Raport Badawczy

Research Report

RB/27/2006

Zagadnienia projektowania algorytmów ewolucyjnych

J. Stańczak

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Zbigniew Nahorski

Warszawa 2006

Zagadnienia projektowania algorytmów ewolucyjnych

Jarosław Stańczak
IBS PAN
stanczak@ibspan.waw.pl

1. Wprowadzenie

W obecnym etapie rozwoju algorytmów ewolucyjnych dawno już minęły czasy stosowania do wszystkich rozwiązywanych zadań tego samego, standardowego zestawu operatorów genetycznych, składającego się z krzyżowania i mutacji. Podobnie jak nie stosuje się już, często „na siłę", kodowania binarnego do wszystkich rozwiązywanych zagadnień. Tendencja do stosowania bardzo prostego algorytmu genetycznego w wersji zaproponowanej przez Hollanda lub zbliżonych do wszystkich rozwiązywanych zadań trwała niestety dość długo, a i obecnie niektórzy użytkownicy algorytmów ewolucyjnych jeszcze jej ulegają. Na szczęście dla algorytmów ewolucyjnych już dość dawno temu zaczęto zauważać niedostatki takiego podejścia i powoli zarzucono używania identycznego, nie tylko co do idei działania, ale także i szczegółów technicznych realizacji, algorytmu genetycznego do rozwiązywania wszystkich zadań, na rzecz metod o różnym stopniu specjalizacji i dopasowania do konkretnego problemu. Oczywiście takie podejście do stosowania algorytmów ewolucyjnych wiąże się z pewnymi dodatkowymi trudnościami połączonymi z każdorazowym dostosowaniem metody do konkretnego problemu, ale umożliwia szybkie osiągnięcie dobrych wyników.

Wizja algorytmu genetycznego jako metody uniwersalnej, zdolnej do rozwiązania niemal każdego problemu optymalizacyjnego przy pomocy identycznych sposobów kodowania rozwiązań, operatorów i metod selekcji okazała się fałszywa. Stosowanie tak pojętego algorytmu genetycznego jest nieefektywne i konieczne jest dostosowywanie metody

do rozwiązywanego zadania, gdyż tylko maksymalnie wykorzystując wiedzę o zadaniu w stosowanej metodzie ewolucyjnej można uzyskać naprawdę dobre wyniki. W innych przypadkach algorytm ewolucyjny działa bardzo powoli, a otrzymywane wyniki są często dalekie od oczekiwań.

To rozczarowanie ideą algorytmu genetycznego spowodowało, że straciła ona przez to nieco na atrakcyjności uniwersalnego narzędzia do wszystkiego, ale zyskała za to miano uniwersalnej ewolucyjnej metody rozwiązywania zadań optymalizacyjnych, do której trzeba dopracować pewne szczegóły, takie jak operatory genetyczne, metodę kodowania wyników, czasem również i metodę selekcji, aby otrzymać w zamian bardzo efektywne narzędzie rozwiązywania przeróżnych trudnych problemów optymalizacyjnych. Tak rozumiany algorytm genetyczny, traktowany jako pewna metoda lub idea ewolucyjnego rozwiązywania zadań, bez wnikania w szczegóły konstrukcyjne algorytmu użytego do rozwiązania konkretnego zadania tworzy pewną nową jakość i nazywana jest często algorytmem ewolucyjnym. Tak rozumiany algorytm ewolucyjny jest natomiast bardzo interesującym wyzwaniem inżynierskim, gdyż odpowiednie jego zaprojektowanie i dopasowanie do rozwiązywanego zdania jest bardzo ciekawym problemem, wymagającym wiedzy zarówno o algorytmach ewolucyjnych jak i o rozwiązywanym problemie oraz na pewno doświadczenia i pewnej intuicji. Dobrze zaprojektowana metoda ewolucyjna „odwdzięczy się" dokonując obliczeń stosunkowo szybko i z dużą dokładnością.

W przypadku algorytmów ewolucyjnych trudno jest podać jakieś konkretne kryteria, formuły czy algorytmy według których można je projektować, ale pewne zasady na pewno warto jest uwzględnić, aby otrzymać dobrze działającą metodę. Dość trudno jest również porównywać różne metody ewolucyjne rozwiązujące dany problem, gdyż mogą różnić się nie tylko jakością generowanych wyników, ale również czasem działania, zajętością pamięci itp. Na to wszystko nakłada się jeszcze losowość działania, więc konkretne rezultaty zależą od danych wejściowych, konkretnej realizacji procesu stochastycznego, jakim jest działający

algorytm ewolucyjny oraz w dużym stopniu od jakości i parametrów wywołania użytych generatorów pseudolosowych. Wobec tego, aby coś można było powiedzieć o właściwościach porównywanych metod ewolucyjnych, należałoby przeprowadzić wiele testów i opracować je metodami bazującymi na statystyce. Jednakże na pewno można podać dużo przykładów w których jedna metoda ewolucyjna rozwiązuje dane zadanie licząc kilkanaście godzin, a inna kilkanaście minut, dając wyniki o zbliżonej dokładności, zużywając przy tym zbliżonej wielkości zasoby komputera. Tak więc na pewno jakaś gradacja jakości tworzonych algorytmów ewolucyjnych jest możliwa. Rozdział ten jest poświęcony sposobom projektowania metod ewolucyjnych działających szybko i z dużą dokładnością, a prezentowane przykłady zastosowań na pewno dadzą jasny pogląd na temat projektowania efektywnych algorytmów ewolucyjnych.

2. Zakodowanie osobnika a operatory ewolucyjne

Projektowanie algorytmu ewolucyjnego powinno się zacząć od przyjęcia metody kodowania rozwiązań. Teoretycznym aspektom tego zagadnieniu poświęcono wiele prac, między innymi: [Arab2001] i nie będą one tu szeroko omawiane. Przedstawione tu metody dotyczą raczej strony praktycznej zagadnienia.

Na wstępie należy jednak podkreślić fakt decydującego wpływu przyjętej metody kodowania na właściwości projektowanego algorytmu (przede wszystkim szybkość działania i dokładność otrzymywanych wyników) oraz sposób działania i implementację operatorów genetycznych. Tak więc zagadnieniu temu należy poświęcić najwięcej uwagi i być może konieczne będzie wypróbowanie kilku metod, aby osiągnąć sukces. Niestety w wielu przypadkach dość trudne jest teoretyczne przewidzenie właściwości projektowanego algorytmu ewolucyjnego na podstawie przyjętej metody kodowania.

Tak więc proces projektowania sprawnie działającego algorytmu ewolucyjnego rozpoczniemy od zaprojektowania struktury osobnika kodującego możliwe rozwiązania zadania. Zapisanie rozwiązań danego problemu za pomocą pewnego zbioru symboli i

odpowiednio wybranej do tego celu struktury danych nie jest zadaniem prostym, a ma duży wpływ na sposób działania i złożoność obliczeniową algorytmu.

Arabas w pracy [Arab2001] podaje pewne ścisłe kryteria, które powinna spełniać przyjęta metoda kodowania. Są to:

- wymaganie, aby każde rozwiązanie z przestrzeni fenotypu (zadania) miało swój odpowiednik w przestrzeni genotypu, w przeciwnym wypadku można zgubić pewne istotne dla rozwiązywanego zadania obszary z poszukiwanym ekstremum;
- wymaganie nie wprowadzania przez metodę kodowania dodatkowych ekstremów do problemu.

Dodatkowo można też wspomnieć o takich cechach jak:

- w niektórych przypadkach istnieje możliwość automatycznego uwzględnienia pewnych ograniczeń narzuconych na rozwiązania w zadaniu dzięki odpowiedniemu doborowi metody kodowania;
- wybranie metody kodowania możliwie prostej i nieangażującej nadmiernie zasobów komputera (zajętości procesora, pamięci) przy obliczaniu funkcji dopasowania, reprodukcji osobników, modyfikowaniu operatorami genetycznymi lub ewentualnych innych działaniach (np. porównywaniu podobieństwa rozwiązań) ;
- zastosowanie metody kodowania, które nie utrudnia algorytmowi ewolucyjnemu rozwiązania zadania np. przez sztuczne powiększenie przestrzeni rozwiązań lub wprowadzenie dodatkowych stopni swobody i parametrów komplikujących zadanie.

Zwykle trudno jest sprawdzić teoretycznie jakie właściwości posiada wybrana specjalizowana metoda kodowania, dlatego też bardzo często konieczne jest sprawdzenie doświadczalne właściwości kilku obiecujących metod i dokonanie wyboru najlepszej na tej podstawie lub też wybór "mniejszego zła", jeśli każda z nasuwających się metod ma pewne wady i zalety.

W zadaniach optymalizacji ciągłej problem wyboru struktury osobnika sprawia mniej problemów, gdyż jest najczęściej narzucony przez specyfikę problemu: jego wymiarowość i ewentualnie występujące w nim ograniczenia. Z reguły stosowane metody kodowania używają wektorów lub macierzy liczb zmiennoprzecinkowych jako reprezentacji rozwiązań. W praktyce już dość rzadko stosuje się tu kodowanie binarne (stałoprzecinkowe, kod NKB lub Graya) współrzędnych wektorów budujących rozwiązania, gdyż zapewniają one znacznie mniejszą dokładność obliczeń niż kodowanie zmiennoprzecinkowe. Oczywiście można ten problem przezwyciężyć stosując odpowiednio długie ciągi kodowe, lecz to z kolei mocno spowalnia obliczenia. Jeśli więc rozwiązywany problem nie ma jakichś bardzo specyficznych wymagań, to zdecydowanie łatwiej jest użyć kodowania zmiennoprzecinkowego w przypadku optymalizacji zadań w R^n.

W przypadku zadań optymalizacji dyskretnej lub mieszanej dyskretno-ciągłej sprawa jest zdecydowanie bardziej skomplikowana. Możliwości wykorzystania różnych struktur danych do zakodowania rozwiązania problemu są ogromne i ograniczone głównie względami praktycznymi związanymi z kosztownością obliczeniową i ewentualnie zajętością pamięci przy zbyt złożonych tworach. Najczęściej stosowane metody to

- wektory binarne, liczb całkowitych, zmiennoprzecinkowych lub bardziej skomplikowanych struktur o stałej lub zmiennej długości.
- macierze złożone z elementów podobnych jak w przypadku wektorów, czasem również o zmiennych wymiarach;
- listy najczęściej o zmiennej długości, zbudowane z węzłów zawierających dane rożnych typów, często złożonych jak podlisty, wektory itp;
- drzewa (binarne i rzadziej wyższych rzędów), zawierające w węzłach dane binarne, całkowite, zmiennoprzecinkowe lub bardziej skomplikowane;

- pewne kompilacje wyżej wymienionych struktur danych o elementach całkowitych, binarnych, rzeczywistych lub zawierających kilka rodzajów wymienionych danych.

Należy jeszcze dodać, że zarówno w przypadku ciągłym jak i dyskretnym osobnik należący do populacji rozwiązań może posiadać jeszcze dodatkowe dane, wymagane przez zastosowaną wersję algorytmu ewolucyjnego (np. strojone na bieżąco pewne parametry algorytmu lub flagi określające wykorzystanie już tego osobnika do jakiegoś celu np. do krzyżowania), dane potrzebne do tworzenia i ewentualnej modyfikacji struktury danych tworzącej osobnika lub też dane wykorzystywane przez niektóre specjalizowane operatory genetyczne (np. wartość aktualnej funkcji celu osobnika) po to, aby nie tracić czasu na ich szukanie lub obliczanie.

Wymienione powyżej typowe możliwości kodowania osobników zapewne nie wyczerpują bogactwa możliwych rozwiązań. Ilustrują jednak złożoność problemu wybrania odpowiedniej reprezentacji problemu, najlepiej dopasowanej do konkretnego zadania. Z reguły możliwe jest wybranie kilku możliwości dla jednego zadania, ale nie wszystkie z nich będą prowadzić do zbudowania naprawdę efektywnego algorytmu ewolucyjnego.

Wybrana metoda kodowania rozwiązań bardzo często ma duży wpływ na projektowane operatory genetyczne. W przypadku występującego w naturze algorytmu genetycznego, któremu podlegają wszystkie istoty żywe na ziemi, „rozwiązania" kodowane są przy pomocy sekwencji czterech zasad azotowych w DNA (istnieją proste organizmy posługujące się w tym celu RNA, substancją pokrewną do DNA), a środowisko w którym się to dzieje i procesy zachodzące w czasie podziału i rozmnażania komórek decydują o tym, że powstała zamierzona przez naturę operacja krzyżowania i raczej jako pewien efekt uboczny czasem zachodzi mutacja replikowanych genów. Bardzo rzadko pojawiają się też przy okazji powielania DNA pewne inne, również raczej niezamierzone działania takie jak odwrócenie fragmentu łańcucha (inwersja), dołączenie jakichś przypadkowych cząstek DNA lub też

zagubienie pewnych jego fragmentów. W przyrodzie wszystkie wymienione sytuacje poza krzyżowaniem prowadzą do błędów w przetwarzaniu DNA. Błędy te są najczęściej śmiertelne dla tak zmienionych komórek, jednakże okazuje się, że w pewnych, bardzo rzadkich sytuacjach prowadzą one do uzyskania przez komórki lub też organizmy potomne pewnych nowych cech, które zdecydowanie powiększają szanse na przeżycie tak zmodyfikowanych komórek lub organizmów potomnych. Natura posługując się takim materiałem jak DNA do przekazywania informacji, jednocześnie ustanowiła zasób możliwych operacji wykonywanych na tym materiale. Jest możliwe, że w naturze inne operacje na DNA nie są możliwe lub są niezwykle mało prawdopodobne lub też są bardzo zawodne i trudne do odpowiedniego sterowania. Nie wiadomo, czy możliwe jest użycie w charakterze podobnym do DNA jakiejś innej substancji (poza jak już wspomniałem marginalnym użyciem RNA). Być może, że tak, jednakże w przypadku takich warunków, jakie panują na ziemi (temperatura, ciśnienie, obecność wody i atmosfery oraz brak silnych promieniowań jonizujących) ta substancja sprawdziła się najlepiej.

W przypadku komputerowych algorytmów genetycznych jest do dyspozycji zupełnie inny rodzaj nośnika informacji, a właściwie jest bardzo dużo możliwości użycia różnych metod kodowania. Dlatego też, jeśli już zostanie wybrana pewna metoda, to konieczne jest użycie do jej modyfikacji operatorów genetycznych, które dobrze będą współgrały z wybraną metodą i dodatkowo z rozwiązywanym zadaniem. Natomiast zupełnie niekonieczne jest nadawanie im na siłę formy lub podobieństwa do metod występujących w naturze. Nie ma też potrzeby stosowania na siłę metody kodowania binarnego, która w bardzo niewielkim stopniu przypomina metodę biologiczną, a już na pewno nie jest metodą uniwersalną, zdolną i odpowiednią do zakodowania wszelkich możliwych zadań. Zarówno zastosowana przez naturę metoda wykorzystująca DNA, jak i kodowanie binarne są tylko pewnymi możliwymi wariantami wykorzystania metod ewolucyjnych, dopasowanymi do konkretnych zadań i środowisk, a nie jedynymi możliwymi i uniwersalnymi sposobami ich zastosowania.

Uniwersalna jest tu metoda ewolucyjna polegająca na budowaniu z prostych cegiełek kodujących rozwiązania, coraz bardziej zaawansowanych tworów (coraz lepszych rozwiązań) dzięki ich modyfikacji przez operatory genetyczne (ewolucyjne) i ukierunkowującemu działaniu zastosowanej metody selekcji osobników (rozwiązań) do następnych pokoleń.

3. Operatory ewolucyjne i metody ich projektowania

Opracowując dobrze dopasowaną do specyfiki problemu metodę kodowania rozwiązań można w znacznym stopniu ułatwić pracę operatorom genetycznym, ale będzie tak pod warunkiem dobrego dopasowania do siebie tych dwóch elementów algorytmu ewolucyjnego. Głównym zadaniem operatorów genetycznych jest przeszukiwanie przestrzeni rozwiązań danego zadania. Ponieważ z zasady algorytmy ewolucyjne używane są do rozwiązywania zadań w których dokładne przeszukanie całej przestrzeni rozwiązań nie jest możliwe z uwagi na jej rozmiar, dlatego też zestaw operatorów w algorytmie rozwiązującym dany problem powinien być na tyle bogaty, aby poradzić sobie z wykryciem obiecujących miejsc, a następnie, po zlokalizowaniu obszaru, gdzie najprawdopodobniej znajduje się poszukiwane ekstremum, optymalizację lokalną w celu znalezienia samego ekstremum. Tak sytuacja wygląda w idealnym przypadku, a w praktycznie algorytm często utyka w ekstremach lokalnych i takim rozwiązaniem trzeba się zadowolić z uwagi na ograniczoną długość trwania obliczeń. Niestety specyfika metod ewolucyjnych jest taka, że nigdy nie będzie można mieć pewności znalezienia ekstremum globalnego przez algorytm ewolucyjny przy zastosowaniu skończonej liczby iteracji. Czyli znalezione rozwiązania właściwie zawsze należy traktować jako suboptymalne. Ale w wielu przypadkach jest to jak najbardziej akceptowalne działanie i właściwie każda praktyczna metoda używana do rozwiązania odpowiednio trudnego zadania również daje tylko rozwiązania przybliżone. Oczywiście mogą się zdarzyć sytuacje, w których taką pewność można mieć np. przy rozwiązywaniu jakiegoś szczególnego zadania, bądź też stosując odpowiednio zaprojektowane operatory genetyczne,

dające taką gwarancję, ale w ogólnym przypadku na pewno nie można mieć gwarancji znalezienia optimum algorytmem ewolucyjnym.

Algorytmy ewolucyjne mają tę właściwość, że dają gwarancję znalezienia ekstremum globalnego z prawdopodobieństwem dążącym do jedynki przy liczbie iteracji algorytmu dążącej do nieskończoności. Tak wypowiada się na temat możliwości wykorzystania algorytmów ewolucyjnych matematyka (oczywiście przy zachowaniu odpowiednich założeń). W praktyce wiadomo, że nikt nie może zapewnić nieskończonej liczby iteracji do znalezienia ekstremum globalnego, a w zasadzie zawsze konieczne jest uzyskanie akceptowalnego wyniku jak najszybciej. W tej sytuacji bardzo duże znaczenie ma dobry wybór operatorów genetycznych, użytych w algorytmie. Mając nieskończenie dużo iteracji do dyspozycji, prawdopodobnie bardzo wiele zestawów operatorów zadziałałoby (trudno jednakże to sprawdzić), ale w przypadku, gdy chcemy uzyskać jak najlepsze wyniki w krótkim czasie (a tak jest zazwyczaj), to problem odpowiedniego (trudno powiedzieć, że optymalnego) wyboru operatorów genetycznych jest bardzo istotny dla dobrego działania metody.

Tradycyjnie operatory ewolucyjne dzielone są na podstawie ich właściwości na eksploracyjne i eksploatacyjne. Właściwie w przypadku często obecnie stosowanych operatorów specjalizowanych podział ten ma znaczenie nieco historyczne, gdyż często trudno jest określić właściwy lub przeważający charakter danego operatora, to jednak dla zrozumienia idei i działania algorytmów ewolucyjnych pojęcia te są bardzo ważne. W szczególności łatwo było przyporządkować charakter działania do dwóch operatorów zaproponowanych przez twórców algorytmów genetycznych. Mutacja (zwana też obecnie często perturbacją) ma właściwości eksploracyjne, gdyż modyfikuje rozwiązanie w sposób dość dowolny wprowadzając do niego zupełnie nowe, nie występujące często dotąd w populacji cechy. Daje ona możliwość szybkiego przemieszczenia rozwiązania w zupełnie inny rejon przeszukiwanego obszaru, umożliwiając np. opuszczenie optimum lokalnego. Tak więc eksploracja to możliwość przemieszczania się w zupełnie nowe obszary przestrzeni

rozwiązań i nabywanie nowych cech przez osobniki populacji. Z kolei krzyżowanie (zwane też rekombinacją) tworzy rozwiązanie (lub rozwiązania) będące w jakimś stopniu mieszaniną cech rodziców, czyli rozwiązań z których powstało. Ma ono wobec tego wybitnie charakter eksploracyjny, gdyż nie powoduje powstawania całkiem nowych rozwiązań, a jedynie umożliwia łatwe wytworzenie (rekombinację) najlepszych cech z istniejącego garnituru genów w populacji. Eksploracja to wykorzystanie tego co już zostało znalezione, tak aby znaleźć najlepsze rozwiązanie w zlokalizowanym obszarze, czyli optymalizacja lokalna. Działanie algorytmu ewolucyjnego polega na zrównoważonym wykorzystaniu tych cech operatorów tak, aby operatory eksploracyjne umożliwiły znalezienie obszaru w którym znajduje się optimum, a operatory eksploatacyjne umożliwiły dokładne zlokalizowanie tego optimum.

Pierwsze algorytmy genetyczne stosowały bardzo ograniczony zestaw operatorów. Podobnie jak w przypadku kodowania, starano się jak najbardziej naśladować naturę i stosowano krzyżowanie i mutację, a w niektórych przypadkach tylko mutację (np. strategie ewolucyjne). Oczywiście metody z takimi operatorami nie mogły dobrze rozwiązywać dowolnych zadań, podobnie jak kodowanie binarne nie mogło dać dobrych wyników w kodowaniu dowolnych zadań. Dzięki bardzo prostym symulacjom komputerowym można potwierdzić to, co wydaje się być oczywiste: im bogatszy jest zestaw operatorów genetycznych, tym łatwiej rozwiązać postawione zadanie, tym mniej iteracji potrzeba do znalezienia dobrego wyniku i tym lepszy wynik można osiągnąć. Zjawisko to można wytłumaczyć tym, że każdy z zastosowanych operatorów w inny nieco sposób modyfikując rozwiązanie, ułatwia wygenerowanie osobników o zupełnie nowych cechach, zwiększa różnorodność populacji, lub też wprowadza dodatkowe szybsze metody optymalizacyjne (heurystyki) zwiększające prawdopodobieństwo znalezienia dobrych rozwiązań lub też wprowadzające szybkie metody optymalizacji lokalnej. Dodatkowo w trakcie obliczeń ewolucyjnych często można zaobserwować ciekawy efekt „wyczerpywania" się możliwości

operatora. Najczęściej dotyczy lub też najłatwiej to zaobserwować na przykładzie operatorów z wbudowanymi heurystykami. Polega to na tym, że jeśli dany operator jest często używany wiele razy po sobie lub w krótkim czasie dla osobników pochodzących od jednego przodka, to nie potrafi już korzystnie zmodyfikować rozwiązania. Jest to często związane z tym, że po prostu dany operator wykonał już to co mógł zdziałać w danym momencie obliczeń (np. jakiś sposób optymalizacji lokalnej) i dalsze jego stosowanie nie ma w tym momencie sensu. Natomiast po pewnym czasie, kiedy to inne operatory wytworzyły zupełnie nową sytuację w rozkładzie populacji, jego rola znów będzie duża. Najlepsze wyniki otrzymuje się, jeśli operatory zmieniają się z iteracji na iterację, choć oczywiście te najlepsze powinny występować częściej. Duża liczba operatorów[1] dobrze zapobiega zjawisku wyczerpania operatora i powoduje coś w rodzaju wzajemnego potęgowania efektów działania operatorów.

Oczywiście zawsze można skonstruować „złośliwe" operatory, które będą pogarszały rozwiązanie lub też w najlepszym przypadku nie będą robiły nic i wtedy oczywiście algorytm nie będzie działać lepiej, ale jeżeli tylko zaprojektowane operatory będą posiadały możliwość poprawy rozwiązania, to na pewno powinny coś pozytywnego zdziałać.

Przy zastosowaniu niestandardowych operatorów lub większej ich liczby, powstaje problem kiedy i jak je wykonywać. Metody standardowe dotyczą tylko typowych operatorów i metod ich wykonywania i nie dają uniwersalnych narzędzi do tego celu – temu zagadnieniu poświęcono wiele prac do których odsyłam zainteresowanego tym tematem czytelnika: [Davi1989], [Davi1990], [Juls1995], [Mula1999],[Nieh2001],[Stan1999], [Stan2000].

Jak już to zostało wcześniej wspomniane, tradycyjnie dzieli się operatory genetyczne na eksploracyjne i eksploatacyjne. Jednakże przy zastosowaniu operatorów specjalizowanych

---

[1] Pojęcie duża liczba operatorów jest bardzo nieprecyzyjne, ale ma oznaczać ono - więcej niż standardowe 2. Zbyt duża ich liczba zapewne też nie jest wskazana, gdyż za wiele czasu będzie trwać ich konstruowanie. W przedstawianych w tej pracy przykładach liczba wykorzystywanych operatorów wynosi od 5 do 12.

lub opartych na heurystykach i wiedzy o rozwiązywanym problemie taki podział może być trudny do przeprowadzenia, gdyż niełatwo jest jednoznacznie określić jaki charakter ma dany operator. Do celów projektowych można pokusić się o przeprowadzenie innego podziału, na operatory działające w sposób czysto losowy (bez żadnych dodatkowych heurystyk) i operatory oparte na heurystykach i wiedzy o rozwiązywanym problemie. Z dużym uproszczeniem można przyjąć, że najczęściej te pierwsze mają charakter bardziej eksploracyjny, a te drugie bardziej eksploatacyjny, ale nie jest to ścisła reguła (krzyżowanie jest na pewno operatorem z kategorii czysto losowych, a ma właściwości eksploatacyjne). Na pewno zastosowanie obydwu grup znacznie polepsza działanie algorytmu ewolucyjnego.

Operatory czysto losowe to metody oparte na zupełnie przypadkowych modyfikacjach rozwiązań, niewykorzystujące żadnej dodatkowej wiedzy o rozwiązywanym problemie, a jedynie ewentualnie dopasowane do specyfiki metody kodowania, ograniczeń itp. Modyfikacje, jakie wprowadzają one do podlegających ich działaniu osobników polegają na czysto losowych zmianach pewnych, najczęściej losowo wybranych części rozwiązań. Są one dość uniwersalne i z niewielkimi zmianami można je stosować w większości rozwiązywanych zadań. Można do nich zaliczyć:

- mutację – losowo zmieniającą wartość pewnego atrybutu (losowo wybranego) rozwiązania na inną, najczęściej dopuszczalną, wylosowaną zgodnie z pewnym rozkładem (np. jednostajnym, Gaussa);
- krzyżowanie – wymiana losowo wybranych fragmentów rozwiązań pomiędzy pewną liczbą (najczęściej dwoma) rozwiązań w populacji; w przypadku optymalizacji z rzeczywistoliczbowym kodowaniem pod pojęciem krzyżowania może się także znaleźć kombinacja liniowa osobników, uśrednienie itp.;
- inwersja – przestawienie kolejności w losowo wybranym fragmencie łańcucha symboli kodujących rozwiązanie;

- przestawienie – przestawienie losowo wybranych fragmentów rozwiązań w obrębie jednego osobnika;
- operatory wielokrotne, czyli operatory kilkukrotnie (liczba powtórzeń może być losowana) powtarzające któreś z wymienionych wcześniej operatorów losowych.

Oczywiście nie każda z tych operacji może być zastosowana w każdym rozwiązywanym zadaniu z uwagi na istniejące ograniczenia lub metodę kodowania. Operatory działające w sposób losowy, zastosowane jako jedyne do rozwiązania zadania, często działają bardzo powoli i przynoszą bardzo niewielkie poprawy, jednakże są bardzo ważnym elementem w algorytmie ewolucyjnym, gdyż to głównie dzięki ich działaniu możliwe jest znalezienie nowych obiecujących obszarów poszukiwań, gdy operatory heurystyczne nie potrafią już nic nowego odnaleźć.

Wymienione operatory nie wyczerpują z pewnością wszystkich możliwości, gdyż istnieją metody posługujące się bardzo specyficznymi metodami operacji wzorowanymi np. na atakach wirusów i reakcjach systemu immunologicznego organizmów żywych, broniącego się przed tego typu infekcjami i innych.

Operatory heurystyczne lub oparte na wiedzy o rozwiązywanym problemie są najczęściej mocno zależne od rozwiązywanego problemu. Ale również i tu można wydzielić pewne klasy operatorów o szerokich możliwościach zastosowania, jak i bardzo szczegółowych, odpowiednich tylko do konkretnego problemu:

- operatory oparte na znajomości funkcji celu zadania;
- operatory oparte na znanych metodach optymalizacji;
- operatory specyficzne dla konkretnego zadania.

Operatory oparte na znajomości wartości funkcji celu zadania są klasą dość uniwersalną i możliwą do zastosowania w bardzo wielu typach rozwiązywanych problemów. Jeśli można porównać wartość funkcji celu rozwiązania przed i po dokonaniu zmiany jakimś operatorem (losowym, bądź heurystycznym) i można proces ten powtórzyć kilkukrotnie,

wybierając na końcu najlepszy wariant, to powstanie dość uniwersalny operator oparty na wiedzy o zadaniu, ale właściwie bardzo mało zależny od konkretnego zadania. W ten sposób można często uzyskać znaczne przyspieszenie obliczeń przy małym nakładzie dodatkowych działań. Metoda taka będzie nieskuteczna w przypadku zadań w których ocena nowego rozwiązania jest bardzo kosztowna obliczeniowo lub jest ograniczony dostęp do możliwości oceny rozwiązania (np. jest nałożony limit na liczbę wywołań funkcji celu). Jednakże w tych sytuacjach można często wykorzystać dane z poprzednich iteracji, np. zapamiętując także poprzednie rozwiązanie (rodzicielskie) i jego wartość funkcji celu, a następnie po analizie danych bieżących i poprzednich, doszukać się jakiegoś trendu i spróbować zmodyfikować rozwiązanie bieżące wykorzystując tę informację. Taka wersja operatora może być szczególnie łatwa do wykorzystania w przypadku optymalizacji ciągłej.

Podobnie dość uniwersalną metodą projektowania operatorów heurystycznych może być zastosowanie jako operatora genetycznego jakiejś znanej i niezbyt skomplikowanej metody optymalizacji lub jej części np. kilku iteracji metodą gradientową w przypadku optymalizacji ciągłej, elementów metody tabu-search, simulated annealing, 2-opt, 3-opt lub innych, zależnie od konkretnego przypadku.

Osobnym zagadnieniem jest problem takiego zaprojektowania operatorów, aby można było automatycznie uwzględnić pewne ograniczenia narzucone na zadanie, bez uciekania się do kosztownych algorytmów naprawy osobników, czy też stosowania funkcji kary za przekroczenie ograniczeń. Uwzględnienie ograniczeń narzuconych w zadaniu przez odpowiednie zaprojektowanie operatorów nie zawsze jest możliwe, ale zawsze dobrze jest spróbować, czy nie jest to realizowalne, gdyż często, jeśli się uda, daje duży zysk w postaci znacznego przyspieszenia obliczeń i często znacznego okrojenia przestrzeni poszukiwań.

W ogólnym przypadku klasa operatorów heurystycznych jest ograniczona jedynie wyobraźnią projektującego i możliwościami obliczeniowymi sprzętu komputerowego. Można stosować łącznie z korzyścią dla procesu optymalizacji genetycznej metody

charakterystyczne tylko dla rozwiązywanego zadania, jak i przedstawione wcześniej metody uniwersalne wraz z metodami losowymi.

Warto pamiętać, że w dziedzinie operatorów heurystycznych zawsze bardzo dobre rezultaty daje zastosowanie operatorów, które posiadają wbudowane pewne proste reguły, dzięki którym osiągają lepsze rezultaty od metod czysto losowych, gdyż reguły te zwiększają prawdopodobieństwo wygenerowania lepszego rozwiązania, koncentrując poszukiwania na bardziej obiecujących rejonach przestrzeni rozwiązań. Oczywiście nie w każdym przypadku przynosi to oczekiwany skutek, dlatego też metody czysto losowe również są potrzebne, aby pochopnie nie porzucić pozornie mało obiecujących rejonów, ale w zdecydowanej większości przypadków użycie operatorów heurystycznych daje bardzo dobre efekty. Sposoby zastosowania takich metod znajdują się w prezentowanych dalej przykładach.

Idea tworzenia algorytmu ewolucyjnego z wieloma specjalizowanymi operatorami pomaga stworzyć bardzo efektywne narzędzie optymalizacyjne, oczywiście trudno jest wymyślić zestaw skutecznych operatorów genetycznych do rozwiązywanego zadania w krótkim czasie. Zaprojektowanie zestawu naprawdę dobrze działających operatorów wymaga pewnego procesu iteracyjnego w którym najpierw powstają proste wersje operatorów np. tylko losowych, a następnie analizując otrzymane wyniki i zachowanie algorytmu można wywnioskować co jeszcze można usprawnić i jakie operatory można zaprojektować, aby całość działała zgodnie z oczekiwaniami.

4. Przykład I – wyznaczanie optymalnego sygnału typu bang-bang do estymacji parametru w obiekcie pierwszego rzędu

W opisywanym przykładzie celem rozpatrywanego zadania jest znalezienie optymalnego sterowania umożliwiającego najlepszą (z minimalnym błędem) estymację parametru układu pierwszego rzędu. Pokazany tu opis zagadnienia jest tylko szkicem metody, koniecznym do zrozumienia rozwiązywanego ewolucyjnie fragmentu zadania, natomiast szczegóły dotyczące całego zagadnienia można znaleźć np. w pracy [Naho2005].

4.1 Opis problemu

Obiekt pierwszego rzędu opisywany jest następującym równaniem różniczkowym:

$$T\frac{dx}{dt}+x(t)=u(t-t_d) \quad \text{dla} \quad x(t_0)=0 \tag{1}$$

gdzie:

T – wartość poszukiwana;

$u(t)$ – sygnał wejściowy;

t_d –opóźnienie;

$x(t)$ – sygnał wyjściowy.

Ogólne rozwiązanie równania opisującego zachowanie obiektu ma postać:

$$x(t)=Ce^{-\frac{t}{T}}+\frac{1}{T}\int_{t_a}^{t_N} e^{-\frac{t-\tau}{T}}u(\tau-t_d)d\tau \tag{2}$$

W rozpatrywanym zadaniu można założyć, że sygnał wejściowy $u(t)$ jest odcinkami stały i może się zmieniać tylko w dyskretnych, równoodległych chwilach czasu. Po dyskretyzacji równania (2) i założeniu zerowych warunków początkowych (C=0) otrzymuje się równanie (3):

$$x_n = ax_{n-1}+bu_{n-k} \tag{3}$$

gdzie:

$x_n=x(t_n)$;

$\Delta=t_n-t_{n-1}$;

$a=e^{-\Delta/T}$;

$k=t_d/\Delta+1$;

$b=1-a$.

Wyjście obiektu mierzone jest z błędem:

$$y_n = x_n+\varepsilon_n \tag{4}$$

Po wprowadzeniu operatora przesunięcia z i podstawieniu wzoru (3) do (4) otrzymuje się wzór na błąd modelu:

$$e_n = y_n - \frac{1-\alpha}{1-\alpha z^{-1}} u_{n-k} \tag{5}$$

gdzie:

zmienna α odpowiada parametrowi a;

błąd e_n odpowiada zakłóceniu ε_n,

ε_n jest ciągiem niezależnych zmiennych losowych o rozkładzie $N(0,\sigma^2)$.

Do wyznaczenie optymalnego sterowania minimalizującego błąd estymacji parametru wykorzystuje się następującą macierz:

$$M(p) = E_{Y|p}\left(\left(\frac{\partial \ln f(Y|p)}{\partial \hat{p}}\right)\left(\frac{\partial \ln f(Y|p)}{\partial \hat{p}}\right)^T\right) \tag{6}$$

gdzie:

$Y=\{y(t_1), y(t_2), ..., y(t_N)\}$ – wielowymiarowa zmienna losowa w zaplanowanych momentach pomiarów;

N – liczba pomiarów;

$f(Y|p)$ – warunkowa gęstość prawdopodobieństw Y przy zadanym wektorze poszukiwanych parametrów p;

$E_{Y|p}$– warunkowa wartość oczekiwana.

Aby rozwiązać postawione zadanie należy minimalizować pewną funkcję skalaryzującą macierz $M^{-1}(p)$. W naszym przypadku macierz $M(p)$ jest skalarem, gdyż wektor p jest jednoelementowy.

Po odpowiednich przekształceniach otrzymujemy macierz informacyjną o postaci:

$$M(a) = \frac{1}{\sigma^2}\sum_{n=k}^{N}\left(\frac{\partial e_n}{\partial a}\right)^2 \tag{7}$$

gdzie:

$$\frac{\partial e_n}{\partial a} = \frac{1 - z^{-1}}{1 - 2a z^{-1} + a^2 z^{-2}} u_{n-k} \tag{8}$$

a następnie powstaje wzór (9), będący poszukiwaną funkcją celu zadania:

$$F = \max_{u_{n-k}} \sum_{n=k+1}^{N} \left\{ a^n \cdot \sum_{l=k+1}^{n} a^{-l} \left[1 + (n-l) \cdot (1 - a^{-1}) \right] \cdot u_{l-k} \right\}^2 \tag{9}$$

gdzie:

n - krok sterowania (n=1,2....N-k);

u - sygnał sterujący $u \in \{0, 1\}$ – optymalny sygnał sterujący jest na ograniczeniu;

k - opóźnienie;

$a = exp(-\Delta/T)$;

Δ - czas trwania 1 kroku;

T - identyfikowany parametr.

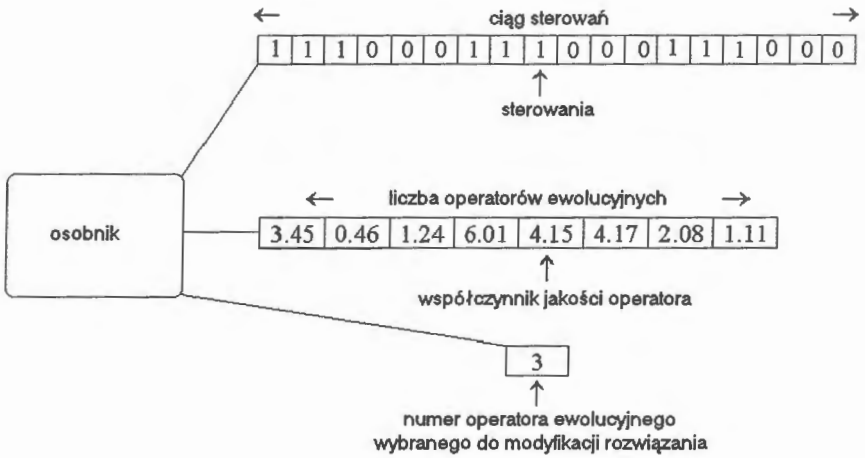
Z warunków zadania wiadomo, że optymalny sygnał wymuszający, umożliwiający identyfikację obiektu ma postać fali prostokątnej, której wartości, bez utraty ogólności rozwiązania, można przyjąć jako 0 i 1. Natomiast optymalne sekwencje sygnałów można znaleźć optymalizując funkcję daną równaniem (9). Znalezienie, metodą sprawdzania wszystkich możliwości, sygnałów już o długościach rzędu 20 trwa bardzo długo – należy sprawdzić 2^{20} możliwości. Do rozwiązania takiego problemu świetnie nadaje się natomiast metoda ewolucyjna.

4.2 Metoda ewolucyjna rozwiązania zadania z kodowaniem binarnym

W pierwszym podejściu zastosowano oczywistą i wynikającą z warunków zadania metodę kodowania binarnego (Rys. 1) z sekwencją sygnałów o stałej długości N (start zawsze od jedynki) i odpowiednio dobrany zestaw operatorów genetycznych:

- **mutacja I** - typowa mutacja binarna;

- **mutacja II** - wyszukuje miejsca, gdzie następuje zmiana 0-1 lub 1-0 i tam przeprowadza negację bitu;
- **przestawienie** - losowe przestawienie fragmentów kodu;
- **operacja 2-opt** - wykorzystuje tzw. metodę 2-optymalną do poprawy rozwiązania;
- **„inteligentne" przestawienie** - przestawienie fragmentów kodu ze sprawdzeniem, czy daje to pozytywny efekt; jeśli nie, to przestawienie jest anulowane;
- **operacja „poszukiwania cyklu"** - poszukuje najczęściej występujących długości cykli i tak modyfikuje rozwiązanie, aby wystąpiły w nim jak najczęściej.



Rys. 1 Schemat osobnika w wersji binarnej.

Symulacje komputerowe pokazały jednak szereg niedostatków takiej metody rozwiązania problemu. Obliczenia trwały dość długo, a wyniki nie były zbyt dokładne. Wynikało to z tego, że otrzymywany sygnał wynikowy był właściwie okresowy o ewentualnie nieco zmiennym okresie trwania, wobec czego zmiany wartości sygnałów w dowolnym miejscu okresu nic pozytywnego nie wnosiły, a wręcz przeciwnie psuły rozwiązania. Jednocześnie należało w taki sam sposób przetwarzać 0 i 1, a wyraźnie widać z

postaci funkcji celu, że obliczanie skomplikowanych wyrażeń, mnożonych następnie przez 0 jest tylko stratą czasu. Dodatkowo rozwiązania zajmują dość dużo miejsca w pamięci.

4.3 Metoda ewolucyjna rozwiązania zadania z kodowaniem całkowitoliczbowym

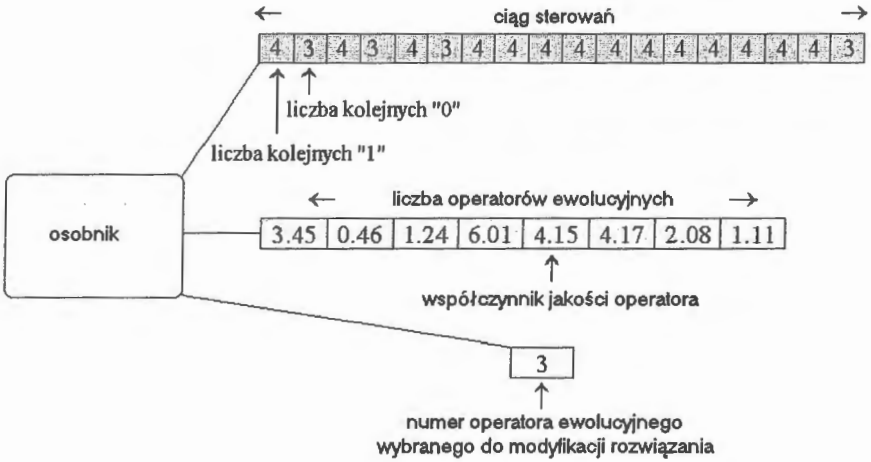
Wobec tych niedostatków wersji z kodowaniem binarnym, w drugiej wersji metody ewolucyjnej zastosowano kodowanie całkowitoliczbowe. Kolejne liczby oznaczały w nim na przemian liczbę jedynek i liczbę zer (pierwsza sekwencja oznacza zawsze liczbę jedynek, aby eksperyment zaczynał się „użytecznym" sygnałem, a nie dodatkowym opóźnieniem) występujących pod rząd w sygnale. Schemat takiego kodowania pokazuje Rys. 2. W metodzie tej liczba elementarnych cegiełek budujących rozwiązanie (liczb zer lub jedynek) nie jest stała, jednakże suma tych liczb ma wartość *N*, tak jak długość rozwiązania w metodzie pierwszej.

W tej metodzie kodowania każda modyfikacja którejś z wartości powoduje zmianę długości cyklu, a nie jakieś przypadkowe jego zaburzenia, jednocześnie bardzo łatwo jest pomijać od razu długie sekwencje zer w obliczeniach. Dodatkowo takie zakodowanie problemu powoduje zajęcie znacznie mniejszej ilości pamięci w komputerze, gdyż występujące najczęściej cykle dłuższe od 2 dają się zapisać krócej niż w przypadku metody binarnej, a nawet najkrótsze cykle o długości 2 nie będą zapisywane jako dłuższe niż dla metody podstawowej. Daje to dodatkowy zysk w postaci mniejszej liczby kopiowanych bajtów przy przetwarzaniu populacji i dodatkowy zysk w szybkości działania metody.

W ulepszonej wersji metody ewolucyjnej zastosowano też nieco zmodyfikowany zestaw operatorów genetycznych:

- **przestawienie** (pojedyncze i wielokrotne) – wymiana długości cykli na losowo wybranych pozycjach;
- **przesunięcie** (pojedyncze i wielokrotne) – zwiększenie długości jednego cyklu kosztem innego;
- **dodanie kolejnego cyklu** - wiąże się to ze skróceniem pozostałych;

- **odjęcie jednego cyklu** - wiąże się to z wydłużeniem pozostałych;
- **wymiana** nowego osobnika o wylosowanej długości cykli;
- **inteligentne wymiana osobnika** - testowanych jest kilka długości cyklu o parametrach zbliżonych do wartości wylosowanej, najlepsza wersja pozostaje.



Rys. 2. Schemat osobnika w wersji całkowitoliczbowej.

Symulacje komputerowe drugiej wersji algorytmu ewolucyjnego pokazały, że dzięki zastosowanym modyfikacjom zadanie liczyło się ok. 10-krotnie szybciej niż w wersji pierwszej przy porównywalnej jakości otrzymywanych rozwiązań. Przyrost prędkości obliczeń wynikał nie tylko ze zmniejszenia rozmiarów rozwiązań i przyspieszenia obliczania funkcji celu, ale też i z wykorzystania znacznie mniej skomplikowanych operatorów, które przy zastosowanej metodzie kodowania znacznie szybciej odnajdowały właściwe rozwiązania niż bardziej specjalizowane wersje zastosowane w metodzie pierwszej. Tak, więc dzięki zastosowaniu odpowiedniej metody kodowania i przystosowanych do niej operatorów ewolucyjnych można uzyskać naprawdę znaczny przyrost szybkości działania i dokładności metody. Nie zawsze możliwe jest przyjęcie odpowiedniego rozwiązana za pierwszym

podejściem – często potrzebny jest proces iteracyjny, w którym projektant algorytmu ewolucyjnego po analizie wyników i sposobie działania algorytmu będzie mógł wybrać najlepsze rozwiązanie. Przykład ten pokazuje również, że przyjmowanie tradycyjnej metody kodowania binarnego, nawet w przypadku w którym wydaje się to być w pełni uzasadnione, nie zawsze jest dobre dla efektywności działania algorytmu ewolucyjnego.

5. Przykład II – optymalizacja funkcji niestacjonarnej w środowisku ciągłym

Optymalizacja w funkcji niestacjonarnych jest dość trudnym zadaniem obliczeniowym. Wymaga ona nie tylko znalezienia optimum (lub suboptimum) funkcji, ale jeszcze nadążania za nim w czasie, gdyż optymalizowana funkcja zmienia swój kształt, a co za tym idzie i położenie optimów w przestrzeni. Zastosowanie AE do rozwiązania tego typu problemów wymaga położenia nacisku na przyspieszenie jego działania i zwiększenie jego zdolności adaptacyjnych (por. [Bran1999], [Cobb1993], [Gold1987]). Metoda z adaptacyjnym doborem operatorów ewolucyjnych wydaje się być dobrym sposobem na polepszenie parametrów AE.

W typowym AE stosuje się operatory ewolucyjne ze sztywno ustalonymi prawdopodobieństwami wyboru operacji modyfikujących populację rozwiązań. Dla stosowanych powszechnie dwóch operatorów: krzyżowania i mutacji stosowano duże, bliskie 1 prawdopodobieństwo wykonania krzyżowania i niewielkie, rzędu 0,1 prawdopodobieństwo mutacji. Wiązało się to ze sposobem działania tych operatorów. Krzyżowanie, operator o charakterze eksploatacyjnym, tworzący potomka będącego rekombinacją cech rodziców powinien działać często, aby prowadzić do tworzenia lepszych rozwiązań. Mutacja, operator o charakterze eksploracyjnym, wprowadzający nowe cechy do osobnika potomnego, powinien być stosowany z umiarem, aby nie zniszczyć tego, co znaleziono dotychczas.

Obecnie stosuje się AE o znacznie wyższym poziomie komplikacji, z kodowaniem rozwiązań i operatorami (często heurystycznymi) dostosowanymi do konkretnych potrzeb zadania. Bardzo trudno jest w takim przypadku ocenić, jaki charakter ma dany operator i jakie

prawdopodobieństwo jego występowania będzie najlepsze zwłaszcza, że dająca najlepsze efekty częstość jego występowania w trakcie obliczeń ewolucyjnych może się zmieniać. Dlatego też zastosowanie metody, w której stosuje się adaptację tych prawdopodobieństw wydaje się bardzo korzystnym rozwiązaniem.

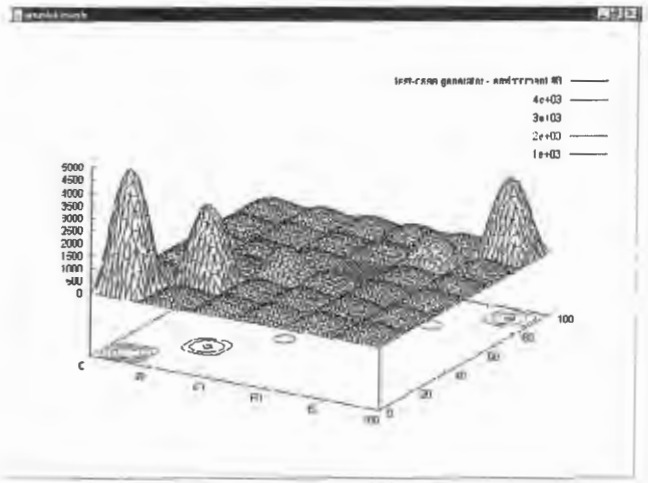
5.1 Optymalizacja w środowisku niestacjonarnym

Optymalizacja funkcji niestacjonarnych, czyli takich, których parametry są zmienne w czasie, jest znacznie trudniejszym przypadkiem niż najczęściej spotykana optymalizacja statyczna. W optymalizacji statycznej należy znaleźć jak najlepsze rozwiązanie i jest na to dowolnie dużo (lub dość dużo) czasu. Ponadto większość wyuczonych lub zauważonych prawidłowości i informacji o optymalizowanym obiekcie nie zmienia się. W przypadku optymalizacji niestacjonarnej ramy czasowe są przeważnie ograniczone, gdyż optimum zmienia swoje położenie. Dodatkowo często żąda się, aby metoda obliczeniowa nadążała za zmieniającym się położeniem optimum. Niestety w przypadku niestacjonarnym informacje o optymalizowanym procesie, pozyskane w trakcie poszukiwania optimum mogą się bardzo szybko dezaktualizować i ich wykorzystanie może prowadzić do złych wyników.

W zagadnieniu optymalizacji obiektów niestacjonarnych można wyróżnić przypadki ze zmiennością cykliczną i ze zmiennością przypadkową. W pierwszym przypadku dobre wyniki rozwiązań można uzyskać stosując metody oparte na pamięci zdarzeń przeszłych, czyli nawiązujące w pewnym sensie do metod optymalizacji statycznej. W drugim przypadku metody takie będą raczej zawodne i istotne jest zwiększenie elastyczności w poszukiwaniu nowych rozwiązań. Te trudności w rozwiązywaniu zadań związanych z optymalizacją niestacjonarną prowadzą do częstego stosowania w nich metod opartych na heurystykach, takich jak symulowane wyżarzanie, czy algorytmy ewolucyjne. Te ostatnie wydają się dysponować szczególnie silnym potencjałem do rozwiązywania takich zadań.

5.2 Zadanie testowe

Do symulacji komputerowych wykorzystane zostało środowisko zaproponowane przez Trojanowskiego i Michalewicza, [Troj1999]. Definiuje ono wielowymiarową przestrzeń (liczba wymiarów jest parametrem metody), podzieloną na wiele hipersześcianów z zawartymi w nich unimodalnymi i paraboloidalnymi funkcjami, wśród których poszukuje się maksimum globalnego. Funkcje z hipersześcianów różnią się wartościami zmiennych w czasie maksimów lokalnych. Zmiany zachodzą cyklicznie. W danej chwili istnieje tylko jedno optimum globalne, w jednym z hipersześcianów. Optima lokalne nie przesuwają się w przestrzeni, a jedynie zmieniają swoją wartość, przybierając większe lub mniejsze wartości. Dla przypadku trójwymiarowego otrzymujemy krajobraz przypominający szachownicę z „górką" o maksimum lokalnym wypadającym w środku każdego pola szachownicy. Wysokości wzniesień są różne i cyklicznie się zmieniają.



Rys. 3. Przykładowy widok środowiska, w którym testowano przedstawione algorytmy

5.3. Zakodowanie rozwiązania i funkcja dopasowania osobnika

Rozwiązywane zadanie jest typowym przykładem optymalizacji funkcji ciągłej w R^n. Osobnik populacji jest więc wektorem liczb rzeczywistych o długości n. Dodatkowo zawiera pewne elementy, wymagane przez zmodyfikowany AE użyty do rozwiązania problemu. Tymi dodatkowymi elementami są:

- wektor liczb rzeczywistych o długości równej liczbie zastosowanych operatorów genetycznych, zawiera on współczynniki jakości operatorów, potrzebne do wyboru operatora;
- wartość bieżącego rozwiązania;
- rozwiązanie poprzednie – rodzicielskie (zakodowane identycznie jak bieżące);
- wartość funkcji celu rozwiązania poprzedniego;
- liczba całkowita – numer aktualnie wybranego operatora genetycznego do modyfikacji rozwiązania;
- flaga (true/false) – identyfikująca, czy dany osobnik był już modyfikowany w danej iteracji algorytmu.

Funkcja dopasowania oparta jest na opisanym wcześniej zadaniu optymalizacji dynamicznej i wprowadza przeskalowanie problemu do zakresu (0,1).

5.4 Operatory ewolucyjne zastosowane do rozwiązania problemu

Efektywność metod opartych na AE polega w dużej mierze na możliwości doboru operatorów ewolucyjnych, dopasowanych do specyfiki rozwiązywanego zadania. W opisywanym przypadku zastosowano następujące operatory:

- mutacja I – wstawienie na wybranej pozycji wektora rozwiązania wylosowanej liczby z zakresu możliwych do przyjęcia wartości w danym wymiarze;
- mutacja II – niewielka perturbacja wartości wektora rozwiązań na wylosowanej pozycji o wartość z przedziału (-5% - +5%);

- mutacja III – działa podobnie jak mutacja I, lecz obowiązkowo dla wszystkich współrzędnych wektora;
- mutacja IV – działa podobnie jak mutacja II, lecz obowiązkowo dla wszystkich współrzędnych wektora;
- mutacja V – zapożyczona z dziedziny algorytmów immuno-genetycznych mutacja wykorzystująca informacje o poprzedniej wartości funkcji celu rozwiązania wg wzoru:

$$\begin{aligned} xc_i' &= xc_i + \alpha \cdot N(0,1) \\ \alpha &= (r_m / \beta) \cdot (domain\_width/2) \cdot exp(-f'(x)) \\ f'(x) &= \frac{f(x) - f_{min}}{f_{max} - f_{min}} \end{aligned} \qquad (10)$$

gdzie xc_i, xc_i' – wartość i-tej współrzędnej rozwiązania przed i po modyfikacji, r_m- współczynnik zakresu mutacji zawarty w przedziale (0,1), $N(0,1)$ – standaryzowany rozkład normalny, β - współczynnik wagi (w symulacjach β=1), $f'(x)$ – znormalizowana poprzednia wartość funkcji dopasowania rozwiązania, f_{min}, f_{max} – minimalna i maksymalna wartość funkcji dopasowania w poprzedniej iteracji AE, *domain_width* – szerokość przedziału, w którym poszukiwane są rozwiązania.

- mutacja VI – standardowa mutacja z modyfikacją współrzędnych wektora rozwiązania przy użyciu wartości wygenerowanych z rozkładu normalnego;
- operator inteligentnej mutacji – na podstawie rozwiązania bieżącego i poprzedniego oraz ich ocen generowany jest hipotetyczny kierunek poprawy, a następnie rozwiązanie jest przesuwane w tym kierunku o pewną niewielką losową wartość;
- krzyżowanie I – wymienia przecięte w wylosowanym miejscu fragmenty wektorów z rozwiązaniami dwóch osobników populacji;
- krzyżowanie II – na wylosowanych pozycjach wektora rozwiązań przeprowadza uśrednienie tych wartości, pochodzących od obu osobników.

5.5. Wyniki symulacji komputerowych

Przeprowadzono symulacje komputerowe dla niestacjonarnego zadania optymalizacyjnego, opisanego w punkcie 3, z podziałem przeszukiwanego fragmentu przestrzeni czterowymiarowej na 64 (4x4x4) hipersześcianów z funkcjami typu paraboloidalnego o nieznanej i zmiennej w czasie wartości optimum lokalnego. Do poszukiwania maksimum globalnego funkcji użyto AE wyposażonego w zestaw operatorów standardowych (mutacja: I, II, krzyżowanie I i II) oraz wszystkich opracowanych do rozwiązania zadania. Do porównania wyników zastosowano współczynnik jakości zwany *offline_error*, zdefiniowany w następujący sposób (por. [Bran1999]):

```
offline_error:=0
for i:=0 to max_iteration
begin
 if i mod τ =0 then
 begin
  f_best:=v_big_number;
  change_optimized_function;
 end;
 modify_and_evaluate_population;
 find(f_curr_best(i));
 if f_curr_best(i)< f_best then f_best:= f_curr_best(i);
 offline_error:= offline_error+f_best;
end;
offline_error:= offline_error/ max_iteration;
```

gdzie τ - oznacza liczbę iteracji, co którą zmieniana jest funkcja celu, f_{best} – najlepsze rozwiązanie znalezione dla bieżącego okresu τ, $f_{curr\_best}(i)$ – rozwiązanie najlepsze w iteracji i, *max_iteration* - liczba iteracji symulacji, *v_big_number* – bardzo duża liczba dodatnia.

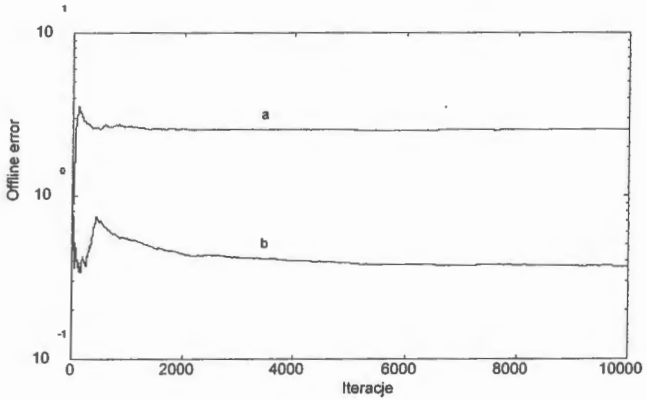
Miara *offline_error* jest dość często używana jako miara dostrojenia algorytmu do poszukiwanego optimum. Współczynnik ten ułatwia ocenę jakości metody rozwiązującej problem. W idealnym przypadku (tzn. gdyby następowało znalezienie optimum w pierwszej iteracji po modyfikacji funkcji celu) powinien wynosić 0, ale szczególnie przez to, że uwzględnia także rozwiązania tuż po przełączeniu funkcji celu ma on znacznie większą

wartość, gdyż algorytm musi mieć trochę czasu na dostrojenie się do nowego „krajobrazu" optymalizowanej funkcji.

We wszystkich symulacjach komputerowych wartość τ wynosiła 10, liczność populacji rodzicielskiej 50 osobników, potomnej 450 osobników. Z obu populacji wybierano kolejną populację rodzicielską (strategia rozwoju populacji $\mu+\lambda$), co nadawało zastosowanej metodzie selekcji cechy selekcji elitarnej (czyli takiej, w której nie można stracić najlepszych znalezionych dotychczas rozwiązań).

Selekcja z cechami elitaryzmu jest rzadko stosowana w optymalizacji problemów niestacjonarnych, gdyż osobniki uznawane za bardzo dobre, po zmianie postaci funkcji celu stają się często bardzo złe. W tej jednak metodzie nie ma to aż takiego znaczenia, gdyż zapamiętywane są także osobniki gorsze, które po zmianie „krajobrazu" stają się dobrym punktem wyjścia do nowych poszukiwań. Wtedy natomiast bardzo pomocna staje się właściwość zachowywania dobrych i różnych rozwiązań bez możliwości ich utraty, wobec czego algorytm ewolucyjny szybko odnajduje nowe optimum. W pracy [Stań2004] porównywano zresztą parametry działania tej metody z innymi typowymi metodami selekcji i wykazywała ona znacznie lepsze właściwości od metod nieelitarnych.

Rys. 4 przedstawia średnią wartość współczynnika błędu *offline_error* wyliczoną z 10 symulacji algorytmu dla przypadku ze standardowymi i specjalizowanymi operatorami (b) i tylko ze standardowymi (a). Z przedstawionych wykresów można wywnioskować, że zastosowanie różnorodnych operatorów specjalizowanych znacząco polepsza efektywność działania algorytmu ewolucyjnego. Ma to szczególne znaczenie w sytuacjach, w których liczy się szybkość obliczeń, czyli np. w przypadku optymalizacji obiektów dynamicznie zmieniających swoje parametry, choć w przypadkach optymalizacji statycznej też nie jest bez znaczenia.



Rys. 4. Wartość współczynnika błędu *offline_error*: a – z zastosowaniem operatorów tradycyjnych, b – z zastosowaniem operatorów specjalizowanych i tradycyjnych.

6. Wnioski

Przedstawione w niniejszej pracy idee dotyczące konstruowania specjalizowanych algorytmów ewolucyjnych umożliwiają tworzenie bardzo szybkich i skutecznych metod ewolucyjnych umożliwiających rozwiązywanie bardzo skomplikowanych zadań, trudnych do rozwiązania metodami tradycyjnymi. Zamieszczone przykłady stanowią ilustrację dla pokazanych metod, a wyniki eksperymentów pokazują, że zaproponowane metody są naprawdę skuteczne.

Literatura

[Arab2001] Arabas J. *Wykłady z algorytmów ewolucyjnych*, WNT,Warszawa, 2001.

[Bran1999] Branke J. *Memory Enhanced Evolutionary Algorithm for Changing Optimisation Problems*, CEC'99, IEEE Press, 1999.

[Cede1997] Cedeno W., Vemuri V. R. *On the Use of Niching for Dynamic Landscapes*, ICEC'97, IEEE Publishing, Inc, 1997.

[Cobb1993] Cobb H., Grefenstette J. J. *Genetic Algorithms for Tracking Changing Environments*, V ICGA'93, Morgan Kauffman, 1993.

[Davi1989] Davis L. *Adapting Operator Probabilities in Genetic Algorithms*, Proc. of the Third International Conference on Genetic Algorithms, 1989.

[Davi1991] Davis L. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[Gold1987] Goldberg D. E., Smith R. E. *Nonstationary Function Optimisation Using Genetic Algorithms with Dominance and Diploidy*, II ICGA, Lawrence Erlbaum Associates, 1987.

[Hada1997] Hadad B. S,. Eick C. F. *Supporting Polyploidy in Genetic Algorithms Using Dominance Vectors*, EP'97, vol. 1213 LNCS, Springer, 1997.

[Juls1995] Julstrom B. A. *What Have You Done for Me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm*, Proc. of the Sixth International Conference on Genetic Algorithms, University of Pittsburgh, 1995.

[Mula1999] Mulawka J. Stańczak J. *Genetic Algorithms with Adaptive Probabilities of Operators Selection*, Proceedings of ICCIMA'99, New Delhi, India, 1999.

[Naho2005] Nahorski Z. Stańczak J. *Optymalny sygnał typu bang-bang do estymacji parametrów w obiekcie pierwszego rzędu*, w: Z. Bubnicki, R. Kulkowski, J. Kacprzyk XV Krajowa Konferencja Automatyki, Tom I, IBS PAN, Warszawa, 2005;

[Nieh2001] Niehaus J. Banzhaf W. *Adaptation of Operator Probabilities in Genetic Programming*, Proc. of 4th EmoGP Conference, Como 2001, Springer, Berlin, 2001, pp. 325-336.

[Stań1999] Stańczak J. *Rozwój koncepcji i algorytmów dla samodoskonalących się systemów ewolucyjnych*, PhD, Politechnika Warszawska, 1999.

[Stań2000] Stańczak J. *Algorytm ewolucyjny z populacją "inteligentnych" osobników*, Materiały IV Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Lądek Zdrój, 2000.

[Stań2004] Stańczak J., Trojanowski K. *Properties of Selection Methods Applied to Non-Stationary Optimization Tasks*, Materiały VII Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Kazimierz Dolny, 2004,str. 171-180;

[Troj1999] Trojanowski K., Michalewicz Z. *Searching for Optima in Non-Stationary Environments*, Proc. Of the 1999 Congress on Evolutionary Computation – CEC'99, IEEE Publ., 1999.

[Troj2003] Trojanowski K. Wierzchoń S. T. *Studying Properties of Multipopulation Heuristic Approach to Non-Stationary Optimisation Tasks*, Proc. of IIPWM'03, 2003.