ZESZYTY NAUKOWE POLITECHNIKI ŚLASKIEJ 2008
Seria: AUTOMATYKA z. 151 Nr kol. 1797

Wojciech CHMIEL, Piotr KADŁUCZKA
Akademia Górniczo-Hutnicza

WIELOFAZOWA METODA RÓŻNICOWANIA POPULACJI W ALGORYTMIE EWOLUCYJNYM

Streszczenie. Zastosowana w algorytmie ewolucyjnym koncepcja zmiennej w czasie strategii sukcesji ma na celu sterowanie zbieżnością algorytmu. Początkowa faza obliczeń różnicuje w większym stopniu rozwiązania w populacji niż fazy późniejsze. Prawdopodobieństwo wstawienia nowo wygenerowanego rozwiązania do populacji, zmienne w kolejnych etapach, jest uzależnione od wartości funkcji przystosowania oraz od pewnej funkcji rozkładu prawdopodobieństwa.
Jako zagadnienie testowe dla zaproponowanego algorytmu przyjęto *NP-trudne* kwadratowe zagadnienie przydziału.

A MULTI-PHASE DIVERSIFICATION METHOD OF POPULATION IN THE EVOLUTIONARY ALGORITHM

Summary. This paper investigates a new advanced evolutionary algorithm for optimization of permutation problems. Implementation of varying in time strategy of succession in evolution algorithms enables controlling the population diversification. In early phases of optimization the diversification of population is greater than the later ones. During all phases the probability of adding solution to population depends on the solution fitness function and certain probability density function.
The experiments were performed for standard test problems of quadratic assignment problems (*QAP*).

1. Wprowadzenie

Problem właściwego sterowania zbieżnością algorytmu ewolucyjnego (*EA*, ang. *evolutionary algorithm*) jest niezwykle istotny oraz bardziej złożony niż w przypadku innych typów algorytmów przybliżonych. Zjawisko braku lub przedwczesnej zbieżności *EA* ([4], [5], [6]) należy rozważać na pierwszym miejscu podczas implementacji tego typu algorytmów. Nie istnieje gotowa recepta pozwalająca na realizację idealnego algorytmu *EA*, gdyż uwarunkowane to jest zbyt wieloma czynnikami, najczęściej mocno związanymi ze specyfiką rozwiązywanego zagadnienia. Cechy działania EA są zależne między innymi od sposobu realizacji pokolenia, przetwarzania populacji, występowania elit, mechanizmu selekcji,

stosowanych operatorów genetycznych oraz szeregu parametrów, np. wielkości populacji, prawdopodobieństw stosowania operatorów itp. Wyżej wymienione elementy realizują dwa przeciwstawne mechanizmy: różnicowania i intensyfikacji, których właściwa proporcja ma gwarantować wysoką efektywność algorytmu. Mechanizmy te jednak nie działają w jednej „płaszczyźnie" - znaczenie ma nie tylko ich „sumaryczna wartość", ale również wielorakie, wzajemne zależności. W poniższym artykule zaproponowano algorytm stosujący zmienny w czasie mechanizm wyboru rozwiązań wstawianych do populacji. Osłabienie zbieżności algorytmu uzyskano przez wprowadzenie podziału populacji na dwie części. Do pierwszej - elitarnej, wprowadzane są rozwiązania, których wartość funkcji oceny jest lepsza od najgorszego w elicie. Natomiast w przypadku drugiej prawdopodobieństwo wstawienia rozwiązania odrzuconego z elity jest zależne od jego jakości oraz fazy procesu przeszukiwania przestrzeni rozwiązań.

2. Zagadnienie QAP

Problem kwadratowego przydziału (ang. *Quadratic Assignent Problem*) [1] należy do klasy zagadnień *NP-trudnych*. Obecnie brak jest metod, które by pozwoliły w sposób dokładny rozwiązać instancje o rozmiarze większym niż 36 w akceptowalnym czasie. Zagadnienie to można zdefiniować następująco. Dany jest zbiór $N=\{1,...,n\}$ oraz dwie $(n\times n)$-wymiarowe macierze $D=[d_{i,k}]$, $F=[f_{j,l}]$. 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 D jest wtedy macierzą odległości pomiędzy pozycjami rozmieszczenia obiektów, podczas gdy macierz F opisuje powiązania (np. liczbę połączeń lub wielkość przepływu) występujące pomiędzy obiektami.

Należy znaleźć permutację $\pi = (\pi(1),...,\pi(n))$ elementów zbioru N, która minimalizuje funkcję celu $\phi(\pi)$ o następującej postaci:

$$\phi(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j} d_{\pi(i)\pi(j)} \qquad (1)$$

Funkcja celu $\phi(\pi)$, $\pi \in \Pi$, określa globalny koszt realizacji lub eksploatacji systemu, natomiast Π jest zbiorem permutacji zbioru N.

3. Sterowanie zbieżnością algorytmu ewolucyjnego

Konstruując *EA*, należy pogodzić dwa sprzeczne podejścia: eksplorację, czyli swobodne poszukiwanie optimum globalnego w całej przestrzeni rozwiązań, oraz eksploatację, tj. możliwie dokładne przeszukiwanie otoczenia optimum lokalnego. Kompromis, pozwalający podążać w stronę optimum lokalnego, a następnie opuścić jego obszar przyciągania, realizowany jest za pomocą przeciwstawnych mechanizmów: intensyfikacji i różnicowania.

Problem przedwczesnej zbieżności algorytmu ewolucyjnego jest konsekwencją stosowanego w algorytmie mechanizmu selekcji oraz sposobu przetwarzania populacji (zjawisko naporu selekcyjnego). Prowadzi on do sytuacji, w której populacja zawiera rozwiązania będące w obszarze przyciągania optimów lokalnych, a zastosowane w algorytmie ewolucyjnym mechanizmy nie pozwalają na dalszą eksplorację przestrzeni

rozwiązań. W celu poprawy efektywności mechanizmów selekcji należy stworzyć odpowiednie proporcje między naporem selekcyjnym, prowadzącym do przedwczesnej zbieżności algorytmu, a różnorodnością populacji [7].

Schemat rozważanego w artykule algorytmu ewolucyjnego jest oparty na selekcji z częściowym zastępowaniem (ang. *Steady-State*), w której rozwiązania potomne konkurują z rodzicielskimi o miejsce w populacji. Zależnie od typu stosowanego operatora, w każdym pokoleniu w populacji wymienione zostanie jedno lub dwa rozwiązania, które zastąpią najgorsze. Wybór osobnika (osobników) realizowany jest po dokonaniu wyboru operatora za pomocą procedury losowania o rozkładzie równomiernym.

Zastosowany schemat należy do metod silnie intensyfikujących, ponieważ prowadzi on do sytuacji w której rozwiązania najlepsze utrzymują się w populacji przez dużą liczbę pokoleń. Takie działanie algorytmu było podstawą do wprowadzenia w nim modyfikacji, polegającej na wprowadzeniu podziału populacji na dwa podzbiory. Pierwszy - elitarny, jest przetwarzany w oparciu o selekcję z częściowym zastępowaniem. Jakość najgorszego rozwiązania w tym zbiorze jest parametrem decydującym o wstawieniu do niej nowo utworzonych rozwiązań. Drugi podzbiór populacji - różnicujący, gromadzi rozwiązania odrzucone przez elitę. Prawdopodobieństwo wstawienia do tej części populacji tego rozwiązania zależy od różnicy wartości funkcji oceny rozwiązania i funkcji oceny najgorszego rozwiązania w elicie, a także fazy procesu optymalizacji. Ogólnie, schemat, zgodnie z którym do obu części populacji jest wstawiane nowe rozwiązanie, jest oparty na regule akceptacji Metropolisa z rozkładem prawdopodobieństwa Boltzmanna. W przyjętym schemacie algorytmu energię układu reprezentuje najgorsze rozwiązanie elitarne. Zgodnie z nim, rozwiązanie lepsze niż najgorsze w elicie zawsze jest wstawiane do podzbioru elitarnego. Natomiast rozwiązanie gorsze niż najgorsze w elicie jest wstawiane do podzbioru różnicującego z pewnym prawdopodobieństwem zależnym od różnicy jakości pomiędzy nim a najgorszym rozwiązaniem z elity oraz od fazy procesu optymalizacji.

4. Algorytm GASA

W eksperymentach obliczeniowych do rozwiązywania zagadnienia *QAP* wykorzystano zmodyfikowany algorytm genetyczny modGA zaproponowany przez Michalewicza [6]. Wprowadzono w nim uproszczenia polegające na losowym wyborze rodziców z populacji w oparciu o rozkład równomierny oraz na współzawodnictwie o miejsce w populacji nowo utworzonych rozwiązań potomnych, na podstawie wartości funkcji celu.

Parametrami algorytmu są: O - zbiór operatorów pseudogenetycznych, $ZP=\{p_i\}$ - zbiór prawdopodobieństw wyboru operatorów ($p_i \geq 0$, $i \in O$, $\sum_{i \in O} p_i = 1$), λ - rozmiar populacji podstawowej, λ_D - rozmiar populacji różnicującej, T_0 – temperatura początkowa, α - parametr sterujący zmianą temperatury w kolejnych iteracjach $\alpha \in (0,1)$, I_{max} - zadana liczba wygenerowanych przez algorytm potomków. Przez P (P_D) oznaczono elitarną populację rozwiązań przetwarzaną przez algorytm (populacje różnicującą), natomiast przez R wygenerowany zbiór rozwiązań wstępnych.

```
Algorytm GASA(l_max, O, ZP, λ, λ_D, T_0, α)
R ← GenerujRozwiązaniaWstępne(λ)
Oceń(R)
P ← UtwórzPopulację(R, λ)
P_D ← UtwórzPopulację(-, λ_D)
l ← 0
T ← T_0
while l < l_max
do
    o ← LosujOperator(O, ZP)
    π_1 ← LosujRodzica(P ∪ P_D, o)
    π' ← GenerujPotomka(π_1, o)
    Oceń(π')
    if π' < Najgorsze(P)
    then {Wstaw(P, π')
    else if Rand[0,1] < e^(-(F(Najgorsze(P))-F(π'))/T)
         then {Wstaw(P_D, π')
    T ← αT
    l ← l+1
π_best ← Najlepsze(P)
return (π_best);
end
```

Rys. 1. Pseudokod algorytmu GASA

W algorytmie zastosowano następujące procedury:

GenerujRozwiązaniaWstępne(λ) - generuje losowo, w oparciu o rozkład równomierny, populację λ rozwiązań początkowych,

Oceń(π) - określa wartości funkcji przystosowania (celu) dla rozwiązania lub rozwiązań,

UtwórzPopulacjęPoczątkową(R,λ) - tworzy uporządkowaną wg wartości funkcji przystosowania (od najlepszego do najgorszego) populację początkową P o rozmiarze λ, w oparciu o zbiór R zawierający rozwiązania wstępne,

LosujOperator(O, ZP) - losuje, w oparciu o rozkład równomierny, operator pseudogenetyczny ze zbioru O z prawdopodobieństwem operatorów ($p_i \geq 0$, $i \in O$, $\sum_{i \in O} p_i = 1$),

LosujRodzica(P, o) - losuje, dla wybranego operatora o, zgodnie z rozkładem równomiernym ze zbioru P, jedno (w przypadku operatora unarnego) lub dwa (w przypadku operatora krzyżowania) rozwiązania-rodziców,

GenerujPotomka(π, o) - w oparciu o wylosowany operator o dokonuje modyfikacji rodzica / rodziców i zwraca potomka / potomków,

Wstaw(P, π) - wstawia potomka / potomków do populacji, w oparciu o wartość funkcji przystosowania. Jeżeli wartość funkcji przystosowania dla potomka jest lepsza niż dla najgorszego rozwiązania ze zbioru P, to umieszcza go w zbiorze P, usuwając rozwiązanie najgorsze. W przeciwnym przypadku potomek nie jest wstawiany do P,

Rand(*a*, *b*) - generator liczb pseudolosowych z zakresu [*a*, *b*],
Najgorsze() /Najlepsze() - zwraca najgorsze/ najlepsze rozwiązanie w populacji.

5. Wyniki badań obliczeniowych

Na podstawie algorytmu *GASA* wykonano eksperymentalne oprogramowanie dla zagadnienia *QAP*, zaimplementowane w języku C++.

Tabela 1
Wartości uśrednionych błędów dla wszystkich instancji testowych

		I_1	I_2	I_3	I_4
O_1	*GASA*	1,36	1,15	1,03	1,71
	GA	1,68	1,29	1,28	1,23
O_2	*GASA*	0,91	1,21	1,14	1,01
	GA	1,47	1,44	1,49	1,31
O_3	*GASA*	1,46	0,63	1,39	1,13
	GA	1,99	1,37	1,23	1,48
O_4	*GASA*	1,34	1,19	1,02	0,56
	GA	1,09	1,09	1,19	2,08
O_5	*GASA*	1,52	1,56	1,24	1,46
	GA	1,49	1,39	1,05	1,52

Przeprowadzone doświadczenia obliczeniowe wykonano w oparciu o zbiór operatorów O={*RM, PMX, OX, LO*}, gdzie operatory *PMX* oraz *OX* są standardowymi operatorami [3] stosowanymi dla zagadnień permutacyjnych, natomiast operator *LO* jest operatorem optymalizacji lokalnej, przeszukującym otoczenie *2-OPT* [2]. Eksperymenty przeprowadzono dla pięciu zestawów prawdopodobieństw losowania operatorów O_1 = {0.1, 0.5, 0.2, 0.2}, O_2={0.1, 0.4, 0.4, 0.1}, O_3={0.1, 0.2, 0.5, 0.2}, O_4={0.1, 0.1, 0.6, 0.2}, O_5={0.1, 0.6, 0.1, 0.2} oraz czterech liczb generowanych potomków I_{max}={I_1, I_2, I_3, I_4}={20000, 25000, 45000, 60000). Pozostałe parametry algorytmu wynoszą: λ=100 (rozmiar populacji elitarnej), λ_D=50 (rozmiar populacji różnicującej), T_0=8000 oraz α=0.999.

Tabela 2
Najlepsze wyniki uzyskane dla *GASA* oraz *GA* podczas testów dla zestawu prawdopodobieństw O_2

Name	QAPLIB	ϕ_{GASA}	I_{max}	E	ϕ_{GA}	I_{max}	E
BUR26A	5426670	5426670	I_1,I_2,I_3,I_4	0,0000	5426670	I_1,I_2,I_3,I_4	0,0000
BUR26B	3817852	3817852	I_1	0,0000	3817852	I_1,I_2,I_3,I_4	0,0000
BUR26C	5426795	5426800	I_2,I_3,I_4	0,0001	5426800	I_2,I_3,I_4	0,0001
BUR26D	3821228	3821230	I_3	0,0001	3821260	I_1,I_3	0,0008
BUR26E	5386879	5386880	I_1,I_3	0,00002	5386880	I_3	0,00002
BUR26F	3782044	3782044	I_1,I_2,I_3,I_4	0,0000	3782044	I_1,I_4	0,0000
BUR26G	10117172	10117200	I_3	0,0003	10117200	I_3,I_4	0,0003
BUR26H	7098658	7098660	I_1,I_2,I_3,I_4	0,00003	7098660	I_1,I_2,I_3,I_4	0,00003
CHR22A	6156	6156	I_3	0,0000	6176	I_3	0,3249
CHR22B	6194	6276	I_1	1,3239	6362	I_1	2,7123
ESC32A	130	134	I_1,I_3,I_4	3,0769	134	I_4	3,0769
ESC32B	160	168	I_1,I_2,I_3,I_4	5,0000	168	I_1,I_2,I_3,I_4	5,0000

ESC32C	642	642	I_1,I_2,I_3,I_4	0,0000	642	I_1,I_2,I_3,I_4	0,0000
ESC32D	200	200	I_1,I_2,I_3,I_4	0,0000	200	I_1,I_2,I_3,I_4	0,0000
ESC32E	2	2	I_1,I_2,I_3,I_4	0,0000	2	I_1,I_2,I_3,I_4	0,0000
ESC32F	2	2	I_1,I_2,I_3,I_4	0,0000	2	I_1,I_2,I_3,I_4	0,0000
ESC32G	6	6	I_1,I_2,I_3,I_4	0,0000	6	I_1,I_2,I_3,I_4	0,0000
ESC32H	438	438	I_1,I_2,I_3,I_4	0,0000	438	I_1,I_2,I_3,I_4	0,0000
ESC64A	116	116	I_1,I_2,I_3,I_4	0,0000	116	I_1,I_2,I_3,I_4	0,0000
KRA30A	88900	88900	I_2	0,0000	88900	I_3,I_4	0,0000
KRA30B	91420	91420	I_1,I_2,I_3	0,0000	91420	I_3	0,0000
LIPA30A	13178	13178	I_1,I_2,I_3,I_4	0,0000	13178	I_3	0,0000
LIPA30B	151426	151426	I_1,I_2,I_3,I_4	0,0000	151426	I_1,I_2,I_3,I_4	0,0000
LIPA40A	31538	31538	I_3,I_4	0,0000	31538	I_1	0,0000
LIPA40B	476581	476581	I_1,I_2,I_3,I_4	0,0000	476581	I_1,I_2,I_3,I_4	0,0000
LIPA50A	62093	62684	I_2,I_3	0,9518	62093	I_4	0,0000
LIPA50B	1210244	1210244	I_1,I_2,I_3,I_4	0,0000	1210244	I_1,I_2,I_3,I_4	0,0000
LIPA60A	107218	108075	I_2	0,7993	108034	I_4	0,7611
LIPA60B	2520135	2520140	I_1,I_3,I_4	0,0002	2520140	I_3,I_4	0,0002
SKO42	15812	15828,00	I_4	0,1011890	15812	I_4	0,0000
SKO49	23386	23414	I_3	0,1197	23444	I_1	0,2480
SKO56	34458	34530	I_4	0,2090	34560	I_2	0,2960
STE36A	9526	9526	I_2,I_3,I_4	0,0000	9550	I_3	0,2519
STE36B	15852	15852	I_3	0,0000	15980	I_1,I_2,I_3	0,8075
STE36C	8239110	8255560	I_2	0,1997	8239110	I_1	0,0000
THO30	149936	150378	I_2,I_3,I_4	0,2948	150280	I_2,I_4	0,2294
THO40	240516	240902	I_3	0,1605	240934	I_2	0,1738
WIL50	48816	48848	I_1,I_2,I_3,I_4	0,0656	48842	I_4	0,0533
$E_{śr}$				0,3238			0,3668

W tabeli 1 przedstawiono wyniki badań eksperymentalnych zagadnienia *QAP*, dla $|N|$=38 zagadnień testowych o rozmiarze n=22-64, zaczerpniętych z biblioteki QAPLIB-A [1]. Uzyskane wyniki porównano z klasycznym algorytmem *GA*. Tabela 1 zawiera uśrednione wartości błędów $E_{śr}$ uzyskane dla całego zestawu 38 zadań testowych (dla 3 przebiegów algorytmu):

$$E_{Śr} = 100\% * \left(\sum_{i \in N} (\phi_i - \phi_{i_{QAPLIB-A}}) / \phi_{i_{QAPLIB-A}} \right) / |N| \qquad (2)$$

gdzie:

N – zbiór instancji testowych dla zagadnienia *QAP*, zaczerpniętych z biblioteki QAPLIB-A,

$|N|$ – liczebność zbioru testowego (38 instancji testowych),

ϕ_i – wartość średnia funkcji przystosowania najlepszego znalezionego rozwiązania π_{best}, dla i-tej instancji testowej z trzech przebiegów algorytmu.

W tabeli 2 porównano najlepsze wyniki oraz względny błąd E, uzyskany dla algorytmów *GASA* oraz *GA* dla zestawu prawdopodobieństw O_2, dla różnych liczb generowanych potomków $I_{\max} \in \{I_1, I_2, I_3, I_4\} = \{20000, 25000, 45000, 60000\}$. Podaną w tabeli wartość względnego błędu E zdefiniowano następująco:

$$E = 100\% * (\phi_{GA/GASA} - \phi_{IQAPLIB-A}) / \phi_{IQAPLIB-A} \quad (3)$$

gdzie: $\phi_{GA/GASA}$ jest wartością funkcji przystosowania najlepszych znalezionych rozwiązań przez algorytm *GA*(*GASA*). W tabeli 2 kolumna I_{max} precyzuje, dla jakiej liczby generowanych potomków uzyskano podany wynik.

6. Podsumowanie

Analiza parametrów jakości optymalizacji zamieszczona w tabeli 1 pozwala postawić tezę, że zastosowanie algorytmu *GASA*, dla wybranego zestawu instancji testowych, prowadzi do uzyskania nieco lepszych wyników niż klasyczny algorytm *GA*.

Wyniki z tabeli 1 pozwalają zauważyć, że:

- uśrednione wartości błędów dla algorytmu *GASA* $E_{śr}$ w serii dwudziestu eksperymentach są niższe w 13 przypadkach,
- w przypadku algorytmu *GASA* zestaw parametrów O_4 umożliwia uzyskanie najlepszych wyników,
- najniższą średnią wartość błędu dla wszystkich eksperymentów obliczeniowych uzyskano dla zestawu prawdopodobieństw (O_4, I_4).

W tabeli 2 porównano, dla wybranego zestawu parametrów sterujących, najlepsze znalezione rozwiązania przez algorytm *GASA* oraz *GA*:

- uśrednione wartości błędów dla algorytmu *GASA* $E_{śr}$ są nieznacznie niższe niż dla algorytmu *GA*,
- liczba znalezionych rozwiązań optymalnych (najlepszych dotąd znanych) jest identyczna dla obu algorytmów. Natomiast algorytm *GASA*, dla różnej liczby generowanych potomków, częściej znajdował najlepsze znane rozwiązania niż algorytm *GA*.

Podsumowując uzyskane wyniki, można stwierdzić, że zastosowanie algorytmu *GASA*, dla wybranego zestawu instancji testowych, przez osłabienie zbieżności procesu optymalizacji prowadzi do uzyskania nieznacznie lepszych rezultatów w porównaniu z klasycznym algorytmem *GA*. Realizowane jest to koszem wydłużenia czasu optymalizacji.

BIBLIOGRAFIA

1. Burkard R.E., Karisch S.E., Rendl F.: QAPLIB-A Quadratic Assignment Problem Library. European Journal of Operational Research, 1991, 55, 115.
2. Croes G. A.: A Method for Solving Traveling-Salesman Problems: Operations Research. Vol. 6, No. 6 (Nov. - Dec., 1958), p. 791-812.
3. Kadłuczka P., Chmiel W.: Zastosowanie własności zagadnienia QAP w konstrukcji algorytmów ewolucyjnych. AGH, Kraków 2004, s. 112-120.
4. Filipowicz B., Wala K.: Algorytmy optymalizacji kwadratowego zagadnienia przydziału. Kwartalnik Elektrotechnika, z.1, Wydawnictwo AGH, Kraków 1992, 32-42.

5. Goldberg D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, 1989.
6. Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer Verlag, 1995.
7. Nissen V.: Evolutionäre Algorithmen Darstellung, Beispiele, betriebswirtschaftliche Anwendungs-möglichkeiten, Deutscher Universitäts Verlag, Wiesbaden 1994.

Recenzent: Prof. dr hab. inż. Zdzisław Duda

Abstract

The paper presents an implementation of evolutionary algorithm basing on the approach used in the simulated annealing algorithm. The algorithm implements multiphase strategy of succession of populations and divides of the population into two subsets "elite" and "divers". During all phases the probability of adding solution to population depends on the solution fitness function and certain probability density function. This approach results in weaker convergence of solutions what enables evolutionary algorithm to exploit more extensively the solution space and in consequence allows to find a solution with better fitness function. The numerical experiments were performed for standard test problems of quadratic assignment problem (QAP) from QAPLIB-A library. We compare the results of proposed algorithm with results obtained from evolution algorithms for QAP problem using standard strategy of succession of populations.