Uniwersytet Ekonomiczny w Katowicach

Studia Ekonomiczne. Zeszyty Naukowe
Uniwersytetu Ekonomicznego w Katowicach
ISSN 2083-8611 Nr 235 · 2015

Michał Kameduła

Uniwersytet Łódzki
Wydział Ekonomiczno-Socjologiczny
Katedra Badań Operacyjnych
michal_kamedula@poczta.o2.pl

Jan Gajda

Uniwersytet Łódzki
Wydział Ekonomiczno-Socjologiczny
Katedra Badań Operacyjnych
jan.b.gajda@wp.pl

WPŁYW PARAMETRÓW STARTOWYCH NA TEMPO ZBIEŻNOŚCI KOEWOLUCYJNEGO ALGORYTMU GENETYCZNEGO

Streszczenie: W pracy tej zaprezentowano procedurę pozwalającą zbadać wpływ parametrów startowych na tempo zbieżności algorytmu genetycznego. Jej zaletą jest fakt, że bierze ona pod uwagę nie tylko samą szybkość znalezienia rozwiązań bliskich optymalnym, ale również stabilność wyników. Przedstawioną metodę wykorzystano następnie do wyboru najlepszej wartości parametrów pewnego koewolucyjnego algorytmu analizy portfelowej. Wykazano przy tym, że dla zadania dwukryterialnego daje on lepsze wyniki, niż niezależne przebiegi zwykłego algorytmu genetycznego przetwarzającego jedną populację na raz. Jednocześnie jednak wymiana informacji pomiędzy niszami podlegającymi koewolucji powinna zostać przerwana, gdy znajdziemy już zestaw rozwiązań bliskich optymalnym. Wykazano też znaczny wpływ parametrów mutacji na zbieżność algorytmu.

Słowa kluczowe: algorytmy genetyczne, koewolucja, zbieżność rozwiązań.

Wprowadzenie

Celem tej pracy było zbadanie, w jaki sposób dobór parametrów startowych algorytmu genetycznego wpływa na szybkość znalezienia rozwiązań suboptymalnych. Testy przeprowadzono na przykładzie algorytmu analizy portfelowej wykorzystującego dane o kursach akcji z indeksu WIG20.

Ustalenie optymalnego składu portfela inwestycyjnego jest w istocie zagadnieniem dwukryterialnym. Naszym celem jest zapewnienie jak najwyższego oczekiwanego zysku przy jednoczesnej minimalizacji ponoszonego przez inwestora ryzyka. Dokładne zmierzenie preferencji i przypisanie im precyzyjnych liczbowych wartości jest jednak zadaniem bardzo skomplikowanym. Dlatego też

w praktyce wyznacza się zwykle pewien zestaw portfeli o zróżnicowanych parametrach, spośród których decydent może wybrać ten, który najlepiej odpowiada jego oczekiwaniom [Laumans i in., 2001, s. 4-5].

Aby otrzymać taki zestaw portfeli z użyciem algorytmu genetycznego posłużono się niszami ewolucyjnymi utworzonymi dzięki mechanizmowi koewolucji. Polega on na równoczesnym prowadzeniu obliczeń dla grupy oddzielonych od siebie podpopulacji, pomiędzy którymi możliwa jest jednak wymiana pewnych informacji genetycznych. Przetwarzany zbiór rozwiązań został więc podzielony na mniejsze części, w których poszukiwano portfeli o nieco innym stosunku zysku do ryzyka. Pomiędzy tymi podzbiorami zachodziła jednak wymiana informacji. Pierwszym zagadnieniem, które zbadano, był więc wpływ parametrów tej wymiany na czas znalezienia rozwiązań.

Zakładano, że koewolucja pozwala dojść w pobliże rozwiązań optymalnych szybciej, niż w przypadku niezależnych przebiegów algorytmu o rozdzielonych populacjach. Porównano więc rezultaty algorytmu koewolucyjnego z wynikami niezależnych przebiegów algorytmu genetycznego dla oddzielonych od siebie nisz. Sprawdzono też, czy wymiana informacji między podpopulacjami jest korzystna przez cały czas trwania obliczeń, czy też w pewnym momencie lepiej jest ją przerwać. Postawiono hipotezę, że w początkowej fazie działania algorytmu wpływ wymiany informacji jest korzystny. Później jednak, po znalezieniu rozwiązań bliskich optymalnym, może ona spowalniać dalsze postępy obliczeń. Zbadano też, jak podobieństwo sąsiadujących ze sobą nisz wpływa na efektywność tej wymiany. Przypuszczano, że im bardziej podobne warunki panują w sąsiednich niszach, tym lepsze efekty możemy uzyskać, przenosząc fragmenty rozwiązań pomiędzy nimi.

Drugim interesującym nas elementem algorytmu był mechanizm mutacji. Wprowadza on niewielkie losowe zaburzenia w kodzie nowo tworzonych rozwiązań. Postanowiono sprawdzić, jak zmiany jego parametrów wpływają na tempo zbieżności generowanych wyników. Zwykle przyjmuje się, że mechanizm ten ma stosunkowo niewielkie znaczenie dla sprawnego działania algorytmu. Zaleca się więc, żeby prawdopodobieństwo mutacji było stosunkowo małe [Goldberg, 2003, s. 31]. Przed przeprowadzeniem testów zakładano, że najlepsze wyniki otrzymamy dla stosunkowo niskiego bazowego prawdopodobieństwa mutacji. Proponowano też, aby w mutowanych rozwiązaniach przeciętnie zmianie ulegały tylko wartości pojedynczych genów. W ten sposób zminimalizowano by szansę, że korzystna mutacja, kiedy już się pojawi, zostanie zniwelowana niepożądanymi zmianami innych fragmentów kodu.

Aby sprawdzić, w jaki sposób poszczególne parametry wpływają na działanie algorytmu, wygodnie jest wykorzystać obliczenia na kracie. Ze względu na ograniczenia czasowe zdecydowano się jednak nie testować wszystkich możliwych zestawów parametrów, a jedynie wybrane ich warianty. Pomysł wykorzystania obliczeń na kracie do wyboru najlepszych parametrów algorytmu genetycznego nie jest nowy [Angelova i Pencheva, 2001]. Na potrzeby tej pracy przygotowano jednak procedurę testową, którą można w łatwy sposób wykorzystać dla dowolnego nowo tworzonego algorytmu. Jedynym warunkiem jest, aby raport wyników zawierał historie postępów funkcji przystosowania.

1. Wykorzystany algorytm genetyczny

Algorytmy genetyczne są klasą metod metaheurystycznych wzorowanych na zachodzących w przyrodzie procesach dziedziczenia i kształtowania gatunków. Wyróżniają się one tym, że operują nie na pojedynczych rozwiązaniach, ale na pewnym ich zbiorze nazywanym populacją. Rozwiązania te, nazywane osobnikami, nie są jednak zapisywane w sposób jawny. Przechowuje się je w odpowiednio zakodowanej postaci, zależnej od specyfiki rozwiązywanego zadania. W kolejnych iteracjach algorytm łączy ze sobą fragmenty kodów poszczególnych rozwiązań. Proces ten nazywany jest krzyżowaniem. Z populacji wybiera się losowo dwa lub więcej osobniki nazywane rodzicami i na podstawie ich kodów tworzy się rozwiązania potomne. Krzyżowanie jest w znacznej mierze operacją losową. Preferuje ono jednak osobniki o najlepszych wartościach przystosowania, będącego odpowiednikiem funkcji celu w klasycznych metodach optymalizacji. Dzięki temu w kolejnych iteracjach jakość rozwiązań stopniowo ulega poprawie. Nowo utworzone rozwiązania mogą też ulec niewielkim losowym zaburzeniom, czyli mutacji. Zwykle ma ona utrzymać odpowiednią różnorodność populacji oraz przeciwdziałać blokowaniu się algorytmu w lokalnych ekstremach funkcji celu. Obszerny opis budowy i zastosowań algorytmów genetycznych można znaleźć m.in. w pracach Goldberga [2003] i Arabasa [2001].

Algorytmy genetyczne mogą z początku wydawać się zbyt losowe, jednak w praktyce okazują się bardzo skuteczne w szerokiej gamie zadań optymalizacji [Goldberg, 2003, s. 141-144]. W szczególności wskazać zaś można wiele zastosowań w interesującej nas tu dziedzinie finansów [Gwiazda, 1998].

W pracy wykorzystano algorytm koewolucyjny, czyli taki, w którym równocześnie przetwarzaniu podlega zestaw autonomicznych podpopulacji [Arabas, 2001, s. 236-239]. Nie są one jednak w pełni od siebie oddzielone. Możliwa jest

ograniczona wymiana informacji pomiędzy nimi. Mechanizm ten używany jest do tworzenia nisz ewolucyjnych [Goldberg, 2003, s. 200-210]. Wykorzystywana jest ona, aby zapewnić wyznaczenie zestawu rozwiązań o różnych własnościach w jednym przebiegu obliczeń. Koewolucja nie jest jedyną znaną techniką rozwiązywania zadań wielokryterialnych za pomocą algorytmów genetycznych. Istnieją również inne metody umożliwiające wyznaczenie zestawu rozwiązań niezdominowanych w jednym przebiegu obliczeń [Laumans i in., 2001]. Dla naszych potrzeb okazała się jednak bardzo wygodna w zastosowaniu.

Celem wykorzystanego algorytmu było znalezienie jak najlepszego zestawu portfeli akcji. Jakość rozwiązań opisywana była ważoną funkcją oczekiwanej stopy zwrotu i jej semiwariancji celowej [Markowitz, 1970, s. 188-201]. Jako wartość celu przyjęto zero, co oznacza że ryzyko związane było wyłącznie z poniesieniem straty. Takie podejście nazywane jest negatywną koncepcją ryzyka [Jajuga, 2007, s. 13]. Wagi obu składników musiały być nieujemne i sumować się do jedności.

Zbiór rozwiązań podzielony był na mniejsze części. Każdej z nich przypisano nieco inne wartości wag obu składników przystosowania. W pierwszej poszukiwano możliwie najbezpieczniejszego portfela. W kolejnych podzbiorach waga zysku stopniowo rosła, a ryzyka malała, aż w ostatnim jedynym kryterium stawał się zysk. Zbiory różniące się od siebie o jeden krok wartości wag uznawano za sąsiadujące. Istotne jest, że zarówno semiwariancja prosta, jak i celowa są nieróżniczkowalne, co uniemożliwia stosowanie ich jako parametrów funkcji celu w klasycznych metodach optymalizacji [DeFusco i in., 2013]. Na szczęście zastosowanie algorytmu genetycznego pozwala nam ominąć ten problem.

Rozwiązania kodowane były jako ciąg liczb rzeczywistych z przedziału od 0 do 20 odpowiadających jednej z branych pod uwagę spółek. Udziały konkretnych akcji w portfelu były wyznaczane jako udział wartości danej liczby w sumie wszystkich wartości w ciągu kodowym. W zapisie tym istotne jest, aby dolna dopuszczalna wartość genu była zerowa. Przyjęcie innych wartości utrudni odkodowanie rozwiązań. Górna wartość ma mniejsze znaczenie. Należy jednak pamiętać, aby zmiany wartości genów w procedurze mutacji były względem niej stosunkowo małe.

Zastosowanie takiego systemu wymusiło specyficzną budowę operacji krzyżowania. Każda z wartości w kodzie nowo tworzonego osobnika pochodziła od losowo wybranego rodzica, ale podlegała korekcie ze względu na łączną sumę wartości w kodach obu rodziców. W każdej iteracji algorytmu tworzono tylko jedno nowe rozwiązanie dla każdej niszy. Aby ustalić, gdzie należy je umieścić, losowano z właściwej podpopulacji trzy rozwiązania. Nowy osobnik zastępował najgorsze z nich. Mechanizm ten zmniejszał szansę utraty najlepszych rozwiązań wskutek zbytniej losowości operacji genetycznych.

Mutacja przeprowadzona była dwustopniowo. Najpierw dla każdego nowo tworzonego rozwiązania sprawdzano, czy mogą wystąpić w nim losowe zaburzenia genów. Jeśli odpowiedź była twierdząca, wykonywano dodatkowe testy dla każdej wartości w jego zapisie kodowym. Wylosowane w ten sposób pozycje były następnie modyfikowane w niewielkim stopniu w górę lub w dół.

2. Procedura testowa

Obliczenia podzielono na dwie główne grupy. W pierwszej badano wpływ zmian wartości kontrolujących ilość nisz i wymianę informacji pomiędzy nimi. W drugiej zmianie ulegały parametry operatora mutacji. Dla każdego badanego zestawu parametrów startowych obliczenia przeprowadzono po tysiąc razy. Następnie zebrano wyniki ze wszystkich przebiegów i podzielono je na grupy według tego, jakie parametry ulegały zmianie w kolejnych seriach obliczeń.

Dla każdej serii testów wyniki następnie rozdzielono według nisz, dla których je uzyskano. Raporty zawierały najlepsze znalezione wartości w kolejnych iteracjach algorytmu. Spośród wszystkich końcowych rozwiązań w danej niszy dla badanego zestawu wartości startowych wybierano najlepszy znaleziony portfel. Następnym krokiem było ustalenie, w której iteracji w każdym z przebiegów znaleziono rozwiązania dostatecznie bliskie najlepszemu. Jako wartości graniczne przyjęto odchylenia od wartości najlepszego przystosowania o nie więcej niż 0,1%, 0,01% i 0,001%.

Dla przebiegów o identycznych parametrach wyznaczano medianę iteracji, w której osiągnięto każdy z progów. W przypadku, gdy żądanej wartości nie udało się uzyskać, przyjmowano do jej wyznaczenia moment zakończenia obliczeń, czyli 100 000 pokoleń. Miało to miejsce głównie w przypadku przebiegów o bardzo niskim prawdopodobieństwie zajścia mutacji. Mediana wydaje się lepszą miarą niż zwykła średnia, gdyż jest mniej wrażliwa na pojawienie się nietypowych wyników.

Znając medianę momentu osiągnięcia celu, można było już łatwo ustalić, które wartości parametrów pozwoliły najszybciej znaleźć rozwiązania dostatecznie bliskie optymalnym. Podejście takie jest znacznie bardziej czasochłonne, niż porównanie tylko pojedynczych przebiegów algorytmu przy różnych wariantach ustawień. Bierze ono jednak pod uwagę stabilność rozwiązań, co dla algorytmów genetycznych wykorzystujących wiele operacji losowych jest bardzo istotne.

Domyślnie populację dzielono na 15 nisz, z których każda zawierała po 30 rozwiązań. Wagi zysku i ryzyka w kolejnych niszach przedstawia tabela 1.

W podstawowym wariancie prawdopodobieństwo wyboru rozwiązania do mutacji wynosiło początkowo 0,1 i rosło o 0,09 po każdych 100 iteracjach bez poprawy najlepszego wyniku w danej niszy. Prawdopodobieństwo mutacji konkretnego genu we wskazanych osobnikach było stałe i wynosiło domyślnie 0,05.

Tabela 1. Wartości wag składowych funkcji przystosowania w kolejnych niszach

Nisza	1	2	3	4	5	6	7	8
Zysk	0	0,071	0,143	0,214	0,286	0,357	0,429	0,5
Ryzyko	1	0,929	0,857	0,786	0,714	0,643	0,571	0,5
Nisza	9	10	11	12	13	14	15	x
Zysk	0,571	0,643	0,714	0,786	0,857	0,929	1	x
Ryzyko	0,429	0,357	0,286	0,214	0,143	0,071	0	x

Powyższą metodę można dostosować dla właściwie dowolnych algorytmów genetycznych. Konieczne jest jedynie, aby program udostępniał w końcowym raporcie nie tylko końcowe wyniki, ale również historie postępów przystosowania najlepszych rozwiązań. Wielokrotne powtórzenie obliczeń i wyznaczenie mediany momentu osiągnięcia celu pozwala wyznaczyć takie parametry, które dają nam dużą szansę znalezienia satysfakcjonujących wyników w pojedynczym przebiegu obliczeń. W przypadku jednak, gdy użytkownik chce być jeszcze ostrożniejszy, medianę można również zastąpić innymi, bardziej restrykcyjnymi, kwantylami rozkładu.

3. Testy wymiany informacji

Zacznijmy od przyjrzenia się wynikom testów wymiany informacji pomiędzy niszami. Przeprowadzono trzy serie obliczeń dla 3, 7 i 15 nisz. W każdym przypadku wymiana mogła być całkowicie wyłączona, zostać ograniczona do 5, 10, 15, 20, 25, 50 lub 75 tysięcy pokoleń, lub też odbywać się bez żadnych ograniczeń. Jak już wyżej wspomniano, dla każdego z wariantów obliczenia powtórzono po 1000 razy. Aby dokonać porównania pomiędzy wariantami wybrano 3 nisze, które występowały w każdym z nich. Wyniki uzyskane dla pierwszej, w której kryterium przystosowania jest jedynie ryzyko, zawiera tabela 2.

Tabela 2. Liczba pokoleń potrzebnych do osiągnięcia wymaganej precyzji dla pierwszej niszy

Nisza pierwsza	3 nisze			7 nisz			15 nisz		
	odchylenie (%)			odchylenie (%)			odchylenie (%)		
Limit	0,1	0,01	0,001	0,1	0,01	0,001	0,1	0,01	0,001
0 tys.	12 917	15 448	21 571	12 993	15 490	21 930	13 057	15 572	21 637
5 tys.	6104	7701	12 194	2275	4792	8963	1310	2681	7770
10 tys.	6654	9434	13 925	2322	4813	11 016	1331	2704	8883
15 tys.	6757	9598	16 404	2290	4704	11 427	1318	2762	9361
20 tys.	6872	9588	16 919	2308	4988	11 296	1325	2775	9119
25 tys.	6668	9500	16 183	2319	4918	11 798	1317	2759	9057
50 tys.	6741	9558	16 816	2280	4777	11 470	1325	2676	8999
75 tys.	6641	9617	16 734	2309	4988	11 633	1312	2667	9084
100 tys.	6682	9348	16 636	2318	4951	11 566	1333	2761	9126

Bez wymiany informacji pomiędzy niszami algorytm działa wyraźnie wolniej niż w pozostałych wariantach. Tempo zbieżności jest też w takim przypadku niezależne od liczby nisz w populacji. Gdy wymiana informacji jest dopuszczalna, liczba pokoleń potrzebna do osiągnięcia kolejnych progów wyraźnie spada. Obserwacje te powtarzały się również w dalszych eksperymentach. Zauważmy, że brak wymiany informacji jest równoznaczny z przeprowadzeniem serii całkowicie niezależnych przebiegów algorytmu genetycznego dla pojedynczych nisz. Możemy więc uznać, że zastosowanie algorytmu koewolucyjnego rzeczywiście daje wyniki szybciej niż całkowicie niezależne przebiegi obliczeń dla oddzielonych od siebie populacji.

Tabela 3 przedstawia wyniki, jakie otrzymaliśmy dla środkowej z badanych nisz. Zysk i ryzyko traktujemy tu jako jednakowo ważne.

Tabela 3. Liczba pokoleń potrzebnych do osiągnięcia wymaganej precyzji dla środkowej niszy

Nisza środkowa	3 nisze			7 nisz			15 nisz		
	odchylenie (%)			odchylenie (%)			odchylenie (%)		
Limit	0,1	0,01	0,001	0,1	0,01	0,001	0,1	0,01	0,001
0 tys.	16 674	20 481	30 705	16 689	20 260	31 003	16 481	20 298	30 207
5 tys.	7147	10 717	20 760	2566	6720	15 507	1504	6088	14 291
10 tys.	8472	12 692	22 078	2660	9790	18 774	1526	7177	17 104
15 tys.	8490	15 241	24 270	2689	9924	21 364	1512	7114	20 310
20 tys.	8681	15 649	26 942	2731	10051	24 524	1509	7363	23 441
25 tys.	8689	16 113	30 450	2697	9941	27 062	1522	7303	26 594
50 tys.	8508	16 651	39 658	2728	9979	29 434	1512	7241	27 549
75 tys.	8570	15 985	38 944	2705	9980	30 657	1501	7127	26 737
100 tys.	8613	16 158	36 733	2726	10057	30 759	1513	7455	27 287

Uzyskane wyniki są podobne, jak w poprzednim przykładzie. Liczba iteracji konieczna do osiągnięcia kolejnych wyznaczonych celów przystosowania była jednak wyższa we wszystkich testowanych wariantach. Wynika to z faktu, że w środkowej niszy optymalna wartość przystosowania jest znacznie bliższa zeru. W rezultacie, używając procentowych odchyleń od najlepszego wyniku, stosujemy bardziej surowe kryterium niż w poprzednim przypadku.

Zwróćmy uwagę, że różnice pomiędzy liczbą iteracji wskazanej dla pełnej wymiany informacji oraz dla przerwania jej po 5 tysiącach pokoleń są tu znacznie wyraźniejsze niż w poprzednim przykładzie. Podczas gdy w pierwszej niszy różnice wynosiły ok. 20%, tym razem pełna wymiana informacji opóźnia znalezienie ostatniego progu ok. dwukrotnie w porównaniu z najlepszym wariantem. Wymagana liczba pokoleń wyraźnie rośnie, również gdy porównujemy kolejne progi w tym samym wariancie ustawień. Możemy więc uznać, że im większej precyzji wyników żądamy, tym korzystniejsze jest stosunkowo wczesne przerwanie wymiany informacji pomiędzy niszami.

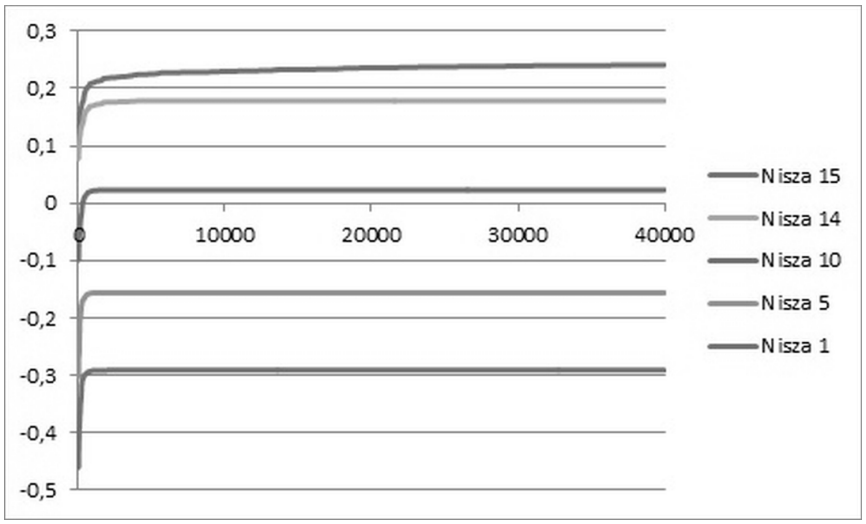
Na koniec przyjrzymy się wynikom uzyskanym dla niszy końcowej, w której jedynym branym pod uwagę kryterium był zysk. Ponieważ znamy dla niej dokładne rozwiązanie, do listy celów dodano moment jego znalezienia, czyli zerowe odchylenie od najlepszego wyniku. Rezultaty testów przedstawia tabela 4.

Tabela 4. Liczba pokoleń potrzebnych do osiągnięcia wymaganej precyzji dla ostatniej niszy

Nisza ostatnia	3 nisze				7 nisz				15 nisz			
	odchylenie (%)				odchylenie (%)				odchylenie (%)			
Limit	0,1	0,01	0,001	0	0,1	0,01	0,001	0	0,1	0,01	0,001	0
0 tys.	57 062	57 996	58 101	58 101	57 537	58 673	58 704	58 714	57 342	58 274	58 324	58 324
5 tys.	47 611	48 579	48 687	48 687	39 931	40 921	41 051	41 052	37 406	38 403	38 458	38 467
10 tys.	47 795	48 628	48 757	48 757	41 195	42 240	42 336	42 336	39 224	40 180	40 181	40 181
15 tys.	49 284	50 343	50 434	50 434	42 985	43 940	43 955	44 012	40 601	41 645	41 767	41 767
20 tys.	51 121	51 961	52 039	52 059	44 146	45 026	45 147	45 147	39 046	40 128	40 255	40 273
25 tys.	52 907	53 953	54 090	54 103	42 672	43 585	43 818	43 818	31 558	32 481	32 536	32 536
50 tys.	42 228	42 875	42 878	42 878	35 592	36 740	36 835	36 844	35 041	36 309	36 444	36 444
75 tys.	41 785	42 319	42 423	42 434	35 749	36 813	36 951	36 951	34 991	36 376	36 497	36 503
100 tys.	42 073	42 707	42 728	42 728	35 636	36 602	36 709	36 715	34 917	36 444	36 616	36 637

Dla 15 nisz korzystne było przerwanie wymiany przed zakończeniem obliczeń, dokładniej zaś po 25 tysiącach iteracji. Rezultaty wyraźnie odbiegają jednak tym razem od dwóch poprzednich przykładów. Wynika to głównie z faktu, że funkcja przystosowania zachowuje się tutaj nieco inaczej niż w pozostałych niszach. Jej postępy w losowo wybranym przebiegu algorytmu dla 15 nisz przedstawia rys. 1.

Zauważmy, że dla niszy 14 wykres bardzo szybko ulega spłaszczeniu. Podobnie zachowuje się przystosowanie w większości pozostałych podpopulacji, również tych, które pominięto dla lepszej czytelności wykresu. Wyjątkiem jest tu nisza 15, dla której dość wyraźny wzrost przystosowania utrzymuje się przez kilkadziesiąt tysięcy pokoleń. Przypuszczalnie wynika to z faktu, że zastosowany operator krzyżowania nie sprawdza się przy wyznaczeniu wartości zmiennych na krańcach dopuszczalnego przedziału. Ponieważ optymalne rozwiązanie w niszy 15 składa się z akcji tylko jednej spółki, uzyskanie go jest w dużej mierze zależne od sprawnego działania mutacji.



Rys. 1. Postępy funkcji przystosowania w wybranych niszach w zależności od numeru iteracji algorytmu

We wszystkich przypadkach ze wzrostem liczby podpopulacji malała liczba pokoleń wymaganych do osiągnięcia kolejnych progów. Gdy warunki w sąsiadujących niszach są do siebie zbliżone, krzyżujemy między nimi bardziej podobne rozwiązania. Tym samym większe są szanse, że przeniesione geny będą dobrze sprawdzać się w nowym środowisku. Zatem im więcej nisz, tym mniej iteracji potrzebujemy na znalezienie się w obszarze rozwiązań suboptymalnych.

Nie oznacza to jednak, że powinniśmy mnożyć nisze ponad potrzebę. Dla większej liczby bardziej podobnych do siebie nisz dobre rozwiązania znajdujemy w mniejszej liczbie iteracji. Jednocześnie jednak wraz z ich liczbą wzrasta łączny czas działania algorytmu. Wykonanie 100 tysięcy iteracji zajmowało dla 3 nisz 25 sekund, dla 15 zaś ok. 2 minut. Tymczasem liczba iteracji wymaganych do znalezienia trzeciego z wymaganych progów przy wzroście liczby nisz z 3 do 15 spadała w najlepszym wypadku o 46%. Zwykle było to zaś wyraźnie

mniej. Zwiększenie liczby podpopulacji wydłuża więc czas potrzebny na uzyskanie wyników bardziej niż oszczędności, jakie możemy uzyskać, odpowiednio zmniejszając liczby pokoleń algorytmu. Zatem jeśli zależy nam na czasie, a jednocześnie nie potrzebujemy oszacowań zbyt wielu różnych wariantów rozwiązań, lepiej jest stosować niewielką ich liczbę.

4. Testy działania mutacji

Drugim z elementów algorytmu, który przetestowano, był mechanizm mutacji. Jak już wspomniano, operacja ta była prowadzona dwustopniowo. Zbadano więc, jakie wartości prawdopodobieństwa w obu testach zapewniają najszybsze znalezienie rozwiązań suboptymalnych. Wykonano w tym celu dwie serie obliczeń. W pierwszej modyfikowano prawdopodobieństwo wyboru rozwiązania, w którym wprowadzano zaburzenia. W drugiej zaś dla ustalonego prawdopodobieństwa wyboru mutowanych rozwiązań zmianie ulegało prawdopodobieństwo zmiany poszczególnych wartości genów.

Zacznijmy od zbadania różnych wartości prawdopodobieństwa wyboru mutowanego rozwiązania. Wykonano testy dla 11 wariantów ustawień. Minimalne prawdopodobieństwo wynosiło 0,01. Następne wartości były zaś wielokrotnościami 0,1. Wymagane progi dokładności były takie same, jak w przypadku testów działania nisz, czyli 0,1%, 0,01% i 0,001%. Dla lepszej przejrzystości wyników zaprezentowano jednak wyłącznie rezultaty uzyskane dla trzeciego z nich. Przypomnijmy też, że domyślne prawdopodobieństwo w teście mutacji drugiego stopnia wynosiło 0,05. Uzyskane wyniki przedstawia tabela 5.

Tabela 5. Mediana liczby pokoleń, po których znaleziono wynik odbiegający od najlepszego o 0,001%, w zależności od niszy i prawdopodobieństwa mutacji pierwszego stopnia

P. mutacji	Nisza														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0,01	44 295	13 422	17 576	34 242	51 245	21 370	80 376	1E+05	82 612	21 471	14 671	11 033	20 896	20 886	1E+05
0,10	9256	6022	8134	10 970	13 886	10 324	21 597	30 810	25 197	8498	5950	5290	6776	6144	38 619
0,20	5554	4235	5629	7239	8946	7212	13 438	18 800	16 231	6427	4171	4099	5193	4823	24 186
0,30	4237	3712	4751	5527	6989	5671	10 079	13 271	12 739	4918	3632	3596	4533	4229	18 725
0,40	3507	3272	4136	4840	5887	4859	8422	11 048	10 176	4361	3160	3331	4167	3884	16 069
0,50	3161	3089	3689	4260	5172	4311	7114	9146	9088	4025	3034	3085	3818	3638	14 219
0,60	2859	2877	3350	3975	4654	3997	6522	8585	7957	3895	2813	2852	3666	3536	13 163
0,70	2684	2775	3264	3731	4325	3815	6210	7841	7397	3512	2701	2821	3513	3379	12 410
0,80	2533	2789	3181	3593	4227	3550	5645	7190	6692	3299	2673	2709	3407	3302	11 712
0,90	2495	2701	3096	3474	3914	3483	5361	6773	6419	3375	2648	2707	3325	3272	11 276
1,00	2366	2636	3039	3424	3828	3526	5149	6513	6380	3252	2612	2647	3262	3218	11 029

Losowanie mutowanych osobników z prawdopodobieństwem 0,01 prowadziło do bardzo wolnych postępów algorytmu. Zwiększenie tej wartości do 0,10 zmniejsza wymaganą liczbę iteracji co najmniej o połowę. W wielu niszach różnica jest jednak jeszcze większa. Dalsze jej podnoszenie pozwala jeszcze bardziej skrócić obliczenia. Choć spodziewano się, że optymalne prawdopodobieństwo będzie stosunkowo niskie, w rzeczywistości okazało się, że najlepsze wyniki uzyskano, gdy wynosiło ono 1. W praktyce więc najlepiej sprawdzała się mutacja jednostopniowa, pomijająca losowy wybór osobników i testująca wystąpienie losowych zaburzeń dla wszystkich genów w populacji.

Oznacza to, że przyjęte na początku założenie o zwiększaniu prawdopodobieństwa mutacji pierwszego stopnia, gdy zbyt długo nie następuje poprawa wyników, okazało się niepotrzebne. Lepsze rezultaty dawało ustalenie tego parametru na stałym wysokim poziomie.

Pozostało nam jeszcze ustalenie, jakie powinno być prawdopodobieństwo zaburzenia konkretnych genów dla rozwiązań wybranych do mutacji. Przetestowano jego wartości z zakresu od 0,01 do 0,25. Ponownie ograniczymy się do przedstawienia wyników dla żądanej dokładności 0,001%. Uzyskane wyniki przedstawia tabela 6.

Tabela 6. Mediana liczby pokoleń, po których znaleziono wynik odbiegający od najlepszego o 0,001%, w zależności od niszy i prawdopodobieństwa mutacji drugiego stopnia

P. mutacji	Nisza														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0,01	20 153	9385	13 094	19 510	24 966	16 527	36 347	50 458	40 890	14 347	10 316	7731	12 376	11 724	82 486
0,025	12 012	6899	9763	12 893	17 335	11 742	26 793	33 449	28 091	10 227	7268	6088	8391	7724	50 444
0,05	9295	5717	8033	10 412	13 713	9735	20 737	27 427	22 858	8325	5806	5202	6801	6246	36 832
0,075	8487	5637	7790	10 132	12 553	9186	19 283	25 805	21 805	8145	5264	4877	6200	5717	32 085
0,1	8243	5236	7452	9902	12 524	9153	19 504	27 084	20 880	8113	5262	4771	5948	5527	30 364
0,125	8619	5663	7490	10 325	13 420	9278	19 996	28 896	22 064	8393	5043	4794	5907	5352	29 999
0,15	9398	5764	8076	10 782	14 385	10 333	21 732	29 703	22 479	7610	5316	5036	5882	5359	30 750
0,175	10 589	6176	8857	11 543	15 279	10 777	24 348	35 066	24 705	8605	5445	5158	5845	5310	31 615
0,2	12 328	6769	9732	12 645	16 588	11 491	24 978	38 060	26 180	8887	6041	5470	6148	5544	33 352
0,225	13 147	6843	10 061	13 042	17 884	12 582	29 397	45 275	28 315	9322	5994	5461	6340	5618	35 343
0,25	14 974	7606	11 311	15 065	19 485	13 612	32 591	54 425	31 872	10 070	6396	5423	6587	5878	38 044

W większości przypadków najlepsze rezultaty dawało prawdopodobieństwo na poziomie 0,1 lub 0,125. Wpływ mutacji drugiego stopnia generalnie nie był aż tak silny, jak stopnia pierwszego. Tym niemniej ustalenie jej prawdopodobieństwa na zbyt wysokim lub zbyt niskim poziomie może wyraźnie spowolnić osiągnięcie wymaganego celu. Dzieje się tak zwłaszcza wówczas, gdy wyma-

gamy wysokiej dokładności wyników, jak np. w niszy nr 8. Należy też zauważyć, że dla niższych progów dokładności preferowana była bardziej intensywna mutacja. Różnice w liczbie wymaganych do osiągnięcia celu iteracji pomiędzy prawdopodobieństwem 0,1 a 0,25 były jednak w takim przypadku bardzo małe.

Przeprowadzone testy wskazały jako optymalne dosyć wysokie łączne prawdopodobieństwo zajścia mutacji. Jest to prawdopodobnie specyficzna cecha zastosowanego algorytmu. Procedura wymiany rozwiązań pomiędzy pokoleniami bazuje na podejściu elitarnym, zabezpieczając populację przed nadmiernym wpływem niepożądanych zaburzeń, jednocześnie zachowując zmiany korzystne. W rezultacie mutacja ma bardzo duży wpływ na tempo zbieżności obliczeń. Odpowiedni dobór jej parametrów pozwala wyraźnie zmniejszyć liczbę pokoleń wymaganą do uzyskania wyników bliskich optymalnym w porównaniu z wariantem bazowym. Projektując nowy algorytm genetyczny, nie powinniśmy więc z góry zakładać, że wpływ mutacji będzie niewielki.

Podsumowanie

Dzięki wymianie informacji pomiędzy niszami algorytmy koewolucyjne działają szybciej, niż całkowicie niezależne przebiegi algorytmu genetycznego dla oddzielonych od siebie populacji. Wydaje się, że wymianę tę jednak zwykle warto przerwać, gdy zbliżymy się do rozwiązań optymalnych. Często następuje to już na dość wczesnym etapie działania algorytmu. Gdy zbieżność rozwiązań jest stosunkowo powolna, wskazane może być jednak dłuższe jej podtrzymywanie. Dzięki zwiększeniu liczby wykorzystywanych nisz możemy zbieżność przyspieszyć, choć w praktyce nie uzyskamy dzięki temu oszczędności czasu. Możliwe jest jednak ograniczenie w pewnym stopniu długości trwania obliczeń w sytuacjach, gdy potrzebujemy oszacowań wielu zróżnicowanych rozwiązań.

Jak się okazało, bardzo duży wpływ na działanie algorytmu miało prawdopodobieństwo mutacji pierwszego stopnia. Właściwy jego dobór pozwalał przyspieszyć działanie algorytmu ok. trzykrotnie. Wpływ prawdopodobieństwa mutacji drugiego stopnia był mniejszy. Nie powinno być ono jednak zbyt wysokie lub niskie, zwłaszcza gdy chcemy uzyskać wyniki o dużej precyzji. Zwykle przyjmuje się, że mutacja ma tylko niewielkie znaczenie dla sprawnego działania algorytmów genetycznych. Uzyskane wyniki wskazują jednak, że nie zawsze musi to być prawdą.

Na koniec warto wyraźnie zaznaczyć, że optymalne wartości parametrów startowych zależą oczywiście od konkretnego przypadku. Sama procedura ich

wyboru, choć czasochłonna, może być jednak z powodzeniem zastosowana właściwie dla dowolnego algorytmu genetycznego. Tym samym może w dłuższym okresie przynieść znaczne oszczędności czasu w przypadkach, gdy spodziewamy się wielokrotnie stosować ten sam algorytm genetyczny dla podobnych danych.

Literatura

Angelova M., Pencheva T. (2011), *Tuning Genetic Algorithm Parameters to Improve Convergence Time*, „International Journal of Chemical Engineering".

Arabas J. (2001), *Wykłady z algorytmów ewolucyjnych*, Wydawnictwo Naukowo-Techniczne, Warszawa.

DeFusco R., McLeavey D., Pinto J., Runkle D. (2013), *Quantitative investment analysis*, Wiley, Hoboken.

Goldberg D. (2003), *Algorytmy genetyczne i ich zastosowania*, Wydawnictwo Naukowo-Techniczne, Warszawa.

Gwiazda T. (1998), *Algorytmy genetyczne – zastosowania w finansach*, Wydawnictwo WSPiZ, Warszawa.

Jajuga K. (2007), *Zarządzanie ryzykiem*, Wydawnictwo Naukowe PWN, Warszawa.

Laumans M., Thiele L., Deb K., Zitzler E. (2001), *On the Convergence and Diversity Preservation of Multi-Objective Evolutionary Algorithms*, TIK Report No. 108, Institut für Technische Informatik und Kommunikationsnetze, Zürich.

Markowitz H. (1970), *Portfolio Selection Efficient Diversification of Investments*, Cowles Foundation Monograph 16, Yale University Press, New Heaven and London.

IMPACT OF THE STARTING PARAMETERS ON THE CONVERGENCE OF RESULTS FOR A COEVOLUTIONARY GENETIC ALGORITHM

Summary: In this work we propose a procedure for testing the impact of starting parameters on the convergence of a genetic algorithm. Although the described solution is quite time consuming it takes into consideration both number of iterations required and stability of obtained results. We then proceed to infer optimal values of such parameters for a certain co-evolutionary portfolio analysis algorithm. We prove, that such an implementation is superior to simple genetic algorithms operating on a single population when dealing with multi-objective fitness functions. However, the exchange of information between different niches should not be enabled for too long. We also point out the big impact that often disregarded mutation procedure can have on the convergence to sub-optimal solutions. Interestingly, both too high and too low probability of mutation can have a noticible negative impact on the performance of a given algorithm.

Keywords: genetic algorithms, co-evolution, convergence.