WYŻSZA SZKOŁA
INFORMATYKI STOSOWANEJ
I ZARZĄDZANIA

ANALIZA SYSTEMOWA W FINANSACH I ZARZĄDZANIU

Wybrane problemy

Pod redakcją
Jerzego HOŁUBCA

WYŻSZA SZKOŁA
INFORMATYKI STOSOWANEJ
I ZARZĄDZANIA

ANALIZA SYSTEMOWA W FINANSACH I ZARZĄDZANIU
Wybrane problemy

Pod redakcją
Jerzego HOŁUBCA

Warszawa 1999

Wykaz opiniodawców artykułów zamieszczonych w tomie:

prof. dr hab. Jerzy **HOŁUBIEC**

prof. dr hab. Janusz **KACPRZYK**

prof. dr hab. Tadeusz **NOWICKI**

prof. dr hab. Stanisław **PIASECKI**

prof. dr hab. Piotr **SZCZEPANIAK**

prof. dr hab. Tadeusz **TRZASKALIK**

doc. dr hab. Sławomir **WIERZCHOŃ**

doc. dr hab. Leszek **ZAREMBA**

Warszawa 1999

ISBN 83-85847-24-3

Przedmowa

Na niniejszą publikację składa się zbiór prac doktorantów Zaocznych Studiów Doktoranckich "Informatyka w zarządzaniu i finansach" działających przy ***Instytucie Badań Systemowych Polskiej Akademii Nauk***.

Prace te były referowane na konferencji BOS'98 "Rozwój średnich i małych miast w XXI wieku w Polsce: Rola badań operacyjnych i systemowych", Kutno, 8-10 czerwca 1998 r.[1], a także na seminariach Studiów Doktoranckich "Informatyka w zarządzaniu i finansach". Nad stroną merytoryczną publikacji czuwał Pan Prof. dr hab. Jerzy Hołubiec oraz grono recenzentów i opiekunów naukowych doktorantów.

Prace dotyczą głównie problemów analizy systemowej oraz jej zastosowań w dziedzinie finansów, a zwłaszcza - teorii portfela, obligacji i problemów inwestycyjnych. Niektóre prace przy analizie finansowej posługują się tzw. algorytmami genetycznymi i sieciami neuronowymi, a także modelowaniem rozmytym i strukturami fraktalnymi. Część prac dotyczy zarządzania i sterowania produkcją.

Wypada zauważyć, iż doktoranci Studiów atakują w swych pracach tematy nowoczesne i znajdujące się w obszarze tzw. frontu badawczego analizy systemów. Wypada im życzyć sukcesów i wytrwałości w pracy, która winna zakończyć się obronioną pracą doktorską.

---

[1] Głównymi organizatorami konferencji było Polskie Towarzystwo Badań Operacyjnych i Systemowych oraz Instytut Badań Systemowych PAN.

Wypada także zaznaczyć, iż wydanie niniejszej publikacji stało się możliwe dzięki wsparciu finansowemu ze strony ***Wyższej Szkoły Informatyki Stosowanej i Zarządzania***, działającej w ramach Fundacji Krzewienia Nauk Systemowych. Fundacja ta została założona w 1991 roku z inicjatywy Prof. L. Kuźnickiego, wówczas Sekretarza Naukowego Polskiej Akademii Nauk. Do zadań Fundacji należy, między innymi, wspieranie i promocja prac młodych pracowników nauki, a zwłaszcza prac doktorantów.

Mamy nadzieję, iż publikacja niniejsza zostanie życzliwie przyjęta przez specjalistów działających w obszarze nauk systemowych.

Rektor WSISiZ

Prof. Roman Kulikowski

ZASTOSOWANIE ALGORYTMÓW GENETYCZNYCH DO ROZWIĄZYWANIA PROBLEMÓW TRUDNYCH I NP ZUPEŁNYCH

Mirosław MICHAŚ
Zaoczne Studia Doktoranckie IBS PAN

1. Problemy P zupełne i NP zupełne

Problem jest kwalifikowany do grupy problemów NP (niedeterministycznych o wielomianowej czasowej złożoności obliczeniowej) jeżeli jest rozwiązywalny w czasie wielomianowym przez niedeterministyczną maszynę Turinga.

P-Problem (którego rozwiązanie jest połączone przez wielomian) jest również klasy NP. Jeżeli przekształcenie do klasy NP jest znane, może być zredukowane do pojedynczej P (w czasie wielomianowym) weryfikacji.

Problem jest nazwany NP-trudnym jeżeli algorytm rozwiązania może być przekształcony w inny NP.- problem. Dużo prościej wykazać, że problem jest NP niż NP-trudny. Problem który jest jednocześnie NP i NP-trudny jest nazywany NP-zupełnym

MASZYNA TURINGA (MT)

Stworzona przez Alana Turniga w 1935 roku teoretyczna maszyna licząca powstała na potrzeby informatyki teoretycznej. Za po-

mocą MT można przedstawić wszystkie języki programowania, w tym języki wysokiego poziomu i architektury komuterowe.

Maszyna posiada nieskończoną taśmę magnetyczną na której mogą być zapisywane i kasowane instrukcje sterujące w regularnych przedziałach czasowych, jednobitowy rejestr pamięci (nazywany czasem stosem) i procesor umożliwiający wykonanie następujących instrukcji: przesunięcie taśmy w prawo i w lewo, zmiana stanu pamięci bazującej na wartości aktualnej taśmy, zapis i kasowanie aktualnej zawartości taśmy. Przy każdym kroku maszyna czyta jeden symbol z taśmy w aktualnym położeniu głowicy czytającej. Dla każdej kombinacji aktualnych stanów i wczytanych symboli maszyna specyfikuje nowy stan i symbol. Stanem może być przesunięcie taśmy o jedną pozycję lub stan osobliwy. Maszyna realizuje instrukcje, dopóki nie osiągnie stanów osobliwych. Stan osobliwy powoduje zatrzymanie maszyny.

Maszyna Turinga zatrzymuje się na polecenie.

(Niedeterministyczna maszyna Turinga jest maszyną równoległą, która może prowadzić kilka obliczeń jednocześnie traktowanych jako równoległe maszyny Turinga bez możliwości wzajemnej komunikacji.

ZŁOŻONOŚĆ OBLICZENIOWA

Złożoność obliczeniowa algorytmu definiuje się jako ilość zasobów komputerowych, potrzebnych do jego wykonania. Podstawowymi zasobami rozważanymi w analizie algorytmów są czas działania i ilość zajmowanej pamięci.

W zależności od jednostek w których mierzymy złożoność wyróżniamy złożoność czasową, i złożoność pamięciową.. Przy złożoności pamięciowej za jednostkę przyjmuje się słowo pamięci maszyny. Złożoność czasowa jest własnością samego algorytmu jako metody rozwiązywania problemu. W agorytmie wyznacza się kluczowe z punktu widzenia wykonywania operacje, takie, że ich liczba jest pro-

porcjonalna do liczby wykonań wszystkich operacji jednostkowych. które nazywa się operacjami dominującymi. Za jednostkę złożoności czasowej przyjmuje się wykonanie jednej operacji dominującej.

Złożoność obliczeniową algorytmu traktuje się jako funkcję rozmiaru danych n. Wyróżnia się złożoność pesymistyczną definiowaną jako ilośc zasobów komputerowych potrzebnych przy najgorszych danych wejściowych rozmiaru n, oraz złożoność oczekiwaną – definiowaną jako ilość zasobów komputerowych , potrzebnych przy typowych danych wejściowych rozmiaru n.

Przez pesymistyczną złożoność czasową algorytmu rozumie się funkcję

$$W(n)=\sup\{t(d):d\in D_n\},$$

gdzie sup oznacza kres górny zbioru.

2. Istota, definicja i podstawy działania algorytmów genetycznych

Algorytmy genetyczne były formalnie zaprezentowane w Stanach Zjednoczonych w latach sześćdziesiątych przez Johna Hollanda z University of Michigan. Ciągły rozwój technologii obliczeniowych spowodował, że stały się one atrakcyjne dla zastosowań optymalizacyjnych. Algorytmy te sprawdzają się dobrze podczas rozwiązywania mieszanych problemów kombinatorycznych (ciągłych i dyskretnych). Trochę gorsze są przy poszukiwaniu lokalnego optimum, lepiej sprawdzają się tu metody przeszukiwania gradientowego.

Z algortymicznego punktu widzenia są bardzo kosztowne pamięciowo i czasowo.

Algorytmy genetyczne są to algorytmy poszukiwania oparte na mechanizmach doboru naturalnego oraz dziedziczności. Łączą w sobie ewolucyjną zasadę przeżycia najlepiej przystosowanych z systema-

tyczną, choć losową wymianą informacji. W każdym pokoleniu powstaje nowy zespół sztucznych organizmów (ciągów bitowych), utworzonych z połączenia fragmentów najlepiej przystosowanych przedstawicieli. poprzedniego pokolenia. Prócz tego wykorzystuje się sporadycznie nową część składową. Pomimo pewnego elementu losowości algorytmy genetyczne nie sprowadzają się do zwykłego przypadkowego błądzenia.

Terminy genetyczne i ich odpowiedniki w algorytmach genetycznych:

Genetyka	Algorytmy Genetyczne
Chromosom	Ciąg kodowy
Gen	Cecha, znak, detektor
Allel	Wariant cechy
Locus	Pozycja
Genotyp	Struktura
Fenotyp	Zbiór parametrów, rozwiązanie, punkt
Epistaza	Nieliniowość

Warunki konieczne zaliczenia algorytmu do GA:

a. GA nie przetwarzają bezpośrednio parametrów zadania lecz ich zakodowaną postać,
b. GA prowadzą poszukiwania, wychodząc nie z pojedynczego punktu, lecz z pewnej ich populacji,
c. GA korzystają tylko z funkcji celu, nie zaś jej pochodnych lub innych pomocniczych informacji.

d. GA stosują probabilistyczne, a nie deterministyczne reguły wyboru

3. Schemat algorytmu genetycznego

Aby zastosować algorytm genetyczny do rozwiązania konkretnego problemu należy określić:

a. genetyczną reprezentację potencjalnych rozwiązań problemu
b. metody generowania populacji rozwiązań początkowych
c. funkcje dopasowywania (ocena potencjalnych rozwiązań)
d. operatory genetyczne zmieniające geny w chromosomach
e. parametry stałe (rozmiar populacji, prawdopodobieństwo zastosowania
f. warunki zakończenia generowania kolejnej populacji

Ogólny algorytm genetyczny może mieć następującą postać:

```
Genetic_algorithm()
{
int t;   /*zmienna t iteruje kolejne populacje*/
t=0;
P(t)=utworzenie_populacji_początkowej ();
/* populacja zazwyczaj tworzona jest losowo*/
ocena_chromosomów ( P(t) )*/
/* chromosomom populacji P(t) przypisywane są wartości funkcji dopasowania*/
while (!warunek_końca() )
{
/* warunkami zakończenia algorytmu genetycznego mogą być zbadanie określonej a priori liczby populacji, mała zmienność wartości
```

funkcji dopasowania dla najlepszych chromosomów w kolejnych populacjach lub inne */

t=t+1;

C(t)=wybór(P(t-1));

/* z poprzedniej populacji P(t-1) wybierane są chromosomy tworzące populację C(t), dla której stosowane będą operatory genetyczne*/

CR(t)=przetworzenie(C(t), PC, PM);

/* dla chromosomów populacji C(t) stosowane są operatory genetyczne krzyżowania z prawdopodobieństwem PC oraz mutacji z prawdopodobieństwem PM; chromosomy wynikowe tworzą populację CR(t) */

ocena_chromosomów (CR(t));

/* chromosomom populacji CR(t) przypisywane są wartości funkcji dopasowania*/

P(t)= wybór(CR(t), P(t-1));

/* wybór chromosomów populacji P(t) z chromosomów populacji CR(t) oraz P(t-1)*/

}

}

Zmienna t oznacza numer kolejnej populacji. Podczas t-tej iteracji algorytm weryfikuje populację P(t)

Procedura utworzenie_populacji_początkowej wyznacza populację początkową P(0). Procedura ocena_chromosomów określa wartość funkcji dopasownia dla chromosomów populacji. Procedura przetworzenie stosuje do chromosomów populacji operatory genetyczne krzyżowania i mutacji odpowiednio z prawdopodobieństwem PC i PM. Procedura wybór wyznacza nową populację uwzględniając wartości funkcji dopasowania.

4. Przykładowy algorytm genetyczny

POSZUKIWANIE MAKSIMUM FUNKCJI $f(x)=x^2$ dla $x \in <0;31>$

(SYMULACJA ODRĘCZNA)

Aby zastosować algorytm genetyczny należy najpierw zakodować wartości zmiennej decyzyjnej zadania.

W tym przypadku można zastosować pięciopozycyjny zapis dwójkowy. Kod pięciobitowy umożliwia operowanie na liczbach całkowitych od 0 (00000) do 31 (11111). Mając funkcję celu (f(x)) i kod można przystąpić do symulacji jednego przebiegu algorytmu genetycznego.

Na początek wybierzemy losowo początkową populację złożoną z czterech ciągów kodowych. W tym celu można wykonać np. 20 rzutów symetryczną monetą. Można do tego użyć generatora liczb losowych.

Działanie algorytmu genetycznego zaczyna się od reprodukcji. Najpierw dobieramy pulę rodzicielską dla następnego pokolenia, kręcąc czterokrotnie tarczą ruletki. Podczas symulacji tego procesu otrzymano po jednej kopii ciągu nr 3. Porównanie tego wyniku z oczekiwaną liczbą kopii ($n \cdot pselect_i$) potwierdza nasze oczekiwania: najlepsi zwiększają swoją reprezentację, średni wychodzą na swoje , a najgorsi wymierają.

Gdy pula rodzicielska złożona z ciągów kodowych poszukujących partnerów jest już zapełniona, przychodzi czas na proces krzyżowania, który przebiega w dwóch etapach:

1. ciągi kodowe zostają skojarzone w sposób losowy w pary,
2. partnerzy krzyżują się ze sobą , przy czym punkt krzyżowania wybierany jest również losowo.

Tabela 1. Odręczna symulacja algorytmu genetycznego

Nr ciągu	Populacja początkowa wygenerowana losowo	Wartość x (liczba całkowita)	Wartość $f(x)=x^2$	Pselect $f_i/\sum f$	Oczekiwana liczba kopii f_i/f	Liczba kopii wylosowanych (wg. Reguły ruletki)
1	01101	13	169	0,14	0,58	1
2	11000	24	576	0,49	1,97	2
3	01000	8	64	0,06	0,22	0
4	10011	19	361	0,31	1,23	1
Suma			1170	1,00	4,00	4,00
Średnia			293	0,25	1,00	1,00
Maksimum			576	0,49	1,97	2,00

Operacja mutacji jest wykonywana losowo dla każdej pozycji ciągu z osobna. Przyjęliśmy, że prawdopodobieństwo mutacji w tym eksperymencie wynosi 0,001. Mając łącznie 20 pozycji/bitów możemy oczekiwać że zmiany dotkną 20·0,001=0,02 bitów w całej populacji. Praktycznie oznacza to, że żaden z bitów nie ulegnie mutacji w naszym doświadczeniu.

Po zakończeniu reprodukcji, krzyżowania i mutacji, nowe pokolenie ciągów musi zostać poddane ocenie. W tym celu dekodujemy otrzymane ciągi i obliczamy dla nich wartości wskaźnika przystosowania. Odpowiednie wyniki dla jednego przebiegu znajdują się w tabeli 2.

Z tabeli widać jak zmieniły się wskaźniki przystosowania w nowej populacji. Średni wskaźnik przystosowania wzrósł z 293 do 439, a

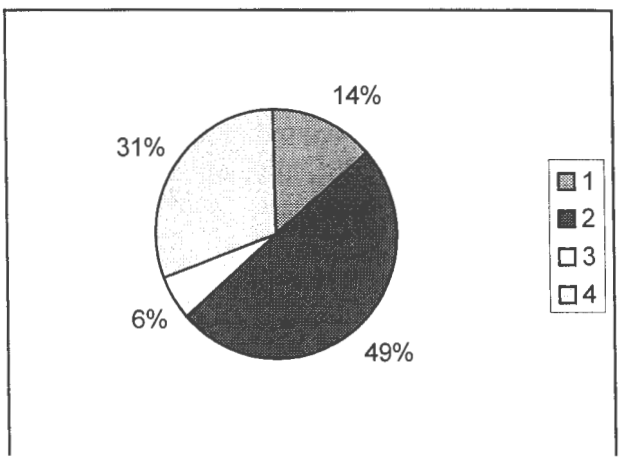
maksymalny z 576 do 729 w jednym tylko pokoleniu.Mimo, że procesy losowe pomagają w stworzeniu sprzyjających okoliczności, sama poprawa nie jest szczęśliwym trafem. Najlepszy ciąg kodowy z pierwszego pokolenia (11000) otrzymał dwie kopie dzięki wysokiej ponadprzeciętnej ocenie. Kiedy jedna z nich została losowo skojarzona z drugim pod względem wyników partnerem (10011), krzyżując się z nim (od również losowej) pozycji 2, jeden z otrzymanych w ten sposób potomków (11011) okazał się być naprawdę dobrym rozwiązaniem.

Tabela 2. Odręczna symulacja algorytmu genetycznego

Pula rodzicielska po reprodukcji (zaznaczono punkty krzyżowania)	Partner wybrany losowo	Punkt krzyżo-- wania wybrany losowo	Nowa populacja	Wartość x	Wartość $f(x)=x^2$
0110|1	2	4	01100	12	144
1100|0	1	4	11001	25	625
11|000	4	2	11011	27	729
10|011	3	2	10000	16	256
Suma					1754
Średnia					439
Maksimum					729

REGUŁA RULETKI

Operacja reprodukcji wybiera ciągi kodowe za pomocą ruletki, której sektory są proporcjonalne do wartości funkcji przystosowania. Tarcza widoczna poniżej została wykalibrowana wg. danych z tabeli 1



Uwagi

Opisany przykład jest odręczną symulacją jednego przebiegu algorytmu genetycznego bez zastosowania operatora mutacji:

1. Populacja początkowa wybrana na podstawie wyniku serii rzutów monetą (czterokrotne powtórzenie pięciu rzutów), przy umowie: orzeł=1, reszka=0.
2. Reprodukcja dokonywana jest według symulowanej reguły ruletki z dokładnością względną 1/8 (trzy rzuty monetą). Koło ruletki startuje od pierwszego chromosomu.
3. Krzyżowanie przeprowadzone na podstawie wyników 2 rzutów monetą, przy umowie: RR=002=0=punkt krzyżowania 1; ...; 112=3=punkt krzyżowania 4.
4. Prawdopodobieństwo krzyżowania pc=1,0.
5. Prawdopodobieństwo mutacji pm=0,001. Oczekiwana lista mutacji =5.4.0,001=0,02. Praktycznie w jednej populacji nie wystąpi ani jedna mutacja.

5. Inne przykłady algorytmów genetycznych

MAKSYMALIZACJA FUNKCJI

Funkcja f jest dodatnia w swojej dziedzinie; jeżeli tak nie jest (a funkcja f jest ograniczona od dołu), to dodajemy do f stałą c>0.

Niech f będzie funkcją k zmiennych, $f(x)=f(x_1,.....,x_k)$, a każda zmienna przyjmuje wartości z przedziału $D_i=[a_i,b_i]$ i $f(x)>0$. Funkcja f będzie optymalizowana z dokładnością p. Aby uzyskać taką dokładność każdy przedział D_i należy podzielić na $(b_i-a_i)\cdot 10^p$ równych podprzedziałów. Oznaczmy przez m(i) najmniejszą liczbę całkowitą taką, że $(b_i-a_i)\cdot 10^p<=2^{m(i)-1}$. Wtedy reprezentacja, w której każda zmienna jest zakodowana jako łańcuch binarny o długości m(i) będzie spełniała wymagania dokładności. Wartość tego łańcucha można wyrazić w postaci dziesiętnej za pomocą wzoru

$$x_i=a_i+\text{decimal}(\text{łańcuch}_2).(b_i-a_i)/(2^{m(i)}-1),$$

gdzie funkcja decimal(łańcuch_2) zwraca wartość dziesiętną łańcucha binarnego łańcuch2.

Np. jeżeli $D_i=[-3;12,1]$, p=4 to m(i)=18 gdyż $(b_i-a_i)\cdot 10^p= 15,1\cdot 10^4$ i $2^{17}<151000<2^{18}$. Łańcuchowi $010001001011010000_2)\ 15,1/(2^{18}.1) = -3+7035.15,1/262143=1,052426$

Każdy chromosom jest reprezentowany przez łańcuch binarny o długości m=m(1)+.....+m(k), gdzie pierwsze m(1) bitów odpowiada zmiennej x_1, kolejna grupa m(2) bitów odpowiada zmiennej x_2, itd.

Znając długość chromosomu generuje się populację składającą się z pop_size chromosomów. Dalej postępujemy zgodnie z podana wyżej procedurą. W każdym pokoleniu oceniamy stosując funkcję f wszystkie chromosomy, wybieramy nową populację i poddajemy ją operacjom krzyżowania i mutacji. Gdy nie obserwujemy poprawy rozwiazań – zatrzymujemy algorytm.

Proces selekcji chromosomów składa się z następujących kroków

1. obliczamy wartość dopasowania $eval(v_i)=f(v_i)$ dla każdego chromosomu v_i, i=1,....,pop_size,
2. obliczamy całkowite dopasowanie $F=\Sigma\{eval(v_i)|i=1, ..., pop\_size\}$,
3. obliczamy prawdopodobieństwo wyboru i-tego chromosomu: $P_i=eval(v_i)/F$,
4. obliczamy dystrybuantę $Q_i= \Sigma\{P_j|j=1, ...,i\}$,
5. utwórz nową populację:

```
for i=1 to pop_size
{
 wylosuj liczbę r
if (r<Q1) then wstaw chromosom vi else wybierz chromosom vi dla którego Qi-
1<r<=Qi
}
```

6. wykonaj krzyżowanie (z ustalonym prawdopodobieństwem P_c):

```
for i=1 to pop_size
{
 wylosuj liczbę r
 if (r<Pc) then wybierz rozpatrywany chromosom do krzyżowania
}
dobierz losowo wybrane chromosomy w pary
for i=1 to k  /*k oznacza liczbę par*/
{
```

wygeneruj losowo liczbę całkowitą pos∈[1,m-1]

pare chromosomów (α_1, .. α_{pos} α_{pos+1}... α_m) i (β_1... $\beta_{pos}\beta_{pos+1}$... β_m)

zamień na parę potomków (α_1, .. α_{pos} β_{pos+1}... β_m) i (β_1... $\beta_{pos}\alpha_{pos+1}$... α_m)

}

wykonaj mutację (z ustalonym prawdopodobieństwem p_m):

for i=1 to pop_size·m

{

wylosuj liczbę r

if (r<P_m) then zamień i-ty bit

}

6. Uwagi końcowe

Przedstawiony algorytm cechuje się : selekcją według reguły ruletki, krzyżowaniem prostym (z kojarzeniem losowym) i mutacją prostą. Własności tego algorytmu zależą od czterech parametrów: rozmiaru populacji (pop_size), prawdopodobieństwa krzyżowania (P_c), prawdopodobieństwa mutacji (P_m) oraz współczynnika wymiany (G). Współczynnik ten został wprowadzony , aby umożliwić mieszanie pokoleń i przybiera on wartości z przedziału [0;1]. W populacjach mieszanych (0<G<1) operacje genetyczne wykonuje się a pop_size.G osobnikach, a otrzymane potomstwo rozmieszcza się w istniejącej populacji na pop_size.G wybranych losowo (bez powtórzeń) miejscach. Jeżeli G=1, mamy do czynienia z populacjami rozłącznymi.

PROBLEM KOMIWOJAŻERA (TSP)

Definicja problemu:

Komiwojażer wyruszając z danego miasta musi odwiedzić dokładnie raz wszystkie miasta na swoim terytorium , powracając do punktu wyjściowego. Dla określonych kosztów podróży pomiędzy

każdym z miast problem TSP polega na optymalnym zaplanowaniu podróży komiwojażera ze względu na koszt całkowity. Dla N miast na terytorium pojedynczym rozwiązaniem TSP jest każda permutacja N elementowa. Liczność pełnej przestrzeni rozwiązań wynosi N!. TSP jest problemem łatwym do zdefiniowania, a trudnym do rozwiązania.

PRZYKŁADOWY ALGORYTM GENETYCZNY DLA TSP

(REPREZENTACJA ŚCIEŻKOWA)

Dla zadania TSP najbardziej naturalną reprezentacją jest reprezentacja ścieżkowa. W zadaniu w którym chcemy wyznaczyć szczególną permutację n miast {1,2.....,n}, chromosom $(i_1, i_2, ..., i_n)$ w reprezentacji ścieżkowej wyznacza drogę: $i_1 \rightarrow i_2 \rightarrow ... \rightarrow i_n$

W tak zapisanej reprezentacji określenie operatorów genetycznych nie jest już tak bezpośrednie. Zastosowanie standardowych operatorów mogłoby prowadzić do powstania chromosomów wyznaczających drogi pomijające pewne miasta z powtórzeniami innych.

Dla reprezentacji ścieżkowej w zadaniu TSP D. Goldberg i R.Lingle zaprezentowali operator krzyżowania PMX (partially maped crossover), dla dwóch chromosomów v_1 i v_2:

$v_1=(i_1,i_2,...,i_n)$

$v_2=(j_1, j_2,...j_n)$

wyznaczane są segmenty określone dwoma parametrami K oraz m:

$v_1=(i_1, i_2, ...i_k, i_{k+1}, ...i_{k+m}, ...i_n)$

$v_2=(j_1, j_2, ...j_k, i_{k+1}, ...j_{k+m}, ...j_m)$

w każdym chromosomie dokonywane są przekształcenia:

$i_{k+l} \rightarrow j_{k+l}$ dla l=0,.....,m

W wyniku tego otrzymujemy dwa chromosomy potomków:

$W_1=(i_{r1}, i_{r2}, ..., i_{rk-1}, j_k, j_{k+1}, ..., j_{k+m}, ..., i_{rn-m-1})$,

$W_2=(j_{p1}, j_{p2}, ..., j_{pk-1}, i_k, i_{k+m}, ..., j_{pn-m-1})$.

Dla chromosomu w_1 geny i_{rl} o wartościach różnych od $j_k, j_{k+1}, ..., j_{k+m}$ pokrywają się z genami i_l, natomiast wartości pozostałych genów wynikaja z przekształceń. Podobnie określone są geny j_{pl} chromosomu w_2.

Jest to oczywiście przykładowa propozycja reprezentacji danych i operatora krzyżowania

Określenie operatora mutacji jest podobne dla wszystkich reprezentacji danych i polega na:

- Wybraniu miasta i umieszczeniu go w losowym miejscu drogi wyznaczonej przez chromosom potomka;
- Wybraniu poddrogi i umieszczeniu jej w losowym fragmencie drogi wyznaczonej przez chromosom potomka;
- Zmianie kierunku wybranego fragmentu drogi wyznaczonej przez chromosom potomka.

Algorytmy genetyczne dla zadania TSP znacznie zwiększają swoją efektywność w wyniku zastosowania heurystycznych przekształceń wzmacniających poszczególne operatory genetyczne. Heurystyka może polegać na lokalnym zastosowaniu strategii poszukiwania, lokalnym minimalizowaniu długości dróg, powtarzaniu pewnych grup przekształceń dla minimalizacji określonych parametrów.

UKŁADANIE PLANU ZAJĘĆ

Rozwiązanie tego problemu można opisać przez

- listę nauczycieli $\{T_1, ..., T_m\}$
- listę odcinków czasowych (godzin) $\{H_1, ..., H_n\}$

– listę klas $\{C_1, ..., C_k\}$

Zadanie polega na wyznaczeniu optymalnego planu (nauczyciele-godziny-klasy). Funkcja celu powinna spełniać pewne wymagania . Obejmują one cele dydaktyczne (na przykład rozłożenie przedmiotów w tygodniu), cele personalne (pozostawienie wolnych popołudni dla nauczycieli pracujących na część etatu) i cele organizacyjne (na przykład każda godzina ma przyporządkowanego dodatkowego nauczyciela na zastępstwa).

Ograniczenia obejmują:

– zadaną liczbę godzin dla każdego nauczyciela i dla każdej klasy, dopuszczalny plan lekcji musi być "zgodny" z tymi liczbami,
– w każdej klasie i godzinie jest tylko jeden nauczyciel,
– nauczyciel nie może uczyć dwóch klas naraz,
– w każdej klasie z zaplanowanymi zajęciami w danej godzinie znajduje się nauczyciel

Najbardziej naturalną prezentacją dla chromosomu przedstawiającego potencjalne rozwiązanie zadania ustalania planu jest reprezentacja macierzowa: macierz $(R)_{ij}$ ($1<=i<=m$ oraz $1<=j<=n$), w której wiersz odpowiada nauczycielowi, a kolumna godzinie, elementami macierzy R są klasy ($r_{ij} \in \{C_1, ..., C_k\}$)1)

W algorytmie zastosowano następujące operatory:

Mutacja rzędu k: ten operator wybiera dwa sąsiednie ciągi k-elementowe z tego samego wiersza macierzy R i zmienia je.

Mutacja dni: ten operator jest specjalnym przypadkiem poprzedniego; wybiera on dwie grupy kolumn (godziny) w macierzy R, które odpowiadają różnym dniom i zamienia je.

Krzyżowanie: dla danych dwóch macierzy R_1 i R_2 operator ustawia wiersze pierwszej macierzy w porządku malejących watości tak zwa-

nej lokalnej funkcji dopasowania (część funkcji dopasowania związana tylko z cechami charakterystycznymi dla nauczyciela).

Najlepszych b wierszy (b jest parametrem ustalanym przez system na podstawie lokalnej funkcji dopasowania obu rodziców) wybiera się jako bloki budujące, a pozostałych m – b wierszy bierze się z macierzy R_2.

Literatura

[1] D.E. Goldberg (1995) "Algorytmy genetyczne I ich zastosowania", WNT Warszawa.

[2] Jerzy Cytowski (1996) "Algorytmy Genetyczne Podstawy I zastosowania", Akademicka Oficyna Wydawnicza PLJ Warszawa.

[3] Bolc, Cytowski (1991) "Metody przeszukiwania heurystycznego" t 1, 2 PWN Warszawa.

[4] Zbigniew Michalkiewicz (1996) "Algorytmy Genetyczne+struktury danych=programy ewolucyjne"WNT, Warszawa.

[5] Banachowski, Rytter/(1996) "Agorytmy i struktury danych" WNT, Warszawa.

[6] S.T. Wierzchoń (1998) "Techniki ewolucyjne. Algorytmy i zastosowania",Materiały seminaryjne Studium Doktoranckie IBS PAN.

[7] D. E. Goldberg, and Richardson, J., (1987) , "Genetic Algorithms with Sharing for Multimodal Function Optimization," Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, pp. 41-49.

[8] D. E. Goldberg, (1989), "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley.

[9] D. E. Goldberg, (1990), "A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing," in: Complex Systems, Vol. 4, Complex Systems Publications, Inc., pp. 445-460.

[10] D. E. Goldberg, (1991), "Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking," in: Complex Systems, Vol.5, Complex Systems Publications, Inc., pp. 139-167.

[11] D. E. Goldberg and Deb, K., (1991), "A Comparitive Analysis of Selection Schemes Used in Genetic Algorithms," in: Foundations of Genetic Algorithms, ed. by Rawlins, G.J.E., Morgan Kaufmann Publishers, San Mateo, CA, pp. 69-93.

[12] D. E. Goldberg, Deb, K., and Clark, J. H., (1992) , "Genetic Algorithms, Noise, and the Sizing of Populations," in: Complex Systems, Vol. 6, Complex Systems Pub., Inc., pp. 333-362.

[13] K.Krishnakumar, (1989) "Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization," SPIE: Intelligent Control and Adaptive Systems, Vol. 1196, Philadelphia, PA.

[14] G Syswerda,., (1989) "Uniform Crossover in Genetic Algorithms," in: Proceedings of the Third International Conference on Genetic Algorithms, Schaffer, J. (Ed.), Morgan Kaufmann Publishers, Los Altos, CA, pp. 2-9.

[15] G. Liepins and M. Hilliard and J. Richardson and M. Palmer (19XX) D. Brown and C. White, "Genetic Algorithms Applications to Set Covering and Traveling Salesman Problems",OR/AI: The Integration of Problem Solving Strategies, pages 29-57.

[16] H. Muhlenbein , (1991), "Parallel Genetic Algorithms and Combinatorial Optimization", SIAM Journal on Optimization,

[17] L. Davis (1991), "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York.

[18] Mitchell A. Potter (1997) The Design and Analysis of a Computational Model of Cooperative Coevolution. Ph.D. thesis, George Mason University, Fairfax, VA.

[19] Jerzy Bala, Kenneth A. De Jong, Jeffrey Huang, Haleh Vafaie, and H. Wechsler (1997). Using Learning to Facilitate the Evolution of Features for Recognizing Visual Concepts. In Special Issue of Evolutionary Computation - Evolution, Learning, and Instinct: 100 Years of the Baldwin Effect, pages 297-311. MIT Press.

[20] William M. Spears (1997). Recombination Parameters. In The Handbook of Evolutionary Computation, T. Baeck, D. Fogel and Z. Michalewicz (editors), IOP Publishing and Oxford University Press.

[21] K. Deb and and William M. Spears (1997). Speciation Methods. In The Handbook of Evolutionary Computation, T. Baeck, D. Fogel and Z. Michalewicz (editors), IOP Publishing and Oxford University

Press. This paper isn't available yet, although a portion entitled "Speciation Using Tag Bits" is currently available online.

[22] William M. Spears and Kenneth A. De Jong (1996) Analyzing GAs Using Markov Models with Semantically Ordered and Lumped States. To appear in Proceedings of FOGA96.

[23] Jayshree Sarma and Kenneth De Jong (1996) An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. To appear in Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN96), Sept. 22-26, 1996, Berlin, Germany.

[24] Bradley C. Wallet, David J. Marchette, Jeffery L. Solka, and Edward J. Wegman (1996) A Genetic Algorithm for Best Subset Selection in Linear Regression. To appear in Proceedings of the 28th Symposium on the Interface.

[25] Jerzy Bala, Kenneth A. De Jong, Jeffrey Huang, Haleh Vafaie, and H. Wechsler (1996) Visual Routine for Eye Detection Using Hybrid Genetic Architectures. To appear in Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, IEEE Computer Society Press.

[26] Haleh Vafaie and Kenneth A. De Jong (1995) Genetic Algorithms as a Tool for Restructuring Feature Space Representations. To appear in Proceedings of the International Conference on Tools with AI, Herndon, VA. IEEE Computer Society Press.

[27] Jerzy Bala, Kenneth A. De Jong, J. Haung, Haleh Vafaie, and H. Wechsler (1995) Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern classification. To appear in Proceedings of the 14th International Joint Conference on Artificial Intelligence, August 19-25, 1995, Montreal, Quebec, Canada.

Załącznik 1 -Inne przykłady zastosowań GA w zadaniach poszukiwania i optymalizacji

Rok	Badacz	Opis
Biologia		
1967	Rosenberg	Symulowana ewolucja populacji organizmów jednokomórkowych
1970	Weinberg	Projekt symulacji populacji komórek z zastosowaniem algorytmu genetycznego z metapoziomu
1984	Perry	Badania nad teorią nisz i specjacją prze użyciu algorytmów genetycznych.
1985	Grosso	Symulacja diploidalnego AG z jawnymi podpopulacjami i migracją
1987	Sannier i Goodman	Adaptacja mechanizmów reagowania na dostępność pokarmu w przestrzeni i czasie za pomocą AG
Informatyka		
1967	Bagley	Adaptacyjny algorytm gry w sześć pionków z zastosowaniem AG
1979	Raghavan i Birchard	Algorytm grupowania oparty na AG
1982	Gerady	Próba identyfikacji automatu probabilistycznego za pomocą AG

1984	Gordon	Adaptacyjna metoda opisu dokumentów przy użyciu AG
1985	Rendell	Zastosowanie AG do poszukiwania funkcji oceny strategii gry
1987	Raghavan i Agarwal	Adaptacyjna metoda grupowania dokumentów przy użyciu AG
Techniki i badania operacyjne		
1981	Goldberg	Identyfikacja modelu amortyzatora za pomocą elementarnego AG
1982	Etter, Hicks i Cho	Projektowanie filtru adaptacyjnego za pomocą elementarnego AG
1983	Goldberg	Optymalizacja pracy gazociągu w reżymach stacjonarnym i nieustalonym za pomocą AG
1985	Davis	Zastosowanie algorytmu genetycznego w zadaniach upakowania i kolorowania grafu
1985	Avis	Projekt zastosowania algorytmu genetycznego w zadaniach szeregowania typu job shop
1985	Davis i Smith	Projektowanie układów VLSI za pomocą AG
1985	Fourman	Kompaktyfikacja układów VLSI za pomocą GA
1985	Goldberg i Kuo	Optymalizacja rurociągu do przesyłu ropy w reżimie stacjonarnym za pomocą AG

1986	Glover	Projektowanie układu klawiatury za pomocą AG
1986	Goldberg i Santami	Optymalizacja konstrukcji (kratownica płaska) za pomocą AG
1986	Goldberg i Smith	Zastosowanie AG w ślepej wersji zagadnienia plecakowego
1986	Minga	Optymalizacja elementów podwozia samolotów za pomocą AG
1987	Davs i Coombs	Optymalizacja wielkości łączy w sieci łącznościowej za pomocą AG z operacjami zaawansowanymi
1987	Davis i Ritter	Układanie planu lekcji metodą symulowanego wyżarzania z AG na metapoziomie
Algorytmy genetyczne		
1962	Holland	Projekt samoadaptującego się komputera komórkowego
1968	Holland	Opracowanie teorii schematów
1971	Holstein	Optymalizacja funkcji dwóch zmiennych przy zastosowaniu reguł kojarzenia i selekcji
1972	Bosworth, Foo i Zegler	Operacje AG-podobne na genach rzeczywistych ze skomplikowanymi wariantami mutacji
1972	Frantz	Badania efektów nieliniowości pozycyjnej i operacyjnej inwersji

1973	Holland	Problem optymalnego wyboru w AG w zagadnieniach dwuramiennego bandyty
1973	Martin	Badania teoretyczne AG podobnych algorytmów probabilistycznych
1975	De Jong	Fundamentalne badania parametryczne elementarnego algorytmu genetycznego w środowisku pięciu funkcji testowych
1975	Holland	Publikacja ANAS
1977	Mecer	AG sterowany AG z metapoziomu
1981	Bethke	Zastosowanie funkcji Walsha do analizy wartości przystosowawczej schematów
1981	Brindle	Badania metod selekcji i dominowania w AG
1983	Pettit i Swigger	Ogólne badania możliwości zastosowania AG w niestacjonarnych zadaniach poszukiwania
1983	Wetzel	Zastosowanie AG w zagadnieniu komiwojażera
1984	Mauldin	Poszukiwanie metod heurystycznych zapewniających utrzymanie różnorodności populacji w elementarnym AG
1985	Baker	Eksperymenty z metodą selekcji przez nadawanie rang
1985	Booker	Koncepcje zgodności częściowej, podziału zasobów i barier reprodukcyjnych

1985	Goldberg i Lingle	Zastosowanie operacji PMX w zagadnieniu komiwojażera i analiza o-schematów
1985	Grefenstette i Fitzpatrick	Badania elementarnego AG w zastosowaniu do funkcji przybliżonych
1985	Schaffer	Optymalizacja wielokryterialna za pomocą Agz podpopulacjami
1985	Goldberd	Oszacowanie optymalnej wielkości populacji ze względu na liczbę efektywnie przetwarzanych schematów
1986	Grefenstette	AG stwrowany AG z metapoziomu
1987	Bridges i Goldberg	Pogłębiona analiza reprodukcji i krzyżowania w l-bitowym AG
1987	Goldberg	Minimalny problem zwodniczy MDP przy reprodukcji i krzyżowaniu
1987	Goldberg i Richardson	Formowanie się nisz i powstawanie gatunków jako efekt zastosowania funkcji współudziału
1987	Goldberg i Segrest	Analiza reprodukcji i mutacji za pomocą łańcuchów markowa
1987	Goldberg i Smith	Optymalizacja funkcji niestacjonarnych za pomocą diploidalnego AG
1987	Olivier, Smith i Holand	Symulacja i analiza permutacyjnych operacji rekombinacji

1987	Schaffer	Analiza wpływu procedur selekcji na propagację schematów
1987	Schaffer i Morishima	Adaptacyjny mechanizm krzyżowania
1987	Whitley	Zastosowanie selekcji na podstawie potomstwa AG
Techniki hybrydowe		
1985	Ackley	Doniesienie o algorytmie konekcyjnym o własnościach zbliżonych do AG
1985	Brady	Zastosowanie operacji quasi-genetycznych w zadaniu komiwojażera
1985	Grefenstette	Zastosowanie operacji genetycznych wspomaganych wiedzą w zagadnieniu komiwojażera
1987	Dolan i Dyer	Propozycja użycia AG do adaptacji topologii sieci konekcyjnej
1987	Grefenstette	Zastosowanie informacji specyficznej dla zadania w poszukiwaniu genetycznym
1987	Liepins, Hillard, Palmer, Morrow	Porównanie ślepych i zachłannych metod poszukiwania dla zagadnień kombinatorycznych
1987	Shaefer	Technika globalnie modyfikowanej reprezentacji adaptacyjnej (ARGOT)
1987	Sirag i Weisser	Sterowanie częstością operacji genetycznych w zagadnieniu komiwojażera za pomocą symulowanego wyżarzania

1987	Suh i Van Gucht	Operacje genetyczne wspomagane wiedzą w zagadnieniu komiwojażera
Przetwarzanie obrazów i rozpoznawanie wzorców		
1970	Cavicchio	Dobór detektorów w zadaniu rozpoznawania wzorców binarnych
1984	Fitzpatrick, Grefenstette	Obróbka obrazów za pomocą AG w celu minimalizacji różnic
1985	Englander	Dobór detektorów przy zadaniu klasyfikacji obrazów
1985	Gillies	Poszukiwanie detektorów cech obrazów za pomocą AG
1987	Stadnyk	Rozpoznawanie klas wzorców przy użyciu częściowego dopasowania
Współbieżne implementacje AG		
1976	Bethke	Wstępne rozważania teoretyczne na temat możliwych implementacji współbieżnych
1981	Grefenstette	Wstępne rozważania teoretyczne na temat możliwych implementacji współbieżnych
1987	Cohoon, Hegde, Martin, Richards	Symulowana implementacja współbieżna optymalnego rozmieszczenia liniowego
1987	Jog i Van Gucht	Algorytm genetczny wspomagany wiedzą z elementami współbieżności

1987	Petty, Leuze i Grefenstette	Implementacja współbieżna na sprzęcie firmy Intel
1987	Such i Van Gucht	Selekcja lokalna we współbieżnym Agdla zagadnienia komiwojażera
1987	Tense	Implementacja współbieżna AG w architekturze HyperCube
Nauki fizyczne		
1985	Shaefer	Zastosowanie AG do rozwiązywania równań nieliniowych związanych z poeierzchniami potencjału
Nauki społeczne		
1979	Reynolds	Zastosowanie metod adaptacyjnych typu AG do modelowania zachowań prehistorycznych myśliwych-zbieraczy
1981	Smith i De Jong	Kalibracja modelu migracji populacyjnych za pomocą AG
1985	Axelrod	Symulacja ewolucji norm zachowania za pomocą AG
1985	Axelrod	Zastosowanie AG do poszukiwania rozwiązań iterowanego dylematu więźnia

WYŻSZA SZKOŁA INFORMATYKI STOSOWANEJ I ZARZĄDZANIA

działa pod auspicjami
Polskiej Akademii Nauk

ZAŁOŻYCIELEM
Wyższej Szkoły Informatyki Stosowanej i Zarządzania
jest
FUNDACJA KRZEWIENIA NAUK SYSTEMOWYCH
powołana z inicjatywy
Prezesa
POLSKIEJ AKADEMII NAUK

FUNDATOREM
Fundacji Krzewienia Nauk Systemowych
jest
POLSKA AKADEMIA NAUK

ORGANEM
sprawującym nadzór
jest
MINISTERSTWO EDUKACJI NARODOWEJ

Wyższa Szkoła Informatyki Stosowanej i Zarządzania
prowadzi studia wyższe na kierunkach:
INFORMATYKA
ZARZĄDZANIE I MARKETING

SIEDZIBA
Instytut Badań Systemowych
Polskiej Akademii Nauk

ISBN 83-85847-24-3