Newtona [57]
2005 r. Algorytm do wyszukiwania rozwiązań w strategiach czystych oparty na przeszukiwaniu tabu [131]
2009 r. Przybliżone rozwiązanie w strategiach czystych [105]

W tabeli 5.2 wymieniono najpopularniejsze algorytmy i ich modyfikacje dotyczące gier n-osobowych (gdzie liczba graczy jest większa od 2). Warto zwrócić uwagę, iż tylko dwie wymienione metody dotyczą rozwiązań przybliżonych. Niestety możliwość ich praktycznego zastosowania jest znikoma, ponieważ dotyczą tylko rozwiązań w strategiach czystych (11 oraz 12 punkty w tabeli). Istotny z punktu widzenia dalszych rozważań jest też algorytm Scarfa. Metoda ta była punktem wyjścia dla wielu algorytmów dla gier n-osobowych, w tym algorytmu prostego podziału opisanego powyżej. Dwie wytłuszczone metody to algorytmy, z którymi porównane zostanie proponowane w rozprawie podejście.

Powyżej przedstawione zostały dwa najpopularniejsze algorytmy stosowane w grach wieloosobowych w postaci normalnej. Jak wskazują badania, najskuteczniejszy i jednocześnie najbardziej uniwersalny algorytm dla gier n-osobowych, to opisana wcześniej globalna metoda Newtona (GNM). Jednocześnie warto zauważyć, że w przypadku gier n-osobowych, jednym z najczęściej modyfikowanych algorytmów są metody oparte na algorytmie Scarfa. Logiczną konsekwencją rozważań będzie porównanie rozwiązania zaproponowanego w rozprawie z dwoma powyższymi algorytmami, a także z podstawowym algorytmem ewolucji różnicowej, który stanowił punkt wyjścia rozważań zawartych w niniejszej rozprawie. W celu porównania wyników wybrano implementacje algorytmów GNM oraz prostego podziału dostępne w programie GAMBIT [95].

ROZDZIAŁ 6

Adaptacyjny algorytm ewolucji różnicowej

Przystosowanie algorytmu ewolucji różnicowej do rozwiązywania jednego z najistotniejszych problemów teorii gier stanowiło ważny aspekt niniejszej rozprawy. Dotychczas w literaturze nie istniał skuteczny sposób opisania genotypu osobnika, który umożliwiałby wyszukiwanie równowag Nasha w grach n-osobowych, a jednocześnie był na tyle elastyczny, aby przy niewielkich modyfikacjach samej tylko funkcji oceny stanowić skuteczne narzędzie także w problemach takich jak wyszukiwanie równowag pareto optymalnych czy też równowag skorelowanych.

Zdecydowana większość istniejących w literaturze algorytmów jest bardzo skuteczna tylko dla konkretnych typów gier, a już niewielka modyfikacja problemu całkowicie uniemożliwia znalezienie jakiegokolwiek rozwiązania. Stąd konieczność opracowania oraz przetestowania pewnej ogólnej metody stanowiącej pomost pomiędzy metaheurystykami a algorytmami klasycznymi.

W niniejszym rozdziale przedstawiona zostanie szczegółowa analiza proponowanego adaptacyjnego algorytmu ewolucji różnicowej dla problemu wyszukiwania równowag Nasha w grach n-osobowych. Na początku przedstawiony zostanie pseudokod algorytmu oraz sposób kodowania genotypu odpowiedni dla problemu. W dalszej kolejności opisany zostanie sposób próbkowania przestrzeni rozwiązań, a także omówiona zostanie koncepcja stanowiąca podstawę powyższego podejścia. Bardzo istotnym elementem jest podejście do adaptacji parametrów w proponowanym algorytmie. Oprócz klasycznych parametrów, takich jak współczynnik krzyżowania i mutacji, opisany zostanie sposób modyfikacji wielkości populacji oraz ustalania wielkości populacji początkowej. Funkcja oceny zaproponowana dla problemu wyszukiwania równowag Nasha omówiona zostanie pod koniec rozdziału, by zakończyć szczegółowym opisem zaproponowanego operatora mutacji określanego jako selektywna mutacja, a także λ modyfikacji, która pozwala istotnie poprawić zbieżność populacji do optimum w końcowym etapie działania algorytmu.

6.1 Ogólna charakterystyka algorytmu

Rozwiązanie przedstawione w niniejszej rozprawie opiera się na przekształceniu zbioru strategii mieszanych poszczególnych graczy w genotyp osobnika. W grze o sumie niezerowej każdy z graczy dysponuje własną macierzą wypłat, której wielkość uzależniona jest od ich liczby, a także od liczby strategii przypisanych dla każdego gracza. Traktując strategię mieszaną gracza (czyli innymi słowy rozkład prawdopodobieństwa nad zbiorem strategii czystych gracza) jako część genotypu możliwe jest istotne ograniczenie jego długości nawet dla bardzo złożonych problemów. W tabeli 6.1 przedstawiona została liczba elementów macierzy, które analizowane są w klasycznych algorytmach, a także długość genotypu dla odpowiadających im problemów. Szczególnie istotny jest fakt, iż wielkość macierzy wypłat obliczana jest na podstawie wzoru $m^n \cdot n$, gdzie n jest liczbą graczy, a m liczbą strategii gracza (ostatni wiersz tabeli). Z kolei wielkość genotypu jest iloczynem liczby strategii oraz liczby graczy. W rozprawie przyjęto dość powszechne założenie, iż liczba strategii każdego z graczy jest równa.

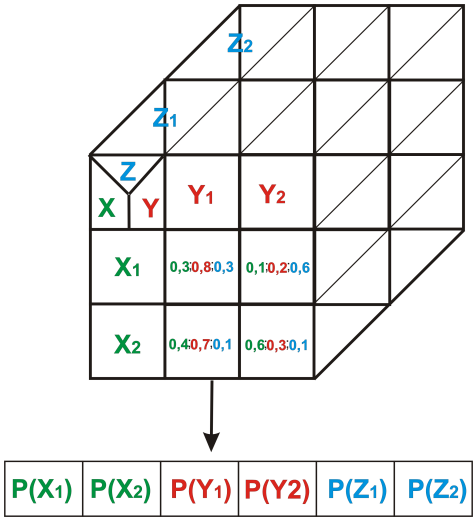
Tabela 6.1: Zależność pomiędzy długością genotypu a złożonością problemu

Gracze	**Strategie**	**Ewolucja różnicowa**	**Klasyczne podejścia**
2	2	$2 \cdot 2 = 4$	$2^2 \cdot 2 = 8$
2	10	$2 \cdot 10 = 20$	$10^2 \cdot 2 = 200$
3	3	$3 \cdot 3 = 9$	$3^3 \cdot 3 = 243$
3	10	$3 \cdot 10 = 30$	$10^3 \cdot 3 = 3000$
5	5	$5 \cdot 5 = 25$	$5^5 \cdot 5 = 15625$
5	10	$5 \cdot 10 = 50$	$10^5 \cdot 5 = 500000$
n	m	$n \cdot m$	$m^n \cdot n$

Warto zaznaczyć, iż w tabeli 6.1 przedstawione zostały długości genotypu osobnika przy założeniu, iż wszystkie strategie każdego z graczy są aktywne i posiadają niezerowe prawdopodobieństwo wyboru. Proponowany w rozprawie algorytm pozwala na ograniczenie liczby strategii aktywnych, dlatego nawet w przypadku złożonych gier, genotyp może zawierać tylko kilkanaście genów. Sam proces tworzenia genotypu osobnika przedstawiony został na rys. 6.1. Pierwszą część genotypu stanowią prawdopodobieństwa wyboru strategii czystych pierwszego gracza, natomiast kolejne części genotypu to strategie pozostałych graczy. Każdy gen w genotypie osobnika przyjmuje wartość z przedziału $\langle 0, 1)$, gdzie wartość zero oznacza zerowe prawdopodobieństwo wyboru danej strategii czystej gracza, z kolei wartość 1 to pewny wybór wskazanej strategii czystej.

W rozdziale dotyczącym równowag Nasha wspomniano, iż problem znalezie-

nia równowagi czystej (gdzie gracze stosują wybrane strategie z prawdopodobieństwem równym 1) jest znacznie prostszy. Istnieją też algorytmy ukierunkowane tylko i wyłącznie na wyszukiwanie równowag czystych. W rozprawie głównym problemem jest wyszukiwanie równowag w strategiach mieszanych - stąd rozwiązania, w których jedna ze strategii gracza wybierana z prawdopodobieństwem równym 1 są pomijane.



Rysunek 6.1: Tworzenie genotypu osobnika

Pseudokod proponowanego podejścia przedstawiony został w alg. 5. Składa się on z dwóch głównych części. Pierwsza faza (wiersze 2 i 3) pozwala na wstępne oszacowanie jakości rozwiązania dla wybranych strategii graczy. Jeżeli wartość ϵ przekroczy ustalony próg, to taki profil strategii rozpatrywany jest jako potencjalna ϵ-równowaga Nasha. Procedura ta opisana zostanie szczegółowo w dalszej części rozprawy. Wiersze 5 - 13 to właściwa część algorytmu adaptacyjnej ewolucji różnicowej, która wykonywana jest tylko wtedy, kiedy dany podzbiór strategii czystych zostanie zaakceptowany jako potencjalne rozwiązanie. W wierszu 14 następuje aktualizacja liczby osobników w populacji. Ponieważ w proponowanym algorytmie rozpatrywane są niewielkie zbiory strategii (do 5 strategii z niezerowym prawdopodobieństwem wyboru dla każdego z graczy), pierwsza faza algorytmu pozwala od razu odrzucić słabe rozwiązania jako nieakceptowalne. Złożoność proponowanego algorytmu jest rzędu:

$$O(c \cdot G \cdot D^2), \tag{6.1}$$

gdzie:

G - liczba iteracji algorytmu ewolucji różnicowej,

D - wymiar problemu,

c - liczba rozpatrywanych układów strategii ($c \ll G$)

Algorytm 5: Adaptacyjny algorytm ewolucji różnicowej

```
begin
1   for Liczba układów strategii do
2       Próbkuj aktualny układ a;
3       if Wynik próbkowania < ε then
4           Utwórz populację początkową S o liczbie osobników NP;
5           t = 0;
6           for Liczba iteracji ADE do
7               for Wielkość populacji do
8                   Mutacja i krzyżowanie DE;
9                   if Przystosowanie V_i ⩾ S_i then
10                      S_i^{t+1} = V_i^t;
                    else
11                      S_i^{t+1} = S_i^t;
12                  Aktualizuj CR, F i t = t + 1 ;
13              if gen mod gen/10 = 0 then
                    Zmniejsz NP
14      Wybierz nowy układ strategii a;
15  Rozwiązanie = zbiór ε-równowag Nasha;
```

Do wyznaczenia oceny osobnika wymagane są macierze wypłat dla poszczególnych graczy. Jednym z najistotniejszych założeń dotyczących proponowanego algorytmu jest stosowanie tylko wybranych strategii czystych poszczególnych graczy, co pozwala na pominięcie dużej części komórek macierzy wypłat przy jednoczesnym zachowaniu możliwości wyznaczenia optimum globalnego.

Proponowane w rozprawie algorytm jest w pewnym stopniu losowy, dlatego możliwe jest tylko pewne oszacowanie średniej wartości rozwiązania, które najczęściej zawiera się w przedziale $\epsilon \in \langle 0.2 : 0.3 \rangle$ (gdzie $\epsilon = 0$ to optimum globalne). Dla porównania, najlepsze algorytmy przybliżone dla gier dwuosobowych pozwalają uzyskać rozwiązanie $\epsilon \approx 0.33$.

6.2 Metoda próbkowania przestrzeni rozwiązań

W sytuacjach, kiedy liczba strategii poszczególnych graczy jest duża, zdecydowana większość strategii graczy ma zerowe (lub bliskie zeru) prawdopodobieństwo wyboru. Oznacza to, iż liczba strategii czystych mających istotny wpływ na jakość rozwiązania jest niewielka i najczęściej stanowi tylko część zbioru wszystkich

strategii. Założenie to zostało potwierdzone w artykule [91]. Oczywiście wybranie wszystkich strategii graczy jako genotypu osobnika również jest możliwe i daje satysfakcjonujące rozwiązanie.

Warto zauważyć, iż w artykule [119] zasugerowano, że z punktu widzenia gracza ograniczona liczba strategii aktywnych jest znacznie lepszym rozwiązaniem. Wybór spośród niewielkiej liczby strategii jest znacznie łatwiejszy, niż w przypadku kiedy dostępny zbiór obejmuje wszystkie strategie. W tej sytuacji logiczne jest uzasadnienie wprowadzenia pewnej metody pozwalającej na losowy wybór niewielkiego podzbioru strategii dla każdego z graczy. Podejście takie określone zostało przez autora rozprawy jako próbkowanie rozwiązań. Próbkowanie rozwiązań przedstawione zostało w algorytmie 6.

Algorytm 6: Próbkowanie przestrzeni rozwiązań

```
   begin
1      for Liczba układów strategii do
2          Ustaw strategie aktywne;
3          for Liczba iteracji DE do
4              for Wielkość populacji do
5                  Mutacja i krzyżowanie DE;
6                  if Przystosowanie V_i ⩾ S_i then
7                      S_i^{t+1} = V_i^t;
                   else
8                      S_i^{t+1} = S_i^t;
9          if Rozwiązanie < wartość ϵ then
10             Algorytm ADE
           else
11             Odrzuć rozwiązanie i wybierz kolejny układ strategii aktywnych;
12             Powrót do kroku 2;
```

W rozprawie zastosowano próbkowanie przestrzeni rozwiązań oparte na ewolucji różnicowej. Zbieżność algorytmu jest najszybsza w kilkuset pierwszych iteracjach, dlatego z dużym prawdopodobieństwem można stwierdzić, iż rozwiązanie nieakceptowalne w pierwszych g iteracjach, gdzie $g \ll Max_{iteracje}$ może zostać odrzucone. Każdy układ strategii, w którym wartość ϵ po g iteracjach jest akceptowalna (wartość $\epsilon < 0.4$), rozwiązywany jest przez adaptacyjny algorytm ewolucji różnicowej. Wartość 0.4 ustalona została na podstawie wartości pesymistycznej uzyskiwanej dla gier dwuosobowych. Najlepsze algorytmy przybliżone gwarantują wartość ϵ nie gorszą niż 0.33 - 0.35. Założono, iż w przypadku gier n-osobowych wartość ta powinna być większa i ustalona została na takim poziomie.

W kolejnym rozdziale przeprowadzone zostaną szczegółowe badania dotyczące

liczby strategii aktywnych (o niezerowym prawdopodobieństwie wyboru) w algorytmach klasycznych. Wyniki te następnie zestawione zostaną z proponowanym algorytmem. Warto zaznaczyć, iż istniejące algorytmy klasyczne generują rozwiązania dla niewielkich zbiorów strategii aktywnych, dlatego ograniczenie tego zbioru w proponowanej metodzie również jest bardzo pożądane.

6.3 Szczegółowy opis parametrów w proponowanym algorytmie

Jednym z najistotniejszych elementów proponowanego algorytmu jest jego adaptacyjny charakter. Ewolucja różnicowa (jak i inne algorytmy metaheurystyczne) wymagają skomplikowanego procesu strojenia licznych parametrów. Oprócz współczynnika krzyżowania i mutacji konieczne jest także określenie wielkości populacji. W trakcie działania algorytmów wartość ta pozostaje niezmienna. Jednocześnie zbieżność algorytmów przybliżonych tego typu została wykazana w licznych publikacjach na ten temat, stąd oczywisty wniosek, iż w kolejnych iteracjach algorytmu różnorodność populacji maleje wraz ze zbliżaniem się osobników do optimum. Prowadzi to do sytuacji powielania genotypu osobników o dobrym przystosowaniu. Stąd wniosek, iż ograniczenie wielkości populacji bez jednoczesnej utraty informacji jest możliwe, a mechanizmy ograniczenia liczby osobników w algorytmie były już wielokrotnie przedstawiane. Szczegóły dotyczące tych modyfikacji przedstawione były w rozdziale drugim niniejszej rozprawy. Zaproponowana poniżej metoda opiera się na dwóch elementach. Po pierwsze, początkowa wielkość populacji powinna być ściśle związana z rozpatrywanym problemem. Złożoność problemu wyszukiwania równowag Nasha i problemów pokrewnych ściśle związana jest z liczbą graczy oraz liczbą strategii poszczególnych graczy. Z punktu widzenia proponowanej metody, 4-osobowa gra z 3 strategiami dostępnymi dla danego gracza jest znacznie mniej złożona niż 3-osobowa gra z 10 strategiami.

Wielkość populacji w proponowanym algorytmie nie jest wartością stałą i ulega zmianom w pewnych odstępach czasu. Autor rozprawy założył, iż zmiany tej wartości nie powinny zachodzić w każdej iteracji. Po pierwsze, wiąże się to z dodatkowym nakładem obliczeniowym, który byłby wykonywany w każdej iteracji. Ponadto, istotne zmiany dotyczące wartości fenotypu osobników są zachodzą tylko co określoną liczbę iteracji. Stąd wniosek, iż zmiana liczby osobników w populacji powinna następować tylko w niektórych iteracjach. Schemat zmiany wielkości populacji zaproponowany w rozprawie przedstawiony został w algorytmie 7.

Pojęcie entropii występujące w algorytmie dotyczy entropii rozkładu prawdopodobieństwa i oznaczone jest jako $H(\cdot)$. Dotyczy ono różnorodności wartości fenotypu poszczególnych osobników. Dane jest ono wzorem:

$$H(g) = ln(\sigma \cdot \sqrt{2 \cdot \Pi \cdot e}), \tag{6.2}$$

Algorytm 7: Modyfikacja wielkości populacji

```
begin
1   poprzednia = H(0);
2   for Liczba iteracji DE do
        ;
3   if warunek then
4       aktualna = H(g);
5       if aktualna < poprzednia then
            NP = NP· 0.95;
        else
            NP = NP· 1.05;
        poprzednia = aktualna;
```

gdzie:

$H(g)$ - wartość entropii obliczana w iteracji g,

σ - odchylenie standardowe dla zbioru wartości fenotypu poszczególnych osobników w populacji.

Początkowe położenie osobników w przestrzeni poszukiwań ustalane jest na podstawie rozkładu jednostajnego. W trakcie działania algorytmu można oczekiwać, iż wartości fenotypu osobników zaczną przypominać rozkład normalny, gdzie pewna liczba elementów będzie cechowała się bardzo dobrym przystosowaniem. Jednocześnie dla części osobników wartość fenotypu będzie niewielka. Pozostałe osobniki w populacji będą gdzieś pomiędzy dwoma powyższymi elementami. Na tej podstawie można przypuszczać, iż zastosowanie entropii związanej z rozkładem normalnym jest tutaj odpowiednie. Powyższe podejście zapewnia zmienność wielkości populacji na podstawie aktualnych wartości funkcji oceny. Dodatkowo, liczebność populacji ograniczona jest dwoma stałymi NP_{max} oraz NP_{min}, które wynoszą odpowiednio NP oraz $\frac{NP}{3}$.

Kolejnym parametrem zmodyfikowanym w algorytmie jest operator mutacji. Jest on znacznie istotniejszy niż operator krzyżowania i to właśnie mutacja przede wszystkim kontroluje szybkość zbieżności populacji. Niestety w wielu publikacjach jej sugerowane wartości wahają się od 0.1 do nawet 0.9. W artykule [87] S. Kukkonen oraz J. Lampinen zwracają uwagę iż niewielkie wartości F prowadzą do szybkiej zbieżności zwiększając jednocześnie ryzyko zatrzymania się populacji w optimum lokalnym. W większości modyfikacji algorytmu parametr F jest wartością ustaloną dla wszystkich osobników. Sytuacja ta ma miejsce nawet dla modyfikacji dotyczących dynamicznej zmiany parametrów.

Zastosowane w rozprawie podejście bazuje na założeniu, iż wartość parametru F powinna być modyfikowana w każdej iteracji algorytmu i jest ściśle związana z parametrem krzyżowania. Wymaga to dodatkowego nakładu obliczeniowego w celu wyznaczenia wartości F. Procedura wyznaczania osobników próbnych przed-

stawiona została w alg. 8.

Algorytm 8: Algorytm mutacji w ADE

begin

1 Uszereguj osobniki w populacji malejąco ze względu na fenotyp;

2 **for** *dla każdego osobnika w populacji* **do**

3 Wybierz osobnika z p najlepszych elementów w populacji;

4 Wybierz losowo dodatkowego osobnika z populacji;

5 **for** *wielkość genotypu osobnika* **do**

$\forall_j \; U_{i,j} = S_{i,j} + \lambda \cdot F_i \cdot (S_{r_p,j} - S_{r,j})$

gdzie:

j - j-ty gen bieżącego osobnika,

p - 20% najlepszych osobników w populacji,

$\lambda \in \langle 0, 1 \rangle$ - czynnik skalujący,

r_p - losowy osobnik z 20% najlepszych osobników w populacji,

F_i - wartość parametru dla i-tego osobnika.

Powyższe podejście związane z wyznaczeniem 20% najlepszych osobników oparte jest na metodzie mutacji stosowanej w algorytmie JADE (opis algorytmu przedstawiony został w rozdziale 3).

Na wartość F istotny wpływ ma wartość μ_F obliczana w każdej iteracji algorytmu. Obliczana jest zgodnie ze wzorem:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot L \; , \tag{6.3}$$

gdzie: c - parametr proporcji ustalony na 0.5.

Nowa wartość μ_F jest średnią wartości μ_F z poprzedniej iteracji oraz wartości L - średniej Lehmera wyznaczanej zgodnie ze wzorem:

$$L = \frac{\sum_{F \in set_F} F^2}{\sum_{F \in set_F} F}, \tag{6.4}$$

gdzie: set_F - zbiór wszystkich wartości F dla populacji.

Powyższe podejście do wyznaczania wartości μ_F przedstawione zostało w [120].

Podejście zbliżone do modyfikacji współczynnika mutacji zastosowane zostało także w przypadku krzyżowania. W algorytmie wprowadzony został wektor wartości krzyżowania, gdzie jednemu osobnikowi odpowiada dokładnie jedna wartość CR z przedziału $\langle 0, 1 \rangle$. Wartości współczynnika krzyżowania ustalane są według wzoru:

$$\forall_i \, CR_i = \mu_{CR} + 0.1 \cdot Normal(-1, 1), \tag{6.5}$$

gdzie:

i - i-ty osobnik,

$Normal(-1, 1)$ - wartość z przedziału generowana zgodnie z rozkładem normalnym.

Natomiast wartość μ_{CR} obliczana jest następująco:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot Avg, \tag{6.6}$$

gdzie:

c, μ_F - parametry związane z proporcją,

Avg - średnia z wartości CR z poprzedniej iteracji.

Wartość parametru CR w pierwszej iteracji ustalana jest na 0.5. Pozostałe elementy algorytmu: akceptowalna wartość ϵ po próbkowaniu rozwiązania, liczba rozpatrywanych układów strategii oraz liczba strategii aktywnych nie mają bezpośredniego wpływu na działanie algorytmu. Liczba rozpatrywanych układów związana jest z oczekiwaną liczbą rozwiązań uzyskanych po zakończeniu działania algorytmu. Możliwe jest tutaj wybranie wartości równej 1. W takiej sytuacji algorytm wywołany zostanie tylko raz dla jednego układu. Z kolei liczba strategii aktywnych związana jest bezpośrednio z preferencjami użytkownika dotyczącymi rozwiązania. Duża wartość tego parametru zwiększa szansę na znalezienie rozwiązania optymalnego, należy jednak pamiętać iż niewielka liczba strategii aktywnych jest zdecydowanie bardziej pożądana z punktu widzenia danego gracza.

Wreszcie ostatni element, czyli wartość ϵ akceptowalna po próbkowaniu rozwiązań. W algorytmie istnieje możliwość manualnego doboru tej wartości jednak sugerowane jest pozostawienie wartości domyślnej równej $\epsilon = 0.4$. W kolejnym rozdziale przedstawione zostaną wyniki eksperymentów dotyczących zależności pomiędzy powyższą wartością ϵ a liczbą znalezionych rozwiązań.

6.4 Funkcja przystosowania w problemie wyszukiwania równowag Nasha

Funkcja oceny jest najistotniejszym elementem każdego algorytmu heurystycznego. Prawidłowe jej zaprojektowanie pozwala na uzyskanie zadowalającego rozwiązania problemu przy jednoczesnym ograniczeniu czasu działania samej metody. Problem wyszukiwania równowag Nasha został sprowadzony do zadania minimalizacji funkcji wielowymiarowej, gdzie wymiar problemu jest proporcjonalny do iloczynu liczby graczy oraz liczby strategii aktywnych danego gracza. Sama funkcja oceny składa się z trzech części, które następnie są sumowane. Pierwsza część dotyczy maksymalnego odchylenia od strategii optymalnej poszczególnych graczy i może zostać opisana następująco:

$$f_1 = \max\{u_1(a) - \max\{u_1(a_{11}, a_{-1}), ..., u_1(a_{1j}, a_{-1})\}, \\ ..., u_i(a) - \max\{u_i(a_{i1}, a_{-i}), ..., u_i(a_{ij}, a_{-i})\}\}, \tag{6.7}$$

gdzie:

$u_i(a_{ij}, a_{-i})$ - wypłata i-tego gracza stosującego strategię czystą j,

$u_i(a)$ - wypłata i-tego gracza dla profilu strategii a.

Innymi słowy chodzi o wyznaczenie wypłaty gracza przy założeniu, że stosuje on wybraną strategię czystą. Z powyższego zbioru wypłat wyznaczana jest wartość maksymalna. Następnie, na podstawie bieżącego genotypu osobnika obliczana jest strategia mieszana gracza. Zgodnie z definicją równowagi Nasha, gracz stosujący strategię mieszaną nie może zyskać więcej, niż gdyby stosował jedną ze swoich strategii czystych. Oznacza to, iż różnica pomiędzy wypłatą gracza stosującego strategię mieszaną, a wypłatą gracza stosującego strategię czystą dającą mu maksymalną wypłatę powinna wynosić zero. Warunek ten jest określony dla wszystkich graczy. Tylko w sytuacji kiedy dla każdego z graczy opisana powyżej różnica jest równa zero, rozwiązanie określane jest jako dokładna równowaga Nasha. W praktyce uzyskanie rozwiązania optymalnego globalnie dla dużych gier jest niezwykle trudne, dlatego ta część funkcji oceny stosowana jest również po zakończeniu działania algorytmu do wyznaczenia wartości ϵ w równowadze przybliżonej.

Warto przytoczyć tutaj inny sposób wyznaczania wartości f_1 w funkcji oceny. W sytuacji, kiedy liczba strategii aktywnych danego gracza jest równa liczbie wszystkich jego strategii spełniony jest warunek:

$$\forall_j \; u_i(a) = u_i(a_{i1}, a_{-i}) = ... = u_i(a_{ij}, a_{-i}), \tag{6.8}$$

gdzie:

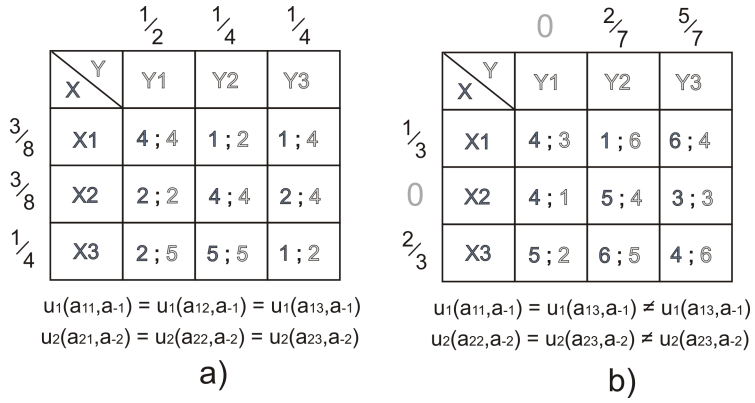
i - indeks gracza,

j - j-ta strategia dostępna dla gracza.

Powyższy wzór oznacza, iż wypłata gracza stosującego dowolną strategię czystą ma zawszę tę samą wartość. Zależność ta przedstawiona jest na rys. 6.2. W celu zachowania spójności, we wszystkich eksperymentach dotyczących wyszukiwania równowag Nasha zastosowana została pierwsza wersja funkcji f_1. Wyniki oparte na drugim podejściu można znaleźć w [15].

Dodatkowym warunkiem koniecznym jest sprawdzenie, czy suma prawdopodobieństw wyboru poszczególnych strategii wszystkich graczy wynosi 1. Dotyczy on tylko strategii aktywnych poszczególnych graczy. Powyższy warunek opisany jest wzorem:

$$f_2 = \sum_{i=1}^{n} |1 - \sum_{j=1}^{m} P(a_{ij})|, \tag{6.9}$$



Rysunek 6.2: Liczba strategii aktywnych a wypłaty graczy. a) wypłaty graczy są równe dla wszystkich strategii aktywnych. b) wypłata gracza stosującego strategię czystą, która nie jest aktywna w strategii mieszanej jest różna od pozostałych

gdzie: $P(a_{ij})$ - prawdopodobieństwo wyboru j-tej strategii i-tego gracza.
Założenie to jest znacznie prostsze do spełnienia, a jednocześnie jest znacznie istotniejsze. W sytuacji, gdy prawdopodobieństwa wyboru strategii chociaż jednego z graczy po sumowaniu nie są równe 1, nie można mówić o rozkładzie prawdopodobieństwa nad zbiorem strategii czystych i jednocześnie o strategii mieszanej.

Trzecim elementem funkcji przystosowania jest założenie dotyczące pomijania strategii czystych. Jednym z głównych założeń niniejszej rozprawy było przedstawienie algorytmu pozwalającego na wyszukiwanie równowag mieszanych. Rozwiązanie w strategiach czystych nie jest w tym wypadku akceptowalne i powinno być automatycznie odrzucone. W tym celu wprowadzono funkcję f_3, która zwiększa wartość funkcji oceny o stałą wartość w sytuacji, kiedy jeden z genów w genotypie osobnika ma wartość równą 1:

$$\forall_i, f_3 = \begin{cases} f_3 + c & \text{kiedy } g[i] = 1, \\ f_3 & \text{w przeciwnym razie,} \end{cases}$$

Parametr c w powyższym wzorze to wartość odpowiadająca kwadratowi iloczynów liczby strategii oraz liczby graczy. Suma trzech powyższych funkcji stanowi funkcję przystosowania:

$$f = f_1 + c \cdot f_2 + f_3, \tag{6.10}$$

Wartość c w powyższym wzorze wpływa na zwiększenie ważności dla funkcji f_2, ponieważ suma prawdopodobieństw równa 1 jest warunkiem koniecznym. Wysoka wartość początkowa parametru c gwarantuje, iż nawet minimalne odchylenie od wartości 1 skutkuje znaczącym wzrostem wartości f i prowadzi do odrzucenia takiego rozwiązania. Warto nadmienić, iż funkcja f_1 dotyczy parametru ϵ i jest niewielka w porównaniu do $c \cdot f_2$, dlatego rozwiązania nawet z dużą wartością ϵ

będą faworyzowane.

6.5 Operator selektywnej mutacji

Przedstawiona w poprzednim podrozdziale funkcja oceny ma interesującą właściwość. Po spełnieniu założenia o sumie prawdopodobieństw dla danego gracza równej 1, minimalizacja wartości f_1 sprowadza się do poprawy wartości wypłaty jednego z graczy (f_1 stanowi maksymalne odchylenie od optimum najgorszego z graczy, czyli gracza o najniższej wypłacie). Prowadzi to do sytuacji, w której w celu poprawy jakości rozwiązania konieczna jest modyfikacja tylko części genotypu odpowiedzialnej za strategię mieszaną gracza o najniższej wypłacie. Proponowany w rozprawie schemat może zostać przedstawiony następująco:

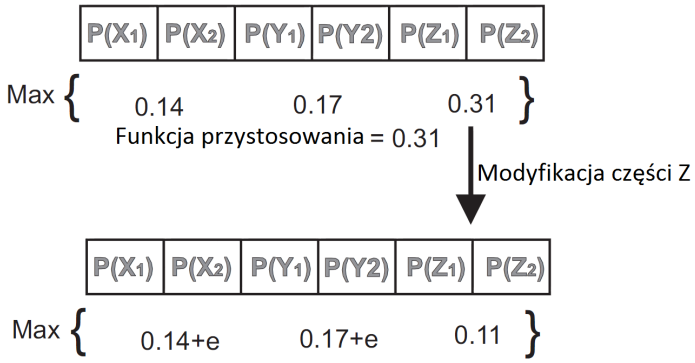
$$S_{i,j} = \begin{cases} Mutuj & \text{gdy } S_{i,j} \in \breve{S}_i, \\ S_{i,j} & \text{w przeciwnym razie.} \end{cases}$$

gdzie: $\breve{S}_i$ - fragment genotypu odpowiadający strategiom gracza posiadającego najmniejszą liczbę jednostek użyteczności.
Zaproponowane powyżej podejście ma solidne podstawy w algorytmach opartych na wyszukiwaniu najlepszej odpowiedzi (strategia dająca graczowi najwyższą wypłatę, biorąc pod uwagę strategie wybierane przez innych graczy). Stosując tę definicję do opisania równowagi Nasha staje się jasne, iż równowaga Nasha jest sytuacją, kiedy każdy z graczy jednocześnie stosuje strategię najlepszej odpowiedzi.

Podejście takie pozwala istotnie ograniczyć złożoność operatora mutacji. Istnieje także możliwość stosowania operatorów mutacji wymiennie - w początkowej fazie działania algorytmu, kiedy nowe rozwiązania o niższej wartości funkcji oceny pojawiają się często, klasyczny schemat wprowadzający więcej zmian w genotypie danego osobniku działa skuteczniej. Z drugiej jednak strony selektywna mutacja zwiększa szansę na znalezienie większej liczby rozwiązań optymalnych lokalnie. Zasada działania operatora przedstawiona została na rys. 6.3.

Przy każdej modyfikacji strategii gracza, w pewnym stopniu zmieniają się wypłaty wszystkich graczy. Wynika to z faktu, iż dla każdego gracza jego wypłata obliczana jest na podstawie własnej macierzy wypłat. Niewielka liczba strategii aktywnych danego gracza znacznie przyspiesza zbieżność algorytmu do optimum, stąd powyższe ograniczenie w połączeniu z selektywną mutacją pozwala na uzyskanie satysfakcjonujących wyników.

Przedstawione w powyższym rozdziale szczegóły dotyczące adaptacyjnego algorytmu ewolucji różnicowej wskazują, iż takie podejście sprowadza problem wyszukiwania równowag Nasha w grach n-osobowych do zadania minimalizacji funkcji wielowymiarowej. Zagadnienie to jest jasno sprecyzowane i pozwala jednoznacznie oceniać jakość uzyskiwanych wyników. Dodatkowo, opisana tutaj funk-



Rysunek 6.3: Operator selektywnej mutacji

cja oceny jest bardzo ogólna i nie dotyczy tylko i wyłącznie wybranego typu gier. Pozwala to przypuszczać, iż opracowany algorytm stanowi dobrą podstawę do dalszych rozważań w dziedzinie zastosowań algorytmów heurystycznych w teorii gier. Powyższe stwierdzenia potwierdzone zostaną w kolejnych rozdziałach poświęconych w całości przygotowaniu zbiorów testowych, przeprowadzeniu doświadczeń oraz opisaniu rezultatów badań.

ROZDZIAŁ 7

Badania eksperymentalne proponowanego algorytmu

W rozdziale tym opisane zostaną badania eksperymentalne dotyczące proponowanego w rozprawie algorytmu. Adaptacyjny algorytm ewolucji różnicowej (ADE) porównany zostanie z algorytmem prostego podziału (SM), a także globalną metodą Newtona (GNM). W rozprawie przedstawiono także implementację podstawowej wersji algorytmu ewolucji różnicowej (DE), która to stanowić będzie istotny element niniejszych badań eksperymentalnych. Jednym z podstawowych celów niniejszego rozdziału jest wykazanie, iż proponowane podejście pozwala uzyskać stabilne rozwiązania dla dowolnych typów gier. ADE umożliwia generowanie kilku rozwiązań unikalnych dla dowolnej gry, a czas działania algorytmu zależy wyłącznie od rozmiaru problemu. W przypadku istniejących algorytmów SM i GNM czas generowania rozwiązania nierzadko różni się nawet dla problemów o jednakowej liczbie graczy i strategii. Jak do tej pory nie sformułowano jasno cech, jakie posiada gra, dla której wyznaczenie równowagi Nasha przy pomocy algorytmów SM oraz GNM jest istotnie trudniejsze. Dlatego tak istotne jest wyznaczenie odchylenia standardowego w odniesieniu do czasu generowania rozwiązania. Wyraźnie widoczne jest to w przypadku, wydaje się, gier niewielkich, gdzie wyznaczenie równowagi Nasha nie powinno stanowić problemu. Stąd też w kwestii czasu generowania rozwiązania widoczna jest pozorna przewaga algorytmów GNM i SM nad algorytmami przybliżonymi. Wraz ze wzrostem wielkości gier, skuteczność wspomnianych algorytmów dokładnych wyraźnie spada. Istotnym celem przeprowadzanych badań jest także wykazanie, iż proponowane podejście pozwala na ograniczenie liczby strategii aktywnych dla danego gracza.

W dalszej części rozdziału opisane zostaną szczegółowo badania dotyczące zbieżności populacji, a także przeprowadzone zostaną testy statystyczne potwierdzające zasadność wniosków wykazanych w pierwszej części rozdziału.

7.1 Przygotowanie zbiorów testowych

W przypadku zagadnienia wyszukiwania równowag Nasha w grach n-osobowych opracowanie odpowiedniego zbioru testowego stanowiło istotny problem. W rozpatrywanej tematyce nie jest dostępny żaden ogólny zbiór testowy, który pozwala jednoznacznie ocenić jakość generowanych rozwiązań i jednocześnie porównać je z optimum globalnym. W tym celu zastosowane zostało narzędzie GAMUT pozwalające generować różne typy gier. Autor rozprawy założył przy tym, iż konieczne jest zróżnicowanie elementów zbioru testowego nie tylko ze względu na rozmiar, ale także na typ gier. Wybrano 3 typy gier, których złożoność wydaje się największa z punktu widzenia istniejących algorytmów.

- gry z kowariancją - gdzie jednym z parametrów jest wariancja pomiędzy strategiami graczy;
- gry losowe - domyślny typ gier, dla którego testowane są algorytmy;
- gry typu RoadMap.

Dla każdego z powyższych typów wygenerowano zbiory dla 3, 4 oraz 5 graczy. Narzędzie GAMUT umożliwia ustalenie wielu parametrów wejściowych dla generowanych gier. Przykładowe parametry, z jakimi wywoływany był program przedstawione zostały poniżej.

```
java␣-jar␣gamut.jar␣-g␣RandomGame␣-players␣2␣-random_params␣-normalize
-min_payoff␣0␣-max_payoff␣1␣-f␣Game.nfg␣-output␣GambitOutput␣-actions␣5
```

Rozpatrywane w badaniach algorytmy SM oraz GNM należą do metod deterministycznych, stąd też każdy przypadek testowy wygenerowany został 30 razy. Następnie wyniki zostały uśrednione. Do testów zastosowano implementacje algorytmów SM oraz GNM dostępne w narzędziu GAMBIT (język C++). Jest to oprogramowanie typu open source zawierające najpopularniejsze algorytmy stosowane zarówno w grach w postaci strategicznej jak i ekstensywnej. Implementacje algorytmu DE oraz ADE również przygotowane zostały w języku C++. Wszystkie badania przeprowadzone zostały na komputerze z procesorem Intel Core 2 Duo 2.13 GHz, 3 MB pamięci cache 2 poziomu.

Parametry dla algorytmu DE ustalone zostały następująco:

- liczba osobników w populacji równa iloczynowi liczby graczy oraz liczby strategii dla jednego gracza;
- schemat mutacji z trzema osobnikami wybranymi losowo z populacji oraz z parametrem mutacji $F = 0.7$ (jest to modyfikacja podstawowego schematu DE, w którym losowo wybierane są tylko dwa osobniki);

- dwumianowy schemat krzyżowania z współczynnikiem krzyżowania równym $CR = 0.5$;
- dla gier 3 oraz 4 osobowych liczba iteracji algorytmu wynosi $l.strategii \cdot 100$, natomiast w przypadku gier 5 osobowych liczba iteracji wynosi 1000.

Należy pamiętać, iż długość genotypu osobnika stanowiła zawsze iloczyn liczby graczy oraz liczby strategii dla danego problemu. Liczba strategii aktywnych dla gracza w początkowej fazie algorytmu odpowiadała liczbie strategii. Zastosowana selekcja gwarantuje, iż do kolejnej iteracji przeniesione zostają osobniki o wyższej wartości funkcji oceny. Rozwiązaniem jest najlepszy osobnik w ostatniej iteracji algorytmu. W przypadku algorytmu ADE zastosowano szereg modyfikacji poprawiających jakość uzyskanych rozwiązań. Jednymi z najistotniejszych zmian jest zastosowany schemat selektywnej mutacji oraz dynamiczna zmiana wielkości populacji. Dwa parametry, istotnie wpływające na jakość rozwiązań, których wartość może być zmieniana przez użytkownika, to:

- liczba strategii aktywnych dla jednego gracza. Przebadano wartości 2, 3 oraz 4.
- maksymalna wartość ϵ akceptowalna jako potencjalne rozwiązanie przy próbkowaniu rozwiązań ustalona na poziomie 0.4.

Warto przypomnieć, iż w artykule [91] wskazano na istnienie równowag Nasha, gdzie liczba strategii aktywnych wynosi $\ln n$, gdzie n jest liczbą strategii danego gracza. Stąd oczywisty wniosek, aby założenie to rozszerzyć także na inne typy gier. Z kolei wartość ϵ powinna być traktowana jako wartość pesymistyczna. Dla gier 2-osobowych wartość ta dla najlepszych algorytmów przybliżonych wynosi w przybliżeniu 0.33. Dla gier n-osobowych autor rozprawy przyjął większy margines błędu dla wartości ϵ, który wynosił 0.4. W tabelach oraz na wykresach stosowano następujące oznaczenie wielkości gier: $xpys$, gdzie x jest liczbą graczy, natomiast y liczbą strategii. Przykładowo $4p9s$ odnosi się do gry 4-osobowej z 9 strategiami dla każdego z graczy.

7.2 Porównanie algorytmów pod kątem czasu generowania rozwiązania

Pierwszy etap badań dotyczył wskazania istotnych problemów występujących w algorytmach SM oraz GNM, a także zaznaczenia, iż proponowane algorytmy ze względu na czas generowania rozwiązania stanowić mogą interesującą alternatywę dla istniejących algorytmów. W algorytmie ADE zastosowano podejście z 3 strategiami aktywnymi. W tabeli 7.1 przedstawiono minimalne, maksymalne, średnie, mediany, a także odchylenia standardowe dla generowania jednego rozwiązania

dla 4 omawianych algorytmów. Tabela podzielona została na 2 części tak, że w pierwszej znajdują się czasy dla istniejących metod, natomiast druga część to proponowane w rozprawie podejścia. Przebadano czas generowania jednego rozwiązania dla gier 3, 4, a także 5-osobowych. Każdy z powyższych zbiorów oddzielony został w tabeli podwójną poziomą linią. Puste miejsca w tabeli oznaczają, iż dany algorytm w żadnym z 30 przypadków nie wygenerował satysfakcjonującego rozwiązania. Warto również dodać, iż przedstawione w tabeli czasy dotyczą tylko i wyłącznie sytuacji, w których rozwiązanie zostało znalezione w czasie krótszym niż maksymalna dopuszczalna wartość (300 sekund dla gier 3-osobowych oraz 600 sekund dla gier 4 i 5-osobowych). Przedstawione czasy dotyczą tylko pierwszego znalezionego rozwiązania (algorytmy GNM oraz ADE). Całkowity czas działania algorytmu nie był w tej sytuacji brany pod uwagę. Przede wszystkim należy zwrócić uwagę na wyraźną przewagę algorytmu GNM dla zagadnienia wartości minimalnej generowania czasu. Tylko w jednym przypadku algorytm ADE okazał się w tej sytuacji lepszy. Dla algorytmu SM czas generowania rozwiązania nawet w przypadku gier 3-osobowych bardzo szybko wzrasta, a największe zbiory testowe okazały się już poza jego zasięgiem. Na rys. 7.1 oraz rys. 7.2 przedstawione zostały wykresy pudełkowe ukazujące zależności czasowe dla algorytmów SM oraz GNM. Dla pozostałych dwóch algorytmów czasy generowania rozwiązań cechują się niewielkim odchyleniem standardowym, dlatego zostały pominięte w zestawieniu.

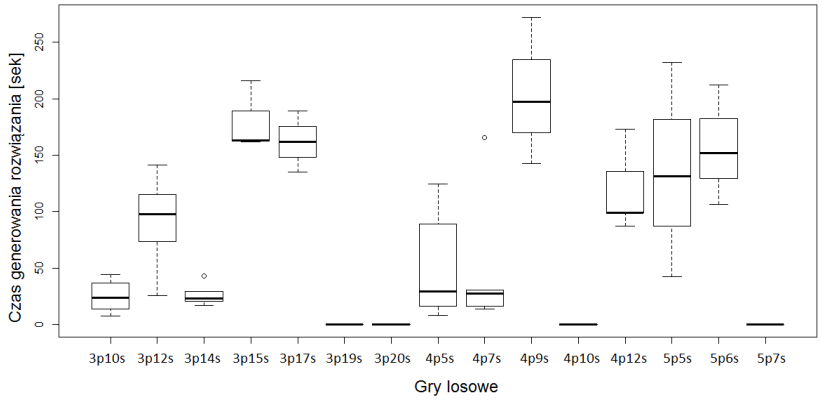
W drugiej części tabeli 7.1 widać bardzo interesujący wzrost czasu generowania rozwiązania dla algorytmów DE oraz ADE. Widać wyraźnie, iż wzrost długości genotypu (na którą wpływa liczba strategii oraz liczby graczy) prowadzi do zwiększenia czasu generowania rozwiązania. Warto jednocześnie zwrócić uwagę na wiersz dotyczący gier 3-osobowych oraz 10 strategii, a także 4-osobowych i 5 strategii. W drugim przypadku dla algorytmu DE długość genotypu wynosi tylko 20 elementów, natomiast w pierwszym 30. Stąd też czas generowania rozwiązania dla gry 4-osobowej jest wyraźnie krótszy. Najistotniejsze ze względu na stabilność generowanych rozwiązań są tutaj wartości odchylenia standardowego. Algorytm GNM posiada co prawda najlepsze wartości w kolumnie minimum, jednak wartości (zwłaszcza dla dużych gier) w kolumnie odchylenia standardowego pozwalają sądzić iż wyniki nie są powtarzalne. Czas generowania rozwiązań często jest zdecydowanie większy niż wartość minimum. Z kolei obydwa proponowane algorytmy DE oraz ADE dają powtarzalne wyniki, a odchylenie standardowe w zdecydowanej większości przypadków jest niewielkie.

Szczegóły dotyczące wartości minimalnych poszczególnych algorytmów dla gier losowych przedstawione zostały na rys. 7.3. Na osi poziomej przedstawione zostały wszystkie rozpatrywane wielkości gier (tylko dla gier losowych), natomiast na osi pionowej zaznaczono czas generowania rozwiązania. Wykres przedstawia zestawienie 4 algorytmów dla gier 3, 4 oraz 5-osobowych. Bardzo istotny element przed-

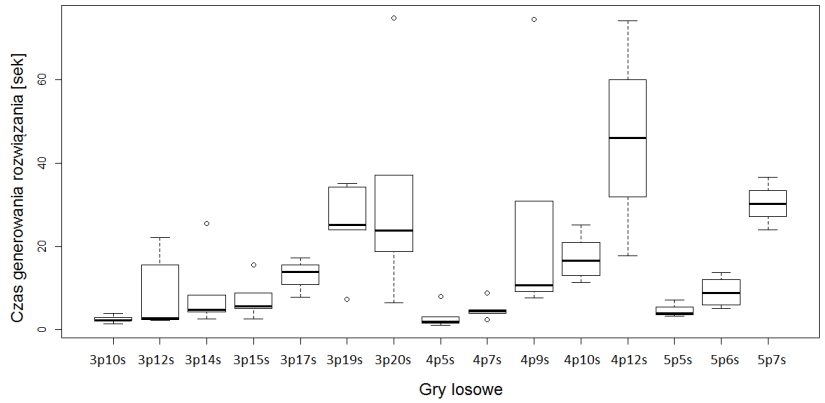
Tabela 7.1: Czas działania algorytmu (znalezienie pierwszego rozwiązania) w sekundach - gry losowe. Pierwsza część tabeli to wyniki dla algorytmów SM oraz GNM, z kolei druga część to proponowane w rozprawie algorytmy DE oraz ADE (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	SM					GNM				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	7,41	44,11	25,16	23,54	14,31	1,35	3,99	2,5	2,29	0,75
3p12s	25,69	141,32	90,79	98,08	48,44	**2,17**	22,2	8,68	2,79	9,57
3p14s	16,58	42,95	26,52	23,28	11,46	**2,61**	25,59	7,34	4,82	7,18
3p15s	161,85	215,74	180,23	163,12	30,75	**2,5**	15,63	7,53	5,67	5,04
3p17s	134,83	189,35	162,09	167,14	38,55	**7,79**	17,27	12,98	13,9	4,8
3p19s	-	-	-	-	-	**7,23**	35,19	25,2	25,23	11,25
3p20s	-	-	-	-	-	**6,48**	74,95	32,23	23,76	29,62
4p5s	8,3	124,4	53,49	29,3	50,81	**1,05**	7,94	2,78	1,82	2,26
4p7s	13,66	165,88	50,81	27,59	64,72	**2,36**	8,81	4,83	4,43	2,4
4p9s	142,63	272,24	204,14	197,56	65,05	**7,58**	74,63	23,08	10,67	24,15
4p10s	-	-	-	-	-	**11,32**	25,11	17,39	16,58	6,19
4p12s	98,74	173,12	115,76	98,74	51,02	**17,83**	74,25	46,04	46,04	39,89
5p5s	42,53	232,32	135,48	131,61	94,95	**3,33**	7,05	4,74	3,86	2,01
5p6s	106,55	212,47	156,97	151,91	53,14	**5,04**	13,76	9,11	8,82	4,13
5p7s	-	-	-	-	-	**24,04**	36,58	30,31	30,31	8,86

gra	DE					ADE				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	2,38	2,4	2,38	2,38	**0,01**	**1,25**	1,42	1,34	1,34	0,05
3p12s	5,85	5,88	5,87	5,87	**0,01**	2,36	2,63	2,45	2,41	0,09
3p14s	12,84	12,87	12,86	12,86	**0,01**	4,45	5,16	4,61	4,52	0,21
3p15s	18,35	18,57	18,4	18,37	**0,07**	5,99	6,82	6,26	6,19	0,28
3p17s	24,86	24,96	24,89	24,89	**0,02**	10,33	12,12	10,79	10,63	0,54
3p19s	54,38	55,43	54,5	54,42	**0,26**	15,49	16,68	15,85	15,92	0,38
3p20s	157,8	158,91	158,09	158,02	**0,27**	20,25	22,88	21,17	20,76	1,06
4p5s	0,73	0,94	0,83	0,82	**0,04**	1,12	1,85	1,37	1,25	0,28
4p7s	4,46	4,56	4,51	4,52	**0,03**	6,13	6,65	6,29	6,24	0,2
4p9s	20,06	20,37	20,21	20,23	**0,1**	22,57	23,73	23,01	22,98	0,41
4p10s	39,14	40,84	39,72	39,57	**0,39**	39,56	40,94	40,11	39,98	0,53
4p12s	124,83	125,46	125,08	125,08	**0,16**	102,75	107,46	104,81	104,8	1,54
5p5s	28,07	29,05	29,03	29,03	**0,01**	22,66	25,13	23,69	23,62	0,7
5p6s	106,84	107,51	107,08	107,04	**0,18**	57,87	68,24	62,47	62,28	3,69
5p7s	201,9	204,75	203,38	203,43	**0,86**	170,84	190,62	177,83	176,3	6,57



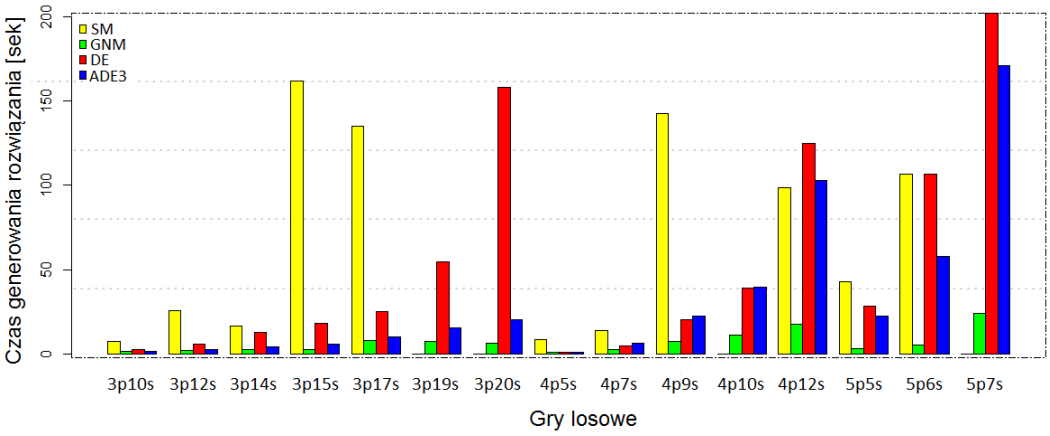
Rysunek 7.1: Wykres pudełkowy dla czasu generowania rozwiązania w algorytmie SM - gry losowe



Rysunek 7.2: Wykres pudełkowy dla czasu generowania rozwiązania w algorytmie GNM - gry losowe

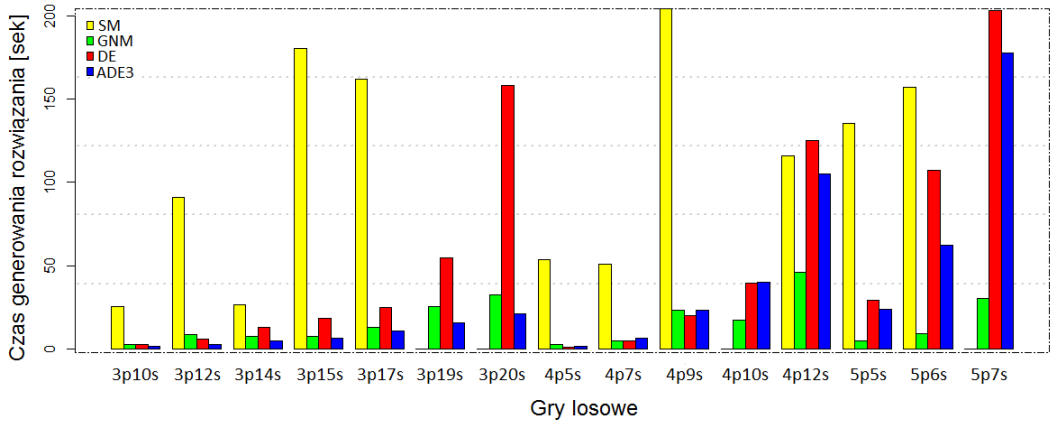
stawiony na wykresie to wyjątkowa nieregularność dotycząca algorytmu SM. Co ciekawe, minimalny czas generowania rozwiązania dla gier 5-osobowych o 6 strategiach jest niższy, niż wartość odpowiadająca grze 4-osobowej i 9 strategiach. W 4 przypadkach metoda ta okazała się nieskuteczna. Zaleta dotycząca czasu generowania rozwiązania dla algorytmu GNM jest tutaj szczególnie widoczna. Jednocześnie warto zwrócić uwagę, iż algorytm ADE w przypadku gier 3-osobowych zapewnia niemal liniowy wzrost czasu generowania rozwiązania. Wyraźnie odstająca wartość w ostatnim przypadku testowym spowodowana była koniecznością zwiększenia liczby iteracji algorytmu.

Wykres przedstawiający średni czas generowania rozwiązania dla 4 algorytmów przedstawiono na rys. 7.4. Niestabilność algorytmu SM jest w tym przypadku szczególnie istotna i nie dotyczy już tylko wybranych wielkości gier (jak to miało miejsce na rys. 7.3). Dla algorytmu GNM wyraźnie widać rozbieżność pomiędzy



Rysunek 7.3: Minimalny czas generowania rozwiązania - gry losowe

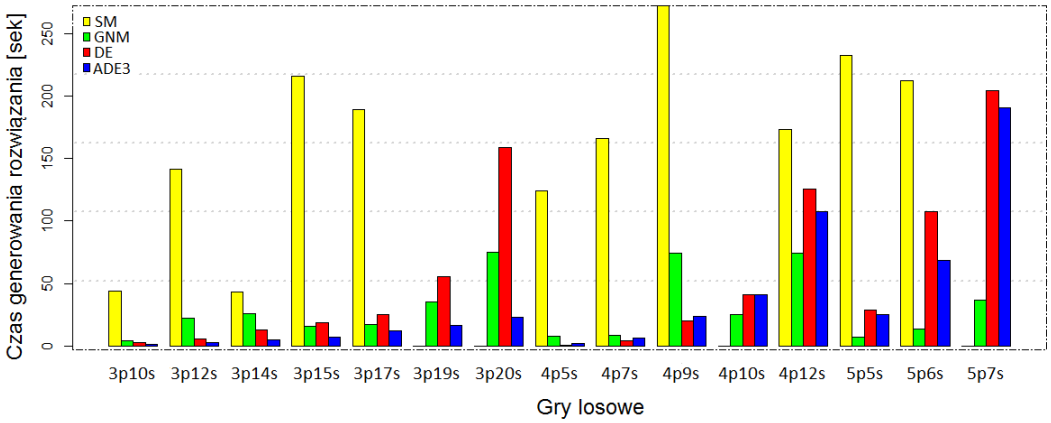
wartościami minimalnymi (rys. 7.3) a średnimi czasami generowania rozwiązania. Z kolei wykresy dotyczące proponowanych w rozprawie podejść wskazują na niemal identyczne wartości, jak miało to miejsce w przypadku poprzedniego wykresu.



Rysunek 7.4: Średni czas generowania rozwiązania - gry losowe

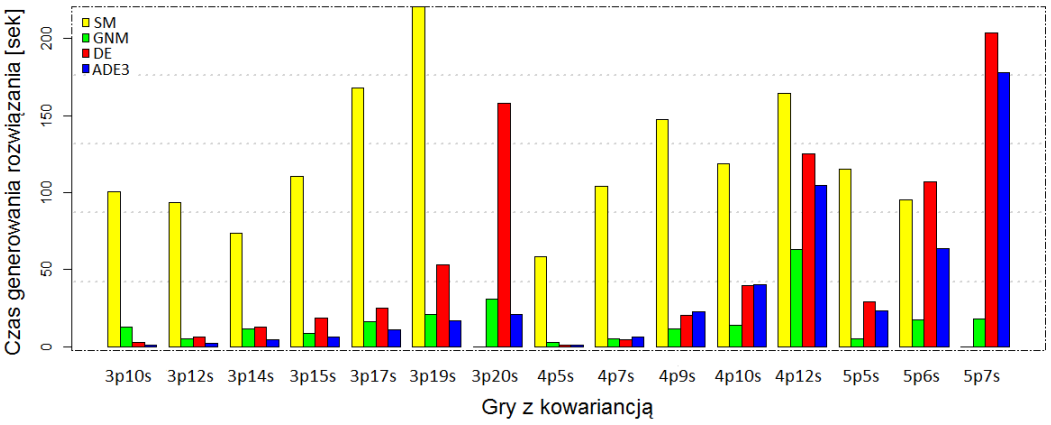
Z kolei rys. 7.5 przedstawia maksymalne czasy generowania rozwiązania dla poszczególnych wielkości gier losowych. Tutaj wyraźnie zaznaczona jest przewaga proponowanego algorytmu ADE, w którym to dla każdej z gier 3-osobowych możliwe było wygenerowanie rozwiązania w czasie znacznie mniejszym niż algorytm GNM. Algorytm SM w tym zestawieniu wypada zdecydowanie najgorzej. Nawet dla niewielkich gier 3-osobowych czas generowania jednego rozwiązania znacznie przekraczał czasy uzyskiwane w pozostałych podejściach.

W ramach zestawienia dodane zostały także wykresy dotyczące dwóch pozostałych typów gier: gier z kowariancją oraz gier typu RoadMap. Na rys. 7.6 przedstawione zostały szczegóły dotyczące średnich czasów generowania rozwią-



Rysunek 7.5: Maksymalny czas generowania rozwiązania - gry losowe

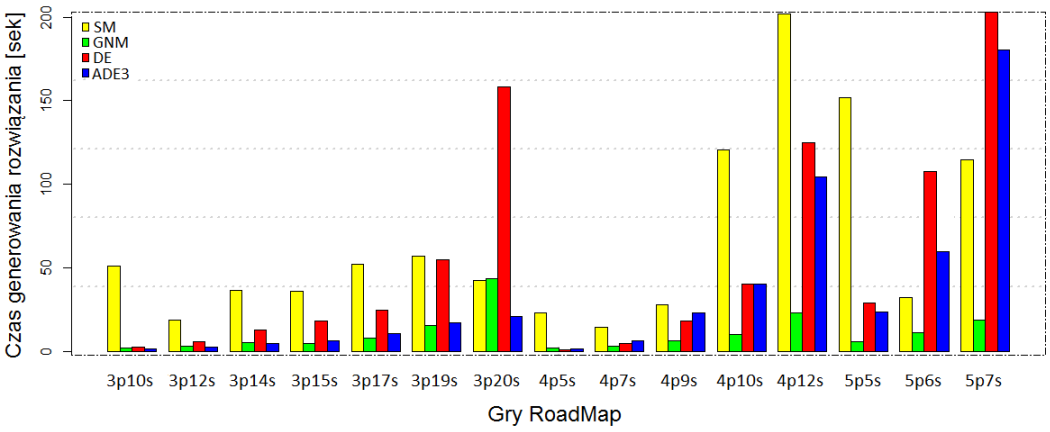
zania dla gier z kowariancją. Algorytm SM pozwala na uzyskanie nieco lepszych wyników niż w przypadku gier losowych - wartości te zwłaszcza dla gier 4 oraz 5-osobowych są mniejsze. Interesujący jest fakt, iż algorytm GNM dla tego szczególnego typu gier wydaje się dawać zdecydowanie gorsze wyniki niż w przypadku gier losowych. Dla gier 3-osobowych przewaga algorytmu ADE jest w tym eksperymencie dość istotna.



Rysunek 7.6: Średni czas generowania rozwiązania - gry z kowariancją

Ostatni rozpatrywany problem dotyczy gier RoadMap. Szczegółowe wyniki czasów dla powyższego typu oraz dla gier z kowariancją przedstawione zostały w dodatku *A*. Z kolei na rys. 7.7 przedstawione zostały średnie czasy generowania rozwiązania dla 4 algorytmów. Jest to kolejny typ gier (po grach z kowariancją), w którym to algorytm SM pozwala uzyskać średnie wyniki w czasie istotnie niższym, niż miało to miejsce w przypadku gier losowych (gry 3-osobowe). Średnie czasy generowania rozwiązania również w przypadku gier typu RoadMap dla algorytmu są gorsze, niż w przypadku rozwiązań dotyczących gier losowych. Wyniki

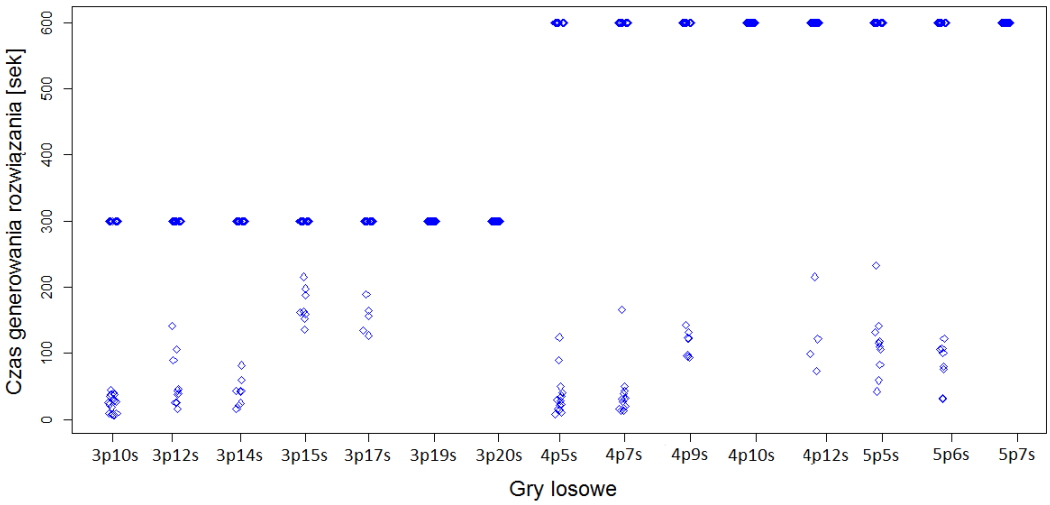
dotyczące zarówno gier RoadMap, jak i gier z kowariancją pozwalają sądzić, iż algorytmy SM oraz GNM nie są skuteczne dla dowolnych typów problemów. Z kolei dla tych samych problemów zarówno DE jak i ADE umożliwiają uzyskanie rozwiązań powtarzalnych a czasy ich generowania są do siebie zbliżone.



Rysunek 7.7: Średni czas generowania rozwiązania - gry RoadMap

Każdy z omawianych algorytmów, dla każdego typu gier uruchomiony został 30 razy. Do tej pory omówione zostały wyniki dotyczące wartości uśrednionych, a także minimalnych oraz maksymalnych. Nie zostało wyraźnie zaznaczone, iż istniejące algorytmy SM oraz GNM nie w każdym wypadku umożliwiały znalezienie satysfakcjonującego rozwiązania. Ponadto, wraz ze wzrostem złożoności problemu, liczba rozwiązań akceptowalnych malała. W celu określenia liczby uruchomień algorytmu, które umożliwiły znalezienie rozwiązania, zastosowano modyfikację wykresu rozrzutu, który ponadto pozwala wskazać zakres zmian w czasie generowania rozwiązania. Na rys. 7.8 przedstawiono wyniki dla algorytmu SM. Jeden punkt na wykresie oznacza jedno uruchomienie algorytmu. Zestawienie obejmuje gry 3, 4 oraz 5-osobowe. Limit działania algorytmu dla gier 3-osobowych wynosił 300 sekund, natomiast dla pozostałych zbiorów - 600 sekund. Nagromadzenie punktów w dolnej części wykresu przy najłatwiejszych problemach 3 oraz 4-osobowych świadczy o tym, iż w dużej części przypadków satysfakcjonujące rozwiązanie zostało wygenerowane w dozwolonym czasie. Wraz ze wzrostem złożoności problemu, liczba uruchomień zakończonych niepowodzeniem rośnie. Świadczy o tym duża liczba punktów nagromadzona przy wartości 300 sekund (dla gier 3-osobowych), a także 600 (dla gier 4 oraz 5-osobowych). W przypadku najbardziej złożonych zbiorów, rozwiązanie w ogóle nie zostało znalezione. Podobne wyniki dotyczące gier z kowariancją, a także gier typu RoadMap przedstawione zostały w dodatku.

Rys. 7.9 przedstawia czasy generowania rozwiązań dla algorytmu GNM. Widać wyraźnie zdecydowaną przewagę algorytmu GNM nad algorytmem SM. Liczba uruchomień algorytmu zakończonych sukcesem jest w tym wypadku zdecydowa-

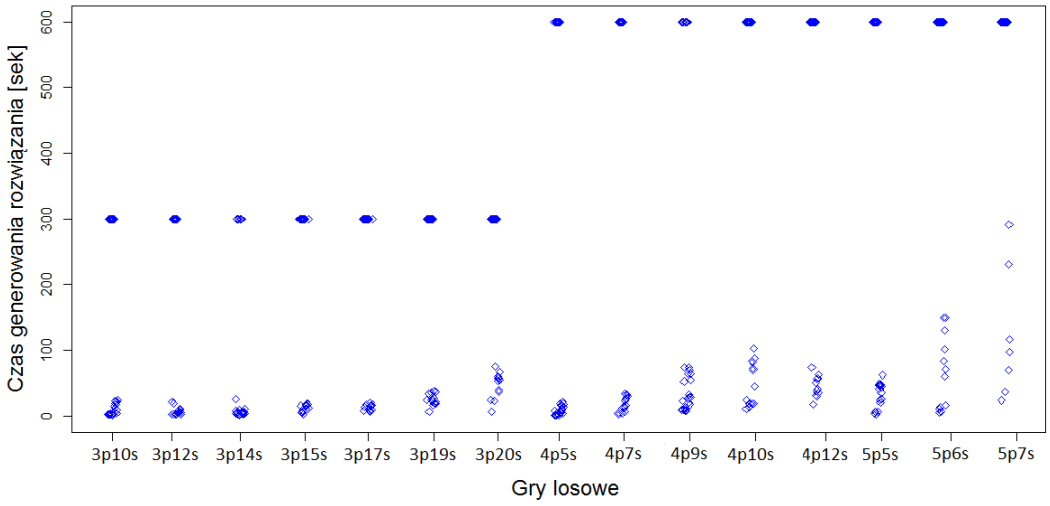


Rysunek 7.8: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm SM

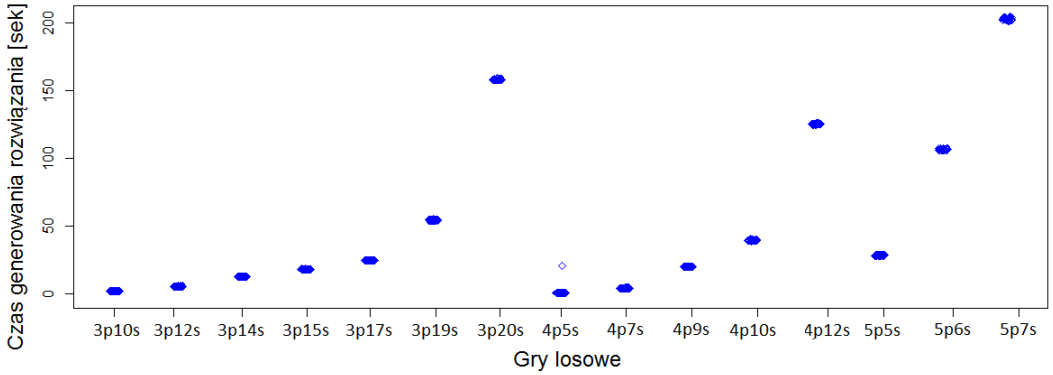
nie większa, choć oczywiście dla najbardziej złożonych gier wartość ta jest wciąż niewielka. Z drugiej strony jednak, dla mniejszych zbiorów testowych liczba znalezionych rozwiązań jest duża. Niestety pomimo zdecydowanie lepszych wyników (w porównaniu z algorytmem SM), metoda ta w większości wypadków nie może zostać zastosowana do generowania rozwiązań dla przedstawionego problemu. Rys. 7.9 podobnie jak rys. 7.8 wyraźnie wskazują na słabe strony istniejących algorytmów. Dodatkowo nie można jednoznacznie stwierdzić, co jest przyczyną takich rozbieżności w wynikach. Dla tych samych typów gier oraz tych samych wielkości gier algorytm nierzadko generuje diametralnie różne wyniki. Kwestia ta nie stanowi elementu niniejszej rozprawy, wydaje się jednak interesujące, aby wskazać w przyszłości cechy gier istotnie zwiększające trudność problemu dla istniejących algorytmów dokładnych.

Podobne doświadczenia przeprowadzone zostały dla dwóch proponowanych algorytmów. Na rys. 7.10 przedstawione zostały wyniki dotyczące algorytmu DE. Dla każdego problemu wszystkie punkty skupione są bardzo blisko siebie. Oznacza to, iż na 30 uruchomień algorytmu, w każdym przypadku rozwiązanie generowane było w podobnym czasie. Jednocześnie we wszystkich 30 przypadkach możliwe było wygenerowanie satysfakcjonującego rozwiązania. Dla najbardziej złożonych problemów, długość genotypu pojedynczego osobnika wynosiła nawet do 60 elementów. Fakt ten wpływa istotnie na czas generowania rozwiązania, co wyraźnie widać na wykresie.

Te same testy przeprowadzone zostały dla algorytmu ADE (rys. 7.11). Podobnie, jak w przypadku algorytmu DE, także algorytm ADE umożliwił wygenerowanie rozwiązań we wszystkich 30 przypadkach dla każdego ze zbiorów. Co ciekawe,



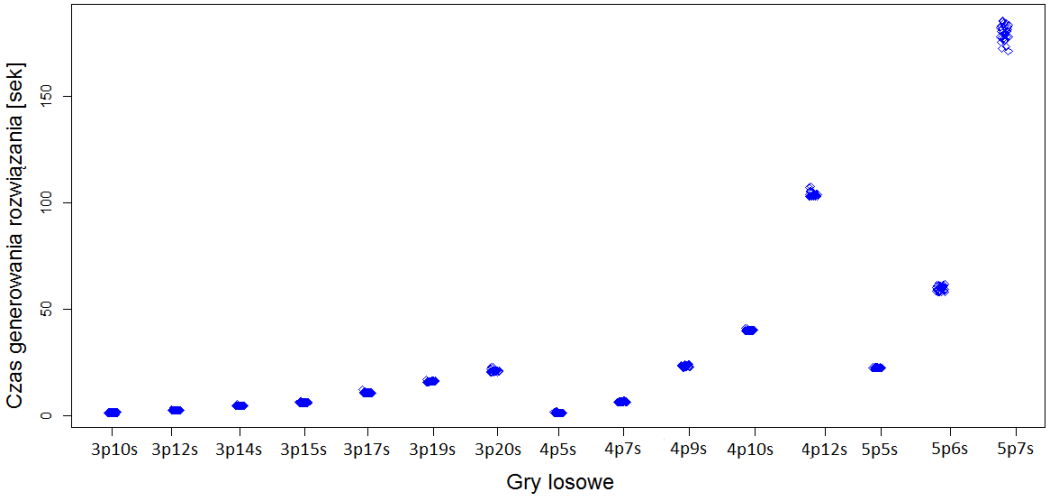
Rysunek 7.9: Czas generowania rozwiązania dla poszczególnych uruchomień - GNM



Rysunek 7.10: Czas generowania rozwiązania dla poszczególnych uruchomień - DE

ze względu na liczbę strategii aktywnych w algorytmie ADE, duże gry 3-osobowe wydają się być szczególnie proste do rozwiązania.

Pomimo pozornej przewagi algorytmów SM oraz GNM, proponowane w rozprawie rozwiązania cechują się dużą powtarzalnością wyników. Dla dowolnie wybranych typów gier możliwe jest generowanie zbliżonych wyników w kolejnych powtórzeniach algorytmu. Rozwiązania przedstawione w literaturze w tym wypadku bardzo wyraźnie odbiegają skutecznością od metod opartych na ewolucji różnicowej.



Rysunek 7.11: Czas generowania rozwiązania dla poszczególnych uruchomień - ADE

7.3 Badanie liczby uzyskanych rozwiązań oraz liczby strategii aktywnych w rozwiązaniu

W poprzednim rozdziale wskazane zostały istotne zalety proponowanego rozwiązania. Kolejna część badań dotyczy liczby generowanych przez algorytm rozwiązań, a także liczby strategii aktywnych dla danego rozwiązania. Należy pamiętać, iż z punktu widzenia użytkownika niewielka liczba strategii aktywnych w rozwiązaniu jest bardzo pożądana. Pierwszym przebadanym elementem jest liczba rozwiązań generowanych przez algorytm. W przypadku algorytmu SM dla każdego problemu generowane jest tylko jedno rozwiązanie. Algorytm DE może zostać uruchomiony kilka razy i za każdym razem generowane jest zupełnie inne rozwiązanie, jednak przyjęto, iż dla każdego przypadku testowego metoda wywoływana jest tylko raz. Tabela 7.2 przedstawia wyniki dotyczące liczby generowanych rozwiązań tylko dla algorytmów GNM oraz ADE. Dodatkowo, dla algorytmu ADE uwzględniono różną liczbę strategii aktywnych. Odpowiednio: $ADE2$ - 2 strategie aktywne dla gracza, $ADE3$ - 3 strategie aktywne oraz $ADE4$ - 4 strategie aktywne. Dla algorytmu ADE ustalone zostało, iż dla każdego przypadku testowego sprawdzane jest 10 różnych podzbiorów strategii aktywnych. Oznacza to, iż maksymalna liczba znalezionych rozwiązań wynosi 10. Algorytm GNM nie posiada takiego ograniczenia, jednak w zestawieniu pominięte zostały rozwiązania uwzględniające równowagi czyste (gdzie każdy z graczy ma tylko jedną strategię aktywną). Algorytm GNM umożliwia wygenerowanie znacznie większej liczby rozwiązań (w kilku przypadkach nawet ponad 20). Z drugiej jednak strony, kolumna dotycząca wartości minimalnej w każdym wierszu wynosi 0. Oznacza to, że na 30

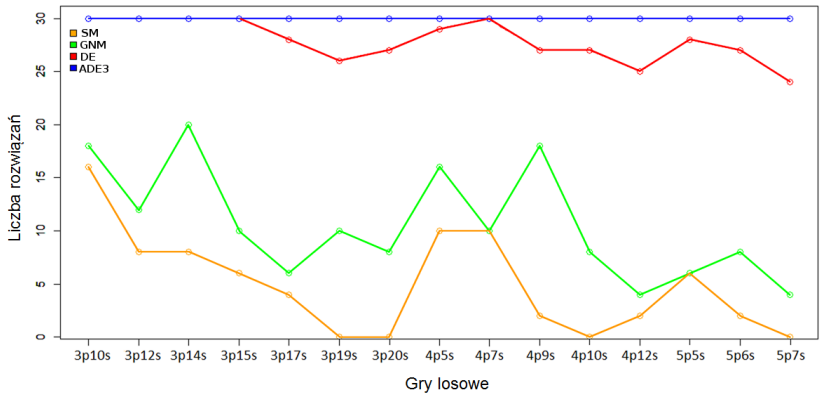
uruchomień algorytmu dla każdej rozpatrywanej wielkości gry przynajmniej raz GNM nie wygenerował żadnego satysfakcjonującego rozwiązania. Kolumna dotycząca średniej dla omawianego algorytmu pozwala sądzić, iż ogólna liczba problemów, dla których wygenerowane zostało rozwiązanie jest niewielka. W kilku przypadkach wartość średnia wahała się od 3 do wartości poniżej jeden.

Dla algorytmu ADE średnia liczba znalezionych rozwiązań jest bliska wartości maksymalnej. Oznacza to, że dla każdego z przypadków testowych algorytm umożliwia wyznaczenie dużej liczby rozwiązań. Oczywiście dla kilku problemów (szczególnie najbardziej złożonych) minimalna liczba wygenerowanych rozwiązań istotnie spadła nawet do 5 lub 6. Jednocześnie warto zaznaczyć, iż dla wersji algorytmu ADE z 4 strategiami aktywnymi liczba znalezionych rozwiązań wzrosła. 4 wylosowane strategie aktywne zwiększają szansę na to, iż wygenerowane rozwiązanie będzie bliżej optimum globalnego. Dla 2 strategii aktywnych zdecydowanie łatwiej zmniejszyć wartość ϵ, jednak rośnie też ryzyko wylosowania strategii, która nie należy do profilu strategii będącego równowagą Nasha co w konsekwencji prowadzi do odrzucenia rozwiązania.

Tabela 7.2: Liczba znalezionych rozwiązań dla algorytmów ADE oraz GNM - gry losowe oraz gry z kowariancją (min- minimum, max-maksimum, śred - wartość średnia oraz std - odchylenie standardowe)

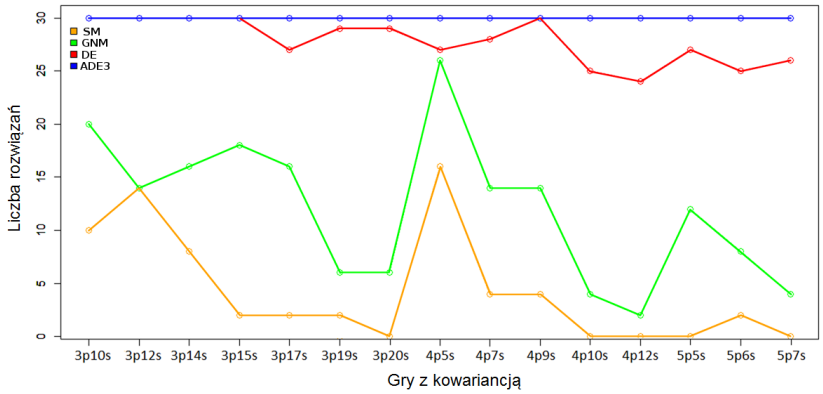
gra	GNM				ADE2				ADE3				ADE4			
	min	max	śred	std	min	max	śred	std	min	max	śred	std	min	max	śred	std
3p10s	0	24	5,7	7,5	10	10	10	0	10	10	10	0	10	10	10	0
3p12s	0	16	3,4	5,2	10	10	10	0	10	10	10	0	10	10	10	0
3p14s	0	16	4,8	5,6	10	10	10	0	10	10	10	0	10	10	10	0
3p15s	0	13	1,7	3,7	8	10	9,3	0,9	9	10	9,9	0,2	10	10	10	0
3p17s	0	15	1,2	3,8	7	10	9,4	1	8	10	9,7	0,6	8	10	9,8	9,5
3p19s	0	17	3,6	5,7	9	10	9,8	0,4	10	10	10	0	7	10	9,4	1
3p20s	0	12	2,1	4,1	8	10	9,4	0,9	8	10	9,5	0,7	8	10	9,6	0,6
4p5s	0	11	3,2	3,9	10	10	10	0	10	10	10	0	10	10	10	0
4p7s	0	7	1,1	2	10	10	10	0	10	10	10	0	10	10	10	0
4p9s	0	11	3	3,8	8	10	9,6	0,8	8	10	9,7	0,7	8	10	9,4	0,8
4p10s	0	7	1,2	2,4	5	10	9,2	1,4	7	10	9,3	1	6	10	9,3	1,2
4p12s	0	11	1	2,9	7	10	9,1	1,1	8	10	9,5	0,7	7	10	8,9	1,3
5p5s	0	7	0,7	1,8	7	10	9	1,2	6	10	8,8	1,4	6	10	8,6	1,5
5p6s	0	11	1,3	3	7	10	8,7	1,3	7	10	8,6	1,1	7	10	9,1	1,2
5p7s	0	7	0,6	1,8	6	10	9,2	1,4	6	10	8,8	1,5	7	10	9,2	1
3p10s	0	17	5,6	5,5	10	10	10	0	10	10	10	0	10	10	10	0
3p12s	0	14	3,4	4,8	10	10	10	0	10	10	10	0	10	10	10	0
3p14s	0	14	2,7	4,2	10	10	10	0	10	10	10	0	10	10	10	0
3p15s	0	21	3,8	5,6	10	10	10	0	10	10	10	0	8	10	9,8	0,5
3p17s	0	12	3,3	4,2	9	10	9,8	0,3	8	10	9,6	0,6	8	10	9,4	0,8
3p19s	0	11	1,8	3,9	8	10	9,6	0,6	7	10	9,4	1,1	7	10	9,2	1,2
3p20s	0	16	1,4	4,1	8	10	9,4	0,7	8	10	9,5	0,8	7	10	9,4	1
4p5s	0	12	5,1	3,9	10	10	10	0	10	10	10	0	10	10	10	0
4p7s	0	14	2,8	4,6	10	10	10	0	10	10	10	0	10	10	10	0
4p9s	0	9	2,2	3,1	8	10	9,2	0,8	7	10	9,2	1,2	8	10	9,4	0,7
4p10s	0	11	1	2,9	7	10	9,4	0,9	7	10	9,4	0,9	7	10	9,1	1
4p12s	0	4	0,2	1	7	10	9	0,9	6	10	8,8	1,5	6	10	9	1,6
5p5s	0	11	1,8	3,1	6	10	9	1,3	6	10	9	1,2	8	10	9,5	0,9
5p6s	0	8	1,2	2,7	5	10	8,7	1,7	7	10	9,6	0,9	8	10	9,7	0,9
5p7s	0	6	0,8	2,1	6	10	8,8	1,7	6	10	8,8	1,4	8	10	9,6	0,8

Nierzadko, aby uznać, iż dany przypadek testowy został pomyślnie rozwiązany, wystarczy znaleźć tylko jedno rozwiązanie satysfakcjonujące. Poniżej przedstawione zostaną informacje dotyczące liczby rozwiązanych przypadków testowych (na 30 możliwych). Za rozwiązanie uznane zostało tutaj znalezienie przynajmniej jednej równowagi, bądź też ϵ- równowagi Nasha. Dotyczy to w szczególności algorytmów GNM oraz ADE, które umożliwiają wygenerowanie więcej niż jednego rozwiązania podczas jednego uruchomienia algorytmu. Rozpatrywana wersja algorytmu ADE dotyczy modyfikacji z 3 strategiami aktywnymi dla jednego gracza. Na rys. 7.12 przedstawione zostały wyniki dotyczące gier losowych. Wyraźnie widać tutaj istotny problem istniejących algorytmów. Dla algorytmu SM nawet dla najprostszej rozpatrywanej sytuacji, gdzie liczba graczy wynosiła 3, a liczba strategii dla pojedynczego gracza 10 rozwiązanie zostało znalezione tylko w 16 przypadkach na 30. Jednocześnie wraz ze wzrostem trudności rozpatrywanych problemów, liczba znalezionych rozwiązań istotnie maleje, by dla najbardziej skomplikowanych przypadków testowych osiągnąć wartość 0. Dla algorytmu GNM sytuacja wygląda podobnie, jednak w tym wypadku procentowa liczba znalezionych rozwiązań wynosiła w najlepszym razie ponad 60%. Oznacza to, iż niemal co drugi przypadek testowy nie został rozwiązany przez żaden z istniejących algorytmów. Jednocześnie dwa zaproponowane w rozprawie algorytmy przybliżone pozwoliły uzyskać nie mniej niż 24 rozwiązania (algorytm DE). Algorytm ADE umożliwił wygenerowanie rozwiązania przybliżonego dla każdej wielkości zbioru testowego.



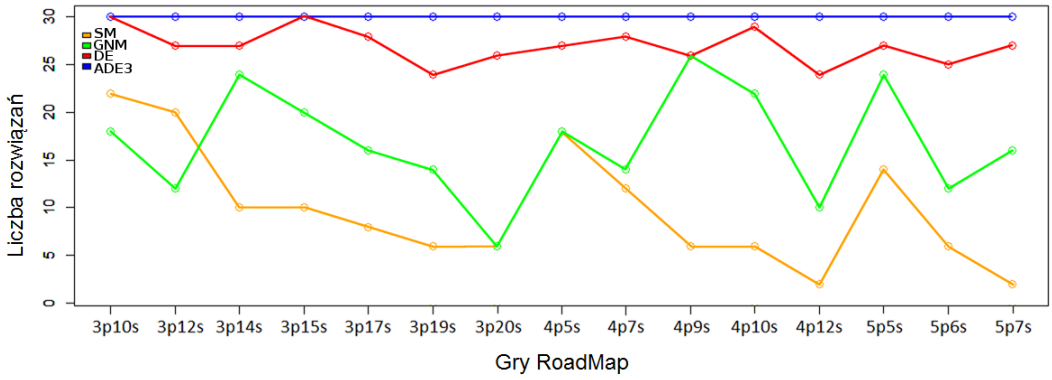
Rysunek 7.12: Liczba znalezionych rozwiązań (na 30 możliwych uruchomień - gry losowe)

Podobne zestawienie dotyczące gier z kowariancją przedstawione zostało na rys. 7.13. Dla tego typu gier wyniki są zbliżone. Algorytmy SM oraz GNM umożliwiły wygenerowanie rozwiązania tylko dla niewielkiego zbioru gier. GNM okazał się jednak nieco bardziej skuteczny, gdyż dla gier 4-osobowych z 5 strategiami rozwiązanie zostało znalezione dla 25 przypadków testowych. Algorytm ADE osiągnął po raz kolejny skuteczność równą 100%.



Rysunek 7.13: Liczba znalezionych rozwiązań (na 30 możliwych uruchomień - gry z kowariancją

Wreszcie na rys. 7.14 przedstawione zostały wyniki dotyczące gier typu RoadMap. Dla algorytmu SM jakość wyników zbliżona jest do tych przedstawionych na dwóch poprzednich wykresach. Natomiast wyraźnie widać poprawę jakości wyników dla algorytmu GNM. Dla 3 rozpatrywanych wielkości gier osiągnięto 25 rozwiązań na 30 możliwych. Warto także zaznaczyć, iż algorytm DE pomimo wyraźnej przewagi nad dwoma powyższymi metodami nie umożliwia osiągnięcia 100% skuteczności, a gorsze wyniki widoczne są dla dużych gier. Należy pamiętać, że dla takich problemów długość genotypu pojedynczego osobnika jest bardzo duża. Liczba strategii aktywnych w równowadze Nasha jest najczęściej niewielka, natomiast operator mutacji w DE prowadzi do modyfikacji całego genotypu osobnika w danej iteracji. Stąd w niektórych przypadkach osiągnięcie punktu zbliżonego do optimum nie jest możliwe. Zmniejszenie liczby strategii aktywnych w ADE umożliwia ograniczenie powyższego problemu.



Rysunek 7.14: Liczba znalezionych rozwiązań (na 30 możliwych uruchomień - gry RoadMap

Ostatni przebadany w tym rozdziale aspekt dotyczył liczby strategii aktyw-

nych dla poszczególnych algorytmów. Wcześniej zaznaczono, iż z punktu widzenia użytkownika ograniczenie liczby strategii aktywnych jest bardzo pożądane. W tabeli 7.3 przedstawione zostało zestawienie dotyczące właśnie liczby strategii aktywnych dla algorytmów SM, GNM oraz DE. Do tej pory wykazano, iż w zdecydowanej większości przypadków, dla bardziej złożonych problemów, istniejące algorytmy dokładne nie umożliwiają skutecznego rozwiązania każdego przedstawionego problemu. Z drugiej jednak strony każde rozwiązanie generowane przez SM oraz GNM cechuje się istotnym ograniczeniem liczby strategii aktywnych. W szczególności algorytm SM ma tutaj dużą przewagę. Średnia liczba strategii aktywnych dla SM nawet dla bardzo złożonych problemów nie przekracza 11 strategii aktywnych dla wszystkich graczy. W przypadku algorytmu GNM średnie wartości strategii aktywnych są większe i dla najbardziej złożonych problemów wahają się od 14 do 17. Ostatni przedstawiony w tabeli algorytm to DE. Niestety tutaj wyraźnie widoczna jest zależność pomiędzy wielkością rozpatrywanej gry a liczbą strategii aktywnych. Ta ostatnia wartość powiązana jest ściśle z liczbą strategii graczy, czyli z długością genotypu osobnika. Wraz ze wzrostem długości genotypu liczba strategii aktywnych wyraźnie rośnie, aby dla gier 3-osobowych z 20 strategiami osiągnąć w pesymistycznym wypadku wartość 43. Wykresy pudełkowe dotyczące liczby strategii aktywnych dla algorytmów SM, GNM oraz DE przedstawione zostały odpowiednio na rys. 7.15, rys. 7.16 oraz rys. 7.17.

Tabela 7.3: Liczba strategii aktywnych w rozwiązaniu - gry losowe (min- minimum, max-maksimum, śred - wartość średnia oraz std - odchylenie standardowe)

gra	SM				GNM				DE			
	min	max	śred	std	min	max	śred	std	min	max	śred	std
3p10s	5	8	6,5	1,3	9	17	11,4	2,8	15	22	18,4	2,2
3p12s	5	10	7,5	2,4	7	16	11,5	3,1	17	24	21,2	2,8
3p14s	5	9	6,5	1,7	5	17	10,8	3,4	19	26	22,6	2,1
3p15s	5	8	6,6	1,2	6	13	9,6	2,7	21	32	25,7	3,2
3p17s	7	8	7,4	0,5	14	16	14,6	1,1	24	34	28,3	3,1
3p19s	-	-	-	-	13	16	14,2	1,1	27	36	32,1	2,9
3p20s	-	-	-	-	12	18	15	3,4	26	43	33,6	4,9
4p5s	6	11	7,8	2,4	9	17	12,6	3	11	18	14,2	2,2
4p7s	6	10	7,6	1,6	8	14	10,4	2,2	14	21	17,9	2,4
4p9s	8	11	9,5	1,3	13	21	16,3	2,5	16	30	23,3	4
4p10s	-	-	-	-	11	18	14,7	3	22	30	25,8	2,8
4p12s	8	10	9	0,7	12	20	16	5,6	23	34	28,7	3,3
5p5s	7	10	9	1,2	7	16	11,6	4,5	16	21	18,4	1,5
5p6s	10	12	11	0,7	14	17	15,2	1,5	16	21	18,4	1,5
5p7s	-	-	-	-	17	18	17,5	0,7	18	25	21,6	2,2

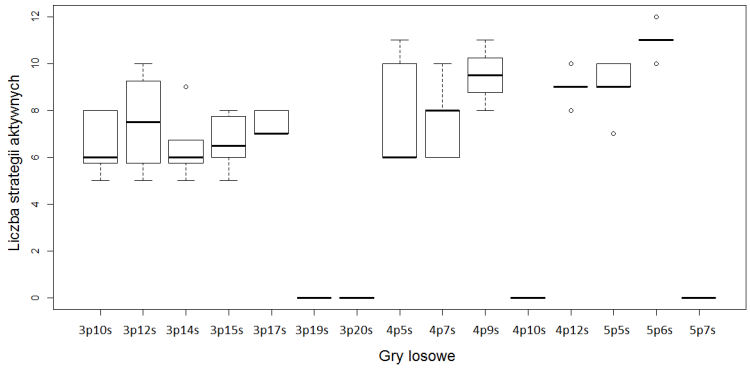
Algorytm ADE pominięty w powyższym zestawieniu łączy zalety istniejących algorytmów oraz algorytmu DE. Liczba strategii aktywnych w ADE ograniczona została przy pomocy parametru i może zostać dowolnie ustalona przez użytkownika. Możliwe jednak jest ustalenie tej wartości tak, aby była równa liczbie stra-

tegii dla danego gracza. W testowanym w rozprawie podejściu założono, iż liczba strategii aktywnych dla jednego gracza wynosi od 2 do 4. Oznacza to, iż dla największego zbioru testowego wartość ta wynosi co najwyżej 20 elementów.

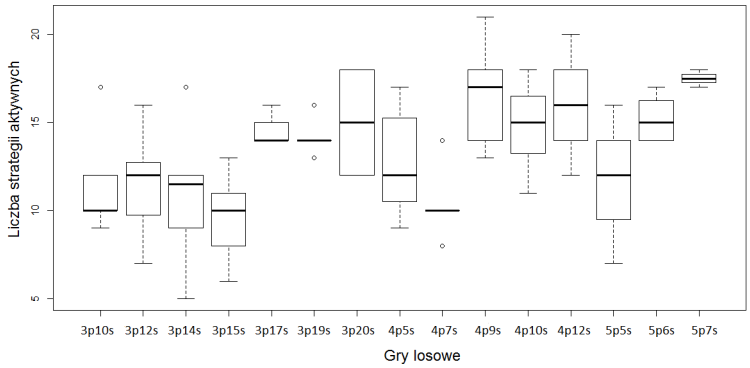
7.4 Badanie jakości rozwiązań w proponowanym algorytmie

Dwa poprzednie podrozdziały pozwoliły wykazać przewagę proponowanego podejścia nad istniejącymi w literaturze rozwiązaniami. Kolejna część rozdziału dotyczy oszacowania jakości rozwiązań generowanych przez algorytmy DE oraz ADE.

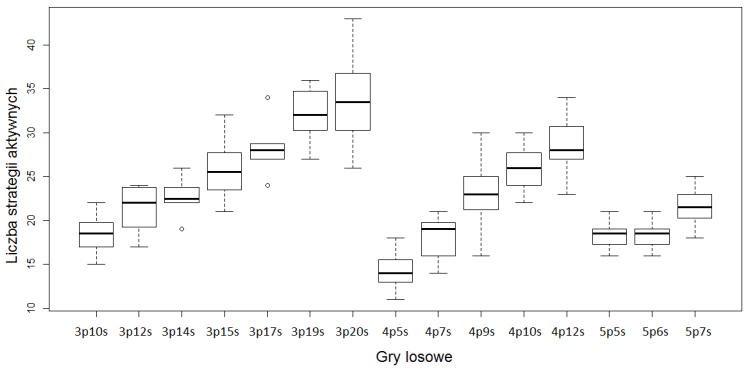
W tabeli 7.4 przedstawione zostały wartości ϵ oznaczające dokładność rozwiązania dla algorytmów DE oraz ADE. Wartość ϵ jest tutaj interpretowana jako pesymistyczna wartość wypłaty najgorszego z graczy. W rozdziale dotyczącym aproksymacji równowag Nasha wspomnianp, iż dla gier 2-osobowych najlepsze algorytmy przybliżone osiągają wartość ϵ równą 0.33. Obecnie nie istnieje żaden algorytm deterministyczny pozwalający na wyznaczenie niższej wartości ϵ w czasie wielomianowym. W tabeli zestawiono wyniki generowane przez algorytm DE oraz trzy wersje algorytmu ADE (z 2, 3 oraz 4 strategiami aktywnymi). W przypadku algorytmu DE za każdym razem rozpatrywany jest pełny zbiór strategii każdego z graczy. Pozwala to uzyskać lepsze wartości przybliżenia - nierzadko niższe od 0.1. Rezultat ten jest bardzo dobry, należy jednak zaznaczyć, iż w takim przypadku liczba strategii aktywnych jest bardzo duża. Rozpatrując to zagadnienie jako problem wielokryterialny, gdzie minimalizowana jest wartość ϵ, liczba strategii aktywnych oraz czas działania algorytmu, metoda ADE posiada istotną przewagę. Niewielkim kosztem obniżenia dokładności pozwala na zmniejszenie czasu działania algorytmu oraz w istotny sposób ogranicza liczbę strategii aktywnych dla każdego z graczy. W tabeli 7.4 pogrubioną czcionką zaznaczone zostały wartości minimalne (tylko dla algorytmu ADE). Wykresy pudełkowe dla wartości ϵ przedstawione zostały na rys. 7.18 (algorytm DE) oraz rys. 7.19 (algorytm ADE z 3 strategiami aktywnymi). Oczywiste jest, iż algorytm DE pozwala na generowanie rozwiązań o niższej wartości ϵ. Jednocześnie, wartości średnie oraz mediany nie odbiegają istotnie od wartości przedstawionych dla algorytmu DE. Oczywiście dla każdej z przedstawionych wersji algorytmu ADE średnie wartości ϵ są znacznie niższe od wspomnianej na początku podrozdziału wartości 0.33. Konkluzja jest więc następująca: algorytm ADE pozwala na ograniczenie liczby strategii aktywnych bez jednoczesnego wyraźnego pogorszenia jakości rozwiązań.



Rysunek 7.15: Wykres pudełkowy dla liczby strategii aktywnych w algorytmie SM - gry losowe



Rysunek 7.16: Wykres pudełkowy dla liczby strategii aktywnych w algorytmie GNM - gry losowe

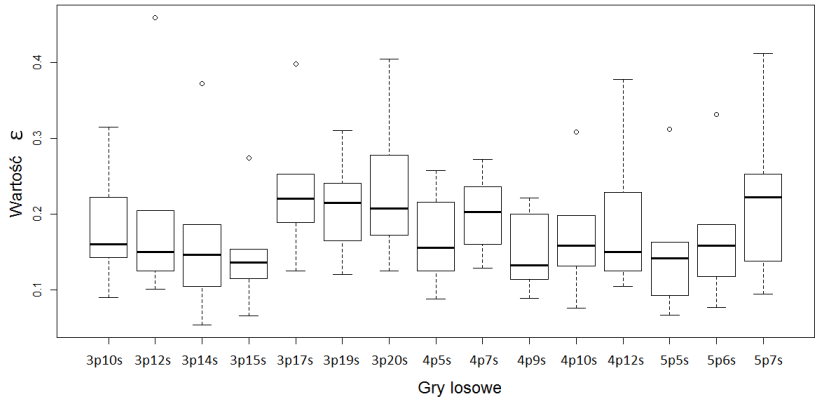


Rysunek 7.17: Wykres pudełkowy dla liczby strategii aktywnych w algorytmie DE - gry losowe

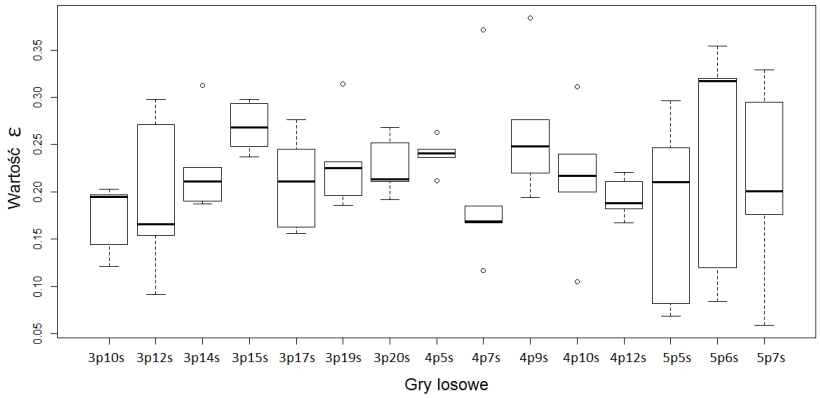
Tabela 7.4: Porównanie wartości ϵ dla algorytmów DE oraz ADE - gry losowe (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	DE					ADE2				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,09072	0,31587	0,18228	0,16051	0,06519	0,15591	0,38495	0,26715	0,25202	0,07661
3p12s	0,10181	0,4599	0,19065	0,1506	0,10121	**0,03213**	0,34522	0,16633	0,12915	0,10622
3p14s	0,05451	0,37208	0,16363	0,14635	0,0875	**0,14157**	0,34508	0,23925	0,25054	0,066
3p15s	0,0662	0,27404	0,14126	0,13695	0,05007	**0,09819**	0,29906	0,22013	0,22492	0,06177
3p17s	0,12516	0,39816	0,23183	0,22054	0,0784	**0,10541**	0,32353	0,21208	0,21337	0,07138
3p19s	0,12175	0,31045	0,20917	0,2153	0,05801	**0,07633**	0,29988	0,19397	0,20401	0,08473
3p20s	0,12574	0,40422	0,2233	0,20708	0,0761	**0,17099**	0,3489	0,24463	0,24141	0,05077
4p5s	0,08814	0,25733	0,1699	0,15683	0,05662	**0,16023**	0,37272	0,25152	0,24231	0,08628
4p7s	0,12963	0,27205	0,19852	0,20387	0,04449	**0,09643**	0,33952	0,21845	0,22415	0,11298
4p9s	0,08974	0,22165	0,15411	0,13328	0,04709	0,1799	0,29133	0,23597	0,23129	0,03969
4p10s	0,07669	0,30827	0,16636	0,15803	0,05668	0,14816	0,3559	0,24806	0,26164	0,08171
4p12s	0,10567	0,37779	0,18776	0,15093	0,08787	**0,13089**	0,328	0,23157	0,24227	0,0861
5p5s	0,06718	0,31254	0,14361	0,14215	0,06326	0,1307	0,35452	0,26809	0,31106	0,09876
5p6s	0,07791	0,33121	0,16609	0,1589	0,06947	0,14516	0,37863	0,23414	0,19155	0,0992
5p7s	0,09552	0,41251	0,22049	0,22293	0,09006	0,19771	0,32463	0,27346	0,28269	0,05012

gra	ADE3					ADE4				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	**0,12129**	0,20317	0,17247	0,19591	0,037	0,14403	0,2849	0,22412	0,23493	0,05601
3p12s	0,09212	0,2983	0,19666	0,1662	0,08602	0,16906	0,23361	0,20172	0,20632	0,02733
3p14s	0,1872	0,3137	0,22581	0,21134	0,0517	0,19097	0,27582	0,23027	0,23104	0,03198
3p15s	0,23713	0,29847	0,26905	0,26852	0,0269	0,19434	0,26542	0,22748	0,22666	0,02551
3p17s	0,15601	0,27614	0,21036	0,21105	0,05176	0,1135	0,33337	0,23049	0,23896	0,08079
3p19s	0,18671	0,31446	0,23118	0,22549	0,05041	0,17754	0,3819	0,2793	0,23977	0,0886
3p20s	0,19218	0,26882	0,2278	0,21383	0,03167	0,18529	0,29463	0,24281	0,25902	0,04551
4p5s	0,21243	0,54348	0,31199	0,2615	0,13207	0,2103	0,25544	0,23586	0,23793	0,01726
4p7s	0,11747	0,3718	0,20231	0,16979	0,09809	0,15885	0,2924	0,23729	0,25743	0,05804
4p9s	0,19417	0,38472	0,26476	0,24815	0,07377	**0,11694**	0,28201	0,20157	0,21188	0,07116
4p10s	**0,10563**	0,31195	0,21509	0,21705	0,07452	0,18613	0,36378	0,23397	0,2059	0,07348
4p12s	0,16742	0,22133	0,19431	0,18841	0,02201	0,13135	0,26853	0,21823	0,23881	0,05286
5p5s	0,06965	0,29659	0,18151	0,21059	0,10094	0,15968	0,34614	0,21549	0,18972	0,07569
5p6s	**0,08466**	0,35498	0,23943	0,31719	0,12664	0,16829	0,35167	0,24879	0,23092	0,06912
5p7s	**0,05924**	0,32998	0,21247	0,20125	0,10669	0,19288	0,33337	0,26133	0,23896	0,06008



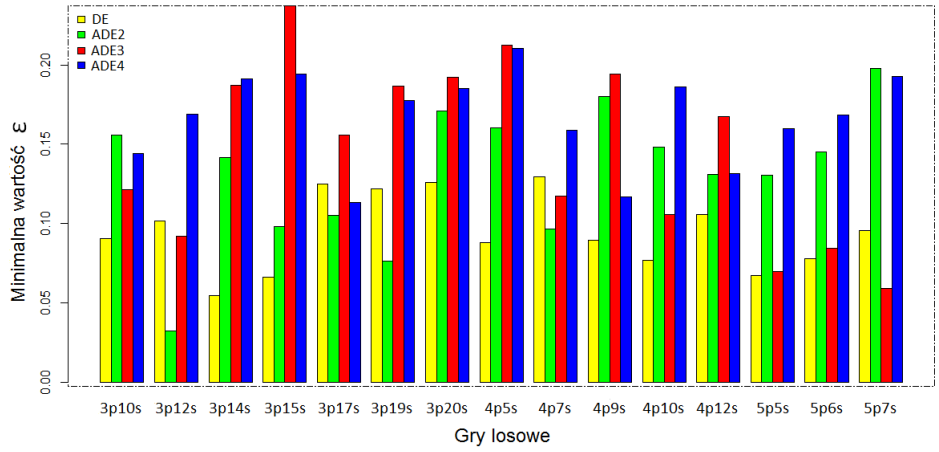
Rysunek 7.18: Wykres pudełkowy dla wartości ϵ w algorytmie DE - gry losowe



Rysunek 7.19: Wykres pudełkowy dla wartości ϵ w algorytmie ADE3 - gry losowe

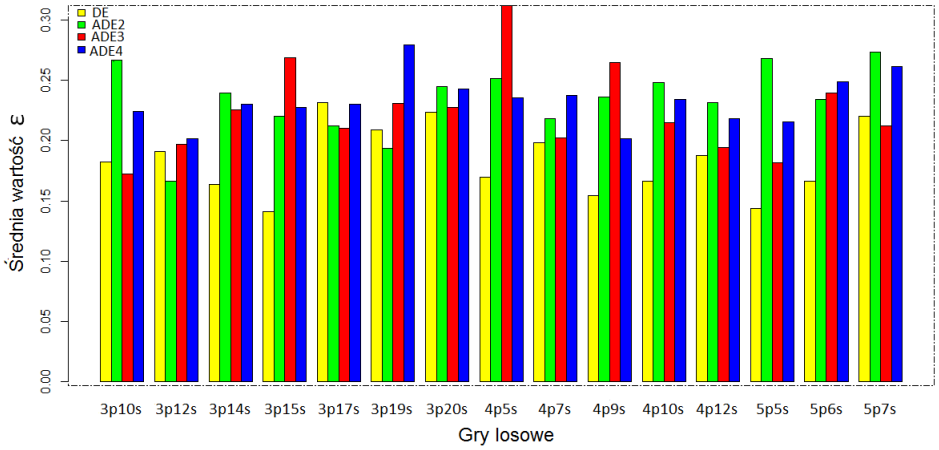
Na rys. 7.20 przedstawiono zestawienie wartości minimalnych dla 3 wersji algorytmu ADE oraz algorytmu DE dla gier losowych (podobne zestawienia dla gier z kowariancją oraz gier typu RoadMap można znaleźć w dodatku na końcu rozprawy). Wyraźnie zaznaczona jest tutaj przewaga algorytmu DE, gdzie dokładność uzyskiwanych rozwiązań dla każdej wielkości zbioru testowego nie przekroczyła progu 0.15. Interesujące wyniki dotyczą algorytmu ADE z dwoma strategiami aktywnymi. Jakość rozwiązań w tym wypadku jest w dużej części lepsza, niż dla pozostałych wersji algorytmu. W momencie wybrania ze zbioru początkowego odpowiednich strategii aktywnych genotyp posiada mniej elementów, niż ma to miejsce dla wersji z 3 oraz 4 strategiami aktywnymi. Pozwala to osiągnąć lepszą zbieżność do optimum w krótszym czasie i poprawić dokładność rozwiązania.

Z kolei dla średniej wartości ϵ różnica pomiędzy algorytmami ADE oraz DE zostaje zniwelowana, co przedstawia rys. 7.21. Oczywiście dokładność uzyskiwanych rozwiązań w przypadku algorytmu DE jest wciąż większa, jednak różnice



Rysunek 7.20: Dokładność uzyskanych rozwiązań dla gier losowych - wartości minimalne

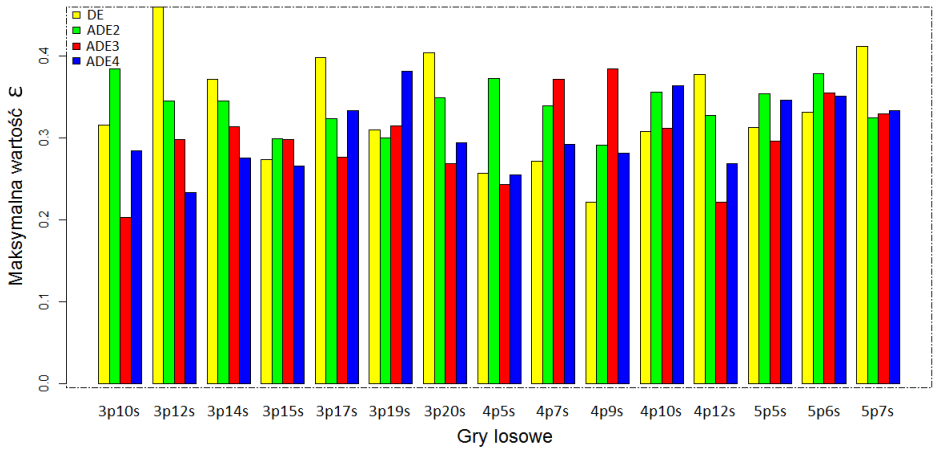
te są niewielkie. Z kolei dla różnych wersji algorytmu ADE nie można wyraźnie wskazać, która modyfikacja ma wyraźną przewagę.



Rysunek 7.21: Dokładność uzyskanych rozwiązań dla gier losowych - wartości średnie

Ostatni prezentowany wykres dotyczy maksymalnych wartości ϵ dla gier losowych. Wyniki można prześledzić na rys. 7.22. Przede wszystkim należy zwrócić uwagę, iż algorytm DE generuje w tym wypadku rozwiązania o znacznie gorszej jakości, a jego przewaga pod względem dokładności nad algorytmem ADE jest niemal całkowicie zniwelowana. Co więcej, w jednym z przypadków wartość $\epsilon = 0.4$ została przekroczona. Algorytm DE nie ma narzuconego odgórnego ograniczenia na wartość ϵ, tak więc możliwe jest wygenerowanie rozwiązania, dla którego ϵ znajduje się powyżej tej wartości. Dla algorytmu ADE dzięki etapowi próbko-

wania rozwiązań, wszelkie układy strategii, dla których uzyskana wartość ϵ jest większa od założonej (w tym wypadku 0.4) są automatycznie pomijane w dalszych rozważaniach. W algorytmie DE nie ma takiego mechanizmu, co może skutkować słabymi rozwiązaniami.



Rysunek 7.22: Dokładność uzyskanych rozwiązań dla gier losowych - wartości maksymalne

Oprócz ustalenia średnich, minimalnych oraz maksymalnych wartości ϵ dla algorytmów DE oraz ADE istotne jest także wskazanie, w ilu przypadkach na 30 uruchomień algorytmu otrzymana wartość przybliżenia była niższa niż założony próg. Badania takie przedstawione zostały w tabeli 7.5 gdzie można zobaczyć zestawienie dotyczące 3 wersji algorytmu ADE: z 2, 3 oraz 4 strategiami aktywnymi. Przyjęto 3 progi dokładności. Były to odpowiednio wartości 0.15, 0.25 oraz 0.35. Dla $\epsilon < 0.35$ właściwie w każdym przypadku dla każdej wersji algorytmu wartość rozwiązania była niższa od założonej. Co ciekawe, najwięcej niepowodzeń zanotowanych zostało dla algorytmu ADE2. Dla wartości $\epsilon < 0.25$ zaznacza się przewaga wersji z 3 oraz 4 strategiami aktywnymi. Wniosek wydaje się tutaj dość oczywisty. Otóż podzbiór strategii aktywnych jest losowany na początku działania algorytmu. Wylosowanie tylko 2 strategii aktywnych (co ma miejsce w przypadku algorytmu ADE2) prowadzi do szybkiej zbieżności oraz obniżenia wartości ϵ. Niestety nieodpowiedni wybór strategii (gdzie żadna z dwóch strategii aktywnych nie jest częścią równowagi Nasha) nie pozwala na uzyskanie satysfakcjonującego rozwiązania. Porównując to zestawienie z rys. 7.20 dotyczącym wartości minimalnych ϵ można zauważyć, iż ADE2 pozwala uzyskać niższe wartości ϵ jednak ogólna liczba satysfakcjonujących rozwiązań spada. Wreszcie ostatnia wartość $\epsilon < 0.15$ powinna być traktowana tylko poglądowo. Liczba rozwiązań, w których wartość ϵ jest mniejsza od 0.15 jest niewielka, a w bardzo wielu przypadkach nie było możliwe wygenerowanie żadnego rozwiązania o takiej wartości.

Tabela 7.5: Liczba rozwiązań algorytmu ADE o wartości ϵ niższej niż ustalona

gra	ADE2			ADE3			ADE4		
	$\epsilon < 0.15$	$\epsilon < 0.25$	$\epsilon < 0.35$	$\epsilon < 0.15$	$\epsilon < 0.25$	$\epsilon < 0.35$	$\epsilon < 0.15$	$\epsilon < 0.25$	$\epsilon < 0.35$
3p10s	0	12	24	12	30	30	6	18	30
3p12s	18	18	30	6	18	30	0	30	30
3p14s	3	12	30	0	24	30	0	24	30
3p15s	3	18	30	0	12	30	0	24	30
3p17s	6	15	30	0	24	30	6	18	30
3p19s	9	18	30	0	24	30	0	18	18
3p20s	0	18	30	0	18	30	0	12	30
4p5s	0	12	21	0	6	24	0	24	30
4p7s	10	15	30	6	24	24	0	12	30
4p9s	0	20	30	0	18	24	12	18	30
4p10s	5	15	25	6	24	30	0	24	24
4p12s	10	15	30	0	30	30	6	24	30
5p5s	5	10	25	12	24	30	0	24	30
5p6s	5	20	25	12	12	24	0	18	24
5p7s	0	10	30	6	18	30	0	18	30
3p10s	20	30	30	24	30	30	0	24	24
3p12s	20	30	30	6	30	30	8	30	30
3p14s	25	25	25	12	30	30	8	30	30
3p15s	5	30	30	18	30	30	0	30	30
3p17s	15	30	30	18	30	30	16	24	30
3p19s	15	30	30	6	30	30	8	30	30
3p20s	20	30	30	18	24	30	16	30	30
4p5s	20	30	30	18	30	30	24	30	30
4p7s	20	30	30	12	24	30	16	24	30
4p9s	10	25	30	12	30	30	0	24	30
4p10s	20	30	30	18	24	30	16	30	30
4p12s	5	25	30	24	30	30	24	30	30
5p5s	15	25	30	18	30	30	24	30	30
5p6s	20	30	30	24	30	30	30	30	30
5p7s	5	20	25	30	30	30	8	24	30
3p10s	0	12	24	6	24	30	8	24	30
3p12s	8	24	30	6	12	24	0	16	30
3p14s	8	18	30	0	12	30	0	16	30
3p15s	16	30	30	0	18	30	16	24	30
3p17s	0	16	24	0	30	30	0	30	30
3p19s	0	0	30	6	6	30	0	0	30
3p20s	0	24	30	0	12	24	8	16	30
4p5s	0	8	24	0	6	30	0	16	30
4p7s	8	16	24	0	18	30	8	16	30
4p9s	0	16	24	6	18	30	0	8	30
4p10s	0	16	30	0	12	30	0	16	30
4p12s	8	24	24	0	6	30	8	16	30
5p5s	0	8	30	0	6	18	0	16	24
5p6s	0	8	24	0	6	24	0	8	30
5p7s	8	8	24	0	18	30	0	0	30

7.5 Badanie zbieżności algorytmu z zastosowaniem miary Q-measure oraz entropii

Skuteczność proponowanego podejścia wykazana została w poprzednich podrozdziałach. W tej części rozdziału przedstawione zostaną szczegóły dotyczące określenia zbieżności populacji przy pomocy miary Q-measure oraz przedstawione zostaną informacje związane z liczebnością populacji w kolejnych iteracjach algorytmu. W [136] zaproponowane zostało interesujące podejście do oceny zbieżności populacji. Podejście to pierwotnie zaproponowane zostało właśnie dla algorytmu ewolucji różnicowej. Na początku wymagane jest wprowadzenie kilku dodatkowych oznaczeń:

- g_{max} - maksymalna liczba iteracji algorytmu;
- E_{max} - maksymalna liczba ewaluacji funkcji ($E_{max} = g_{max} \cdot NP$);
- n_t - liczba powtórzeń algorytmu;
- E_i - liczba ewaluacji funkcji w i-tym powtórzeniu algorytmu;
- ϵ - dokładność rozwiązania.

Każda próba ma maksymalną liczbę iteracji. W przypadku, gdy po założonej liczbie g_{max} rozwiązanie o dokładności równej ϵ nie zostanie znalezione - próba uznawana jest za nieważną.

Pierwszym badanym elementem jest wartość miary zbieżności obliczana według wzoru:

$$C = \frac{\sum_{j=1}^{n_c} E_j}{n_c}, \tag{7.1}$$

gdzie: j jest liczbą uruchomień zakończonych sukcesem, gdzie w liczbie iteracji mniejszej niż g_{max} znalezione zostało rozwiązanie o wartości ϵ.

W tabeli 7.6 przedstawiono średnią liczbę ewaluacji funkcji wymaganych do osiągnięcia zadanej wartości ϵ. Przebadano 3 wielkości gier losowych: 3-osobową grę z 15 strategiami oznaczoną jako $3p15s$, 4-osobową grę z 9 strategiami - $4p9s$ oraz grę 5-osobową z 6 strategami - $5p6s$. Dla każdej z tych gier testy dotyczyły algorytmu ADE z 2, 3 oraz 4 strategiami aktywnymi, natomiast ustalone wartości ϵ to 0.15, 0.25 oraz 0.35. Maksymalna liczba ewaluacji funkcji w jednym przebiegu algorytmu wynosiła dla gier 3 i 4-osobowych 22500, natomiast dla gier 5-osobowych 50000. W tabeli podane są średnie wartości wymaganej liczby ewaluacji. Warto zauważyć, iż dla algorytmu $ADE2$ wartości te są przeważnie istotnie mniejsze od wersji z 3 oraz 4 strategiami aktywnymi. W wersji z 2 strategiami

Tabela 7.6: Liczba ewaluacji funkcji wymagana do osiągnięcia założonej wartości ϵ (przy stałej liczbie osobników w populacji)

	ϵ	ADE2	ADE3	ADE4
3p15s	0.15	12622	14490	16815
3p15s	0.25	11793	9803	12459
3p15s	0.35	9969	6975	9875
4p9s	0.15	14060	13302	13907
4p9s	0.25	13149	11938	12433
4p9s	0.35	11511	8763	11736
5p6s	0.15	35741	34020	33037
5p6s	0.25	31082	25888	26757
5p6s	0.35	21520	24807	26712

Tabela 7.7: Liczba rozwiązań algorytmu ADE o wartości ϵ niższej niż ustalona

	ϵ	ADE2	ADE3	ADE4
3p15s	0.15	33%	36%	40%
3p15s	0.25	56%	50%	60%
3p15s	0.35	84%	86%	80%
4p9s	0.15	56%	43%	40%
4p9s	0.25	65%	63%	60%
4p9s	0.35	83%	70%	90%
5p6s	0.15	33%	36%	40%
5p6s	0.25	70%	63%	63%
5p6s	0.35	73%	86%	80%

aktywnymi długość genotypu osobnika jest krótsza, co znacznie ułatwia znalezienie rozwiązania o założonej wartości ϵ. Oczywiste jest także, iż większa wartość ϵ oznacza istotnie mniejszą wymaganą liczbę wywołań funkcji oceny.

Warto zestawić powyższe wyniki z danymi dotyczącymi wartości procentowej wszystkich uruchomień algorytmu zakończonych sukcesem. Wyniki te przedstawione zostały w tabeli 7.7 i odnoszą się do tych samych zbiorów testowych. Ogólna liczba uruchomień algorytmu, w której osiągnięto założoną wartość ϵ nie zależy istotnie od liczby strategii aktywnych w algorytmie ADE. Wartość ϵ bliska optimum ($\epsilon = 0.15$) prowadziła do obniżenia liczby uruchomień algorytmu zakończonych sukcesem.

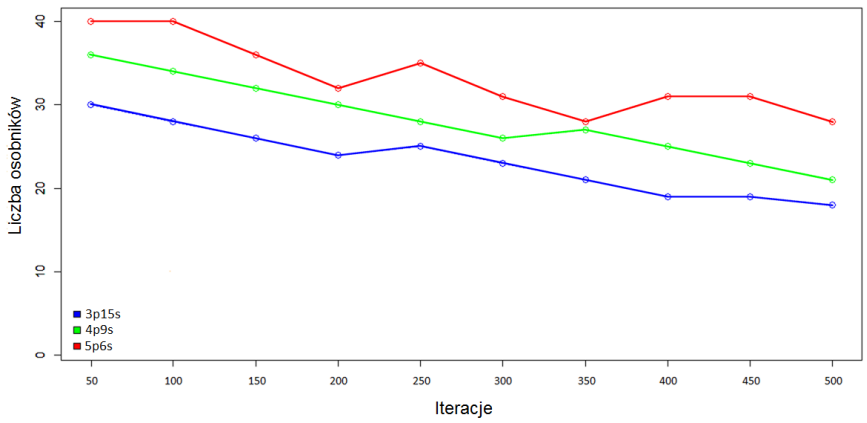
Wprowadzona w algorytmie dynamiczna zmiana wielkości populacji oparta na entropii ma na celu ograniczenie liczby ewaluacji funkcji, co prowadzi do skrócenia czasu jego działania. Podejście to opiera się na założeniu, iż wraz z czasem działania algorytmu różnorodność populacji maleje. Pozwala to na niewielkie

ograniczenie wielkości populacji bez utraty jej różnorodności. W poniższych badaniach przedstawiono wielkości populacji dla 3 wybranych gier losowych: 3, 4 oraz 5-osobowych. Maksymalna liczba osobników w populacji ustalone zostały odpowiednio na 30, 36 oraz 40.

Na rys. 7.23 przedstawiono wielkości populacji dla algorytmu $ADE2$ dla 3 przykładowych gier. Wykres ograniczono tylko do pierwszych 500 iteracji, kiedy to zmiany są najbardziej widoczne. Należy przypomnieć, iż na wielkość populacji nałożone zostały pewne ograniczenia:

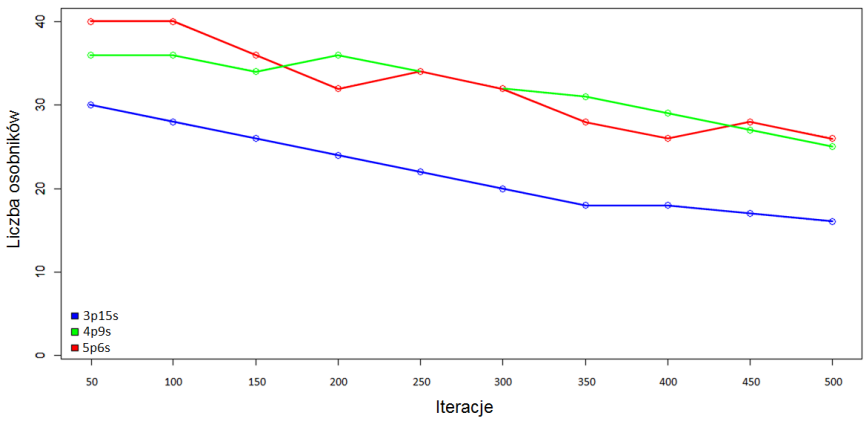
- maksymalna liczba osobników w populacji nie może przekroczyć wartości początkowej;
- minimalna liczba osobników w populacji to $\frac{1}{3}$ maksymalnej liczby osobników.

Wielkość populacji modyfikowana jest co 50 iteracji i widać wyraźnie, iż liczba osobników jet stopniowo ograniczana aż do osiągnięcia pewnego założonego minimum. Skuteczność tego podejścia jest wyraźnie widoczna w przypadku gry 4-osobowej, gdzie właściwie przy każdym odczycie liczebność populacji maleje. Dla gry 5-osobowej dwukrotnie liczebność populacji zostaje powiększona. Oznacza to zahamowanie procesu zbieżności i konieczność dodania osobników do populacji. Przedstawione badania stanowią tylko pewne uogólnienie. Interesujące może wydawać się sprawdzenie, czy zmiany populacji nie powinny następować częściej, nie jest to jednak przedmiotem badań niniejszej rozprawy.



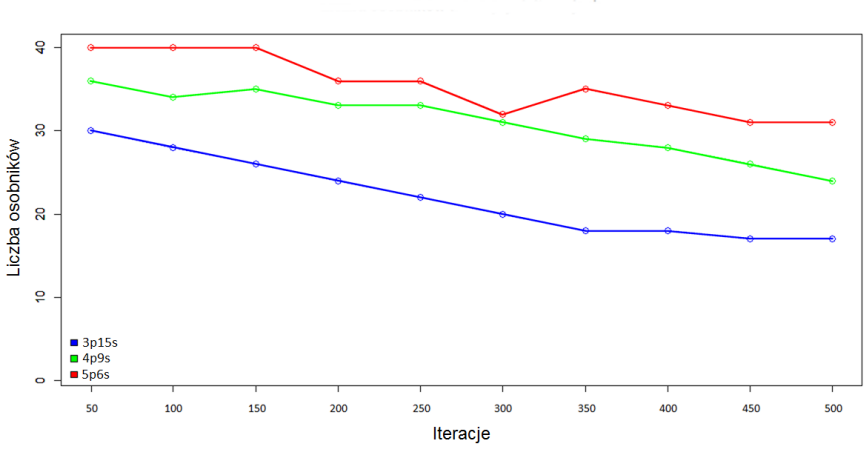
Rysunek 7.23: Analiza dynamicznie zminianej wielkości populacji dla przykładowych gier - algorytm $ADE2$

Druga wersja algorytmu ADE z 3 strategiami aktywnymi przedstawiona została na rys. 7.24. Z kolei dla gry 3-osobowej liczba osobników jest sukcesywnie zmniejszana. Dla pozostałych dwóch gier widać wyraźne wahania i wzrosty wielkości populacji.



Rysunek 7.24: Analiza dynamicznie zminianej wielkości populacji dla przykładowych gier - algorytm $ADE3$

Interesującą zależność można zaobserwować na rys. 7.25, gdzie przedstawione zostały wyniki dla algorytmu z 4 strategiami aktywnymi. W przypadku gry 5-osobowej przez pierwsze 150 iteracji wielkość populacji jest niezmienna. Oznacza to dużą zmienność populacji. Można wyciągnąć wniosek, iż dla tego problemu, gdzie genotyp osobnika zawiera 20 elementów, populacja początkowa jest na tyle różnorodna, iż nawet po 150 iteracjach większość osobników nie jest do siebie podobna oraz ich fenotyp nie wskazuje na zbliżenie się do optimum globalnego.



Rysunek 7.25: Analiza dynamicznie zminianej wielkości populacji dla przykładowych gier - algorytm $ADE4$

7.6 Statystyczna weryfikacja proponowanego podejścia

Dwa zaproponowane w rozprawie algorytmy (ewolucja różnicowa oraz adaptacyjny algorytm ewolucji różnicowej) mają w pewnym stopniu charakter losowy. Jednocześnie algorytm adaptacyjny stanowi rozwinięcie podstawowej wersji algorytmu o liczne elementy związane z adaptacją parametrów. W tej części rozdziału przedstawiony zostanie test Wilcoxona. Test Wilcoxona to test nieparametryczny służący do porównywania wyników dwóch prób zależnych. Jest on odpowiednikiem testu t Studenta dla prób zależnych. W teście określone zostały następujące hipotezy:

- H_0 - brak różnicy przy porównaniu próbek,
- H_1 - różnica pomiędzy dwoma próbkami.

W celu przeprowadzenia testu wprowadzono kryterium zastępcze uwzględniające oprócz wartości ϵ także czas działania algorytmu oraz liczbę strategii aktywnych. Przyjęto prostą metodę ważonych celów z 3 funkcjami oraz trzema wagami $w_1 = w_2 = w_3 = 0.33$. Dla przeprowadzonych analiz hipoteza H_0 została odrzucona przy wartości $p < 0.01$. Oznacza to, iż rezultaty generowane przez algorytm ADE są istotnie lepsze od algorytmu DE.

Jednocześnie w celu sprawdzenia powtarzalności wyników dwóch zaproponowanych algorytmów powyższej analizie poddano każdy z algorytmów. DE oraz ADE uruchomione zostały odpowiednio po 200 razy. W każdym z przypadków zbiór podzielony został na połowę, następnie wyniki z pierwszych 100 uruchomień algorytmu zestawione zostały z pozostałymi uruchomieniami. We wszystkich przypadkach, dla gier 3, 4 oraz 5-osobowych wynik testu wskazywał na brak różnicy pomiędzy próbkami (duża wartość p). Wniosek jest więc następujący: próbki cechują się podobnymi rezultatami, a wyniki są powtarzalne.

ROZDZIAŁ 8

Analiza skuteczności algorytmu w innych problemach teorii gier

W poprzednim rozdziale przedstawione zostały wyniki badań dotyczące zastosowania algorytmu ewolucji różnicowej oraz jego adaptacyjnej wersji w jednym z najistotniejszych problemów teorii gier, jakim jest wyszukiwanie równowag Nasha i równowag przybliżonych. Proponowane metody porównane zostały z istniejącymi algorytmami. Zbadane zostały takie elementy jak: czas działania algorytmu, dokładność rozwiązania, czy też liczba strategii aktywnych. Należy jednak pamiętać, iż równowagi Nasha są tylko jednym z wielu istniejących w teorii gier problemów. Sam problem równowag Nasha określany jest jako problem klasy $PPAD$, przy czym istnieją też problemy należące z całą pewnością do klasy NP. Wśród nich można wyróżnić szereg problemów powiązanych z koncepcją równowagi Nasha, często określane jako równowagi Nasha z dodatkowymi właściwościami. W poniższym rozdziale krótko opisane zostaną przykładowe problemy, a także propozycja zastosowania proponowanych w rozprawie metod do rozwiązania tych problemów.

Wreszcie w ostatniej części rozdziału przedstawione zostaną wyniki badań dotyczące problemu określanego jako wyszukiwanie punktów ogniskowych w grach koordynacyjnych. Nie jest to problem należący do klasy NP, jednak z punktu widzenia niniejszej rozprawy istotne jest wskazać, iż proponowany adaptacyjny algorytm ewolucji różnicowej może być traktowany jako narzędzie ogólne w szeroko rozumianej tematyce teorii gier, w problemach nie tylko związanych ściśle z pojęciem równowagi Nasha.

8.1 Klasyfikacja równowag Nasha z dodatkowymi właściwościami

W 1989 roku I. Gilboa oraz E. Zemel przedstawili pewne rozważania nad złożonością dwóch problemów: równowagi Nasha oraz równowagi skorelowanej [54].

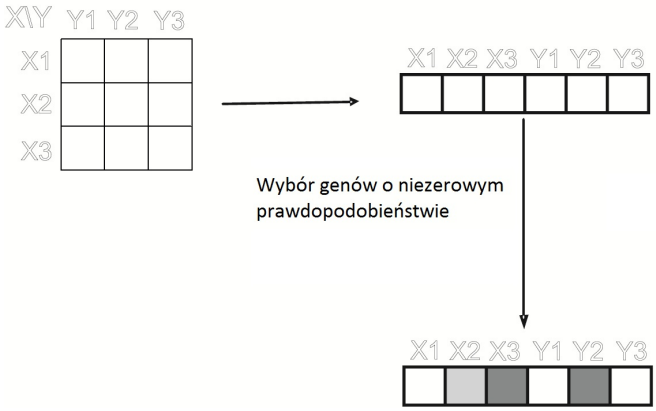
Równowaga skorelowana została wspomniana już wcześniej w niniejszej rozprawie. Należy przypomnieć, iż jest to problem istotnie łatwiejszy niż równowaga Nasha. Rozważania w dalszej części tego rozdziału dotyczą równowag Nasha z dodatkowymi właściwościami. Poniżej przedstawione zostały problemy rozważane przez I. Gilboę:

- równowaga Nasha z maksymalną wypłatą: mając daną grę G oraz wartość r, wskazać, czy istnieje równowaga Nasha, w której każdy z graczy otrzymuje oczekiwaną wypłatę nie mniejszą niż r;
- równowaga Nasha w podzbiorze: mając daną grę G oraz podzbiór strategii S dla każdego z graczy, sprawdzić, czy istnieje równowaga Nasha, gdzie wszystkie strategie niezawarte w zbiorze S mają zerowe prawdopodobieństwo wyboru;
- równowaga Nasha zawierająca podzbiór: dla danej gry G oraz podzbioru strategii S danego dla każdego z graczy, sprawdzić, czy istnieje równowaga Nasha, w której każda strategia ze zbioru S ma niezerowe prawdopodobieństwo wyboru;
- równowaga Nasha z maksymalnym wsparciem: mając daną grę G i liczbę całkowitą $r \geqslant 1$, sprawdzić, czy istnieje równowaga Nasha taka, że przynajmniej r strategii dla danego gracza ma niezerowe prawdopodobieństwo wyboru.

Interesującym problemem jest równowaga Nasha w podzbiorze. W przypadku algorytmu ADE możliwe jest wskazanie podzbioru strategii, na podstawie którego zostanie zbudowany genotyp osobnika. W tej sytuacji zaproponowane w rozprawie podejście bez problemu może zostać tutaj zastosowane. W przypadku algorytmu DE nie jest to tak oczywiste, dlatego będą wymagane pewne eksperymenty, które przeprowadzone zostaną w dalszej części rozdziału. Warto rozwinąć tutaj problem o wyszukiwanie ϵ równowagi Nasha w podzbiorze, czyli przybliżenia równowagi. Niestety w literaturze nie istnieje żaden algorytm umożliwiający przeprowadzenie analizy porównawczej, dlatego wszystkie omawiane wyniki dotyczyć będą tylko i wyłącznie proponowanych w rozprawie rozwiązań.

Drugim interesującym problemem jest wyznaczenie równowagi zawierającej wybrane strategie, przy czym prawdopodobieństwo wyboru strategii nie może być niższe od zadanej wartości. Dla algorytmów DE oraz ADE problem ten może zostać rozwiązany następująco: w początkowej fazie działania algorytmu wybrane elementy genotypu są oznaczone, a w funkcji oceny następuje sprawdzenie, czy prawdopodobieństwo wyboru danej strategii nie jest niższe niż zadana wartość.

Na rys. 8.1 jasnoszarym odcieniem zaznaczono elementy genotypu, których prawdopodobieństwo wyboru powinno wynosić przynajmniej a. Z kolei ciemnosiwy kolor odpowiada strategiom, których prawdopodobieństwo wyboru powinno



Rysunek 8.1: Przekształcenie gry w genotyp z ustalonymi strategiami

wynosić przynajmniej b. Wartości a oraz b powinny zawierać się w przedziale $(0, 1\rangle$. Podejście umożliwiające zrealizowanie tego zadania opiera się na dodatkowej funkcji kary. Funkcja kary może zostać określona następująco:

$$\forall_i, F_{kary} = \begin{cases} F_{kary} + k & \text{kiedy } g[i] = 0, i \in G, \\ F_{kary} & \text{w przeciwnym wypadku,} \end{cases}$$

gdzie:

G - zbiór indeksów genów o niezerowym prawdopodobieństwie,

k - wartość stała równa 1.

Powyższa funkcja gwarantuje, że każdy gen w zbiorze G z zerowym prawdopodobieństwem wyboru zwiększa wartość funkcji kary. Końcowa postać funkcji przystosowania dla tego problemu powinna być określona następująco:

$$f = f_1 + f_2 + F_{penalty}, \tag{8.1}$$

gdzie f_1 oraz f_2 to elementy podstawowej funkcji oceny opisane dokładnie w rozdziale 6.

8.2 Równowagi Nasha z wykluczeniem strategii

Jednym z zagadnień przedstawionych przez I. Gilboę jest równowaga Nasha z wykluczeniem określonych strategii graczy. Jest to problem o złożoności NP. W poniższym podrozdziale przedstawione zostały wyniki dotyczące wspomnianego zagadnienia dla algorytmu DE. Algorytmy SM oraz GNM nie są przystosowane do rozwiązywania tego typu problemów, z kolei algorytm ADE umożliwia wskazanie podzbioru strategii aktywnych. W tym kontekście wszystkie pozostałe ele-

menty traktowane są jako strategie wykluczone. Niniejsze wyniki dotyczą tylko algorytmu DE. Zastosowane zostały wartości parametrów wykorzystane w poprzednich badaniach.

W tabeli 8.1 przedstawione zostały czasy działania algorytmu DE dla klasycznego problemu wyszukiwania równowag Nasha. Czasy te zestawiono z problemem, gdzie wykluczone zostały odpowiednio: jedna (DE ex_1), dwie (DE ex_2) oraz trzy (DE ex_3) strategie czyste. Przede wszystkim należy zauważyć, iż pomimo większej złożoności problemu wykluczenia strategii wyniki we wszystkich przypadkach są powtarzalne. Ponadto, nawet dla największych gier (3-osobowe z 20 strategiami oraz 4-osobowe z 12 strategiami) złożoność problemu nie wpłynęła w żaden istotny sposób na czas działania algorytmu. Było to przewidywalne, ponieważ liczba iteracji, a także wielkość populacji została zachowana. Dodatkowo, dostosowanie funkcji oceny do przedstawionego problemu wymagało dodania jednego prostego elementu - funkcji kary, która uniemożliwia włączenie do rozwiązania strategii zabronionych. Nie jest to operacja kosztowna obliczeniowo.

Większa złożoność problemu przy jednoczesnym zachowaniu zbliżonego czasu działania algorytmu nie prowadzi do obniżenia dokładności rozwiązań. W tabeli 8.2 przedstawione zostały szczegółowe wyniki dotyczące dokładności rozwiązań. Porównanie dotyczy algorytmu ewolucji różnicowej DE (wyniki dotyczące wartości ϵ pochodzą z poprzedniego rozdziału) oraz algorytmu DE z wykluczeniem 1, 2 oraz 3 strategii. Wytłuszczone elementy wskazują na najlepsze znalezione wartości ϵ. Nie można jednoznacznie stwierdzić, iż problem dotyczący wykluczenia strategii prowadzi jednoznacznie do pogorszenia jakości uzyskiwanych rozwiązań. Interesująca wydaje się być obserwacja, iż w przypadku największych zbiorów testowych wykluczenie strategii prowadzi do poprawienia wartości ϵ w kolumnie *max* dotyczącej najgorszych rozwiązań. Jedną z przyczyn może być tutaj niejakie ograniczenie długości genotypu (poprzez wykluczenie, czyli odrzucenie pewnych jego fragmentów).

Tabela 8.1: Czas działania algorytmu (znalezienie pierwszego rozwiązania) w sekundach - dla gier losowych. DE to oryginalne wartości czasu działania, natomiast DE ex_1, DE ex_2 oraz DE ex_3 to odpowiednio DE z wykluczeniem 1, 2 oraz 3 strategii (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	DE					DE ex_1				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	2,38	2,4	2,38	2,38	0,01	2,38	2,44	2,39	2,38	0,02
3p12s	5,85	5,88	5,87	5,87	0,01	5,85	5,92	5,87	5,87	0,02
3p14s	12,84	12,87	12,86	12,86	0,01	12,83	12,9	12,85	12,85	0,02
3p15s	18,35	18,57	18,4	18,37	0,07	18,36	18,46	18,39	18,38	0,02
3p17s	24,86	24,96	24,89	24,89	0,02	24,85	24,91	24,87	24,87	0,01
3p19s	54,38	55,43	54,5	54,42	0,26	54,81	54,89	54,85	54,85	0,02
3p20s	157,8	158,91	158,09	158,02	0,27	157,5	158,99	157,95	157,94	0,03
4p5s	0,73	0,94	0,83	0,82	0,04	0,78	0,88	0,81	0,81	0,02
4p7s	4,46	4,56	4,51	4,52	0,03	4,47	4,53	4,48	4,47	0,02
4p9s	20,06	20,37	20,21	20,23	0,1	20,44	20,5	20,48	20,47	0,12
4p10s	39,14	40,84	39,72	39,57	0,39	39,13	39,76	39,48	39,44	0,24
4p12s	124,83	125,46	125,08	125,08	0,16	124,84	125,97	124,9	124,91	0,14
5p5s	28,07	29,05	29,03	29,03	0,01	28,12	29,12	28,75	28,67	0,03
5p6s	106,84	107,51	107,08	107,04	0,18	106,89	107,64	107,21	107,17	0,09
5p7s	201,9	204,75	203,38	203,43	0,86	201,66	203,77	202,81	202,65	0,94

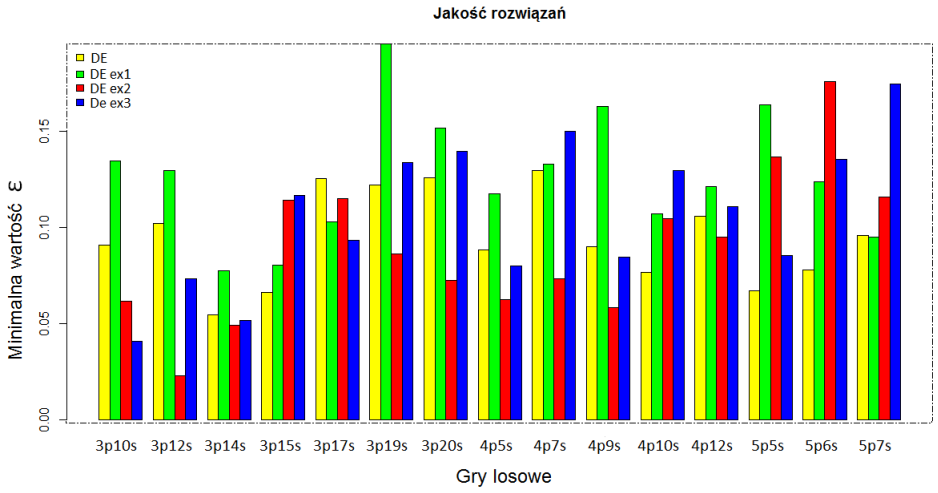
gra	DE ex_2					DE ex_3				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	2,41	2,47	2,44	2,44	0,02	2,23	2,44	2,27	2,26	0,05
3p12s	5,87	5,93	5,89	5,89	0,01	5,82	6,04	5,95	5,92	0,04
3p14s	12,8	12,99	12,84	12,82	0,05	12,68	13,35	12,93	12,91	0,18
3p15s	18,33	18,53	18,39	18,37	0,05	18,89	19,39	19,04	19,02	0,12
3p17s	25,12	25,65	25,33	25,35	0,11	24,34	25,94	25,23	25,69	0,29
3p19s	54,32	54,98	54,62	54,67	0,2	54,92	55,31	55,14	55,08	0,05
3p20s	157,58	159,74	156,78	159,09	0,81	158,26	162,3	160,29	160,04	1,33
4p5s	0,73	0,95	0,89	0,83	0,02	0,76	0,94	0,85	0,83	0,03
4p7s	4,01	4,05	4,03	4,03	0,01	4,12	4,45	4,24	4,22	0,18
4p9s	20,01	20,42	20,25	20,21	0,38	20,94	21,88	21,39	21,34	0,43
4p10s	39,66	40,32	40,04	40,07	0,19	39,52	40,25	39,98	39,86	0,34
4p12s	124,83	125,77	125,08	125,03	0,14	125,04	125,97	125,78	125,73	0,25
5p5s	28,07	29,14	28,75	28,64	0,21	28,12	28,64	28,45	28,41	0,12
5p6s	105,73	107,32	106,46	106,24	0,77	107,23	108,35	107,96	107,83	0,24
5p7s	202,34	203,56	202,87	202,53	0,53	202,46	205,61	203,87	203,66	0,87

Tabela 8.2: Dokładność generowanych rozwiązań dla algorytmu DE oraz algorytmu DE z wykluczeniem 1, 2, oraz 3 strategii (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	DE					DE ex_1				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,09072	0,31587	0,18228	0,16051	0,06519	0,1346	0,39328	0,19304	0,16577	0,04189
3p12s	0,10181	0,4599	0,19065	0,1506	0,10121	0,1293	0,28066	0,18086	0,17117	0,03508
3p14s	0,05451	0,37208	0,16363	0,14635	0,0875	0,07745	0,27503	0,18187	0,12283	0,03512
3p15s	**0,0662**	0,27404	0,14126	0,13695	0,05007	0,08046	0,39506	0,16187	0,13841	0,04033
3p17s	0,12516	0,39816	0,23183	0,22054	0,0784	**0,10281**	0,27399	0,16215	0,14889	0,03526
3p19s	0,12175	0,31045	0,20917	0,2153	0,05801	0,19515	0,2481	0,21021	0,20408	0,10418
3p20s	0,12574	0,40422	0,2233	0,20708	0,0761	0,15147	0,31717	0,22602	0,18806	0,09195
4p5s	0,08814	0,25733	0,1699	0,15683	0,05662	0,11719	0,3519	0,18273	0,14651	0,05448
4p7s	0,12963	0,27205	0,19852	0,20387	0,04449	0,13269	0,4412	0,19659	0,18709	0,03037
4p9s	0,08974	0,22165	0,15411	0,13328	0,04709	0,16258	0,46576	0,26276	0,25131	0,04744
4p10s	**0,07669**	0,30827	0,16636	0,15803	0,05668	0,10706	0,27437	0,18989	0,15395	0,06857
4p12s	0,10567	0,37779	0,18776	0,15093	0,08787	0,12098	0,31567	0,24251	0,23829	0,03186
5p5s	**0,06718**	0,31254	0,14361	0,14215	0,06326	0,16356	0,35732	0,24773	0,22674	0,06431
5p6s	**0,07791**	0,33121	0,16609	0,1589	0,06947	0,12367	0,28567	0,1754	0,16541	0,03467
5p7s	0,09552	0,41251	0,22049	0,22293	0,09006	**0,09471**	0,42371	0,19673	0,17698	0,08577

gra	DE ex_2					DE ex_3				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	**0,06151**	0,29908	0,16441	0,15324	0,04124	0,04095	0,20049	0,10085	0,09072	0,03898
3p12s	**0,02285**	0,25804	0,14871	0,12188	0,03074	0,07321	0,26297	0,14032	0,12267	0,04397
3p14s	**0,04923**	0,33081	0,16832	0,16884	0,07637	0,05166	0,26059	0,13824	0,12404	0,08242
3p15s	0,11421	0,32088	0,18013	0,16996	0,04215	0,11675	0,39355	0,21784	0,19533	0,08674
3p17s	0,11474	0,25194	0,16277	0,16344	0,07756	0,09331	0,33335	0,17469	0,16939	0,11457
3p19s	**0,08594**	0,42507	0,24134	0,23124	0,11208	0,13358	0,25728	0,19682	0,21429	0,09904
3p20s	**0,07239**	0,22355	0,18442	0,17456	0,07427	0,13925	0,32627	0,21255	0,20778	0,10265
4p5s	**0,06243**	0,29355	0,20607	0,12134	0,11154	0,07987	0,33277	0,1271	0,11035	0,08513
4p7s	**0,07326**	0,35924	0,16996	0,16098	0,07453	0,14978	0,26943	0,18262	0,17248	0,11248
4p9s	**0,05842**	0,25515	0,19672	0,15354	0,09571	0,08449	0,24458	0,16939	0,16759	0,09788
4p10s	0,10455	0,23769	0,16534	0,15631	0,06475	0,12953	0,28603	0,18628	0,21202	0,11751
4p12s	**0,09485**	0,30667	0,19423	0,16992	0,04182	0,1108	0,37369	0,20355	0,19861	0,09888
5p5s	0,13644	0,31562	0,21643	0,19462	0,09451	0,08532	0,25621	0,15763	0,16421	0,07461
5p6s	0,17565	0,28416	0,23145	0,21435	0,09737	0,13522	0,36224	0,19462	0,18467	0,11485
5p7s	0,11564	0,31573	0,25212	0,24254	0,09522	0,17432	0,47165	0,25561	0,23461	0,08741

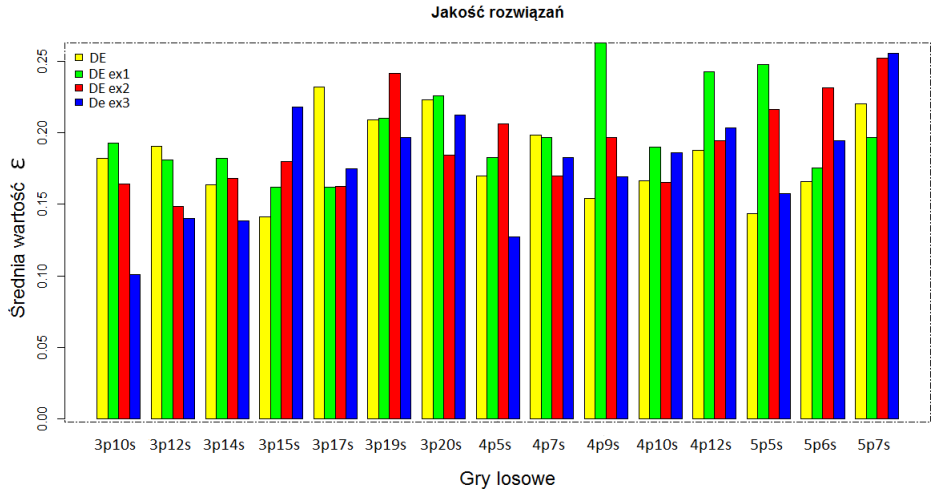
Na rys. 8.2 przedstawiono zestawienie wartości minimalnych ϵ dla algorytmu DE, a także algorytmu DE z wykluczeniem strategii. Warto zwrócić tutaj uwagę na dwie kwestie. Po pierwsze, przedstawione wyniki potwierdzają komentarze dotyczące tabeli 8.2, gdzie wyniki generowane przez algorytm DE nie są wyraźnie lepsze, niż w przypadku algorytmu DE z wykluczeniem strategii. Świadczy to o tym, iż ogólna funkcja oceny umożliwia traktowanie problemu zdefiniowanego przez I. Gilboę na równi z klasycznym problemem wyszukiwania równowag Nasha bez istotnego spadku jakości rozwiązań. Druga istotna kwestia to fakt, iż problem wykluczenia strategii zdaje się prowadzić do nieznacznie lepszych wyników, niż ma to miejsce dla problemu wykluczenia 1 oraz 3 strategii. Jednocześnie, to właśnie DE z wykluczeniem 2 strategii pozwoliło uzyskać wynik najbardziej zbliżony do optimum globalnego.



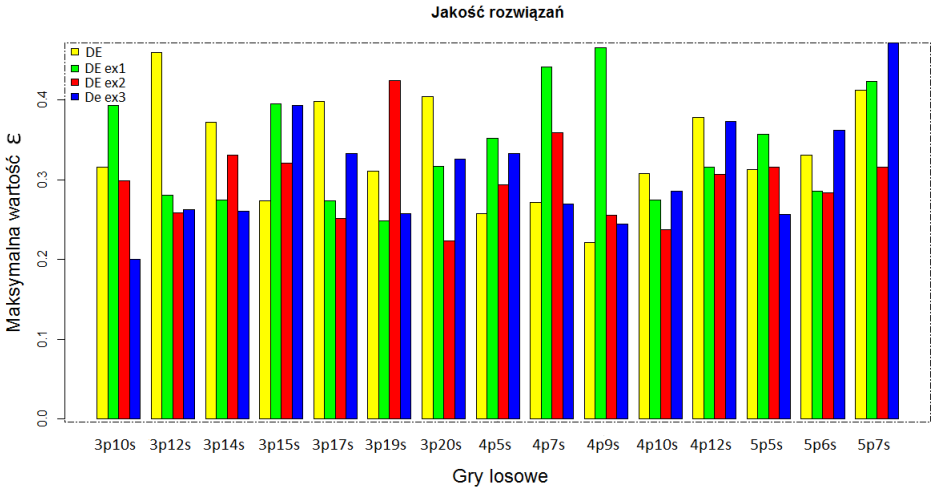
Rysunek 8.2: Dokładność uzyskanych rozwiązań dla algorytmów DE oraz DE z wykluczeniem strategii - wartości minimalne dla gier losowych

Rys. 8.3 dotyczy zostawienia średnich wartości ϵ dla DE oraz DE z wykluczeniem strategii. Wartości dla wszystkich przypadków są do siebie zbliżone, a jakość rozwiązań uzyskiwanych dla problemu wyszukiwania równowag Nasha z wykluczeniem strategii jest zbliżona do wartości ϵ uzyskiwanych przy klasycznym problemie wyszukiwania równowag Nasha. Podobne zestawienie dla wartości maksymalnych ϵ przedstawione zostały na rys. 8.4.

Podsumowując zagadnienie wyszukiwania równowag Nasha z wykluczeniem wybranych strategii można uznać, iż sposób rozwiązania tego problemu przy pomocy algorytmu DE jest zbliżony do klasycznego problemu wyszukiwania równowag Nasha. Takie uproszczenie możliwe było dzięki zastosowaniu pewnej ogólnej funkcji oceny.



Rysunek 8.3: Dokładność uzyskanych rozwiązań dla algorytmów DE oraz DE z wykluczeniem strategii - wartości średnie dla gier losowych



Rysunek 8.4: Dokładność uzyskanych rozwiązań dla algorytmów DE oraz DE z wykluczeniem strategii - wartości maksymalne dla gier losowych

8.3 Równowagi zawierające strategie o ustalonym prawdopodobieństwie

Bardzo istotnym problemem poruszanym przez I. Gilboę jest wyszukiwanie równowag Nasha z ustalonymi minimalnymi wartościami wyboru strategii aktywnych. Jest to również problem o złożoności NP. W poniższym podrozdziale przedstawione zostały wyniki dotyczące porównania algorytmu DE oraz algorytmu ADE dla wspomnianego problemu. Założone zostały 3 różne warianty zagadnienia:

- jedna ze strategii każdego z graczy powinna posiadać prawdopodobieństwo wyboru równe przynajmniej 0.3;
- dwie ze strategii każdego z graczy powinny posiadać prawdopodobieństwo wyboru równe przynajmniej 0.15;
- jw. ale dotyczy trzech strategii z prawdopodobieństwem wyboru równym przynajmniej 0.1.

Ponadto, podobnie jak w poprzednim rozdziale, także i tutaj do funkcji oceny wprowadzona została pewna funkcja kary zapewniająca, iż dla wskazanej arbitralnie strategii prawdopodobieństwo jej wyboru będzie wynosiło przynajmniej założoną wartość. Dla algorytmu ADE liczba strategii ustalonych wynosiła od 3 (dla pierwszego wariantu problemu, gdzie wybrana została jedna strategia o prawdopodobieństwie przynajmniej 0) do 5 strategii aktywnych. Najistotniejszym aspektem niniejszych badań jest wykazanie, iż problemy związane z wyszukiwaniem równowag Nasha określane jako problemy klasy NP są efektywnie rozwiązywane przez proponowane w rozprawie podejście, a ogólna funkcja oceny jest efektywna, także dla innych problemów związanych z wyszukiwaniem równowag Nasha. Ponadto, proponowany w rozprawie algorytm ADE może być w prosty sposób dostosowany do bardziej złożonych problemów.

Pierwsze zestawienie prezentowane w tabeli 8.3 dotyczy wartości ϵ dla algorytmu DE. Oznaczenia w tabeli dotyczą odpowiednio:

- s_1 - jedna strategia z minimalnym prawdopodobieństwem 0.3;
- s_2 - dwie strategie z prawdopodobieństwami minimalnymi 0.15;
- s_3 - trzy strategie z prawdopodobieństwami minimalnymi 0.1.

W tabeli wytłuszczone zostały wartości minimalne dla poszczególnych zagadnień. Warto zwrócić uwagę na dwa istotne aspekty. Po pierwsze, podejście, w którym trzy strategie każdego z graczy posiadają ustalone prawdopodobieństwo minimalne na poziomie 0.1 umożliwiło uzyskanie najniższych wartości ϵ. Ponadto,

pierwsza część tabeli, gdzie zaprezentowane zostały wyniki dotyczące zagadnienia z jedną strategią aktywną z minimalnym prawdopodobieństwem na poziomie 0.3 wskazują na największe wartości ϵ. Każdy z genów w genotypie osobnika losowany jest z przedziału $\langle 0, 1\rangle$. Wartości te wybierane są zgodnie z rozkładem jednostajnym, stąd w przypadku, gdy jeden z genów powinien mieć wartość większą niż 0.3, to aż w 30% przypadków wylosowana wartość będzie mniejsza od wspomnianej wartości granicznej. W takiej sytuacji dany osobnik podlega funkcji kary, która dyskwalifikuje dane rozwiązanie. Stąd oczywisty wniosek, iż poprawa jakości rozwiązania (a co za tym idzie obniżenia wartości ϵ) jest znacznie łatwiejsza w przypadku, gdy trzy strategie gracza powinny mieć wartość prawdopodobieństwa nie mniejszą niż 0.1.

Długość genotypu w przypadku algorytmu ADE jest istotnie mniejsza, niż miało to miejsce dla algorytmu DE. Stąd też długość genotypu w przypadku algorytmu DE w połączeniu z bardziej złożonym problemem uniemożliwiła znalezienie satysfakcjonującego rozwiązania dla największych gier. Świadczą o tym wykreślone pola w tabeli 8.3. Dla algorytmu ADE znalezienie rozwiązania możliwe było w każdym z przypadków 8.4. Oczywiście w przypadku omawianego problemu jasne jest, iż dla ograniczonego zbioru strategii aktywnych dla algorytmu ADE jakość rozwiązań w tym wypadku będzie gorsza niż dla algorytmu DE. Podobna sytuacja miała miejsce w klasycznym problemie wyszukiwania równowag Nasha. Wytłuszczone wartości minimalne w tabeli 8.4 również wskazują na fakt, iż 3 strategie aktywne z prawdopodobieństwem minimalnym równym 0.1 to w przypadku algorytmu ADE zagadnienie mniej złożone i umożliwiające w większości przypadków wygenerowanie rozwiązania z niższym ϵ niż ma to miejsce w pozostałych przypadkach. Tutaj jednak różnice te nie są tak widoczne, jak miało to miejsce w algorytmie DE.

Tabela 8.3: Dokładność generowanych rozwiązań dla algorytmu DE dla problemu wyszukiwania równowag Nasha z ustalonymi minimalnymi wartościami wyboru strategii aktywnych (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra		DE s1					DE s2			
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,12119	0,32343	0,21373	0,19132	0,0674	0,1058	0,37344	0,22969	0,22116	0,07584
3p12s	0,12245	0,34826	0,24639	0,26344	0,07052	**0,05224**	0,35402	0,16394	0,13572	0,07961
3p14s	0,12865	0,37065	0,2415	0,22736	0,07275	0,08594	0,42571	0,19786	0,18257	0,0748
3p15s	0,09154	0,36955	0,192	0,18765	0,07836	0,05418	0,28183	0,15977	0,15195	0,06505
3p17s	0,08545	0,3797	0,19528	0,17242	0,08727	**0,05753**	0,29014	0,16111	0,14641	0,07052
3p19s	0,07409	0,36647	0,18383	0,16809	0,08253	**0,02974**	0,29168	0,14143	0,13251	0,0669
3p20s	0	0	0	0	0	0	0	0	0	0
4p5s	0,08606	0,34866	0,2034	0,1863	0,08569	0,06684	0,2261	0,13966	0,13678	0,04257
4p7s	0,07144	0,38371	0,19873	0,18947	0,0785	0,09539	0,32403	0,16556	0,13664	0,06135
4p9s	**0,0534**	0,29229	0,15807	0,14963	0,06495	0,06027	0,34143	0,18814	0,17918	0,07885
4p10s	0	0	0	0	0	0,07111	0,26008	0,13701	0,11404	0,05601
4p12s	0	0	0	0	0	0	0	0	0	0
5p5s	0,07226	0,34812	0,19234	0,18242	0,08081	0,07487	0,27784	0,16319	0,1595	0,054
5p6s	0,09088	0,29785	0,17938	0,17562	0,05705	0,04454	0,28943	0,15874	0,15762	0,06851
5p7s	0	0	0	0	0	0	0	0	0	0

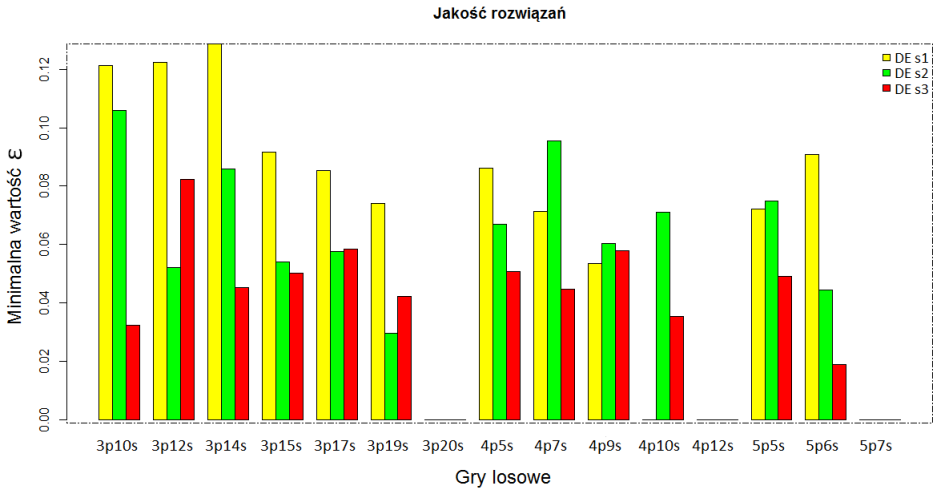
gra		DE s2			
	min	max	śred	med	std
3p10s	**0,03241**	0,22229	0,15128	0,16349	0,05441
3p12s	0,0824	0,28534	0,15262	0,1223	0,05945
3p14s	**0,04534**	0,35113	0,14894	0,14045	0,07625
3p15s	**0,05022**	0,23824	0,15202	0,15952	0,06121
3p17s	0,05841	0,30723	0,17896	0,18017	0,07372
3p19s	0,04225	0,28715	0,15995	0,16595	0,06812
3p20s	0	0	0	0	0
4p5s	**0,05075**	0,54517	0,15268	0,10852	0,11517
4p7s	**0,04464**	0,25852	0,1599	0,1679	0,07688
4p9s	0,05777	0,31697	0,16274	0,14848	0,06993
4p10s	**0,03536**	0,38595	0,12751	0,12328	0,08088
4p12s	0	0	0	0	0
5p5s	**0,04914**	0,34063	0,17317	0,17938	0,06747
5p6s	**0,01886**	0,36953	0,13445	0,117	0,08828
5p7s	0	0	0	0	0

Tabela 8.4: Dokładność generowanych rozwiązań dla algorytmu ADE dla problemu wyszukiwania równowag Nasha z ustalonymi minimalnymi wartościami wyboru strategii aktywnych (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra		ADE 3					ADE 4			
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,13712	0,38031	0,2169	0,20407	0,07745	0,09422	0,21147	0,14949	0,14576	0,04504
3p12s	0,22328	0,35863	0,30031	0,305	0,04403	0,23069	0,33907	0,27247	0,25374	0,04059
3p14s	0,15423	0,38603	0,29775	0,29485	0,08527	**0,13309**	0,37934	0,25384	0,23812	0,10132
3p15s	0,15019	0,3709	0,28481	0,30593	0,07945	0,15579	0,33863	0,24094	0,24603	0,05705
3p17s	0,20816	0,33945	0,28345	0,31363	0,05873	0,27731	0,38888	0,30169	0,28678	0,03948
3p19s	0,14555	0,37165	0,28466	0,28814	0,08117	**0,08521**	0,32369	0,1987	0,18218	0,10233
3p20s	0,17635	0,38356	0,27469	0,2584	0,08227	0,18272	0,36402	0,28252	0,28124	0,06178
4p5s	0,16221	0,36322	0,21995	0,20822	0,06838	**0,06832**	0,37808	0,12803	0,08359	0,11199
4p7s	0,18484	0,29776	0,24092	0,24585	0,03647	0,13908	0,2403	0,17355	0,17854	0,03528
4p9s	0,14437	0,30596	0,2115	0,20907	0,05502	**0,09171**	0,35266	0,20714	0,17471	0,09487
4p10s	0,17035	0,31031	0,24073	0,24589	0,06	0,12889	0,26547	0,20998	0,20743	0,04345
4p12s	0,18579	0,38028	0,25884	0,24951	0,07564	0,17344	0,35936	0,2221	0,19205	0,0702
5p5s	0,21426	0,38701	0,2992	0,28801	0,06406	**0,11179**	0,16401	0,13818	0,13871	0,01854
5p6s	0,12817	0,39419	0,27214	0,27483	0,09875	0,14343	0,2984	0,18953	0,16426	0,06328
5p7s	0,25629	0,39201	0,33509	0,35729	0,06054	**0,10733**	0,2434	0,15416	0,13292	0,05412

gra		ADE 5			
	min	max	śred	med	std
3p10s	**0,07918**	0,24497	0,16278	0,15796	0,0619
3p12s	**0,15863**	0,31246	0,24681	0,24592	0,05736
3p14s	0,22371	0,3756	0,27847	0,26343	0,0561
3p15s	**0,11221**	0,33762	0,16867	0,12693	0,08813
3p17s	**0,11649**	0,35284	0,26111	0,26245	0,08583
3p19s	0,10498	0,35757	0,25383	0,25694	0,08461
3p20s	**0,1327**	0,24249	0,20248	0,20971	0,03737
4p5s	0,08842	0,18808	0,14142	0,14013	0,03488
4p7s	**0,12415**	0,2741	0,21347	0,23201	0,05729
4p9s	0,11122	0,21716	0,15834	0,13831	0,04797
4p10s	**0,08355**	0,22601	0,1538	0,16657	0,05359
4p12s	**0,12288**	0,19123	0,14767	0,13905	0,02899
5p5s	0,15504	0,33385	0,20776	0,1947	0,0733
5p6s	**0,09151**	0,21229	0,15332	0,1698	0,05789
5p7s	0,13511	0,32097	0,20726	0,19749	0,07909

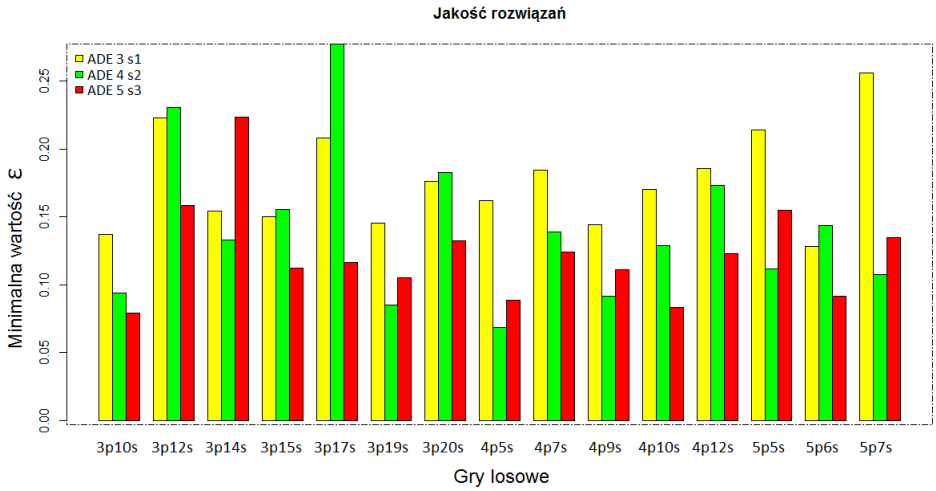
Na rys. 8.5 przedstawione zostały wartości minimalne ϵ w omawianym problemie dla poszczególnych gier. Dla największych gier nie było możliwe wygenerowanie satysfakcjonującego rozwiązania. Ponadto, na wykresie wyraźnie widać, iż dla wersji z 3 ustalonymi minimalnymi wartościami wyboru strategii wartości ϵ są wyraźnie niższe.



Rysunek 8.5: Minimalny DE

Podobne różnice dla różnych problemów zaobserwowane zostały w przypadku algorytmu ADE - rys. 8.6. Jak można się było spodziewać, wartości ϵ dla algorytmu ADE są nieco wyższe, niż miało to miejsce dla algorytmu DE. Wskazuje to wyraźnie, iż sama skuteczność przedstawionych algorytmów nie zmienia się pomimo wprowadzenia do problemu dodatkowych założeń.

Rozważania dotyczące wyników badań dla problemu wyszukiwania równowag Nasha, a także wyszukiwania równowag Nasha z dodatkowymi właściwościami przedstawione w dwóch powyższych rozdziałach wyraźnie wskazują, iż część problemów teorii gier może być traktowana jako problemy wielokryterialne. Oprócz tak ważnej dokładności uzyskiwanych rozwiązań oraz czasu działania algorytmu niemniej istotny jest także problem ograniczenia liczy strategii aktywnych dla pojedynczego gracza. Brak algorytmów dotyczących tego problemu w literaturze prowadzi do marginalizowania wspomnianego zagadnienia. W powyższych badaniach wskazane zostały słabe strony istniejących algorytmów oraz zaproponowana została metoda oparta na algorytmach metaheurystycznych umożliwiająca wyszukiwanie przybliżonych równowag Nasha. Jednocześnie wskazano, iż dla algorytmów SM oraz GNM bardzo poważnym problemem jest brak powtarzalności uzyskiwanych rozwiązań. Dla problemów o takiej samej złożoności istniejące metody nierzadko tylko w niektórych przypadkach umożliwiają wygenerowanie rozwiązań. Zaproponowane w rozprawie metody cechują się tak ważną w tym kontekście



Rysunek 8.6: Minimalny ADE

powtarzalnością.

Pokazano także, iż zaproponowane algorytmy mogą być w łatwy sposób dostosowane do znacznie bardziej złożonych problemów teorii gier i umożliwiają wyszukiwanie równowag Nasha z dodatkowymi właściwościami. Świadczy to o dużej elastyczności wspomnianych metod.

ROZDZIAŁ 9

Podsumowanie

Ewolucja różnicowa należy do heurystycznych algorytmów populacyjnych. Ze względu na adaptacyjny charakter mutacji w niej występujący, skuteczność algorytmu jest bardzo wysoka. Nierzadko jednak dla pewnych problemów konieczne jest wprowadzenie licznych modyfikacji podstawowej wersji metody. Poprzez dodawanie mechanizmów takich jak przeszukiwanie lokalne tworzone są złożone systemy hybrydowe dostosowane do jednego szczególnego problemu. Bardzo często samo dobranie odpowiednich parametrów dla operatorów genetycznych stanowi złożone zagadnienie. Pomimo licznych publikacji w tym zakresie, problem wciąż nie jest jednoznacznie rozwiązany, a dobór parametrów wydaje się zależeć od konkretnego zagadnienia.

Interesującą dziedziną, w której możliwe jest wykazanie skuteczności wspomnianego algorytmu jest teoria gier. Mnogość problemów występujących w tej dziedzinie implikowała konieczność ograniczenia do jednego wybranego zagadnienia, a mianowicie wyszukiwania równowag Nasha w grach n-osobowych. W analogiczny sposób definiowana jest ϵ-równowaga Nasha, przy czym ϵ oznacza tutaj pewne odchylenie od optimum. Jest to jeden z najistotniejszych problemów współczesnej informatyki.

Wyszukiwanie ϵ-równowag Nasha może zostać rozpatrywane analogicznie jako problem optymalizacji funkcji z parametrami rzeczywistymi, co było pierwotnym obszarem zastosowań ewolucji różnicowej. Istniejące algorytmy (w tym opisane w rozprawie algorytmy SM oraz GNM) świadczą o wyraźnej dysproporcji pomiędzy metodami dostępnymi dla gier 2-osobowych a gier n-osobowych. W drugim przypadku szczególnie odczuwalny jest brak metod umożliwiających generowanie rozwiązań przybliżonych dla dużych gier. Zaproponowane w pracy algorytmy (algorytm ewolucji różnicowej oraz jego adaptacyjna wersja) pozwalają zmniejszyć wartość ϵ, a co za tym idzie, obniżyć wartość błędu wynikającą ze stosowania przez graczy ϵ-równowag Nasha. Ponadto, zaproponowane w rozprawie podejścia umożliwiają generowanie powtarzalnych wyników (co nie jest możliwe w przy-

padku istniejących rozwiązań takich jak SM oraz GNM). W przypadku algorytmu adaptacyjnego jednym z istotnych problemów było ograniczenie liczby strategii aktywnych dostępnych dla pojedynczego gracza.

Teza pracy "Zaproponowane modyfikacje algorytmu ewolucji różnicowej poprawiają jakość rozwiązań w problemie wyszukiwania ϵ-równowag Nasha w grach o sumie niezerowej. Przekształcenie powyższego zagadnienia do problemu optymalizacji funkcji ciągłej pozwala na wprowadzenie ogólnej funkcji oceny umożliwiającej jego rozwiązywanie bez wyraźnego zwiększenia kosztu obliczeniowego algorytmu" została potwierdzona. W wyniku przeprowadzonych badań zrealizowane zostały następujące cele:

1. Przeanalizowane zostały istniejące wersje algorytmu ewolucji różnicowej (rozdziały 2 oraz 3). Szczególny nacisk położono na rozpoznanie najistotniejszego elementu algorytmu, czyli operatora mutacji. Wskazano również najpopularniejsze modyfikacje, a także przedstawiono szereg wariantów hybrydowych. Na tym etapie możliwe było wskazanie parametrów, których odpowiednie ustalenie nierzadko stanowi najtrudniejsze zadanie dla użytkownika. Parametr mutacji oraz wielkość populacji to elementy istotnie wpływające na zbieżność algorytmu do optimum, dlatego z punktu widzenia dalszych badań wprowadzenie adaptacyjnej wersji ich doboru stanowiło konieczność.

2. Kolejnym etapem było dokładne rozpoznanie problemów istniejących w teorii gier a następnie zawężenie obszaru badań. Teoria gier jest dziedziną bardzo rozległą i konieczne było wcześniejsze wskazanie najistotniejszych zagadnień (ze szczególnym uwzględnieniem typów równowag, w tym także równowagi Nasha - podrozdział 4.3 oraz 4.4). Konieczne było także wskazanie pewnego ogólnego zarysu typów gier. Zagadnienia te opisane zostały w podrozdziale 4.2.

3. Dokonana została analiza istniejących algorytmów do wyszukiwania równowag Nasha dla gier 2-osobowych oraz n-osobowych. Na tym etapie wyznaczone zostały dwie najskuteczniejsze metody stosowane przy wyznaczaniu równowag w grach n-osobowych. Podstawowe kryteria, którymi kierowano się w tym zagadnieniu dotyczyły po pierwsze tego, aby metody były jak najbardziej ogólne i umożliwiały rozwiązywanie różnych klas gier w postaci strategicznej. Dodatkowo zastosowane w nich podejścia powinny być w miarę możliwości odmienne tak, aby porównanie dotyczyło odmiennych mechanizmów generowania równowag.

4. W rozdziale 6 szczegółowo opisane zostały mechanizmy działania adaptacyjnej ewolucji różnicowej. Przybliżono także zaproponowaną w rozprawie selektywną mutację oraz ogólną funkcję oceny dla problemu wyszukiwania równowag Nasha.

5. Badania eksperymentalne wspomnianego algorytmu adaptacyjnego, jego podstawowej wersji, a także ich zestawienie z algorytmami SM oraz GNM przedstawione zostały w rozdziale 7. Podrozdział 7.1 poświęcono na opis metody generowania zbiorów testowych, a także określenia ich maksymalnej wielkości. Wybrane zostały trzy typy gier, co pozwoliło wskazać skuteczność proponowanej metody nie tylko dla najpopularniejszych gier losowych, ale także dla innych ich typów. Analiza dotyczyła wielu aspektów ze szczególnym uwzględnieniem jakości rozwiązań oraz liczby strategii aktywnych dla poszczególnych algorytmów. W pierwszej kolejności wskazana została pozorna przewaga istniejących algorytmów - na tym etapie przedstawione zostały przede wszystkim wyniki dotyczące czasu działania algorytmów. Kolejne zagadnienie dotyczyło liczby strategii aktywnych dostępnych dla każdego z graczy. Wykazano tutaj słabe strony istniejących algorytmów i brak możliwości manipulacji wspomnianą liczbą (co z punktu widzenia użytkownika jest bardzo pożądane). Wreszcie ostatnia część rozdziału dotyczyła badania jakości rozwiązań na podstawie wartości ϵ (podrozdział 7.4) a także statystycznego potwierdzenia wyników (podrozdział 7.6).

6. Rozprawa dotyczyła różnych problemów spotykanych w teorii gier, dlatego ostatni rozdział miał na celu wykazania skuteczności proponowanej metody także dla bardziej złożonych zagadnień związanych z punktami równowagi. W rozdziale 8.1 wskazano kilka przykładowych problemów, a także przedstawiono modyfikację ogólnej funkcji oceny umożliwiającą ich rozwiązanie. Wyniki dotyczące omawianych zagadnień przedstawione zostały w dalszej części rozdziału. Szczególnie istotne było porównanie jakości rozwiązań dla algorytmów ewolucji różnicowej oraz jej adaptacyjnej wersji.

W wyniku przedstawionych w pracy rozważań możliwe było wysnucie następujących wniosków:

- Adaptacja parametrów oraz dynamiczna zmiana wielkości populacji odciąża użytkownika od kłopotliwego problemu doboru odpowiednich wartości. Dodatkowo wprowadzenie λ modyfikacji umożliwia poprawę zbieżności w końcowej fazie działania algorytmu.

- Podejście oparte na selektywnej mutacji pozwala ograniczyć zakres zmian w genotypie osobnika, co jest szczególnie pożądane na etapie eksploatacji rozwiązań. Wspomniany operator wydaje się być skuteczny dla problemów nie tylko związanych z teorią gier, ale także dla zagadnień, w których w funkcji oceny zastosowana jest funkcja *min* lub *max* - gdzie wartość funkcji oceny wybierana jest ze zbioru danych.

- Wprowadzenie ogólnej funkcji oceny w problemach teorii gier pozwala w prosty sposób dostosować metodę do bardziej złożonych zagadnień. Obec-

nie można zaobserwować trend, w którym to metaheurystyki dostosowane są tylko i wyłącznie do jednego problemu, co w zasadzie neguje pierwotny obszar ich zastosowań. Poprzez ogólną funkcję oceny możliwa jest poprawa skuteczności ewolucji różnicowej bez istotnego zawężenia dziedziny, w jakiej może być stosowana.

- Algorytm ADE umożliwia generowanie powtarzalnych rozwiązań w problemie wyszukiwania równowag Nasha w grach n-osobowych. Dodatkowo, w przypadku bardziej złożonych zagadnień związanych z punktami równowagi, algorytm ADE umożliwia generowanie podobnych rozwiązań bez istotnego zwiększania nakładu obliczeniowego.

Oprócz wniosków związanych z algorytmem ewolucji różnicowej możliwe było także zaobserwowanie pewnych zależności związanych ściśle z teorią gier i wyszukiwaniem równowag Nasha:

- Podejście oparte na próbkowaniu rozwiązań pozwala na wyznaczenie zbioru rozwiązań dla każdego problemu. Wraz ze wzrostem wielkości gry jednocześnie zwiększa się liczba oczekiwanych równowag Nasha. W połączeniu z założeniem, iż każda gra posiada równowagę o wsparciu równym $\ln n$ losowy wybór strategii aktywnych w danej iteracji algorytmu zwiększa szansę na znalezienie rozwiązania.

- Wprowadzenie do gier n-osobowych algorytmu umożliwiającego generowanie rozwiązań przybliżonych umożliwiło istotne zwiększenie złożoności rozpatrywanych problemów. Szczególna reprezentacja genotypu osobnika, w której liczba graczy oraz liczba strategii traktowana jest równorzędnie również miała istotny wpływ na możliwość analizy bardziej złożonych problemów.

- Ograniczenie zbioru strategii aktywnych dla poszczególnych graczy jest bardzo istotne z punktu psychologicznego. Gracz operujący niewielką liczbą strategii może znacznie łatwiej podjąć decyzję, niż w przypadku, gdy liczba strategii aktywnych pokrywa cały zbiór strategii dostępnych dla pojedynczego gracza. Rozwiązanie zaproponowane w rozprawie pozwala sterować wielkością tego zbioru dając użytkownikowi więcej możliwości.

Analiza wyników przeprowadzonych badań eksperymentalnych pozwoliła wysnuć szereg wniosków. Stanowi jednocześnie dobry punkt wyjścia do dalszych badań nad zagadnieniem generowania równowag Nasha w grach n-osobowych. Pytaniem otwartym pozostaje maksymalny rozmiar gier, dla których równowagi Nasha mogą być efektywnie wyznaczane przez adaptacyjny algorytm ewolucji różnicowej. Ponadto, istotnym zagadnieniem jest także wyznaczenie zbioru strategii

aktywnych należących do rozwiązania. Zagadnienie to związane jest ściśle z optymalizacją kombinatoryczną i dobre efekty powinno przynieść w tym wypadku zastosowanie algorytmów mrowiskowych lub symulowanego wyżarzania.

Zagadnienia związane z algorytmem ewolucji różnicowej powinny być rozwijane w kierunku adaptacji parametrów, co prowadzi do ograniczenia ingerencji użytkownika w sam algorytm. Jednocześnie duża skuteczność opisanej metody pozwala sądzić, iż inne metaheurystyki mogą umożliwiać generowanie satysfakcjonujących rezultatów w zbliżonych problemach. Potwierdzenie tego mogłoby skutkować próbą budowy hiperheurystyki stosowanej w problemach teorii gier.

Bibliografia

[1] H. A. Abbass, R. Sarker. The pareto differential evolution algorithm. International Journal on Artificial Intelligence Tools, 2002, s. 531–552.

[2] T. Abbott, D. Kane, P. Valiant. Finding nash equilibria in 0-1 bimatrix games is as hard as finding equilibria in rational bimatrix games. 2005.

[3] D. Agonafer et al. A simulation-based multi-objective design optimization of electronic packages under thermal cycling and bending. Microelectronics Reliability, 2004, wolumen 44, s. 1977–1983.

[4] A. Al-Ani et al. A combined ant colony and differential evolution feature selection algorithm. LNCS, 2008, wolumen 5217/2008, s. 1–12.

[5] J. Alves-Foss et al. Quantum genetic algorithms. Proceedings of the Genetic and Evolutionary Computation Conference GECCO, 2000, s. 373.

[6] K. J. Arrow, G. Debreu. Existence of an equilibrium for a competitive economy. Econometrica, 1954, wolumen 22, s. 265–290.

[7] J. Aubin. *Mathematical Methods of Game and Economic Theory.* North-Holland Publ. CO. 1979.

[8] R. Aumann. Subjectivity and correlation in randomized strategies. Journal of Mathematical Economics, 1974, wolumen 1(1), s. 67–96.

[9] B. V. Babu, M. L. Jehan. Differential evolution for multi-objective optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, 2003, wolumen 4, s. 2696–2703.

[10] I. Barany, S. Vempala, A. Vetta. Nash equilibria in random games. Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005, wolumen 1, s. 123–131.

[11] M. Batouche, H. Talbi. Hybrid particle swarm with differential evolution for multimodal image registration. In Proceedings of the IEEE International Conference on Industrial Technology, 2004, wolumen 3, s. 1567–1573.

[12] E. Borel. *Sur les jeux ou interviennent le hasard et l'habilité des joueurs, the English translation [Elements of the Theory of Probability].* Prentice-Hall 1924.

[13] U. Boryczka, P. Juszczuk. Comparative study of the differential evolution and the approximation algorithm for computing the optimal mixed strategies in zero-sum games. Computational Collective Intelligence. Technologies and Applications Lecture Notes in Computer Science, 2010, wolumen 6421, s. 363–372.

[14] U. Boryczka, P. Juszczuk. Using differential evolution to find optimal mixed strategies in two players matrix games. 16th International Conference on Soft Computing - MENDEL, 2010, s. 535–541.

[15] U. Boryczka, P. Juszczuk. A new evolutionary method for generating nash equilibria in bimatrix games with known support. Central European Journal of Computer Science, 2012.

[16] U. Boryczka, P. Juszczuk. Solving the sudoku with the differential evolution. Zeszyty Naukowe Politechniki Białostockiej, 2012.

[17] U. Boryczka, P. Juszczuk, L. Kłosowicz. A comparative study of various strategies in differential evolution. Evolutionary Computing and Global Optimization, KAEiOG'09, 2009, s. 19–26.

[18] B. Boskowic, J. Brest, S. Greiner. Selfadapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 2006, wolumen 10(6), s. 646–657.

[19] B. Boskowic et al. Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computation: a Fusion of Foundations, Methodologies and Applications, 2007, wolumen 11(7), s. 617–629.

[20] H. Bosse, J. Byrka, E. Markakis. New algorithms for approximate nash equilibria in bimatrix games. Journal Theoretical Computer Science, 2010, wolumen 411, s. 164–173.

[21] J. Brest, M. S. Maucec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. Special Issue on scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, 2011, wolumen 15, s. 2157–2174.

[22] C. S. Chang, D. Du. Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system. IEE Proceedings of Electric Power Applications, 2000, wolumen 147, s. 206–212.

[23] H. C. Chang, T. T. Chang. Application of differential evolution to passive shunt harmonic filter planning. In Proceedings of the Eight International Conference on Harmonics and Quality of Power, 1999, s. 149–153.

[24] X. Chen, X. Deng. 3 - nash is ppad - complete. Electronic Colloquium on Computational Complexity, 2005.

[25] J. P. Chiou, F. S. Wang. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. In IEEE World Congress on Computational Intelligence, 1998, s. 627–632.

[26] B. Codenotti, M. Pagan, S. De Rossi. An experimental analysis of lemke-howson algorithm. The Computing Research Repository, 2008, wolumen 811.

[27] B. Codenotti, D. Štefankovic. On the computational complexity of nash equilibria for (0, 1) bimatrix games. Journal Information Processing Letters, 2005, wolumen 94, s. 145–150.

[28] C. A. Coello Coello, E. Mezura-Montes, J. Velázquez-Reyes. A comparative study of differential evolution variants for global optimization. Proceedings of the 8th annual conference on Genetic and evolutionary computation GECCO, 2006, s. 485–492.

[29] V. Conitzer, A. Gilpin, T. Sandholm. Mixed-integer programming methods for finding nash equilibria. Proceedings of the 20th national conference on Artificial intelligence, 2005, wolumen 2, s. 495–501.

[30] R. Cressman. *Evolutionary Dynamics and Extensive Form Games*. MIT Press 2003.

[31] R. Dasilva et al. Using game theory to analyze wireless ad hoc networks. IEEE Communications Surveys & Tutorials, 2005, wolumen 7, s. 46–56.

[32] C. Daskalakis, P. Goldberg. The complexity of computing a nash equilibrium. ACM Press, 2006, s. 71–78.

[33] C. Daskalakis, A. Mehta, Ch. Papadimitriou. A note on approximate nash equilibria. Proceedings of the 2nd Workshop on Internet and Network Economics, 2006, s. 297–306.

[34] C. Daskalakis, A. Mehta, Ch. Papadimitriou. Progress in approximate nash equilibria. 8th ACM conference on Electronic commerce, 2007, s. 355 – 358.

[35] F. H. de Brito et al. Hawk-dove tournament: A new selection technique for genetic algorithms based on evolutionary game theory. Proceedings of the 2009 International Conference on Genetic and Evolutionary Methods, 2009, s. 54–58.

[36] Ch. Deng, Y. Y. Deng, B. Zhao. A novel binary differential evolution without scale factor f. Advanced Computational Intelligence (IWACI), 2010 Third International Workshop on, 2010, s. 250 – 253.

[37] X. Deng, S. Teng, Ch. Xi. Computing nash equilibria: Approximation and smoothed complexity. 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, s. 603–612.

[38] J. Dickhaut, T. Kaplan. A program for finding nash equilibria. The Mathematica Journal, 1991, wolumen 1(4), s. 87–93.

[39] D. Dolinar, M. Milanovic, O. Tezak. Snubber design approach for dc-dc converter based on differential evolution method. Proceedings of the Eight IEEE International Workshop on Advanced Motion Control, 2004, s. 87–91.

[40] M. Dorigo. Optimization, learning and natural algorithms. Phd Thesis, 1992.

[41] D. Easley, J. Kleinberg. *Reasoning about a Highly Connected World*. Cambridge University Press 2010.

[42] R. Eberhart, J. Kennedy. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 1995, wolumen IV, s. 1942–1948.

[43] R. C. Eberhart, J. Kennedy. A discrete binary version of the particle swarm algorithm. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 1997, s. 4104–4109.

[44] A. P. Engelbrecht. *Computational intelligence - An introduction*. John Wiley and Sons Ltd 2007.

[45] A. P. Engelbrecht, N. Franken, G. Pampara. Binary differential evolution evolutionary computation. IEEE Congress on CEC 2006, 2006, s. 1873 – 1879.

[46] A.P. Engelbrecht, M. G. H. Omran, A. Salman. Differential evolution methods for unsupervised image classification. Proceedings of the IEEE Congress on Evolutionary Computation, 2005, wolumen 2, s. 966–973.

[47] M.G. Epitropakis. Finding multiple global optima exploiting differential evolution's niching capability. IEEE Symposium on Differential Evolution (SDE), 2011, s. 1 – 8.

[48] M. Fei et al. A modified binary differential evolution algorithm. Lecture Notes in Computer Science, 2010, s. 49–57.

[49] L. Fogel, A. J. Owens, M. J. Walsh. Artificial intelligence through simulated evolution. John Wiley & Sons, 1966.

[50] D. P. Foster, V. V. Rakesh. Calibrated learning and correlated equilibrium. Games and Economic Behaviour, 1997, wolumen 21, s. 40–55.

[51] W. Froelich, P. Juszczuk. Learning fuzzy cognitive maps using a differential evolution algorithm. Polish Journal of Environmental Studies, 2009, wolumen 18, numer 3B, s. 108–112.

[52] D. Fudenberg, J. Tirole. *Game Theory.* Cambridge MA: MIT Press 1991.

[53] Jr. C. D. Gelatt, S. Kirkpatrick, M. P. Vecchi. Optimization by simulated annealing. Science, 1983, s. 671–680.

[54] I. Gilboa. Nash and correlated equilibria: Some complexity considerations. Games and Economic Behavior, 1989, s. 80–93.

[55] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Professional 1989.

[56] P. W. Goldberg, Ch. Papadimitriou. Reducibility among equilibrium problems. Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, 2006, s. 61 – 70.

[57] S. Govindan, R. Wilson. Computing nash equilibria by iterated polymatrix approximation. Journal of Economic Dynamics and Control, 2003, wolumen 27.

[58] S. Govindan, R. Wilson. A global newton method to compute nash equilibria. Journal of Economic Theory, 2003, wolumen 110, s. 65–86.

[59] R. J. Grave, A. C. Sandersons. Modeling and convergence analysis of a continuous multi-objective differential evolution algorithm. Proceedings of IEEE Congress on Evolutionary Computaton, 2005, s. 228–235.

[60] M. Gravel et al. Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem. European Journal of Operational Research, 2007, wolumen 178, s. 634–638.

[61] J. M. Graves, A. C. Sanderson, F. Xue. Pareto-based multi-objective differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation, 2003, wolumen 2, s. 862–869.

[62] M. Griss, R. Letsinger. Games at work agent-mediated e-commerce simulation. Raport instytutowy, HP Laboratories Technical Report, 2000.

[63] D. Grosu, J. Widger. Computing equilibria in bimatrix games by parallel support enumeration. 7th International Symposium on Parallel and Distributed Computing, 2008, wolumen 1, s. 250–256.

[64] P. Hammerstein, P. Selten. *Handbook of game theory*. University of Bonn 1994.

[65] H. Hao et al. Discrete differential evolution algorithm for the job shop scheduling problem. Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, 2009, s. 879–882.

[66] J. C. Harsanyi. The tracing procedure: a bayesian approach to defining a solution for n-person noncooperative games. International Journal of Game Theory, 1975, wolumen 4, s. 61–94.

[67] A. Haurie, J. Krawczyk. *An Introduction to Dynamic Games*. 2000.

[68] R. He et al. Three eit approaches for static imaging of head. Engineering in Medicine and Biology Society, 2004, wolumen 1, s. 578–581.

[69] T. Hendtlass. A combined swarm differential evolution algorithm for optimization problems. In Proceedings of the Fourteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 2001, wolumen 2070, s. 11–18.

[70] P. J. Herings, J. A. P. Peters. A differentiable homotopy to compute nash equilibria of n-person games. Economic Theory, 2001, wolumen 18(1), s. 159–185.

[71] D. R. Hofstadter. *Basic Books: Dilemmas for Superrational Thinkers, Leading Up to a Luring Lottery*. 1983.

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

[73] H. J. Huang, F. S. Wang. Fuzzy decision-making design of chemical plant using mixed-integer hybrid differential evolution. Computers & Chemical Engineering, 2002, wolumen 26, s. 1649–1660.

[74] Q. A. K. Huang, V. L. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation, 2009, wolumen 13, s. 398–417.

[75] A. J. J. Ialman, I. M. Ioup. A new variable dimension simplicial algorithm to find equilibria on the product space of unit simplices. Mathematical Programming, 1987, wolumen 37, s. 319–355.

[76] A. A. Jatoth, R. K. Rajasekhar. Hybrid differential artificial bee colony algorithm. Journal of Computational and Theoretical Nanoscience, 2012, wolumen 9(2), s. 249–257.

[77] Y. Y. Jing. A differential evolution with simulated annealing updating method. 2006 International Conference on Machine Learning and Cybernetics, 2006, s. 2103 – 2106.

[78] A. E. Kanlikilicer, A. Keles, U. A. Sima. Experimental analysis of binary differential evolution in dynamic environments. Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, 2007, s. 2509–2514.

[79] D. Karaboga. An idea based on honey bee swarm for numerical numerical optimization. Technical Report-TR06, 2005.

[80] H. Kazemipoor, P. Shahnazari-Shahrezaei, R. Tavakkoli-Moghaddam. Differential evolution and simulated annealing algorithms for a multi-skilled project scheduling problem. American Journal of Scientific Research, 2011, wolumen 33, s. 136–146.

[81] J. Kiefer, J. Wolfowitz. Stochastic estimation of the maximum of a regression function. Annals of Mathematical Statistics, 1952, wolumen 3, s. 462–466.

[82] L. Kockesen. *Game Theory Lecture Notes*. MIT 2008.

[83] S. C. Kontogiannis, P. N. Panagopoulou. Polynomial algorithms for approximating nash equilibria of bimatrix games. Journal Theoretical Computer Science, 2009, wolumen 410, s. 1599–1606.

[84] S. C. Kontogiannis, P. G. Spirakis. Well supported approximate equilibria in bimatrix games: A graph theoretic approach. Mathematical Foundations of Computer Science, 2007, wolumen 4708, s. 596–608.

[85] S. Krink, T. Paterlini. High performance clustering with differential evolution. Proceedings of the IEEE Congress on Evolutionary Computation, 2004, wolumen 2, s. 2004–2011.

[86] A. Kyprianou, M. Panet, K. Worden. Identification of hysteretic systems using the differential evolution algorithm. Journal of Sound and Vibration, 2001, wolumen 248, s. 289–314.

[87] J. Lampinen, S. Kukkonen. An empirical study of control parameters for generalized differential evolution. Raport instytutowy, KanGAL Report Number 2005014, 2005.

[88] J. Lampinen, I. Zelinka. On stagnation of the differential evolution algorithm. International Conference on Soft Computing, 2000, s. 76–83.

[89] C. E. Lemke, J. T. Howson. Equilibrium points of bimatrix games. SIAM Journal on Applied Mathematics, 1964, wolumen 12, s. 413–423.

[90] K. Leyton-Brown, Y. Shoham. *Essentials of game theory: a concise, multidisciplinary introduction*. Morgan and Claypool 2008.

[91] R. Lipton, E. Markakis, A. Mehta. Playing large games using simple strategies. 2003, s. 36–41.

[92] R. Lipton, N. Young. Simple strategies for large zero-sum games with applications to complexity theory. Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, 1994, s. 734 – 740.

[93] O. Mangasarian. Equilibrium points in bimatrix games. Journal of the Society for Industrial and Applied Mathematics, 1964, wolumen 12, s. 778–780.

[94] D. Manimegalai, S. Selvi. Scheduling jobs on computational grid using differential evolution algorithm. Proceedings of the 12th international conference on Networking, VLSI and signal processing, 2010, s. 118–123.

[95] R. D. McKelvey, A. M. McLennan, T. L. Turocy. Gambit: Software tools for game theory. Version 0.2006.01.20, 2010.

[96] N. Megiddo, Ch. Papadimitriou. On total functions, existence theorems, and computational complexity. Theoretical Computer Science, 1991, wolumen 81, s. 317–324.

[97] N. Metropolis et al. Equations of state calculations by fast computing machines. Journal of Chemical Physics, 1953, s. 1087–1092.

[98] Z. Michalewicz. *Algorytmy genetyczne + struktury danych = programy ewolucyjne.* Wydawnictwa Naukowo-Techniczne 1992.

[99] I. Milchtaich. Computation of completely mixed equilibrium payoffs in bimatrix games. International Game Theory Review, 2006, wolumen 8, s. 483–487.

[100] S. Monro, H. Robbins. A stochastic approximation method. Annals of Mathematical Statistics, 1951, wolumen 3, s. 400–407.

[101] O. Morgenstern, J. von Neumann. *Theory of Games and Economic Behavior.* Princeton University Press 1944.

[102] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. Technical Report C3P 826, 1989.

[103] R. B. Myerson. *Game Theory: Analysis of Conflict.* Cambridge, MA: Harvard University Press 1991.

[104] J. Nash. Noncooperative games. Annals of Mathematics, 1951, wolumen 54, s. 286–295.

[105] T. Nguyen, E. Tardos. Approximate pure nash equilibria via lovasz local lemma. Proceedings of the 5th International Workshop on Internet and Network Economics, 2009, s. 160–171.

[106] A. Nikolakopoulos, H. Sarimveis. A line up evolutionary algorithm for solving nonlinear constrained optimization problems. Computers & Operations Research, 2005, wolumen 32, s. 1499–1514.

[107] E. Nudelman, R. Porter, Y. Shoham. Simple search methods for finding a nash equilibrium. AAAI-04, 2004, s. 664–669.

[108] P. Ordeshook. *Game theory and political theory.* Cambridge University Press 1986.

[109] G. Owen. *Teoria gier.* PWN 1975.

[110] M. Pant, T. K. Sharma. Differential operators embedded artificial bee colony algorithm. International Journal of Applied Evolutionary Computation (IJAEC), 2011, wolumen 2, s. 1–14.

[111] Ch. Papadimitriou. Algorithms, games and the internet. Proceedings of the thirty-third annual ACM symposium on Theory of computing, 2001, s. 749–753.

[112] Christos Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. Journal of Computer and System Sciences - Special issue: 31st IEEE conference on foundations of computer science, 1994, s. 498–532.

[113] T. Płatkowski. *Wstęp z teorii gier*. Uniwersytet Warszawski 2012.

[114] D. G. Pearce. Rationalizable strategic behavior and the problem of perfection. Econometrica, 1984, wolumen 52, s. 1029–1050.

[115] S. Pesko. Differential evolution for small tsps with constraints. International Scientific Conference „Challenges in Transport and Communication“, 2006.

[116] M. J. Powell. The theory of radial basis function approximation. Advances in Numerical Analysis, 1990, wolumen 2.

[117] K. Price, R. Storn. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, wolumen 11, s. 341 – 359.

[118] A. K. Qin, P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. Proceedings of IEEE Congress on Evolutionary Computation, 2005, wolumen 2, s. 1785–1791.

[119] A. Rubinstein. *Modelling Bounded Rationality*. Mit Press 1998.

[120] A. C. Sanderson, J. Zhang. *Adaptive Differential Evolution - A Robust Approach to Multimodal Problem Optimization*. Springer 2009.

[121] J.G Sauer. Discrete differential evolution with local search to solve the traveling salesman problem: Fundamentals and case studies. 7th IEEE International Conference on Cybernetic Intelligent Systems, 2008. CIS 2008., 2008, s. 1–6.

[122] H. E. Scarf. The approximation of fixed points of a continuous mapping. SIAM Journal of Applied Mathematics, 1967, wolumen 15, s. 1328–1343.

[123] S.H. Schanuel, L.K. Simon, W.R. Zame. The algebraic geometry of games and the tracing procedure. Game equilibrium models II, methods, morals and markets, 1991, wolumen 1, s. 9–43.

[124] H. P. Schwefel. Evolutionsstrategie und numerische optimierung. PhD thesis, 1975.

[125] L. S. Shapley. A note on the lemke-howson algorithm. Mathematical Programming Study 1: Pivoting and Extensions, 1974, s. 175–189.

[126] M. Sion. On general minimax theorems. Pacific Journal of Mathematics, 1958, wolumen 8, s. 171–176.

[127] S. Smale. A convergent process of price adjustment and global newton methods. Journal of Mathematical Economics, 1976, wolumen 3, s. 107–120.

[128] S. F. Smith. A learning system based on genetic adaptive algorithms. PhD Thesis, 1980.

[129] R. Storn. Differential evolution design of an iir-filter. Proceedings of IEEE International Conference on Evolutionary Computation, 1996, s. 268–273.

[130] R. Storn. On the usage of differential evolution for function optimization. Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96, 1996, s. 519–523.

[131] A. Sureka, P. R. Wurman. Using tabu best-response search to find pure strategy nash equilibria in normal form games. Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, 2005, s. 1023–1029.

[132] K. Tang, Z. Yang, X. Yao. Self-adaptive differential evolution with neighborhood search. Proceedings of IEEE Congress on Evolutionary Computation, 2008, s. 1110 – 1116.

[133] M. Tennenholtz. *Game Theory and Artificial Intelligence.* Springer Berlin/Heidelberg 2002.

[134] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. Soft Computation: a Fusion of Foundations, Methodologies and Applications, 2006, wolumen 10, s. 673–686.

[135] T. Tyszka. *Konflikty i Strategie.* Wydawnictwo Naukowo-Techniczne 1978.

[136] Feoktistov V. *Differential Evolution - In Search of Solutions.* Springer Science+Business Media, LLC 2006.

[137] L. van Der Heyden, G. van der Laan, A. J. J. Talman. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. Mathematics of Operations Research, 1987, s. 377–397.

[138] G. van der Laan, A. J. J. Talman. A restart algorithm for computing fixed points without an extra dimension. Mathematical Programming, 1979, wolumen 20, s. 33–48.

[139] V. V. Vazirani. *Algorytmy aproksymacyjne.* WNT 2005.

[140] J. von Neumann. Zur theorie der gesellschaftsspiele. Mathematische Annalen - Contributions to the Theory of Games, 1928, wolumen 4, s. 13–42.

[141] A. Wald. Generalization of a theorem by von neumann concerning zero-sum two-person games. Annals of Mathematics, 1945, wolumen 46, s. 281–286.

[142] J. D. Williams. *The Compleat Strategyst: Being a Primer on the Theory of Games of Strategy.* McGraw-Hill , New York 1966.

[143] R. Wilson. Computing equilibria of n-person games. Journal on Applied Mathematics, 1971, wolumen 21, s. 80–87.

[144] Z. Xiangyin. Deaco: Hybrid ant colony optimization with differential evolution. Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)., 2008, s. 921 – 927.

[145] X-F. Xie, W-J. Zhang. Depso: Hybrid particle swarm with differential evolution operator. In Proceedings of the IEEE International Conference on System, Man, and Cybernetics, 2004, wolumen 4, s. 3816–3821.

[146] F. Xue. *Multi-Objective Differential Evolution: Theory and Applications*. Rensselaer Polytechnic Institute 2004.

[147] X. S. Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press 2008.

[148] X. S. Yang. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization, 2010, s. 65–74.

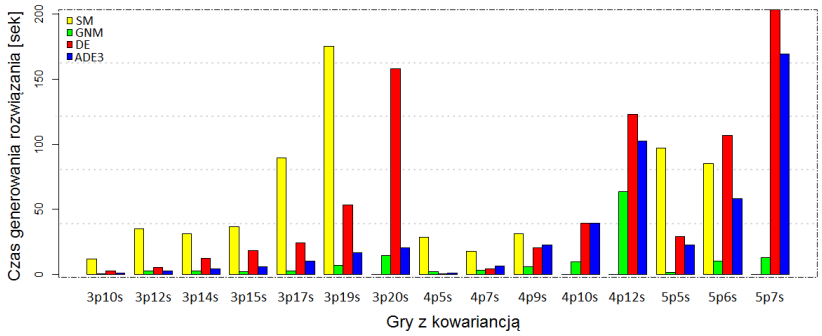
[149] D. Zaharie. Critical values for the control parameters of differential evolution algorithms. International Mendel Conference on Soft Computing, 2002.

[150] D. Zaharie. Control of population diversity and adaptation in differential evolution algorithms. 9th International Conference on Soft Computing, MENDEL, 2003, s. 41–46.

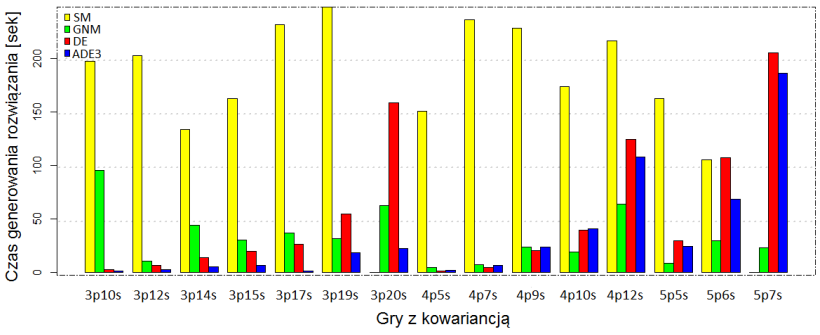
[151] D. Zaharie. Influence of crossover on the behavior of differential evolution algorithms. Journal of Applied Soft Computing, 2009, wolumen 9, s. 1126–1138.

DODATEK A

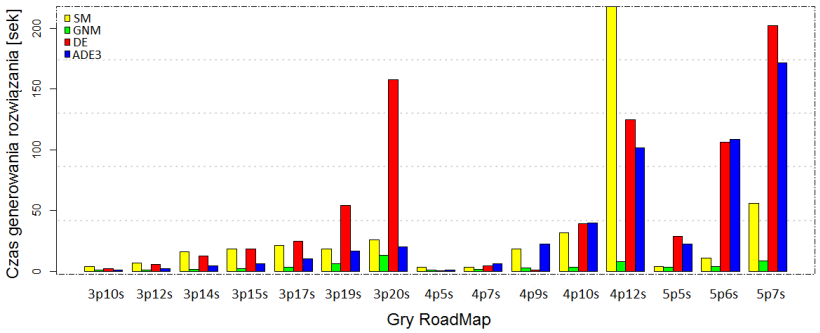
Szczegółowe wyniki dla wybranych problemów wyszukiwania równowag Nasha



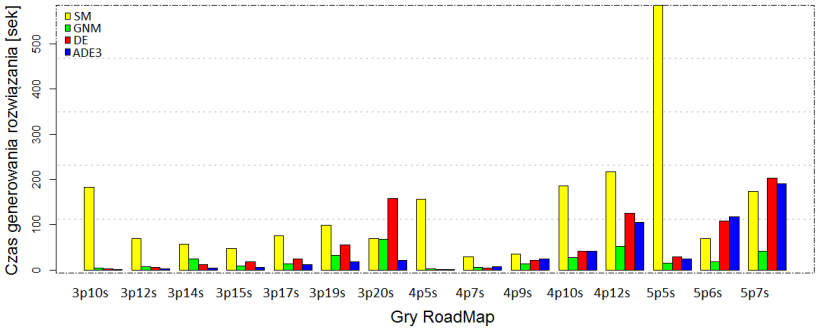
Rysunek A.1: Minimalny czas generowania rozwiązania - gry z kowariancją



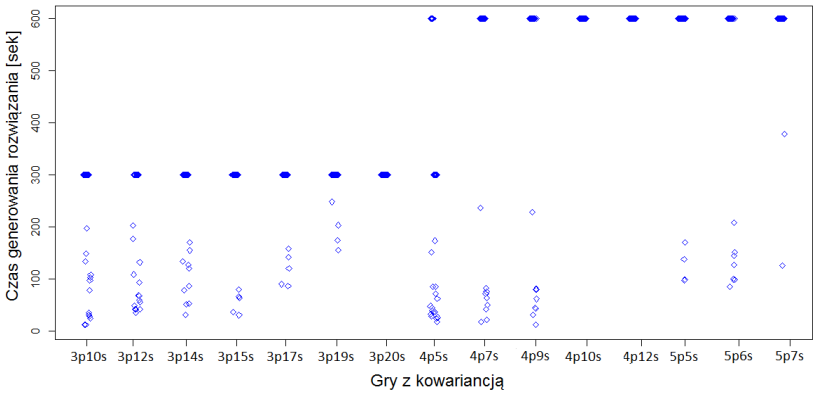
Rysunek A.2: Maksymalny czas generowania rozwiązania - gry z kowariancją



Rysunek A.3: Minimalny czas generowania rozwiązania - gry RoadMap



Rysunek A.4: Maksymalny czas generowania rozwiązania - gry RoadMap



Rysunek A.5: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm SM

Tabela A.1: Czas działania algorytmu (znalezienie pierwszego rozwiazania) w sekundach - gry z kowariancją. Pierwsza część tabeli to wyniki dla algorytmów SM oraz GNM, z kolei druga część to proponowane w rozprawie algorytmy DE oraz ADE (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

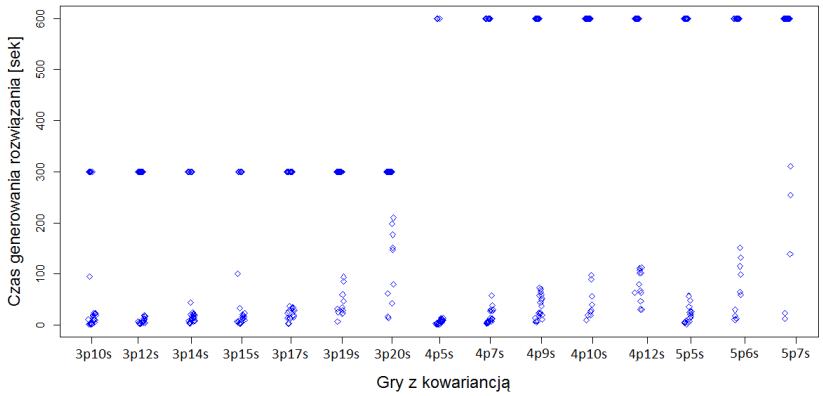
gra	SM					GNM				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	12,01	197,04	100,79	133,86	84,2	0,54	95,26	12,6	2,28	31,01
3p12s	34,77	202,34	93,41	48,18	70,69	2,4	10,7	5,02	4,75	2,88
3p14s	31,29	133,31	73,63	64,96	44,25	2,6	44,22	11,9	6,32	14,25
3p15s	36,44	162,54	110,6	121,71	57,14	2,28	30,36	8,44	5,76	8,59
3p17s	89,63	231,46	168,08	183,16	72,1	2,39	36,46	16,34	13,39	11,63
3p19s	175,36	247,72	220,64	220,64	38,29	7,1	31,22	21,26	25,46	12,59
3p20s	-	-	-	-	-	14,33	62,06	30,89	16,28	27,01
4p5s	28,36	150,71	58,19	42,07	41,32	1,81	4,35	2,754	2,72	0,84
4p7s	17,46	236,23	104,4	91,34	81,95	3,27	6,89	5,06	4,35	1,53
4p9s	31,31	228,2	147,73	183,68	103,25	5,67	23,36	11,77	8,96	6,63
4p10s	-	-	-	-	-	9,5	18,77	14,13	14,13	6,55
4p12s	-	-	-	-	-	63,43	63,43	63,43	63,43	7,48
5p5s	97,37	162,68	115,4	146,67	68,49	1,58	8,72	5,05	5,31	2,36
5p6s	85,11	105,35	95,23	95,23	14,31	10,24	29,58	17,48	15,06	8,62
5p7s	-	-	-	-	-	12,87	23,04	17,95	17,95	7,19

gra	DE					ADE				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	2,51	2,8	2,58	2,38	0,01	1,2	1,4	1,26	1,25	0,06
3p12s	5,32	6,41	6,27	5,87	0,01	2,38	2,71	2,5	2,5	0,11
3p14s	12,35	13,47	12,86	12,86	0,01	4,43	5,03	4,67	4,62	0,21
3p15s	18,15	19,57	18,39	18,37	0,09	5,99	6,6	6,2	6,18	0,17
3p17s	24,44	25,91	24,89	24,88	0,02	10,3	11,4	10,76	10,75	0,41
3p19s	53,13	54,51	53,43	54,43	0,04	16,67	18,04	17,17	16,85	0,52
3p20s	157,99	158,52	158,08	158,02	0,17	20,27	22,07	20,74	20,57	0,56
4p5s	0,68	0,89	0,82	0,82	0,04	1,11	1,81	1,3	1,22	0,2
4p7s	4,48	4,72	4,53	4,51	0,02	6,12	6,66	6,33	6,29	0,21
4p9s	20,16	20,31	20,19	20,23	0,09	22,45	23,85	22,99	23,07	0,41
4p10s	39,47	39,58	39,55	39,55	0,03	39,46	40,67	40,13	40,22	0,43
4p12s	122,96	124,22	125,01	124,98	0,12	102,47	107,55	104,53	104,72	2,08
5p5s	29,02	29,64	29,33	29,11	0,01	22,46	24,23	23,46	23,49	0,6
5p6s	106,93	107,19	107,04	107,02	0,09	58,46	68,24	63,72	64,25	3,36
5p7s	203,38	205,39	203,96	203,68	0,61	169,57	186,25	177,8	177,57	4,46

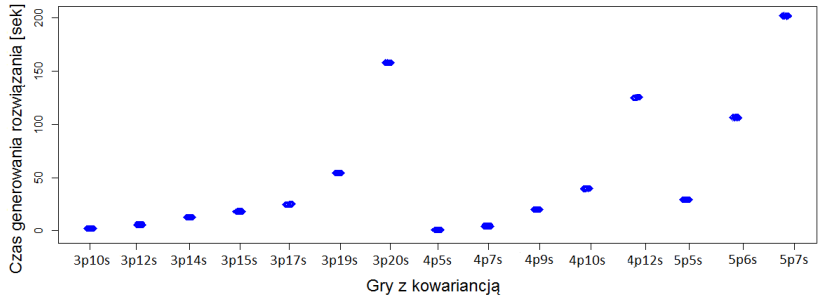
Tabela A.2: Czas działania algorytmu (znalezienie pierwszego rozwiazania) w sekundach - gry RoadMap. Pierwsza część tabeli to wyniki dla algorytmów SM oraz GNM, z kolei druga część to proponowane w rozprawie algorytmy DE oraz ADE (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	SM					GNM				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	3,89	182,31	51	34,69	53,06	1,32	4,45	2,08	1,59	1,09
3p12s	6,87	70,12	18,77	14,43	18,83	1,25	8,11	2,99	1,75	2,67
3p14s	16,01	57,51	36,43	32,41	19,46	1,8	24,81	5,28	2,76	7,04
3p15s	18,27	48,08	35,75	39,46	11,34	2,09	9,26	4,88	3,92	2,8
3p17s	21,24	75,87	51,91	55,28	22,86	3,54	13,15	7,92	7,62	3,94
3p19s	18,5	98,91	57,03	53,68	40,3	6,32	32,49	15,45	9,47	11,17
3p20s	25,86	69,69	42,5	31,97	23,73	13,08	68,83	43,72	49,27	28,28
4p5s	3,32	157,11	22,8	5,21	50,44	1,23	2,84	1,92	1,92	0,5
4p7s	3,56	28,89	14,71	14,83	10,33	1,77	5,72	2,98	2,34	1,45
4p9s	18,4	36,19	27,76	28,71	8,93	2,9	13,69	6,46	5,85	3,25
4p10s	31,81	186,15	120,76	144,33	79,82	3,55	28,01	10,32	7,91	6,89
4p12s	217,51	217,51	202,17	202,17	21,68	8,07	52,03	22,86	15,44	17,35
5p5s	3,9	585,12	151,72	45,75	223,86	3,12	14,84	5,65	4,36	3,46
5p6s	11,08	69,25	32,11	16,01	32,25	3,7	18,26	10,96	10,91	5,89
5p7s	56,01	173,28	114,64	114,64	82,92	8,73	41,26	18,7	12,66	12,96

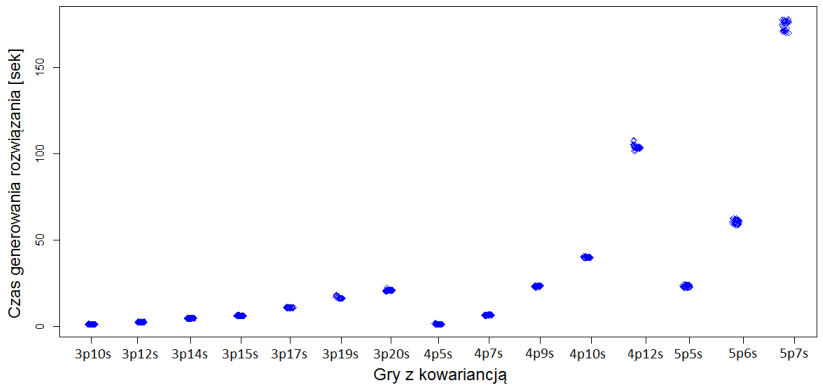
gra	DE					ADE				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	2,34	2,39	2,37	2,36	0,01	1,25	1,48	1,31	1,27	0,07
3p12s	5,85	5,88	5,86	5,86	0,01	2,38	2,78	2,56	2,56	0,13
3p14s	12,84	12,86	12,85	12,85	0,01	4,43	5,12	4,64	4,58	0,22
3p15s	18,36	18,38	18,37	18,37	0,01	5,99	6,52	6,18	6,08	0,19
3p17s	24,73	25,14	24,85	24,82	0,12	10,33	11,76	10,9	10,83	0,5
3p19s	54,38	55,5	54,85	54,46	0,04	16,66	18,4	17,35	17,1	0,71
3p20s	157,82	158,75	158,14	158,03	0,33	20,25	21,48	20,77	20,66	0,51
4p5s	0,75	0,86	0,82	0,84	0,03	1,12	1,85	1,33	1,23	0,22
4p7s	4,5	4,57	4,53	4,53	0,02	6,12	6,78	6,41	6,35	0,26
4p9s	0,79	20,72	18,03	20,38	6,96	22,14	23,85	22,8	22,62	0,51
4p10s	39,19	41,58	40,37	40,1	0,83	39,46	40,94	40,04	39,95	0,48
4p12s	124,81	125,22	125,02	125,05	0,15	101,47	105,67	104,19	104,56	1,17
5p5s	29,01	29,14	29,06	29,04	0,05	22,56	24,23	23,37	23,51	0,58
5p6s	106,37	108,57	107,76	107,94	0,7	58,46	68,24	63,05	62,71	3,33
5p7s	201,83	203,96	202,89	202,9	0,59	171,62	190,62	180,24	178,41	6,26



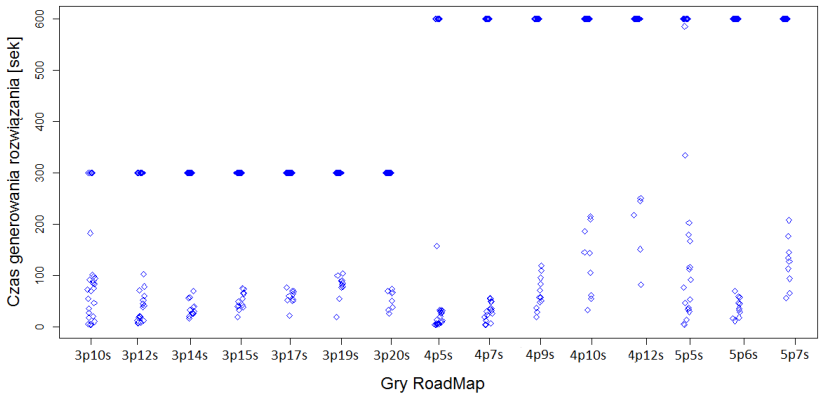
Rysunek A.6: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm GNM



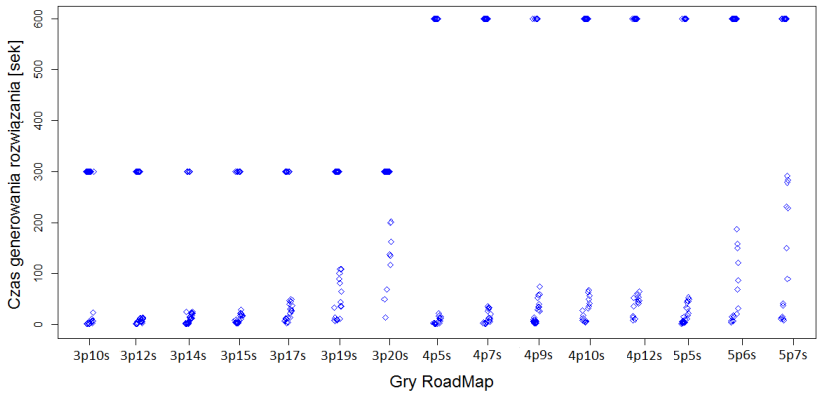
Rysunek A.7: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm DE



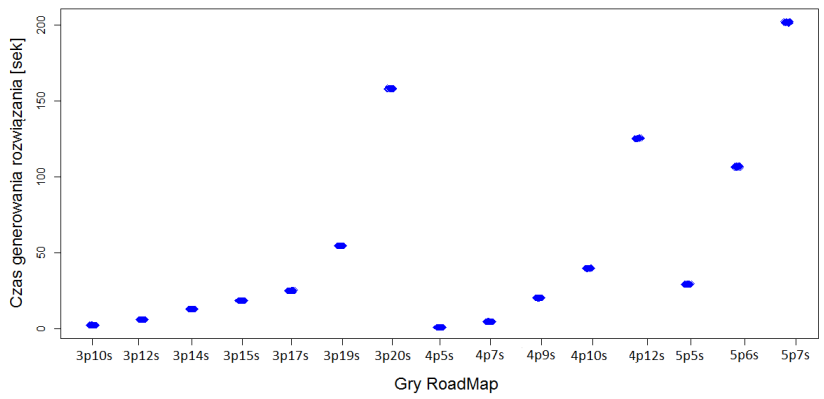
Rysunek A.8: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm ADE



Rysunek A.9: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm SM



Rysunek A.10: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm GNM



Rysunek A.11: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm DE

Rysunek A.12: Czas generowania rozwiązania dla poszczególnych uruchomień - algorytm ADE

Tabela A.3: Liczba znalezionych rozwiązań dla algorytmów ADE oraz GNM - gry RoadMap (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	GNM				ADE2				ADE3				ADE4			
	min	max	śred	std	min	max	śred	std	min	max	śred	std	min	max	śred	std
3p10s	0	11	3,6	4,11	10	10	10	0	10	10	10	0	10	10	10	0
3p12s	0	17	4,2	6,1	10	10	10	0	10	10	10	0	10	10	10	0
3p14s	0	22	6,2	5,8	10	10	10	0	10	10	10	0	10	10	10	0
3p15s	0	20	6,2	6,7	9	10	9,8	0,4	7	10	9,4	1,1	9	10	9,6	0,5
3p17s	0	14	3,3	4,7	8	10	9,6	0,8	8	10	9,8	0,7	7	10	9,5	0,9
3p19s	0	11	3,6	4,6	6	10	9,1	1,5	6	10	9,2	1,5	6	10	9,5	1
3p20s	0	8	1,2	2,6	6	10	9	1,2	6	10	9,5	1,4	6	10	9,3	1,6
4p5s	0	14	2,9	4,9	10	10	10	0	10	10	10	0	10	10	10	0
4p7s	0	9	1,9	2,8	10	10	10	0	10	10	10	0	10	10	10	0
4p9s	0	13	5,2	3,5	5	10	8,9	1,5	7	10	9,2	1,4	6	10	9,1	1,5
4p10s	0	12	3,5	3,8	6	10	9,4	1,2	6	10	9,5	1,5	6	10	9,5	1,3
4p12s	0	11	2	3,4	7	10	9,5	1	5	10	9	1,8	6	10	9	1,5
5p5s	0	10	2,8	3,3	8	10	9,7	0,7	8	10	9,8	0,5	7	10	9,6	1,1
5p6s	0	16	2,3	4,3	7	10	9,4	1,1	7	10	9,4	1	8	10	9,6	0,7
5p7s	0	11	2,8	3,7	8	10	9,5	0,8	6	10	9,3	1,1	6	10	9	1,7

Tabela A.4: Liczba strategii aktywnych w rozwiązaniu - gry z kowariancją (min-minimum, maxmaksimum, śred - wartość srednia oraz std - odchylenie standardowe)

gra	SM				GNM				DE			
	min	max	śred	std	min	max	śred	std	min	max	śred	std
3p10s	5	7	6	1	8	15	12,3	2,2	13	21	17,8	2,5
3p12s	6	11	8	1,6	8	21	13,5	4,5	18	25	21,5	2,2
3p14s	5	10	7,2	2	6	19	13,6	3,9	21	27	24,6	2,2
3p15s	7	9	7,8	0,8	7	20	13	4,3	19	30	26,1	3,9
3p17s	11	13	11,5	0,8	11	19	14,8	2,8	23	37	29,2	4,9
3p19s	7	8	7,2	0,5	9	17	12,6	4	29	37	31,2	2,9
3p20s	-	-	-	-	11	24	15,3	7,5	31	39	34,6	2,4
4p5s	6	11	8,2	1,7	10	17	12,2	2,5	11	16	13,7	1,5
4p7s	6	9	7,2	1,1	9	17	12,8	2,3	15	22	18,4	1,7
4p9s	6	8	6,8	0,7	9	20	12,5	4	20	28	23,9	2,3
4p10s	-	-	-	-	13	17	15	2,8	17	33	25,6	4,5
4p12s	-	-	-	-	20	19	22	1,3	26	33	28,4	2,5
5p5s	8	11	9,8	1,3	10	17	13,3	2,4	16	21	18,4	1,5
5p6s	8	11	9,5	1	12	20	16,2	3,5	16	21	18,4	1,5
5p7s	-	-	-	-	13	19	16	4,2	18	28	22,4	2,7

Tabela A.5: Liczba strategii aktywnych w rozwiązaniu - gry RoadMap (min- minimum, maxmaksimum, śred - wartość srednia oraz std - odchylenie standardowe)

gra	SM				GNM				DE			
	min	max	śred	std	min	max	śred	std	min	max	śred	std
3p10s	5	9	5,8	1,4	5	13	8,4	3,3	12	21	17,6	2,7
3p12s	5	6	5,4	0,8	7	18	10,8	4,2	18	24	21	2,1
3p14s	5	5	5,2	0,4	9	20	12,2	3,7	20	26	23,4	2,2
3p15s	5	7	5,5	1	9	23	13,7	5	23	31	26,1	2,7
3p17s	5	9	7	1,6	7	17	12,3	3,3	26	36	30,1	2,8
3p19s	6	9	7	1,7	7	16	11,1	3,2	26	38	32,1	3,8
3p20s	5	8	6,6	1,5	11	20	15,3	4,5	30	38	33,9	2,6
4p5s	6	8	6,7	0,9	6	13	8,5	2,5	11	16	13,5	1,7
4p7s	6	10	7,6	1,6	8	13	10,4	2	15	22	18	2,1
4p9s	6	10	7,6	2	7	20	13,3	4,2	21	26	23,4	1,8
4p10s	6	10	8,3	2	8	20	13,3	3,9	21	29	24,5	3,1
4p12s	6	8	7,2	0,8	11	20	15	3,7	19	33	28,8	4,1
5p5s	7	11	8,7	1,5	8	16	10,9	2,6	15	21	18,2	1,8
5p6s	7	10	8	1	9	14	11,1	1,7	17	21	19,2	1,3
5p7s	10	13	11,6	1,1	10	14	12	1,4	14	23	20,6	2,5

Tabela A.6: Porównanie wartości ϵ dla algorytmów DE oraz ADE - gry z kowariancją (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

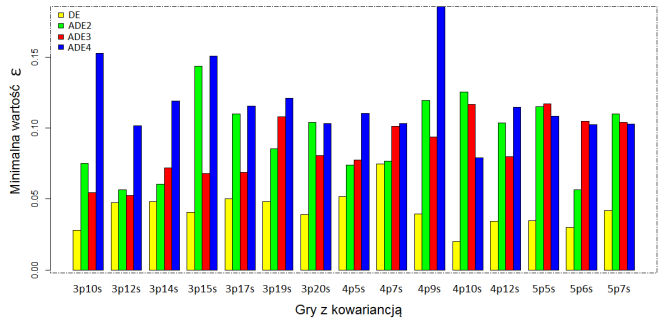
gra	DE					ADE2				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,0281	0,15655	0,09092	0,09106	0,03743	0,07498	0,20405	0,1194	0,0971	0,05337
3p12s	0,04722	0,15157	0,09638	0,09938	0,02941	0,05636	0,18001	0,12016	0,13489	0,05114
3p14s	0,04823	0,19098	0,10472	0,09893	0,04502	0,0604	0,37865	0,1483	0,10722	0,1166
3p15s	0,0408	0,17539	0,1036	0,11028	0,03901	0,14336	0,22726	0,1976	0,20499	0,0316
3p17s	0,05013	0,25832	0,10879	0,09962	0,05635	0,10985	0,21989	0,16044	0,15255	0,04058
3p19s	0,04833	0,33296	0,15945	0,14882	0,08895	0,08552	0,22618	0,14705	0,14641	0,04999
3p20s	0,03928	0,36049	0,14715	0,14552	0,09391	0,10387	0,1675	0,13466	0,13246	0,0286
4p5s	0,05161	0,19611	0,1088	0,10749	0,03688	0,07382	0,2159	0,14608	0,1255	0,05699
4p7s	0,07453	0,27563	0,12462	0,10584	0,05407	0,07664	0,23945	0,1518	0,1475	0,05483
4p9s	0,03962	0,3661	0,13283	0,0939	0,09414	0,11928	0,28776	0,1862	0,1892	0,06291
4p10s	0,02007	0,36277	0,14222	0,11868	0,09813	0,12534	0,16751	0,14536	0,14267	0,0155
4p12s	0,03417	0,37477	0,16951	0,11076	0,11929	0,1037	0,30666	0,22043	0,23462	0,06717
5p5s	0,03482	0,28833	0,12593	0,11268	0,08158	0,11487	0,32668	0,18612	0,15749	0,08294
5p6s	0,03004	0,29085	0,11533	0,0993	0,0688	0,05655	0,21906	0,13569	0,12538	0,05433
5p7s	0,04176	0,29404	0,15696	0,13852	0,09668	0,10979	0,3545	0,21733	0,1967	0,09371

gra	ADE3					ADE4				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,05439	0,20602	0,13325	0,14322	0,05498	0,15241	0,37145	0,2287	0,19547	0,10111
3p12s	0,05245	0,24329	0,18984	0,22993	0,07946	0,10165	0,22697	0,17762	0,19093	0,05622
3p14s	0,07199	0,24142	0,1597	0,15482	0,06188	0,11903	0,19386	0,15878	0,16112	0,03121
3p15s	0,06811	0,19519	0,13814	0,13484	0,0465	0,15071	0,1926	0,17228	0,17291	0,01899
3p17s	0,06864	0,18391	0,12631	0,14078	0,05171	0,1154	0,26971	0,17306	0,15356	0,06823
3p19s	0,10802	0,22968	0,16744	0,15664	0,04515	0,12085	0,21597	0,1754	0,1824	0,04262
3p20s	0,08081	0,32587	0,16565	0,13497	0,09394	0,103	0,18541	0,14981	0,15541	0,03622
4p5s	0,07757	0,20648	0,13782	0,12526	0,05482	0,11019	0,20259	0,14089	0,12538	0,04177
4p7s	0,10128	0,29434	0,18532	0,15444	0,07787	0,10331	0,29736	0,18098	0,16163	0,08305
4p9s	0,0937	0,21387	0,15027	0,16967	0,05372	0,18534	0,26406	0,22357	0,22244	0,03232
4p10s	0,11643	0,27355	0,16778	0,14016	0,06287	0,07913	0,24203	0,16151	0,16245	0,08291
4p12s	0,07985	0,21032	0,12085	0,10731	0,05175	0,11473	0,16379	0,13854	0,13783	0,02107
5p5s	0,11719	0,17642	0,15045	0,14931	0,02165	0,10827	0,23159	0,15588	0,14184	0,05322
5p6s	0,10486	0,17285	0,12731	0,11782	0,02726	0,10251	0,13711	0,12195	0,12409	0,01651
5p7s	0,10401	0,23271	0,1547	0,14786	0,04745	0,10273	0,27533	0,19768	0,20633	0,07159

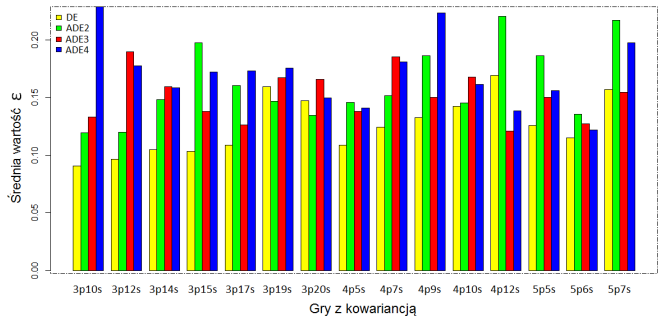
Tabela A.7: Porównanie wartości ϵ dla algorytmów DE oraz ADE - gry RoadMap (min- minimum, max-maksimum, śred - wartość średnia, med - mediana oraz std - odchylenie standardowe)

gra	DE					ADE2				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,09481	0,28702	0,15906	0,1461	0,05702	0,16132	0,36116	0,26467	0,25482	0,07351
3p12s	0,06614	0,24063	0,16732	0,1778	0,0551	0,12631	0,26097	0,20011	0,18492	0,05544
3p14s	0,08837	0,23971	0,17062	0,16639	0,04346	0,13932	0,30171	0,21051	0,19613	0,06918
3p15s	0,01366	0,16508	0,11164	0,11829	0,03795	0,1309	0,20713	0,16406	0,1591	0,0347
3p17s	0,05274	0,23532	0,12677	0,1188	0,05165	0,16646	0,35085	0,25462	0,25058	0,07831
3p19s	0,04775	0,25882	0,16177	0,16284	0,07456	0,25046	0,30576	0,28515	0,29218	0,02581
3p20s	0,07091	0,27725	0,13687	0,13241	0,05631	0,17592	0,25283	0,21278	0,21118	0,03439
4p5s	0,0704	0,34088	0,20108	0,1939	0,07633	0,23782	0,36421	0,29837	0,29573	0,06083
4p7s	0,07287	0,37306	0,21578	0,21349	0,07799	0,13911	0,3656	0,24267	0,233	0,09922
4p9s	0,07194	0,33559	0,19635	0,20645	0,07991	0,19913	0,37082	0,28229	0,2796	0,08456
4p10s	0,06411	0,35036	0,21213	0,21617	0,08341	0,15453	0,28951	0,21767	0,21332	0,06583
4p12s	0,11548	0,4429	0,27728	0,29335	0,09478	0,11152	0,39528	0,24679	0,24018	0,11614
5p5s	0,09249	0,31053	0,17177	0,13921	0,07146	0,21175	0,33041	0,28274	0,2944	0,05135
5p6s	0,06954	0,35345	0,19184	0,18126	0,08076	0,18999	0,36087	0,28974	0,30406	0,07651
5p7s	0,10574	0,3735	0,23845	0,22329	0,09131	0,11052	0,35744	0,26592	0,29786	0,10973

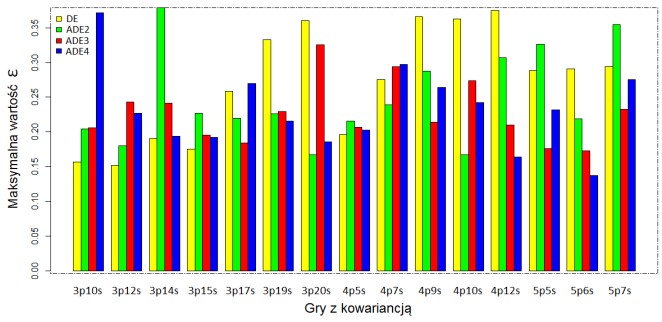
gra	DE					ADE2				
	min	max	śred	med	std	min	max	śred	med	std
3p10s	0,11362	0,25099	0,20004	0,22152	0,05682	0,14733	0,2906	0,21318	0,2074	0,06762
3p12s	0,07618	0,35664	0,25466	0,29259	0,10902	0,16443	0,26511	0,22122	0,22767	0,04714
3p14s	0,22383	0,33826	0,2763	0,25078	0,05512	0,17814	0,3105	0,25233	0,26035	0,06775
3p15s	0,16146	0,26803	0,22008	0,20833	0,04507	0,10447	0,29367	0,16989	0,14072	0,08931
3p17s	0,17319	0,24987	0,20451	0,20969	0,03068	0,16605	0,2318	0,19547	0,19201	0,03443
3p19s	0,14024	0,34156	0,27116	0,29918	0,07922	0,26813	0,3533	0,30453	0,29835	0,03549
3p20s	0,17008	0,39578	0,29535	0,34065	0,09455	0,13966	0,33781	0,24429	0,24985	0,08182
4p5s	0,23835	0,322	0,28893	0,29113	0,03154	0,19607	0,2612	0,22807	0,22751	0,03414
4p7s	0,21514	0,29245	0,24324	0,2279	0,03191	0,14051	0,29152	0,23326	0,2505	0,06687
4p9s	0,12663	0,31759	0,22339	0,2143	0,07567	0,2393	0,27555	0,26233	0,26723	0,01584
4p10s	0,23349	0,27986	0,26066	0,27585	0,02451	0,19416	0,2581	0,23166	0,23719	0,02936
4p12s	0,22086	0,31809	0,26758	0,2701	0,03481	0,14059	0,32021	0,22864	0,22688	0,07591
5p5s	0,23609	0,39529	0,31036	0,30036	0,06866	0,22036	0,38839	0,27962	0,25486	0,07641
5p6s	0,21349	0,39698	0,29992	0,29035	0,06942	0,20721	0,37394	0,28399	0,2774	0,06853
5p7s	0,19339	0,32467	0,25581	0,24509	0,05778	0,25928	0,34736	0,31149	0,31966	0,0375



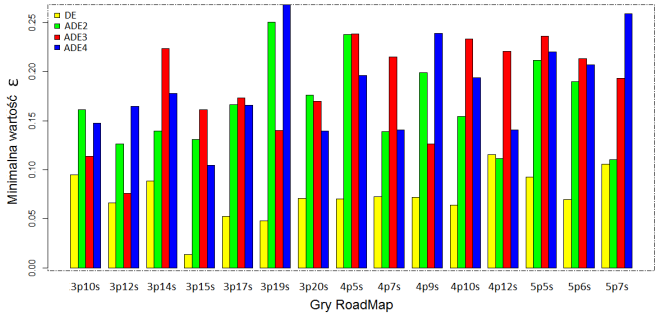
Rysunek A.13: Dokładność dla uzyskiwanych rozwiązań - gry z kowariancją - wartości minimalne



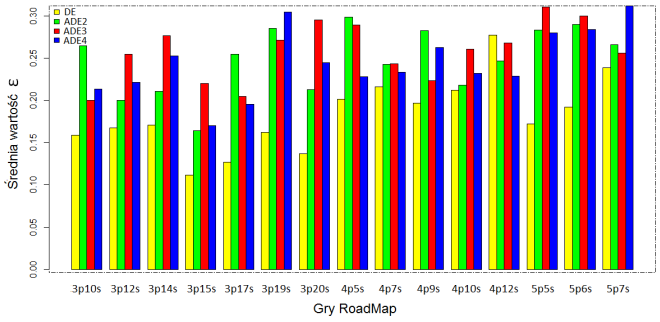
Rysunek A.14: Dokładność dla uzyskiwanych rozwiązań - gry z kowariancją - wartości średnie



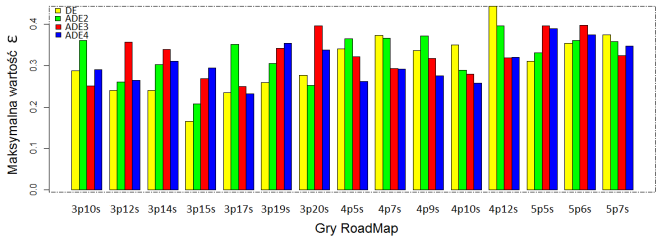
Rysunek A.15: Dokładność dla uzyskiwanych rozwiązań - gry z kowariancją - wartości maksymalne



Rysunek A.16: Dokładność dla uzyskiwanych rozwiązań - gry RoadMap - wartości minimalne



Rysunek A.17: Dokładność dla uzyskiwanych rozwiązań - gry RoadMap - wartości średnie



Rysunek A.18: Dokładność dla uzyskiwanych rozwiązań - gry RoadMap - wartości maksymalne

Spis rysunków

Spis tablic