AUTOMATYKA • 2005 • Tom 9 • Zeszyt 1–2

Piotr Kadłuczka*, Wojciech Chmiel*

Efektywność algorytmu ewolucyjnego wykorzystującego warunkową wartość oczekiwaną funkcji celu

1. Wprowadzenie

Artykuł zawiera propozycje wykorzystania w konstrukcji algorytmów ewolucyjnych wartości oczekiwanej oraz warunkowej wartości oczekiwanej funkcji celu rozwiązań częściowo ustalonych, jako dodatkowego parametru, będącego źródłem informacji o własnościach przestrzeni rozwiązań. Zaprojektowane operatory pseudogenetyczne stosują powyższe wielkości w ocenie perspektywiczności nowo tworzonych rozwiązań.

W pracy zaprezentowano opis implementacji operatorów oraz wyniki badań eksperymentalnych dla przykładowych zagadnień testowych z biblioteki *QAPLIB-A*. Przeprowadzono badania porównawcze efektywności zaprojektowanych operatorów z typowymi operatorami genetycznymi, stosowanymi dla zagadnień permutacyjnych: *PMX, OX, CX* itp.

Przedstawione wyniki prac i propozycje rozwiązań mogą być wykorzystane także dla innych zagadnień kombinatorycznych generalizowanych przez *QAP*.

2. Model matematyczny zagadnienia *QAP*

Zagadnienie *QAP* (*Quadratic Assignment Problem*) modeluje wiele ważnych zagadnień, dla których rozwiązania można przedstawić w postaci permutacji.

Przykładami zastosowań praktycznych zagadnienia są:

- problem lokalizacji kooperujących ze sobą zakładów produkcyjnych, maszyn w hali w zautomatyzowanych systemach produkcji, centrów handlowych w projektowanych miastach;
- organizacja biur, terminali przeładunkowych, oddziałów szpitalnych [1];
- projektowanie rozmieszczenia elementów elektronicznych w układach o wielkiej skali integracji (*VLSI*), problem połączeń Steinberga [2];

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

– wyważanie turbin silników odrzutowych [3];
– problem dopasowania molekularnego [4];
– projektowanie klawiatur, przełączników ATM oraz wiele innych [5].

Problem *QAP* należy do klasy zagadnień *NP*-trudnych, co wymusza stosowanie do jego rozwiązania metod przybliżonych już dla zagadnień o rozmiarach powyżej 30. Z uwagi, że jest ono znacznie trudniejsze niż inne zagadnienia optymalizacji kombinatorycznej, a z drugiej strony modeluje ważną klasę problemów decyzyjnych, prace nad rozwojem zastosowanych metod przybliżonych dla tego zagadnienia należą do aktualnego i intensywnego nurtu badań naukowych.

Model matematyczny zagadnienia *QAP* możemy 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*

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

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

3. Algorytm ewolucyjny

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 nowo utworzonych rozwiązań potomnych o miejsce w populacji, na podstawie wartości funkcji celu.

3.1. Opis algorytmu *modGA*

Parametrami algorytmu (rys. 1) są:

O – zbiór operatorów pseudogenetycznych (U – podzbiór operatorów unarnych),
$p=\{p_i\}$ – zbiór prawdopodobieństw wyboru operatorów ($p_i \geq 0$, $i \in O$, $\sum_{i \in O} p_i = 1$),

λ – rozmiar populacji,
I_{max} – zadana liczba wygenerowanych przez algorytm potomków.

W algorytmie U jest podzbiorem zbioru operatorów pseudogenetycznych O, do którego należą operatory unarne, tzn. takie, które do utworzenia potomka potrzebują jednego rodzica (tworząc na jego podstawie także jednego potomka). Przez P oznaczono populację rozwiązań przetwarzaną przez algorytm, natomiast przez R wygenerowany zbiór rozwiązań wstępnych.

```
algorytm modGA(I_max, O, p, λ)
R ← GenerujRozwiązaniaWstępne(λ)
Oceń(R)
P ← UtwórzPopulacjęPoczątkową(R, λ)
I ← 0
while I < I_max
do
    o_i ← LosujOperator(O, p)
    if o_i ∈ U
    then
        π_1 ← LosujRodzica(P)
        π^1 ← GenerujPotomka(π_1, o_i)
        Oceń(π^1)
        Wstaw(π^1)
        I ← I + 1
    else
        π_1 ← LosujRodzica(P)
        π_2 ← LosujRodzica(P)
        {π^1, π^2} ← GenerujPotomka(π_1, π_2, o_i)
        Oceń(π^1, π^2)
        Wstaw(π^1, π^2)
        I ← I + 2
π_best ← ZwróćNajlepsze(P)
return (π_best);
end modGA
```

Rys. 1. Pseudokod algorytmu *modGA*

Opis zastosowanych w algorytmie procedur:

GenerujRozwiązaniaWstępne() – generuje losowo, w oparciu o rozkład równomierny, populację rozwiązań początkowych;

Oceń() – określa wartość funkcji przystosowania (celu) dla rozwiązania lub rozwiązań;

UtwórzPopulacjęPoczątkową() – tworzy uporządkowana według wartości funkcji przystosowania (od najlepszego do najgorszego) populację początkową P o rozmiarze l, w oparciu o zbiór R zawierający rozwiązania wstępne;

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

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

GenerujPotomka() – w oparciu o wylosowany operator dokonuje modyfikacji rodzica / rodziców i zwraca potomka / potomków;

Wstaw() – 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;

ZwróćNajlepsze() – zwraca najlepsze rozwiązania w populacji P.

3.2. Operatory genetyczne

W przeprowadzonych badaniach komputerowych zastosowano typowe operatory genetyczne dla zagadnień permutacyjnych, dlatego skrótowy opis sygnalizuje jedynie zasadę ich działania:

1. Operator mutacji *RM* (*random mutation*)

Operator *RM* [7] dokonuje przestawienia wartości na dwóch wylosowanych pozycjach rozwiązania, dzięki temu zawsze otrzymujemy rozwiązanie dopuszczalne.

Dla zagadnień permutacyjnych, których przykładem jest *QAP*, operatory krzyżowania są bardziej skomplikowane, ponieważ proste krzyżowanie może prowadzić do utworzenia rozwiązań niedopuszczalnych. Klasyczne jedno- lub dwupunktowe operatory krzyżowania mogą być stosowane jedynie łącznie z procedurą naprawczą. Spośród operatorów zapewniających dopuszczalność w algorytmie zastosowano operatory krzyżowania: zachowujący bezwzględny porządek elementów w permutacji (*PMX*), porządek względny (*OX*) i wykorzystujący relację *„pozycja-wartość”* między rozwiązaniami (*COMX*).

2. Operator krzyżowania z częściowym odwzorowaniem *PMX* (*partially matched crossover*)

W operatorze *PMX* [8] sekcja kojarzenia jest wykorzystywana do wykonywania krzyżowania za pomocą operacji wymiany typu „*pozycja za pozycją*". Miejsca krzyżowania są określane w sposób losowy, a następnie dokonywana jest wymiana podciągów położonych pomiędzy punktami przecięcia i zapełniane miejsca, dla których nie zachodzi konflikt. Reszta pozycji rozwiązań uzupełniana jest tak, aby zachować bezwzględny porządek elementów, wynikający z odwzorowania pomiędzy pozycjami rozwiązań.

3. Operator krzyżowania łączącego *COMX* (*composition crossover*)

W operatorze *COMX* [7], potomków tworzy się w oparciu o relację „*pozycja-wartość*" między rozwiązaniami rodzicielskimi. Wartość kolejnych pozycji pierwszego rozwiązania decyduje o kolejności wstawienia wartości z drugiego rozwiązania do rozwiązania potomnego. Aby otrzymać drugiego potomka postępujemy analogicznie.

4. Operator krzyżowania z porządkowaniem *OX* (*order crossover*)

Zamiast stosowania wymiany „*pozycja za pozycją*" jak w operatorze *PMX*, w *OX* [9] stosuje się przesunięcia elementów, w celu zapełnienia „dziur" powstałych podczas przenoszenia elementów odwzorowywanych pozycji permutacji. Poczynając od drugiego punktu przecięcia u jednego z rodziców, kopiuje się kolejne wartości z drugiej permutacji, omijając te, które już tam są. Po osiągnięciu końca ciągu rozpoczyna się od pierwszego miejsca u drugiego rodzica. Analogicznie otrzymuje się drugie rozwiązanie potomne.

4. Warunkowa wartość oczekiwana funkcji celu w konstrukcji algorytmu ewolucyjnego

Przedstawione propozycje zastosowania warunkowej wartości oczekiwanej funkcji celu dotyczą dwóch podstawowych ról, jakie może ten parametr spełniać w konstrukcji algorytmu. Pierwsza grupa dotyczy sposobu modyfikacji rozwiązań realizowanych przez operatory genetyczne, a druga sterowania algorytmem. Opisane metody zastosowania są jedynie pojedynczymi przykładami licznych możliwości, jakie można zrealizować w oparciu o ten parametr.

4.1. Zmodyfikowane operatory genetyczne

Propozycją wykorzystania warunkowej wartości oczekiwanej jest próba poprawy efektywności działania operatorów genetycznych. Warunkową wartość oczekiwaną funkcji celu – $E(\phi(\pi) \mid_{i \in D})$ w uproszczeniu oznaczoną $E(\phi(\pi))$, obliczamy dla rozwiązań częściowo ustalonych, na pozycjach różniących dwa rozwiązania. Rozważając rozwiązania: rodzica – π_1 i potomka – π^1, ustalonymi pozycjami $i \in D$ są pozycje różnicujące oba rozwiązania: $D = \{i : \pi_1(i) \neq \pi^1(i)\}$. Przykładem modyfikacji operatora może być zastąpienie losowego działania operatora działaniem ukierunkowanym za pomocą warunkowej wartości oczekiwanej. Może to dotyczyć, na przykład, w operatorach krzyżowania algorytmu naprawczego usuwającego „kolizje" i „dziury", wyboru pozycji przecięcia, długości sekcji kojarzenia lub ustaleniu pozycji rozwiązania nie podlegających modyfikacji itp.

W realizowanym obliczeniowo algorytmie wprowadzono do operatorów krzyżowania *PMX* i *CX* oraz operatora mutacji *RM* wybór na podstawie $E(\phi(\pi))$ punktów przecięcia rozwiązań rodzicielskich, ze zbioru ustalonych K zestawów punktów (a przez to zbioru pozycji ustalonych rozwiązania $\pi_1 - D_k$)

$$D^* = \min_{k=1,\ldots,K} \left\{ E(\phi(\pi^1) \mid_{i \in D_k}) \right\} \tag{2}$$

Oczywiście jest to jedynie jeden przykład z wielu wymienionych na wstępie, których efektywność można ocenić w wyniku serii żmudnych badań testowych.

4.2. Sterowanie algorytmem

Sterowanie przebiegiem algorytmu może być realizowane w oparciu o warunkową wartość oczekiwaną funkcji celu rozwiązań aktualnych na wiele sposobów, których efektywność można ocenić jedynie eksperymentalnie. W rozważanym przykładzie tego typu zmodyfikowano wyjściowy algorytm *modGA*, w którym użycie operatora określa prawdopodobieństwo, będące parametrem algorytmu. W przypadku operatora mutacji wynosi ono p_{RM} i z takim prawdopodobieństwem operator ten dokona losowej zmiany w losowo wybranym z populacji rozwiązaniu. Następnie rozwiązanie potomne po obliczeniu wartości funkcji przystosowania wstawiane jest do populacji.

W przypadku proponowanego rozwiązania wykorzystującego warunkową wartość oczekiwaną procedura realizująca operator mutacji jest wywoływana częściej, ale liczba skutecznych realizacji operatora pozostaje bez zmian. Podobnie jak poprzednio wybór rozwiązania i przestawianych pozycji permutacji dokonywany jest losowo. Następnie obliczana jest warunkowa wartość oczekiwana dla rodzica – π_1 i potomka – π^1, przy ustalonych pozycjach $i \in D$ różniących oba rozwiązania

$$\Delta E = E(\phi(\pi^1)) - E(\phi(\pi_1)) \tag{3}$$

W przypadku gdy następuje poprawa warunkowej wartości oczekiwanej podejmowana jest próba wstawienia rozwiązania potomnego do populacji. Kryterium wstawienia jest tak jak poprzednio wartość funkcji przystosowania. W przypadku przeciwnym próba wstawienia jest dokonywana z malejącym prawdopodobieństwem, zależnym od różnicy wartości oczekiwanej (2), co jest realizowanie przez losowanie z rozkładem równomiernym $\delta \in [0, 1]$, a następnie dokonanie porównania (α – stała)

$$\delta < e^{-\frac{\Delta E}{\alpha}} \tag{4}$$

W przypadku niespełnienia warunku (4) próba wstawienia rozwiązania potomnego do populacji nie jest wykonywana i ponownie wywoływany jest operator *RM*. W wyniku zastosowanego mechanizmu następuje bardziej intensywne wykorzystanie mutacji, która nie działa już w pełni losowo, ale w pewnym sensie ukierunkowuje algorytm w stronę podprzestrzeni rozwiązań „bardziej perspektywicznych". W świetle przytoczonych we wcześniejszym artykule wyników eksperymentów dotyczących ogólnych własności warunkowej

wartości oczekiwanej, podejrzewamy, że mechanizm ten realizuje intensyfikacje w początkowej fazie pracy algorytmu (poprawa zbieżności, gdy rozwiązania populacji są gorsze), natomiast w fazie późniejszej jest mechanizmem różnicowania, przeciwdziałającym przedwczesnej zbieżności algorytmu.

5. Eksperymenty komputerowe

Na podstawie algorytmu *modGA* wykonano eksperymentalne oprogramowanie, zaimplementowane dla zagadnienia *QAP* w dwóch wersjach:

1) podstawowej,
2) wykorzystującej warunkową wartość oczekiwaną (oznaczoną *modGA-1*).

Wszystkie eksperymenty komputerowe wykonane w oparciu o algorytm korzystają z następującego zbioru parametrów: I_{max} = 10 000 iteracji, λ = 100 (rozmiar populacji), zbioru operatorów O={*RM, PMX, COMX, OX*}. oraz prawdopodobieństw losowania operatorów wynoszących odpowiednio p_{RM} = 0,1, p_{PMX} = 0,3, p_{COMX} = 0,2, p_{OX} = 0,4.

W tabeli 1 przedstawiono wyniki badań eksperymentalnych zagadnienia *QAP*, dla 38 zagadnień testowych o rozmiarze n = 22÷64, zaczerpniętych z biblioteki *QAPLIB-A* [10], porównując obie wersje algorytmu.

QAPLIB-A jest biblioteką zagadnień testowych dla zagadnienia kwadratowego przypisania. Została ona stworzona w 1991 r. i jest rozwijana do dzisiaj przez: R. Burkarda, S. Karischa i F. Rendla. Zawiera instancje testowe opisujące rzeczywiste i wygenerowane zadania:

BurXX – (Burkard i Offermann) – Macierz odległości opisuje przeciętny czas pisania par liter przez stenotypistę, podczas gdy macierz przepływów określa częstotliwość występowania par liter w różnych językach (w oparciu o 100 000 przykładów) (asymetryczne).

ChrXX – (Christofides i Benavent) – Macierz odległości jest macierzą przyległości ważonego drzewa, macierz przepływów jest macierzą przyległości pewnego grafu pełnego (symetryczne).

EscXX – (Eschermann i Wunderlich) – Macierze zawierają dane dla samotestujących się obwodów elektronicznych. Celem zadania była minimalizacja czasu testowania sprzętu.

KraXX – (Krarup i Pruzan) – Opisują rzeczywiste zagadnienie planowania kliniki w Regensburgu.

LipaXX – (Li i Pardalos) – zadania powstały w oparciu o generator instancji testowych.

SkoXX – (Skorin i Kapov) – Macierze odległości oraz przepływu zawierają liczby pseudolosowe.

SteXX – (Steinberg) – Opisują problem minimalizacji długości połączeń kablowych. W pierwszym problemie macierz odległości opisuje odległości na Manhattanie.

ThoXX – (Thonemann i Bölte) – Macierze odległości zawierają odległości pomiędzy punktami na prostokącie.

WilXX – (Wilhelm i Ward) – Macierze odległości zawierają dystans pomiędzy punktami na prostokącie.

Tabela 1
Wyniki eksperymentów komputerowych dla opisanych algorytmów *modGA* oraz *modGA-1* dla zadań testowych zaczerpniętych z biblioteki *QAPLIB-A*

Nazwa	$\phi_{QAPLIB\text{-}A}$	*modGA*	*modGA-1*							
			PMX		*OX*		*RM*		*modGA-1**	
		ϕ^*_{ap}	ϕ_{ap}	$E_{[\%]}$	ϕ_{ap}	$E_{[\%]}$	ϕ_{ap}	$E_{[\%]}$	ϕ_{ap}	$E_{[\%]}$
BUR26A	5426670	5441750	5461450	0,362	5444640	0,053	5447710	0,110	5451240	0,174
BUR26B	3817852	3830030	3852200	0,579	3842550	0,327	3829660	–0,010	3841630	0,303
BUR26C	5426795	5450390	5468040	0,324	5433880	–0,303	5441070	–0,171	5449070	–0,024
BUR26D	3821228	3836120	3842360	0,163	3825380	–0,280	3838450	0,061	3829300	–0,178
BUR26E	5386879	5416120	5402990	–0,242	5403980	–0,224	5425150	0,167	5391140	–0,461
BUR26F	3782044	3809440	3790550	–0,496	3786670	–0,598	3790880	–0,487	3787220	–0,583
BUR26G	10117172	10136400	10169100	0,323	10188800	0,517	10166500	0,297	10176900	0,400
BUR26H	7098658	7132490	7119810	–0,178	7147040	0,204	7186760	0,761	7127250	–0,073
CHR22A	6156	7218	7420	2,799	7806	8,146	7252	0,471	7182	–0,499
CHR22B	6194	7128	7172	0,617	7606	6,706	7302	2,441	7240	1,571
CHR25A	3796	6442	6864	6,551	6510	1,056	7800	21,080	7006	8,755
ESC32A	130	184	186	1,087	184	1,056	210	14,130	184	0,000
ESC32C	642	644	644	0,000	644	1,056	644	0,000	646	0,311
ESC32D	200	218	210	–3,670	214	1,056	224	2,752	222	1,835
ESC32E	2	2	2	0,000	2	1,056	2	0,000	2	0,000
ESC32F	2	2	2	0,000	2	1,056	2	0,000	2	0,000
ESC32G	6	6	6	0,000	6	1,056	6	0,000	6	0,000
ESC32H	438	456	482	5,702	462	1,056	472	3,509	450	–1,316
ESC64A	116	128	140	9,375	128	1,056	126	–1,563	124	–3,125
KRA30A	88900	98880	99080	0,202	101220	1,056	100170	1,305	97820	–1,072
KRA30B	91420	99360	100770	1,419	100070	1,056	100880	1,530	99860	0,503
LIPA30A	13178	13510	13509	–0,007	13495	1,056	13516	0,044	13536	0,192
LIPA30B	151426	182580	183036	0,250	179367	1,056	182142	–0,240	181986	–0,325
LIPA40A	31538	32313	32194	–0,368	32249	1,056	32259	–0,167	32255	–0,179
LIPA40B	476581	584408	579409	–0,855	579911	1,056	582401	–0,343	580658	–0,642
LIPA50A	62093	63252	63395	0,226	63289	1,056	63335	0,131	63269	0,027
LIPA50B	1210244	1485030	1477090	–0,535	1482010	1,056	1474430	–0,714	1480280	–0,320
LIPA60A	107218	109048	108989	–0,054	109014	1,056	109174	0,116	109050	0,002
LIPA60B	2520135	3134900	3144690	0,312	3113170	1,056	3144920	0,320	3138410	0,112
SKO42	15812	16960	17248	1,698	17286	1,056	17214	1,498	17294	1,969
SKO49	23386	25620	25946	1,272	25784	1,056	25442	–0,695	25824	0,796
SKO56	34458	37782	38248	1,233	38250	1,056	37892	0,291	37442	–0,900
STE36A	9526	11488	11684	1,706	12358	1,056	11306	–1,584	11798	2,698
STE36B	15852	27646	25678	–7,119	25268	1,056	25544	–7,603	25038	–9,434
STE36C	8239110	10053400	10094700	0,411	9539680	1,056	10658800	6,022	10045300	–0,081
THO30	149936	166914	165250	–0,997	162500	1,056	162780	–2,477	162842	–2,440
THO40	240516	276588	273080	–1,268	267368	1,056	276204	–0,139	274430	–0,780
WIL50	48816	51538	51560	0,043	51812	1,056	51428	–0,213	51502	–0,070
Średnia				0,549		1,056		1,069		–0,075

W tabeli 1 przyjęto następujące oznaczenia:

$\phi_{\text{QAPLIB-A}}$ – najlepsza znana wartość funkcji przystosowania, zaczerpnięta z biblioteki *QAPLIB-A*;

ϕ^*_{ap} – wartość funkcji przystosowania najlepszego znalezionego rozwiązania – π_{bestp}, dla *modGA*;

ϕ_{ap} – wartość funkcji przystosowania najlepszego znalezionego rozwiązania – π_{bestp}, dla *modGA-1*;

$E = 100\%*(\phi_{ap} - \phi^*_{ap})/\phi^*_{ap}$ – względna różnica funkcji celu najlepszego rozwiązania dla algorytmu *modGA-1* w stosunku do *modGA*;

modGA – wyniki dla klasycznej wersji algorytmu *modGA*;

modGA-1 – wyniki dla wersji algorytmu, wykorzystujących $E(\phi(\pi))$:

- *PMX* – zastosowanie zmodyfikowanego operatora *PMX*;
- *OX* – zastosowanie zmodyfikowanego operatora *OX*;
- *RM* – zastosowanie zmodyfikowanego operatora *RM*;

*modGA-1** – zastosowanie zmodyfikowanego sterowania algorytmem.

6. Podsumowanie

Zamieszczone w tabeli 1 wyniki pozwalają zauważyć, że modyfikacja działania operatorów genetycznych przez wprowadzenie warunkowej wartości oczekiwanej pozwala osiągać wyniki porównywalne z algorytmem *modGA*. Dla kolejno rozważanych operatorów *PMX, OX, RM*, najlepsze wyznaczone rozwiązania były porównywalnej jakości (średnio gorsze o ok. 0,5%, 0,2%, 1%) – co pozwoliło uzyskać dla wielu instancji rozwiązania lepsze. Średnią ocenę jakości uzyskanych wyników pogarsza występowanie specyficznych zadań testowych, dla których osiągnięto bardzo złe wyniki (np. dla *CHR25A* i operatora *RM* – 21% gorsze). Uwzględniając wydajność operatorów *PMX, OX, COMX, RM*, która wynosiła odpowiednio 30%, 6%, 1%, 37% (procent rozwiazań utworzonych przez dany operator, które weszły do populacji – dla *modGA*) można podejrzewać, że lepsze wyniki operatorów krzyżowania w stosunku do mutacji mogą być spowodowane większą liczbą pozycji ustalonych rozwiązania, a operatora *OX* w stosunku do *PMX* niższą jego wydajnością.

Algorytm *modGA-1**, w którym w oparciu o $E(\phi(\pi))$ realizowane jest sterowanie użyciem operatora mutacji, pozwala uzyskać najlepsze wyniki. Pozostawienie w pełni losowej mutacji pozwala różnicować populację, co daje znacznie lepsze efekty niż deterministyczny wybór pozycji ustalonych przy stosowaniu zmodyfikowanego operatora *RM*.

Przedstawione w artykule eksperymenty obliczeniowe potwierdzają zasadność użycia warunkowej wartości oczekiwanej w konstrukcji algorytmu ewolucyjnego. W porównaniu do standardowego algorytmu *modGA* po wprowadzeniu mechanizmów wykorzystujących $E(\phi(\pi))$ nastąpiła poprawa uzyskiwanych wyników dla dużej liczby zadań testowych.

Sprawdzenie pozostałych propozycji i zoptymalizowanie sposobu wykorzystania tego dodatkowego parametru oceny (dot. miejsca wstawienia, liczby i istotności pozycji ustalonych, złożoności obliczeniowej wprowadzanych mechanizmów) powinna zaowocować dalszą poprawą efektywności algorytmu.

Literatura

[1] Bermúdez R., Cole M. H.A.: *Genetic Algorithm Approach to Door Assignments in Breakbulk Terminals*. Final Report MBTC-1102, Mack-Blackwell Transportation Center, University of Arkansas, Fayetteville, Arkansas 2001
[2] Steinberg L.: *Backboard Wiring Problem: A Placement Algorithm*. SIAM Review, 3, 1967, 37
[3] Mason, A., Rönnqvist, M.: *Solution Methods for the Balancing of Jet Turbines.* Computers and Operations Research, 24, 2, 1997, 153
[4] Phillips A.T., Rosen J.B.: *A Quadratic Assignment Formulation of the Molecular Conformation Problem.* Journal of Global Optimization, 4, 1994, 229
[5] Borst S., Ramakrishnan K.G.: *An Application of Quadratic Assignment Problem to ATM Switch Design.* Mathematical programming in telecommunications, 1997
[6] Michalewicz Z.: *Genetic algorithms + data structures = evolution programs.* Berlin, Springer-Verlag 1992
[7] Wala K., Chmiel W.: *Evolution Algorithm for Quadratic Assignment Problem.* Kraków, University of Mining and Metallurgy Press, Elektrotechnika, 1, 1, 1997, 409
[8] Goldberg D., Lingle R.: *Alleles, loci, and the Traveling Salesman Problem.* Proc. International Conference on Genetic Algorithms and their Applications, 1985
[9] Oliver, I.M., Smith D.J., Holland J.R.C.: *A Study of Permutation Crossover Operations on the Traveling Salesman Problem.* Proceedings of the Fourth International Conference on Genetic Algorithms, 32, 1991
[10] Burkard R.E., Karisch S.E., Rendl F.: *QAPLIB-A Quadratic Assignment Problem Library.* European Journal of Operational Resarch, 55, 1991, 115