Instytut Podstawowych Problemów Techniki
Polskiej Akademii Nauk

Analiza algorytmów genetycznych jako układów dynamicznych

mgr inż. Stefan Kotowski

Rozprawa doktorska napisana pod kierunkiem
prof. dr. hab. Witolda Kosińskiego

Warszawa, Czerwiec 2008

Podziękowania

Pragnę złożyć serdeczne podziękowania wszystkim osobom, które swoimi cennymi uwagami przyczyniły się do realizacji niniejszej rozprawy i w znacznym stopniu pomogły mi zrealizować badania w niej zawarte.

Pracę tę dedykuję wszystkim moim najbliższym, a w szczególności mojej Rodzinie

Stefan Kotowski

Spis treści

Wstęp

W obliczeniach ewolucyjnych wykorzystujemy idee i inspiracje zapożyczone z procesów naturalnej ewolucji i adaptacji zauważonych w przyrodzie. Klasyczne obliczeniowe systemy optymalizacyjne są znakomite dla dokładnych i ścisłych obliczeń, lecz także wrażliwe i nieodporne. Stawiają one wymagania, aby zadania optymalizacyjne były sformułowane w sposób jednoznaczny i dokładny, w precyzyjnej formie matematycznej. Nie są one przystosowane do zadań niedokładnych, o złożonych danych także wtedy, gdy funkcja celu (oceny) nie jest dana dokładną formułą matematyczną lub jest zaburzona i nieprecyzyjna.

Obliczenia ewolucyjne pozwalają takie problemy podejmować, a nawet rozwiązywać. Są one uzupełnieniem klasycznych metod obliczeniowych. Większość technik obliczeń ewolucyjnych ma swoje źródła ideowe i inspiracje w ewolucji molekularnej immunologii, genetyce populacyjnej. Terminologia używana w obliczeniach ewolucyjnych jest zapożyczona z tych dziedzin i jest odbiciem tych powiązań. Przykładem są: algorytmy genetyczne, genotyp, fenotyp, środowisko itd. Jednocześnie stosując i badając algorytmy ewolucyjne, można przybliżyć się do zrozumienia szeregu zjawisk biologicznych, chociaż nie jest ich głównym celem budowa poprawnych biologicznie modeli.

Celem badań algorytmów ewolucyjnych jest rozwój efektywnych systemów obliczeniowych rozwiązujących złożone zadania praktyczne występujące w rzeczywistości. Obliczenia ewolucyjne są dziedziną, która wyłoniła się niedawno i przeżywa gwałtowny rozwój w ostatnich latach.

Wszystkie algorytmy ewolucyjne maja dwie istotne cechy, które wyróżniają je wśród innych algorytmów przeszukiwania. Po pierwsze, ich dziedziną są populacje i to je odróżnia od innych algorytmów. Ponadto osobniki komunikują się i wymieniają informacje w ramach populacji. Komunikowanie się i wymiana informacji jest wynikiem selekcji i rekombinacji (krzyżowania)w algorytmach. Operatory genetyczne tworzą potomków z rodziców, tj. nowych osobników z istniejących osobników. Różne sposoby definiowania reprezentacji osobników i wielość sposobów implementacji funkcji wartościującej (funkcji oceny, przystosowania), operacji selekcji, krzyżowania i mutacji, tj. operatorów przeszukiwania, definiują odmienne algorytmy.

W algorytmach genetycznych wykorzystuje się kodowanie potencjalnych rozwiązań (osobników) w postaci chromosomów i oddziaływanie operatorów genetycznych na te chromosomy. Jest to równoważne przekształceniu oryginalnego zadania (optymalizacyjnego) z przestrzeni, w której początkowo zostało ono oryginalnie sformułowane (mówimy tutaj czasami o tzw. przestrzeni fizycznej), do innej przestrzeni. Sposób tego przekształcenia (reprezentacji) jest kluczowy dla sukcesu algorytmu genetycznego. Odpowiednia reprezentacja ułatwia rozwiązanie zadania, natomiast zła – utrudnia. Tak więc istotnym problemem algorytmów genetycznych jest pytanie, jaką reprezentację należy wybrać, aby efektywnie przeszukiwać dziedzinę zadania (funkcji

celu).

Cel i zakres rozprawy

W rozprawie chcemy przedstawić uzyskane wyniki dotyczące postaci rozkładu granicznego algorytmów genetycznych (ewolucyjnych) modelowanych jako łańcuchy Markowa. (Vose [101], Rowe [68]). Dotychczasowe wyniki badań algorytmów genetycznych (ewolucyjnych) modelowanych łańcuchami Markowa kończyły się na twierdzeniach egzystencjalnych. Mówiły one, że istnieje rozkład graniczny dla łańcuchów Markowa opisujących algorytmy genetyczne, przy czym na ogół ograniczano się do prostego algorytmu genetycznego [101, 76, 63].

W proponowanym opracowaniu chcemy przedstawić następny krok na tej drodze. Przedstawimy postać rozkładu granicznego łańcucha Markowa, opisującego konkretną postać algorytmu genetycznego. Rozszerzymy też twierdzenie o zbieżności na algorytmy ze zmienną mutacją oraz z funkcją przystosowania zależną od populacji i selekcją elitarną.

Jawna postać rozkładu granicznego ma znaczenie egzystencjalne, ze względu na to, że do jej wyznaczenia niezbędna jest wiedza o rozkładzie funkcji przystosowania na całej jej dziedzinie, a więc także aprioryczna znajomość rozwiązania zadania optymalizacyjnego. Niemniej opis stanu granicznego operatora, wskazuje na możliwość opracowania metod sterowania (lub ich adaptacji) parametrami algorytmów i uzyskanie szybszej zbieżności.

Uzyskana postać rozkładu granicznego pozwala także na klasyfikację algorytmów genetycznych bazującą na metodach teorii układów dynamicznych, a więc uwzględniających rzeczywistą dynamikę przeszukiwania, niezależnie od dotychczasowej klasyfikacji taksonomicznej. W pracy będziemy chcieli też porównać wyniki proponowanej klasyfikacji dynamicznej z klasyfikacją taksonomiczną dotychczas stosowaną.

Na mocy znajomości postaci rozkładu granicznego przedstawimy optymalny - w sensie probabilistycznym - algorytm genetyczny. W postaci ogólnej taki algorytm jest niewyznaczalny, jednak jego istnienie pozwala na poprawianie, a nawet quasi-optymalizację poszczególnych algorytmów i to w dość szerokiej klasie.

Klasyfikacja oparta na własnościach dynamicznych i postać optymalnego algorytmu jest próbą odpowiedzi na wyzwania, które postawiło słynne Twierdzenie No Free Lunch Theorem [98]. Zaproponowana klasyfikacja przedstawia wyniki, w pewnym sensie komplementarne do tego twierdzenia.

Bazując na tych wynikach, teoretycznych, przedstawimy badania empiryczne nad klasyfikacją algorytmów, w których jako niezmienniki (inwarianty) klasyfikacji przyjęto entropię i wymiar fraktalny oraz jego przybliżenia: wymiar pudełkowy i informacyjny.

Algorytm optymalizacyjny

Algorytmy ewolucyjne są metodami optymalizacji, które korzystają z ograniczonej wiedzy o badanym zadaniu. Równocześnie nasza wiedza o wykorzystywanym algorytmie jest często także ograniczona. Spróbujmy umiejscowić algorytmy ewolucyjne w szerszej klasie *algorytmów optymalizacyjnych*. Skorzystamy przy tym z zaproponowanego przez Wolperta i Macready [98] definicji optymalizacyjnego algorytmu

ewolucyjnego jako operatora przeszukiwania. Wprowadźmy abstrakcyjną przestrzeń przeszukiwania $\mathcal{X}$ (search space) o dużej, ale skończonej liczbie elementów (mocy), oraz przestrzeń możliwych wartości kosztów (cost values) Y, też skończonej. W aplikacjach to drugie założenie jest automatycznie spełnione, gdyż elementy w Y mogą być utożsamione z 32 lub 64 bitowymi reprezentacjami liczb rzeczywistych.

Podstawowym obiektem w zagadnieniach optymalizacyjnych jest funkcja celu, zwana też funkcją kosztów $f : \mathcal{X} \to Y$ (cost function). Zbiór wszystkich takich funkcji, tj. $F = Y^{\mathcal{X}}$ reprezentuje zbiór wszystkich dopuszczalnych problemów.

Weźmy pod uwagę pary elementów (punktów) leżące na wykresie funkcji f, tj. (x, y), gdzie $x \in \mathcal{X}, y \in Y$ i takie, że $y = f(x)$. Pierwszy element takiej pary jest punktem z przestrzeni poszukiwań, zaś drugi element reprezentuje wartość, jaką przyjmuje funkcja kosztów w tym punkcie. Dalej będziemy się zajmowali skończonymi zbiorami takich par, a dokładniej ich ciągami tworzącymi się w wyniku działania algorytmu genetycznego.

Mówimy, że mamy do czynienia z próbką o rozmiarze m, gdzie m-naturalne, gdy dany jest skończony ciąg takich punktów o długości m. Dla podkreślenia tego faktu autorzy [99] piszą $d_m^x(1) \equiv \{(d_m^x(1), d_m^y(1)), \ldots, (d_m^x(m), d_m^y(m))\}$. Przestrzeń wszystkich próbek o rozmiarze m, oznaczany przez D_m jest iloczynem kartezjańskim $D_m = (\mathcal{X} \times Y)^m$. Suma nieskończona tych wszystkich przestrzeni jest przestrzenią wszystkich możliwych próbek o dowolnym rozmiarze $D \equiv \bigcup_{m \geq 0} D_m$. Algorytm optymalizacyjny jest najczęściej reprezentowany przez iteracyjne przekształcenie zbioru ostatnio odwiedzanych punktów w pewien nowy zbiór w $\mathcal{X}$.

Definicja. *Algorytm optymalizacyjny a jest przekształceniem określonym na D o wartościach w X*

$$a : D \to X$$

spełniającym dodatkowy warunek

$$a : D \ni d \longmapsto \{x | x \notin d^x\}$$

.

Warunek definicji oznacza wymaganie, aby projekcja $pr_{\mathcal{X}}(d)$ nie miała punktów wspólnych z wartością $f(d)$. Trzeba sobie tutaj zdać sprawę z pewnego ważnego ograniczenia na algorytm, jakie stawia ten warunek definicji [98].

Rozróżnienie pomiędzy zadaniem optymalizacyjnym a algorytmem (jego dobór) jest trudne. Ponadto takie rozróżnienie jest operacją sztuczną, ponieważ zmienia ono główną ideę algorytmów genetycznych, w których funkcja dopasowania otrzymana z funkcji celu, jest głównym elementem algorytmu genetycznego i tworzy postać algorytmu. Jednak w tej rozprawie będą używane oba pojęcia algorytmu.

Przedstawiana rozprawa wpisuje się, w zamierzeniach i nadziei autora, w nurt badań, które prowadzą na drogę, na której wyłania się "zunifikowana teoria" algorytmów ewolucyjnych (genetycznych). Jak pisze R.Galar [28] "Przez adaptacyjne zbocza, wierzchołki, siodła i grzbiety" *"Algorytmy ewolucyjne mają popularność i renomę metod, które - choć nie bardzo wiadomo dlaczego, pozwalają uzyskać przyzwoite rezultaty w zadaniach trudnych do "ugryzienia" innymi sposobami. Taki, nieco magiczny status, pasujący do epoki New Age, Algorytmy Ewolucyjne zdają się dzielić z sieciami neuronowymi i technikami rozmytymi. Zunifikowana teoria algorytmów ewolucyjnych wydaje się więc wisieć w powietrzu, ale - jak sugeruje doświadczenie, z zapowiedziami takich teorii - warto póki co miarkować entuzjazm sceptycyzmem. Tak bowiem się składa, że choć wyraźnie wyłaniają się zgęstki podobieństw, zdające*

się upoważniać do generalizacji, to zarazem obszar konfuzji otaczający podstawowe koncepty nie wydaje się maleć w sposób znaczący.... Specjaliści od optymalizacji,.. przenieśli w ich obręb szereg nawyków , które zniekształcają raczej istotę rzeczy (np. dążenie do asymptotycznej zbieżności)." Mimo tej opinii i pozostając w tej konwencji, można stwierdzić, że autor tej rozprawy wszedł na drogę asymptotycznej zbieżności i sądzi, że przedstawione w pracy wyniki przybliżają nas do wyłonienia się teorii algorytmów, chociaż bez ambicji jej zunifikowania, która zawęzi wspomniany "obszar konfuzji". Dalej (R. Galar) stawia tezę, że *"ewolucja jest procesem postępującej eksploracji bezpośredniego otoczenia, a nie gmeraniem w pudle z wszelakimi rozwiązaniami".* Wyniki autora stawiają tezę przeciwną: w procesie realizowanym przez algorytm ewolucyjny zmierzamy do sytuacji, w której wszystkie rozwiązania mogą wystąpić, ale ich prawdopodobieństwo wyraźnie się różnicuje, preferując rozwiązania lepsze w istotny sposób.

Tezy rozprawy

Rozprawa stawia sobie trzy tezy:
1. **Istnieje rozkład graniczny i można go opisać jawną zależnością. Zależność ta wskazuje sposób ulepszania algorytmu.**
2. **Dla każdego wyjściowego algorytmu genetycznego istnieje algorytm genetyczny, optymalny w sensie probabilistycznym.**
3. **Możliwa jest klasyfikacja algorytmów genetycznych na podstawie ich entropii i wymiaru fraktalnego trajektorii. Może być ona pożyteczna przy projektowaniu następnych algorytmów genetycznych.**

W pracy zdefiniowano prosty algorytm genetyczny w terminach skończonego multizbioru potencjalnych rozwiązań (osobników danej populacji), na którym są określone operacje krzyżowania, mutacji i selekcji, każdy z pewnym prawdopodobieństwem. Operacje te, działając iteracyjnie, tworzą nową populację. Istnienie funkcji przystosowania (dopasowania), określonej na osobnikach populacji, pozwala powiązać prawdopodobieństwo selekcji osobników do nowej populacji z wartościami, jakie funkcja przystosowania przyjmuje na osobniku. Rozpatrując przejście pomiędzy wektorami prawdopodobieństwa, z jakim w kolejnych pokoleniach pojawiają się populacje, otrzymujemy operator Markowa. W teorii operatorów Markowa oraz operatorów dodatnich dowiedzionych jest wiele różnych twierdzeń, dotyczących istnienia punktów stałych oraz zbieżności ciągu iteracji operatora (przykładowo [52, 69, 85]). Korzystając z tych wyników znaleźliśmy warunki wystarczające i konieczne stabilności operatora Markowa związanego z pewną klasą algorytmów genetycznych.

Problemy optymalizacyjne a algorytmy genetyczne

Problemy optymalizacyjne pełnią ważną rolę we współczesnych zastosowaniach matematyki do zagadnień modelowych w fizyce i technice. Wiele rozwiązań konkretnych problemów nie byłoby możliwe do uzyskania, gdyby nie zostały odpowiednio rozwinięte metody optymalizacji funkcji rzeczywistych, tj. poszukiwanie minimum czy maksimum danego problemu fizycznego. Wszystkie zadania optymalizacji mają w całej swej różnorodności kilka wspólnych cech. Jeśli przestrzeń ta ma strukturę liniową, to wtedy zakłada się, że jest skończenie wymiarowa. Wśród elementów tego zbioru szukamy rozwiązań zadania. Ponadto na przestrzeni $\mathcal{X}$ jest określona funkcja

jakości (lub funkcja celu) $\phi : \mathcal{X} \to \mathbf{R}$, przyporządkowująca elementowi $x \in \mathcal{X}$ jego ocenę ze zbioru liczb rzeczywistych $\mathbf{R}$. W funkcji jakości ϕ zawieramy różne kryteria, których spełnienia wymagamy od pożądanego rozwiązania. Funkcja ϕ nie musi być gładka. Dzięki niej możemy porównywać otrzymane rozwiązania.

Znalezienie zadowalającego rozwiązania dla jednego z takich problemów wymaga dużej wiedzy o zadaniu i znajomości algorytmów optymalizacyjnych. Należy zdawać sobie sprawę z tego, że rozwiązywanie praktycznych problemów jest w dużej mierze sztuką, w której znaczącą rolę pełni nie tylko praktyka, co konieczność korzystania z metod heurystycznych.

Zazwyczaj sam fakt wykorzystywania w obliczeniach komputerów powoduje, że w efekcie znajdujemy jedynie pewne przybliżone rozwiązanie. Dzieje się tak dlatego, że liczby w komputerze są reprezentowane przez skończony ciąg bitów, co powoduje kumulowanie się błędów zaokrągleń. Ponadto w obliczeniach komputerowych nie realizujemy zazwyczaj procesów ciągłych, a jedynie ich dyskretne przybliżenia i to przez odpowiednią skończoną liczbę dyskretnych kroków.

Obok powyższego pojawiają się inne problemy: niedoskonałość metod numerycznych, czy duża złożoność obliczeniowa. W takich przypadkach zastosowanie znanego dokładnego algorytmu nie jest możliwe i jesteśmy zmuszeni poszukiwać innych podejść do rozwiązywanego problemu optymalizacyjnego. Pewną klasą algorytmów, które dowiodły swojej przydatności w najtrudniejszych zadaniach praktycznych są *algorytmy probabilistyczne*. Algorytmy te znajdują rozwiązanie problemu (zwykle suboptymalne) z prawdopodobieństwem mniejszym od 1. Natomiast ich przebieg działania zależy nie tylko od właściwości zadania, ale także od czynników losowych.

Jednym z najczęściej używanych sposobów poszukiwania rozwiązań niegładkich problemów optymalizacyjnych jest *algorytm genetyczny*. Jest on usytuowany w nurcie *obliczeń ewolucyjnych*. Pojawił się jako pewna implementacja procesu ewolucji (por. Holland [35] i Hollstien [36]) doboru organizmów żywych zauważona w przyrodzie, gdy osobniki danego gatunku będąc lepiej przystosowanymi do warunków, w których żyją, mają większe prawdopodobieństwo przeżycia niż osobniki gorzej przystosowane.

Obliczenia ewolucyjne należą do rozwijającej się dziedziny wiedzy zwanej inteligencją obliczeniową (ang. *computational intelligence*) wyrosłej z dziedziny zwanej sztuczną inteligencją [49, 13]. Rozwijane w tej dziedzinie metody obliczeniowe czasami noszą nazwę *metod miękkich*, jako przeniesienie z angielskiego *soft computing*, gdyż stosowane tutaj podstawy algorytmów nie są dobrze umotywowane, a inspirację dla nich zaczerpnięto z przyrody. Brak jest jednak pełnych dowodów ich poprawności.

Algorytmy probabilistyczne, a wśród nich algorytmy genetyczne, na ogół nie przeszukują przestrzeni poszukiwań (odpowiednik przestrzeni potencjalnych rozwiązań problemu optymalizacyjnego) w sposób wyłącznie losowy. Zwykle bazują one na pewnych heurystykach, np. na założeniach dotyczących kształtu i regularności funkcji jakości zadania. Heurystyczne są tutaj nie tylko założenia o funkcji jakości (inaczej celu), ale też same metody tworzenia algorytmu obliczeniowego [11].

Skoro kod (chromosom) opisuje budowę wszystkich żywych organizmów i służy do przechowywania i przekazywania materiału genetycznego, można się pokusić, by w sposób choćby uproszczony zastosować taki kod w obliczeniach komputerowych. Tutaj mamy do dyspozycji najprostszy sposób, jakim jest ciąg bitów. W ten sposób pojawiają się algorytmy wykorzystujące mechanizmy ewolucji, które nazywamy algorytmami ewolucyjnymi, a szczególnym ich rodzajem są **binarne algorytmy genetyczne** [15, 30] (BAG), będące tematem artykułu Kiesia i Michalewicza [45]. Nato-

miast artykuł Profesora Lasoty [52] był inspiracją obecnych badań.

Jak już wspomnieliśmy, organizm ma tym większe szanse przeżycia w swoim środowisku, im bardziej jest on do tego środowiska przystosowany. Aby powstawały coraz lepsze organizmy, musi być tworzona wielka liczba nowych odmian gatunku, które są następnie "testowane" w środowisku. Naturalna selekcja powoduje, że organizm słabo przystosowany umiera i jego kod DNA zanika, natomiast dobrze przystosowany ma większe szanse przetrwania, czyli rozpowszechnienia swojego kodu DNA w większej liczbie potomków. Proces tworzenia, jak i umierania jest też do pewnego stopnia losowy. Jednak jego losowość ma swoje "preferencje" wynikłe za każdym razem z oceny stopnia dostosowania (przystosowania) każdego osobnika do otoczenia (środowiska). Ten aspekt losowości jest niezmiernie ważny – odróżnia on algorytmy ewolucyjne (tutaj genetyczne) od całkowicie przypadkowych metod, np. metody Monte Carlo. Proces tworzenia nowych osobników jest realizowany za pomocą modyfikacji kodu DNA w wyniku krzyżowania i mutacji. Główna idea w procesie tworzenia nowych populacji (generacji) polega na całkowitym oddzieleniu procesu tworzenia nowych osobników i procesu oceny ich przystosowania. Istotnie, te dwa mechanizmy, pozornie nie mając ze sobą nic wspólnego, są odpowiedzialne za cały świat przyrody.

Złożenie operacji krzyżowania, mutacji i selekcji spełnia własność Markowa, przez co istnieje analogia do operatora (macierzy) Markowa. Zbieżność ciągu operacji może być w pewnych sytuacjach badana metodami znanymi z teorii operatorów Markowa. I takie podejście jest reprezentowane w niniejszej rozprawie.

ROZDZIAŁ 1

Algorytmy genetyczne. Historia i typy

1.1. Obliczenia ewolucyjne

Początki algorytmów ewolucyjnych można zauważyć w końcu lat 1950 (Bremerman, Box, Friedberg) [10, 9, 24]. Dziedzina ta pozostała wówczas mało znaną przez następne lata. Związane to było z niewielką mocą komputerów i ich niedostępnością, jak i słabością metodologiczną takiego podejścia.

Podstawowe prace (Holland, Rechenberg, Schwefel) [35], które nadały impuls rozwojowi obliczeń ewolucyjnych, pojawiły się w latach 1970 i od tego czasu narastało zainteresowanie tymi metodami i ich praktyczne zastosowanie. Doprowadziło to do gwałtownego rozwoju tych metod. Można sądzić, że główną przyczyną tak intensywnego rozwoju tych metod jest ich elastyczność i zdolność adaptacyjna, połączona z odpornym zachowaniem i całościowością przeszukiwania. Obliczenia ewolucyjne są raczej koncepcją, ideą rozwiązywania trudnych zadań optymalizacyjnych, niż zbiorem użytecznych algorytmów.

1.2. Algorytmy genetyczne

Niech $\mathcal{X}$ oznacza przestrzeń rozwiązań zadania optymalizacyjnego opisanego przez funkcję przystosowania

$$f : \mathcal{X} \to \mathbf{R} \ , \ \mathcal{X} \subset \mathbf{R}^n \tag{1.1}$$

dla której stosujemy algorytm genetyczny. Każdy element $x \in \mathcal{X}$ kodujemy w postaci chromosomu o długości l. Funkcja kodująca

$$\varphi : \mathcal{X} \to \{0,1\}^l = B \tag{1.2}$$

przekształca elementy z przestrzeni $\mathcal{X}$ w chromosomy w przestrzeni B .

Załóżmy, że algorytm genetyczny działa na r-elementowej populacji. Każda populacja tworzy podzbiór $[P^r]$ w przestrzeni iloczynu kartezjańskiego B^r. Dla i-tego pokolenia oznaczymy populację jako $[P_i^r]$, a każdy element tego podzbioru jest wektorem

$$P_i^r = [x_1^i, x_2^i, \ldots, x_r^i] \ . \tag{1.3}$$

Inne elementy tego podzbioru $[P_i^r]$ są tworzone z elementów wektora P_i^r przez permutację ich składowych. Zauważmy, że każda składowa tego wektora jest elementem przestrzeni B i jest możliwe, że pewne współrzędne mogą się powtarzać, a więc różne współrzędne wektora są tymi samymi elementami przestrzeni B.

Można powiedzieć, że populacja jest klasą równoważności punktów z przestrzeni wektorowej B^r, w której klasa równoważności jest określona poprzez klasę wszystkich możliwych permutacji zbioru r liczb $\{1, 2, ..., r\}$, które mogą być użyte do permutacji składowych współrzędnych danego punktu z B^r.

Zauważmy, że możemy utożsamiać punkty z $\mathcal{X}$ z ich zakodowaną postacią w B przy działaniu w przestrzeni $\mathcal{X}^r$. Jako trajektorię algorytmu genetycznego definiujemy zbiór

$$Tr = \bigcup_{i=1}^{L} [P_i^r] \ , \tag{1.4}$$

gdzie L jest liczbą kroków (pokoleń) realizowanych przez algorytm genetyczny.

Niech p_m i p_c będą odpowiednio prawdopodobieństwami mutacji i krzyżowania, gdy p_s jest prawdopodobieństwem selekcji, wszystkie są niezależnymi od pokolenia. Wówczas dla takiego algorytmu genetycznego prawdopodobieństwo wystąpienia populacji $[P_{i+1}^r]$ w pokoleniu $i+1$ po populacji $[P_i^r]$ w pokoleniu i , jest prawdopodobieństwem warunkowym.

$$\mathcal{P}(P_{i+1}^r | P_i^r \ , f(P_i^r), p_m, p_c, p_s) \ . \tag{1.5}$$

Populacja początkowa jest generowana na podstawie rozkładu równomiernego prawdopodobieństwa na zbiorze B , a więc każdy punkt z B ma to samo prawdopodobieństwo znalezienia się w $[P_1^r]$. Następna populacja, występująca po niej w kolejnym pokoleniu jest efektem działania algorytmu genetycznego i w wyniku tego ma niejednostajny rozkład prawdopodobieństwa.

Na mocy założeń (1.5) wynika, że prawdopodobieństwo wystąpienia każdej populacji zależy od populacji poprzedniej i nie zależy od historii (to jest nie zależy od wcześniejszych populacji; zaś prawdopodobieństwa p_m, p_c i p_s można traktować jako parametry funkcji $\mathcal{P}$.

Można zauważyć, że generowanie trajektorii algorytmu genetycznego zdefiniowanych przez (1.4), jest ergodycznym procesem Markowa i nazywając kolejne populacje (punkty trajektorii) stanami procesu, można stwierdzić, że każdy stan jest osiągalny z prawdopodobieństwem 1.

1.2.1. Algorytmy genetyczne

Algorytmy genetyczne różnią się od strategii ewolucyjnych i programowania ewolucyjnego reprezentacją osobników i operatorami przeszukiwania. Algorytmy genetyczne preferują binarne kodowanie potencjalnych rozwiązań jako chromosomów i stosowanie operatorów genetycznych do tych chromosomów. Jest to równoważne przekształceniu oryginalnego zadania z jednej przestrzeni do innej przestrzeni. Jest oczywiste, że sposób reprezentacji jest kluczowy dla sukcesu algorytmu genetycznego. Dobra reprezentacja ułatwia rozwiązanie zadania podczas, gdy zła je utrudnia. Istotny jest problem dostosowania algorytmu genetycznego do zadania, w szczególności interesuje nas odpowiedź na pytanie: jak dobrać reprezentację, która może być efektywnie przeszukiwana [67].

Kanoniczny algorytm genetyczny [100, 57, 58] (zwany prostym algorytmem | genetycznym) wykorzystuje reprezentację dwójkową, jednopunktowe krzyżowanie i

mutację punktową. Dwójkowa reprezentacja oznacza, że każdy osobnik reprezentowany jest przez pewną ilość bitów 0 lub 1. *Krzyżowanie jednopunktowe* realizujemy w ten sposób, że bierzemy dwa ciągi dwójkowe x i y o długości l, a następnie generujemy punkt krzyżowania pomiędzy pozycją 1 a $l-1$ losowo, np. c. W wyniku tego pierwszy potomek składa się z pierwszych c bitów osobnika x i pozostałych $l-c$ bitów z osobnika y. Drugi potomek ma pierwsze c bitów z y i $l-c$ bitów z x. *Mutacji* poddajemy każdy bit w ten sposób, że dla każdego bitu danego osobnika z pewnym prawdopodobieństwem zmieniamy 0 na 1 lub 1 na 0. Po nich następuje *selekcja* osobników do nowej populacji. Złożenie tej trójki operatorów nosi czasem nazwę *operatora przeszukiwania*.

Kanoniczny algorytm genetyczny ma postać:

1. Utwórz losowo populację początkową $P(0)$ i niech $i = 0$;
2. Powtarzaj
   - **(a)** Wyznacz dopasowanie każdego osobnika w $P(i)$.
   - **(b)** Wybierz rodziców z $P(i)$ na podstawie ich przystosowania z prawdopodobieństwem:
     $$p_i = \frac{f_i}{\sum_{j=1}^{r} f_j} \tag{1.6}$$
     gdzie f_i oznacza przystosowanie i-tego osobnika
   - **(c)** Do wybranych rodziców zastosuj krzyżowanie.
   - **(d)** Krzyżowane osobniki poddaj mutacji.
   - **(e)** Utwórz nową populację, pokolenie $P(i+1)$, zastępując rodziców potomkami.
3. Dopóki spełnione jest kryterium stopu.

Obliczenia ewolucyjne mają więcej wariantów niż trzy wymienione. Są to także algorytmy koewolucyjne, sztuczne systemy immunologiczne, układy samoadaptacyjne i wiele innych [15, 17, 30, 45].

1.3. Teoria algorytmów ewolucyjnych

1.3.1. Strategie ewolucyjne

Strategie ewolucyjne wykorzystują reprezentacje osobników zbliżone do reprezentacji naturalnej (dziesiętnej), a nie preferują reprezentacji binarnej (genetycznej) osobników. Najczęściej osobniki są reprezentowane jako wektory liczb rzeczywistych. Strategie ewolucyjne wykorzystują deterministyczne schematy (metody) selekcji, mutację Gaussowską oraz dyskretne lub uśredniające krzyżowanie. Nie podejmują one naśladownictwa ewolucji na poziomie genetycznym. Jest ono raczej fenotypowe [8]. Selekcja jest realizowana poprzez dwa podstawowe schematy: $(\lambda + \mu)$ i (λ, μ). Tutaj μ opisuje rozmiar populacji (równoważny liczbie rodziców), a λ liczbę potomków generowaną przez wszystkich rodziców w danej populacji. W strategii $(\lambda + \mu)$ jest generowanych λ potomków przez μ rodziców, a następnie μ najlepszych elementów jest wybieranych spośród $\lambda + \mu$ kandydatów. W strategii (λ, μ) jest wybieranych μ najlepszych osobników spośród λ potomków. Stąd warunkiem jest, aby $\lambda \geq \mu$.

Realizacja mutacji w strategiach ewolucyjnych następuje poprzez dodawanie liczby losowej o rozkładzie Gaussowskim do rodzica. Wówczas mutacja ma następującą postać :

$$x_i' = x_i + N_i(0, \sigma_i),$$

gdzie $N_i(0, \sigma_i)$ jest liczbą losową o rozkładzie normalnym ze średnią zero i odchyleniem standardowym σ_i , a n liczb losowych jest generowane niezależnie.

Ważnym parametrem mutacji Gaussowskiej jest odchylenie standartowe σ_i. Wybór tej wielkości ma istotny wpływ na zachowanie i przebieg strategii ewolucyjnej. Jednak jej optymalna wartość zależy zarówno od zadania, jak i jego rozmiaru. Powstała propozycja, aby σ_i była fragmentem osobnika i ewoluowała automatycznie razem z nim. Taka operacja nazywana jest samoadaptacją . Jest to jedna z różnic między strategiami ewolucyjnymi i algorytmami genetycznymi. W wielu implementacjach najpierw ewoluuje σ_i, a potem x_i jest mutowany za pomocą tego nowego σ_i .

Mutowanie wielu składowych niezależnie może być dla wielu problemów nieodpowiednie ze względu na to, że w zadaniu składowe mogą nie być zupełnie niezależne. Aby sprostać temu problemowi, powinna być dołączona kowariancja jako następna dodatkowa część (fragment) osobnika. Nie jest wyjaśniona sytuacja, kiedy taka samoadaptacja jest lepsza dla większości zadań, gdyż przestrzeń poszukiwań rośnie wykładniczo, gdy powiększamy rozmiar osobników. Rekombinacja w strategiach ewolucyjnych przyjmuje dwie główne postaci: rekombinacji dyskretnej i pośredniej. Rekombinacja dyskretna miesza losowo współrzędne dwóch osobników (wektorów) rodzicielskich. Dwoje rodziców $x = (x_1, x_2, ..., x_n)$ i $y = (y_1, y_2, ..., y_n)$ generuje potomków $x' = (x_1', x_2', \ldots, x_n')$ i $y' = (y_1', y_2', ..., y_n')$ w następujący sposób:

$$x_i' = \begin{cases} x_i & \text{z prawdopodobieństwem } p_r \\ y_i & \text{w przeciwnym razie} \end{cases} \tag{1.7}$$

Tutaj p_r oznacza prawdopodobieństwo rekombinacji, zaś y' tworzymy jako uzupełnienie x'. Rekombinacja pośrednia realizowana jest jako pewien typ uśredniania z parametrem wagi $\alpha \in (0, 1)$, która też może być generowana losowo lub ustalona. Dwoje rodziców $x = (x_1, x_2, \ldots, x_n)$ i $y = (y_1, y_2 \ldots, y_n)$ generuje potomków $x' = (x_1', x_2', \ldots, x_n')$ i $y' = (y_1', y_2' \ldots, y_n')$ w następujący sposób:

$$x_i' = x_i + \alpha(y_i - x_i)$$

Implementacja (w pseudokodzie) strategii ewolucyjnej (μ, λ):

1. Utwórz populację początkową z μ osobników i niech $k = 1$. Każdy osobnik jest parą liczb rzeczywistych $(x_i, \eta_i), \forall i \in \{1, ...\mu\}$, η w tym przypadku oznacza odchylenie standardowe σ;

2. Wyznacz dopasowanie dla każdego osobnika (x_i, η_i), $\forall i \in \{1, ..., \mu\}$ danej populacji.

3. Każdy z rodziców $(x_i, \eta_i), i = 1, ..., \mu$ tworzy średnio λ/μ potomków, tak, że utworzonych zostaje λ potomków:

$$\eta_k'(j) = \eta_i(j) \exp(\tau' N(0, 1) + \tau N_j(0, 1) \tag{1.8}$$

$$x_k'(j) = x_i(j) + \eta_i'(j) N_j(0, 1) \tag{1.9}$$

Tutaj $N(0, 1)$ oznacza jednowymiarową liczbę losową o rozkładzie normalnym ze średnią zero i odchyleniem standardowym jeden. Czynniki τ i τ' są pewnymi liczbami dobranymi empirycznie, jak $(\sqrt{2\sqrt{n}})^{-1}$ oraz $(\sqrt{2n})^{-1}$

4. Wyznacz przystosowanie każdego potomka (x'_i, η'_i), $\forall i \in \{1, ..., \lambda\}$.

5. Uporządkuj potomków $(x'_i, \eta'_i), \forall i \in \{1, ..., \lambda\}$ w porządku niemalejącym zgodnie z wartościami ich funkcji dopasowania i wybierz μ najlepszych potomków spośród λ osobników jako rodziców do następnej populacji.

6. Zatrzymaj, jeśli spełnione jest kryterium stopu; w przeciwnym przypadku $k := k + 1$ i przejdź do kroku 3.

1.3.2. Programowanie ewolucyjne

Było ono wprowadzone jako próba tworzenia sztucznej inteligencji i zostało zastosowane do ewolucji automatów skończonych. Programowanie ewolucyjne jest podobne do strategii ewolucyjnych dla zadań optymalizacji numerycznej. Korzysta ono z osobników w postaci wektorów liczb rzeczywistych oraz mutacji Gaussowskiej i samoadaptacji, jak poprzednio. Najistotniejszą różnicą między programowaniem ewolucyjnym i strategiami ewolucyjnymi jest rekombinacja i selekcja. Programowanie ewolucyjne nie korzysta z rekombinacji lub krzyżowania, lecz wykorzystuje losową konkurencję jako mechanizm selekcji. Oczywiście nie ma powodu– z algorytmicznego punktu widzenia– by programowanie ewolucyjne nie mogło mieć rekombinacji i by strategie ewolucyjne nie mogły mieć losowego schematu selekcji. Najbardziej znaczące różnice między programowaniem ewolucyjnym a strategiami ewolucyjnymi występują przy rekombinacji i selekcji [57].

Początki programowania ewolucyjnego i strategii ewolucyjnych różnią się. Programowanie ewolucyjne zostało zaproponowane jako symulacja inteligencji poprzez ewoluujący automat skończony, podczas gdy strategie ewolucyjne zajmowały się optymalizacją parametrów numerycznych. Zastosowanie rekombinacji do automatów skończonych w sposób poprawny było problematyczne.

Implementacja programowania ewolucyjnego:

1. Utwórz populację początkową z μ osobników i ustal $k = 1$. Każdy osobnik jest dany jako para wektorów rzeczywisto-liczbowych (x_i, η_i) $\forall i \in \{1, ..., \mu\}$, gdzie x_i są zmiennymi a η'_i są odchyleniami standardowymi mutacji Gaussowskiej.

2. Oceń stopień dopasowania każdego osobnika (x_i, η_i) $\forall i \in \{1, ..., \mu\}$ populacji.

3. Każdy rodzic $(x_i, \eta_i), i = 1, ..., \mu$, tworzy jednego potomka (x'_i, η'_i) w sposób następujący: dla $j = 1, ...n$,

$$\eta'_i(j) = \eta_i(j) exp(\tau' N(0, 1) + \tau N_j(0, 1) \tag{1.10}$$

$$x'_i(j) = x_i(j) + \eta'_i(j) N_j(0, 1) \tag{1.11}$$

   gdzie $N(0, 1)$ oznacza jednowymiarową liczbę losową o rozkładzie normalnym ze średnią zero i odchyleniem standardowym jeden. Czynniki τ i τ' są pewnymi liczbami dobranymi empirycznie; zalecane są $(\sqrt{2\sqrt{n}})^{-1}$ oraz $(\sqrt{2n})^{-1}$

4. Wyznacz przystosowanie każdego potomka (x'_i, η'_i), $\forall i \in \{1, ..., \lambda\}$.

5. Przeprowadź porównanie par w zbiorze będącym połączeniem zbiorów rodziców (x_i, η_i) i potomków (x'_i, η'_i), $\forall i \in \{1, ..., \mu\}$. Dobierz do każdego osobnika losowo jednostajnie q konkurentów spośród wszystkich rodziców i potomków. Przy każdym porównaniu osobnik, którego dopasowanie jest nie mniejsze niż konkurentów, wygrywa.

6. Wybierz μ pierwszych[1] osobników spośród $x_i, \eta_i)$ i (x'_i, η'_i), $\forall i \in \{1, ..., \mu\}$ jako rodziców do następnej populacji.

7. Zatrzymaj, jeśli spełnione jest kryterium stopu; w przeciwnym przypadku $k := k + 1$ i idź do 3.

1.3.3. Programowanie genetyczne

Jest to metoda zastosowania przeszukiwania ewolucyjnego w przestrzeni struktur drzewiastych, które mogą być interpretowane jako programy komputerowe w językach odpowiednich do modyfikacji przez mutację i rekombinację. Wykorzystuje się tu język LISP, a także inne języki i kod maszynowy.

1.4. Teoria obliczeń ewolucyjnych

Prace teoretyczne poświęcone algorytmom ewolucyjnym koncentrują się na trzech głównych zagadnieniach. Pierwszym jest analiza teoretyczna zbieżności i próba wyznaczenia wskaźnika zbieżności dla algorytmów genetycznych, strategii ewolucyjnych i programowania genetycznego. Drugim tematem jest badanie stopnia trudności zadania dla algorytmów ewolucyjnych. Głównym zagadnieniem jest badanie, jakie zadania są trudne dla algorytmów, a jakie są łatwe. Jeśli mamy charakterystykę zadania, która rozróżnia zadania łatwe i trudne, to możemy lepiej zrozumieć, kiedy i jak działają algorytmy ewolucyjne. Ma to olbrzymie znaczenie praktyczne towarzyszące rozważaniom teoretycznym. Skupiając się na tym zagadnieniu, badano zwodniczość algorytmów. Prace te próbowały scharakteryzować zadania, które są trudne do rozwiązania przez swoją zwodniczość. Trzeba podnieść problematyczność tego podejścia. Innym podejściem była analiza bloków budujących i teoria schematów, analiza oparta na funkcjach Walsha i krajobrazów dopasowania oraz korelacji odległości dopasowania. Wszystkie te metody tworzą pewien postęp w lepszym wyjaśnianiu, jak i kiedy algorytmy działają. Pomimo tych analiz, algorytmy genetyczne kryją jeszcze wiele tajemnic [76, 100, 91].

Trzecim zagadnieniem jest złożoność obliczeniowa algorytmów ewolucyjnych. Jest to jedna z najważniejszych dziedzin, gdzie równocześnie osiągnięto niewielki postęp. Algorytmy stosowane są do optymalizacji kombinatorycznej i numerycznej w duchu oryginalnej metody przeszukiwania i adaptacji. Zbadano wiele algorytmów i ich złożoność dla wielu problemów. Jednak nadal jest niejasne, kiedy algorytmy mogą być lepsze niż inne algorytmy aproksymacyjne lub heurystyczne przy kryterium najlepszej lub średniej złożoności czasowej. Brak jest konkretnych wyników dotyczących złożoności obliczeniowej algorytmów ewolucyjnych dla nietrywialnych zadań, zwłaszcza dla zadań kombinatorycznych.

1.4.1. Optymalizacja ewolucyjna

Optymalizacja ewolucyjna jest dziedziną najbardziej aktywnego stosowania obliczeń ewolucyjnych. Większość prac z optymalizacji ewolucyjnej zajmuje się optymalizacją numeryczną. Przy zastosowaniu algorytmów genetycznych do optymalizacji

[1] Najpierw uporządkuj wszytkich wygrywających według wartości ich dopasowania.

numerycznej wektory liczb rzeczywistych są zwykle kodowane dwójkowo (ciągi binarne). Próbowano różnych metod kodowania (kody Graya, kody delta). Jednak dotychczas jest problemem znalezienie najlepszego kodowania i nie wiadomo, kiedy konieczne jest przekształcenie liczb rzeczywistych w ciągi dwójkowe. Strategie ewolucyjne i programowanie ewolucyjne wykorzystują liczby rzeczywiste bezpośrednio i pomijają problem znalezienia odpowiedniego kodowania osobników. Istnieje pewna ilość prac porównujących reprezentację dwójkową algorytmów genetycznych i rzeczywistoliczbową, która jest wykorzystywana w programowaniu ewolucyjnym. Istnieje potrzeba dalszych studiów tego zagadnienia i poznania, dlaczego algorytm zachowuje się lepiej (lub gorzej) dla pewnych zadań. Algorytmy ewolucyjne stały się narzędziem rozwiązywania różnorodnych zadań optymalizacji kombinatorycznej [8].

1.4.2. Uczenie ewolucyjne

Uczenie ewolucyjne jest podejściem ewolucyjnym w uczeniu maszynowym. Najbardziej obiecującym jest zastosowanie podejścia ewolucyjnego w uczeniu się ze wzmocnieniem. W systemach klasyfikacyjnych używamy algorytmów genetycznych do generowania nowych klasyfikatorów poprzez mutację i krzyżowanie. Jako funkcje przystosowania traktujemy moc klasyfikatora. Algorytm genetyczny jest stosowany do klasyfikatora po pewnej liczbie cykli operacji celem lepszego przybliżenia jego mocy. Istnieją dwa główne podejścia do sytemów klasyfikujących: Metoda Michigan i metoda Pitts i uczenie koewolucyjne [13, 58].

Metoda Michigan i metoda Pitts

W podejściu Michigan każdy osobnik z populacji jest klasyfikatorem. W podejściu Pitts każdy osobnik w populacji reprezentuje zupełny układ klasyfikatorów. Cała populacja zawiera pewną liczbę konkurujących układów klasyfikujących.

Uczenie koewolucyjne

Koewolucja polega na równoczesnej ewolucji dwóch lub więcej środowisk ze sprzężonym wspólnym dopasowaniem. Uczenie koewolucyjne może przebiegać w dwóch formach. W pierwszej, dwie lub więcej populacji ewoluują w tym samym czasie. Dopasowane w jednej populacji zależy od osobników w innej populacji. Brak jest wymiany informacji (krzyżowania) między populacjami. Jest to koewolucja na poziomie populacji. Drugą formą koewolucji jest poziom indywidualny. Wprowadzona jest wówczas tylko jedna populacja. Dopasowanie jednego osobnika zależy od innych osobników tej samej populacji. Obydwie formy koewolucji mają dynamiczne otoczenie i dynamiczną funkcję wartościującą [99, 13].

1.5. Operatory przeszukiwania

Istnieje wiele operatorów przeszukiwania i ich lista nie jest zamknięta. Są one wykorzystywane w różnych algorytmach ewolucyjnych. Wiele z nich jest wyspecjalizowanych w rozwiązywaniu szczególnych zadań. Można jednak wyróżnić powszechnie używane, niejako klasyczne typy operatorów, nie pretendując do zupełności opisu [1, 16, 78, 84, 57].

1.5.1. Operatory rekombinacji

Istotą każdego operatora rekombinacji (krzyżowania) jest dziedziczenie informacji (genów) od dwojga (lub więcej) rodziców przez potomków. W większości przypadków operatory wykorzystują dwoje rodziców. Z wielu rodziców korzystamy tylko w specjalnych przypadkach.

Rekombinacja wektorów rzeczywistoliczbowych

Operatory takie są wykorzystywane w przekształcaniu wektorów liczb rzeczywistych. W strategiach ewolucyjnych rekombinacja jest niezależna od zmiennych i parametrów strategii. Natomiast może być ona różna dla zmiennych i dla parametrów.

Rekombinacja dyskretna

W tym przypadku wektor potomny posiada składowe pochodzące od dwóch lub więcej rodziców. Natomiast nie ma zmian żadnej składowej. Dla dwojga danych rodziców $x = (x_1, x_2, ..., x_n)$ i $y = (y_1, y_2, ..., y_n)$ osobniki potomne $x' = (x'_1, x'_2, ..., x'_n)$ i $y' = (y'_1, y'_2, ..., y'_n)$ powstają w następujący sposób:

$$x'_i = \begin{cases} x_i & \text{z prawdopodobieństwem } p_r \\ y_i & \text{w przeciwnym razie} \end{cases} \tag{1.12}$$

Tutaj p_r jest prawdopodobieństwem rekombinacji, zaś y' jest uzupełnieniem x'. W przypadku ogólnym y_i jest tworzony losowo $\forall i$ z rozkładem jednorodnym dla całej populacji. Liczba rodziców jest taka sama jak rozmiar populacji.

Rekombinacja pośrednia

W tym przypadku składowa potomka jest kombinacją liniową (średnią) odpowiednich składowych rodziców.

Dwoje rodziców $x = (x_1, x_2, ..., x_n)$ i $y = (y_1, y_2, ..., y_n)$ generuje potomków $x' = (x'_1, x'_2, ..., x'_n)$ i $y' = (y'_1, y'_2, ..., y'_n)$ w następujący sposób:

$$x'_i = x_i + \alpha(y_i - x_i) \tag{1.13}$$

z parametrem wagi $\alpha \in (0, 1)$, która też może być generowana losowo lub ustalona; może być ona różna dla każdego i [57].

Rekombinacja ciągów dwójkowych

Jest wiele sposobów realizacji rekombinacji, ale najpopularniejsze jest krzyżowanie k-punktowe i krzyżowanie jednostajne.

1.5.2. Krzyżowanie k-punktowe

Stosujemy je do stringów z dowolnego alfabetu. Dla dwojga danych rodziców o długości n i k liczb losowych $r_1, r_2, ..., r_k$ generowanych losowo, jednorodnie i bez powtórzeń pomiędzy 1 i $n - 1$, tworzymy potomka, biorąc elementy (segmenty rozdzielone przez $r_1, r_2, ..., r_k$) stringu (ciągu) przemiennie tak, że pierwszy element

jest z pierwszego rodzica, drugi z drugiego, trzeci z pierwszego itd.

Krzyżowanie jednostajne

Przy tym krzyżowaniu potomek jest generowany poprzez dobieranie każdego bitu z odpowiedniego bitu jednego z rodziców przemiennie. Innymi operatorami krzyżowania mogą być tasowanie, krzyżowanie w macierzach i krzyżowanie permutacyjne [95].

1.5.3. Operator mutacji

Mutacja ciągów binarnych polega najczęściej na losowej zmianie bitu z pewnym prawdopodobieństwem (niewielkim). To prawdopodobieństwo nazywamy prawdopodobieństwem mutacji. Zmiana bitu może być stosowana do ciągów dowolnego alfabetu.

Mutacja wektorów rzeczywistoliczbowych

Mutacja wektorów rzeczywistych bazuje na pewnych dobranych rozkładach prawdopodobieństwa takich, jak jednostajny, normalny (Gaussowski), Cauchy'ego:

$$x_i' = x_i + N_i(0, \sigma_i) \tag{1.14}$$

gdzie $N_i(0, \sigma_i)$ jest liczbą losową o rozkładzie normalnym ze średnią zero i odchyleniem standardowym σ_i, a n liczb losowych jest generowane niezależnie.

Jak wspomniano już w 1.3.1, ważnym parametrem mutacji Gaussowskiej jest odchylenie standardowe σ_i. Wybór odchylenia ma istotny wpływ na zachowanie i przebieg strategii ewolucyjnej. Jego optymalna wartość zależy zarówno od zadania, jak i od jego rozmiaru. W pewnych algorytmach wprowadzono σ_i jako fragment osobnika. Następnie ewoluuje ona automatycznie, razem z nim. Taka operacja nazywana jest samoadaptacją. Mutacja Cauchy'ego różni się od normalnej rozkładem generującym liczby losowe [53].

1.5.4. Selekcja

Procedura selekcji wyznacza prawdopodobieństwo wybrania osobnika do tworzenia potomków przez rekombinację i (lub) mutację. W celu wyszukiwania osobników lepiej dostosowanych, elementy o większym przystosowaniu powinny mieć duże prawdopodobieństwo selekcji, podczas gdy nieprzystosowane powinny być selekcjonowane z małym prawdopodobieństwem. Różne metody selekcji tworzą różne metody wyznaczania prawdopodobieństwa selekcji. Czasami nacisk selekcyjny ma wpływ na to, jak duże powinno być prawdopodobieństwo wyboru osobników dobrze dopasowanych w porównaniu z osobnikami niedopasowanymi. Wyższemu prawdopodobieństwu odpowiada mocniejszy nacisk selekcyjny.

Głównymi typami selekcji jest selekcja ruletkowa (proporcjonalna), rangowa i turniejowa.

Selekcja ruletkowa

Niech f_i oznacza przystosowanie i-tego osobnika. Wówczas prawdopodobieństwo selekcji osobnika dane jest przez:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{1.15}$$

Metoda ta może powodować trudności w pewnych przypadkach. Gdy populacja zawiera osobniki o wysokim dostosowaniu, ale nie najlepsze, wówczas mogą one szybko zdominować populację i uniemożliwią eksplorację innych potencjalnie lepszych osobników. Także w sytuacji, gdy osobniki w populacji mają zbliżone przystosowania, utrudnione jest znalezienie lepszych, gdyż prawdopodobieństwa selekcji są podobne. Trudności te można rozwiązać metodą skalowania dopasowania.

Selekcja rangowa

Metoda ta polega na uporządkowaniu osobników według ich wartości przystosowania i następnie wyznaczaniu prawdopodobieństwa selekcji na podstawie rangi (pozycji), a nie przystosowania. Prawdopodobieństwo selekcji nie zależy od przystosowania bezpośrednio a tylko pośrednio. Zachowując stały nacisk selekcyjny możemy unikać problemów występujących przy selekcji ruletkowej. Jest wiele różnych metod selekcji rangowej; jedną z nich przedstawia

$$P_i = \frac{i}{\sum_{j=1}^{n} j} \tag{1.16}$$

Selekcja turniejowa

Zarówno selekcja ruletkowa, jak i selekcja rangowa wykorzystują całą informację o populacji. Selekcja turniejowa korzysta tylko z informacji częściowej do wyznaczenia prawdopodobieństwa selekcji. Przykład selekcji turniejowej przytoczono w opisie programowania ewolucyjnego, inną jest selekcja Boltzmanowska opisana przez Golgberga [30].

Selekcja elitarna

W selekcji tej do następnej populacji zawsze jest kopiowany najlepszy osobnik, bez jakiejkolwiek modyfikacji. Może też być kopiowanych kilku najlepszych osobników do następnej populacji. Selekcja elitarna towarzyszy innemu rodzajowi selekcji [56, 53].

1.5.5. Twierdzenie o schematach

Twierdzenie o schematach jest jednym z pierwszych wyników teoretycznych podejmujących próbę opisania algorytmów genetycznych i wyjaśnienia ich działania. Jego sformułowanie wprowadza pojęcie schematu i wyznacza prawo rozpowszechniania się schematów wśród osobników populacji algorytmu, zapisanych w postaci kodu binarnego.

Twierdzenie o schematach Hollanda [44, 30, 66, 93] jest powszechnie uważane za podstawę wyjaśnienia mocy algorytmów genetycznych. Schematem jest wzorzec, który opisuje pewien ciąg binarny (podciąg) z elementami określonymi na pewnych pozycjach ciągu (stringu).

Dla przykładu rozważmy ciąg dwójkowy (binarny string) długości 6. Schemat $1**0*1$ opisuje zbiór wszystkich stringów o długości 6 z jedynką na pierwszej pozycji i szóstej oraz zerem na pozycji czwartej. Natomiast gwiazdka * jest dżokerem. Oznacza to, że pozycje druga, trzecia i piąta mogą mieć wartość zarówno zero, jak i jeden. Jako rząd schematu definiujemy liczbę ustalonych pozycji we wzorcu, podczas gdy długością definiującą (δH) jest odległość pomiędzy pierwszą i ostatnią wyspecyfikowaną pozycją. Rzędem $1**0*1$ jest trzy, a jego długością definiującą jest pięć. Dopasowaniem schematu jest średnie dopasowanie wszystkich ciągów pasujących do schematu. Dopasowanie schematu jest miarą wartości tak kodowanego rozwiązania zadania, wyznaczanego przez funkcję wartościującą zadania. Twierdzenie o schematach stanowi, że krótkie, niskiego rzędu schematy z lepszym dopasowaniem średnim przyrastają wykładniczo w kolejnych pokoleniach. Wyraża to równanie:

$$m(H, t+1) \geq \frac{m(H,t)f(t)}{a_t}[1-p] \tag{1.17}$$

gdzie $m(H,t)$ jest liczbą ciągów (stringów) należących do schematu H w pokoleniu t, natomiast $f(H)$ jest uzyskanym dopasowaniem schematu H i a_t jest wyznaczana jako średnie dopasowanie w pokoleniu t. Prawdopodobieństwo zniszczenia schematu p jest prawdopodobieństwem, że krzyżowanie lub mutacja zniszczą schemat H. Można to wyrazić wzorem:

$$p = \frac{\delta(H)}{l-1}p_c + o(H)p_m \tag{1.18}$$

gdzie $o(H)$ jest liczbą ustalonych pozycji, l jest długością kodu, p_m jest prawdopodobieństwem mutacji, a p_c jest prawdopodobieństwem krzyżowania. Stąd schematy o krótkiej długości definiującej $\delta(H)$ mają mniejsze szanse zniszczenia.

Twierdzenie uwzględnia małe, niezerowe prawdopodobieństwa, że string należący do schematu powstanie poprzez mutację pojedyńczego ciągu lub rekombinację dwóch ciągów, które poprzednio nie należały do schematu.

1.5.6. Hipoteza bloków budujących

Algorytmy genetyczne są dość łatwe do implementacji, lecz ich zachowanie jest trudne do wytłumaczenia, dlaczego prowadzą one tak często do sukcesu w generowaniu rozwiązań o wysokim dopasowaniu. Hipoteza Bloków Budujących (BBH) składa się z:

- Opisu mechanizmu adaptacji realizowanego poprzez rekombinację "bloków budujących", niskiego rzędu, o niewielkiej długości definiującej schematów o wysokim, średnim dopasowaniu.
- Hipotezy, że algorytm genetyczny realizuje adaptację przez pośrednie i efektywne wykorzystanie tych abstrakcyjnych mechanizmów adaptacji [30]. Krótkie niskiego rzędu o wysokim dostosowaniu schematy są próbkowane i zachowywane z ciągami o potencjalnie wyższym dopasowaniu w procesie selekcji. W ten sposób przez przetwarzanie poszczególnych schematów (bloków budujących) redukujemy złożoność naszego zadania, poprzez tworzenie ciągów o wysokiej częstości pojawiania się, tworzymy coraz lepsze ciągi (rozwiązania) spośród najlepszych rozwiązań cząstkowych w poprzednich próbkach [89, 66, 44].
- Hipoteza Bloków Budujących była też ostro krytykowana ze względu na brak teoretycznych analiz i uzasadnień oraz wyników eksperymentalnych potwierdzających ją.

Z teoretycznego punktu widzenia wiele wypowiedzi o Hipotezie Bloków Budujących nie ma podstaw teoretycznych, a w wielu wypadkach są one po prostu niespójne. Z eksperymentalnych danych wynika, że krzyżowanie jednostajne przewyższa krzyżowanie jedno i dwupunktowe dla wielu funkcji dopasowania badanych przez Syswerda [95]. Podsumowując te uwagi, Fogel stwierdził: "*ogólnie, krzyżowanie jednostajne generuje lepsze zachowanie aniżeli krzyżowanie dwupunktowe, które z kolei jest lepsze od jednopunktowego.* Wynik ten zaprzecza Hipotezie Bloków Budujących, ponieważ krzyżowanie jednostajne w sposób maksymalny niszczy schematy.

1.5.7. Problematyka rozprawy

Przedmiotem badania asymptotycznego zachowania trajektorii algorytmów genetycznych mogą być własności graniczne ich trajektorii. Jako główne narzędzie tych badań, w rozprawie będzie wykorzystana entropia procesu oraz wymiar fraktalny trajektorii. Celem badań będzie wyjaśnienie istoty działania binarnych algorytmów genetycznych i programów ewolucyjnych poprzez analizę zachowania trajektorii procesu iteracyjnego generowanego przez badane algorytmy genetyczne.

Efektem proponowanych badań będzie próba klasyfikacji algorytmów genetycznych i wyjaśnienie mechanizmów ich działania. Propozycja ta została opracowana przy założeniu, że pojęcia i metody teorii układów dynamicznych oraz dynamiki symbolicznej zastosowane do analizy algorytmów ewolucyjnych przyczyniają się do wyjaśnienia podstaw ich działania i pozwolą na wypracowanie zasad projektowania nowych algorytmów.

1.5.8. No Free Lunch

Słynne twierdzenie No Free Lunch Theorem [98] stwierdza, że nie istnieje lepszy lub gorszy algorytm optymalizacyjny dla wszystkich zadań. Algorytm działający lepiej na pewnej klasie zagadnień będzie gorszym dla innej klasy zagadnień. W związku z tym ustalenie odpowiedniości między algorytmem a zadaniem optymalizacyjnym jest ważne zarówno z punktu widzenia praktyki, jak i teorii algorytmów.

Pojawia się też problem, jak można wykorzystać już posiadaną wiedzę o sposobie zachowania algorytmu genetycznego na pewnej klasie zagadnień do prognozy jego zachowania na innej klasie oraz jak takie uogólnienie można mierzyć. Istotne stało się zrozumienie (wyjaśnienie) relacji pomiędzy tym, jak dobrze dany algorytm się zachowuje, a zagadnieniem optymalizacyjnym, które on rozwiązuje [98]. Zagadnieniem otwartym jest problem pomiaru efektywności algorytmu genetycznego dla danego problemu.

Twierdzenie No Free Lunch ustala, ze średnio, dowolny algorytm dla wszystkich zadań optymalizacyjnych zachowuje się tak samo i jest to spełnione dla dowolnej miary zachowania. Taką szczególną miarą zachowania algorytmu jest entropia jego trajektorii.

Natomiast twierdzenie No Free Lunch (NFL) nie jest spełnione w przypadku algorytmów koewolucyjnych. To stawia pytanie, czy istnieje algorytm najlepszy w tej klasie i jaką wartość przyjmuje dla niego entropia.

Postać graniczna operatora Markowa opisującego algorytm genetyczny pozwala na postawienie paru hipotez.

Istotne są własności graniczne operatora a nie populacja, nawet jeśli jest "optymalna". Prawdopodobieństwo otrzymania danej populacji w stanie granicznym

jest stałe i niezależne od populacji wyjściowej, czy prawdopodobieństwo innych elementów, a zatem i populacji, jest tak małe , że następuje przybliżenie (obcięcie) prawdopodobieństw do zera na skutek zaokrągleń?.

Permutacje w zbiorze wartości funkcji przystosowania reprezentują zarówno algorytmy, jak i funkcje. Rozróżnić należy pomiędzy algorytmem a jego zachowaniem. Istnieje nieskończenie wiele algorytmów, lecz jeśli zbiór wartości jest skończony, to istnieje tylko skończona liczba zachowań. Można klasyfikować algorytmy z dokładnością do permutacji punktów przestrzeni przeszukiwania generowanej przez algorytm. Wtedy mogą być równoważne algorytmy przy różnej entropii, gdyż mogą być różne prawdopodobieństwa poszczególnych stanów, przy zachowaniu tej samej kolejności punktów. Czy te różnice wynikają z tego, że dowód Ornsteina dotyczy przesunięć Bernoulliego ciągów nieskończonych, a więc zbiorów ciągłych, a w algorytmach rozważamy zbiory dyskretne.

W optymalizacji (informatyce) istnieją warunki powodujące, że wyniki wszystkich procedur rozwiązujących pewien typ zadań są statystycznie identyczne. Sposób opisu takich warunków, wprowadzony został przez D. H. Wolperta i W.G.Macready [98] w związku z problematyką przeszukiwania i optymalizacji i został określony jako NFL. Wcześniej, korzystając z innej terminologii, zagadnienie NFL zostało przedstawione w zadaniach uczenia maszynowego przez C. Schaffera [77].

Z formalnego punktu widzenia NFL ma miejsce, gdy rozkład prawdopodobieństw dla występujących zadań jest taki, że wszystkie sposoby rozwiązywania dają wyniki o tym samym rozkładzie. W przypadku przeszukiwania przykładowe zadania to funkcja wartościująca, a wynikiem jest ciąg wartości otrzymanych w procesie oceny proponowanych rozwiązań z dziedziny funkcji. Przy typowej interpretacji wyników, przeszukiwanie jest procesem optymalizacyjnym. NFL w przeszukiwaniu występuje wtedy i tylko wtedy, gdy rozkład wartości funkcji wartościującej jest niezmienniczy względem permutacji punktów przestrzeni potencjalnych rozwiązań. Warunek ten nie jest spełniony dokładnie w praktyce, lecz prawie wszędzie istnieje domniemanie, że NFL jest w przybliżeniu spełnione.

Wiele zadań obliczeniowych jest rozwiązywanych metodą poszukiwania dobrego rozwiązania w przestrzeni dopuszczalnych rozwiązań. Przepis, w jaki sposób wybierać kolejne potencjalne rozwiązania do oceny, nazywamy algorytmem przeszukiwania. W szczególności różne algorytmy przeszukiwania mogą dawać różne wyniki, lecz wobec wszystkich zadań są one nierozróżnialne. Wynika to z tego, że jeśli algorytm uzyskuje lepsze wyniki dla pewnych zadań, to odbywa się to kosztem gorszych rezultatów dla innych zadań. W tym znaczeniu w przeszukiwaniu "nie ma nic za darmo". Zazwyczaj przeszukiwanie jest interpretowane jako pewien rodzaj optymalizacji i to prowadzi do stwierdzenia, że w optymalizacji spełnione jest NFL.

Twierdzenie NFL stanowi, że "dwa dowolne algorytmy są równoważne, jeśli ich zachowanie jest uśrednione względem wszystkich możliwych zadań". Stąd NFL sugeruje, że dopasowanie algorytmu do zadania daje wyższe średnie zachowanie, niż stosowanie wybranego ustalonego algorytmu do wszystkich zadań. Igel, Toussaint i English [37, 38, 19, 22] ustalili ogólne warunki, dla których NFL jest spełnione. Jeśli jest to (fizycznie) możliwe, nie jest to koniecznie spełnione dokładnie. Droste, Jansen, i Wegener [19] dowiedli twierdzenia, które interpretują, że w praktyce NFL jest spełnione prawie wszędzie.

Rozważmy sytuację, gdy praktyk optymalizacji staje przed zadaniem opracowania dobrego algorytmu. Posiadając pewną wiedzę o genezie zadania, praktyk może wykorzystywać wiedzę dobierając algorytm, który będzie dobrze rozwiązywał zadanie.

Jeżeli jednak praktyk nie potrafi wykorzystać wiedzy lub jej poprostu nie posiada, to można postawić pytanie, kiedy (czy) pewien algorytm generalnie przewyższa inne dla rzeczywistych zadań. Autorzy NLF theorem "prawie wszędzie" stwierdzają, że odpowiedź jest (istotnie) przecząca, lecz zostawiają pewną swobodę interpretacji, kiedy twierdzenie dotyczy praktyczych zadań [19].

Zadanie jest, zasadniczo, funkcją (oceniającą) przyporządkowującą proponowanemu rozwiązaniu jej wartości. Algorytm przeszukiwania przyjmuje fukcję wartościującą jako wejście i ocenia kolejno proponowane rozwiązania. Wynikiem działania algorytmu jest ciąg otrzymanych wartości funkcji oceny. Wolpert i Macready przyjmują, że algorytm nigdy nie ocenia powtórnie danego rozwiązania oraz że jakość zachowania algorytmu jest mierzona na wyjściu. Dla prostoty pomijamy losowość algorytmu. Przy tych założeniach, przebieg algorytmu po wszystkich możliwych zadaniach generuje wszystkie możliwe wyniki dokładnie raz. Ponieważ zachowanie algorytmu jest oceniane, mierzone na wyjściu, algorytm jest nierozróżnialny w tym, jak często osiąga on dany poziom zachowania (jakości).

Pewne miary skali zachowania wskazują, jak dobrze algorytm wykonuje optymalizację funkcji oceny. Wspólną miarą zachowania jest ostatni indeks ostatniej wartości w ciągu wyjściowym. Jest to liczba ewaluacji niezbędna do minimalizacji funkcji oceny. Dla pewnych algorytmów czas niezbędny do wyznaczenia minimum jest proporcjonalny do liczby ewaluacji. Oryginalne twierdzenie NFL zakłada, że wszystkie funkcje oceniające mają jednakowe prawdopodobieństwo zostania argumentem algorytmu przeszukiwania. Dlatego wyjaśniono, że NFL jest spełnione wtedy i tylko wtedy, gdy przestawienie funkcji oceny nie ma wpływu na prawdopodobieństwo ich wykorzystania. Warunek ten nie jest spełniony dokładnie, lecz dla NFL istotą jest stopień spełnienia, a nie postawa: wszystko albo nic. Jeżeli warunki NFL spełnione są w przybliżeniu, to wszystkie algorytmy dają w przybliżeniu ten sam rezultat dla wszystkich funkcji oceny. I to jest uzasadnieniem, że w praktyce NFL jest "prawie wszędzie" spełnione.

Zbiór wszystkich funkcji oceny jest $Y^{\mathcal{X}}$, gdzie $\mathcal{X}$ jest skończoną przestrzenią rozwiązań a Y jest skończonym zbiorem. Zbiorem wszystkich permutacji $\mathcal{X}$ jest J. Zmienna losowa F ma rozkład na $Y^{\mathcal{X}}$ z prawdopodobieństwem $\forall j \in J, F_j$ jest zmienną losową o rozkładzie w $Y^{\mathcal{X}}$, z $pr\{F_j = f\} = pr\{F = f_{j-1}\} \forall f \in Y^X$. Niech $a(f)$ oznacza wynik (wyjście) algorytmu przeszukiwania przy wejściu f. Jeżeli $a(F)$ i $b(F)$ mają ten sam rozkład dla wszystkich algorytmów przeszukiwania a i b, wówczas F ma rozkład spełniający NFL. Warunek ten jest spełniony wtedy i tylko wtedy, gdy F i F_j mają ten sam rozkład $\forall j \in J$. Innymi słowy, NFL jest spełnione wtedy i tylko wtedy, jeżeli rozkład dla funkcji oceny jest niezmienniczy względem permutacji przestrzeni rozwiązań.

Wolpert i Macready sformułowali dwa podstawowe twierdzenia NFL. Pierwsze dotyczy funkcji ustalonej, która nie zmienia się podczas przeszukiwania a drugie dotyczy funkcji, która może się zmieniać.

Twierdzenie 1.5.1 *1: Dla dowolnej pary algorytmów a_1 i a_2*

$$\Sigma_f P(h_m^y | f, m, a_1) = \Sigma_f P(h_m^y | f, m, a_2).$$

gdzie f jest funkcją oceny a h_m^y jest ciągiem jej wartości po m krokach algorytmu. Twierdzenie NFL możemy interpretować stwierdzając, że "uogólniona uniwersalna" strategia optymalizacyjna jest nierealizowalna (niemożliwa) i jedyną możliwością,

aby jedna strategia przewyższała inną jest dostosowanie jej do specyfiki rozwiązywanego zadania. Jednak algorytm może przewyższać inny dla danego zadania, nawet jeśli nie jest on dobrany specjalnie dla tego zadania. Jest to możliwe, gdy obydwa algorytmy należą do grupy najgorszych dla danego zadania. Wolpert i Macready rozwinęli metodę porównania algorytmu i zadania . Stąd stwierdzenie, że algorytm jest lepiej dopasowany do zadania niż inny nie oznacza, że algorytm jest dobrze dostosowany do zadania.

1.5.9. Algorytmy koewolucyjne a NFL

.

Wolpert i Macready dowiedli [99] istnienia Free Lunches w optymalizacji koewolucyjnej. Ich analiza obejmuje zadanie gry ('self-play'). W zagadnieniu tym gracze współpracują, aby wygenerować championa (zwycięzcę), który pokona jednego lub kilku przeciwników w kolejnych grach z wieloma uczestnikami. Wówczas funkcją oceny jest wyłonienie dobrego gracza, bez innej funkcji oceny. Algorytm korzysta z gracza i jakości jego gry dla wyłonienia lepszego gracza. Najlepszy gracz spośród wszystkich uwzględnionych w algorytmie jest zwycięzcą (championem). Wolpert i Macready pokazali, że pewne algorytmy koewolucyjne są generalnie lepsze od innych algorytmów wyłaniania najlepszego gracza. Wyłanianie najlepszego poprzez auto-grę jest interesujące w algorytmach ewolucyjnych i teorii gier. Wynik nie ma zastosowania do koewolucji środowisk przyrodniczych biologicznych, gdyż w nich nie wyłania się championa.

Twierdzenie NFL sugeruje, że dopasowanie algorytmu do zadania prowadzi do lepszego średniego zachowania aniżeli korzystanie z ustalonego algorytmu do wszystkich zadań. Na mocy tego wybór najlepszego algorytmu może być poprawnie postawiony jedynie w kontekście zadania optymalizacyjnego. Aby podjąć to zagadnienie, można postawić pytanie, czy istnieje najlepszy algorytm genetyczny dla dostatecznie szerokiej klasy zadań optymalizacyjnych, który zawsze generuje rozwiązanie optymalne dla tej dostatecznie szerokiej klasy zadań. Ponadto powstaje pytanie, czy możliwe jest dobranie takich operatorów selekcji i mutacji, które są najlepsze dla danej klasy zagadnień. Odpowiedź nie jest pozytywna.

Fakt ten postuluje konieczność poszukiwania algorytmów dostosowanych do zadań. Większość metod badawczych algorytmów genetycznych jako podstawę przyjmuje podejście probabilistyczne, gdzie główną metodą jest badanie statystycznych własności ciągów populacji.

W przedstawianej rozprawie nie tyle odchodzimy od podejścia statystycznego, co podejmujemy wykorzystanie metod układów dynamicznych. Sądzimy, że w tym przypadku celowe jest wykorzystanie metod dynamiki symbolicznej razem z metodami analogii pomiędzy zachowaniem algorytmu genetycznego a dynamiką przekształceń jednowymiarowych.

Praca jest próbą wprowadzenia rozszerzenia metod analizy algorytmów genetycznych o metody badań jakościowych układów dynamicznych. Wprowadzenie metod i wyników jakościowej analizy układów dynamicznych, zwłaszcza analizy układów jednowymiarowych, może przybliżyć nas do wyjaśnienia podstaw działania algorytmów i ich własności.

1.6. Izomorfizm

Jednym z podstawowych zagadnień, które stawia NFL, jest zagadnienie, jak należy dobierać algorytm do zadania optymalizacyjnego, gdyż nie istnieje najlepszy uniwersalny algorytm optymalizacyjny. Dobór algorytmu do zadania jest sprawą doświadczenia i praktyki. Ale nie istnieje definicja, a tym bardziej teoria praktyki. Mimo to podejmowane są próby teoretycznego opracowania zadania doboru algorytmu i problemu optymalizacyjnego. My także podejmiemy to zagadnienie. Z zagadnieniem doboru algorytmu i zadania związany jest problem równoważności algorytmów. Jego badanie prowadzić będziemy na podstawie teorii układów dynamicznych. Podstawowym pojęciem jest izomorfizm transformacji, a taką transformacją jest algorytm genetyczny. Podejmiemy więc analizę problemu równoważności algorytmów genetycznych.

1.6.1. Klasyfikacja

Podstawowym narzędziem badań będzie badanie entropii procesu (trajektorii algorytmu). Entropia będzie tu więc traktowana jako statystyka trajektorii algorytmu genetycznego oraz jako wskaźnik klasyfikacji [62]. Wyznaczenie entropii na podstawie definicji jest, jak już pisaliśmy, trudne. Doświadczalne wyznaczenie entropii wymagałoby wielokrotnego uruchomienia algorytmu i na podstawie uzyskanych wyników wyznaczenia prawdopodobieństw przejść pomiędzy stanami, które posłużyłyby do wyznaczenia entropii. Zatem bezpośrednie wyznaczenie entropii jest nie tylko złożone, lecz i niedokładne.

Dlatego do wyznaczenia entropii procesu można skorzystać z Twierdzenia Lempel-Ziv oraz opracowanych na jego podstawie programów kompresji. Programy te będą wyznaczały entropię na podstawie słów (bloków) o ustalonej długości oraz entropię resztkową *(ang. excess entropy)* oraz inne rodzaje entropii. Pozwoli to na rozróżnianie przypadków, w których klasyczne pojęcie entropii nie daje zadowalających wyników.

Wykorzystując taką metodę pomiaru entropii, przeprowadzone zostanie badanie entropii algorytmów w zależności od parametrów algorytmu takich, jak prawdopodobieństwo mutacji, krzyżowania i selekcji, wielkości populacji, typu algorytmu [21, 14] itd.), typu selekcji i jej parametrów.

Możliwe jest także badanie zmienności przebiegu entropii w kolejnych fazach realizacji algorytmu genetycznego. Badanie zmienności entropii związane jest z zagadnieniem zbieżności algorytmu, gdyż w końcowej fazie działania algorytmu należy oczekiwać, że entropia będzie malała do pewnej ustalonej wielkości [62]. Badania obejmą też zmienność entropii w zależności od zmian średniego przystosowania populacji. Badanie entropii i wymiaru fraktalnego trajektorii będziemy prowadzić w przestrzeni argumentów zadania, w przestrzeni kodowej oraz w przestrzeni wartości funkcji dopasowania.

Dalsze badanie to ustalenie, na ile algorytmy genetyczne są wrażliwe na uwarunkowania takie, jak nałożenie lub odrzucenie więzów oraz innych warunków dodatkowych. Istotnym problemem poddanym badaniom będzie także zagadnienie określania stopnia trudności funkcji przystosowania.

1.6.2. Problematyka izomorfizmu

Przyjmijmy podstawowe definicje. Niech X, Y będą niepustymi zbiorami $\alpha : X \to Y$ jest przekształceniem, jeśli każdemu elementowi $x \in X$ jest przyporządkowany element $y = \alpha(x) \in Y$.

Jeśli $\alpha : X \to Y$, $\beta : Y \to Z$ są przekształceniami, to złożeniem tych przekształceń jest przekształcenie $\varphi : X \to Z$ zdefiniowane wzorem $\varphi = \beta(\alpha(x))$ dla $x \in X$ i oznaczane symbolem $\varphi = \beta \circ \alpha$ lub $\varphi = \beta\alpha$.

Niech $i_x : X \to X$ będzie przekształceniem tożsamościowym zbioru X, tzn. $i_x(x) = x \ \forall x \in X$.

Inne spojrzenie $i_x : X \to X$ jest przekształceniem tożsamościowym, jeśli dla dowolnych zbiorów Y, Z oraz przekształceń $\alpha : X \to Y$ oraz $\beta : Y \to Z$ zachodzą równości: $\alpha i_x = \alpha$ i $i_y\beta = \beta$ (kategoryjna definicja tożsamości).

Jeśli przekształcenie $\alpha : X \to Y$ jest różnowartościowe oraz dla każdego elementu $y \in Y$ istnieje element $x \in X$ taki, że $y = \alpha(x)$, to przyporządkowując elementowi $y \in Y$ jedyny element $x \in X$, spełniający również $y = \alpha(x)$, określamy przekształcenie $\beta : Y \to X$, które nazywamy przekształceniem odwrotnym do przekształcenia $\alpha : X \to Y$.

Inna definicja: dla $\alpha : X \to Y$ przekształcenie $\beta : Y \to X$ jest odwrotne, jeśli: $\beta\alpha = i_x$ oraz $\alpha\beta = i_y$.

Piszemy $\beta = \alpha^{-1}$

Grupa przekształceń zbioru X to dowolny zbiór przekształceń $\alpha : X \to X$ spełniający:

1. jeśli $\alpha, \beta \in G$, to $\beta\alpha \in G$,
2. $i_x \in G$,
3. jeśli $\alpha \in G$, to $\alpha^{-1} \in G$

Półgrupa to spełnienie tylko (1); jeśli dodatkowo (2) to półgrupa z tożsamością.

Niech X ma pewną strukturę algebraiczną, tzn. $\{X', J_x\}$, gdzie J_x jest zbiorem operacji, funkcji określonych dla elementów z X.

Niech $\{Y, J_y\}$ będzie innym, niepustym zbiorem ze strukturą algebraiczną J_y.

Przekształcenie $\alpha : X \to Y$ nazywamy homomorfizmami, jeśli α zachowuje struktury algebraiczne tych zbiorów, tzn. $\alpha(J_x) = J_y$.

Przekształcenie $\alpha : X \to Y$ jest izomorfizmem, jeśli istnieje $\alpha^{-1} : Y \to X$, które jest homomorfizmem.

Abstrakcyjną algebrą nazywamy $\{X, J_x\}$, gdzie J_x zawiera zbiór działań na X, w szczególności zawiera działanie wieloargumentowe, tzn. $J_x \ni f : X^n \to X$, to f nazywamy działaniem n-argumentowym na zbiorze X. Jeśli $X = \mathbf{R}$, to działaniem dwuargumentowym na X jest dodawanie. Jeśli J_x jest skończony, to piszemy $\{X, f_1, \ldots, f_n\}$, wtedy homomorfizm $\alpha : X \to Y$ jest określony poprzez:

$\alpha : X \to Y$, $(Y, h_1, \ldots, h_p)$ oraz $\forall f_i \in J_x$, $h_i = \alpha$, $(f_i) \in J_y$, $k \leq p$, f_i-k-elementowa operacja g_n.

Spełniona jest przemienność tzn.

$$\forall (x_1, \ldots, x_n), \quad \alpha(f_i(x_1, \ldots, x_n)) = h_i(\alpha(x_1), \ldots, \alpha(x_n)) \,. \tag{1.19}$$

Jednym z zasadniczych zagadnień zarówno teorii algorytmów genetycznych, jak i praktycznego ich wykorzystania, jest pytanie, kiedy dane dwa (lub więcej) algorytmy

są równoważne. Równoważność rozumiana jest jako własność algorytmu (przekształcenia generującego przeszukiwanie dziedziny zadania optymalizacyjnego) w probabilistycznie równoważny sposób. Ta równoważność prowadzi nas do izomorfizmu przekształceń zachowujących miarę i warunków, kiedy takie dwa przekształcenia możemy uważać za takie same (równoważne, izomorficzne lub sprzężone). Kwestia, kiedy dwa (lub więcej) zachowujące miarę przekształcenia są izomorficzne lub sprzężone poza zbiorami miary zero (małymi), jest badana za pomocą niezmienników (invariantów) takiego izomorfizmu lub sprzężenia. Niezmienniki takie mogą być dwóch typów. Pierwszym jest posiadanie lub nieposiadanie pewnej własności przez obydwa przekształcenia. Drugim typem niezmienników jest przypisanie przekształceniom (transformacjom) pewnego obiektu (obiektów) matematycznego (np. liczby) do każdego przekształcenia w ten sposób, że każde przekształcenie zachowujące miarę, z pewnej klasy, ma ten sam obiekt w swojej grupie. Aby taki obiekt (niezmiennik) miał zastosowanie, powinien być obliczalny dla interesującej nas klasy przekształceń.

Znane są dwa inwarianty tego typu.

Jednym typem jest grupa wartości własnych przekształceń (zachowujących miarę). Wyznacza ona pewną podgrupę. Przekształcenia zachowujące miarę mają tę samą grupę wartości własnych. Dla zbioru wszystkich ergodycznych przekształceń zachowujących miarę taki niezmiennik jest zupełny. Oznacza to, że jeśli dwa przekształcenia zachowujące miarę z dyskretnym spektrum mają tę samą grupę wartości własnych, to są one izomorficzne.

Drugim niezmiennikiem jest entropia. Entropia przypisuje każdemu przekształceniu zachowującemu miarę T nieujemną liczbę $h(T)$ i jeśli T_1 jest izomorficzne z T_2, to $h(T_1) = H(T_2)$. Dowiedzione przez D.S. Ornsteina [61, 25, 26, 96] twierdzenie stanowi, że w zbiorze wszystkich przesunięć Bernoulliego entropia jest zupełnym niezmiennikiem, co oznacza, że dwa przesunięcia Bernoulliego o tej samej entropii są izomorficzne. Wynik ten był dla nas inspiracją do prowadzenia badań.

-

ROZDZIAŁ 2

Model Markowa algorytmów genetycznych

W pracy wprowadzamy tylko podstawowe i niezbędne - dla zrozumienia niniejszych wyników - pojęcia z algorytmów genetycznych nie wdając się w dokładne wyjaśnienia. Dlatego odsyłamy dociekliwego czytelnika do artykułu [45] oraz przeglądowego Rowe [68] i łatwo dostępnych książek w języku polskim, które na początku wprowadzają od podstaw ideę algorytmów genetycznych, a następnie omawiają bardziej szczegółowo zagadnienia ważne dla tego tematu. Najważniejsze z nich to książki Michalewicza [56], Goldberga [30] i Cytowskiego [15]. Bardziej zaawansowane podejście z punktu widzenia pewnych układów dynamicznych przynoszą pozycje Vose [100] oraz polskojęzyczna monografia Schaefera [75]. Większość oznaczeń pochodzi właśnie z pozycji [100, 68].

Głównym celem tego rozdziału pracy jest badanie zachowania asymptotycznego SGA i niezależności rozkładu granicznego od populacji początkowej. Sądzimy, że zmiana kolejności sekwencji na mutacja–selekcja nie wpływa na prawdziwość naszych wyników, jeśli istnieje dla niej zależność analogiczna do (2.17), opisująca prawdopodobieństwo uzyskania wskazanej populacji, gdy startuje się z danej populacji początkowej. Egzystencjalnie zmiana nie ma wpływu, natomiast wyniki konkretne mogą być absolutnie różne.

2.1. Prosty algorytm genetyczny jako układ dynamiczny

Dzięki dobrze rozwiniętym metodom układów dynamicznych możemy się pokusić o przedstawienie podstawowych koncepcji algorytmów genetycznych w tym języku. Dla dobrego przedstawienia wszystkich wyników należy rozszerzyć listę potrzebnych dla tego celu pojęć, dodając pojęcia charakterystyczne dla algorytmów genetycznych.

Wśród badaczy zajmujących się algorytmami ewolucyjnymi (a w szczególności genetycznymi) przyjęto używać terminologię z biologii. I tak przetwarzany w danej iteracji (w danym kroku) multizbiór jest nazywany populacją (ang. *population*), rozwiązania należące do przestrzeni potencjalnych rozwiązań nazywane są osobnikami (ang. *individual*) lub fenotypami, a zamiast o kroku obliczeń mówi się o pokoleniu (ang. *generation*). Nowe osobniki, zwane potomkami (ang. *offspring*), są otrzymywane z osobników z poprzedniego pokolenia, zwanych rodzicami (ang. *parents*) w wyniku działania operacji selekcji (ang. *selection*). Elementy przestrzeni kodowej nazywane są chromosomami (ang. *chromosomes*), ze względu na ich podobieństwo do chromosomów składających się z cząsteczek kwasu DNA lub genotypami.

Wyróżnione funkcjonalnie fragmenty chromosomu nazywane są genami (ang. *gene*). Operatory podstawowe to *krzyżowanie* (ang. *crossover*), jeżeli dokonują wymiany informacji między osobnikami, natomiast *mutacja* (ang. *mutation*) modyfikuje pojedynczego osobnika. Używana jest często zamiennie funkcja jakości, czy funkcja celu, zaś jej odpowiednik, przetransformowany do przestrzeni kodowej to *funkcja przystosowania* (ang. *fitness function*), a wartość funkcji przystosowania dla konkretnego osobnika nazywana jest jego przystosowaniem. Funkcja przystosowania wraz z topologią przestrzeni kodowej definiowaną przez wybrane operatory jest nazywana krajobrazem przystosowania (ang. *fitness landscape*).

2.2. Zbiór możliwych populacji

Prosty algorytm genetyczny (ang.*simple genetic algorithm*) (SGA) jest modelem binarnych algorytmów genetycznych BGA [45], w których stosuje się selekcję proporcjonalną, wykorzystującą prawdopodobieństwo wyznaczone poprzez funkcję przystosowania, punktową mutację i standardowy algorytm krzyżowania, określany dla dwóch osobników.

W algorytmie genetycznym składnikiem zależnym od rozwiązywanego zadania, oprócz funkcji przystosowania, jest sposób kodowania elementów przestrzeni rozwiązań w chromosomy. Wtedy działanie algorytmu jest prawie niezależne od zadania, a wprowadzane operatory krzyżowania i mutacji (zwane w [45] *operatorami przemieszczenia*) działają nie na rozwiązaniach, ale na elementach przestrzeni kodowej.

W binarnym algorytmie genetycznym działamy na elementach przestrzeni kodowej, która jest obrazem przestrzeni rozwiązań podległej pewnej operacji kodowania (binarnego). Elementy przestrzeni kodowej są tutaj reprezentowane przez binarne chromosomy, które mają tę samą długość l, a więc zbiorem wszystkich chromosomów, czyli przestrzenią kodową Z, jest zbiór $Z = \{0,1\}^l$. Możemy napisać Z jako zbiór $\{z_0, ..., z_{s-1}\}$, gdzie $s = 2^l$. W dalszej części naszego rozdziału, dla prostoty, zamiast terminu *element przestrzeni kodowej* niekiedy będziemy używać terminu *komórka*.

Populację, czyli skończony multizbiór o rozmiarze r, zwanym rozmiarem populacji (ang. *PopSize*), który składa się z pewnej liczby tych samych kopii elementów przestrzeni kodowej, opisujemy jako (właściwie - utożsamiamy z) uporządkowaną s-tkę liczb wymiernych, ułamków, przy czym każdy element tej s-tki reprezentuje względną liczbę kopii elementu w populacji do liczby wszystkich elementów multizbioru. Innymi słowy, jeśli a_k jest liczbą kopii elementu z_k w populacji o rozmiarze r, to na k-tym miejscu w p wystąpi ułamek:

$$p_k = \frac{a_k}{r} \ , \text{ natomiast } p = (p_0, ..., p_{s-1}) \ , \tag{2.1}$$

przy czym

$$\sum_{k=0}^{s-1} p_k = 1 \ . \tag{2.2}$$

Tę s-tkę ułamków p odpowiadającą populacji nazywamy *wektorem populacji*. Czasami dla skrótu samo p będziemy nazywać populacją.

Własność (2.2) pozwala traktować poszczególne współrzędne populacji p jako prawdopodobieństwa występowania danego elementu z przestrzeni kodowej w populacji. Tym samym, ze względu na (2.2), p staje się wektorem probabilistycznym

(por.[52]). Dalsze uszczegółowienia tego wątku można znaleźć w pozycji [100] oraz artykule przeglądowym [68].

Warto rozróżnić populację od wektora populacji. Konkretna realizacja algorytmu genetycznego operuje na generowanych populacjach, choć przejście między populacjami jest losowe. Wektor populacji, natomiast, jest wektorem probabilistycznym i wokół niego skupia się nasze zainteresowanie.

Głównym celem naszej analizy, dotyczącej procesu losowego, a nie jego konkretnej realizacji, jest wyznaczenie własności asymptotycznych tego procesu, a w szczególności rozkładu granicznego wektora probabilistycznego.

Może warto dla przyszłych rozważań, szczególnie w punkcie poświęconym operatorowi krzyżowania, określić zbiór wszystkich możliwych populacji Λ przez

$$\Lambda = \{p \in \mathbf{R}^s : \forall k,\ p_k \geq 0,\ \text{jest wymierne oraz } \sum_{i=0}^{s-1} p_i = 1\ \}. \tag{2.3}$$

W tej definicji nie dopuszczamy, aby współrzędne wektora p przyjmowały dowolne, nawet niewymierne (rzeczywiste) wartości. Jednak, gdy przyjmiemy, że rozmiar populacji dąży do nieskończoności, zbiór możliwych populacji staje się gęsty w następującym zbiorze

$$\bar{\Lambda} = \{x \in \mathbf{R}^s : \forall k,\ x_k \geq 0,\ \text{oraz } \sum_{i=0}^{s-1} x_i = 1\ \}. \tag{2.4}$$

To oznacza, że każdy element sympleksu $\bar{\Lambda}$ może być dowolnie blisko rzeczywistej populacji przez odpowiednie zwiększenie rozmiaru populacji.

2.3. Operator selekcji

Istotnym zadaniem przy badaniu dynamiki algorytmu genetycznego jest wyznaczenie prawdopodobnego rozkładu(zawartości) następnej populacji, gdy mamy daną aktualną populację i operatory genetyczne oraz ich charakterystyczne parametry. Zaczniemy od operatora selekcji.

Dana jest populacja $p = (p_0, ..., p_{s-1})$ oraz funkcja przystosowania $f : Z \to \mathbf{R}^+$, która jest złożeniem funkcji jakości ϕ z funkcją kodującą oraz, jeśli to było konieczne, z innymi funkcjami przekształcającymi funkcję jakości w postać funkcji wymaganych przez rozpatrywany problem własności, np. nieujemności czy ograniczenia od dołu, itp. Zakładając stosowanie selekcji proporcjonalnej (por. [45, 56]) możemy wyznaczyć prawdopodobieństwo, że element z_k wystąpi w następnej populacji

$$\frac{f(z_k)p_k}{\bar{f}(p)}, \tag{2.5}$$

gdzie $\bar{f}(p)$ jest średnim przystosowaniem populacji p wyznaczonym przez

$$\bar{f}(p) = \sum_{k=0}^{s-1} f(z_k)p_k. \tag{2.6}$$

To pozwoli na wyznaczenie nowej s-tki (wektora) q składającej się z tych prawdopodobieństw. W tym celu określmy macierz diagonalną $\boldsymbol{S}$ o wymiarze s, w której na głównej przekątnej występują wartości funkcji kolejnych elementów przestrzeni kodowej, tzn.

$$S_{kk} = f(z_k). \tag{2.7}$$

Pozwala to na konsekwentny zapis w postaci:

$$q = \mathcal{F}p = \frac{1}{\bar{f}(p)}\boldsymbol{S}p, \tag{2.8}$$

który określa rozkład prawdopodobieństwa w następnej populacji po zastosowaniu operatora selekcji. Zauważmy, że zapis ten ma zastosowanie dla przypadku, gdy wychodzimy z konkretnej populacji o znanej względnej liczbie kopii elementu w populacji, ale także, gdy obiektem, na który działa operator selekcji, jest tylko rozkład prawdopodobieństwa występowania elementów przestrzeni kodowej w populacji. Często w tym miejscu dla skrótu mówi się o rozkładzie prawdopodobieństwa populacji. Już w tym miejscu widać, że nawet startując z $p \in \Lambda$, wynik w (2.8) nie musi być w tym samym zbiorze, gdyż wartości elementów macierzy $\boldsymbol{S}$ mogą być niewymierne. Tak więc bezpiecznie jest napisać, iż $q \in \bar{\Lambda}$.

Wartym podkreślenia jest fakt, że dzięki takiej definicji selekcji, zawierającej wartości funkcji przystosowania, będziemy w stanie w następstwie określić operator przejścia T (por.(2.18)) działający na rozkładach prawdopodobieństwa (wektorach prawdopodobieństwa) wszystkich populacji, włączając w definicję populacji wartości funkcji przystosowania. Dzięki temu operator T staje się operatorem liniowym, niezależnym od wartości funkcji przystosowania dla konkretnych elementów przestrzeni kodowej, występujących w konkretnej populacji. Przy odpowiedniej definicji tworzenia nowej generacji stosunki wartości funkcji przystosowania, będące podstawą wyznaczania selekcji proporcjonalnej, mogą być "ukryte" w wartości wektora prawdopodobieństwa uporządkowanego elementami zbioru wszytkich możliwych populacji (por. definicję zbioru wszystkich możliwych populacji W z początku rozdziału, oraz wyrażenia na prawdopodobieństwa (2.16) i (2.17)) .

2.4. Operator mutacji

Przejdźmy do operatora mutacji. Dla prostego operatora genetycznego naturalnym jest rozpatrzeć najpierw mutację binarną, równomierną, o parametrze μ. Oznacza to, że dowolny gen w chromosomie, komórce (elemencie przestrzeni kodowej), może być zmutowany z prawdopodobieństwem μ.

Wyjdźmy z dowolnego elementu z_j. Wiemy, że prawdopodobieństwo występowania tego elementu jest q_j. Prawdopodobieństwo przejścia w element z_i na drodze mutacji z populacji q jest

$$\sum_{j=0}^{s-1} U_{ij} q_j, \tag{2.9}$$

gdzie U_{ij} jest elementem macierzy $\boldsymbol{U}$ opisującej prawdopodobieństwa mutacji z elementu z_j w element z_i, w przypadku gdy $i \neq j$. Gdy $i = j$, jest to prawdopodobieństwo przetrwania elementu z_i w trakcie mutacji.

Sposób wyznaczania elementów tej macierzy pokazuje następujący przykład. Gdy z_i różni się od z_j na c pozycjach, to

$$U_{ij} = \mu^c (1-\mu)^{l-c}. \tag{2.10}$$

Składając operacje mutacji i selekcji otrzymujemy zależność

$$p(t+1) = \boldsymbol{U} \circ \mathcal{F}p(t) = \frac{1}{\bar{f}(p(t))}\boldsymbol{U}\boldsymbol{S}p(t), \tag{2.11}$$

gdzie zmienna t oznacza numer populacji, kroku iteracji.

Dla poprawności naszych wyników zawartych w następnych rozdziałach nie ma potrzeby ograniczania się jedynie do macierzy mutacji $\boldsymbol{U}$, której elementy dane są przez (2.10). Wyniki będą poprawne dla przypadku ogólniejszego, w szczególności dla niebinarnych operatorów mutacji. Będzie się jedynie wymagać, aby elementy macierzy $\boldsymbol{U}$ były nieujemne oraz ich suma w każdej kolumnie była równa jeden. To oznacza, że macierz $\boldsymbol{U}$ przeprowadza wektory prawdopodobieństwa w wektory prawdopodobieństwa, a to oznacza, że jest macierzą Markowa [52].

Większość wyników naszego artykułu dotyczy algorytmu genetycznego, w którym został pominięty następny element prostego algorytmu genetycznego, a mianowicie krzyżowanie.

2.5. Operator krzyżowania

Aby określić operator krzyżowania $\boldsymbol{C}$, musimy wprowadzić kilka dodatkowych pojęć. Niech macierze $\boldsymbol{C}_0, ..., \boldsymbol{C}_{s-1}$ będą takimi, że element (i, j) macierzy $\boldsymbol{C}_k$ oznacza prawdopodobieństwo, że element z_i skrzyżowany z elementem z_j wygeneruje element z_k.

Dla przybliżenia wprowadzanych pojęć rozpatrzmy skrótowo przypadek chromosomu o długości $l = 2$. Wówczas elementy przestrzeni kodowej mają postać

$$z_0 = 00,\ z_1 = 01,\ z_2 = 10, z_3 = 11. \tag{2.12}$$

Gdy krzyżowanie jest jednolite, tzn. wszystkie elementy mogą podlegać krzyżowaniu ze wszystkimi, macierz $\boldsymbol{C}_0$ ma postać

$$\boldsymbol{C}_0 = \begin{pmatrix} 1,0 & 0,5 & 0,5 & 0,25 \\ 0,5 & 0,0 & 0,25 & 0,0 \\ 0,5 & 0,25 & 0,0 & 0,0 \\ 0,25 & 0,0 & 0,0 & 0,0 \end{pmatrix} \tag{2.13}$$

Macierze $\boldsymbol{C}_k$ są symetryczne. Następnie tworzymy operator $\boldsymbol{C}$ w działaniu na dowolną populację p przepisem

$$\boldsymbol{C}(p) = (p \cdot \boldsymbol{C}_0 p, ..., p \cdot \boldsymbol{C}_{s-1} p), \tag{2.14}$$

gdzie kropka $\cdot$ oznacza formalny iloczyn skalarny dwóch wektorów z s-wymiarowej przestrzeni.

Działanie prostego algorytmu genetycznego [100, 68, 75] przy przejściu od danej populacji do następnej jest opisywane operatorem $\mathcal{G}$, będącym złożeniem trzech operatorów: selekcji, mutacji i krzyżowania

$$\mathcal{G} = \boldsymbol{C} \circ \boldsymbol{U} \circ \mathcal{F}. \tag{2.15}$$

Czytelnika zainteresowanego szczegółowym opisem tych mechanizmów odsyłamy do pozycji bibliografii [100, 68].

2.6. Model z selekcją i mutacją dla populacji skończonych

Niech $p = (p_0, ..., p_{s-1})$ będzie wektorem populacji. Gdybyśmy rozpatrywali $p \in \overline{\Lambda}$, wówczas operatory opisane w rozdziale 2 przeprowadzałyby zbiór $\overline{\Lambda}$ w siebie. Jednak kiedy mamy do czynienia z populacją o skończonej liczbie elementów, to znaczy $p \in \Lambda$, wówczas te operatory mogą wyprowadzać populację poza zbiór Λ. W tym przypadku postępujemy następująco. Jeśli mamy daną populację p, to losujemy ze zwracaniem r-elementów ze zbioru Z, przy czym prawdopodobieństwo wylosowania elementów $z_0, ..., z_{s-1}$ opisane jest wektorem $\mathcal{G}(p)$, gdzie

$$\mathcal{G}(p) = \frac{1}{\overline{f}(p)} \boldsymbol{U}\boldsymbol{S}p \ . \tag{2.16}$$

Te r-elementów to nasza nowa populacja q.

Oznaczmy przez W zbiór wszystkich możliwych populacji r-elementowych złożonych z elementów wybranych ze zbioru Z, przy czym elementy w populacji mogą się powtarzać. Zbiór ten jest skończony i jego moc oznaczamy przez M. Można wykazać, że liczba M dana jest pewną formułą kombinatoryczną. Teraz w dowolny sposób ponumerujemy populacje, tzn. $W = \{w^1, \ldots, w^M\}$. Oczywiście dla danej populacji $w^k = (w_0^k, \ldots, w_{s-1}^k)$, gdzie $k \in \{1, \ldots, M\}$, liczba w_i^k dla $i \in \{0, \ldots, s-1\}$ oznacza prawdopodobieństwo wylosowania z populacji w^k komórki z_i (czyli względny udział komórki z_i w populacji w^k). Załóżmy, że zaczynamy naszą implementację od dowolnej populacji $p = w^k$. W następnym kroku każda z populacji $w^1, \ldots, w^M$ może wystąpić z prawdopodobieństwem, które można wyznaczyć. Prawdopodobieństwo wystąpienia populacji $q = w^l$ w następnym pokoleniu jest równe

$$r! \prod_{j=0}^{s-1} \frac{\left(\mathcal{G}(p)_j\right)^{rq_j}}{(rq_j)!} \ . \tag{2.17}$$

Po dwóch krokach każda z populacji $w^1, \ldots, w^M$ będzie występować z pewnym prawdopodobieństwem, które jest dwukrotnym złożeniem powyższego wzoru. Podobnie w trzecim kroku. I tak dalej. Tak więc ma sens rozpatrywanie rozkładu prawdopodobieństwa, z jakim w kolejnych krokach pojawiają się odpowiednie populacje.

Oznaczmy

$$\Gamma = \{x \in \mathbf{R}^M : \forall k \ x_k \geq 0 \text{ oraz } ||x|| = 1\},$$

gdzie $\|x\| = x_1 + ... + x_M$, dla $x = (x_1, ..., x_M)$. Zbiór Γ składa się ze wszystkich możliwych rozkładów prawdopodobieństwa dla populacji. Tak więc opisana przez nas implementacja przeprowadza w każdym kroku zbiór Γ w siebie.

Wprowadźmy teraz na zbiorze Γ podstawowy dla dalszych rozważań probabilistyczny operator przejścia (ang. *transition operator*)

$$T(\cdot) : \mathbf{N} \times \Gamma \to \Gamma, \tag{2.18}$$

którego działanie określaliśmy opisowo powyżej. Jeśli $u \in \Gamma$, to przez $T(t)u = \big((T(t)u)_1, \ldots, (T(t)u)_M\big)$ oznaczamy rozkład prawdopodobieństwa dla M populacji w kroku numer t , jeśli zaczynaliśmy naszą implementację prostego algorytmu genetycznego $\mathcal{G}$ (por. (2.16)) od rozkładu prawdopodobieństwa dla M populacji równego $u = (u_1, \ldots, u_M) \in \Gamma$, przy t - krotnym zastosowaniu powyższego rozumowania.

Zauważmy, że $(T(t)u)_k$ dla $k \in \{1 \ldots, M\}$ oznacza prawdopodobieństwo wystąpienia w kroku numer t populacji w^k . Ze względu na definicję $\mathcal{G}(p)$ w (2.16),(2.17) oraz uwagę zrobioną na końcu punktu poświęconego operatorowi selekcji, operator przejścia T jest liniowy.

Dla przybliżenia sposobu działania algorytmów genetycznych warto odwołać się do obrazu często występującego w błądzeniu losowym punktów po pewnym zbiorze [52], gdyż samo działanie (prostego) algorytmu genetycznego jest podobne do następującego schematu. W przestrzeni wszystkich możliwych populacji $\bar{\Lambda}$ wędruje punkt, którego następne losowe położenie jest efektem działania SGA na bieżącej populacji. Wiemy, że w chwili początkowej punkt był jedną z możliwych populacji, ponumerowanych liczbami $1, 2, ..., M$, odpowiednio z prawdopodobieństwami $u_1, u_2, ..., u_M$. Wiemy też, że jeśli w chwili t (w pokoleniu o numerze t) mieliśmy populację p o numerze k, tzn. populację w^k, to prawdopodobieństwo, że w chwili $t+1$ (w pokoleniu o numerze $t+1$) osiągnie się populację q o numerze l, tzn. populację w^l, wynosi p_{lk} i to prawdopodobieństwo nie zależy od numeru kroku, w którym to przejście następuje. Przy tych właśnie oznaczeniach prawdopodobieństwo p_{lk} jest dane wzorem (2.17).

Utwórzmy nieujemną, kwadratową macierz $\boldsymbol{T}$ o rozmiarze M, której elementy tworzą p_{lk}, $l, k = 1, 2, ..., M$. Wówczas rozkład prawdopodobieństwa położenia naszego punktu w kroku t dany jest wzorem

$$\boldsymbol{T}^t u \quad t = 0, 1, 2, ...$$

Wzór definiuje nam każdy element macierzy opisującej prawdopodobieństwa przejścia między dowolnymi populacjami. Elementy te nie zależą od numeru kroku algorytmu. Wprowadzony powyżej operator przejścia $T(t)$ jest związany z powyższą macierzą następującą zależnością

$$T(t) = \boldsymbol{T}^t \ .$$

Tak określona macierz jest macierzą Markowa. Fakt ten pozwala na wykorzystanie dorobku teorii operatorów Markowa do analizy zbieżności algorytmów genetycznych.

Niech $e_k \in \Gamma$ będzie wektorem, który na k-tym miejscu ma jedynkę oraz zero na pozostałych miejscach. Tak więc e_k jest takim rozkładem, w którym populacja w^k występuje z prawdopodobieństwem 1. Zapis $T(t)w^k$ będziemy rozumieli jako

$$T(t)w^k = T(t)e_k. \tag{2.19}$$

W ten sposób opisany jest przypadek, gdy nasze doświadczenie rozpoczynamy od konkretnej populacji w^k. W dalszym ciągu będziemy zakładać, że zachodzi następujący warunek:

$$U_{jj} > 0 \ \text{ dla } \ j \in \{0, ..., s-1\}. \tag{2.20}$$

Warunek ten, w przypadku mutacji binarnej, opisanej poprzednio i danej wzorem (2.10) jest w rzeczywistości ograniczeniem na dopuszczalne wartości parametru μ. Parametr ten charakteryzuje prawdopodobieństwo mutacji pojedyńczego genu w chromosomie (elemencie przestrzeni kodowej), tutaj często nazywanym komórką. Spojrzenie na wzór (2.10) mówi, że spełnienie nierówności (2.20) jest możliwe przy warunku

$$0 \leq \mu < 1. \tag{2.21}$$

Załóżmy teraz, że mamy dany dowolny rozkład prawdopodobieństwa dla M populacji $u = (u_1, \ldots, u_M) \in \Gamma$. Łatwo obliczyć, że wtedy dla $i \in \{0, \ldots, s-1\}$ prawdopodobieństwo wylosowania komórki z_i wynosi

$$\sum_{k=1}^{M} w_i^k \cdot u_k \ , \tag{2.22}$$

gdzie w_i^k to prawdopodobieństwo wylosowania z k-tej populacji komórki z^i, a u_k prawdopodobieństwo wystąpienia k-tej populacji. **Populacją oczekiwaną** będziemy nazywać wektor z przestrzeni $\mathbf{R}^s$ którego i-ta współrzędna będzie dana wzorem (2.22). Ponieważ $u_k \geq 0,\ w_k^i \geq 0$ dla $k \in \{1, \ldots, M\},\ i \in \{0, \ldots, s-1\}$ oraz

$$\sum_{i=0}^{s-1} \left(\sum_{k=1}^{M} u_k \cdot w_i^k \right) = \sum_{k=1}^{M} u_k \left(\sum_{i=0}^{s-1} w_i^k \right) = \sum_{k=1}^{M} u_k = 1 \ ,$$

więc nasz wektor należy do $\overline{\Lambda}$. Z (2.22) wynika, że populacja oczekiwana opisana jest wzorem

$$\sum_{k=1}^{M} w^k \cdot u_k \tag{2.23}$$

Oczywiście populacja oczekiwana może nie być żadną ze wszystkich możliwych populacji r-elementowych.

Dla każdego $u \in \Gamma$ oraz dla dowolnego t mamy dany pewien rozkład prawdopodobieństwa dla M populacji $T(t)u$. Stąd wynika, że mamy daną również populację oczekiwaną w tym kroku.

Oznaczmy przez $R(t)u = \big((R(t)u)_0, \ldots, (R(t)u)_{s-1}\big)$ populację oczekiwaną w kroku t , jeśli zaczynaliśmy nasze doświadczenie od rozkładu $u \in \Gamma$. Mamy oczywiście $R(t)u \in \overline{\Lambda}$.

Definicja 2.6.1 *Będziemy mówili, że model jest asymptotycznie stabilny, jeśli istnieje $u^* \in \Gamma$ takie, że:*

$$T(t)u^* = u^* \quad \text{dla} \quad t = 0, 1, \ldots \tag{2.24}$$

$$\lim_{t \to \infty} ||T(t)u - u^*|| = 0 \quad \text{dla } \forall u \in \Gamma \ . \tag{2.25}$$

Ponieważ dla $k \in \{1, \ldots, M\}$ mamy

$$\big|\big(T(t)u\big)_k - u_k^*\big| \leq ||T(t)u - u^*|| \ , \tag{2.26}$$

więc z (2.25) otrzymujemy

$$\lim_{t \to \infty} \big(T(t)u\big)_k = u_k^* \ . \tag{2.27}$$

To oznacza, że prawdopodobieństwo wystąpienia populacji w^k w kroku numer t zmierza do pewnej ustalonej liczby u_k^* , niezależnej od początkowego rozkładu u . Ma to miejsce również w szczególnym przypadku, gdy naszą implementację rozpoczęliśmy od jednej konkretnej populacji $p = w^j$.

Twierdzenie 2.6.1 *Jeśli model jest asymptotycznie stabilny, to*

$$\lim_{t\to\infty} ||R(t)u - p^*|| = 0 \quad \text{dla} \quad u \in \Gamma \ , \tag{2.28}$$

gdzie $p^* \in \overline{\Lambda}$ *jest populacją oczekiwaną odpowiadającą rozkładowi* u^*. *W szczególności mamy również*

$$\lim_{t\to\infty} ||R(t)p - p^*|| = 0 \quad \text{dla} \quad p \in W. \tag{2.29}$$

Dowód . Z (2.23) mamy

$$R(t)u = \sum_{i=1}^{M} w^i \cdot \big(T(t)u\big)_i$$

oraz

$$p^* = \sum_{i=1}^{M} w^i \cdot u_i^* \ .$$

Tak więc

$$||R(t)u - p^*|| = ||\sum_{i=1}^{M} w^i \cdot \big(T(t)u\big)_i - \sum_{i=1}^{M} w^i \cdot u_i^*|| =$$

$$= \sum_{j=0}^{s-1} |\sum_{i=1}^{M} w_j^i \cdot \big(T(t)u\big)_i - \sum_{i=1}^{M} w_j^i \cdot u_i^*| \leq$$

$$\leq \sum_{j=0}^{s-1}\sum_{i=1}^{M} w_j^i |\big(T(t)u\big)_i - u_i^*| = \sum_{i=1}^{M} \left(\sum_{j=0}^{s-1} w_j^i\right) |\big(T(t)u\big)_i - u_i^*| = ||T(t)u - u^*||.$$

Stąd na podstawie (2.25) zachodzi (2.28). Biorąc pod uwagę przyjęte oznaczenie zadane zależnością (2.19), wzór (2.29) jest szczególnym przypadkiem (2.28). □

Z twierdzenia wynika, że jeśli model jest asymptotycznie stabilny, to populacja oczekiwana stabilizuje się zmierzając do $p^* \in \bar{\Lambda}$ niezależnie od warunków początkowych.

Przyjmujemy, że z komórki z_a można otrzymać z_b w jednej mutacji (lub w jednym kroku) z dodatnim prawdopodobieństwem, jeśli $U_{ba} > 0$. Zakładamy, że z komórki z_a można otrzymać komórkę z_b z dodatnim prawdopodobieństwem w n-mutacjach (lub w n-krokach), jeśli istnieją komórki $z_{i_o}, ..., z_{i_n}$ takie, że $z_{i_o} = z_a$, $z_{i_n} = z_b$ oraz każdą komórkę z_{i_j} dla $j = 1, ..., n$ można otrzymać z komórki $z_{i_{j-1}}$ w jednym kroku z dodatnim prawdopodobieństwem.

Przytoczony tutaj w Twierdzeniu 2.6.1 wynik ma fundamentalne znaczenie dla analizy zbieżności algorytmów genetycznych [88].

Definicja 2.6.2 *Model nazywamy punktowo asymptotycznie stabilnym, jeśli istnieje populacja* w^j *taka, że*

$$\lim_{t\to\infty} (T(t)u)_j = 1 \text{ dla } u \in \Gamma \ . \tag{2.30}$$

Warunek (2.30) Definicji 2.6.2 oznacza, że w kolejnych krokach prawdopodobieństwo pojawienia się populacji innej niż w^j zmierza do zera. Jest to szczególny przypadek asymptotycznej stabilności, gdzie

$$u^* = e_j.$$

Twierdzenie 2.6.2 *Model jest punktowo asymptotycznie stabilny wtedy i tylko wtedy, gdy istnieje dokładnie jedna komórka z_a o tej własności, że można ją otrzymać z dowolnej komórki w skończonej ilości kroków z dodatnim prawdopodobieństwem. W takiej sytuacji populacja w^j składa się wyłącznie z komórek z_a oraz zachodzi*

$$T(t)w^j = w^j \ . \tag{2.31}$$

Ponadto prawdopodobieństwo wystąpienia w kroku numer t populacji innej niż w^j zmierza do zera w postępie geometrycznym, tzn. istnieją $\lambda \in (0,1)$, $D \in \boldsymbol{R}_+$ takie, że

$$\sum_{\substack{i=1 \\ i \neq j}}^{M} (T(t)u)_i \leq D \cdot \lambda^t \ . \tag{2.32}$$

□

Dowody twierdzeń i wspomagających je lematów są umieszczone w oryginalnych artykułach [86, 87, 85], do których odwołujemy się w Bibliografii.

Liczby λ i D dla konkretnego modelu możemy wyznaczyć. Wzór (2.30) mówi nam, że gdybyśmy w rzeczywistości postępowali według opisanego przez nas algorytmu , to populacja w^j pojawi się po skończonej ilości kroków. Ze wzoru (2.31) wynika, że z populacji w^j otrzymujemy w^j z prawdopodobieństwem równym 1, czyli, jeśli w^j raz się pojawi, to od tego momentu będziemy mieli stale populację w^j.

Z twierdzenia 2.6.2 wynika, że taka jak wyżej zbieżność do jednej populacji może pojawić się tylko przy bardzo szczególnych założeniach. To uzasadnia sens badania asymptotycznej stabilności takiej, jak w Definicji 3.1.

Definicja 2.6.3 *Przez komórkę osiągalną rozumiemy taką komórkę $z_a \in Z$, którą można otrzymać z dowolnej innej w skończonej ilości kroków z dodatnim prawdopodobieństwem. Oznaczymy przez Z^* zbiór komórek z_a o tej własności.*

Twierdzenie 2.6.3 *Model jest asymptotycznie stabilny wtedy i tylko wtedy, gdy $Z^* \neq \emptyset$.* □

Twierdzenie 2.6.4 *Załóżmy, że model jest asymptotycznie stabilny. W tej sytuacji zachodzi następująca równoważność:*

(war) $u_k^* > 0$ *wtedy i tylko wtedy, gdy populacja w^k składa się wyłącznie z komórek należących do zbioru Z^*.*

Uwaga 2.6.1 *Jeśli $Z^* = Z$, to $\forall k \in \{1, \ldots, M\}$ $u_k^* > 0$.*

□

Podsumujmy nasze wyniki:

1. $Z^* = \emptyset \Rightarrow$ brak asymptotycznej stabilności;
2. $Z^* \neq \emptyset \Rightarrow$ asymptotyczna stabilność, przy czym:
3. Z^* jest zbiorem jednoelementowym $\Rightarrow$ punktowa asymptotyczna stabilność (zbieżność w pewnym sensie do jednej populacji);

4. Z^* jest zbiorem zawierającym więcej niż jeden element $\Rightarrow$ asymptotyczna stabilność, ale brak punktowej asymptotycznej stabilności (stabilizuje się prawdopodobieństwo otrzymania poszczególnych populacji w kolejnych krokach, lecz brak zbieżności do jednej populacji).

Uwaga Dla konkretnego modelu spełniającego założenia Twierdzenia 2.6.4 można otrzymać efektywne oszacowanie u_k^* i p_k^* od dołu.

ROZDZIAŁ 3

Układy dynamiczne a algorytmy genetyczne

3.1. Algorytmy genetyczne jako układy dynamiczne

Algorytmy genetyczne generują struktury zbiorów rozwiązań (a także trajektorie) o pewnych szczególnych własnościach, które możemy badać metodami układów dynamicznych, a w szczególności teorii ergodycznej. Możliwa jest sytuacja, w której wyniki uzyskane dla algorytmów genetycznych są przenaszalne na algorytmy ewolucyjne. Ma to miejsce, gdy proces przeszukiwania algorytmu ewolucyjnego możemy opisać macierzą Markowa. Wykorzystanie łańcuchów Markowa do modelowania algorytmów genetycznych podjęto w szeregu prac [101, 100, 76, 20]. Modele te są dotychczas pewną ideą pozwalającą badać własności graniczne algorytmów genetycznych i ewolucyjnych. Są one konstruowane w sposób na tyle ogólny , że obejmują szerokie klasy algorytmów i pozwalają na prowadzenie badań własności jakościowych algorytmów. Na podstawie modelu Markowa udowodniona została zbieżność algorytmów i istnienie rozkładu granicznego. Uzyskano też wyniki określające parametry zbieżności. Jednak wyniki te są wynikami jakościowymi i niewiele mówią o zachowaniu konkretnego algorytmu. Trudności związane z budową modelu konkretnego algorytmu wynikają ze złożonego mechanizmu działania algorytmu. Zależy on od sposobu kodowania, mutacji, krzyżowania, selekcji i wielkości parametrów określających te operatory genetyczne, a także wzajemnego oddziaływania tych czynników. Oddziaływania te są nieliniowe i trudne do modelowania oraz analizy. Wpływ mutacji na działanie algorytmu zależy od sposobu kodowania zadania. Zagadnienie kodowania jest także ściśle związane z prawdopodobieństwem mutacji. Jeśli ze zmianą kodowania zmienia się prawdopodobieństwo mutacji tak, że prawdopodobieństwa przejść są zachowane, to algorytm zachowuje się tak samo. W związku z tym algorytm z autonomicznym dostrajaniem mutacji powinien zachowywać się tak samo, niezależnie od sposobu kodowania. Sprawa dostrajania parametrów i ogólniej sterowania parametrami powinna być rozpatrywana razem z zagadnieniem kodowania. Niezależne rozpatrywanie tych zagadnień może być błędne. Należy także brać pod uwagę problem nadmiarowości kodowania. Staje się wtedy istotną wielkość nadmiarowości i sposób ingerencji nadmiarowości w znaczenie kodu [67].

Nawet algorytm z dodatnim prawdopodobieństwem mutacji nie przeprowadza każdego elementu w każdy z tym samym prawdopodobieństwem, gdy kodowanie jest nadmiarowe. Czy można dobrać prawdopodobieństwo mutacji tak, aby skompensować nadmiarowość?. To pytanie pozostaje otwarte.

Mutacja nie ma wpływu na rozkład graniczny, gdy macierze opisujące poszczególne operacje genetyczne są przemienne [89]. W praktycznych zadaniach takie przypadki nie zachodzą. Wobec tego macierze te nie są przemienne, a zatem mutacja ma wpływ na rozkład graniczny. Interakcja między parametrami algorytmów powoduje, że wyniki badań indywidualnego wpływu poszczególnych parametrów na zachowanie algorytmów nie pozwalają na ich uogólnienie na inne przypadki. Istnieje potrzeba wprowadzenia klasyfikacji algorytmów na podstawie ich własności dynamicznych.

3.1.1. Algorytmy, struktury, losowość

Algorytmy genetyczne jako szczególna wersja algorytmów ewolucyjnych oparta na binarnej reprezentacji osobników i operatorach selekcji, krzyżowania oraz mutacji, są dziś powszechnie wykorzystywanymi metodami rozwiązywania różnorodnych zagadnień optymalizacyjnych. Jednak mimo intensywnych badań, podstawy działania takich algorytmów nadal nie są w pełni wyjaśnione. Teoria algorytmów ,mimo podejmowania jej przez wiele ośrodków i znamienitych badaczy, jest nadal cząstkowa i nie daje zadowalających (dostatecznych) podstaw do projektowania algorytmów dopasowanych do zadania optymalizacyjnego. Algorytmy genetyczne są układami (operatorami) losowymi, silnie nieliniowymi i istnieją trudności z uzyskaniem mocnych wyników teoretycznych. Projektując algorytm genetyczny programista kieruje się doświadczeniem. Brak jest natomiast niekwestionowanych metod pozwalających na jednoznaczne przyporządkowanie algorytmu do rozwiązywanego zadania optymalizacyjnego.

3.1.2. Zbieżność binarnego algorytmu genetycznego

W teorii obliczeń ewolucyjnych najważniejszymi zagadnieniami są: badanie zbieżności algorytmów ewolucyjnych, a także asymptotycznego zachowania trajektorii algorytmów, badanie stopnia trudności problemu (zadania) ze względu na rodzaj algorytmu oraz problematyka złożoności obliczeniowej algorytmów. Jednym z najważniejszych problemów jest ustalenie zależności pomiędzy typem algorytmu ewolucyjnego a zadaniem optymalizacyjnym (funkcją dopasowania), do którego jest on stosowany. Istotne jest, jakie kryteria należy stosować przy takim doborze. Jest to problem otwarty i praktycy pokonują go opierając się na doświadczeniu i wskazówkach heurystycznych.

Cechą charakterystyczną algorytmu genetycznego jest zwiększanie się udziału osobników lepiej dopasowanych w populacji i wzmacnianie ich wpływu na kształtowanie się następnych populacji, poprzez dłuższe przeżycie i tworzenie większej ilości potomków. Jednak sposoby wprowadzenia tego mechanizmu do algorytmu genetycznego różnią się bardzo w zależności od przekonań i wiedzy programistów. Parametrami decydującymi o działaniu algorytmu są mechanizmy reprodukcji i dziedziczenia, rozmiary populacji i metody selekcji osobników. Potrzebne są wyniki teoretyczne opisujące zależności między metodami kodowania (reprezentacji) osobników a własnościami operatorów przeszukujących przestrzeń kodową.

Zbieżność algorytmu genetycznego jest głównie uzależniona od sposobu generowania nowych populacji oraz rodzaju i parametrów selekcji. Badania tych zagadnień są podstawowymi tematami teorii algorytmów genetycznych i badań empirycznych.

Wobec szybkiego rozwoju algorytmów genetycznych, ich typów, jak i zmienności parametrów powstaje potrzeba wypracowania takich metod doboru operatorów genetycznych, które optymalizują algorytmy dla pewnej, możliwie szerokiej klasy zadań

[16]. Ciąg populacji algorytmu genetycznego można rozpatrywać jako łańcuch Markowa (z czasem dyskretnym). Model łańcucha Markowa pozwala na badanie algorytmu genetycznego przy użyciu procesów równoważnych łańcuchom Markowa. Ze względu na to, że przestrzeń kodowa jest przestrzenią binarną, a algorytmy działają na ciągach binarnych (osobnikach), celowe jest podjęcie badań algorytmów jako przesunięcia Bernoulliego.

To stwierdzenie jest podstawą klasyfikacji. Klasyfikację proponujemy prowadzić na podstawie entropii trajektorii algorytmu [21, 14, 27].Hipoteza, że można klasyfikować algorytmy genetyczne na podstawie entropii (lub wymiaru fraktalnego) trajektorii, opiera się na twierdzeniu Ornsteina [61], według którego entropia jest niezmiennikiem izomorfizmu dla przesunięć Bernoulliego. Oczywiście taka klasyfikacja może być tylko przybliżona, gdyż trajektoria algorytmu genetycznego jest zbiorem skończonym.

Wyznaczając zatem entropię trajektorii algorytmów genetycznych możemy stwierdzić, czy są one izomorficzne. Bezpośrednie wyznaczanie entropii na podstawie trajektorii nie jest możliwe bez znajomości rozkładu prawdopodobieństwa przejścia między stanami populacji. Wymaganie znajomości rozkładu i uwzględnienie tej informacji przy wyznaczaniu entropii powoduje, że niemożliwe jest zastosowanie tej metody, gdy chcemy badać konkretne realizacje algorytmu i jego trajektorię. Wyznaczanie entropii możliwe jest na podstawie obserwacji pojedynczej trajektorii algorytmu i nie wymaga jakiejkolwiek wiedzy o rozkładzie prawdopodobieństw przejść pomiędzy poszczególnymi stanami procesu. Warunki do tego określa twierdzenie Lempel-Ziv (tzw. metoda przedrostkowa) [1,29]. Rozważając trajektorię jako nieskończenie długi tekst, wyznaczamy entropię poprzez zliczanie przedrostków [74]. Dwa procesy, które maja tę samą entropię są równoważne (w sensie złożoności) i mogą być przekształcone jeden na drugi poprzez przekształcenie, które komutuje z przesunięciem w przestrzeni trajektorii.

3.2. Główne kierunki badań

W algorytmach genetycznych różnorodność populacji i napór selekcyjny są dwoma istotnymi zagadnieniami i konkurują ze sobą [6]. Można postawić hipotezę, że są one sprzężone i mają istotny wpływ na zbieżność algorytmów genetycznych.

Zbieżność algorytmów genetycznych jest jednym z najbardziej fundamentalnych zagadnień teoretycznych, badanych między innymi za pomocą skończonych łańcuchów Markowa [70, 41]. Tworzono też metaalgorytm genetyczny do sterowania parametrami innego algorytmu genetycznego . Staranne badanie wpływu parametrów sterujących na algorytmy genetyczne przedstawiono w [59].

Próbowano też rozstrzygnąć problem: który z operatorów, mutacji czy krzyżowania, ma większy wpływ na działanie algorytmu genetycznego? Wynikiem tych analiz było stwierdzenie, że wpływ operatora mutacji na działanie algorytmu genetycznego jest większy, niż to początkowo przypuszczano [56]. Stąd badanie własności operatora mutacji jest szczególnie uzasadnione przy próbie wyjaśnienia mechanizmów działania algorytmów genetycznych.

W badaniach tych podjęta zostanie próba klasyfikacji algorytmów genetycznych z wykorzystaniem pojęcia entropii [21, 79, 7]. Propozycja ta wykorzystuje fakt izomorfizmu operatorów ergodycznych w przypadku, gdy mają one tą samą entropię.

Ponieważ entropia określa własności mieszania punktów trajektorii operatora, więc jej badanie dla operatorów ergodycznych jest w istocie badaniem własności mieszania algorytmu genetycznego.

Dotychczas sporadycznie wykorzystywano pojęcie entropii do badania lub konstrukcji algorytmów genetycznych, a inspiracja do zastosowania tego pojęcia wywodziła się z teorii informacji. Entropię wykorzystywano do sterowania operacją krzyżowania przy rozwiązywaniu zagadnień klasyfikacyjnych. W innej procedurze entropia wykorzystywana jest do sterowania szybkością zbieżności populacji i zapożyczona jest z metody symulacyjnego wyżarzania .

Podejmowano też próby wykorzystania entropii do wyznaczania miar określających zachowanie algorytmu genetycznego [97], jednak i w tym przypadku inspiracja stosowania entropii pochodziła z teorii informacji .

Ergodyczność operatora mutacji jest własnością jednowymiarowego działania operatora mutacji, gdyż taki jest mechanizm działania tego operatora. Jednak wyznaczone w ten sposób punkty odpowiadają punktom w przestrzeni wielowymiarowej. W związku z tym celowe wydaje się podjęcie badań symulacyjnych wyjaśniających, na ile własność ergodyczności dotyczy reprezentacji punktów w przestrzeni wielowymiarowej.

Najbardziej efektywną metodą badania jest wyznaczenie wymiaru pudełkowego trajektorii generowanych przez operatory algorytmów genetycznych [4, 33, 46]. Wymiar pudełkowy jest (dyskretnym) przybliżeniem wymiaru fraktalnego, więc jego badanie może być innym sposobem przybliżonego wyznaczania entropii tych operatorów, gdyż istnieje ścisła zależność między entropią a wymiarem fraktalnym [3]. Istnienie zależności między entropią operatora a wymiarem fraktalnym jego trajektorii jest podstawą do przeprowadzenia analizy porównawczej tak otrzymanych wyników, to jest entropii i wymiaru pudełkowego. Jeżeli klasyfikacja uzyskana na podstawie entropii będzie się pokrywała z klasyfikacją uzyskaną na postawie wymiaru pudełkowego, to będzie to potwierdzeniem naszej hipotezy.

3.3. Parametry

3.3.1. Samoadaptacja parametrów operatorów genetycznych

Zagadnienie doboru parametrów algorytmu jest, jak do tej pory, zadaniem słabo zdefiniowanym i złożonym, o nieokreślonej strukturze. Równocześnie sądzimy, że sam (metalgorytm dobudowany do algorytmu genetycznego) algorytm genetyczny może lepiej rozwiązuje tego typu zadanie niż inne metody. Dotyczy to sytuacji, gdy dopasowanie parametrów algorytmu genetycznego następuje na podstawie aktualnego stanu procesu (aktualnej populacji). Możliwe jest to poprzez wprowadzenie reguł zmiany, doboru parametrów, na podstawie stanu populacji i wartości funkcji przystosowania. Także możliwe jest to przy równoczesnej modyfikacji funkcji przystosowania. Można też rozszerzyć wartość chromosomu poprzez dołączenie parametrów do chromosomu i poddanie całości procesowi ewolucji. Następuje w ten sposób powiązanie wielkości i zmian parametrów z aktualnym stanem procesu, populacji i rozwiązania. Możliwe jest też równolegle włączenie osobnego algorytmu (metalgorytmu) modyfikującego parametry właściwego algorytmu rozwiązujacego postawione zadanie.

3.3.2. Twierdzenie Markowa

Działanie algorytmu genetycznego jest wynikiem złożenia przekształceń populacji realizowanych kolejno przez operatory selekcji, krzyżowania i mutacji. Iteracyjne powtarzanie tych działań nad populacją osobników będących elementami przestrzeni rozwiązań prowadzi do uzyskania rozwiązań zadań optymalizacyjnych.

Definicja 3.3.1 *Operatory Markowa ([52])*
Dana jest macierz kwadratowa

$$\boldsymbol{P} = (p_{ij}) \quad i,j = 1,....,M\,, \quad p_{ij} \geq 0\,, \sum_{k=1}^{M} p_{kj} = 1 \quad ,i,j = 1,....,M, \tag{3.1}$$

macierz taka jest macierzą Markowa.

Rozważmy dowolny wektor probabilistyczny f_i, $f_i \geq 0$, dla $i = 1,....M$ spełniający warunki

$$\sum_{k=1}^{M} f_k = 1. \tag{3.2}$$

Badamy ciąg $\boldsymbol{P}^n f$ dla $n = 0, 1, ...$

Definicja 3.3.2 $\boldsymbol{P}$ *jest asymptotycznie stabilny, jeśli dla każdego* f *spełniającego powyższe warunki , ciąg* $\boldsymbol{P}^n f$ *zmierza do tej samej granicy* f^**, niezależnie od wyboru* f*.*

Warunek konieczny i dostateczny asymptotycznej stabilności podał Markow.

Twierdzenie 3.3.1 *Operator (macierz) Markowa* $\boldsymbol{P} : \boldsymbol{R}^M \to \boldsymbol{R}^M$ *jest asymptotycznie stabilny wtedy i tylko wtedy, gdy dla pewnego naturalnego* n *istnieje wskaźnik* $i \in \{1, ..., M\}$ *taki, że wszystkie wyrazy* i*-tego wiersza macierzy* $\boldsymbol{P}^n$ *są dodatnie.*

Algorytm możemy traktować jako losowanie następnej populacji z rozkładem wyznaczonym przez daną populację. Jeżeli liczba populacji jest skończona, to funkcja przejścia może być opisana skończoną liczbą prawdopodobieństw przejść, które tworzą macierz kwadratową o wymiarze równym ilości możliwych populacji. Macierz taka jest macierzą Markowa.

Jeśli prawdopodobieństwo mutacji jest dodatnie $p_m > 0$, to dodatnie jest też prawdopodobieństwo przejścia z dowolnej populacji do każdej innej. Zatem macierz Markowa opisująca funkcje przejścia prostego algorytmu genetycznego ma wszystkie wyrazy dodatnie. Taki operator jest asymptotycznie stabilny na mocy powyższego twierdzenia Markowa. Rozkład graniczny algorytmu jest wówczas niezależny od populacji początkowej. Jest to najprostszy przypadek stabilności asymptotycznej prostego algorytmu genetycznego.

3.3.3. Algorytm elitarny

Algorytmy z selekcją elitarną (zachowanie najlepszego osobnika w następnej populacji) mogą być modelowane modelem Markowa, w macierzy którego występują zera. Warunkiem jest, aby w pewnym momencie w macierzy występował wiersz niezerowy. W algorytmie elitarnym podczas tworzenia następnej populacji zachowywany jest najlepszy osobnik (element) populacji poprzedniej. Przy takiej selekcji macierz

Markowa, opisująca prawdopodobieństwa przejścia od danej populacji do następnej zawiera elementy zerowe. Wynika to z tego, że możliwe jest tylko przejście do populacji, w której element najlepszy jest nie gorszy niż aktualny element najlepszy. Stąd prawdopodobieństwo przejścia do populacji, której element najlepszy byłby gorszy od aktualnego jest równe zero. Jeżeli jednak weźmiemy populację składającą się wyłącznie z samych elementów najgorszych w danej przestrzeni poszukiwań, to z tej populacji możliwe jest przejście do wszystkich innych populacji (przy dodatnim prawdopodobieństwie mutacji), niezależnie od tego, czy element najgorszy jest jeden, czy też jest ich wiele. To samo dotyczy także elementów najlepszych. Zatem wiersz, w którym występuje opisana najgorsza populacja ma wszystkie elementy niezerowe. W takim przypadku spełniony jest warunek twierdzenia Markowa o asymptotycznej stabilności wymagający, aby istniała potęga macierzy Markowa o niezerowym wierszu (którego wszystkie wyrazy są niezerowe). Zatem, na mocy twierdzenia Markowa można stwierdzić, że algorytm elitarny ma rozkład graniczny niezależny od rozkładu początkowego.

3.3.4. Algorytm genetyczny o zmiennych parametrach

Algorytm genetyczny jest procesem adaptacyjnym. W związku z tym naturalnym jest, aby w trakcie jego wykonywania podlegały adaptacji także jego parametry i to na mocy wewnętrznej dynamiki algorytmu. Istnieje hipoteza, że w różnych stadiach realizacji algorytmu zmieniają się wartości parametrów, w sposób optymalny dla danego etapu poszukiwań rozwiązania. Samoadaptacja parametrów operatorów genetycznych może następować poprzez dobór tych parametrów według pewnego rozkładu prawdopodobieństwa. Rozszerzamy wektor poszukiwań o dodatkowe elementy, które podlegają ewolucji razem z całym algorytmem i równocześnie wpływają na przebieg algorytmu. Możliwe jest też prowadzenie równoległej adaptacji zbioru operatorów.

W przypadku, gdy parametry mutacji, krzyżowania lub selekcji zależą od danej populacji i są dodatnie, rozkład prawdopodobieństw przejścia jest stały i niezależny od kroku macierzy i tworzy macierz Markowa. Jej wszystkie elementy są dodatnie pod warunkiem, że mutacja jest dodatnia, niezależnie od wielkości tej mutacji.

Twierdzenie 3.3.2 *Algorytm genetyczny o parametrach dostrajanych poprzez dołączenie parametrów do chromosomu i poddanie ich działaniu operatorów genetycznych razem z całym chromosomem, jest asymptotycznie stabilny i jego rozkład graniczny jest niezależny od populacji początkowej.*

Twierdzenie 3.3.3 *Algorytm genetyczny, w którym parametry są dostrajane w zależności od wartości funkcji przystosowania lub jej rozkładu, jest asymptotycznie stabilny i jego rozkład graniczny nie zależy od populacji początkowej.*

Twierdzenie 3.3.4 *Algorytm genetyczny, w którym parametry dobierane są poprzez równolegle działający algorytm i przyjmują niezerowe wartości prawdopodonieństw mutacji, krzyżowania i selekcji, jest asymptotycznie stabilny i jego rozkład graniczny nie zależy od populacji początkowej.*

Definicja 3.3.3 *Przestrzenią stanów algorytmu genetycznego nazywamy zbiór, do którego należą wszystkie populacje (lub ich jednoznaczne reprezentacje), jakie może wytworzyć ten algorytm.*

Algorytm genetyczny dokonuje przekształcenia (iteracji) populacji w danej przestrzeni stanów w kolejnych pokoleniach $t = 1, 2, 3, ...$ Ta metoda tworzenia ciągu populacji jest niedeterministyczna. W związku z tym populacje tworzą ciąg zmiennych losowych o wartościach w przestrzeni stanów, któremu odpowiada ciąg rozkładów prawdopodobieństwa. Wówczas działanie algorytmu odpowiada losowaniu populacji ze zbioru, zgodnie z rozkładem prawdopodobieństwa. Ciąg populacji algorytmu genetycznego tworzy zatem (proces stochastyczny) łańcuch Markowa (z czasem dyskretnym), ponieważ operatory genetyczne w klasycznych algorytmach genetycznych nie zależą od poprzednich.

3.4. Algorytmy genetyczne a układy dynamiczne

Zajmiemy sie tutaj ważnym związkiem wyników rozdziału 2 z teorią łańcuchów Markowa i ich asymptotycznymi zachowaniami. Teoria takich łańcuchów jest dobrze opisana w pozycji [39], z której zaczerpnięto definicje i odpowiednie twierdzenia.

Definicja 3.4.1 *Łańcuch Markowa nazywamy jednorodnym (w czasie), gdy istnieje macierz* $\boldsymbol{P} = (p_{ij})$ *będąca dla każdego* n *jego macierzą przejścia w* n*-tym kroku.*

Definicja 3.4.2 *Łańcuch Markowa nazywamy nieprzywiedlnym, gdy wszystkie stany wzajemnie komunikują się; oznacza to, że* $\forall(j,k)$ *istnieje* n*, że* $p_{jk}(n) > 0$

Definicja 3.4.3 *Okresem stanu* j *nazywamy liczbę*

$$o(j) = NWD\{n : p_{jj}(n) > 0\}.$$

Jest to największy wspólny dzielnik zbioru takich n*, że powrót do stanu* j *może nastąpić po* n *krokach. Stan* j *nazywamy okresowym, gdy* $o(j) > 1$ *i nieokresowym, gdy* $o(j) = 1$*.*

Twierdzenie 3.4.1 *W nieprzywiedlnym łańcuchu Markowa wszystkie stany mają ten sam okres.*

Definicja 3.4.4 *Nieprzywiedlny łańcuch Markowa nazywamy okresowym, gdy jego stany są okresowe z okresem* $d > 1$*. W przeciwnym przypadku łańcuch nazywamy nieokresowym.*

Twierdzenie 3.4.2 *Jeśli łańcuch Markowa jest nieprzywiedlny i nieokresowy, to dla każdej pary stanów* i, j *mamy* $p_{ij} > 0$ *dla dostatecznie dużych* n*.*

Definicja 3.4.5 *Mówimy, że rozkład prawdopodobieństwa jest rozkładem stacjonarnym dla łańcucha Markowa o macierzy przejścia* $\boldsymbol{P}$*, gdy*

$$\pi = \boldsymbol{P}\pi,$$

czyli, gdy $\sum_j \pi_j = 1$ *oraz dla każdego* i *jest* $\pi_i \geq 0$ *i* $\pi_i = \sum_j \pi_j p_{ij}$*.*

Twierdzenie 3.4.3 *Niech* P *będzie macierzą przejść dla skończonego łańcucha Markowa. Łańcuch ten jest nieprzywiedlny i nieokresowy wtedy i tylko wtedy,gdy istnieje takie całkowite* $n > 0$*, że każdy element macierzy* $\boldsymbol{P}^n$ *jest dodatni. Wówczas także każdy element macierzy* $\boldsymbol{P}^m$*, gdzie* $m > n$*, jest dodatni.*

Twierdzenie 3.4.4 *Twierdzenie graniczne. Jeśli* $\boldsymbol{P}$ *jest macierzą opisującą nieprzywiedlny i nieokresowy (regularny) łańcuch Markowa, to*

- $\boldsymbol{P}^n$ *zbiega dla* $n \to \infty$ *do macierzy stochastycznej* $\boldsymbol{Q}$.
- *Każdy wiersz macierzy* $\boldsymbol{Q}$ *jest tym samym wektorem* π^*, *i każda składowa* π^* *jest dodatnia.*

Macierz $\boldsymbol{Q}$ *nazywamy macierzą graniczną, a wektor* π^**–wektorem granicznym łańcucha.*

Twierdzenie 3.4.5 *[12] Niech* $\boldsymbol{P}$ *będzie macierzą losową. Jeśli wektor* $\pi = (\pi_i)$ *spełnia warunek* $\pi\boldsymbol{P} = \pi$, *oraz* $\sum_{i=1}^{M} \pi_i = 1$, *wówczas spełnione są następujące zależności:*

(i) *istnieje suma* $\boldsymbol{Q} = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \boldsymbol{P}^n$,

(ii) $\boldsymbol{Q}$ *jest macierzą losową ,*

(iii) $\boldsymbol{Q}P = \boldsymbol{P}Q = \boldsymbol{Q}$,

(iv) *Jeśli* $\boldsymbol{P}$ $v = \boldsymbol{v}$ *to* $\boldsymbol{Q}$ $v = \boldsymbol{v}$.

Twierdzenie 3.4.6 *[12] Wszystkie wiersze* $\boldsymbol{Q}$ *są identyczne. Każdy wiersz* $\boldsymbol{Q}$ *jest równy* π.

Uwaga. Przesunięcie Markowa z macierzą o tych samych wierszach jest przesunięciem Bernoulliego. Zatem rozkład graniczny można rozpatrywać jako przesunięcie Bernoulliego. Tak więc kwestia izomorfizmu może być rozstrzygana tak, jak dla przesunięcia Bernoulliego, a więc na mocy entropii. Problemem jest na ile to samo dotyczy całego przebiegu algorytmu (przesunięcia Markowa, które jest równoważne przesunięciu Bernoulliego).

Wniosek 3.4.1 *Prosty algorytm genetyczny jest równoważny przesunięciu Bernoulliego.*

Wniosek 3.4.2 *Entropia jest niezmiennikiem izomorfizmu prostego algorytmu genetycznego.*

Stwierdzenie, że prosty algorytm genetyczny jest równoważny przesunięciu Bernoulliego, pozwala na wykorzystanie wszystkich metod badawczych i wyników uzyskanych przy analizie tych przekształceń do badania algorytmów genetycznych. Wynik ten znakomicie rozszerza zakres narzędzi, które mogą być wykorzystane w analizie algorytmów i ich projektowaniu.

Twierdzenie 3.4.7 *Zbieżność* $||\boldsymbol{P}^n - \boldsymbol{Q}||$ *gdy* $n \to \infty$ *jest wykładnicza.*

Twierdzenie 3.4.8 *Dla prawie każdego algorytmu genetycznego istnieje algorytm optymalny w sensie probabilistycznym. Oznacza to, że algorytm ten startując z dowolnego rozkładu początkowego już w pierwszym kroku generuje rozkład graniczny. Operatorem opisującym taki algorytm jest macierz* $\boldsymbol{Q}$ *(zdefiniowana w poprzednich twierdzeniach).*

Dowód.

Niech będzie dany dowolny wektor początkowy $\boldsymbol{c} = (c_1, c_2, ..., c_M)$ przy czym $\sum_i c_i = 1$. Weźmy dowolną kolumnę $\boldsymbol{Q}$ np. i-tą, wszystkie elementy tej kolumny mają wartość π_i

$$c_1\pi_i + c_2\pi_i + c_3\pi_i + \cdots + c_M\pi_i = \pi_i(c_1 + c_2 + \cdots + c_M) = \pi_i.$$

A więc $\boldsymbol{c}Q = \pi_i$. □

Twierdzenie to jest komplementarne do twierdzenia No Free Lunch dla algorytmów genetycznych. Oznacza to, że o ile No Free Lunch zajmuje się całym universum algorytmów i zadań, to powyższe twierdzenie zajmuje się pojedyńczym algorytmem i zadaniem. O ile tamto mówi, że dla universum wszystkie algorytmy i zadania są średnio takie same, to nasze twierdzenie pokazuje, że dla prawie każdego (pojedyńczego) algorytmu ewolucyjnego i (pojedyńczego) zadania optymalizacyjnego istnieje algorytm nie tylko lepszy, ale i najlepszy w sensie probabilistycznym. I nie można go doprowadzić do zadania deterministycznego, gdyż wtedy musi być spełnione twierdzenie o punktowej asymptotycznej stabilności. Pozostaje problem, czy na podstawie pewnych danych, można wyznaczyć odpowiednie przybliżenie algorytmu optymalnego. Porównanie z twierdzeniem o punktowej asymptotycznej stabilności wskazuje kierunek poszukiwań. Może też być pułapką gdyż wydaje się, że nie ma „gładkiego" przejścia między tymi dwoma typami zbieżności.

Kwestia optimum może być niejednoznaczna, gdyż dwa algorytmy dające rozkład graniczny w jednym kroku mogą mieć różny rozkład graniczny. Wówczas lepszy jest ten, który daje większe prawdopodobieństwo najlepszemu rozwiązaniu. Należy więc uwzględnić entropię rozkładu granicznego.

Weźmy dwa algorytmy optymalizacyjne dla tej samej funkcji. Za lepszy uważamy ten, który ma większe prawdopodobieństwo dla optimum (problemem jest sytuacja, gdy są dwa lub więcej optima). Niemożliwe jest przejście do (0,0,...,1,0,...,0), gdyż taki algorytm byłby deterministyczny (w granicy) a Q jest dodatnia.

Ciąg algorytmów w „mierze" entropii jest ciągiem Cauchy'ego. Entropia może być dowolnie bliska zera , ale nie jest zbieżna do zera. Dla każdego algorytmu istnieje zatem algorytm lepszy, ale nie istnieje najlepszy algorytm. Te rozważania dotyczyły nieskończonej liczby algorytmów. Dla skończonej liczby algorytmów zawsze istnieje najlepszy, mimo, że nie potrafimy wyznaczyć go efektywnie.

Jeżeli mamy algorytm z dostrajaniem parametrów, najlepszą drogę (regułę) zmiany parametrów określa rozkład graniczny. Parametry zakodowane w osobnikach populacji ewoluują tworząc rozkład graniczny.

3.4.1. Problem sposobu kodowania i mutacji

Porównywane są różne sposoby kodowania, np. binarne, Graya, rzeczywiste i prowadzone są ich wartościowania, najczęściej przy porównaniu odległości, otoczenia sąsiedztwa. Podobnie rozważa się dobór i sterowanie parametrami, niezależnie od zadania kodowania. Natomiast wydaje się, że te problemy są sprzężone. Zmiana kodowania zmienia prawdopodobieństwo przejścia między elementami przy tej samej mutacji. Po zmianie kodowania można tak zmienić mutację i krzyżowanie, aby macierz przejścia między elementami była prawie równa poprzedniej. Taka zmiana niczego nie zmienia w działaniu algorytmu i jego rozkładzie granicznym, natomiast zmienia schematy (te powinny zależeć od sposobu kodowania). Nasuwa się pytanie, czy mogżna znaleźć taka zmianę, która zachowa schematy (a zmieni rozkład graniczny),

oraz czy istnieje taka zmiana, która zachowa obydwa, schematy i rozkład. Jest to problem jednoznaczności lub niejednoznaczności algorytmów genetycznych. Wiąże się to z problemem dostrajania parametrów lub sterowania parametrami [32, 65]. Jeżeli dostrajanie parametrów działa w sposób optymalny, to algorytm przy dostrajaniu parametrów powinien się zachowywać tak samo (optymalnie), niezależnie od sposobu kodowania. Inaczej jest, gdy dostrajanie zależy od sposobu kodowania.

3.4.2. Wyznaczanie rozkładu granicznego

Nasuwa się pytanie, czy możliwe jest doświadczalne wyznaczenia rozkładu granicznego lub jego przybliżenia. Odpowiedź na to pytanie daje następujący wniosek.

Wniosek 3.4.3 *Poszczególne składowe rozkładu stacjonarnego opisują proporcje czasów spędzanych przez proces w poszczególnych stanach, przy uśrednieniu po długich czasach.*

Powyższy wniosek daje możliwość przybliżonego wyznaczenia prawdopodobieństw będących składowymi rozkładu granicznego. Problemem jest to, że wynik dotyczy populacji, natomiast badamy doświadczalnie wystąpienia poszczególnych stanów algorytmu. Można jednak przyjąć, że tak uśrednione dane będą zbliżone do populacji oczekiwanej dla rozkładu granicznego.

3.4.3. Operator graniczny algorytmu elitarnego

Algorytm genetyczny z selekcją elitarną modelowany jest macierzą blokową składającą się z podmacierzy odpowiadających uporządkowanym elementom przeszukiwanej przestrzeni [81]. Porządek podmacierzy odpowiada porządkowi elementów i określony jest przez wartość funkcji dopasowania dla każdego elementu. Dla każdych dwóch elementów x, y, przypisujemy większy priorytet elementowi o większej wartości funkcji przystosowania f. Stąd, dla $f(x) > f(y) \Rightarrow pr(x) > pr(y)$. Najlepszy element ma priorytet 1, najgorszy element ma priorytet n. Oznaczamy przez $P(i)$ podmacierze opisujące przejścia dla klasy populacji o najlepszym elemencie o priorytecie i, w ramach tej klasy. Algorytm elitarny pozwala tylko na przejście do populacji o najlepszym osobniku nie gorszym od wyjściowego. Wobec tego macierz $\boldsymbol{P}$ ma postać:

$$\boldsymbol{P} = \begin{bmatrix} \begin{bmatrix} & & \\ & P(1) & \\ & & \end{bmatrix} & \begin{matrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{matrix} & \\ \begin{matrix} + & \cdots & + \\ + & & \cdot \\ \cdot & & \cdot \\ + & \cdots & + \end{matrix} & \begin{bmatrix} & & \\ & P(2) & \\ & & \end{bmatrix} & \begin{matrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{matrix} \\ \begin{matrix} & + & \\ & \cdot & \\ & \cdot & \\ & \cdot & \end{matrix} & \begin{matrix} \cdot & \cdots & \cdot \end{matrix} & \\ \begin{matrix} + & \cdots & + \\ + & & \cdot \\ \cdot & & \cdot \\ + & \cdots & + \end{matrix} & \begin{matrix} + & \cdots & + \\ + & & \cdot \\ \cdot & & \cdot \\ + & \cdots & + \end{matrix} & \begin{bmatrix} & & \\ & P(n) & \\ & & \end{bmatrix} \end{bmatrix}.$$

Macierz[1] ta generuje łańcuch pochłaniający opisany przez macierz $P(1)$. Powoduje to, że rozkład graniczny algorytmu elitarnego odpowiada rozkładowi granicznemu macierzy $P(1)$. Oznaczając ten rozkład przez π^{1*} możemy wyznaczyć operator, optymalny w sensie probabilistycznym, algorytmu elitarnego w postaci:

$$\boldsymbol{Q}_{opt} = \begin{bmatrix} \pi_1^{1*}, & \pi_2^{1*} & \dots & \pi_k^{1*} & \dots & 0 \\ \pi_1^{1*}, & \pi_2^{1*} & \dots & \pi_k^{1*} & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \pi_1^{1*}, & \pi_2^{1*} & \dots & \pi_k^{1*} & \dots & 0 \end{bmatrix}.$$

Macierz ta ma tyle wierszy, ile występuje w macierzy $\boldsymbol{P}$, a k odpowiada rozmiarowi macierzy $P(1)$. Algorytm elitarny opisany powyższym operatorem w jednym kroku generuje rozkład graniczny odpowiadający oryginalnemu algorytmowi opisanemu macierzą $\boldsymbol{P}$. Dowód tego faktu jest analogiczny do dowodu Twierdzenia 3.4.8.

3.5. Postać graniczna algorytmu genetycznego

W rozprawie wyznaczony został operator opisujący postać graniczną danego algorytmu genetycznego. Operator taki opisuje działanie algorytmu genetycznego w nieskończonym ciągu kroków, ale niewiele mówi o tym, jaka pojawi się następna populacja, gdy dana jest populacja poprzednia. Postać graniczna opisuje bardziej rozkład populacji po nieskończonej ilości kroków niż efekt działania algorytmu w danej chwili. Postać operatora granicznego opisuje działanie algorytmu genetycznego globalnie, zostawiając dla konkretnego kroku jedynie prawdopodobieństwo. Jest to pewnego rodzaju prawo wielkich liczb dla algorytmów. Gdyby istniała możliwość skonstruowania algorytmu o operatorze takim, jak operator graniczny, to taki algorytm realizowałby w każdym kroku zachowanie takie, jakie realizuje algorytm po nieskończonej ilości kroków. A więc statystycznie byłoby to znaczne przyśpieszenie działania algorytmów.

3.5.1. Istnienie algorytmu optymalnego

Optymalny algorytm optymalizacyjny to taki algorytm, który wyznacza poszukiwane rozwiązanie w jednym kroku działania algorytmu. W przypadku algorytmów probabilistycznych, algorytmem optymalnym jest algorytm, który w pierwszym kroku daje największe prawdopodobieństwo wyznaczenia optimum. Postać takiego algorytmu została przedstawiona w tym rozdziale. Wynik taki ma znaczenie egzystencjalne, stawia także zadanie wypracowania metod konstruowania takich algorytmów. Niezbędne do tego są zarówno badania eksperymentalne, jak i dalsze analizy teoretyczne. Przedstawione w tej pracy rezultaty opisujące rozkłady graniczne dotyczą populacji a nie osobników.

[1] Użylismy tutaj wyjątkowo lewostronnego działania macierzy przejścia $\boldsymbol{P}$ na wektor probabilistyczny.

3.5.2. Algorytm genetyczny jako operator rzutowy

Algorytm genetyczny generuje rozkład graniczny prawdopodobieństwa pojawienia się wszystkich populacji. Rozkład graniczny jest punktem stałym operatora Markowa. Rozszerzeniem tych wyników jest fakt, że dla każdego prostego algorytmu genetycznego, algorytmu elitarnego oraz algorytmu z samoadaptacją parametrów, istnieje najlepszy algorytm genetyczny w sensie probabilistycznym. Postać takiego algorytmu została podana w tej pracy. Niezależnie od początkowego rozkładu prawdopodobieństwa, algorytm ten generuje rozkład graniczny w jednym kroku i ten rozkład jest punktem stałym algorytmu. Najlepszy algorytm w sensie probabilistycznym jest operatorem rzutowym.

ROZDZIAŁ 4

Entropia i wymiar fraktalny w klasyfikacji

Rozwojowi algorytmów genetycznych towarzyszy od początków ich pojawienia się zagadnienie ich klasyfikacji. Jest ono ściśle związane z samym pojęciem i rozumieniem algorytmów, jak i słynnym twierdzeniem No Free Lunch Theorem, które stwierdzając, że nie istnieje uniwersalny, najlepszy algorytm genetyczny, niejako wymusza dopasowywanie algorytmu do zadania optymalizacyjnego. Pojęcie lepszego lub gorszego algorytmu optymalizacyjnego można rozważać tylko w kontekście zadania, do którego jest on stosowany. Bez samego zadania optymalizacyjnego, dla którego mamy zamiar stosować odpowiedni algorytm, wartościowanie - a tym bardziej przypisanie do jakiejś klasy - nie jest uzasadnione. Także pojęcie algorytmu jest związane z zadaniem optymalizacyjnym. Rozróżnienie algorytmu od zadania budzi wiele wątpliwości. Funkcja dopasowania może być traktowana jako parametr algorytmu.

4.1. Entropia

Oznaczmy przestrzeń probabilistyczną jako (X, B, m).

Przyjmijmy, że $\zeta(A) = \{A_1, \ldots, A_k\}$, jest skończonym rozbiciem mierzalnym przestrzeni X na zbiory o prawdopodobieństwach (p_i) odpowiadających miarom $m(A_i)$. Wówczas entropię rozbicia definiujemy jako funkcję

$$H(\zeta(A) = H(A) = -\sum_{i=1}^{k} m(A_i) \log(m(A_i)).$$

Niezmiennik P relacji równoważności jest niezmiennikiem zupełnym. Jeśli kiedykolwiek przekształcenia T i S mają własność P, to T i S są równoważne. Entropia jako niezmiennik ma pewne ograniczenia. Nie jest ona dobrym niezmiennikiem sprzężenia dla całej klasy transformacji o zerowej entropii. Jest ważne, aby wyznaczyć klasę przekształceń zachowujących miarę, dla których entropia jest dobrym niezmiennikiem. Prowadzi to do wprowadzenia przekształceń, które są "przeciwieństwem" przekształceń o zerowej entropii.

Automorfizm Bernoulliego i automorfizm Kołmogorowa

W roku 1958 Andrei Kołmogorow postawił hipotezę (pytanie), że entropia jest niezmiennikiem zupełnym dla (wszystkich) przesunięć Bernoulliego. Odpowiedź przedstawił w 1969 r. Ornstein [61].

Twierdzenie 4.1.1 *Niech T_1 i T_2 będą przesunięciami Bernoulliego, których przestrzenie stanów są przestrzeniami Lebesgu'a. Jeśli $h(T_1) = h(T_2)$, wówczas T_1 jest sprzężone z T_2, a zatem izomorficzne przy założeniu o przestrzeni stanów, że przeliczalny iloczyn kartezjański przestrzeni (stanów) Lebesgu'a, jest przestrzenią Lebesgu'a.*

Twierdzenie to redukuje zagadnienie izomorfizmu (sprzężenia) przesunięć Bernoulliego do jego (ich) przestrzeni stanów, ponieważ entropia zależy od przestrzeni stanów. Równocześnie istnieje możliwość , że przesunięcie Bernoulliego z dwupunktową przestrzenią stanów może być sprzężone z przesunięciem Bernoulliego o przeliczalnie nieskończonej przestrzeni stanów (przykład Mieszałkina)[96]. Definicja przesunięcia Bernoulliego podana zostanie poniżej.

4.1.1. Porównywanie przekształceń zachowujących miarę

Rozważmy nieskończony ciąg utworzony z k symboli. Dla ułatwienia oznaczenia, te k symboli wybieramy spośród $\{0, 1, ..., k-1\}$. Niech $X = \prod_1^\infty \{1, ..., k\}$. Element $x \in X$ oznaczamy przez $\{x_1, x_2, ...\}$ lub $\{x_1x_2x_3...\}$. W przypadku $k = 2$ jest to ciąg dwójkowy (binarny). Niech $p_1, ..., p_k$ będą liczbami nieujemnymi, takimi, że $p_1 + \cdots + p_k = 1$. Zdefiniujemy dla $t \geq 1$ cylinder (blok) o długości n jako

$$[a_1, \ldots, a_n]_{t,\ldots,t+n-1} = \{x \in X : x_{t+1} = a_1, \ldots, x_{t+n} = a_n\}.$$

Definicja 4.1.1 *Zdefiniujemy miarę μ na cylindrze jako*

$$\mu([a_1, \ldots, a_n]_{t,\ldots,t+n-1} = p_{a_1} \cdots p_{a_n}.$$

Miara probabilistyczna określona na X, znów oznaczona przez μ jest jednoznacznie zdefiniowana na σ–algebrze generowanej przez zbiory cylindryczne (cylindrów). Miare μ nazywamy $(p_1, \ldots, p_k)$–miarą Bernoulliego, a X jest przestrzenią przesunięć Bernoulliego. Jednostronne przesunięcie Bernoulliego TB na X określone jest poprzez

$$(x_1x_2x_3 \ldots) \mapsto (x_2x_3x_4 \ldots),$$

tzn. $TB(x_1x_2x_3 \ldots) = (x_2x_3x_4 \ldots)$ Podobnie definiujemy dwustronne przesunięcie Bernoulliego jako

$$(\ldots \hat{x}_0x_1x_2 \ldots) \mapsto (\ldots \hat{x}_1x_2x_3 \ldots)$$

na $X = \prod_{-\infty}^{\infty} \{1, ..., k\}$, przy czym $\hat{}$ oznacza zerową współrzędną (położenie) w ciągu.

Zauważmy, że przesunięcie zachowuje miare μ.

Dla $X = \prod_1^\infty \{1, ..., k\}$, jak poprzednio, niech $\boldsymbol{P} = [P_{ij}]$ będzie macierzą losową o wymiarze $k \times k$ z operacją prawostronną [1]. Załóżmy, że $\boldsymbol{\pi} = [\pi_i]$, gdzie każde $\pi_i > 0$ spełnia $\sum_i \pi_i = 1$ oraz jest prawym wektorem własnym, tzn. $\boldsymbol{P\pi} = \boldsymbol{\pi}$.

[1] Choe [12] rozpatruje macierze losowe z operacjami lewostronnymi.

Definicja 4.1.2 *Definiujemy ν jako*

$$\nu([a_1, \ldots, a_n]_{t,\ldots,t+n-1}) = P_{a_n a_{n-1}} \ldots P_{a_2 a_1} \pi_{a_1}.$$

Wówczas istnieje jednoznaczna miara probabilistyczna niezmiennicza względem przesunięcia, oznaczana ponownie jako ν, określona na σ–algebrze generowanej przez zbiór cylindrów. Miarę ν nazywamy wówczas miarą Markowa, a X przestrzenią przesunięć Markowa.

Niech $Pr(B|A)$ oznacza prawdopodobieństwo zdarzenia B, pod warunkiem zdarzenia A. Wówczas P określa prawdopodobieństwo przejścia

$$Pr(x_{n+1} = j | x_n = i) = P_{ji}.$$

Jest to prawdopodobieństwo warunkowe, spełniające zależność

$$\sum_{b=1}^{k} Pr(b|a) = 1$$

dla każdego $a \in \{0, 1, ..., k-1\}$. Przesunięcia Markowa są przesunięciami Bernoulliego, gdy kolumny P są identyczne (równe). Przesunięcie oznaczamy przez jak poprzednio przez $\boldsymbol{T}B$.

Rozważania poprzednie pozwalają na utożsamienie miary Markowa lub miary Bernoulliego z miarą na odcinku $[0, 1]$ poprzez rozwinięcie dwójkowe. Oznacza to, że ciąg dwójkowy $(a_1 a_2 a_3 \ldots)$ jest utożsamiany z $\sum_{n=1}^{\infty} a_n 2^{-n}$. W przypadku, gdy $p \notin \{0, \frac{1}{2}, 1\}$ mamy do czynienia z $(p, 1-p)$–miarą Bernoulliego na przedziale $[0, 1]$ [12].

Izomorfizm (równoważność) przekształceń.

Nasuwa się pytanie, jak można porównywać i ewentualnie klasyfikować przekształcenia zachowujące miarę, określone na przestrzeniach probabilistycznych. Związane to jest z obserwacją, że wśród zespołu obiektów, różne obiekty mogą posiadać tę samą strukturę matematyczną i wówczas możemy je identyfikować (uważać za takie same, podobne, analogiczne). Relacja równoważności jest zbiorem reguł, które ustalają, kiedy dwa elementy $x, y \in X$ są tożsame z matematycznego punktu widzenia. Formalnie niezbędne są do tego trzy warunki:

- (zwrotność) $x \sim x$ for $x \in X$
- (symetria) jeśli $x \sim y$, to $y \sim x$ dla $x, y \in X$ oraz
- (przechodniość) jeśli $x \sim y$ i $y \sim z$, wówczas $x \sim z$ dla $x, y, z \in X$.

Na podstawie tej definicji możemy wprowadzić pojęcie klasy równoważności zawierającej x, jako zbioru wszystkich y spełniających $x \sim y$ i oznaczanej przez $[x]$.

Własności wszystkich elementów danej klasy równoważności możemy badać poprzez analizę własności dowolnego elementu (reprezentanta) tej klasy.

Definicja 4.1.3 *Niech (X_1, μ_1) i (X_2, μ_2) będą przestrzeniami mierzalnymi (z miarą).*

- *Przekształcenie $\phi : (X_1, \mu_1) \to (X_2, \mu_2)$ nazywamy bijekcją prawie wszędzie i zachowującym miarę, wówczas (X_1, μ_1) i (X_2, μ_2), jeśli istnieją $E_1 \subset X_1$ i $E_2 \subset X_2$ takie, że $\mu_1(E_1) = 0 = \mu_2(E_2)$ oraz $\phi X_1 \backslash E_1 \to X_2 \backslash E_2$ jest jednoznaczne i "na".*

- *Ponadto, jeśli istnieje prawie wszędzie przekształcenie (bijekcja) $\phi : (X_1, \mu_1) \to (X_2, \mu_2)$ takie, że ϕ oraz ϕ^{-1} są mierzalne i zachowują miarę, wówczas (X_1, μ_1) i (X_2, μ_2) nazywamy izomorficznymi, a ϕ nazywamy izomorfizmem (X_1, μ_1) i (X_2, μ_2). (Przez odwrotność ϕ przyjmujemy odwrotność $\phi : X_1 \backslash E_1 \to X_2 \backslash E_2$).*

Przykład 4.1.1 *[12] Niech $X = \prod_1^\infty \{0, 1\}$ będzie przestrzenią przesunięć Bernoulliego z prawdopodobieństwami $(\frac{1}{2}, \frac{1}{2})$, co oznacza, że każdy symbol (z dwóch symboli) występuje z prawdopodobieństwem $\frac{1}{2}$. Wówczas przestrzeń X jest izomorficzna z odcinkiem $[0, 1]$ z miarą Lebesgue'a. Można to pokazać biorąc $x = (b_1, b_2, b_3, \ldots) \in X$ i definiując przekształcenie*

$$\phi(x) = \sum_{n=1}^{\infty} b_n 2^{-n}$$

które jest prawie wszędzie bijekcją i zachowuje miarę.

Definicja 4.1.4 *Niech $T_1 : (X_1, \mu_1) \to (X_1, \mu_1)$ i $T_2 : (X_2, \mu_2) \to (X_2, \mu_2)$ będą przekształceniami zachowującymi miarę. Mówimy, że są one izomorficzne, jeśli istnieje izomorfizm $\phi : (X_1, \mu_1) \to ((X_2, \mu_2)$ taki, że $\phi \circ T_1 = T_2 \circ \phi$, tzn. przemienny jest następujący diagram (komutuje):*

$$\begin{array}{ccc} (X_1, \mu_1) & \xrightarrow{T_1} & (X_1, \mu_1) \\ {\scriptstyle \Phi}\downarrow & & {\scriptstyle \Phi}\downarrow \\ (X_2, \mu_2) & \xrightarrow{T_2} & (X_2, \mu_2) \end{array}$$

(Zakładamy że $T_1(X_1 \backslash E_1) \subset X_1 \backslash E_1$ oraz $T_2(X_2 \backslash E_2) \subset X_2 \backslash E_2$, gdzie E_1 i E_2 są zdefiniowane w poprzedniej definicji.)

Przykład 4.1.2 Przekształcenia izomorficzne .

Przekształcenia Bernoulliego $(\frac{1}{2}, \frac{1}{2})$ przestrzeni $X_1 = \prod_1^\infty \{0, 1\}$ są izomorficzne poprzez izomorfizm ϕ zdefiniowany w poprzednim przykładzie. Przyjmijmy $T_1 : X_1 \to X_1$ jako

$$T_1((b_1, b_2, b_3, \ldots)) = (b_2, b_3, b_4, \ldots),$$

oraz zdefiniujmy $T_2 : X_2 \to X_2$ jako

$$T_2(x) = 2x(mod 1)$$

Wówczas T_1 i T_2 są izomorficzne, ponieważ $\phi \circ T_1 = T_2 \circ \phi$.

4.1.2. Pomiar entropii

Najkorzystniejszą metodą wyznaczania entropii jest wykorzystanie algorytmów kompresji zbiorów (trajektorii) [7, 74]. Algorytmy kompresji zbiorów korzystają z twierdzenia Lempel-Ziv. Wyszukują one powtarzające się sekwencje znaków (przedrostki), analizują ich położenie oraz rozkład. Program kompresji analizuje powtarzające się

sekwencje (strings) i zapisuje częściej powtarzające się ciągi przy wykorzystaniu mniejszej ilości bitów niż występujące rzadko. W związku z tym stosunek wielkości zbioru wyjściowego poddanego kompresji i zbioru uzyskanego w wyniku przeprowadzenia kompresji jest miarą (przybliżeniem) entropii procesu generującego ten zbiór i przybliżenie to jest tym dokładniejsze im dłuższy jest zbiór badany. Możliwe jest także bezpośrednie porównywanie dwóch trajektorii poprzez kompresję jednego zbioru, a następnie dołączenie do zbioru bazowego zbioru porównywanego i poddanie tak skonstruowanego zbioru kompresji. Jeżeli wielkości tych zbiorów są zbliżone, to zbiory mają tę samą entropię (a więc istnieje izomorfizm). Wynika to z faktu, że zachodzi nierówność dla entropii zbiorów A, i B, przy czym równość zachodzi tylko wtedy, gdy zbiory A i B są niezależne. Pozwala to na pomiar podobieństwa (odległości) trajektorii (w sensie złożoności). Pytaniem jest, na ile ten wskaźnik odległości informacyjnej opisuje różnice między trajektoriami algorytmów genetycznych jako układów dynamicznych. I tu jako potwierdzenie można przywołać model algorytmu genetycznego jako (łańcucha Markowa) przesunięcie Bernoulliego.

Wskaźnikiem ściśle związanym z entropią jest wymiar fraktalny trajektorii. Możliwe jest też zastosowanie wymiaru fraktalnego jako miary podobieństwa i wskaźnika klasyfikacji algorytmów. W tym przypadku skorzystamy z metody wstępnie opracowanej w pracach [64, 2, 47].

Zaproponowana metoda odbiega od typowych sposobów badania algorytmów genetycznych, w których podstawą jest podejście probabilistyczne. Nasze badania bazujące na wynikach uzyskanych w teorii układów dynamicznych i dynamice symbolicznej nastawione są na weryfikację empiryczną i wypracowanie metod pozwalających na eksperymentalną ocenę algorytmów i ich zachowania.

4.2. Wymiar fraktalny

Dziedziną badań teorii ergodycznej jest przestrzeń z miarą i przekształcenia zachowujące miarę. Przestrzenią z miarą jest zbiór X z miarą m, (znormalizowaną do 1 nazywamy prawdopodobieństwem), zdefiniowaną na mierzalnej σ-algebrze jej podzbiorów $\mathcal{B}$.

Definicja 4.2.1 *Przestrzenie probabilistyczne* $(X_1, \mathcal{B}_1, m_1), (X_2, \mathcal{B}_2, m_2)$ *nazywamy* ***izomorficznymi****, jeśli istnieją* $M_1 \in \mathcal{B}_1, M_2 \in \mathcal{B}_2$ *z* $m_1(M_1) = 1 = m_2(M_2)$ *i odwracalne przekształcenia zachowujące miarę* $\phi : M_1 \to M_2$. *i.e.* $m_1(\phi^{-1}(E)) = m_2(E)$ *dla każdego podzbioru mierzalnego* $E \in \mathcal{B}_2$.

Aby możliwe było badanie algorytmów genetycznych i ich podobieństwa (lub więcej - izomorfizmu), należy rozważyć (przekształcenie) operator zdefiniowany na przestrzeni probabilistycznej.

Definicja 4.2.2 *Załóżmy,że* $(X_1, \mathcal{B}_1, m_1), (X_2, \mathcal{B}_2, m_2)$ *są przestrzeniami probabilistycznymi łącznie z przekształceniami zachowującymi miarę* $T_1 : X_1 \to X_1, T_2 : X_2 \to X_2$. *Przekształcenia* T_1 *są* ***izomorficzne*** T_2, *jeśli istnieją* $M_1 \in \mathcal{B}_1, M_2 \in \mathcal{B}_2$ *z* $m_1(M_1) = m_2(M_2) = 1$ *takie, że*

- $T_1(M_1) \subseteq M_1$, $T_2(M_2) \subseteq M_2$,

➤ *wówczas istnieją odwracalne przekształcenia zachowujące miarę*

$$\phi : M_1 \to M_2 \quad with\ \phi(T_1(x)) = T_2(\phi((x)) \quad for\ all\ x \in M_1.$$

W naszym podejściu główną rolę odgrywa przesunięcie Bernoulliego [96, 12] zdefiniowane poprzednio.

Oznaczamy przez $\mathtt{T}_i$ operator, który przekształca i-te pokolenie (punkt trajektorii) w następne. Mając rozkład prawdopodobieństwa charakteryzujący przekształcenie $\mathtt{T}_i$ z aktualnej populacji do następnej możemy zdefiniować entropię tego przekształcenia:

$$H(\mathtt{T}_i) = -\sum_{j=1}^{M} \mathcal{P}(P^r_{i+1,j}|P^r_i, f(P^r_i), p_m, p_c, p_s) \cdot \log \mathcal{P}(P^r_{i+1,j}|P^r_i, f(P^r_i), p_m, p_c, p_s) \tag{4.1}$$

gdzie $[P^r_{i+1,j}]$ oznacza populacją j z przestrzeni kodowej B, $j = 1, 2, \ldots, 2^{Nr}, \ldots, M$, jaką algorytm może przyjąć w kroku $i + 1$.

Entropia jest funkcją prawdopodobieństwa mutacji i selekcji. Wzrasta ona ze wzrostem prawdopodobieństwa mutacji i maleje, gdy rośnie nacisk selekcyjny. Tak więc entropia może być miarą interakcji pomiędzy operatorami selekcji i mutacji. Entropia trajektorii jest związana ze złożonością obliczeniową algorytmu ewolucyjnego.

Wyznaczanie prawdopodobieństwa przekształcenia $\mathtt{T}_i$, jak i entropii H_i, jest trudne. Proponujemy zastąpienie jej wymiarem fraktalnym, który jest zależny od entropii [3] i może charakteryzować losowe własności algorytmów genetycznych. W [29] wprowadzono statystyczne i dynamiczne metody analizy algorytmów genetycznych.

Twierdzenie 4.2.1 *(Ornstein) Dwa dowolne przesunięcia Bernoulliego z tą samą entropią są izomorficzne.*

Twierdzenie 4.2.2 *(Choe)[12] Niech $X = \prod_1^\infty \{0, 1\}$ będzie $(p, 1 - p)$ przestrzenią z przesunięciem Bernoulliego, którą podobnie jak przedział jednostkowy $[0, 1)$ wyposażono w metrykę Euklidesową. Niech X_p oznacza zbiór wszystkich ciągów binarnych $x \in X$ takich, że*

$$X_p = \{x \in X : \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} x_i = p\}.$$

Wówczas wymiar Hausdorffa x_p jest równy jego entropii $-p \log_2 p - (1-p) \log_2 (1-p)$ przesunięcia Bernoulliego.

Podobny wynik jest spełniony dla przestrzeni przesunięć Markowa.

Tak więc możemy wykorzystać wymiar Hausdorffa lub jego przybliżenie jako niezmiennik równoważności algorytmów.

4.2.1. Wymiar pudełkowy, informacyjny

Przyjmijmy oznaczenie dla s- wymiarowej miary Hausdorffa na podzbiorze $E \subset \mathbf{R}^l$, gdzie $s \geq 0$. Jeśli $E \subset \bigcup_i U_i$ i średnica U_i, oznaczona jako $\delta(U_i)$ jest mniejsza od ϵ dla każdego i, to mówimy, że $\{U_i\}$ jest ϵ-pokryciem E. Dla $\epsilon > 0$, zdefiniujemy

$$\mathcal{H}^s_\epsilon(E) = \inf \sum_{i=1}^{\infty} [\epsilon(U_i]^s \tag{4.2}$$

gdzie infimum jest brane po wszystkich ϵ-pokryciach $\{U_i\}$ zbioru E. Granica $\mathcal{H}^s_\epsilon$, gdy $\epsilon \to 0$ oznaczana przez $\mathcal{H}^s(E)$, jest s-wymiarową miarą Hausdorffa zbioru E.

Zauważmy, że w przestrzeni $\mathbf{R}^l$ można dowieść, że $\mathcal{H}^l(E) = \kappa_l \mathcal{L}^l(E)$ gdzie $\mathcal{L}^l$ jest l-wymiarową miarą Lebesgue, a κ_l jest stosunkiem miary l–wymiarowej kostki do l–wymiarowej kuli wpisanej w kostkę.

Jest oczywiste, że $\mathcal{H}^s_\epsilon(E)$ rośnie, gdy największa średnica ϵ zbioru U_i dąży do zera, gdyż rozważamy coraz mniejsze elementy, których nie uwzględniamy w większej skali.

Jednakże miara Hausdorffa $\mathcal{H}^s_\epsilon(E)$ maleje, gdy s wzrasta i dla dużych s przyjmuje wartość 0. Wówczas wymiar Hausdorffa E definiujemy jako

$$\dim(E) = \inf\{s : \mathcal{H}^s(E) = 0\}, \tag{4.3}$$

i można sprawdzić, że $\dim(E) = \sup\{s : \mathcal{H}^s(E) = \infty\}$.

$$\dim_{box}(S) = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)} \tag{4.4}$$

Jeżeli granica nie istnieje, możemy rozważać **górny wymiar pudełkowy** i **dolny wymiar pudełkowy**, który odpowiada górnej i dolnej granicy powyższego wyrażenia. Stąd wymiar pudełkowy jest dobrze zdefiniowany tylko wtedy, gdy górny i dolny wymiar pudełkowy są sobie równe. Górny wymiar pudełkowy jest czasami nazywany **wymiarem entropijnym**, wymiarem Kołmogorowa lub wymiarem Minkowskiego, podczas gdy dolny wymiar pudełkowy nazywamy **dolnym wymiarem Minkowskiego**. Te wymiary są ściśle związane z wymiarem Hausdorffa. Tylko w bardzo wyselekcjonowanych przypadkach ważnym jest rozróżnienie między nimi. Innym przybliżeniem wymiaru Hausdorffa jest wymiar korelacyjny.

Obydwa wymiary pudełkowe są skończenie addytywne, t.j. jeśli $\{A_1, A_2, ..., A_n\}$ jest skończoną rodziną zbiorów

$$\dim_{box}(\{A_1 \cup A_2 \cup .... \cup A_n\}) = \max\{\dim_{box}(A_1), \dim_{box}(A_2), ..., \dim_{box}(A_n)\}.$$

Podobnie, jak wymiar Hausdorffa, wymiar pudełkowy jest jednym z wymiarów, który może być wykorzystany do opisu fraktali. Dla wielu fraktali wszystkie te wymiary są równe. W przypadku zbioru Cantora wynoszą one $\log(2)/\log(3)$. Jednak definicje te nie są równoważne. Zachodzi nierówność pomiędzy wymiarem Hausdorffa i pudełkowym.

$$\dim_H(S) \leq \dim_{dolnybox}(S) \leq \dim_{górnybox}(S) \tag{4.5}$$

Często obydwie nierówności mogą być ostre. Górny wymiar pudełkowy może być większy od dolnego, gdy zachowanie fraktala zależy od skali. Wymiar pudełkowy nie zapewnia własności stabilności, jakiej oczekiwalibyśmy od wymiaru. Przykładowo można sądzić, że dodanie przeliczalnego zbioru nie będzie miało wpływu na wymiar zbioru. Ta własność nie jest prawdziwa dla wymiaru pudełkowego.

$$\dim_{box}(\{0, 1, 1/2, 1/3, 1/4, ...\}) = \frac{1}{2}$$

Po wprowadzeniu wymiaru fraktalnego (Hausdorffa) możemy badać trajektorie GA algorytmów genetycznych lub ich atraktory. Algorytmy możemy uważać za równoważne, jeśli mają tę samą złożoność obliczeniową przy rozwiązywaniu takich samych zadań. Jako miarę złożoności obliczeniowej można wziąć iloczyn rozmiaru populacji i ilości kroków, po której otrzymujemy rozwiązania zadania. Ten sposób szacowania złożoności obliczeniowej algorytmów genetycznych łączy złożoność pamięciową i czasową.

Przy wykonywaniu zadania algorytm genetyczny wyznacza trajektorię, która powinna zbiegać do pewnego atraktora. Oczekujemy, że idealny algorytm genetyczny wyznaczy rozwiązanie optymalne w ten sposób, że trajektoria zmierza do atraktora będącego zbiorem jednowymiarowym. Jednak algorytm bez selekcji ma za atraktor całą przestrzeń. Tak więc możemy powiedzieć, że algorytmy są równoważne, jeśli tworzą podobne atraktory. Proponuję więc wykorzystanie wymiaru fraktalnego jako miary podobieństwa atraktorów.

Definicja 4.2.3 *Dwa algorytmy genetyczne są równoważne, jeżeli realizują trajektorie o tym samym wymiarze fraktalnym.*

Tak więc obok entropii także wymiar fraktalny będzie wykorzystywany jako wskaźnik lub miara przy klasyfikacji algorytmów genetycznych.

Przejście od entropii do niezmiennika, jakim jest wymiar fraktalny może być zrealizowane przy pomocy szczególnych wskaźników. Takim wskaźnikiem jest wymiar informacyjny zdefiniowany przez

$$D_I(T) = -\lim_{\epsilon \to 0} \frac{\sum_{i=1}^{W(\epsilon)} p_i \ln p_i}{\ln(\frac{1}{\epsilon})} \tag{4.6}$$

gdzie $W(\epsilon)$ jest liczbą elementów trajektorii, zawartych w l-wymiarowej kostce której krawędź jest równa ϵ, a $p_i = \frac{N_i}{N}$ jest częstością występowania $i - tego$ elementu , a N_i – liczbą punktów w i-tej hiperkostce, N -liczbą punktów trajektorii.

W dalszej analizie zastępujemy wymiar informacyjny (4.6) jego przybliżeniem, jakimi są wymiar pudełkowy lub pojemnościowy. W [64] wymiar pudełkowy wyprowadzono ze wzoru przybliżonego.

Wykorzystamy to podejście do przybliżenia wymiaru. Niech $N(R, \epsilon)$ będzie najmniejszą liczbą K-wymiarowych kostek o krawędzi równej ϵ, które pokrywają trajektorię 1.4 $Tr \subset \mathcal{X}$, i $\mathcal{X}$ jest l-wymiarową przestrzenią przeszukiwania. Rozważmy przypadek, kiedy $\epsilon = 2^{-k}$ i podzielmy długość krawędzi kostki przez dwa. Wówczas następujący iloraz przybliża wymiar pudełkowy trajektorii ([64])

$$D_c(R) \approx \frac{\log_2 N(R, 2^{-(k+1)}) - \log_2 N(R, 2^{-k})}{\log_2 2^{k+1} - \log 2^k} = \log_2 \frac{N(R, 2^{-(k+1)}}{N(R, 2^{-k})}, \tag{4.7}$$

ponieważ $\log_2 x = \log_2 e \ln x$. Wyrażenie (4.7) pozwala na wyznaczenie wymiaru pudełkowego przy wzrastającej liczbie kostek, gdy ich krawędzie maleją poprzez dzielenie przez dwa.

4.3. Badania eksperymentalne

Bazując na tych wynikach teoretycznych przedstawiamy badania empiryczne nad klasyfikacją algorytmów, w których jako niezmienniki (invarianty) klasyfikacji przyjęto entropię i wymiar fraktalny oraz jego przybliżenia: wymiar pudełkowy i informacyjny.

Przeprowadzono eksperyment obliczeniowy w celu podania odpowiedzi na kilka pytań. Pierwsze z nich to pytanie: jakie parametry dobrać do algorytmu, aby

skutecznie i jak najbardziej efektywnie znajdował interesujące nas rozwiązanie? Ogólniejsze pytanie: który algorytm jest lepszy, gorszy lub równorzędny inaczej sparametryzowanemu algorytmowi? Jako miary oceny postanowiono użyć wymiaru fraktalnego, który z definicji jest jedną z charakterystyk trajektorii układów dynamicznych. Podjęto próbę klasyfikacji algorytmów genetycznych na podstawie szczególnego wymiaru pojemnościowego, tzw. wymiaru pudełkowego, ich trajektorii [51, 50].

Jest on uważany za pewne przybliżenie klasycznego wymiaru fraktalnego Hausdorffa. Zrobiono to mając nadzieję, że właśnie ten wskaźnik pozwoli nam porównać różne algorytmy, stwierdzając, że skoro mają taki sam wymiar fraktalny, to niezależnie od różnych parametrów początkowych, są one sobie równoważne. Proponowana próba klasyfikacji jest oparta, jak mówiono już wcześniej, na modelu algorytmu jako procesu Markowa. Procesy Markowa są równoważne przesunięciom Bernoulliego. Do ich analizy zatem, możemy wykorzystać aparat analityczny wypracowany dla tych przesunięć. To stwierdzenie jest podstawą metody obliczeniowej. Propozycję klasyfikacji prowadzono na podstawie właśnie wymiaru fraktalnego trajektorii algorytmu.

Hipoteza, że można w ten właśnie sposób klasyfikować algorytmy genetyczne opiera się na twierdzeniu Ornsteina, które stwierdza, że entropia jest niezmiennikiem izomorfizmu dla przesunięć Bernoulliego oraz entropia i wymiar fraktalny mają tę samą wartość [61]. Napisano własne oprogramowanie do realizacji algorytmów wyznaczania wymiaru ich trajektorii .

Pozwoliło to na sprzężenie go w sposób w pełni zautomatyzowany z zewnętrznym programem wyliczającym wymiar fraktalny poszczególnych trajektorii algorytmów. Umożliwiło to generowanie podsumowujących plików i raportów zbierających wszystkie interesujące nas faktory i wskaźniki. Trzecim, równie istotnym głosem przemawiającym za takim rozwiązaniem, był fakt, iż budowa własnego systemu pozwoliła na swobodne dokładanie do niego nowych modułów.

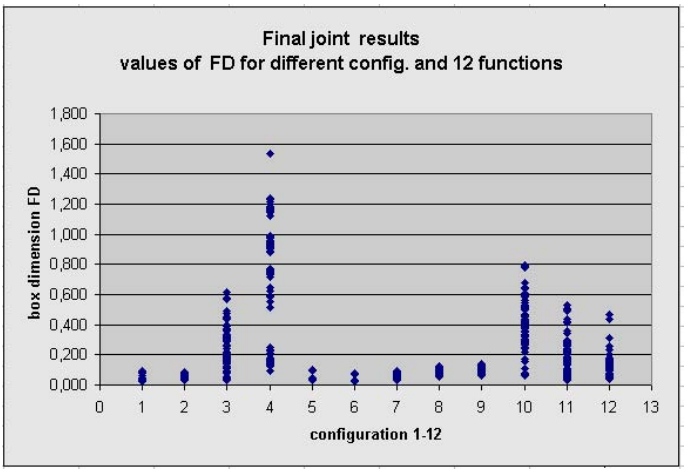
Cały system został napisany w Javie i jego idea została oparta na interfejsach, po to, by w łatwy i przejrzysty sposób można było dodawać nowe moduły, czyli nową funkcjonalność dla całego systemu. Dlatego też zostały zaimplementowane następujące interfejsy (przepisy) do tworzenia nowych klas:
Chromosom - do implementowania różnych reprezentacji chromosomu
Estymator - do implementowania różnych funkcji oceny i różnych problemów
Selector - do implementowania różnych operatorów selekcji
Mutator - do implementowania różnych operatorów mutacji
Reproductor - do implementacji różnych operatorów krzyżowania
Improver - do implementacji różnych opcji poprawiających efektywność algorytmów, czyli usprawnień algorytmu.

W systemie zaimplementowano 10 funkcji testowych. Większość z nich operuje na dziesięciu zmiennych, trzy operują tylko na dwóch. W systemie zaimplementowano zarówno stochastyczne, jak i deterministyczne operatory selekcji.

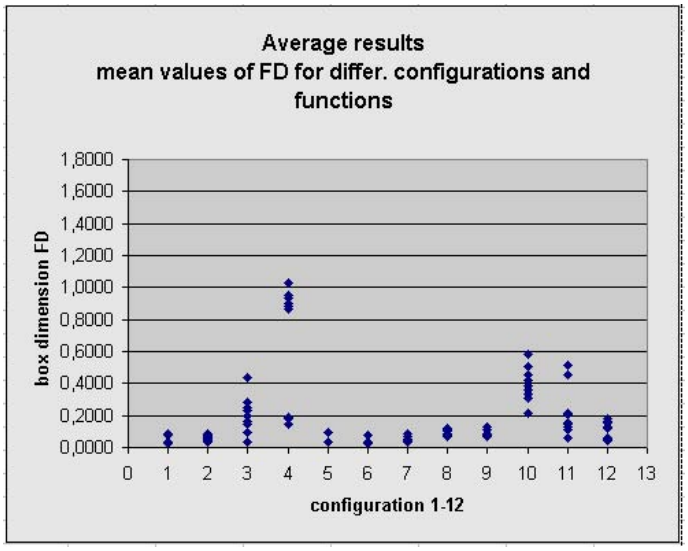
Podstawowy moduł systemu FDBin oferuje wyliczanie wymiaru pudełkowego dowolnego zbioru punktów w n-wymiarowej przestrzeni. Warunkiem jest, aby każdy z elementów tego zbioru wejściowego był zakodowany binarnie w odpowiedni sposób. Proces wyliczania wymiaru pudełkowego jest dwuetapowy.

Wyniki przedstawione są na wykresach zamieszczonych poniżej.

Gdy przyjrzymy się bliżej uzyskanym zestawieniom, zauważymy, że w większości przypadków wartości wymiaru pudełkowego nie są przypadkowe. Przy tej samej funkcji oraz dla tej samej konfiguracji liczby te są wyraźnie skupione przy jednej



Rysunek 4.1. Wymiar fraktalny - wyniki zbiorcze



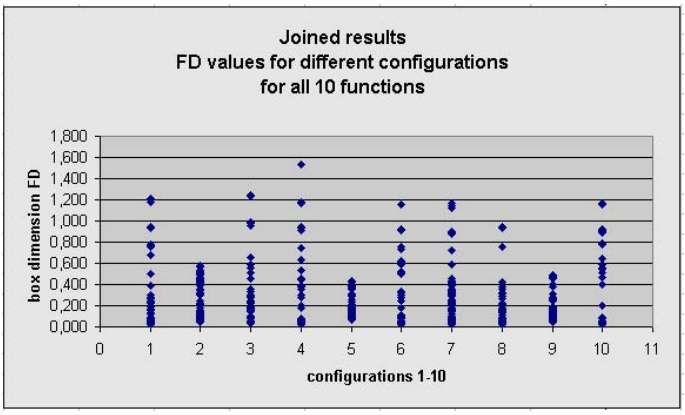
Rysunek 4.2. Wymiar fraktalny - wyniki uśrednione

wartości. Cała próbka dziesięciu niezależnych przebiegów algorytmu osiąga bardzo zbliżone wyniki wymiaru pudełkowego. Dowodzi to, jak już wspomnieliśmy wyżej, że wyliczanie wymiaru fraktalnego jest zasadne. Idąc dalej i poszerzając nasz horyzont obserwacji zauważamy, iż nawet przy różnych funkcjach ta sama konfiguracja nadal zachowuje pewną prawidłowość. Wciąż widzimy wyraźne skupienie wartości wymiaru pudełkowego.

Wyniki zbiorcze działania zaprojektowanego systemu do wyznaczania wymiaru pudełkowego trajektorii algorytmów są przedstawione na 3 rysunkach.

4.3.1. Klasyfikacja

Postać graniczna algorytmu pozwala na dokładniejsze analizy związane z klasyfikacją algorytmów, a nawet wprowadzenie pewnej hierarchii w klasyfikacji. Równoważność algorytmów można rozpatrywać na gruncie macierzy przejścia, rozkładu granicznego, operatora granicznego, wektorów własnych operatora opisującego algorytm oraz entropii i porównywać z klasyfikacją taksonomiczną lub opartą na twierdzeniu



Rysunek 4.3. Wymiar fraktalny - wyniki z chmurą trajektorii

o schematach albo metodach statystycznych. Ważną byłaby klasyfikacja oparta o uporządkowanie (trajektorię) populacji (permutację populacji), jak i kluczową teoretycznie entropię i wymiar fraktalny oraz jego przybliżenia.

- Dwa algorytmy genetyczne są równoważne, jeżeli mają tę samą macierz przejścia (operator Markowa).
- Dwa algorytmy genetyczne są równoważne, jeżeli ich operatory przejścia mają ten sam rozkład graniczny.
- Dwa algorytmy genetyczne są równoważne, jeśli generują ten sam operator graniczny.
- Algorytmy genetyczne są równoważne, jeśli mają tę samą entropię trajektorii.
- Dwa algorytmy genetyczne są równoważne, jeśli ich trajektorie mają ten sam wymiar fraktalny (Hausdorffa, pudełkowy, informacyjny, itd.).
- Dwa algorytmy genetyczne są równoważne, jeśli generują te same permutacje populacji (uporządkowanie).

Klasyfikacja taka wykorzystuje wielkości probabilistyczne, takie jak rozkład prawdopodobieństwa, rozkład graniczny, wektory własne, macierze Markowa, wymiar fraktalny i wreszcie permutacja populacji oraz entropia. Są to wielkości probabilistyczne. Problemem jest, czy te wskaźniki klasyfikacji powinny być odpowiednio równe, a kiedy możemy dopuścić ich odchylenia. Ustalenie, które wielkości wskaźników i jakie zakresy tych odchyleń są odpowiednie dla każdego z nich, może być przedmiotem analizy.

ROZDZIAŁ 5

Podsumowanie

5.1. Klasyfikacja w literaturze

Wyniki D. Battle, M. Vose [6], oraz T. Jansen [40] są cząstkowymi wynikami potwierdzającymi nasze rozważania i wyniki dotyczące równoważności (izomorfizmu) algorytmów genetycznych (ewolucyjnych) [60]. Opisane tam wyniki przyjmują jako niezmienniki izomorfizmów występowanie takich samych schematów lub podobieństwo wariancji i kowariancji oraz korelacji trajektorii algorytmów genetycznych oraz innych charakterystyk statystycznych. Podobieństwo schematów i parametrów statystycznych prowadzi do tego, że entropia tych procesów jest zbliżona lub taka sama, a zatem nie tylko nie przeczy, ale i potwierdza nasze wyniki, że entropia jest niezmiennikiem izomorfizmu. Wyniki zamieszczone w tych artykułach są cząstkowymi rezultatami potwierdzającymi równoważność algorytmów z tą samą entropią.

Zagadnienie dyskretyzacji a algorytmy

Dla zadanego problemu optymalizacyjnego generowanie procesu ewolucyjnego jest poprzedzone nie tylko definicją funkcji wartościującej (dopasowania), ale też jej dziedziny, która zależy od przyjętej dyskretyzacji (dokładności obliczeń), tj. kroku próbkowania (często ciągłej) przestrzeni argumentów. To powoduje, że w zasadzie mamy do czynienia z różnymi procesami ewolucyjnymi dla różnej dyskretyzacji tego samego problemu optymalizacyjnego. Z tym jest związany już zaobserwowany fakt, że entropia generowanego procesu może zależeć od sposobu dyskretyzacji i nie jest to niczym zaskakującym. Przy zmianie dyskretyzacji możemy zmieniać prawdopodobieństwa pojawiania się pewnych punktów, a zatem i entropię procesu [80]. Przy pewnym kroku dane zadanie może być stabilne, a przy innym staje się niestabilne. Następuje zatem zmiana rozkładu prawdopodobieństwa pojawiania się punktów, a więc zmienia się entropia. Fakt ten pokazuje, że nawet niewielka zmiana dziedziny funkcji wartościującej w problemie optymalizacyjnym może spowodować, że algorytm optymalizacyjny zachowuje się zupełnie inaczej. Już przy małej modyfikacji zadania, dobry dotychczas algorytm staje się nieodpowiedni.

5.2. Główne wyniki rozprawy

W rozprawie przedstawiono postać graniczną operatora algorytmu genetycznego. Uzyskanie postaci granicznej algorytmu genetycznego pozwoliło na przedstawienie

optymalnego algorytmu genetycznego w sensie probabilistycznym. Postać optymalnego algorytmu genetycznego w sensie probabilistycznym daje wskazówki, jak projektować algorytmy genetyczne. Może być ona podstawą do dalszych badań zarówno teoretycznych, jak i eksperymentalnych. Ciekawą własnością operatora granicznego przedstawioną w pracy jest fakt, że kolumny [1] macierzy operatora są sobie równe i są równe wektorowi własnemu odpowiadającemu największej wartości własnej równej jeden.

Wynik ten pozwala na sformułowanie postaci najlepszego algorytmu genetycznego w sensie probabilistycznym. Daje to też podstawy teoretyczne do podjęcia badań nad realizacją optymalnych algorytmów genetycznych. Wynik ten wskazuje, że najlepszym algorytmem jest algorytm, w którym daną populację można otrzymać z tym samym prawdopodobieństwem z każdej innej populacji.

Fakt, że operator graniczny jest reprezentowany przez macierz o tych samych kolumnach pozwolił też na udowodnienie, że algorytm genetyczny jest równoważny przesunięciu Bernoulliego, a więc można go klasyfikować na mocy entropii i wymiaru fraktalnego. Dowód, że algorytm genetyczny jest równoważny przesunięciu Bernoulliego, ma szersze znaczenie i pozwala na uzasadnione wykorzystanie całego dorobku metod i wyników wypracowanych przy badaniach przesunięć Bernoulliego do badania algorytmów genetycznych i ewolucyjnych. Otwiera to nowe, rozległe i wartościowe pole badań.

[1] dla macierzy z operacją lewostronną będą to jednakowe wiersze.

Literatura

[1] A. Agapie, *Theoretical Analysis of Mutation Adaptive Evolutionary Algorithms,* Evolutionary Computation **9** (2) pp.127-146.

[2] Amigo J. M., Szczepański J., Wajnryb M., Sanchez–Vives M.V., *Estimating the Entropy of Spike Trains Via Lempel–Zif Complexity*, Neural Comput. **16**, 2004, 716–736.

[3] Argyris J., Faust G., Haase M., *An Exploration of Chaos,* North Holland, 1994.

[4] Badii R. and Politi, *On the Fractal Dimention of Filtered Chaotic Signals,* in Mayer-Kress, Dimensions and Entropies. Springer-Verlag , 1985 .

[5] Barnsley M. F.: Lecture notes on iterated function systems, in *Chaos and Fractals. The Mathematics Behind the Computer Graphics, Proc.Symp. Appl. Math.*, vol. 39, R. L. Devaney and L. Keen (eds.) American Mathematical Society, Providence, Rhode Island, pp. 127-144, 1989.

[6] D.L. Battle and M.D. Vose, *Isomorphisms of Genetic Algorithms,* Artifitial Intelligence 60 155-165, Elsevier Science Publishers, 1993.

[7] D. Benedetto, E. Cagliotti and V. Loreto, *Language Trees and Zipping,* Physical Reviev Letters, Vol.**88**, n.4, 2002.

[8] Hans Georg Beyer,*The Theory of Evolution Strategies,* Springer, 2001.

[9] Box G.E.P., *Evolutionary operation: A method for increasing industrial productivity* Appl. Statistics, vol. **VI,** no.2, pp. 81–101, 1957.

[10] Bremerman H.J., *Optimization through evolution and recombination*, in Self–Organizing Systems, M. C. Yovits et al. Eds. Washington, DC, Spartan, 1962.

[11] Burgin E., Eberbach E., *Evolutionary Optimization in an Algorithmic Setting*, arXiv:cs/0611077 [pdf].

[12] G. H. Choe *Computational Ergodic Theory*, Springer,Heidelber, New York 2005.

[13] P. Cichosz, *Systemy uczące się*, WNT, Warszawa 2000.

[14] J. Crutchfield, D. Feldman, *Regularities Unseen, Randomness, Observed Levels of Entropy Convergence.* Santa Fe Institute Working Paper 01-02-012

[15] J. Cytowski, *Algorytmy genetyczne: podstawy i zastosowania*, Seria: Problemy Współczesnej Nauki - Teoria i Zastosowania Nr 18, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1996.

[16] K. Deb, S. Agrawal, *Understanding Interactions among Genetic Algorithms Parameters,* in: Foundations of Genetic Algorithms-5, Ed. W. Banzhaf and C. Reeves, Morgan Kaufman, 1999.

[17] K. De Jong, *Evolutionary Computation: Recent Developments and Open Issues, in Evolutionary Algorithms in Engineering and Computer Science,* Ed. K. Miettinen, P. Neittaanmaki, M.M. Makela, J, Periaux, John Wiley & Sons, 1999.

[18] P. Del Moral and L. Miclo,*Asymptotic Results for genetic Algorithms with Applications to Nonlinear Estimation,* in Leila Kallel, Bart Nauds, Alex Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Springer, 2001.

[19] S. Droste, T.Jansen, I. Wegener, *Perhaps not a free lunch but at least a free appetizer.* Genetic and Evolutionary computation Conference (GECCO '99), pp.833-839, Morgan Kaufman, 1999.

[20] Eiben A.E., Aarts E.H.L., Van Hee K.M., *Global Convergence of Genetic Algorithms; On Infinite Markov Chain Analysis,* w: Schwefel H.P., Manner R.(Eds.), Proceedings of the First International Conference on Parallel Problem Solving of Natute., Lecture Notes in Computer Science, Vol.496, Springer-Verlag, 1991.

[21] Martin N., England J., *Mathematical Theory of Entropy,* Addison-Wesley Publishing Company, 1981.

[22] English, T., *No More Lunch: Analysis of Sequential Search*,in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, pp. 227-234. 2004, http:/BoundedTheoretics.comĆEC04.pdf.

[23] Falconer K.J.: Fractal geometry *Math. Found. Appl.* John Wiley, Chichester, pp. 15 -25, 1990.

[24] Friedberg R.M., *A learning machine: Part 1,* IBM J., vol.**2,** no.1, pp. 2–13, Jan. 1958.

[25] Friedman N.A., Ornstein D.S.: *On isomorphisms of weak Bernoulli transformations,* Adv. in Math. , **5**, pp. 365-394, 1970.

[26] S. W. Fomin, I.P. Kornfeld, J.G.Sinaj, *Teoria ergodyczna,* PWN, Warszawa, 1987.

[27] G. Frizelle, Y.M. Suhov, *An entropic measurement of queueing behaviour in a class of manufacturing operations.* Proc. Royal Soc. London A (2001) 457, 1579-1601.

[28] R.Galar *Przez adaptacyjne zbocza, wierzchołki, siodła i grzbiety* , IV Krajowa Konferencja "Algorytmy Ewolucyjne i Optymalizacja Globalna", Lądek Zdrój, 5-8 czerwca 2000, str. 83–90.

[29] J.Garnier, L.Kallel, *Statistical Distribiution of the Convergence Time of Evolutionary Algorithms for Long Path Problems,* IEEE Transaction on Evolutionary Computation, vol.**4**, no1, April 2000.

[30] D. E. Goldberg, *Algorytmy genetyczne i ich zastosowania*, WNT,Warszawa, 1995.

[31] Goldberg D.E., Segrest P., *Finite Markov Chain Analysis of Genetic Algorithms,* Greffenstette J.J. Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[32] Greffenstette J.J., *Optimization of Control Parameters for Genetic Algorithms,* IEEE Transactions on Systems, Man, and Cybernetics, Vol.16, No.1, 1986.

[33] Harrison J.: An introduction to fractals, in *Chaos and Fractals. The Mathematics Behind the Computer Graphics, Proc.Symp. Appl. Math.*, vol. 39, R. L. Denamney and L. Keen (eds.) American Mathematical Society, Providence, Rhode Island, pp. 107-126, 1989.

[34] Ho, Y.C., Pepyne, D.L. (2002), *Simple Explanation of the No-Free-Lunch Theorem and Its Implications,*Journal of Optimization Theory and Applications, **115,** 549.

[35] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[36] R. B. Hollstien, *Artificial Genetic Adaptation in Computer Control Systems*, Ph.D. Thesis, University of Michigan, 1971.

[37] Ch. Igel, M. Toissant, *On Classes of Functions for which No Free Lunch results Hold,* arXiv:cs.NE/0108011 v1, 21 Aug 2001,

[38] Igel, C., and Toussaint, M., *A No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions*, Journal of Mathematical Modelling and Algorithms **3,** 313–322, 2004.

[39] Jakubowski J. Sztencel R., *Wstęp do teorii prawdopodobieństwa* SCRIPT, Warszawa, 2000.

[40] T. Jansen *On Classification of Fitness Function*, Leila Kallel, Bart Naudts, Alex Rovers (Eds.) Theoretical Aspects of Evolutionary Computing, Springer 2001, ISBN 3-540-67396-2

[41] M.Iosifescu, *Skończone procesy Markowa i ich zastosowania,* PWN, Warszawa, 1988.

[42] L. Kallel, B.Nauds, and C. Reeves, *Properties of Fitness Functions and Search Landscapes,* in Leila Kallel, Bart Nauds, Alex Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Springer, 2001.

[43] L. Kallel, B. Naudts, *Candidate Longpaths for the Simple genetic Algorithms,* in:Foundations of Genetic Algorithms-5, Ed. W. Banzhaf and C. Reeves, Morgan Kaufman, 1999.

[44] van Kemenade C. K., *Recombinative Evolutionary Search* Universiteit Leiden, 1999.

[45] P. Kieś i Z. Michalewicz, *Podstawy algorytmów genetycznych*, Matematyka Stosowana. Matematyka dla Społeczeństwa, **1** (44), 2000, 68–91.

[46] Kieś P., *Dimension of Attractors Generated by a Genetic Algorithms,* in Proc. of Wokshop Intelligent Information Systems IX, held in Bystra, Poland, June 12-16, pp.40-45, 2000.

[47] Kocarev L., Szczepański J., *Finite-Space Lyapunov Exponents and Pseudochaos*, Physical Review Letters, **93** 234101, (2004).

[48] Kocarev L., Szczepański J., Amigo J. M., Tomovski I., *Discrete Chaos–I: Theory* IEEE Transaction on Circuits and Systems–1: regular Papers, vol **53**, 6, 2006.

[49] Konar Amit *Computational Intelligence, Principles, Techniques and Applications*, Springer-Verlag, Berlin Heildelberg, 2005.

[50] W. Kosinski, S. Kotowski, *Fractal Dimension of Trajectory as Invariant of Genetic Algorithms,* in: Formal Methods and Intelligent Techniques in Control, Decision making, Multimedia and Robotics, Proceedings of the 2-nd International Conference, Polish-Japanese Institute of Information Technology, Warsaw 2000, pp. 58–65.

[51] S. Kotowski, W. Kosiński, Z. Michalewicz, J. Nowicki, B. Przepiórkiewicz, *Fractal dimension of trajectory as invariant of genetic algorithms,* ICAICS, 9-th International Conference on Artifical Intelligence and Soft Computing, 2008, LNAI, Springer, Berlin, Heidelberg, vol. 5097, 2008, pp. 414 - 425 .

[52] A. Lasota, *Asymptotyczne własności półgrup operatorów Markowa*, Matematyka Stosowana. Matematyka dla Społeczeństwa, **3** (45), 2002, 39–51.

[53] Lobo F. G., Lima C, F,. Michalewicz Z., *Parameter Setting in Evolutionary Algorithms* Springer, 2007.

[54] E.Lutton, *Genetic Algorithms and Fractals,* in Evolutionary Algorithms in Engineering and Computer Science, Ed. K. Miettinen, P. Neittaanmaki, M.M. Makela, J, Periaux, John Wiley & Sons, 1999.

[55] Mayer-Kress, *Dimentions and Entropies.* Springer-Verlag, 1985.

[56] Z. Michalewicz, *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT, Warszawa, 1996.

[57] Michalewicz Z.: *Genetic Algorithms + Data Structures = Evolution Programs,* 3rd, rev. edition, Springer, Berlin, Heidelberg *et al.*, 1996.

[58] Z. Michalewicz, D. Fogel, *How to Solve It: Modern heuristics*, Springer, 2000.

[59] Michalewicz Z. and Martin Shmidt, *Evolutionary Algorithms and Constrained Optimization,*in Evolutionary Optimization Ed. By R. Sarker, M. Mohammadian, Xin Yao, Kluwer Academic Publishers, 2002.

[60] B.Naudts, L. Kallel, *A Comparison of Predictive Measures of Problem Difficulty in Evolutionary Algorithms,* IEEE Transaction on Evolutionary Computation, vol.**4,** no 1, April 2000,

[61] Ornstein D.S.: *E*rgodic theory, randomness and dynamical systems, Yale Univ. Press, 1974.

[62] Ossowski A., *Statistical and Topological Dynamics of Evolutionary Algorithms,* in Proc. of Wokshop Intelligent Information Systems IX, held in Bystra, Poland, June 12-16, pp.94-103, 2000.

[63] Paszyńska A., *Projektowanie wspomagane komputerowo a problemy zbieżności algorytmów genetycznych* Praca doktorska pod kierunkiem: dr hab. Ewy Grabskiej, Kraków, 2007.

[64] Peitgen H.O., Jurgens H., Saupe D., *Granice Chaosu, Fraktale,* PWN, Warszawa 1996.

[65] Prugel-Bennett, *On the Limit Of Long Strings*, in:Foundations of Genetic Algorithms-5, Ed. W. Banzhaf and C. Reeves, Morgan Kaufman, 1999.

[66] Rana S., *Examining the Role of Local Optima and Schema Processing in Genetic Search* Colorado State University, Fort Collins, Colorado, Summer 1999.

[67] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, Springer, Berlin Heildelberg New Tork, 2006.

[68] J. E. Rowe, *The dynamical system models of the simple genetic algorithm*, w Theoretical Aspects of Evolutionary Computing, Leila Kallel, Bart Naudts, Alex Rogers (Eds.), Springer, 2001, pp.31–57.

[69] R. Rudnicki, *On asymptotic stability and sweeping for Markov operators*, Bull. Polish Acad. Sci. Math., **43** (1995), 245–262.

[70] G. Rudolf, *Convergence Analysis of Canonical Genetic Algorithms,* IEEE Transactions on Neural Networks, vol.**5,** no.1 January 1994,

[71] W. Siedlecki, J. Sklansky, *On automatic feature selection*, Int. J Pattern Recognition and Artificial Intelligence, **2** (2), 197–220, 1988.

[72] J.D. Shaffer, M. Mani, L. Eshelman, K. Mathias, *The Effect of Incest Prevention on Genetic, Drift,* in:Foundations of Genetic Algorithms-5, Ed. W. Banzhaf and C. Reeves, Morgan Kaufman, 1999.

[73] C.A. Schippers, *Multi Parent Scannig Crossover and Genetic Drift,* in Leila Kallel, Bart Nauds, Alex Rogers (Eds.), *Theoretical Aspects of Evolutionary Computing*, Springer, 2001.

[74] Paul C. Shields, *The Ergodic Theory of Discrete Sample Paths,* Graduate Studies in Mathematics Vol.13, American Mathematical Society, 1996.

[75] R. Schaefer, *Podstawy genetycznej optymalizacji globalnej*, Wydawnictwo Uniwersytetu Jagielońskiego, Kraków 2002.

[76] R. Schaefer, *Foundations of Global Genetic Optimization* Springer, Series: Studies in Computational Intelligence , Vol. 74, 2007, XII, 222 p. ISBN: 978-3-540-73191-7

[77] Schaffer, Cullen (1994), *A conservation law for generalization performance,* International Conference on Machine Learning, H. Willian and W. Cohen, Editors. San Francisco: Morgan Kaufmann, pp.295-265.

[78] Schaffer J., Caruana R., Eshelman L., Das R., *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization,* w Shaffer J., Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[79] J.I. Shapiro, *Statistical Mechanics Theory of Genetic Algorithms*, in: Leila Kallel, Bart Nauds, Alex Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Springer, 2001.

[80] Sobczyk K., *Information Dynamics: Premises, Challenges and Results*, Mechanical Systems and Signal Processing, **15** (3) 2001, 475–498.

[81] Suzuki J., *A Markov Chain Analysis on A Genetic Algorithm*, Proceedings of the Fifth International Conference on genetic Algorithms, Morgan Kaufmann Publishers, 1993,146–153.

[82] Sharkovsky A.N., Kolyada S.F., Sivak A.G. and Fedorenko V.V., *Dynamics of One-Dimentional* Maps, Kluwer Academic Publishers, 1997.

[83] C. Schumacher, M.D. Vose, L. D. Whitley, *The No Free lunch and Problem Description Length,* in. Genetic and Evolutionary Computation Conference (GECCO2001), pp.565-570, Morgan Kaufman, 2001.

[84] Sirag D.J., Weisser P.T., *Toward a Unified Thermodynamic Genetic Operator*, w Greffenstette J.J. Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[85] J. Socała, *Asymptotic behaviour of the iterates of nonnegative operators on a Banach lattice*, Ann. Polon. Math., **68** (1), 1998, 1–16.

[86] J. Socała, W. Kosiński, *On convergence of a simple genetic algorithm,* ICAICS, 9-th International Conference on Artifical Intelligence and Soft Computing, 2008, LNAI, vol. 5097, Springer, Berlin, Heidelberg, 2008, pp. 489–498.

[87] J. Socała, W. Kosiński, *Lower-bound function method in the converegence analysis of genetic algorithms,* (in Polish: Zastosowanie metody funkcji dolnej do badania zbieżności algorytmów genetycznych,) Matematyka Stosowana. Matematyka dla Społeczeństwa, PTM, Warszawwa, **8** (49), 2007 , 33–44.

[88] J. Socała, W. Kosiński, S. Kotowski, *O asymptotycznym zachowaniu prostego algorytmu genetycznego,* Matematyka Stosowana. Matematyka dla Społeczeństwa, PTM, Warszawa, **6 (47), 2005, 70–86.**

[89] W. M. Spears, *Evolutionary Algorhms*, The Role of Mutation and Recombination, Springer 2000.

[90] W. M. Spears and L. De Jong, *Dinning with Gas: Operator Lunch Theorems,* in:Foundations of Genetic Algorithms-5, W. Banzhaf and C. Reeves (Eds.), Morgan Kaufman, 1999.

[91] P. Stagge and C.Igel, *Structure Optimizations and Isomorphisms,* in: Leila Kallel, Bart Nauds, Alex Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Springer, 2001.

[92] C.A. Schippers, *Multi Parent Scannig Crossover and Genetic Drift,* in Leila Kallel, Bart Nauds, Alex Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Springer, 2001.

[93] C.R. Stephens, H, Waelbbroek, R. Aguirre, *Schemata as Building Blocks:Does Size Matter?,* in:Foundations of Genetic Algorithms-5, Ed. W. Banzhaf and C. Reeves, Morgan Kaufman, 1999.

[94] Strauss C. E., Wolpert D.H. and Wolf D.R., *Alpha, evidence, and the entropic prior*, w: Maximum Entropy and Bayesian Methods, Reading, Ma, Addison Wesley, 1992.

[95] Syswerda G., *Uniform Crossoverin genetic Algoruthms*, Proceedings of the Third International Conference on Genetic Algorithms, eds. J. D. Schaffer, 1989, Morgan Kaufman.

[96] W. Szlenk, *An Introduction to the Theory of Smooth Dynamical Systems.*, PWN, Warszawa, John Wiley&Sons, Chichester, 1984.

[97] F. Vavak, T.C. Fogarty, *A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Nonstationary Enviroments.* Proc. of the Society for the Study of Artificial Intelligence and Simulation of Behaviour Workshop on Evolutionary Computing, 1996, pp. 301-307.

[98] D. Volpert, W. G. Macready, *No Free Lunch theorems for Optimization,* IEEE Transaction on Evolutionary Computation, vol.**1** no 1, April 1997.

[99] Wolpert, D.H., and Macready, W.G. *Coevolutionary free lunches,* IEEE Transactions on Evolutionary Computation, **9**,(6),(2005), 721-735.

[100] Vose M.D., *The Simple Genetic Algorithms,* MIT Press, Boston, 1999.

[101] Vose M.D., *Modelling Simple Genetic Algorithms*, Evolutionary Computation, **3** (4) 453-472, 1996.