ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: AUTOMATYKA z. 131 Nr kol.1476

Wojciech CHMIEL, Piotr KADŁUCZKA
Akademia Górniczo-Hutnicza w Krakowie

ZASTOSOWANIE TRANSFORMACJI PERMUTACJI DO PRZESTRZENI WEKTOROWEJ W ALGORYTMACH EWOLUCYJNYCH*

Streszczenie. W pracy przedstawiono wyniki badań eksperymentalnych procesu optymalizacji realizowanego za pomocą algorytmu ewolucyjnego dla testowego zagadnienia przydziału z kwadratową funkcją celu. Opierając się na zagadnieniu porównano algorytm ewolucyjny korzystający z metod transformujących zagadnienie permutacyjne do postaci kombinacji z działającymi wyłącznie na permutacjach.

TRANSFORMATION PERMUTATION TO VECTOR SPACE IN EVOLUTION ALGORITHM PROBLEMS

Summary. This paper presents results of experimental examination of optimization process realized with the aid of genetic algorithm on the test example of quadratic assignment problem. The investigated genetic algorithm realizes original genetic search process. In this algorithm we introduce technique normally used to optimize objective functions of any kind in Cartesian space where derivatives are not available or impossible to determine.

1. Wstęp

W dziedzinie optymalizacji kombinatorycznej zagadnienia, dla których rozwiązania można sformułować w postaci permutacji, stanowią ważną klasę problemów decyzyjnych.

Klasycznymi typami problemów permutacyjnych są: kwadratowy problem przypisania, kolorowania grafu, zagadnienia harmonogramowania, zagadnienia komiwojażera oraz wiele innych ważnych zagadnień optymalizacyjnych. Badania nad algorytmami przybliżonymi, dostarczającymi rozwiązań dla zagadnień, w których zastosowanie metod dokładnych jest niemożliwe ze względu na liczność przestrzeni rozwiązań, stanową obecnie jedną z najszybciej rozwijających się gałęzi nauki.

* Praca sfinansowana przez KBN nr grantu: 11.11.120.227 (AGH, Kraków).

2. Ewolucyjny proces przeszukiwania przestrzeni rozwiązań

Algorytmy ewolucyjne powstały z relaksacji założeń i wprowadzania „innowacji" w klasycznych algorytmach genetycznych *(AG)*, które zostały zaproponowane w 1975 roku przez Hollanda, jako ogólnego przeznaczenia metaheurystykę odwołująca się do praw ewolucji [10]. Dzięki swoim cechom dają one możliwość przeszukiwania wielkich przestrzeni rozwiązań, wobec których klasyczne metody optymalizacji okazują się bezradne.

Zastosowanie technik EA w stosunku do konkretnego zadania wymaga spełnienia niewielu prostych warunków, takich jak określenie reprezentacji rozwiązania, zdefiniowanie zbioru pseudogenetycznych operatorów (zwanych też krótko operatorami genetycznym) oraz funkcji oceny (przystosowania) rozwiązań (ang. *fitness function*).

Dziedziczona wiedza zgromadzona w procesie poszukiwania rozwiązania jest kodowana w postaci listy rozwiązań zapisanej w zbiorze *P*, zwanym dalej populacją, gdzie $M = |P|$ jest rozmiarem populacji. Działanie operatorów genetycznych modeluje ewolucję populacji P, bowiem każdy operator genetyczny generuje nowe rozwiązania (zwane potomkami) bazując na poprzednich rozwiązaniach (zwanych rodzicami).

Zastosowany przez nas algorytm należy do algorytmów zwanych *modGA*, zmodyfikowanych w stosunku do klasycznego AG. Został on zaproponowany przez Michalewicza w monografii [11].

Modyfikacja algorytmu *modGA*, w odniesieniu do klasycznego *AG*, polega na tym, że nowy zbiór *P* jest formowany w następujący sposób: tylko część $r < M$ rozwiązań 'starego' zbioru *P* jest wybierana jako rodzice do przekształcenia, a *M-r* rozwiązań przenoszona jest do nowej populacji bez zmian. Rozwiązania z wartością większą niż średnia wartość funkcji przystosowania rozwiązań zbioru *P* mają większą szansę zostania rodzicami, natomiast rozwiązania z wartością funkcji przystosowania mniejszą niż średnia mają większą szansę być wybrane do odrzucenia. Nowy zbiór *P* składa się z *(M-r)* rozwiązań starego zbioru oraz *r* nowych rozwiązań-potomków otrzymanych na drodze przekształcenia przez operatory genetyczne *r* rodziców wybranych ze starego zbioru *P*. W ten sposób w jednej iteracji tylko *r* $(r \leq M)$ rozwiązań podlega procesowi selekcji i przekształcania, dzięki czemu algorytm *modGA* należy do klasy *Steady State GA*.

W artykule przedstawiamy algorytm *HGEN-I*, gdzie podczas jednej iteracji jest wybierany jeden operator w sposób przypadkowy (rozkład normalny), a następnie *r*, $r \in \{1,2\}$ rozwiązań jest selekcjonowanych z populacji i poddawanych przekształceniu. Jeśli został wybrany operator binarny, wtedy selekcjonujemy dwa rozwiązana, jedno - jeśli unarny.

Założono także, że zbiór P jest liniowo uporządkowany za pomocą funkcji przystosowania: pierwsze rozwiązanie zbioru jest rozwiązaniem najlepszym Π_{best},..., ostatnie rozwiązanie (rozwiązanie nr M) jest rozwiązaniem najgorszym Π_{worst} w zbiorze.

Podstawowym problemem, z jakim borykają się tego typu algorytmy jest częste „utknięcie" procesu optymalizacji w minimum lokalnym. Jest to spowodowane tym, iż najlepsze rozwiązania w populacji mają wielokrotnie większą szansę selekcji na rodziców niż pozostałe, co powoduje powstanie tzw. „superosobników", a co za tym idzie, eliminację innego niż reprezentowany przez nie (być może bardziej obiecującego) „materiału genetycznego".

W poniższym artykule zaproponowano zastosowanie algorytmu wykorzystującego, oprócz typowych operatorów przeznaczonych dla zagadnień permutacyjnych, jak mutacja, PMX, OX, operator optymalizacji lokalnej korzystający z „tablicy inwersji". Operator ten umożliwia iteracyjne poszukiwanie lepszych rozwiązań w przestrzeni, w której następne rozwiązanie jest tworzone na podstawie informacji o rozwiązaniu aktualnym. W operatorze tym zastosowano zaproponowaną przez Yureta i De la Mazę [16] procedurę pozwalającą na optymalizację wartości funkcji nieróżniczkowalnej w przestrzeni kartezjańskiej, zwaną *dynamic hill climbing*.

3. Algorytmy ewolucyjne dla problemów permutacyjnych

Do podstawowych metod stosowanych w ewolucyjnych algorytmach, poszukujących przybliżonych rozwiązań w zagadnieniach permutacyjnych, można zaliczyć:

- funkcje kary, gdzie na nowo wygenerowane rozwiązanie nie narzucamy ograniczeń koniecznych na to, aby było ono permutacją. Jakość nowo wygenerowanego rozwiązania jest proporcjonalna do funkcji oceny i odwrotnie proporcjonalna np. do liczby „nielegalnych" (powtarzających się) elementów w permutacji,
- stosowanie operatorów generujących jedynie „legalne" permutacje. Są to najczęściej operatory OX (order crossover), PMX (partially matched crossover), CX (cycle crossover) oraz operatory problemowo zorientowane, jak np. AEX (alternating edges crossover), STC (subtour chunking crossover) stosowane dla problemu TSP,
- metody transformacji – umożliwiające rozwiązywanie problemów permutacyjnych metodami stosowanymi dla innego typu zagadnień.

Poniżej prezentujemy rezultaty poszukiwań przybliżonych algorytmów ewolucyjnych. Realizują one ewolucyjny proces poszukiwania dla problemów permutacyjnych, należących

do klasy zagadnień *NP-trudnych* na przykładzie zagadnienia o kwadratowym wskaźniku jakości, zwykle oznaczanym symbolem *QAP (ang. Quadratic Assignment Problem)*. Algorytm przetestowano opierając się na bogatym zbiorze przykładowych zadań zaczerpniętych z biblioteki *QAPLIB-A*, będących problemami testowymi przeznaczonymi dla zagadnień aproksymacyjnych.

4. Zagadnienie QAP

Należy pamiętać, że *QAP* jest szczególnie trudnym zagadnieniem permutacyjnym. Rozwiązanie tego zagadnienia metodami dokładnymi dla rozmiaru $n > 15$ napotyka znaczne trudności obliczeniowe, co jest powodem licznych prac w zakresie rozwoju metod przybliżonych dla tego zagadnienia (por. [1, 3, 7, 8]).

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:

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

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. Macierz 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 permutacji, określa koszt globalny eksploatacji systemu.

W badanym przez nas algorytmie zastosowano naturalną reprezentację rozwiązania QAP, permutację oraz operatory genetyczne zapewniające dopuszczalność otrzymanych rozwiązań-potomków oraz funkcję oceny rozwiązań $f(\Pi)$.

5. Transformacja zagadnień permutacyjnych

Koncepcja transformacji analitycznej została z sukcesem zastosowana w stosunku do wielu problemów w dziedzinie fizyki oraz inżynierii. Typowym przykładem może tu być transformacja Fouriera pozwalająca na transformację sygnału z dziedziny czasu w dziedzinę częstotliwości. Dzięki takiej operacji wiele złożonych działań w dziedzinie czasu odpowiada o wiele prostszym w dziedzinie częstotliwości. Odwrotna transformacja Fouriera pozwala na powrót do dziedziny czasu. Podstawowym problemem w zastosowaniu TF był czas, w jakim

można dokonać takiej transformacji. Cooley i Tukey [6] odkryli nowy algorytm o złożoności $O(n \log n)$ zamiast $O(n^2)$, co umożliwiło bardzo szybki rozwój metod przetwarzania sygnałów, a co za tym idzie, niezwykle szybki rozwój dziedzin z nim związanych.

Można zdefiniować algorytm o złożoności obliczeniowej $O(n \log n)$, który pozwoli na mapowanie permutacji do przestrzeni wektorowej. Oczywiście, istnieje metoda o takiej samej złożoności obliczeniowej, pozwalająca na odwrotna operację, tj. mapowanie rozwiązań z przestrzeni wektorowej do przestrzeni permutacji. Tego typu transformacja została opisana po raz pierwszy w opracowaniu Knutha [9] i nosi nazwę *inversion table*. Została ona stworzona w celu uzyskania matematycznego narzędzia umożliwiającego dowodzenie twierdzeń oraz opis własności permutacji, w szczególności do określenia miary odległości pomiędzy dwoma punktami w przestrzeni permutacji.

6. Tablica inwersji

Rozważmy pewną permutację n liczb naturalnych $\{\Pi_1, \Pi_2, .., \Pi_n\}$ zbioru liczb $\{1,2,..,n\}$ oraz jej tablicę inwersji $\{a_1, a_2, .., a_n\}$, gdzie element a_j jest określony przez liczbę elementów w permutacji, na lewo od elementu j, które są większe niż element j.

$$a_{\Pi_j} = \sum_{i=1}^{i=j} \begin{cases} 1 & \text{gdy } \Pi_i > \Pi_{ij} \\ 0 & \text{w p.p} \end{cases} \qquad (1)$$

gdzie $0 \le a_1 \le n-1$, $0 \le a_1 \le n-2$, ..., $a_n = 0$.

Inaczej mówiąc, n wymiarowa przestrzeń permutacji jest transformowana do pewnego typu $n-1$ wymiarowej liniowej przestrzeni dyskretnej. Jak widać, taka transformacja umożliwia mapowanie pomiędzy dwoma przestrzeniami o zupełnie innych własnościach, co może zostać wykorzystane w procesie optymalizacji. Teraz opiszemy odwrotną operację.

Weźmy pewien łańcuch zdefiniowany następująco: $\alpha=[m_1,n_1]$, $[m_1,n_1]$,..., $[m_n, n_n]$ oraz pewien pusty łańcuch $\varepsilon=0$. Możemy teraz zdefiniować operację binarną $\otimes$, która w oparciu o dwa łańcuchy $([\mathbf{m}.\ \mathbf{n}]\alpha)$, $([\mathbf{m}.\ \mathbf{n}]\beta)$, gdzie α, β są podłańcuchami bez pierwszego elementu, tworzy nowy łańcuch zgodnie z regułą:

$$([m,n]\alpha) \otimes ([m',n']\beta) = \begin{cases} ([m,n](\alpha \otimes (m'-m,n')\beta)) & \text{dla } m \le m' \\ ([m',n']([m-m'-1,n]\alpha) \otimes \beta) & \text{dla } m > m' \end{cases} \qquad (2)$$

oraz $\varepsilon \otimes \alpha = \alpha \otimes \varepsilon = \alpha$ i $\otimes$ jest operacją łączną, tj. $\alpha \otimes (\beta \otimes \gamma) = (\alpha \otimes \beta) \otimes \gamma$.

Można dowieść że:

$$[\Pi_1,1]\otimes[\Pi_2,2]\otimes\ldots\otimes[\Pi_n,n]=[0,a_1]\,[0,a_2]\ldots[0,a_n]. \quad (3)$$

Inaczej mówiąc, złożenie listy elementów, gdzie przestrzenią są wartości tablicy inwersji oraz liczby od 1 do n, generuje listę elementów, gdzie pole liczb jest odpowiednim elementem permutacji. Co więcej, można tego dokonać opierając się na algorytmie o złożoności $O(n \log n)$. Ze względu na szczupłość miejsca pominiemy opis algorytmu implementującego powyższe rozważania.

7. Algorytm ewolucyjny

Algorytm *HGEN-I* wykorzystuje dwa operatory unarne i dwa operatory krzyżowania. Zauważmy, że w przypadku formalizacji rozwiązania zagadnienia w postaci permutacji Π informacją zawartą w rozwiązaniu jest tylko *kolejność* elementów. W takim przypadku w algorytmie genetycznym można zastosować tylko operatory zmieniające kolejność elementów permutacji. W przeprowadzonych badaniach komputerowych zastosowano dwa unarne operatory genetyczne:

1. Operator mutacji RM (random mutation- zob. [4, s. 125-134]).

2. Operator optymalizacji lokalnej LO.

(a). Wybierz w sposób losowy (rozkład równomierny) rozwiązanie Πz populacji;

(b). Utwórz odpowiadającą rozwiązaniu Π tablicę odwrotną x, wyzeruj wektor u oraz w sposób losowy wygeneruj początkowy kierunek szukania określony przez wektor v;

(c). $iter=0$;

(d). Jeśli $f(x+v)\geq f(x)$ oraz $iter<maxIter$, idź do **(e)**; wp.p idź do **(f)**;

(e). Wygeneruj w sposób losowy nowy kierunek szukania $v=randomVector(v)$ oraz podstaw $iter:=iter+1$ oraz idź do **(d)**;

(f). Jeśli $f(x+v)>f(x)$, podstaw $v=v/2$ oraz idź do **(j)**; wpp. idź do **(g)**;

(g). Jeśli $iter=0$, podstaw $x=x+v$, $u=u+v$, $v=2v$ oraz idź do **(j)**; wp.p idź do **(h)**;

(h). Jeśli $f(x+u+v)<f(x)$, podstaw $x=x+v+u$, $u=u+v$, $v=2v$ oraz idź do **(j)**; wp.p idź do **(i)**;

(i). Podstaw $x=x+v$; $u=v$; $v=2v$ oraz idź do **(j)**;

(j). Jeśli $|v| < minV$ to idź do **(k)**; wpp. idź do **(c)**;

(k). STOP – zwróć otrzymane rozwiązanie.

Operator optymalizacji lokalnej implementuje procedurę pozwalającą na optymalizację wartości funkcji nieróżniczkowalnej w przestrzeni kartezjańskiej, zwaną *dynamic hill*

climbing. Parametrami procedury są: *maxIter* – maksymalna liczba losowych prób wygenerowania wektora określającego kierunek poprawy rozwiązania, oraz *minV* – minimalna długość wektora określającego kierunek poprawy rozwiązania. Zastosowanie dodatkowego wektora *u* pamiętającego poprzedni „obiecujący" kierunek poszukiwania i jego liniowa kombinacja z wektorem *v*, ułatwia algorytmowi pokonywanie „siodeł" optymalizowanej funkcji.

Dla rozwiązań, w których ważna jest kolejność elementów, operatory krzyżowania są bardziej skomplikowane, ponieważ klasyczne (ślepe) krzyżowanie może prowadzić do wyznaczenia rozwiązań niedopuszczalnych. Tak więc należy zastosować ściśle określone heurystyczne operatory krzyżowania. W naszych eksperymentach komputerowych stosowaliśmy dwa operatory krzyżowania:

3. Operator PMX (partially matched crossover- zob., [4, s.172-174]).

4. Operator OX (order crossover- zob., [4, s.172-174]).

ALGORYTM HGEN-I:

Aby wyznaczyć rozwiązanie przybliżone Π_{approx} wykonaj następującą procedurę:

Krok 1. Wyznaczenie populacji początkowej.

Wygeneruj, w sposób losowy, *M* permutacji Π oraz określ funkcje przystosowania *f(Π)* dla każdej z nich. Z wygenerowanych rozwiązań utwórz populację początkową P o rozmiarze *M*, uporządkowaną wg wartości funkcji przystosowania *f(Π)*, tzn. permutacja nr 1 jest najlepsza ($\Pi_{best} = \arg\min\{f(\Pi): \Pi \in P\}$), a ostatnia permutacja nr *M* jest najgorsza.

Krok 2. Wybór operatora genetycznego.

Wylosuj typ operatora genetycznego ze zbioru {*RM, LO, OX, PMX*}, gdzie każdy z operatorów jest losowany z prawdopodobieństwem $p_i \geq 0$, $i \in \{RM, LO, OX, PMX\}$, przy czym $p_{RM} + p_{LO} + p_{OX} + p_{PMX} = 1$.

Krok 3. Wybór rodziców rodzica

Dla wybranego operatora wylosuj, zgodnie z rozkładem równomiernym, ze zbioru *P*, zależnie od typu operatora, jedno (w przypadku operatora unarnego) lub dwa (w przypadku operatora krzyżowania) rozwiązania, które nazywamy rodzicami.

Krok 4. Generowanie potomków.

Za pomocą wylosowanego operatora genetycznego dokonaj modyfikacji rozwiązania-rodzica/rozwiązań-rodziców i wyznacz w ten sposób rozwiazanie-potomka/rozwiązania-potomków.

Krok 5. Poprawa rozwiązań w zbiorze P.

Dla każdego potomka oblicz wartość funkcji przystosowania. Jeżeli ta wartość jest lepsza od wartości najgorszego rozwiązania zbioru P, to umieść takiego potomka w zbiorze P, usuwając zarazem z tego zbioru rozwiązanie najgorsze.

Krok 6. Warunek STOP'u.

Jeżeli wygenerowano zadaną liczbę L potomków, to STOP zwróć najlepsze rozwiązanie zbioru P-$\Pi_{approx} = arg\ min\ \{f(\Pi): \Pi \in P(O)\}$. W przeciwnym wypadku idź do kroku 2.

Wielkości takie, jak rozmiar populacji M, liczba iteracji algorytmu L, prawdopodobieństwa selekcji operatorów RM - p_{RM}, LO - p_{LO}, OX -p_{OX}, PMX - p_{PMX} są parametrami algorytmu *HGEN-I*.

8. Wyniki badań komputerowych

Na podstawie algorytmu *HGEN-I* wykonano eksperymentalny program komputerowy, zakodowany w języku C++, pod systemem WINDOWS NT, zaimplementowany dla zagadnienia *QAP*. W tabeli 1 przedstawiono wyniki badań eksperymentalnych dla zagadnienia *QAP*, zbioru 33 testów o rozmiarze n=26-60 zaczerpniętych z biblioteki QAPLIB-A [1]. Wszystkie eksperymenty komputerowe wykonane w oparciu o algorytm *HGEN-I* korzystają z następującego zbioru parametrów: L=10000 iteracji, M = 100 (rozmiar populacji) Zbiór operatorów $H=\{RM, LO, OX, PMX\}$. Przy ustalonych powyżej parametrach wykonano serie obliczeń dla operatorów ze zbioru H. Przyjęto $p_{LO} = 0.1$, $p_{OX} = 0.4$, $p_{PMX} = 0.3$.

Otrzymane wyniki porównano z najlepszym wariantem hybrydowego algorytmu z długoterminową pamięcią częstotliwościową *HGEN-2* przestawionego w publikacji [5].

Przyjęto oznaczenia:

$f_{QAPLIB-A}$ - najlepsza znana wartość funkcji przystosowania dla zaczerpnięta z biblioteki QAPLIB-A,

f_{start} - wartość funkcji przystosowania najlepszego rozwiązania w populacji startowej $\Pi_{start} = arg\ min\ \{f(\Pi): \Pi \in P(O)\}$,

f_{ap} - wartość funkcji przystosowania najlepszego znalezionego rozwiązania dla Π_{approx},

$E = 100\%(f_{ap} - f_{QAPLIB-A})/f_{QAPLIB-A}$,

L_{ap} - liczba iteracji po ilu zostało znalezione najlepsze rozwiązanie Π_{approx}.

Tabela 1

Wyniki badań eksperymentalnych

Nazwa	$f_{QAPLIB-A}$	f_{BAM}	HGEN-2			HGEN-I		
			f_{ap}	E [%]	L_{ap}	f_{ap}	E [%]	L_{ap}
BUR26A	5426670	5666080	5426670	0,00000	169	5433520	0,12623	9253
BUR26B	3817852	4008670	3817850	-0,00005	5475	3827790	0,26030	9481
BUR26C	5426795	5735520	5426960	0,00304	548	5430960	0,07675	9439
BUR26D	3821228	4079800	3821410	0,00476	4262	3824450	0,08432	9966
BUR26E	5386879	5667490	5387320	0,00819	4920	5389560	0,04977	9466
BUR26F	3782044	4030400	3782040	-0,00011	4309	3788980	0,18339	7621
CHR22A	6156	10508	6314	2,56660	1171	6976	13,32034	9021
CHR22B	6194	10940	6470	4,45593	6745	6918	11,68873	9747
CHR25A	3796	14076	4180	10,11591	487	6912	82,08641	9537
ESC32A	130	370	140	7,69231	7786	162	24,61538	9858
ESC32B	160	408	168	5,00000	5945	204	27,50000	8799
ESC32C	642	752	642	0,00000	9220	642	0,00000	5627
ESC32D	200	312	200	0,00000	7080	208	4,00000	2470
ESC32E	2	6	2	0,00000	9030	2	0,00000	1
ESC32F	2	24	2	0,00000	9239	2	0,00000	57
ESC32G	6	12	6	0,00000	8463	6	0,00000	83
ESC32H	438	630	438	0,00000	7819	450	2,73973	9014
KRA30A	88900	124630	90790	2,12598	2321	98380	10,66367	9659
KRA30B	91420	127200	91890	0,51411	510	97910	7,09910	9258
LIPA30A	13178	13726	13390	1,60874	736	13445	2,02610	8945
LIPA30B	151426	191360	151426	0,00000	8094	175147	15,66508	9932
LIPA40A	31538	32525	31895	1,13197	5396	32113	1,82320	9682
LIPA40B	476581	604619	476581	0,00000	7961	573423	20,32016	9895
LIPA50A	62093	63823	62764	1,08064	3819	63096	1,61532	8348
LIPA50B	1210244	1524310	1210240	-0,00033	4595	1467180	21,23010	9651
LIPA60A	107218	109816	108254	0,95626	1263	108743	1,42234	9817
LIPA60B	2520135	3219390	2995440	18,86030	5037	3076940	22,09425	8873
SKO42	15812	19294	15908	0,60713	1670	16662	5,37566	9674
SKO49	23386	28164	23526	0,59865	3014	24946	6,67066	9600
SKO56	34458	41110	34672	0,62105	2463	36686	6,46584	9332
THO30	149936	202012	151206	0,84703	545	157326	4,92877	9174
THO40	240516	309522	241824	0,54383	410	256512	6,65070	9942
WIL50	48816	54250	48904	0,18027	6356	50650	3,75696	9823

9. Podsumowanie

Należy stwierdzić, że algorytm *HGEN-I* jest algorytmem przeznaczonym do rozwiązywania szerokiej gamy zadań permutacyjnych, a nie specjalizowaną procedurą ukierunkowaną na rozwiązywaniem jedynie problemu *QAP*. Zaproponowany operator optymalizacji lokalnej pracujący w dyskretnej przestrzeni wektorowej, co jest możliwe dzięki zastosowaniu transformacji, może być doskonałym uzupełnieniem innych procedur z dziedziny algorytmów ewolucyjnych. Otwiera on nowe możliwości dla algorytmów przybliżonych, w szczególności gdy procedura przypadkowego generowania początkowego kierunku poszukiwania zostanie zastąpiona procedurą korzystającą, ze zgromadzonej wiedzy uzyskanej podczas procesu optymalizacji lub z wiedzy o samym o problemie.

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, s. 406-415.
2. Burkard R.E., Karisch S.E., Rendl F.: QAPLIB-A Quadratic Assignment Problem Library, European Journal of Operational Resarch, 55, 1991, s.115-119.
3. Brukard R.E., Stratmann K.H.: Numerical investigation on quadratic assigment problems. Naval Research Logistics Quaterly, vol.25, 1978, s.129-147.
4. Chmiel W.: Algorytm ewolucyjny z ograniczonym wyborem operatorów genetycznych., Zeszyty Naukowe Politechniki Śląskiej, seria Automatryka, z.125, 1998, s.125-134.
5. Chmiel W., Kadłuczka P.: Algorytm hybrydowy z długoterminową pamięcią częstotliwościową., University of Mining and Metallurgy Press, Automatyka vol.3, Kraków 1999, s. 59-71.
6. Cooley P.M., Tukey J.W.: An Algorithm for the Machine Computation of Complex Fourier Series, Mathematics of Computation, vol.19, 1965
7. Filipowicz B., Wala K.: Algorytmy optymalizacji kwadratowego zagadnienia przydziału. Kwartalnik Elektrotechniki, z.1, 1992, Wydawnictwo AGH w Krakowie.
8. Finke G., Burkard R.E, Rendl F.: Quadratic assigments problems. Annals of Discrete Mathematics, vol.31, North-Holland, 1987, s. 61 -82.
9. Knuth D.E.: The Art. Of Computer Programming, Addison-Wesley Publishing Company, Inc., 1973
10. Holland J.: Adaptation in natural and artificial systems. Univ. of Michigan Press, Ann Arbor, MI, 1975.
11. Michalewicz Z.: Genetic algorithms + data structures = evolution programs, Springer-Verlag, Berlin 1992.
12. Michalewicz Z.: Heuristic Methods for Evolutionary Computation Techiques, Journal of Heuristics, 1, Kluwer Academic Publishers, 1995, s. 177-206.
13. Turrini S.: Optimization in Permutation Spaces, Western Research Laboratory Report vol. 1, Palo Alto, California 1996.
14. Wala K., Chmiel W.: An improved genetic algorithm for *NP-hard* permutation problems, Proc. of Third Int. Symposium on Methods and Models in Automation and Robotics, 10-13 September 1996, Międzyzdroje, Poland, vol.3, s. 1163-1166.
15. Wala K., Chmiel W.: Evolution Algorithm for Quadratic Assignment Problem, University of Mining and Metallurgy Press, Elektrotechnika 1,1, Kraków, 1997, s: 409-414.
16. Yuret D., De La Maza M.: Dynamic Hill Climbing, AI Expert,1994, p.26-31

Recenzent: Prof. dr hab.inż. A.Niederliński

Abstract

Permutation problems are an important class of decision problems in the combinatorial optimization domain. Classical instances of permutation problems include quadratic assignment problem *(QAP)*, graph coloring, production scheduling problems, as well as

variety of design problems. The paper presents the results of computer investigation of approximate algorithm, realized evolution artificial search process, for *QAP* as a hard instance of permutation problems and, on the other hand, there is a rich Quadratic Assignment Problem Library, called QAPLIB-A, with test task of this problem for approximate algorithm examinations. *QAP* generalizes many NP-hard combinatorial optimization problems, including the travelling salesman problem, and until now even quite small instances for exact algorithm are computationally intractable.

Evolutionary algorithms (*EAs*) are powerful search techniques taking inspiration from genetics and natural selection. They can effectively explore very large solutions spaces. The ones requirements for applying EAs to the problem are: solution representation of the problem to be solved, a set of pseudo-genetic operators called briefly genetic operators and a evaluation function of the solution called fitness function.

We examine evolution search process called *HGEN-I*, where in course of one iteration initially one genetic operator is randomly chosen and then r, $r \in \{1,2\}$, solutions are selected from the population and processed: one solution if unary operator is chosen and two in case of crossover operator. Thus the algorithm also belongs to the class of *Steady State GAs* and in this way it has all features of the *modGA*. Additionally in one of unary operators we introduce technique normally used to optimize any KIND of objective functions in Cartesian space where derivatives are not available or impossible to determine. It is possible due to analytical transformation between permutation and vector spaces.