Anton Smoliński
Wydział Informatyki
Zachodniopomorski Uniwersytet Technologiczny w Szczecinie
anton.smolinski@zut.edu.pl

Samoadaptacyjna Optymalizacja Genetyczna

1. Algorytmy genetyczne

Algorytmy genetyczne są jedną z najczęściej stosowanych metod heurystycznych do rozwiązywania różnorakich problemów. Ich popularność wywodzi się przede wszystkim z uniwersalności zastosowań, oraz braku ograniczeń, które często występują w innych metodach. Stosowanie algorytmów genetycznych nie wymaga zgłębiania się w rozwiązywany problem, gdyż wystarczy znaleźć odpowiedź na pytanie jak porównać dwa rozwiązania i wybrać „lepsze". Dzięki swoim właściwościom algorytmy genetyczne doskonale sprawdzają się w rozwiązywaniu problemów rzeczywistych, oraz takich, w których przestrzeń rozwiązań jest na tyle duża, że inne metody stają się zbyt czasochłonne [5, 12].

Algorytmy genetyczne nazują na rozwiązaniu naśladującym pewne właściwości spotykane w naturze, takie jak dziedziczenie, czy wymiana genów, oraz dobór naturalny. Można wskazać, że działają one na zasadzie symulacji, w której tworzony jest zbiór pewnych rozwiązań problemu (zwany populacją). Pojedyncze rozwiązanie kodowane jest w tzw. chromosom zawierający informację na temat danego rozwiązania. Kodowanie ma szczególne znaczenie dla problemów rzeczywistych, gdyż prócz przedstawienia samego rozwiązania umożliwia także wprowadzenie pewnych ograniczeń na przestrzeń poszukiwań, dzięki czemu nie trzeba martwić się w kolejnych etapach algorytmu, czy generowane nowe rozwiązania należą do danej przestrzeni. Zagadnienie doboru odpowiedniego kodowania ma znaczący wpływ również na szybkość i dokładność działania algorytmów genetycznych i zostało opisane i przebadane w wielu pracach [14, 15].

W każdym kroku (iteracji) algorytmu genetycznego populacja zostaje przetworzona za pomocą kilku operatorów genetycznych takich jak selekcja, krzyżowanie i mutacja. Pozwalają one na wybranie oraz utworzenie nowych rozwiązań problemu, które w rezultacie dążą do lepszego dopasowania całej populacji. Przez wiele lat, odkąd opracowano ideę algorytmów genetycznych (klasyczną formę tych algorytmów przedstawił John Holland w 1970 r.), opracowywano wiele różnych operatorów, których zadaniem było usprawnienie przeszukiwania przestrzeni rozwiązań dla konkretnych problemów. Powstało wiele

prac oraz badań nad wpływem dobranych operatorów oraz ich parametrów na to, jak szybko i jak dokładnie algorytmy genetyczne wyszukują rozwiązanie quasi optymalne [1, 3, 4, 7, 8, 13].

Jedną z poważniejszych wad algorytmów genetycznych jest wspomniane wyszukanie rozwiązań quasi optymalnych. Oznacza to, że każdy przebieg (iteracja) algorytmu ulepsza pewne rozwiązania znajdujące się w populacji, dzięki czemu populacja jako ogół staje się coraz bardziej dopasowana do problemu, jednakże znalezienie rozwiązania optymalnego jest bardzo rzadkie. Wynika to z faktu w pewnej mierz losowego przeszukiwania przestrzeni rozwiązań i kłopotem z dokładnością reprezentacji rozwiązania. Aby rozwiązać ten problem często tworzy się hybrydy metod genetycznych i klasycznych, gdzie po znalezieniu pewnego rozwiązania quasi-optymalnego, metody klasyczne rozpoczynają dodatkowe dostrajanie celem polepszenia wyniku zwróconego przez algorytm genetyczny (oczywiście pod warunkiem, że znana jest metoda na dokładne rozwiązanie rozpatrywanego problemu). Nie ulega jednak wątpliwości, że to właśnie algorytmy genetyczne w rozsądnym czasie zwracają dobre wyniki, co w wielu przypadkach (szczególnie problemach rzeczywistych) jest wystarczające.

Drugim problemem algorytmów genetycznych jest złożoność obliczeniowa. Jako iż algorytmy te nie są zależne od rozpatrywanego problemu nie sposób określić ile iteracji należy wykonać aby osiągnąć akceptowalne rozwiązanie. Istnieje kilka metod zatrzymania algorytmu, z których najczęściej rozpowszechniona i najprostsza jest określenie z góry ilości iteracji algorytmu. Inne sposoby badają zbieżność populacji bądź najlepszego wyniku w populacji (np. jeżeli nie poprawił się przez zadaną liczbę iteracji, bądź kolejne wyniki są od siebie nie lepsze niż zadana dokładność). Jednak wraz z rozwojem wieloprocesorowych maszyn a także prostym dostępem do systemów rozproszonych problem złożoności przestaje być krytycznym. Struktura algorytmów genetycznych zakłada równoległe wykonywanie większości operacji, tak więc wykonywanie ich sekwencyjnie jest pewnym ograniczeniem ich możliwości [2, 5, 9, 15].

Uogólniając pewne właściwości algorytmów genetycznych powstała nowa klasa rozwiązywania problemów – obliczenia ewolucyjne. W odróżnieniu od swojego pierwowzoru obliczenia ewolucyjnie nie posiadają ściśle określonej struktury operatorów genetycznych ani reprezentacji. Bazują one w luźny sposób na operatorach genetycznych wprowadzając pewne urozmaicenia jak subpopulacje, zmienność operatorów bądź ich parametrów, czy inne zmiany. Natomiast jak w przypadku zwykłych algorytmów genetycznych celem tym zmian jest jedna (lub dwie) z poniższych zmian:

- Zwiększenie dokładności otrzymywanych obliczeń,
- Zwiększenie szybkości zwracania danych obliczeń.

2. Adaptacje w algorytmach genetycznych

Przy rozpatrywaniu algorytmów genetycznych oraz dobierania odpowiednich operatorów i parametrów do rozpatrywanego problemu ważne jest zrozumienie dwóch pojęć: naporu selekcyjnego, oraz różnorodności.

Napór selekcyjny można rozumieć jako szybkość z jaką populacja zbiega do pewnego rozwiązania quasi-optymalnego. Różnorodność określa natomiast różnice w pojedynczej populacji. Oczywistym faktem jest, że te dwa pojęcia posiadają przeciwstawne zwroty. Otóż silny napór selekcyjny powoduje, że populacja staje się jednorodna, natomiast duża różnorodność populacji spowalnia napór selekcyjny. Z jednej strony używając algorytmów genetycznych oczekuje się znalezienia dobrych wyników w krótkim czasie, jednak zbyt szybki napór selekcyjny i mała różnorodność pogarsza dokładność zwracanych rozwiązań, a może wręcz prowadzić do wpadnięcia do pułapki optimów lokalnych [9, 10, 11].

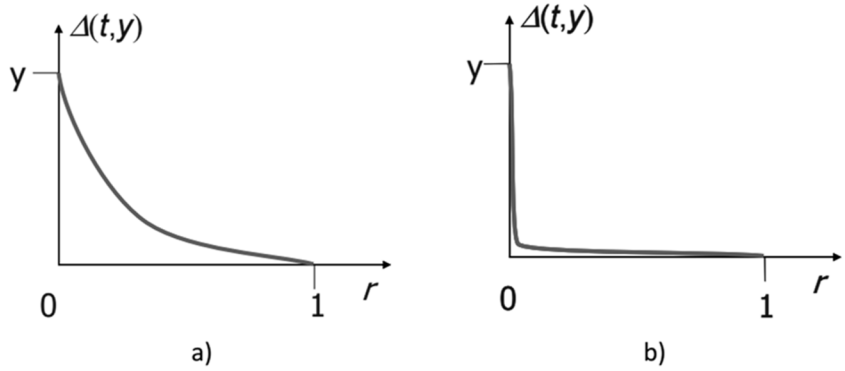
Wielkościami tych zjawisk steruje się za pomocą dobierania parametrów poszczególnych operatorów. Aby odpowiednio dobrać parametry często przeprowadza się kilka(-naście, -dziesiąt) przebiegów algorytmu z różnymi parametrami, gdyż nie odkryto jeszcze sposobu na dobór odpowiednich parametrów do konkretnego problemu. Często pojawia się więc paradoks, gdyż by wykonać optymalizację za pomocą algorytmów genetycznych należy najpierw zoptymalizować same algorytmy, a dokładnie – parametry poszczególnych operatorów.

Sposobem, który w dużej mierze rozwiązuje problem doboru odpowiednich parametrów jest stosowanie metod adaptacyjnych. Dzięki nim w trakcie swego działania programy ewolucyjne same dostrajają potrzebne im parametry. Przykładowo, różnorodność populacji jest ważniejszy na samym początku pracy algorytmów, natomiast im dokładniejsze wyniki są otrzymywane, zaczyna się w większym stopniu liczyć napór selekcyjny i dostrajanie lokalne [1, 6, 9, 10].

Aby osiągnąć adaptację stosuje się kilka różnych metod. Najpopularniejszą jest wprowadzanie zmiennych parametrów, np. mutacji zależnych od numeru iteracji algorytmu. Najpopularniejszą z takich metod jest mutacja nierównomierna, która zakłada, że w początkowych fazach algorytmu mutacji przede wszystkim powinny podlegać najbardziej znaczące bity w bitowej reprezentacji chromosomu, dzięki czemu zapewnia się dużą zmienność w rozwiązaniach, a następnie w trakcie jego działania zmieniać mniej znaczące bity – dzięki czemu kolejne rozwiązania nie różnią się zbytnio od siebie. Przykład zależności zmian można prześledzić na rysunku 1. Wykres a) zawiera zależność funkcji $\Delta(t,y)$ – czyli wartości zmiany chromosomu w wyniku mutacji, od liczby losowej (r) i maksymalnego przedziału zmian (y) [1, 9].

Innym sposobem na adaptacje algorytmów ewolucyjnych (adaptacja powoduje, że algorytmy należą już do wyższej klasy – ewolucyjnych a nie tylko genetycznych), jest stosowanie zmiennych wielkości populacji. Uzależnienie

wielkości populacji od kolejnych iteracji ma podobny wynik jak uzależnienie zmian w operatorze mutacji. Przy dużych populacjach naturalnym jest fakt, że osobniki są zróżnicowane (choćby za sprawą wielkości danej populacji). Jednocześnie powoduje to zwiększenie ilości wykonywanych operacji, gdyż każdy z osobników musi być osobno oceniony, a następnie należy utworzyć ponowną, dużą populację. Małe populacje wymagają znacznie mniej operacji, lecz wadą w tym przypadku jest niska różnorodność osobników (nawet, jeżeli mocno się od siebie różnią, reprezentują niewielką grupę przestrzeni rozwiązań) [9, 10].



Rysunek 1. Zależność zmian wprowadzanych przez mutację nierównomierną w a) początkowej fazie algorytmu ewolucyjnego, b) końcowej fazie algorytmu ewolucyjnego

Niektórzy badacze wykorzystując także metody hybrydowe do tworzenia algorytmów ewolucyjnych nazywając je adaptacją. Najpopularniejsze są w tym przypadku hybrydy pomiędzy algorytmami genetycznymi a metodami klasycznymi (dla danego problemu), które mają za zadanie zwiększyć dokładność rozwiązania wskazanego przez AG. Innym sposobem jest stosowanie metod np. logiki rozmytej jako metaalgorytm doboru parametrów AG [6].

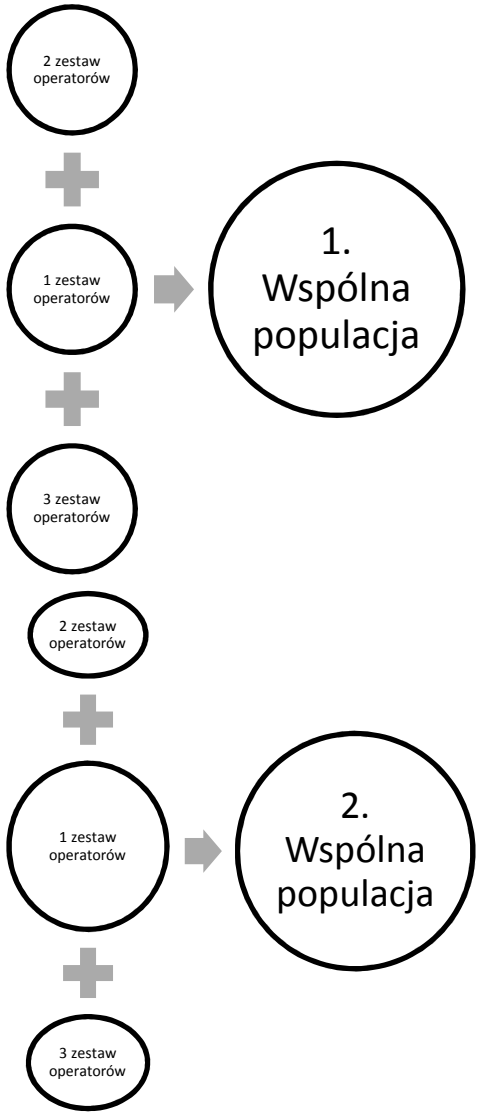
Przy rozpatrywaniu znaczenia adaptacji warto również wspomnieć tzw. No Free Lunch Theorem opracowaną przez Wolpert, Macready [9]. Teoria ta opisuje matematyczne podstawy znanego powszechnie faktu, że każda metoda specjalizacyjna będzie lepsza od metody uniwersalnej w zadanym problemie. Dodatkowo można to rozumieć, że odpowiednio dobrane parametry algorytmu genetycznego tworzą rozwiązanie bardziej wyspecjalizowane, od algorytmu w którym te parametry nie są dobierane – mimo że obydwa powinny po pewnym czasie dać dobre wyniki. Jednak problemy z metodami specjalizowanymi wynikają przy próbie ich zastosowania do innych problemów, niż te dla których zostały dostrojone. W tym przypadku radzą sobie znacznie gorzej od metod uniwersalnych. Oczywiście szczególnym przypadkiem są algorytmy genetyczne, które z założenia są metodami uniwersalnymi, jednak zmieniając parametry można je dostosować. Zmiana

parametrów będzie miała wpływ na szybkość osiąganych wyników, jednak nie tak wielką, jak zmiana całej metody wyszukiwania odpowiedniego rozwiązania. Adaptacje wprowadzane do metod genetycznych dodatkowo pozwalają na osiągnięcie „lepszych" wyników gdy nie znamy dokładnego rozwiązania dla danego problemu, czy klasy problemów.

3. Idea metody SOG

Opracowane przez autora podejście Samoadaptacyjnej Optymalizacji Genetycznej bazuje na kilku założeniach i modyfikacjach przetestowanych przez innych badaczy. Metoda ta ma na celu uzyskania efektu synergii w wyniku połączenia różnych modyfikacji algorytmów genetycznych, a także usprawnienie adaptacji celem samoistnego dostrajania się algorytmu do istoty rozwiązywanego problemu. W tym celu proponuje się wprowadzenie wielopoziomowej adaptacji: Metod, Operatorów oraz Parametrów.

W odróżnieniu do powszechnie stosowanych metod adaptacji, metoda SOG przenosi adaptację na poziom metaalgorytmu. Oznacza to, że poszczególne metody rozwiązywania zadań, bądź operatory także stanowią przedmiot adaptacji i mogą być wzmacniane, bądź wygłuszane w zależności od rozpatrywanego zadania. Aby tego dokonać metoda zakłada wprowadzenia obliczeń ewolucyjnych nie tylko na poziomie rozwiązań, lecz na poziomie poszczególnych algorytmów. Schemat danej adaptacji został przedstawiony na rysunku 2. W ramach danej metody uruchamiane są różne metody oraz operatory genetyczne pracujące na osobnych subpopulacjach. W kroku selekcji oceniane są nie tylko pojedyncze rozwiązania, lecz całe populacje zwracane przez zestawy operatorów. Populacje o najlepszym dopasowaniu powodują że utworzone je operatory otrzymują priorytet w kolejnych obliczeniach, utożsamiany z wielkością subpopulacji którą przetwarza, natomiast pozostałe zestawy operatorów otrzymują zubożenie swojej populacji. W ten sposób schemat doboru naturalnego przekłada się nie tylko na samych rozwiązaniach, lecz także na sposób wyszukiwania nowych rozwiązań. Biorąc pod uwagę, że różne optymalne dla danego problemu zestawy operatorów i parametrów mogą być różne, po każdej selekcji subpopulacje są łączone, a następnie znów dzielone w losowy sposób.



Rysunek 2. Schemat działania metaalgorytmu adaptacyjnego SOG

Prócz wprowadzenia konkurowania pomiędzy metodami, operatorami oraz ich parametrami, metoda SOG wprowadza także najpopularniejsze usprawnienia algorytmów genetycznych, które przyczyniają się do zwiększenia ich dokładności a także szybkości obliczeń. W tym celu korzysta się z dobrze przebadanych poprawek do klasycznego algorytmu genetycznego i spuścizny badaczy, aby

uzyskać nową wartość dodaną wynikającą z ich połączenia. Proponowane modyfikacje można zauważyć na rysunku 3. Na rysunku tym widnieją dwa kolory przedstawiające modyfikacje wspierające szybkość działania programów ewolucyjnych, a także modyfikacje wspierające dokładność znajdowanych rozwiązań.

Modyfikacje wpływające na dokładność znajdowanych rozwiązań, a więc Kodowanie zmiennopozycyjne, Mutacja nierównomierna, oraz adaptacyjna elitarność, to jedne z częściej stosowanych modyfikacji w algorytmach genetycznych. Kodowanie zmiennopozycyjne pozwala na zwiększenie przestrzeni poszukiwań, spowodowane zmianą zapisu poszczególnych liczb. Dodatkowo zmiana ta wprowadza pewną właściwość liczb rzeczywistych, której nie sposób było osiągnąć za pomocą kodowania binarnego – otóż podobne rozwiązania, mają podobną reprezentację, czyli niewiele się od siebie różnią. W przy kodowaniu binarnym, aby osiągnąć dany efekt należało stosować zapis binarny w Kodzie Graya.

Rysunek 3. Modyfikacje AG proponowane w metodzie SOG

Mutacja nierównomierna, wspominana już w niniejszym artykule wprowadza kolejny poziom adaptacyjności na poziomie poszczególnych parametrów, zmiennych w trakcie przeprowadzania obliczeń. Podobnie adaptacyjna elitarność oznacza wprowadzenie elitarności zależnej od ilości iteracji wykonanych przez algorytm. Elitarność jest jednym z najpowszechniej stosowanych środków do zwiększenia naporu selekcyjnego. Zakłada ona porównanie osobników potomnych z osobnikami rodzicielskimi i wybranie „lepszych", bądź w niektórych przypadkach jawne przeniesienie najlepszych osobników z populacji rodzicielskiej do populacji potomnej bez wykonywania na nich innych operacji genetycznych [5, 9, 12, 13].

Przyspieszenie obliczeń w metodzie SOG realizowane jest głównie za pomocą ich zrównoleglania. W tym przypadku bezwzględny czas obliczeń będzie dłuższy, niż obliczenia sekwencyjne (gdyż należy doliczyć czas synchronizacji), jednakowo czas względny, który będzie odczuwalny przez użytkownika jest znacznie krótszy z powodu asynchronicznego przeprowadzania obliczeń i wykorzystywania większej ilości rdzeni komputera.

Dodatkowym usprawnieniem prędkości działania jest propozycja wysunięta przez Goldberga, polegająca na przetwarzaniu tzw. Nieporządnych algorytmów genetycznych (messy genetic algorithms). Reprezentacja taka umożliwia przetwarzanie rozwiązań bez kompletu genów określających dane rozwiązanie. W rezultacie chromosomy mogą posiadać różną długość, a co za tym idzie zużycie pamięci do przechowywania populacji będzie mniejsze. W przypadku tej reprezentacji istotnym czynnikiem jest określenie jak interpretować brakujące geny chromosomu. Nadmiarowość genów jest natomiast redukowana poprzez wyciągnięcie średniej z danych wartości. Najczęściej stosowanymi operatorami przy tej reprezentacji są operatory przecinania i sklejania [5, 9, 11].

Zdaniem autora metody modyfikacje te powinny zapewnić doskonały kompromis pomiędzy uniwersalnością a specjalnością metody do rozwiązywania problemów, szczególnie gdy rozwiązywane są problemy rzeczywiste, gdyż właśnie one stanowią główny cel stosowania opracowanej metody. Podstawy teoretyczne poszczególnych modyfikacji, różne badania, oraz powszechność stosowania pozwalają mieć nadzieję graniczącą z pewnością że metoda ta przyniesie oczekiwane po niej wyniki.

Niestety na chwilę obecną metoda ta dopiero została opracowana koncepcyjnie na podstawie dogłębnych badań literaturowych i badań nad poszczególnymi elementami algorytmów genetycznych. Przetestowanie działania metody i rzeczywiste wyniki, jakie można za jej pomocą osiągnąć zostaną przedstawione w przyszłych artykułach autora na podstawie już prowadzonych badań w danym zakresie, które nie są jeszcze gotowe do publikacji.

W dalszej części niniejszego artykułu przedstawione zostały badania poszczególnych modyfikacji algorytmów genetycznych zastosowanych przy

metodzie SOG. Badania te miały na celu określenie jak dane modyfikacje wpływają na czas oraz szybkość osiąganych wyników.

4. Badania modyfikacji algorytmów genetycznych

Przeprowadzone badania nad poszczególnymi algorytmami genetycznymi wykonano w środowisku .NET. Głównym powodem wyboru tego środowiska była możliwość opracowania algorytmów w paradygmacie obiektowym, oraz uproszczone procedury zrównoleglania obliczeń.

Aplikacja pozwalająca na badanie poszczególnych algorytmów genetycznych składa się z szeregu współdziałających klas, w których najważniejszą rolę odgrywa dziedziczenie. Taki sposób zapisu algorytmu gwarantuje ich kompatybilność – każda badana wersja zawierająca różne modyfikacje posiada taką samą strukturę, interfejs oraz operuje na tych samych danych. Rozszerzając klasę bazową, którą jest klasyczny algorytm genetyczny poszczególne badane metody przeciążały jeden, bądź dwa operatory genetyczne. Dzięki temu zabiegowi badane metody różniły się tylko i wyłącznie modyfikacją, która była przedmiotem tych badań. Dodatkowo obiekt chromosomu pozwalał na konwersję przechowywanych wartości poszczególnych genów zarówno na postać bitową (wymaganą przez pewne operatory), jak i postać zmiennopozycyjną (badaną w innych operatorach).

By dokładniej poznać różnice pomiędzy różnymi modyfikacjami wygenerowano również pojedynczą populację początkową, która została przekazana do wszystkich metod, tak więc wyeliminowano kolejny czynnik wpływający na osiąganie wyniki. Wszystkie modyfikacje zostały również przetestowane na tym samym zestawie parametrów, by również czynnik różnych parametrów nie zakłócał wyników tych badań. W rezultacie badaniu zostały poddane jedynie modyfikacje w strukturze algorytmu, a wszystkie pozostałe elementy zostały ujednolicone dla badanych algorytmów.

Badanie zostało przeprowadzone na 5 funkcjach (1 lub 2 parametrycznych), które reprezentują różne klasy problemów i były używane przez innych badaczy do ukazania zróżnicowania działania algorytmów [9]:

- $x\sin(10\pi x)+1$,
- $21.5+x\sin(4\pi x)+y\sin(20\pi y)$,
- $\frac{8x}{8}$,
- $x\,sgn(x)$,
- $0.5+\frac{\sin^2\sqrt{x^2+y^2}-0.5}{1+0.001(x^2+y^2)^2}$.

Na wskazanych funkcjach przetestowano 5 podejść, algorytmów:

- Klasyczny algorytm genetyczny
- Algorytm genetyczny z reprezentacją zmiennopozycyjną
- Algorytm „nieporządny”
- Elitarność w algorytmach genetycznych
- Mutację nierównomierną

W każdym z badanych podejść wykorzystano selekcję proporcjonalną opartą na kole ruletki i stałą populację. Klasyczny algorytm genetyczny bazując na reprezentacji binarnej korzystał z operatora jednopunktowego krzyżowania, oraz mutacji dla każdego bitu z zadanym prawdopodobieństwem. Przy reprezentacji zmiennopozycyjnej operator krzyżowania tworzył potomków odpowiednio w 1/3 i 2/3 odległościach pomiędzy osobnikami rodzicielskimi, natomiast operator mutacji zmieniał wartość genu losowo +/- o wartość z przedziału od 0 do zadanego prawdopodobieństwa mutacji procentowej wartości danego genu. Nieporządne algorytmy wprowadziły zmianę jedynie w operatorze krzyżowania, który losowo przyjmował wartość albo pierwszego, albo drugiego rodzica, albo średnią z ich wartości i w ten sposób generował dwoje potomków. Elitarność zarówno w operatorze krzyżowania jak i mutacji sprawdzała czy osobnik rodzicielski jest lepszy od osobnika nowo utworzonego, przy czym jako wynik operatory zwracały lepszy zbiór osobników. Natomiast ostatnia modyfikacja, mutacji nierównomiernej wprowadziła dodatkową funkcję, która na podstawie aktualnego numeru populacji wyliczała wartość o jaką zmieni się wartość mutującego genu. Funkcja ta ma postać:

$$x \cdot 1 - r^{1-\frac{itt}{60}},$$

gdzie *r* jest losową wartością przyjmującą wartość 0 lub 1, *itt* jest numerem aktualnej iteracji, a *x* maksymalną zmianą w genie, która w zależności od liczby losowej z równym prawdopodobieństwem wynosi:

$$\begin{cases} max - x \\ x - min \end{cases},$$

gdzie *max* i *min* to odpowiednio górne i dolne ograniczenia przestrzeni rozwiązań.

Każdą z metod uruchomiono 10 krotnie (z tymi samymi parametrami i populacją początkową), by pomimo losowości móc ustalić średnią wartość poszczególnych parametrów. Badaniu podlegały dwie wartości – dopasowanie najlepszego wyniku uzyskanego daną metodą, a także liczbę iteracji potrzebnych do uzyskania najlepszego wyniku. Warunkiem zakończenia poszczególnych algorytmów było osiągnięcie zadanej liczby iteracji bez poprawy rozwiązania.

Aplikacja do testowania równolegle uruchamiała 10 instancji pojedynczej metody w osobnych wątkach. Po wykonaniu wszystkich przebiegów dla jednej z metod uruchamiało kolejne 10 instancji dla kolejnej metody, aż zostały przebadane wszystkie badane modyfikacje.

5. Wyniki

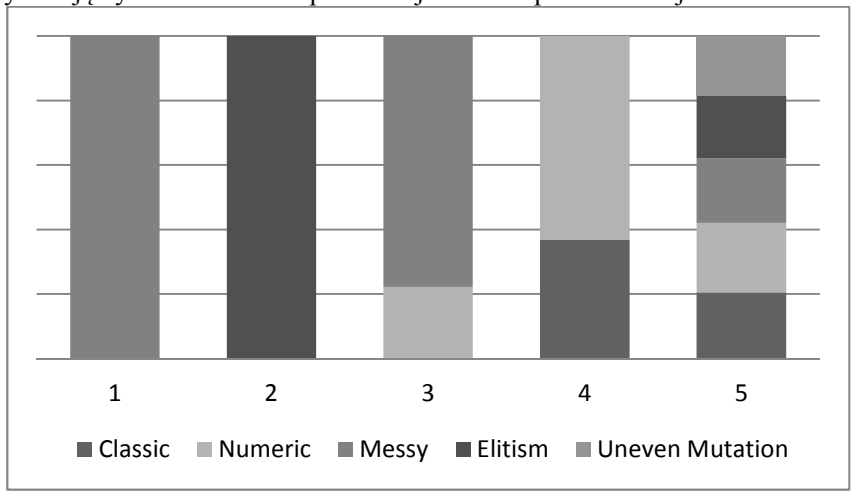
Wyniki dla parametrów:

- Wielkość populacji: 75,
- Iteracji bez polepszenia rozwiązania: 25 (warunek stopu),
- Parametr krzyżowania: 0.65,
- Parametr mutacji: 0.015,

można prześledzić na rysunkach 4 i 5.

Rysunek 4 przedstawia dokładność poszczególnych metod (zaznaczonych różnymi kolorami) dla poszczególnych funkcji (numery funkcji na osi odciętych). Im większy udział metody, tym lepsze otrzymywane były średnie wyniki w porównaniu z innymi metodami. Jak można łatwo zauważyć nie można znaleźć jednoznacznego zwycięzcy. Każda badana modyfikacja poprawiała wyniki w innej klasie problemu. Wyjątkiem jest tutaj Mutacja nierównomierna, która jedynie w ostatniej funkcji osiągnęła wyniki zbliżone do innych funkcji.

Na rysunku 5 można natomiast prześledzić liczbę iteracji, która była potrzebna każdej z metod do osiągnięcia najlepszej dla niej wartości rozwiązania problemu. Interpretując wykres, im mniejszy udział odpowiedniej metody (koloru) w słupku wskazującym na konkretny problem, tym szybciej uzyskała ona wynik. Zgodnie z oczekiwaniami, najbardziej ogólne, uniwersalne, podejście, czyli klasyczny algorytm genetyczny na rozwiązanie każdego z zadań potrzebował mniej więcej podobnej liczby iteracji. Pozostałe metody w zależności od problemu potrzebowały więcej, bądź mniej przebiegów. Najbardziej wrażliwą na różne klasy problemów modyfikacją było stosowanie reprezentacji zmiennoprzecinkowej.

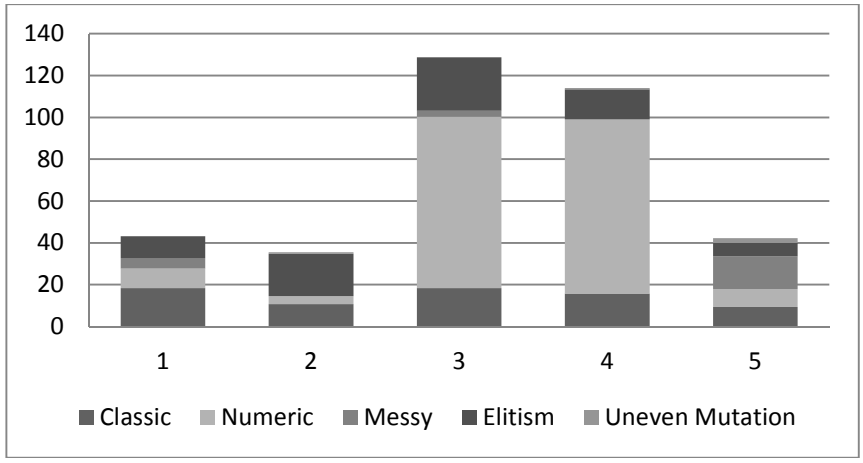


Rysunek 4. Dokładność rozwiązań zwracanych przez metodę

Porównując obydwa wykresy, można wnioskować, że najlepsze wyniki dało zastosowanie nieporządnych algorytmów genetycznych (messy) gdyż zarówno w dokładności zwracanych rozwiązań, jak i szybkości działania ma zadowalające wyniki ogólne. Jednakże dla każdego z rodzajów problemu inne modyfikacje wykazały się lepszymi osiągnięciami.

Metoda klasyczna zachowywała się stabilnie niezależnie od problemu, co jednoznacznie wskazuje na jej uniwersalność i niezależność od rozpatrywanych problemów. Reprezentacja numeryczna w przypadku funkcji 4 (zgodnie z kolejnością podaną w rozdziale 4) przy każdym przebiegu poprawiała swój rezultat znacząco wyprzedzając inne metody. Można zauważyć prawidłowość, że metoda ta działa albo długo i daje bardzo dobre wyniki, lub gdy nie jest w stanie ich poprawić szybko kończy swoją pracę. Nieporządne algorytmy genetyczne, oprogramowane jako nietypowy operator krzyżowania (nie badano właściwości zmiennej długości chromosomu na pamięć potrzebną do wykonania programu), okazała się jedną z lepszych modyfikacji. Wprowadzenie elitarności do algorytmów genetycznych wydłużyło ich czas działania w stosunku do klasycznego podejścia, jednak dla pewnej klasy problemów dało zaskakująco dobre rezultaty. Natomiast największym rozczarowaniem okazało się zastosowanie mutacji nierównomiernej, która jedynie dla ostatniej funkcji przy porównywalnych wynikach do pozostałych metod zakończyła swoje działanie w rekordowo krótkim czasie.

Warto jednak zauważyć, że z powodu dużej losowości występującej przy każdym przebiegu iteracji, a także innej populacji początkowej przy każdym wykonywanym teście (również generowanej losowo) wyniki mogą być różne nawet przy zastosowaniu tych samych parametrów. Przedstawione w niniejszej pracy rezultaty są uśrednieniem trendu widocznego przy dłuższym badaniu tych parametrów i mogą nie być weryfikowalne przy zaledwie kilku powtórzeniach danego badania.

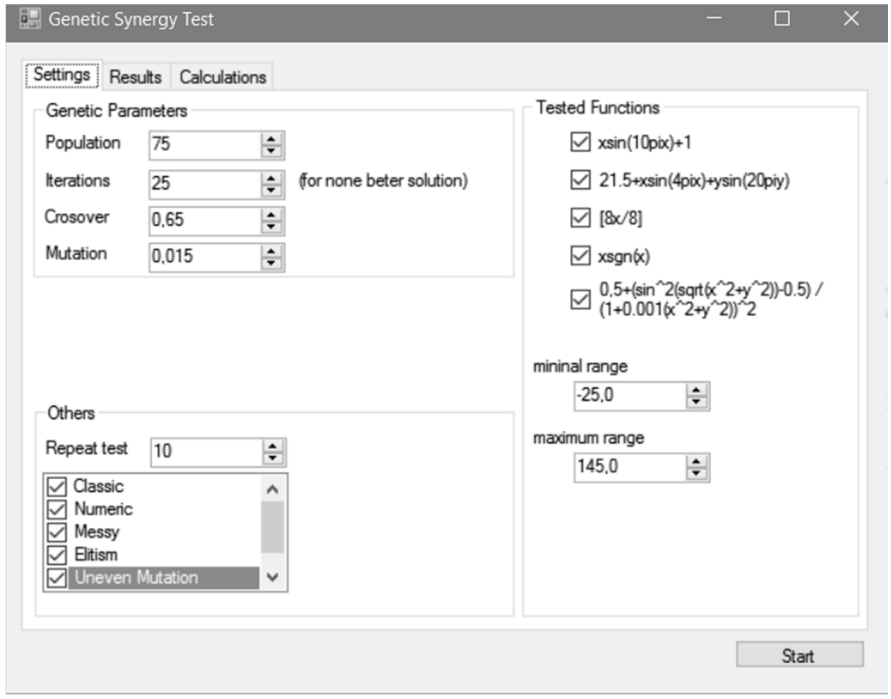


Rysunek 5. Ilość iteracji potrzebna poszczególnym metodom do osiągnięcia najlepszego wyniku.

6. Podsumowanie

Opracowana metoda samoadaptacyjnego algorytmu genetycznego ma na celu wytworzenie uniwersalnego algorytmu, który niezależnie od klasy problemu, a przede wszystkim z myślą o problemach rzeczywistych, będzie zwracać wyniki lepsze zarówno pod względem dokładności, jak i szybkości od klasycznego podejścia algorytmów genetycznych. W tym celu autor powołuje się na przebadane i dobrze udokumentowane modyfikacje, których porównanie w opisanych badaniach wskazało, że każda z nich jest lepsza w pewnej klasie problemów od podejścia klasycznego. Zastosowanie tych modyfikacji umożliwi wykorzystanie mocnych stron każdej z nich, co powinno odbić się korzystnie na wynikach otrzymywanych przez taki algorytm.

W ramach badań wytworzono również aplikację, za pomocą której można badać wpływ poszczególnych operatorów oraz podejść na pracę algorytmu genetycznego, eliminując (bądź minimalizując) wpływ losowości dzięki zapewnieniu jednakowych warunków testowania dla wszystkich badanych przypadków. Rozwój danej aplikacji (przedstawionej na rysunku 6) umożliwi badanie parami, oraz grupami poszczególnych modyfikacji celem maksymalizacji efektu synergii ich stosowania.



Rysunek 6. Aplikacja umożliwiająca testowanie poszczególnych modyfikacji algorytmów genetycznych

Dalsze badania autora skupiają się na określeniu siły wpływu poszczególnych modyfikacji na osiągane wyniki programu ewolucyjnego, a także na metodach związanych z oceną populacji zwracaną przez poszczególne metody (aby umożliwić rywalizację metod i ocenę ich działania w czasie rzeczywistym).

Bibliografia

1. Beluch W.: *Wpływ operatorów mutacji na skuteczność poszukiwań AE*, Obliczenia ewolucyjne, laboratorium 4, Wydział Mechaniczny Technologiczny, Politechnika Śląska,
   http://www.imio.polsl.pl/Dopobrania/Lab%20OE%2004%20(mutacja).pdf [dostęp: 29.05.2016]
2. Deb K.: *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley & Sons Ltd, UK, 2001
3. Fogel D. B., Atmar J. W.: *Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Process Using Linear Systems*, Springer-Verlag, Biological Cybernetics vol. 63, 1990

4. Freisleben B, Merz P.: *New Genetic Local Search Operators for the Traveling Salesman Problem*, Springer Berlin Heidelberg, Applications Of Evolutionary Computation Evolutionary Computation In Computer Science And Operations Research, 2005
5. Goldberg D. E.: *Algorytmy genetyczne i ich zastosowania*, WNT, Warszawa 1998
6. Grygierek K.: *Samoadaptacyjna metoda algorytmów genetycznych w optymalizacji przestrzennych kratownic*, Modelowanie Inżynierskie nr52, Gliwice 2014
7. Gwiazda T. D.: *Algorytmy genetyczne kompendium tom1*, Wydawnictwo Naukowe PWN, Warszawa 2007
8. Gwiazda T. D.: *Algorytmy genetyczne kompendium tom2,* Wydawnictwo Naukowe PWN, Warszawa 2007
9. Michalewicz Z.: *Algorytmy genetyczne+struktury danych=programy ewolucyjne,* Wydawnictwa Naukowo-Techniczne, 1996
10. Paszyńska A.: *Projektowanie wspomagane komputerowo a problemy zbieżności algorytmów genetycznych*, rozprawa doktorska, Instytut Podstawowych Problemów Techniki PAN, 2007
11. Przewoźniczek M.: *Nowy szablon z kodowaniem nieporządnym jako remedium na typowe wady algorytmu genetycznego,* rozprawa doktorska, Wydział Informatyki i Zarządzania, Politechnika Wrocławska, Wrocław 2012
12. Słowik A.: *Właściwości i zastosowania algorytmów ewolucyjnych w optymalizacji*, Metody informatyki stosowanej, Koszalin 2007
13. Studniarski M.: *Wykłady z algorytmów genetycznych Część 1: Podstawowe informacje o algorytmach genetycznych*, Wydział Matematyki i Informatyki Uniwersytetu Łódzkiego, http://math.uni.lodz.pl/~marstud/AGwyklad1-2.pdf [dostęp: 26.12.2015]
14. Turing A. M.: *Computing Machinery and Intelligence. „Mind"*, Tom 59, nr 236, X/1950
15. Zitzler E.: *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, rozprawa doktorska, Swiss Federal Institute of Technology Zurich, 1999

Streszczenie

W artykule przedstawiono nowe podejście do adaptacyjnych Algorytmów genetycznych. Koncepcja samoadaptacyjnej optymalizacji genetycznej opiera się na wprowadzeniu meta-algorytmu, w ramach którego poszczególne algorytmy genetyczne (z różnymi operatorami oraz parametrami) rywalizują między sobą. Artykuł zawiera wstępne badania, ukazujące działanie różnych modyfikacji

algorytmów genetycznych na wybranych problemach. Przeprowadzone eksperymenty wskazują, że użycie strategii samoadaptacji w proponowanym zakresie może przynieść obiecujące rezultaty.

Opisywane w niniejszym dokumencie prace ukazują porównanie modyfikacji takich jak: reprezentacja numeryczna chromosomów, nieporządne algorytmy genetyczne, mutacja nierównomierna czy elitarność. Wyniki różnych podejść zostały również porównane do klasycznego podejścia (reprezentacja binarna, jednopunktowe krzyżowanie).

Słowa kluczowe: Algorytmy genetyczne, Adaptacja genetyczna, reprezentacja numeryczna chromosomów, nieporządne algorytmy genetyczne, mutacja nierównomierna, elitarność

Abstract

This paper presents a new way of adaptive in genetic algorithms. Concept of self-adaptive genetic optimization was based on meta-algorithm, where different operators with different parameters competitive with each other. The paper contains preliminary research, showing how the various genetic algorithms modification react with different problems. Conducted experiments suggest that developed self-adaptive strategy for real problem optimization using genetic algorithms may return promising results.

Described research compare genetic modification as: chromosome numeric representation, messy genetic algorithms, uneven mutation and elitism. The results of different approach have been also compared to result of classic genetic algorithm (with binary representation, one-point crossing).

Keywords: Genetic algorithms, genetic adaptive, real problem optimization, numeric representation, messy genetic algorithms, uneven mutation, elitism.