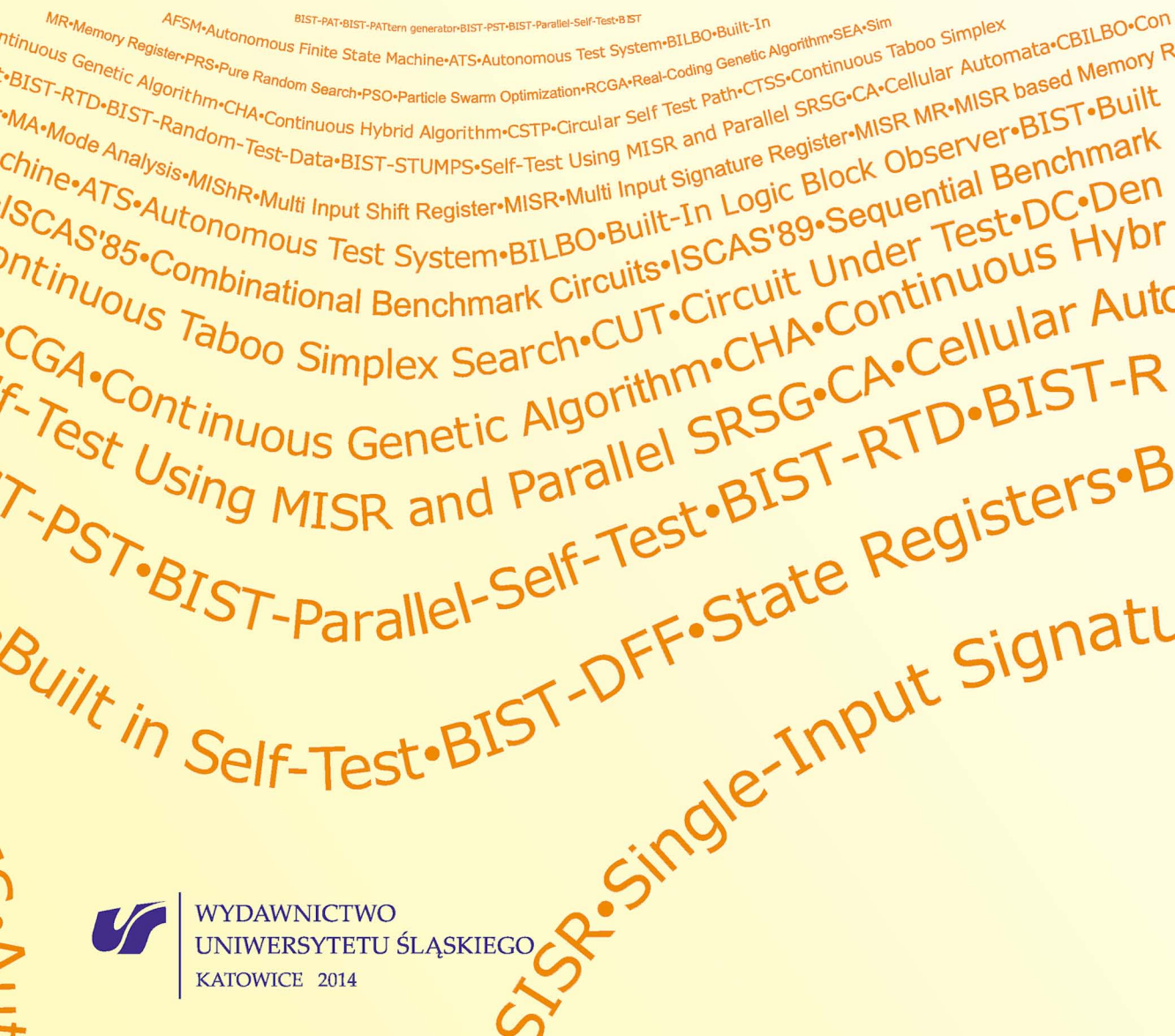
Nowe ujęcie wybranych zagadnień optymalizacji

Ireneusz Gościniak



WYDAWNICTWO
UNIWERSYTETU ŚLĄSKIEGO
KATOWICE 2014

Nowe ujęcie wybranych zagadnień optymalizacji

Gdy uruchamiamy sztuczny proces ewolucyjny, stawiamy się w roli stwórcy i to my określamy reguły symulacji, co przypomina ustawianie fizyki świata. Możemy chcieć zwrócić szczególną uwagę na konkretne mechanizmy naturalnej ewolucji albo rozważać elementy nie występujące w systemach biologicznych (np. kojarzenie więcej niż dwóch rodziców). Możemy nawet odejść zupełnie od praw natury ...

Zbigniew Michalewicz, David B. Fogel:
Jak to rozwiązać czyli nowoczesna heurystyka
Warszawa: WNT, 2006, s. 188

NR 3204

Ireneusz Gościniak

Nowe ujęcie wybranych zagadnień optymalizacji

Wydawnictwo Uniwersytetu Śląskiego • Katowice 2014

Redaktor serii:	Informatyka i Inżynieria Biomedyczna
	Piotr Porwik
Recenzent:	Sławomir Wierzchoń

Redaktor
Justyna Szmyt

Projekt okładki
Hanna Gościniak

Redaktor techniczny
Małgorzata Pleśniar

Korekta
Justyna Szmyt

Łamanie
Ireneusz Gościniak

ISSN 0208-6336
ISBN 978-83-8012-046-4
(wersja drukowana)
ISBN 978-83-8012-047-1
(wersja elektroniczna)

Wydawca
Wydawnictwo Uniwersytetu Ślskiego
ul. Bankowa 12B, 40-007 Katowice
www.wydawnictwo.us.edu.pl
e-mail: wydawus@us.edu.pl

Wydanie I. Ark. druk. 11,25 Ark. wyd. 13,5 Papier offset. kl. III, 90 g Cena 28 zł (+VAT)

Druk i oprawa: „TOTEM.COM.PL Sp.z o.o." Sp.K.
ul. Jacewska 89
88-100 Inowrocław

Spis treści

Ważniejsze symbole

$A(\varepsilon)$	–	pokrycie
$Ab = [ab_1, ..., ab_n]$	–	wektor opisujący przeciwciało
$Ag = [ag_1, ..., ag_n]$	–	wektor opisujący antygen
D_f	–	wymiar fraktalny (indeks Hausdorffa)
D_I	–	informacyjny wymiar fraktalny
D_{M-B}	–	wymiar fraktalny Minkowskiego-Bouliganda
D_q	–	uogólniony wymiar fraktalny
$E_i = [e_{i1}, e_{i2}, ..., e_{id}]$	–	wektor opisujący drapieżnika
F, f	–	funkcja celu, kryterium
$f(\alpha)$	–	widmo osobliwości (spektrum multifraktalne)
$f_i(X(t))$	–	stan i-tego wyjścia układu sprzężenia zwrotnego dla wektora X w chwili t
$F(X(t)) = [f_0(X(t)), f_1(X(t)), ..., f_{n-1}(X(t))]$	–	wektor funkcji
g	–	osobnik, genom, element przestrzeni przeszukiwania $\mathbb{G}$
$H(\mathfrak{U})$	–	entropia zmiennej losowej $\mathfrak{U}$
I	–	macierz mapowania wejść
Map	–	mapowanie genotyp – fenotyp
$N(l)$	–	liczba segmentów o skali l
N_I	–	wektor negacji wejść
N_O	–	wektor negacji wyjść
O	–	macierz mapowania wyjść
Op	–	operacja przeszukiwania
$p(\mathfrak{u})$	–	funkcja prawdopodobieństwa zmiennej losowej $\mathfrak{U}$

S	–	zbiór przestrzeni euklidesowej
T	–	macierz połączeń
$U = [u_0, u_1, ..., u_n]$	–	wektor wejściowy
v	–	funkcja przystosowania $v : \mathbb{X} \rightarrow \mathbb{V}$
$v\,(t)$	–	prędkość
$v'\,(t)$	–	przyspieszenie (zmiana prędkości w jednostce czasu)
$V_i^E = [v_{i1}^E, v_{i2}^E, ..., v_{id}^E]$	–	wektor prędkości drapieżników
V_i^Z	–	wektor prędkości zdobyczy
$X = [x_0, x_1, ..., x_n]$	–	wektor stanu rejestru
$\hat{x}$	–	ekstremum lokalne
$x^\star$	–	rozwiązanie, element zbioru rozwiązań $X^\star$
$X^\star$	–	zbiór rozwiązań
Y	–	przeciwdziedzina
$Z_i = [z_{i1}, z_{i2}, ..., z_{id}]$	–	wektor opisujący zdobycz
Z_q	–	funkcja podziału (q-ty moment)
α	–	wskaźnik osobliwości (wykładnik Höldera)
α_{max}	–	miejsce zerowe funkcji widma multifraktalnego
α_{min}	–	miejsce zerowe funkcji widma multifraktalnego
α_s	–	przedziałowy wykładnik Höldera
ε	–	współczynnik skali
$\mu\,(B)$	–	probabilistyczna miara określona jako prawdopodobieństwo wystąpienia obiektu w pudełku $B_i\,(l)$, miara multifraktalna (przeskalowana miara multifraktalna)
$\Theta = [\vartheta_1, \vartheta_2, ..., \vartheta_n]$	–	wektor losowy
Ξ	–	obszar tolerancji
$\delta_{\mathfrak{B}}\,(\mathfrak{f})$	–	dylatacja obiektu $\mathfrak{f}$
$\varepsilon_{\mathfrak{B}}\,(\mathfrak{f})$	–	erozja obiektu $\mathfrak{f}$
$\tau\,(q)$	–	funkcja skalująca
Ψ	–	strumień błędów
Ω	–	przestrzeń rozwiązań
$\mathfrak{B}$	–	elementy strukturalne (sonda)

$\mathfrak{D}$	–	dziedzina
$\mathfrak{f}$, $\mathfrak{g}$	–	obiekt przekształceń morfologicznych
$\mathfrak{M}$	–	macierz próby
$\mathfrak{N}(l)$	–	liczba segmentów o skali l
$\mathbb{G}$	–	przestrzeń przeszukiwania
$\mathbb{R}$	–	zbiór liczb rzeczywistych
$\mathbb{S}$	–	przestrzeń rozwiązań
$\mathbb{X}$	–	dziedzina, przestrzeń problemu
$\oplus$	–	suma modulo 2

Ważniejsze oznaczenia w języku angielskim

AFSM	–	Autonomous Finite State Machine
ATS	–	Autonomous Test System
BILBO	–	Built-In Logic Block Observer
BIST	–	Built In Self-Test
BIST-DFF	–	State Registers
BIST-PAT	–	BIST-PATtern generator
BIST-PST	–	BIST-Parallel-Self-Test
BIST-RTD	–	BIST-Random-Test-Data
BIST-STUMPS	–	Self-Test Using MISR and Parallel SRSG
CA	–	Cellular Automata
CBILBO	–	Concurrent BILBO
CGA	–	Continuous Genetic Algorithm
CHA	–	Continuous Hybrid Algorithm
CSTP	–	Circular Self Test Path
CTSS	–	Continuous Taboo Simplex Search
CUT	–	Circuit Under Test
DC	–	Density Clustering
DFT	–	Design For Testability
ECTS	–	Enhanced Continuous Taboo Search
GMO	–	Genetically Modified Organisms
ICARIS	–	International Conference on Artificial Immune Systems
IEEE	–	Institute of Electrical and Electronics Engineers
IRM	–	Immune Recuirement Mechanism

ISCAS	–	International Symposium on Circuits and Systems
ISCAS'85	–	Combinational Benchmark Circuits
ISCAS'89	–	Sequential Benchmark Circuits
LFSR	–	Linear Feedback Shift Register
MA	–	Mode Analysis
MIShR	–	Multi Input Shift Register
MISR	–	Multi Input Signature Register
MISR MR	–	MISR based Memory Register
MLSL	–	Multi Level Single Linkage
MR	–	Memory Register
PRS	–	Pure Random Search
PSO	–	Particle Swarm Optimization
RCGA	–	Real-Coding Genetic Algorithm
SEA	–	Simplex and Evolution Algorithm
SEREG	–	SErial REGister
SISR	–	Single-Input Signature Register
S-IMM	–	Semi Immune Algorithm
SP	–	Scan Path
S-PSO	–	Semi Particle Swarm Optimization
STP	–	Self Test Path
TGO	–	Topographical Global Optimization
TMSL	–	Topographical Multilevel Single Linkage
TPS	–	Test-Per-Scan

Wstęp

Obserwacje systemów organizmów żywych stają się inspiracją do tworzenia nowoczesnych technik obliczeniowych. Motto rozprawy zaczerpnięte z książki Z. Michalewicza i D.B. Fogela „Jak to rozwiązać czyli nowoczesna heurystyka" kryje w sobie bardzo ważną konkluzję – tworzenie systemu jako metafory czy też zbioru metafor funkcjonowania organizmów żywych znosi związane z tym ograniczenia. Jednocześnie na taki system zostają narzucone znacznie większe ograniczenia związane z ich implementacją. Z twierdzenia NFL [223] (*No Free Lunch Theorem*) wynika, że nie istnieje uniwersalny algorytm optymalizacyjny dla wszystkich klas zadań. Jest to konsekwencja zależności pomiędzy zachowaniem algorytmu a rozwiązywanym problemem. Stanowi to natomiast inspirację do kreowania nowych rozwiązań, prowadzenia badań nad zachowaniem algorytmu i jego przydatnością do rozwiązywania zadań określonej klasy. W większości przypadków prowadzi to do próby zwiększenia efektywności procesu obliczeniowego poprzez modyfikację istniejących algorytmów.

Szczególnie interesującą grupę stanowią algorytmy implementujące koewolucję w środowiskach przyrodniczych, gdyż do nich nie stosuje się twierdzenie NFL. Algorytmy ewolucyjne od algorytmów stochastycznych odróżnia bardzo efektywny, adaptacyjny mechanizm przeszukiwania przestrzeni rozwiązań. Przez to w procesie optymalizacji algorytmy stochastyczne wymagają przeprowadzenia większej liczby iteracji, natomiast są one mniej wrażliwe na zatrzymanie procesu optymalizacyjnego w lokalnym optimum. O przydatności algorytmu decydować więc będą reguły dobrze opracowane dla algorytmów stochastycznych, a określenie ich metafor – czyli *de facto* tworzenie zupełnie nowych algorytmów – wcale nie jest rzeczą trywialną. Nowego algorytmu należy więc poszukiwać w grupie algorytmów implementujących koewolucję w środowiskach przyrodniczych i bazującego na regułach opracowanych dla algorytmów stochastycznych. Na uwagę zasługuje również zmysł wzroku. Wzrok, przebiegając po obrazie, koncentruje się na jego szczegółach i buduje całe wyobrażenie o nim. Przypomina to fazy eksploracyjną i eksploatacyjną algorytmu optymalizacyjnego. Można przy tym odnieść wrażenie, że jego działanie jest optymalne.

Nietypowe spojrzenie na algorytm optymalizacyjny pozwoliło na stworzenie nowego algorytmu, a prace nad jego rozwojem – na umieszczenie jego

metafor w grupie sztucznego życia. Powstałe w ten sposób algorytmy są dalej efektywnymi algorytmami optymalizacji, a proponowane podejście wprowadza w ich działanie nowe właściwości.

Zatem oryginalnym dorobkiem pracy jest: przedstawienie nowego algorytmu obserwacji – „eyetracking" oraz jego metafor w grupie algorytmów immunologicznych i algorytmów optymalizacji rojem cząstek; wykazanie nowych właściwości tych algorytmów: zachowania przypominającego mechanizm obserwacji, mechanizmu koewolucji – determinującego zachowanie algorytmu i niezależnego od wpływu środowiska. Algorytm cechuje niski koszt pracy. Realizacja postawionych założeń narzuciła konieczność opracowania efektywnego mechanizmu mutacji dla algorytmu immunologicznego. W odniesieniu do algorytmu optymalizacji rojem cząstek zdefiniowano funkcje scenariuszy zachowań. Zaproponowano grupę systemów immunologicznych będącą odpowiednikiem systemu wielopopulacyjnego oraz zdefiniowano metody wymiany informacji pomiędzy systemami w grupie. Przedstawiono umocowanie teoretyczne działania algorytmów, a także poparto to badaniami symulacyjnymi. W badaniach zastosowano analizy fraktalną i multifraktalną, wykazując ich przydatność w badaniach nad zachowaniem algorytmów.

W rozwiązywaniu złożonych zadań optymalizacyjnych algorytmy ewolucyjne zajmują wiodącą pozycję. Optymalizacja struktury diagnostycznej układu cyfrowego jest zagadnieniem wielokryterialnym i stanowi swego rodzaju wyzwanie.

W powszechnym użytku znajduje się coraz więcej urządzeń elektronicznych, zwłaszcza takich, które w swojej budowie zawierają elementy systemów cyfrowych. Diagnostyka jest tak ważnym elementem w procesie produkcji i eksploatacji, że opłacalne staje się wprowadzenie znacznego nadmiaru układowego do wnętrza obwodu scalonego, celem ułatwienia procesu testowania. Wprowadzenie struktur diagnostycznych do wnętrza układu jest niewątpliwe efektem ich ewolucji. Przyjęty w 1990 roku standard IEEE 1149.1 pozwala na uproszczenie procesu testowania układów i urządzeń w nie wyposażonych. Ciągle rozwijanym zagadnieniem jest tak zwana diagnostyka wbudowana BIST (*Built in Self-Test*). W ostatnich latach zaproponowano dziesiątki różnych rozwiązań struktur BIST, a w szczególności ATS (*Autonomous Test System*). Struktura ATS stanowi rodzaj rejestru, w którym sprzężenie zwrotne realizowane jest przez testowany (kombinacyjny lub sekwencyjny) układ cyfrowy CUT (*Circuit Under Test*), a także elementy struktury diagnostycznej.

W większości przypadków struktury diagnostyczne bazują na rejestrach z liniowym sprzężeniem zwrotnym. Szczególnie przydatne są rejestry, których sprzężenie zwrotne realizuje wielomian pierwotny. Ta grupa rejestrów i ich właściwości oraz przydatność w diagnostyce są bardzo dobrze zbadane. Duże zainteresowanie budzi stosowanie rejestrów z nieliniowym sprzężeniem zwrotnym, pozwalające na zmniejszenie nadmiaru układowego testera. W większości przypadków struktury diagnostyczne budowane są z typowych bloków

o sztywnej architekturze i dobrze przebadanych właściwościach. Rejestry stosowane w diagnostyce jako generatory testów muszą wytwarzać sekwencję stanów o ściśle określonych cechach. Jest to związane z długością sekwencji testującej i wytworzonymi w niej stanami. Natomiast w przypadku kompakcji odpowiedzi testowej istotne jest uzyskanie najdłuższego cyklu.

Większość wytwarzanych obecnie układów to układy wielomodułowe, których moduły mogą być testowane niezależnie. Zatem niewiele prac odnosi się do budowy i optymalizacji struktur diagnostycznych w układach wielomodułowych. Kompleksowe podejście do diagnostyki układu wielomodułowego może prowadzić do nowych rozwiązań, również w zakresie diagnostyki pojedynczego modułu. Tak więc nowych rozwiązań struktur diagnostycznych należy poszukiwać dla układów wieloomodułowych, zakładając niewielki nadmiar układowy i elastyczność architektury.

Rozprawa prezentuje nowe ujęcie problemu architektury struktury diagnostycznej oraz jej optymalizacji, która bazuje na nietypowym podejściu do tego zagadnienia oraz wskazuje jej podstawę teoretyczną. Oryginalnym dorobkiem pracy w tym zakresie jest: propozycja architektury wbudowanego testowania, bazującej na tak zwanej modyfikacji liniowej, wprowadzenie opisu struktury diagnostycznej, określenie podstaw teoretycznych tej koncepcji, potwierdzenie sformułowanych podstaw teoretycznych, a jednocześnie weryfikacja skuteczności diagnostycznej proponowanych rozwiązań metodami symulacyjnymi, opartymi na modelowaniu z wykorzystaniem układów testowych ISCAS'89 [42], wykazanie stałych cech modułów podczas testowania, przedstawienie formalnego zapisu dowolnej struktury diagnostycznej wraz z opisem ram optymalizacji oraz koncepcji narzędzia symulacyjnego wykorzystywanego w prowadzonych badaniach – przedstawiono również oryginalne wykorzystanie algorytmu genetycznego, uzyskując wysoką efektywność optymalizacji. Ta część publikacji przedstawia kompletny system opisu dowolnej struktury diagnostycznej wraz z metodą jej optymalizacji. Rozwiązania te otwierają drogę do dalszych badań, jednak ich prowadzenie jest poważnie ograniczone możliwościami obliczeniowymi sprzętu komputerowego.

Stosowanym narzędziem optymalizacyjnym, a zarazem badawczym dla proponowanej struktury diagnostycznej stał się algorytm genetyczny. Jego działanie i stosowanie są nie tylko łatwe, ale i efektywne. Mimo to, właśnie te badania zrodziły potrzebę poszukiwania nowych algorytmów optymalizacji, mając na uwadze rodzące się koncepcje badań. Jednakże przeprowadzone eksperymenty wykazały duże podobieństwo uzyskanych wyników. Pod względem merytorycznym tę część opracowania można uznać za zamkniętą, natomiast samo narzędzie może być rozwijane w kierunku aplikacji komercyjnej lub może stanowić inspirację do sformalizowania opisu innych zagadnień optymalizacji w narzędziach badawczych lub komercyjnych. Rezygnacja z próby bezwarunkowego dostosowania opracowywanych algorytmów do optymalizacji struktur

diagnostycznych pozwoliła na uzyskanie znacznie ciekawszych rezultatów badań właściwości tych algorytmów.

Zatem w pracy wyodrębniają się dwie części, które mimo wspólnej bazy w postaci algorytmów ewolucyjnych, prezentują odrębne i zamknięte tematycznie problemy.

Rozdział 1

Algorytmy optymalizacji – wybrane zagadnienia

W rozdziale tym zostaną przybliżone pojęcia związane z zagadnieniami optymalizacji – będą one stanowić bazę dalszych rozważań. Obszerne omówienie tych zagadnień, z których tylko podstawowe pojęcia przedstawiono poniżej, znajdziemy między innymi w pracy [217]. Optymalizacja jest gałęzią matematyki stosowanej i analizy numerycznej. Taksonomia klasyfikuje metody optymalizacji w zależności od ich konstrukcji i zasad działania z punktu widzenia teorii. Algorytmy optymalizacji dzielą się na: deterministyczne i probabilistyczne.

Działanie algorytmów deterministycznych jest całkowicie niezależne od warunków początkowych. Oznacza to, że kilkukrotne uruchomienie takiego algorytmu dla dowolnego stanu i dowolnych danych wejściowych za każdym razem doprowadzi do takiego samego wyniku.

Istnieją problemy, które obejmują większość istotnych, praktycznych zagadnień występujących w przemyśle i dla nich nie są znane efektywne algorytmy deterministyczne. W takiej sytuacji z pomocą mogą przyjść algorytmy probabilistyczne, dla których to w wielu rozwiązywanych problemach potrafimy jedynie znajdować rozwiązania przybliżone – i to nie we wszystkich przypadkach.

Szczególnie istotne znaczenie w rodzinie algorytmów probabilistycznych mają algorytmy bazujące na podejściu Monte Carlo – losowym (zgodnie ze znanym rozkładem) wytworzeniu zbioru elementów (próby), który podlega modyfikacji w kolejnych iteracjach.

Algorytm optymalizacji charakteryzuje:
1) sposób przypisania oceny do elementów,
2) sposoby selekcji do dalszego przetwarzania,
3) sposób działania operacji przeszukiwania,
4) sposób zbierania i wykorzystania informacji o swoim stanie.

Heurystyki stosowane w optymalizacji są funkcjami pomagającymi zdecydować, który z zestawu możliwych rozwiązań należy przetwarzać. Heurystyka stosowana w optymalizacji jest funkcją będącą częścią algorytmu optymalizacji, wykorzystującego informacje gromadzone przez algorytm, aby pomóc zdecydować, który element przestrzeni rozwiązań jest kandydatem na rozwiązanie i powinien być przetworzony oraz jak następny element może być wytworzony. Heurystyki zazwyczaj zależą od klasy rozwiązywanego problemu.

Metaheurystyki stanowią metodę obliczeniową, łącząc w sposób abstrakcyjny funkcje celu (funkcje oceniające przydatność elementu) lub heurystyki. Połączenie to jest często wykonywane przez wykorzystanie statystyki próbek z przestrzeni przeszukiwań lub na podstawie modelu pewnego naturalnego zjawiska lub procesu fizycznego. Przeznaczone są one do rozwiązywania ogólnej klasy problemów i nie gwarantują znalezienia rozwiązania oraz nie można przewidywać czasu ich działania.

Na obserwacji natury bazują algorytmy, których inspiracją stała się inteligencja roju. Naśladują one zachowania zwierząt i ptaków. Do tej klasy algorytmów należą między innymi algorytmy: mrówkowe (ACO *Ant Colony Optimization*), pszczele (BA *Bee Algorithms*) i roju cząstek (PSO *Particle Swarm Optimization*).

Algorytm PSO jest metodą obliczeniową bazującą na pewnym zbiorze rozwiązań, która optymalizuje problem przez iteracyjne doskonalenie kandydatów na rozwiązanie w odniesieniu do danego środka.

Pionierskimi opracowaniami dotyczącymi optymalizacji rojem cząstek były prace [125], [166]. PSO ma wiele podobieństw do technik obliczeń ewolucyjnych, takich jak algorytmy genetyczne – jednak w przeciwieństwie do nich, PSO nie ma operatorów ewolucyjnych (krzyżowania i mutacji). Metaheurystyki, takie jak PSO nie gwarantują znalezienia optymalnego rozwiązania. Algorytm ten może być stosowany do rozwiązania problemów optymalizacji dyskretnej [126].

Algorytmy ewolucyjne naśladują naturalną ewolucję i traktują kandydatów rozwiązania jako osobniki, które konkurują w środowisku wirtualnym. Są one szczególnym przypadkiem algorytmów stochastycznych. Algorytmy ewolucyjne stanowią niejako metaforę organizmów biologicznych, zapożyczając od nich terminologię i mechanizmy działania. Algorytmy te w każdej iteracji zwanej pokoleniem przetwarzają populację reprezentującą próbę losową należącą do dziedziny rozwiązań, czyli środowiska. Populacja jest zbiorem osobników przetwarzanych w każdej iteracji. Stopień przydatności osobników zostaje poddany weryfikacji środowiska za sprawą tzw. funkcji przystosowania (w zadaniu optymalizacyjnym zwanej funkcją celu lub funkcją kosztów).

W algorytmach ewolucyjnych o rozkładzie osobników w środowisku decydują mechanizmy adaptacji zapożyczone z biologii. Należą do nich operatory genetyczne, takie jak: mutacja, krzyżowanie oraz selekcja. Operatory te realizują funkcje odpowiedzialne za eksplorację środowiska oraz eksploatację obszarów ekstremów lokalnych. Osobnik opisany jest genotypem, składającym się z chromosomów i mającym swoją reprezentację w fenotypie, ocenianym przez funkcję przystosowania, która może zawierać elementy skalowania czy też kary. Mechanizmy adaptacji czynią te algorytmy bardziej efektywnymi niż całkowicie przypadkowe przeszukiwanie przestrzeni rozwiązań.

Oprócz metod inspirowanych naturą ewolucji, istnieją również metody, które naśladują procesy fizyczne, takie jak symulowane wyżarzanie (*Simu-*

lated Annealing), hartowanie równolegle (*Parallel Tempering*), jak również techniki niemające bezpośredniego odwzorowania w rzeczywistości – poszukiwanie tabu (*Tabu Search*) i optymalizacji losowej (*Random Optimization*).

Algorytmy immunologiczne stanowią próbę zastosowania mechanizmów naturalnego systemu immunologicznego [69]. Można je podzielić na populacyjne, do których należą algorytmy: selekcji klonalnej i selekcji negatywnej oraz sieciowe, budujące sieć idiotypową w modelu ciągłym lub dyskretnym. W sztucznych systemach immunologicznych najczęściej wykorzystuje się model komórki typu B, zawierającej jedno przeciwciało. W odniesieniu do zadań optymalizacji rolę antygenu pełni funkcja opisująca problem, a przeciwciałem jest jego rozwiązanie. Sztuczny system immunologiczny broniąc się przed wpływem antygenu, wytwarza i udoskonala przeciwciała, które stanowią rozwiązanie.

Algorytm selekcji klonalnej realizuje to zadanie w dwóch etapach – ekspansji klonalnej oraz hipermutacji. Pierwszy etap dokonuje selekcji oraz klonowania najlepszych przeciwciał, natomiast drugi realizuje mutację w celu lepszego dopasowania przeciwciał. Algorytm selekcji negatywnej eliminuje przeciwciała, które rozpoznają własne struktury jako obce. Model sieci idiotypowej różni się od selekcji klonalnej i negatywnej. W tym przypadku limfocyty B są zdolne do rozpoznawania siebie nawzajem. Pamięć immunologiczna tworzy sieciową strukturę (sieć idiotypową), która podlega modyfikacjom w miarę pojawiania się kolejnych patogenów (antygenów rozpoznawanych przez limfocyt).

Celem optymalizacji jest znalezienie najlepszych możliwych elementów $x^\star$ ze zbioru $\mathbb{X}$ na podstawie zestawu kryteriów – funkcji celu:

$$F = \{f_i : \mathbb{X} \to Y_i : 0 < i \leqslant n, Y_i \subseteq \mathbb{R}\}\,. \tag{1.1}$$

Funkcja celu nie musi być wyrażeniem matematycznym, może być złożonym algorytmem. Dziedzina $\mathbb{X}$ jest przestrzenią problemu i reprezentuje dowolny typ elementów. Ze względu na liczbę kryteriów można wyróżnić algorytmy optymalizacji jednokryterialnej dla $n = 1$ i wielokryterialnej dla $n > 1$. W najprostszym przypadku funkcja celu jest więc odwzorowaniem $f : \mathbb{X} \to Y$, gdzie $Y \subseteq \mathbb{R}$.

W przypadku optymalizacji wielokryterialnej rozwiązaniem jest zbiór elementów optymalnych $x^\star \in X^\star \subseteq \mathbb{X}$, nie można tutaj mówić o globalnym optimum. Różnorodne podejścia do ustalenia zbioru rozwiązań $X^\star$ prowadzą do uzyskania różnych rezultatów. Przykładem może być metoda ważonych celów lub podział na podpopulacje, realizujące odmienne cele w algorytmie genetycznym – opisy metod można znaleźć w większości materiałów przeglądowych, na przykład [164] lub [217].

Optymalizacja globalna obejmuje wszystkie techniki, które mogą być wykorzystane do wyznaczenia najlepszego elementu $x^\star$ przestrzeni problemu $\mathbb{X}$ dla kryterium F (dla $n = 1$).

Elementem działania algorytmu optymalizacji lub też celem samym w sobie może być wyznaczenie lokalnego optimum sąsiadujących ze sobą elementów, stanowiących podzbiór $\mathbb{X}$ (optymalizacja wielomodalna). W odniesieniu do algorytmu genetycznego cel ten można osiągnąć przez tworzenie stabilnych podpopulacji (gatunków) związanych z różnymi poddziedzinami funkcji (niszami) – patrz [82].

Lokalne optimum $\hat{x} \in \mathbb{X}$ funkcji celu $f : \mathbb{X} \to Y$ to maksimum lokalne $f(\hat{x}) \geqslant f(x)$ lub minimum lokalne $f(\hat{x}) \leqslant f(x)$ dla wszystkich x będących w sąsiedztwie $\hat{x}$.

Jeśli $\mathbb{X} \subseteq \mathbb{R}$, to

$$\forall \hat{x} \exists \varepsilon > 0 : f(\hat{x}) \square f(x)\ \forall x \in \mathbb{X}, |x - \hat{x}| < \varepsilon, \tag{1.2}$$

gdzie $\square$ oznacza: $\leqslant$ dla minimum, $\geqslant$ dla maksimum lokalnego.

Optimum globalne $x^\star \in \mathbb{X}$ funkcji celu $f : \mathbb{X} \to Y$, to maksimum $f(x^\star) \geqslant f(x)\ \forall x \in \mathbb{X}$ lub minimum $f(x^\star) \leqslant f(x)\ \forall x \in \mathbb{X}$.

Algorytmy optymalizacji globalnej są to algorytmy optymalizacji, które uniemożliwiają zbieżność w lokalnym optimum, zwiększając w ten sposób prawdopodobieństwo znalezienia globalnego optimum.

Funkcja przystosowania $v : \mathbb{X} \to \mathbb{V}$, gdzie $\mathbb{V} \subseteq \mathbb{R}^+$ określa przydatność elementu x, jako kandydata rozwiązania lub w kolejnym etapie przetwarzania. W większości przypadków sprowadza się to do wyznaczenia funkcji celu $v(x) = f(x)\ \forall x \in \mathbb{X}$.

Kandydatem na rozwiązanie jest element x przestrzeni $\mathbb{X}$ analizowanego problemu optymalizacji. W grupie algorytmów ewolucyjnych, kandydaci na rozwiązanie są zwykle nazywani fenotypami. Przestrzeń rozwiązań $\mathbb{S}$ stanowią wszystkie możliwe rozwiązania, a to oznacza, że $X^\star \subseteq \mathbb{S} \subseteq \mathbb{X}$.

Heurystyka odwzorowuje kandydatów reprezentujących rozwiązanie we współczynnik ich użyteczności $h : \mathbb{X} \to \mathbb{R}$. W odniesieniu do algorytmów ewolucyjnych przestrzenią przeszukiwania $\mathbb{G}$ problemu optymalizacji jest zbiór (populacja) wszystkich g elementów (osobników), które mogą być przetwarzane przez operacje przeszukiwania. Element $g \in \mathbb{G}$ nazywany jest genotypem i zbudowany jest z elementarnych obiektów zwanych genami (chromosomami). Element g może mieć reprezentację w postaci wektora liczb rzeczywistych (kodowanie fenotypowe) lub inny sposób kodowania, na przykład binarny. Przy przejściu z genotypu na fenotyp zachodzi proces mapowania $\mathbb{G} \to \mathbb{X}$, który jest relacją odwzorowującą element $\mathbb{G}$ przestrzeni wyszukiwania do elementów przestrzeni problemu $\mathbb{X}$. Proces optymalizacji jest przeważnie inicjowany przez tworzenie losowych genotypów.

Operacje wyszukiwania są wykorzystywane przez algorytmy optymalizacji w celu zbadania przestrzeni $\mathbb{G}$. Przetwarzają one zbiór elementów zwanych populacją $Pop \subseteq \mathbb{G} \times \mathbb{X}$. Mutacja i krzyżowanie są przykładami takich operacji. Operacje, które tworzą z istniejących nowe rozwiązania mają bardzo duży wpływ na szybkość algorytmu, zbieżność i różnorodność populacji.

Problem optymalizacji definiuje więc pięć elementów ($\mathbb{X}$, F, $\mathbb{G}$, Op, Map), czyli określenie przestrzeni problemu $\mathbb{X}$, funkcji celu F, przestrzeni przeszukiwania $\mathbb{G}$, zbioru operacji przeszukiwania Op oraz Map – mapowania genotyp - fenotyp.

Zachowanie algorytmu jest bezpośrednio związane z utrzymaniem właściwej równowagi między eksploracją a eksploatacją przestrzeni przeszukiwań. Eksploracja i eksploatacja są celami sprzecznymi.

W problemach ukierunkowanych na czas optymalizacji możemy wyróżnić optymalizację online (*Online Optimization*), realizowaną w przedziale czasowym między dziesięć milisekund do kilku minut oraz offline (*Offline Optimization*), której czas realizacji może trwać nawet kilka dni. Szybkość obliczeń i ich precyzja są celami sprzecznymi.

Strukturę programu ewolucyjnego można przedstawić w następujący sposób:

1. Losowe tworzenie populacji początkowej i jej ocena.
2. Selekcja osobników.
3. Przetworzenie wyselekcjonowanych osobników.
4. Ocena osobników.
5. Sprawdzenie warunku zakończenia.
6. Iteracja od punktu 2.

Zatrzymanie procesu optymalizacji bazuje na obserwacji zachowania algorytmu uwzględniając następujące kryteria:

1) czas obliczeń,
2) liczbę iteracji algorytmu lub rozwoju osobnika,
3) brak postępu optymalizacji,
4) na podstawie porównania z podobnym problemem testowym,
5) uzyskanie satysfakcjonującego rozwiązania.

Można oczywiście stosować dowolną kombinację tych kryteriów.

W pracy [60] zauważono, że w algorytmie ewolucyjnym efektywność przemieszczania populacji zależy od obszaru, w którym się ona znajduje.

Poniżej zostaną przedstawione wybrane algorytmy stosowane w zadaniach optymalizacji, które zdaniem autora wykazują podobieństwo koncepcji lub właściwości algorytmów proponowanych w pracy.

1.1. Algorytmy stochastyczne

Stochastyczne metody optymalizacji często znajdują zadowalające rozwiązania, gdy zawodzą metody deterministyczne. Algorytmy te operują na pojedynczym kandydacie pretendującym do rozwiązania z pewnej przestrzeni problemu, realizując przejście do następnego, najbliższego kandydata.

Przykładem algorytmu stochastycznego może być algorytm PRS (*Pure Random Search*) [7]:

1. Losowe tworzenie punktów próby początkowej.

2. Obliczenie funkcji przystosowania dla próby.
3. Wybór najlepszego rozwiązania.
4. Sprawdzenie kryterium stopu.
5. Losowe tworzenie kolejnej próby (niezależne od poprzednich).
6. Iteracja od punktu 2.

Tworzenie próby początkowej polega najczęściej na losowaniu z rozkładem równomiernym (lub zbliżonym do niego) pewnej liczby punktów, należących do przestrzeni rozwiązań. Ocena próby losowej opiera się na obliczeniu funkcji celu. Przetworzenie może bazować na usunięciu elementów o słabym przystosowaniu. Inną możliwością jest zastąpienie każdego elementu próby wynikiem działania prostej metody lokalnej, polegającej na wytworzeniu sąsiadów elementu próby – druga faza działania algorytmu. Metoda optymalizacji generuje w określony sposób zbiór punktów przestrzeni rozwiązań [192]. Przetworzona próba może identyfikować obszary ekstremów lokalnych. W dalszych obliczeniach może być ona zastąpiona lub uzupełniona nowymi elementami. Analiza elementów próby jest wykorzystana do rozstrzygania warunku zatrzymania i identyfikacji najlepszych rozwiązań. Warunek zatrzymania sprawdza osiągnięcie kryterium stopu przez algorytm.

Tworzenie kolejnej próby losowej może być wynikiem operacji losowych na elementach próby poprzedniej w przypadku, gdy zdefiniowany jest sposób przetwarzania losowego lub losowania z założonym rozkładem prawdopodobieństwa – na przykład tak, jak w algorytmach *Monte Carlo*.

Algorytm PRS może stanowić bazę tworzenia innych algorytmów optymalizacyjnych – tego typu algorytm omawiany jest np. w pracy [7].

Algorytm stochastyczny może być realizowany w dwóch fazach. Faza globalna polega na losowaniu według określonego rozkładu punktów w przestrzeni rozwiązań. Jest ona odpowiedzialna za asymptotyczną poprawność w sensie probabilistycznym oraz asymptotyczną, probabilistyczną gwarancję odnalezienia wszystkich ekstremów [114]. Problemem staje się wybór odpowiednich generatorów liczb losowych oraz występowanie punktów leżących zbyt blisko siebie, których redukcję zaproponowano w pracy [209]. Wybór punktu startowego dla fazy lokalnej wprowadza główne różnice pomiędzy tymi algorytmami. Zakłada się ponadto, że metody lokalne są ściśle malejące – to znaczy, że wytwarzane tą metodą elementy próby zbiegają do ekstremum lokalnego [192]. Przykładem takich algorytmów, bazujących na algorytmie PRS, są Single-Start oraz Multistart – patrz [194]. W algorytmie Single-Start metoda lokalna inicjowana jest przez najlepszy wygenerowany punkt. W metodzie Multistart metoda lokalna ma charakter ściśle malejący i inicjowana jest przez każdy wygenerowany punkt. Metoda ta wykrywa ekstrema lokalne. Algorytmy PRS, Single-Start i Multistart wykazują asymptotyczną poprawność w sensie probabilistycznym. Multistart umożliwia tworzenie optymalnych i suboptymalnych kryteriów zatrzymania [26], [35], [36], [69]. Wadą tej metody jest wielokrotne wykrywanie tego samego ekstremum lokalnego. Usu-

wają ją metody klastrowania (*clustering methods*), których założeniem jest jednokrotne wykonanie metody lokalnej w obszarze ekstremum lokalnego. Klaster jest zbiorem punktów przybliżającym obszar ekstremum lokalnego. Przykładowy opis algorytmu klastrowania można przedstawić w następujący sposób:

1. Losowe tworzenie punktów próby według rozkładu równomiernego.
2. Przekształcenie zbioru wylosowanych punktów.
3. Wyznaczenie klastrów.
4. W każdym wyznaczonym klastrze wykonanie metody lokalnej i zapamiętanie ekstremum lokalnego.
5. Sprawdzenie kryterium stopu.
6. Powyższe punkty powtarzane są aż do osiągnięcia kryterium stopu.

Przekształcenie zbioru wylosowanych punktów może być realizowane jako redukcja lub koncentracja [98]. Redukcja tworzy zredukowaną próbę przez usunięcie elementów o najgorszym przystosowaniu. Etap redukcji nie wpływa na właściwości metod klastrowania, zapewniając asymptotyczną poprawność w sensie probabilistycznym, ale nie zapewnia on asymptotycznej probabilistycznej gwarancji sukcesu odnalezienia wszystkich ekstremów lokalnych.

W metodzie koncentracji dla każdego punktu realizuje się metodę lokalną, np. największego spadku [208] lub losowych kierunków [221]. W pracy [48] przedstawiono metodę koncentracji z wykorzystaniem algorytmów genetycznych.

Klastrem jest zbiór punktów wyznaczony regułami klastrowania (*clustering rules*) względem zadanego punktu – zarodnika (*seed point*). Zarodnik jest punktem o najlepszej wartości przystosowania – nienależącym do klastra lub otrzymanym w wyniku działania metody lokalnej. Przykładem metod klastrowania są: DC (*Density Clustering*) [98], SL (*Single Linkage*) [192], a także wywodząca się z metod klastrowania metoda MLSL (*Multi Level Single Linkage*) [193].

Metoda DC buduje klaster iteracyjnie od zarodnika, dołączając punkty jeszcze nieprzyłączone do klastra. Metoda MA (*Mode Analysis*) buduje klaster, dołączając obszary przestrzeni rozwiązań (np. hipersześciany) [192] – daje to możliwość szybkiego przybliżenia basenów przyciągania. Metoda MLSL nie ma fazy redukcji oraz może nie mieć fazy klastrowania [193]. W metodzie tej punkt może należeć do więcej niż jednego klastra, co pozwala znacznie lepiej wykrywać ekstrema lokalne. Bardziej efektywne mogą być metody topograficzne (TGO – *Topographical Global Optimization*) [6]. Można je tak modyfikować, aby budowały klastry i gromadziły o nich informacje [194]. W fazie globalnej odrzucane są wylosowane (z rozkładem równomiernym) punkty leżące zbyt blisko zaakceptowanych już punktów. Algorytm wykorzystuje informację zawartą w topografie (*topograph*) – grafie skierowanym, którego krawędzie łączą najbliższych sąsiadów (zwrot krawędzi w kierunku większej wartości funkcji celu). Klastry mogłyby być tworzone przez łącze-

nie sąsiadujących grafów, definiując w ten sposób baseny przyciągania [194]. W pracy [6] zaproponowano metodę TMSL (*Topographical Multilevel Single Linkage*), w której to do minimum topografu (węzła o najmniejszej wartości funkcji celu) zastosowano metodę MLSL (algorytm ten wyznacza minimum globalne).

Cechy charakterystyczne dla tej grupy algorytmów będzie można dostrzec w algorytmach omawianych dalej. Algorytmy stochastyczne mają silną podbudowę teoretyczną, dzięki temu znajdują się nieustannie w kręgu zainteresowań. Prace nad nimi dotyczą szczególnie modyfikacji podnoszących ich efektywność.

Spośród pozycji szerzej opisujących problematykę optymalizacji stochastycznej, można tu podać następujące – [55], [71], [201], [217] oraz [229].

1.2. Algorytmy genetyczne

Podwaliny algorytmów genetycznych daje praca [113]. W odróżnieniu od algorytmów stochastycznych, dzięki mechanizmom adaptacji, rozkład prawdopodobieństwa wystąpienia osobników w środowisku zmienia się z każdym krokiem iteracji. Algorytm genetyczny można przedstawić w następujący sposób:

1. Losowe tworzenie populacji początkowej.
2. Selekcja (reprodukcja*).
3. Wykonanie operacji genetycznych (krzyżowanie i mutacja).
4. Sukcesja*.
5. Sprawdzenie kryterium stopu.
6. Iteracja od punktu 2.

* Kroki te nie są obowiązkowe i mogą występować w bardziej złożonych algorytmach.

Operator selekcji przekształca populację osobników w nową populację – przekształcenie to z reguły ma charakter losowy. Reprodukcja jest rodzajem selekcji – jej celem jest wyłonienie osobników rodzicielskich. Można wyróżnić następujące rodzaje selekcji (za [194]):

- proporcjonalną (ruletkową) – polega na losowaniu ze zwracaniem osobników w populacji (prawdopodobieństwo wylosowania osobnika jest proporcjonalne do wartości funkcji przystosowania),
- turniejową – polega na losowaniu grupy osobników (przeważnie dwóch) bez powtarzania i wyborze najlepszego,
- elitarną – polega na zagwarantowaniu najlepiej przystosowanym osobnikom przejścia do następnej populacji; pozostałe osobniki podlegają innej selekcji,
- rangową – polega na określeniu rangi osobnika, co wpływa na prawdopodobieństwo jego wyboru do następnej populacji.

Operacje genetyczne przetwarzają populację w osobniki potomne – potomstwo (*offspring*), mają one charakter losowy. Mutacja jest podstawową operacją, polegającą na losowej zmianie genotypu osobnika [164]. O zasięgu mutacji decyduje jej rozkład wynikający z zastosowanego generatora liczb pseudolosowych – mały zasięg ma charakter eksploatacyjny [155], natomiast duży ma charakter eksploracyjny [202]. W pracy [79] zauważono, że jedynym mechanizmem odpowiedzialnym za eksplorację jest mutacja. W trakcie pracy algorytmu niezwykle trudne staje się utrzymanie właściwej równowagi pomiędzy eksploracją a eksploatacją. Krzyżowanie ma charakter wieloosobniczy (przeważnie dwa osobniki). Celem krzyżowania jest uzyskanie potomków mających cechy rodziców. Krzyżowanie ma charakter eksploatacji środowiska ograniczonego cechami rodziców [200] – potomek posiada cechy rodziców. W artykule [206] wprowadzono operator *invert-over*, który realizuje zarówno funkcję krzyżowania, jak i mutacji. Operator ten pozwala na uzyskanie silnego nacisku selektywnego – koncentrowania poszukiwań na najlepszych osobnikach.

Podstawy działania algorytmu genetycznego wyjaśnia twierdzenie o schematach i hipoteza bloków budujących [113].

Sukcesja jest rodzajem selekcji (często selekcji elitarnej), mającej na celu promowanie najlepszych rozwiązań w następnym pokoleniu. Reszta populacji podlega zwykłym regułom selekcji.

Zdefiniowanie warunku zatrzymania jest bardzo trudne i z reguły opiera się na obserwacji zachowania algorytmu.

Metody grupowania osobników w niszach [9] stosuje się celem identyfikacji ekstremów lokalnych. Wyłanianie nisz może być realizowane na podstawie:

- odległości (podobieństwa) chromosomów,
- koewolucji,
- deformacji funkcji przystosowania.

W algorytmach genetycznych stosuje się również metody usuwania podobnych osobników – preselekcję i ścisk [82] oraz wprowadzanie losowych osobników – losowych imigrantów [9].

Algorytmy genetyczne (jak również i inne algorytmy z grupy ewolucyjnych) w łatwy sposób mogą być realizowane jako obliczenia równoległe – przez przypisanie pojedynczemu osobnikowi oddzielnego procesu lub podział populacji na subpopulacje [163]. Równorzędne populacje mogą działać w pewnej hierarchii. Mogą zostać wydzielone obliczenia prowadzone jako zgrubne i precyzyjne. Wymiana materiału genetycznego pomiędzy subpopulacjami zachodzi przez migrację osobników (bądź imigrację [9]). W takich systemach może zachodzić więcej niż jeden proces ewolucji – procesy te oddziałują wzajemnie na siebie.

Algorytm genetyczny jako metoda niedeterministyczna nie daje gwarancji znalezienia za każdym razem tego samego rozwiązania. Istnieją możliwości

tworzenia hybrydowych algorytmów genetycznych w połączeniu z metodami lokalnymi dla operatorów mutacji [155] bądź krzyżowania [200].

Do przeglądowych pozycji książkowych można tu zaliczyć [163], [164], [165], [202].

1.3. Algorytm immunologiczny

Algorytm IRM (*Immune Recruirement Mechanism*) był jednym z pierwszych algorytmów wykorzystujących mechanizmy immunologiczne w optymalizacji [24]. W pracy [67] zaproponowano modyfikację algorytmu *aiNet* [65], [66] (algorytmu uczenia sieci idiotypowej), tworząc algorytm selekcji klonalnej o nazwie *CLONALG*. Natomiast w pracy [219] przedstawiono modyfikację algorytmu *CLONALG,* wprowadzając mechanizmy preselekcji i zatłoczenia. Opis działania ww. algorytmu jest następujący:

1. Inicjalizacja – losowe utworzenie zbioru przeciwciał (ich liczba jest stała).
2. Klonowanie i mutacja – każde przeciwciało produkuje ustaloną liczbę klonów, które poddawane są mutacji (sposób mutacji określono w opracowaniu [219]).
3. Selekcja – dla każdego przeciwciała znajdowany jest najbardziej podobny klon (podobieństwo można mierzyć na przykład odległością euklidesową lub Hamminga), który je zastępuje, jeśli jest lepiej dopasowany do antygenu (środowiska).
4. Supresja – dla każdego przeciwciała znajdowane jest przeciwciało do niego najbardziej podobne. Z pary przeciwciał pozostaje to, które jest bardziej podobne do antygenu, drugie z pary przeciwciał zastępowane jest utworzonym losowo przeciwciałem.
5. Kroki algorytmu 2-4 powtarzane są aż do osiągnięcia kryterium stopu.

W tym algorytmie antygen reprezentuje funkcję celu w rozpatrywanej przestrzeni rozwiązań. Wszystkie przeciwciała powinny wytwarzać klony. Przeciwciała, które nie będą wytwarzały klonów, mogą pozostać w bezruchu (tzn. nie będą przemieszczać się w kierunku ekstremów lokalnych). Wytwarzane w punkcie 3 algorytmu przeciwciała poruszają się w kierunku ekstremów lokalnych i skupiają się w nich. Powoduje to zmniejszenie różnorodności populacji przeciwciał w obszarze eksploatacji. W obszarach tych skupione przeciwciała przemieszczają się w niewielkim zakresie, co można uznać za swoistego rodzaju bezruch. Prowadzi to do stagnacji algorytmu – algorytm nie wykonuje eksploracji przestrzeni rozwiązań. Temu zjawisku skutecznie przeciwdziała punkt 4 algorytmu, usuwając nadmiar podobnych przeciwciał. Problemem pozostaje sposób i liczba wytwarzania klonów przez przeciwciała, czyli sposób, w jaki zmutowane klony mają eksploatować przestrzeń otaczającą przeciwciało, które je wytworzyło – te zagadnienia omawiane są w pracy [211].

Odniesienie do optymalizacji środowisk niestacjonarnych zawiera praca [79], w której zaproponowano algorytm *Sais:*

1. Inicjalizacja populacji.
2. Ocena komórek.
3. Selekcja klonów.
4. Rekrutacja komórek do nowej populacji.
5. Iteracja od punktu 2 do osiągnięcia kryterium stopu.

Wysoką efektywność pracy algorytmu uzyskano dzięki selekcji, a nie destrukcji komórek. Algorytmy immunologiczne wykazują naturalną równowagę pomiędzy eksploracją a eksploatacją – dzięki temu są one chętnie stosowane w środowiskach niestacjonarnych [212]. Pomimo bardzo interesujących właściwości, rozwój algorytmów bazujących na selekcji klonalnej nie był tak dynamiczny, jak rozwój opisywanych poniżej algorytmów PSO.

Olbrzymim źródłem wiedzy o możliwościach stosowania algorytmów immunologicznych są materiały tematycznej konferencji ICARIS. Do przeglądowych pozycji książkowych można tu zaliczyć [218].

1.4. Algorytm PSO

Działanie algorytmu polega na przeszukiwaniu przestrzeni przez cząstki, których pozycja i prędkość są modyfikowane według prostych wzorów matematycznych. Każda cząstka w ruchu pod wpływem lokalnej, najlepszej pozycji lidera kieruje się również w jego stronę. Takie zachowanie powoduje przemieszczanie cząstek w kierunku najlepszych rozwiązań. Dużą zaletą jest pamiętanie stanu poprzedniego, co pozwala na eksplorację znanej okolicy, a jednoczenie nie hamuje eksplorowania przestrzeni dalszych.

Podstawowy algorytm PSO można sformułować w następujący sposób:

1. Losowe tworzenie próby początkowej (zbioru punktów reprezentujących cząstki) – dla każdej cząstki określone zostają położenie i prędkość początkowa.
2. Obliczenie funkcji przystosowania dla próby.
3. Wybór najlepszego rozwiązania – określenie najlepszego sąsiada oraz wytypowanie lidera roju.
4. Sprawdzenie kryterium stopu.
5. Tworzenie kolejnej próby według zdefiniowanej reguły – nowy wektor prędkości obliczany jest na podstawie parametrów cząstki, jej najlepszego sąsiada oraz lidera roju, a nowe położenie cząstki określane jest na podstawie jej parametrów (położenia oraz nowego wektora prędkości).
6. Iteracja od punktu 2.

W trakcie kolejnych iteracji następuje adaptacja roju do środowiska. Lider roju reprezentuje pozycję o najlepszym przystosowaniu (najlepsze rozwiązanie). Przyporządkowanie sąsiadów cząstce roju realizowane jest przeważnie

raz – na początku obliczeń, co ułatwia wyznaczenie najlepiej zaadaptowanego sąsiada. Inteligencja roju wykazuje następujące cechy:

1. Bliskości – sposób wykonywania obliczeń, określających położenie cząstki.
2. Jakości – zapamiętuje najlepsze położenia lidera oraz przystosowanie.
3. Zróżnicowania odpowiedzi – zróżnicowanie położenia cząstek w roju.
4. Stabilności – rój cząstek zmienia swoje zachowanie pod wpływem zmian położenia lidera, nie pod wpływem zmian środowiska.
5. Adaptacyjności – zmiana zachowania roju cząstek jest funkcją zmian zachowania lidera.

Istnieje wiele modyfikacji omawianego algorytmu, znaczną ich liczbę wymienia pozycja [196]. Interesującą modyfikacją są dynamicznie zmieniające się topologie sąsiedztwa [167]. W pracy [191] zaproponowano zwiększające się zróżnicowanie cząstek. Inny sposób modyfikacji to zmiana zasad aktualizacji prędkości [188]. Prostota algorytmu pozwala na stosowanie elementów z innych podejść [8].

W publikacji [118] używana jest wersja lokalnego sąsiedztwa, w której pod uwagę brane jest tylko zachowanie sąsiadów. W pracy [145] zaproponowano wprowadzenie dodatkowego elementu wiedzy o środowisku (*repellor*), który wpływa na zachowanie cząstek, kierując rój w obszary środowiska o lepszej adaptacji. W opracowaniu [104] zaproponowano koewolucję dwóch rojów, z których jeden optymalizuje współczynniki kary, a drugi poszukuje optymalnego rozwiązania (roje poruszają się w przestrzeniach optymalnych rozwiązań współczynników kary dla roju przeciwnego). Innym przykładem systemu koewolucyjnego jest model drapieżników i ich zdobyczy (*predator-prey*), wykorzystujący model teorii gier [10]. Podejście to było stosowane w optymalizacji grupowania rozmytego [121]. Do prezentowanego poniżej algorytmu wydaje się być podobny algorytm pokazany w pracy [107]. Ma on jednak inną mechanikę działania w stosunku do algorytmu proponowanego w pracy. Jest to algorytm drapieżnik-zdobycz, który bierze przykład z zachowania ławicy sardynek i polujących orek. Drapieżniki podążają do środka ławicy sardynek – wygląda to jak rozpędzanie zdobyczy, która ucieka przed drapieżnikami. Przyczynia się to do tego, że cząstki unikają lokalnych optimów rozwiązania, aby w efekcie znaleźć optymalne rozwiązanie globalne. Drapieżniki odgrywają rolę eksploatacyjną (realizują konwergencję), natomiast zdobycze uciekając przed drapieżnikami realizują eksplorację przestrzeni rozwiązań (pełnią rolę dywersyfikacji algorytmu). Algorytm realizuje scenariusz, w którym wybierany jest najbliższy drapieżnik i określane jest czy cząstka będąca zdobyczą ucieka (co zależy od odległości zdobyczy i drapieżnika – na tej podstawie wyliczana jest prędkość ucieczki). Opis ten wskazuje na istotne różnice pomiędzy algorytmem prezentowanym w pracach [107] i [108], a proponowanym w poniższej monografii. Podejście wykorzystujące mechanizmy koewolucji czy kooperacji gatunków, jak również grupowania osobników jest stosowane w wielu algorytmach. W pracy [127] zaproponowano wykorzystanie algorytmu

wyznaczającego klastry (*k-means clustering algorithm*), grupującego cząstki w określonej (predefiniowanej) liczbie klastrów. W celu uzyskania stabilności klastrów, algorytm klastrowania wykonywany jest trzykrotnie (uwaga: wyznaczanie klastrów może być realizowane przez sam algorytm PSO, jak również potwierdzenie stabilności klastra – co realizują proponowane w pracy algorytmy).

W strukturze algorytmu SOS [38] występuje populacja macierzysta, przeszukująca przestrzeń rozwiązań i populacje potomne, które śledzą lokalne optima. Populacja macierzysta jest stale analizowana, celem sprawdzania warunków tworzenia potomków – całkowita liczba osobników jest stała.

W pracy [44] zaproponowano algorytm *nbest*PSO, który został zaprojektowany do lokalizowania wielu rozwiązań. Algorytm *nbest*PSO definiuje sąsiedztwo jako najmniejszą odległość pomiędzy cząstkami. Najlepsze sąsiedztwo określa się na podstawie średniej wartości najbliższych cząsteczek.

W [45] zaproponowano algorytm *Niche*PSO – główny rój może stworzyć subrój w przypadku, gdy zostaje zidentyfikowana nisza. Kryterium tworzenia subroju jest brak istotnych zmian w kolejnych iteracjach (stagnacja – przypisek autora), natomiast subrój może wchłaniać cząstki lub inne subroje, zależnie od promienia (odległości).

Algorytm PSO bazujący na specjacji (SPSO) przedstawiono w pracach [150], [178]. Algorytm ten dynamicznie dostosowuje liczbę i rozmiar rojów przez utworzenie uporządkowanej listy cząstek, uszeregowanych według ich sprawności, łącząc cząstki konkretnego gatunku w bliskiej przestrzeni. W każdym pokoleniu SPSO ma na celu identyfikację wielu liderów (zarodników) gatunków w roju. Wszystkie cząstki w otoczeniu lidera są przypisane do tego samego gatunku. W ulepszonej wersji SPSO wprowadzono mechanizm do usuwania duplikatów cząstek [179].

W [27] zaproponowano algorytm adaptacyjny *niching*PSO (ANPSO), w którym promień gatunków wyznaczany jest na podstawie statystyki populacji. Wprowadzono również inną ulepszoną wersję SPSO, stosując regresję najmniejszych kwadratów (rSPSO) [28].

Podejście atomowe rojów zastosowano do jednoczesnego śledzenia wielu ekstremów w dynamicznym środowisku [29], [30]. W tym podejściu algorytm zapewnia utrzymanie różnorodności roju, a zasada wykluczania zapewnia, że nie więcej niż jeden rój otacza pojedynczy pik. W [30] wprowadzona została antykonwergencja, która przez wymianę informacji między wszystkimi subrojami pozwala na wykrycie nowych ekstremów. Strategia ta okazała się skuteczna dla problemu środowiska niestacjonarnego [37].

Aby określić liczbę klastrów w algorytmie *k-means* PSO (w pracy [181]), zaproponowano algorytm, w którym cząstki są generowane przez połączenie kilku probabilistycznych rozkładów – każdy klaster odpowiada innej dystrybucji. Następnie, znalezienie optymalnej liczby k jest równoważne znalezieniu najlepiej dopasowanych modeli. Algorytm ten daje lepsze wyniki dla pro-

blemów opisanych w środowiskach stacjonarnych niż algorytmy SPSO [150] i ANPSO [27].

W CESO [157] współpracują ze sobą dwa roje, z których jeden wykorzystuje ewolucję różnicową (CDE) [207], a drugi – model PSO. Rój, który korzysta z CDE jest odpowiedzialny za różnorodność, podczas gdy rój PSO śledzi globalne optimum (wyniki konkurencyjne odnotowano w [157]).

Do lokalizowania i śledzenia wielu ekstremów w dynamicznym środowisku zaproponowano w [151] szybki wielorojowy algorytm (FMSO), którego inspiracją był algorytm SOS [38]. W FMSO rój rodzic jest używany do identyfikacji zmian środowiska, natomiast roje potomne są wykorzystywane do wyszukiwania lokalnego optimum w swoich podprzestrzeniach. Każdy rój dziecko ma określony promień wyszukiwania. Jeżeli odległość między dwoma rojami jest mniejsza niż ich promień, to następuje usunięcie roju gorszego. Gwarantuje to, że nie więcej niż jeden rój dziecko obejmuje pojedyncze ekstremum.

Powyższa dyskusja określa kluczowe problemy systemów wielorojowych, tj.: jak określić obiecujący obszar w przestrzeni rozwiązań i jak realizować ruch cząstek w kierunku różnych podobszarów oraz jak określić potrzebną liczbę subrojów i jak generować subroje.

Próbą rozwiązania tych problemów jest *Clustering* PSO (CPSO) [147]. W CPSO każda cząstka uczy się – zna własną (wynikającą z przebiegu działania algorytmu) najlepszą pozycję oraz pozycję najbliższego sąsiada, co ma wpływ na ruch cząstek. Korzystając z hierarchicznego sposobu grupowania, cały rój w CPSO można podzielić na subroje, które obejmują różne regiony przestrzeni rozwiązań. CPSO realizuje adaptacyjne wykrywanie podregionów. Strategia dla globalnej, najlepszej cząstki została również wprowadzona w CPSO [147].

W [224] zastosowano pewne uproszczenia w porównaniu z oryginalnym CPSO: w [224] usunięto proces szkolenia, a w [147] wprowadzono metodę hierarchicznego grupowania, która obejmuje dwa etapy – zgrubne grupowanie i precyzowanie klastrów. W [224] metoda grupowania hierarchicznego jest uproszczona tylko do jednej fazy. Wyniki eksperymentalne, przedstawione w [224] wskazują lepszą wydajność klastrów cząstek roju w lokalizacji i śledzeniu optimum w dynamicznym środowisku w porównaniu z innymi modelami PSO, opartymi na wielu rojach.

Obecnie tylko algorytmy hybrydowe potrafią realizować wysoko wydajne przeszukiwanie przestrzeni rozwiązań, uzyskując jednocześnie dużą precyzję podczas eksploracji ekstremów.

1.5. Programowanie genetyczne

Interesujące podejście do rozwiązywania problemów zaproponowano w [139] i nazwano je programowaniem genetycznym. W trakcie procesu ewolucyjnego program powinien sam się rozwijać, tworząc populację wykony-

walnych programów – pojedyncze programy rywalizują ze sobą (słabe wymierają, mocniejsze reprodukują się). Przestrzeń rozwiązań jest przestrzenią programów kodowanych w strukturze drzewiastej. Drzewo składa się z węzłów nieterminalnych (operatorów lub funkcji, których parametrami są węzły podrzędne) oraz węzłów terminalnych (reprezentujących wartości: stałe, zmienne, funkcje bezparametrowe). Dla takiej reprezentacji osobników zdefiniowano odpowiednie operatory genetyczne. Inną reprezentacją osobnika mogą być łańcuchy funkcyjne lub wielowartościowe łańcuchy funkcyjne [95]. Zastosowanie tych ostatnich rozszerza możliwości operacji genetycznych tak, jak dla algorytmów genetycznych. Krótkie omówienie możliwości ich wykorzystania zawiera dodatek A niniejszej rozprawy.

1.6. Uwagi dotyczące działania algorytmów

Wiele problemów jest traktowanych jako niezmienne – są one reprezentowane przez stałe środowisko. Jednak przez zmianę zasobów, zadań czy też innych elementów systemu problem zmienia się ze statycznego na dynamiczny. Tego typu problemy reprezentowane są przez zmienne środowisko. W rezultacie, wcześniejsze podejścia do rozwiązania problemu nie mogą być stosowane w nowym środowisku. Utrata różnorodności występuje z powodu samej natury metaheurystyki, realizującej konwergencję populacji w ekstremum globalnym – co jest głównym celem optymalizacji w środowisku statycznym. Stanowi to istotną wadę w przypadku optymalizacji w środowisku dynamicznym. Znalezienie optimum globalnego nie jest jedynym celem w warunkach dynamicznych – bardzo ważne jest śledzenie zmian w przestrzeni problemu. Również w środowisku stacjonarnym mechanizm konwergencji może powodować przedwczesną zbieżność w ekstremum lokalnym. Eksploracja przestrzeni rozwiązań i eksploatacja ekstremum uważane są za cele sprzeczne – przeciwstawne, tzn. zwiększenie eksploracji powoduje zmniejszenie eksploatacji, a znalezienie kompromisu jest wyjątkowo trudne. Należy tu podkreślić, że w przypadku środowiska stacjonarnego tylko przeszukiwanie przestrzeni rozwiązań może doprowadzić do odszukania ekstremum globalnego, natomiast w przypadku środowiska zmiennego umożliwi odszukanie obszarów zmian. Eksploatacja pozwoli określić wartość ekstremum w stałym środowisku, natomiast w zmiennym środowisku dodatkowo pozwoli podążać za jego zmianami. Ze względu na odmienne funkcje eksploracji i eksploatacji ich optymalne działanie powinno być celem efektywnie pracującego algorytmu, a nie tylko rodzajem kompromisu. Należy poszukiwać rozwiązań czyniących je niezależnymi od siebie tak, jak w przypadku faz globalnej i lokalnej algorytmu stochastycznego. Rozwiązania można szukać w systemach koewolucyjnych. Działanie takich systemów jest zależne od siebie, natomiast ich funkcje mogą być różne.

Jak już wspomniano, eksploracja i eksploatacja są celami sprzecznymi. Zachowanie algorytmu w dużej mierze zależy od funkcji celu. W dwutomowej pracy [99] i [100] przedstawiono zbiór stosowanych operatorów krzyżowania i mutacji, dostosowanych do rozwiązywania różnej klasy zadań. Oznacza to brak w miarę uniwersalnego podejścia do tego zagadnienia. Problemem pozostaje jeszcze zależność zachowania się algorytmu od środowiska albo inaczej – poszukiwanie algorytmu zdolnego do efektywnego rozwiązywania zadań różnych klas. Przedstawione dalej algorytmy podejmują próbę rozwiązania głównych problemów zaprezentowanych powyżej.

Również szybkość obliczeń i ich precyzja są celami sprzecznymi i jest to raczej związane z charakterem rozwiązywanego problemu – optymalizacją online lub offline. W tym przypadku wydziela się obliczenia, które są prowadzone jako zgrubne, co podnosi szybkość obliczeń kosztem precyzji.

Algorytmy przedstawione powyżej mają wiele cech wspólnych. Definicja problemu w działaniu algorytmu oraz sposób jego eliminacji są więc bardzo podobne i wywołują podobny skutek. Zarówno w algorytmie genetycznym, jak i immunologicznym mechanizm eksploatacji ekstremów lokalnych i mechanizm eksploracji przestrzeni rozwiązań są od siebie zależne. Większość algorytmów stosowanych w zmiennych środowiskach jest adaptacją algorytmów stosowanych w środowiskach stacjonarnych. Prezentowane w pracy algorytmy były projektowane z myślą o ich zastosowaniu w środowiskach niestacjonarnych. Nowoczesne algorytmy PSO o dużej efektywności należy zakwalifikować jako algorytmy hybrydowe. Prezentowane algorytmy przedstawiono raczej jako bazowe – do przyszłych modyfikacji. Opierając się na przedstawionym opisie nietrudno zauważyć wręcz konieczność wykonania określonych modyfikacji, które stworzą algorytm hybrydowy o dużej wydajności. Proponowane w pracy algorytmy porównano z różnymi algorytmami, również starszymi, stąd też biorą się odwołania do starszych pozycji literatury.

Rozdział 2

Algorytm „eyetracking”

Opisywany poniżej autorski algorytm powstał w wyniku analizy pracy algorytmów stochastycznych, genetycznych, PSO, a w szczególności immunologicznych, jednak określenie jego jednoznacznej przynależności jest kłopotliwe. Nazwa algorytmu powstała na podstawie cechy wyróżniającej go spośród pozostałych algorytmów – „eyetracking”.

2.1. Opis działania algorytmu „eyetracking”

Algorytm realizuje kooperację (w systemach ewolucyjnych określanej terminem koewolucji) dwóch systemów, nazywanych dalej elementami próby i zarodnikami (analogicznie do systemów stochastycznych). Słowny opis algorytmu przedstawiono poniżej:

1. Losowe tworzenie elementów próby początkowej oraz zarodników.
2. Aktywacja zarodnika – losowy wybór zarodnika.
3. Redukcja zarodników – sprawdzenie czy w zadanym otoczeniu aktywnego zarodnika znajdują się inne zarodniki; zarodniki gorzej przystosowane zastępowane są w sposób losowy; zarodnik, który pozostał staje się zarodnikiem aktywnym.
4. Aktywacja elementów próby (wyznaczenie klastra) – elementy próby w otoczeniu zarodnika stają się aktywne.
5. Redukcja aktywnych elementów próby – spośród aktywnych elementów znajdujących się w odległości mniejszej niż zadana gorzej przystosowane przestają być aktywne i zostają zastąpione elementami losowymi.
6. Przetworzenie – zastąpienie aktywnych elementów próby i aktywnego zarodnika nowymi, będącymi wynikami określonej reguły przetwarzania – wykonanie metody lokalnej (reguła przetwarzania została podana poniżej w sposób opisowy).
7. Iteracja od punktu 2 do osiągnięcia kryterium stopu.

Zarodniki oraz elementy próby początkowej tworzone są w sposób losowy. Elementów próby jest znacznie więcej niż zarodników. Elementy próby tworzą rodzaj siatki, której węzły pod wpływem metod przetwarzania przemieszczają się w stronę ekstremów lokalnych. Gęstość elementów próby w obszarach ekstremów lokalnych będzie większa niż na pozostałych obszarach. Aktywny zarodnik, korzystając z informacji o aktywnych elementach, określa swoją nową

pozycję kierując się również w stronę ekstremów. Prędkość przemieszczania (zarówno elementów próby, jak i zarodników) zależy od gęstości elementów próby i jest tym większa, im mniejsza jest ich gęstość. Dzięki temu zarodniki wykorzystując informacje z aktywnych elementów próby poruszają się znacznie szybciej poza obszarami ekstremów lokalnych, natomiast w obszarach ekstremów poruszają się wolniej. To właśnie ruch zarodników przypomina ruch oczu oglądających obraz. Generalnie elementy próby przemieszczają się wolniej od zarodników. Elementy próby są odpowiedzialne za eksplorację przestrzeni rozwiązań, a zarodniki za eksploatację ekstremów lokalnych – jednak między tymi elementami zachodzi ścisła współzależność. Redukcja zarodników świadczy o identyfikacji ekstremów lokalnych, a jednocześnie pomaga w eksploracji środowiska, ponieważ zredukowane zarodniki zastępowane są losowo tworzonymi zarodnikami. Natomiast redukcja elementów próby jest wskaźnikiem eksploatacji ekstremów lokalnych i zwiększa eksplorację przestrzeni rozwiązań przez jej losowe próbkowanie. Stanowi to mechanizm samoregulacji eksploracji i eksploatacji realizowanej przez algorytm.

Elementy próby wykonują swoistego rodzaju aproksymację funkcji środowiska. Zarodniki inicjują proces optymalizacji ich położenia. Zbliżanie się elementów próby powoduje ich zagęszczanie. Działanie to ma charakter adaptacyjny i decyduje o rozkładzie próbkowania przestrzeni rozwiązań. Można więc wnioskować, że dzięki mechanizmowi adaptacji rozkład próbkowania będzie optymalny. Kooperacja zarodników i elementów próby w znaczny sposób osłabia wpływ środowiska na działanie algorytmu. Przedstawiony powyżej algorytm ma charakter bazowy, na którym zostaną oparte dwa kolejne algorytmy.

Algorytm ten ma oczywistą wadę, istotną dla środowisk stacjonarnych – jest nią wielokrotne wykrywanie ekstremów lokalnych. Wada ta może być w prosty sposób zniwelowana, a rozwiązania takie można z łatwością odnaleźć w literaturze.

O zachowaniu elementów próby i zarodników decyduje metoda przetwarzania. Dla tego algorytmu zrealizowana została ona jako metafora systemu immunologicznego oraz systemu PSO. Ze względu na to, że metafory te nie są implementowane jako kanon tych algorytmów, dalej będą nazywane algorytmem semiimmunologicznym (S-IMM) oraz semi-PSO (S-PSO).

Odstępstwem od standardowego podejścia do algorytmu immunologicznego jest reprezentowanie antygenu przez zarodniki, a nie przez środowisko. Ponadto, antygeny zmieniają położenie pod wpływem przeciwciał, które reprezentowane są przez elementy próby. Z podobną sytuacją mamy do czynienia w przyrodzie, gdy wirusy mutują, by obronić się przed działaniem systemu immunologicznego. Natomiast u podstaw redukcji przeciwciał stoi mechanizm autoimmunizacji. Powoduje on autoagresję wobec własnych komórek. Z kolei usuwanie najsłabszych ze zgrupowanych i otoczonych przeciwciałami antygenów stanowi interpretację mechanizmu ich redukcji.

W odniesieniu do systemów wielopopulacyjnych zaproponowano grupę systemów immunologicznych oraz zdefiniowano wszystkie pojęcia związane z jej funkcjonowaniem. Wymiana materiału genetycznego odbywa się przez szczepionkę oraz surowicę.

Algorytm S-PSO implementuje strategię osaczania (*beset game*) jako koewolucję dwóch systemów: drapieżników reprezentowanych przez elementy próby oraz zdobyczy, którą są zarodniki. Zdobycz jest osaczana przez grupę drapieżników. Drapieżniki poruszają się w kierunku zdobyczy oraz w kierunku lidera grupy, który stara się odciąć drogę ucieczki zdobyczy. Zdobycz, na podstawie obserwacji lidera drapieżników i najsłabszego z nich, ucieka w stronę bezpiecznego schronienia, czyli ekstremum lokalnego. Osaczenie więcej niż jednej zdobyczy powoduje eliminację najsłabszych z nich. Natomiast nadmierne zbliżenie drapieżników powoduje redukcję tych najsłabszych. Mechanizmy te wydają się być naturalnymi elementami walki o przetrwanie. Zasada działania algorytmów *predator-prey*, dotychczas opisywanych w literaturze, była odmienna. Złamaniem kanonu tych algorytmów jest również adaptacja współczynników wagowych, kreująca scenariusze zachowań.

Szczegółowy opis algorytmów zostanie przedstawiony w kolejnych rozdziałach. Właściwości algorytmu pozwalają na jego efektywną pracę zarówno w stacjonarnym, jak i niestacjonarnym środowisku. Algorytm ten wykazuje dużą efektywność eksploracyjną oraz eksploatacyjną przestrzeni rozwiązań. Jak dla wszystkich algorytmów tej grupy efektywność ich pracy zależy od strojenia. Proponowany algorytm ma jeszcze jedną szczególną właściwość – jego zachowanie w dużym stopniu zależy od parametrów pracy, natomiast w niewielkim od środowiska.

2.2. Baza teoretyczna działania algorytmu

Poniżej przedstawiono teoretyczne uzasadnienie rozwiązań implementowanych w algorytmie.

Niech będzie dana funkcja celu $f : \Omega \to Y$, gdzie $\Omega \subset \mathbb{R}^n$ jest przestrzenią rozwiązań i $Y \subset \mathbb{R}$. Element próby $e_i \in \Omega$, zarodniki $z_j \in \Omega$ dla $i = 1, 2, ..., n$ i $j = 1, 2, ..., m$, gdzie: n – liczba punktów próby, m – liczba zarodników.

Definicja 2.1 (obszar tolerancji)**.** *Obszar tolerancji Ξ jest obszarem przestrzeni rozwiązań ($\Xi \subset \Omega$), którego środkiem jest element zbioru Ω, a otoczenie określone jest przez promień r, w obrębie którego rozwiązania można uznać za równoważne.*

Obszar tolerancji Ξ_e elementów próby e_i o promieniu r_e i obszar tolerancji Ξ_z zarodników z_i o promieniu r_z mogą być różne $\Xi_e \neq \Xi_z$. Zatem dla elementów próby $e_k \neq e_l$, gdzie $1 \leqslant k \leqslant n$ i $1 \leqslant l \leqslant n$, funkcja celu $f(e_k) \equiv f(e_l)$ wtedy i tylko wtedy, gdy $\|e_k - e_l\| < r_e$. Natomiast dla zarodników $z_k \neq z_l$,

gdzie $1 \leqslant k \leqslant m$ i $1 \leqslant l \leqslant m$, funkcja celu $f(z_k) \equiv f(z_l)$ wtedy i tylko wtedy, gdy $\|z_k - z_l\| < r_z$.

Entropia ([11], [197]) stanowi miarę ilości informacji o przestrzeni rozwiązań uzyskanej w systemie.

Definicja 2.2 (entropia informacji w systemie)**.** *Entropia H zmiennej losowej $\mathfrak{U}$ z funkcją prawdopodobieństwa $P(\mathfrak{U} = \{\mathfrak{u}_i\})$ – która jest miarą wystąpienia elementów próby i zarodników w i-tym obszarze przestrzeni rozwiązań – jest zdefiniowana wzorem:*

$$H(\mathfrak{U}) = -\sum_i P(\mathfrak{U} = \mathfrak{u}_i) \cdot \log_2 P(\mathfrak{U} = \mathfrak{u}_i) \tag{2.1}$$

i jest miarą informacji uzyskanej przez system.

Twierdzenie 2.1 (entropia sygnału przekształconego [197])**.** *Żadne przekształcenie nie może zwiększyć entropii dowolnego sygnału lub inaczej*

$$\underset{f}{\forall} \quad H(f(\mathfrak{U})) \leqslant H(\mathfrak{U}). \tag{2.2}$$

Twierdzenie 2.2 (brak wzrostu entropii systemu)**.** *Penetracja przestrzeni rozwiązań przez dwa lub więcej elementów znajdujących się w obszarze tolerancji nie powoduje wzrostu entropii uzyskanej w systemie.*

Dowód. Na podstawie definicji 2.1 elementy próby losowej znajdujące się we wspólnym obszarze tolerancji reprezentują taką samą informację. Dowód wynika z twierdzenia 2.1, które mówi, że żadne przekształcenie f nie może zwiększyć informacji statystycznej [197]. □

Twierdzenie 2.3 (maksymalnej entropii systemu)**.** *Entropia uzyskana w systemie osiąga wartość maksymalną wtedy i tylko wtedy, gdy penetracja przestrzeni rozwiązań jest równomierna.*

Dowód. Wynika to z właściwości, że entropia osiąga jedno i tylko jedno maksimum przy równomiernym rozkładzie próby losowej (prawdopodobieństwo wystąpienia dla każdego elementu próby jest jednakowe) [197]. □

Z twierdzenia 2.3 wynika jednoznacznie, że największa ilość informacji o przestrzeni rozwiązań uzyskiwana będzie przy równomiernym rozkładzie próby. Jak wynika z twierdzenia 2.2, gęstość próby determinowana będzie przez obszar tolerancji, gdyż nadmierny jej wzrost nie przyczyni się do istotnego wzrostu informacji o przestrzeni rozwiązań. Na tej podstawie powinny być budowane reguły eksploracji przestrzeni rozwiązań.

Dla bardzo małego obszaru tolerancji, którego konsekwencją jest brak zbiegania się elementów próby, a co za tym idzie brak ich redukcji, można przyjąć, że jest to rodzaj stagnacji. Zwiększanie obszaru tolerancji spowoduje pojawienie się redukcji elementów próby. Dalsze zwiększanie obszaru

tolerancji sprawi, że zwiększy się liczba redukcji. Wpłynie to na zwiększenie szybkości przeszukiwania przestrzeni rozwiązań kosztem dokładności. Gdy obszar tolerancji będzie zbyt duży, działanie algorytmu będzie przypominało działanie generatora liczb losowych. Będą następowały bardzo częste redukcje i mechanizmy adaptacji zanikną.

2.3. Środowisko testowe

Do testów algorytmów wykorzystano dwa typy środowisk: stacjonarne i niestacjonarne (zmienne). Obszerne omówienie zmiennego środowiska zawierają między innymi opracowania [39], [40], opisujące środowisko testowe MPB (*Moving Peaks Benchmark* – Dodatek C) oraz GDBG [146] (*Generalized Dynamic Benchmark Generator* – Dodatek D). Środowisko GDBG jest znacznie bardziej złożone od środowiska MPB. Badania efektywności pracy proponowanych algorytmów, opierając się na tych środowiskach, przedstawiono w rozdziałach 2.7 i 2.8.

Bezpośrednie zastosowanie zaprezentowanego tam modelu do analizy zachowania omawianego algorytmu wydaje się problematyczne ze względu na to, że:

- sposób prowadzenia badań ukierunkowany jest, w większości przypadków, na uzyskanie takiego strojenia algorytmu, by uzyskiwane wyniki były najlepsze. Stanowi to dość silne uproszczenie. Nie daje możliwości porównania złożonych zagadnień, w których przestrzeń życiowa osobników nie zajmuje całego dopuszczalnego obszaru możliwych rozwiązań czy też może być współdzielona z innymi osobnikami (populacjami).
- Stosowany sposób obliczania błędów wyznaczania ekstremów lokalnych odniesiony do wartości średniej populacji będzie świadczył co najwyżej o grupowaniu osobników. Zakładając, że średni rozkład osobników w populacji będzie równomierny, to tak obliczony błąd osiągnie duże wartości. Ze względu na sposób działania proponowanego algorytmu, porównanie uzyskanych wyników z innymi opracowaniami będzie bardzo trudne.

Uzasadnieniem dla przyjętego w pracy rozwiązania dla środowiska testowego może być fakt, że wielu autorów opracowań w tym zakresie proponuje własne rozwiązania, które są *de facto* modyfikacjami stosowanych już środowisk. Możliwa jest także adaptacja już istniejącego środowiska, jednak wynikiem tych modyfikacji będzie brak możliwości porównania wyników. Dla zmiennego środowiska zaproponowano więc własne rozwiązanie, wzorowane na pracy [219].

W dodatku C zaprezentowano scenariusze, jakie musi realizować algorytm w czasie procesu optymalizacji. Przedstawione środowisko również wymaga od algorytmu takich działań. Wynika to z wyodrębniania w środowisku mniejszych obszarów (w dalszym opisie prezentowanych na rysunku 2.2), co skutkuje pojawieniem się nowego ekstremum. Przemieszczanie zmienia jego

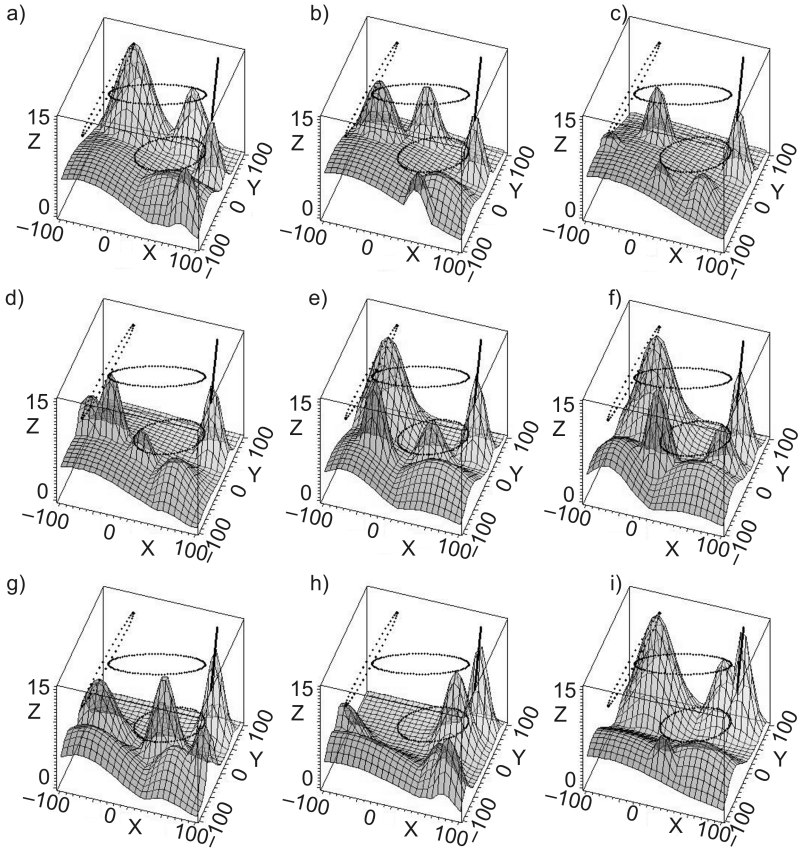
położenie oraz jest możliwe powstanie ekstremum w wyniku zmian wartości wierzchołka, który do tej pory nim nie było. Ekstremum o charakterze globalnym może stać się ekstremum lokalnym. Przemieszczanie ekstremów lokalnych oraz zmiana wartości funkcji przystosowania w zastosowanym środowisku jest większa niż w przykładowym środowisku, omówionym w dodatku C.

Model zmiennego środowiska oparto na zmianach parametrów funkcji unimodalnej, opisanej wzorami:

$$f = \max_{i=1..n} f_i \quad \text{i} \quad f_i(x,y) = h_i \cdot 100^{-\left(\frac{(x-t_{x,i})^2}{w_{x,i}^2} + \frac{(y-t_{y,i})^2}{w_{y,i}^2}\right)}, \tag{2.3}$$

gdzie: h_i – wysokość; $t_{x,i}$, $t_{y,i}$ – przesunięcie x i y; $w_{x,i}$, $w_{y,i}$ – nachylenie zbocza; n – liczba wierzchołków.

Widok przykładowych zmian środowiska przedstawiono na rys. 2.1.



Rysunek 2.1. Widok zmian środowiska z zaznaczonymi zmianami położenia wierzchołków

Zmiany środowiska mogą zachodzić w sposób ciągły lub skokowo co pewną liczbę iteracji. W analizowanym modelu zmiennego środowiska założono zmiany skokowe, pomiędzy którymi występuje faza stabilna. Pełny cykl zmian realizowany jest w 100 krokach.

W modelu wykorzystano sześć funkcji, których parametry podano w tabeli 2.1, gdzie: *Par* – parametry; c – cykle w 100 krokach – 1 cykl=2π rad. (zmieniane parametry w cyklu podane są w nawiasach).

Tabela 2.1. Parametry funkcji

Par	Funkcja					
	f_1	f_2	f_3	f_4	f_5	f_6
h_i	[7,6;15,3]	[10;15]	[7;10]	[10,0;14,3]	[5,5;7,5]	[5,5;6,5]
$t_{x,i}$	[-92;-48]	[70,1;80,0]	[-9,5;89,0]	[-59;59]	-70	50
$t_{y,i}$	[-32;72]	[25;84]	[-89,0;9,7]	[-31,0;81,5]	-70	-50
$w_{x,i}$	75	35	50	50	[100;300]	[15;50]
$w_{y,i}$	75	35	50	50	[100;300]	100
c	$2,0(h_1)$ $2,0(t_{x,1})$ $2,0(t_{y,1})$	$1,0(h_2)$ $1,0(t_{x,2})$ $1,0(t_{y,2})$	$1,0(h_3)$ $1,0(t_{x,3})$ $1,0(t_{y,3})$	$1,3(h_4)$ $1,3(t_{x,4})$ $1,3(t_{y,4})$	$2,0(h_5)$ $1,4(t_{x,5})$ $0,8(t_{y,5})$	$2,0(h_6)$ $1,0(t_{x,6})$
śr. h_i	11,47	12,48	8,67	11,99	6,50	6,00

Do porównania pracy algorytmów wykorzystuje się również stacjonarne funkcje testowe. Wykorzystane zostały tu funkcje z opracowania [143] oraz [154] (Dodatek E). Badania efektywności pracy proponowanych algorytmów, bazując na tych środowiskach, przedstawiono w rozdziale 2.6.

2.4. Grupowy system immunologiczny implementujący autoimmunizację

Grupowy system immunologiczny jest odpowiednikiem systemu wielopopulacyjnego – osobnik odpowiada populacji. Każdy osobnik ma swój system immunologiczny. Na każdego osobnika – jego system immunologiczny – oddziałuje środowisko. Każdy z osobników żyje w wyznaczonym rejonie środowiska. Obszary te mogą się pokrywać lub tylko mieć części wspólne. Systemy immunologiczne osobników mogą oddziaływać na siebie [90]. Na system immunologiczny mogą oddziaływać wirusy, szczepionki i surowice.

Układ odpornościowy może mieć pewne dysfunkcje. Bywa też, że własne tkanki organizmu, do tej pory tolerowane przez układ odpornościowy, z różnych powodów stają się przedmiotem jego ataku. Autoantygeny albo

zmieniają swoją antygenowość, np. w wyniku zetknięcia się z wirusem, albo niezmienione mogą być zaatakowane przez zmienione (zmutowane) limfocyty. Przełamanie tolerancji wobec własnych antygenów prowadzi do chorób z obszaru autoagresji – choroby autoimmunizacyjnej.

Wirusy są skomplikowanymi cząsteczkami organicznymi, niemającymi struktury komórkowej, które należy uznać za organizmy żywe, tzn. takie, które powinny wykazywać dziedziczną zmienność wpływającą na możliwości rozmnożenia się, czyli być zdolne do ewolucji. Chorobotwórcze działanie wirusów ma zasięg miejscowy lub ogólny.

Szczepionki są preparatami pochodzenia biologicznego, stosowanymi w celu wywołania odpowiedzi immunologicznej pod wpływem kontaktu z antygenem.

Surowice (antytoksyny) są preparatami leczniczymi, zawierającymi swoiste przeciwciała skierowane przeciwko egzotoksynom lub jadom.

W przypadku degradacji układu odpornościowego stosowane są przeszczepy szpiku kostnego, pozwalające na jego odbudowę. Zabieg taki poprzedzony jest zniszczeniem zdegradowanego układu odpornościowego.

Choroby powodują pewnego rodzaju destabilizację układu odpornościowego (autoimmunizacja czy wirusy), natomiast zadaniem szczepionek i surowic jest wspomaganie układu odpornościowego.

Przedstawiony opis ma tylko zasygnalizować wybrane problemy organizmów żywych, których próbę implementacji przedstawiono w dalszej części.

Ingerencją w materiał genetyczny organizmów w celu zmiany ich właściwości dziedzicznych zajmuje się inżynieria genetyczna. Jej rolą jest wprowadzaniu do komórek organizmu określonego odcinka DNA innego organizmu. Do przenoszenia DNA wykorzystywane są m.in. wirusy. Inżynieria genetyczna stosowana jest również do wytwarzania tzw. organizmów transgenicznych – organizmów zmodyfikowanych genetycznie, w skrócie GMO (*Genetically Modified Organisms*). Ma ona również duże znaczenie w poznaniu funkcji pełnionych przez określone geny. Metody inżynierii genetycznej, w odniesieniu do rozpatrywanych zagadnień, można określić w następujący sposób: metodami inżynierii genetycznej są metody analizy modyfikacji i wytwarzania materiału genetycznego, który może zostać użyty na przykład jako surowica, szczepionka lub wirus.

W sztucznym systemie immunologicznym autoagresję komórek można zdefiniować w następujący sposób.

Definicja 2.3 (autoimmunizacja – autoagresja)**.** *Z przeciwciał zgrupowanych w odległości mniejszej niż zadana pozostaje tylko jedno o największej wartości funkcji przystosowania. Pozostałe przeciwciała są usuwane i zastępowane losowo utworzonymi przeciwciałami.*

Jak już wspomniano, usuwanie najsłabszych ze zgrupowanych i otoczonych przeciwciałami antygenów stanowi interpretację mechanizmu ich redukcji.

Definicja 2.4 (wirus)**.** *Wirus aktywuje najbardziej podobne do siebie przeciwciała, które mutując pod jego wpływem zbliżają się do niego. Wirus może również mutować pod wpływem przeciwciał.*

Powoduje to zmniejszenie odległości przeciwciał, które stają się bardziej podatne na autoimmunizację. Wirusy mogą oddziaływać na wszystkie osobniki grupy. Aby to zrealizować, wprowadzono parametr tolerancji, definiujący odległość, w promieniu której przeciwciała i antygeny uznawane są za identyczne (dla przeciwciał i antygenów parametr ten może być różny).

Definicja 2.5 (przeszczep)**.** *Przeszczepem będzie zastąpienie losowo wybranych przeciwciał biorcy losowo wybranymi przeciwciałami dawcy, znajdujących się we wspólnym obszarze środowiska.*

Losowe zastąpienie przeciwciał jednego osobnika przeciwciałami drugiego ma charakter eksploracyjny. Podobnie jak wpływ wirusów, wpływ przeszczepu na pracę systemu immunologicznego w niniejszej pracy nie będzie bliżej analizowany.

Implementacja autoimmunizacji wprowadza pewnego rodzaju niestabilność systemu immunologicznego – zapobiega nadmiernemu grupowaniu przeciwciał, a co za tym idzie stagnacji algorytmu.

Szczepionki i surowice muszą zostać wytworzone i może się to odbywać w organizmach innych osobników lub przy wykorzystaniu metod inżynierii genetycznej. Poniżej przedstawiono ich definicję.

Definicja 2.6 (surowica)**.** *Surowicą będą przeciwciała lub ich klony wytworzone przez innego osobnika pod wpływem antygenu, który jest najbardziej podobny do aktywnego antygenu osobnika pobierającego surowicę, przy założeniu, że znajdują się one we wspólnym obszarze środowiska.*

Zatem możemy analizować dwa rodzaje surowicy, dalej oznaczane jako: surowica I – przeciwciała i surowica II – klony wytworzone przez przeciwciała.

Definicja 2.7 (szczepionka)**.** *Szczepionką będzie antygen innego osobnika, który jest najbardziej podobny do aktywnego antygenu osobnika szczepionego, przy założeniu, że znajduje się on we wspólnym obszarze środowiska.*

Takie podejście stanowi pewne uproszczenie, niemniej jednak może być bazą do analizy zjawisk, które będą wytworzone w systemie immunologicznym podczas stosowania szczepionki czy surowicy. Podjęto więc próbę sformułowania algorytmu selekcji klonalnej implementującego autoimmunizację, działanie wirusów oraz wymianę materiału genetycznego, bazującą na surowicy i szczepionce.

Poniżej przedstawiono opis działania algorytmu selekcji klonalnej, implementującego omawiane powyżej elementy:

1. Pierwsza populacja przeciwciał, antygenów i wirusów tworzona jest w sposób losowy.

2. W sposób losowy wybierany jest aktywny antygen.*Ten punkt algorytmu implementuje szczepienie. U dawcy aktywowany jest antygen znajdujący się w najmniejszej odległości od aktywnego antygenu biorcy. Jeżeli antygen dawcy jest lepszy od antygenu biorcy, to go zastępuje. Antygeny odpowiadają za aktywowanie przeciwciał, a co za tym idzie za eksplorację środowiska; zatem antygeny nie powinny się skupiać. Skupianie się antygenów identyfikuje ekstremum lokalne, jednak przechowywanie więcej niż jednego antygenu identyfikującego ekstremum lokalne wydaje się być zbędne. W promieniu tolerancji aktywnego antygenu sprawdzana jest obecność innych antygenów. Jeżeli takie występują, domniemywa się istnienie w tym obszarze ekstremum lokalnego. Najlepszy antygen staje się aktywny, a pozostałe antygeny zastępowane są w sposób losowy. Dzięki temu cały obszar przestrzeni rozwiązań jest nieustannie przeszukiwany.
3. *System immunologiczny zostaje wystawiony na działanie wirusów. Wirusy znajdujące się w pobliżu aktywnego antygenu aktywują najbardziej podobne do siebie przeciwciała. Przeciwciała te mutując pod wpływem wirusa zbliżają się do siebie. Wirusy mogą również mutować pod wpływem przeciwciał. Powoduje to zmniejszenie odległości przeciwciał, które stają się bardziej podatne na autoimmunizację.
4. *Surowica I. U dawcy aktywowany jest antygen znajdujący się w najmniejszej odległości od aktywnego antygenu biorcy. Antygen ten aktywizuje przeciwciała, które zostają wszczepione biorcy. W następnym kroku algorytmu wszczepione przeciwciała mogą być aktywizowane przez antygen.
5. Aktywny antygen aktywizuje przeciwciała, które są do niego najbardziej podobne (w dalszym opisie nazywane aktywnymi przeciwciałami). W tym punkcie algorytmu implementowana zostaje autoimmunizacja. Polega ona na selekcji, w wyniku której pozostają najlepsze przeciwciała niepozostające we wspólnym obszarze tolerancji. Pozostałe przeciwciała są usuwane jako podobne i zastępowane losowo utworzonymi przeciwciałami. Zapobiega to nadmiernemu skupianiu przeciwciał na bardzo małym obszarze. Dzięki temu wytworzone przez nie klony będą efektywniej realizowały eksploatację tego fragmentu przestrzeni rozwiązań. W istotny sposób wpływa to również na funkcję kosztu.
6. Aktywne przeciwciała produkują klony w liczbie zależnej od wartości funkcji przystosowania – im wyższa wartość funkcji przystosowania, tym przeciwciało produkuje więcej klonów.
7. *Surowica II. U dawcy aktywowany jest antygen znajdujący się w najmniejszej odległości od aktywnego antygenu biorcy. Antygen ten aktywizuje przeciwciała, które produkują klony. Surowica – wyprodukowane klony – zostają wszczepione biorcy.
8. Następnie wyprodukowane klony podlegają mutacji (hipermutacji), która jest zależna od wartości przystosowania klonu (antygenu, który go wyprodukował) – im większa wartość funkcji przystosowania, tym mniej-

sza mutacja. Założono, że zmutowane klony powinny znajdować się wewnątrz obszaru ograniczonego cechami aktywnych przeciwciał. Klony wykonują więc eksploatację ograniczonej przestrzeni rozwiązań.

9. Klony najbardziej podobne do aktywnych przeciwciał, które je wyprodukowały, zastępują je, jeżeli ich wartość funkcji przystosowania jest większa. Zadaniem przeciwciał jest tworzenie węzłów siatki podlegającej adaptacji. Przeciwciała powinny poruszać się w kierunku ekstremów lokalnych, jednak znacznie wolniej w obszarach o większej wartości funkcji celu (w obszarach o większej wartości funkcji celu powinno ich być więcej niż w pozostałych obszarach). Jeżeli wśród klonów znalazło się przeciwciało o wartości funkcji przystosowania większej niż antygen, który je aktywował, to go zastępuje. Antygeny bardzo szybko się przemieszczają w kierunku ekstremum lokalnego.
10. Najsłabsze z aktywnych przeciwciał są usuwane i zastępowane przeciwciałami tworzonymi w sposób losowy. Usuwanie przeciwciał z obszarów o najmniejszej wartości funkcji dostosowania może utrudniać eksplorację zmiennego środowiska. W stosunku do klasycznego algorytmu tę funkcję przejął mechanizm autoimmunizacji. W omawianym algorytmie będzie to realizowane w bardzo ograniczonym zakresie, gdy będą użyte przeciwciała surowicy I, celem utrzymania stałej liczby przeciwciał.
11. Punkty od 2 do 10 algorytmu powtarzają się w sposób iteracyjny do osiągnięcia kryterium stopu.

Kroki 2^*, 3^*, 4^* i 7^* są opcjonalne. W tych krokach algorytmu są implementowane elementy inżynierii genetycznej, kierunkujące rozwój populacji przeciwciał i antygenów.

Produkcja zmutowanych klonów odpowiada za eksploatację przestrzeni rozwiązań i może być realizowana w następujący sposób:
zmutowany klon jest tworzony w odległości $d_{Ab-C} = \sqrt{\sum_{i=1}^{n}(k_i - \vartheta_i)^2}$ od przeciwciała przy założeniu, że odległość klonu od antygenu nie będzie większa niż $d_{Ab-Ag} = \sqrt{\sum_{i=1}^{n} k_i^2}$, gdzie: $K = Ab - Ag$ – wektor odległości przeciwciała $Ab = [ab_1, ..., ab_n]$ od antygenu $Ag = [ag_1, ..., ag_n]$, i $\Theta = [\vartheta_1, ..., \vartheta_n]$ – wektor losowy, n– wymiar przestrzeni, $i = 1, .., n$ (w analizowanym przypadku $n = 2$ i $\vartheta_i \in [-1,3; 1,3]$).

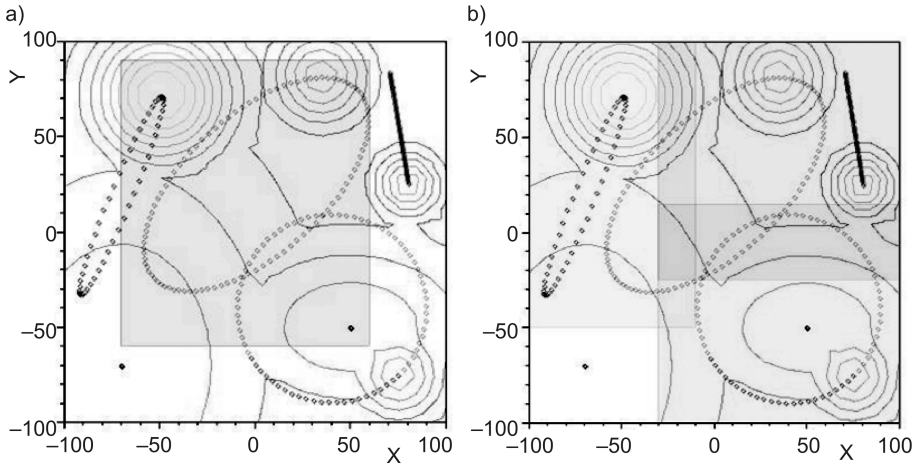
Uwaga dotycząca rozkładu próbkowania przestrzeni rozwiązań

Twierdzenia 2.2 i 2.3 można odnieść do zmutowanych klonów w przestrzeni rozwiązań ograniczonej cechami przeciwciał, wytwarzających te klony. Liczba klonów nie musi być duża, gdyż w kolejnych iteracjach zbliżają się one do siebie – ich zagęszczenie rośnie. Jak już wspomniano, działanie to ma charakter adaptacyjny i decyduje o rozkładzie próbkowania przestrzeni rozwiązań. Można więc wnioskować, że dzięki mechanizmowi adaptacji rozkład próbkowania będzie optymalny.

Z omawianą sytuacją możemy mieć do czynienia np. w systemach kooperujących lub wtedy, gdy analizowana przestrzeń rozwiązań musi zostać podzielona na mniejsze obszary.

2.4.1. Zachowanie algorytmu immunologicznego

Z systemem immunologicznym może być kojarzony osobnik mający taki system. Każdy z systemów immunologicznych znajduje się w wyznaczonym obszarze przestrzeni rozwiązań. Obszary te mają części wspólne. Z taką sytuacją możemy mieć do czynienia np. w systemie współpracujących robotów lub wtedy, gdy analizowana przestrzeń musi zostać podzielona na mniejsze obszary podczas obliczeń równoległych. Poniżej analizowane są dwa schematy życia osobników w zmiennym środowisku. Pierwszy z nich przypomina klasyczny, wielopopulacyjny system, w którym obszary przestrzeni rozwiązań pokrywają się. W tym przypadku analizowana będzie grupa dwóch systemów immunologicznych. W drugim schemacie rozpatrywana przestrzeń rozwiązań pokrywa się tylko w obszarach granicznych, tak jak to przedstawiono na rysunku 2.2, stanowiąc około 20% całkowitej powierzchni (ciemniejsze pasy). W drugim przypadku analizowana będzie grupa trzech systemów immunologicznych.



Rysunek 2.2. Analizowane obszary życia osobników w zmiennym środowisku – zmiany położenia ekstremów zaznaczono punktami: a) dwa osobniki, b) trzy osobniki

Nietrudno zaobserwować, że liczba ekstremów lokalnych w cyklach życiowych będzie się zmieniać. Poniżej zostaną opisane wybrane parametry oraz ich wpływ na pracę algorytmu. W analizowanych przykładach środowisko podlega zmianom, a osobniki nie poruszają się.

Dla tak zamodelowanego zmiennego środowiska przeprowadzono próbę identyfikacji zmian położenia ekstremów lokalnych z wykorzystaniem grupowego systemu immunologicznego. Założono pewne stałe parametry pracy algorytmu: liczba antygenów jest równa 6 oraz liczba przeciwciał jest równa 20. Algorytm realizowany jest w 100 cyklach zmian środowiska, pomiędzy którymi występują 4 fazy stabilne.

W pierwszym scenariuszu dla każdego z osobników zdefiniowano różne obszary tolerancji. Obszary tolerancji przeciwciał i antygenów dla osobnika pierwszego są większe niż dla drugiego. Dane zamieszczone w tabelach są wartościami średnimi, przeliczonymi na jeden cykl pracy algorytmu. Przy omawianiu rezultatów przeprowadzonych eksperymentów w nawiasach okrągłych podano odnośniki do parametrów podanych w tabelach.

Tabela 2.2. Dane opisujące zachowanie algorytmu immunologicznego (dwa osobniki)

Parametry \ Sposób wymiany	Brak		Szczepionka		Surowica I		Surowica II	
System immunologiczny	I	II	I	II	I	II	I	II
1. Produkowane klony	8,94	16,20	8,67	17,68	9,88	21,13	5,31	11,86
2. Aktywne przeciwciała	3,87	3,46	3,82	3,44	4,12	3,81	2,30	3,10
3. Modyfikacja przeciwciał	1,21	1,25	0,92	1,25	1,21	1,58	0,26	0,55
4. Modyfikacja antygenu	0,59	0,59	0,37	0,46	0,66	0,75	0,69	0,46
5. Redukowane przeciwciała	1,73	0,45	0,92	0,36	2,54	1,51	0,11	0,09
6. Redukowane antygeny	0,45	0,31	0,43	0,34	0,43	0,33	0,52	0,28
7. Średni błąd wyznaczenia ekstremum lokalnego	0,29	0,27	0,26	0,21	0,27	0,22	0,25	0,20

Pierwszymi istotnymi parametrami są liczby produkowanych klonów (1) oraz aktywnych przeciwciał (2), które mają wpływ na eksploatację obszaru ograniczonego cechami aktywnych przeciwciał. Zbyt duża liczba przeciwciał spowoduje nadmierną produkcję klonów. Ta nadprodukcja wpłynie na wydłużenie czasu obliczeń, czyli koszt pracy algorytmu.

Liczba modyfikowanych przeciwciał (3) i antygenów (4) świadczy o postępie eksploatacyjnym algorytmu. Natomiast parametry określające redukcję przeciwciał (5) i antygenów (6) mówią o intensywności eksploracji.

Parametry te mogą być regulowane przez obszar tolerancji, jak również oddziaływanie wirusów. Redukcje przeciwciał (5) będą świadczyły o ich zbieżności w obszarach ekstremów lokalnych, natomiast redukcja antygenów (6) będzie świadczyła o identyfikacji ekstremów lokalnych. Uwaga: zbyt duża

liczba przeciwciał i antygenów może powodować wydłużenie czasu identyfikacji zmian środowiska.

Liczba aktywnych przeciwciał jest podobna we wszystkich przypadkach. Aktualizowanie przeciwciał jest ściśle związane z produkcją klonów, dlatego u drugiego osobnika parametr ten jest większy niż u pierwszego. Wskaźniki redukcji antygenów i przeciwciał są większe u pierwszego osobnika, co świadczy o większej intensywności eksploracji. W przypadku wymiany materiału genetycznego, bazującego na szczepionce i surowicy I modyfikacja antygenu występuje częściej u drugiego osobnika. Natomiast dla pierwszego osobnika modyfikacja antygenu jest częstsza w przypadku surowicy II. Jest to oczywiste, gdyż pierwszy osobnik ma lepsze właściwości eksploracyjne, a surowicą II są klony przeciwciał, które w tym przypadku podnoszą jego skuteczność eksploatacyjną. Ponieważ osobnik pierwszy wykazuje znacznie większą intensywność eksploracji przestrzeni rozwiązań, więc skutkuje to nieznacznie większym średnim błędem wyznaczenia ekstremum lokalnego.

Przeciwciała tworzą samoadaptującą się siatkę, której węzły przemieszczają się w kierunku ekstremów lokalnych poza obszarami basenów przyciągania szybciej niż w ich wnętrzu.

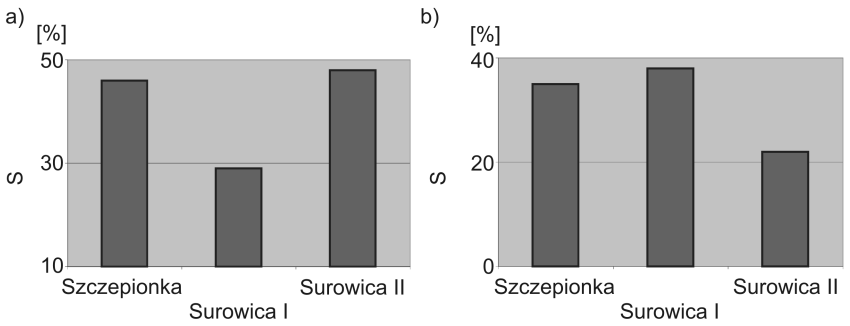
Przemieszczanie się przeciwciał można opisać parametrem średniej drogi przebytej poza obszarami basenów przyciągania, która jest średnio o 53% większa u pierwszego osobnika niż u drugiego. Wewnątrz obszarów basenów przyciągania prędkość przemieszczania się przeciwciał osobnika pierwszego jest również większa o średnio 40%. Prędkość przemieszczania przeciwciał poza obszarami basenów przyciągania jest również większa niż wewnątrz o średnio 48% u pierwszego osobnika i 33% u drugiego. Konsekwencją tego jest większy udział przeciwciał wewnątrz obszarów ekstremów lokalnych i wynosi średnio 59%.

Antygeny mają znacznie większą prędkość przemieszczania w kierunku najlepszych rozwiązań niż przeciwciała, średnio o 60%. Zachowują się one podobnie jak przeciwciała – ich średnia droga przebyta poza obszarami basenów przyciągania jest średnio o 18% większa u pierwszego osobnika i większa o około 25% niż średnia droga przebyta wewnątrz obszarów basenów przyciągania. Również konsekwencją tego jest większy udział antygenów wewnątrz obszarów ekstremów lokalnych, który wynosi (średnio) 75%.

Te dane również uwidaczniają znacznie większą intensywność przeszukiwania przestrzeni rozwiązań u pierwszego osobnika.

Wskaźnikiem skuteczności wymiany materiału genetycznego jest oddziaływanie na antygen, czyli przeprowadzenie jego modyfikacji z udziałem pobranego materiału genetycznego.

W przypadku szczepienia zbierana jest informacja o modyfikacji antygenu biorcy, która to jest miarą efektywności wymiany materiału genetycznego. Tak jak należało przypuszczać, osobnik pierwszy wymienia więcej antyge-



Rysunek 2.3. Skuteczność wymiany materiału genetycznego dla osobnika: (a) pierwszego, (b) drugiego; gdzie S – skuteczność wymiany materiału genetycznego

nów, ponieważ ma większą intensywność eksploracji, co skutkuje mniejszą dokładnością wyznaczania ekstremów lokalnych (patrz rysunek 2.3).

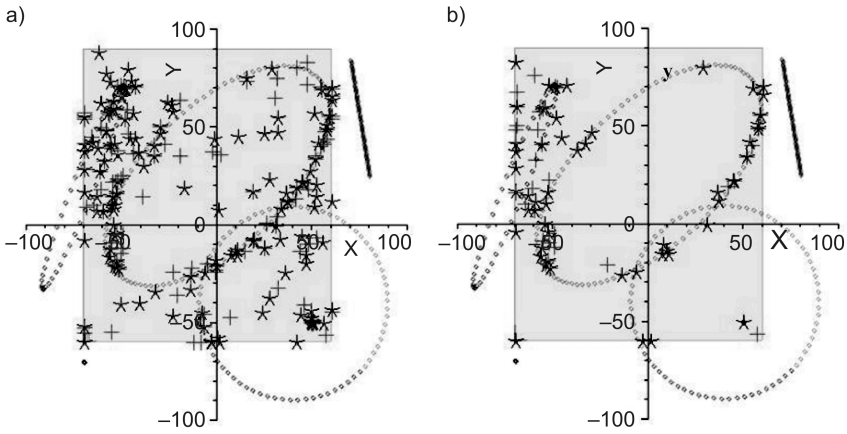
W przypadku podawania surowicy I monitorujemy utworzone przeciwciała surowicy, których jest średnio $4,7$ dla pierwszego osobnika i $6,1$ dla drugiego. Natomiast udział przeciwciał wszczepionych w kontenerze przeciwciał informuje, jaka ilość materiału genetycznego została przyjęta przez biorcę – dla pierwszego osobnika jest ich średnio 23%, a dla drugiego 33%. Większa liczba wszczepionych przeciwciał będzie skutkowała większą liczbą produkowanych przez nie klonów na większym obszarze, a co za tym idzie – zwiększeniem skuteczności wymiany materiału genetycznego, tak jak to uwidacznia rysunek 2.3. Ponieważ wszczepianie przeciwciał powoduje zmianę liczby przeciwciał w kontenerze, więc aby zachować ich stałą liczbę usuwane są najsłabsze z nich, w ilości około 3%.

W przypadku podawania surowicy II zbierana jest informacja o ilości wszczepionych przeciwciał surowicy u biorcy – jest ich średnio u pierwszego osobnika $13,5$, a u drugiego $9,6$. Przeciwciała surowicy zachowują się jak klony, w związku z tym sprawdzany był również ich wpływ na przeciwciała systemu immunologicznego. Wskaźnik ten okazał się bardzo niski, co sugeruje marginalny wpływ na działanie systemu immunologicznego w tym zakresie. W tym przypadku podanie surowicy II u osobnika pierwszego jest bardziej efektywne (patrz rysunek 2.3).

Na podstawie powyższych danych można stwierdzić, że surowica II poprawia warunki eksploatacyjne, a surowica I warunki eksploracyjne systemu immunologicznego. Szczepionka wykazuje wysoką skuteczność.

Poniżej przedstawiono rysunki obrazujące położenie antygenów, określających najlepsze rozwiązania. Dobrą skuteczność wyznaczania ekstremów potwierdzają również uzyskane wartości liczbowe.

Podobną analizę przeprowadzono dla trzech osobników, dla których zdefiniowano takie same obszary tolerancji. Analizowane dane uzyskano dla mniej-



Rysunek 2.4. Antygeny znajdujące się w pobliżu ekstremów lokalnych (a) i identyfikujące ekstrema globalne (b)

szych obszarów tolerancji zarówno antygenów, jak i przeciwciał niż w przykładzie pierwszym, a liczba faz stabilnych wynosi 16.

Tabela 2.3. Dane opisujące zachowanie algorytmu immunologicznego (trzy osobniki)

Sposób wymiany / Parametry	Brak	Szczepionka	Surowica I	Surowica II
1. Produkowane klony	15,71	16,03	11,75	16,52
2. Aktywne przeciwciała	3,12	3,15	2,30	3,21
3. Modyfikacja przeciwciał	1,57	1,74	0,91	1,55
4. Modyfikacja antygenu	0,37	0,37	0,31	0,38
5. Redukowane przeciwciała	0,36	0,42	0,17	0,55
6. Redukowane antygeny	0,04	0,05	0,03	0,05
7. Średni błąd wyznaczenia ekstremum lokalnego	0,06	0,05	0,05	0,06

W odniesieniu do przykładu pierwszego zwiększono liczbę produkowanych klonów (1) przy podobnym wskaźniku aktywnych przeciwciał (2). Dzięki temu zwiększy się eksploatacja obszaru ograniczonego cechami aktywnych przeciwciał. Wpływa to bezpośrednio na dokładność wyznaczania ekstremów lokalnych.

Jak już wspomniano, liczby aktualizowanych przeciwciał (3) i antygenów (4) świadczą o postępie eksploatacyjnym algorytmu.

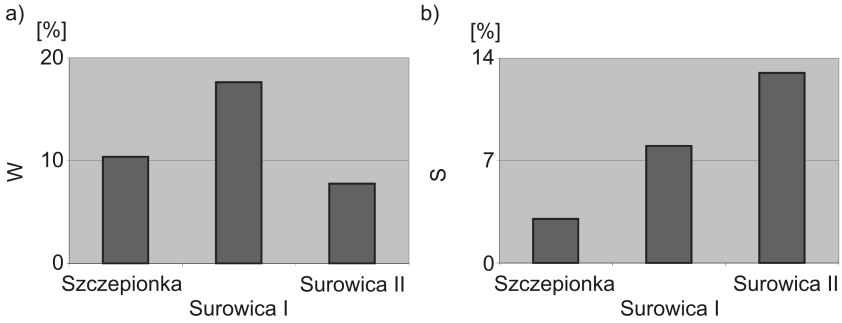
W analizowanym przypadku zdefiniowano mniejszy obszar tolerancji. Efektem tego są znacznie niższe niż w poprzednim przykładzie parametry określające redukcję przeciwciał (5) i antygenów (6). Ma to bezpośredni wpływ na zmniejszenie postępu eksploracyjnego algorytmu immunologicznego.

Znacznie mniejszy postęp eksploracyjny algorytmu potwierdzają również parametry średniej drogi, które są mniejsze o około 77% w stosunku do poprzedniego przykładu.

Parametr średniej drogi przeciwciał przebytej poza obszarami basenów przyciągania jest średnio o 51% większy niż wewnątrz tych obszarów. Udział przeciwciał wewnątrz obszarów ekstremów lokalnych jest nieznacznie większy niż poza nimi i wynosi (średnio) 54%. Ze względu na to, że średnia wartość powierzchni obszarów przyciągania jest mniejsza od pozostałej części przestrzeni rozwiązań, to gęstość przeciwciał wewnątrz obszarów przyciągania będzie większa niż poza nimi.

Antygeny mają znacznie większą prędkość przemieszczania w kierunku najlepszych rozwiązań niż przeciwciała – średnio o 64%. Ich średnia droga przebyta poza obszarami basenów przyciągania jest większa o 63%, niż średnia droga przebyta wewnątrz obszarów basenów przyciągania. Również konsekwencją tego jest większy udział antygenów wewnątrz obszarów ekstremów lokalnych i wynosi (średnio) aż 95%. Ze względu na mniejszą wartość obszaru tolerancji, wskaźnik ten jest znacznie wyższy niż w poprzednim przykładzie. Znacznie mniejsza intensywność przeszukiwania przestrzeni rozwiązań skutkuje zwiększeniem eksploatacji basenów przyciągania. Konsekwencją tego jest dużo mniejszy średni błąd wyznaczenia ekstremum lokalnego niż w poprzednim przypadku.

Na rysunku 2.5 przedstawiono parametry opisujące skuteczność wymiany materiału genetycznego.



Rysunek 2.5. Parametry opisujące skuteczność wymiany materiału genetycznego: (a) wskaźnik wymiany materiału genetycznego – W, (b) skuteczność wymiany materiału genetycznego – S

W przypadku wymiany materiału genetycznego zbierana jest informacja o zainicjowaniu takiej wymiany. W analizowanym przykładzie wskaźnik

ten jest znacznie zróżnicowany. Powodem tego jest zarówno niewielki obszar wspólny, stanowiący $20,27\%$ całej przestrzeni rozwiązań, jak i mechanizm wymiany.

W przypadku szczepienia wskaźnik wymiany materiału genetycznego jest bardzo mały, co jest widoczne na diagramie z rysunku 2.5b. Wynika to z faktu, że aby zaszła modyfikacja, to antygen musi znajdować się we wspólnym obszarze.

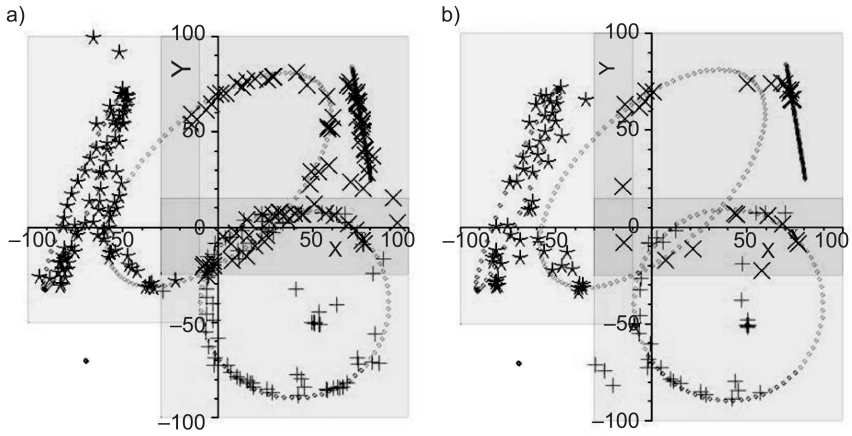
W przypadku podawania surowicy I monitorujemy wskaźnik przyjętych przeciwciał, który wynosi 1%. Wskaźnikiem skuteczności jest modyfikacja antygenu przez klony wszczepionych przeciwciał – wskaźnik ten wynosi 8%, tak jak na wykresie z rysunku 2.5b. Wskaźnik ten jest również niski. Konsekwencją tego jest niski udział przeciwciał wszczepionych w kontenerze przeciwciał, który wynosi średnio 10%. Ponieważ wszczepianie przeciwciał powoduje zmianę liczby przeciwciał w kontenerze, więc celem zachowania stałej liczby przeciwciał usuwamy najsłabsze z nich, określając jednocześnie ich średnią liczbę – w tym przypadku jest ona bardzo mała i wynosi $0,3\%$.

W przypadku podawania surowicy II zbierana jest informacja o produkowanych przeciwciałach surowicy, których liczba średnio wynosi $7,5$. Wskaźnik przeciwciał surowicy wszczepionych biorcy wynosi 58%. Wynika to z faktu, że wszczepiane przeciwciała surowicy muszą znajdować się wewnątrz obszaru biorcy. Natomiast wskaźnik modyfikacji antygenu przez przeciwciała surowicy jest tu najwyższy i wynosi 13%. Przeciwciała surowicy zachowują się jak klony, w związku z tym mają one wpływ na eksploatację przestrzeni obszarów granicznych, dlatego też surowica II ma w tym przypadku największą skuteczność.

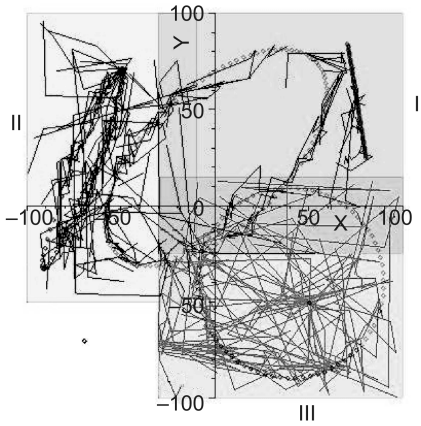
Poniżej przedstawiono rysunki obrazujące położenie antygenów, określających najlepsze rozwiązania i jak można zaobserwować algorytm realizuje to bardzo dobrze, o czym świadczą również średnie wartości błędu wyznaczania ich położenia ekstremum.

Zachowanie się antygenów można przedstawić w zależności od wartości funkcji dostosowania, dzieląc obszar rozwiązań na pełny zakres: obszary basenu przyciągania i sąsiedztwo ekstremów lokalnych. Na rysunku 2.7 przedstawiono drogi poruszania się przeciwciał w tak zdefiniowanych obszarach.

Jeżeli rysunek 2.7 (III) przypomina ruch „chaotyczny", przeszukujący całą przestrzeń rozwiązań, to rysunki 2.7 (I) i 2.7 (II) przypominają „eyetracking" [117]; antygen przemieszcza się w kierunku ekstremum lokalnego i podąża za jego zmianami. Jak widać, antygeny dość dokładnie odzwierciedlają zmiany środowiska. Analiza pracy algorytmu ewolucyjnego jest rzeczą skomplikowaną i w większości przypadków sprowadza się do porównania błędów wyznaczania ekstremów lokalnych czy ekstremum globalnego. W następnym rozdziale przedstawiono możliwości analiz fraktalnej i multifraktalnej w ocenie pracy algorytmu.



Rysunek 2.6. Antygeny znajdujące się w pobliżu ekstremów lokalnych (a) i identyfikujące ekstrema globalne (b)



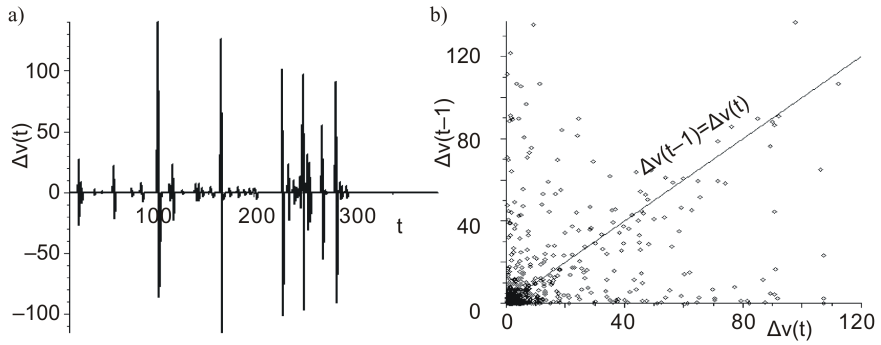
Rysunek 2.7. Drogi przebyte przez antygeny w zależności od wartości funkcji dostosowania

2.4.2. Analizy fraktalna i multifraktalna pracy algorytmu immunologicznego

Jak w poprzednim rozdziale, tak i tutaj analiza pracy algorytmu realizowana jest w zmiennym środowisku. Zachowanie przeciwciał i antygenów można przedstawić jako wykres zmiany prędkości w kolejnych krokach iteracji $\Delta v\,(t) = v\,(t) - v\,(t-1)$ – rysunek 2.8a. Wysokie piki dodatnie oznaczają redukcję. Kolejne cykle przypominają silnie tłumione oscylacje. Świadczy to o szybkiej stabilizacji ruchu przeciwciał. Podobnie zachowują się antygeny.

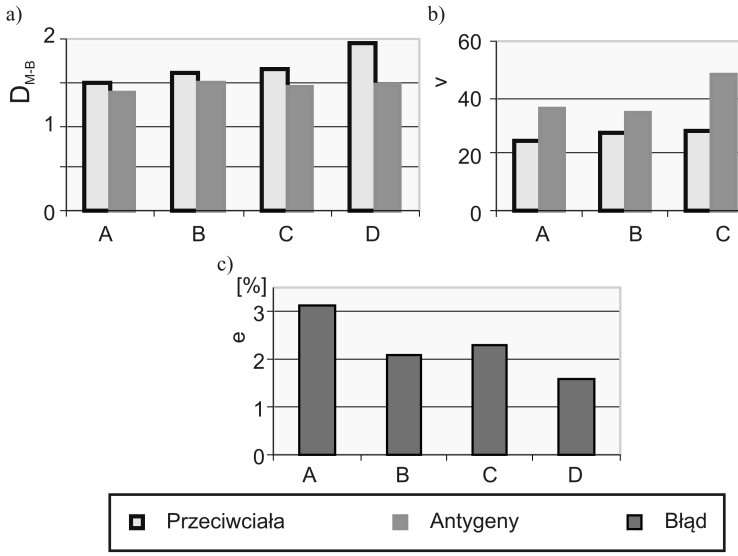
Zachowanie przeciwciał i antygenów można przedstawić jako wykres zmiany prędkości w poprzednim kroku iteracji odniesionej do zmiany prędko-

ści w następnym kroku iteracji – rysunek 2.8b. Przeciwciała leżące nad prostą $\Delta v\,(t-1) = \Delta v\,(t)$ zmniejszają swoją prędkość. Przeciwciała znajdujące się blisko osi $\Delta v\,(t)$ o dużych wartościach zmiany prędkości podlegały redukcji.



Rysunek 2.8. Wykresy ilustrujące zmiany prędkości przeciwciał w kolejnych krokach iteracji (a) i w stosunku do kroku poprzedniego (b)

Większość przeciwciał ma małe prędkości zmian. Więcej przeciwciał w poprzednim kroku miało większą zmianę prędkości niż w następnym (podobnie zachowują się antygeny).

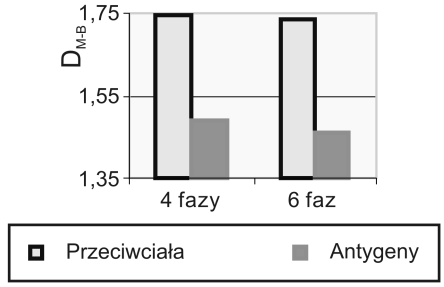


Rysunek 2.9. Wpływ obszaru tolerancji na parametry opisujące zachowanie przeciwciał i antygenów: (a) wymiar fraktalny, (b) prędkość, (c) błąd względny wyznaczenia ekstremów

Analiza ta obrazuje zachowanie przeciwciał i antygenów, jednak nie daje możliwości porównania zachowania algorytmu w zależności od różnych jego

parametrów. Możliwości takie dają analizy fraktalna i multifraktalna. Zachowanie przeciwciał i antygenów może być regulowane przez parametr tolerancji. Na rysunku 2.9 przedstawiono charakterystyki będące funkcją parametru tolerancji; i tak: A – oznacza ustawienia podstawowe, B – zwiększony obszar tolerancji dla antygenów, C – zwiększony obszar tolerancji dla przeciwciał, D – zwiększony obszar tolerancji przeciwciał i antygenów.

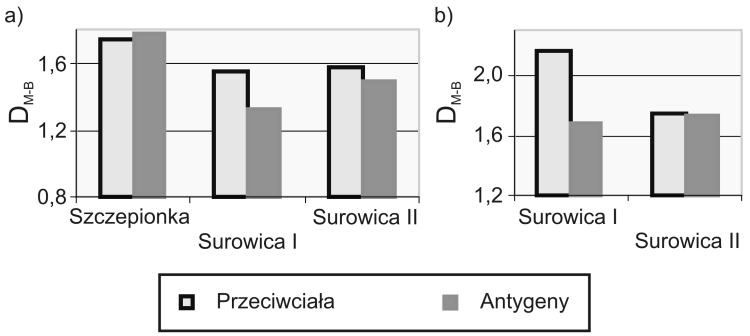
Na rysunku 2.9a przedstawiono przykładowe wymiary fraktalne, opisujące zachowanie przeciwciał i antygenów w zależności od zmiany obszaru tolerancji. Wzrost średniej prędkości przemieszczania przeciwciał powoduje równomierną eksplorację przestrzeni rozwiązań, co skutkuje wzrostem wartości wymiaru fraktalnego. Natomiast wzrost prędkości antygenów nie powoduje istotnego wzrostu wartości wymiaru fraktalnego, gdyż podlegają one silnemu grupowaniu. Na rysunku 2.9b pokazano średnie zmiany prędkości przeciwciał i antygenów, co odpowiada omawianemu zachowaniu. Natomiast rysunek 2.9c przedstawia średni błąd względny wyznaczenia ekstremów. Wysoka eksploracja przestrzeni rozwiązań inicjowana redukcjami przeciwciał i antygenów pozwala lepiej identyfikować zmiany środowiska. W połączeniu ze wzrostem prędkości przeciwciał i antygenów pozwala na ich grupowanie i eksploatację basenów przyciągania, co skutkuje zmniejszeniem błędu wyznaczania ekstremów.



Rysunek 2.10. Zachowanie przeciwciał i antygenów w zależności od liczby faz

W algorytmie można zwiększyć liczbę faz stabilnych. Spowoduje to zmniejszenie błędu wyznaczania ekstremów – odpowiednio dla czterech faz uzyskano $3,1\%$, a dla sześciu faz $2,7\%$. Zachowanie się przeciwciał i antygenów zobrazowano przez wyznaczenie wymiaru fraktalnego. Z rysunku 2.10 wynika, że zwiększenie liczby faz powoduje wzrost grupowania.

W grupowym systemie immunologicznym możliwa jest wymiana materiału genetycznego. Na rysunku 2.11 przedstawiono zachowanie przeciwciał i antygenów opisane wymiarem fraktalnym. Jedynym przypadkiem, kiedy wymiar fraktalny antygenów jest wyższy od wymiaru fraktalnego przeciwciał jest wymiana antygenów. Szczepionka natomiast wpływa na większe grupowanie antygenów. Wykres z rysunku 2.11a sporządzono na podstawie danych pocho-

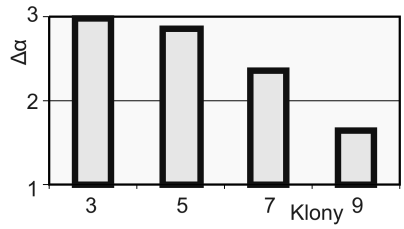


Rysunek 2.11. Wymiana materiału genetycznego dla: (a) dwóch osobników, (b) trzech osobników

dzących z wymiany materiału genetycznego dla dwóch osobników, a wykres z rysunku 2.11b dla trzech osobników.

W przypadku szczepienia występuje zwiększenie grupowania przeciwciał, co nie poprawia w istotny sposób grupowania antygenów. Ten rodzaj wymiany powoduje, że wymiar fraktalny dla antygenów jest wyższy niż dla przeciwciał. W przypadku dwóch osobników, gdy przestrzenie rozwiązań są takie same, surowica I poprawia grupowanie antygenów. Dzieje się tak za sprawą poprawy eksploracji przestrzeni rozwiązań przez przeciwciała. Mechanizm ten nie jest wyraźnie widoczny na wykresie, gdyż jak wspomniano towarzyszy mu redukcja najsłabszych przeciwciał, a pobierane przeciwciała znajdują się w nieznacznej odległości od basenów przyciągania. Ten rodzaj wymiany znacznie poprawia rozkład przeciwciał w przypadku trzech osobników, gdyż jest to związane z lepszą eksploracją obszarów granicznych. Ponieważ surowica II ma lepsze właściwości eksploatacyjne, więc na tym wykresie występują istotne różnice w wartości wymiaru fraktalnego dla przeciwciał (patrz rysunek 2.11b). Potwierdza to wnioski sformułowane w poprzednim podrozdziale.

Zachowanie się przeciwciał zależnie od liczby produkowanych klonów można również określić na podstawie $\Delta\alpha$, wyznaczonego ze spektrum multifraktalnego – rysunek 2.12.



Rysunek 2.12. Wpływ liczby klonów na zachowanie przeciwciał

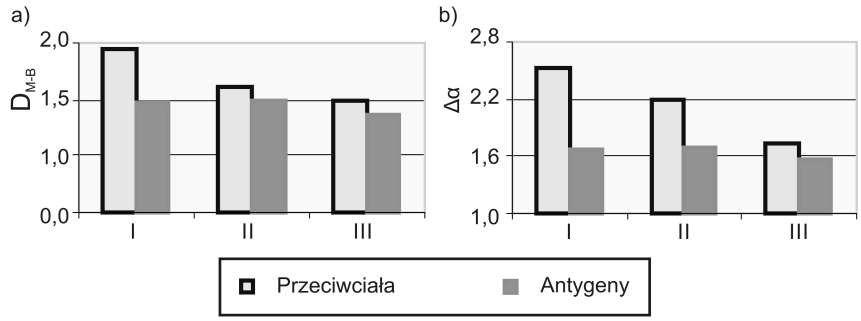
Wyraźnie widoczny jest wzrost grupowania przeciwciał w przestrzeni rozwiązań wraz ze zwiększaniem liczby klonów.

W celu porównania wpływu mutacji na zachowanie przeciwciał i antygenów, zdefiniowano trzy takie operatory przez określenie parametrów wektora losowego:

1. $\vartheta_i \in [-1,3; 1,3]$,
2. $\vartheta_i \in [-1,0; 1,0]$,
3. $\vartheta_i \in \{-1,0; -0,5; -0,25; 0,25; 0,5; 1,0\}$.

Założenie, że odległość klonu od antygenu nie będzie większa niż odległość przeciwciała od antygenu określa kierunek mutacji. Kolejne operatory mutacji mają charakter coraz bardziej ograniczający przemieszczanie przeciwciał.

Zachowanie przeciwciał i antygenów można opisać wymiarem fraktalnym, tak jak na rysunku 2.13a lub przedstawić jako $\Delta\alpha$, wyznaczone na podstawie analizy multifraktalnej – rysunek 2.13b.

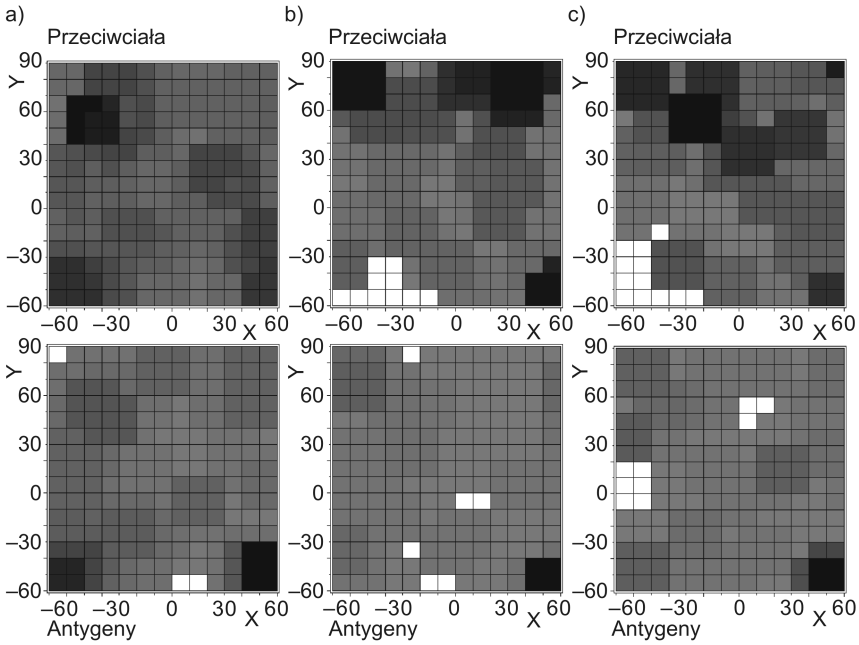


Rysunek 2.13. Wpływ operatora mutacji przedstawiony przez: (a) analizę fraktalną, (b) analizę multifraktalną

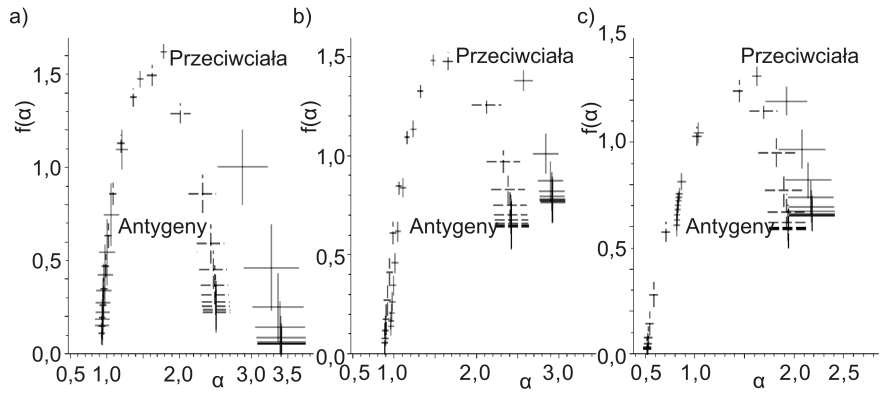
Obie metody w podobny sposób ilustrują zachowanie się przeciwciał i antygenów w zależności od operatora mutacji. Dla lepszego zilustrowania wpływu operatora mutacji, na rysunku 2.14 przedstawiono przykłady obrazów danych będące wynikiem zachowania się przeciwciał i antygenów. Dane te wykorzystywane są do wyznaczenia wymiaru fraktalnego oraz spektrum multifraktalnego obrazu i opisują prawdopodobieństwo wystąpienia przeciwciał i antygenów w danym obszarze.

Spektrum multifraktalne (rysunek 2.15) w wyraźny sposób obrazuje różnice w zachowaniu przeciwciał i antygenów, polegające na wzroście ich grupowania.

Obie prezentowane metody bardzo dobrze nadają się do oceny działania omawianego algorytmu immunologicznego i mogą być wykorzystane do porównania algorytmów oraz przy doborze parametrów ich pracy. Metoda ta może być szczególnie przydatna do analizy zachowania algorytmu w przestrzeni o dużej liczbie wymiarów.



Rysunek 2.14. Przykładowe obrazy danych będące wynikiem zachowania się przeciwciał i antygenów pod wpływem różnych operatorów mutacji (oznaczenia I, II, III do rysunków a, b, c – patrz opis)



Rysunek 2.15. Spektrum multifraktalne dla omawianych operatorów mutacji (oznaczenia I, II, III do rysunków a, b, c – patrz opis)

2.5. Algorytm S-PSO osaczania – „beset game”

W żartobliwy sposób strategię działania algorytmu przedstawiają słowa piosenki:

Nie o to chodzi by złowić króliczka
Ale by gonić go...
Zieliński A., Osiecka A., Króliczek, Skaldowie, Muza 1969.

Proponowany algorytm przedstawia koewolucję dwóch systemów cząstek: drapieżników i ich zdobyczy (*predator-prey*). Drapieżniki posługują się strategią osaczania – drapieżników jest więcej niż zdobyczy. Osaczana przez grupę drapieżników zdobycz porusza się szybciej niż one. Wymknięcie się zdobyczy z jednaj obławy powoduje wpadnięcie w następną. Zdobycz na podstawie obserwacji lidera drapieżników i najsłabszego z nich ucieka w stronę bezpiecznego schronienia, czyli ekstremum lokalnego. W algorytmie tym sąsiedztwo drapieżników i zdobyczy musi być wyznaczane w każdej iteracji. Drapieżniki poruszają się w kierunku zdobyczy oraz w kierunku lidera grupy, który stara się odciąć drogę ucieczki zdobyczy. Drapieżniki i zdobycz skupiają się w obszarach ekstremów lokalnych. Osaczenie więcej niż jednej zdobyczy powoduje eliminację najsłabszych z nich. Natomiast nadmierne zbliżenie drapieżników powoduje redukcję tych najsłabszych. Mechanizmy te wydają się być naturalnymi elementami walki o przetrwanie. Wyeliminowane osobniki są zastępowane losowo utworzonymi.

Omawiany algorytm można opisać w następujący sposób:

1. Losowe tworzenie drapieżników ($E_i = [e_{i1}, e_{i2}, ..., e_{id}]$) oraz zdobyczy ($Z_i = [z_{i1}, z_{i2}, ..., z_{id}]$) – system koewolucyjny.
2. Aktywacja zdobyczy – losowy wybór cząstki.
3. Eliminacja zdobyczy – sprawdzenie czy w zadanym otoczeniu aktywnej zdobyczy znajdują się inne osobniki tej grupy; zdobycze gorzej przystosowane zastępowane są w sposób losowy; zdobycz, która pozostała staje się aktywna.
4. Aktywacja drapieżników – drapieżniki w otoczeniu zdobyczy stają się aktywne.
5. Eliminacja drapieżników – spośród aktywnych drapieżników eliminowane są drapieżniki słabe, znajdujące się zbyt blisko silnych i zastępowane nieaktywnymi drapieżnikami, utworzonymi losowo.
6. Przetworzenie – zastąpienie aktywnych elementów próby i aktywnego zarodnika według następującej reguły: wyznaczenie wartości funkcji przystosowania dla aktywnych drapieżników ($f(E_i)$) oraz wybór najlepszego ($E_{i,best}$) i najgorszego ($E_{i,worst}$) drapieżnika. Obliczenie wektorów prędkości dla drapieżników (V_i^E) i dla zdobyczy (V_i^Z) według następujących wzorów:

$$V_i^E(t+1) = w^E V_i^E(t) + c_1^E \varphi_1^E (E_{i,best} - E_i) + c_2^E \varphi_2^E (Z_i - E_i), \quad (2.4)$$

$$V_i^Z(t+1) = w^Z V_i^Z(t) + c_1^Z \varphi_1^Z (E_{i,best} - Z_i) + c_2^Z \varphi_2^Z (Z_i - E_{i,worst}), \quad (2.5)$$

gdzie: φ_1^E, φ_2^E, φ_1^Z, φ_2^Z – są wartościami losowymi z przedziału $[0, 1]$; c_1^E, c_2^E, c_1^Z, c_2^Z – są współczynnikami wagowymi przyśpieszenia *(acceleration constants)*, zwane też współczynnikami uczenia (*learning rates*); w^E,

w^Z – są współczynnikami wagowymi bezwładności ruchu cząstki (*inertia weight*). Wartości tych współczynników podlegają adaptacji, celem uzyskania odpowiednich scenariuszy zachowań. W algorytmie podawane są tylko maksymalne wartości współczynników wagowych. Duży wpływ na zachowanie ma współczynnik wagowy bezwładności, dlatego jego adaptacja ma istotne znaczenie jeśli chodzi o zachowanie cząstek. Natomiast nowe pozycje drapieżników i pozycja zdobyczy wyznaczane są według wzorów:

$$E_i(t+1) = E_i(t) + V_i^E(t+1), \tag{2.6}$$

$$Z_i(t+1) = Z_i(t) + V_i^Z(t+1). \tag{2.7}$$

7. Iteracja od punktu 2 do osiągnięcia kryterium stopu.

Ruch drapieżników zależy od położenia najlepszego drapieżnika i od położenia zdobyczy. Natomiast ruch zdobyczy zależy od położenia drapieżników najlepszego i najgorszego – w kierunku położenia najlepszego i w przeciwnym do położenia najgorszego drapieżnika. Przetwarzanie to ma charakter lokalny i pozwala wyznaczać ekstrema lokalne.

Drapieżniki są odpowiedzialne za eksplorację przestrzeni rozwiązań, a zdobycz za eksploatację ekstremów lokalnych – jednak między nimi występuje współzależność, która ogranicza wpływ środowiska na ich zachowanie. Współczynniki wagowe są odpowiedzialne za kreowanie scenariuszy zachowań, które to zależą od wzajemnego położenia aktywnych elementów.

W bliskiej odległości od zdobyczy drapieżnik będzie starał się odciąć jej drogę ucieczki (zdobycz ucieka w kierunku ekstremum lokalnego). Znacznie mniejszy wpływ na jego zachowanie ma kierunek, w którym się do tej pory poruszał. Również bezpośredni ruch drapieżnika w kierunku zdobyczy może ułatwić jej ucieczkę. Dlatego też w takiej sytuacji współczynniki wagowe w^E oraz c_2^E powinny mieć mniejszy wpływ na zachowanie drapieżnika. Natomiast dominujący wpływ na zachowanie drapieżnika powinien mieć współczynnik c_1^E odpowiedzialny za poruszanie się w kierunku lidera drapieżników. W takiej sytuacji na ruch lidera drapieżników dominujący wpływ będą miały współczynniki w^E oraz c_2^E, czyli będzie się on kierował w stronę zdobyczy.

W przypadku znacznej odległości drapieżnika od zdobyczy będzie on szybko podążał w jej stronę. Dlatego też istotny wpływ na jego zachowanie będą miały współczynniki wagowe w^E oraz c_2^E. W tym scenariuszu współczynnik c_1^E będzie miał mniejsze znaczenie w zachowaniu drapieżnika.

Scenariusze zachowania zdobyczy będą również zależały od jej położenia względem drapieżników. W przypadku złego położenia zdobyczy, czyli znacznej odległości od lidera drapieżników istotny wpływ na jej zachowanie będą miały współczynniki wagowe w^Z oraz c_2^Z, określające szybką ucieczkę ze złej lokalizacji. Natomiast przy niewielkiej odległości zdobyczy od lidera drapieżników, oznaczającego dotarcie do celu ucieczki, czyli ekstremum lokalnego, istotny wpływ na jej zachowanie będzie miał współczynnik wagowy c_1^Z. Po-

niżej przedstawiono przykłady realizacji funkcji współczynników wagowych. Wariant I:

$$c_1^E = \frac{\|E_{i,best} - E_i\|}{\|E_{i,best} - E_i\| + \|Z_i - E_i\|} \cdot c_{1,max}^E, \tag{2.8}$$

$$c_2^E = \frac{\|Z_i - E_i\|}{\|E_{i,best} - E_i\| + \|Z_i - E_i\|} \cdot c_{2,max}^E, \tag{2.9}$$

$$w^E = c_2^E \tag{2.10}$$

oraz

$$c_1^Z = \frac{\|E_{i,best} - Z_i\|}{\|E_{i,best} - Z_i\| + \|Z_i - E_{i,worst}\|} \cdot c_{1,max}^Z, \tag{2.11}$$

$$c_2^Z = \frac{\|Z_i - E_{i,worst}\|}{\|E_{i,best} - Z_i\| + \|Z_i - E_{i,worst}\|} \cdot c_{2,max}^Z, \tag{2.12}$$

$$w^Z = c_2^Z. \tag{2.13}$$

Wariant II:

$$c_1^E = \frac{\min\left\{\|E_{i,best} - E_i\|, \|Z_i - E_i\|\right\}}{\|E_{i,best} - E_i\|} \cdot c_{1,max}^E, \tag{2.14}$$

$$c_2^E = \frac{\min\left\{\|E_{i,best} - E_i\|, \|Z_i - E_i\|\right\}}{\|Z_i - E_i\|} \cdot c_{2,max}^E, \tag{2.15}$$

$$w^E = \frac{\min\left\{\|E_{i,best} - E_i\|, \|Z_i - E_i\|\right\}}{\|Z_i - E_i\|} \cdot w_{max}^E \tag{2.16}$$

lub

$$w^E = \left(1 - \frac{\min\left\{\|E_{i,best} - E_i\|, \|Z_i - E_i\|\right\}}{\|E_{i,best} - E_i\|}\right) \cdot w_{max}^E, \tag{2.17}$$

oraz

$$c_1^Z = \frac{\min\left\{\|E_{i,best} - Z_i\|, \|Z_i - E_{i,worst}\|\right\}}{\|E_{i,best} - Z_i\|} \cdot c_{1,max}^Z, \tag{2.18}$$

$$c_2^Z = \frac{\min\left\{\|E_{i,best} - Z_i\|, \|Z_i - E_{i,worst}\|\right\}}{\|Z_i - E_{i,worst}\|} \cdot c_{2,max}^Z, \tag{2.19}$$

$$w^Z = \frac{\min\left\{\|E_{i,best} - Z_i\|, \|Z_i - E_{i,worst}\|\right\}}{\|Z_i - E_{i,worst}\|} \cdot w_{max}^Z \tag{2.20}$$

lub

$$w^Z = \left(1 - \frac{\min\left\{\|E_{i,best} - Z_i\|, \|Z_i - E_{i,worst}\|\right\}}{\|E_{i,best} - Z_i\|}\right) \cdot w_{max}^Z, \tag{2.21}$$

gdzie w^Z (równanie 2.17) i w^E (równanie 2.21) będą przyjmowały wartość 0.

W powyższych zależnościach $c^E_{1,max}$, $c^E_{2,max}$, w^E_{max}, $c^Z_{1,max}$, $c^Z_{2,max}$, w^Z_{max} są maksymalnymi wartościami współczynników wagowych.

W scenariuszu algorytmu można uwzględnić sytuację, w której to drapieżniki znajdują się na ściśle wydzielonych, sąsiadujących ze sobą terenach polowań. Z kolei zdobycz może poruszać się po całej tej przestrzeni. Algorytm ten, jako algorytm PSO, należy do grupy algorytmów stochastycznych. Omawiany algorytm w sposób bardzo efektywny łączy stochastyczne przeszukiwanie przestrzeni rozwiązań z przeszukiwaniem opisanym zachowaniem cząstek; wykazuje on również właściwość obserwacji – eyetracking.

2.6. Analiza działania algorytmów w stacjonarnych środowiskach testowych

W rozdziale 2.4 przeprowadzono ocenę pracy grupowego systemu immunologicznego w relatywnie prostym środowisku. W literaturze światowej zaproponowano znacznie bardziej złożone środowiska, które zostały przybliżone w dodatkach niniejszej pracy. Ocenę działania algorytmu przeprowadzono więc przez porównanie skuteczności wyznaczania ekstremum globalnego funkcji testowych w odniesieniu do wybranych algorytmów optymalizacyjnych. Funkcje testowe opisują środowisko stacjonarne. Oznaczenia porównywanych algorytmów, przyjęte w tabelach są następujące: S-IMM – omawiany algorytm (*Semi - Artificial Immune Algorithm*), S-PSO – omawiany algorytm (*Semi - Particle Swarm Optimization*), PSO (*Particle Swarm Optimization*) [143], RCGA (*Real-Coding Genetic Algorithm*) [25], CGA (*Continuous Genetic Algorithm*) [52], CHA (*Continuous Hybrid Algorithm*) [53], ECTS (*Enhanced Continuous Taboo Search*) [54], CTSS (*Continuous Taboo Simplex Search*) [51], SEA (*Simplex and Evolution Algorithm*) [144].

Tabela 2.4 przedstawia zestawienie liczby iteracji potrzebnych do osiągnięcia 100% wskaźnika sukcesu (*success rate*). Tylko dla algorytmu SEA i funkcji *Easoma, Zakharova* oraz *Rosenbrocka* wskaźnik sukcesu wynosi 97%. Omawiany algorytm porównywany jest z siedmioma algorytmami optymalizacyjnymi dla czternastu funkcji testowych. Tego typu analiza, stanowiąca podstawę większości artykułów, pozwala na porównanie efektywności algorytmu, jednak nie obrazuje jego zachowania. Mechanizm zachowania wydaje się być istotny przy doborze algorytmu do rozwiązywanego problemu. W pośredni sposób taką informację uzyskujemy stosując różne funkcje testowe. Wiedza ta może być jednak istotna przy rozwiązywaniu nietypowych problemów. Wskaźnik sukcesu określa możliwość rozwiązania problemu, a liczba iteracji algorytmu, pozwalająca na osiągnięcie sukcesu jest funkcją kosztu. Bardziej precyzyjne określenie kosztu pracy algorytmu uzyskujemy określając liczbę wyznaczeń funkcji przystosowania.

Celem lepszego zilustrowania działania algorytmu, tabelę 2.4 można uzupełnić danymi zaprezentowanymi na rysunku 2.16, określającymi uzyskiwane

Tabela 2.4. Porównanie liczby iteracji proponowanych algorytmów (S-IMM, S-PSO) z innymi algorytmami dla wybranych funkcji testowych

Algorytm / Funkcja	S-IMM	S-PSO	PSO	RCG	CGA	ECTS	CHI	CTS	SEA
Easoma	573	653	740	642	1504	1284	952	325	197*
Fichiera	259	307	580	–	430	–	132	98	258
Shuberta	671	721	800	946	575	370	345	283	420
Goldsteina i Price'a	381	425	480	270	410	231	259	119	–
Zakharova	584	613	380	437	620	195	215	78	90*
Rosenbrocka	469	497	1660	596	960	480	459	369	266*
Branina RCOS	430	433	740	490	620	254	295	125	272
De Jounga 3D	344	358	500	449	750	338	371	155	–
Shekela $_{4,7}$ 4D	557	621	29180	1143	680	910	620	590	–
Shekela $_{4,10}$ 4D	731	768	30160	1235	650	898	635	555	–
Zakharova 5D	982	1066	1530	1115	1350	2254	950	–	–
Zakharova 10D	2709	2764	7440	2190	6991	4630	100	–	–
Rosenbrocka 5D	2291	2423	33100	4150	3990	2142	3290	–	–
Rosenbrocka 10D	6259	6340	106960	8100	21563	15720	14563	–	–

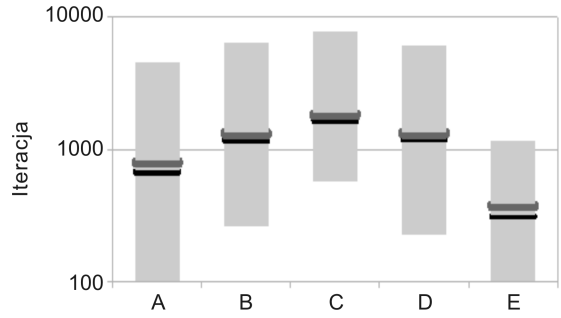
* – wskaźnik sukcesu wynosi 97%.

wartości minimalne, średnie i maksymalne liczby iteracji algorytmu oraz ich medianę i odchylenie standardowe. Cieńszą linią (ciemniejszą) oznaczono dane algorytmu S-IMM, a linią grubszą (jaśniejszą) dane algorytmu S-PSO. Na wykresach szary obszar pokazuje rozrzut wartości uzyskiwany dla różnych środowisk testowych, natomiast linia pozioma reprezentuje wartość średnią.

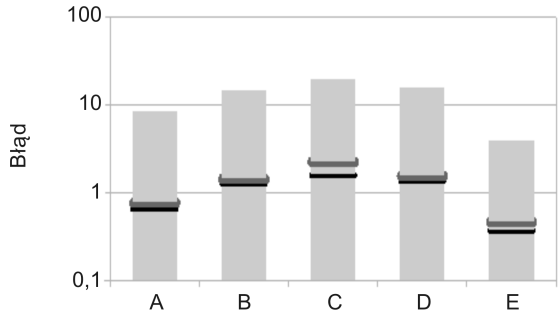
Parametry pracy algorytmu muszą być dobrane w zależności od środowiska. Dla algorytmów S-IMM i S-PSO dobrano podobne parametry pracy. Obszary tolerancji zdobyczy w algorytmie S-PSO (antygenów w algorytmie S-IMM) dla funkcji De Jounga, Shekela są dwukrotnie większe, a dla funkcji Zakharova 10D i Rosenbrocka 10D są pięciokrotnie większe od pozostałych. Wzrost obszarów tolerancji skutkuje zmniejszeniem dokładności wyznaczania ekstremum. Obszary tolerancji drapieżników (przeciwciał) ustawione w algorytmie są takie same, z wyjątkiem funkcji o problemach 10D. W przypadku tych funkcji obszary tolerancji zdobyczy (antygenów) i drapieżników (przeciwciał) są identyczne. Dla funkcji De Jounga, Shekela obszary tolerancji zdobyczy (antygenów) są dwukrotnie mniejsze niż obszary tolerancji drapieżników (przeciwciał), natomiast w pozostałych przypadkach – czterokrotnie mniejsze.

Liczba drapieżników (przeciwciał) wynosi 100, a liczba zdobyczy (antygenów) – 10. Liczba klonów wytwarzanych przez przeciwciała dla funkcji o wymiarach 5D i 10D wynosi 4, dla pozostałych funkcji wartość ta jest równa 3. Ustawienia te pozwolą na porównanie zachowania algorytmu z takimi samymi ustawieniami w różnych środowiskach testowych. Konsekwencją takiego kroku jest nieoptymalna efektywność pracy algorytmu – mimo to uzyskiwane rezultaty są zadowalające, co ilustrują wyniki zaprezentowane na rysunku 2.16.

Na rysunku 2.17 przedstawiono wykres obrazujący dane dotyczące błędu wyznaczenia ekstremum: minimalny, średni, maksymalny, mediana oraz odchylenie standardowe. Uzyskiwane rezultaty dla obydwu algorytmów są bardzo podobne.



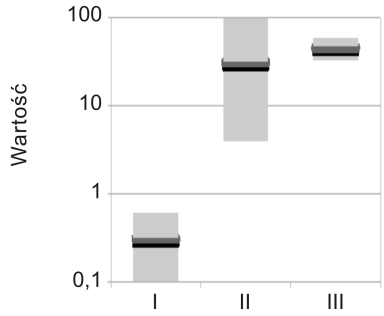
Rysunek 2.16. Liczba iteracji algorytmu: A – minimalna, B – średnia, C – maksymalna, D – mediana, E – odchylenie standardowe



Rysunek 2.17. Błąd wyznaczenia ekstremum: A – minimalny, B – średni, C – maksymalny, D – mediana, E – odchylenie standardowe

Aby lepiej ocenić efektywność pracy algorytmu, na rysunku 2.18 podano wskaźniki jego pracy reprezentujące liczbę wyznaczeń funkcji przystosowania, które zostały przeliczone na jedną iterację algorytmu. Wartość ta zależy od liczby cząstek aktywnych oraz redukowanych. Koszt pracy algorytmu wyraźnie zależy od środowiska i jest tym większy, im trudniej jest wyznaczyć ekstremum. Nietrudno jednak zauważyć, że wskaźniki eliminacji cząstek oraz

aktywnych cząstek, przypadające średnio na jeden cykl pracy algorytmu są bardzo podobne, a środowisko ma na nie niewielki wpływ. Właściwość ta daje szerokie możliwości stosowania tego algorytmu oraz sterowania jego pracą.



Rysunek 2.18. Wskaźniki wyznaczania funkcji dostosowania przeliczone na jeden cykl algorytmu: I – eliminacja zdobyczy (antygenów), II – eliminacja drapieżników (przeciwciał), III – aktywne drapieżniki (przeciwciała)

Mimo iż wszystkich cząstek jest 110, to średnio mniej niż 8% dla algorytmu S-PSO oraz około 19% dla algorytmu S-IMM (ze względu na produkcję klonów przeciwciał) bierze udział w obliczeniach jednego cyklu i podlega mechanizmom adaptacji. Pozostałe cząstki pełnią również bardzo ważną funkcję, gdyż zawierają informacje o pozostałej przestrzeni rozwiązań. Ma to duże znaczenie dla poprawnego działania tego algorytmu (patrz opis działania algorytmu).

W pracy [154] przedstawiono podział środowisk testowych na następujące grupy:

A: funkcje jednomodalne i proste funkcje wielomodalne: 1) sfery, 2) Rosenbrocka;

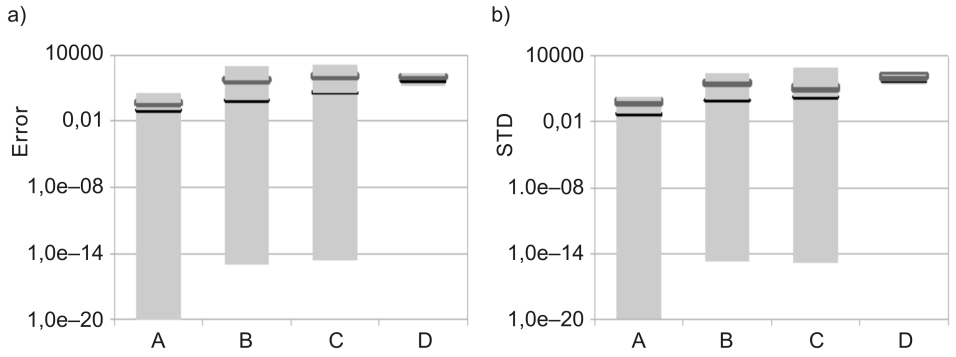
B: nieobrócone funkcje wielomodalne: 3) Ackleya, 4) Griewanka, 5) Weierstrasa, 6) Rastrigina 7) nieciągła Rastrigina, 8) Schwefela;

C: obrócone funkcje wielomodalne: 9) obrócona Ackleya, 10) obrócona Griewanksa, 11) obrócona Weierstrasa, 12) obrócona Rastrigina, 14) obrócona Schwefela;

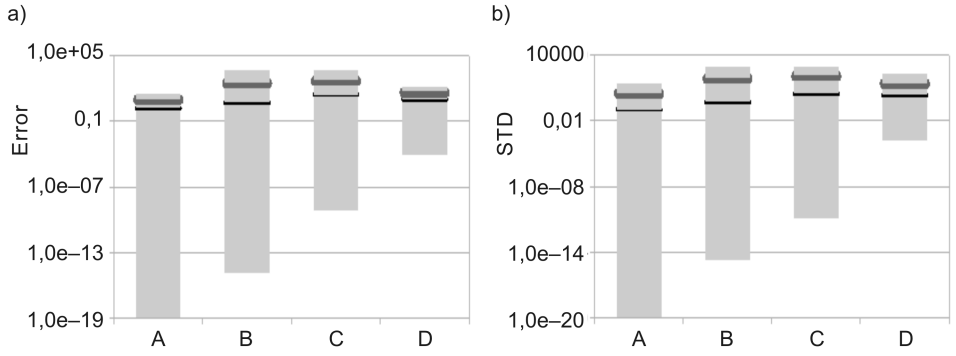
D: złożone funkcje: 15) złożona funkcja 1 (CF1) [153], 16) złożona funkcja 5 (CF5) [153].

Uzyskiwane rezultaty zostały porównane z grupą efektywnych algorytmów – przeważnie hybrydowych: PSO-w (*PSO with inertia weight*) [199]; PSO-cf (*PSO with constriction factor*) [61]; PSO-w-local (*Local version of PSO with inertia weight*), PSO-cf-local (*local version of PSO with constriction factor*) [128], UPSO (*Unified Particle Swarm Optimization*) [180], FIPS (*Fully informed particle swarm*) [161], FDR-PSO (Fitness Distance Ratio based Particle Swarm Optimization) [183]; CPSO-H (Cooperative PSO) [214]; CLPSO (*Cellular PSO*) [154].

Uzyskiwane wartości średnie, prezentowane na wykresach – rysunki 2.19 i 2.20 mieszczą się w zakresach uzyskiwanych dla poszczególnych grup algorytmów. Wzorując się na opracowaniu [154] założono maksymalną liczbę wyznaczeń funkcji przystosowania na 30000 dla przestrzeni 10D oraz 200000 dla przestrzeni 30D. Koszt pracy algorytmów jest więc taki sam. Uzyskiwane rezultaty są nieznacznie lepsze dla algorytmu immunologicznego.



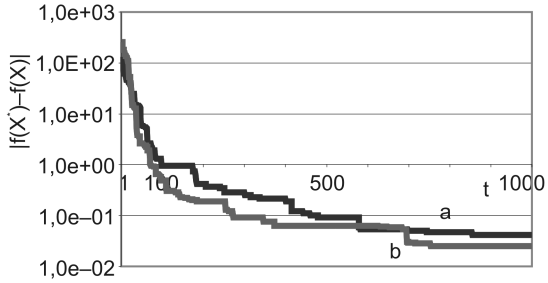
Rysunek 2.19. Błąd (a) oraz odchylenie standardowe (b) wyznaczenia ekstremów w przestrzeni 10D dla grup funkcji testowych: A, B, C, D (opis w tekście)



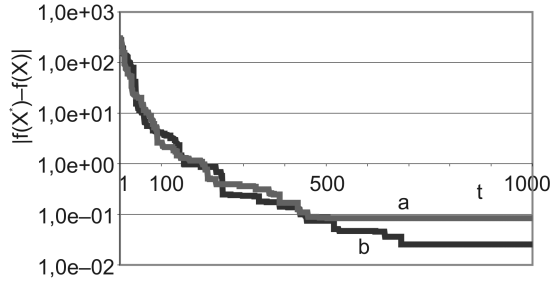
Rysunek 2.20. Błąd (a) oraz odchylenie standardowe (b) wyznaczenia ekstremów w przestrzeni 30D dla grup funkcji testowych: A, B, C, D (opis w tekście)

Przedstawione poniżej omówienie koncentruje się na algorytmie S-PSO, gdyż działania algorytmu S-IMM obszernie prezentuje rozdział 2.4. Rysunek 2.21 przedstawia historię wyznaczania ekstremum globalnego algorytmem S-PSO dla funkcji Goldsteina i Price'a. W przypadku charakterystyki b obszar tolerancji jest mniejszy o 5%. Wykres ten pokazuje wyraźny wpływ obszaru tolerancji na precyzję wyznaczenia ekstremum. Jednocześnie wyznaczenie ekstremum z mniejszą precyzją jest znacznie szybsze. Jest to oczywiste, gdyż czas

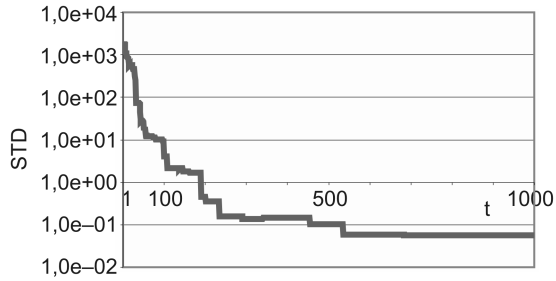
i precyzja wyznaczenia ekstremum globalnego są celami sprzecznymi. Obszar tolerancji pozwala na określenie, który z tych celów będzie realizowany. Analogiczne wykresy prezentowane są dla funkcji Rosenbrocka.



Rysunek 2.21. Historia wyznaczenia ekstremum funkcji Goldsteina i Price'a

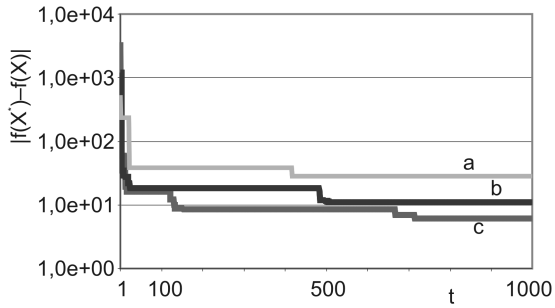


Rysunek 2.22. Historia wyznaczenia ekstremum funkcji Rosenbrocka



Rysunek 2.23. Historia zmian odchylenia standardowego błędu wyznaczenia ekstremum funkcji Fichiera

Odchylenie standardowe wyznaczenia ekstremum globalnego przedstawiono na rysunku 2.23. Wykres ten pokazuje, że dla wszystkich prób błąd wyznaczenia ekstremum lokalnego maleje w czasie. Ten wzrost precyzji wyznaczenia ekstremum ogranicza jednak obszar tolerancji. Obrazuje to zbieżność algorytmu w wyznaczeniu ekstremum globalnego.



Rysunek 2.24. Historia wyznaczenia ekstremum funkcji Zakharova

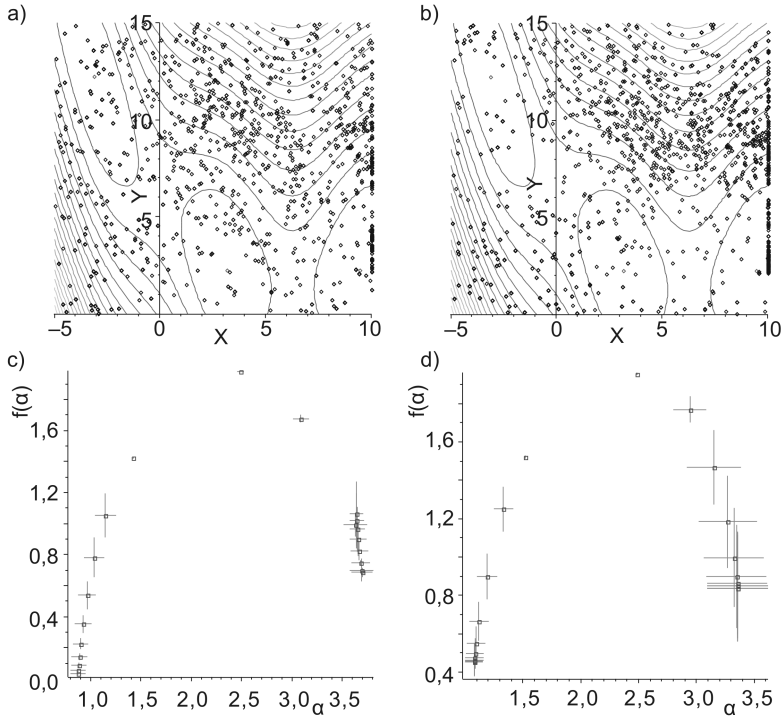
Rysunek 2.24 przedstawia historię wyznaczenia ekstremum globalnego przez prezentację błędu jego wyznaczania. Dla kolejnych przebiegów obszary tolerancji maleją o 5%. Przebieg c ma najmniejszy obszar tolerancji i jednocześnie najmniejszy błąd wyznaczenia ekstremum. Zwraca uwagę fakt, że odnalezienie ekstremum we wszystkich przypadkach zachodzi bardzo szybko.

Tabela 2.5. Dane statystyczne z pracy algorytmu uwzględniające kryterium zatrzymania [213]

Funkcja	Algorytm / Parametr	LDW-PSO	CR-PSO	SE-PSO	D-PSO
Easoma	min	807	813	793	806
	avg	5141	5196	4245	3985
	max	17999	18205	14478	13446
Shuberta	min	2512 (0,94)	2535 (0,94)	1794 (0,98)	1805 (0,99)
	avg	5778 (0,96)	5890 (0,96)	3528 (0,99)	2940
	max	9958	10224	5838	4510
Branina RCOS	min	942 (0,96)	949 (0,96)	856 (0,99)	902 (0,94)
	avg	4002 (0,97)	4043 (0,97)	1766	1658 (0,98)
	max	8496	8698	3088	2664
$Shekela_{4,7}$	min	3516 (0,51)	3318 (0,52)	1613 (0,48)	1885 (0,45)
	avg	8170 (0,62)	8379 (0,63)	2648 (0,54)	2380 (0,54)
	max	12845 (0,82)	13337 (0,81)	5283 (0,72)	3703 (0,73)
$Shekela_{4,10}$	min	4342 (0,62)	4239 (0,63)	1726 (0,49)	1768 (0,45)
	avg	9107 (0,71)	9319 (0,71)	2729 (0,56)	2313 (0,54)
	max	14685 (0,93)	15186 (0,92)	5328 (0,78)	3703 (0,79)

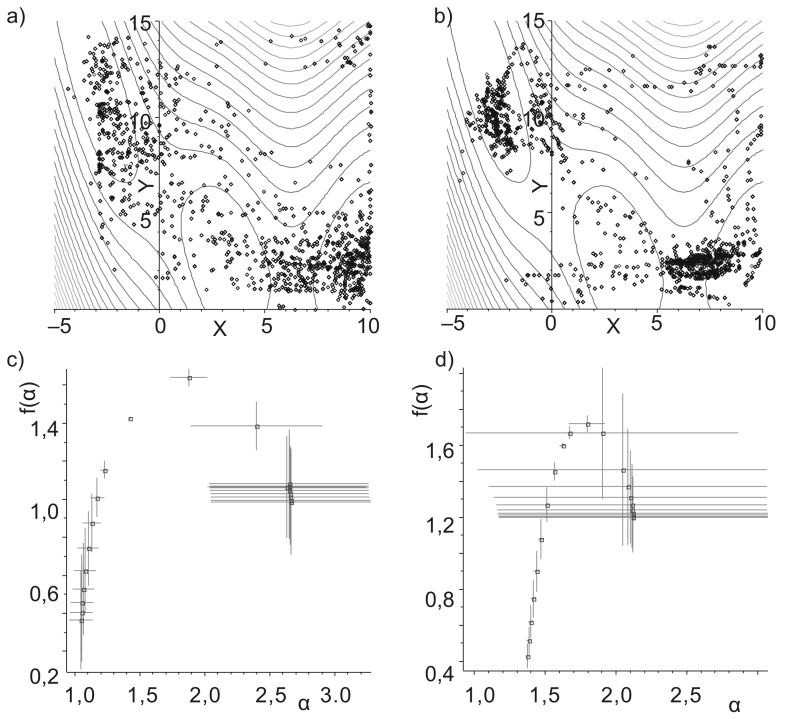
Efektywne kryterium zatrzymania pracy algorytmu jest trudne do osiągnięcia. Pokazują to między innymi badania zamieszone w pracy [213]. W proponowanym algorytmie za wskazanie ekstremum odpowiadają redukcje zdobyczy (patrz opis działania algorytmu). Redukcje te, powtarzając się w tych samych obszarach w połączeniu z niewielkimi zmianami wartości funkcji przystosowania, są pewnym wskaźnikiem ekstremum. Uzyskane wartości kosztu spełniają takie właśnie kryterium. Porównując je z wartościami średnimi, uzyskanymi na podstawie danych prezentowanych w opracowaniu [213] (w nawiasach podano współczynnik skuteczności wyznaczenia ekstremum), możemy stwierdzić, że koszt obliczeń proponowanego algorytmu (w większości przypadków) jest większy – daje on jednak pewność wyznaczenia ekstremum globalnego.

Algorytmy (za [213]) przytoczone w tabeli 2.5 to: LDW-PSO (*Linearly Decreasing Weight Particle Swarm Optimization*), CR-PSO (*Center PSO*), SE-PSO (*Simple PSO*), D-PSO (*Dynamic PSO*).



Rysunek 2.25. Rozkład drapieżników w środowisku (a, b) oraz spektrum multifraktalne prezentowanych rozkładów (c, d)

Do oceny pracy algorytmu może być wykorzystana analiza multifraktalna. Uzyskane widmo multifraktalne daje informacje o efektywności przeszukiwania przestrzeni rozwiązań przez drapieżniki i o zachowaniu zdobyczy. Na rysunkach 2.25a i b przedstawiono rozkład drapieżników w środowisku. Wy-



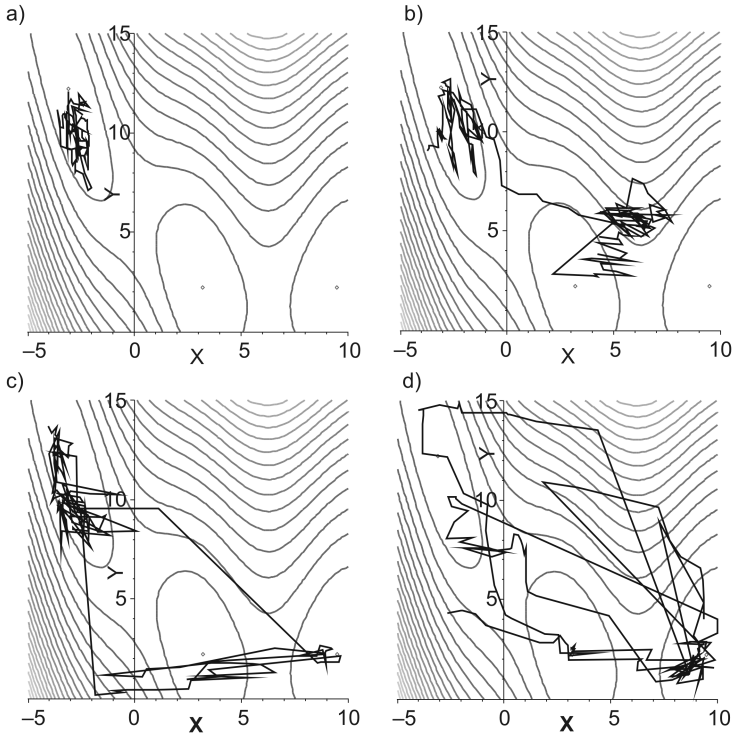
Rysunek 2.26. Rozkład zdobyczy w środowisku (a, b) oraz spektrum multifraktalne prezentowanych rozkładów (c, d)

konują one równomierną eksplorację przestrzeni rozwiązań w poszukiwaniu zdobyczy. Należy tu zwrócić uwagę na fakt, że rozkład drapieżników w pojedynczej iteracji nie jest równomierny, gdyż osaczają one zdobycz. Z porównania wykresów 2.25a i b wynika, że na rysunku 2.25a rozkład drapieżników jest bardziej równomierny. Potwierdzają to wykresy analizy multifraktalnej, gdzie widmo multifraktalne z rysunku 2.25d odpowiada rozkładowi drapieżników z rysunku 2.25b.

Podobną analizę można przeprowadzić obserwując zachowanie zdobyczy w przestrzeni rozwiązań. Na rysunkach 2.26a i b widać wyraźne jej grupowanie w ekstremach lokalnych. Grupowanie zdobyczy ilustruje również widmo multifraktalne. Wykres analizy multifraktalnej 2.26d i odpowiadający mu rozkład zdobyczy z rysunku 2.26b potwierdzają to grupowanie.

Ponadto, analiza multifraktalna może być szczególnie użyteczna przy opracowywaniu zagadnień wielowymiarowych. Wyniki tej analizy mogą być pomocne przy strojeniu algorytmu, a także przy automatyzacji procesu dostrajania algorytmu w czasie jego pracy.

Rysunek 2.27 ilustruje zachowanie osaczanej zdobyczy. Na kolejnych wykresach (rysunki 2.27a, b, c, d) widoczny jest wyraźny wzrost prędkości przemieszczania zdobyczy, spowodowany wzrostem wartości maksymalnych



Rysunek 2.27. Przykładowe drogi przemieszczania się osaczanej zdobyczy

współczynników wagowych. Na rysunku 2.27a zdobycz nie potrafi się wymknąć, co uwidacznia silny charakter eksploatacyjny algorytmu. Natomiast rysunki 2.27b, c, i d ilustrują następujące po sobie osaczenia i ucieczki. Dobrą równowagę pomiędzy eksploracją a eksploatacją pokazują wykresy 2.27b i 2.27c. Na rysunku 2.27b można zaobserwować niewielką przewagę eksploatacji nad eksploracją, natomiast na wykresie 2.27c widać niewielką przewagę eksploracji nad eksploatacją. Silny charakter eksploracyjny algorytmu jest widoczny na rysunku 2.27d. Zachowanie zdobyczy z rysunku 2.27c jest najlepsze, gdyż porusza się ona bardzo szybko poza obszarami ekstremów, zmierzając do kolejnego z nich. Wewnątrz ekstremum zdobycz porusza się znacznie wolniej realizując eksploatację.

Zachowanie to ilustruje również efekt „eyetracking”. Dla rysunku 2.27a można powiedzieć, że wzrok utkwił w jednym punkcie, a dla rysunku 2.27d, że jest rozbiegany. Natomiast na rysunkach 2.27b i c przebiega on po obrazie, koncentrując się na wybranych szczegółach.

Dzięki tym właściwościom algorytm powinien wykorzystywać dużą efektywność pracy również w środowisku niestacjonarnym.

2.7. Analiza działania algorytmów w niestacjonarnym środowisku testowym Moving Peaks Benchmark (MPB)

W rozdziale 2.3 przedstawiono proste środowisko testowe, które było wzorowane na środowisku MPB, a w rozdziale 2.4 przeprowadzono serię badań opierając się na proponowanym środowisku. Zastosowanie nowego środowiska uniemożliwia jednak bezpośrednie porównanie efektywności pracy w odniesieniu do nowoczesnych algorytmów prezentowanych w literaturze światowej. Środowisko MPB jest obecnie uważane za jedno z prostszych środowisk testowych, jednak w literaturze światowej to właśnie ono ma najwięcej odwołań.

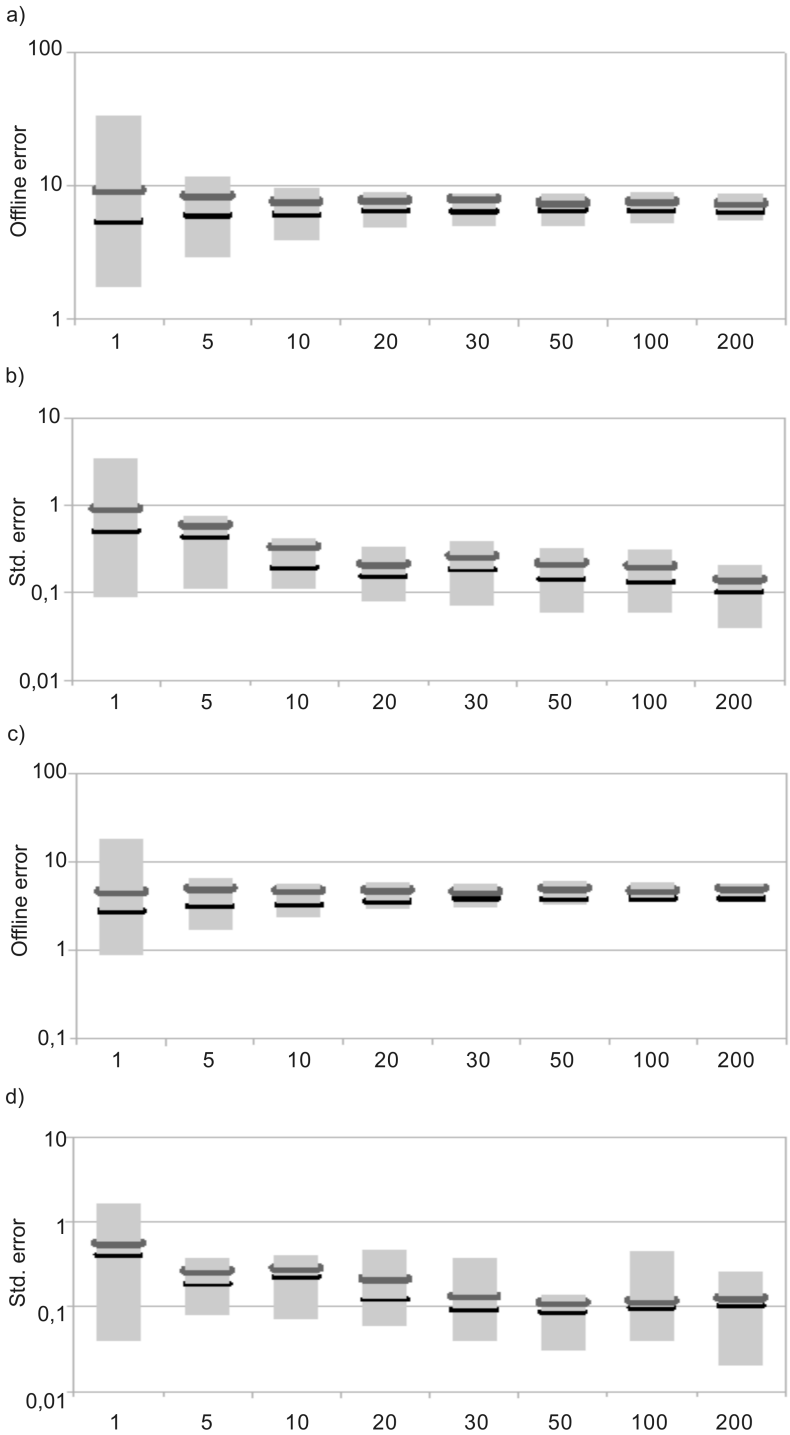
Wzorując się na artykule [226] oraz materiałach dostępnych na portalu opisującym środowisko MPB, przeprowadzono badania pozwalające ocenić efektywność pracy proponowanych algorytmów. Na wykresach z rysunku 2.28 przedstawiono parametr *Offline error* oraz odchylenie standardowe w zależności od liczby wyznaczeń wartości funkcji przystosowania oraz liczby ekstremów. Parametry pracy algorytmu są ustawiane zgodnie ze scenariuszami proponowanymi przez autora środowiska testowego. Zostały one zamieszczone w dodatku C. Dla algorytmu S-IMM określono liczbę klonów na 3. Na wykresach szary obszar oznacza rozrzut parametrów uzyskiwanych przez grupę porównawczą algorytmów. Cienką linią (ciemną) zaznaczono wartości średnie uzyskiwane przez algorytm S-IMM, a linią grubszą (ciemnoszarą) przedstawiono wartości średnie uzyskiwane przez algorytm S-PSO. Dla rysunku 2.28 grupę porównawczą stanowią następujące algorytmy: mQSO (*Quantum multi-Swarm Optimization*) [30], AmSO (*Anti-convergence Swarm Optimization*) [32], mPSO (*Particle multi-Swarm Optimization*) [123], HmPSO (*Hibernating multi-Swarm Optimization*) [124], APSO (*Adaptive Particle Swarm Optimization*) [190], FTMPSO (*finder–tracker PSO*) [226].

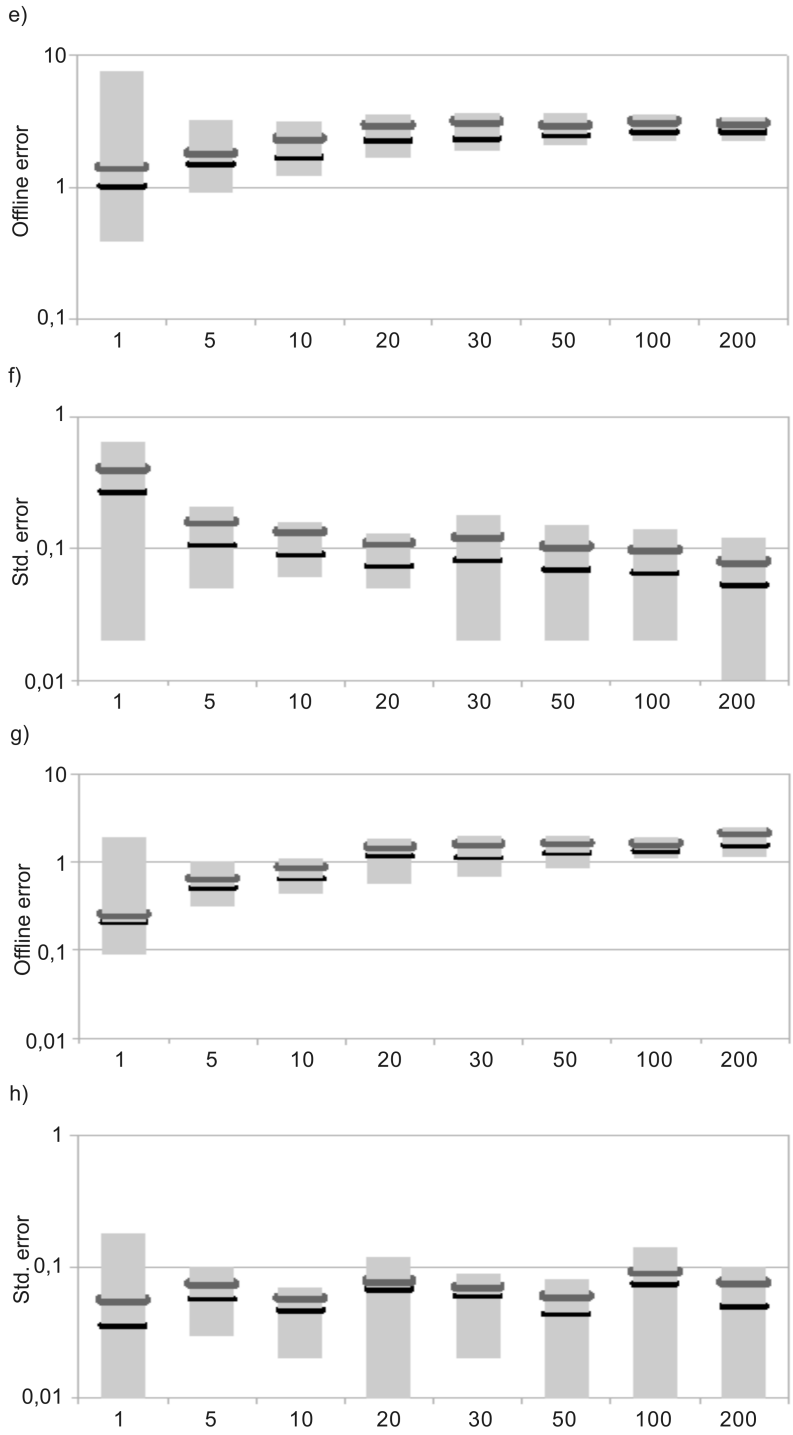
Rezultaty uzyskiwane przez algorytm immunologiczny są nieznacznie lepsze od proponowanego algorytmu S-PSO. Na jego precyzję istotny wpływ ma obszar tolerancji, dlatego też uzyskiwane wartości nie są minimalne. Algorytm S-IMM ma również odmienny sposób penetracji przestrzeni rozwiązań, który bazuje na produkcji klonów. Czynniki te w istotny sposób wpływają na uzyskiwane rezultaty.

Wzorując się na artykułach [224] oraz [226], proponowane algorytmy można porównać z algorytmami bazującymi na wyznaczaniu klastrów i wielopopulacyjnymi: CPSO (*Clustering Particle Swarm Optimizer*) [224], CGAR (*Clustering Genetic Algorithm without change detection*) [148], CDER (*Clustering Differential Evolution algorithm without change detection*) [148], CPSOR (*Clustering Particle Swarm Optimization algorithm without change detection*) [148] oraz FTMPSO (*Finder–Tracker PSO*).

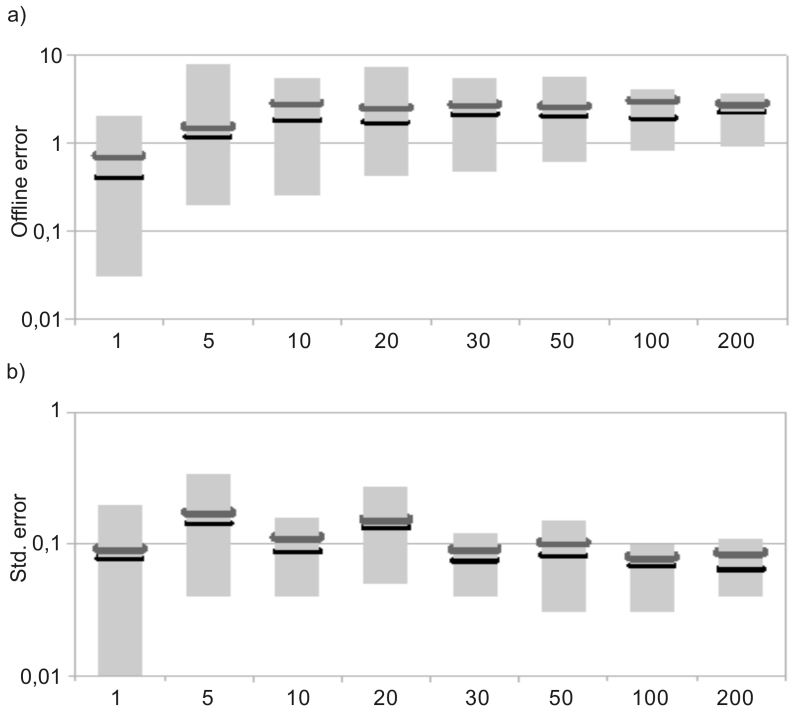
Podobnie jak na wykresach z rysunku 2.28, również na rysunku 2.29 obserwujemy podobne zachowanie obu proponowanych algorytmów.

Jak już wspomniano, środowisko testowe MPB jest dosyć popularne, co umożliwia utworzenie szerokiej grupy porównawczej algorytmów: PSO (*Par-*


a)
Offline error
100
10
1
1
5
10
20
30
50
100
200
b)
Std. error
10
1
0,1
0,01
1
5
10
20
30
50
100
200
c)
Offline error
100
10
1
0,1
1
5
10
20
30
50
100
200
d)
Std. error
10
1
0,1
0,01
1
5
10
20
30
50
100
200



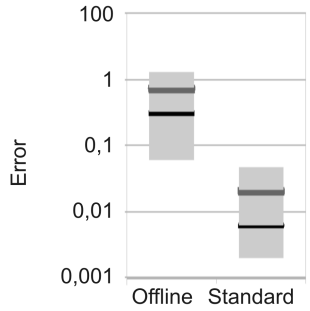
Rysunek 2.28. Ocena pracy algorytmu w zależności od liczby obliczeń funkcji przystosowania (parametr *Change Frequency*) a, b) 500, c, d)1000, e, f) 2500, g, h) 10000 oraz liczby ekstremów równej: 1, 5, 10, 20, 30, 50, 10, 200



Rysunek 2.29. Ocena pracy algorytmu dla liczby obliczeń funkcji przystosowania równej 5000 oraz liczby ekstremów równej: 1, 5, 10, 20, 30, 50, 10, 200

ticle Swarm Optimization) [31], DE (*Differential Evolution*) [162], EO (*Extremal Optimisation*) [169], mQSO (*multi-population Quantum Swarm Optimization*) [31], AmQSO (*adaptive multi-population QSO*) [32], CLPSO (*Cellular PSO*) [102], FMSO (*Fast Multi-Swarm Optimization*) [50], RPSO (*using Regression PSO*) [116], mCPSO [67], SPSO (*Standard PSO*) [70], rSPSO (*Using regression SPSO*) [28], mPSO (*Particle multi-Swarm Optimization*) [123], HmPSO (*Hibernating multi-Swarm Optimization*) [124], PSO-CP (*PSO-Composite Particles*) [156], CESO (*Collaborative Evolutionary-Swarm Optimization*) [157], ESCA (*Evolutionary Swarm Cooperative Optimization Algorithm*) [158], RVDEA (*Relocation Variable Dynamic Evolutionary Algorithm*) [222], SFA (*Speciation Firefly Algorithm*) [170], APSO (*Adaptive Particle Swarm Optimization*) [190], CLDE (*Cellular DE*) [171], MDAIS (*Multi-set Dynamic Artificial Immune System*) [198]. Dla tej grupy algorytmów określono liczbę ekstremów na 10, a liczbę wyznaczeń funkcji dostosowania na 2500.

Otrzymane rezultaty mieszczą się w zakresach grupy porównawczej algorytmów. Również na wykresach z rysunku 2.30 algorytm immunologiczny wypada nieco lepiej niż algorytm S-PSO.

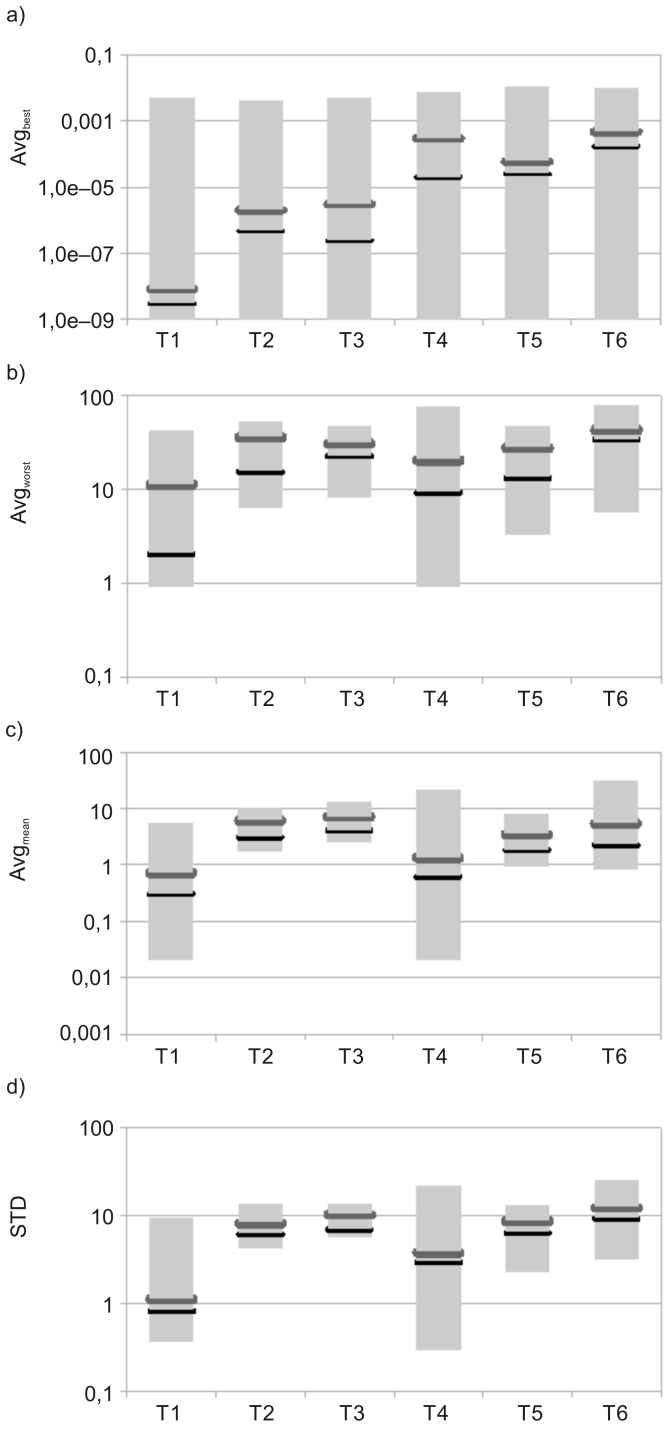


Rysunek 2.30. Ocena pracy algorytmów S-PSO i S-IMM na tle grupy porównawczej

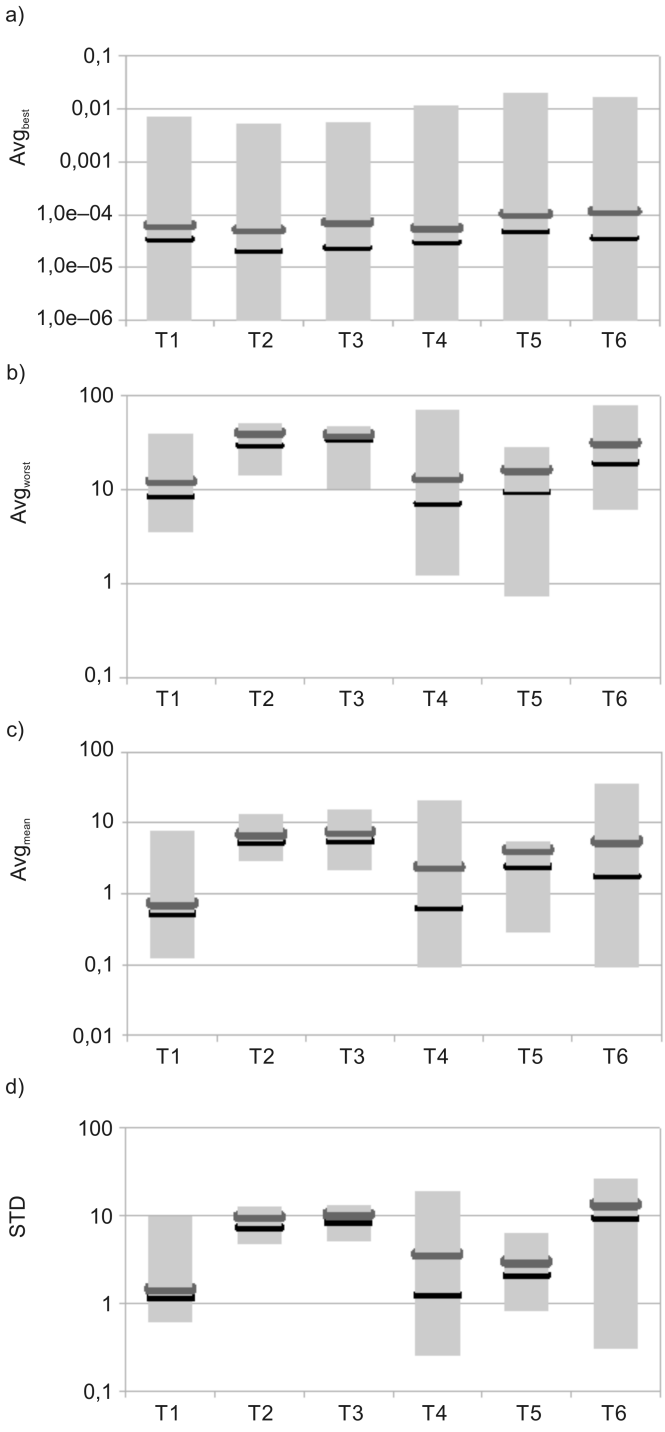
2.8. Analiza działania algorytmów w niestacjonarnym środowisku testowym Generalized Dynamic Benchmark Generator (GDBG)

Środowisko testowe GDBG jet uznawane za jedno z bardziej złożonych środowisk testowych (zostało ono ogólnie przedstawione w dodatku D). Jego popularność jest również dużo mniejsza w porównaniu ze środowiskiem testowym MPB. Podobnie jak w powyżej prezentowanych badaniach, otrzymane rezultaty porównano z zakresami wartości uzyskiwanymi przez grupę algorytmów porównawczych. Do grupy algorytmów porównawczych należą: DAI (*Dynamic Artificial Immune*) [68], DantS (*Differential Ant-Stigmergy*) [135], SGA (*Simple Genetic Algorithm*) [147], Self-ADE (*Self-Adaptive Differential Evolution*) [41], EEM (*Ensemble of Explicit Memories*) [228], SPSO (*Standard PSO*), CPSO (*Clustering Particle Swarm Optimizer*) [70]. Podobnie jak w przypadku środowiska MPB, przyjmuje się również i tu określony scenariusz ustawień. By dokonać oceny pracy algorytmu, porównuje się wartości średnie najlepszych, najgorszych cząstek oraz cząstek medianowych, a także określa się odchylenie standardowe. Na prezentowanych poniżej wykresach opis T1...T6 oznacza scenariusze zmian środowiska. Badania przeprowadzono dla proponowanych dla środowiska funkcji bazowych: rotacji szczytów, złożenia funkcji sfer, złożenia funkcji Rastrigina, złożenia funkcji Ackleya, hybrydowego złożenia funkcji.

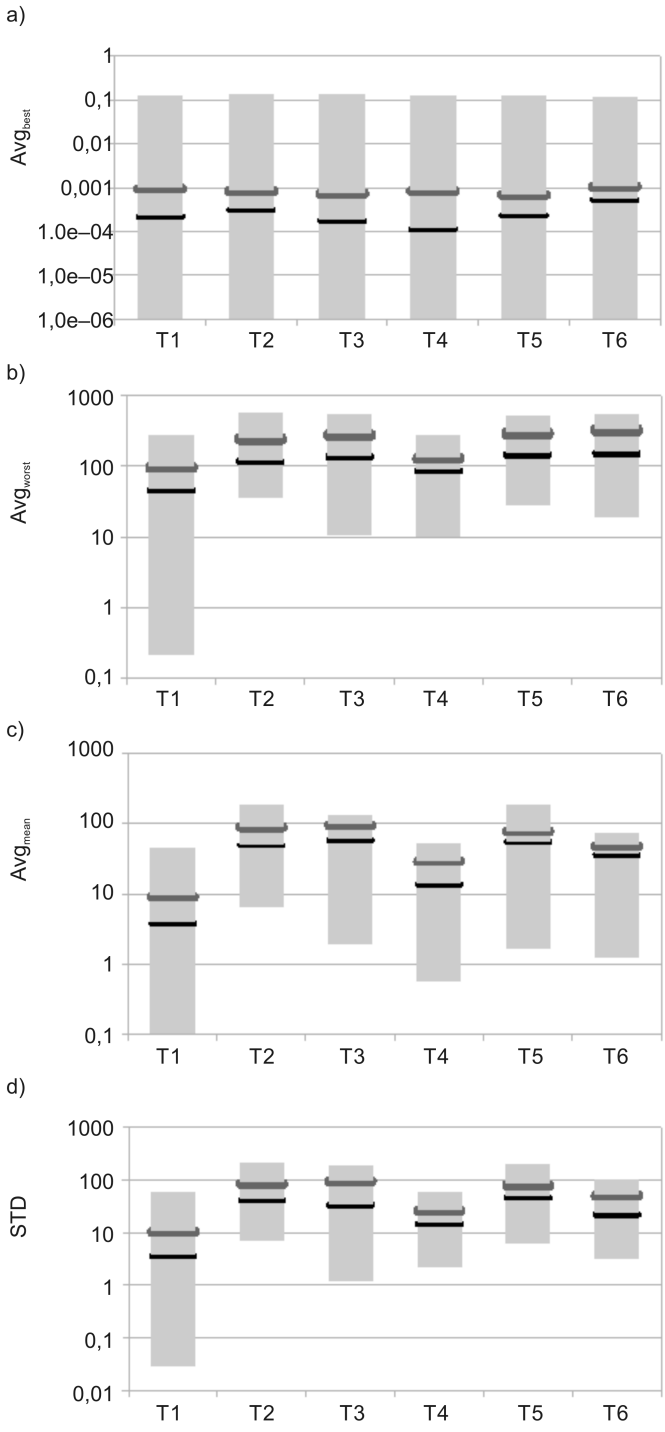
Rezultaty uzyskiwane dla algorytmu S-IMM są nieznacznie lepsze od rezultatów uzyskiwanych dla algorytmu S-PSO.



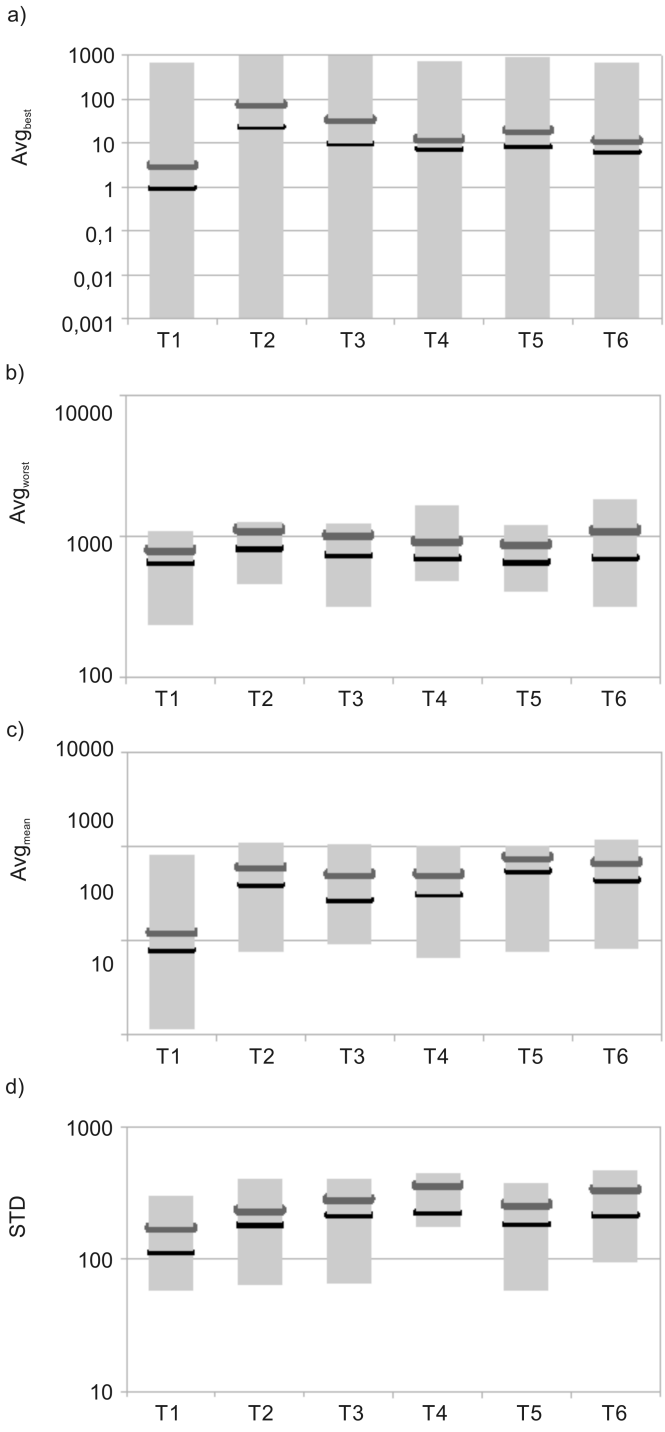
Rysunek 2.31. Rezultaty pracy algorytmów dla funkcji rotacji szczytów z dziesięcioma ekstremami



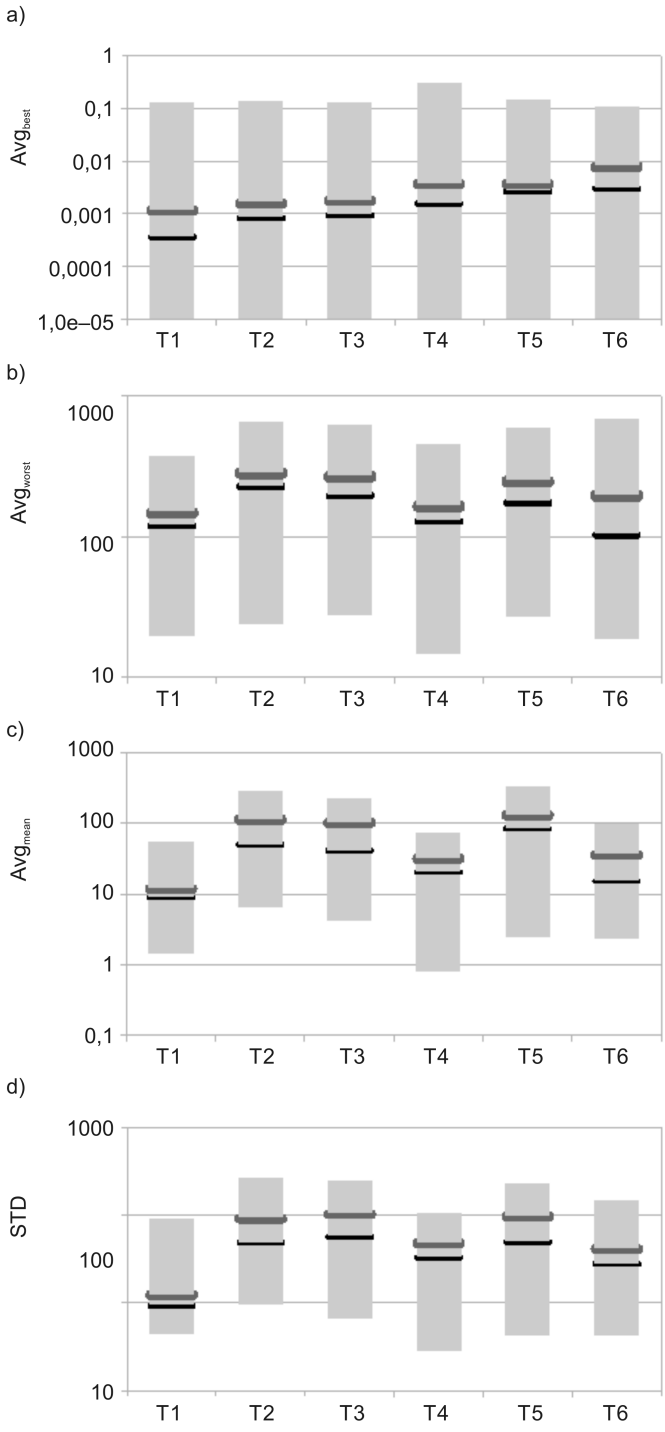
Rysunek 2.32. Rezultaty pracy algorytmów dla funkcji rotacji szczytów z pięćdziesięcioma ekstremami



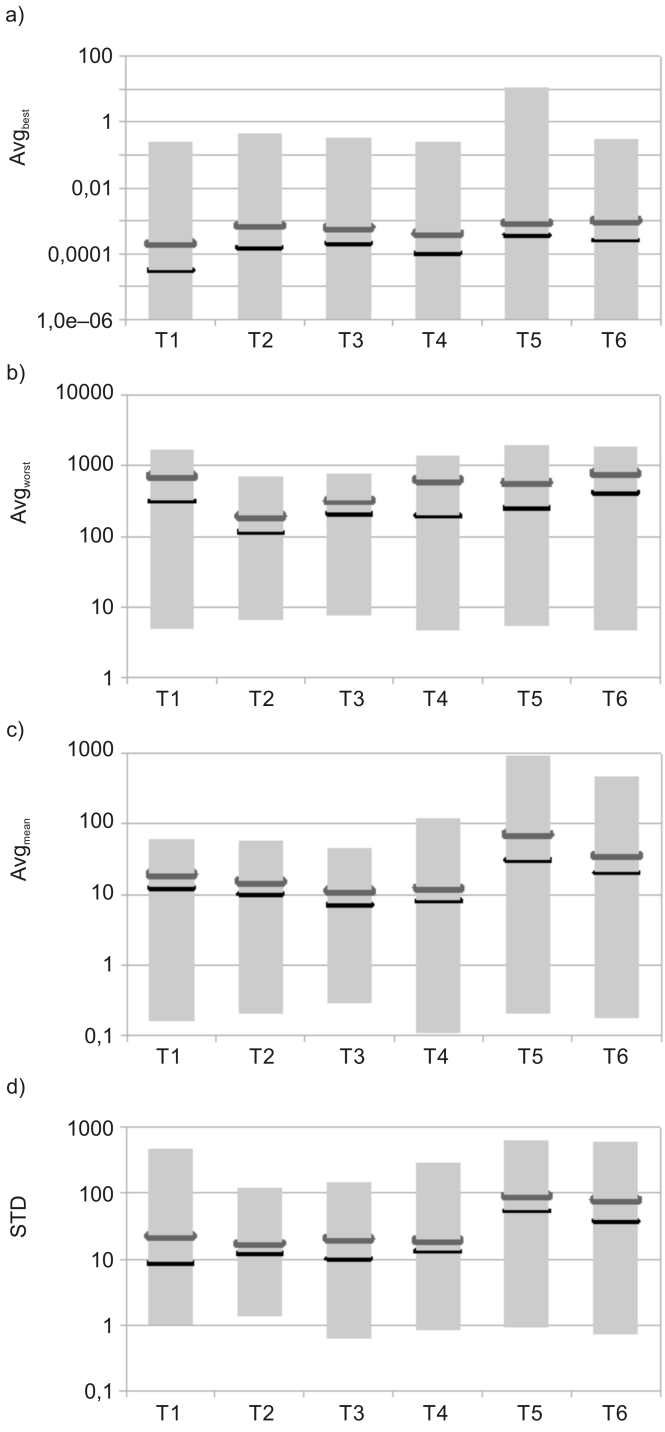
Rysunek 2.33. Rezultaty pracy algorytmów dla złożenia funkcji sfer z dziesięcioma ekstremami



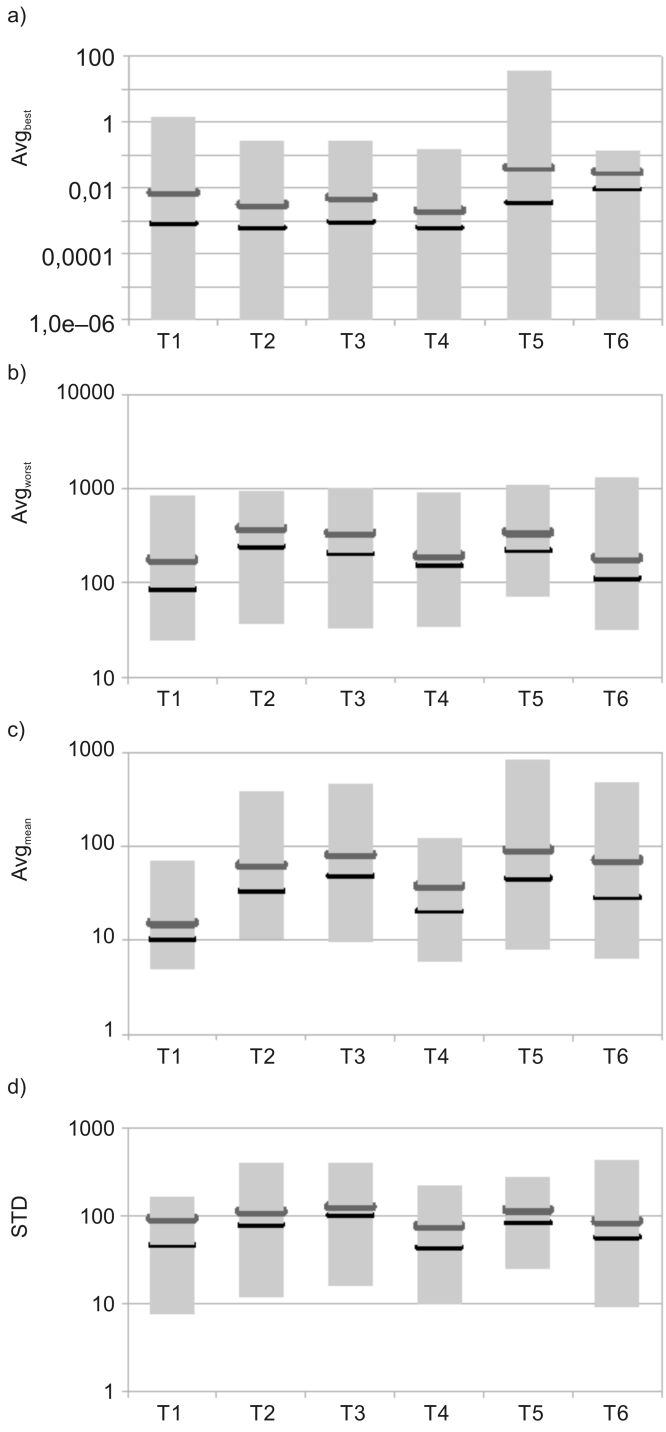
Rysunek 2.34. Rezultaty pracy algorytmów dla złożenia funkcji Rastrigina z dziesięcioma ekstremami



Rysunek 2.35. Rezultaty pracy algorytmów dla złożenia funkcji Griewanka z dziesięcioma ekstremami



Rysunek 2.36. Rezultaty pracy algorytmów dla złożenia funkcji Ackleya z dziesięcioma ekstremami



Rysunek 2.37. Rezultaty pracy algorytmów dla hybrydowego złożenia funkcji z dziesięcioma ekstremami

2.9. Uogólnione wnioski

Przedstawione w tej części pracy algorytmy mają wspólny algorytm bazowy, stanowiący nowe ujęcie zagadnienia. Różnią się one od siebie metodą opisu ruchu cząstek. Tylko ta różnica powoduje umieszczenie proponowanych algorytmów w dwóch grupach – jako algorytmy: immunologiczny i optymalizacji rojem cząstek. Są one nowymi rozwiązaniami w każdej z grup algorytmów. Jednocześnie charakteryzują się efektywną eksploracją przestrzeni rozwiązań i eksploatacją ekstremów lokalnych.

Algorytmy mają mechanizm samoregulacji eksploracji i eksploatacji przestrzeni rozwiązań. Ponadto, wykonują adaptację lokalnego obszaru przeszukiwania – efektem tego jest quasi-optymalny rozkład próbkowania tej przestrzeni.

Algorytmy te posiadają własność zbliżoną do mechanizmu obserwacji. Funkcja środowiska ma niewielki wpływ na ich działanie. Działanie algorytmów opisuje zachowania znane w systemach przyrodniczych mających wysoką efektywność. Konsekwencją implementacji była zmiana interpretacji funkcji elementów w stosunku do stosowanego do tej pory podejścia. Wadą algorytmów jest wielokrotne wykrywanie ekstremów lokalnych.

Efektywność pracy algorytmów jest wysoka, co potwierdza przedstawiona teoria i przeprowadzone badania symulacyjne. Niewielki koszt pracy algorytmu S-PSO daje szansę jego wykorzystania w optymalizacji online. Interesujące wydaje się też wykorzystanie tego algorytmu we współpracy z innymi rozwiązaniami, tworząc w ten sposób systemy hybrydowe.

Dzięki przedstawionym właściwościom algorytmy te mogą mieć istotne znaczenie praktyczne.

Rozdział 3

Rejestry ze sprzężeniem zwrotnym jako element architektury wbudowanego testowania

Rejestry ze sprzężeniem zwrotnym są powszechnie stosowane w strukturach BIST (*Built-In Self-Test*). Prace nad tego typu rozwiązaniami prowadzone są w wielu zespołach na świecie – wykaz ważniejszych grup badaczy europejskich i amerykańskich przedstawia tabela 3.1.

Tematykę diagnostyki podnoszą również pozycje książkowe, które mają charakter przeglądowy, ukazując aktualny stan wiedzy w tym zakresie – przykładem takiego opracowania jest [203].

Na szczególną uwagę zasługują tutaj prace A. Hławiczki oraz jego zespołu. Godne zainteresowania są również prace A. Kraśniewskiego oraz D. Badury, które wnoszą istotny wkład w badania nad wykorzystaniem rejestrów z nieliniowym sprzężeniem zwrotnym w strukturach BIST. Rozwiązania te charakteryzuje niewielki nadmiar układowy. Zastosowanie algorytmów genetycznych w realizacji zadań optymalizacji struktur diagnostycznych znajdujemy między innymi w pracach zespołu P. Prinetto, a także w pracach zespołu A. Hławiczki [175].

Zastosowanie rejestru nieliniowego może znacznie uprościć koncepcję testowania oraz zminimalizować nadmiar układowy, jednak stosowanie rejestrów z nieliniowym sprzężeniem zwrotnym ma znacznie szerszy charakter. Algorytmy syntezy rejestrów generujących ciągi o określonych właściwościach znajdują się w pracy [215]. Wspomniana rozprawa przedstawia możliwości ich stosowania jako rejestrów liczących. Dla rejestrów ze sprzężeniem zwrotnym można określić cechy, które są istotne w diagnostyce: graf rejestru ma najdłuższy cykl lub najdłuższą ścieżkę, graf rejestru jest cykliczny, graf rejestru ma cykl o zadanej długości, graf rejestru ma zadaną sekwencję stanów, graf rejestru ma zadaną sekwencję stanów w cyklu o podanej długości, graf rejestru ma stany w sekwencji rosnącej. Omówienie możliwości optymalizacji rejestrów o takich właściwościach przy wykorzystaniu algorytmów genetycznych zawarto w pracy [97]. Przeprowadzono również badania możliwości syntezy sprzężenia zwrotnego rejestru z wykorzystaniem programowania genetycznego [91]. W badaniach zastosowano bibliotekę GPKernel [119] z modyfikacjami właściwymi dla rozwiązywanych problemów. Dzięki stosowaniu gramatyk, możliwe jest użycie rejestrów ze sprzężeniami zwrotnymi liniowym i nieliniowym zwrotnym, w tym rejestrów hybrydowych. Te ostatnie mają elementy opisu charakterystyczne dla rejestrów ze sprzężeniami zwrotnymi

Tabela 3.1. Zespoły europejskie i amerykańskie

Zespół		Publikacje
A. Hławiczka	z Wydziału Automatyki, Elektroniki i Informatyki Politechniki Śląskiej w Gliwicach	[14], [15], [16], [172], [175]
A. Kraśniewski	z Politechniki Warszawskiej (badania koncentrują się na strukturach FPGA)	[140], [141], [142], [195], [186]
D. Badura	z Uniwersytetu Śląskiego w Katowicach	[12], [13], [15], [17], [18], [59], [86]
J. Tyszer	z Politechniki Poznańskiej	[189]
P. Prinetto	z Wydziału Automatyki i Informatyki Politechniki w Turynie	[58], [62], [63], [64]
H.J. Wunderlich	z Instytutu Informatyki Technicznej Uniwersytetu w Stuttgarcie	[105], [129], [130], [152]
S. Hellebrand	z Uniwersytetu Paderborn	[105], [106], [152]
P. Girard	z LIRMM Montpellier	[75]
D. Nikolos	z Wydziału Inżynierii Komputerowej i Informatyki na Uniwersytecie w Patrze	[23], [204]
ś.p. J. Hlavicka	z Czeskiego Uniwersytetu Technicznego w Pradze	[77], [78]
O. Novak	z Politechniki w Libercu	[172]
K. Chakarbarty	z Wydziału Elektrycznego i Inżynierii Komputerowej na Uniwersytecie Duke	[49], [149]
J. Rajski	z Korporacji Mentor Graphics	[106], [189]
S.M. Reddy	z Uniwersytetu Iowa	[227]
E. McCluskey	z Centrum Niezawodnego Przetwarzania Danych na Uniwersytecie Stanford	[210]

liniowym i nieliniowym. Ze względu na to, że koncepcje te ukierunkowane są na tworzenie autonomicznych kompaktorów i generatorów testów, nie zostały zamieszczone w niniejszej rozprawie. Omówienie koncepcji diagnostycznych, zweryfikowanych przy użyciu układów testowych ISCAS'85 ([43]) zawiera rozprawa [96]. Prace te zostały zakończone pełnym sukcesem i stanowiły bazę do rozwoju koncepcji testowania, zamieszczonych w niniejszej rozprawie. Przyczyniły się one również do zgłębienia wiedzy z zakresu algorytmów ewolucyjnych i ich stosowania w optymalizacji struktur diagnostycznych. Wiedza

ta okazała się nie bez znaczenia przy opracowaniu niniejszej rozprawy. Omówienie możliwości doboru elementów struktury diagnostycznej przedstawione zostało również w pracy [88].

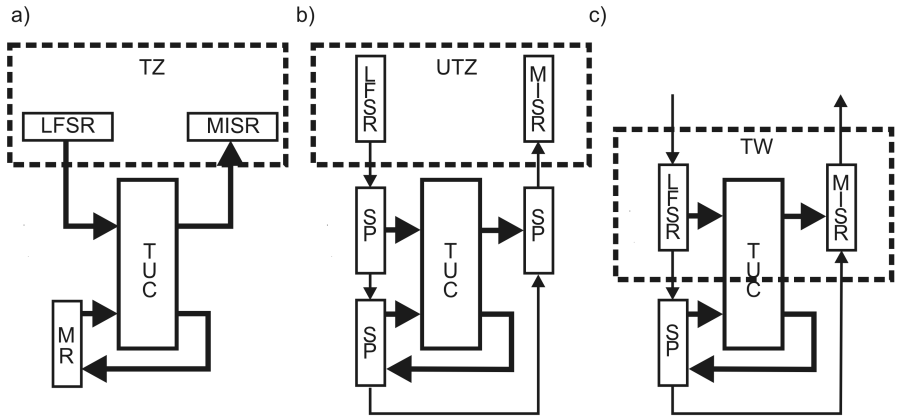
Aby efektywnie przeprowadzić proces badawczy, niezbędne jest narzędzie symulacyjne. Pierwsze konstrukcje symulatora, dające zadowalające czasy symulacji zostały opisane w pracach [89], [93]. Symulator, przedstawiony w rozdziale 4.3, jest następstwem tych prac.

Najobszerniej tematykę: analizy, syntezy oraz zastosowania rejestrów liniowych w testowaniu układów cyfrowych omawia monografia [111]. W dysertacji [18] znajduje się obszerne omówienie właściwości, a także możliwości wykorzystania rejestrów z nieliniowym sprzężeniem zwrotnym do budowy struktur diagnostycznych. Praca [83] podejmuje tematykę wykorzystania rejestrów z nieliniowym sprzężeniem zwrotnym w strukturach diagnostycznych. Natomiast takie struktury diagnostyczne, jak pierścień samotestujący [142] i ścieżka samotestująca [18] są charakterystyczną odmianą rejestru z nieliniowym sprzężeniem zwrotnym. Dobór struktury diagnostycznej stanowi tutaj niejako odrębne zagadnienie, gdyż ma charakter wielokryterialnego problemu optymalizacyjnego. Do tego typu optymalizacji stosowane są metody sztucznej inteligencji. W rozdziale tym zostanie przedstawiony opis rejestrów ze sprzężeniem zwrotnym oraz wynikające z niego możliwości tworzenia genomu dla metod ewolucyjnych.

3.1. Architektura wbudowanego testowania

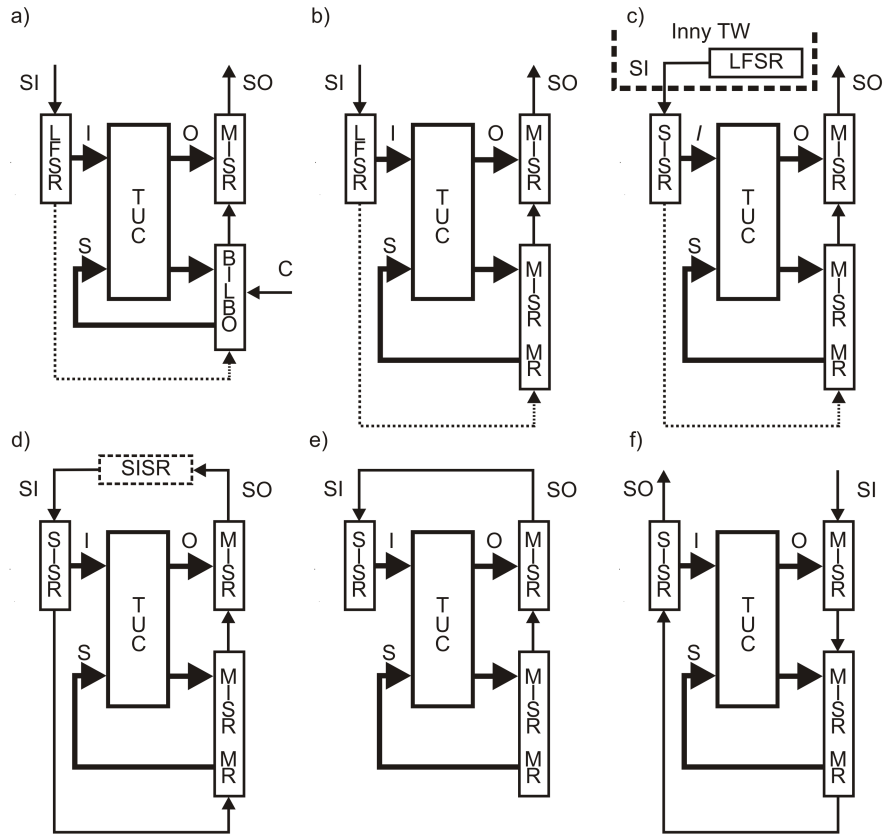
W większości przypadków koncepcja wbudowanego testowania przekształca testowany układ sekwencyjny w testowany układ kombinacyjny połączony ze strukturą testera, pełniącą funkcję generatora testów i kompaktora. Techniki DFT – projektowania ułatwiającego testowanie (*Design For Testability*) określają drogę rozwoju technologii produkcji układów scalonych oraz kierunki poszukiwań nowych rozwiązań. Standard IEEE 1149.1 determinuje obecnie strategię testowania układów cyfrowych [33], [120], [177]. Klasyfikacja struktur testowania, a w szczególności struktur wbudowanego testowania wydaje się być niezwykle trudna. Jest to spowodowane pozornie niewielkimi zmianami, tworzącymi nową architekturę testowania układu. Dlatego też w tym rozdziale omówiono ewolucję struktury diagnostycznej bazującą na klasyfikacji prezentowanej w monografii [111]. Przedstawiony w ten sposób zakres zmian architektur testowania pozwoli lepiej dostrzec uniwersalność, proponowanego w dalszej części pracy, rozwiązania. Wyznaczenie testów dla układów cyfrowych jest bardzo czasochłonne i opiera się na założonym modelu uszkodzeń testowanego układu. Metoda pseudolosowego generowania testów jest bardziej naturalna i umożliwia zwiększenie współczynnika pokrycia uszkodzeń związanych z dynamiką (np. opóźnienia, hazardy) oraz uszkodzeń, które są niezależne od przyjętego modelu.

Rysunek 3.1a pokazuje przykład konwencjonalnego testera zewnętrznego. Tester połączony jest z testowanym układem magistralą wieloprzewodową. Podczas testowania wystąpienie zróżnicowanych stanów S w bloku pamięci MR (*Memory Register*) zależy od funkcji testowanego układu i sekwencji testów I. Konsekwencją tego może być bardzo niska skuteczność testowania [111]. Sekwencja testów pseudolosowych nie zawsze jest właściwa do pobudzenia uszkodzenia i propagacji wytworzonego błędu na wyjście O. Aby usunąć te wady, w miejsce rejestru MR wprowadzono rejestr SP (*Scan Path*). Powoduje to przekształcenie testowanego układu sekwencyjnego w układ kombinacyjny – rysunek 3.1b [2], [109], [111], [112]. Wprowadzenie rejestrów SP do wejść i wyjść testowanego układu pozwala na szeregowe łączenie testera z testowanym układem. Rejestry ścieżki brzegowej w czasie normalnej pracy układu są przezroczyste. Następnym krokiem jest przeniesienie testera do wnętrza układu. Rejestr SP podłączony do wejść układu zostaje zastąpiony rejestrem LFSR (*Linear Feedback Shift Register*) – pełniącym funkcję generatora testów, a rejestr SP podłączony do wyjść układu rejestrem MISR (*Multi Input Signature Register*) – pełniącym funkcję kompaktora (rysunek 3.1c).



Rysunek 3.1. Testowanie układu sekwencyjnego przy użyciu: a) testera zewnętrznego TZ, b) rejestru SP oraz uproszczonego testera zewnętrznego UTZ, c) testera wewnętrznego TW

Rysunek 3.1 ilustruje więc ewolucję testerów i od tej chwili omawiane będą techniki wbudowanego testowania, których przykładowe architektury prezentowane są na rysunku 3.2. Przedstawiona na rysunku 3.2a struktura BIST-RTD (*BIST-Random-Test-Data*) powstała w wyniku zamiany rejestru *SP* układu z rysunku 3.1c na wielofunkcyjny rejestr BILBO (*Built-In Logic Block Observer*) [2], [112]. Rejestr BILBO w czasie normalnej pracy pełni funkcję rejestru MR, a w czasie testowania rejestru MISR. W pracy [74] zastąpiono rejestr BILBO rejestrem MISR MR (*MISR based Memory Register*), otrzymując strukturę BIST-PST (*BIST-Parallel-Self-Test*) – rysunek

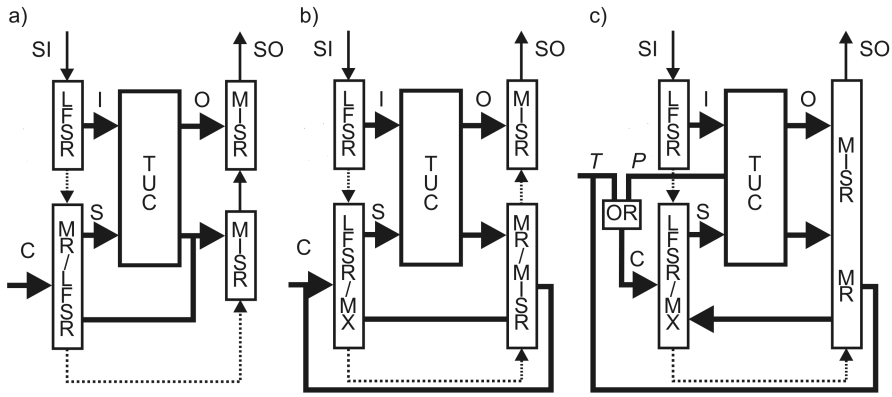


Rysunek 3.2. Testery wewnątrzukładowe: a) BIST-RTD, b) BIST-PST, c) architektura z rejestrem zakłócającym, d) architektura bazująca na CSTP, e, f) struktury bazująca na STP

3.2b. Linią przerywaną zaznaczono możliwe modyfikacje struktury testowania [112]. Należy tu zwrócić uwagę na znaczenie tych modyfikacji. Połączenie rejestru LFSR z blokiem MISR MR zwiększa efektywność generowania testów, można to również osiągnąć przez zakłócanie pracy rejestru. Natomiast połączenie rejestru MISR MR z kompaktorem MISR eliminuje maskowanie błędów w rejestrze MISR MR. Rejestr zakłócający nie musi być częścią testera wewnątrzukładowego – rysunek 3.2c. Wszystkie rejestry można połączyć w pierścień, podobnie jak w strukturze CSTP (*Circular Self Test Path*), opisywanej w pracach [140], [141], [142], [195], [186], oraz architekturze SEREG (*SErial REGister*) [46], której modyfikację z udziałem rejestru SISR (*Single-Input Signature Register*), zaproponowaną w [133], przedstawia rysunek 3.2d. Bazową strukturą modyfikacji struktur testowania z rysunków 3.2e i 3.2f jest struktura STP (*Self Test Path*), opisywana w pracach [12], [13], [17], [18]. Generowanie testów dla wejść pierwotnych układu w architekturach z rysunków 3.2a, 3.2b

i 3.2c nie zależy od funkcji testowanego układu. W pozostałych przypadkach – 3.2d, 3.2e, 3.2f – skuteczność testowania związana zarówno z generowaniem testów, jak i kompakcją odpowiedzi testowej zależy bardzo od funkcji testowanego układu.

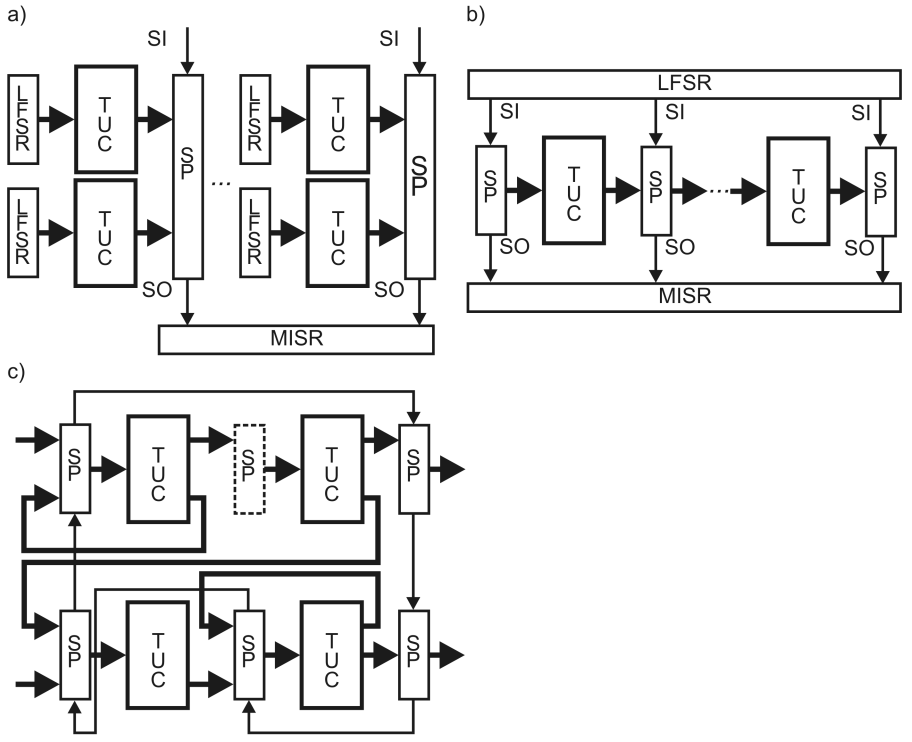
Celem uzyskania grafu przejść z pierścieniem o cyklu gwarantującym maksymalną skuteczność testowania w miejsce bloku pamięci MR wprowadzono rejestr CBILBO (*Concurrent BILBO*) z podwójną liczbą przerzutników. Umożliwia to niezależne budowania generatora testów i kompaktora odpowiedzi testowej na magistrali stanu. Na schemacie z rysunku 3.3a rejestr MR pracuje jako generator testów, a na rysunku 3.3b jako kompaktor odpowiedzi testowej.



Rysunek 3.3. Testery wewnątrzukładowe z rejestrem CBILBO: a) BIST-DFF – rejestr MR pracuje jako generator testów, b) BIST-DFF – rejestr MR pracuje jako kompaktor MISR, c) przykład struktury BIST-PAT

Lepszą skuteczność testowania tych struktur można osiągnąć łącząc parami rejestry generatora testów (LFSR i LFSR MR) oraz rejestry kompaktorów, uzyskując zamiast krótkich – długie rejestry [111]. W pracach [73], [74] rejestr LFSR (włączony w magistralę stanu) wykorzystano do generowania stanów w czasie normalnej pracy – rysunek 3.3c pokazuje architekturę BIST-PAT (*PATtern*).

Złożonym problemem staje się stosowanie technik wbudowanego testowania dla układów wielomodułowych. Omówienie tego problemu zawiera między innymi praca [230]. W strukturach wielomodułowych należy rozwiązać problem jednoczesnego testowania wielu modułów. Architekturę struktury TPS (*Test-Per-Scan*) z kilkoma generatorami testu przedstawiono na rysunku 3.4a [184]. Natomiast na rysunku 3.4b pokazano strukturę testera BIST-STUMPS (*Self-Test Using MISR and Parallel SRSG*) [2], [19], [20], [111]. Istotnym problemem budowy wewnętrznych ścieżek jest właściwy dobór tworzących je elementów. Propozycję takiego rozwiązania przedstawiono w artykule [136]. Do budowy wewnętrznych ścieżek samotestujących wykorzystano algorytm



Rysunek 3.4. Struktury testowania układów wielomodułowych: a) tester wewnątrzukładowy typu test-per-scan, b) BIST-STUMPS, c) CSTP

genetyczny. Na rysunku 3.4c przedstawiono strukturę CSTP, w której nie wszystkie rejestry zostały połączone w pierścień [142]. W technice pierścienia samotestującego CSTP do generacji testów i kompakcji odpowiedzi wykorzystuje się rejestry szeregowe, w tym rejestry należące do testowanego układu, które zostają połączone w pierścień [142], [195]. Rejestry te nie mają dodatkowych sprzężeń zwrotnych. W tym przypadku sprzężenie zwrotne realizowane jest przez badany układ. Technika ta charakteryzuje się małym nadmiarem układowym, potrzebnym do jej realizacji. W artykule [140] wykazano, że CSTP jako analizator sygnatur jest co najmniej tak samo skuteczny, jak standardowy. Czas rozruchu pierścienia samotestującego jest bardzo krótki. Podczas generowania testów właściwości pierścienia samotestującego nie są optymalne, ponieważ występuje cykliczne powtarzanie stanów [13]. Struktura pierścienia samotestującego była omawiana również w pracy [63]. Zastosowanie CA (*Cellular Automata*) do modyfikacji pierścienia samotestującego zaprezentowano w pracach [63], [64].

W analizie struktury diagnostycznej można brać pod uwagę graf przejść autonomicznego układu sekwencyjnego AFSM (*Autonomous Finite State Ma-*

chine), którego blok pamięci tworzą rejestry podłączone do wejść testowanego układu [12], [13], [17], [18], [46], [109]. Graf przejść zawierający cykl ułatwia uzyskanie maksymalnej skuteczności testowania. W przypadku grafu o dużo mniejszej liczbie węzłów uzyskanie właściwej sekwencji testującej może okazać się niemożliwe, natomiast w przypadku grafów o cyklach większych występuje powtarzanie się stanów, co może powodować konieczność wydłużenia testu. Nie ma uniwersalnej metody projektowania, pozwalającej otrzymać AFSM o grafie przejść z pierścieniem optymalnym, a proponowane algorytmy są efektywne dla układów zawierających maksymalnie 30–40 komórek.

Przedstawione powyżej architektury nie wyczerpują wszystkich możliwości ich modyfikacji. Jednoznacznie można stwierdzić, że żadna ze struktur wbudowanego testowania nie ma uniwersalnego zastosowania, przy optymalnych właściwościach diagnostycznych. Jak nietrudno zauważyć, budowane są one z typowych bloków o określonych funkcjach, których konfiguracja musi być dobrana w zależności od funkcji TUC (Testowany Układ Cyfrowy – *Circuit Under Test*). Sposób połączeń między nimi określa właściwości struktury. Takie podejście, będące niejako kanonem BIST, poważnie ogranicza możliwości tworzenia architektur testerów wbudowanych. Wydaje się, że jedyne ograniczenia dla takich architektur wprowadza standard IEEE 1149.1.

Pytanie „Jak zaprojektować tester, aby uzyskać maksymalną skuteczność diagnostyczną, przy czasie testowania zbliżonym do testu deterministycznego?" pozostaje ciągle bez odpowiedzi. Niestandardowe podejście do tego problemu zaproponowano w niniejszej rozprawie i zweryfikowano dzięki zastosowaniu algorytmu genetycznego.

3.2. Rejestr ze sprzężeniem zwrotnym

Do opisu działania układów sekwencyjnych używany jest graf przejść układu. Jest on szczególnie wygodny w opisie zachowania rejestrów ze sprzężeniem zwrotnym. Obszerne omówienia tych zagadnień odnajdujemy w [18], [81] oraz [111], a ich elementy przytoczono poniżej.

Definicja 3.1 (cykl w grafie rejestru)**.** *Cyklem w grafie rejestru nazywamy uporządkowany zbiór wszystkich stanów, przez które rejestr przechodzi tak, że stanem następującym po ostatnim stanie cyklu jest stan początkowy.*

Definicja 3.2 (długość cyklu)**.** *Długością cyklu oznaczamy liczbę wszystkich stanów w cyklu rejestru.*

Definicja 3.3 (w pełni cykliczny graf rejestru)**.** *Graf przejść rejestru ze sprzężeniem zwrotnym jest w pełni cykliczny, jeżeli do każdego węzła grafu wchodzi jedna i tylko jedna gałąź.*

Definicja 3.4 (nie w pełni cykliczny graf rejestru)**.** *Graf przejść rejestru jest nie w pełni cykliczny, gdy do co najmniej jednego węzła grafu wchodzą przynajmniej dwie gałęzie.*

Wynika z tego, że w grafie będą istniały drzewa węzłów grafu połączone z węzłami odpowiadającymi stanom należącym do cykli. Rejestry z nieliniowym sprzężeniem zwrotnym mają (przeważnie) nie w pełni cykliczny graf przejść.

Definicja 3.5 (maksymalny cykl w grafie rejestru)**.** *Dla n – bitowego rejestru z nieliniowym sprzężeniem zwrotnym długość maksymalnego cyklu wynosi* 2^n*, natomiast maksymalny cykl rejestru z liniowym sprzężeniem zwrotnym wynosi* $2^n - 1$.

Definicja 3.6 (równoważna postać grafów)**.** *Dwa grafy są równoważne, jeżeli w wyniku wzajemnie jednoznacznego odwzorowania węzłów i gałęzi grafów drogi pomiędzy dowolnymi dwoma, odpowiadającymi sobie węzłami grafów są takie same.*

W sprzężeniach rejestrów można wyróżnić funkcje liniową i nieliniową. Rejestry z liniowym sprzężeniem zwrotnym nazywane są rejestrami liniowymi, a z nieliniowym sprzężeniem zwrotnym – rejestrami nieliniowymi. Rejestr hybrydowy jest rejestrem przesuwającym, mającym funkcje liniową i nieliniową sprzężenia zwrotnego i przynależy do grupy rejestrów nieliniowych. Rejestr taki można więc opisać wzorem:

$$\begin{aligned} X^T(t+1) = \; & N_I^T \oplus \left(T_{OP}\left(N_O^T \oplus X^T(t)\right) \oplus U^T(t) \oplus \right. \\ & \oplus O_0 F_0^T\left(I_0\left(N_O^T \oplus X^T(t)\right)\right) \oplus \\ & \oplus O_1 F_1^T\left(I_1\left(N_O^T \oplus X^T(t)\right)\right) \oplus \\ & \left. ... \oplus O_{m-1} F_{m-1}^T\left(I_{m-1}\left(N_O^T \oplus X^T(t)\right)\right)\right), \end{aligned} \tag{3.1}$$

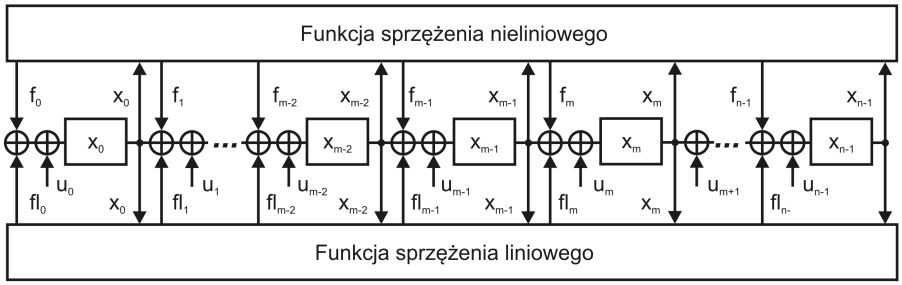
gdzie:

$$T_{OP} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-2} & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-2} & a_{1,n-1} \\ a_{2,0} & a_{2,1} & \cdots & a_{2,n-2} & a_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-2} & a_{n-1,n-1} \end{bmatrix}, \tag{3.2}$$

n – długość rejestru, $a_{i,j}$ – przyjmuje wartość 1, jeżeli realizowane jest połączenie pomiędzy j-tą a i-tą komórką rejestru.

Na rysunku 3.5 przedstawiono schemat rejestru hybrydowego.

Na podstawie opisu macierzy T_{OP} dokonuje się klasyfikacji liniowych sprzężeń zwrotnych [94]. Wejścia i wyjścia przerzutników rejestru mogą być



Rysunek 3.5. Rejestr hybrydowy

negowane. Takie rejestry mają również równoważne grafy przejść. Elementy wektorów negacji N_I^T i N_O^T przyjmują wartość 1, w przypadku realizacji operacji negacji (indeks O oznacza wyjście, a I wejście elementu rejestru). Zmieniając wartości wektorów negacji uzyskujemy rejestry o równoważnych grafach przejść. Sposób podłączenia nieliniowej funkcji F^T sprzężenia zwrotnego rejestru przesuwającego można określić przez macierze mapowania wejść oraz wyjść. W macierzy mapowania indeks kolumny odpowiada indeksowi wektora wejściowego (macierz I) lub wyjściowego (macierz O) funkcji, a indeks wiersza odpowiada komórce rejestru, do którego jest przypisana. Każda funkcja musi mieć swoją parę macierzy mapowania. Funkcji F^T nie można zredukować do funkcji realizowanej funktorami XOR, czyli przedstawić za pomocą macierzy połączeń. Graf przejść rejestru szeregowego z nieliniowym sprzężeniem zwrotnym będzie cykliczny, jeżeli funkcja nieliniowa nie zależy od zmiennej x_{n-1} [18].

Elementy struktur diagnostycznych budowane są przeważnie na bazie rejestrów z liniowym sprzężeniem zwrotnym. Elementy te połączone są ze strukturą testowanego układu, którego funkcja ma charakter kombinacyjny. Całość może więc być opisana jako rejestr hybrydowy.

Definicja 3.7 (rejestr autonomiczny)**.** *Rejestrem autonomicznym ze sprzężeniem zwrotnym nazywamy rejestr z zerowym wektorem wejściowym.*

Wzór (3.1) przyjmie więc postać:

$$\begin{aligned} X^T(t+1) = \quad & N_I^T \oplus \left(T_{OP}\left(N_O^T \oplus X^T(t)\right) \oplus \right. \\ & \oplus O_0 F_0^T \left(I_0\left(N_O^T \oplus X^T(t)\right)\right) \oplus \\ & \oplus O_1 F_1^T \left(I_1\left(N_O^T \oplus X^T(t)\right)\right) \oplus \\ & \left. \ldots \oplus O_{m-1} F_{m-1}^T \left(I_{m-1}\left(N_O^T \oplus X^T(t)\right)\right)\right). \end{aligned} \tag{3.3}$$

Zależność ta pozwala opisać dowolną strukturę diagnostyczną dla układów jedno- i wielomodułowego.

3.3. Przestrzeń rozwiązań

Analizując wyrażenie (3.3) możemy określić potencjalną przestrzeń rozwiązań, w której znajduje się optymalna struktura diagnostyczna.

$X(t)$ to wektor o rozmiarze n, określający stan początkowy elementów pamięci testowanych układów, a także elementów struktury diagnostycznej. Ustawiane mogą być bity rejestru ścieżki sterująco-obserwującej oraz włączone do niej elementy pamięci układu. Pozostałe elementy pamięci mogą mieć ściśle określony stan, wynikający z założeń projektowych dla stanu początkowego.

N_I, N_O to wektory negacji wejść/wyjść o rozmiarze n. Ze względu na istniejący już nadmiar układowy komórek należących do ścieżki sterująco-obserwującej, realizacja negacji nie będzie kosztowna – dla pozostałych elementów pamięci można założyć jej brak (chociaż realizacja sprzętowa wprowadzenia negacji w trakcie testowania jest możliwa przy niewielkim nadmiarze układowym – jedna bramka XOR).

I_k, O_k to macierze mapowania o n wierszach i liczbie kolumn odpowiadającej: in_k – liczbie wejść lub out_k – liczbie wyjść testowanego bloku kombinacyjnego. Każda funkcja bloku musi mieć własną parę macierzy mapowania. Metoda podłączenia bloku w układzie jest ściśle określona i ograniczona przez projekt, jednak w ten sposób można modelować zmianę kolejności komórek w rejestrze ścieżki brzegowej.

T to macierz połączeń, jej rozmiar to $n \times n$. Definiuje opis połączeń struktury diagnostycznej, w tym liniowe sprzężenia zwrotne rejestrów MISR, LFSR i SR oraz możliwe połączenia między nimi.

Zakładając wszystkie możliwe zmiany, otrzymamy następującą liczbę modyfikowanych elementów:

$$\begin{aligned} X(t) &\rightarrow n, \\ N_I &\rightarrow n, \\ N_O &\rightarrow n, \\ I_k &\rightarrow \binom{n}{in_k}, \\ O_k &\rightarrow \binom{n}{out_k}, \\ T &\rightarrow n^2. \end{aligned}$$

Łączną liczbę modyfikowanych elementów dla m modułów wyraża zależność:

$$3n + \sum_{k=0}^{m-1} \binom{n}{in_k} + \sum_{k=0}^{m-1} \binom{n}{out_k} + n^2. \tag{3.4}$$

Dla struktury pierścienia samotestującego liczba modyfikowanych elementów macierzy połączeń wynosi $l \leqslant n$, natomiast dla struktury ścieżki samotestującej $l \leqslant n-1$. Dla dowolnej struktury testowania, przy założeniu, że

dla każdej komórki rejestru występują nie więcej niż dwa połączenia (jedno połączenie z następną komórką, drugie jako liniowe sprzężenie zwrotne), liczba modyfikowanych elementów macierzy połączeń wynosi $l \leqslant 2n$.

Zakładając, że mapowanie wejść i wyjść dla układu jest niedopuszczalne, a liczba elementów rejestru wynosi 10, to liczba możliwych rozwiązań, w przypadku dowolnej struktury diagnostycznej wynosi maksymalnie:

$$2^{5n} = 2^{50}.$$

Ten prosty przykład ilustruje ogrom złożoności problemu doboru optymalnej struktury diagnostycznej. Ze względu na ograniczenia projektowe, liczba dobieranych elementów byłaby znacznie mniejsza, jednak dla większych układów i dla układów wielomodułowych analizowana przestrzeń rozwiązań będzie ogromna. Wobec powyższego w optymalizacji struktury diagnostycznej jedynie stosowanie algorytmów ewolucyjnych może przynieść zadowalające rezultaty.

3.4. Ocena skuteczności testowania

Rejestry szeregowe ze sprzężeniem zwrotnym są wykorzystywane jako generatory testów i kompaktory. Generatory testów powinny cechować się brakiem powtórzeń w generowanym teście. Zbiór stanów rejestru powinien zapewniać pobudzenie wszystkich uszkodzeń, mogących wystąpić w układzie. Taki sposób generowania testów dotyczy układów kombinacyjnych. Wprowadzenie elementów struktury diagnostycznej do układu sekwencyjnego może powodować jego przekształcenie do układu kombinacyjnego. Skutkuje to uproszczeniem koncepcji testowania. Poniżej omówione zostaną wybrane pojęcia związane z efektywnością generowania testów.

Definicja 3.8 (pokrycie stanów)**.** *Pokrycie stanów definiowane jest jako stosunek liczby różnych pobudzeń testowych do liczby pobudzeń testu wyczerpującego.*

Analogiczną miarą do pokrycia stanów może być entropia stanów rejestru. Jeżeli X będzie reprezentować stany rejestru, to entropię można zdefiniować w następujący sposób:

Definicja 3.9 (entropia)**.** *Niech n-bitowy sygnał X w chwili t przyjmuje wartość ze zbioru $\{0, 1, ..., 2^n - 1\}$ z prawdopodobieństwami:*
$\mathbb{X}(t) = [P(X(t) = 0), P(X(t) = 1), ..., P(X(t) = 2^n - 1)]^T$.

Entropia sygnału X w chwili t jest definiowana jako [11], [197]:

$$H(X(t)) = -\sum_{i=0}^{2^n-1} P(X(t) = i) \cdot \log_2 P(X(t) = i). \tag{3.5}$$

Definicja 3.10 (entropia stanów rejestru – entropia rejestru)**.** *Entropia H_r jest miarą zróżnicowania stanów rejestru i jest funkcją prawdopodobieństwa p_i wystąpienia wszystkich stanów rejestru. Entropię n-bitowego generatora testu można przedstawić wzorem:*

$$H_r = -\sum_{i=0}^{2^n-1} p_i \cdot \log_2 p_i, \tag{3.6}$$

gdzie p_i – prawdopodobieństwo wystąpienia i-tego stanu n-bitowego rejestru.

Entropia stanów rejestru osiągnie wartość maksymalną, gdy nie wystąpi powtórzenie stanu rejestru, to znaczy, że rejestr będzie miał graf przejść z cyklem o maksymalnej długości.

Rejestry szeregowe ze sprzężeniem zwrotnym wykorzystywane są również podczas testowania do kompakcji odpowiedzi. Podczas testowania w kompaktorach zachodzi zjawisko maskowania błędów (*Error Aliasing*). Zjawisko to polega na tym, że w trakcie testowania uszkodzonego układu na wyjściu kompaktora pojawia się taki sam stan, jak dla sprawnego układu. Stan kompaktora można przedstawić w postaci liczby zwanej sygnaturą. Poniżej zaprezentowane zostaną podstawowe miary efektywności kompakcji.

Definicja 3.11 (prawdopodobieństwo wykrycia uszkodzenia)**.** *Prawdopodobieństwo wykrycia uszkodzenia jest zdefiniowane jako prawdopodobieństwo otrzymania sygnatury błędnej w przypadku uszkodzenia układu.*

Definicja 3.12 (entropii sygnatur)**.** *Entropia jest miarą ilości informacji zawartej w kompaktorze i jest funkcją prawdopodobieństwa wystąpienia wszystkich sygnatur [11], [133], [197]. Entropię n-bitowego kompaktora można przedstawić wzorem:*

$$H_S = -\sum_{i=0}^{2^n-1} p_i \cdot \log_2 p_i, \tag{3.7}$$

gdzie p_i – prawdopodobieństwo otrzymania i-tej sygnatury.

Wartość entropii sygnatur wyznaczamy symulacyjnie, podobnie jak prawdopodobieństwo wykrycia uszkodzenia. W artykule [3] zaobserwowano, że wartość prawdopodobieństwa wykrycia uszkodzenia jest maksymalna, gdy wartość entropii na wyjściu układu jest maksymalna. Z wniosków zawartych w rozprawie [133] wynika, że przyjęcie przez entropię sygnatur wartości maksymalnej jest warunkiem wystarczającym do tego, aby wartość prawdopodobieństwa maskowania wyniosła 2^{-n}. Entropia kompaktora jest miarą zarówno generowania testów, jak i kompakcji odpowiedzi. Osiąganie przez entropię niskich wartości może być skutkiem niepobudzenia uszkodzeń lub zamaskowania błędów w kompaktorze. Również dla kompaktorów zbudowanych na bazie rejestrów LFSR i NLFSR wykazano na podstawie analizy procesów Markowa,

że prawdopodobieństwo maskowania osiąga wartość 2^{-n} w przypadku, gdy rejestr ma graf przejść z cyklem o maksymalnej długości [18], [72], [110], [220].

W celu uzyskania wysokiej efektywności testowania dla struktury pierścienia testującego czy też ścieżki samotestującej wymagane jest uzyskanie wysokiej wartości entropii w trakcie testowania. Jeżeli otrzymanie wysokiej entropii części rejestru podłączonej do wejść układu testowanego (zbiór stanów wejściowych stanowi test) będzie skutkowało uzyskiwaniem wysokiej entropii wyjść układu, to należy wnioskować, że efektywność diagnostyczna jest wysoka.

3.5. Optymalizacja struktury rejestru do celów diagnostycznych

Zastosowanie rejestru nieliniowego może znacznie uprościć koncepcję testowania oraz zminimalizować nadmiar układowy. W diagnostyce istotne są rejestry, które mają następujące cechy:

1) rejestr ma graf o najdłuższym cyklu,
2) rejestr ma graf z cyklem o zadanej długości,
3) rejestr posiada graf ze ścieżką o zadanej długości,
4) rejestr ma graf ze ścieżką, której węzły mają zadaną sekwencję stanów.

Pierwszy z rejestrów będzie miał dobre właściwości jako kompaktor odpowiedzi testowej. Pozostałe rejestry mogą pracować jako generatory testów, jednak najlepiej do tego celu będzie dostosowany ostatni rejestr.

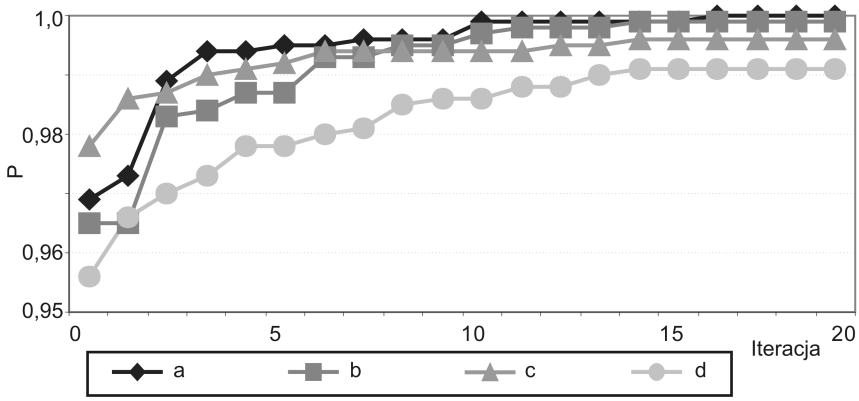
Poniżej przedstawiono możliwości optymalizacji rejestrów z nieliniowym sprzężeniem zwrotnym, celem uzyskania podanych wyżej właściwości. Opis genomu rejestru reprezentuje równanie 3.3. Jego częścią może być funkcja sprzężenia nieliniowego. Jest ona opisana przez tablicę prawdy. W optymalizacji architektury rejestru wykorzystano algorytm S-IMM. W przeciwieństwie do analizowanych do tej pory przypadków, genom opisujący rejestr ma charakter binarny. Podobieństwo genomów przeciwciał i antygenów określane więc będzie jako odległość Hamminga.

Analizowane są następujące architektury rejestru:

1) rejestr hybrydowy z optymalizowaną funkcją sprzężenia nieliniowego,
2) rejestr nieliniowy z optymalizowaną funkcją sprzężenia nieliniowego,
3) rejestr hybrydowy ze stałą funkcją sprzężenia nieliniowego,
4) rejestr nieliniowy ze stałą funkcją sprzężenia nieliniowego.

Pierwsza z przedstawionych architektur rejestru ma największy zakres zmian w procesie optymalizacji. Możliwe są zmiany funkcji nieliniowej sprzężenia zwrotnego, połączenia tej funkcji z rejestrem, połączeń liniowych rejestru (funkcji liniowej sprzężenia zwrotnego), negacji wejść i wyjść komórek rejestru. Natomiast ostatnia architektura ma najmniejsze możliwości doboru w procesie optymalizacji. Jak ilustrują to przedstawione poniżej wyniki przeprowadzonych badań, ma to istotny wpływ na proces optymalizacji.

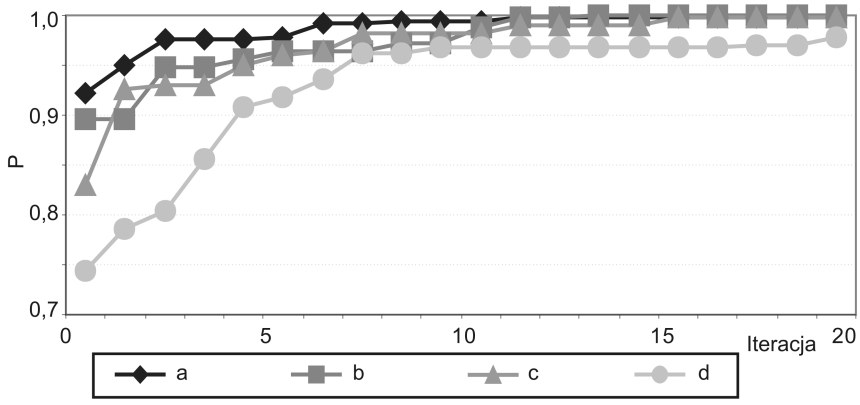
W pierwszym przypadku przyjętym kryterium końcowym jest znalezienie najdłuższej możliwej ścieżki, czyli maksymalnego cyklu w grafie przejść. Rysunek 3.6 ilustruje średnią wartość dostosowania P, opisującą możliwość uzyskania rozwiązania dla kolejnych iteracji algorytmu. Na rysunkach 3.6 – 3.9 oznaczenia literowe odpowiadają omówionym powyżej architekturom rejestrów.



Rysunek 3.6. Średnia wartość funkcji dostosowania P dla kolejnych iteracji algorytmu

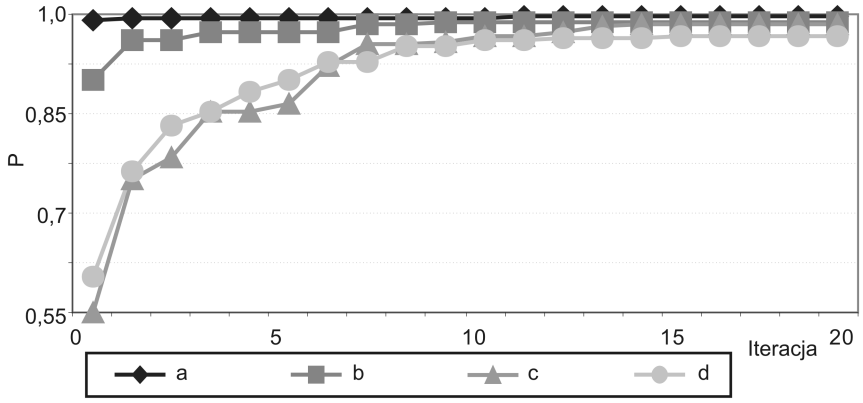
W widoczny sposób zauważalny jest wpływ architektury rejestru na otrzymane rezultaty. Najlepsze wyniki uzyskuje tutaj rejestr hybrydowy.

W drugim przypadku (rysunek 3.7) poszukiwana jest funkcja sprzężenia zwrotnego dla rejestru, realizująca cykl o zadanej długości.



Rysunek 3.7. Średnia wartość funkcji dostosowania P dla kolejnych iteracji algorytmu

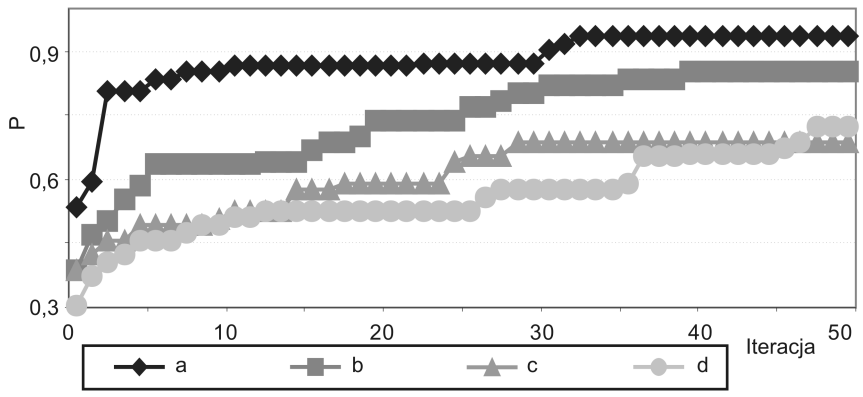
Podobnym kryterium jest poszukiwanie funkcji sprzężenia zwrotnego dla rejestru, realizującej ścieżkę o zadanej długości – co obrazują wykresy na rysunku 3.8. W tym przypadku graf przejść nie jest cykliczny.



Rysunek 3.8. Średnia wartość funkcji dostosowania P dla kolejnych iteracji algorytmu

Porównując charakterystyki z rysunku 3.8 wykazano, że możliwość optymalizacji funkcji sprzężenia zwrotnego daje lepsze rezultaty w przypadku doboru funkcji sprzężenia zwrotnego. Tworzą się więc dwie podobne pod względem skuteczności grupy rozwiązań.

W ścieżce lub cyklu grafu przejść istnieje zadana sekwencja stanów rejestru. Realizowana optymalizacja polega na znalezieniu rejestru, w którym kolejne węzły grafu przejść rejestru będą miały zadaną sekwencję stanów.



Rysunek 3.9. Średnia wartość funkcji dostosowania P dla kolejnych iteracji algorytmu

Kolejne wartości grafu przejść stanów mogą być bardzo zróżnicowane. Jak ilustrują to wykresy z rysunku 3.9, uzyskanie zadanej sekwencji w grafie nie dało pełnego sukcesu. Z zadaniem najlepiej radził sobie rejestr hybrydowy.

W tym miejscu należy podać kilka uwag o działaniu algorytmu optymalizującego architekturę rejestru. Antygen aktywizował średnio cztery przeciwciała, z których każde produkowało przeciętnie około pięciu klonów. W przeliczeniu na jeden cykl pracy algorytmu, jedno przeciwciało było redukowane i zastępowane nowym, jako najbardziej podobne. Co trzy, cztery cykle był redukowany antygen. Mutacja klonów realizowała przeszukiwanie przestrzeni rozwiązań ograniczonej cechami aktywnych przeciwciał. Takie działanie ma charakter lokalny. Ze względu na wytwarzaną liczbę klonów, należy oszacować koszt algorytmu na podobny do algorytmu genetycznego, który ma 20 osobników w populacji. Zastosowany algorytm jest algorytmem przeszukiwania lokalnego, natomiast algorytm genetyczny – przetwarzającym w każdym cyklu całą populację, co powoduje, że uzyskanie najlepszego rozwiązania może być szybsze. Wskazuje na to porównanie wyników uzyskanych w pracy [97]. Uwidacznia się to na charakterystykach przez gwałtowne zmiany wartości, dlatego też w przypadku optymalizacji realizowanej dla dużych wielowymiarowych przestrzeni rozwiązań algorytm genetyczny może szybciej znajdować rozwiązania suboptymalne.

Rozdział 4

Zwiększanie skuteczności diagnostycznej

Entropia może być miarą zarówno efektywności generowania testów, jak i kompakcji odpowiedzi testowej, jak to określają definicje 3.10 oraz 3.12. W pracy [133] przeprowadzono omówienie właściwości entropii podczas testowania, głównie w odniesieniu do kompakcji odpowiedzi testowej. Analizowany jest tam wpływ sygnału błędów na pracę kompaktora. W taki sam sposób, jak w artykule [134], entropię można wykorzystać do opisu pracy rejestru zakłócanego. W modelu tym sygnał błędu zostaje zastąpiony sygnałem zakłócenia. Uzupełnienia wymaga przypadek, gdy rejestr w czasie pracy przechodzi przez pewną liczbę stanów, która jest dużo mniejsza od liczby wszystkich możliwych stanów. Celem wprowadzenia w to zagadnienie, poniżej przedstawiono niezbędne definicje i twierdzenia.

Definicja 4.1 (entropia łączna)**.** *Niech sygnały X i U przyjmują wartości odpowiednio ze zbiorów $\{0,1,...,2^m-1\}$ oraz $\{0,1,...,2^n-1\}$ w sposób opisany za pomocą prawdopodobieństwa łącznego $P\{X(t)=i, U(t)=j\}$, gdzie $i=0,...,2^m-1$ oraz $j=0,...,2^n-1$.*

Entropia łączna sygnałów X i U jest zdefiniowana jako:

$$\begin{aligned} H(X,U) &= -\sum_{i=0}^{2^m-1}\sum_{j=0}^{2^n-1} P(X(t)=i, U(t)=j)\cdot \\ &\quad \cdot \log_2 P(X(t)=i, U(t)=j). \end{aligned} \tag{4.1}$$

Twierdzenie 4.1 (entropia łączna)**.**

$$H(X,U) \leqslant H(X) + H(U). \tag{4.2}$$

Równość zachodzi, gdy sygnały X i U są niezależne.

Definicja 4.2 (entropia warunkowa)**.** *Entropia warunkowa sygnału X, gdy znamy sygnał U, jest zdefiniowana zależnością:*

$$H(X|U) = H(X,U) - H(U). \tag{4.3}$$

Entropia warunkowa określa ilość informacji, jaką trzeba dostarczyć, by poznać sygnał X, jeśli znany jest sygnał U. Uwzględniając równanie (4.2)

można wykazać, że entropia warunkowa nie może być większa od entropii bezwarunkowej. Natomiast przekształcenie równania (4.3) do postaci:

$$H(X,U) = H(X|U) + H(U) \tag{4.4}$$

pozwala wyznaczyć entropię łączną w sytuacji, gdy sygnały są wzajemnie zależne.

Twierdzenie 4.2 (górna granica entropii rejestru zakłócanego)**.** *Niech $X(t)$ oznacza stan rejestru w chwili t, $U(t)$ wartość sygnału zakłócającego tuż przed przejściem rejestru do stanu następnego $X(t+1)$, a $X(0)$ jest znanym stanem początkowym rejestru. Entropia rejestru X nie jest większa niż suma entropii warunkowej wejściowego sygnału zakłócającego U, czyli:*

$$H(X(t)) \leqslant \sum_{i=0}^{t-1} H(U(i)|X(i)) \leqslant \sum_{i=0}^{t-1} H(U(i)) \leqslant n. \tag{4.5}$$

Dowód analogicznego twierdzenia zamieszczono w artykule [134].

Górna granica entropii identyfikuje rejestr o maksymalnym cyklu.

Twierdzenie 4.3 (warunek konieczny wzrostu entropii rejestru zakłócanego)**.** *Warunkiem koniecznym wzrostu entropii rejestru zakłócanego jest, aby zakłócenie nie było skorelowane ze stanem rejestru, tzn. entropie zakłócenia i rejestru muszą być niezerowe:*

$$H(U(t)|X(t)) > 0. \tag{4.6}$$

Dowód analogicznego twierdzenia zamieszczono w artykule [134].

Rozpatrzmy rejestr cykliczny. Z twierdzenia 4.1 wynika, że sygnał X może zostać przekształcony w sygnał o entropii mniejszej lub równej sygnałowi przekształconemu. Zatem niech przekształconym sygnałem X będzie sygnał U. Funkcja przekształcająca sygnał może być zrealizowana jako funkcja liniowa (bramki XOR). Dla takiej funkcji możliwe jest uzyskanie grafu przejść z maksymalnym cyklem, a tym samym spełnienie twierdzeń 4.2 i 4.3. Jeżeli rejestr cykliczny zastąpimy rejestrem z nieliniowym sprzężeniem zwrotnym, to uzyskamy rejestr hybrydowy. Natomiast taki sposób tworzenia strumienia zakłócającego nazywany dalej będzie modyfikacją liniową. Jak już wspomniano na początku rozdziału 3., w pracy [97] przeprowadzono badania optymalizacji rejestrów mających określone cechy, które są istotne w diagnostyce. W optymalizacji struktury rejestru wykorzystywany był algorytm genetyczny, natomiast w opracowaniu [91] do tego celu zastosowano programowanie genetyczne. Prace te potwierdziły efektywność modyfikacji liniowej. W dalszej części rozprawy modyfikacja ta będzie stosowana w optymalizacji struktury diagnostycznej.

Z rozdziału 3.1 wynika, że graf przejść rejestru z cyklem realizującym testowanie wyczerpujące może być znacznie dłuższy od testu deterministycznego. Należy więc w obszarze wejść wytworzyć zbiór stanów rejestru, zawierający zbiór pobudzeń testu deterministycznego. Rozwiązaniem tego problemu może być zakłócanie pracy rejestru. Testowanie będzie realizowane w zakresie podgrafu rejestru.

Definicja 3.3 określa cykliczny graf przejść rejestru, który można również zdefiniować jak poniżej.

Definicja 4.3 (cykliczny graf przejść)**.** *Stan następny rejestru opisany jest funkcją* $X(t+1) = f(X(t), U(t))$ *przy stanie rejestru* $X(t)$ *i zakłóceniu* $U(t)$*. Rejestr ma cykliczny graf przejść, jeżeli dla wszystkich zakłóceń* $U(t)$ *i dla każdego* t *spełniona jest zależność* $X(t+1) \neq X(t)$*.*

Definicja 4.4 (ścieżka w grafie przejść)**.** *Zbiór kolejnych stanów w grafie jest ścieżką, jeżeli dla wszystkich zakłóceń* $U(t)$ *funkcja* $X(t+1) = f(X(t), U(t))$ *spełnia zależność* $X(t+1) \neq X(t)$ *(patrz definicja 4.3).*

Z definicji 4.3 i 4.4 jednoznacznie wynika, że ścieżka jest podgrafem.

Twierdzenie 4.4 (entropia ścieżki)**.** *Jeżeli entropia zakłócanego* p*-bitowego obszaru* n*-bitowego rejestru* X *(gdzie* $p \leqslant n$ *i analizowana liczba stanów jest równa* 2^p*) osiąga wartość maksymalną, to graf przejść, opisujący zachowanie rejestru stanowi ścieżkę grafu o długości* 2^p*.*

Dowód. Z właściwości entropii wynika, że osiąga ona wartość maksymalną, gdy prawdopodobieństwo każdego ze stanów jest jednakowe. Zachodzi to w przypadku, gdy występują wszystkie możliwe stany, których jest 2^p. W takim przypadku każdy węzeł grafu ma tylko jedno wyjście tworząc ścieżkę. □

Jeśli w twierdzeniu 4.4 przyjmiemy $p = n$, to rejestr ma graf z cyklem o maksymalnej długości.

Twierdzenie 4.5 (wzrost entropii ścieżki)**.** *Jeżeli zakłócenie* $U(t)$ *jest niezależne od stanu rejestru* $X(t)$ *i zakłócany jest* p*-bitowy obszar* n*-bitowego rejestru* X *(gdzie* $p \leqslant n$ *i analizowana liczba stanów jest równa* 2^p*), to wzrost entropii rejestru* X *jest taki sam, jak zakłócanego (analizowanego) obszaru rejestru.*

Dowód. Na podstawie twierdzenia 4.2 entropia rejestru X nie będzie większa niż suma entropii warunkowej wejściowego sygnału zakłócającego $U(t)$ i maksymalnie osiągnie wartość p. □

Twierdzenie 4.6 (maskowanie uszkodzeń dla ścieżki)**.** *Jeżeli entropia zakłócanego* p*-bitowego obszaru* n*-bitowego rejestru* X *(gdzie* $p \leqslant n$*) osiąga wartość maksymalną, to prawdopodobieństwo maskowania uszkodzeń* P_M *wynosi:*

$$P_M = 2^{-p}. \tag{4.7}$$

Dowód. Jeżeli entropia zakłócanego p-bitowego obszaru rejestru osiąga wartość maksymalną, oznacza to, że rejestr przyjmuje 2^p możliwych stanów. Prawdopodobieństwo osiągnięcia każdego ze stanów jest jednakowe i wynosi 2^{-p} i jest równe prawdopodobieństwu uzyskania sygnatury poprawnej w przypadku uszkodzenia układu. □

W przypadku, gdy entropia zakłócanego obszaru nie osiąga wartości maksymalnej, oznacza to powtarzanie się stanu w tym obszarze. Prawdopodobieństwo maskowania nie może być większe, gdyż powtórzenie stanu rejestru związane jest z powtórzeniem stanu pozostałej części rejestru, którego prawdopodobieństwo przyjmuje minimalną wartość, równą 2^{p-n}.

W testowaniu pseudoprzypadkowym długość testu powinna być zbliżona do testu deterministycznego. Jak wynika z przeprowadzonej analizy, zakłócenie w obszarze wejść może pozwolić na uzyskanie takiego testu, jednocześnie nie pogarszając warunków kompakcji. Podsumowując, dla rejestrów mających równoważny podgraf grafu (ścieżkę) o opisanych wyżej właściwościach, efektywność diagnostyczna jest taka sama. Oznacza to dalej, że badanie efektywności struktury diagnostycznej ograniczone zostanie do badania jej właściwości w zakresie testowania (długości testu). Problemem pozostaje, jak zrealizować takie zakłócenie przy minimalnym nadmiarze układowym i czasie projektowania struktury diagnostycznej. Najmniejszy nadmiar układowy wprowadzi realizację zakłóceń wewnątrz struktury diagnostycznej. Jednak w przypadku korelacji stanu rejestru i sygnału zakłócającego, jak wynika z równania (4.2), uzyskanie maksymalnego wzrostu entropii nie będzie osiągalne.

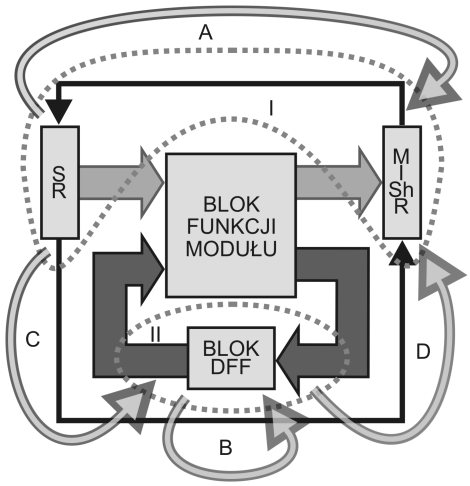
4.1. Koncepcja struktury diagnostycznej

Proponowana metoda modyfikacji liniowej może być alternatywnym rozwiązaniem w stosunku do budowy wewnętrznych ścieżek testujących dzięki mniejszemu nadmiarowi układowemu, potrzebnemu do jej realizacji.

4.1.1. Moduł z elementami struktury diagnostycznej

Jako bazową strukturę diagnostyczną przyjęto pierścień samotestujący. Zbudowany jest on na bazie ścieżki sterująco-obserwującej. Na rysunku 4.1 przedstawiono strukturę diagnostyczną [85]. Struktura pierścienia samotestującego znajduje się w obszarze oznaczonym jako I.

W prezentowanych w rozdziale 3.1 architekturach wbudowanego testowania rejestry przesuwające LFSR i MISR wyposażone były w liniowe sprzężenia. Odpowiednikiem tych sprzężeń są sprzężenia realizowane w obrębie grupy A, przedstawione na rysunku 4.1 i zaznaczone symbolicznie za pomocą strzałki. W obrębie struktury ścieżki brzegowej, tworzącej pierścień CSTP, modyfikacja ta realizuje dowolne liniowe sprzężenie zwrotne rejestru przesuwającego.



Rysunek 4.1. Struktura diagnostyczna jednego modułu

Obszar wewnętrznych elementów pamięci oznaczono na rysunku 4.1 jako B. Włączenie komórek pamięci reprezentowanych jako komórki ścieżki sterująco-obserwującej do pierścienia samotestującego wiązałoby się z dużym nadmiarem układowym. Możliwa jest jednak modyfikacja tego obszaru przy mniejszym nadmiarze układowym. Modyfikacja ta nie tworzy układu (połączeń) rejestru LFSR w klasycznym jego rozumieniu.

Modyfikacja liniowa polegająca na podłączeniu wejść wewnętrznych elementów pamięci ze strukturą CSTP przedstawiona jest na rysunku 4.1 i oznaczona jako C. Połączenie to ma za zadanie „stymulować" obszar bloków pamięci oraz w pośredni sposób wejścia testowanego układu.

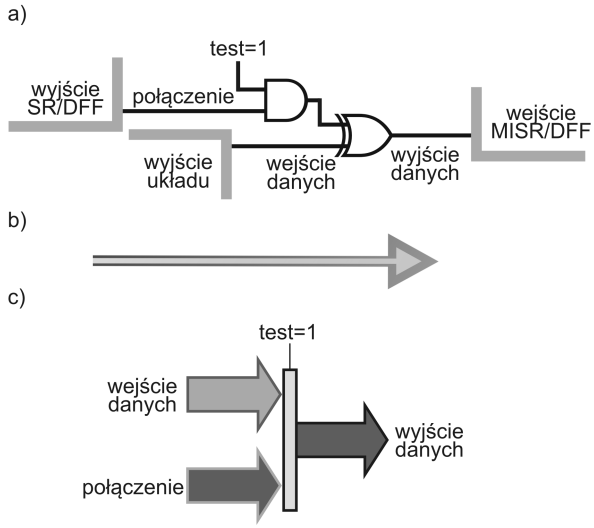
Modyfikacja liniowa realizująca podłączenia wyjść wewnętrznych elementów pamięci z elementami struktury CSTP została przedstawiona na rysunku 4.1 i oznaczona jako D. Ma ona za zadanie „obserwować" obszar wewnętrznej pamięci, a co za tym idzie i podłączone do niej wyjścia układu.

Dla przyjętego modelu układu sekwencyjnego wydzielono obszar elementów pamięci DFFs oraz strukturę CSTP, obejmującą tylko wejścia i wyjścia pierwotne układu, która stanowi naturalną implementację dla standardu IEEE 1149.1. Proponowana modyfikacja liniowa nie powoduje istotnych zmian budowy tych elementów.

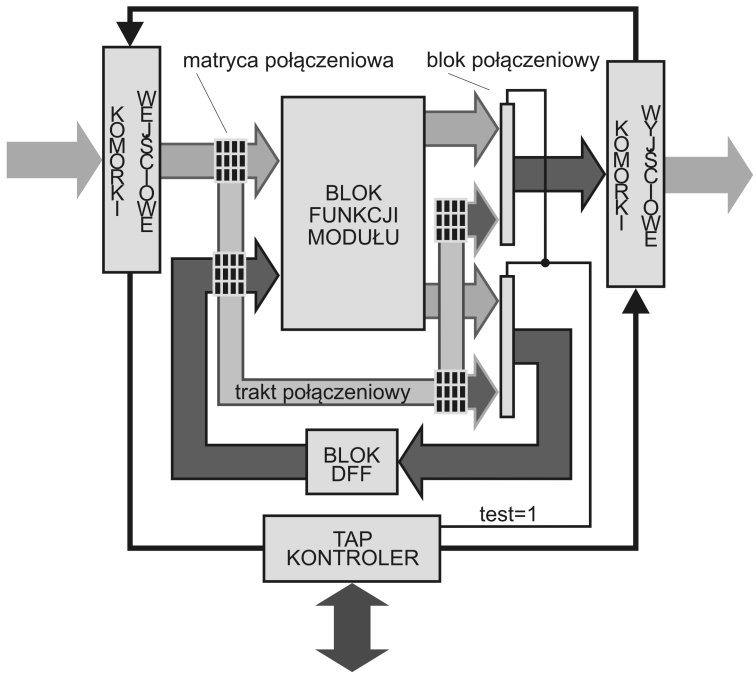
4.1.2. Sprzętowa realizacja struktury modyfikacji liniowej

Modyfikacja liniowa bazuje na idei zaczerpniętej z budowy rejestrów LFSR. Schemat komórki modyfikacji liniowej przedstawia rysunek 4.2.

Podobny schemat (rysunek 4.2a) ma punkt testowy (*test point*) prezentowany w pracy [216], w którym zamiast bramki XOR użyto bramki OR. Grupy połączeń liniowych zostały tak dobrane, aby umożliwić ich praktyczne



Rysunek 4.2. Modyfikacja liniowa: a) schemat komórki, b) symbol, c) schemat blokowy

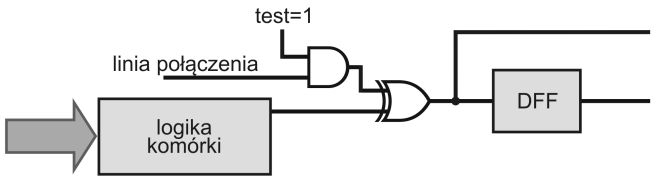


Rysunek 4.3. Realizacja metody modyfikacji liniowej

wykorzystanie. Realizacja tej techniki może być szczególnie dogodna w układach ASIC SC (Standard Cell). Bloki połączeń liniowych umieszczone są poza

logiką układu. Połączenia realizowane są za pomocą traktów i matryc połączeniowych. Przykład makrobloku ze strukturą modyfikacji liniowej przedstawiono na rysunku 4.3.

Realizacja struktury testowania bazującej na modyfikacji liniowej może być interesująca dla układów ASIC GA (*Gate Array*) programowalnych maską. Poniżej, na rysunku 4.4 przedstawiono propozycję modyfikacji komórki FPGA, celem realizacji połączeń liniowych struktury testowania.



Rysunek 4.4. Modyfikacja liniowa komórki FPGA

Połączenia realizowane są za pomocą traktów struktury FPGA. Sygnał *test* połączony jest z modułem TAP Controler. Testowanie struktur FPGA jest oddzielnym zagadnieniem bazującym na właściwościach tych układów.

Prezentowana struktura może być wykorzystana jako sygnał pobudzania wewnątrz modułu lub do propagacji sygnału na zewnątrz modułu, zwiększając skuteczność diagnostyczną.

4.1.3. Układy z modułami połączonymi ścieżką brzegową

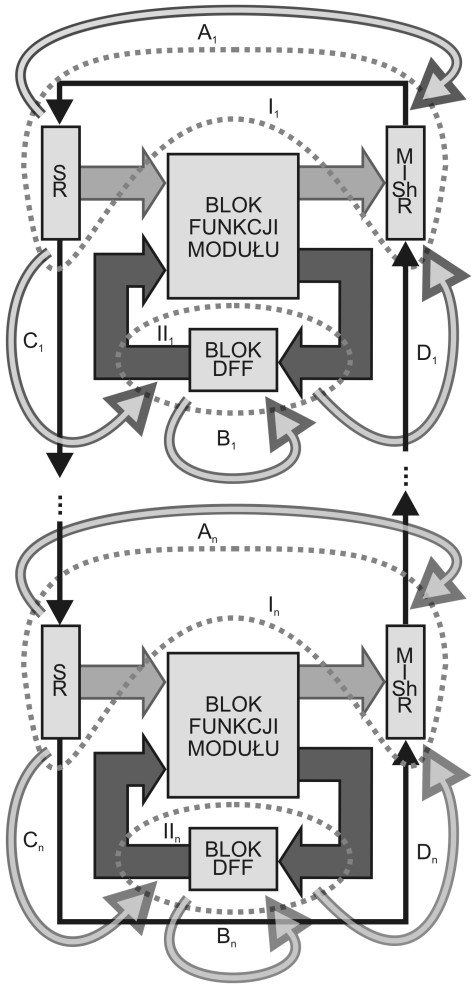
Do badań eksperymentalnych zamodelowano układy testowe ISCAS'89 w strukturze pierścienia samotestującego, tworzącego układy jedno-, dwu- i trzymodułowe, tak jak na rysunku 4.5 [87].

Struktura pierścienia samotestującego jest budowana na podstawie ścieżki brzegowej (zgodnie ze standardem IEEE 1149.1) [84]. Do optymalizacji struktury testowania zastosowano algorytm genetyczny, wyposażony w mechanizm redukcji połączeń. Parametry pracy algorytmu są jednakowe dla wszystkich przeprowadzonych eksperymentów (można więc przypuszczać, że dobór parametrów pracy algorytmu genetycznego może powodować uzyskanie lepszych rezultatów).

Dla każdego modułu (benchmarku) zdefiniowano liczbę możliwych modyfikacji liniowych w każdej z grup.

4.1.4. Wpływ sprzężeń liniowych w poszczególnych grupach na skuteczność diagnostyczną

Weryfikację koncepcji przeprowadzono opierając się na modelowaniu uszkodzeń w układach testowych ISCAS'89. Bazowy schemat struktury testowania wygląda jak na rysunku 4.5. Natomiast układy wielomodułowe symulowano stosując odpowiednio dwa lub trzy takie same moduły. Budowę



Rysunek 4.5. Moduły połączone ścieżką brzegową: I.1, I.n – grupa rejestrów zewnętrznych, tworzących strukturę pierścienia samotestującego; II.1, II.n – grupa elementów pamięci układu sekwencyjnego; A.1, A.n – sprzężenia liniowe w obszarze rejestrów tworzących pierścień samotestujący; B.1, B.n – sprzężenia liniowe w obszarze elementów pamięci (sprzężenia te w sposób dowolny łączą wewnętrzne elementy pamięci); C.1, C.n – sprzężenia liniowe o charakterze zakłócającym (zwiększającym entropię obszaru II); D1, Dn – sprzężenia liniowe o charakterze zakłócającym/obserwującym

struktury testowania i wyznaczenie skuteczności testowania wykonano dla następujących przypadków:

1) sprzężenia liniowe realizowane są wewnątrz struktury CSTP,
2) tworzone są połączenia liniowe wewnątrz bloku pamięci,

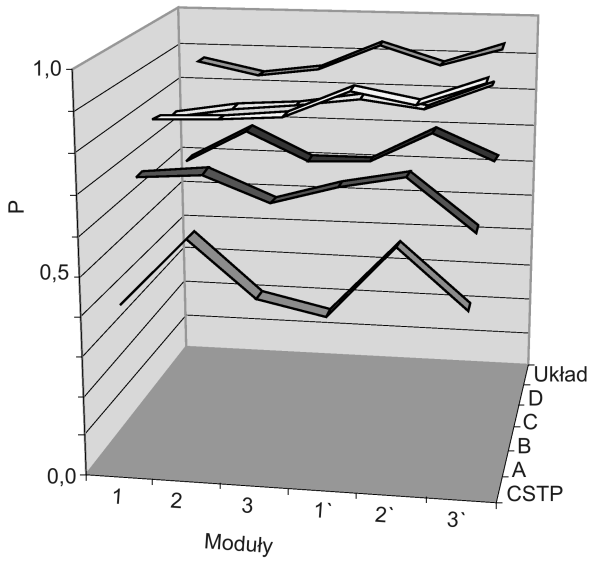
Tabela 4.1. Liczba linii połączeń

Moduł	Liczba linii połączeń				
	A	B	C	D	Razem
S208	11	8	8	1	28
S289	9	14	14	6	43
S344	20	15	15	11	61
S349	20	15	15	11	61
S382	9	21	21	6	57
S386	28	18	18	14	78
S400	9	21	21	6	57
S420	19	16	16	1	52
S444	9	21	21	6	57
S510	26	6	6	7	45
S526	9	21	21	6	57
S641	59	19	19	24	121
S713	58	19	19	23	119
S820	37	5	5	19	66
S832	37	5	5	19	66
S838	35	32	32	1	100
S935	39	29	29	23	120
S1196	28	18	18	14	78
S1238	28	18	18	14	78
S1423	22	74	74	5	175
S1488	27	6	6	19	58
S1494	27	6	6	19	58

3) połączenia liniowe łączą elementy pamięci wewnętrznej z elementami struktury CSTP,
4) połączenia liniowe łączą elementy struktury CSTP z elementami pamięci wewnętrznej,
5) wykonane są wszystkie podane powyżej połączenia liniowe.

Skuteczność testowania wyznaczono również dla podstawowej struktury CSTP. Algorytm genetyczny optymalizuje połączenia liniowe wewnątrz grup rejestrów. Grupy te oznaczono odpowiednio literami A, B, C i D. Tak wprowadzony podział pozwala wykonać znacznie lepszą analizę struktury testowania, a w konsekwencji jej późniejszą implementację. Badania przeprowadzono dla

układów jedno- i wielomodułowych, zbudowanych z jednakowych modułów oraz dla układów zbudowanych z różnych modułów. Poniżej przedstawiono uzyskane wyniki badań.



Rysunek 4.6. Prawdopodobieństwo wykrycia uszkodzenia dla układów (1) jedno-, (2) dwu- i (3) trzymodułowych, realizowane w algorytmie wielopopulacyjnym (druga populacja oznaczona przez znak ')

Modyfikacja połączeń liniowych grupy A pozwala na zwiększenie efektywności generowania testów w porównaniu ze strukturą CSTP, pozbawioną tej modyfikacji. Realizacja połączeń liniowych w tej grupie wpływa bezpośrednio na wzrost wykrywania uszkodzeń.

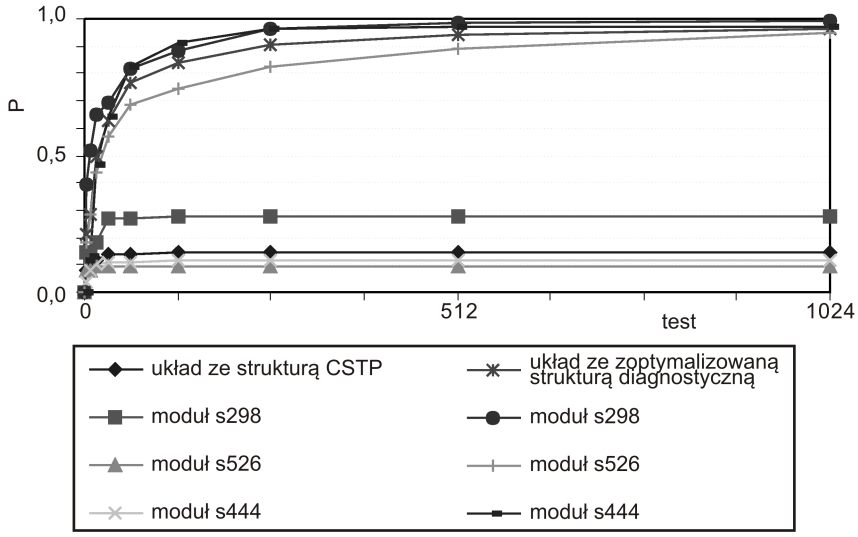
Modyfikacja połączeń liniowych grupy B zwiększa entropię stanów w obszarze II. Połączenia realizowane są pomiędzy dowolnymi elementami pamięci, łącząc ich wyjścia z wejściami. Struktura połączeń nie tworzy typowego rejestru LFSR. Połączenia te mogą również zwiększać propagację sygnałów przez dodatkowe połączenie wyjść o słabej propagacji sygnału do wejść o większej propagacji sygnału. Modyfikacja ta wpływa na zwiększenie skuteczności testowania.

Modyfikacja połączeń liniowych grupy C pozwala na zwiększenie entropii stanów w obszarze bloku pamięci przez wprowadzenie dodatkowych sygnałów o większej entropii do wejść o niskiej entropii. Z przeprowadzonych badań wynika, że wzrost skuteczności testowania, spowodowany modyfikacją liniową grupy C, jest znaczący. Modyfikację tę należy więc uznać jako istotny czynnik zwiększania skuteczności testowania.

Modyfikacja połączeń liniowych grupy D zwiększa propagację sygnałów z obszaru wewnętrznego elementów pamięci do struktury CSTP. Wpływa to również na zwiększenie skuteczności testowania.

Z przeprowadzonych badań wynika, że każda z grup połączeń ma swój wkład w podnoszenie skuteczności testowania. Jednoczesne ich użycie daje najlepsze rezultaty, co ilustruje wykres z rysunku 4.6.

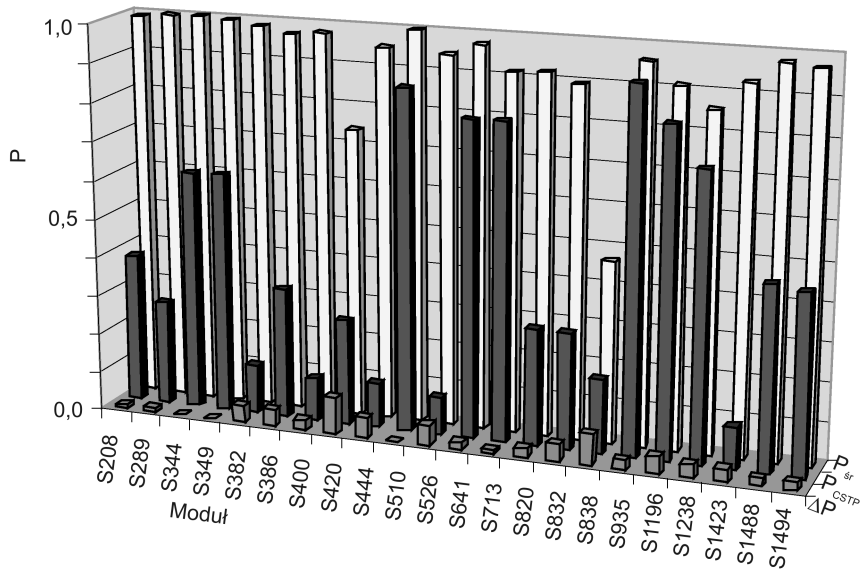
Jak wynika z przeprowadzonych badań, liczba połączeń w obrębie grupy nie jest duża i nie zwiększa się przy wzroście liczby modułów. Wzrost liczby połączeń liniowych nie wpływa w istotny sposób na wzrost skuteczności diagnostycznej. Uzyskiwanie średniej liczby połączeń liniowych, obserwowane dla wszystkich grup, może być spowodowane przez użytą metodę optymalizacji. Najlepsze rezultaty uzyskujemy stosując modyfikację liniową we wszystkich grupach jednocześnie. Nieznaczne zmniejszenie wartości prawdopodobieństwa wykrycia uszkodzeń obserwowane jest wraz ze wzrostem liczby modułów.



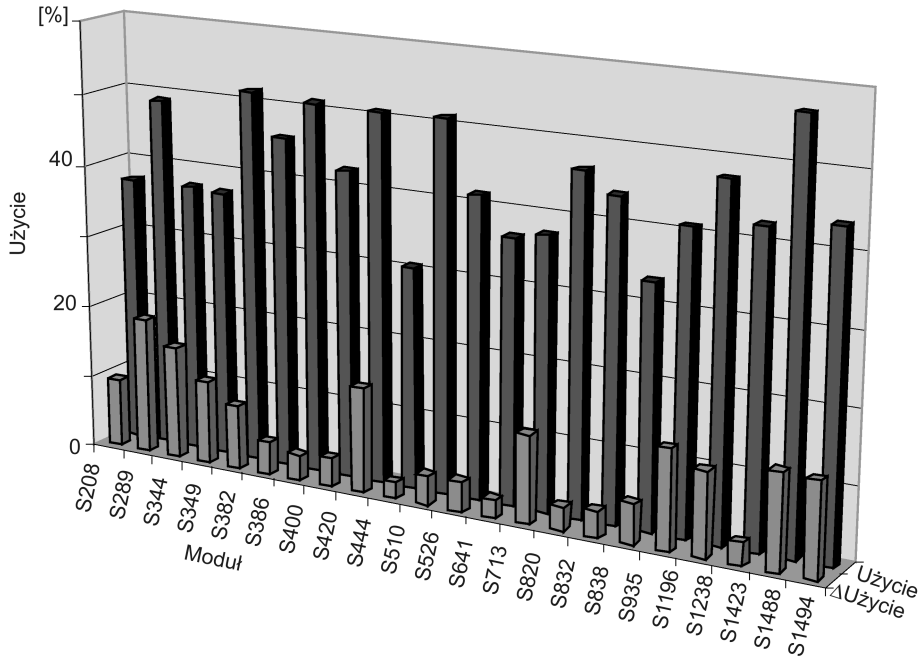
Rysunek 4.7. Wpływ proponowanej struktury testowania na wykrywanie uszkodzeń w układach wielomodułowych

Modyfikacja liniowa wpływa na zwiększenie i wyrównanie wartości prawdopodobieństwa wykrycia uszkodzenia dla modułów składowych w strukturach wielomodułowych – rysunek 4.7. Można również stwierdzić, że dla odpowiednio dobranej struktury testowania nie występuje korelacja strumieni, wpływająca na zmniejszenie entropii w czasie testowania (zgodnie z twierdzeniem 4.3).

Na rysunku 4.8 przedstawiono wartości średnie prawdopodobieństwa wykrycia uszkodzenia dla struktury testowania z rysunku 4.5. Jak wynika z prezentowanych danych, uzyskano wysoką skuteczność testowania. Relatywnie niskie wartości prawdopodobieństwa wykrycia uszkodzenia otrzymano dla



Rysunek 4.8. Średnie prawdopodobieństwo wykrycia uszkodzenia dla układów z rysunku 4.5



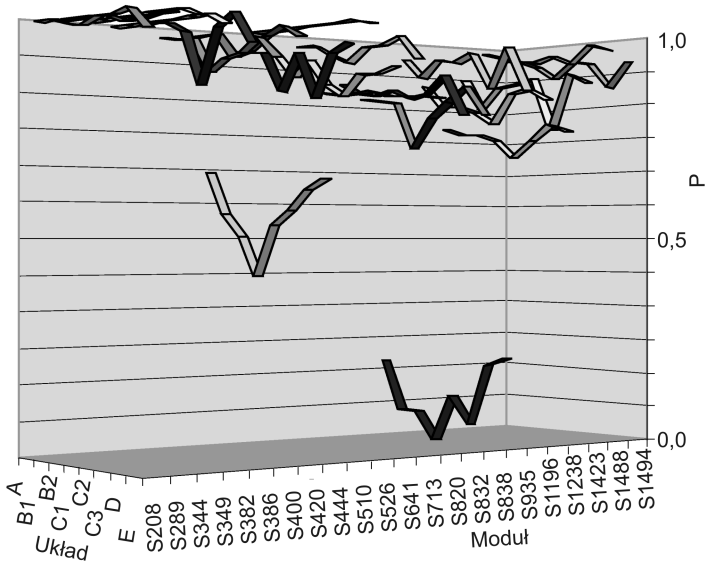
Rysunek 4.9. Procentowy udział połączeń dla poszczególnych modułów

Tabela 4.2. Prawdopodobieństwo wykrycia uszkodzenia oraz udział połączeń liniowych dla przykładowych układów dwumodułowych

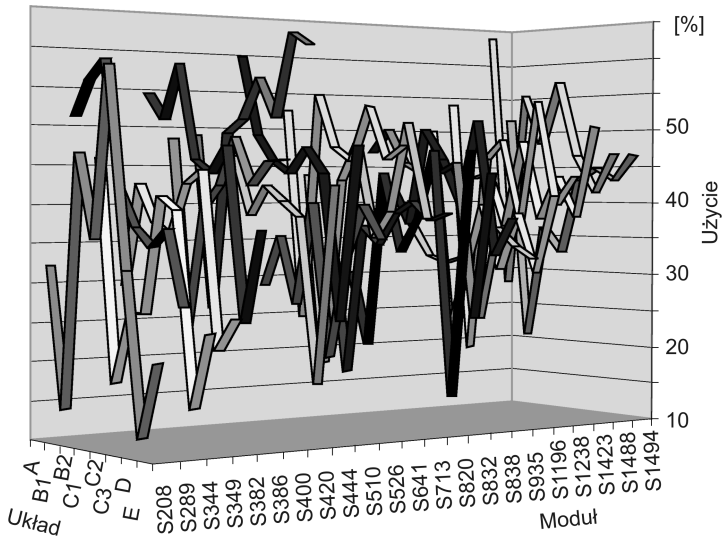
Moduł 1			Moduł 2			Układ	
Moduł	P [%]	Użycie	Moduł	P [%]	Użycie	P [%]	Użycie
s208	1,00	12	s510	1,00	40	1,00	36
s344	1,00	15	s641	0,98	50	0,99	48
s349	0,99	22	s1238	0,87	40	0,89	45
s382	1,00	25	s400	0,99	61	0,99	56
s420	0,76	17	s382	0,98	46	0,87	41
s444	0,95	25	s1494	0,98	33	0,98	42
s526	0,93	46	s953	0,98	34	0,96	40
s820	0,93	15	s713	0,90	38	0,92	39
s832	0,92	25	s298	0,98	44	0,93	43
s838	0,52	33	s1494	0,93	43	0,77	38
s1196	0,92	31	s298	0,99	37	0,93	42
s1488	0,99	41	s386	0,97	45	0,98	43
s1423	0,94	38	s208	0,98	32	0,95	38

Tabela 4.3. Prawdopodobieństwo wykrycia uszkodzenia oraz udział połączeń liniowych dla przykładowych układów trzymodułowych

Moduł 1			Moduł 2			Moduł 3			Układ	
Moduł	P [%]	Użycie	Moduł	P [%]	Użycie	Moduł	P [%]	Użycie	P [%]	Użycie
s208	0,96	32	s832	0,87	44	s1196	0,90	41	0,90	39
s208	0,97	21	s1238	0,86	44	s344	0,99	25	0,89	30
s349	0,98	26	s420	0,78	42	s641	0,95	37	0,90	35
s386	0,94	45	s444	0,96	47	s641	0,95	38	0,95	44
s444	0,95	53	s1423	0,93	50	s208	0,94	54	0,93	52
s526	0,94	47	s382	0,96	37	s820	0,87	47	0,91	44
s1488	0,98	47	s400	0,97	60	s510	0,99	36	0,98	47
s1494	0,96	47	s838	0,53	38	s953	0,91	33	0,83	39
s298	0,99	28	s713	0,90	39	s838	0,45	41	0,69	36

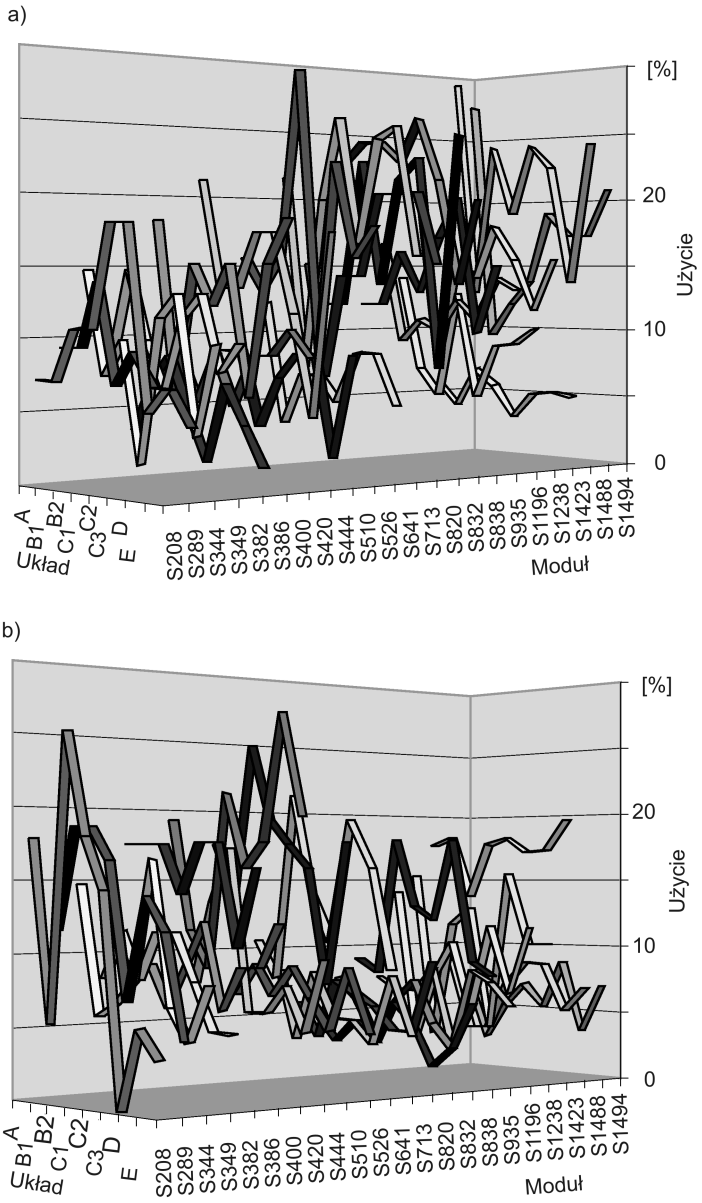


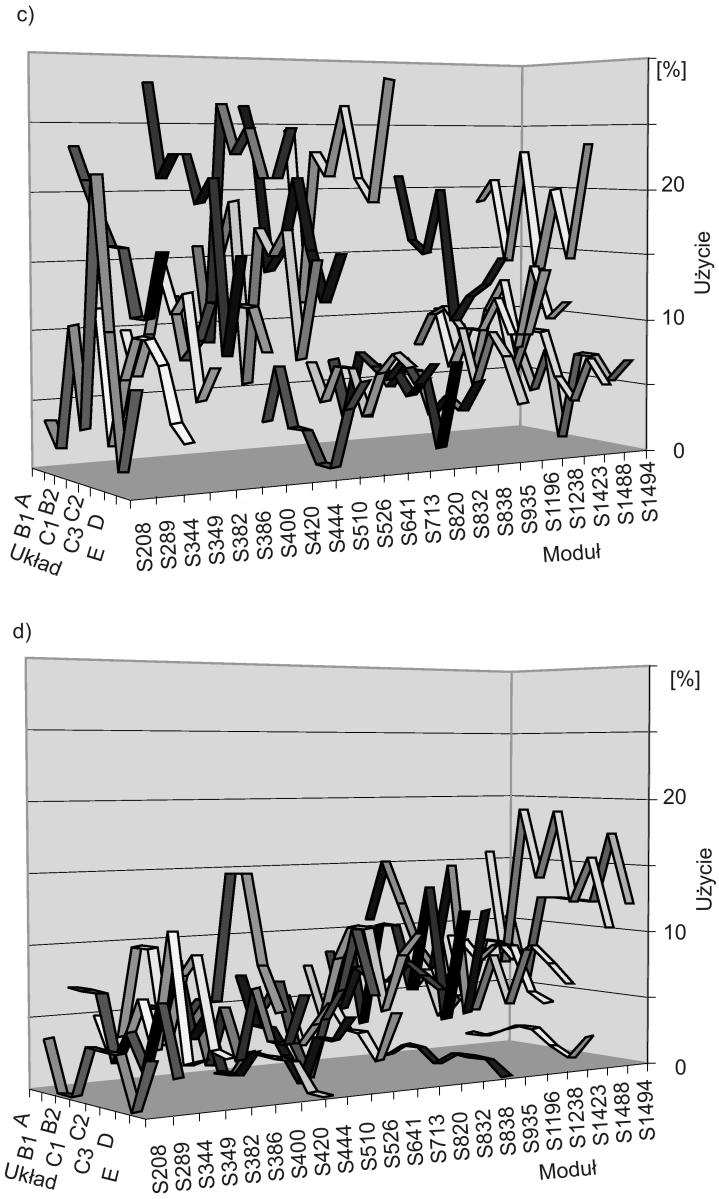
Rysunek 4.10. Prawdopodobieństwo wykrycia uszkodzenia w module; P – prawdopodobieństwo wykrycia uszkodzenia



Rysunek 4.11. Procentowy udział połączeń liniowych w strukturze

modułów s420, s838. Odchylenie wartości prawdopodobieństwa wykrycia uszkodzenia dochodzi do $0,09$ (bez układów s420, s838 wynosi $0,05$). Wykorzystanie połączeń liniowych waha się od 30% do 57%, a odchylenie tych wartości wynosi od 2% do 19% (rysunek 4.9).


a)
[%]
20
10
0
Użycie
A
B1
B2
C1
C2
C3
D
E
Układ
S208
S289
S344
S349
S382
S386
S400
S420
S444
S510
S526
S641
S713
S820
S832
S838
S935
S1196
S1238
S1423
S1488
S1494
Moduł
b)
[%]
20
10
0
Użycie
A
B1
B2
C1
C2
C3
D
E
Układ
S208
S289
S344
S349
S382
S386
S400
S420
S444
S510
S526
S641
S713
S820
S832
S838
S935
S1196
S1238
S1423
S1488
S1494
Moduł



Rysunek 4.12. Udział połączeń liniowych w poszczególnych grupach: a) grupa A, b) grupa B, c) grupa C, d) grupa D

Algorytm genetyczny wyposażono w mechanizm redukcji połączeń liniowych, jednak nie włączono go jako składnika funkcji dostosowania – został

on zaimplementowany w operatorze mutacji. Mimo to występują rozwiązania o mniejszej liczbie połączeń i dużej skuteczności testowania. Pozwala to wnioskować, że uwzględnienie tego składnika w funkcji dostosowania i dobór parametrów pracy algorytmu genetycznego umożliwią uzyskiwanie wyższej skuteczności testowania, połączonej ze znaczną redukcją elementów realizujących sprzężenie liniowe. Badania te powtórzono dla układów zbudowanych z różnych modułów.

Jak wynika z danych prezentowanych w tabelach 4.2 i 4.3, dla układów wielomodułowych, zbudowanych z różnych modułów, prawdopodobieństwo wykrycia uszkodzenia jest również wysokie. Na rysunku 4.10 przedstawiono zestawienia wyników prawdopodobieństwa wykrycia uszkodzenia dla modułów w analizowanych układach:

A – układ jednomodułowy,
B1 – układ zbudowany z dwóch jednakowych modułów – moduł pierwszy,
B2 – układ zbudowany z dwóch jednakowych modułów – moduł drugi,
C1 – układ zbudowany z trzech jednakowych modułów – moduł pierwszy,
C2 – układ zbudowany z trzech jednakowych modułów – moduł drugi,
C3 – układ zbudowany z trzech jednakowych modułów – moduł trzeci,
D – układ zbudowany z dwóch różnych modułów,
E – układ zbudowany z trzech różnych modułów.

Widać wyraźnie, że uzyskiwane wartości prawdopodobieństwa wykrycia uszkodzenia nie zależą w istotny sposób od analizowanego układu (jednomodułowy, wielomodułowy jednorodny, wielomodułowy niejednorodny). Wraz ze wzrostem wielkości modułów, występują większe różnice w wartościach prawdopodobieństwa wykrycia uszkodzenia.

Procentowy udział połączeń liniowych dla modułów, w zależności od analizowanego układu, przedstawiono na rysunku 4.11.

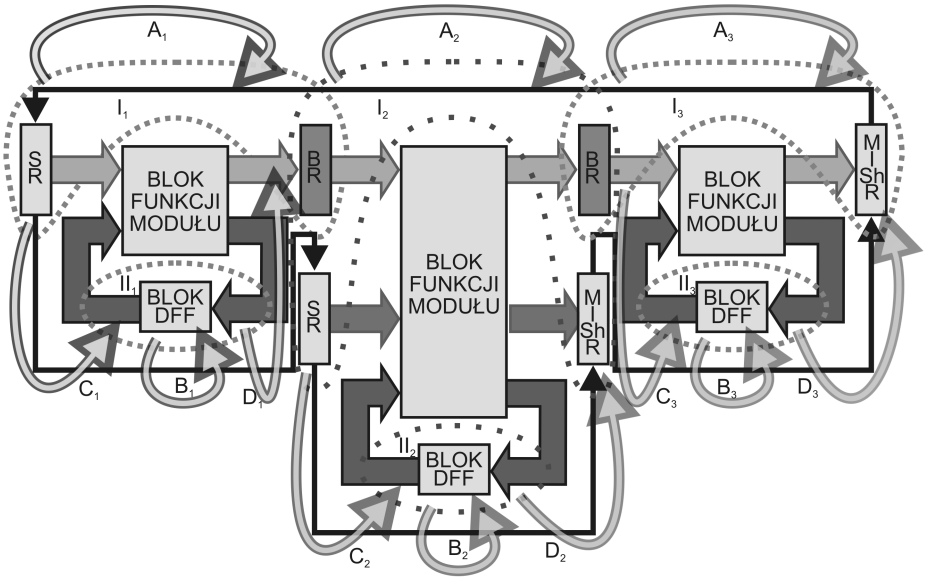
Zróżnicowanie wartości jest bardzo duże i nie zależy ani od wielkości modułu, ani od układu, w którym moduł był użyty.

Analizując realizację połączeń wewnątrz grup (rysunek 4.12) można stwierdzić, że w grupach C i D liczba realizowanych połączeń zależy od cech modułu. Grupa D nie wymaga dużej liczby połączeń liniowych. Zależność taka jest znacznie słabiej zauważalna w grupie B.

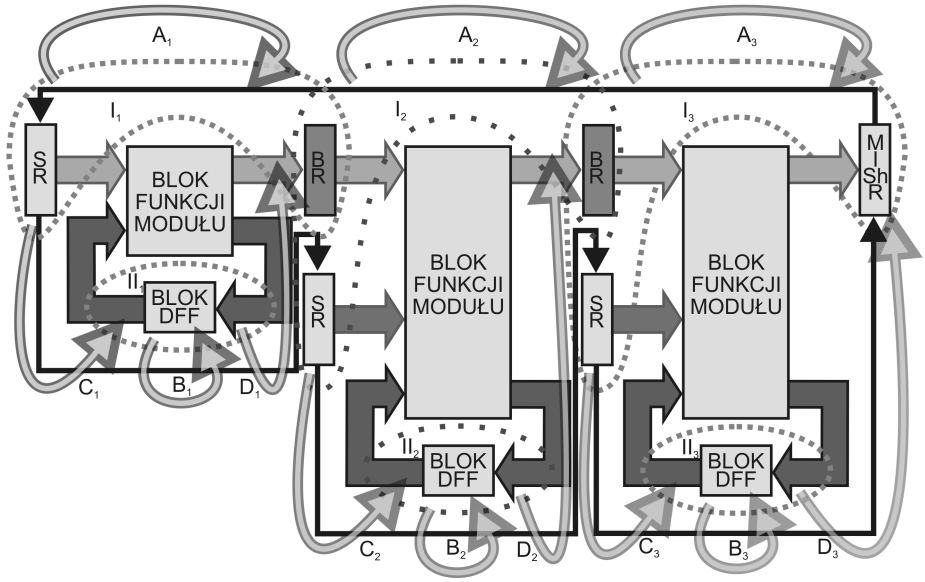
4.1.5. Układy z modułem centralnym

W analizowanych uprzednio układach moduły połączone były ze sobą tylko za pośrednictwem ścieżki brzegowej. W układach badanych poniżej moduły mają wspólne bloki rejestru. Jest to zgodne z przyjętą definicją modułu [85].

Definicja 4.5 (moduł)**.** *Moduł jest częścią układu odseparowaną od jego reszty komórkami rejestru.*



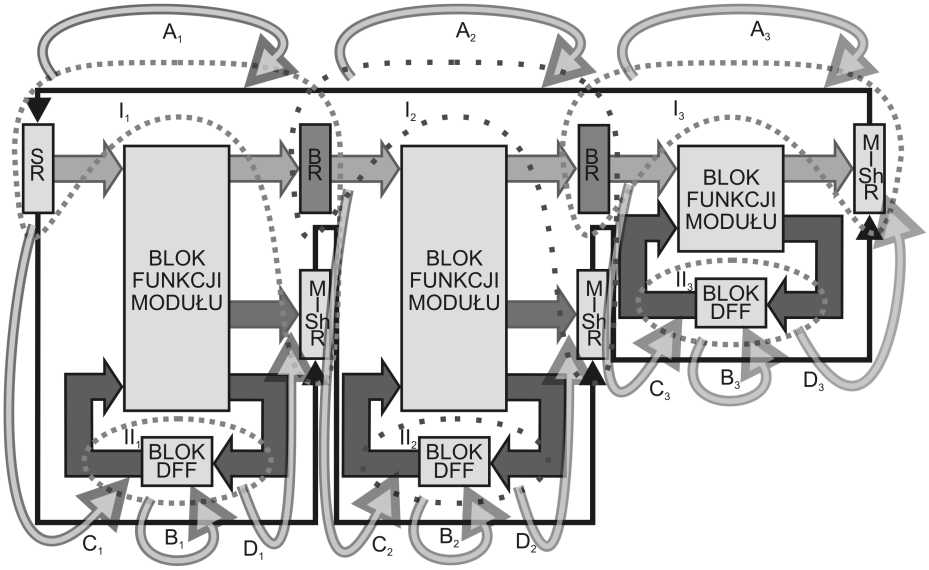
Rysunek 4.13. Układ 1 z modułem centralnym



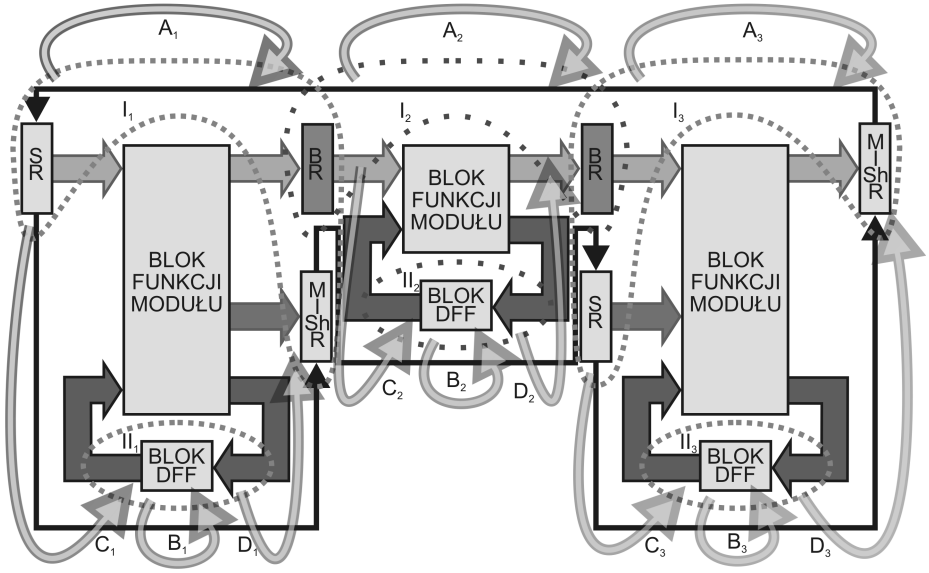
Rysunek 4.14. Układ 2 z modułem centralnym

Dla trzech modułów możemy podać cztery przykładowe architektury, które będą zawierały moduł centralny – obrazują to następujące schematy:

1. Układ 1 (U1) – część wejść i część wyjść modułu centralnego jest połączona ze strukturą CSTP (rysunek 4.13).



Rysunek 4.15. Układ 3 z modułem centralnym



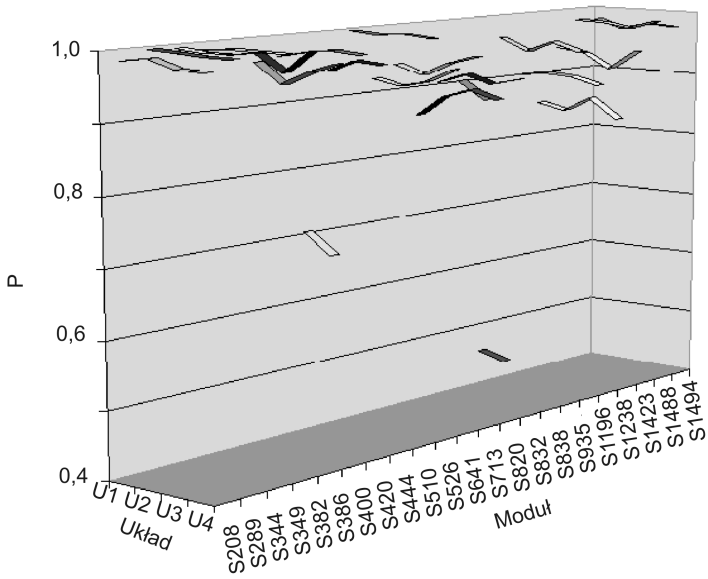
Rysunek 4.16. Układ 4 z modułem centralnym

2. Układ 2 (U2) – wyjścia modułu centralnego są odseparowane od struktury CSTP (rysunek 4.14).
3. Układ 3 (U3) – wyjścia modułu centralnego są odseparowane od struktury CSTP (rysunek 4.15).

4. Układ 4 (U4) – wyjścia i wejścia modułu centralnego są odseparowane od struktury (rysunek 4.16).

Rejestr BR nie należy do ścieżki samotestującej, a struktura diagnostyczna nie buduje w tym obszarze rejestru przesuwającego (dla układów rzeczywistych można rozpatrywać brak rejestrów BR). Grupy modyfikacji liniowej są analogiczne do już uprzednio rozpatrywanych. Dla każdego schematu użyto analizowanych modułów, jednak możliwość ich lokalizacji zależała głównie od liczby wejść i wyjść. Przeprowadzone eksperymenty miały za zadanie scharakteryzować struktury diagnostyczne pod względem efektywności oraz złożoności.

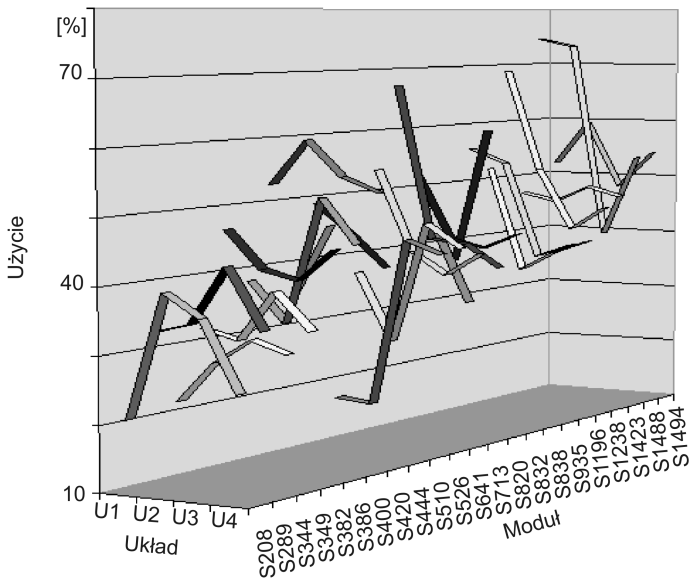
Prawdopodobieństwo wykrycia uszkodzenia dla modułów w analizowanych strukturach testowania przedstawiono na rysunku 4.17.



Rysunek 4.17. Prawdopodobieństwo wykrycia uszkodzenia dla modułów w analizowanych strukturach testowania

Procentowe użycie połączeń liniowych dla modułów w analizowanych strukturach testowania przedstawiono na rysunku 4.18.

Algorytm genetyczny efektywnie optymalizuje strukturę diagnostyczną, przez co podnosi skuteczność wykrywania uszkodzeń. Prawdopodobieństwo wykrywania uszkodzeń dla każdego modułu jest wysokie. Schemat oraz położenie układu nie wpływają w istotny sposób na tę wartość. Można natomiast obserwować nieznaczne zmniejszanie się wartości wykrycia uszkodzenia wraz ze wzrostem złożoności modułu. Oznacza to, że metoda będzie bardziej efektywna dla małych modułów. Relatywnie niskie wartości prawdopodobieństwa wykrycia uszkodzenia uzyskano dla modułów s420 i s838 (jest to związane z ich budową).



Rysunek 4.18. Procentowe użycie połączeń liniowych dla modułów w analizowanych strukturach testowania

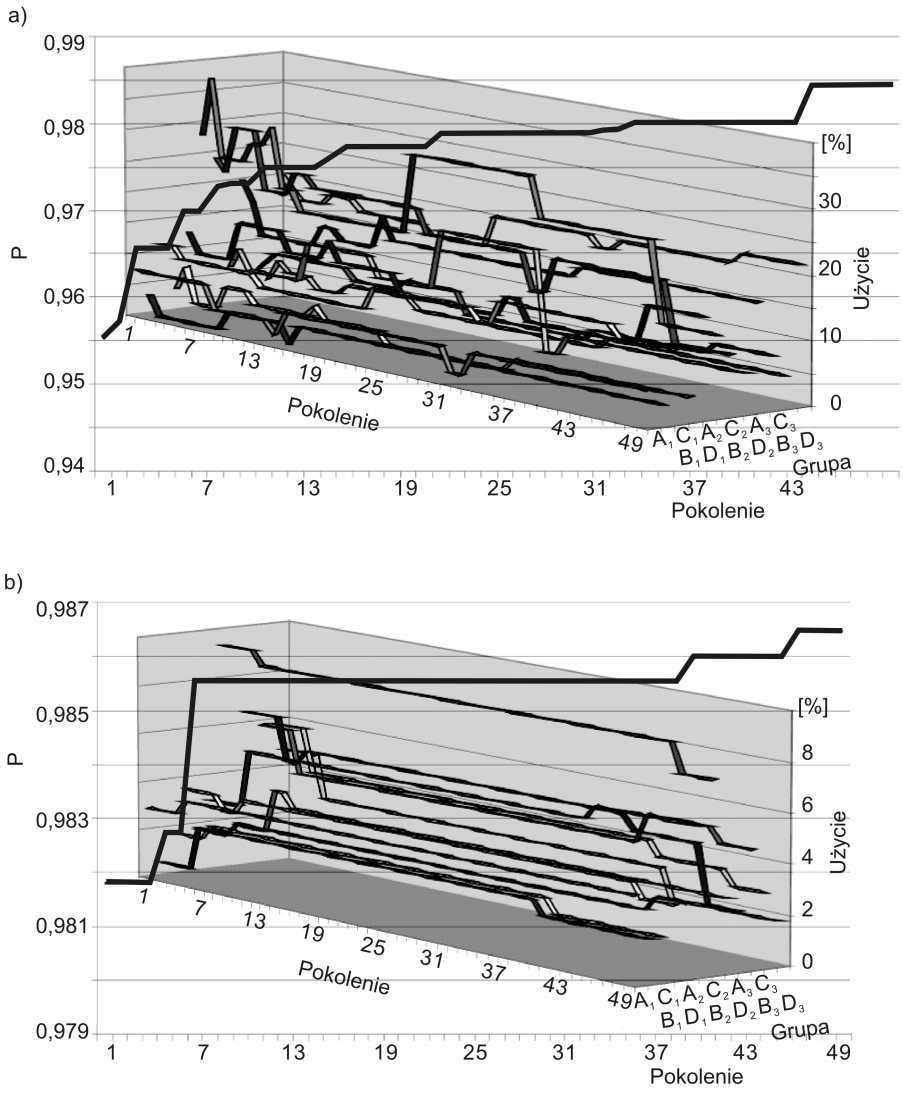
W tabeli 4.4 przedstawiono konfigurację struktur trzymodułowych, których proces optymalizacji ilustrują rysunki 4.19 i 4.20.

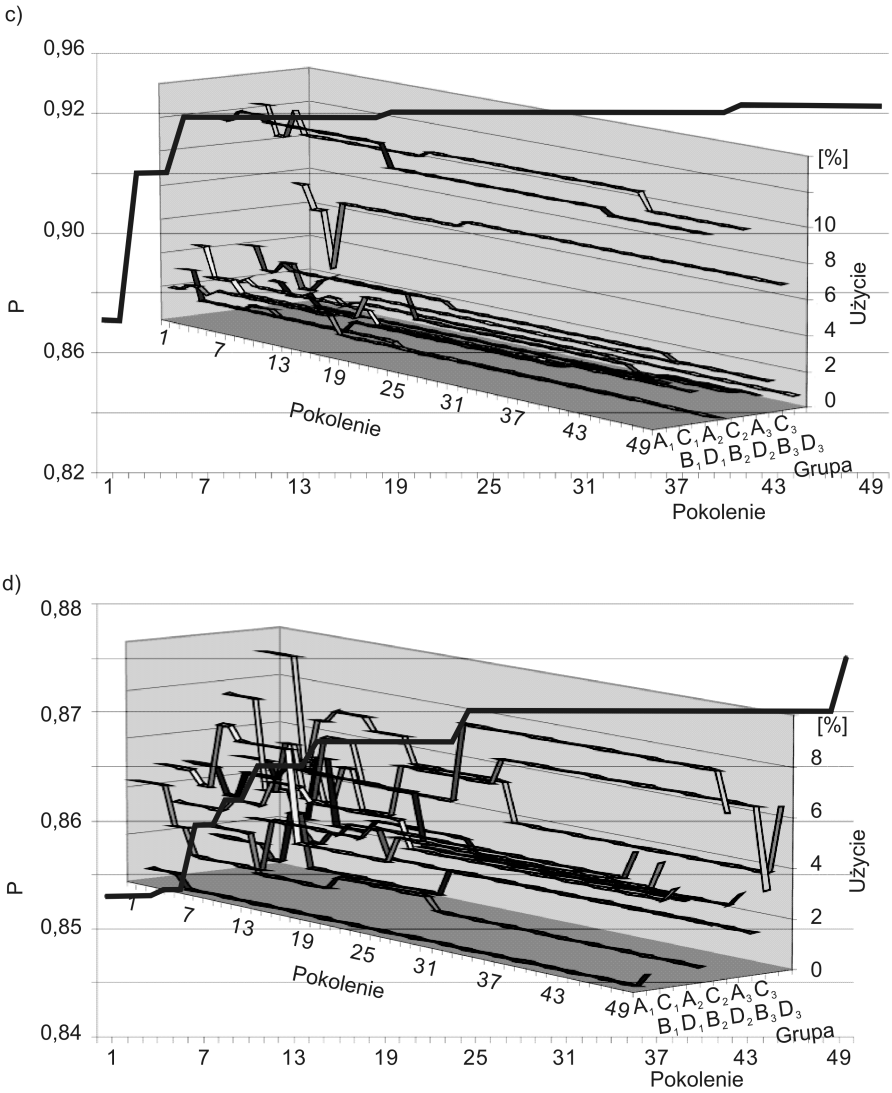
Tabela 4.4. Konfiguracja struktur trzymodułowych

Rysunek	Moduł 1	Moduł 2	Moduł 3	Układ
4.19a	s344	s1494	s641	U1
4.19b	s298	s1498	s349	U1
4.19c	s344	s1423	s820	U2
4.19d	s420	s444	s1196	U2
4.20a	s641	s820	s1196	U3
4.20b	s1238	s208	s953	U4
4.20c	s820	s526	s510	U4*
4.20d	s820	s526	s510	U4**

* pierwsza populacja, ** druga populacja.

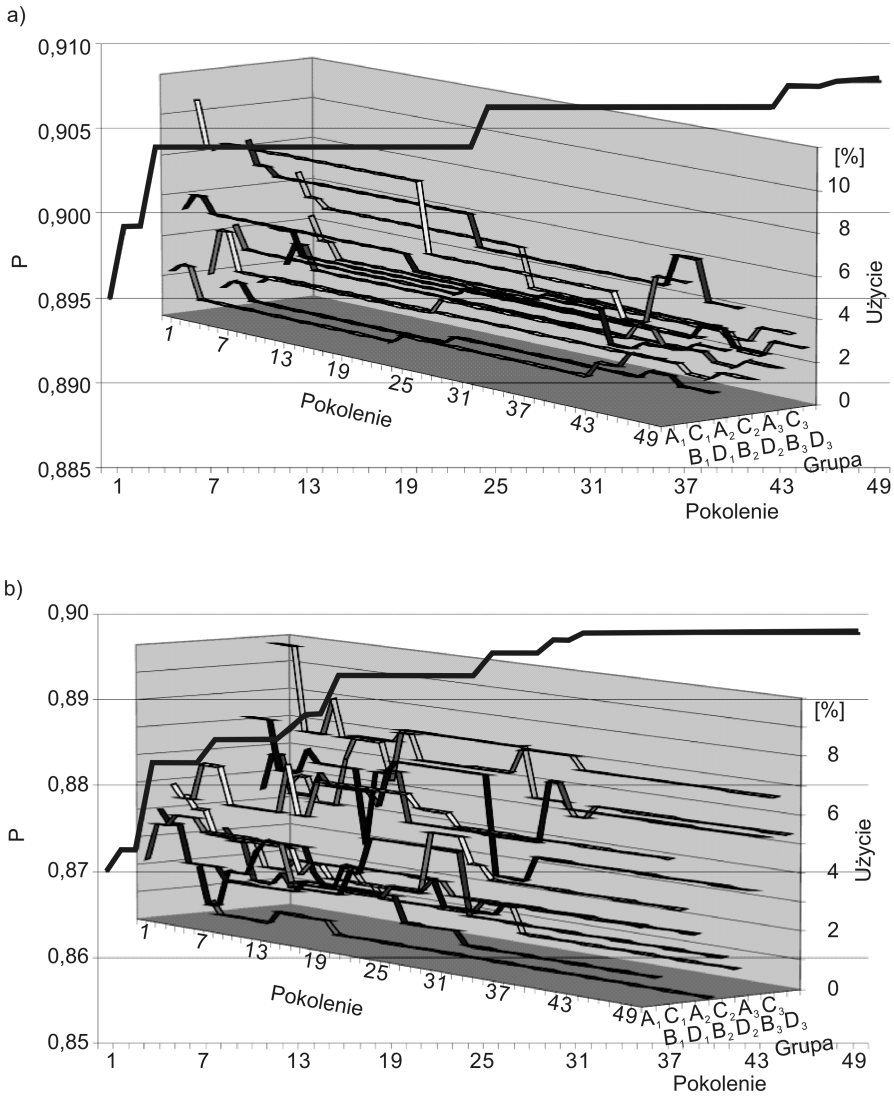
Jak już wcześniej wspomniano, algorytm genetyczny optymalizuje stan początkowy, negację wejść i wyjść, podłączenia układu do struktury testowania oraz architekturę samej struktury diagnostycznej.

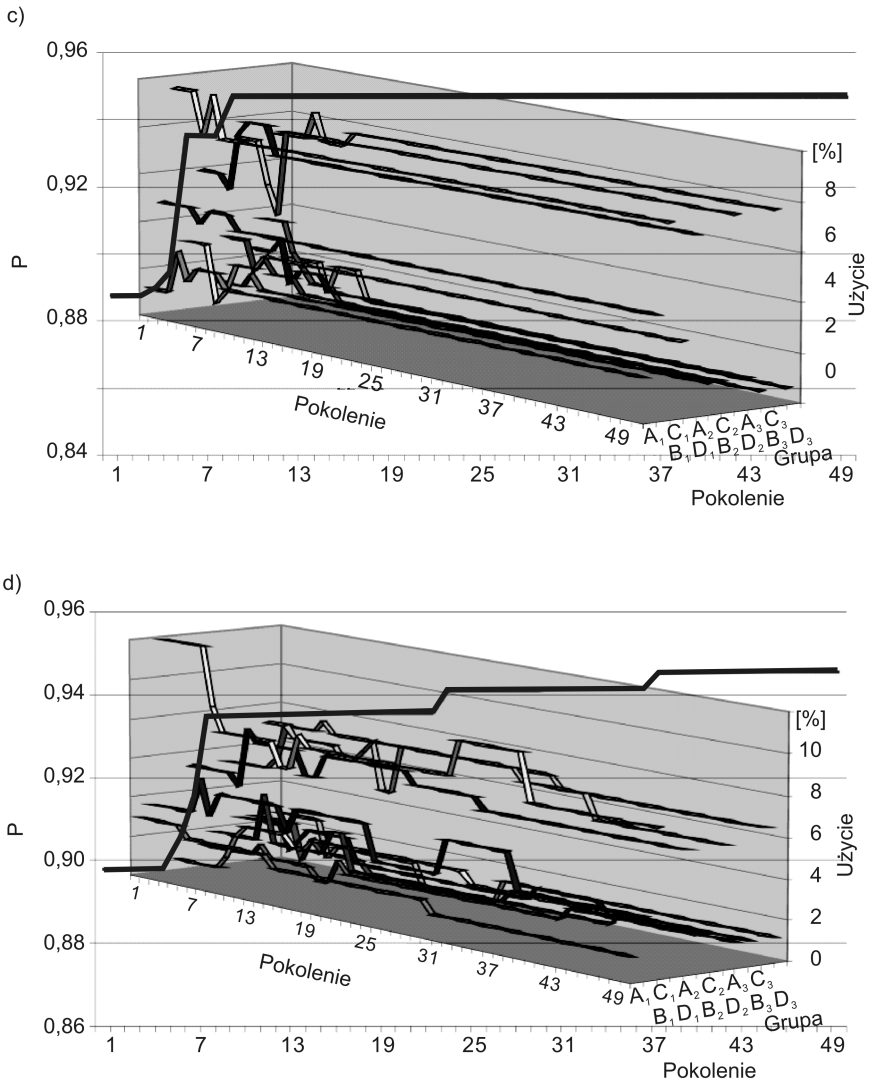

a)
P
0,99
0,98
0,97
0,96
0,95
0,94
[%]
30
20
10
0
Użycie
1
7
13
19
25
31
37
43
49
Pokolenie
A1 C1 A2 C2 A3 C3
B1 D1 B2 D2 B3 D3
Grupa
1 7 13 19 25 31 37 43
Pokolenie
b)
P
0,987
0,985
0,983
0,981
0,979
[%]
8
6
4
2
0
Użycie
1 7 13 19 25 31 37 43 49
Pokolenie
A1 C1 A2 C2 A3 C3
B1 D1 B2 D2 B3 D3
Grupa
1 7 13 19 25 31 37 43 49
Pokolenie



Rysunek 4.19. Prawdopodobieństwo wykrycia uszkodzenia oraz procentowy udział połączeń liniowych w kolejnych pokoleniach: a, b) układ 1; c, d) układ 2

Obserwowany jest jednoczesny wzrost prawdopodobieństwa wykrycia uszkodzenia wraz ze zmianą elementów struktury diagnostycznej. Zmiany prawdopodobieństwa wykrycia uszkodzenia zachodzą w tych samych okresach – mierzonych pokoleniami – co zmiany w architekturze struktury diagnostycznej. Wyniki tego doświadczenia ilustrują rysunki 4.19 i 4.20. Jest to eksperymentalne potwierdzenie twierdzenia o wzroście entropii rejestru


a)
P
0,910
0,905
0,900
0,895
0,890
0,885
[%]
10
8
6
4
2
0
Użycie
1
7
13
19
25
31
37
43
49
Pokolenie
A1 C1 A2 C2 A3 C3
B1 D1 B2 D2 B3 D3
Grupa
b)
P
0,90
0,89
0,88
0,87
0,86
0,85
[%]
8
6
4
2
0
Użycie
Pokolenie
Grupa



Rysunek 4.20. Prawdopodobieństwo wykrycia uszkodzenia oraz procentowy udział połączeń liniowych w kolejnych pokoleniach:
a) układ 4; b, c, d) układ 4

zakłócanego. Oznacza to równocześnie, że pozostałe składowe genomu mają mniejszy wpływ na efektywność diagnostyczną optymalizowanej struktury.

Badania te dowodzą, że możliwe jest realizowanie takiej struktury przy niewielkiej liczbie połączeń. Eksperyment przeprowadzono również dla sys-

temu wielopopulacyjnego – dwie populacje. Dla obydwu populacji uzyskano wysoką skuteczność diagnostyczną.

Pomimo że rejestry BR nie tworzą struktury diagnostycznej, wejścia i wyjścia modułów podłączonych do nich są objęte modyfikacją liniową (nie jest tworzony typowy rejestr przesuwający z liniowym sprzężeniem zwrotnym).

Można postawić tezę, że wprowadzenie nieskorelowanego strumienia o wysokiej entropii stanów (zrealizowane przez układ bramek z rysunku 4.2a) do wnętrza bloku kombinacyjnego w węźle o niskiej entropii stanów wpłynie na wzrost skuteczności diagnostycznej podczas testowania. Mimo iż badań takich nie przeprowadzono, to odpowiednikiem takiego połączenia jest sprzężenie C z rysunku 4.1. Możliwe jest oczywiście zrealizowanie odpowiednika sprzężenia D z rysunku 4.1.

Tego typu rozwiązanie wpłynie na czas propagacji sygnałów w czasie normalnej pracy, jednak może rzutować na podniesienie skuteczności diagnostycznej takich układów, jak s420 i s838. Koncepcja ta znacznie rozszerza przestrzeń możliwych rozwiązań. Pomimo iż analiza entropii wewnątrz układu podczas testowania nie będzie wymagała modelowania uszkodzeń, to do jej realizacji będzie potrzebny system o bardzo dużej mocy obliczeniowej.

4.2. Uwagi dotyczące stosowania algorytmu genetycznego w optymalizacji struktury BIST

Projektowanie algorytmu genetycznego jest sztuką, bazującą jedynie na doświadczeniu programisty. Na podstawie doświadczeń przyjmuje się: liczbę osobników z zakresu 50–100, stosunkowo duże prawdopodobieństwo krzyżowania (80–95%), przy czym prawdopodobieństwo mutacji jest bardzo małe i obejmuje 1 lub 2 geny. Analiza wpływu parametrów algorytmu genetycznego na jego działanie omawiana jest między innymi w artykule [165]. Wykazano, że wpływ operatora mutacji na działanie algorytmu genetycznego ma istotne znaczenie [164]. Mutację można interpretować jako wprowadzenie pewnego, przypadkowego zróżnicowania w populacji. Krzyżowanie prowadzi do losowego połączenia cech osobników rodzicielskich. Stanowi więc formę przeszukiwania przestrzeni rozwiązań, ograniczonej cechami tych osobników. W opracowaniu [205] wykazano, że krzyżowanie jednostajne jest lepsze od jedno- i dwupunktowego. Można więc stwierdzić, że wykonuje lepszą eksplorację tej przestrzeni. Przy silnym naporze selekcyjnym operator krzyżowania będzie eksploatował basen przyciągania. Natomiast zróżnicowanie populacji, czyli eksplorację przestrzeni rozwiązań, będzie realizował operator mutacji.

Optymalizację struktury diagnostycznej dokonano w systemie dwupopulacyjnym i przyjęto, że każda z nich będzie liczyć 40 osobników. Miało to ochronić przed stagnacją tego procesu optymalizacji. Mechanizm redukcji elementów struktury diagnostycznej został zaimplementowany w operatorze mutacji. Dla typowych ustawień operatorów krzyżowania i mutacji nie zaobserwowano

postępu optymalizacji. Dopiero ustawienia mutacji rzędu 20% pozwoliły na wzrost efektywności pracy algorytmu (duże zróżnicowanie populacji – intensywne przeszukiwanie przestrzeni rozwiązań). Jednak wzrost mutacji znacznie ponad tę wartość powodował brak powtarzalności wyników. Naruszało to w istotny sposób mechanizm zbieżności algorytmu. Natomiast istotnym elementem sukcesu była właściwa eksploracja przestrzeni rozwiązań. Uzyskiwane, zadowalające rezultaty doboru struktury diagnostycznej potwierdzają tylko poprawność procesu optymalizacji, przy tak dobranych parametrach. Jest to również potwierdzeniem, że istotnym czynnikiem sukcesu w stosowaniu algorytmów genetycznych jest doświadczenie. Należy więc poszukiwać nowych algorytmów, które mają dużą efektywność zarówno eksploracji, jak i eksploatacji. Rezultaty prowadzonych badań w tym zakresie stały się przyczynkiem do powstania nowych algorytmów, prezentowanych w pierwszej części rozprawy. Dużym utrudnieniem ich stosowania, jak również jakichkolwiek algorytmów ewolucyjnych, w tym również omawianego algorytmu genetycznego, jest bardzo czasochłonny, symulacyjny sposób modelowania uszkodzeń. Metoda ta jest opisana w kolejnym rozdziale. Użycie algorytmu genetycznego okazało się wystarczające do weryfikacji omawianej koncepcji struktury diagnostycznej, jednak w przypadku zastosowania komercyjnego, należałoby wykorzystać bardziej efektywny algorytm, na przykład proponowany w pierwszej części pracy algorytm S-IMM.

4.3. Narzędzie do modelowania i optymalizacji struktur BIST

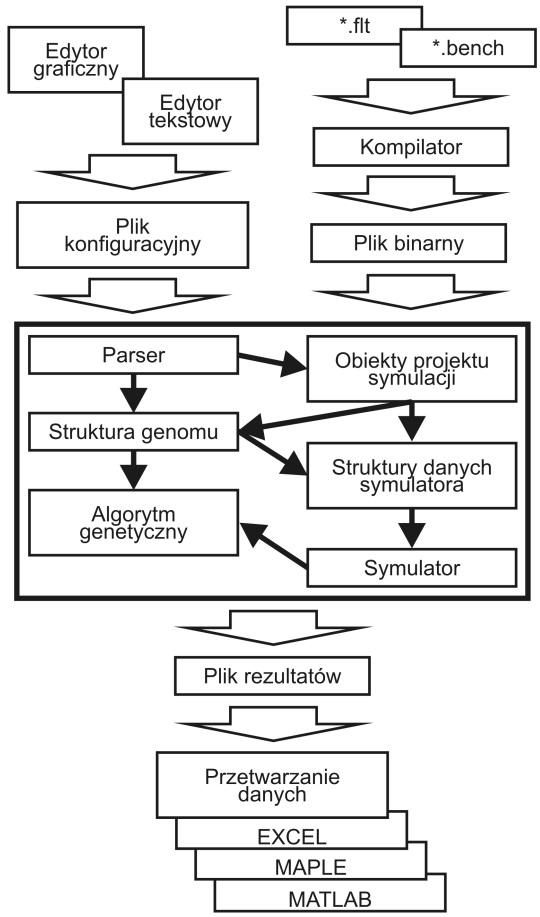
BSDL jest językiem powstałym na bazie filozofii ścieżki brzegowej [57], [176] (standard ścieżki brzegowej został zdefiniowany w IEEE 1149.1 – 1990, natomiast BSDL został włączony do standardu w 1994 roku). Nie jest on językiem opisu sprzętu a raczej zbiorem definicji przydatnych w określaniu przepływu danych w strukturze ścieżki brzegowej podczas generowania testów i diagnostyki błędów. Jego składnia jest podzbiorem języka VHDL. Język BSDL nie pozwala na opis elementów diagnostycznych oraz opis struktury testowania. Poniżej przedstawiono elementy języka składniowo spójne z elementami języka BSDL, pozwalające na opis dowolnej struktury diagnostycznej oraz elementów odpowiedzialnych za sterowanie przebiegiem optymalizacji tej struktury. Zostało to wykorzystane w budowie narzędzia symulacyjnego, służącego do badania i optymalizacji struktur diagnostycznych.

4.3.1. Etapy procesu optymalizacji struktury diagnostycznej

Proces optymalizacji przebiega według następującego schematu [96]:

1. Przygotowanie plików konfiguracyjnych z wykorzystaniem edytora graficznego lub edytora tekstowego – to najważniejszy krok. Pliki konfiguracyjne

zapisane są za pomocą składni zbliżonej do języka BSDL. Opisują architekturę struktury BIST oraz zakres zmian, dopuszczalny w procesie optymalizacji. Plik ten zawiera również dane sterujące procesem optymalizacji.



Rysunek 4.21. Schemat procesu optymalizacji struktury diagnostycznej

2. Przygotowanie plików funkcji testowych – kompilacja plików *.bench i *.flt do plików *.bin. Pliki modułów układu zostają wstępnie przekompilowane i jednocześnie zostają przygotowane odpowiednie struktury danych, podnoszące wydajność procesu symulacji funkcji układu wraz z modelowaniem uszkodzeń.
3. Wykonanie optymalizacji projektu w programie symulatora. Działanie programu jest podzielone na następujące etapy:
   a) wczytanie pliku konfiguracyjnego, zawierającego opis jednej lub wielu konfiguracji struktury testowania,
   b) sprawdzenie poprawności danych zawartych w pliku konfiguracyjnym,

c) utworzenie struktur danych wykorzystywanych przez program oceniający w czasie symulacji,
d) utworzenie struktur danych wykorzystywanych przez algorytm genetyczny do optymalizacji konfiguracji,
e) przeprowadzenie procesu optymalizacji podanych konfiguracji,
f) wygenerowanie raportów.

4. Interpretacja wyników. Ten etap jest niezależny od omawianego narzędzia i może być wspomagany przez liczną grupę programów matematycznych.

Symulator oblicza funkcję (wzór 3.3), korzystając z odpowiednio przygotowanych struktur danych. Poniżej zostaną opisane wybrane atrybuty pliku konfiguracyjnego, służące do opisu struktury testowania (reprezentujące opis omówionego wyżej równania) oraz opisujące proces optymalizacji. Edytor graficzny jest to uproszczony edytor schematów z możliwością ustawienia niezbędnych parametrów konfiguracyjnych. Edytor zastępuje żmudną, ręczną edycję plików konfiguracyjnych symulatora przez zautomatyzowanie procesu generowania projektów.

4.3.2. Plik konfiguracyjny

Podstawową strukturą stosowaną w pliku konfiguracyjnym jest atrybut. Każdy atrybut jest identyfikowany nazwą oraz jest deklarowany na rzecz pewnego obiektu. Definicja atrybutu może zawierać parametry, umieszczane pomiędzy znakami cudzysłowu. Jeżeli występuje kilka parametrów, to oddzielane są one znakiem „&". Ogólny format zapisu parametru ma postać:

attribute < nazwa atrybutu > *of* < nazwa obiektu > *:*
entity is [["< parametry atrybutu >*, " & ...]*
"< parametry atrybutu >*"]*;

Parametry zawarte w atrybucie mogą zostać niezdefiniowane. W pliku konfiguracyjnym muszą znajdować się następujące atrybuty:

- *PROJECT_GROUPS* – deklaracja nazw projektów opisanych plikiem konfiguracyjnym,
- *TESTREG_GPROG_PARAMS* – parametry sterujące pracą algorytmu genetycznego: wielkość populacji, liczba generacji, prawdopodobieństwo krzyżowania, prawdopodobieństwo mutacji, stosowanie strategii elitarnej,
- *SIM_MIGRATE* – parametry migracji osobników pomiędzy populacjami,
- *SIM_PARAMS* – parametry symulatora, metody oceny jak również nazw plików modułów oraz ich aliasów w projekcie,
- *FUNCTION_PARAMS* – parametry użytych modułów.

Dla wszystkich zadeklarowanych projektów musi zostać stworzona część pliku konfiguracyjnego, definiująca dany projekt. Każda definicja projektu składa się z atrybutów grup, atrybutów opisujących strukturę testowania oraz atrybutów procesu optymalizacji. Na podstawie definicji grup tworzona jest przestrzeń rozwiązań. Poniżej przedstawiono atrybuty grup:

- *TESTREG_MOD_MAP_GROUPS* – grupy połączeń dla modułów,

- *TESTREG_LINEAR_FEEDBACK_GROUPS* – grupy liniowych sprzężeń zwrotnych,
- *TESTREG_INITIAL_STATE_GROUPS* – grupy stanu początkowego komórek,
- *TESTREG_INVERSION_GROUPS* – grupy inwersji wejść i wyjść komórek,
- *TESTREG_CONNECTION_MAP_GROUPS* – architektura struktury diagnostycznej; kwalifikacja typów sprzężeń liniowych dla atrybutu *TESTREG_LINEAR_FEEDBACK_GROUPS*: *E* – zewnętrzne, *I* – wewnętrzne, *IE* – zewnętrzno-wewnętrzne, *T* – sprzężenie komórki z sobą, *brak oznaczenia* – brak sprzężenia. Grupa wykorzystywana jest w definicji atrybutu *TESTREG_CONNECTION_MAP_GROUPS*.

Nazwy grup muszą być unikalne w ramach projektu. Zestawienie parametrów dla atrybutów przedstawiono w tabeli 4.5.

Tabela 4.5. Zestawienie parametrów dla atrybutów

Atrybut	Definicja parametrów
TESTREG_MOD_MAP_GROUPS	<nazwa grupy>(<numer wyprowadzenia >[,<numer wyprowadzenia>...])
TESTREG_LINEAR_FEEDBACK_GROUPS	<nazwa grupy>(<numer komórki> [,<numer komórki>...])
TESTREG_INITIAL_STATE_GROUPS	<nazwa grupy> (<lista wartości>)
TESTREG_INVERSION_GROUPS	<nazwa grupy> (<lista wartości>)
TESTREG_CONNECTION_MAP_GROUPS	<nazwa grupy>(<numer komórki>[, <numer komórki>...])
TESTREG_MODULS	<numer komórki> (<alias układu>,<numer wejścia>, <numer wyjścia>)
TESTREG_CONNECTION_MAP_GROUPS	<numer komórki> (<negacja wejścia>,<stan początkowy>, <negacja wyjścia>,<grupa sprzężenia liniowego>, {<komórka> [, <komórka>...]})

Wydruk 4.1 Fragment pliku konfiguracyjnego

```
attribute PROJECT_GROUPS of MyProject:
entity is " G1";
...
attribute TESTREG_MOD_MAP_GROUPS of G1 :
entity is " MapIn (0..3)";
attribute TESTREG_LINEAR_FEEDBACK_GROUPS of G1 :
entity is ;
attribute TESTREG_INITIAL_STATE_GROUPS of G1 :
entity is " Ini (0,1) " ;
attribute TESTREG_INVERSION_GROUPS of G1 :
entity is " NotIn (0,1) ," &
" NotOut (0,1) " ;
...
attribute TESTREG_CONNECTION_MAP_GROUPS of G1 :
entity is " B (5..7),"&
" C (0..3)," &
" D (5..7)," &
" A (0..4)" ;
...
attribute TESTREG_MODULS of G1 :
entity is " 0..3 (TEST1, MapIn[], )," &
" 4 (TEST1, ,0 ), "&
" 5..7 (TEST1, 4++, 1++)";
attribute TESTREG_CELLS of G1 : entity is
" 0 (NotIn[], Ini[], NotOut[], , {4, A[] }),"  &
" 1..3 (NotIn[], Ini[], NotOut[], ,{ 0++, A[] }),"&
" 4 (NotIn[], Ini[], NotOut[], , {3, A[],D[] }),"&
" 5..7 (NotIn[], Ini[], NotOut[, ,{ B[], C[] })" ;
...
```

Globalne parametry symulatora mogą zostać zmodyfikowane w obrębie projektu przez atrybut

- *TESTREG_GENOMES_PARAMS* – redefinicja parametrów optymalizacji.

Opis struktury diagnostycznej definiują następujące parametry:

- *TESTREG_MODULS* – połączenia modułów ze strukturą diagnostyczną,
- *TESTREG_CELLS* – połączenia komórek struktury diagnostycznej oraz ich konfiguracja,
- *GROUPS_PARAMS* – parametry specyficzne dla poszczególnych grup.

W pliku konfiguracyjnym musi zostać zdefiniowany co najmniej jeden projekt.

Grupy tworzą składowe genomu. Nazwa grupy może być poprzedzona zakresem [<class>::], określającym sposób optymalizacji – jej dodatkowe kryteria (w analizowanym systemie nie został on zaimplementowany).

Atrybut *TESTREG_CONNECTION_MAP_GROUPS* opisuje całą strukturę połączeń. Negacja oznaczana jest znakiem *!*. Jeżeli parametr ma być optymalizowany, to należy wpisać odpowiednią nazwę grupy w formacie *<nazwa grupy>[],* uprzednio zdefiniowaną. Grupa sprzężenia liniowego określa, w jakiego rodzaju sprzężeniu komórka bierze udział i do jakiej grupy należy, zgodnie z specyfikacją *TESTREG_LINEAR_FEEDBACK_GROUP*. W przypadku braku tego typu sprzężeń pole to może pozostać puste. Pole *<komórka>*określa numer komórki, do której wyjścia podłączona jest definiowana komórka. W miejscu tym może pojawić się opis połączenia, zdefiniowany zgodnie ze specyfikacją atrybutu *TESTREG_CONNECTION_MAP_GROUP*.

Przykładowa struktura BIST składa się ze ścieżki brzegowej oraz grup połączeń tworzących dodatkowe elementy struktury testowania, tak jak pokazano to na wydruku 4.1. Poniżej przedstawiono fragment pliku konfiguracyjnego, który opisuje projekt takiej struktury. Wszystkie istotne elementy tego pliku zostały omówione powyżej.

Na podstawie pliku konfiguracyjnego tworzona jest macierz połączeń, opisująca strukturę diagnostyczną. W opisie struktury występują elementy, które będą optymalizowane algorytmem genetycznym. Dla normalnych projektów macierze te mogą zawierać setki elementów. Ponieważ liczba jedynek w macierzy jest niewielka, a liczba optymalizowanych elementów również będzie dużo mniejsza od rozmiaru macierzy w programie, taka macierz przechowywana jest w postaci zoptymalizowanych struktur danych.

4.3.3. Moduły

Bazą weryfikacji struktur testowania są układy ISCAS'89, które są sekwencyjne synchroniczne. Aby opisać funkcję takiego układu zależnością (3.3), należy wyodrębnić funkcję kombinacyjną układu. Elementy pamięci wewnętrznej oraz wejścia i wyjścia układu reprezentowane są przez elementy wektora X. Opis przykładowego układu testowego s27.bench w formacie ISCAS'89 przedstawiono poniżej. Do pełnego opisu wymagana jest lista modelowanych uszkodzeń zawarta w pliku s27.flt.

Wydruk 4.2 Pliki opisujące moduł

s27.bench	s27.flt
# 4 inputs	*G1 /0*
# 1 outputs	*G2 /0*
# 3 D-type flipflops	*G3 /0*
# 2 inverters	*G5 /0*
# 8 gates (1 ANDs + 1 NAND	*G6 /1*
# + 2 ORs + 4 NORs)	*G7 /0*
INPUT(G0)	*G8 /0 /1*
INPUT(G1)	*G9 /0*
INPUT(G2)	*G10 /0 /1*
INPUT(G3)	*G11 /0 /1*
OUTPUT(G17)	*G12 /0 /1*
G5 = DFF(G10)	*G13 /0 /1*
G6 = DFF(G11)	*G14 /0 /1*
G7 = DFF(G13)	*G15 /1*
G14 = NOT(G0)	*G16 /1*
G17 = NOT(G11)	*G17 /0 /1*
G8 = AND(G14, G6)	*G8 -> G15 /0*
G15 = OR(G12, G8)	*G8 -> G16 /0*
G16 = OR(G3, G8)	*G11 -> G6 /0 /1*
G9 = NAND(G16, G15)	*G11 -> G10 /0*
G10 = NOR(G14, G11)	*G12 -> G13 /0*
G11 = NOR(G5, G9)	*G12 -> G15 /0*
G12 = NOR(G1, G7)	*G14 -> G8 /1*
G13 = NOR(G2, G12)	*G14 -> G10 /0*

Pliki w formacie ISCAS'89 podlegają kompilacji. Poniżej podano przykłady elementów tej funkcji, z podaną informacją o modelowanych uszkodzeniach.

W procesie obliczeniowym wykorzystywane są rejestry XMM procesora PENTIUM. Dzięki temu podczas jednokrotnego przeliczenia funkcji możliwe jest modelowanie do 127 uszkodzeń – na jednym bicie wyliczana jest poprawna wartość. Modelowanie uszkodzeń bazuje na modelowaniu kodu, znanego również z programowania gier komputerowych. Modelowanie kodu polega na wykonaniu wstawek kodu zmieniających funkcję obliczeniową. Metoda ta daje

Tabela 4.6. Przykłady modelowania składowych funkcji modułu

Opis	ASM
INPUT(G1)	*MOVAPS XMM0,[EDX+_primary_input_2];*
G1 /0	*MOVAPS [EDX+G1],XMM0; # sa0*
G8 = AND(G14, G6)	*MOVAPS XMM0,[EDX+G14]; # sa1*
G14 -> G8 /1	*ANDPS XMM0,[EDX+G6];*
G8 /0 /1	*MOVAPS [EDX+G8],XMM0; # sa0, sa1*
OUTPUT(G17)	*MOVAPS [EDX+G17],XMM0; # sa0, sa1*
G17 /0 /1	*MOVAPS XMM0,[EDX+G17];*
	MOVAPS [EDX+_primary_output_1],XMM0;
G5 = DFF(G10)	*MOVAPS XMM0,[EDX+_input_from_dff_1];*
G5 /0	*MOVAPS [EDX+G5],XMM0; # sa0*
	MOVAPS XMM0,[EDX+G10];
	MOVAPS [EDX+_output_to_dff_1],XMM0;

najlepsze rozwiązania pod względem optymalizacji kodu. Skompilowana funkcja jest plikiem binarnym, zawierającym wszystkie struktury danych, binarny kod funkcji oraz struktury modelowania uszkodzeń.

Rozdział 5

Podsumowanie

Funkcjonowanie organizmów żywych oparte jest na mechanizmach optymalizacji. Inspirują one do próby ich wykorzystania w technikach obliczeniowych. Aby uzyskać informacje o obrazie, oko ludzkie wykonuje ruch, który przechodząc przez różne punkty obrazu koncentruje się na szczegółach (wykazano, że ma on charakter fraktalny). Można stwierdzić, że ruch ten ma fazy eksploracyjną i eksploatacyjną. Codziennie korzystając z tego zmysłu doświadczamy jego potęgi. W rozdziale 2. monografii zdefiniowano algorytm optymalizacyjny, mający właściwość podobną do mechanizmu obserwacji. Jego zasada działania opiera się na koewolucji dwóch systemów cząstek, z których jeden ma funkcję eksploracyjną, a drugi eksploatacyjną. Podział ten jest umowny i wynika z analizy zachowania systemów cząstek. Koewolucja ma silny wpływ na zachowanie algorytmu – silniejszy niż wpływ środowiska.

Metaforami tego algorytmu są: algorytm S-IMM, zaprezentowany w rozdziale 2.4 oraz S-PSO, omówiony w rozdziale 2.5. Algorytmy te łamią kanony przyjęte dla typowych algorytmów w tych grupach. Różnice między nimi wynikają z realizacji funkcji odpowiedzialnej za ruch cząstek. Łączą one w sobie wysoką efektywność losowego przeszukiwania przestrzeni rozwiązań połączoną z przeszukiwaniem wynikającym z ruchu cząstek. W przypadku algorytmu S-IMM opracowano efektywny system mutacji, natomiast w przypadku algorytmu S-PSO – scenariusze zachowań, bazujące na funkcjach współczynników wagowych. Prezentowany w pracy (w rozdziale 2.4) grupowy system immunologiczny jest odpowiednikiem systemu wielopopulacyjnego. W rozdziale 2.4 zostały zdefiniowane wszystkie pojęcia, związane ze współdziałaniem systemów immunologicznych – głównie dotyczących wymiany informacji, która bazuje na różnych rodzajach przekazywania materiału genetycznego. Algorytm S-IMM implementuje mechanizm autoimmunizacji, występujący w systemach naturalnych. Można stwierdzić, że omawiany grupowy system immunologiczny reprezentuje model sztucznego życia. Algorytm S-PSO również opisuje sytuację wziętą z życia – strategię osaczania. Rezultaty badań właściwości grupowego systemu immunologicznego przedstawiono w rozdziałach 2.4.1 i 2.4.2. Wyniki te ilustrują zachowanie omawianego algorytmu oraz uwidaczniają różnice wynikające z różnych sposobów wymiany materiału genetycznego dla grupy systemów immunologicznych. Ponadto, rezultaty badań uzyskane w rozdziale 2.4.2, jak również twierdzenie 2.3 po-

twierdzają słuszność koncepcji. Zaprezentowany algorytm oraz jego metafory mają nieudokumentowaną do tej pory w literaturze światowej właściwość eyetracking. Ponieważ kryterium stopu dla tej grupy algorytmów określane jest na podstawie ich zachowania, więc parametry omówione w rozdziale 2.4.1 dają znacznie lepszą możliwość kontroli jego pracy – w porównaniu z innymi algorytmami. W rozdziałach 2.6, 2.7, 2.8 porównano działanie algorytmów S-PSO i S-IMM z innymi algorytmami w odniesieniu do środowisk testowych. Badania te wykazały ich dobre właściwości, jednak szczegółowa analiza parametrów pracy algorytmów pokazuje kolejną ich cechę – brak istotnego wpływu środowiska na ich pracę. Bezpośrednie porównanie algorytmów pokazuje uzyskanie zbliżonych rezultatów dla podobnych parametrów pracy, co świadczy o równoważności tych algorytmów. Proponowane algorytmy mają jeszcze jedną, ważną zaletę – niski koszt pracy, który przypada na jedną iterację.

W badaniach właściwości algorytmów wykorzystano analizy fraktalną i multifraktalną. Przeprowadzone eksperymenty potwierdzają ich przydatność w tym zakresie.

Algorytmy tej grupy przeważnie stosowane są w zadaniach, dla których reguły optymalizacji nie mają charakteru deterministycznego, a proces optymalizacji jest bardzo złożony. Do takich zadań można zaliczyć optymalizację struktur diagnostycznych układów cyfrowych. Szczególnym wyzwaniem staje się optymalizacja struktury wielomodułowej.

Ze względu na charakter wbudowanego testera, istotny staje się sposób opisu architektury takiego układu – zatem model omówiony w rozdziale 4.1.1 może mieć znaczenie praktyczne. Zaproponowana architektura struktury diagnostycznej jest zgodna ze standardem ścieżki brzegowej. Łamie ona jednak przyjęty kanon realizacji takiej struktury. W rozdziale 4. wykazano, że właściwości struktury diagnostycznej (model rejestru liczącego) podczas testowania zależą od ścieżki w grafie przejść, opisującej jej zachowanie, nie zaś od właściwości poza tym zakresem. W rozdziale tym wykazano również, że uzyskanie dobrych właściwości kompakcji i generowania testów w zakresie testowania może być zrealizowane przez zakłócanie. Badania zaprezentowane w rozdziale 4.1 wykazały, że stworzenie takich warunków jest możliwe wewnątrz struktury diagnostycznej. Dowodzą one również, że architektura struktury diagnostycznej jest zależna od funkcji modułu – jego cech. Daje to możliwość definiowania dodatkowych reguł budowy struktury diagnostycznej, a także ma wpływ na efektywność pracy algorytmu optymalizacyjnego. Analizując proces optymalizacji struktury diagnostycznej zaobserwowano wzrost skuteczności diagnostycznej, połączony ze zmianą architektury tej struktury testowania oraz to, że istnieje wiele architektur o podobnej skuteczności diagnostycznej. Ponadto, obserwowany jest wzrost skuteczności diagnostycznej połączony z redukcją elementów tej struktury. Jedynie zmiany architektury mogą wewnątrz struktury testowania efektywnie tworzyć strumienie zakłócające pracę obszarów

rejestru – zgodnie z twierdzeniem 4.5. Uzyskane rezultaty badań wskazują (zgodnie z twierdzeniem 4.3), że istnieje wiele równoważnych pod względem diagnostycznym rozwiązań, gdyż uzyskanie ściśle skorelowanego strumienia byłoby znacznie trudniejsze. Proponowana w pracy struktura diagnostyczna charakteryzuje się dużą efektywnością wykrywania uszkodzeń oraz może być stosowana w układach wielomodułowych, co zostało wykazane przy wykorzystaniu układów testowych ISCAS'89. Jest ona zgodna ze standardem IEEE 1149.1 i pozwala na minimalizację kosztów, związanych z projektowaniem struktur BIST. Przedstawione rozwiązanie łamie kanon stosowania sztywnej architektury wbudowanego testera.

Zaproponowana w rozdziale 4.3 metoda opisu struktury diagnostycznej oraz jej optymalizacji są zgodne z koncepcją języka BSDL. Pozwalają one na opis dowolnej struktury testowania układu jedno- i wielomodułowego (między innymi takich, jakie pokazano w rozdziale 3.1). Architektura ta pozwala na implementację punktu testowego wewnątrz testowanego układu [216] oraz klas optymalizacji dla grup, które byłyby odpowiedzialne za specyfikę optymalizacji w obrębie grupy. Rozwój tego narzędzia pozwoliłby na jego wykorzystanie nie tylko w zakresie badawczym. Jedynym ograniczeniem rozwoju i stosowania omawianej metody jest moc obliczeniowa systemu komputerowego, dla którego będzie przeznaczony i to stanowiło istotny problem prowadzenia badań. Jednak drugim aspektem ograniczającym wydawał się algorytm optymalizacji. Poszukiwania nowych rozwiązań pozwoliły na stworzenie pierwszej części pracy – tu również moc obliczeniowa komputera okazała się mieć istotne znaczenie w zakresie prowadzonych badań. Pomimo to obie części pracy zostały zrealizowane w wyczerpującym zakresie.

W dodatku A zaprezentowano możliwości wykorzystania wielowartościowych łańcuchów funkcyjnych w reprezentacji genomu. Jak już wspomniano, koncepcja wielowartościowych łańcuchów funkcyjnych była rozwijana przez J. Gawina. Połączenie algorytmów genetycznych i łańcuchów funkcyjnych daje możliwości zbliżone do programowania genetycznego. Ze względu na oryginalny charakter tego rozwiązania zdecydowano się je zamieścić w niniejszej pracy.

Dodatek A

Wielowartościowe łańcuchy funkcyjne

Pomiędzy algorytmami genetycznymi, programowaniem genetycznym i podejściem immunologicznym występują istotne różnice, mogące decydować o ich przydatności do rozwiązania określonego problemu.

Po dokonaniu próby wyboru metody modelowania funkcji okazało się, że najlepsze cechy mają łańcuchy funkcyjne, których szczególnym przypadkiem są łańcuchy binarne. Z kolei łańcuchy binarne są szczególną formą zapisu binarnych drzew decyzyjnych (*Binary Decision Diagram*). Łańcuchy funkcyjne pozwalają na modelowanie funkcji na poziomie bramek i przesłań międzyrejestrowych. Pozwalają one również na opis układów kombinacyjnych, sekwencyjnych oraz mikroprogramowalnych. Łańcuchy funkcyjne mogą być wykorzystane do symulacji i generowania testów oraz syntezy układów. Powyższe cechy łańcuchów funkcyjnych skłaniają do podjęcia próby wykorzystania ich w narzędziach optymalizacji, stosujących algorytmy ewolucyjne.

A.1. Składnia wielowartościowego łańcucha funkcyjnego

W opisie funkcji wielowartościowych łańcuchów funkcyjnych występują dwa typy ogniw o następujących formatach [80], [81]:

- ogniwa nieterminalne (zmienne) – *liczba_odgalezien*|*nazwa* (podłańcuchy następujące po tej zmiennej odpowiadają kolejnym wartościom zmiennej od $n-1$ do 0).

  Zmienną łańcucha może być:

1) identyfikator zmiennej sterującej układu (np. CS),
2) pojedynczy bit dowolnego pola układu (np. $FLAG\,[2]$),
3) warunek w postaci porównania wartości dowolnego pola układu z liczbą lub z wartością innego pola (np. $?Q = 15$),
4) zmiana wartości jednego z pól układu (np. $?\,[C]$),

- ogniwa terminalne – $=$ *wyniki*.

Pozostałe symbole używane w opisie wielowartościowych łańcuchów funkcyjnych:

_ lub *(wyrazenie)* – dotychczasowa wartość pola opisywanego łańcuchem,

$\sim$ – negacja składnika,

$(INPUT, H\_IMP)$ – określenie stanu pola układu dla pól dwukierunkowych,

$INPUT!nazwa\_funkcji$ – wywołanie funkcji.

Liczbę odgałęzień i-tego ogniwa L określamy w następujący sposób:

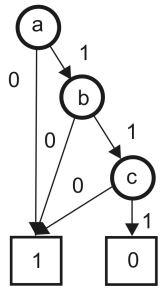
$$odgalezienie\,(L_i) = \begin{cases} n & \Leftrightarrow L_i \in nazwazmiennej \\ 0 & \Leftrightarrow L_i \in wynik \end{cases} \tag{A.1}$$

Waga i-tego ogniwa:

$$waga\,(L_i) = \begin{cases} waga\,(L_{i-1}) + odgalezienie\,(L_i) - 1 \Leftrightarrow \\ \qquad\qquad\qquad\qquad \Leftrightarrow L_i \in N \cup \{0\} \\ 1 \qquad\qquad\qquad\qquad\quad \Leftrightarrow \quad i = -1 \end{cases} \tag{A.2}$$

Przykład reprezentacji funkcji $y =\sim abc$ w zapisie wielowartościowych łańcuchów funkcyjnych i za pomocą binarnego diagramu decyzyjnego – BDD jest omówiony poniżej.

Na rysunku A.1 przedstawiono wielowartościowy łańcuch funkcyjny w postaci reprezentującej łańcuch binarny oraz drzewo binarne, opisane tym łańcuchem.



$$Y = a, b, c, 0, 1, 1, 1$$

Rysunek A.1. Binarny diagram decyzyjny opisujący drzewo binarne

Przykład wielowartościowego łańcucha funkcyjnego:

$$Lan = 3|A, B, = 4, = 5, = 6, C, = 7, = 8 \tag{A.3}$$

W pracy [81] przedstawiono następujące algorytmy działań na łańcuchach funkcyjnych: wydzielenia podłańcucha, redukcji łańcucha, wycięcia podłańcucha, podstawienia łańcucha, równoważenia dwóch łańcuchów, operacji logicznej na dwóch łańcuchach, wyszukania w łańcuchu wyniku odpowiadającego zbiorowi aktualnych wartości n-zmiennych.

Tabela A.1. Interpretacja zapisu wielowartościowego łańcucha funkcyjnego

Indeks	0	1	2	3	4	5	6	7
Opis	$3\|A$	B	$=4$	$=5$	$=6$	C	$=7$	$=8$
Odgałęzienia	3	2	0	0	0	2	0	0
Wagi	3	4	3	2	1	2	1	0

A.2. Operatory genetyczne na łańcuchach funkcyjnych

Wykorzystanie łańcuchów funkcyjnych w narzędziach projektowych, stosujących algorytmy ewolucyjne, powoduje konieczność rozszerzenia działań o operacje genetyczne takie, jak mutacja i krzyżowanie. Poniżej przedstawiono ich graficzną interpretację oraz opis ich algorytmów.

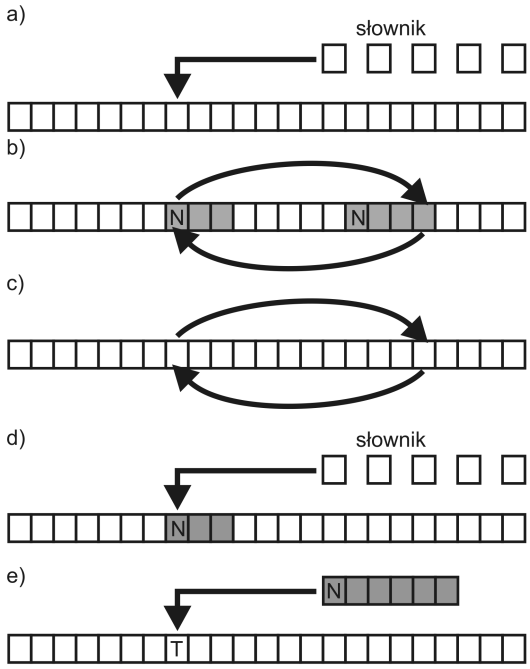
Działanie operatorów mutacji, przedstawione na rysunku A.2, można zdefiniować w następujący sposób:

a) wylosowanie ogniwa (terminalnego lub nieterminalnego) w łańcuchu funkcyjnym i zamiana na ogniwo z odpowiadającego mu słownika ogniw (warunek: ogniwa nieterminalne muszą mieć taką samą liczbę odgałęzień. Operacja jest typowa dla AG i PG),
b) wylosowanie dwóch ogniw nieterminalnych w łańcuchu funkcyjnym, wydzielenie podłańcuchów tych ogniw i ich zamiana (warunek: podłańcuchy nie mogą być zagnieżdżone – operacja jest typowa dla PG),
c) wylosowanie dwóch ogniw tego samego typu (terminalnych lub nieterminalnych) w łańcuchu funkcyjnym i ich zamiana (warunek: ogniwa nieterminalne muszą mieć taką samą liczbę odgałęzień – operacja jest typowa dla AG i PG),
d) wylosowanie ogniwa nieterminalnego w łańcuchu funkcyjnym i zastąpienie wydzielonego podłańcucha tego ogniwa symbolem terminalnym (operacja jest typowa dla PG),
e) wylosowanie ogniwa terminalnego w łańcuchu funkcyjnym i zastąpienie go utworzonym losowo podłańcuchem (operacja jest typowa dla PG).

Działanie operatorów krzyżowania, przedstawione na rysunku A.3, można zdefiniować w następujący sposób:

- Krzyżowanie łańcuchów niezrównoważonych (operacja jest typowa dla PG).

a) wylosowanie ogniwa terminalnego w pierwszym łańcuchu funkcyjnym, wylosowanie ogniwa nieterminalnego i wydzielenie podłańcucha w drugim łańcuchu funkcyjnym, następnie zamiana ogniwa terminalnego z podłańcuchem,
b) wylosowanie ogniwa (terminalnego lub nieterminalnego) w pierwszym łańcuchu funkcyjnym, wylosowanie tego samego typu ogniwa w drugim łań-



Rysunek A.2. Operatory mutacji dla wielowartościowych łańcuchów funkcyjnych (opis w tekście)

cuchu funkcyjnym i zamiana ogniw (warunek: ogniwa nieterminalne muszą mieć taką samą liczbę odgałęzień),

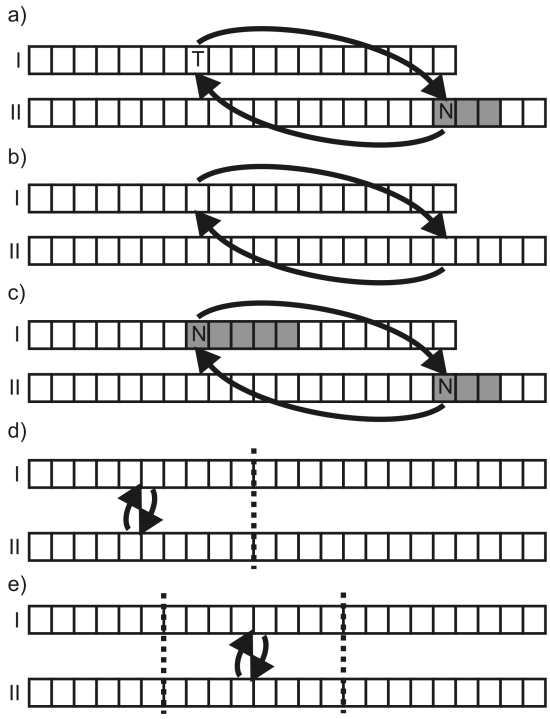
c) wylosowanie ogniwa nieterminalnego i wydzielenie podłańcucha w pierwszym łańcuchu funkcyjnym, wylosowanie ogniwa nieterminalnego oraz wydzielenie podłańcucha w drugim łańcuchu funkcyjnym, zamiana.

- Krzyżowanie zrównoważonych łańcuchów (operacja jest typowa dla AG):

d) krzyżowanie jednopunktowe – wylosowanie ogniwa będącego punktem krzyżowania i zamiana odpowiadających sobie części łańcuchów funkcyjnych,

e) krzyżowanie dwupunktowe – wylosowanie dwóch ogniw będących punktami krzyżowania i zamiana środkowych części łańcuchów funkcyjnych.

Zaprezentowane powyżej operacje genetyczne mają cechy charakterystyczne zarówno dla algorytmów genetycznych, jak i programowania genetycznego. Zastosowanie łańcuchów funkcyjnych daje znacznie większe możliwości operowania genomami w stosunku do genomów opisanych słowami binarnymi lub drzewami. Mogą one być z powodzeniem stosowane w systemach immunologicznych.



Rysunek A.3. Operatory krzyżowania dla wielowartościowych łańcuchów funkcyjnych (opis w tekście)

A.3. Opis funkcji układu

Opis funkcji układu zawiera elementy specyficzne dla języka VHDL. Port opisuje połączenie układu ze środowiskiem:

entity XOR
port (
A: in bit;
B: in bit;
C: out bit
)

Część opisująca zachowanie układu może zawierać postać łańcucha funkcyjnego:

behaviour
begin
C:= A,B,=0,=1,B,=1,=0;
end

Funkcja układu może mieć postać niezdefiniowaną:

C:= <notdefined>;

Zachowanie układu wynika również z sekwencji testującej, która jest zapisywana w postaci tabeli:

Tabela A.2. Sekwencja testująca

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Opis funkcji układu w sekcji *behaviour*, a także odpowiedź układu w kolumnie C są zależne od działania programu optymalizującego. Optymalizacja funkcji układu bazuje na algorytmie genetycznym i bibliotece *GAlib*. Funkcja dostosowania opiera się na rezultatach wykonania sekwencji testującej.

A.4. Przykładowe funkcje

Pierwszą analizowaną funkcją jest XOR, której opis przedstawiono powyżej. Program pozwolił na uzyskanie następujących rozwiązań:

$C := A, =\sim B, = B;$

$C := A, B, = 0, ?1 < 0, = 0, = 1, B, = 1, = 0;$

$C := A, B, = 0, B, = B, = 1, = B;$

$C := A, B, 0, 1, B, 1, 0;$

$C := B, A, = 0, B, = 1, = 0, A, = 1, A, = 1, = 0;$

$C := B, ?1 <\sim 1, = 1|1, A, = 0, = 1, = 1| \sim 1;$

Wszystkie przedstawione rozwiązania realizują funkcję XOR, natomiast pierwszy zapis reprezentuje postać optymalną.

Drugim przykładem jest funkcja kombinacyjna FUN, opisana w następujący sposób:

```
entity FUN
port (
 A: in bit;
 B: in bit;
 C: in bit;
 D: in bit;
 E: out bit
)
behaviour
begin
 E:= <notdefined>;
end
```

Sekwencja testująca opisana jest w następujący sposób:

Tabela A.3. Sekwencja testująca

A	B	C	D	E
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	1	0

Tabela A.4. Uzupełnienie sekwencji testującej z tabeli A.3 do testu wyczerpującego

A	B	C	D	E
0	1	1	0	0
1	0	1	0	1
1	0	1	1	1
1	1	1	0	0

Przedstawiony test jest testem funkcjonalnym, gdyż nie zawiera wszystkich możliwych stanów wejściowych. Dla tak opisanej funkcji układu uzyskano następujące rezultaty:

$E := C, B, = 0, = 1, A, = D, = 0;$
$E := C, B, = 0, = 1, A, = D, = A;$
$E := C, =\sim B, D, = A, = D;$
$E := A, =\sim C\&D, B, = 0, = C;$
$E := C, =\sim B, A, = D, = 0;$
$E^* := C, =\sim B, = D\&A;$
$E := D, C, =\sim B, = A, = C;$
$E := C, =\sim B, A, = D, =\sim 0\&C;$

Zapis $E^* := C, =\sim B, = D\&A;$ jest zapisem optymalnym.

Przedstawiony powyżej test będzie realizował testowanie wyczerpujące, gdy zostanie uzupełniony w sposób przedstawiony poniżej.

Dla tak określonej funkcji uzyskano następujące rezultaty:

$E^* := C, =\sim B, A, = D, = A;$

$E^* := C, =\sim B, A, = D, =\sim 1;$

$E^* := A, C, =\sim B, = D, =\sim B\&C;$

$E^* := C, B, =\sim B, = 1, = A\&D;$

$E^* := C, =\sim B, D, = A, = 0;$

$E := C, A, =\sim B, B, = A, = 1, D, = A, = B\&D;$

$E := C, B, =\sim 1, = C, A, C, =\sim B, = D, = A;$

$E^* := C, =\sim B, A, = D\&1, = A;$

(* – wyrażenie posiadające minimalną liczbę węzłów)

Tabela A.5. Łańcuchy funkcyjne dla testów funkcjonalnego i wyczerpującego

Test wyczerpujący	Test funkcjonalny
$E := C, =\sim B, A, = D, =\sim 1;$	$E := C, B, = 0, = 1, A, = D, = 0;$
$E := C, =\sim B, A, = D, = A;$	$E := C, \sim B, A, = D, = 0;$
$E := C, =\sim B, D, = A, = 0;$	$E := C, B, = 0, = 1, A, = D, = A;$
	$E := C, =\sim B, D, = A, = D;$

Zestawienie rezultatów spełniających warunek dla testów funkcjonalnego i wyczerpującego przedstawiono w tabeli A.5. Wielowartościowe łańcuchy funkcyjne w połączeniu z algorytmem genetycznym są równoważne programowaniu genetycznemu.

Dodatek B

Analizy fraktalna i mutifraktalna

Analizy fraktalna i mutifraktalna są stosowane w wielu dziedzinach nauki i techniki. Przykłady zastosowania analizy fraktalnej w odniesieniu do algorytmów genetycznych prezentowane są w pracach [122], [131], [132], [138]. Fraktalny charakter prostego algorytmu genetycznego wykazano w pracy [122]. W opracowaniach [131] i [132] wymiar fraktalny stosowano w analizie genomu. Analizę trajektorii osobników w algorytmie genetycznym, bazującą na wymiarze fraktalnym omawia praca [138]. W niniejszej monografii analizy fraktalna i multifraktalna są stosowane do oceny zachowania algorytmów przez badanie rozkładów elementów próby w przestrzeni rozwiązań. Poniżej zdefiniowano podstawowe pojęcia, leżące u podstaw tych analiz. W pracy [160] Mandelbrot zdefiniował wymiar fraktalny:

Definicja B.1 (wymiar fraktalny)**.** *Obiekt ma wymiar fraktalny (albo indeks Hausdorff'a) D_f, zdefiniowany jako:*

$$D_f = \lim_{l \to 0} \frac{\ln \mathfrak{N}(l)}{-\ln l}, \tag{B.1}$$

gdzie $\mathfrak{N}(l)$ liczba kul o promieniu l, potrzebna do pokrycia zbioru w przestrzeni d-wymiarowej.

D_f jest wymiarem Hausdorffa zbioru i na podstawie zależności (B.1) wyrażenie przedstawione poniżej jest także poprawne:

$$\mathfrak{N}(l) \propto (l)^{-D_f} \quad \text{dla} \quad l \to 0. \tag{B.2}$$

Pojęcie fraktala można zdefiniować w różny sposobów [21], [22], [160]; na przykład:

Definicja B.2 (fraktal)**.** *Fraktal jest obiektem, który ma niecałkowity wymiar fraktalny.*

Fraktale podlegają prawu opisanemu równaniem (B.1), niezależnie od zakresu skali. Przez pokrywanie obiektu pudełkami zmieniającymi wielkości i licząc pudełka, które obejmują obiekt można oszacować wymiar pudełkowy, który jest równoważny wymiarowi fraktalnemu.

Fraktal może zostać scharakteryzowany przez jego wymiar fraktalny D_f, który definiuje miarę, z jaką fraktal wypełnia przestrzeń. Fraktale są homogeniczne (jednorodne), ponieważ składają się z geometrycznie powtórzonych

obiektów dla każdej skali. Dla takich obiektów wymiar fraktalny jest taki sam, niezależnie od skali.

Niestety, w naturze fraktale nie są jednorodne (homogeniczne). Rzadko występuje identyczny motyw, powtórzony we wszystkich skalach – fraktale są różnorodne (heterogeniczne). Oznacza to, że fraktal może mieć różne wymiary w różnych skalach. Takie właśnie obiekty nazywane są multifraktalami. Nie należy ich jednak mylić z fraktalami mieszanymi, dla których można wyodrębnić trzy wymiary, zależnie od skali. Dla małych skal wyznaczamy wymiar teksturowy, dla dużych – wymiar strukturalny, a dla wszystkich– wymiar ogólny [103]. Należy tu podkreślić, że dwa obiekty mogą mieć ten sam wymiar fraktalny, a jednak wyglądać zupełnie inaczej.

Pojęcie multifraktala oparte jest nie na zbiorach (jak w przypadku fraktali), lecz na miarach związanych z tymi zbiorami. W wyniku pomiarów otrzymujemy nie pojedynczą wartość wymiaru fraktalnego, lecz spektrum wymiarów. Analiza multifraktalna różni się więc zasadniczo od obliczenia zwykłego wymiaru fraktalnego i niesie ze sobą większy zasób informacji. Obszerne omówienie tej tematyki zilustrowane licznymi przykładami przedstawiono między innymi w pozycjach [159], [173] i [174].

B.1. Analiza fraktalna

Analiza fraktalna bazuje na wymiarze fraktalnym. Przestrzeń rozwiązań reprezentowana jest przez macierz, której komórki – pudełka lub hipersześciany – są elementami tej przestrzeni. Dane macierzy reprezentują prawdopodobieństwa wystąpienia elementu próby w danym pudełku (hipersześcianie dla przestrzeni n-wymiarowej). Wymiar fraktalny Minkowskiego-Bouliganda D_{M-B}, który jest stosowany w analizie fraktalnej, można zdefiniować w następujący sposób:

Definicja B.3 (wymiar fraktalny Minkowskiego-Bouliganda)**.** *Niech S będzie n-wymiarową przestrzenią euklidesową z metryką d oraz niech $A \subset S$. Jeśli istnieje:*

$$D_{M-B} = \lim_{\varepsilon \to 0} \left(n - \frac{\log \mathrm{vol}_n A\left(\varepsilon\right)}{\log \varepsilon} \right), \tag{B.3}$$

gdzie vol_n *oznacza objętość w przestrzeni n-wymiarowej oraz*

$$A(\varepsilon) = \bigcup_{x \in A} \left\{ y \in S : d(x,y) < \varepsilon \right\}, \tag{B.4}$$

to wówczas nazywamy ją wymiarem Minkowskiego-Bouliganda zbioru A.

Morfologia matematyczna zajmuje się badaniem struktury geometrycznej danego obiektu w przestrzeni $\mathbb{R}^n$. W rozpatrywanym przypadku analiza zo-

staje ograniczona do przestrzeni $3D$. Obiekt – jego funkcję – można opisać w następujący sposób:

$$\mathfrak{f} : \mathfrak{D} \rightarrow \mathfrak{W}, \tag{B.5}$$

gdzie: $\mathfrak{D}$ dziedzina analizowanego obszaru ($2D$), $\mathfrak{W} \subset \mathbb{R}^+$ przeciwdziedzina.

Analiza ta przebiega analogicznie do przekształcenia morfologicznego obrazu w odcieniach szarości. Do analizy wykorzystywane są elementy strukturalne $\mathfrak{B}$, które charakteryzują otoczenie punktu analizowanego obszaru. Transformacje morfologiczne: $\delta_{\mathfrak{B}}(\mathfrak{f})$ dylatacja obszaru $\mathfrak{f}$ oraz $\varepsilon_{\mathfrak{B}}(\mathfrak{f})$ erozja są wynikiem zależności zachodzących pomiędzy badanym obiektem a sondą [101]. Erozję i dylatację można zdefiniować w następujący sposób:

$$\mathfrak{g} = \varepsilon_{\mathfrak{B}}(\mathfrak{f}) \Leftrightarrow \underset{\mathfrak{p}\epsilon\mathfrak{D}}{\forall} \mathfrak{g}(\mathfrak{p}) = \min_{\mathfrak{b}\epsilon\mathfrak{B}} \{\mathfrak{f}(\mathfrak{p}+\mathfrak{b})\}, \tag{B.6}$$

$$\mathfrak{g} = \delta_{\mathfrak{B}}(\mathfrak{f}) \Leftrightarrow \underset{\mathfrak{p}\epsilon\mathfrak{D}}{\forall} \mathfrak{g}(\mathfrak{p}) = \max_{\mathfrak{b}\epsilon\mathfrak{B}} \{\mathfrak{f}(\mathfrak{p}+\mathfrak{b})\}, \tag{B.7}$$

gdzie $\mathfrak{g}$ – obiekt po przekształceniu morfologicznym.

W równaniu B.3 (dla $n = 3$) $\mathrm{vol}_3 A(\varepsilon)$ może być przedstawione jako różnica pomiędzy obrazami będącymi rezultatem transformacji dylatacji i erozji oraz opisane wyrażeniem $\delta_{\mathfrak{B}}(\mathfrak{f}) - \varepsilon_{\mathfrak{B}}(\mathfrak{f})$. Rezultatem dylatacji jest „rozszerzanie", a erozji „kurczenie" się obiektu. Jeżeli $\mathrm{vol}_3 A(\varepsilon) = 0$, oznacza to brak zróżnicowania obrazu, a wymiar fraktalny osiąga wartość maksymalną. Im większe zróżnicowanie obrazu, tym mniejszą wartość przyjmuje wymiar fraktalny.

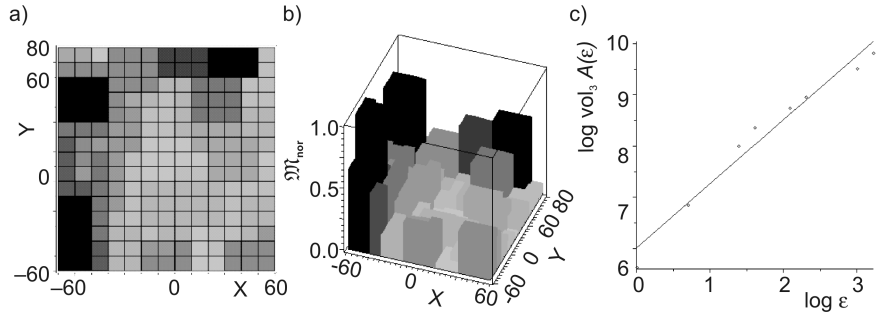
Uzyskane wyniki operacji erozji i dylatacji powinny mieć taki sam rozmiar, jak dane przetwarzane. Aby uniknąć wpływu efektów brzegowych na rezultaty obliczeń, należy rozszerzyć zbiór przetwarzanych danych o wartości skrajnych próbek.

Metoda ta może być wykorzystana do oceny różnic w zachowaniu się obiektów – wysoka wartość wymiaru fraktalnego będzie świadczyła o równomierności ich rozłożenia w przestrzeni rozwiązań (zbliżone wymiary fraktali mogą świadczyć o podobnym zachowaniu się przeciwciał lub antygenów). Należy tu jednak przypomnieć, że taki sam wymiar fraktalny mogą mieć obiekty o całkiem innym wyglądzie. Celem wizualizacji, dane zostały poddane normalizacji:

$$\mathfrak{M}_{i,norm} = 1 - \frac{\mathfrak{M}_i - \min(\mathfrak{M})}{\max(\mathfrak{M}) - \min(\mathfrak{M})}, \tag{B.8}$$

gdzie: $\mathfrak{M}$ – macierz próby, $\mathfrak{M}_i$ – i-ty element macierzy próby, $\mathfrak{M}_{i,norm}$ – i-ty element znormalizowanej macierzy próby, min, max – funkcje określające minimalną i maksymalną wartości macierzy próby.

Na rysunku B.1 przedstawiono dane po przekształceniach morfologicznych $\delta_{\mathfrak{B}}(\mathfrak{f}) - \varepsilon_{\mathfrak{B}}(\mathfrak{f})$, jako znormalizowaną macierz w postaci mapy $2D$ (rysunek B.1a), natomiast jako diagram $3D$ na rysunku B.1b. Sposób wyznaczania



Rysunek B.1. Obraz danych 2D (a) i 3D (b) po przekształceniach morfologicznych oraz sposób wyznaczania wartości wymiaru fraktalnego (c)

wartości wymiaru fraktalnego z charakterystyki log – log ilustruje rysunek B.1c.

Na wykresie B.1c naniesione są punkty [log vol$_3 A(\varepsilon)$,log (ε)] (dla różnych wartości ε) i wykonana jest aproksymacja linią prostą. Wartość $\lim_{\varepsilon\to 0} \frac{\log \mathrm{vol}_3 A(\varepsilon)}{\log \varepsilon}$ jest nachyleniem prostej. Wymiar fraktalny obliczamy na podstawie wzoru (B.3) dla $n = 3$.

Twierdzenie B.1 (osiąganie maksymalnej entropii informacji)**.** *Wymiar fraktalny D_{M-B} osiąga wartość maksymalną wtedy i tylko wtedy, gdy entropia uzyskana w systemie osiąga wartość maksymalną.*

Dowód. Na podstawie twierdzenia 2.3 dla równomiernego rozkładu elementów próby w przestrzeni rozwiązań $\delta_{\mathfrak{B}}(\mathfrak{f}) - \varepsilon_{\mathfrak{B}}(\mathfrak{f})$ otrzymujemy $\mathrm{vol}_3 A(\varepsilon) = 0$ dla każdej ze skal ε. Ze względu na granicę logarytmu dla wartości 0 równą $-\infty$ musimy rozpatrzyć $\mathrm{vol}_3 A(\varepsilon)$ dążące do zera i stałe dla każdej ze skal, ε czyli $\mathrm{vol}_3 A(\varepsilon) = const$. Z tego wynika, że $\lim_{\varepsilon\to 0} \frac{1}{\log \varepsilon} \cdot \log \mathrm{vol}_3 A(\varepsilon) = 0$. Na podstawie wzoru (B.3) $D_{M-B} = n$. □

B.2. Analiza multifraktalna

Matematyka może zdefiniować wymiar na wiele różnych sposobów, na przykład wymiar: Hausdorffa, topologiczny, euklidesowy i pudełkowy [76], [182]. Miara, dla której D_q zmienia się w zależności od q-tego momentu, jest miarą multifraktalną. Poniższe rozważania mają przybliżyć tę miarę.

Niech μ będzie probabilistyczną miarą samopodobieństwa definiowaną na obiekcie $S \subset \mathbb{R}^n$, gdzie $\mu(B)$ jest miarą probabilistyczną określoną jako prawdopodobieństwo wystąpienia obiektu w pudełku $B_i(l)$ i $\mathfrak{N} \propto \frac{1}{l^2}$ jest liczbą pudełek w siatce.

Uogólniony wymiar fraktalny D_q albo inaczej widmo osobliwości $f(\alpha)$ może zostać policzone przy użyciu techniki zliczania pudełek – α jest wskaźnikiem osobliwości (wykładnikiem Höldera). Przykrywamy obiekt S siatką pudełek $(B_i(l))_{i=1}^{\mathfrak{N}}$ o rozmiarze l. Funkcja podziału (albo q-ty moment) Z_q jest zdefiniowana jako:

$$Z_q(l) = \sum_{\mu(B)\neq 0} [\mu(B)]^q = \sum_{i=1}^{\mathfrak{N}} p_i^q(l). \tag{B.9}$$

Dla obiektów samopodobnych oraz danej liczby rzeczywistej q, $\tau(q)$ może zostać zdefiniowana jako liczba dodatnia, spełniająca warunek:

$$\sum_{i=1}^{\mathfrak{N}} p_i^q r_i^{\tau(q)} = 1, \tag{B.10}$$

gdzie p_i reprezentuje prawdopodobieństwa ($\sum_{i=1}^{\mathfrak{N}} p_i = 1$) dla określonego współczynnika podziału (fragmentacji) r_i.

Funkcja $\tau : \mathbb{R} \to \mathbb{R}$ jest funkcją malejącą oraz:

$$\lim_{q\to-\infty} \tau(q) = \infty \quad \text{i} \quad \lim_{q\to\infty} \tau(q) = -\infty. \tag{B.11}$$

Uogólnione wymiary (wymiary Renyiego) D_q i funkcja skalująca $\tau(q)$ są zdefiniowane jako:

$$\tau(q) = D_q(1-q) = \lim_{l\to 0} \frac{\ln Z_q(l)}{-\ln l}, \tag{B.12}$$

gdzie $\tau(q)$ pełni rolę wymiaru fraktalnego.

Uogólnione wymiary są otrzymane z założenia prawa zachowania funkcji podziału w granicy, gdy $l \to 0$ i $\mathfrak{N} \to \infty$:

$$Z_q \propto l^{D_q(q-1)} \quad \text{lub} \quad Z_q \propto l^{-\tau(q)}. \tag{B.13}$$

Definicja B.4 (uogólniony wymiar fraktalny)**.** *Uogólniony (pudełkowy) wymiar fraktalny D_q, gdzie $q \in \Re$, jest zdefiniowany w następujący sposób:*

$$D_q = \lim_{l\to 0} \frac{1}{1-q} \frac{\ln \sum_{i=1}^{\mathfrak{N}} p_i^q(l)}{-\ln\, l}, \tag{B.14}$$

gdzie: indeks i etykietuje pojedyncze pudełka (hipersześciany dla n-wymiarowej przestrzeni), wielkości l i $p_i(l)$ wskazują względną wagę i-tego pudełka albo prawdopodobieństwo wystąpienia obiektu w pudełku, stąd:

$$p_i(l) = \frac{\mathfrak{N}_i(l)}{\mathfrak{N}}, \tag{B.15}$$

gdzie: $\mathfrak{N}_i(l)$ jest wagą i-tego pudełka i $\mathfrak{N}$ jest całkowitą wagą obiektu.

Miarą multifraktalną nazywać będziemy każdą taką miarę, dla której $D_q \neq const$ względem q.

Kiedy $q = 0$,

$$D_0 = D_f = \lim_{l \to 0} \frac{\ln \mathfrak{N}(l)}{-\ln l}, \tag{B.16}$$

gdzie: $\mathfrak{N}(l)$ jest liczbą pudełek zapewniającą minimalne pokrycie.

Kiedy $q = 1$, może zostać zastosowana reguła L'Hopitala, dając:

$$D_1 = \lim_{l \to 0} \frac{\sum_{i=1}^{\mathfrak{N}} p_i \ln p_i}{-\ln l}. \tag{B.17}$$

D_1 jest znany jako wymiar informacyjny, który określa, jak wzrasta uporządkowanie, gdy $l \to 0$.

D_2 jest znany jako wymiar korelacyjny i wskazuje korelację między parami elementów w każdym pudełku. Uogólnione wymiary D_3, D_4,... są związane z korelacjami między 3, 4 itd. krotnościami elementów w każdym pudełku.

Rozważmy spektrum multifraktalne $f(\alpha)$. Waga p_s segmentu s skalowanego pudełkiem l jest następująca:

$$p_s(l) \propto l^{\alpha_s}, \tag{B.18}$$

gdzie α_s jest przedziałowym wykładnikiem Höldera, zdefiniowanym jako:

$$\alpha_s = \frac{\ln p_s(l)}{\ln l}. \tag{B.19}$$

Liczba segmentów $\mathfrak{N}_s$ skali s o wielkości pudełka l odpowiada:

$$\mathfrak{N}_s(l) \propto l^{f_s}. \tag{B.20}$$

Wykładniki α_s i f_s mogą więc zostać użyte do określenia $f(\alpha)$. W wielu przypadkach $f(\alpha) = \dim_H S_\alpha$ odpowiada wymiarowi Hausdorffa-Besicovicha dla zbioru obiektów należących do S [76]. W większości przypadków spektrum multifraktalne $f(\alpha)$ może zostać wyznaczone z $\tau(q)$ przez transformację Legendre'a, która jest opisana poniżej.

$$f(\alpha) = \inf_{-\infty < q < \infty} (\tau(q) + \alpha q), \tag{B.21}$$

$f(\alpha)$ może zostać wyprowadzone z $\tau(q)$ i vice versa, z tożsamości:

$$f(\alpha(q)) = q \cdot \alpha(q) + \tau(q) = q \cdot \alpha(q) - (q-1) D_q, \tag{B.22}$$

$$\alpha(q) = \frac{\mathrm{d}}{\mathrm{d}q}\tau(q) = \frac{\mathrm{d}}{\mathrm{d}q}(q-1)D_q. \tag{B.23}$$

Wiadomo, że $f(\alpha)$ jest funkcją ściśle wypukłą, a $\alpha(q)$ jest malejącą funkcją zmiennej q. Natomiast $\tau(q)$ to wykładnik korelacyjny (masowy) q-tego rzędu.

W praktyce, aby policzyć $\tau(q)$, używając funkcji podziału, wymagane są trzy następujące kroki:

1. Pokrycie obiektu pudełkami $(B_i(l))_{i=1}^{\mathfrak{N}}$ o rozmiarze l i policzenie odpowiednich miar pudełka $\mu_i = \mu(B_i(l)) = p_i(l)$.
2. Policzenie funkcji podziału Z_q dla różnych wartości l.
3. Jeżeli wykresy $\log - \log$ dla Z_q w zależności do l są liniami prostymi, wtedy nachyleniu linii $\tau(q)$ odpowiada wykładnikowi q.

Podsumowując, $\tau(q)$ i D_q można otrzymać z równań (B.9), (B.12), wartości $f(\alpha)$ natomiast mogą zostać określone tak, jak powyżej albo jako [56]:

$$f(q) = \lim_{l\to 0} \frac{\sum_{i=1}^{\mathfrak{N}} \mu_i(q,l) \ln\, \mu_i(q,l)}{\ln\, l}, \tag{B.24}$$

$$\alpha(q) = \lim_{l\to 0} \frac{\sum_{i=1}^{\mathfrak{N}} \mu_i(q,l) \ln\, p_i(q,l)}{\ln\, l}, \tag{B.25}$$

gdzie $\mu_i(q,l)$ są znormalizowanymi prawdopodobieństwami, określonymi wzorem:

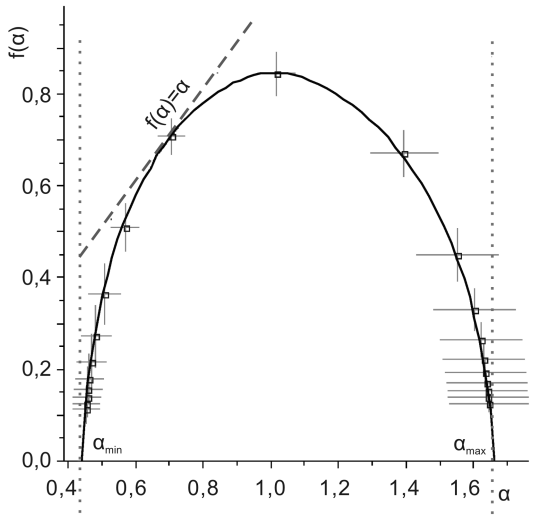
$$\mu_i(q,l) = \frac{p_i^q(l)}{\sum_{j=1}^{\mathfrak{N}} p_j^q(l)}. \tag{B.26}$$

p_i – wyznaczmy na podstawie wzoru (B.15), μ_i obliczamy dla $-10 \leqslant q \leqslant 10$. $f(q)$ i $\alpha(q)$ (wzory B.24 i B.25) wyznaczamy z nachylenia prostej – tak samo, jak przy wymiarze pudełkowym i nanosimy na wykres $f(\alpha)$.

Funkcje $\tau(q)$, D_q i $f(\alpha)$ mają typowe formy dla miar samopodobieństwa. Na przykład, rozważana $f : [\alpha_{min}, \alpha_{max}] \to \mathbb{R}$, wtedy α_{min} i α_{max} są zboczami asymptot funkcji ściśle wypukłej τ. Geometria przekształcenia Legendre'a określa, że f jest ciągła dla $[\alpha_{min}, \alpha_{max}]$ i $f(\alpha_{min}) = f(\alpha_{max}) = 0$. Można pokazać, że $f(\alpha)$ jest styczna do $f(\alpha) = \alpha$ w $q = 1$. Typowa funkcja $f(\alpha)$ oraz pewne jej właściwości są pokazane na rysunku B.2.

Reasumując, z wykresu $f(\alpha)$ można odczytać następujące informacje:

1) $\tau(0) = D_0$ i $q = 0$ odpowiada maksimum – maksymalna wartość $f(\alpha)$ jest miarą wymiaru pojemnościowego D_q,
2) dla $q = 1$, $\tau(q) = 0$ mamy $f(\alpha) = \alpha$; $\frac{\mathrm{d}}{\mathrm{d}\alpha}(f(\alpha) - \alpha) = q - 1 = 0$ – w ten sposób $f(\alpha)$ jest styczny do $f(\alpha) = \alpha$ w $q = 1$, punkt styczności odpowiada wymiarowi informacyjnemu D_1,



Rysunek B.2. Wykres funkcji $f(\alpha)$

3) pozostałe, uogólnione wymiary są rozmieszczone wzdłuż wykresu $f(\alpha) - \alpha$, dla q dodatniego na lewo, dla q ujemnego na prawo od maksimum i określają korelację między i-tymi krotnościami elementów w każdym pudełku,
4) rozpiętość dziedziny $f(\alpha) - \alpha$ ograniczona jest liniami asymptotycznymi do $f(\alpha)$, gdy $q \to \infty$ zbocze ma α_{min}, a gdy $q \to -\infty$ zbocze ma α_{max} i jest ona określona jako $\Delta\alpha = \alpha_{max} - \alpha_{min}$; jest ona miarą grupowania się obiektów (niejednorodności zbioru) – na przykład dla fraktali wszystkie wymiary uogólnione będą równe D_0 (jeśli $\Delta\alpha = 0$, to mamy do czynienia z fraktalem),
5) wykres $f(\alpha) - \alpha$ nie jest wykresem symetrycznym.

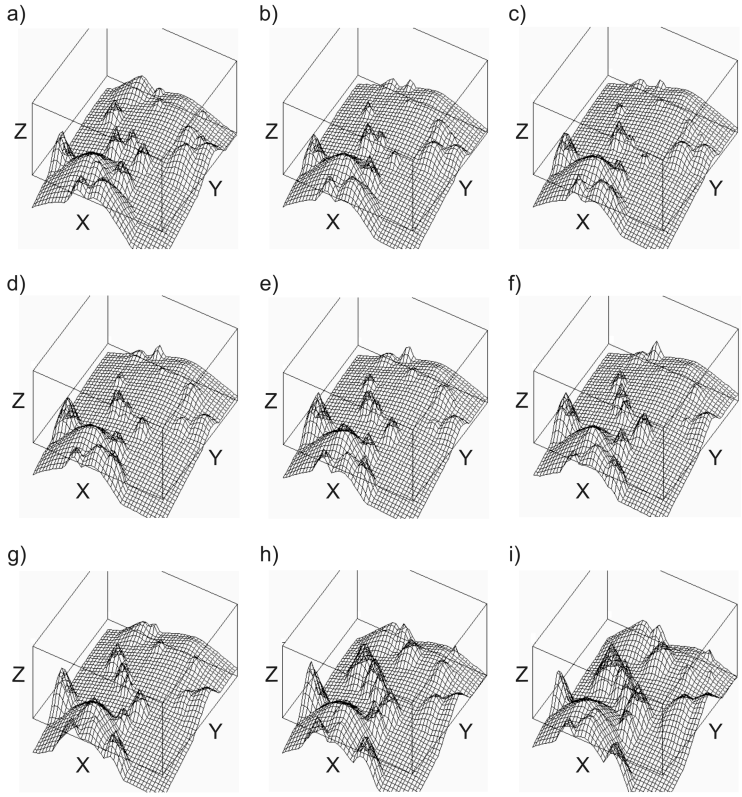
Ze względu na ograniczenia związane ze stosowanym algorytmem w wielu pracach rezultaty wyznaczania spektrum osobliwości są tylko wiarygodne dla wąskiego zakresu wartości q [5].

W monografii analizy fraktalna i multifraktalna są wykorzystane do badania zachowania omawianych algorytmów.

Dodatek C

Środowisko testowe – Moving Peaks Benchmark (MPB)

Branke [39], [40] oraz Morrison i Dejong [168] zaproponowali (niezależnie) podobne, wielowymiarowe środowiska testowe. Aktualne rozwiązanie stanowi rozszerzoną wersję tych środowisk.



Rysunek C.1. Przykład zmiennego środowiska [40]

W zmiennym środowisku algorytm optymalizujący może realizować następujące scenariusze:

- dostosowanie do obecnego rozwiązania, które podlega zmianom,

- „przełączenie" na wcześniejsze nieco gorsze rozwiązanie, które w wyniku zmian stało się optymalnym,
- „przeskok" (przejście doliny) na pojawiające się zupełnie nowe rozwiązanie optymalne.

Funkcję n-wymiarową o m szczytach można sformułować następująco:

$$F\left(\vec{x},t\right)=\max\left(B\left(\vec{x}\right),\max_{i=1\ldots m}P\left(\vec{x},h_i\left(t\right),w_i\left(t\right),\vec{p_i}\left(t\right)\right)\right), \tag{C.1}$$

gdzie: $B(\vec{x})$ – jest funkcją stałą – bazową, a P – to zmienne w czasie funkcje wierzchołków, opisane następującymi parametrami: h_i – wysokość, w_i – szerokość, $\vec{p_i}$ – położenie.

Współrzędne: wysokość i szerokość każdego wierzchołka – są inicjowane w sposób losowy, a następnie zmieniane przez dodanie zmiennej losowej o rozkładzie Gaussa.

Zmiana jednego wierzchołka może być opisana jako:

$$\begin{aligned}\sigma &\in N(0;1),\\ h_i(t) &= h_i(t-1)+height\_severity\cdot\sigma,\\ w_i(t) &= w_i(t-1)+width\_severity\cdot\sigma,\\ \vec{p_i}(t) &= \vec{p_i}(t-1)+\vec{v_i}(t).\end{aligned} \tag{C.2}$$

Wektor przesunięcia $\vec{v_i}$ jest liniową kombinacją wektora losowego $\vec{r}$ oraz poprzedniego wektora przesunięcia $\vec{v_i}(t-1)$ i znormalizowany do długości s:

$$\vec{v_i}(t) = \frac{s}{\left|\vec{r}+\vec{v_i}(t-1)\right|}\left(\left(1-\lambda\right)\vec{r}+\lambda\vec{v_i}\left(t-1\right)\right). \tag{C.3}$$

Położenie i-tego szczytu jest przemieszczane przez wektor $\vec{v_i}$ w kierunku dowolnym ($\lambda = 0$) lub w zależności od kierunku poprzedniego ($\lambda > 0$). Parametr λ pozwala kontrolować tendencję zmian. Losowy wektor $\vec{r}$ jest generowany losowo dla każdego wymiaru i normalizowany względem długości s.

Złożoność funkcji może być łatwo skalowana przez zwiększenie liczby wymiarów, liczby szczytów oraz funkcji bazowej.

Typowe ustawienia środowiska:

- liczba wierzchołków: 10,
- wymiar przestrzeni rozwiązań: 5,
- wysokość wierzchołków: $[30;70]$,
- zmiana wysokości: $7,0$,
- wysokość początkowa: $50,0$,
- szerokość wierzchołków: $[1;12]$,
- zmiana szerokości: $1,0$,
- liczba wyznaczeń funkcji (Change Frequency): 5000,

- długość przesunięcia: $1,0$,
- współczynnik korelacji λ: 0,
- zakres przestrzeni zadania: $[0;100]^n$,
- brak funkcji bazowej.

Dodatek D

Środowisko testowe – Generalized Dynamic Benchmark Generator (GDBG)

Środowisko testowe GDBG [146] bazuje na omówionym MPB, zaprezentowanym w dodatku C. Środowisko to zawiera sześć funkcji oraz sześć scenariuszy, które do zmian środowiska wykorzystują między innymi obrót i przesunięcie. Problem optymalizacyjny jest opisany przez funkcję:

$$F = f(x, \phi, t), \tag{D.1}$$

gdzie: f – funkcja dostosowania, x – rozwiązanie problemu, ϕ – stan środowiska (reprezentowany przez parametry sterujące), t – czas.

Zmiana środowiska realizowana jest przez zmianę parametrów sterujących $\Delta\phi$:

$$\phi(t+1) = \phi(t) \oplus \Delta\phi. \tag{D.2}$$

Zmiana ta będzie więc przedstawiona w następujący sposób:

$$f(x, \phi, t+1) = f(x, \phi \oplus \Delta\phi, t). \tag{D.3}$$

Środowisko testowe realizuje następujące scenariusze zmian parametrów sterujących:

T1 – mały krok:

$$\Delta\phi = \alpha \cdot \|\phi\| \cdot r \cdot \phi_{severity}, \tag{D.4}$$

T2 – duży krok:

$$\Delta\phi = \|\phi\| \cdot (\alpha \cdot sign(r) + (\alpha_{max} - \alpha) \cdot r) \cdot \phi_{severity}, \tag{D.5}$$

T3 – zmiana losowa:

$$\Delta\phi = N(0,1) \cdot \phi_{severity}, \tag{D.6}$$

T4 – zmiana chaotyczna:

$$\phi(t+1) = A \cdot (\phi(t) - \phi_{min}) \cdot (1 - (\phi(t) - \phi_{min}) / \|\phi\|), \tag{D.7}$$

T4 – zmiana powtarzająca się:

$$\phi(t+1) = \phi_{min} + \frac{\|\phi\|}{2} \cdot \left(\sin\left(\frac{2\pi}{P}t + \varphi\right) + 1\right), \tag{D.8}$$

T5 – zmiana powtarzająca się z zaszumieniem:

$$\begin{aligned}\phi(t+1) = \quad & \phi_{min} + \frac{\|\phi\|}{2} \cdot \left(\sin\left(\frac{2\pi}{P}t + \varphi\right) + 1\right) + \\ & + N(0,1) \cdot noisy_{severity},\end{aligned} \tag{D.9}$$

gdzie: $\|\phi\|$ – zakres zmian ϕ, zmiany nasilenia, $\phi_{severity}$ – stała określająca dokładność zmian ϕ, ϕ_{min} – minimalna wartość ϕ, $noisy_{severity} \in (0;1)$ – dokładność cyklicznych zmian zaszumienia, $\alpha \in (0;1)$ i $\alpha_{max} \in (0;1)$ – stałe ustawione na $0,04$ i $0,1$, A – stała z zakresu $(1,0;4,0)$, jeżeli ϕ jest wektorem, to jej początkowa wartość powinna być różna od $\|\phi\|$ dla zmian chaotycznych, P – jest okresem zmian, a φ fazą początkową, r – jest liczbą losową z zakresu $(-1;1)$, $sign(x)$ – jest funkcją znaku zwracającą wartości $\{-1;0;1\}$, $N(0;1)$ – oznacza liczbę losową o rozkładzie normalnym ze średnią zero i odchyleniem standardowym równym jeden. Przestrzeń problemu oraz ekstrema są tworzone przez następujące funkcje:

1. Funkcja rotacji szczytów – rysunek D.1a.
2. Złożenie funkcji sfer – rysunek D.1b.
3. Złożenie funkcji Rastrigina – rysunek D.1c.
4. Złożenie funkcji Griewanka – rysunek D.1d.
5. Złożenie funkcji Ackleya – rysunek D.1e.
6. Hybrydowe złożenie funkcji – rysunek D.1f.

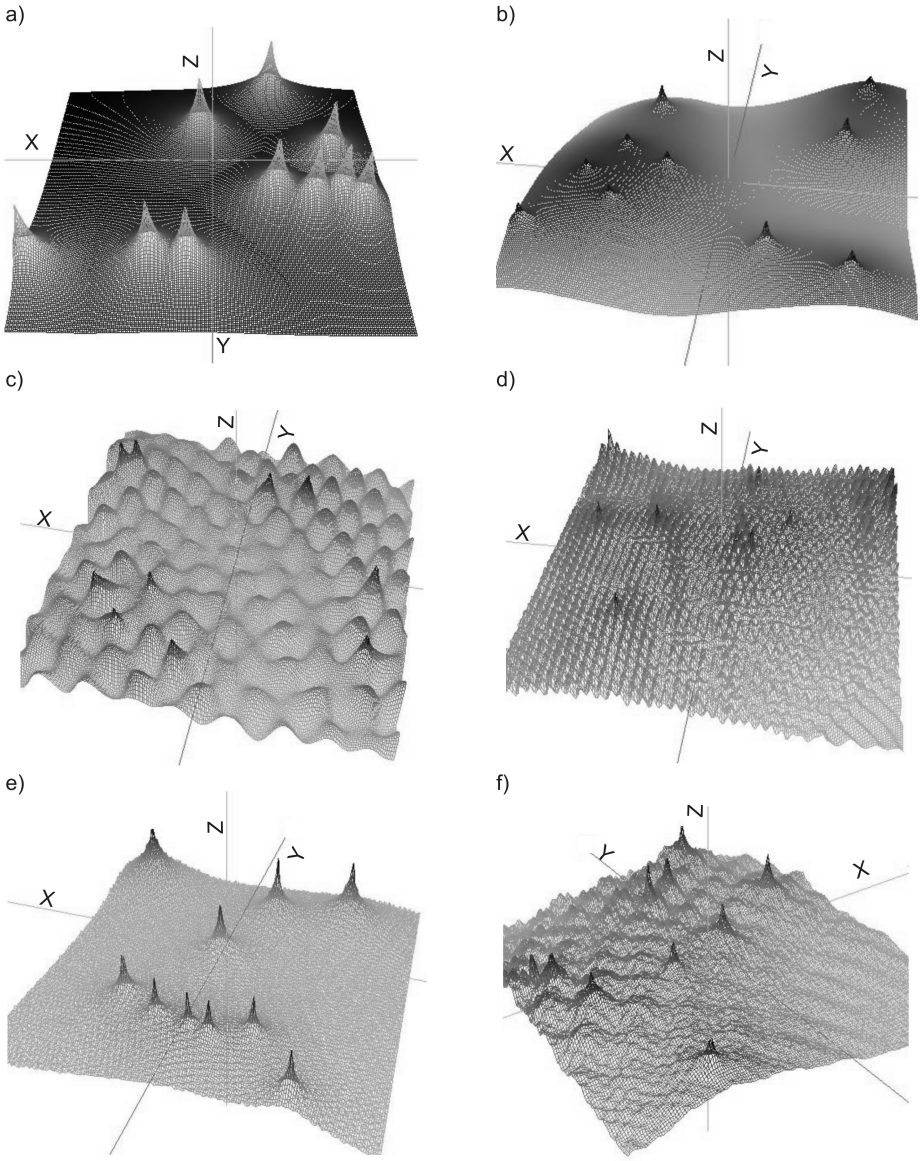
Dla wszystkich funkcji testowych parametry środowiska przyjmują następujące wartości:

- wymiary: $n = 10$, $n \in [5;15]$,
- zakres przestrzeni zadania: $x \in [5;15]^n$,
- częstotliwość zmian: $frequency = 10000 \cdot n$,
- liczba zmian: $num\_change = 60$,
- okres: $P = 12$,
- dokładność cyklicznych zmian zaszumienia: $noisy_{severity} = 0,8$,
- stała dla zmian chaotycznych: $A = 3,67$,
- dokładność kroku: $\alpha = 0,04$,
- maksymalna wartość α: $\alpha_{max} = 0,1$,
- zakres wysokości: $h \in [10;100]$,
- wysokość początkowa: $initial\_height = 50$,
- dokładność wysokości: $\varphi\_h_{severity} = 5,0$.

Chaotyczna inicjalizacja: jeżeli ϕ jest wektorem, to początkowe wartości elementów w ϕ powinny być losowane za pomocą równomiernego rozkładu w ramach $\|\phi\|$.

Dla wszystkich złożeń funkcji:

- Liczba podstawowych funkcji: $m = 10$
- Wskaźnik zbieżności: $\sigma_i = 1,0;\ i = 1,2,...,n$
- $C = 2000$



Rysunek D.1. Funkcje testowe środowiska GDBG [146]

Dodatek E

Stacjonarne środowiska testowe

W celu porównania pracy algorytmów, wykorzystuje się stacjonarne funkcje testowe ([143], [154]). Poniżej przedstawiono ich opis:

1. Funkcja *Easoma*

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left(-\left((x_1-\pi)^2+(x_2-\pi)^2\right)\right), \tag{E.1}$$

gdzie: $-100 < x_i < 100$; $i = 1, 2$; ekstrema lokalne nieudokumentowane w literaturze; minimum globalne: $f(x^*) = -1$ w punkcie $x^* = (\pi; \pi)$.

2. Funkcja *Fichiera B2*

$$f(x) = x_1 + 2x_2 - 0,3\cos(3\pi x_1) - 0,4\cos(4\pi x_2) + 0,7, \tag{E.2}$$

gdzie: $-100 < x_i < 100$, $i = 1, 2$; ekstrema lokalne nieudokumentowane w literaturze; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0)$.

3. Funkcja *Shuberta*

$$\begin{aligned} f(x) = & \left(\sum_{j=1}^{5} j\cdot\cos\left((j+1)\,x_1 + j\right)\right)\cdot \\ & \cdot\left(\sum_{j=1}^{5} j\cdot\cos\left((j+1)\,x_2 + j\right)\right), \end{aligned} \tag{E.3}$$

gdzie: $-10 < x_i < 10$, $i = 1, 2$; 760 minimów lokalnych; 18 minimów globalnych: $f(x^*) = -186,7309$.

4. Funkcja *Goldsteina i Price'a.*

$$\begin{aligned} f(x) = & \left(1+(x_1+x_2+1)^2\cdot\right. \\ & \cdot\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right))\cdot \\ & \cdot\left(30+(2x_1-3x_2)^2\cdot\right. \\ & \cdot\left.\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right), \end{aligned} \tag{E.4}$$

gdzie: $-2 < x_i < 2, i = 1, 2$; 4 minima lokalne; minimum globalne: $f(x^*) = 3$ w punkcie $x^* = (0; -1)$.

5. Funkcja *Zakharova*

$$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0,5i \cdot x_i\right)^2 + \left(\sum_{i=1}^{n} 0,5i \cdot x_i\right)^4, \tag{E.5}$$

gdzie: $-5 < x_i < 10$, $i = 1, 2, ..., n$; ekstrema lokalne nieudokumentowane w literaturze; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; ...; 0)$.

6. Funkcja *Rosenbrocka*

$$f(x) = \sum_{i=1}^{n-1} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right), \tag{E.6}$$

gdzie: $-5 < x_i < 5$, $i = 1, 2, 3, ..., n$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (1; ...; 1)$.

7. Funkcja *Branina RCOS*

$$f(x) = \left(x_2 - \frac{5}{4\pi^2}x_1 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10, \tag{E.7}$$

gdzie: $-5 < x_1 < 10$, $0 < x_2 < 15$; minimum globalne: $f(x^*) = 0,39788$ w punktach $x^* = (-\pi; 12,275), (\pi; 2,275), (9,42478; 2,475)$.

8. Funkcja *De Jounga 3D*

$$f\,(x) = x_1^2 + x_2^2 + x_3^2, \tag{E.8}$$

gdzie: $-5,12 < x_i < 5,12$, $i = 1; 3$, minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; 0)$.

9. Funkcja *Shekela $S_{\,4,n}$ 4D*

$$f\,(x) = -\sum_{i=1}^{n} \left[(x - a_i)^T (x - a_i) + c_i\right]^{-1} \tag{E.9}$$

oraz $x = (x_1, x_2, x_3, x_4)^T$ i $a_i = (a_1, a_2, a_3, a_4)^T$, dla $S_{4,7}$ ($n = 7$ – 7 minimów), minimum globalne: $S_{4,7}\,(x^*) = -10,40294$; dla $S_{4,10}$ ($n = 10$ – 10 minimów), minimum globalne: $S_{4,10}\,(x^*) = -10,53641$, gdzie $-5,12 < x_i < 5,12$, $i = 1, 3$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; 0)$.

i	a_i^T				c_i
1	4,0	4,0	4,0	4,0	0,1
2	1,0	1,0	1,0	1,0	0,2
3	8,0	8,0	8,0	8,0	0,2
4	6,0	6,0	6,0	6,0	0,4
5	3,0	7,0	3,0	7,0	0,4
6	2,0	9,0	2,0	9,0	0,6
7	5,0	5,0	3,0	3,0	0,3
8	8,0	1,0	8,0	1,0	0,7
9	6,0	2,0	6,0	2,0	0,5
10	7,0	3,6	7,0	3,6	0,5

10. Funkcja sfery

$$f(x) = \sum_{i=1}^{D} x_i^2, \tag{E.10}$$

gdzie: $-100 < x_i < 100$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

11. Funkcja *Ackleya*

$$\begin{aligned} f(x) = \quad & -20 \exp\left(-0,2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \\ & - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e, \end{aligned} \tag{E.11}$$

gdzie: $-32,768 < x_i < 32,768$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

12. Funkcja *Griewanka*

$$f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{E.12}$$

gdzie: $-600 < x_i < 600$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

13. Funkcja *Weierstrasa*

$$\begin{aligned} f(x) = \quad & \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right) \right] \right) - \\ & - D \sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k x_i \cdot 0.5\right) \right], \end{aligned} \tag{E.13}$$

gdzie: $a = 0,5$, $b = 3$, $k_{max} = 20$, $-0,5 < x_i < 0,5$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

14. Funkcja *Rastrigina*

$$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right), \tag{E.14}$$

gdzie: $-5,12 < x_i < 5,12$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0, 0, ..., 0)$.

15. Funkcja *Rastrigina* – nieciągła

$$f(x) = \sum_{i=1}^{D} \left(y_i^2 - 10\cos(2\pi y_i) + 10 \right), \tag{E.15}$$

i $y_i = \begin{cases} x_i & |x_i| < \frac{1}{2} \\ \dfrac{\text{round}(2x_i)}{2} & |x_i| \geqslant \frac{1}{2} \end{cases}$ dla $i = 1, 2, ..., D$,

gdzie: $-5,12 < x_i < 5,12$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$, round – zaokrąglenie.

16. Funkcja *Schwefela*

$$f(x) = 418,9829D - \sum_{i=1}^{D} x_i \sin\left(|x_i|^{\frac{1}{2}} \right), \tag{E.16}$$

gdzie: $-500 < x_i < 500$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (420,96; 420,96; ...; 420,96)$.

Opis obrotu

Ortogonalna macierz obrotu:

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,D} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ m_{D,1} & m_{D,2} & \cdots & m_{D,D} \end{bmatrix} \tag{E.17}$$

i $\mathbf{x} = [x_1, x_2, ..., x_D]^T$, $\mathbf{y} = [y_1, y_2, ..., y_D]^T$ oraz $y_i = m_{i,1}x_1, m_{i,2}x_2, ..., m_{i,D}x_D$, gdzie $i = 1, 2, ..., D$.

17. Funkcja *Ackleya* – obrócona

$$\begin{aligned} f(x) = \quad & -20\exp\left(-0,2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2} \right) - \\ & - \exp\left(\tfrac{1}{D}\textstyle\sum_{i=1}^{D}\cos(2\pi y_i) \right) + 20 + e \end{aligned} \tag{E.18}$$

i $\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$, gdzie $-32,768 < x_i < 32,768$.

18. Funkcja Griewanka – obrócona

$$f(x) = \sum_{i=1}^{D} \frac{y_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{y_i}{\sqrt{i}}\right) + 1 \tag{E.19}$$

i $\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$, gdzie: $-600 < x_i < 600$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

19. Funkcja *Weierstrasa* – obrócona

$$\begin{aligned} f(x) = & \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k (y_i + 0,5)\right) \right] \right) \\ & - D \sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k \cdot 0,5\right) \right] \end{aligned} \tag{E.20}$$

i $a = 0,5$, $b = 3$, $k_{max} = 20$, oraz $\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$, gdzie: $-0,5 < x_i < 0,5$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

20. Funkcja *Rastrigina* – obrócona

$$f(x) = \sum_{i=1}^{D} \left(y_i^2 - 10\cos(2\pi y_i) + 10 \right) \tag{E.21}$$

i $\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$, gdzie: $-5,12 < x_i < 5,12$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

21. Funkcja *Rastrigina* – obrócona i nieciągła

$$f(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10 \right) \tag{E.22}$$

i $z_i = \begin{cases} y_i & \text{dla} \quad |y_i| < \frac{1}{2} \\ \dfrac{\text{round}(y_i)}{2} & \text{dla} \quad |y_i| \geqslant \frac{1}{2} \end{cases}$ dla $i = 1, 2, ..., D$,

oraz $\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$, gdzie: $-5,12 < x_i < 5,12$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (0; 0; ...; 0)$.

22. Funkcja *Schwefela* – obrócona

$$f(x) = 418,9829 \times D - \sum_{i=1}^{D} z_i \tag{E.23}$$

i $z_i = \begin{cases} y_i \sin\left(|y_i|^{\frac{1}{2}}\right) & \text{dla } |y_i| \leqslant 500 \\ 0,001\left(|y_i| - 500\right)^2 & \text{dla } |x_i| > 500 \end{cases}$ dla $i = 1, 2, ..., D$,

oraz $\mathbf{y} = \mathbf{y}' + 420,96$; $\mathbf{y}' = \mathbf{M} \cdot (\mathbf{x} + 420,96)$; gdzie: $-500 < x_i < 500$; minimum globalne: $f(x^*) = 0$ w punkcie $x^* = (420,96; 420,96; ..., 420,96)$.

Złożenie funkcji

$$F(x) = \sum_{i=1}^{n} \left\{ w_i \cdot \left[f_i' \frac{x - o_i + o_{i\text{old}}}{\lambda_i \cdot \mathbf{M}_i} + \text{bias}_i \right] \right\} + f_{\text{bias}}, \tag{E.24}$$

$$w_i = \exp\left(-\frac{\sum_{k=1}^{D} (x_k - o_{ik} + o_{ik\text{old}})^2}{2D\sigma_i^2} \right), \tag{E.25}$$

$$w_i = \begin{cases} w_i & \text{dla} \quad w_i = \max(w_i) \\ w_i \cdot (1 - \max(w_i))^{10}) & \text{dla} \quad w_i \neq \max(w_i) \end{cases}, \tag{E.26}$$

$$w_i = \frac{w_i}{\sum_{i=1}^{n} w_i}, \tag{E.27}$$

$$\lambda_i = \sigma_i \frac{X_{\max} - X_{\min}}{x_{\max,i} - x_{\min,i}}, \tag{E.28}$$

$$f_i'(x) = C \cdot \frac{f_i(x)}{|f_{\max i}|}, \tag{E.29}$$

$$|f_{\max i}| = f_i\left(\frac{z}{\lambda_i} \cdot \mathbf{M}_i \right), \tag{E.30}$$

gdzie: $z = X\,\max$, $n = 10$, $D = 10$, $C = 2000$, $f_{\text{bias}} = 0$,
bias $= [0, 100, 200, 300, 400, 500, 600, 700, 800, 900]$,
$[o_1, o_2, ..., o_9] = [0, 0, ..., 0]$ oraz $i = 1, 2, ..., n$;
$\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_n$ – ortogonalne macierze obrotu.

23. Funkcja CF1
    CF1 stanowi złożenie następujących funkcji: $f_1, f_2, ..., f_{10}$ – funkcja sfery oraz $[\lambda_1, \lambda_2, ..., \lambda_{10}] = \left[\frac{5}{100}, \frac{5}{100}, ..., \frac{5}{100}\right]$, $[\sigma_1, \sigma_2, ..., \sigma_{10}] = [1, 1, ..., 1]$,
    gdzie: $-500 < x_i < 500$; minimum globalne: $f(x^*) = 0$.
24. Funkcja CF5

CF5 stanowi złożenie następujących funkcji:

1. $f_{1-2}(x)$ – funkcje Rastrigina,
2. $f_{3-4}(x)$ – funkcje Weierstrasa,
3. $f_{5-6}(x)$ – funkcje Griewanka,
4. $f_{7-8}(x)$ – funkcje Ackleya,
5. $f_{9-10}(x)$ – funkcje sfery,

dla $[\lambda_1, \lambda_2, ..., \lambda_{10}] = \left[\frac{1}{5}, \frac{1}{5}, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}\right]$ i $[\sigma_1, \sigma_2, ..., \sigma_{10}] = [1, 1, ..., 1]$, gdzie: $-500 < x_i < 500$; minimum globalne: $f(x^*) = 0$.

Bibliografia

[1] E.H.L. Aarts, I.H.M. Korst, P.I. Zwietering: *Deterministic and Randomized Local Search.* Eindhoven University of Technology Department of Mathematics and Computing Science, Computing Science Note, 1993.

[2] M. Abramovici, M.A. Brauer, A.D Friedman: *Digital Systems Testing and Testable Design.* IEEE Press, 1990.

[3] V.D. Agrawal: *An Information Theoretic Approach to Digital Fault Testing.* IEEE Transactions on Computers, August 1981, s. 528–587.

[4] V.D. Agrawal, Ch.R. Kime, K.K. Saluja: *A Tutorial on Built-In Self-Test, Part II.* IEEE Design & Test of Computers, June 1993, s. 69–77.

[5] M. Alber, J. Peinke: *Improved Multifractal Box-Counting Algorithm.* Virtual Phase Transitions, and Negative Dimensions, Phys. Rev. E, Vol. 57, No. 5, 1998, s. 5489–5493.

[6] M. Ali, C. Storey: *Topographical Multilevel Single Linkage.* Journal of Global Optimization **5**, 1994, s. 349–358.

[7] R.S. Anderssen, P. Bloomfield: *Properties of the Random Search in Global Optimization.* Journal of Optimization Theory and Applications **16**, 1975, s. 383–398.

[8] P.J. Angeline: *Using Delection to Improve Particle Swarm Optimization.* In: Proc. of the 1998 IEEE Congress on Evolutionary Computation, Piscataway, NJ, USA,.IEEE Press, 1998, s. 84–89.

[9] J. Arabas: *Wykłady z algorytmów ewolucyjnych.* Warszawa WNT, 2001.

[10] J. Arquilla, D. Ronfeldt: *Swarming and the Future of Conflict.* RAND National Defense Research Institute, Santa Monica, CA, U.S. , 2000.

[11] R.B. Ash: *Information Theory.* John Wiley & Sons, New York, 1967.

[12] D. Badura: *Design of Circuit with Self-Test Path. Proc.* FTSD'90, Varna, June 1990.

[13] D. Badura: *Efficiency of Self-Test Path as a Test Pattern Generator and a Test Response Compactor.* Proc. FTCS'89, Baden-Baden, September 1989, s. 365–375.

[14] D. Badura, A. Hławiczka: *A Linear Feedback Shift Register as the Modifier to Optimize Error Masking in-Built Self-Testing.* Informatik Informationen Reporte Akademie der Wissenschaften der DDR, Fault Tolerant Systems and Diagnostics, Special Issue Vol. 2, Berlin, 1988, s. 209–214.

[15] D. Badura, A. Hławiczka: *Condensed Circular Self-Test Path: A Low-Cost Circular BIST.* Proc. European Test Workshop, 1996, s. 65–69.

[16] D. Badura, A. Hławiczka: *Multivalued Logic Circuit Testing with Boundary -Scan Path.* Proc. FTSD'88 Conf., Suhl, June 1988, s. 253–258.

[17] D. Badura: *Self - Test Path in Self-Diagnostic Systems. Proc.* COGNITIV'90, Madryt, November 1990.

[18] D. Badura: *Techniki projektowania samotestowalnych układów i pakietów cyfrowych wykorzystujące rejestry szeregowe z nieliniowym sprzężeniem zwrotnym.* Wydawnictwo Uniwersytetu Śląskiego, 1992.

[19] P.H. Bardell, W.H. McAnney, J. Savir: *Built-In Test for VLSI: Pseudorandom Techniques.* John Wiley & Sons, New York, 1987.

[20] P.H. Bardell, W.H. McAnney: *Self-Testing of Multichip Logic Module.* Proc. International Test Conf., 1982, s. 200–204.

[21] M. Barnsley: *Fractals Everywhere.* Academic Press, San Diego, 1988.

[22] M. Barnsley: *Superfractals.* Cambridge University Press, 2006.

[23] M. Bellos, D, Kagaris, D. Nikolos: *Test Set Embedding Based on Phase Shifters.* Proc. the 4th European Dependable Computing Conf., Lecture Notes In Computer Science, Vol. 2485, 2002, s. 90–101.

[24] H. Bersini, F. Verela: *The Immune Learning Mechanism: Reinforcement and Recruitment and their Applications.* Computing with Biological Metaphors, Chapman Hall, 1994, s. 166–192.

[25] M. Bessaou, P. Siarry: *A Genetic Algorithm with Real-Value Coding to Optimize Multimodal Continuous Functions.* Structural and Multidiscipline Optimization, Vol. 23, 2001, s. 63–74.

[26] B. Betro, F. Schoen: *Sequential Stopping Rules for the Multistart Algorithm in Global Optimisation.* Mathematical Programming, Vol. 38, Number 3, 1987, s. 271–286.

[27] S. Bird, X. Li: *Adaptively Choosing Niching Parameters in a PSO.* In Proc. 2006 Genetic Evol. Comput. Conf., 2006, s. 3–10.

[28] S. Bird, X. Li: *Using Regression to Improve Local Convergence.* IEEE Congress on Evolutionary Computation, CEC, 2007, s. 592–599.

[29] T. Blackwell, J. Branke: *Multi-Swarm Optimization in Dynamic Environments.* Applications of Evolutionary Computing, Springer, Vol. 3005, 2004, s. 489–500.

[30] T. Blackwell, J. Branke: *Multi-Swarms, Exclusion, and Anti-Convergence in Dynamic Environments.* IEEE Transactions on Evolutionary Computation, Vol. 10, 2006, s. 459–472.

[31] T. Blackwell, J. Branke: *Multi-Swarms, Exclusion and Anti-Convergence in Dynamic Environments.* IEEE Transactions on Evolutionary Computation, 2006, s. 51–58.

[32] T. Blackwell, J. Branke, X. Li: *Particle Swarms for Dynamic Optimization Problems.* Swarm Intelligence: Introduction and Applications, Springer, 2008, s. 193–217.

[33] H. Bleeker: *Boundary-Scan Test.* Kluwer Academic Publishers, Boston, 1993.

[34] C.G.E. Boender, A.H.G. Rinnooy Kan, C. Vercellies: *Stochastic Optimization Methods.* In: G. Andreatta, F. Mason, P. Serafini: *Advanced School on Stochastics in Combinatorial Optimization.* World Scientific Pub Co Inc, 1987, s. 94–112.

[35] C.G.E. Boender, A.H.G. Rinnooy Kan: *Bayesian Stopping Rules for Multistart Global Optimization Methods.* Mathematical Programming, Vol. 37, Issue 1, 2, February 1987, s. 59–80.

[36] C.G.E. Boender, A.H.G. Rinnooy Kan: *On When to Stop Sampling for the Maximum.* Journal of Global Optimization, Vol. 1, No. 4, 1991, s. 331–340.

[37] J. Branke: *Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems.* Congr. Evol. Comput., Vol. 3, 1999, s. 1875–1882.

[38] J. Branke, T. Kaussler, C. Schmidth, H. Schmeck: *A Multi-Population Approach to Dynamic Optimization Problems.* 4th Int. Conf. Adaptive Computing in Design and Manufacturing **200**, s. 299–308.

[39] J. Branke: *Evolutionary Optimization in Dynamic Environment.* Kluwer Academic Publisher, Boston – Dordrecht – London, 2002.

[40] J. Branke: *The Moving Peaks Benchmark.* http://www.aifb.uni-karlsruhe.de/~jbr/MovePeaks/movpeaks/, (dostęp 08/10/2008).

[41] J. Brest, A. Zamuda, B. Boskovic, M.S. Maucec, V. Zumer: *Dynamic Optimization Using Self-Adaptive Differential Evolution.* IEEE Congress on Evolutionary Computation, 2009.

[42] F. Brglez, D. Bryant, K. Kozminski: *Combinational Profiles of Sequential Benchmark Circuits.* Proc. Int. Symp. on Circuits and Systems, s. 1929–1934.

[43] F. Brglez, H. Fujiwara: *A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran.* Proc. International Symposium on Circuits and Systems, Special Session on ATPG and Fault Simulation, June 1985, s. 663–698.

[44] R. Brits, A.P. Engelbrecht, F. van den Bergh: *Solving Systems of Unconstrained Equations Using Particle Swarm Optimization.* IEEE Conf. Syst. Man, and Cybern., 2002, s. 102–107.

[45] R. Brits, A. Engelbrecht, F. van den Bergh: *A Niching Particle Swarm Optimizer.* 4th Asia-Pacific Conf. Simulated Evolution and Learning, Vol. 2, 2002, s. 692–696.

[46] O. Brynestad, E.J. Aas, A.E. Vallestad: *State Transition Graph Analysis as a Key to BIST Fault Coverage.* Proc. International Test Conf., 1990, s. 537–543.

[47] L.T, Bui, J. Branke, H.A. Abbass: *Multiobjective Optimization for Dynamic Environments.* Congress on Evolutionary Computation, IEEE, 2005, s. 2349–2356.

[48] E. Cabib, R. Schaefer, H. Telega: *A Parallel Genetic Clustering for Inverse Problems.* LNCS 1541, Springer, 1998, s. 551–556.

[49] K. Chakarbarty, S.R. Das: *Test-Set Embedding Based on Width Compression for Mixed-Mode BIST.* IEEE Transactions on Instrumentation and Measurement, Vol. 49, No. 3, June 2000, s. 671–678.

[50] L. Changhe, S. Yang: *Fast Multi-Swarm Optimization for Dynamic Optimization Problems.* 4th International Conf. on Natural Computation, 2008, s. 624–628.

[51] R. Chelouah, P. Siarry: *A Hybrid Method Combining Continuous Taboo Search and Nelder-Mead Simplex Algorithms for The Global Optimization of Multiminima Functions.* European Journal of Operational Research, Vol. 161, 2005, s. 636–654.

[52] R. Chelouah, P. Siarry: *A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions.* Journal of Heuristics, Vol. 6, 2000, s. 191–213.

[53] R. Chelouah, P. Siarry: *Genetic and Nelder-Mead Algorithms Hybridized for a More Accurate Global Optimization of Continuous Multiminima Functions.* European Journal of Operational Research, Vol. 148, 2003, s. 335–348.

[54] R. Chelouah, P. Siarry: *Tabu Search Applied to Global Optimization.* European Journal of Operational Research, Vol. 123, 2000, s. 256–270.

[55] H.F. Chen: *Stochastic Approximation and Its Applications.* Kluwer Academic Publishers, 2003.

[56] A.B. Chhabra, C. Meneveau, R.V. Jensen, K.R. Sreenivasan: *Direct Determination of the f(α) Singularity Spectrum and its Application to Fully Developed Turbulence.* Phys. Rev. A, **40-9**, 1989, s. 5284–5294.

[57] D. Chiles, J. DeJaco: *Using Boundary Scan Description Language in Design.* Proc. International Test Conf., 1991, s. 865–868.

[58] S. Chiusano, F. Corno, P. Prinetto, M.S. Reorda: *C2BIST: A New BIST Architecture.* IEEE European Test Workshop, 1997, s. 96–99.

[59] M. Chodacki, I. Gościniak, D. Badura: *Genetic Programming for the BIST Structures Designing.* International Carpathian Control Conf., Poland, 2004, s. 53–58.

[60] A. Chorazyczewski, R. Galar: *Evolutionary Dynamics in Space of States.* Proc. Congress on Evolutionary Computation, 2001, s. 1366–1373.

[61] M. Clerc, J.Kennedy: *The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space.* IEEE Trans. Evol. Comput., Vol. 6, No. 1, February 2002, s. 58–73.

[62] F. Corno, N. Gaudenzi, P. Prinetto, M.S. Reorda: *On the Identification of Optimal Cellular Automata for Built-In Self-Test of Sequential Circuits.* Proc. 16th IEEE VLSI Test Symposium, 1998, s. 424–429.

[63] F. Corno, P. Prinetto, M.S. Reorda: *Circular Self-Test Path for FSM.* IEEE Design & Test of Computers, Winter 1996, s. 50–58.

[64] F. Corno, M.S. Reorda: *A New BIST Architecture for Sequential Circuits.* IEEE European Test Workshop, 2000, s. 167–172.

[65] L.N. De Castro, F.J. von Zuben: *An Evolutionary Immune Network for Data Clustering.* SBRN'2000, IEEE Computer Society Press, 2000, s. 84–89.

[66] L.N. De Castro, F.J. von Zuben: *Artificial Immune Systems: Part I – Basic Theory and Applications.* Technical Report TR-DCA 01/99, 1999.

[67] L.N. De Castro, F.J. von Zuben: *The Clonal Selection Algorithm with Engineering Applications.* GECCO'00, 2000, s. 36–37.

[68] F.O. De Franc, F.J. von Zuben: *A Dynamic Artificial Immune Algorithm Applied to Challenging Benchmarking Problems.* IEEE Congress on Evolutionary Computation, 2009.

[69] L. Devroye: *Progressive Global Random Search of Continuous Functions.* Mathematical Programming No. 15, 1978, s. 330–342.

[70] W. Du, B. Li: *Multi-Strategy Ensemble Particle Swarm Optimization for Dynamic Optimization.* Information Sciences **178**, 2008, s. 3096–3109.

[71] G. Dzemyda, V. Saltenis, A. Zilinskas: *Stochastic and Global Optimization.* Kluwer Academic Publishers, 2002.

[72] E.B. Eichelberger, E. Lindbloom, J.A. Waicukauski, T.W. Williams: *Structured Logic Testing.* Prentice Hall Inc., New Jersey, 1991.

[73] B. Eschermann: *Optimized Synthesis Techniques for Testsble Sequential Circuits.* IEEE Trans. On CAD. Vol. 11, No. 3, March 1992, s. 301–312.

[74] B. Eschermann, H.J. Wunderlich: *Parallel Self-Test and Synthesis of Control Units.* Proc. of 2nd ETC, 1991, s. 73–82.

[75] C. Fagot, O. Gascuel, P. Girard, C. Landrault: *Calculating Efficient LFSR Seeds for Built-In Self Test.* Proc. IEEE European Test Workshop, 1999, s. 7–14.

[76] K. Falconer: *Fractal Geometry.* Mathematical Foundations and Applications, Wiley, New York, 2003.

[77] P. Fiser, J. Hlavicka: *Column-Matching Based BIST Design Method.* Proc. 7th IEEE Europian Test Workshop, Corfu Greece, 2002, s. 15–16.

[78] P. Fiser: *Pseudo-Random Pattern Generator Design for Column-Matching BIST.* Proc. 10th Euromicro Conf. on Digital System Design Architectures, Methods and Tools, 2007, s. 657– 663.

[79] A. Gaspar, P. Collard: *From Gas to Artificial Immune Systems: Improving Adaptation in Time Dependent Optimization.* Proc. Congress on Evolutionary Computation, Vol. 3, IEEE Press, 1999, s. 1859–1866.

[80] J. Gawin, P. Fochtman: *Substitution Algorithm for Binary Chains.* Control and Cybernetics, Vol. 21, No. 3/4, 1992.

[81] J. Gawin: *Metoda modelowania funkcji układów.* Gliwice: Politechnika Śląska, 1994 [rozprawa doktorska].

[82] D. Goldberg: *Algorytmy genetyczne w zastosowaniach.* Warszawa: WNT, 1995.

[83] I. Gościniak: *A Method for Increasing of Self-Testing Effectiveness in the Circular Self-Test Path Base BIST.* IEEE European Test Workshop, Cagliari, Italy, May 1997, s. 114–115.

[84] I. Gościniak: *A New Approach to Linear Connections Building BIST Structure Based on CSTP Structure.* The 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI System, Cannes, France, October 2004, s. 256–263.

[85] I. Gościniak: *A Way of BIST Structure Optimization.* World Congress in Computer Science, Computer Engineering & Applied Computing – WORLDCOMP'07, Las Vegas, Nevada, USA, June 2007, s. 206–212.

[86] I. Gościniak, D. Badura: *Modyfikacja ścieżki samotestującej dla zwiększenia skuteczności testowania.* Elektronika, nr 5, SIGMA: Warszawa, 1996, s. 27–31.

[87] I. Gościniak: *BIST Structure for ASIC Circuits.* Proc. International Conf. on Computational Intelligence for Modelling, Control and Automation – CIMCA, Vol. I, Vienna, Austria, November 2005, s. 840–845.

[88] I. Gościniak: *Criteria of Elements Selection in Digital Circuits Self-Testing Structure.* ASIS, Czech Republic, September 2000, s. 179–183.

[89] I. Gościniak: *Design for Testability Using Both ACTIVE-CAD and Universal Binary Simulator.* ASIS, Krnov, Czech Republic, September 1998, s. 137–142.

[90] I. Gościniak: *Genetic Modification, Artificial Individual, Vaccines and Serum – Genetic Engineering in Evolutionary Systems.* Evolutionary Computation and Global Optymization, Prace Naukowe Elektronika z. 156, Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 2006, s. 147–153.

[91] I. Gościniak: *Genetic Programming in Feedback Registers Designing.* Proc. IFAC Workshop on Programmable Devices and Systems, Institute of Electronics, Faculty of Automatic Control, Electronics and Computer Science, Gliwice: Silesian University of Technology, Poland, Cracow, November 2004, s. 253–256.

[92] I. Gościniak: *Immune Algorithm in Non-Stationary Optimization Task.* Proc. International Conf. on Computational Intelligence for Modelling Control & Automation Vienna, 2008, s. 750–755.

[93] I. Gościniak, A. Kalarus, R. Bałaga: *Uniwersalny program symulatora dla struktur technik BIST.* XIXth International Workshop Advanced Simulation of Systems. Krnov, Czech Republic, September 1997, s. 331–336.

[94] I. Gościniak: *Metoda doboru i ocena efektywności ułatwionego testowania i samotestowania układów cyfrowych.* Wydział Automatyki, Elektroniki i Informatyki Politechniki Śląskiej w Gliwicach – Instytut Elektroniki, Gliwice, 1997 [rozprawa doktorska].

[95] I. Gościniak: *Multi-Valued Function Chains in Evolutionary Algorithm.* Proc. International Conf. on Computational Intelligence for Modelling. Control and Automation – CIMCA, Vienna, Austria, November 2005, s. 41–46.

[96] I. Gościniak: *Tool for Simulation and Optimization BIST Structures.* EUROCON'2007 – The International Conf. on Computer as a Tool, Warsaw, Poland, September 2007, s. 2256–2262.

[97] I. Gościniak, M. Chodacki: *Genetic Algorithms for the Designing Feedback Shift Register.* 6th IEEE International Workshop on Design and Diagnostics of Electronic Circuits and Systems, Poznań, Poland, April 2003, s. 301–302.

[98] C. Guus, E. Boender, H. Romeijn: *Stochastic Methods.* In: R. Horst, P.M. Pardalos, (eds.): *Handbook of Global Optimization.* Kluwer Academic Publishers, 1995.

[99] T.D. Gwiazda: *Algorytmy genetyczne. Kompendium. Tom I. Operator krzyżowania dla problemów numerycznych.* Warszawa: Wydawnictwo Naukowe PWN, 2007.

[100] T.D. Gwiazda: *Algorytmy genetyczne. Kompendium. Tom II. Operator mutacji dla problemów numerycznych.* Warszawa: Wydawnictwo Naukowe PWN, 2007.

[101] R. Haralick, S. Sternberg, X. Zhuang: *Image analysis using mathematical morphology.* IEEE Trans Pattern Anal. Machine Intell Pami-9 **4**, 1987, s. 532–550.

[102] A.B. Hashemi, M.R. Meybodi: *Cellular Pso: a PSO for dynamic environments.* Advances in Computation and Intelligence, 2009, s. 422–433.

[103] H. Hastings, G. Sugihara: *Fractals: A User's Guide for The Natural Sciences.* Oxford University Press, Oxford – New York – Tokyo, 1994.

[104] Q. He, L. Wang: *An Effective co-Evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems.* Engineering Applications of Artificial Intelligence, **20**, 2006, s. 89–99.

[105] S. Hellebrand, H.G. Liang, H.J. Wunderlich: *A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters.* Proc. IEEE International Test Conf., Atlantic City, NJ, October 2000, s. 778–784.

[106] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, B. Courtois: *Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers.* IEEE Transactions on Computers, Vol. 44, No. 2, February 1995, s. 223–233.

[107] M. Higashitani, A. Ishigame, K. Yasuda: *Particle Swarm Optimization Considering the Concept of Predator-Prey Behavior.* Proc. of the 2006 IEEE Congress on Evolutionary Computation, 2006, s. 434–437.

[108] M. Higashitani, A. Ishigame, K. Yasuda: *Pursuit-Escape Particle Swarm Optimization.* Trans. On Electrical and Electronic Eng., Vol. 3, No. 1, 2008, s. 136–142.

[109] A. Hławiczka (red.): *Łatwo testowalne układy i pakiety cyfrowe – projektowanie i testowanie.* Warszawa: WNT, 1993.

[110] A. Hławiczka, K. Mostowski: *Wpływ liniowych sprzężeń zwrotnych na skuteczność analizy sygnaturowej uszkodzeń.* Prace IPI PAN nr 665, 1989.

[111] A. Hławiczka: *Rejestry liniowe – analiza, synteza i zastosowania w testowaniu układów cyfrowych.* Gliwice: Wydawnictwo Politechniki Śląskiej, 1997.

[112] A. Hławiczka: *Struktury testerów wewnętrznych dla samotestowalnych układów sekwencyjnych (część 4).* Elektronika, nr 7–8, 1997, s. 33–38.

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

[114] R. Horst, P.M. Pardalos, (eds.): *Handbook of Global Optimization.* Kluwer, 1995.

[115] H.H. Hoos, T. Stützle: *Stochastic Local Search Foundations and Applications.* Morgan Kaufmann – Elsevier, 2004.

[116] X. Hu, R.C. Eberhart: *Adaptive particle swarm optimization: detection and response to dynamic systems.* In: IEEE Congress on Evolutionary Computation, CEC2002, 2002, s. 1666–1670.

[117] http://gurneyjourney.blogspot.com/2009/09/eye-tracking-and-composition-part-3.html, (dostęp 09/2009).

[118] X. Hu, R.C. Eberhart, Y. Shi: *Engineering Optimization with Particle Swarm.* Proc. 6th World Multiconf. Syst. Cybern. Inform, 2002, s. 53–57.

[119] H. Hürner: *A C++ Class Library for Genetic Programming.* Vienna: The Vienna University of Economics, 1996.

[120] *IEEE Standard Test Access Port and Boundary-Scan Architecture.* New York, 1990.

[121] W.S. Jang, H.I. Kang, B.H. Lee, K.I. Kim, D.I. Shin, S.C. Kim: *Optimized Fuzzy Clustering by Predator Prey Particle Swarm Optimization.* IEEE/CEC, 2007, s. 3232–3238.

[122] J. Juliany, M.D. Vose: *The Genetic Algorithm Fractal.* Evolutionary Computation, MIT Press, 1994, s. 165–180.

[123] M. Kamosi, A.B. Hashemi, M.R. Meybodi: *A New Particle Swarm Optimization Algorithm for Dynamic Environments.* Swarm, Evolutionary, and Memetic Computing, 2010, s. 129–138

[124] M. Kamosi, A.B. Hashemi, M.R. Meybodi: *A Hibernating Multi-Swarm Optimization Algorithm for Dynamic Environments.* Proc. of World Congress on Nature and Biologically Inspired Computing, Kitakyushu, Japan, 2010, s. 370–376.

[125] J. Kennedy, R.C. Eberhart: *Particle Swarm Optimization.* Proc. IEEE Int. Conf. On Neural Networks, Piscataway, NJ, 1995, s. 1942–1948.

[126] J. Kennedy, R.C. Eberhart: *A Discrete Binary Version of the Particle Swarm Algorithm.* In: Proc. of the 1997 IEEE International Conf. on Systems, Man, and Cybernetics, Piscataway, NJ, USA, 1997.IEEE Press, 1997, s. 4104–4108.

[127] J. Kennedy: *Stereotyping: Improving Particle Swarm Performance with Cluster Analysis.* Congr. Evol. Comput., 2002, s. 1507–1512.

[128] J. Kennedy, R. Mendes: *Population Structure and Particle Swarm Performance.* Proc. IEEE Congr. Evol. Comput., Honolulu, HI, 2002, s. 1671–1676.

[129] G. Kiefer, H. Vranken, E.J. Marinissen, H.J. Wunderlich: *Application of Deterministic Logic BIST on Industrial Circuits.* Journal of Electronic Testing, Theory and Applications **17**, Kluwer Academic Publishers, 2001, s. 351–362.

[130] G. Kiefer, H.J. Wunderlich: *Deterministic BIST with Partial Scan.* Journal of Electronic Testing, Vol. 16, No. 3, 2000, s. 169–177.

[131] P. Kieś: *Information Dimension of a Population's Attracted and Population's Entropy.* Proc. International Symposium on Intelligent Information Systems X, Physica Verlag, 2001, s. 271–280.

[132] P. Kieś: *Information Dimension of a Population's Attractor in a Binary Genetic Algorithm.* Proc. International Conf. Artificial Neural Nets and Genetic Algorithms in Prague, Czech Republic, 2001, s. 232–235

[133] M. Kopeć: *Analiza i zwiększanie skuteczności pierścieni testujących.* Gliwice: Politechnika Śląska, 1994 [rozprawa doktorska].

[134] M. Kopeć: *Can Nonlinear Compactors Be Better than Linear Ones.* IEEE Transactions on Computers, Vol. 44, November 1995, s. 1275–1282.

[135] P. Korosec, J. Silc: *The Differential Ant-Stigmergy Algorithm Applied to Dynamic Optimization Problems.* In: IEEE Congress on Evolutionary Computation, 2009.

[136] Z. Kotasek, J. Strandel, D. Mika: *Methodologies of RTL Partial Scan Analysis and Their Comparison, Proc.* Poznań: IEEE Workshop on Design and Diagnostic of Electronic Circuits and Systems, Poland, UNI-DRUK, 2003, s. 233–238.

[137] R. Kothari, D. Ha: *Experimental Results on Aliasing Errors in Circular BIST Designs.* Proc. European Test Conf., 1993, s. 466–474.

[138] S. Kotowski, W. Koniński, Z. Michalewicz, J. Nowicki, B. Przepiórkiewicz: *Fractal Dimension of Trajectory as Invariant of Genetic Algorithms.* Artificial Intelligence and Soft Computing – ICAISC, LNCS V5097, 2008, s. 414–425.

[139] J.R. Koza: *Genetic Programming: On the Porgramming of Computers by Means of Natural Selection.* MIT Press, 1992.

[140] A. Kraśniewski, S. Pilarski: *Circular Self-Test Path, A Low-Cost BIST Technique for VLSI Circuits.* IEEE Transactions on Computer-Aided Design, Vol. 8, No. 1, January 1989, s. 46–55.

[141] A. Kraśniewski, S. Pilarski: *Effectiveness of Using a Test Response Compactor for Test Pattern Generation.* Proc. FTSD'87 Conf., Varna, 1987, s. 284–289.

[142] A. Kraśniewski: *Projektowanie samotestowalnych układów cyfrowych wielkiej skali integracji.* Prace Naukowe Politechniki Warszawskiej, Warszawa: WPW, 1989.

[143] H.C. Kuo, J.R. Chang, C.H. Liu: *Particle Swarm Optimization for Global Optimization Problems.* Journal of Marine Science and Technology, Vol. 14, No. 3, 2006, s. 170–181.

[144] H.C. Kuo, J.R. Chang, K.S. Shyu: *A Hybrid Algorithm of Evolution and Simplex Methods Applied to Global Optimization.* Journal of Marine Science and Technology, Vol. 12, 2004, s. 280–289.

[145] A. Leontitsis, D. Kontogiorgos, J. Pange: *Repel the Swarm to the Optimum!.* Applied Mathematics and Computation 173 (**1**), 2006, s. 265–272.

[146] C. Li, S. Yang, T.T. Nguyen, E.L. Yu, X. Yao, Y. Jin, H.G. Beyer, P.N. Suganthan: *Benchmark Generator for CEC'2009 Competition on Dynamic Optimization.*, CEC'2008, 2008, s. 1-14.

[147] C. Li, S. Yang: *A Clustering Particle Swarm Optimizer for Dynamic Optimization.* IEEE Congress on Evolutionary Computation, 2009, s. 439–446.

[148] C. Li, S. Yang: *A General Framework of Multi-Population Methods with Clustering in Undetectable Dynamic Environments.* IEEE Transactions on Evolutionary Computation, Vol. 16, No. 4, 2012, s. 556–577.

[149] L. Li, K. Chakarbarty: *Test Set Embedding for Deterministic BIST Using a Reconfigurable Interconnection Network.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 9, September 2004, s. 1289–1305.

[150] X. Li: *Adaptively Choosing Neighborhood Bests Using Species in a Particle Swarm Optimizer for Multimodal Function Optimization.* Genetic Evol. Comput. Conf., 2004, s. 105–116.

[151] C. Liand, S. Yang: *Fast Multi-Swarm Optimization for Dynamic Optimization Problems.* 4th Int. Conf. Natural Comput., Vol. 7, 2008, s. 624–628.

[152] H.G. Liang, S. Hellebrand, H.J. Wunderlich: *Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST.* Proc., Baltimore: IEEE International Test Conf., MD, 2001, s. 894–902.

[153] J.J. Liang, P.N. Suganthan, K. Deb: *Novel Composition Test Functions for Numerical Global Optimization.* Proc. Swarm Intell. Symp., Available: http://www.ntu.edu.sg/home/EP-NSugan, June 2005.

[154] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar: *Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions.* Evolutionary Computation, IEEE Transactions on Vol. 10, Issue 3, June 2006, s. 281–295.

[155] M. Littman, D. Ackley: *Adaption in Constant Utility Nonstationary Environment.* ICGA'91, 1991, s. 136–142.

[156] L. Liu, S. Yang, D. Wang: *Particle Swarm Optimization with Composite Particles in Dynamic Environments.* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **40**, 2010, s. 1634–1648.

[157] R.I. Lung, D. Dumitrescu: *A Collaborative Model for Tracking Optima in Dynamic Environments.* IEEE Congress on Evolutionary Computation, 2007, s. 564–567.

[158] R.I. Lung, D. Dumitrescu: *Evolutionary Swarm Cooperative Optimization in Dynamic Environments.* Natural Computing **9**, 2010, s. 83–94.

[159] S. Lynch: *Dynamical Systems with Applications using Maple.* Birkhauser, 2000.

[160] B.B. Mandelbrot: *The Fractal Geometry of Nature.* W.H.Freeman, New York, 1983.

[161] R. Mendes, J. Kennedy, J. Neves: *The Fully Informed Particle Swarm: Simpler, maybe better.* IEEE Trans. Evol. Comput., Vol. 8, June 2004, s. 204–210.

[162] R. Mendes, A. Mohais: *DynDE: a Differential Evolution for Dynamic Optimization Problems.* IEEE Congress on Evolutionary Computation, 2005, s. 2808–2815.

[163] Z. Michalewicz, D.B. Fogel: *How to Solve It: Modern Heuristics.* Springer, (*Jak to rozwiązać czyli nowoczesna heurystyka.* Warszawa: WNT), 2006.

[164] Z. Michalewicz: *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd, rev. edition, Springer, Berlin, Heidelberg et al., (*Algorytmy genetyczne + struktury danych = programy ewolucyjne.* Warszawa: WNT, 1996), 2000.

[165] Z. Michalewicz, M. Shmidt: *Evolutionary Algorithms and Constrained Optimization, in Evolutionary Optimization.* Edited by R. Sarker, M. Mohammadian, Y. Xin, Kluwer Academic Publishers, 2002.

[166] M.M. Millonas: *Swarms, Phase Transitions and Collective Intelligence.* In: C.G. Langton (ed.), *Artificial Life III.* Addison Wesley, Reading, MA, 1994.

[167] A. Mohais, R. Mendes, Ch. Ward, Ch. Postoff: *Neighborhood restructuring in particle swarm optimization.* In: Shichao Zhang and Ray Jarvis, (ed.), LNCS 3809, Proc. of the 18th Australian Joint Conf. on Artificial Intelligence, Springer, 2005, s. 776–785.

[168] R.W. Morrison, K.A. De Jong: *A Test Problem Generator for Non-Stationary Environments.* In IEEE Congress on Evolutionary Computation, Vol. 3, 1999, s. 2047–2053.

[169] I. Moser, T. Hendtlass: *A Simple and Efficient Multi-Component Algorithm for Solving Dynamic Function Optimisation Problems.* IEEE CEC, 2007.

[170] B. Nasiri, M.R. Meybodi: *Speciation based firefly algorithm for optimization in dynamic environments.* International Journal of Artificial Intelligence, Vol. 8, 2012, s. 118–132.

[171] N. Noroozi, A.B. Hashemi, M.R. Meybodi: *CellularDE: a Cellular Based Differential Evolution for Dynamic Optimization Problems.* Adaptiveand Natural Computing Algorithms, 2011, s. 340–349.

[172] O. Novak, Z. Pliva, J. Nosek, A. Hlawiczka, T. Garbolino, K. Gucwa: *Test-Per-Clock Logic BIST with Semi-Deterministic Test Patterns and Zero-Aliasing Compactor.* Journal of Electronic Testing: Theory and Applications **20**, Manufactured in The United States, Kluwer Academic Publishers, 2004, s. 109–122.

[173] E.S. Oczeretko: *Wymiar fraktalny w analizie sygnałów i obrazów biomedycznych.* Białystok: Wydawnictwo Uniwersytetu w Białymstoku, 2006.

[174] E. Ott: *Chaos w układach dynamicznych.* Warszawa: Wydawnictwo Naukowo-Techniczne, 1997.

[175] G. Papa, T. Garbolino, F. Novak, A. Hławiczka: *Deterministic Test Pattern Generator Design with Genetic Algorithm.* Journal of Electrical Engineering, Vol. 58, No. 3, 2007, s. 121–127.

[176] K.P. Parker, S. Oresjo: *A Language for Describing Boundary-Scan Devices.* Proc. International Test Conf., 1990, s. 222–234.

[177] K.P. Parker: *The Boundary-Scan Handbook.* Boston: Kluwer Academic Publishers, 1992.

[178] D. Parrott, X. Li: *A Particle Swarm Model for Tracking Multiple Peaks in a Dynamic Environment Using Speciation.* 2004 Congr. Evol. Comput., 2004, s. 98–103.

[179] D. Parrott, X. Li: *Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation.* IEEE Trans. Evol. Comput., Vol. 10, No. 4, 2006, s. 440–458.

[180] K.E. Parsopoulos, M.N. Vrahatis: *UPSO – A Unified Particle Swarm optimization Scheme.* Lecture Series on Computational Sciences, 2004, s. 868–873.

[181] A. Passaroand, A. Starita: *Particle Swarm Optimization for Multimodal Functions Clustering Approach.* Journal of Artif. Evol. and Appl., Vol. 2008, 2008, s. 8:1–8:15.

[182] H.O. Peitgen, H. Jürgens, D. Saupe: *Chaos and Fractals: New Frontiers of Science.* Springer-Verlag, New York, 1992.

[183] T. Peram, K. Veeramachaneni, C.K. Mohan: *Fitness-Distance-Ratio Based Particle Swarm Optimization.* Swarm Intelligence Symp., 2003, s. 174–181.

[184] S. Pilarski, T. Kameda: *A Probabilistic Analisys of Test-Responce Compaction.* IEEE Computer Society Press, 1995.

[185] S. Pilarski, A. Kraśniewski, T. Kameda: *Estimating Testing Effectiveness of the Circular Self-Test Path Technique.* IEEE Transactions on Computer-Aided Design, Vol. 11, No. 10, October 1992, s. 1301–1316.

[186] S. Pilarski, A. Kraśniewski, T. Kameda: *Estimating Testing Effectiveness of the Circular Self-Test Path Technique.* Raport CSS/LCCR (Simon Fraser University, Burnaby, B.C.Canada) TR 90-10, 1990, s. 1301–1316.

[187] M.C. Plessis, A.P. Engelbrecht: *Differential Evolution for Dynamic Environments with Unknown Numbers of Optima.* Journal of Global Optimization, http://dx.doi.org/10.1007/s10898-012-9864-9, 2012.

[188] R. Poli, C.Di. Chio, W.B. Langdon: *Exploring extended particle swarms: A genetic programming approach* (A. Marco, M. de Oca: *Particle Swarm Optimization*). In: Proc. of the 2005 Conf. on Genetic and Evolutionary Computation, New York, USA, ACM Press, 2005, s. 169–176.

[189] J. Rajski, N. Tamarapalli, J. Tyszer: *Automated Synthesis of Phase Shifters for Built-In Self-Test Applications.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 19, No. 10, 2000, s. 1175–1188.

[190] I. Rezazadeh, M.R. Meybodi, A. Naebi: *Adaptive Particle Swarm Optimization Algorithm for Dynamic Environments.* Advances in Swarm Intelligence, 2011, s. 120–129.

[191] J. Riget, J. Vesterstroem: *A Diversity-Guided Particle Swarm Optimizer – the ARPSO.* Technical Report 2002-02, Department of Computer Science, University of Aarhus, 2002.

[192] A.H.G. Rinnooy Kan, G.T. Timmer: *Stochastic Methods for Global Optimization, Part 1. Clustering Methods.* Mathematical Programming **39**, 1987, s. 27–56.

[193] A.H.G. Rinnooy Kan, G.T. Timmer: *Stochastic Methods for Global Optimization, Part 2. Multilevel Methosds.* Mathematical Programming **39**, 1987, s. 57–78.

[194] R. Schafer: *Podstawy genetycznej optymalizacji globalnej.* Kraków: WUJ, 2002.

[195] M. Schroeder: *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise.* W.H. Freeman, New York, 1991, s. 41–45.

[196] D. Sedighizadeh, E. Masehian: *Particle Swarm Optimization Methods, Taxonomy and Applications.* International Journal of Computer Theory and Engineering, Vol. 1, No. 5, December 2009, s. 1793–8201.

[197] J. Seidler: *Nauka o informacji. Tomy 1 i 2.* Warszawa: WNT, 1983.

[198] J. Shahabi: *A Multi-Set Artificial Immune System for Searching Optima in Dynamic Environments.* Computer Engineering, Eastern Mediterranean University, August 2012 [rozprawa doktorska].

[199] Y. Shi, R.C.Eberhart: *A Modified Particle Swarm Optimizer.* Proc. IEEE Congr. Evol. Comput., 1998, s. 69–73.

[200] S. Smith: *The Simplex Method and Evolutionary Algorithms.* ICEC'98, 1998, s. 799–804.

[201] J.C. Spall: *Introduction to Stochastic Search and Optimization.* Wiley-Interscience, 2003.

[202] W.M. Spears: *Evolutionary Algorithms: The Role of Mutation and Recombination.* Natural Computing Series, Springer, 2000.

[203] Ch.E. Stroud: *A Designer's Guide to Built-in Self-Test.* Kluwer Academic Publishers, 2002.

[204] S. Sudireddy, J. Kakade, D. Kagaris: *Deterministic Built-In TPG with Segmented FSMs.* Proc. 14th IEEE International On-Line Testing Symposium, 2008, s. 261–266.

[205] G. Syswerda: *Uniform Crossover in Genetic Algorithms.* Proc. Third International Conf. on Genetic Algorithms, (J. Schaffer (ed.)), Morgan Kaufmann Publishers, Los Altos, CA, 1989. s. 2–9.

[206] G. Tao, Z. Michalewicz: *Inver-Over Operator for the TSP.* In: Proc. PPSN, Springer, s. 803–812, 1998.

[207] R. Thomsen: *Multimodal Optimization Using Crowding-Based Differential Evolution.* Congr. Evol. Comput., Vol. 2 , 2004, s. 1382–1389.

[208] A.A. Torn: *Cluster Analysis Using Seed Points and Density-Determined Hyperspheres as an Aid to Global Optimization Systems.* Man, and Cybernetics, IEEE Transactions on, August Vol. 7, Issue 8, 1977, s. 610–616.

[209] A.A. Torn, S. Viitanen: *Topographical Global Optimization Using Pre-Sampled Points.* Global Optimization **5**, 1994, s. 267–276.

[210] N.A. Touba, E.J. McCluskey: *Altering a Pseudo-Random Bit Sequence for Scan-Based BIST.* Proc.International Test Conf., 1996, s. 167–175.

[211] K. Trojanowski, S. Wierzchoń: *Immune-Based Algorithms for Dynamic Optimizaion.* Information Sciences, Vol. 179, Issue 10, April 2009, s. 1495–1515.

[212] K. Trojanowski, Z. Michalewicz: *Searching for Optima in Non Stationary Environments.* Proc. of the Congress on Evolutionary Computation, Vol. 3, IEEE Press, Piscataway, 1999, s. 1843–1850.

[213] I. Tsoulos, A. Stavrakoudis: *Enhancing pso methods for global optimization.* Applied Mathematics and Computation Vol. 216, No. 10, 2010, s. 2988–3001.

[214] F. van den Bergh, A.P. Engelbrecht: *A cooperative approach to particle swarm optimization.* IEEE Trans. Evol. Comput., Vol. 8, June 2004, s. 225–239.

[215] F. Wagner: *Projektowanie krótkich rejestrów liczących.* Gliwice: Wydawnictwo Politechniki Śląskiej, ZN **526**, 1977.

[216] L.T. Wang, Ch.W. Wu, X. Wen: *VLSI Test Principles and Architectures.* Elsevier Inc., 2006.

[217] T. Weise: *Global Optimization Algorithms – Theory and Application.* http://www.it-weise.de/,[wersja: 2009-06-26], 2009.

[218] S.T. Wierzchoń: *Sztuczne systemy immunologiczne, Teoria i zastosowania.* Exit, 2001.

[219] S.T. Wierzchoń: *Multimodal Optimization with Artificial Immune Systems.* Intelligent Information Systems, Physica-Verlag, 2001, s. 167–179.

[220] T.W. Williams, W. Daehn, M. Gruetzner, C.W. Starke: *Aliasing Errors in Signature Analysis Registers.* IEEE Design & Test of Computers, April 1987, s. 39–45.

[221] R. Wit: *Metody programowania nieliniowego.* Warszawa: WNT, 1986.

[222] Y.G. Woldesenbet, G.G. Yen: *Dynamic evolutionary algorithm with variable relocation.* IEEE Transactions on Evolutionary Computation, Vol. 13, 2009, s. 500–513.

[223] D. Wolpert, W.G. Macready: *No Free Lunch Theorems for Optimization.* IEEE Transaction on Evolutionary Computation, Vol. 1, No. 1, April 1997, s. 67–82.

[224] S. Yang, C. Li: *A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments.* IEEE Transactions on Evolutionary Computation, Vol. 14, No. 6, 2010, s. 959–974.

[225] D. Yazdani, M.R. Akbarzadeh, B. Nasiri, M.R. Meybodi : *A new artificial fish swarm algorithm for dynamic optimization problems.* IEEE Congress on Evolutionary Computation, http://dx.doi.org/10.1109/CEC.2012.6256169, 2012, s. 1–8.

[226] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M.R. Meybodi: *A Novel Multi-Swarm Algorithm for Optimization in Dynamic Environments Based on Particle Swarm Optimization.* Applied Soft Computing, Vol. 13, Issue 4, 2013, s. 2144–2158.

[227] Ch. Yu, S.M. Redy, I. Pomeranz: *Weighted Pseudo-Random BIST for N-detection of Single Stuck-at Faults.* Proc. 13^{th} Asian Test Symposium ATS, 2004, s. 178–183.

[228] E.L. Yu, P.N. Suganthan: *Evolutionary Programming with Ensemble of Explicit Memories for Dynamic Optimization.* IEEE Congress on Evolutionary Computation, 2009, s. 431–438.

[229] A. Zhigljavsky, A. Zilinskas: *Stochastic Global Optimization.* Springer, 2008.

[230] Y. Zorian, H. Bederr: *Design Self-Testable Multi-Chip Module.* European Design and Test Conf., ED&TC, 1996, s. 181–185.

Ireneusz Gościniak

A new approach to selected optimization problems

Summary

In solving complex optimization tasks evolutionary algorithms have a leading position. Unusual look at the optimization algorithms presented in the thesis, led to the creation of the new algorithm and work on its development to put its metaphors in a group of artificial life. The resulting algorithms are still the effective optimization algorithms and the proposed approach introduces new properties in their operation. The study presents a new algorithm of observation - as the base algorithm and its metaphors placed in a group of immune algorithm and particle swarm optimization algorithms. Research on the mechanics of these algorithms demonstrated new properties, i.e.: behavior resembling observation, and co-evolution mechanism determines the behavior of independence on influences of the environment. Implementation of the assumptions imposed the need to develop effective mechanism of mutation for immune algorithm. The functions of behavior scenarios were defined for the particle swarm optimization algorithm. A group of immune systems is proposed which is an equivalent to the multi-population system and methods of information exchange between systems in the group are defined. The thesis presents a theoretical background of algorithms' operation and a simulation study. To check the efficiency of the algorithms the typical test environment for stationary and non-stationary problems were applied. In the study, fractal and multifractal analysis was used and its usefulness was demonstrated in research on behavior of algorithms. Optimization of diagnostic structure of digital circuit is an issue of multimodal optimization and is a particular kind of challenge. A comprehensive approach to test multi-module circuit may lead to new solutions, also in terms of a single module testing. Such concepts are included in this study, basing on an untypical approach to testing multi-module circuit, the conclusion has a strong theoretical base. The original achievements in this dissertation are as follows: a proposal of BIST architecture based on the so-called linear modification, the introduction of the diagnostic structure description, and determination of the theoretical basis of this concept, confirmation of the formulated theoretical basement and simultaneously the verification of the diagnostic efficiency of the proposed solutions by means of simulation methods basing on modeling with using ISCAS'89 benchmark, the demonstration of permanent features of modules during testing, the presentation of a formal description of any diagnostic structure with a description of the optimization framework and the concept of simulation tools used in the current research. Simultaneously, the study shows the original use of a genetic algorithm to give a high efficiency optimization. This part of the study presents a complete system of description of any diagnostic structure with the optimization method. The solutions presented in the dissertation open the way for the further research. This dissertation is composed of two parts, despite of the common basis in a form of evolutionary algorithms, they are present different and closed thematically issues.

Keywords: optimization, multi-criteria optimization, multimodal optimization, evolutionary algorithms, genetic algorithms, immune algorithms, particle swarm optimization algorithms, a group of immune system, the algorithm of observation, exchange of genetic material, fractal analysis, multifractal analysis, beset game algorithm, immune algorithm with auto-aggression, stationary problems, non-stationary problems, BIST structure, BIST structure optimization, BIST structure description, multi-modular circuit BIST.

Ireneusz Gościniak

Ein neuer Ansatz zur ausgewählten Optimierungsprobleme

Zusammenfassung

Bei der Lösung komplexer Optimierungsaufgaben nehmen die evolutionären Algorithmen eine führende Position ein. Der ungewöhnliche Blickwinkel auf die Optimierungsalgorithmen in der vorgestellten Arbeit führte zur Gründung eines neuen Algorithmus und die Arbeit auf seine Entwicklung - seine Metaphern in einer Gruppe von künstlichen Lebens zu stellen. Die daraus resultierenden Algorithmen sind immer noch die effektive Optimierungsalgorithmen und der vorgeschlagene Ansatz stellt neue Eigenschaften in ihren Betrieb vor. Die Studie stellt einen neuen Algorithmus der Beobachtung vor - als Basis-Algorithmus und seine Metaphern gelegt in einer Gruppe von Immunalgorithmus und Algorithmen der Partikelschwarmoptimierung. Forschung auf die Mechanik dieser Algorithmen demonstriert neue Eigenschaften, dh: Verhalten ähnelt Beobachtung und Co-Evolution Mechanismus bestimmt das Verhalten der Unabhängigkeit auf Einflüsse der Umwelt. Die Umsetzung der Annahmen verhängte die Notwendigkeit zur Entwicklung des effizienten Mechanismus der Mutation für den Immunalgorithmus. Für den Algorithmus der Partikelschwarmoptimierung wurden die Funktionen der Verhaltensszenarien definiert. Es ist eine Gruppe von Immunsystemen vorgeschlagen, welche ein äquivalent zu dem Multi-Population System ist und es wurde eine Methode von Informationsaustausch zwischen Systemen in der Gruppe definiert. Die Arbeit stellt einen theoretischen Hintergrund der Betrieb-Algorithmen und eine Simulationsstudie vor. Um die Effizienz der Algorithmen zu überprüfen, wurde die typische Testumgebung für stationäre und nicht-stationäre Probleme angewandt. In der Studie wurde die fraktal und multifraktalen Analyse verwendet und ihre Nützlichkeit in der Forschung auf das Verhalten von Algorithmen dargestellt. Optimierung der Diagnosestruktur der digitalen Schaltung ist eine Frage der multimodalen Optimization und ist eine besondere Art von Herausforderung. Ein umfassender Ansatz zur Testung der Multi-Modul-Schaltung kann zu neuen Lösungen führen, auch in Bezug auf ein einzelnes Modul-Tests. Diese Studie umfasst solche Konzepte, basierend auf einer untypischen Ansatz zur Prüfung Multi-Modul-Schaltung. Der Abschluss hat einen starken theoretische Grundsatz. Die ursprünglichen Leistungen in dieser Dissertation sind wie folgt: einen Vorschlag von BIST-Architektur auf der Basis der so genannten linearen Modifikation, die Einführung der Diagnosestrukturbeschreibung und Bestimmung der theoretischen Grundlagen dieses Konzepts, die Bestätigung der formulierten theoretischen Grundlage und gleichzeitig die überprüfung der diagnostischen Effizienz der vorgeschlagenen Lösungen durch Simulationsmethoden basierend auf Modellierung mit dem ISCAS'89 Benchmark, die Demonstration der dauerhaften Merkmale der Module während der Prüfung, die Präsentation der formalen Erläuterung jeglicher Diagnosestruktur mit der Beschreibung des Optimierungsrahmens als auch das Konzept der Simulationswerkzeuge, die in der jetzigen Forschung eingesetzt werden. Gleichzeitig zeigt die Studie die ursprüngliche Verwendung eines genetischen Algorithmus, um eine hohe Wirkungsgrad-Optimierung zu geben. Dieser Teil der Studie präsentiert ein komplettes System zur Beschreibung der Diagnosestruktur mit dem Optimierungsverfahren. Diese Dissertation besteht aus zwei Teilen, die trotz der gemeinsamen Basis in Form von evolutionären Algorithmen, präsentiert unterschiedlich und thematisch geschlossene Probleme.

Schlüsselwörter: Optimierung, evolutionäre Algorithmen, genetische Algorithmen, Immun-Algorithmen, Partikel-Schwarm-Algorithmen, eine Gruppe von Immunsystems, den Algorithmus der Beobachtung, Austausch von genetischem Material, fraktale/multifraktalen Analyse, bedrängt Spiel Algorithmus, Immunalgorithmus mit Auto-Aggression, BIST Strukturoptimierung, BIST Strukturbeschreibung, multi-modulare Schaltung BIST.

Więcej o książce

CENA 28 ZŁ
(+ VAT)

ISSN 0208-6336
ISBN 978-83-8012-046-4