ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Konrad WALA, Wojciech CHMIEL
Akademia Górniczo-Hutnicza

OPERATORY GENETYCZNE DLA PERMUTACYJNYCH ZAGADNIEŃ OPTYMALIZACJI

Streszczenie: W pracy przedstawiono wyniki badań eksperymentalnych procesu optymalizacji, realizowanego za pomocą algorytmu genetycznego dla kwadratowego problemu przydziału. Celem badań jest określenie wpływu różnych operatorów genetycznych na efektywność procesu optymalizacji.

GENETIC OPERATORS FOR OPTIMIZATION OF PERMUTATION PROBLEMS

Summary: This paper presents results of experimental examination of optimization process realized with the aid of genetic algorithm on an example of quadratic assigment problem. The purpose of these investigations is determination of the genetic operators influence on the optimization process efficiency.

GENETISCHE OPERATOREN FÜR DIE PERMUTATIONSPROBLEME DER OPTIMIERUNG

Zusammenfassung: In dieser Forschungsarbeit werden die Ergebnisse der Experimentaluntersuchungen des Optimierungsvorganges vorgestellt. Dieses Vorgang wird mit der Hilfe eines genetischen Algoritmus realisiert und auf den Beispiel des Quadratproblems der Zuteilung präsentiert. Das Ziel der Untersuchungen ist die Bestimmung des Einflusses der verschiedenen genetischen Operatoren auf die Effektivität des Optimierungsvorganges.

1.Wprowadzenie

Istnieje znaczna liczba *NP-trudnych* zagadnień optymalizacji kombinatorycznej o dużym znaczeniu praktycznym, które można sformalizować w postaci zagadnień permutacyjnych, tj. zagadnień, gdzie rozwiązanie ma postać permutacji. Należy tu cała klasa zagadnień harmonogramowania oraz zagadnienia przydziału zadań do zasobu, jak też

standardowe zagadnienia optymalizacji kombinatorycznej, zagadnienie komiwojażera. Prace z zakresu poprawiania efektywności przybliżonych algorytmów rozwiązywania *NP-trudnych* zagadnień permutacyjnych dużego rozmiaru należą do aktualnych zadań badawczych. W pracy przedstawiono wyniki badań komputerowych algorytmu przybliżonego dla zagadnień permutacyjnych, w którym do poprawy efektywności procesu szukania rozwiązań zastosowano mechanizm uczenia się maszyn. Bazą mechanizmu uczenia się jest metoda genetyczna.

Szczególnie trudnym zagadnieniem permutacyjnym jest problem przydziału o kwadratowym wskaźniku jakości, zwykle oznaczanym symbolem *QAP* *(ang. Quadratic Assigment Problem)*. Rozwiązanie zagadnienia *QAP* o rozmiarze $n > 15$ napotyka na znaczne trudności obliczeniowe, co jest powodem licznych prac w zakresie rozwoju metod przybliżonych dla tego zagadnienia (por. [1], [2], [3], [4], [8], [9]).

Zagadnienie *QAP* sformalizujemy w następujący sposób. Dany jest zbiór $N=\{1,...,n\}$ i dwie $(n\times n)$ wymiarowe macierze $A=[a_{i,k}]$, $B=[b_{j,l}]$, należy znaleźć permutację $\Pi=(\Pi(1),...,\Pi(n))$ elementów zbioru N, która minimalizuje funkcje celu $f(\Pi)$ o postaci:

$$f(\Pi) = \min_{\Pi\in\Psi} f(\Pi) = \sum_{i=1}^{n}\sum_{k=1}^{n} a_{i,k}\, b_{\Pi(i),\Pi(k)}$$

W terminologii alokacji obiektów zbiór N jest zbiorem numerów obiektów, a $\Pi(i)\in N$, $i=1,...,n$, określa numer obiektu przydzielonego do pozycji i. Macież A jest wtedy macierzą odległości pomiędzy pozycjami rozmieszczenia obiektów, podczas gdy macierz B opisuje powiązania występujące pomiędzy obiektami. Natomiast funkcja celu $f(\Pi)$, $\Pi\in\Psi$, gdzie Ψ jest zbiorem n-elementowych dopuszczalnych permutacji, określa koszt globalny eksploatacji systemu.

2. Ogólna struktura algorytmu genetycznego

Ogólnie, algorytm genetyczny (por. monografie [5], [6], [7]) jest iteracyjną procedurą poprawiania rozwiązań zawierającą zbiór *POPULACJA* najlepszych znanych rozwiązań zagadnienia oraz zbiór pseudogenetycznych operatorów nazywanych krótko operatorami genetycznymi. Na początek pracy algorytmu rozwiązania ze zbioru *POPULACJA* są wyznaczane na drodze losowania. Przyjęto, że zbiór *POPULACJA* ma stały rozmiar, tj. liczba M rozwiązań w zbiorze *POPULACJA* nie zmienia się podczas procesu optymalizacji. Eksploatowany przez nas, pod nazwą algorytmu genetycznego *GA*, proces optymalizacji można opisać za pomocą następujących kroków (por. [9]).

ALGORYTM *GA*.

Krok 1. Wyznaczenie populacji początkowej.

Wygeneruj, w sposób losowy, M permutacji n-elementowych i umieść je w zbiorze *POPULACJA*.

Krok 2. Wybór operatora genetycznego.

Wylosuj numer operatora genetycznego (każdy z siedmiu operatorów jest losowany z prawdopodobieństwem p_r, $r=1, \ldots, 7$; przy czym $p_1, p_2, p_3, p_4, p_5, p_6, p_7 \geq 0$).

Krok 3. Wybór rodzica/rodziców.

Dla wybranego operatora wylosuj, zgodnie z rozkładem równomiernym, ze zbioru *POPULACJA*, zależnie od typu operatora jedno lub dwa rozwiązania, które nazywamy rodzicami.

Krok 4. Generowanie potomków.

Za pomocą wylosowanego operatora genetycznego dokonaj modyfikacji permutacji-rodzica/permutacji-rodziców i wyznacz w ten sposób permutacje-potomka/ permutacje-potomków.

Krok 5. Poprawa rozwiązań w zbiorze *POPULACJA*.

Dla każdego potomka oblicz wartość funkcji celu $f(\Pi_{pot})$ potomka Π_{pot} i jeśli wartość ta jest mniejsza od najgorszego rozwiązania Π_{max} populacji (tj. $f(\Pi_{pot})<f(\Pi_{max})$, gdzie $f(\Pi_{max})=max\{ f(\Pi): \Pi \in POPULACJA \}$), to umieść potomka Π_{pot} w zbiorze *POPULACJA* usuwając zarazem z tego zbioru najgorsze rozwiązanie Π_{max}.

Krok 6. STOP.

Jeśli wygenerowano zadaną liczbę L potomków to STOP, wydrukuj najlepsze rozwiązanie zbioru *POPULACJA*. W przeciwnym wypadku idź do kroku 2.

Parametrami algorytmu są wielkości $M, L, p_1, p_2, p_3, p_4, p_5, p_6, p_7$.

3.Operatory genetyczne

Podkreślamy, że w przypadku zagadnień permutacyjnych rozwiązania są ciągami mającymi określoną strukturę, a mianowicie są n-elementowymi permutacjami, tj. każdy element zbioru $N=\{1,\ldots,n\}$ występuje w rozwiązaniu $\Pi=(\Pi(1),\ldots,\Pi(j),\ldots,\Pi(n))$ dokładnie jeden raz. Tak więc zbiór *POPULACJA* jest zbiorem permutacji oraz potomkowie, wyznaczani przez operatory genetyczne, także muszą być permutacjami. Możliwe są tu dwa rozwiązania.

Do tego celu można użyć pewnego rodzaju "algorytmu naprawy" rozwiązań, który zastosowany do wygenerowanego przez operator potomka przywraca go do właściwej przestrzeni szukania, tj.do zbioru Ψ permutacji n-elementowych. Często zamiast używać

algorytmu naprawy potomka wprowadza się wiedzę o problemie do operatora i w ten sposób operator "inteligentnie" może unikać konstrukcji rozwiązań niedopuszczalnych.

Stosunkowo łatwo można sobie poradzić z operatorami unarnymi (tj. operatorami jednoargumentowymi), które dokonują modyfikacji elementów pojedynczych ciągów. Zauważmy, że w takim przypadku, operator mutacji zmieniający wartość tylko jednego elementu permutacji jest niedopuszczalny. Można natomiast zmienić dwa lub więcej elementów permutacji, ale tak, ażeby dalej spełnione było podstawowe ograniczenie: każdy element zbioru N występuje w rozwiązaniu dokładnie jeden raz. W konsekwencji takie modyfikacje sprowadzają się do czynności przestawienia elementów permutacji, co jest zwykle realizowane przez operator inwersji.

Jednakże, w przypadku stosowania tylko operatorów unarnych, jest mała nadzieja na znalezienie nawet dobrej permutacji (nie mówiąc już o najlepszej). Siła algorytmów genetycznych wynika z wymiany informacji strukturalnej, w przypadku rozwiązań-permutacji, między "dobrymi" rozwiązaniami. Tak więc potrzebne są operatory krzyżowania (operatory binarne), które eksploatują ważne podobieństwa pomiędzy rozwiązaniami-rodzicami i generują rozwiązania-potomków w postaci permutacji. Poniżej są opisane unarne i binarne operatory genetyczne, których przydatność do rozwiązywania zagadnień permutacyjnych sprawdziliśmy w "naszym" algorytmie genetycznym.

1. Operator inwersji podciągu

Operator inwersji 1 tworzy, na podstawie jednego rozwiązania wybranego ze zbioru *POPULACJA*, jednego potomka w następujący sposób:

a. wybiera dwie liczby naturalne i,j takie, że $1 \le i < j \le n$,

b. zmienia miejscami elementy permutacji znajdujące się pomiędzy pozycjami i oraz j w ten sposób, że element z pozycji i jest zamieniany z elementem pozycji j, $(i+1)$ z $(j-1)$, itd.

Jeżeli przodek jest rozwiązaniem o postaci:

$\Pi=(\Pi(1),\Pi(2)...,\Pi(i),\Pi(i+1),...,\Pi(j-1),\Pi(j),\Pi(j+1),...,\Pi(n))$, to potomek ma postać:

$\Pi=(\Pi(1),\Pi(2)...,\Pi(j),\Pi(j-1),...,\Pi(i+1),\Pi(i),\Pi(j+1),...,\Pi(n))$.

2. Dwuelementowy operator inwersji

Operator inwersji 2 tworzy, na podstawie jednego rozwiązania (rodzica) wybranego ze zbioru *POPULACJA*, jednego potomka w następujący sposób:

a. wybiera dwie liczby naturalne i,j takie że $1 \le i < j \le n$,

b. zmienia miejscami dwa elementy permutacji znajdujące się na pozycjach i oraz j.

3. Jednopunktowy operator krzyżowania

Operator krzyżowania 3 tworzy, na podstawie dwóch rozwiązań (pary rodziców)

$\Pi'=(\Pi'(1),\Pi'(2),...,\Pi'(n))$, $\Pi''=(\Pi''(1),\Pi''(2),...,\Pi''(n))$, dwóch-czterech potomków w następujący sposób:

a. wybierz liczbę naturalną i taką, że $1 \leq i \leq n$,

b. utwórz dwa rozwiązania wstępne $\Pi^1=(\Pi'(1),\Pi'(2),...,\Pi'(i),\Pi''(i+1),...,\Pi''(n))$,
$\Pi^2=(\Pi''(1),\Pi''(2),...,\Pi''(i),\Pi'(i+1),...,\Pi'(n))$

c. Każde rozwiązanie wstępne popraw za pomocą procedury *ALFA*.

Procedura *ALFA* zaproponowana w [9] jest algorytmem poprawy rozwiązań wstępnych. Jeżeli rozwiązanie wstępne jest n-elementową permutacją, to procedura *ALFA* nie zmienia jego postaci, i w tym przypadku rozwiązanie wstępne jest bezpośrednio potomkiem. W przeciwnym wypadku procedura *ALFA* poprawia rozwiązanie wstępne do permutacji n-elementowej generując na podstawie jednego rozwiązania wstępnego dwie permutacje-dwóch potomków.

PROCEDURA *ALFA*

a. Jeśli rozwiązanie wstępne $\Pi^j (j=1,2)$ jest permutacją n-elementową, to rozwiązanie Π^j jest potomkiem → *POWRÓT*.

b. Sprawdzaj elementy ciągu Π^j od lewej strony do prawej. Pierwszy taki element, który występuje dwukrotnie zastąp przez element α, gdzie α jest najmniejszym elementem zbioru $N=\{1,...,n\}$, który nie występuje w rozwiązaniu częściowym Π^j. Otrzymanego w ten sposób potomka oznacz symbolem Π^{j+2}.

c. Sprawdzaj elementy ciągu Π^j od prawej strony do lewej. Pierwszy taki element, który w ciągu występuje dwukrotnie, zastąp przez element α. Otrzymanego w ten sposób potomka oznacz symbolem Π^j → *POWRÓT*.

W monografii [8] opisano operatory krzyżowania *PMX* (ang. partially matched crossover) oraz *OX* (ang. order crossover). Są to realizacje "inteligentnych" operatorów krzyżowania, które generują potomków bezpośrednio w postaci permutacji. Działanie tych operatorów zostanie przedstawione za pomocą przykładów. W każdym z operatorów *PMX* i *OX*, dla wybranej ze zbioru *POPULACJA* pary rodziców, ustalane są na drodze losowania (rozkład równomierny) dwa punkty rozcięcia, które wyznaczają sekcje kojarzenia, tj. odcinki permutacji podlegające wymianie; w demonstrowanych poniżej przykładach sekcje kojarzenia znajdują się między dwoma pionowymi kreskami.

4. Dwupunktowy operator krzyżowania PMX

W operatorze *PMX* sekcja kojarzenia jest wykorzystywana do wykonywania krzyżowania za pomocą operacji wymiany typu pozycja-za-pozycją. Rozważmy parę rodziców:

$\Pi' = (1,2\,|3,4,5,6|7,8,9,10)$

$\Pi'' = (4,1|2,3,7,8|5,9,10,6)$

Najpierw odwzorowywana jest permutacja Π'' do permutacji Π'; zmieniają miejsce następujące wartości permutacji: 2 i 3, 3 i 4, 7 i 5, 8 i 6. Podobnie odwzorowanie Π' do Π'' polega na zmianie wartości 3 i 2, 4 i 3, 5 i 7, 6 i 8. W rezultacie otrzymujemy dwóch potomków Π^1 i Π^2:

$\Pi^1 = (1,4 \mid 2,3,7,8 \mid 5,6,9,10)$

$\Pi^2 = (2,1 \mid 3,4,5,6 \mid 7,9,10,8)$

5. Dwupunktowy operator krzyżowania OX

Zamiast stosowania operacji wymiany pozycja-za-pozycją, używaną w operatorze *PMX* do realizacji odwzorowania, operator *OX* stosuje przesunięcia w celu zapełnienia "dziur" (oznaczonych w przykładzie symbolem "-") powstałych podczas przenoszenia elementów odwzorowywanych pozycji permutacji. Rozważmy parę rodziców:

$\Pi' = (1,2 \mid 3,4,5,6 \mid 7,8,9,10)$

$\Pi'' = (4,1 \mid 2,3,7,8 \mid 5,9,10,6)$

Na przykład, kiedy permutacja Π'' jest odwzorowywana do permutacji Π', wartości 3,4,5,6 tworzą dziury w permutacji Π'':

$\Pi'' = (-,1 \mid 2,-,7,8 \mid -,9,10,-)$

Dziury, leżące poza sekcją kojarzenia, są zapełniane za pomocą operacji przesunięcia, która ustala pozycje permutacji potrzebne do krzyżowania:

$\Pi'' = (2,7 \mid -,-,-,- \mid 8,9,10,1)$

Następnie dziury są zapełniane wartościami z sekcji kojarzenia partnera (tj. permutacji Π'). Wykonując tę operację oraz uzupełniające krzyżowanie otrzymujemy potomków Π^1 i Π^2:

$\Pi^1 = (4,5 \mid 2,3,7,8 \mid 6,9,10,1)$

$\Pi^2 = (2,7 \mid 3,4,5,6 \mid 8,9,10,1)$

6. Dwupunktowy operator krzyżowania GX

Opracowaliśmy także dwupunktowy operator krzyżowania *GX* (ang. greedy crossover), w którym dziury powstałe podczas przenoszenia elementów odwzorowywanych pozycji permutacji są zapełniane za pomocą reguły zachłannej badanego zagadnienia; w przeprowadzonych eksperymentach reguły dla zagadnienia *QAP*. Rozważmy parę rodziców:

$\Pi' = (1,2 \mid 3,4,5,6 \mid 7,8,9,10)$

$\Pi'' = (4,1 \mid 2,3,7,8 \mid 5,9,10,6)$

Opiszemy teraz proces konstrukcji potomka Π^2 pary rodziców Π' i Π''; podobnie wyznaczamy potomka Π^1. Kiedy permutacja Π'' jest odwzorowywana do permutacji Π', wartości *3,4,5,6* sekcji kojarzenia permutacji Π' tworzą dziury w permutacji Π'':

$\Pi'' = (-,1 \mid 2,-,7,8 \mid -,9,10,-)$

Zbiór wartości permutacji Π'' leżących w sekcji kojarzenia tak zmodyfikowanej permutacji Π'' oznaczamy symbolem W; w prezentowanym przykładzie $W = \{2,7,8\}$. Następnie zastępujemy sekcję kojarzenia permutacji Π'' sekcją permutacji Π' konstruując w ten sposób potomka Π^2:

$\Pi^2=(-,1 \mid 3,4,5,6 \mid -,9,10,-)$

Zbiór numerów pozycji potomka Π^2 leżących poza sekcją kojarzenia, w których występują dziury, oznaczamy symbolem S; w prezentowanym przykładzie $S = \{1,7,10\}$.

Dziury występujące na pozycjach zbioru S zapełniamy wartościami ze zbioru W za pomocą następującej reguły zachłannej.

Niech $\gamma(i)$ jest numerem pozycji permutacji, która jest maksymalnie "odległa" od pozycji i: $a_{i,\gamma(i)} = max\{a_{ij}: i \neq j\}$. Reguła zachłanna ustala dla pozycji $i \in S$ taką wartość $\Pi(i)$, która minimalizuje składnik: $a_{i,\gamma(i)} b_{\Pi(i),\Pi(\gamma(i))}$ funkcji celu. W tym celu, w każdej iteracji zapełniania dziur w Π^2, szukamy takiego $i \in S$, ażeby $a_{i,\gamma(i)} = max\{a_{k,\gamma(k)}: k \in S\}$, a następnie wartość permutacji $\Pi(i)$ (ewentualnie i $\Pi(\gamma(i))$) wyznaczmy minimalizując czynnik $b_{..}$:

$$b_{\Pi(i),\Pi(\gamma(i))} = \begin{cases} min\{b_{j,\Pi(\gamma(i))} : j \in W\} & \text{dla } \gamma(i) \notin S \\ min\{b_{j,k} : j,k \in W; j \neq k\} & \text{dla } \gamma(i) \in S \end{cases}$$

Zauważmy, że w przypadku $\gamma(i) \in S$ oprócz wartości $\Pi(i)$ dla pozycji i permutacji Π^2 wyznaczamy wartość $\Pi(\gamma(i))$ dla pozycji $\gamma(i) \in S$.

Zbiór rozwiązań ewoluującej populacji testuje jednocześnie wiele obszarów zbioru rozwiązań Ψ, przy czym operatory krzyżowania umożliwiają próbkowanie nowych, i to najbardziej "obiecujących", obszarów przestrzeni szukania (por.[6]). W celu szczegółowego zbadania "obiecujących" obszarów wprowadziliśmy operator optymalizacji lokalnej, który dokładnie sprawdza niewielkie obszary zbioru rozwiązań.

7. *Operator optymalizacji lokalnej*

Niech otoczenie $S(\Pi)$ rozwiązania $\Pi=(\Pi(1),...,\Pi(i-1),\Pi(i),...,\Pi(j-1),\Pi(j),...,\Pi(n))$ jest określone za pomocą formuły:

$$S(\Pi)=\{\Pi': \Pi'=(\Pi(1),...,\Pi(i-1),\Pi(j),...,\Pi(j-1),\Pi(i),...,\Pi(n))\}.$$

Operator optymalizacji lokalnej wyznacza jednego potomka Π^1 na podstawie jednego rodzica Π, wybranego ze zbioru *POPULACJA*, w następujący sposób:

a. Określ otoczenie $S(\Pi)$,

b. Znajdź rozwiązanie $\Pi' = arg\ min\{f(\hat{\Pi}): \hat{\Pi} \in S(\Pi); \hat{\Pi} \neq \Pi\}$,

c. Jeżeli $f(\Pi') < f(\Pi)$ to podstaw $\Pi:=\Pi'$ i przejdź do instrukcji **(a)** w przeciwnym przypadku przejdź do instrukcji **(d)**,

d. Podstaw $\Pi^1:=\Pi$.

4. Eksperymenty obliczeniowe

Opisany powyżej algorytm genetyczny (łącznie z operatorami genetycznymi), dla zagadnienia *QAP*, został zakodowany w języku C na komputery zgodne z IBM PC pracujące pod systemem operacyjnym MS DOS. Przedstawimy wybrane wyniki procesu optymalizacji dla testu Nugenta [8] rozmiaru n=30. Przyjęto, że liczba iteracji algorytmu genetycznego jest równa liczbie wygenerowanych potomków. Rozwiązania początkowe (startowe) były generowane w sposób losowy przy równomiernym rozkładzie, przy czym każdy eksperyment rozpoczynał się od tych samych rozwiązań startowych - tej samej wygenerowanej losowo populacji startowej. Rozmiar populacji we wszystkich zaprezentowanych eksperymentach był taki sam, $M=30$. Funkcja celu najlepszego znanego rozwiązania zadania testowego wynosi *6124*.

W tablicy *1* zaprezentowano wyniki obliczeń dla następujących parametrów algorytmu genetycznego: liczba iteracji $L=10000$, $p_1=p_2=0.1$, $p_7=0.0$, natomiast prawdopodobieństwa losowania trzeciego operatora - p_3 - znajdują się w pierwszej kolumnie tablicy *1*. W kolejnych seriach eksperymentów obliczeniowych bierze udział tylko jeden z operatorów *GX, PMX, OX;* w kolumnie drugiej tablicy *1* są umieszczone prawdopodobieństwa losowania operatora p_i, $i \in P=\{X, PMX, OX\}$, przy czym w każdym eksperymencie były spełnione warunki: $p_3+p_i=0.8$, $p_j=0.0$ dla $j \in P$ oraz $j \neq i$. W następnych kolumnach podane są wartości funkcji celu najlepszych, po wykonaniu L iteracji, rozwiązań znajdujących się w zbiorze *POPULACJA*, gdzie symbolem f_i oznaczono wartości funkcji celu tej serii eksperymentów, w której badano efektywność operatora $i \in P$.

Tablica *1*

p_3	p_i	f_{PMX}	f_{OX}	f_{GX}
0.8	0.0	6620	6620	6620
0.7	0.1	6638	6506	6508
0.6	0.2	6584	6644	6494
0.5	0.3	6468	6658	6392
0.4	0.4	6416	6600	6572
0.3	0.5	6536	6610	6570
0.2	0.6	6554	6680	6440
0.1	0.7	6478	6462	6436
0.0	0.8	7242	6756	6616

W tablicy *2* przedstawiono wyniki obliczeń dla następujących parametrów algorytmu: liczby iteracji eksperymentów wynoszą $L1=10000$ i $L2=100000$, prawdopodobieństwa p_1, p_2, p_3 są podane, odpowiednio, w pierwszej, drugiej i trzeciej kolumnie tablicy *2*, natomiast operator optymalizacji lokalnej (operator *7*) jest nieaktywny we wszystkich seriach eksperymentów $(p_7=0.0)$. Podobnie jak w tablicy *1*, w kolumnie czwartej tablicy *2* są umieszczone wartości prawdopodobieństwa p_i, $i \in P$. W każdej serii eksperymentów jest aktywny tylko jeden z operatorów zbioru P, tj. jeżeli badany jest operator $i \in P$ $(p_i \geq 0)$, to wtedy dla operatorów $j \neq i$, $j \in P$: $p_j=0.0$. W następnych kolumnach podane są wartości funkcji celu najlepszych, po wykonaniu $L1$ lub $L2$ iteracji, rozwiązań znajdujących się w zbiorze *POPULACJA*, gdzie symbolem f_i oznaczono wartości funkcji celu tej serii eksperymentów, w której badano efektywność operatora $i \in P$.

Tablica *2*

p_1	p_2	p_3	p_i	f_{PMX}		f_{OX}		f_{CX}	
				L1	*L2*	*L1*	*L2*	*L1*	*L2*
0.1	0.0	0.2	0.7	6888	6708	6712	6544	6722	6682
0.1	0.0	0.3	0.6	6650	6606	6800	6522	6824	6652
0.2	0.0	0.5	0.3	6614	6528	6714	6526	6808	6502
0.1	0.0	0.1	0.8	6916	6726	6734	6632	6902	6470
0.1	0.2	0.3	0.4	6518	6272	6420	6272	6484	6252
0.1	0.2	0.2	0.5	6506	6336	6430	6250	6400	6278

Tablica *3* ilustruje wpływ operatora optymalizacji lokalnej $(p_7 \geq 0.0)$ na efektywność procesu optymalizacji. Liczba iteracji dla wszystkich eksperymentów jest taka sama i wynosi $L=10000$. Prawdopodobieństwa p_1, p_2, p_3 są podane, odpowiednio, w pierwszej, drugiej i trzeciej kolumnie tablicy. Podobnie jak w eksperymentach przedstawionych w poprzednich tablicach, w każdej serii eksperymentów jest aktywny tylko jeden z operatorów zbioru P. Jeżeli badany jest operato\r $i \in P$ $(p_i \geq 0.0)$ to pozostałe operatory $j \in P$, $j \neq i$, są nieaktywne (tj. $p_j=0.0$). W kolumnie czwartej tablicy *3* podane są łączne prawdopodobieństwa losowania $p=p_i+p_7$ operatora i oraz operatora optymalizacji lokalnej; w każdej serii eksperymentów prawdopodobieństwo losowania operatora i równa się $p_i=p-p_7$. Symbolem f_i oznaczono wartości funkcji celu najlepszych rozwiazań uzyskiwanych w seriach eksperymentów, w których badano efektywność operatora $i \in P$.

Tablica 3

p_1	p_2	p_3	p	f_{PMX}		f_{OX}		f_{GX}	
				$p_7=0.0$	$p_7=0.1$	$p_7=0.0$	$p_7=0.1$	$p_7=0.0$	$p_7=0.1$
0.1	0.0	0.2	0.7	6888	6220	6712	6224	6722	6194
0.1	0.0	0.3	0.6	6650	6246	6800	6170	6824	6246
0.2	0.0	0.5	0.3	6614	6182	6714	6188	6808	6220
0.1	0.0	0.1	0.8	6916	6216	6734	6156	6902	6212
0.1	0.2	0.3	0.4	6518	6162	6420	6190	6484	6196
0.1	0.2	0.2	0.5	6506	6224	6430	6176	6400	6170

Z przeprowadzonych eksperymentów, które częściowo są przedstawione w tablicach *1, 2, 3,* wynikają następujące wnioski:

- W algorytmie genetycznym wystarczy zaimplementować jeden z omówionych operatorów krzyżowania, przy czym wybór operatora nie ma większego znaczenia dla procesu optymalizacji.
- •• Znaczne wydłużenie procesu optymalizacji przynosi nieznaczną poprawę funkcji celu.
- ••• Istotne znaczenie dla poprawy efektywności procesu optymalizacji ma zastosowanie operatora optymalizacji lokalnej, który szczegółowo sprawdza niewielkie "obiecujące" obszary rozwiązań.

LITERATURA

[1] Brown D.E., Huntley Ch.L., Spollane A.R.: A parallel genetic heuristic for the quadratic assigment problem. Proc. of the Third Int. Conference on Genetic Algorithms, Georg Manson Univ., 1989, pp.406-415.

[2] Brukard R.E., Stratmann K.H.: Numerical investigation on quadratic assigment problems. Naval Research Logistics Quaterly, vol.25, 1978, pp.129-147.

[3] Filipowicz B., Wala K.: Algorytmy optymalizacji kwadratowego zagadnienia przydziału. Kwartalnik Elektrotechnki, z.1, 1992, Wydawnictwo AGH w Krakowie.

[4] Finke G., Burkard R.E, Rendl F.: Quadratic assigments problems. Annals of Discrete Mathematics, vol.31, North-Holland, 1987, pp.61 -82.

[5] Goldberg D.E.:Genetic algorithms in search, optimization and machine learning. Addison-Wesley Pub. Company, Inc.,N.Y.,1989.

[6] Holland J.:Adaptation in natural and artificial systems. Univ. of Michigan Press, Ann Arbor, MI, 1975.

[7] Michalewicz Z.:Genetic algorithms + data structures = evolution programs. Springer-Verlag, Berlin, 1992.

[8] Nugent C.E, Vollman T.E., Ruml J.: An experimental comparison of techniques for the assigment of facilities to locations. Operation Research, vol.16, 1968, pp.150-173.

[9] Wala K., Chmiel W.:Local search and genetic heuristics for the permutational optimization problems. Materiały 7. Sympozjum Międzynarodowego "System-Modelling-Control", t.2, Zakopane, maj 1993, pp. 252-257.

Recenzent: Prof. dr inż. Henryk Kowalowski

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

There are a number of combinatorial optimization problems which can be formalized as permutational optimization problems, for instance the traveling salesman problem (*TSP*), quadratic assignment problem (*QAP*), and so forth. Usually these ones are *NP-hard* problems.

The paper presents the results of a computer (numerical) investigation of the optimization process realized with the aid of genetic algorithm on the example of the quadratic assignment problem. The investigation has been realized by means of experimental software package which is specially prepared for IBM PC compatibile computers. The purpose of these investigations is determination of the genetic operators influence on the optimization process efficiency. The algorithm exploits seven genetic operators where four of them are crossover operators, two are inversion operators and one is a local search operator. We present a collection of numerical results for Nugent's ($n=30$) test task which give the characteristic of subjection of the optimization process to genetic operators. One can find out a notable influence of the local search operator on the optimization process efficiency.